AI Assisted Visual Communication Through Generative Models



Arnab Ghosh

St Cross College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2021

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Arnab Ghosh, St Cross College

Acknowledgements

I would like to thank my supervisor, Prof. Philip Torr, for having faith in me and providing the opportunity to pursue a PhD. His infectious energy and creativity have been pivotal in making this thesis possible.

I have also had the pleasure of working closely with many collaborators during my PhD. My mentors in my initial projects – Puneet Dokania, Richard Zhang, Eli Shechtman, Oliver Wang, Alyosha Efros – have played a significant role in my development as a researcher. I have also enjoyed subsequent collaborations with Harkirat, Rodrigo, Viveka and Adnane. Further thanks go to the computer vision teams in Adobe (particularly Eli, Richard, Oliver) for hosting me. Richard, Puneet, Eli, Oliver and Alyosha have also taken the time to provide detailed and thoughtful feedback on drafts of my papers on numerous occasions.

I have been fortunate to be part of the computer vision community in Oxford. There are too many past and present members in TVG to name here, along with many others from VGG, OVAL and AVL from whom I have learned things from whilst sharing an "office". One of the best perks of studying at Oxford has been the impressive array of seminar speakers who have given talks here and then been available for further discussions afterwards. I am also grateful to my transfer report and confirmation of status examiners, Professors Pawan Kumar and Ioannis Havoutis. My thesis examiners, Professor Victor Prisacariu and Dr Aaron Hertzmann, also provided insightful comments on the manuscript and the viva itself turned out to be a stimulating discussion.

I have thoroughly enjoyed my time here in Oxford, both inside and outside of the lab, thanks to Christian, Tom, Aravindh, Namhoon, Qizhu, Li, Harkirat, Daniela, Oscar, Viveka .

Maddy, and then later Cassandra, have also been helpful regarding all administrative matters. And thanks to Jerry, as well as the ARC team, for managing the group's computing resources.

I am grateful to my family for their continual support during my PhD. In particular, I am thankful to my parents for always having my back through thick and thin.

Finally, I am grateful to have been funded by the Microsoft Studentship.

Abstract

Expressing ideas in our minds which are inevitably visual into words had been a necessity. Lack of tools to express oneself creatively using visual communication had been a bottleneck. With the advent of AI and generative models starting to be applied to graphics based tasks, this elusive elysian fields seems to be a step closer. This thesis tries to tackle the challenge using the 3 pronged approach of GANs, VAEs and Continuous Normalizing Flows.

We first propose MAD-GAN, an intuitive generalization to the Generative Adversarial Networks (GANs) and its *conditional variants* to address the well known problem of *mode collapse*. First, MAD-GAN is a multi-agent GAN architecture incorporating multiple generators and one discriminator. Second, to enforce that different generators capture diverse high probability modes, the discriminator of MAD-GAN is designed such that along with finding the real and fake samples, it is also required to identify the generator that generated the given fake sample. Intuitively, to succeed in this task, the discriminator must learn to push different generators towards different identifiable modes. We perform extensive experiments on synthetic and real datasets and compare MAD-GAN with different variants of GAN. We show high quality diverse sample generations for challenging tasks such as image-to-image translation and face generation. In addition, we also show that MAD-GAN is able to disentangle different modalities when trained using highly challenging diverse-class dataset (For examplethe dataset with images of forests, icebergs, and bedrooms). In the end, we show its efficacy on the unsupervised feature representation task. We introduce a similarity based competing objective (MAD-GAN-Sim) which encourages different generators to generate diverse samples based on a user defined similarity metric. We show its performance on the image-to-image translation, and also show its effectiveness on the unsupervised feature representation task.

We then propose an interactive GAN-based sketch-to-image translation method that helps novice users create images of simple objects. As the user starts to draw a sketch of a desired object type, the network interactively recommends plausible completions, and shows a corresponding synthesized image to the user. This enables a feedback loop, where the user can edit their sketch based on the network's recommendations, visualizing both the completed shape and final rendered image while they draw. In order to use a single

trained model across a wide array of object classes, we introduce a gating-based approach for class conditioning, which allows us to generate distinct classes without feature mixing, from a single generator network.

In Appendix C we propose a semi supervision based strategy for deep generative modelling for human body analysis. Deep generative modelling for human body analysis is an emerging problem with many interesting applications. However, the latent space learned by such approaches is typically not interpretable, resulting in less flexibility. In this work, we present deep generative models for human body analysis in which the body pose and the visual appearance are disentangled. Such a disentanglement allows independent manipulation of pose and appearance, and hence enables applications such as pose- transfer without specific training for such a task. Our proposed models, the Conditional-DGPose and the Semi-DGPose, have different characteristics. In the first, body pose labels are taken as conditioners, from a fully-supervised training set. In the second, our structured semi-supervised approach allows for pose estimation to be performed by the model itself and relaxes the need for labelled data. Therefore, the Semi-DGPose aims for the joint understanding and generation of people in images. It is not only capable of mapping images to interpretable latent representations but also able to map these representations back to the image space. We compare our models with relevant baselines, the ClothNet-Body and the Pose Guided Person Generation networks, demonstrating their merits on the Human3.6M, ChictopiaPlus and DeepFashion benchmarks.

Finally, Training Neural Ordinary Differential Equations (ODEs) is often computationally expensive. Indeed, computing the forward pass of such models involves solving an ODE which can become arbitrarily complex during training. Recent works have shown that regularizing the dynamics of the ODE can partially alleviate this. In this paper we propose a new regularization technique: randomly sampling the end time of the ODE during training. The proposed regularization is simple to implement, has negligible overhead and is effective across a wide variety of tasks. Further, the technique is orthogonal to several other methods proposed to regularize the dynamics of ODEs and as such can be used in conjunction with them. We show through experiments on normalizing flows, time series models and image recognition that the proposed regularization can significantly decrease training time and even improve performance over baseline models.

Contents

List of Figures **iv**

List of Tables **x**

1 Introduction **1**
- 1.1 Communication . . . 1
- 1.2 Why is visual communication important? . . . 6
- 1.3 Taxonomy of Generative Models . . . 7
- 1.4 Generative Models for Visual Communication . . . 9
- 1.5 Major Challenges . . . 9
- 1.6 Key Ideas . . . 11
- 1.7 Thesis Outline & Contributions . . . 12

2 Background **14**
- 2.1 Generative Adversarial Networks . . . 14
- 2.2 Variational Autoencoders . . . 15
- 2.3 Neural ODEs . . . 17
  - 2.3.1 Continuous Normalizing Flows . . . 17

3 MADGAN: Multi-Agent Diverse Generative Adversarial Networks **19**
- 3.1 Introduction . . . 21
- 3.2 Related Work . . . 23
- 3.3 Preliminaries . . . 24
- 3.4 Multi-Agent Diverse GAN . . . 25
  - 3.4.1 Multi-Agent GAN Architecture . . . 25
  - 3.4.2 Enforcing Diverse Modes . . . 26
- 3.5 Experiments . . . 29
  - 3.5.1 Non-Parametric Density Estimation . . . 32
  - 3.5.2 Stacked and Compositional MNIST . . . 34

3.5.3 Diverse Samples for Image-to-Image Translation and Comparison to InfoGAN . . . 36
3.5.4 Diverse-Class Data Generation . . . 37
3.5.5 Diverse Face Generation . . . 38
3.5.6 Unsupervised Representation Learning . . . 38
3.6 Conclusion . . . 38
3.7 Insights & Discussions . . . 39
3.8 Insights for Diversity in MAD-GAN . . . 39
3.9 Similarity based competing objective . . . 40
3.9.1 Approach . . . 40
3.9.2 Experiments . . . 42
3.9.3 Diverse Samples for Image-to-Image Translation . . . 42

4 Interactive Sketch & Fill: Multiclass Sketch-to-Image Translation 44
4.1 Introduction . . . 46
4.2 Related Work . . . 48
4.3 Method . . . 49
4.3.1 Shape completion . . . 49
4.3.2 Appearance synthesis . . . 51
4.4 Experiments . . . 53
4.4.1 Single Class Generation . . . 54
4.4.2 Multi-Class Generation . . . 55
4.5 Discussion . . . 58

5 STEER: Simple Temporal Regularization For Neural ODEs 61
5.1 Introduction . . . 63
5.2 Related Work . . . 64
5.3 Methodology . . . 66
5.3.1 Neural ODEs . . . 66
5.3.2 Temporal Regularization . . . 66
5.3.3 Stiff ODEs . . . 68
5.4 Experiments . . . 69
5.4.1 Generative Models . . . 69
5.4.2 Times Series Models . . . 71
5.4.3 Feedforward Models . . . 72
5.4.4 A Toy Example of Stiff ODE . . . 73
5.4.5 A Toy example of a Timeseries Model: . . . 75

5.5 Conclusion 75
5.6 Modified Picard's Iteration: 76
5.7 Stiff ODE: Ablation Studies 80
5.8 Generative Models: Ablation Studies 83

6 Discussion & Future Work **86**

A MADGAN Paper **89**

B iSketchNFill Paper **99**

C DGPose Paper **110**

D STEER Paper **138**

Bibliography **152**

List of Figures

1.1 Efficacy of visual communication 1
1.2 Standard reconstruction of the development of writing. There is a possibility that the Egyptian script was invented independently from the Mesopotamian script [40]. 2
1.3 The Kish tablet from Sumer, with pictographic writing. This may be the earliest known writing, 3500 BC. Ashmolean Museum 3
1.4 Visual communication complements communication using characters in a language 4
1.5 Taxonomy of Generative Models, this thesis mainly draws upon Generative Adversarial Networks 8
1.6 DALL-E [122] on the left and BigGAN[11] images on the right 9
1.7 Angel of Air Unreal Engine [80] on the left and Coral Reef City [2] on the right 10

3.1 Diverse-class data generation using MAD-GAN. Diverse-class dataset contains images from different classes/modalities (in this case, forests, icebergs, and bedrooms). Each row represents generations by a particular generator and each column represents generations for a given random noise input z. As shown, once trained using this dataset, generators of MAD-GAN are able to disentangle different modalities, hence, each generator is able to generate images from a particular modality. 21
3.2 Multi-Agent Diverse GAN (MAD-GAN). The discriminator outputs $k + 1$ softmax scores signifying the probability of its input sample being from either one of the k generators or the real distribution. 23
3.3 Visualization of different generators getting pushed towards different modes. Here, M_1 and M_2 could be a cluster of modes where each cluster itself contains many modes. The arrows abstractly represent generator specific gradients for the purpose of building intuition. 26

3.4 A toy example to understand the behaviour of different GAN variants in order to compare with MAD-GAN (each method was trained for 198000 iterations). The orange bars show the density estimate of the training data and the blue ones for the generated data points. After careful cross-validation, we chose the bin size of 0.1. 30
3.5 A toy example to understand the behavior of MAD-GAN with different number of generators (each method was trained for $1,98,000$ iterations). The orange bars show the density estimate of the training data and the blue ones for the generated data points. After careful cross-validation, we chose the bin size of 0.1. 30
3.7 Diverse generations for *edges-to-handbags* generation task. In each sub-figure, the first column is the input, columns 2-4 are generations by MAD-GAN (using three generators), and columns 5-7 are generations by InfoGAN (using three categorical codes). Clearly different generators of MAD-GAN are producing diverse results capturing different colors, textures, design patterns, *etc.*. However, InfoGAN generations are visually almost the same, indicating mode collapse. 33
3.9 InfoGAN for *edges-to-handbags* task by sharing the discriminator and Q Network. In each sub-figure, the first column is the input, columns $2-4$ are generations when input is categorical code besides conditioning image, and columns $5-7$ are generations with noise as an additional input. The generations for both the architectures are visually the same irrespective of the categorical code value, which clearly indicates that it is not able to capture diverse modes. 34
3.11 Diverse generations for *night-to-day* image generation task. First column in each sub-figure represents the input. The remaining three columns show the diverse generations of three different generators of MAD-GAN (Our). . . . 35
3.13 Face generations using MAD-GAN. Each sub-figure shows generations by a single generator. The first generator is generating faces with very dark background. The second one is generating female faces with long hair in light background, while the third one is generating faces with colored background and casual look (based on facial direction and expression). 35
3.14 Face generations using MAD-GAN. Each generator employed is DCGAN. Each row represents a generator. Each column represents generations for a given random noise input z. Note that, the first generator is generating faces pointing to the left. The second generator is generating female faces with long hair, while the third generator generates images with light background. 37

3.15 MAD-GAN-Sim compared with MAD-GAN. All the generators share parameters of all the layers except the last one. The two proposed diversity enforcing objectives, 'competing' (MAD-GAN-Sim) and 'generator identification' (MAD-GAN), are shown at the end of the discriminator 40
3.17 MAD-GAN-Sim: Diverse generations for 'edges-to-handbags' image generation task. First column in each sub-figure represents the input. The remaining three columns show the diverse outputs of different generators. It is evident that different generators are able to produce very diverse results capturing color (brown, pink, black), texture, design pattern, shininess, among others. 43
3.19 MAD-GAN-Sim: Diverse generations for 'night to day' image generation task. First column in each sub-figure represents the input. The remaining three columns show the diverse outputs of different generators. It is evident that different generators are able to produce very diverse results capturing different lighting conditions, different sky patterns (cloudy vs clear sky), different weather conditions (winter, summer, rain), different landscapes, among many other minute yet useful cues. 43

4.1 (**Top**) Given a user created incomplete object outline (first row), our model estimates the complete shape and provides this as a recommendation to the user (shown in gray), along with the final synthesized object (second row). These estimates are updated as the user adds (green) or removes (red) strokes over time – previous edits are shown in black. (**Bottom**) This generation is class-conditioned, and our method is able to generate distinct multiple objects for the same outline (For example'circle') by conditioning the generator on the object category. 46
4.2 **Video of our interface** We can see two versions of our interface. The left side shows how a user can quickly generate multiple objects using a few strokes, while the right side shows the utility of multimodal completions where the user can quickly explore different possible shape generations while drawing. Full video available at our website. **Please view with Acrobat Reader.** 47
4.3 **Our two-stage approach** First, we complete a partial sketch using the shape generator G_S. Then we translate the completed sketch into an image using the appearance generator G_A. Both generators are trained with their respective discriminators D_S, and D_A. 50

4.4 **First stage (Shape Generator)** To achieve multi-modal completions, the shape generator is designed using inspiration from non-image conditional model [103] with the conditioning input provided at multiple scales, so that the generator network doesn't ignore the partial stroke conditioning. 50
4.5 **Simulated Inputs** Three sizes of occluders were used to simulate partial outlines. 51
4.6 **Simplified Edges** The 2nd edgemap is obtained using the technique of [66], while the 3rd is the intermediate edgemap using [93] and further simplified using [134] which looks closer to what a human would sketch. 51
4.7 **Conditioning variants for the Appearance Generator** Our model uses gating on all the residual blocks of the generator and the discriminator, other forms of conditioning such as (naive concatenation in input only, all layers, AC-GAN like latent regressor [109]) are evaluated as well. 52
4.8 **Injecting conditioning with modified residual layers (Left)** A "vanilla" residual block without conditioning applies a residual modification to the input tensor. **(Mid-left)** The $\mathcal{H}(X)$ block is softly-gated by scalar parameter α and shift β. **(Mid)** Adaptive Instance Normalization [63] applies a channel-wise scaling and shifting after an instance normalization layer. **(Mid-right)** Channel-wise gating adds restrictions to the range of $\boldsymbol{\alpha}$. **(Right)** We find that channel-wise gating (without added bias) produces the best results empirically. 52
4.9 **Example Sketch & Fill Progression.** The **first row** represents the progressive addition of new strokes on the canvas, the **second row** shows the autocompleted sketch, and the **third row** is the final generated image. As the sparse strokes are changed by the user, the completed shape and generated image evolve as well. Note that changing a stroke locally produces coherent changes in other parts of the image. 54
4.10 **Conditioning injection comparison.** We show results across methods on the outline→image task using the **SkinnyResNet** architecture. Naive Concatenation **Concat** often confuses classes, such as oranges and basketballs, while gating mechanisms such as the **ChannelGate** method succeed. The gating method also improves results for the **EncoderDecoder** architecture. 59
4.11 **Directly mapping from partial outline to image** Our proposed system uses a 2-stage approach, using a completed edge map as an intermediate. Here, we show results when directly mapping from the partial outline to the image. When the outline is well-defined, the network can generate realistic images. However, when the outline is sparse, the network struggles with the geometry. 60

4.12 **Multiclass Sketch & Fill results** A few input strokes (first row) are enough to automatically complete the class specific outlines (second) and appearance (last). 60

5.1 Flow fields of a standard Neural ODE (left) vs one equipped with STEER (right). These models transform a simple initial distribution $p(z_{t_0})$ to the target distribution $p(z_t)$. STEER learns simpler flows, which enable faster training. 63
5.2 Effective behavior of Neural ODEs with STEER regularization. The solution is reached at a time $t_1 - b$ instead of time t_1. 67
5.3 Plots showing the path taken by FFJORD with STEER regularization. STEER regularization ($b = 0.25$ and $b = 0.375$) performs the same transformation in shorter paths. 69
5.4 Standard Neural ODE struggles while STEER regularization helps to fit a stiff ODE with stiffness ratio $r = 1000$. 73
5.5 Model with STEER regularization achieves a much lower MSE on a Stiff ODE as the stiffness ratio increases. 74
5.6 A simple addition of t to $A.sin(\phi t)$ changes the qualitative performance of Neural ODE vs STEER. The top row demonstrates the 3 sample trajectories that have been irregularly sampled using STEER, while the bottom row is with a standard Neural ODE. The green points indicate the irregularly sampled points, while the rest of the points are predicted using each trained model. The amplitude A of the model is the same across the various samples but the frequency ϕ is variable across the samples. As we see from the sampled trajectories STEER models the qualitative behavior of the trajectories better than standard Neural ODEs. 76
5.7 Conditions of the proof. 77
5.8 Comparison of the losses for the various choices of b across varying stiffness ratios. 81
5.9 Comparison of the losses for the various choices of standard deviation (std) of the Gaussian across varying stiffness ratios. 81
5.10 Comparison of the losses for the various number of units in the hidden layer for the case of STEER with $b = 0.124$ and a stanard Neural ODE across varying stiffness ratios. 82
5.11 Comparison of a standard Neural ODE along with the Gaussian and Uniform distributions from which the end time t_1 can be sampled. 83

5.12 We experiment with 2 cases which have multiple e^{-kt} in the $\frac{dy}{dt}$ function to be estimated. STEER regularization doesn't totally fail in these scenarios although the performance reduces. 84
5.13 Failure cases of the proposed STEER regularization. Adding a periodic term to the equation $\frac{dy}{dt}$ makes it harder to learn. Changing the behavior of the steady state from $y = 3$ to $y = 7$ causes STEER to not reach a viable steady state solution. Note that a standard Neural ODE also fails in these cases . . 84

List of Tables

3.1 Synthetic experiment on 1D GMM (Fig. 3.4). 31
3.2 Synthetic experiment with different number of MAD-GAN generators (same setup as in Fig. 3.4). 31
3.3 Stacked-MNIST experiments and comparisons. Note that three generators are used for MAD-GAN. 33
3.4 Compositional-MNIST experiments and comparisons. Note that three generators are used for MAD-GAN. 33

4.1 **Single-class generation, 2-stage vs 1-stage**. We evaluate the result quality from different task pipelines. 54
4.2 **Multi-class generation, 2-stage vs 1-stage**. We evaluate the result quality from different task pipelines. Accuracy is computed by a fixed, pretrained classification network, on the resulting images. 55

5.1 Test accuracy and training time comparison on various image datasets. Unless mentioned as days, the time reported is in hours. 70
5.2 Per-time-point classification accuracies on the Human Activity dataset. . . . 70
5.3 Per-sequence classification AUC on Physionet. 70
5.4 Test Mean Squared Error (MSE) ($\times 10^{-2}$) on the MuJoCo dataset. 71
5.5 Test accuracies on Feedforward models on various datsets. 72
5.6 Comparison of b in case of Continuous Normalizing Flows. Greater values of b lead to faster convergence. Altering t_1 such that $t_1 + b = t_1^{original}$ leads to faster convergence times. 85

1 Introduction

What I cannot create,
I do not understand

Richard Feynman

1.1 Communication





Figure 1.1: We are able to recognize the bovine creature almost as succinctly as we understand the wolf. However if we're magically transported to the past when this was being painted we wouldn't understand any of the language being spoken.

Visual communication had been an important method of communication between humans since time immemorial. If we go back to some of the earliest known cave paintings more

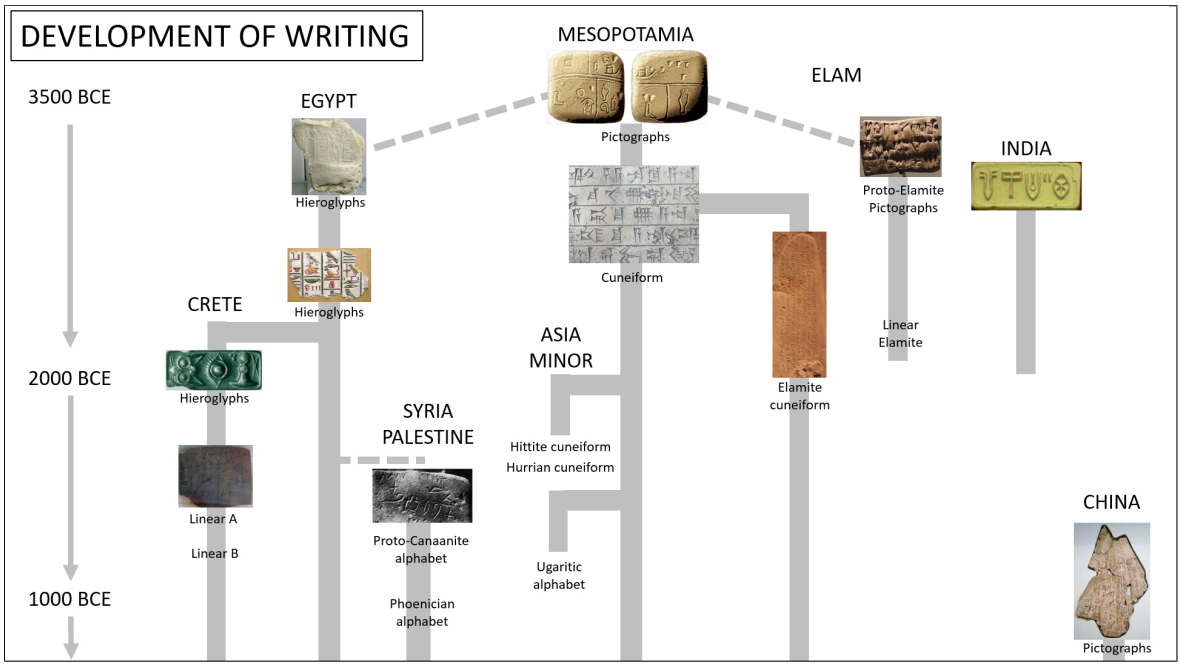


Figure 1.2: Standard reconstruction of the development of writing. There is a possibility that the Egyptian script was invented independently from the Mesopotamian script [40].

than 35,000 years ago, we would still recognize what they were trying to communicate. As seen in Figure 1.1. Some recent theories indicate that language evolved in humans as a form of communication since the tools to communicate visually were non-ubiquitous [96]. If a wolf was nearby and someone started drawing it would already be too late to save anyone.

The tools at the time didn't enable us to draw a wolf fast enough to stop us from being eaten. Thus humans invented language as a means to communicate [96]. If we suddenly invent a time machine and go back thousands of years to the caves with the earliest cave paintings, we would still understand the bull and the impending danger to the human settlement. However if we try to communicate in either English or any other modern language with the prehistoric cave dwellers neither of us would be able to communicate with the other. However, if we showed them a TikTok video of another human, even the prehistoric cave dwellers would most likely be able to identify with the emotions shown in the video.

Spoken language is physically manifested as written language. Theories suggest that human beings developed language around 35,000 BC [40] as evidenced by cave paintings from the period of the Cro-Magnon Man (50,000-30,000 BC) which appear to express concepts concerning daily life. These pictographs suggest a language because, in some instances, they seem to tell a story (say, of a hunting expedition in which specific events



Figure 1.3: The Kish tablet from Sumer, with pictographic writing. This may be the earliest known writing, 3500 BC. Ashmolean Museum

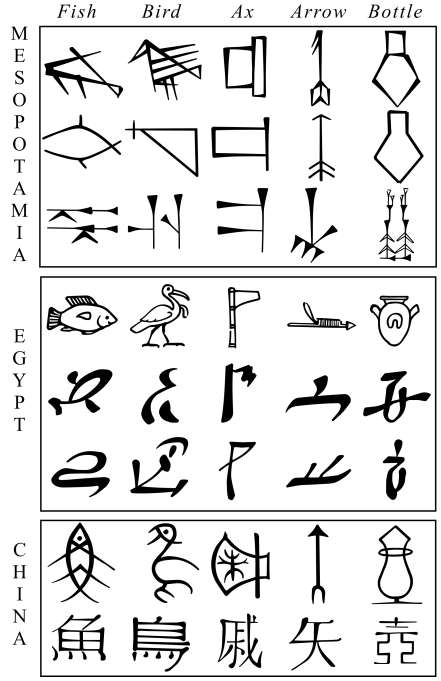
occurred) rather than being mundane pictures of animals and people.

Language in its written form, however, does not emerge until its invention in Sumer, southern Mesopotamia around 3500 -3000 BC [40]. This early version of writing was referred to as cuneiform and consisted of making specific marks in wet clay with a reed implement. The writing system of the Egyptians was already in use before the rise of the Early Dynastic Period (3150 BC) [40]. This Egyptian writing system is thought to have developed from Mesopotamian cuneiform (though this theory is disputed) and came to be known as heiroglyphics.

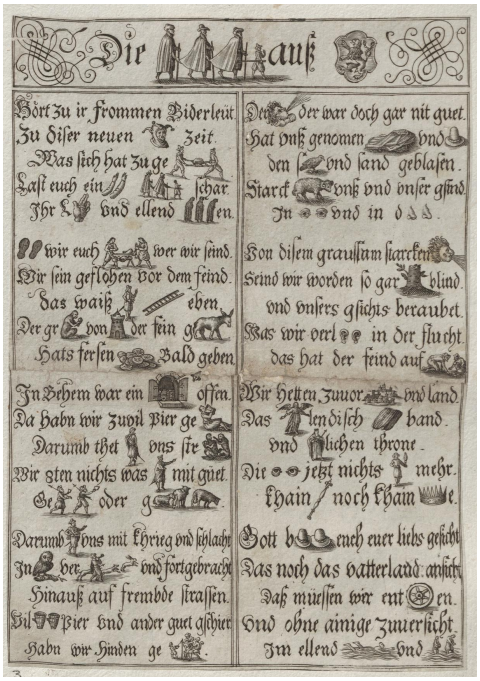
The Greeks conceptualized the phoenetic writing systems ("phoenetic" from the Greek phonein - "to speak clearly"), and later the Romans, came from Phoenicia [40]. The Phoenician writing system, though quite different from that of Mesopotamia, still owes its development to the Sumerians and their advances in the written word. Writing was also developed in Mesoamerica by the Maya around 500 BC [40] and, also independently, by the Chinese. All of these developments were independent of the Near East or Europe.

Writing was first invented by the Sumerians as a means of long-distance communication which was necessitated by trade. The rise of cities in Mesopotamia often lacked resources in the nearby region. This lack of resources necessitated long-distance trade. Long distance trade required people to communicate across the expanses between cities or regions.

Pictographs were the earliest form of writing – objects represented by symbols – and served to aid in remembering such things as which parcels of grain had gone to which destination or how many sheep were needed for events like sacrifices in the temples. These pictographs were impressed onto wet clay which was then dried, and these became official records of commerce. As the most popular beverage in ancient Mesopotamia was beer, many of the earliest records have to do with the sale of beer [40] . With pictographs, one



(a) Comparative evolution of Cuneiform, Egyptian and Chinese characters



(b) A German rebus, circa 1620

Figure 1.4: Visual communication complements communication using characters in a language

could tell how many jars or vats of beer were involved in a transaction but not necessarily what that transaction meant.

To enable humans to convey concepts more complex than financial transactions or lists of items, a more elaborate writing system was required, and this was developed in the Sumerian city of Uruk (3200 BC) [40]. Pictograms, though still in use, gave way to phonograms – symbols which represented sounds – and those sounds were the spoken language of the people of Sumer. Phonograms, enabled humans to easily convey precise meaning and so, in the example of the two sheep and the temple of Inanna [40], one could now make the distinction whether the sheep were going to or coming from the temple, whether they were living or dead. It also enabled to explicitly note what role they played in the life of the temple. Previously, one had only static images in pictographs showing objects like sheep and temples. Phonograms enabled humans a dynamic means of conveying motion to or from a location.

As we see in Fig. 1.2 and Fig. 1.3 some of the earliest found evidence of language were pictographic. Fig. 1.3 shows the Kish tablet from Sumer. This was one of the earliest known human writing found by humans. We can clearly identify the hand like symbol depicted multiple times in the tablet. As we go through the journey of the evolution of written language,

we see progressively the letters becoming more abstract. Fig. 1.4a demonstrates how some of the earliest pictographs were made progressively more abstract. The same trend was observed in the cultures of Mesopotamia, Egypt and China. Fig. 1.4b depicts the Rebus method. It combines the use of illustrated pictures with individual letters to depict words or phrases. For example: the word "been" might be depicted by a rebus showing an illustrated bumblebee next to a plus sign (+) and the letter "n". It was a favourite form of heraldic expression used in the Middle Ages to denote surnames. The rebus method formed an intermediate system which used both phonetic characters as well as images for communication.

Phonetic communication as a bottleneck Language as a means of communication has immense benefits. The common cultural shared context enables a much higher bandwidth communication using very few bits of information. The common cultural context of a language ensures that not everything has to be explicitly elucidated, while still allowing effective communication. However, verbal and written communication's biggest strength is also a weakness. When we talk about our Italian holiday, it can mean so many different things to different people. Some would focus on the feel of the air, others on the mediterranean weather, while some on piazzas. Some would envision the passion for the Ferrari F1 team while others might imagine the obvious taste of the delicious pizzas. Certain communities such as the Sami people of the Arctic Lapland might have no shared context and would have a hard time even imagining it. All of the above is compressed in a single phrase titled "Italian holiday". In the previous centuries phonetic language emerged as the dominant media for human communication. It allowed us to write books, elaborate essays and theses to communicate intricate ideas. It helped to herald revolutions [99] and to unify people with cultural identities. However, armed with extremely powerful cameras and even more powerful machine learning systems visual communication is complementing text based communication in a multitude of domains. Business meetings without involving infographics in Powerpoint presentations are unimaginable. A major portion of communication on the internet involves the use of photos and videos to complement the accompanying text. This trend indicates that in the 21st century visual communication would play a leading role in shaping and complementing dominant forms of phonetic language based communication. These trends indicate that visual communication is back as a dominant form of human communication.

1.2 Why is visual communication important?

- **Expressing Emotions:** Emotions are hard to express via text on the internet. Especially when we are trying to express sarcasm or a joke. This led to the advent of sites such as giphy for expressing feelings visually.
- **Saves time:** As the proverb says: "A picture is worth a thousand words". A picture communicates details which would take an essay to express via text. We also have seen a shift to more visual media in social media as well. Internet chatrooms led to Facebook which further paved the way for more visually appealing social media such as Snapchat, Instagram and Tiktok.
- **Promotes consistency:** In different languages concepts are represented differently. Several cultural norms of communication also changes from the orient to the west.
- **Provides Clarity:** Huge amounts of information is lost while communicating via text. When we talk about our "Italian Holiday", there is so much information encoded within those 2 words. For some it might mean the delicious food in Sicily, for others it might mean the brilliant shine of the mediterranean sea. However glancing through the photos gives us such a better description of the context being spoken of than just the 2 words: "Italian Holiday".
- **Increases retention:** Humans are more inclined to think visually. Hence dreams are almost always vivid visuals rather than pages of essays. We remember visual ads much better than ads with lots of text.
- **Universally understood:** A picture of a pizza gives a sense of the toppings and universally understood in any culture and any country. However explaining pizza toppings along with the crust is a huge challenge everyone has inevitably faced.
- **More impactful:** Images/ videos are more persuasive. Hence in big banner ads or Superbowl ads its almost always a video to drive call to action. Even platforms such as Facebook which originally started with text based check-ins has shifted to image or video based ads.

Current state of the art in Visual Communication GIFs are the go to medium for expressing our emotions at the moment. Platforms such as Instagram, Snapchat and Tiktok have changed how we communicate visually. Tiktok allows us to make videos which would have taken hours of work on Adobe Premier Pro. Instagram and the modern smartphone cameras have rendered the bulky DSLRs obsolete. We now have Google Lens which can identify anything you don't know in a matter of seconds.

Future of visual communication Interesting products are coming to the forefront. Interactive product design in Virtual reality via the Oculuses and the Holo lenses of the world. Just like when you search "How to" on google, and it starts suggesting "how to give a swashbuckling speech on Toastmasters". Products from Adobe are coming up which starts autocompleting your sketches and also photo-realisitically renders it. So with a few strokes of your stylus you can design & customise your own luxury sports car or your new set of fancy boots. In the future when you can 3D print items, you can quickly design your furniture assisted with AI and just 3D print it at home. When you watch a movie, you'd like to see yourself as the protagonist on the top of Burj Khalifa rather than Mr Tom Cruise. Ads in the future would feature you in that Armani suit and give you the fear or missing out leading your fear to buy that Armani suit or dress.

The tools for visual communication are becoming an even faster media of communication than traditional forms of text. It thus becomes imperative to design the future of visual communication tools enhanced by data driven learning approaches. Generative models learn the underlying generative function that best fits the data. Once a generative model is trained it should ideally support generation of new samples from the underlying data distribution. Generative models spearheaded by Generative Adversarial Networks [49] and Variational Autoencoders [76] introduced applications which previously had been deemed impossible.

1.3 Taxonomy of Generative Models

Typical generative models are trained with maximum likelihood learning. The learnt density function can either be implicit as in the case of Generative Adversarial Networks[49] and Generative Stochastic Networks [3]. On the other hand the density can be explicitly modeled. Variational Autoencoders [76] approximate the density function using the Evidence Lower Bound (ELBO). In case of autoregressive models such as PixelRNN[142] the density is

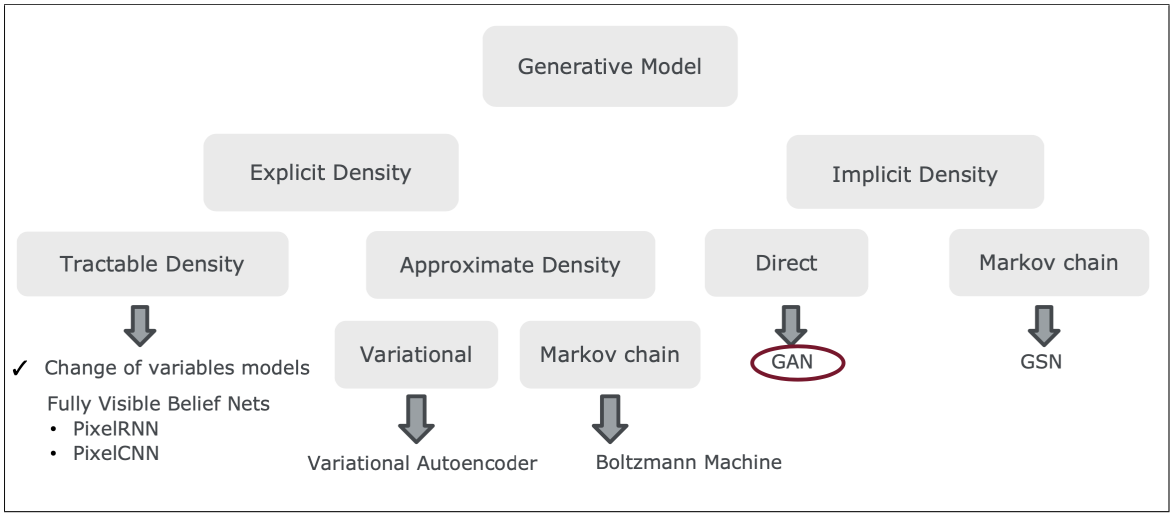


Figure 1.5: Taxonomy of Generative Models, this thesis mainly draws upon Generative Adversarial Networks

tractable. Continuous Normalizing Flows [16, 52] is also a form of explicit density estimation technique. These are schematically represented in Fig. 1.5.

Deep generative models models are function approximators, parametrized by a neural network with several layers. These networks are trained to approximate a complicated high dimensional probability distribution. On successful training, these trained models could either be used to estimate likelihood of a particular sample under the distribution. The trained deep generative model often allows us to draw samples from the distribution. Deep generative models has been one of the most active areas of research in recent years because of its incredible promise. Although, the promise hasn't been fulfilled and hence the increased research interest. In recent years three major avenues of deep generative model has taken centre-stage amongst many others. Variational Autoencoders(VAEs), Normalizing Flows(NFs) and Generative Adversarial Networks(GANs) are the ones that have shown the highest quality results. These are the models which have shown the highest resolution which can be potentially used for effective and reliable Visual Communication applications. Markov chain based models such as Generative Stochastic Networks(GSN) and Boltzmann machines have complicated training procedures which often don't converge within reasonable time frames for larger resolution/ more complicated image distributions. Models which deal with explicit tractable density functions such as PixelRNN/ PixelCNN are too slow for scalable image generations since it has to do O(10^6) forward passes through the network to render a high resolution (1024x1024) image.



Figure 1.6: DALL-E [122] on the left and BigGAN[11] images on the right

1.4 Generative Models for Visual Communication

As machine learning models start being more widespread in computer graphics, machine learning is starting to assist creative artists as a tool. Mario Klingemann's work [79] on generating surreal images with BigGAN[11] or Katherine Crowson's [22] work on VQGAN[41] + CLIP[121]. OpenAI's DALL-E [122] allows us to define images which only exists in our imagination. Such as a snail like harp as shown on the left of Fig. 1.6. Interpolating between classes also gives interesting results as shown in the right of the figure. As these techniques start getting more mature, we would be able to do more complicated and creative tasks. Applications such as interactively generating small video clips akin to GIFs [131] is starting to gain traction.

1.5 Major Challenges

There exists multitude of challenges before these techniques become widely used in products of daily use. With the advent of the pandemic, companies such as Facebook and Nvidia are investing hugely into the concepts of Metaverse and virtual worlds in general. It would just a matter of time before these techniques are used in everyday products for complementing traditional forms of communication in an ever virtual environment. Most of these visions of the future of communication in virtual worlds would be powered by



Figure 1.7: Angel of Air Unreal Engine [80] on the left and Coral Reef City [2] on the right

advances in Generative Models.

Let's delve deeper into some of the major challenges in the sphere of research in Generative Models. Specifically, we would consider the challenges which this thesis addressed with its contributions.

Mode Collapse in GANs: Mode Collapse was a huge challenge in the training of GANs. The training objective of GANs penalizes quality of generations much more than the overall coverage of the model in terms of the modes of the distribution. This caused the generators to pick and choose a set of modes it would master and not care about the other modes of the distribution. In terms of the original GAN game, the discriminator was not penalized heavily under such a scenario, hence the generator coverage in terms of the modes was limited.

Improper Conditioning in Class Conditional GANs: Conditional GANs worked remarkably well in the case of Pix2pix [66] and CycleGAN [164]. The design of the U-Net like architecture had been its greatest strength as well as its biggest weakness. The conditioning provided to the U-Net like generator network was a high dimensional image. This made it hard to add low dimensional conditioning for more granular control of the generation process. This posed a problem, since a different model had to be trained for edges2handbags and edges2shoes, although they were doing very similar tasks. However, because of the network design, it was extremely hard to inculcate such a switch reliably into the network.

Assistance in Human Creativity: Sketch-RNN [54] introduced a novel framework of autocompleting user's doodles. Although it was interesting at the time, it came with its own set of flaws. The doodles were non realistic and didn't resemble how humans might want to design or communicate effectively. These doodles hardly resembled realistic sketches of the respective classes. This meant that the resultant sketches couldn't easily be photorealistically rendered using Pix2pix kind of networks.

If Adobe's illustrator had an AI assistant for helping out novice users it would make art much more fun and faster. The assistant's guardrails would ensure that the creations are more realistic in terms of the underlying data distribution.

Extremely slow training in Continuous Normalizing Flows: Neural ODEs[16] also enabled an alternate strategy for generative modeling known as Continuous Normalizing Flows[52]. It used continuous transformation of the distribution from a source distribution to a target distribution. However training these models were excessively slow even on the high end Nvidia GPUs. The time taken to converge was orders of magnitude slower for training high resolution GANs or VAEs as well. Thus, in order for these Continuous Normalizing Flow based models to be competitive against other classes of Generative Models it is critical to enable faster convergence.

1.6 Key Ideas

The key ideas introduced in this thesis are:

Reformulation of the GAN framework in the multi-generator scenario: In Chapter 3 we introduce a multi-generator formulation for increasing the capacity of GANs. In this scenario, if the discriminator's objective is not modified, the different generators are not incentivized to learn different distributions thus wasting parameters. The implicit objective for the discriminator to also identify the generator forces the discriminator to push the different generators towards diverse modes. This distribution of modes between the different generators helps in correct attribution by the disciminator in case it considers the image a fake one.

Gating of residual neural networks: In Chapter 4 we introduce a simplification of the proposed multi-generator framework. In practice, the GPU capacity limits the number of generators that can be introduced in the multi-generator framework. Requiring a different

model for each task is prohibitively expensive. For instance requiring a different model for converting edges to handbags and a different model to convert from edges to shoes is a waste of resources. This becomes relevant especially in resource constrained applications especially in the context of mobile devices. Thus we introduced a simple strategy to modulate the activations of various residual blocks based on the class conditioning. This allows us to have a single model to generate multiple different classes. It also allows for a multi-task model where the task conditions the activations of the residual blocks.

Auto-completion of sketches using a simple data occlusion strategy: In Chapter 4 apart from the gating strategy we also introduce a sketch recommendation system. Sketch-RNN [54] was interesting as a proof of concept but it wasn't trained on sketches which could be photorealistically rendered. Since the completions of Sketch-RNN were quite rough it couldn't be used for more complex designing systems. For instance it couldn't be used for designing cars or customizing your shoes. Thus we introduced a simple strategy for creating partial sketches by simulating partial canvases and occluding the rest of the sketch. This allowed a simple model which generates the full sketch given the partial sketch.

Unified framework for pose recognition and pose guided image generation: In Appendix C we propose a semi supervision based strategy for deep generative modelling for human body analysis This allows a single unified framework for pose recognition as well as pose guided image generation. This allowed pose transfer of a subject within a single unified framework.

Simple strategy of randomizing limits of ODE integration: In Chapter 5 we introduce a simple regularization strategy to improve the performance of Neural ODEs in terms of faster computation within the constraints of the underlying ODE solver. Our experiments showed that by simply randomizing the end time of the integration leads to simpler flow trajectories in the generative modeling scenario.

1.7 Thesis Outline & Contributions

Chapter 2 discusses the technical background necessary for the understanding of the technical contributions. Chapter 3 (MAD-GAN) discusses the approach to alleviate some of the mode collapse issues in GANs with the introduction of multiple generators and a novel

training strategy to implicitly distribute the modes among the various generators. Chapter 4 (iSketchNFill) discusses an approach to tackle the problem of proper conditioning of class conditional GANs with a new framework of novel residual network based design. It also uses a simple strategy to autocomplete user's partial sketches and interactively also photorealistically renders the auto-completed sketches. Appendix C discusses the DGPose paper for deep generative modelling for human body analysis. This work proposed a unified framework for pose recognition as well as pose guided image generation. Chapter 5 (STEER) discusses an elegant strategy to speed up the training of Neural ODEs by randomly sampling the limits of the integration by ODE. This regularization also increases the accuracy in certain sequence prediction tasks where the sequence is irregularly spaced.

2 Background

2.1 Generative Adversarial Networks

Here we present a brief review of GANs [49]. Given a set of samples $\mathcal{D} = (x_i)_{i=1}^n$ from the true data distribution p_d, the GAN learning problem is to obtain the optimal parameters θ_g of a generator $G(z; \theta_g)$ that can sample from an approximate data distribution p_g, where $z \sim p_z$ is the prior input noise (For example samples from a normal distribution). In order to learn the optimal θ_g, the GAN objective (Eq. (3.1)) employs a discriminator $D(x; \theta_d)$ that learns to differentiate between a *real* (from p_d) and a *fake* (from p_g) sample x. The overall GAN objective is:

$$\min_{\theta_g} \max_{\theta_d} V(\theta_d, \theta_g) := \mathbb{E}_{x \sim p_d} \log D(x; \theta_d) + \mathbb{E}_{z \sim p_z} \log \Big(1 - D(G(z; \theta_g); \theta_d)\Big) \tag{2.1}$$

The above objective is optimized in a block-wise manner where θ_d and θ_g are optimized one at a time while fixing the other. For a given sample x (either from p_d or p_g) and the parameter θ_d, the function $D(x; \theta_d) \in [0, 1]$ produces a score that represents the probability of x belonging to the true data distribution p_d (or probability of it being real). The objective of the discriminator is to learn parameters θ_d that maximizes this score for the true samples (from p_d) while minimizing it for the fake ones $\tilde{x} = D(z; \theta_g)$ (from p_g). In the case of generator, the objective is to minimize $\mathbb{E}_{z \sim p_z} \log \Big(1 - D(G(z; \theta_g); \theta_d)\Big)$, equivalently maximize $\mathbb{E}_{z \sim p_z} \log D(G(z; \theta_g); \theta_d)$. Thus, the generator learns to maximize the scores for

the fake samples (from p_g), which is exactly the opposite to what discriminator is trying to achieve. In this manner, the generator and the discriminator are involved in a minimax game where the task of the generator is to maximize the mistakes of the discriminator. Theoretically, at equilibrium, the generator learns to generate real samples, which means $p_g = p_d$.

2.2 Variational Autoencoders

Deep generative models (DGMs) come in two broad flavours – Variational Autoencoders (VAEs) [76, 124], and Generative Adversarial Networks (GANs) [51]. In both cases, the goal is to learn a generative model $p_\theta(\mathbf{x}, \mathbf{z})$ over data $\mathbf{x}$ and latent variables $\mathbf{z}$, with parameters θ. Typically the model parameters θ are represented in the form of a neural network.

VAEs express an objective to learn the parameters θ that maximise the marginal likelihood (or evidence) of the model denoted as $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})dz$. They introduce a conditional probability density $q_\phi(\mathbf{z}|\mathbf{x})$ as an approximation to the unknown and intractable model posterior $p_\theta(\mathbf{z}|\mathbf{x})$, employing the variational principle in order to optimise a surrogate objective $\mathcal{L}(\phi, \theta; \mathbf{x})$, called the evidence lower bound (ELBO), as

$$\begin{aligned} \log\, p_\theta(\mathbf{x}) &\geq \mathcal{L}_{\text{VAE}}(\phi, \theta; \mathbf{x}) \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]. \end{aligned} \tag{2.2}$$

The conditional density $q_\phi(\mathbf{z}|\mathbf{x})$ is called the recognition or inference distribution, with parameters ϕ also represented in the form of a neural network. Lastly, VAEs also admit an extension to *conditional* generative models (CVAEs) [137], simply by incorporating a conditioning variable $\mathbf{y}$, to derive

$$\begin{aligned} \log\, p_\theta(\mathbf{x}|\mathbf{y}) &\geq \mathcal{L}_{\text{CVAE}}(\phi, \theta; \mathbf{x}|\mathbf{y}) \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})}\left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z}|\mathbf{y})}{q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})}\right]. \end{aligned} \tag{2.3}$$

On the other hand, in the context of structured semi-supervised learning, one can factor the latent variables into unstructured or non-interpretable variables $\mathbf{z}$ and structured or interpretable variables $\mathbf{y}$ without loss of generality [77, 132]. For learning in this framework, the objective can be expressed as the combination of supervised and unsupervised objectives. Let $\mathcal{D}_u$ and $\mathcal{D}_s$ denote the unlabelled and labelled subset of the dataset $\mathcal{D}$, and let the

joint recognition network factorise as $q_\phi(\mathbf{y}, \mathbf{z}|\mathbf{x}) = q_\phi(\mathbf{y}|\mathbf{x})q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})$. Then, the combined objective summed over the entire dataset corresponds to

$$\begin{aligned}\mathcal{L}_{\text{SS}}(\theta, \phi; \mathcal{D}) = &\sum_{\mathbf{x}_u \in \mathcal{D}_u} \mathcal{L}_u(\theta, \phi; \mathbf{x}_u) \\ &+ \gamma \sum_{(\mathbf{x}_s, \mathbf{y}_s) \in \mathcal{D}_s} \mathcal{L}_s(\theta, \phi; \mathbf{x}_s, \mathbf{y}_s)\end{aligned} \tag{2.4}$$

where $\mathcal{L}_u$ and $\mathcal{L}_s$ are defined as

$$\mathcal{L}_u(\theta, \phi; \mathbf{x}_u) = \mathcal{L}_{\text{VAE}}(\theta, \phi; \mathbf{x}_u), \text{ and} \tag{2.5}$$

$$\begin{aligned}\mathcal{L}_s(\theta, \phi; \mathbf{x}_s, \mathbf{y}_s) = &\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_s, \mathbf{y}_s)} \left[\log \frac{p_\theta(\mathbf{x}_s, \mathbf{z}|\mathbf{y}_s)}{q_\phi(\mathbf{z}|\mathbf{x}_s, \mathbf{y}_s)}\right] \\ &+ \alpha \log q_\phi(\mathbf{y}_s|\mathbf{x}_s),\end{aligned} \tag{2.6}$$

respectively. Here, the hyper-parameter γ (Eq. 2.4) controls the relative weight between the supervised and unsupervised dataset sizes, and α (Eq. 2.6) controls the relative weight between generative and discriminative learning.

Note that by the factorisation of the generative model, VAEs necessitate the specification of an explicit likelihood function $p_\theta(\mathbf{x}|\mathbf{z})$, which can often be difficult. GANs, on the other hand, attempt to sidestep this requirement by learning a surrogate to the likelihood function, while avoiding the learning of a recognition distribution. Here, the generative model $p_\theta(\mathbf{x}, \mathbf{z})$, viewed as a mapping $G : \mathbf{z} \mapsto \mathbf{x}$, is setup in a two-player minimax game with a "discriminator" $D : \mathbf{x} \mapsto \{0, 1\}$, whose goal is to correctly identify if a data point $\mathbf{x}$ came from the generative model $p_\theta(\mathbf{x}, \mathbf{z})$ or the true data distribution $p(\mathbf{x})$. Such objective is defined as

$$\begin{aligned}\mathcal{L}_{\text{GAN}}(D, G) = &\mathbb{E}_{p(\mathbf{x})} \left[\log D(\mathbf{x})\right] \\ &+ \mathbb{E}_{p_\theta(\mathbf{z})} \left[1 - \log D(G(\mathbf{z}))\right].\end{aligned} \tag{2.7}$$

In fact, in our structured model, generation is defined as a function of pose and appearance as $G(\mathbf{y}, \mathbf{z})$. Crucially, learning a customised approximation to the likelihood can result in a much higher quality of generated data, particularly for the visual domain [73].

A more recent family of DGMs, VAEGANs [85], bring together these two different approaches into a single objective that combines both the VAE and GAN objectives directly as

$$\mathcal{L} = \mathcal{L}_{\text{VAE}} + \mathcal{L}_{\text{GAN}}. \tag{2.8}$$

This marries better the likelihood learning with the inference-distribution learning, providing a more flexible family of models.

2.3 Neural ODEs

Neural ODEs can be interpreted as a continuous equivalent of Residual networks [57]. The transformation of a residual layer is $\mathbf{z}_{t+1} = \mathbf{z}_t + f_t(\mathbf{z}_t)$, where the representation at layer t is $\mathbf{z}_t$ and $f_t(\mathbf{z}_t)$ is a function parametrized by a neural network. The output dimension of the function $f_t(\mathbf{z}_t)$ should be the same as the input dimension to allow it to be added to the previous layer's features $\mathbf{z}_t$. This transformation can be rewritten as

$$\frac{\mathbf{z}_{t+\delta_t} - \mathbf{z}_t}{\delta_t} = f_t(\mathbf{z}_t), \quad \delta_t = 1, \tag{2.9}$$

which can be interpreted as finite difference approximation of the derivative. In contrast to traditional deep neural networks where $\delta_t = 1$ is fixed, Neural ODEs [16] introduced a continuous version by analysing Eq. 5.1 in the limit $\delta_t \to 0$. The resulting equation becomes

$$\frac{d\mathbf{z}(t)}{dt} = f(\mathbf{z}(t), t). \tag{2.10}$$

To transform an input $\mathbf{x}$ to an output $\mathbf{y}$, Residual networks use a sequence of k functions to transform the representation. In an analogous manner, Neural ODEs evolve the representation from the initial time $t = t_0,\ \mathbf{z}(t_0) = \mathbf{x}$, to the final time $t = t_1,\ \mathbf{z}(t_1) = \mathbf{y}$. The representation $\mathbf{z}(t)$ at any given time t can then be obtained by solving Eq. 5.2 using an off-the-shelf ODE solver with the function $f(\mathbf{z}(t), t)$ parameterized by the neural network. To update the parameters of the function $f(\mathbf{z}(t), t)$ such that $\mathbf{z}(t_1) \approx \mathbf{y}$ we backpropagate through the operations of the ODE solver to obtain the appropriate gradients.

2.3.1 Continuous Normalizing Flows

Continuous normalizing flow based generative models [52, 16] provide an alternate perspective to generative modeling. Let $p(\mathbf{x})$ be the likelihood of a datapoint $\mathbf{x}$ and $p(\mathbf{z})$ be the likelihood of the latent $\mathbf{z}$. FFJORD [52] showed that under certain conditions the change in log probability also follows a differential equation $\frac{d\log(p(\mathbf{z}(t)))}{dt} = -\text{tr}\left(\frac{df}{d\mathbf{z}(t)}\right)$. Thus the total change in log probability can be obtained by integrating the equation which leads to

$$\log p(\mathbf{z}(t_1)) = \log p(\mathbf{z}(t_0)) - \int_{t_0}^{t_1} \text{tr}\left(\frac{df}{d\mathbf{z}(t)}\right) dt. \tag{2.11}$$

This allows to compute the point $\mathbf{z_0}$ which generates a given datapoint $\mathbf{x}$, and $\log p(\mathbf{x})$ under the model by solving the initial value problem:

$$\underbrace{\begin{bmatrix} \mathbf{z_0} \\ \log p(\mathbf{x}) - \log p_{\mathbf{z_0}}(\mathbf{z_0}) \end{bmatrix}}_{\text{solutions}} = \underbrace{\int_T^{t_0} \begin{bmatrix} f(\mathbf{z}(t), t; \theta) \\ -\text{tr}\left(\frac{df}{d\mathbf{z}(t)}\right) \end{bmatrix} dt,}_{\text{dynamics}} \qquad \underbrace{\begin{bmatrix} \mathbf{z}(T) \\ \log p(\mathbf{x}) - \log p(\mathbf{z}(T)) \end{bmatrix} = \begin{bmatrix} \mathbf{x} \\ \mathbf{0} \end{bmatrix}}_{\text{initial values}}. \tag{2.12}$$

The combined dynamics of $\mathbf{z}(t)$ and the log-density of the sample is integrated backwards in time from T to t_0. The log likelihood $\log p(\mathbf{x})$ can be computed by adding $\log p_{\mathbf{z_0}}(\mathbf{z_0})$ to the solution of Eq. 5.8. A major drawback of continuous normalizing flow based models is extremely slow convergence. STEER regularization enables faster convergence by sampling $T \sim \text{Uniform}(t_1 - b, t_1 + b)$ at each iteration.

3

MADGAN: Multi-Agent Diverse Generative Adversarial Networks

Abstract

We propose MAD-GAN, an intuitive generalization to the Generative Adversarial Networks (GANs) and its *conditional variants* to address the well known problem of *mode collapse*. First, MAD-GAN is a multi-agent GAN architecture incorporating multiple generators and one discriminator. Second, to enforce that different generators capture diverse high probability modes, the discriminator of MAD-GAN is designed such that along with finding the real and fake samples, it is also required to identify the generator that generated the given fake sample. Intuitively, to succeed in this task, the discriminator must learn to push different generators towards different identifiable modes. We perform extensive experiments on synthetic and real datasets and compare MAD-GAN with different variants of GAN. We show high quality diverse sample generations for challenging tasks such as image-to-image translation and face generation. In addition, we also show that MAD-GAN is able to disentangle different modalities when trained using highly challenging diverse-class dataset (For example the dataset with images of forests, icebergs, and bedrooms). In the end, we show its efficacy on the unsupervised feature representation task. In Section 3.9, we introduce a similarity based competing objective (MAD-GAN-Sim) which encourages different generators to generate diverse samples based on a user defined similarity metric. We show its performance on the image-to-image translation, and also show its effectiveness on the unsupervised feature representation task.



Figure 3.1: Diverse-class data generation using MAD-GAN. Diverse-class dataset contains images from different classes/modalities (in this case, forests, icebergs, and bedrooms). Each row represents generations by a particular generator and each column represents generations for a given random noise input z. As shown, once trained using this dataset, generators of MAD-GAN are able to disentangle different modalities, hence, each generator is able to generate images from a particular modality.

3.1 Introduction

Generative models have attracted considerable attention recently. The underlying idea behind such models is to attempt to capture the distribution of high-dimensional data such as images and texts. Though these models are highly useful in various applications, it is computationally expensive to train them as they require intractable integration in a very high-dimensional space. This drastically limits their applicability. However, recently there has been considerable progress in *deep generative models* – conglomerate of deep neural networks and generative models – as they do not explicitly require the intractable integration, and can be efficiently trained using back-propagation algorithm. Two such famous examples are Generative Adversarial Networks (GANs) [49] and Variational Autoencoders [76].

In this work we focus on GANs as they are known to produce sharp and plausible images. Briefly, GANs employ a generator and a discriminator where both are involved in a minimax game. The task of the discriminator is to learn the difference between *real* samples (from true data distribution p_d) and *fake* samples (from generator distribution p_g). Whereas, the task of the generator is to maximize the mistakes of the discriminator. At convergence, the generator learns to produce real looking images. A few successful applications of GANs are video generation [146], image inpainting [115], image manipulation [162], 3D object generation [153], interactive image generation using few brush strokes [162], image super-resolution [89], diagrammatic abstract reasoning [46] and conditional GANs [105, 123].

Despite the remarkable success of GAN, it suffers from the major problem of *mode collapse* [4, 14, 18, 104, 127]. Though, theoretically, convergence guarantees the generator

learning the true data distribution. However, practically, reaching the true equilibrium is difficult and not guaranteed, which potentially leads to the aforementioned problem of mode collapse. Broadly speaking, there are two schools of thought to address the issue: (1) improving the learning of GANs to reach better optima [4, 104, 127]; and (2) explicitly enforcing GANs to capture diverse modes [14, 18, 94]. Here we focus on the latter.

Borrowing from the multi-agent algorithm [1] and coupled GAN [94], we propose to use multiple generators with one discriminator. We call this framework the Multi-Agent GAN architecture, as shown in Fig. 3.2. In detail, similar to the standard GAN, the objective of each generator here is to maximize the mistakes of the *common* discriminator. Depending on the task, it might be useful for different generators to share information. This is done using the initial layer parameters of generators. Another reason behind sharing these parameters is the fact that initial layers capture low-frequency structures which are almost the same for a particular type of dataset (for example, faces), therefore, sharing them reduces redundant computations. However, when the dataset contains images from completely different modalities, one can avoid sharing these parameters. Naively using multiple generators may lead to the *trivial solution* where all the generators learn to generate *similar* samples. To resolve this issue and generate different visually plausible samples capturing diverse high probability modes, we propose to modify the objective function of the discriminator. In the modified objective, along with finding the real and the fake samples, the discriminator also has to correctly identify the generator that generated the given fake sample. Intuitively, in order to succeed in this task, the discriminator must learn to push generations corresponding to different generators towards different identifiable modes. Combining the Multi-Agent GAN architecture with the diversity enforcing term allows us to generate diverse plausible samples, thus the name Multi-Agent Diverse GAN (MAD-GAN).

As an example, an intuitive setting where mode collapse occurs is when a GAN is trained on a dataset containing images from different modalities/classes. For example, a diverse-class dataset containing images such as forests, iceberg, and bedrooms. This is of particular interest as it not only requires the model to disentangle intra-class variations, it also requires inter-class disentanglement. Fig. 3.1 demonstrates the surprising effectiveness of MAD-GAN in this challenging setting. Generators among themselves are able to disentangle inter-class variations, and each generator is also able to capture intra-class variations.

In addition, we analyze MAD-GAN through extensive experiments and compare it with several variants of GAN. First, for the proof of concept, we perform experiments in controlled settings using synthetic dataset (mixture of Gaussians), and complicated

Stacked/ Compositional MNIST datasets with hand engineered modes. In these settings, we empirically show that our approach outperforms all other GAN variants we compare with, and is able to generate high quality samples while capturing large number of modes. In a more realistic setting, we show high quality diverse sample generations for the challenging tasks of *image-to-image translation* [66] (conditional GAN) and *face generation* [18, 120]. Using the SVHN dataset [107], we also show the efficacy of our framework for learning the feature representation in an unsupervised setting.

We also provide theoretical analysis of this approach and show that the proposed modification in the objective of discriminator allows generators to learn together as a mixture model where each generator represents a mixture component. We show that at convergence, the global optimum value of $-(k+1)\log(k+1) + k\log k$ is achieved, where k is the number of generators.

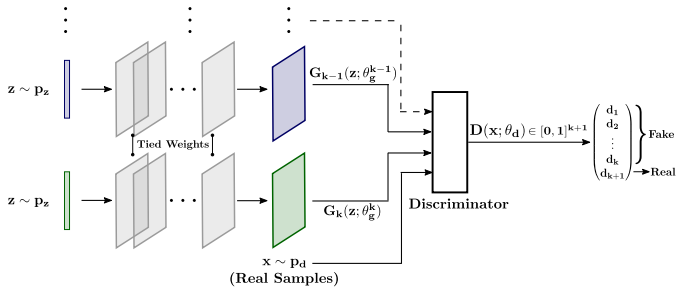


Figure 3.2: Multi-Agent Diverse GAN (MAD-GAN). The discriminator outputs $k+1$ softmax scores signifying the probability of its input sample being from either one of the k generators or the real distribution.

3.2 Related Work

The recent work called InfoGAN [18] proposed an information-theoretic extension to GANs in order to address the problem of mode collapse. Briefly, InfoGAN disentangles the latent representation by assuming a factored representation of the latent variables. In order to enforce that the generator learns factor specific generations, InfoGAN maximizes the mutual information between the factored latents and the generator distribution. Che et al. [14] proposed a mode regularized GAN (ModeGAN) which uses an encoder-decoder paradigm. The basic idea behind ModeGAN is that if a sample from the true data distribution p_d belongs to a particular mode, then the sample generated by the generator (fake sample) when the true sample is passed through the encoder-decoder is likely to belong to the same mode. ModeGAN assumes that there exists enough true samples from a mode for the generator to

be able to capture it. Another work by Metz et al. [104] proposed a surrogate objective for the update of the generator with respect to the unrolled optimization of the discriminator (UnrolledGAN) to address the issue of convergence of the training process of GANs. This improves the training process of the generator which in turn allow the generators to explore better coverage to true data distribution.

Liu et al. [94] presented Coupled GAN, a method for training two generators with shared parameters to learn the joint distribution of the data. The shared parameters guide both the generators towards similar subspaces but since they are trained independently on two domains, they promote diverse generations. Durugkar et al. [37] proposed a model with multiple discriminators whereby an ensemble of multiple discriminators have been shown to stabilize the training of the generator by guiding it to produce better samples.

W-GAN [5] is a recent technique which employs integral probability metrics based on the earth mover distance rather than the JS-divergences that the original GAN uses. BEGAN [9] builds upon W-GAN using an autoencoder based equilibrium enforcing technique alongside the Wasserstein distance. DCGAN [120] was a seminal technique which used a fully convolutional generator and discriminator for the first time along with the introduction of batch normalization thus stabilizing the training procedure, and was able to generate compelling generations. GoGAN [70] introduced a training procedure for the training of the discriminator using a maximum margin formulation alongside the earth mover distance based on the Wasserstein-1 metric. [7] introduced a technique and theoretical formulation stating the importance of multiple generators and discriminators in order to completely model the data distribution. In terms of employing multiple generators, our work is closest to [7, 94, 45]. However, while using multiple generators, our method explicitly enforces them to capture diverse modes.

3.3 Preliminaries

Here we present a brief review of GANs [49]. Given a set of samples $\mathcal{D} = (x_i)_{i=1}^n$ from the true data distribution p_d, the GAN learning problem is to obtain the optimal parameters θ_g of a generator $G(z; \theta_g)$ that can sample from an approximate data distribution p_g, where $z \sim p_z$ is the prior input noise (For examplesamples from a normal distribution). In order to learn the optimal θ_g, the GAN objective (Eq. (3.1)) employs a discriminator $D(x; \theta_d)$ that learns to differentiate between a *real* (from p_d) and a *fake* (from p_g) sample x. The

overall GAN objective is:

$$\min_{\theta_g} \max_{\theta_d} V(\theta_d, \theta_g) := \mathbb{E}_{x \sim p_d} \log D(x; \theta_d) + \mathbb{E}_{z \sim p_z} \log \Big(1 - D(G(z; \theta_g); \theta_d)\Big) \tag{3.1}$$

The above objective is optimized in a block-wise manner where θ_d and θ_g are optimized one at a time while fixing the other. For a given sample x (either from p_d or p_g) and the parameter θ_d, the function $D(x; \theta_d) \in [0, 1]$ produces a score that represents the probability of x belonging to the true data distribution p_d (or probability of it being real). The objective of the discriminator is to learn parameters θ_d that maximizes this score for the true samples (from p_d) while minimizing it for the fake ones $\tilde{x} = D(z; \theta_g)$ (from p_g). In the case of generator, the objective is to minimize $\mathbb{E}_{z \sim p_z} \log \Big(1 - D(G(z; \theta_g); \theta_d)\Big)$, equivalently maximize $\mathbb{E}_{z \sim p_z} \log D(G(z; \theta_g); \theta_d)$. Thus, the generator learns to maximize the scores for the fake samples (from p_g), which is exactly the opposite to what discriminator is trying to achieve. In this manner, the generator and the discriminator are involved in a minimax game where the task of the generator is to maximize the mistakes of the discriminator. Theoretically, at equilibrium, the generator learns to generate real samples, which means $p_g = p_d$.

3.4 Multi-Agent Diverse GAN

In the GAN objective, one can argue that the task of a generator is much harder than that of the discriminator as it has to produce real looking images to maximize the mistakes of the discriminator. This, along with the minimax nature of the objective raise several challenges for GANs [4, 14, 18, 104, 127]: (1) mode collapse; (2) difficult optimization; and (3) trivial solution. In this work we propose a new framework to address the first challenge of *mode collapse* by increasing the capacity of the generator while using well known tricks to partially avoid other challenges [4].

Briefly, we propose a *Multi-Agent GAN architecture* that employs multiple generators and one discriminator in order to generate different samples from high probability regions of the true data distribution. In addition, theoretically, we show that our formulation allows generators to act as a mixture model with each generator capturing one component.

3.4.1 Multi-Agent GAN Architecture

Here we describe our proposed architecture (Fig. 3.2). It involves k generators and one discriminator. In the case of homogeneous data (all the images belong to same class, For

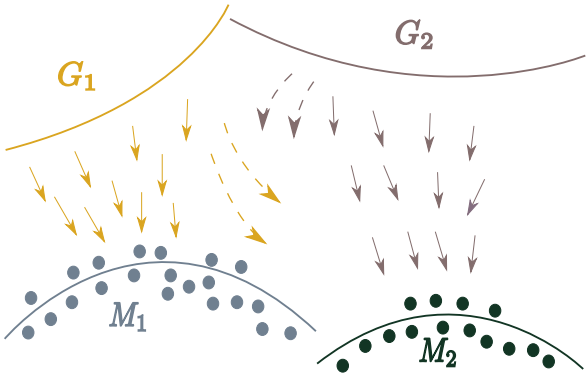


Figure 3.3: Visualization of different generators getting pushed towards different modes. Here, M_1 and M_2 could be a cluster of modes where each cluster itself contains many modes. The arrows abstractly represent generator specific gradients for the purpose of building intuition.

example faces or birds), we allow all the generators to share information by tying most of the initial layer parameters. This is essential to avoid redundant computations as initial layers of a generator capture low-frequency structures which are almost the same for a particular type of dataset. This also allows different generators to converge faster. However, in the case of *diverse-class* data (For example dataset with a mixture of different classes such as forests, icebergs *etc.*), it is necessary to avoid sharing these parameters to allow each generator to capture content specific structures. Thus, the extent to which one should share these parameters depends on the task at hand.

More specifically, given $z \sim p_z$ for the i-th generator, similar to the standard GAN, the first step involves generating a sample (for example, an image) $\tilde{x}_i$. Since each generator receives the same latent input sampled from the same distribution, naively using this simple approach may lead to the *trivial solution* where all the generators learn to generate similar samples. In what follows, we propose an intuitive solution to avoid this issue and allow the generators to capture diverse modes.

3.4.2 Enforcing Diverse Modes

Inspired by the discriminator formulation for the semi-supervised learning [127], we use a generator identification based objective function that, along with minimizing the score $D(\tilde{x};\theta_d)$, requires the discriminator to identify the generator that generated the given fake sample $\tilde{x}$. In order to do so, as opposed to the standard GAN objective function where the discriminator outputs a scalar value, we modify it to output $k+1$ soft-max scores. In more detail, given the set of k generators, the discriminator produces a soft-

max probability distribution over $k+1$ classes. The score at $(k+1)$-th index $(D_{k+1}(.))$ represents the probability that the sample belongs to the true data distribution and the score at $j \in \{1,\ldots,k\}$-th index represents the probability of it being generated by the j-th generator. Under this setting, while learning θ_d, we optimize the cross-entropy between the soft-max output of the discriminator and the Dirac delta distribution $\delta \in \{0,1\}^{k+1}$, where for $j \in \{1,\ldots,k\}$, $\delta(j) = 1$ if the sample belongs to the j-th generator, otherwise $\delta(k+1) = 1$. Thus, the objective of the discriminator, which is optimizing θ_d while keeping θ_g constant (refer Eq. (3.1)), is modified to:

$$\max_{\theta_d} \mathbb{E}_{x\sim p}\, H(\delta, D(x;\theta_d))$$

where, $Supp(p) = \cup_{i=1}^{k} Supp(p_{g_i}) \cup Supp(p_d)$ and $H(.,.)$ is the negative of the cross entropy function. Intuitively, in order to correctly identify the generator that produced a given fake sample, the discriminator must learn to push different generators towards different identifiable modes. However, the objective of each generator remains the same as in the standard GAN. Thus, for the i-th generator, the objective is to minimize the following:

$$\mathbb{E}_{x\sim p_d} \log D_{k+1}(x;\theta_d) + \mathbb{E}_{z\sim p_z} \log(1 - D_{k+1}(G_i(z;\theta_g^i);\theta_d))$$

To update the parameters, the gradient for each generator is simply computed as $\nabla_{\theta_g^i} \log(1 - D_{k+1}(G_i(z;\theta_g^i);\theta_d)))$. Notice that all the generators in this case can be updated in parallel. For the discriminator, given $x \sim p$ (can be real or fake) and corresponding δ, the gradient is $\nabla_{\theta_d} \log D_j(x;\theta_d)$, where $D_j(x;\theta_d)$ is the j-th index of $D(x;\theta_d)$ for which $\delta(j) = 1$. Therefore, using this approach requires *very minor modifications to the standard GAN optimization algorithm* and can be easily used with different variants of GAN. An intuitive visualization is shown in Fig. 3.3.

Theorem 1 shows that the above objective function actually allows generators to form a mixture model where each generator represents a mixture component and the global optimum of $-(k+1)\log(k+1) + k\log k$ is achieved when $p_d = \frac{1}{k}\sum_{i=1}^{k} p_{g_i}$. Notice that, at $k = 1$, which is the case with one generator, we obtain exactly the same Jensen-Shannon divergence based objective function as shown in [49] with the optimal value of $-\log 4$.

Theorem 1. *Given the optimal discriminator, the objective for training the generators boils down to minimizing*

$$KL\Big(p_d(x)||p_{avg}(x)\Big) + kKL\Big(\frac{1}{k}\sum_{i=1}^{k} p_{g_i}(x)||p_{avg}(x)\Big) - (k+1)\log(k+1) + k\log k \tag{3.2}$$

where, $p_{avg}(x) = \frac{p_d(x)+\sum_{i=1}^{k} p_{g_i}(x)}{k+1}$. *The above objective function obtains its global minimum if* $p_d = \frac{1}{k}\sum_{i=1}^{k} p_{g_i}$ *with the objective value of* $-(k+1)\log(k+1) + k\log k$.

Proof. The joint objective of all the generators is to minimize the following:

$$\mathbb{E}_{x\sim p_d} \log D_{k+1}(x) + \sum_{i=1}^{k} \mathbb{E}_{x\sim p_{g_i}} \log(1 - D_{k+1}(x))$$

Using Corollary 1, we substitute the optimal discriminator in the above equation and obtain:

$$\begin{aligned}
&\mathbb{E}_{x\sim p_d} \log \left[\frac{p_d(x)}{p_d(x) + \sum_{i=1}^{k} p_{g_i}(x)}\right] + \\
&\sum_{i=1}^{k} \mathbb{E}_{x\sim p_{g_i}} \log \left[\frac{\sum_{i=1}^{k} p_{g_i}(x)}{p_d(x) + \sum_{i=1}^{k} p_{g_i}(x)}\right] \\
&= \mathbb{E}_{x\sim p_d} \log \left[\frac{p_d(x)}{p_{avg}(x)}\right] + k\mathbb{E}_{x\sim p_g} \log \left[\frac{p_g(x)}{p_{avg}(x)}\right] \\
&- (k+1)\log(k+1) + k\log k
\end{aligned} \tag{3.3}$$

where, $p_g = \frac{\sum_{i=1}^{k} p_{g_i}}{k}$ and $p_{avg}(x) = \frac{p_d(x)+\sum_{i=1}^{k} p_{g_i}(x)}{k+1}$. Note that, Eq. (3.3) is exactly the same as Eq. (3.2). When $p_d = \frac{\sum_{i=1}^{k} p_{g_i}}{k}$, both the KL terms become zero and the global minimum is achieved. □

Corollary 1. *For fixed generators, the optimal distribution learned by the discriminator D has the following form:*

$$\begin{aligned}
D_{k+1}(x) &= \frac{p_d(x)}{p_d(x) + \sum_{i=1}^{k} p_{g_i}(x)}, \\
D_i(x) &= \frac{p_{g_i}(x)}{p_d(x) + \sum_{i=1}^{k} p_{g_i}(x)}, \forall i \in \{1, \cdots, k\}.
\end{aligned}$$

where, $D_i(x)$ *represents the* i*-th index of* $D(x;\theta_d)$, p_d *the true data distribution, and* p_{g_i} *the distribution learned by the* i*-th generator.*

Proof. For fixed generators, the objective function of the discriminator is to maximize

$$\mathbb{E}_{x\sim p_d} \log D_{k+1}(x) + \sum_{i=1}^{k} \mathbb{E}_{x_i\sim p_{g_i}} \log D_i(x_i)$$

where, $\sum_{i=1}^{k+1} D_i(x) = 1$ and $D_i(x) \in [0, 1], \forall i$. The above equation can be written as:

$$\int_x p_d(x) \log D_{k+1}(x) dx + \sum_{i=1}^{k} \int_x p_{g_i}(x) \log D_i(x) dx$$
$$= \int_{x \in p} \sum_{i=1}^{k+1} p_i(x) \log D_i(x) dx \tag{3.4}$$

where, $p_{k+1}(x) := p_d(x), p_i(x) := p_{g_i}(x), \forall i \in \{1, \cdots, k\}$, and $Supp(p) = \bigcup_{i=1}^{k} Supp(p_{g_i}) \bigcup Supp(p_d)$, Therefore, for a given x, the optimum of objective function defined in Eq. (3.4) with constraints defined above can be obtained using Proposition 1. □

Proposition 1. *Given* $\mathbf{y} = (y_1, \cdots, y_n)$*,* $y_i \geq 0$*, and* $a_i \in \mathbb{R}$*, the optimal solution for the objective function defined below is achieved at* $y_i^* = \frac{a_i}{\sum_{i=1}^{n} a_i}, \forall i$

$$\max_{\mathbf{y}} \sum_{i=1}^{n} a_i \log y_i, \quad s.t. \ \sum_{i}^{n} y_i = 1$$

Proof. The Lagrangian of the above problem is:

$$L(\mathbf{y}, \lambda) = \sum_{i=1}^{n} a_i \log y_i + \lambda(\sum_{i=1}^{n} y_i - 1)$$

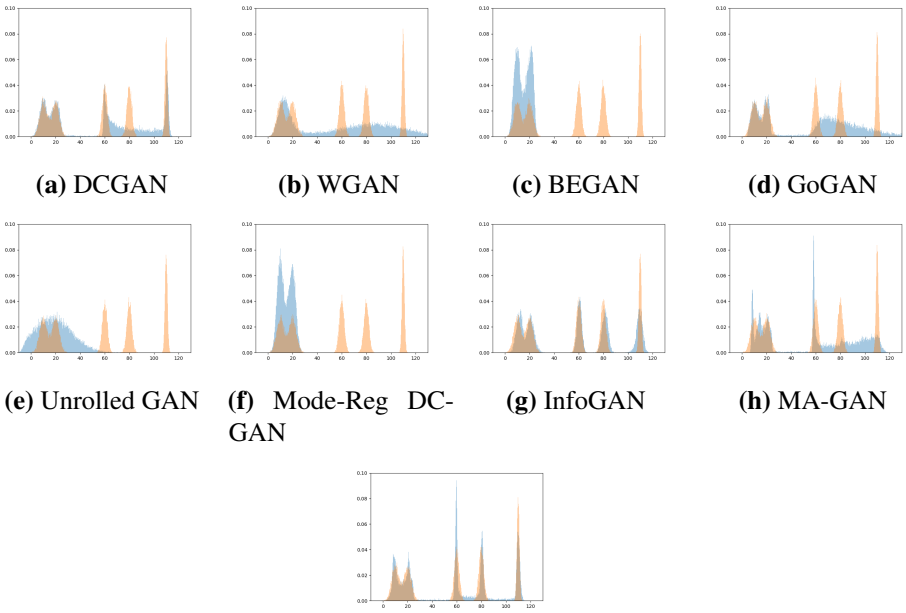
Differentiating w.r.t y_i and λ, and equating to zero,

$$\frac{a_i}{y_i} + \lambda = 0 \ , \ \sum_{i=1}^{n} y_i - 1 = 0$$

Solving the above two equations, we obtain $y_i^* = \frac{a_i}{\sum_{i=1}^{n} a_i}$. □

3.5 Experiments

We present an extensive quantitative and qualitative analysis of MAD-GAN on various synthetic and real-world datasets. First, we use a simple 1D mixture of Gaussians and also Stacked/Compositional MNIST dataset (1000 modes) to compare MAD-GAN with several known variants of GANs, such as DCGAN [120], WGAN [5], BEGAN [9], GoGAN [70], Unrolled GAN [104], Mode-Reg GAN [14] and InfoGAN [18]. Furthermore, we created another baseline, called *MA-GAN (Multi-Agent GAN)*, which is a trivial extension of GAN with multiple generators and one discriminator. As opposed to MAD-GAN, MA-GAN has a simple Multi-Agent architecture without modifications to the objective of the discriminator.

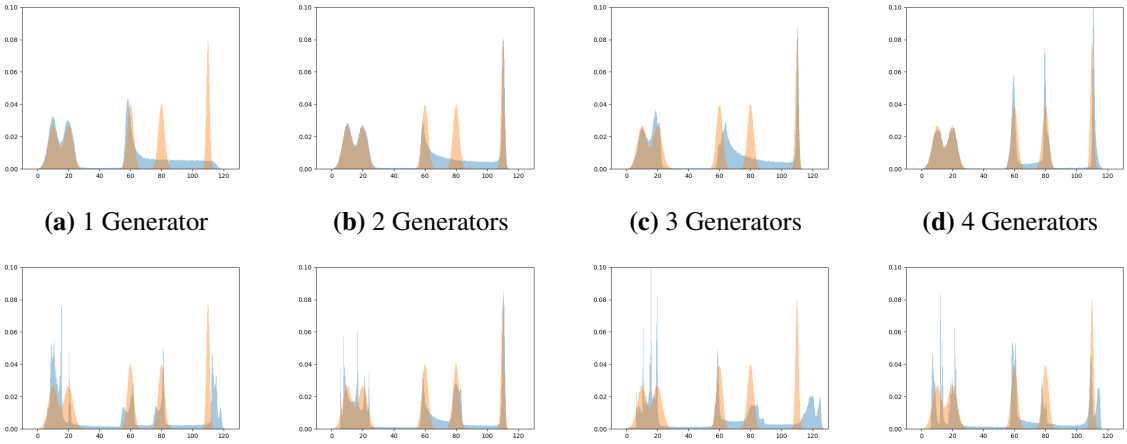


(a) DCGAN **(b)** WGAN **(c)** BEGAN **(d)** GoGAN

(e) Unrolled GAN **(f)** Mode-Reg DC-GAN **(g)** InfoGAN **(h)** MA-GAN

(i) MAD-GAN (Our)

Figure 3.4: A toy example to understand the behaviour of different GAN variants in order to compare with MAD-GAN (each method was trained for 198000 iterations). The orange bars show the density estimate of the training data and the blue ones for the generated data points. After careful cross-validation, we chose the bin size of 0.1.



(a) 1 Generator **(b)** 2 Generators **(c)** 3 Generators **(d)** 4 Generators

(e) 5 Generators **(f)** 6 Generators **(g)** 7 Generators **(h)** 8 Generators

Figure 3.5: A toy example to understand the behavior of MAD-GAN with different number of generators (each method was trained for $1,98,000$ iterations). The orange bars show the density estimate of the training data and the blue ones for the generated data points. After careful cross-validation, we chose the bin size of 0.1.

GAN Variants	**Chi-square**$(\times 10^5)$	**KL-Div**
DCGAN [120]	0.90	0.322
WGAN [5]	1.32	0.614
BEGAN [9]	1.06	0.944
GoGAN [70]	2.52	0.652
Unrolled GAN [104]	3.98	1.321
Mode-Reg DCGAN [14]	1.02	0.927
InfoGAN [18]	0.83	0.21
MA-GAN	1.39	0.526
MAD-GAN (Our)	**0.24**	**0.145**

Table 3.1: Synthetic experiment on 1D GMM (Fig. 3.4).

# Generators	**Chi-square** $(\times 10^7)$	**KL-Div**
1	1.27	0.57
2	1.38	0.42
3	3.15	0.71
4	0.39	0.28
5	3.05	0.88
6	0.54	0.29
7	0.97	0.78
8	4.83	0.68

Table 3.2: Synthetic experiment with different number of MAD-GAN generators (same setup as in Fig. 3.4).

This comparison allows us to understand the effect of explicitly enforcing diversity in the objective of the MAD-GAN. We use KL-divergence [83] and number of modes recovered [14] as the criterion for comparisons and show superior results compared to all the other methods. Additionally, we show diverse generations for the challenging tasks of *image-to-image translation* [66], *diverse-class data generation*, and face generation. It is non-trivial to devise a metric to evaluate diversity on these high quality generation tasks, so we perform qualitative assessment. Note that, the image-to-image translation objective is known to learn the delta distribution, thus, it is agnostic to the input noise vector. However, we show that MAD-GAN is able to produce highly plausible diverse generations for this task. In the end, we show the efficacy of MAD-GAN in unsupervised feature representation learning task.

In the case of InfoGAN [18], we varied the dimension of the categorical variable, depicting the number of modes, to obtain the best cross-validated results.

3.5.1 Non-Parametric Density Estimation

In order to understand the behavior of MAD-GAN and different state-of-the-art GAN models, we first perform a very simple synthetic experiment, much easier than generating high-dimensional complex images. We consider a distribution of 1D GMM [10] having five mixture components with modes at 10, 20, 60, 80 and 110, and standard deviations of 3, 3, 2, 2 and 1, respectively. While the first two modes overlap significantly, the fifth mode stands isolated as shown in Fig. 3.4. We train different GAN models using $200,000$ samples from this distribution and generate $65,536$ data points from each model. In order to compare the learned distribution with the ground truth distributions, we first estimate them using bins over the data points and create the histograms. These histograms are carefully created using different bin sizes and the best bin (found to be 0.1) is chosen. Then, we use Chi-square distance and the KL-divergence to compute distance between the two histograms. From Fig. 3.4 and Tab. 3.1 it is evident that MAD-GAN is able to capture all the clustered modes which includes significantly overlapped modes as well. MAD-GAN obtains the minimum value in terms of both Chi-square distance and the KL-divergence. In this experiment, both MAD-GAN and MA-GAN used four generators. In the case of InfoGAN, we used 5 dimensional categorical variable, which provides the best result.

To understand the effect of varying the number of generators in MAD-GAN, we use the same synthetic experimental setup, the real data distribution is the same GMM with 5 Gaussians. For better non-parametric estimation of the distribution, we use 1 million sample points from the real distribution (instead of $65,536$). We generate an equal number of points from each of the generators such that they sum up to 1 million. The results are shown in Fig. 3.5 and corresponding Tab. 3.2. It is quite clear that as the number of generators are increased up to 4, the sampling keeps getting more realistic. In the case when multiple modes are significantly overlapped/clustered, a generator can capture a cluster of modes. Therefore, for this real data distribution, 4 generators are enough to capture all the 5 modes. With 5 or more generators, all the modes were still captured, but the two overlapping modes have more than two generation peaks. This is mainly because multiple generators capture this region and all the generators (mixture components) were assigned *equal weights* during sampling.

Other work using more than one generators [94, 7] also use the number of generators as a hyper-parameter as knowing a-priori the number of modes in a real-world data (For exampleimages) in itself is an open problem.

GAN Variants	KL Div	# Modes Covered
DCGAN [120]	2.15	712
WGAN [5]	1.02	868
BEGAN [9]	1.89	819
GoGAN [70]	2.89	672
Unrolled GAN [104]	1.29	842
Mode-Reg DCGAN [14]	1.79	827
InfoGAN [18]	2.75	840
MA-GAN	3.4	700
MAD-GAN (Our)	**0.91**	**890**

Table 3.3: Stacked-MNIST experiments and comparisons. Note that three generators are used for MAD-GAN.

GAN Variants	KL Div	# Modes Covered
DCGAN [120]	0.18	980
WGAN [5]	0.25	1000
BEGAN [9]	0.19	999
GoGAN [70]	0.87	972
Unrolled GAN [104]	0.091	1000
Mode-Reg DCGAN [14]	0.12	992
InfoGAN [18]	0.47	990
MA-GAN	1.62	997
MAD-GAN (Our)	**0.074**	**1000**

Table 3.4: Compositional-MNIST experiments and comparisons. Note that three generators are used for MAD-GAN.



Figure 3.7: Diverse generations for *edges-to-handbags* generation task. In each sub-figure, the first column is the input, columns 2-4 are generations by MAD-GAN (using three generators), and columns 5-7 are generations by InfoGAN (using three categorical codes). Clearly different generators of MAD-GAN are producing diverse results capturing different colors, textures, design patterns, *etc.*. However, InfoGAN generations are visually almost the same, indicating mode collapse.



Figure 3.9: InfoGAN for *edges-to-handbags* task by sharing the discriminator and Q Network. In each sub-figure, the first column is the input, columns $2-4$ are generations when input is categorical code besides conditioning image, and columns $5-7$ are generations with noise as an additional input. The generations for both the architectures are visually the same irrespective of the categorical code value, which clearly indicates that it is not able to capture diverse modes.

3.5.2 Stacked and Compositional MNIST

We now perform experiments on a more challenging setup, similar to [14, 104], in order to examine and compare MAD-GAN with other GAN variants. [104] created a Stacked-MNIST dataset with $25,600$ samples where each sample has three channels stacked together with a random MNIST digit in each of them. Thus, it creates 1000 distinct modes in the data distribution. [104] used a stripped down version of the generator and discriminator pair to reduce the modeling capacity. We do the same for fair comparisons and used the same architecture as mentioned in their paper. Similarly, [14] created Compositional-MNIST whereby they took 3 random MNIST digits and place them at the 3 quadrants of a 64×64 dimensional image. This also resulted in a data distribution with 1000 hand-designed modes. The distribution of the resulting generated samples was estimated using a pretrained MNIST classifier to classify each of the digits either in the channels or the quadrants to decide the mode it belongs to.

Tables 3.3 and 3.4 provide comparison of our method with variants of GAN in terms of KL divergence and the number of modes recovered for the Stacked and Compositional MNIST datasets, respectively. In Stacked-MNIST, as evident from the Tab. 3.3, MAD-GAN outperforms all other variants of GAN in both the criteria. Interestingly, in the case of Compositional-MNIST, as shown in Tab. 3.4, MAD-GAN, WGAN and Unrolled GAN were able to recover all the 1000 modes. However, in terms of KL divergence, the distribution generated by MAD-GAN is the closest to the true data distribution.



Figure 3.11: Diverse generations for *night-to-day* image generation task. First column in each sub-figure represents the input. The remaining three columns show the diverse generations of three different generators of MAD-GAN (Our).



Figure 3.13: Face generations using MAD-GAN. Each sub-figure shows generations by a single generator. The first generator is generating faces with very dark background. The second one is generating female faces with long hair in light background, while the third one is generating faces with colored background and casual look (based on facial direction and expression).

3.5.3 Diverse Samples for Image-to-Image Translation and Comparison to InfoGAN

Here we present experimental results on the challenging task of image-to-image translation [66] which uses conditional variant of GANs [105]. Conditional GAN for this task is known to learn the delta distribution, thus, generates the same image irrespective of the variations in the input noise vector. Generating diverse samples in this setting in itself is an open problem. We show that MAD-GAN is able to generate diverse samples in these experiments as well. We use three generators for MAD-GAN experiments and show three diverse generations. Note that, we do not claim to capture all the possible modes present in the data distribution because firstly we cannot estimate the number of modes a priori, and secondly, even if we could, we do not know how diverse the generations would be after using certain number of generators. We follow the same approach as [66] and employ patch based conditional GAN.

We compare MAD-GAN with InfoGAN [18] in these experiments as it is closest to our approach and can be used in image-to-image translation task. Theoretically, latent codes in InfoGAN should enable diverse generations. However, InfoGAN can only be used when the bias introduced by the categorical variables have significant impact on the generator network. For image-to-image translation and high resolution generations, the categorical variable does not have sufficient impact on the generations. As will be seen shortly, we validate this hypothesis by comparing our method with InfoGAN for this task. For the InfoGAN generator, to capture three kinds of distinct modes, the categorical code is chosen to take three values. Since we are dealing with images, in this case, the categorical code is a 2D matrix in which we set one third of the entries to 1 and remaining to 0 for each category. The generator is fed input image along with categorical code appended channel wise to the image. Architecture of the Q network is same as that of the pix2pix discriminator [66], except that the output is a vector of size 3 for the prediction of the categorical codes. Note that, we tried different variations of the categorical codes but did not observe any significant variation in the generations.

Fig. 3.7 shows generations by MAD-GAN and InfoGAN for the *edges-to-handbags* task, where given the edges of handbags, the objective is to generate real looking handbags. Clearly, each MAD-GAN generator is able to produce meaningful images but different from remaining generators in terms of color, texture, and patterns. However, InfoGAN generations are almost the same for all the three categorical codes. The results shown for InfoGAN are obtained by not sharing the discriminator and Q network parameters.



Figure 3.14: Face generations using MAD-GAN. Each generator employed is DCGAN. Each row represents a generator. Each column represents generations for a given random noise input z. Note that, the first generator is generating faces pointing to the left. The second generator is generating female faces with long hair, while the third generator generates images with light background.

To make our baseline as strong as possible, we did some more experiments with InfoGAN for the *edges-to-handbags* task. For Fig. 3.9, we conducted two experiments by sharing all the initial layers of the discriminator and Q network. In the first experiment, the input is the categorical code besides the conditional image. In the second experiment, noise is also added as an input. The details of the architecture are given in Appendix **??**. In Fig. 3.9, we show the results of both these experiments side by side. There are not many perceivable changes as we vary the categorical code values, as the generator learns to ignore the input noise (as was also seen in [66]).

In addition, in Fig. 3.11, we show diverse generations for the *night-to-day* task, where given night images, the objective is to generate their corresponding day images. As can be seen, the generated day images in Fig. 3.11 differ in terms of lighting conditions, sky patterns, weather conditions, and many other minute yet useful cues.

3.5.4 Diverse-Class Data Generation

To further explore the mode capturing capacity of MAD-GAN, we experimented with a much more challenging task of diverse-class data generation. In detail, we trained MAD-GAN (three generators) on a combined dataset consisting of various highly diverse images such as *islets*, *icebergs*, *broadleaf-forest*, *bamboo-forest*, and *bedroom*, obtained from the Places dataset [161]. Images were randomly selected from each of them, creating a training dataset of $24,000$ images. The generators have the same architecture as that of DCGAN. In this case, as the images in the dataset belong to different classes, we did not share the generator parameters. As shown in Fig. 3.1, to our surprise, we found that even in this

highly challenging setting, the generations from different generators belong to different classes. This clearly indicates that the generators in MAD-GAN are able to disentangle inter-class variations. In addition, each generator for different noise input is able to generate diverse samples, indicating intra-class diversity.

3.5.5 Diverse Face Generation

Here we show diverse face generations (CelebA dataset) using MAD-GAN where we use DCGAN [120] as each of our three generators. Again, we use the same setting as provided in DCGAN. The high quality face generations are shown in the Fig. 3.14. We show additional generations in Fig. 3.13.

3.5.6 Unsupervised Representation Learning

Similar to DCGAN [120], we train our framework using SVHN dataset [107]. The trained discriminator is used to extract features. Using these features, we train an SVM for the classification task. For the MAD-GAN, with three generators, we obtained misclassification error of 17.5% which is almost 5% better than the results reported by DCGAN (22.48%). This clearly indicates that our framework is able to learn a better feature space in an unsupervised setting.

3.6 Conclusion

We presented a very simple and effective framework, Multi-Agent Diverse GAN (MAD-GAN), for generating diverse and meaningful samples. We showed the efficacy of our approach and compared it with various variants of GAN that it captures diverse modes while producing high quality samples. We presented a theoretical analysis of MAD-GAN with conditions for global optimality. Looking forward, an interesting future direction would be to estimate a priori the number of generators needed for a particular dataset. It is not clear how to do that given that we do not have access to the true data distribution. In addition, we would also like to theoretically understand the limiting cases that depend on the relationship between the number of generators and the complexity of the data distribution. Another interesting direction would be to exploit different generators such that their combinations can be used to capture diverse modes.

3.7 Insights & Discussions

Here, we first give better insights about the Theorem 1 and discuss how and when MAD-GAN leads to diverse generations. In Section 3.9, we introduce another way of getting different generators to generate diverse samples. We introduce intuitive similarity based competing objective (MAD-GAN-Sim) which encourages different generators to generate diverse samples.

3.8 Insights for Diversity in MAD-GAN

One obvious question that could arise is that *is it possible that all the generators learn to capture the same mode?*. The short answer is, theoretically *yes* and in practice *no*. Let us begin with the discussion to understand this. Theoretically, according to Theorem 1 if $p_{g_i} = p_d$, for all i, then also the minimum objective value can be achieved. This implies, in worst case, MAD-GAN would perform same as the standard GAN. However, as discussed below, this is possible in following *highly unlikely* situations:

- all the generators *always generate exactly similar* samples so that the discriminator is not able to differentiate them. In this case, the discriminator will learn a uniform distribution over the generator indices, thus, the gradients passed through the discriminator will be exactly the same for all the generators. However, this situation in general is not possible as all the generators are initialized differently. Even a slight variation in the samples from the generators will be enough for the discriminator to identify them and pass different gradient information to each generator. In addition, the objective function of generators is *only* to generate *real* samples, thus, there is nothing that encourages them to generate exactly the same samples.

- the discriminator does not have enough capacity to learn the optimal parameters. This is in contrast to the assumption made in Theorem 1, which is that the discriminator is *optimal*. Thus, it should have enough capacity to learn a feature representation such that it can correctly identify samples from different generators. In practice, this is a very easy task and we did not have to modify anything up to the feature representation stage of the architecture of the discriminator. We used the standard architectures for all the tasks.

Hence, with random initializations and sufficient capacity generator/discriminator, we can easily avoid the trivial solution in which all the generators focus on exactly the same region

of the true data distribution. This has been very clearly supported by various experiments showing diverse generations by MAD-GAN.

3.9 Similarity based competing objective

We have discussed the MAD-GAN architecture using generator identification based objective. In this section, we propose a different extension to the standard GAN : *similarity based competing objective*, which we call as MAD-GAN-Sim. Here, we augment the GAN objective function with a diversity enforcing term. It ensures that the generations from different generators are diverse where the diversity depends on a user-defined task-specific function.

The architecture is same as that of MAD-GAN discussed in Section 3.4.1 (refer Fig. 3.15).

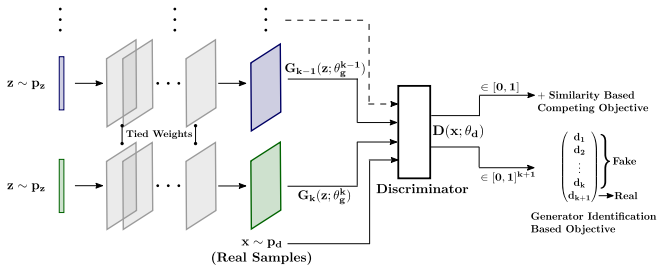


Figure 3.15: MAD-GAN-Sim compared with MAD-GAN. All the generators share parameters of all the layers except the last one. The two proposed diversity enforcing objectives, 'competing' (MAD-GAN-Sim) and 'generator identification' (MAD-GAN), are shown at the end of the discriminator

3.9.1 Approach

The approach presented here is motivated by the fact that the samples from different modes must look different. For example, in the case of images, these samples should differ in terms of texture, color, shading, and various other cues. Thus, different generators must generate dissimilar samples where the dissimilarity comes from a task-specific function. Before delving into the details, let us first define some notations in order to avoid clutter. We denote θ_g^i as the parameters of the i-th generator. The set of generators is denoted as $K = \{1, \cdots, k\}$. Given a random noise z to the i-th generator, the corresponding generated sample $G_i(z; \theta_g^i)$ is denoted as $g_i(z)$. Using these notations and following the above discussed intuitions, we

impose following constraints over the i-th generator while updating its parameters:

$$D(G_i(z;\theta_g^i);\theta_d) \geq D(G_j(z;\theta_g^j);\theta_d) + \Delta\Big(\phi(g_i(z)),\phi(g_j(z))\Big), \quad \forall j \in K \setminus i \tag{3.5}$$

where, $\phi(g_i(z))$ denotes the mapping of the generated image $g_i(z)$ by the i-th generator into a feature space and $\Delta(.,.) \in [0,1]$ is the similarity function. Higher the value of $\Delta(.,.)$ more similar the arguments are. Intuitively, the above set of constraints ensures that the discriminator score for each generator should be higher than all the other generators with a margin proportional to the similarity score. If the samples are similar, the margin increases and the constraints become more active. We use unsupervised learning based representation as our mapping function $\phi(.)$. Precisely, given a generated sample $g_i(z)$, $\phi(g_i(z))$ is the feature vector obtained using the discriminator of our framework. This is motivated by the feature matching based approach to improve the stability of the training of GANs [127]. The $\Delta(.,.)$ function used in this work is the standard cosine similarity based function. The above mentioned constraints can be satisfied by maximizing an equivalent unconstrained objective function as defined below:

$$U(\theta_g^i,\theta_d) := f\Bigg(D(G_i(z;\theta_g^i);\theta_d) - \frac{1}{k-1}\sum_{j\in K\setminus i}\Big(D(G_j(z;\theta_g^j);\theta_d) + \Delta(\psi_i,\psi_j)\Big)$$

where, $f(a) = \min(0,a)$, $\psi_i = \phi(g_i(z))$, and $\psi_j = \phi(g_j(z))$. Intuitively, if the argument of $f(.)$ is positive, then the desirable constraint is satisfied and there is no need to do anything. Otherwise, maximize the argument with respect to θ_g^i. Note that instead of using all the constraints independently, we use the average of all of them. Another approach would be to use the constraint corresponding to the j-th generator that maximally violates the set of constraints shown in Eq. 3.5. Experimentally we found that the training process of the average constraint based objective is more stable than the maximum violated constraint based objective. The intuition behind using these constraints comes from the well know 1-slack formulation of the structured SVM framework [69, 140]. Thus, the overall objective for the i-th generator is:

$$\min_{\theta_g^i} \; V(\theta_d,\theta_g^i) - \lambda\, U(\theta_g^i,\theta_d)$$

where $\lambda \geq 0$ is the hyperparameter. Algorithm 3.9.1 shows how to compute gradients corresponding to different generators for the above mentioned objective function. Notice that,

once sampled, the same z is passed through all the generators in order to enforce constraints over a *particular generator* (as shown in Eq. 3.5). However, in order for constraints to not to contradict with each other while updating another generator, a different z is sampled again from the p_z. The Algorithm 3.9.1 is shown for the batch of size one which can be trivially generalized for any given batch sizes. In the case of the discriminator, the gradients will have exactly the same form as the standard GAN objective. The only difference is that in this case the fake samples are being generated by k generators, instead of one.

[tb] [1] θ_d; $p(z)$; $\theta_g^i, \forall i \in \{1, \cdots, k\}$; λ. each generator $i \in \{1, \cdots, k\}$ Sample noise from the given noise prior $z \sim p_z$. Obtain the generated sample $G_i(z; \theta_g^i)$ and corresponding feature vector $\psi_i = \phi(G_i(z; \theta_g^i))$. $\nu \leftarrow 0$. each generator $j \in \{1, \cdots, k\} \setminus i$ Compute feature vector $\psi_j = \phi(G_j(z; \theta_g^j))$. $\nu \leftarrow \nu + D(G_j(z; \theta_g^j); \theta_d) + \Delta(\psi_i, \psi_j)$. $\nu \leftarrow D(G_i(z; \theta_g^i); \theta_d) - \frac{\nu}{k-1}$. $\nu \geq 0$ $\nabla_{\theta_g^i} \log(1 - D(G_i(z; \theta_g^i); \theta_d)))$. $\nabla_{\theta_g^i} \Big(\log(1 - D(G_i(z; \theta_g^i); \theta_d))) - \lambda U(\theta_g^i, \theta_d) \Big)$.

3.9.2 Experiments

We present the efficacy of MAD-GAN-Sim on the real world datasets.

3.9.3 Diverse Samples for Image-to-Image Translation

We show diverse and highly appealing results using the diversity promoting objective. We use cosine based similarity to enforce diverse generations, an important criteria for real images. As before, we show results for the following two situations where diverse solution is useful: (1) given the edges of handbags, generate real looking handbags as in Fig. 3.17; and (2) given night images of places, generate their equivalent day images as in Fig. 3.19. We clearly notice that each generator is able to produce meaningful and diverse images.



Figure 3.17: MAD-GAN-Sim: Diverse generations for 'edges-to-handbags' image generation task. First column in each sub-figure represents the input. The remaining three columns show the diverse outputs of different generators. It is evident that different generators are able to produce very diverse results capturing color (brown, pink, black), texture, design pattern, shininess, among others.

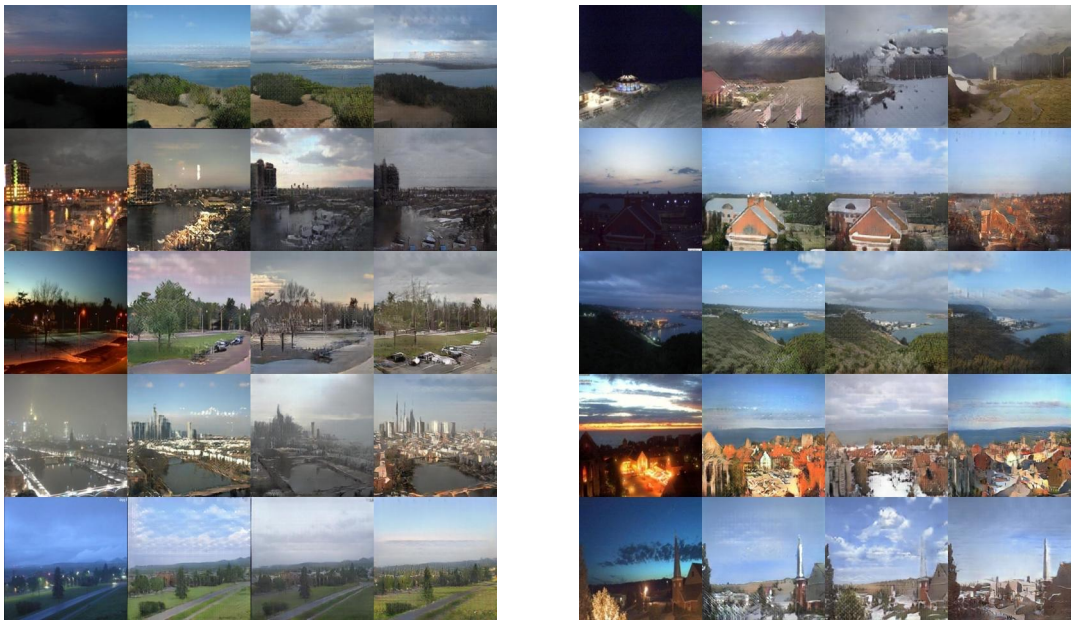


Figure 3.19: MAD-GAN-Sim: Diverse generations for 'night to day' image generation task. First column in each sub-figure represents the input. The remaining three columns show the diverse outputs of different generators. It is evident that different generators are able to produce very diverse results capturing different lighting conditions, different sky patterns (cloudy vs clear sky), different weather conditions (winter, summer, rain), different landscapes, among many other minute yet useful cues.

4

Interactive Sketch & Fill: Multiclass Sketch-to-Image Translation

Abstract

We propose an interactive GAN-based sketch-to-image translation method that helps novice users create images of simple objects. As the user starts to draw a sketch of a desired object type, the network interactively recommends plausible completions, and shows a corresponding synthesized image to the user. This enables a feedback loop, where the user can edit their sketch based on the network's recommendations, visualizing both the completed shape and final rendered image while they draw. In order to use a single trained model across a wide array of object classes, we introduce a gating-based approach for class conditioning, which allows us to generate distinct classes without feature mixing, from a single generator network.

4.1 Introduction

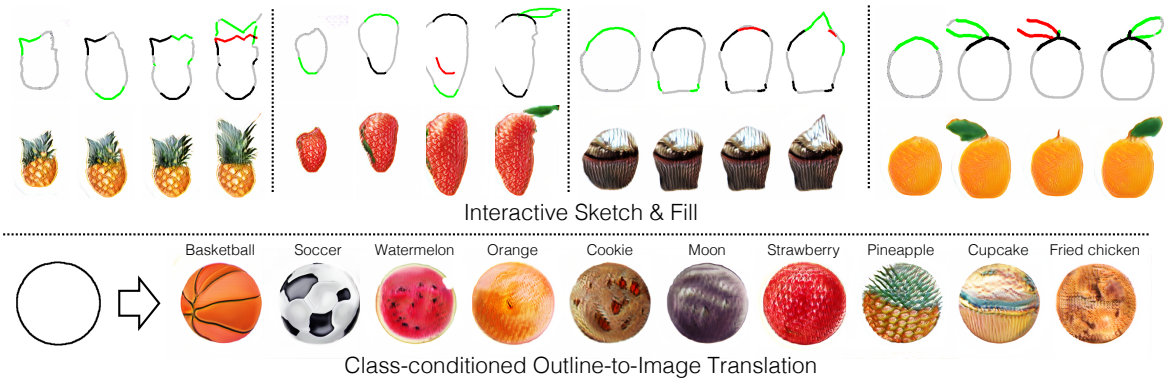


Figure 4.1: (**Top**) Given a user created incomplete object outline (first row), our model estimates the complete shape and provides this as a recommendation to the user (shown in gray), along with the final synthesized object (second row). These estimates are updated as the user adds (green) or removes (red) strokes over time – previous edits are shown in black. (**Bottom**) This generation is class-conditioned, and our method is able to generate distinct multiple objects for the same outline (For example'circle') by conditioning the generator on the object category.

Conditional GAN-based image translation [66, 128, 164] models have shown remarkable success at taking an abstract input, such as an edge map or a semantic segmentation map, and translating it to a real image. Combining this with a user interface allows a user to quickly create images in the target domain. However, such interfaces for object creation require the entire edge or label map as input, which is a challenging task as users typically create drawings *incrementally*. Furthermore, completing a line drawing without any feedback may prove difficult for many, as untrained practitioners generally struggle at free-hand drawing of accurate proportions of objects and their parts [20], 3D shapes and perspective [130]. As a result, it is much easier with current interactive image translation methods to obtain realistic looking images by editing *existing* images [28, 118] rather than creating images from scratch.

We propose a new GAN-based interactive image generation system for drawing objects from scratch that: 1) generates full images given *partial* user strokes (or sketches); 2) serves as a *recommender system* that suggests or helps the user *during* their creative process to help them generate a desired image; and 3) uses a single conditional GAN model for *multiple* image classes, via a gating-based conditioning mechanism. Such a system allows for creative input to come from the user, while the challenging task of getting exact object proportions correct is left to the model, which constantly predicts a plausible completion of the user's sketch (Fig. 4.1).



Figure 4.2: Video of our interface We can see two versions of our interface. The left side shows how a user can quickly generate multiple objects using a few strokes, while the right side shows the utility of multimodal completions where the user can quickly explore different possible shape generations while drawing. Full video available at our website. **Please view with Acrobat Reader.**

Unlike other related work, we use sparse object outlines/sketches/simplified-edges instead of dense edge maps as the user input as these are closer to the lines that novice users tend to draw [21]. Our model first completes the user input and then generates an image conditioned on the completed shape. There are several advantages to this two-stage approach. For one, we are able to give the artist feedback on the general object shape in our interactive interface (similar to ShadowDraw [90]), allowing them to quickly refine higher level shape until it is satisfactory. Second, we found that splitting completion and image generation to work better than going directly from partial outlines to images, as the additional intermediate supervision on full outlines/sketches breaks the problem into two easier sub-problems – first recover the geometric properties of the object (shape, proportions) and then fill in the appearance (colors, textures).

For the second stage, we use a multi-class generator that is conditioned on a user supplied class label. This generator applies a gating mechanism that allows the network to focus on the important parts (activations) of the network specific to a given class. Such an approach allows for a clean separation of classes, enabling us to train a single generator and discriminator across *multiple* object classes, thereby enabling a finite-size deployable model that can be used in multiple different scenarios.

To demonstrate the potential of our method as an interactive tool for stroke-based image generation, we collect a new image dataset of ten simple object classes (pineapple, soccer, basketball, etc.) with white backgrounds. In order to stress test our conditional generation mechanism, six of the object classes have similar round shapes, which requires the network to derive texture information from the class conditioning. Fig. 4.2 shows a short video of an interactive editing session using our system. Along with these simple objects, we also demonstrate the potential of our method on more complicated shapes, such as faces and shoes. Code and other details are available at our website.

4.2 Related Work

Interactive Generation Interactive interfaces for freehand drawing go all the way back to Ivan Sutherland's Sketchpad [138]. The pre-deep work most related to us, ShadowDraw [90], introduced the concept of generating multiple shadows for novice users to be able to draw sketches. PhotoSketcher [39] introduces a retrieval based method for obtaining real images from sketches. More recently, deep recurrent networks have been used to generate sketches [54, 44]. Sketch-RNN [54] provides a completion of partial strokes, with the advantage of intermediate stroke information via the Quickdraw dataset at training time. SPIRAL [44] learns to generate digits and faces using a reinforcement learning approach. Zhu et al. [162] train a generative model, and an optimization-based interface to generate possible images, given color or edge constraints. The technique is limited to a single class and does not propose a recommendation for the completion of the shape. SketchyGAN [17] also aimed at generating multi-class images but lacks interactive capability. In contrast to the above, our method provides interactive prediction of the shape and appearance to the user and supports multiple object classes.

Generative Modeling Parametric modeling of an image distribution is a challenging problem. Classic approaches include autoencoders [59, 145] and Boltzmann machines [135]. More modern approaches include autoregressive models [38, 111], variational autoencoders (VAEs) [76], and generative adversarial networks (GANs). GANs and VAEs both learn mappings from a low-dimensional "latent" code, sampled stochastically, to a high-dimensional image through a feedforward pass of a network. GANs have been successful recently [30, 120, 6], and hybrid models feature both a learned mapping from image to latent space as well as adversarial training [32, 35, 86, 18].

Conditioned Image Generation The methods described above can be conditioned, either by a low-dimensional vector (such as an object class, or noise vector), a high-dimensional image, or both. Isola et al. [66] propose "pix2pix", establishing the general usefulness of conditional GANs for image-to-image translation tasks. However, they discover that obtaining multimodality by injecting a random noise vector is difficult, a result corroborated in [102, 115, 163]. This is an example of mode collapse [50], a phenomenon especially prevalent in image-to-image GANs, as the generator tends or ignore the low-dimensional latent code in favor of the high-dimensional image. Proposed solutions include layers which

better condition the optimization, such as Spectral Normalization [159, 106], modifications to the loss function, such as WGAN [5, 53] or optimization procedure [58], or modeling proposals, such as MAD-GAN [48] and MUNIT [64]. One modeling approach is to add a predictor from the output to the conditioner, to discourage the model from ignoring the conditioner. This has been explored in the classification setting in Auxiliary-Classifier GAN (ACGAN) [109] and regression setting with InfoGAN [18] and ALI/BiGAN ("latent regressor" model) [35, 32], and is one half of BicycleGAN model [163]. We explore a complementary approach of architectural modification via gating.

Gating Mechanisms Residual networks [57], first introduced for image classification [82], have made extremely deep networks viable to train. Veit et al. [144] find that the skip connection in the architecture enables test-time removal of blocks. Follow-up work [143] builds in block removal during training time, with the goal of subsets of blocks specializing to different categories. Inspired by these results, we propose the use of gating for image generation and provide a systematic analysis of gating mechanisms.

The adaptive instance normalization (AdaIn) layer has similarly been used in arbitrary style transfer [63] and image-to-image translation [64], and Feature-wise Linear Modulation (FiLM) [117]. Both methods scale and shift feature distributions, based on a high-dimensional conditioner, such as an image or natural language question. Gating also plays an important role in sequential models for natural language processing: LSTMs [60] and GRU [19]. Similarly, concurrent work [72], [114] use a AdaIN-style network to modulate the generator parameters.

4.3 Method

We decouple the problem of interactive image generation into two stages: object shape completion from sparse user sketches, and appearance synthesis from the completed shape. More specifically, as illustrated in Fig. 4.3 we use the Shape Generator G_S for the automatic shape (outline/sparse-sketch/simplified-edge) generation and the Appearance Generator G_A for generating the final image as well as the adversary discriminators D_S and D_A. Example usage is shown in our user interface in Fig. 4.2.

4.3.1 Shape completion

The shape completion network G_S should provide the user with a visualization of its completed shape(s), based on the user input, and should keep on updating the suggested

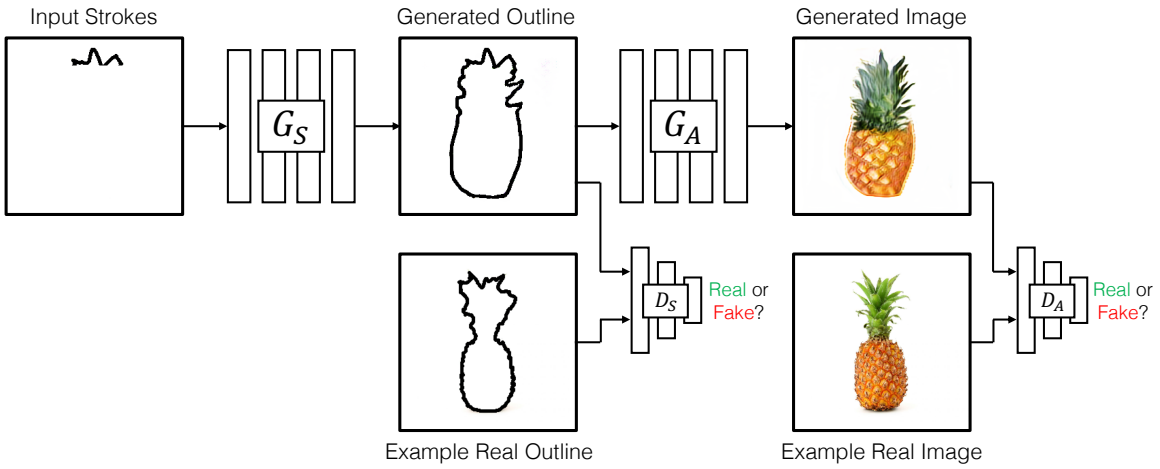


Figure 4.3: Our two-stage approach First, we complete a partial sketch using the shape generator G_S. Then we translate the completed sketch into an image using the appearance generator G_A. Both generators are trained with their respective discriminators D_S, and D_A.

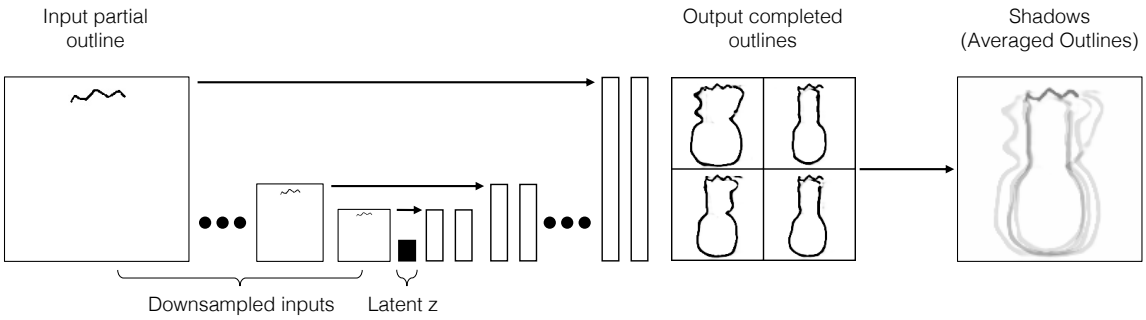


Figure 4.4: First stage (Shape Generator) To achieve multi-modal completions, the shape generator is designed using inspiration from non-image conditional model [103] with the conditioning input provided at multiple scales, so that the generator network doesn't ignore the partial stroke conditioning.

shape(s) interactively. We take a data-driven approach for this whereby, to train the network, we simulate partial strokes (or inputs) by removing random square patches from the full outline/ full sparse sketch/ full simplified edges. The patches are of three sizes (64×64, 128×128, 192×192) and placed at a random location in the image of size 256×256 (see Fig. 4.5 for an example). To extend the technique beyond outlines and generate more human-like sketches, we adopt the multistage procedure depicted in Fig. 4.6. We refer to these generated sketches as "simplified edges". We automatically generate data in this manner, creating a dataset where for a given full outline/sketch or a simplified edge-map, 75 different inputs are created. The model, shown in Fig. 4.3, is based on the architecture used for non-image conditional generations in [103]. We modify the architecture such that

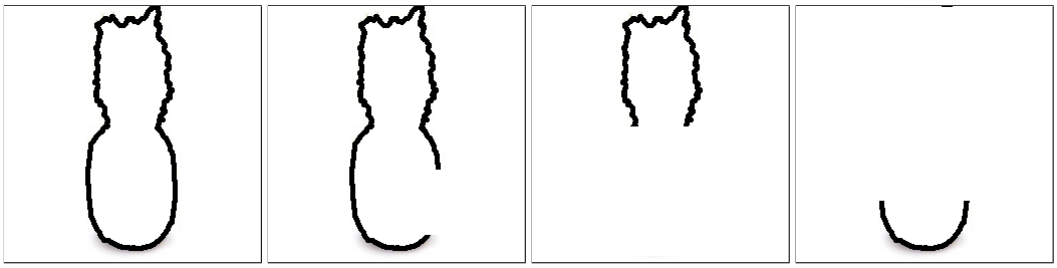


Figure 4.5: Simulated Inputs Three sizes of occluders were used to simulate partial outlines.

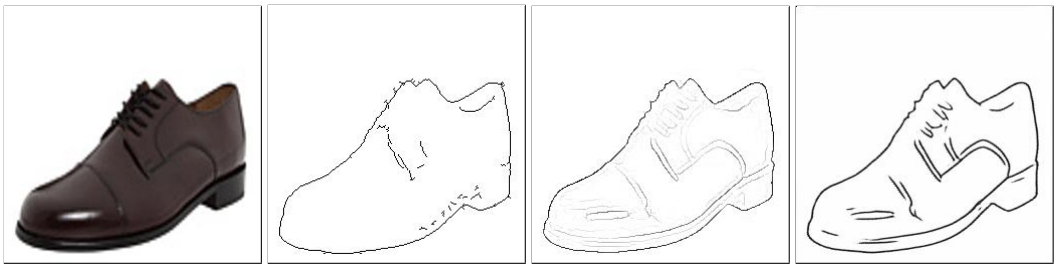


Figure 4.6: Simplified Edges The 2nd edgemap is obtained using the technique of [66], while the 3rd is the intermediate edgemap using [93] and further simplified using [134] which looks closer to what a human would sketch.

the conditioning input is provided to the generator and discriminator at multiple scales as shown in Fig. 4.4. This makes the conditioning input an active part of the generation process and helps in producing multimodal completions.

4.3.2 Appearance synthesis

An ideal interactive sketch-to-image system should be able to generate multiple different image classes with a single generator. Beside memory and time considerations (avoiding loading/using a separate model per class, reducing overall memory), a single network can share features related to outline recognition and texture generation that are common across classes, which helps training with limited examples per class.

As we later show, class-conditioning by concatenation can fail to properly condition the network about the class information in current image translation networks [66, 163]. To address this, we propose an effective soft gating mechanism, shown in Fig. 4.7. Conceptually, our network consists of a small external gating network that is conditioned on the object class (encoded as a 1-hot vector). The gating network outputs parameters that are used to modify the features of the main generator network. Given an input feature tensor X_l,

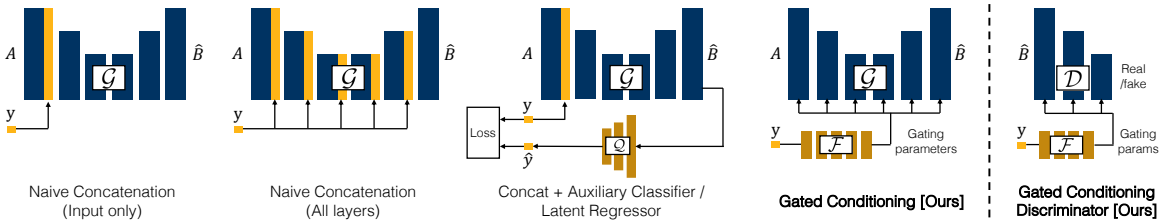


Figure 4.7: Conditioning variants for the Appearance Generator Our model uses gating on all the residual blocks of the generator and the discriminator, other forms of conditioning such as (naive concatenation in input only, all layers, AC-GAN like latent regressor [109]) are evaluated as well.

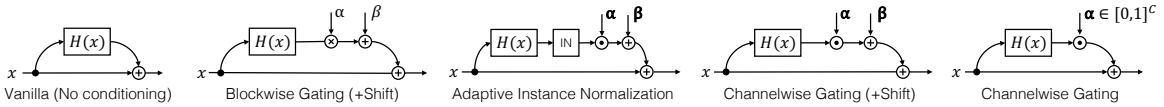


Figure 4.8: Injecting conditioning with modified residual layers (Left) A "vanilla" residual block without conditioning applies a residual modification to the input tensor. **(Mid-left)** The $\mathcal{H}(X)$ block is softly-gated by scalar parameter α and shift β. **(Mid)** Adaptive Instance Normalization [63] applies a channel-wise scaling and shifting after an instance normalization layer. **(Mid-right)** Channel-wise gating adds restrictions to the range of $\boldsymbol{\alpha}$. **(Right)** We find that channel-wise gating (without added bias) produces the best results empirically.

"vanilla" ResNet [57] maps it to

$$X_{l+1} = X_l + \mathcal{H}_l(X_l). \tag{4.1}$$

Changes in resolution are obtained by upsampling before or downsampling after the residual block. Note that we omit l subscript from this point forward to reduce clutter. Our gating network augments this with a predicted scalar α for each layer of the network using a learned network $\mathcal{F}(\mathbf{y})$, where $\mathbf{y}$ is the conditioning vector:

$$X + \alpha\, \mathcal{H}(X), \text{where } \alpha \in [0, 1] \tag{4.2}$$

If the conditioning vector $\mathbf{y}$ has no use for a particular block, it can predict α close to zero and effectively switch off the layer. During training, blocks within the main network can transform the image in various ways, and $\mathcal{F}$ can modulate such that the most useful blocks are selected. Unlike previous feature map conditioning methods such as AdaIn [141], we apply gating to *both* the generator and discriminator. This enables the discriminator to select blocks which effectively judge whether generations are real or fake, conditioned on the class input. Some blocks can be shared across regions in the conditioning vector, whereas other blocks can specialize for a given class.

A more powerful method is to apply this weighting channel-wise using a vector $\boldsymbol{\alpha}$:

$$X + \boldsymbol{\alpha} \odot \mathcal{H}(X), \text{where } \boldsymbol{\alpha} \in [0, 1]^c, \tag{4.3}$$

where $\odot$ represents channel-wise multiplication. This allows specific channels to be switched "on" or "off", providing additional degrees of freedom. We found that this channelwise approach for gating provides the strongest results. AdaIn describes the case where an Instance Normalization [141] (IN) operation is applied before scaling and shifting the feature distribution. We constrain each element of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in $[-1, 1]$. We additionally explored incorporating a bias term after the soft-gating, either block-wise using a scalar $\beta \in [-1, 1]$ per layer, or channel-wise using a vector $\boldsymbol{\beta} \in [-1, 1]^c$ per layer but we found that they did not help much, and so we leave them out of our final model. Refer Fig. 4.8 for pictorial representation of various gatings.

Finally, we describe our network architecture, which utilizes the gated residual blocks described above. We base our architecture on the proposed residual **Encoder-Decoder** model from MUNIT [64]. This architecture is comprised of 3 `conv` layers, 8 residual blocks, and 3 `up-conv` layers. The residual blocks have 256 channels. First, we deepen the network, based on the principle that deeper networks have more valid disjoint, partially shared paths [144], and add 24 residual blocks. To enable the larger number of residual blocks, we drastically reduce the width to 32 channels for every layer. We refer to this network as **SkinnyResNet**. Additionally, we found that modifying the downsampling and upsampling blocks to be residual connections as well improved results, and also enables us to apply gating to *all* blocks. When gating is used, the gate prediction network, $\mathcal{F}(\mathbf{y})$, is also designed using residual blocks. Additional architecture details are in the supplementary material.

4.4 Experiments

We first compare our 2 step approach for interactive image generation on existing datasets such as the UTZappos Shoes dataset [157] and CelebA-HQ [74]. State-of-the-art techniques such as pix2pixHD [148] are used to generate the final image from the autocompleted sketches. We finally evaluate our approach on a multi-class dataset that we collected to test our proposed gating mechanism.

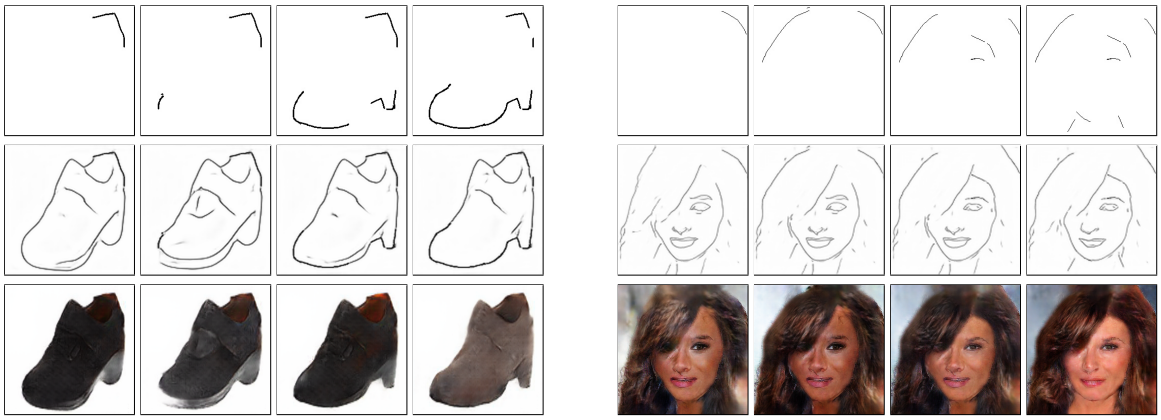


Figure 4.9: Example Sketch & Fill Progression. The **first row** represents the progressive addition of new strokes on the canvas, the **second row** shows the autocompleted sketch, and the **third row** is the final generated image. As the sparse strokes are changed by the user, the completed shape and generated image evolve as well. Note that changing a stroke locally produces coherent changes in other parts of the image.

Trained task	FID
Faces	
Partial Simplified Edges → Image	383.02
Partial Simplified Edges → Simplified Edges → Image	374.67
Shoes	
Partial Simplified Edges → Image	170.45
Partial Simplified Edges → Simplified Edges → Image	154.32

Table 4.1: Single-class generation, 2-stage vs 1-stage. We evaluate the result quality from different task pipelines.

4.4.1 Single Class Generation

Datasets We use the edges2shoes[66], CelebA-HQ[74] datasets to test our method on single class generation. We simplify the edges to attempt to more closely resemble how humans would draw strokes by first using the preprocessing code of [93] further reducing the strokes with a sketch simplification network [134].

Architecture We use the architecture described in Section 4.3.1 for shape completion. In this case, each dataset only contains a single class, so we can use an off-the-shelf network,

Trained task	Avg Acc
Partial edges → Image	73.12 %
Partial outline → Image	88.74 %
Partial outline → Full outline → Image [Ours]	**97.38%**

Table 4.2: Multi-class generation, 2-stage vs 1-stage. We evaluate the result quality from different task pipelines. Accuracy is computed by a fixed, pretrained classification network, on the resulting images.

such as pix2pixHD [149] for rendering.

Results As seen in Fig. 4.9, our 2 step technique allows us to complete the simplified edge maps from the partial strokes and also generate realistic images from the autocompleted simplified edges. Table 4.1 also demonstrates, across two datasets (faces and shoes), that using a 2 step procedure produces stronger results than mapping directly from the partial sketch to the completed image.

Method	SkinnyResNet		EncDec	
	Class. Acc [%]	AMT Fool. Rate [%]	Class. Acc [%]	AMT Fool. Rate [%]
Ground truth	100.0	50.0	100.0	50.0
1 gen/class	***97.0***	17.7±1.46	–	–
Concat (In)	62.6	15.0±1.4	39.2	7.5±1.06
Concat (All)	64.5	15.3±1.41	51.4	5.4±0.88
Cat(In)+Aux-Class	65.6	14.5±1.5	–	–
Cat(All)+Aux-Class	67.0	19.7±1.42	–	–
BlockGate(+bias)	89.6	19.6±1.34	–	–
BlockGate	**99.6**	17.3±1.61	–	–
AdaIn	94.5	14.9±1.47	–	–
ChanGate(+bias)	94.1	14.8±1.43	–	–
ChanGate	***97.0***	**23.4±1.99**	92.7	14.1±1.48

4.4.2 Multi-Class Generation

Datasets To explore the efficacy of our full pipeline, we introduce a new outline dataset consisting of 200 images (150 train, 50 test) for each of 10 classes – basketball, chicken, cookie, cupcake, moon, orange, soccer, strawberry, watermelon and pineapple. All the

images have a white background and were collected using search keywords on popular search engines. In each image, we obtain rough outlines for the image. We find the largest blob in the image after thresholding it into a black and white image. We fill the interior holes of the largest blob and obtain a smooth outline using the Savitzky–Golay filter [129].

Architecture For the shape completion, we use the architecture in Section 4.3.1. For class-conditioned image generation, test the gated architectures in Section 4.3.2.

Results In order to test the fidelity of the automatically completed shapes, we evaluate the accuracy of a trained classifier on being able to correctly label a particular generation. We first test in Table 4.2 that our 2 stage technique is better than 1 step generation. We evaluate the results on the multi-class outline to image generations on two axes: adherence to conditioning and realism. We first test the conditioning adherence – whether the network generates an image of the correct class. Off-the-shelf networks have been previously used to evaluate colorizations [160], street scenes [66, 149], and ImageNet generations [127]. We take a similar approach and fine-tune a pretrained InceptionV3 network [139] for our 10 classes. The generations are then tested with this network for classification accuracy. Results are presented in Table 4.4.1.

To judge the generation quality, we also perform a “Visual Turing test" using Amazon Mechanical Turk (AMT). Turkers are shown a real image, followed by a generated image, or vice versa, and asked to identify the fake. An algorithm which generates a realistic image will “fool" Turkers into choosing the incorrect image. We use the implementation from [160]. Results are presented in Table 4.4.1, and qualitative examples are shown in Fig. 4.10.

Gating Architectures We compare our proposed model to the residual **Encoder-Decoder** model [64]. In addition, we compare our proposed gating strategy and **SkinnyResNet** architecture to the following methods for conditional image generation:

- **Per-class**: a single generator for each category; this is the only test setting with *multiple* networks, all others train a single network
- **Concat (In)**: naive concatenation, input layer only
- **Concat (All)**: naive concatenation, all layers
- **Concat (In)+Aux-Class**: we add an auxiliary classifier, both for input-only and all layers settings

- **BlockGate(+Bias), BlockGate**: block-wise soft-gating, with and without a bias parameter
- **AdaIn**: Adaptive instance normalization
- **ChannelGate(+Bias), ChannelGate**: channel-wise soft-gating, with and without a bias parameter

Does naive concatenation effectively inject conditioning? In Fig. 4.10, we show a selected example from each of the 10 classes. The per-class baseline trivially adheres to the conditioning, as each class gets to have its own network. However, when a single network is trained to generate all classes, naive concatenation is unable to successfully inject class information, for either network and for either type of concatenation. For the **EncoderDecoder** network, basketballs, oranges, cupcakes, pineapples, and fried chicken are all confused with each other. For the **SkinnyResNet** network, oranges are generated instead of basketballs, and pineapples and fried chicken drumsticks are confused. As seen in Table 4.4.1, classification accuracy is slightly higher when concatenating all layers (64.5%) versus only the input layer (62.6%), but is low for both.

Does gating effectively inject conditioning? Using the proposed soft-gating, on the other hand, leads to successful generations. We test variants of soft-gating on the **SkinnyResNet**, and accuracy is dramatically improved, between 89.6% to 99.6%, comparable to using a single generator per class (97.0%). Among the gating mechanisms, we find that channel-wise multiplication generates the most realistic images, achieving an AMT fooling rate of 23.4%. Interestingly, the fooling rate is higher than the per-class generator of 17.7%. Qualitatively, we notice that per-class generators sometimes exhibits artifacts in the background, as seen in the generation of “moon". We hypothesize with the correct conditioning mechanism, the single generator across multiple classes has the benefit of seeing more training data and finding common elements across classes, such as clean, white backgrounds.

Is gating effective across architectures? As seen in Table 4.4.1, using channelwise gating instead of naive concatenation improves performance both accuracy and realism *across* architectures. For example, for the **EncoderDecoder** architecture, gating enables successful generation of the pineapple. Both quantitatively and qualitatively, results are better for our proposed **SkinnyResNet** architecture.

Do the generations generalize to unusual outlines? The training images consist of the outlines corresponding to the geometry of each class. However, an interesting test scenario is whether the technique generalizes to unseen shape and class combinations. In Fig. 4.1, we show that an input circle not only produces circular objects, such as a basketball, watermelon,

and cookie, but also noncircular objects such as strawberry, pineapple, and cupcake. Note that both the pineapple crown and bottom are generated, even without any structural indication of these parts in the outline.

4.5 Discussion

We present a two-stage approach for interactive object generation, centered around the idea of a shape completion intermediary. This step both makes training more stable and also allows us to give coarse geometric feedback to the user, which they can choose to integrate as they desire.

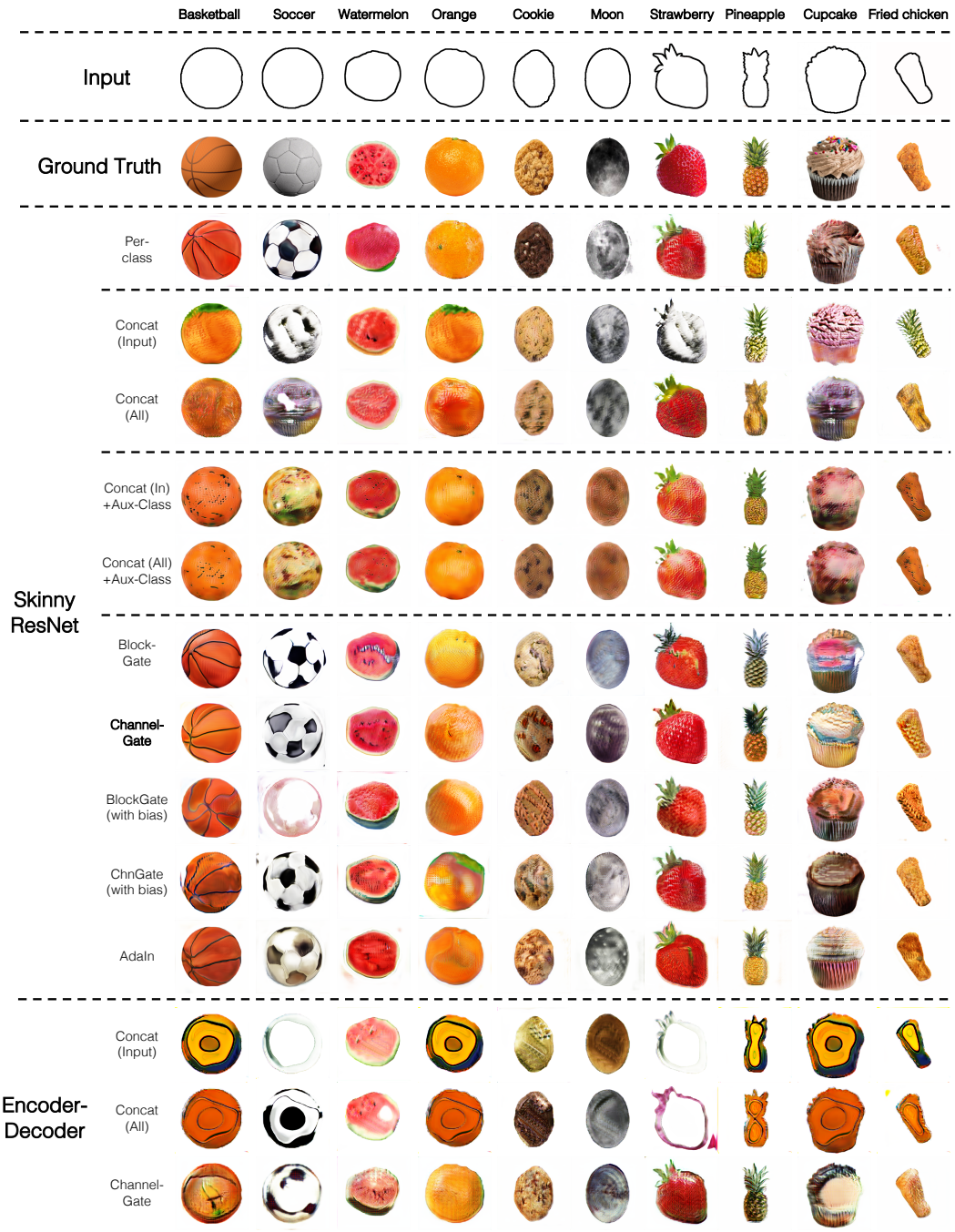


Figure 4.10: Conditioning injection comparison. We show results across methods on the outline→image task using the **SkinnyResNet** architecture. Naive Concatenation **Concat** often confuses classes, such as oranges and basketballs, while gating mechanisms such as the **ChannelGate** method succeed. The gating method also improves results for the **EncoderDecoder** architecture.

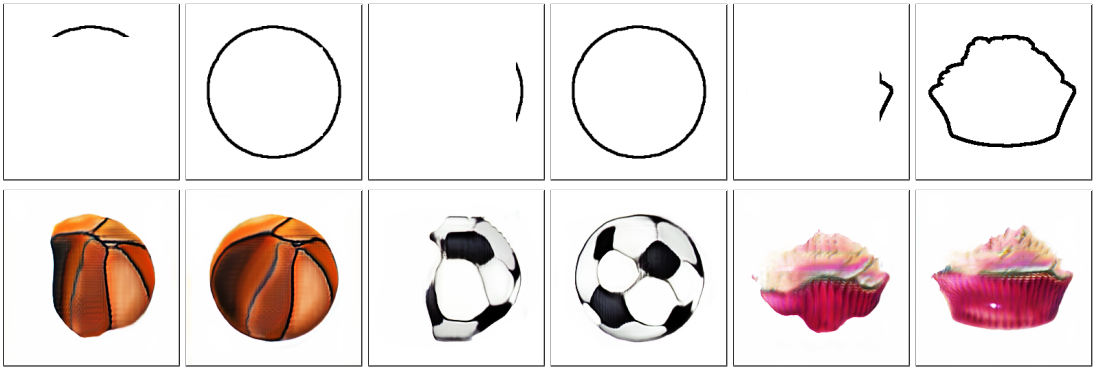


Figure 4.11: Directly mapping from partial outline to image Our proposed system uses a 2-stage approach, using a completed edge map as an intermediate. Here, we show results when directly mapping from the partial outline to the image. When the outline is well-defined, the network can generate realistic images. However, when the outline is sparse, the network struggles with the geometry.

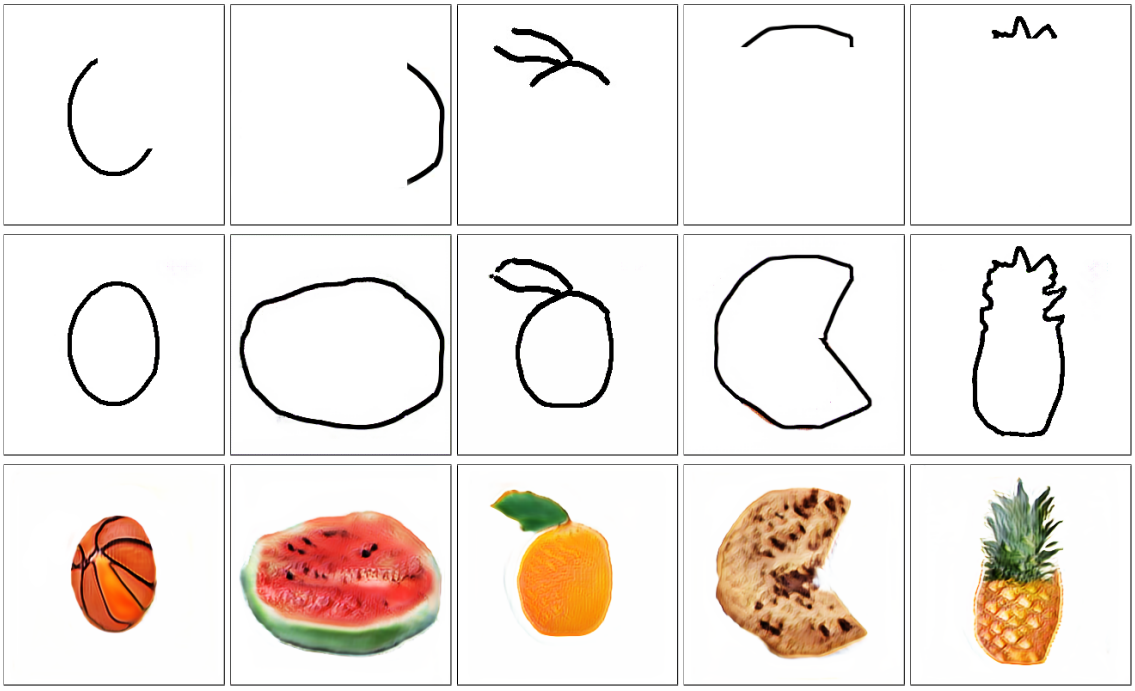


Figure 4.12: Multiclass Sketch & Fill results A few input strokes (first row) are enough to automatically complete the class specific outlines (second) and appearance (last).

5

STEER: Simple Temporal Regularization For Neural ODEs

Abstract

Training Neural Ordinary Differential Equations (ODEs) is often computationally expensive. Indeed, computing the forward pass of such models involves solving an ODE which can become arbitrarily complex during training. Recent works have shown that regularizing the dynamics of the ODE can partially alleviate this. In this work we propose a new regularization technique: randomly sampling the end time of the ODE during training. The proposed regularization is simple to implement, has negligible overhead and is effective across a wide variety of tasks. Further, the technique is orthogonal to several other methods proposed to regularize the dynamics of ODEs and as such can be used in conjunction with them. We show through experiments on normalizing flows, time series models and image recognition that the proposed regularization can significantly decrease training time and even improve performance over baseline models.

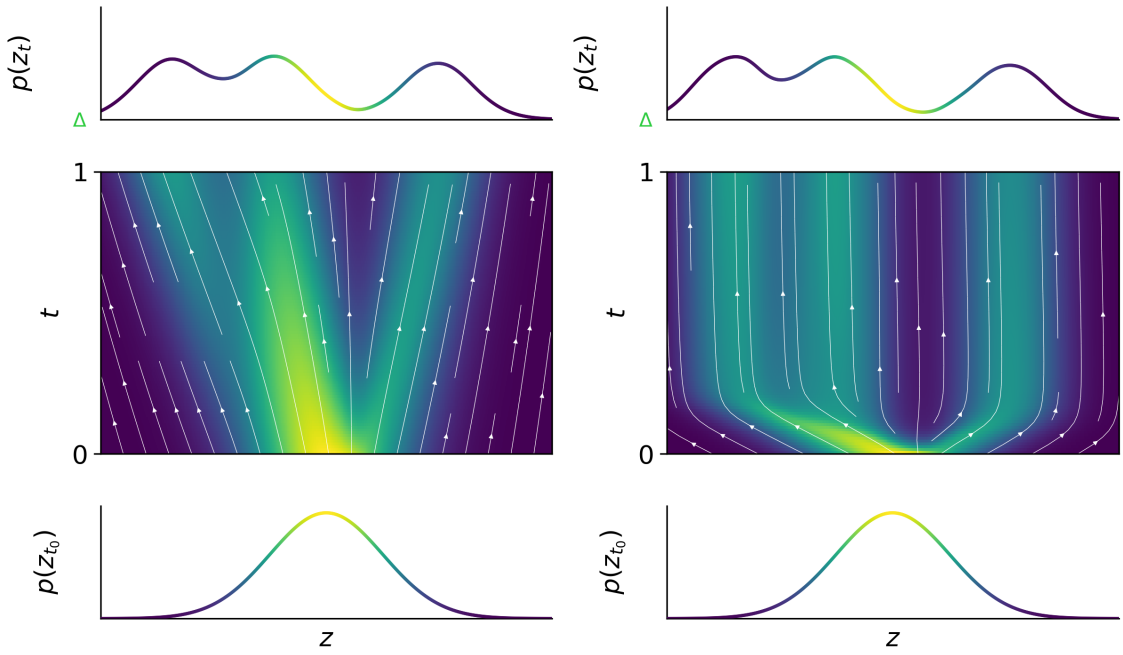


Figure 5.1: Flow fields of a standard Neural ODE (left) vs one equipped with STEER (right). These models transform a simple initial distribution $p(z_{t_0})$ to the target distribution $p(z_t)$. STEER learns simpler flows, which enable faster training.

5.1 Introduction

Neural Ordinary Differential Equations (ODE) are an elegant approach for building continuous depth neural networks [16]. They offer various advantages over traditional neural networks, for example a constant memory cost as a function of depth and invertibility by construction. As they are inspired by dynamical systems, they can also take advantage of the vast amount of mathematical techniques developed in this field. Incorporating continuous depth also enables efficient density estimation [52] and modeling of irregularly sampled time series [125].

The aforementioned advantages come at the expense of a significantly increased computational cost compared with traditional models like Residual Networks [57]. The training dynamics of Neural ODEs become more complicated as training progresses and the number of steps taken by the adaptive ODE solver (typically referred to as the number of function evaluations) can become prohibitively large [36]. In fact, several works have argued that this is one of the most pressing limitations of Neural ODEs [16, 36, 43]. It is therefore important to design methods that can help reduce the computational cost of this tool.

A natural approach for tackling this issue is to explore regularization techniques that simplify the underlying dynamics for these models. Indeed, simpler dynamics have shown benefits in terms of faster convergence and a lower number of function evaluations both during training and testing [36]. [43] recently proposed an interesting technique for explicitly regularizing the dynamics based on ideas from optimal transport showing significant improvements in the effective convergence time. In this work, we propose a regularization technique in order to further accelerate training. Our regularization is orthogonal to previous approaches and can therefore be used in conjunction with them. Our technique is extremely simple: during training we randomly sample the upper limit of the ODE integration.

Specifically, the solver in Neural ODEs takes as input the integration limits (t_0, t_1). We propose to introduce regularization by randomly perturbing the end time parameter t_1 to be $t_1 \pm \delta$ where δ is a random variable. Empirically, we demonstrate that, in a wide variety of tasks, the proposed regularization achieves comparable performance to unregularized models at a significantly lower computational cost. We test our regularization technique on continuous normalizing flows [52], irregularly sampled time series [125] and image recognition tasks and show that it improves performance across all these tasks. It also performs better in certain instances of Stiff ODEs. Stiff ODEs are a widely studied topic in the field of engineering[24]. During training the dynamics learnt by the implicit model parametrized by the neural network can turn into an instance of a Stiff ODE. Techniques which could tackle the instances of Stiff ODE would prove to be instrumental as these implicit deep learning models mature.

5.2 Related Work

Connections between deep neural networks and dynamical systems were made as early as [84]. The adjoint sensitivity technique was explored in [116]. Some early libraries such as dolfin [42, 119], implemented the adjoint method for neural network frameworks. Previous works have shown the connection between residual networks and dynamical systems [126, 55, 97, 151, 147, 8]. By randomly dropping some blocks of a residual network, it was demonstrated that deep neural networks could be trained [62] or tested [144, 47] with stochastic depth, and that such a training procedure produces ensembles of shallower networks.

Neural ODEs [16] introduced a framework for training ODE models with backpropagation. Recent works [100, 158, 23, 67, 101, 27, 56, 108] have tried to analyze this framework theoretically and empirically. Neural ODEs were extended to work with flow based generative models [52]. In contrast to standard flow based models such as [78, 31, 112, 113], continuous

normalizing flows do not put any restrictions on the model architecture. Furthermore, Neural ODEs have been applied in the context of irregularly sampled time series models [125], reinforcement learning [34] and for set modeling[92]. Neural ODEs have also been explored for harder tasks such as video generation [156, 71] and for modeling the evolving dynamics of graph neural networks [154]. Recent work has also extended Neural ODEs to stochastic differential equations [91, 95, 61, 110, 152, 68].

The methods most relevant to our work are [75, 43] and [36]. A regularization based on principles of optimal transport which simplifies the dynamics of continuous normalizing flow models, was introduced in [43] to make training faster. Equations which are easier to learn were analyzed in [75] whereby they used a differential surrogate for the time cost of standard numerical solvers, using higher-order derivatives of solution trajectories. Augmented Neural ODEs [36] introduced a simple modification to Neural ODEs by adding auxillary dimensions to the ODE function and showing how this simplified the dynamics. Our work is complementary to these methods, and we empirically demonstrate that our method can be used to further improve the performance of these techniques.

There is a huge treasure of work which deals with instability in training ODEs[25, 26]. Curtiss et al. [24] were one of the first works that dealt with solving stiff equations. It was observed in Curtiss and Hirschfelder [24] that explicit methods failed for the numerical solution of ordinary differential equations that model certain chemical reactions. It is very difficult to integrate "stiff" equations by ordinary numerical methods. Small errors are rapidly magnified if the equations are integrated. The other case might be that the errors oscillate rapidly. Implicit methods such as the second order Adams-Moulton (AM2) [150] are elegant techniques which deal with some of the problems of stiff ODEs. In the context of Neural ODEs[16] implicit computations reduces the efficiency of the solvers. This effect is especially apparent in complex modeling tasks such as generative modeling using continuous normalizing flows[52] and irregularly sampled timeseries models[125]. Hence an analysis of simple scenarios of stiff equations which cause problems for standard Neural ODEs are discussed in the following sections.

5.3 Methodology

5.3.1 Neural ODEs

Neural ODEs can be interpreted as a continuous equivalent of Residual networks [57]. The transformation of a residual layer is $\mathbf{z}_{t+1} = \mathbf{z}_t + f_t(\mathbf{z}_t)$, where the representation at layer t is $\mathbf{z}_t$ and $f_t(\mathbf{z}_t)$ is a function parametrized by a neural network. The output dimension of the function $f_t(\mathbf{z}_t)$ should be the same as the input dimension to allow it to be added to the previous layer's features $\mathbf{z}_t$. This transformation can be rewritten as

$$\frac{\mathbf{z}_{t+\delta_t} - \mathbf{z}_t}{\delta_t} = f_t(\mathbf{z}_t), \quad \delta_t = 1, \tag{5.1}$$

which can be interpreted as finite difference approximation of the derivative. In contrast to traditional deep neural networks where $\delta_t = 1$ is fixed, Neural ODEs [16] introduced a continuous version by analysing Eq. 5.1 in the limit $\delta_t \to 0$. The resulting equation becomes

$$\frac{d\mathbf{z}(t)}{dt} = f(\mathbf{z}(t), t). \tag{5.2}$$

To transform an input $\mathbf{x}$ to an output $\mathbf{y}$, Residual networks use a sequence of k functions to transform the representation. In an analogous manner, Neural ODEs evolve the representation from the initial time $t = t_0,\ \mathbf{z}(t_0) = \mathbf{x}$, to the final time $t = t_1,\ \mathbf{z}(t_1) = \mathbf{y}$. The representation $\mathbf{z}(t)$ at any given time t can then be obtained by solving Eq. 5.2 using an off-the-shelf ODE solver with the function $f(\mathbf{z}(t), t)$ parameterized by the neural network. To update the parameters of the function $f(\mathbf{z}(t), t)$ such that $\mathbf{z}(t_1) \approx \mathbf{y}$ we backpropagate through the operations of the ODE solver to obtain the appropriate gradients.

5.3.2 Temporal Regularization

The formulation of a standard Neural ODE is given by

$$\mathbf{z}(t_1) = \mathbf{z}(t_0) + \int_{t_0}^{t_1} f(\mathbf{z}(t), t, \theta)dt = \text{ODESolve}(\mathbf{z}(t_0), f, t_0, t_1, \theta), \tag{5.3}$$

where $\mathbf{z}(t)$ is the latent representation at any time t. θ are the parameters of the function f and (t_0, t_1) are the limits of the integration. The initial condition of the ODE is set as $\mathbf{z}(t_0) = \mathbf{x}$.

Neural ODEs often take an excessively large time to train, especially in the case of continuous normalizing flows [52] . The dynamics become more complicated as training

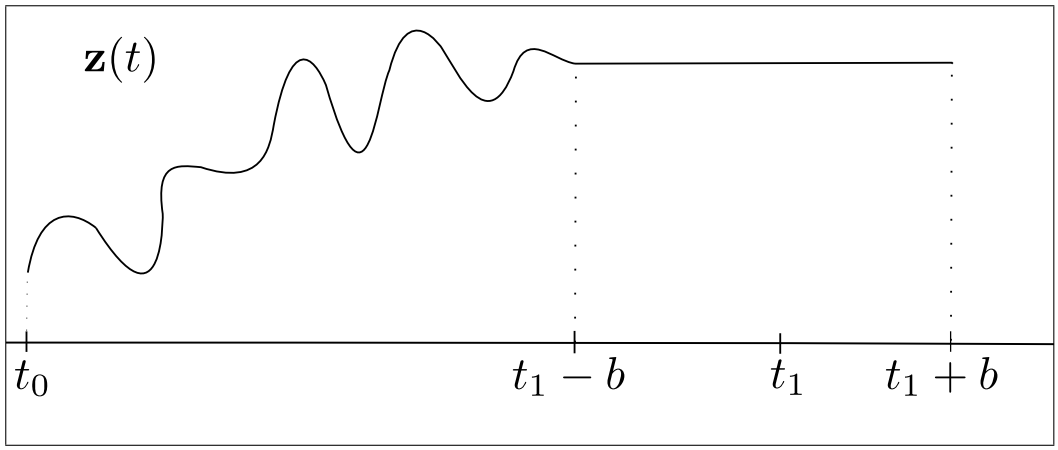


Figure 5.2: Effective behavior of Neural ODEs with STEER regularization. The solution is reached at a time $t_1 - b$ instead of time t_1.

progresses thus leading to an increase in the number of function evaluations. In the context of Neural ODEs, the dynamics of the ODE and training time are intricately related. In practice, simpler dynamics [36, 43] often lead to faster training with fewer function evaluations. Regularization is commonly used in machine learning to ease the training of a model and improve its generalization. Some common instances include Batchnorm [65] and early stopping [155].

Neural ODEs continuously evolve the representation $\mathbf{z}(t)$ with time t. Thus in the context of neural ODEs, an alternative type of regularization is possible, namely temporal regularization. In this work, we propose a simple regularization technique that exploits this temporal property. We propose to perturb the final time t_1 for the integration of the Neural ODE in Eq. 5.3 during training. At each training iteration, the final time t_1 for the integration is stochastically sampled. More formally, if we assume $t_0 < t_1$, then our regularization can be formulated as

$$\begin{aligned} \mathbf{z}(t_1) &= \mathbf{z}(t_0) + \int_{t_0}^{T} f(\mathbf{z}(t), t, \theta) dt = \text{ODESolve}(\mathbf{z}(t_0), f, t_0, T, \theta), \\ T &\sim \text{Uniform}(t_1 - b, t_1 + b), \\ b &< t_1 - t_0 : \text{parameter controlling interval of uniform distribution.} \end{aligned} \tag{5.4}$$

We call the technique STEER, which stands for Simple Temporal Regularization. Since Neural ODEs model a continuous series of transformations, applying STEER is essentially equivalent to performing a variable number of transformations on the input $\mathbf{x}$ in each training iteration, thereby introducing a new form of stochasticity in the training process.

We observe that the proposed regularization effectively ensures convergence to the solution at time $t_1 - b$. The effective behavior of $\mathbf{z}(t)$ can be see in Fig. 5.2

5.3.3 Stiff ODEs

Stiff ODEs have been extensively researched in the field of differential equations [12]. Inspite of years of research a concrete definition of Stiff ODEs has not been agreed upon [136]. We highlight how some of the problems of Stiff ODEs could impact the training dynamics of Neural ODEs in specific scenarios. Certain discussions of Stiff ODEs define that the stiffness concerns the underlying mathematical form of the equation, integration time and a set of initial data.

In terms of the mathematical equation, stiffness is a problem that can arise when numerically solving ODEs. In such systems there is typically a fast and a slow transient component which causes adaptive ODE solvers to take exceedingly small steps. This increases the number of function evaluations and the time taken to integrate the function. Furthermore, in Neural ODEs, the function $\frac{d\mathbf{z}}{dt}$ is parameterized by a neural network which is updated during training, and could become stiff. This would be hard for a traditional ODE solver to integrate efficiently, thereby making this a limitation.

Stiffness ratio is one of the techniques used to characterize some forms of stiff equations. The stiffness ratio is used to characterize the stiffness for the general case of linear constant coefficient systems. These systems can be represented as

$$\frac{d\mathbf{z}}{dt} = \mathbf{A}\mathbf{z} + f(t), \tag{5.5}$$

where $\mathbf{z}, f(t) \in \mathbb{R}^n$ and $\mathbf{A}$ is a $n \times n$ matrix with eigenvalues $\lambda_i \in \mathbb{C}, i = 1, 2, \ldots, n$ (assumed distinct) and corresponding eigenvectors $\mathbf{c_i} \in \mathbb{C}^n, i = 1, 2, \ldots, n$. The eigenvalues and eigenvectors can be complex. The general solution of Eq. 5.9 takes the form

$$\mathbf{z}(t) = \sum_{i=1}^{n} \kappa_i \exp(\lambda_i t)\mathbf{c}_i + \mathbf{g}(t). \tag{5.6}$$

If λ_i is complex, the term $\exp(\lambda_i t)$ varies sinusoidally. The stiffness ratio is defined as $\frac{\max_{i=1\ldots n}|Re(\lambda_i)|}{\min_{i=1\ldots n}|Re(\lambda_i)|}$, where $Re(\lambda_i)$ denotes the real component of $\lambda_i \in \mathbb{C}$. Ordinarily, temporal regularization might fail in the face of stiffness. However, in our empirical experiments we observed that at least in the case of toy problems, STEER actually performed better. This analysis is provided in the next section.

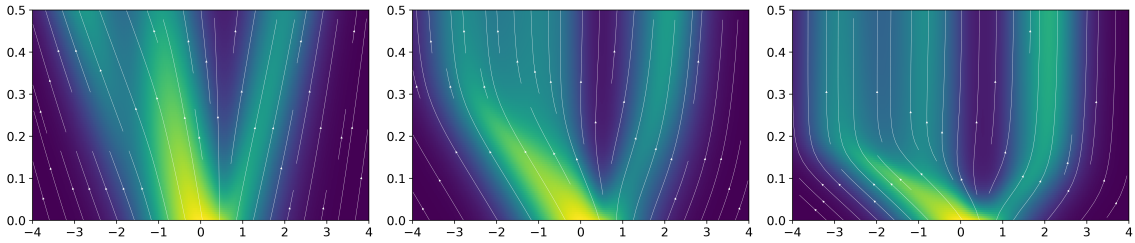


Figure 5.3: Plots showing the path taken by FFJORD with STEER regularization. STEER regularization ($b = 0.25$ and $b = 0.375$) performs the same transformation in shorter paths.

5.4 Experiments

In this section, we demonstrate the effectiveness of the proposed method on several classes of problems. First, we conduct experiments on continuous normalizing flow based generative models, and compare against FFJORD [52] and RNODE [43]. Then, we conduct experiments on irregularly sampled time-series models, and compare against the models and baselines proposed in Latent ODEs [125]. We also conduct experiments on feedforward models for image recognition, and compare against Neural ODEs and Augmented Neural ODEs [36]. We finally evaluate STEER and standard Neural ODEs on a toy stiff ODE experiment.

5.4.1 Generative Models

Continuous normalizing flow based generative models [52, 16] provide an alternate perspective to generative modeling. Let $p(\mathbf{x})$ be the likelihood of a datapoint $\mathbf{x}$ and $p(\mathbf{z})$ be the likelihood of the latent $\mathbf{z}$. FFJORD [52] showed that under certain conditions the change in log probability also follows a differential equation $\frac{d\log(p(\mathbf{z}(t)))}{dt} = -\text{tr}\left(\frac{df}{d\mathbf{z}(t)}\right)$. Thus the total change in log probability can be obtained by integrating the equation which leads to

$$\log p(\mathbf{z}(t_1)) = \log p(\mathbf{z}(t_0)) - \int_{t_0}^{t_1} \text{tr}\left(\frac{df}{d\mathbf{z}(t)}\right) dt. \tag{5.7}$$

This allows to compute the point $\mathbf{z_0}$ which generates a given datapoint $\mathbf{x}$, and $\log p(\mathbf{x})$ under the model by solving the initial value problem:

$$\underbrace{\begin{bmatrix} \mathbf{z_0} \\ \log p(\mathbf{x}) - \log p_{\mathbf{z_0}}(\mathbf{z_0}) \end{bmatrix}}_{\text{solutions}} = \underbrace{\int_T^{t_0} \begin{bmatrix} f(\mathbf{z}(t), t; \theta) \\ -\text{tr}\left(\frac{df}{d\mathbf{z}(t)}\right) \end{bmatrix} dt}_{\text{dynamics}}, \qquad \underbrace{\begin{bmatrix} \mathbf{z}(T) \\ \log p(\mathbf{x}) - \log p(\mathbf{z}(T)) \end{bmatrix} = \begin{bmatrix} \mathbf{x} \\ \mathbf{0} \end{bmatrix}}_{\text{initial values}}. \tag{5.8}$$

	MNIST		CIFAR-10		IMAGENET-32	
	BITS/DIM	TIME	BITS/DIM	TIME	BITS/DIM	TIME
FFJORD, VANILLA.	0.974	68.47	3.40	> 5 days	3.96	> 5 days
FFJORD RNODE. [43]	0.973	24.37	3.398	31.84	2.36	30.1
FFJORD, VANILLA. + STEER ($b = 0.5$)	0.974	51.21	3.40	86.34	3.84	> 5 days
FFJORD, VANILLA. + STEER ($b = 0.25$)	0.976	58.21	3.41	103.4	3.87	> 5 days
FFJORD RNODE + STEER ($b = 0.5$)	**0.971**	**16.32**	**3.397**	**22.24**	**2.35**	**24.9**
FFJORD RNODE + STEER ($b = 0.25$)	0.972	19.71	3.401	25.24	2.37	27.84

Table 5.1: Test accuracy and training time comparison on various image datasets. Unless mentioned as days, the time reported is in hours.

Model	**Accuracy**
Latent ODE (ODE enc)	0.846 ± 0.013
Latent ODE (RNN enc.)	0.835 ± 0.010
ODE-RNN	0.829 ± 0.016
Latent ODE (ODE enc, STEER)	$\mathbf{0.880 \pm 0.011}$
Latent ODE (RNN enc, STEER)	0.865 ± 0.021
ODE-RNN (STEER)	0.872 ± 0.012

Table 5.2: Per-time-point classification accuracies on the Human Activity dataset.

Model	**AUC**
Latent ODE (ODE enc)	0.829 ± 0.004
Latent ODE (RNN enc.)	0.781 ± 0.018
ODE-RNN	$\mathbf{0.833 \pm 0.009}$
Latent ODE (ODE enc, STEER)	0.810 ± 0.018
Latent ODE (RNN enc, STEER)	0.772 ± 0.014
ODE-RNN (STEER)	0.811 ± 0.007

Table 5.3: Per-sequence classification AUC on Physionet.

The combined dynamics of $\mathbf{z}(t)$ and the log-density of the sample is integrated backwards in time from T to t_0. The log likelihood $\log p(\mathbf{x})$ can be computed by adding $\log p_{\mathbf{z_0}}(\mathbf{z_0})$ to the solution of Eq. 5.8. A major drawback of continuous normalizing flow based models is extremely slow convergence. STEER regularization enables faster convergence by sampling $T \sim \text{Uniform}(t_1 - b, t_1 + b)$ at each iteration.

We focus on the task of density estimation of image datasets using the same architectures and experimental setup as FFJORD [52]. More details are provided in the supplementary material. The performance of the model is evaluated on multiple datasets, namely MNIST [88], CIFAR-10 [81] and ImageNet-32 [29], and the results are shown in Table 5.1. The performance is also analyzed in the simple scenario of a 1D Mixture of Gaussians as shown in Fig. 5.3.

It can be seen in Table 5.1 that models with STEER regularization converge much faster. In some cases, such as on CIFAR-10, our model is 30% faster than the state of the art.

Fig. 5.3 shows the evolution of the probability distribution from the latent distribution $p(\mathbf{z})$ to the data distribution $p(\mathbf{x})$ at different test times. It can be seen that the trajectories reach the final states at an earlier test time with STEER regularization.

Model	Interpolation (% Observed Pts.)				Extrapolation (% Observed Pts.)			
	10 %	20 %	30 %	50 %	10 %	20 %	30 %	50 %
Latent ODE (ODE enc)	0.360	0.295	0.300	0.285	1.441	1.400	1.175	1.258
Latent ODE (RNN enc.)	2.477	0.578	2.768	0.447	1.663	1.653	1.485	1.377
ODE-RNN	1.647	1.209	0.986	0.665	13.508	31.950	15.465	26.463
Latent ODE (ODE enc, STEER)	**0.230**	**0.235**	**0.246**	**0.210**	**1.01**	**0.98**	**1.01**	**1.04**
Latent ODE (RNN enc., STEER)	2.31	0.561	2.534	0.386	1.541	1.498	1.236	1.119
ODE-RNN (STEER)	1.528	1.09	0.796	0.603	11.508	29.34	14.23	23.528

Table 5.4: Test Mean Squared Error (MSE) ($\times 10^{-2}$) on the MuJoCo dataset.

5.4.2 Times Series Models

For time series models, the function is evaluated at multiple times $t_0, ..t_n$. We employ the regularization between every consecutive pair of points (t_i, t_{i+1}). More formally, if the time is split into consecutive intervals $(t_0, t_1)...(t_{n-1}, t_n)$, we start with t_0 and the initialization $\mathbf{z}(t_0)$ and sample the end time as $T \sim \text{Uniform}(t_1 - b, t_1 + b)$. The resulting integration using the ODE solver gives us the latent representation $\mathbf{z}(t_1)$. We repeat the same procedure where the limits of integration are now (t_1, t_2). In this iteration t_1 is not altered while $T \sim \text{Uniform}(t_2 - b, t_2 + b)$ is sampled to obtain $\mathbf{z}(t_2)$ and so on. In the case of irregularly sampled time series models, the length of the interval $t_{i+1} - t_i$ can itself vary. The parameter b in this case is adaptive such that $b = |t_{i+1} - t_i| - \epsilon$. The ϵ parameter is added to avoid numerical instability issues. In the case of time series models the parameter b changes at each forward pass thus adding another source of stochasticity.

Latent ODEs [125] introduced irregularly sampled time series models in which Neural ODEs perform much better. Indeed these datasets break the basic assumptions of a standard RNN, namely that the data are not necessarily collected at regular time intervals. This is the case in several real life scenarios such as patient records or human activity recognition which are observed at irregularly sampled timepoints.

The Mujoco experiments consist of a physical simulation of the "Hopper" model from the Deepmind control suite. The initial condition uniquely determines the trajectory. It consists of $10,000$ sequences of 100 time points each. The Human Activity dataset [125] consists of sensor data fitted to individuals. The sensor data is collected using tags attached to the belt, chest and ankles of human performers. It results in 12 features in total. The task involves classifying each point as to which activity was being performed. Physionet [133] consists of time series data from initial 48 hours of ICU admissions. It consists of sparse measurements where only some of the features' measurements are taken at an instant. The measurement

	NODE	NODE(STEER)	ANODE	ANODE(STEER)
MNIST	$97.2\% \pm 0.2$	$98.3\% \pm 0.3$	$98.2\% \pm 0.1$	$\mathbf{98.6\% \pm 0.3}$
CIFAR-10	$53.7\% \pm 0.4$	$54.9\% \pm 0.5$	$60.6\% \pm 0.4$	$\mathbf{62.1\% \pm 0.2}$
SVHN	$81.0\% \pm 0.6$	$81.8\% \pm 0.3$	$83.5\% \pm 0.5$	$\mathbf{84.1 \pm 0.2}$

Table 5.5: Test accuracies on Feedforward models on various datsets.

frequency is also irregular which makes the dataset quite challenging. To predict mortality, binary classifiers were constructed by passing the hidden state to a 2 layer binary classifier.

We use the same architecture and experiment settings as used in Latent ODEs [125]. For the Mujoco experiment 15 and 30 latent dimensions were used for the generative and recognition models respectively. The ODE function had 500 units in each of the 3 layers. A 15 dimensional latent state was used for the Human Activity dataset. For the Physionet experiments the ODE function has 3 layers with 50 units each. The classifier consisted of 2 layers with 300 units each.

In the case of irregularly sampled timeseries with Latent ODEs, the number of function evaluations is not a major concern since the training time is quite small, so we compare only in terms of accuracy. It can be seen in Table 5.2 and Table 5.4 that STEER regularization consistently improves the performance of ODE based models. Table 5.4 demonstrates that the extrapolation results are much better with STEER regularization, indicating that it may help improve the generalization of the model. However in the case of Physionet Table 5.3, the results do not improve over standard Neural ODEs, although the performance remains very close.

5.4.3 Feedforward Models

In feedforward learning for classification and regression, the Neural ODE framework can be used to map the input data $\mathbf{x} = \mathbf{z}(t_0) \in \mathbb{R}^d$ to features $\mathbf{z}(t) \in \mathbb{R}^d$ at any given time t. For standard neural ODEs, one integrates in the range (t_0, t_1), and for STEER the integration is performed in the range (t_0, T) where $T \in \text{Uniform}(t_1 - b, t_1 + b)$. Once the features $\mathbf{z}(T)$ at the final time T are obtained, a linear layer $\mathbb{R}^d \to \mathbb{R}$ is used to map the features to the correct dimension for downstream tasks such as classification or regression.

We evaluate STEER on a similar set of experiments as performed in [36] on the MNIST [88], CIFAR-10 [81] and SVHN [107] datasets using convolutional architectures for $f(\mathbf{z}(t), t)$. We use the same architectures and experimental settings as used in [36] for both Neural

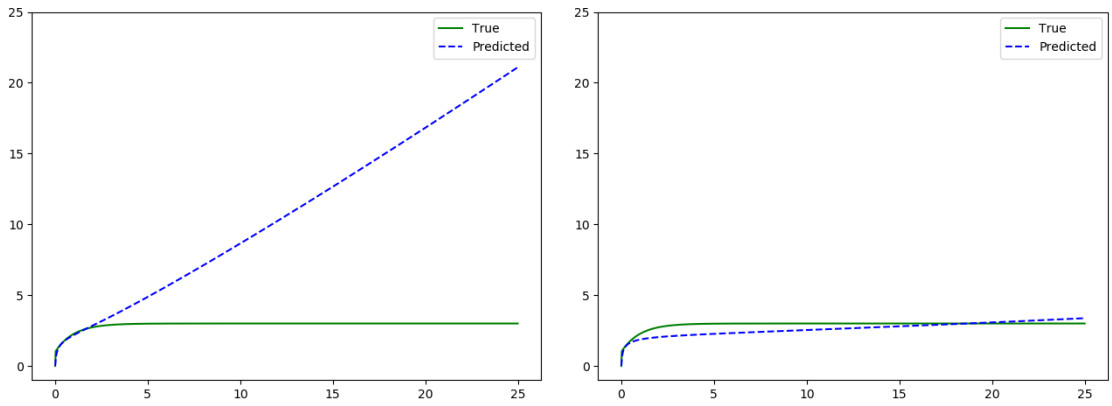


Figure 5.4: Standard Neural ODE struggles while STEER regularization helps to fit a stiff ODE with stiffness ratio $r = 1000$.

ODEs (NODE) and Augmented Neural ODEs (ANODE). For the MNIST experiments, 92 filters were used for NODEs while 64 filters were used for ANODEs to compensate for 5 augmented dimensions. For the CIFAR-10 and SVHN experiments, 125 filters were used for NODEs while 64 filters were used for ANODEs with 10 augmented dimensions. Such a design was employed to ensure a roughly similar number of parameters for a fair comparison.

Table 5.5 demonstrates the performance of STEER regularization with $b = 0.99$. STEER provides a small boost in accuracy and performs consistently across all datasets.

5.4.4 A Toy Example of Stiff ODE

In this experiment, we analyze a toy example of stiff ODE from [13]. The aim of this experiment is to highlight the vulnerability of Neural ODEs to stiffness. The experimental setting for section 4.4 was similar to the one used in the official github repository [1] of Neural ODE. Instead of using a 2-D ODE we used a 1-D ODE to be able to better visualize and analyze the behavior. We used Eq 12 in the paper as mentioned in the numerical methods book by [13] on page 752. It was surprising that a Neural ODE was able to model a 2 dimensional model with ease while the stiff equation was extremely hard for the Neural ODE to model.

The ODE is given by

$$\frac{dy}{dt} = -1000y + 3000 - 2000e^{-t}, \tag{5.9}$$

[1] https://github.com/rtqichen/torchdiffeq/blob/master/examples/ode_demo.py

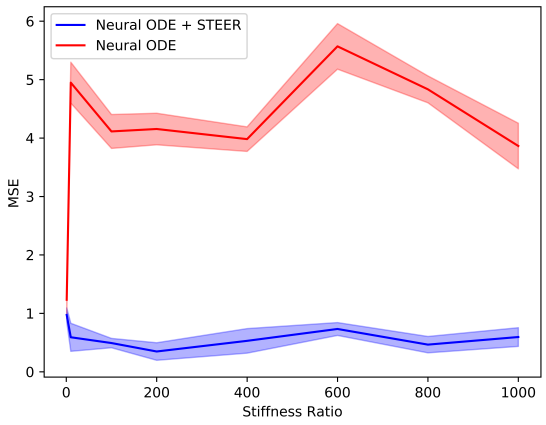


Figure 5.5: Model with STEER regularization achieves a much lower MSE on a Stiff ODE as the stiffness ratio increases.

with the initial condition of $y(0) = 0$. The solution for the above stiff ODE is $y = 3 - 0.998e^{-1000t} - 2.002e^{-t}$. We use a simple architecture with a single hidden layer of dimension 500 for both the standard Neural ODE and the one with STEER. The effective input dimension is 2 resulting by concatenating y and t, and the output dimension is 1.

As shown in Fig. 5.4, the solution is initially dominated by the fast exponential term e^{-1000t}. After a short period $(t < 0.005)$, this transient dies down and the solution is dictated by the slow exponential e^{-t}. The training interval $[0, 15]$ is broken down into small intervals of length $(t_1 - t_0) = 0.125$. We sample 1000 random initial points t_0 and use $y(t_0)$ as the initialization and then try to estimate the value of the function $y(t_1)$ at $t_1 = t_0 + 0.125$. It can be seen in Fig. 5.4 that standard Neural ODE doesn't show the same asymptotic behavior as the actual solution, whereas STEER regularization enables better asymptotic behavior. The minimum Mean Squared Error(MSE) loss over all the points of the trajectory in the test interval $[0, 25]$ for standard Neural ODE is 3.87, and with STEER regularization with $b = 0.124$ it is 0.52.

To further study the behaviour with varying stiffness ratio, the formula in Eq. 5.9 is generalized as $\frac{dy}{dt} = -ry + 3r - 2re^{-t}$. The generalized equation has the same asymptotic behavior as the original. It also reaches a steady state at $y = 3$. The $-ry$ term would introduce a term of e^{-rt} and another term of e^{-t} would be present in the solution. Thus the stiffness ratio as described in Subsection 5.3.3 is exactly r. The plot in Fig. 5.5 shows that as the stiffness increases, the STEER regularization compares better in terms of MSE to models without the regularization. The experiments were performed using the dormand-prince [33] ODE solver.

While we do not promise to have proposed a solution for the difficult problem of Stiff-ODEs [13], we empirically observed that it gives improvement on toy examples. We want to highlight that further research is needed in this important area of Stiffness. We show analysis about the choice of b for our technique in the appendix.

5.4.5 A Toy example of a Timeseries Model:

In this experiment, we analyze a toy example of irregularly sampled timeseries models. The experimental setting was the same as in [125]. The models were tested on a toy dataset of periodic trajectories. The initial point was sampled from a standard Gaussian. Gaussian noise was added to the subsequent observations. Each trajectory has a 100 observations. A fixed subset of these points from each trajectory are sampled. The entire set of 100 observations are attempted to be reconstructed. In the original experiment by [125] the amplitude was the same throughout the trajectories while the frequency was varied for the $A.sin(\phi t)$ where ϕ is the frequency of the sine component. We made a simple change to the experimental setting whereby we also added a t component to the sampled periodic trajectories. Thus the final form of the trajectories being sampled are $A.sin(\phi t) + t$ where A is constant and ϕ can vary within the sampled trajectories. In this experimental setting we see that standard Neural ODEs start to produce trajectories which are qualitatively quite different from the actual sampled trajectories. STEER on the other hand produces trajectories which match the qualitative behavior of the sampled trajectories more closely.

5.5 Conclusion

In this paper we introduced a simple temporal regularization method for Neural ODEs. The regularization consists of perturbing the final time of the integration of the ODE solver. The main advantage of our work is the empirical evidence showing the regularization results in simpler dynamics for Neural ODEs models on a wide variety of tasks. This is demonstrated for a specific stiff ODE that is challenging for a basic Neural ODE solver, faster continuous normalizing flow models and improved or comparable accuracy for time series models and deep feedforward networks. We further show that our approach is orthogonal to other regularization approaches and can be thus combined with them to further improve Neural ODEs.

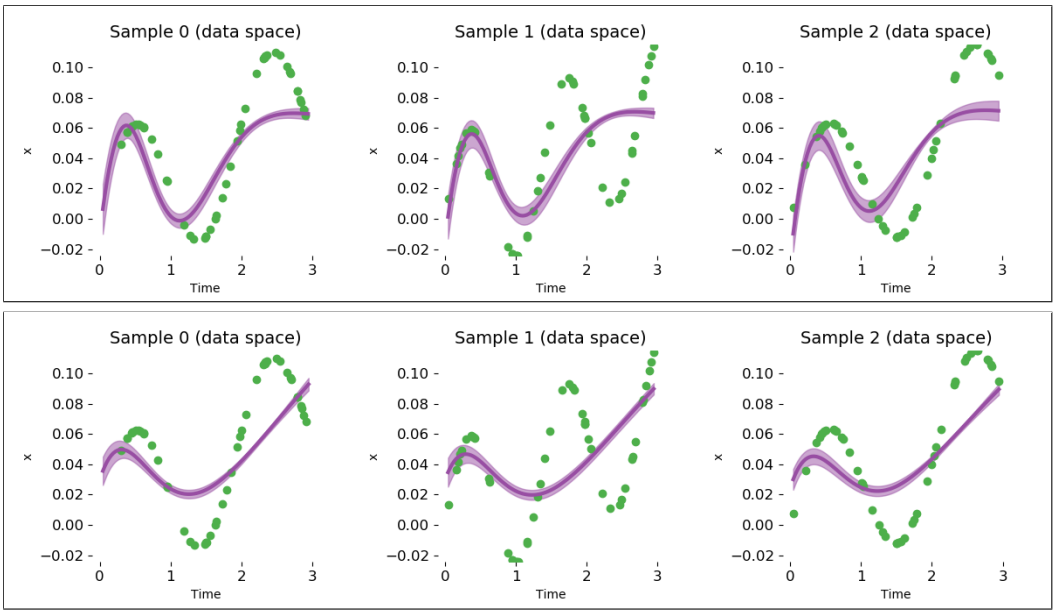


Figure 5.6: A simple addition of t to $A.sin(\phi t)$ changes the qualitative performance of Neural ODE vs STEER. The top row demonstrates the 3 sample trajectories that have been irregularly sampled using STEER, while the bottom row is with a standard Neural ODE. The green points indicate the irregularly sampled points, while the rest of the points are predicted using each trained model. The amplitude A of the model is the same across the various samples but the frequency ϕ is variable across the samples. As we see from the sampled trajectories STEER models the qualitative behavior of the trajectories better than standard Neural ODEs.

5.6 Modified Picard's Iteration:

Let the differential equation be

$$\frac{d\phi}{dt} = f(t, \phi(t)), \quad \phi(t_0) = z_0. \tag{5.10}$$

The modified Picard's iteration can be formulated as

$$\phi_0(t) = z_0, \tag{5.11a}$$

$$\phi_{k+1}(t) = z_0 + \int_{t_0}^{t+\delta_{k+1}} f(s, \phi_k(s))ds, \tag{5.11b}$$

$$\delta_{k+1} \sim \text{Uniform}(-b, b). \tag{5.11c}$$

Although the modified Picard's iteration is very close to our technique, it is not the exact same process that is simulated by our proposed technique. There are subtle differences which make the proposed technique different to the one simulated by the modified Picard's iteration.

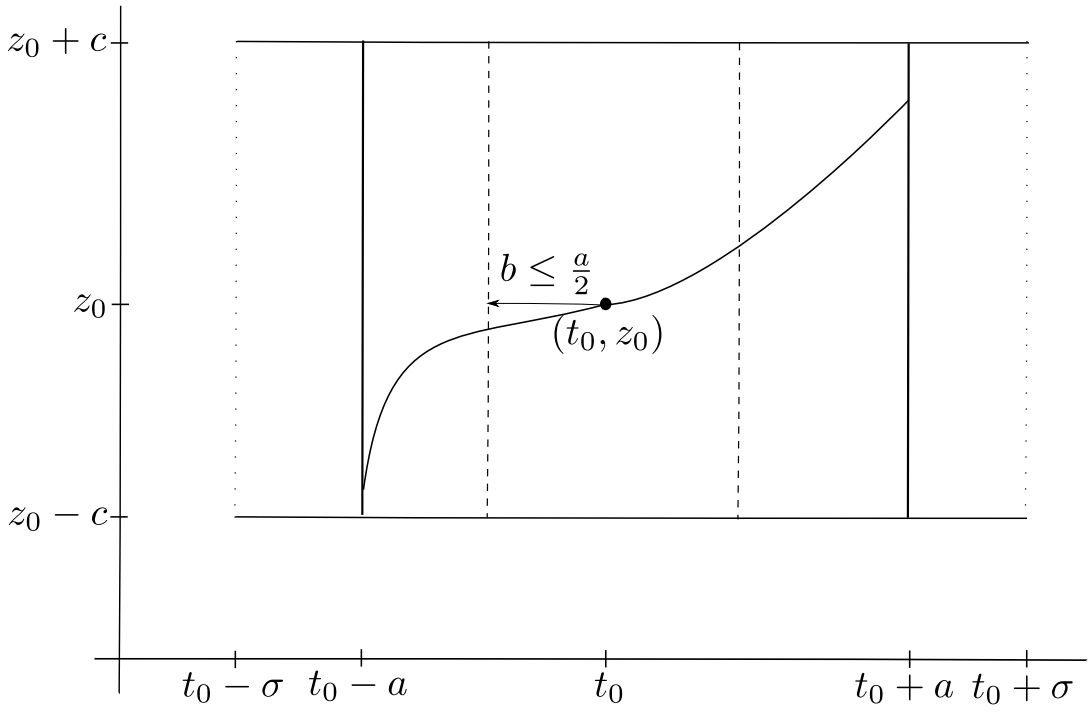


Figure 5.7: Conditions of the proof.

Picard's iteration constructs a sequence of approximate functions $\{\phi_k(t)\}$ which eventually converge to the desired solution. Eq. 5.11a defines the initial approximation $\phi_0(t)$ as the initial condition z_0 of the ODE. Eq. 5.11b describes the recurrence relation that relates $\phi_{k+1}(t)$ to $\phi_k(t)$. The recurrence relation adds a δ_{k+1} (Eq. 5.11c) term which is randomly sampled.

It can be seen that the right-hand side Eq. 5.11b defines an operator that maps a function ϕ to a function $T[\phi]$ as

$$T[\phi_k](t) = \phi_0 + \int_{t_0}^{t+\delta} f(s, \phi_k) ds, \quad \delta \sim \text{Uniform}(-b, b). \tag{5.12}$$

The following theorem shows the existence of a unique solution for the modified Picard's iteration for our method.

Theorem 2. *Suppose that* $f : \mathbb{R}^2 \to \mathbb{R}$ *satisfies the Lipschitz condition* $|f(t, x_2) - f(t, x_1)| \leq L|x_2 - x_1|$ *where* $0 \leq L < \infty$. *Suppose that* f *is continuous and* $\exists\ M$ *such that* $0 \leq$

$M < \infty,\ |f(t,x)| \leq M,\ \forall (t,x)$. *Then the sequence* $\{\phi_k\}$ *generated by the iteration* $\phi_{k+1} = T[\phi_k], \phi_0(t) = z_0$ *is a contraction in expectation.*

The sequence $\{\phi_k(t)\}$ *converges to a unique fixed point* $\phi^*(t)$.

Proof. Let the starting range of t for the analysis be $(t_0 - \sigma, t_0 + \sigma)$. Let c be such that $\forall i, |\phi_i(t) - \phi_0| \leq c$. The solution only exists in the range $t \in (t_0 - \frac{a}{2}, t_0 + \frac{a}{2})$ where $a < \min(\frac{c}{M}, \frac{1}{2L})$. The parameter b for STEER is chosen as $b \leq \frac{a}{2}$ to ensure that the final effective time after sampling $\delta\ t + \delta \in (t_0 - a, t_0 + a)$.

Let $\phi_{k+1}(t) = T[\phi_k] = \phi_0 + \int_{t_0}^{t+\delta} f(s, \phi_k) ds, \quad \delta \sim \text{Uniform}(-b, b)$ is well defined on $[t_0 - a, t_0 + a]$. $\phi_{k+1}(t)$ is continuous since both $\phi_k(t)$ and f are continuous.

$\phi_{k+1}(t) \in \mathbb{R}$ since $|\phi_{k+1}(t) - \phi_0| = |\int_{t_0}^{t_1} f(s, \phi_k(s)) ds| \leq M|t - t_0| \leq Ma < c$. This is by choice of a.

Let the metric on the space of solutions Φ be defined such that if $\Delta(\phi_k, \phi_{k+1}) = \max_{[t_0-a, t_0+a]} |\phi_k(t) - \phi_{k+1}(t)|$. Φ is a complete metric space which implies that all Cauchy sequences converge. We show using Lemma 1 that the operator T is a contraction in expectation. Lemma 3 shows the convergence of the sequence of functions $\{\phi_k\}$. Finally Lemma 4 shows why the fixed point is unique with high probability. □

Lemma 1. $\mathbb{E}_{|\delta_2| - |\delta_1|}\, \Delta(T\phi_1, T\phi_2) \leq \frac{1}{2}\Delta(\phi_1, \phi_2)$

Proof.

$$
\begin{aligned}
&|T\phi_2(t) - T\phi_1(t)| \\
&= |\int_{t_0}^{t+\delta_2} (f(s, \phi_2(s)) ds - \int_{t_0}^{t+\delta_1} (f(s, \phi_1(s))) ds| \\
&= |\int_{t_0}^{t} (f(s, \phi_2(s) - f(s, \phi_1(s))) ds + \int_{t}^{t+\delta_2} (f(s, \phi_2(s))) ds - \int_{t}^{t+\delta_1} (f(s, \phi_1(s))) ds| \\
&\leq |\int_{t_0}^{t} (f(s, \phi_2(s) - f(s, \phi_1(s))) ds + \int_{t}^{t+\delta_2} (f(s, \phi_2(s))) ds - \int_{t}^{t+\delta_1} (f(s, \phi_1(s))) ds| \\
&\leq |L \int_{t_0}^{t} |\phi_2(s) - \phi_1(s)| ds + M|\delta_2| - M|\delta_1|| \\
&\leq |L\Delta(\phi_2, \phi_1) \int_{t_0}^{t} ds + M(|\delta_2| - |\delta_1|)| \\
&\leq |L\Delta(\phi_2, \phi_1)(t - t_0) + M(|\delta_2| - |\delta_1|)| \\
&\leq |\Delta(\phi_2, \phi_1) La + M(|\delta_2| - |\delta_1|)| \\
&\leq |\frac{1}{2}\Delta(\phi_2, \phi_1) + M(|\delta_2| - |\delta_1|)|
\end{aligned}
\tag{5.13}
$$

Since a is chosen such that $a < min(\frac{c}{M}, \frac{1}{2L})$, hence $La < \frac{1}{2}$. If $\delta_i \sim U(-b, b)$ then $|\delta_i| \sim U(0, b)$. Further $\mathbb{E}_{|\delta_2|-|\delta_1|}(|\delta_2| - |\delta_1|) = 0$ using Lemma 2. Thus $\mathbb{E}_{|\delta_2|-|\delta_1|}\, \Delta(T\phi_1, T\phi_2) \leq \frac{1}{2}\Delta(\phi_1, \phi_2)$ □

Lemma 2. $\mathbb{E}_{|\delta_{i+1}|-|\delta_i|}(|\delta_{i+1}| - |\delta_i|) = 0$ *since* $|\delta_i|, |\delta_{i+1}| \sim$ *Uniform*$(0, b)$ *. The difference of 2 uniform random variables* $U(0, b)$ *follows the standard triangular distribution.*

Proof. Let $X_1 = |\delta_{i+1}|, X_2 = |\delta_i|$ are independent $U(0, b)$ random variables. Let $Y = X_1 - X_2$. The joint probability density of X_1 and X_2 is $f_{X_1,X2}(x_1, x_2) = 1$, $0 < x_1 < b, 0 < x_2 < b$ Using the cumulative distribution technique, the c.d.f of Y is

$$\begin{aligned} F_Y(y) &= P(Y \leq y) \\ &= P(X_1 - X_2 \leq y) \\ &= \begin{cases} \int_0^{b+y} \int_{x_1-y}^{b} 1 dx_2 dx_1 & -B < y < 0 \\ 1 - \int_y^b \int_0^{x_1-y} 1 dx_2 dx_1 & 0 \leq y < b \end{cases} \\ &= \begin{cases} \frac{b^2}{2} + by + \frac{y^2}{2} & -b < y < 0 \\ 1 - \frac{b^2}{2} + by - \frac{y^2}{2} & 0 \leq y < b \end{cases} \end{aligned} \tag{5.14}$$

Differentiating w.r.t y yields the probability distribution function :

$$f_Y(y) = \begin{cases} y + b & -b < y < 0 \\ y - b & 0 \leq y < b \end{cases} \tag{5.15}$$

From the properties of standard triangular distribution, $E_Y[Y] = 0$

□

Lemma 3. *The sequence of functions* $\{\phi_k\}$ *obtained using the transformation T as* $\phi_0(t) = \phi_0,\ \phi_{k+1} = T\phi_k$ *converges.*

Proof. $\Delta(\phi_2, \phi_1) = \Delta(T\phi_1, T\phi_0) \leq \frac{1}{2}\Delta(\phi_1, \phi_0)$.

Similarly, $\Delta(\phi_3, \phi_2) = \Delta(T\phi_2, T\phi_1) \leq \frac{1}{2}\Delta(\phi_2, \phi_1) \leq \frac{1}{4}\Delta(\phi_1, \phi_0)$.

In general $\Delta(T\phi_{n+1}, T\phi_n) \leq \left(\frac{1}{2}\right)^n \Delta(\phi_1, \phi_0)$

$\implies \Sigma_{n=0}^{\infty}\Delta(T\phi_{n+1}, T\phi_n) \leq \Delta(\phi_1, \phi_0)\Sigma_{n=0}^{\infty}\left(\frac{1}{2}\right)^n$

Since the above sum converges and the completeness of Φ proves that the sequence $\{\phi_k\}$ converges. □

Lemma 4. *T has at most one fixed point with high probability.*

Proof. Suppose there were 2 distinct fixed points ϕ_1 and ϕ_2. By the definition of a fixed point $T\phi_k = \phi_k$, hence we obtain $\Delta(T\phi_1, T\phi_2) = \Delta(\phi_1, \phi_2)$ which contradicts Lemma 1 with high probability as Lemma 1 shows that $|T\phi_2(t) - T\phi_1(t)| \leq \frac{1}{2}\Delta(\phi_2, \phi_1) + M(|\delta_2| - |\delta_1|)$. Hence $\Delta(T\phi_1, T\phi_2)$ would not be equal to $\Delta(\phi_1, \phi_2)$ with high probability. □

5.7 Stiff ODE: Ablation Studies

As discussed in the experiments section of the paper, we use the same setting as the one described in [13].

The ODE is given by

$$\frac{dy}{dt} = -1000y + 3000 - 2000e^{-t}, \tag{5.16}$$

with the initial condition of $y(0) = 0$. We use the generalized version of the above equation which is $\frac{dy}{dt} = -ry + 3r - 2re^{-t}$. The generalized equation has the same asymptotic behavior as the original. It also reaches a steady state at $y = 3$. Varying r effectively varies the stiffness ratio of the underlying ODE. It thus allows us to analyze the behavior of the various hyperparameters across a wide range of underlying problem difficulty. The experiments were performed using the dormand-prince [33] ODE solver.

Since the experimental setting of the stiff ODE converges in minutes rather than hours, we test out a variety of settings for b. We try to identify strategies for choosing the hyperparameter b for the proposed STEER regularization. We also consider the use of a Gaussian distribution in place of the Uniform distribution. We further test out the effect of the capacity on the regularization. We perform experiments to observe the behavior by varying the number of units in the hidden layer of the neural network.

Effect of varying b: As we observe from Fig. 5.8 as the parameter b varies, we obtain a range of behavior in terms of the MSE error across a wide variety of stiffness ratios. The general trend indicates that larger b such as $b = 0.124, 0.115, 0.085$ have similar behavior and achieve the minimum error in general. Smaller b on the other hand shows behavior similar to standard Neural ODE as is evident from the plots of $b = 0.025, 0.045$. An intermediate value of $b = 0.065$ shows behavior which is better than very small values of b while worse behavior than the large values of b. This indicates that higher values of b are better for the proposed STEER regularization as long as b is less than the length of the original interval.

Effect of varying distributions: In the proposed STEER regularization we use the Uniform distribution to sample the end point of the integration. The simplicity of the Uniform

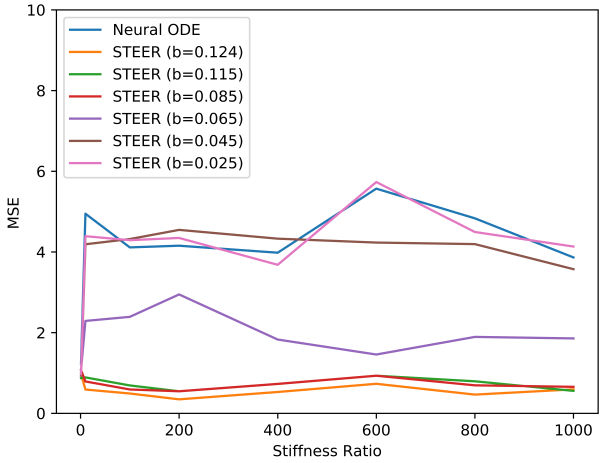


Figure 5.8: Comparison of the losses for the various choices of b across varying stiffness ratios.

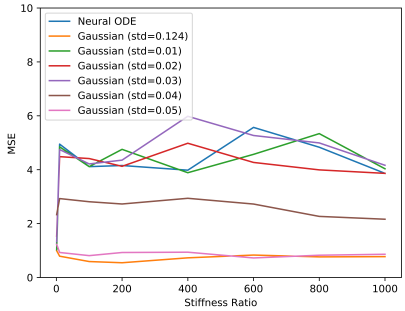


Figure 5.9: Comparison of the losses for the various choices of standard deviation (std) of the Gaussian across varying stiffness ratios.

distribution adds an elegance to the proposed technique. We want to analyze whether it is the inherent stochasticity that makes the technique effective or the particular choice of the Uniform distribution. As we see from Fig. 5.9 we observe a similar trend as we had seen in Fig. 5.8. Greater the stochasticity in the end time the better the performance in terms of MSE.

Delving deeper into the experimental setting of the Gaussian distribution, we sample an end time $t \sim N(t_1, std)$, where t_1 was the original end time of the integration and std is the parameter controlling the standard deviation of the Gaussian distribution. Fig. 5.9 indicates that higher standard deviations $std = 0.124, 0.05$ lead to better performance in terms of MSE. Very small standard deviations start approaching a behavior that is similar to the one shown by standard Neural ODE as exemplified by $std = 0.01, 0.02, 0.03$. It is

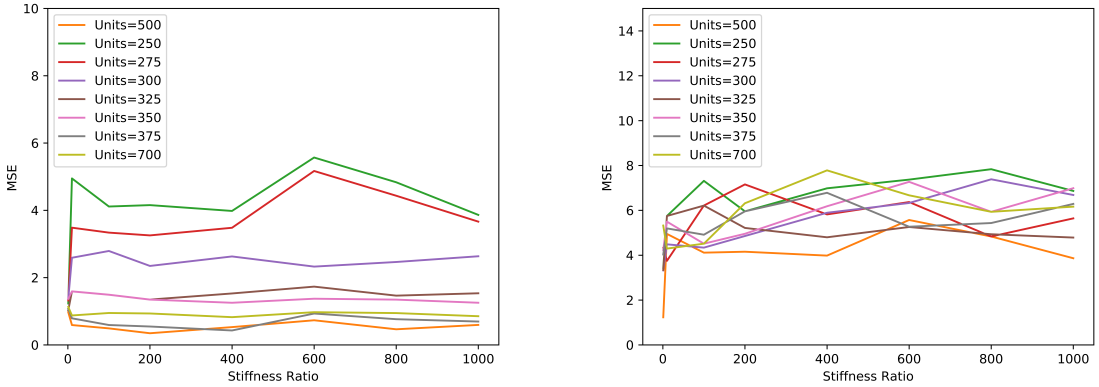


Figure 5.10: Comparison of the losses for the various number of units in the hidden layer for the case of STEER with $b = 0.124$ and a stanard Neural ODE across varying stiffness ratios.

interesting to note that the transition from $std = 0.03$ to $std = 0.04$ is rather abrupt and shows an intermediate behavior between the smaller and larger values of std.

As we observe from Fig. 5.9 we see the effective behavior of the best approaches with the Uniform distribution and the Gaussian distribution respectively compared alongside a standard Neural ODE. This plot indicates that the high stochasticity in the cases of the Uniform and Gaussian distributions leads to lower losses in terms of the MSE. We observe a slight advantage of using Uniform distribution rather than the Gaussian distribution.

Although we observe in this case that the Gaussian distribution leads to similar behavior as the Uniform distribution, it comes along with its own implementation challenges. We observe from Fig. 5.9 that better performance is obtained when std is high. On the flipside when std is high, there might be some sampled values of t which might be less than the initial time t_0. To avoid such scenarios, we would have to employ clipping on one side. Clipping on only one side would skew the resulting distribution. Clipping on both sides would add another parameter $clip$. It would decide how much to clip on either side of t_1. To simplify the technique and reducing the number of hyperparameters we chose to use the Uniform distribution. While Gaussian distribution with intelligent clipping could be a viable alternative we leave its analysis for future work.

Effect of varying the network capacity: To complete our ablation study, we also compare the effect of varying the number of units in the hidden layer of the network. In case of STEER regularization, the reduction of the number of units generally hurts as shown in

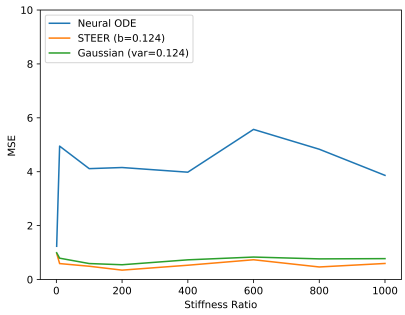


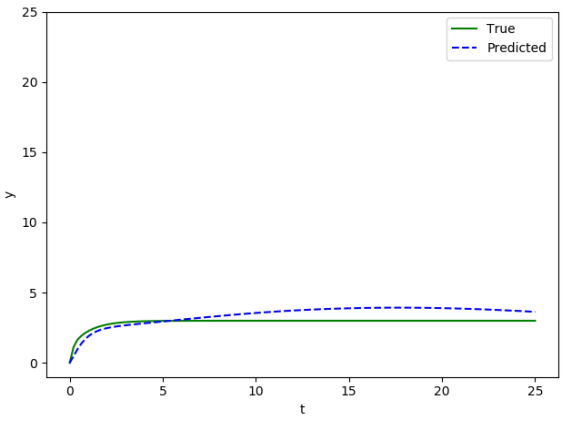
Figure 5.11: Comparison of a standard Neural ODE along with the Gaussian and Uniform distributions from which the end time t_1 can be sampled.

Fig. 5.10. The worst performance is obtained with 250 hidden units. The performance in terms of MSE keeps getting better by increasing the number of hidden units till 500. Increasing further to 700 leads to similar behavior but slight reduction in performance. In case of a standard Neural ODE on the other hand, the behavior is quite erratic as seen in Fig. 5.10 . Increasing the number of units in the hidden layer doesn't consistently decrease the MSE error.

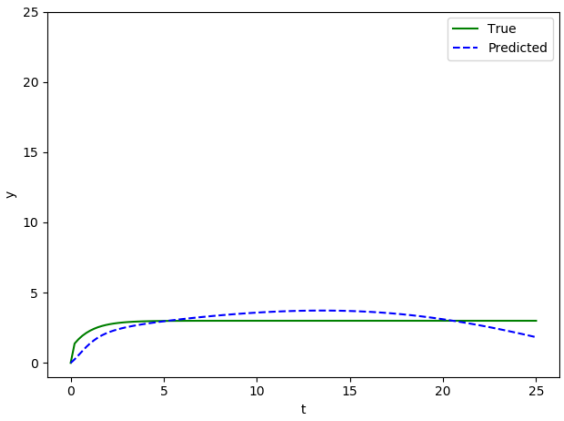
Other forms and failure cases: We analyze other instances of the equation to delve deeper into the effective behavior. When additional terms in form of e^{-kt} are added to the differential equation, our proposed STEER regularization is able to reasonably reach steady state solutions as shown in Fig. 5.12. We analyze some failure cases in Fig. 5.13. We observe that when a $sin(t)$ term is added our proposed STEER regularization struggles to fit the periodic behavior. Smaller changes such as changing the steady state from $y = 3$ to $y = 7$, causes our regularization to fail to reach the steady state solution. This indicates that stiff ODEs need further analysis. Alternative techniques or regularizations may be required to deal with harder instances of stiff ODEs.

5.8 Generative Models: Ablation Studies

We analyze the effect of varying b in case of continuous normalizing flow based generative models. We conduct ablation studies on the setting of RNODE [43] since it converges much faster. Multiple possible values of b could be experimented, since convergence is fast. As we observe from Table 5.6 there is a general trend which is similar to the one we observed in the case of stiff ODEs. The greater the stochasticity due to a larger value of b, the faster the convergence time. The best results were obtained by using $b = 0.5$. In case

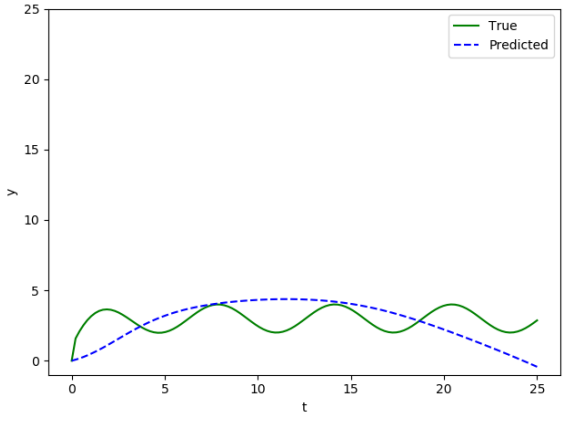


$\frac{dy}{dt} = -1000y + 3000 - 2000(e^{-t} + e^{-10t})$

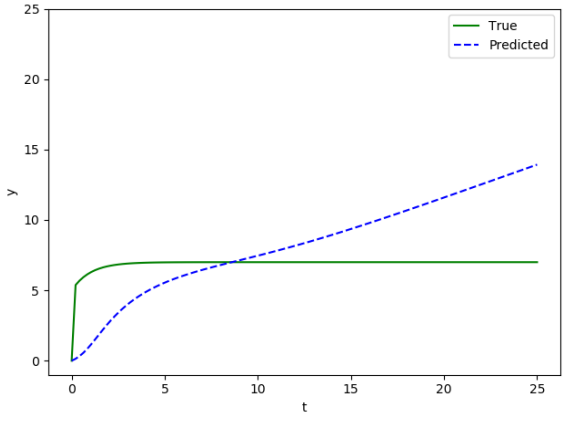


$\frac{dy}{dt} = -1000y + 3000 - 2000(e^{-t} + e^{-10000t})$

Figure 5.12: We experiment with 2 cases which have multiple e^{-kt} in the $\frac{dy}{dt}$ function to be estimated. STEER regularization doesn't totally fail in these scenarios although the performance reduces.



$\frac{dy}{dt} = -1000y + 3000 - 2000e^{-t} + 1000sin(t)$



$\frac{dy}{dt} = -1000y + 7000 - 2000e^{-t}$

Figure 5.13: Failure cases of the proposed STEER regularization. Adding a periodic term to the equation $\frac{dy}{dt}$ makes it harder to learn. Changing the behavior of the steady state from $y = 3$ to $y = 7$ causes STEER to not reach a viable steady state solution. Note that a standard Neural ODE also fails in these cases

	MNIST	
	BITS/DIM	TIME
$t_1 : t_1 + b = t_1^{original} = 1$		
FFJORD RNODE + STEER ($b = 0.5$)	0.971	16.32
FFJORD RNODE + STEER ($b = 0.375$)	0.973	17.13
FFJORD RNODE + STEER ($b = 0.25$)	0.972	19.71
FFJORD RNODE + STEER ($b = 0.125$)	0.973	22.32
$t_1 : t_1 = t_1^{original} = 1$		
FFJORD RNODE + STEER ($b = 0.5$)	0.974	25.81
FFJORD RNODE + STEER ($b = 0.375$)	0.98	25.72
FFJORD RNODE + STEER ($b = 0.25$)	0.971	25.23
FFJORD RNODE + STEER ($b = 0.125$)	0.976	24.32

Table 5.6: Comparison of b in case of Continuous Normalizing Flows. Greater values of b lead to faster convergence. Altering t_1 such that $t_1 + b = t_1^{original}$ leads to faster convergence times.

of generative modeling, to obtain faster convergence, the end time t_1 had to be constrained such that $t_1 + b = t_1^{original}$. The original ending time was $t_1^{original} = 1$ for the experiments in Table 5.6. The bottom half of Table 5.6 demonstrates that if the t_1 is not altered i.e. $t_1 = t_1^{original}$, faster convergence is not observed.

6
Discussion & Future Work

In Chapter 3, we presented a very simple and effective framework, Multi-Agent Diverse GAN (MAD-GAN), for generating diverse and meaningful samples. We showed the efficacy of our approach and compared it with various variants of GAN that it captures diverse modes while producing high quality samples. We presented a theoretical analysis of MAD-GAN with conditions for global optimality. Looking forward, an interesting future direction would be to estimate a priori the number of generators needed for a particular dataset. It is not clear how to do that given that we do not have access to the true data distribution. In addition, we would also like to theoretically understand the limiting cases that depend on the relationship between the number of generators and the complexity of the data distribution. Another interesting direction would be to exploit different generators such that their combinations can be used to capture diverse modes.

In Chapter 4, we present a two-stage approach for interactive object generation, centered around the idea of a shape completion intermediary. This step both makes training more stable and also allows us to give coarse geometric feedback to the user, which they can choose to integrate as they desire. Although the results are attractive at first glance, it only works with images and sketches that are aligned. Deep face drawing [15] showed much better results with a similar approach but highly engineered towards optimizing for face images. Such high quality results should be available for other use cases such as designing cars and apparel. In the future if such a technique was to be deployed into production for the future of design it has to be much more versatile to be adaptible to new classes. A

crucial requirement for the model to work in the wild would be to automatically detect keypoints such that the orientaion in which the user wants to draw is obvious. Once the rough autocompleted image has been generated a StyleGAN [72] like backend could help in enhancing the details of the generated image.

In Appendix C via the DGPose paper, we have presented a comprehensive deep generative model framework for human pose analysis in images. Our models are based on a principled VAEGAN approach and allow the disentanglement of body posture and visual appearance, aiming for the independent manipulation of such factors. With our conditional-VAEGAN model, the Conditional-DGPose, differently from previous art, we have taken such manipulation to extreme cases, For exampleby performing cross-domain *pose-transfer* and by hallucinating multiple people in a variety of unseen or even unrealistic poses. Moreover, we have achieved state-of-the-art results on image reconstruction conditioned on pose, *outperforming* the closest related comparable baseline [87]. With a single-stage structured semi-supervised VAEGAN architecture, the Semi-DGPose, we pursued the joint *understanding* and *generation* of people in images, not only mapping images to partially interpretable latent representations but also mapping these representations back to the image space. Importantly, such an approach simultaneously allows for reconstruction, direct manipulation, sampling, pose estimation, indirect pose-transfer, and semi-supervised learning. These joint capabilities differentiate the Semi-DGPose from other methods in the literature and demonstrate a real-world application of structured deep generative models with the highly relevant potential of being less dependable of fully-labelled data. We have systematically evaluated our methods on well-known benchmarks, the Human3.6M, the ChictopiaPlus, and the DeepFashion datasets, comparing our results with the closest related baseline methods in the literature [87, 98]. Such results and comparisons highlight the novelty and effectiveness of our approaches and its capabilities, despite the significant challenge posed by our aimed goal. We believe that we have shown and reinforced the relevance of employing an interpretable and structured latent space, which allows for semi-supervised learning, as well as the importance of tackling the problem with single-stage end-to-end architectures.

In Chapter 5, we introduced a simple temporal regularization method for Neural ODEs. The regularization consists of perturbing the final time of the integration of the ODE solver. The main advantage of our work is the empirical evidence showing the regularization results in simpler dynamics for Neural ODEs models on a wide variety of tasks. This is demonstrated for a specific stiff ODE that is challenging for a basic Neural ODE solver, faster continuous normalizing flow models and improved or comparable accuracy for time

series models and deep feedforward networks. We further show that our approach is orthogonal to other regularization approaches and can be thus combined with them to further improve Neural ODEs.

Neural ODEs operate under the broad umbrella of 'Implicit Methods in Deep Learning'. The last decade in deep learning was characterized by layer based deep neural networks. The limits of that framework are starting to approach whereby one can fit only so many layers in the GPU of a computer. Thus, alternate forms of computational frameworks are emerging whereby the function is not explicitly modeled. Rather a pseudo function is learnt from which the original function that needed to be approximated can be retrieved. This shows a different model of computation and previous work has shown signs that it is much more parameter efficient than resnet based deep neural networks.

Our work is in the broad area of Neural ODEs which provides continuous depth neural networks. This line of work would have a similar impact as most supervised deep learning techniques. However,current continuous depth neural networks show limitations that are improved through our proposed regularization scheme. Thus, our models converge faster during training by reducing the number of function evaluations and effective training time. The resulting trained models also require fewer function evaluations at inference, thus reducing the computation at test time too. Our research could potentially encourage the search for more such efficient techniques for training neural ODEs.

Although we have highlighted some drawbacks of existing Neural ODEs in the context of Stiff ODEs. We do not fully understand why our solution works better in the case of stiff ODEs. The issue of stiff ODEs has a rich history of research in engineering. Inevitably during training, there is a possibility that the dynamics learnt by the neural network might be an instance of a stiff equation. This cannot be prevented with the current formulation. As Neural ODEs become more stable and mature we need to understand the effects of stiffness. These models can be truly useful if we understand it from all possible points of failure.

A
MADGAN Paper

Contributions:

- Arnab Ghosh and Viveka Kulharia came up with the ideas and conducted the experiments
- Vinay Namboodiri, Philip Torr and Puneet Dokania supervised the project and helped in writing of the manuscript.

This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

Multi-Agent Diverse Generative Adversarial Networks

Arnab Ghosh*
University of Oxford, UK
arnabg@robots.ox.ac.uk

Viveka Kulharia*
University of Oxford, UK
viveka@robots.ox.ac.uk

Vinay Namboodiri
IIT Kanpur, India
vinaypn@iitk.ac.in

Philip H.S. Torr
University of Oxford, UK
philip.torr@eng.ox.ac.uk

Puneet K. Dokania
University of Oxford, UK
puneet@robots.ox.ac.uk

Abstract

We propose MAD-GAN, an intuitive generalization to the Generative Adversarial Networks (GANs) and its conditional variants *to address the well known problem of* mode collapse. *First, MAD-GAN is a multi-agent GAN architecture incorporating multiple generators and one discriminator. Second, to enforce that different generators capture diverse high probability modes, the discriminator of MAD-GAN is designed such that along with finding the real and fake samples, it is also required to identify the generator that generated the given fake sample. Intuitively, to succeed in this task, the discriminator must learn to push different generators towards different identifiable modes. We perform extensive experiments on synthetic and real datasets and compare MAD-GAN with different variants of GAN. We show high quality diverse sample generations for challenging tasks such as image-to-image translation and face generation. In addition, we also show that MAD-GAN is able to disentangle different modalities when trained using highly challenging diverse-class dataset (*e.g. *dataset with images of forests, icebergs, and bedrooms). In the end, we show its efficacy on the unsupervised feature representation task.*

1. Introduction

Generative models have attracted considerable attention recently. The underlying idea behind such models is to attempt to capture the distribution of high-dimensional data such as images and texts. Though these models are highly useful in various applications, it is computationally expensive to train them as they require intractable integration in a very high-dimensional space. This drastically limits their applicability. However, recently there has been considerable progress in *deep generative models* – conglomerate of deep neural networks and generative models – as they do not explicitly require the intractable integration, and can be efficiently trained using back-propagation algorithm. Two such famous examples are Generative Adversarial Networks (GANs) [11] and Variational Autoencoders [14].



Figure 1: Diverse-class data generation using MAD-GAN. Diverse-class dataset contains images from different classes/modalities (in this case, forests, icebergs, and bedrooms). Each row represents generations by a particular generator and each column represents generations for a given random noise input z. As shown, once trained using this dataset, generators of MAD-GAN are able to disentangle different modalities, hence, each generator is able to generate images from a particular modality.

In this paper we focus on GANs as they are known to produce sharp and plausible images. Briefly, GANs employ a generator and a discriminator where both are involved in a minimax game. The task of the discriminator is to learn the difference between *real* samples (from true data distribution p_d) and *fake* samples (from generator distribution p_g). Whereas, the task of the generator is to maximize the mistakes of the discriminator. At convergence, the generator learns to produce real looking images. A few successful applications of GANs are video generation [26], image inpainting [22], image manipulation [29], 3D object generation [27], interactive image generation using few brush strokes [29], image super-resolution [17], diagrammatic abstract reasoning [15] and conditional GANs [20, 24].

Despite the remarkable success of GAN, it suffers from the major problem of *mode collapse* [2, 7, 8, 19, 25].

*Joint first author

Though, theoretically, convergence guarantees the generator learning the true data distribution. However, practically, reaching the true equilibrium is difficult and not guaranteed, which potentially leads to the aforementioned problem of mode collapse. Broadly speaking, there are two schools of thought to address the issue: (1) improving the learning of GANs to reach better optima [2, 19, 25]; and (2) explicitly enforcing GANs to capture diverse modes [7, 8, 18]. Here we focus on the latter.

Borrowing from the multi-agent algorithm [1] and coupled GAN [18], we propose to use multiple generators with one discriminator. We call this framework the Multi-Agent GAN architecture, as shown in Fig. 2. In detail, similar to the standard GAN, the objective of each generator here is to maximize the mistakes of the *common* discriminator. Depending on the task, it might be useful for different generators to share information. This is done using the initial layer parameters of generators. Another reason behind sharing these parameters is the fact that initial layers capture low-frequency structures which are almost the same for a particular type of dataset (for example, faces), therefore, sharing them reduces redundant computations. However, when the dataset contains images from completely different modalities, one can avoid sharing these parameters. Naively using multiple generators may lead to the *trivial solution* where all the generators learn to generate *similar* samples. To resolve this issue and generate different visually plausible samples capturing diverse high probability modes, we propose to modify the objective function of the discriminator. In the modified objective, along with finding the real and the fake samples, the discriminator also has to correctly identify the generator that generated the given fake sample. Intuitively, in order to succeed in this task, the discriminator must learn to push generations corresponding to different generators towards different identifiable modes. Combining the Multi-Agent GAN architecture with the diversity enforcing term allows us to generate diverse plausible samples, thus the name Multi-Agent Diverse GAN (MAD-GAN).

As an example, an intuitive setting where mode collapse occurs is when a GAN is trained on a dataset containing images from different modalities/classes. For example, a diverse-class dataset containing images such as forests, iceberg, and bedrooms. This is of particular interest as it not only requires the model to disentangle intra-class variations, it also requires inter-class disentanglement. Fig. 1 demonstrates the surprising effectiveness of MAD-GAN in this challenging setting. Generators among themselves are able to disentangle inter-class variations, and each generator is also able to capture intra-class variations.

In addition, we analyze MAD-GAN through extensive experiments and compare it with several variants of GAN. First, for the proof of concept, we perform experiments in controlled settings using synthetic dataset (mixture of Gaussians), and complicated Stacked/Compositional MNIST datasets with hand engineered modes. In these settings, we empirically show that our approach outperforms all other GAN variants we compare with, and is able to generate high quality samples while capturing large number of modes. In a more realistic setting, we show high quality diverse sample generations for the challenging tasks of *image-to-image translation* [12] (conditional GAN) and *face generation* [8, 23]. Using the SVHN dataset [21], we also show the efficacy of our framework for learning the feature representation in an unsupervised setting.

We also provide theoretical analysis of this approach and show that the proposed modification in the objective of discriminator allows generators to learn together as a mixture model where each generator represents a mixture component. We show that at convergence, the global optimum value of $-(k+1)\log(k+1) + k\log k$ is achieved, where k is the number of generators.

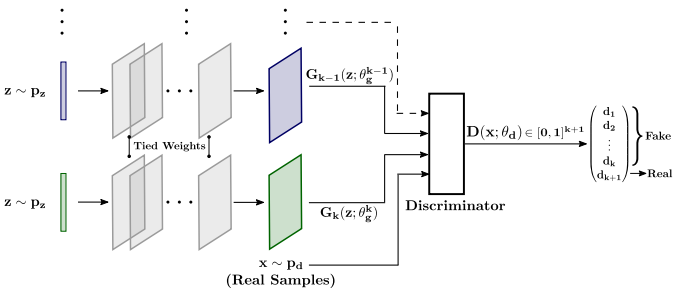


Figure 2: Multi-Agent Diverse GAN (MAD-GAN). The discriminator outputs $k+1$ softmax scores signifying the probability of its input sample being from either one of the k generators or the real distribution.

2. Related Work

The recent work called InfoGAN [8] proposed an information-theoretic extension to GANs in order to address the problem of mode collapse. Briefly, InfoGAN disentangles the latent representation by assuming a factored representation of the latent variables. In order to enforce that the generator learns factor specific generations, InfoGAN maximizes the mutual information between the factored latents and the generator distribution. Che *et al*. [7] proposed a mode regularized GAN (ModeGAN) which uses an encoder-decoder paradigm. The basic idea behind ModeGAN is that if a sample from the true data distribution p_d belongs to a particular mode, then the sample generated by the generator (fake sample) when the true sample is passed through the encoder-decoder is likely to belong to the same mode. ModeGAN assumes that there exists enough true samples from a mode for the generator to be able to capture it. Another work by Metz *et al*. [19] proposed a surrogate objective for the update of the generator with respect to the

unrolled optimization of the discriminator (UnrolledGAN) to address the issue of convergence of the training process of GANs. This improves the training process of the generator which in turn allow the generators to explore better coverage to true data distribution.

Liu *et al*. [18] presented Coupled GAN, a method for training two generators with shared parameters to learn the joint distribution of the data. The shared parameters guide both the generators towards similar subspaces but since they are trained independently on two domains, they promote diverse generations. Durugkar *et al*. [9] proposed a model with multiple discriminators whereby an ensemble of multiple discriminators have been shown to stabilize the training of the generator by guiding it to produce better samples.

W-GAN [3] is a recent technique which employs integral probability metrics based on the earth mover distance rather than the JS-divergences that the original GAN uses. BE-GAN [5] builds upon W-GAN using an autoencoder based equilibrium enforcing technique alongside the Wasserstein distance. DCGAN [23] was a seminal technique which used a fully convolutional generator and discriminator for the first time along with the introduction of batch normalization thus stabilizing the training procedure, and was able to generate compelling generations. GoGAN [13] introduced a training procedure for the training of the discriminator using a maximum margin formulation alongside the earth mover distance based on the Wasserstein-1 metric. [4] introduced a technique and theoretical formulation stating the importance of multiple generators and discriminators in order to completely model the data distribution. In terms of employing multiple generators, our work is closest to [4, 18, 10]. However, while using multiple generators, our method explicitly enforces them to capture diverse modes.

3. Preliminaries

Here we present a brief review of GANs [11]. Given a set of samples $\mathcal{D} = (x_i)_{i=1}^n$ from the true data distribution p_d, the GAN learning problem is to obtain the optimal parameters θ_g of a generator $G(z;\theta_g)$ that can sample from an approximate data distribution p_g, where $z \sim p_z$ is the prior input noise (*e.g*. samples from a normal distribution). In order to learn the optimal θ_g, the GAN objective (Eq. (1)) employs a discriminator $D(x;\theta_d)$ that learns to differentiate between a *real* (from p_d) and a *fake* (from p_g) sample x. The overall GAN objective is:

$$\min_{\theta_g}\max_{\theta_d} V(\theta_d,\theta_g) := \mathbb{E}_{x\sim p_d}\log D(x;\theta_d) + \mathbb{E}_{z\sim p_z}\log\big(1-D(G(z;\theta_g);\theta_d)\big) \quad (1)$$

The above objective is optimized in a block-wise manner where θ_d and θ_g are optimized one at a time while fixing the other. For a given sample x (either from p_d or p_g) and the parameter θ_d, the function $D(x;\theta_d) \in [0,1]$ produces a score that represents the probability of x belonging to the true data distribution p_d (or probability of it being real). The objective of the discriminator is to learn parameters θ_d that maximizes this score for the true samples (from p_d) while minimizing it for the fake ones $\tilde{x} = D(z;\theta_g)$ (from p_g). In the case of generator, the objective is to minimize $\mathbb{E}_{z\sim p_z}\log\big(1-D(G(z;\theta_g);\theta_d)\big)$, equivalently maximize $\mathbb{E}_{z\sim p_z}\log D(G(z;\theta_g);\theta_d)$. Thus, the generator learns to maximize the scores for the fake samples (from p_g), which is exactly the opposite to what discriminator is trying to achieve. In this manner, the generator and the discriminator are involved in a minimax game where the task of the generator is to maximize the mistakes of the discriminator. Theoretically, at equilibrium, the generator learns to generate real samples, which means $p_g = p_d$.

4. Multi-Agent Diverse GAN

In the GAN objective, one can argue that the task of a generator is much harder than that of the discriminator as it has to produce real looking images to maximize the mistakes of the discriminator. This, along with the minimax nature of the objective raise several challenges for GANs [2, 7, 8, 19, 25]: (1) mode collapse; (2) difficult optimization; and (3) trivial solution. In this work we propose a new framework to address the first challenge of *mode collapse* by increasing the capacity of the generator while using well known tricks to partially avoid other challenges [2].

Briefly, we propose a *Multi-Agent GAN architecture* that employs multiple generators and one discriminator in order to generate different samples from high probability regions of the true data distribution. In addition, theoretically, we show that our formulation allows generators to act as a mixture model with each generator capturing one component.

4.1. Multi-Agent GAN Architecture

Here we describe our proposed architecture (Fig. 2). It involves k generators and one discriminator. In the case of homogeneous data (all the images belong to same class, *e.g*. faces or birds), we allow all the generators to share information by tying most of the initial layer parameters. This is essential to avoid redundant computations as initial layers of a generator capture low-frequency structures which are almost the same for a particular type of dataset. This also allows different generators to converge faster. However, in the case of *diverse-class* data (*e.g*. dataset with a mixture of different classes such as forests, icebergs *etc*.), it is necessary to avoid sharing these parameters to allow each generator to capture content specific structures. Thus, the extent to which one should share these parameters depends on the task at hand.

More specifically, given $z \sim p_z$ for the i-th generator, similar to the standard GAN, the first step involves gen-

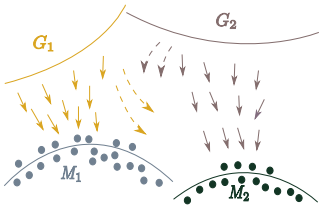


Figure 3: Visualization of different generators getting pushed towards different modes. Here, M_1 and M_2 could be a cluster of modes where each cluster itself contains many modes. The arrows abstractly represent generator specific gradients for the purpose of building intuition.

erating a sample (for example, an image) $\tilde{x}_i$. Since each generator receives the same latent input sampled from the same distribution, naively using this simple approach may lead to the *trivial solution* where all the generators learn to generate similar samples. In what follows, we propose an intuitive solution to avoid this issue and allow the generators to capture diverse modes.

4.2. Enforcing Diverse Modes

Inspired by the discriminator formulation for the semi-supervised learning [25], we use a generator identification based objective function that, along with minimizing the score $D(\tilde{x};\theta_d)$, requires the discriminator to identify the generator that generated the given fake sample $\tilde{x}$. In order to do so, as opposed to the standard GAN objective function where the discriminator outputs a scalar value, we modify it to output $k+1$ soft-max scores. In more detail, given the set of k generators, the discriminator produces a soft-max probability distribution over $k+1$ classes. The score at $(k+1)$-th index $(D_{k+1}(.))$ represents the probability that the sample belongs to the true data distribution and the score at $j \in \{1,\ldots,k\}$-th index represents the probability of it being generated by the j-th generator. Under this setting, while learning θ_d, we optimize the cross-entropy between the soft-max output of the discriminator and the Dirac delta distribution $\delta \in \{0,1\}^{k+1}$, where for $j \in \{1,\ldots,k\}$, $\delta(j)=1$ if the sample belongs to the j-th generator, otherwise $\delta(k+1)=1$. Thus, the objective of the discriminator, which is optimizing θ_d while keeping θ_g constant (refer Eq. (1)), is modified to:

$$\max_{\theta_d} \mathbb{E}_{x\sim p}\, H(\delta, D(x;\theta_d))$$

where, $Supp(p) = \cup_{i=1}^{k} Supp(p_{g_i}) \cup Supp(p_d)$ and $H(.,.)$ is the negative of the cross entropy function. Intuitively, in order to correctly identify the generator that produced a given fake sample, the discriminator must learn to push different generators towards different identifiable modes. However, the objective of each generator remains the same as in the standard GAN. Thus, for the i-th generator, the objective is to minimize the following:

$$\mathbb{E}_{x\sim p_d} \log D_{k+1}(x;\theta_d) + \mathbb{E}_{z\sim p_z} \log(1 - D_{k+1}(G_i(z;\theta_g^i);\theta_d))$$

To update the parameters, the gradient for each generator is simply computed as $\nabla_{\theta_g^i} \log(1 - D_{k+1}(G_i(z;\theta_g^i);\theta_d)))$. Notice that all the generators in this case can be updated in parallel. For the discriminator, given $x \sim p$ (can be real or fake) and corresponding δ, the gradient is $\nabla_{\theta_d} \log D_j(x;\theta_d)$, where $D_j(x;\theta_d)$ is the j-th index of $D(x;\theta_d)$ for which $\delta(j)=1$. Therefore, using this approach requires *very minor modifications to the standard GAN optimization algorithm* and can be easily used with different variants of GAN. An intuitive visualization is shown in Fig. 3.

Theorem 1 shows that the above objective function actually allows generators to form a mixture model where each generator represents a mixture component and the global optimum of $-(k+1)\log(k+1) + k\log k$ is achieved when $p_d = \frac{1}{k}\sum_{i=1}^{k} p_{g_i}$. Notice that, at $k=1$, which is the case with one generator, we obtain exactly the same Jensen-Shannon divergence based objective function as shown in [11] with the optimal value of $-\log 4$.

Theorem 1. *Given the optimal discriminator, the objective for training the generators boils down to minimizing*

$$KL\Big(p_d(x)||p_{avg}(x)\Big) + kKL\Big(\frac{1}{k}\sum_{i=1}^{k} p_{g_i}(x)||p_{avg}(x)\Big) - (k+1)\log(k+1) + k\log k \tag{2}$$

where, $p_{avg}(x) = \frac{p_d(x)+\sum_{i=1}^{k} p_{g_i}(x)}{k+1}$. *The above objective function obtains its global minimum if* $p_d = \frac{1}{k}\sum_{i=1}^{k} p_{g_i}$ *with the objective value of* $-(k+1)\log(k+1) + k\log k$.

Proof. The joint objective of all the generators is to minimize the following:

$$\mathbb{E}_{x\sim p_d} \log D_{k+1}(x) + \sum_{i=1}^{k} \mathbb{E}_{x\sim p_{g_i}} \log(1 - D_{k+1}(x))$$

Using Corollary 1, we substitute the optimal discriminator in the above equation and obtain:

$$\begin{aligned}
&\mathbb{E}_{x\sim p_d} \log\left[\frac{p_d(x)}{p_d(x)+\sum_{i=1}^{k} p_{g_i}(x)}\right] + \\
&\sum_{i=1}^{k} \mathbb{E}_{x\sim p_{g_i}} \log\left[\frac{\sum_{i=1}^{k} p_{g_i}(x)}{p_d(x)+\sum_{i=1}^{k} p_{g_i}(x)}\right] \\
&= \mathbb{E}_{x\sim p_d} \log\left[\frac{p_d(x)}{p_{avg}(x)}\right] + k\mathbb{E}_{x\sim p_g} \log\left[\frac{p_g(x)}{p_{avg}(x)}\right] \\
&- (k+1)\log(k+1) + k\log k
\end{aligned} \tag{3}$$

where, $p_g = \frac{\sum_{i=1}^{k} p_{g_i}}{k}$ and $p_{avg}(x) = \frac{p_d(x)+\sum_{i=1}^{k} p_{g_i}(x)}{k+1}$. Note that, Eq. (3) is exactly the same as Eq. (2). When $p_d = \frac{\sum_{i=1}^{k} p_{g_i}}{k}$, both the KL terms become zero and the global minimum is achieved. □

Corollary 1. *For fixed generators, the optimal distribution learned by the discriminator D has the following form:*

$$D_{k+1}(x) = \frac{p_d(x)}{p_d(x) + \sum_{i=1}^{k} p_{g_i}(x)},$$
$$D_i(x) = \frac{p_{g_i}(x)}{p_d(x) + \sum_{i=1}^{k} p_{g_i}(x)}, \forall i \in \{1, \cdots, k\}.$$

where, $D_i(x)$ represents the i-th index of $D(x; \theta_d)$, p_d the true data distribution, and p_{g_i} the distribution learned by the i-th generator.

Proof. For fixed generators, the objective function of the discriminator is to maximize

$$\mathbb{E}_{x \sim p_d} \log D_{k+1}(x) + \sum_{i=1}^{k} \mathbb{E}_{x_i \sim p_{g_i}} \log D_i(x_i)$$

where, $\sum_{i=1}^{k+1} D_i(x) = 1$ and $D_i(x) \in [0, 1], \forall i$. The above equation can be written as:

$$\int_x p_d(x) \log D_{k+1}(x) dx + \sum_{i=1}^{k} \int_x p_{g_i}(x) \log D_i(x) dx$$
$$= \int_{x \in p} \sum_{i=1}^{k+1} p_i(x) \log D_i(x) dx \quad (4)$$

where, $p_{k+1}(x) := p_d(x)$, $p_i(x) := p_{g_i}(x), \forall i \in \{1, \cdots, k\}$, and $Supp(p) = \bigcup_{i=1}^{k} Supp(p_{g_i}) \bigcup Supp(p_d)$, Therefore, for a given x, the optimum of objective function defined in Eq. (4) with constraints defined above can be obtained using Proposition 1. □

Proposition 1. *Given $\mathbf{y} = (y_1, \cdots, y_n)$, $y_i \geq 0$, and $a_i \in \mathbb{R}$, the optimal solution for the objective function defined below is achieved at $y_i^* = \frac{a_i}{\sum_{i=1}^{n} a_i}, \forall i$*

$$\max_{\mathbf{y}} \sum_{i=1}^{n} a_i \log y_i, \quad s.t. \sum_{i}^{n} y_i = 1$$

Proof. The Lagrangian of the above problem is:

$$L(\mathbf{y}, \lambda) = \sum_{i=1}^{n} a_i \log y_i + \lambda(\sum_{i=1}^{n} y_i - 1)$$

Differentiating w.r.t y_i and λ, and equating to zero,

$$\frac{a_i}{y_i} + \lambda = 0 \ , \ \sum_{i=1}^{n} y_i - 1 = 0$$

Solving the above two equations, we obtain $y_i^* = \frac{a_i}{\sum_{i=1}^{n} a_i}$. □

GAN Variants	Chi-square($\times 10^5$)	KL-Div
DCGAN [23]	0.90	0.322
WGAN [3]	1.32	0.614
BEGAN [5]	1.06	0.944
GoGAN [13]	2.52	0.652
Unrolled GAN [19]	3.98	1.321
Mode-Reg DCGAN [7]	1.02	0.927
InfoGAN [8]	0.83	0.21
MA-GAN	1.39	0.526
MAD-GAN (Our)	**0.24**	**0.145**

Table 1: Synthetic experiment on 1D GMM (Fig. 4).

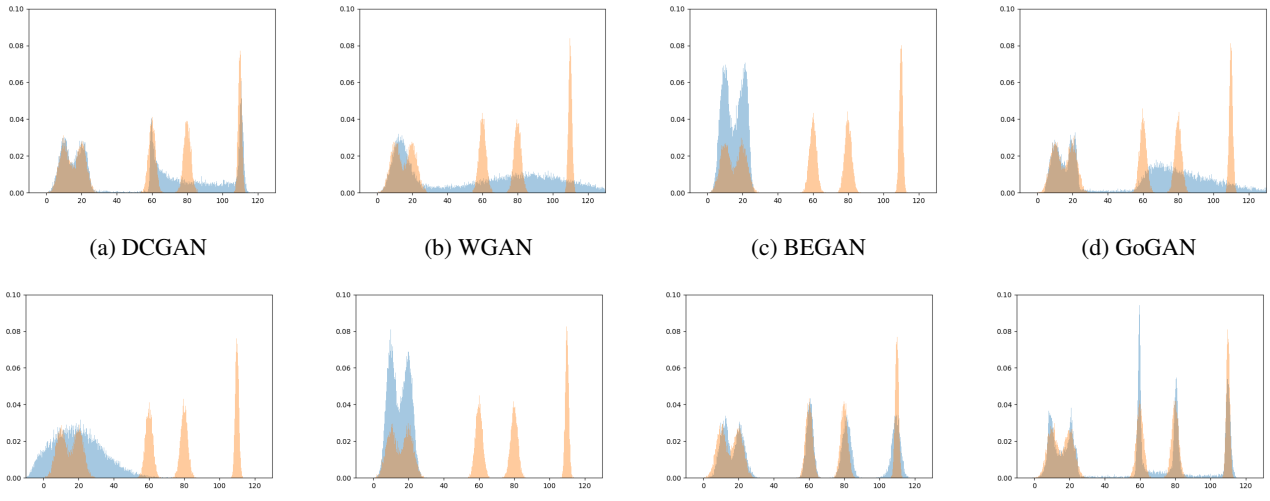
5. Experiments

We present an extensive quantitative and qualitative analysis of MAD-GAN on various synthetic and real-world datasets. First, we use a simple 1D mixture of Gaussians and also Stacked/Compositional MNIST dataset (1000 modes) to compare MAD-GAN with several known variants of GANs, such as DCGAN [23], WGAN [3], BEGAN [5], GoGAN [13], Unrolled GAN [19], Mode-Reg GAN [7] and InfoGAN [8]. Furthermore, we created another baseline, called *MA-GAN (Multi-Agent GAN)*, which is a trivial extension of GAN with multiple generators and one discriminator. As opposed to MAD-GAN, MA-GAN has a simple Multi-Agent architecture without modifications to the objective of the discriminator. This comparison allows us to understand the effect of explicitly enforcing diversity in the objective of the MAD-GAN. We use KL-divergence [16] and number of modes recovered [7] as the criterion for comparisons and show superior results compared to all the other methods. Additionally, we show diverse generations for the challenging tasks of *image-to-image translation* [12], *diverse-class data generation*, and face generation. It is non-trivial to devise a metric to evaluate diversity on these high quality generation tasks, so we perform qualitative assessment. Note that, the image-to-image translation objective is known to learn the delta distribution, thus, it is agnostic to the input noise vector. However, we show that MAD-GAN is able to produce highly plausible diverse generations for this task. In the end, we show the efficacy of MAD-GAN in unsupervised feature representation learning task. We provide detailed overview of the architectures, datasets, and the parameters used in our experiments in the supplementary.

In the case of InfoGAN [8], we varied the dimension of the categorical variable, depicting the number of modes, to obtain the best cross-validated results.

5.1. Non-Parametric Density Estimation

In order to understand the behavior of MAD-GAN and different state-of-the-art GAN models, we first perform a



(a) DCGAN (b) WGAN (c) BEGAN (d) GoGAN

(e) Unrolled GAN (f) Mode-Reg DCGAN (g) InfoGAN (h) MAD-GAN (Our)

Figure 4: A toy example to understand the behaviour of different GAN variants in order to compare with MAD-GAN (each method was trained for 198000 iterations). The orange bars show the density estimate of the training data and the blue ones for the generated data points. After careful cross-validation, we chose the bin size of 0.1.

very simple synthetic experiment, much easier than generating high-dimensional complex images. We consider a distribution of 1D GMM [6] having five mixture components with modes at 10, 20, 60, 80 and 110, and standard deviations of 3, 3, 2, 2 and 1, respectively. While the first two modes overlap significantly, the fifth mode stands isolated as shown in Fig. 4. We train different GAN models using $200,000$ samples from this distribution and generate $65,536$ data points from each model. In order to compare the learned distribution with the ground truth distributions, we first estimate them using bins over the data points and create the histograms. These histograms are carefully created using different bin sizes and the best bin (found to be 0.1) is chosen. Then, we use Chi-square distance and the KL-divergence to compute distance between the two histograms. From Fig. 4 and Tab. 1 it is evident that MAD-GAN is able to capture all the clustered modes which includes significantly overlapped modes as well. MAD-GAN obtains the minimum value in terms of both Chi-square distance and the KL-divergence. In this experiment, both MAD-GAN and MA-GAN used four generators. In the case of InfoGAN, we used 5 dimensional categorical variable, which provides the best result.

5.2. Stacked and Compositional MNIST

We now perform experiments on a more challenging setup, similar to [7, 19], in order to examine and compare MAD-GAN with other GAN variants. [19] created a Stacked-MNIST dataset with $25,600$ samples where each sample has three channels stacked together with a random MNIST digit in each of them. Thus, it creates 1000 distinct

GAN Variants	KL Div	# Modes Covered
DCGAN [23]	2.15	712
WGAN [3]	1.02	868
BEGAN [5]	1.89	819
GoGAN [13]	2.89	672
Unrolled GAN [19]	1.29	842
Mode-Reg DCGAN [7]	1.79	827
InfoGAN [8]	2.75	840
MA-GAN	3.4	700
MAD-GAN (Our)	**0.91**	**890**

Table 2: Stacked-MNIST experiments and comparisons. Note that three generators are used for MAD-GAN.

GAN Variants	KL Div	# Modes Covered
DCGAN [23]	0.18	980
WGAN [3]	0.25	1000
BEGAN [5]	0.19	999
GoGAN [13]	0.87	972
Unrolled GAN [19]	0.091	1000
Mode-Reg DCGAN [7]	0.12	992
InfoGAN [8]	0.47	990
MA-GAN	1.62	997
MAD-GAN (Our)	**0.074**	**1000**

Table 3: Compositional-MNIST experiments and comparisons. Note that three generators are used for MAD-GAN.



Figure 6: Diverse generations for *edges-to-handbags* generation task. In each sub-figure, the first column represents the input, columns 2-4 represents generations by MAD-GAN (using three generators), and columns 5-7 are generations by InfoGAN (using three categorical codes). It is evident that different generators of MAD-GAN are able to produce diverse results capturing different colors, textures, design patterns, among others. However, InfoGAN generations are visually almost the same, indicating mode collapse.

modes in the data distribution. [19] used a stripped down version of the generator and discriminator pair to reduce the modeling capacity. We do the same for fair comparisons and used the same architecture as mentioned in their paper. Similarly, [7] created Compositional-MNIST whereby they took 3 random MNIST digits and place them at the 3 quadrants of a 64×64 dimensional image. This also resulted in a data distribution with 1000 hand-designed modes. The distribution of the resulting generated samples was estimated using a pretrained MNIST classifier to classify each of the digits either in the channels or the quadrants to decide the mode it belongs to.

Tables 2 and 3 provide comparison of our method with variants of GAN in terms of KL divergence and the number of modes recovered for the Stacked and Compositional MNIST datasets, respectively. In Stacked-MNIST, as evident from the Tab. 2, MAD-GAN outperforms all other variants of GAN in both the criteria. Interestingly, in the case of Compositional-MNIST, as shown in Tab. 3, MAD-GAN, WGAN and Unrolled GAN were able to recover all the 1000 modes. However, in terms of KL divergence, the distribution generated by MAD-GAN is the closest to the true data distribution.

5.3. Diverse Samples for Image-to-Image Translation and Comparison to InfoGAN

Here we present experimental results on the challenging task of image-to-image translation [12] which uses conditional variant of GANs [20]. Conditional GAN for this task is known to learn the delta distribution, thus, generates the same image irrespective of the variations in the input noise vector. Generating diverse samples in this setting in itself is an open problem. We show that MAD-GAN is able to generate diverse samples in these experiments as well. We use three generators for MAD-GAN experiments and show three diverse generations. Note that, we do not claim to capture all the possible modes present in the data distribution because firstly we cannot estimate the number of modes a priori, and secondly, even if we could, we do not know how diverse the generations would be after using certain number of generators. We follow the same approach as [12] and employ patch based conditional GAN.



Figure 8: Diverse generations for *night-to-day* image generation task. First column in each sub-figure represents the input. The remaining three columns show the diverse generations of three different generators of MAD-GAN (Our).

We compare MAD-GAN with InfoGAN [8] in these experiments as it is closest to our approach and can be used in image-to-image translation task. Theoretically, latent codes in InfoGAN should enable diverse generations. However, InfoGAN can only be used when the bias introduced by the categorical variables have significant impact on the generator network. For image-to-image translation and high resolution generations, the categorical variable does not have



Figure 9: Face generations using MAD-GAN. Each generator employed is DCGAN. Each row represents a generator. Each column represents generations for a given random noise input z. Note that, the first generator is generating faces pointing to the left. The second generator is generating female faces with long hair, while the third generator generates images with light background.

sufficient impact on the generations. As will be seen shortly, we validate this hypothesis by comparing our method with InfoGAN for this task. For the InfoGAN generator, to capture three kinds of distinct modes, the categorical code is chosen to take three values. Since we are dealing with images, in this case, the categorical code is a 2D matrix in which we set one third of the entries to 1 and remaining to 0 for each category. The generator is fed input image along with categorical code appended channel wise to the image. Architecture of the Q network is same as that of the pix2pix discriminator [12], except that the output is a vector of size 3 for the prediction of the categorical codes. The discriminator and the Q network parameters are unshared. We observed that sharing them and even adding noise besides categorical code did not make any difference. Note that, we tried different variations of the categorical codes but did not observe any significant variation in the generations.

Fig. 6 shows generations by MAD-GAN and InfoGAN for the *edges-to-handbags* task, where given the edges of handbags, the objective is to generate real looking handbags. Clearly, each MAD-GAN generator is able to produce meaningful images but different from remaining generators in terms of color, texture, and patterns. However, InfoGAN generations are almost the same for all the three categorical codes. In addition, in Fig. 8, we show diverse generations for the *night-to-day* task, where given night images of places, the objective is to generate their corresponding day images. As can be seen, the generated day images in Fig. 8 differ in terms of lighting conditions, sky patterns, weather conditions, and many other minute yet useful cues.

5.4. Diverse-Class Data Generation

To further explore the mode capturing capacity of MAD-GAN, we experimented with a much more challenging task of diverse-class data generation. In detail, we trained MAD-GAN (three generators) on a combined dataset consisting of various highly diverse images such as *islets*, *icebergs*, *broadleaf-forest*, *bamboo-forest*, and *bedroom*, obtained from the Places dataset [28]. Images were randomly selected from each of them, creating a training dataset of $24,000$ images. The generators have the same architecture as that of DCGAN. In this case, as the images in the dataset belong to different classes, we did not share the generator parameters. As shown in Fig. 1, to our surprise, we found that even in this highly challenging setting, the generations from different generators belong to different classes. This clearly indicates that the generators in MAD-GAN are able to disentangle inter-class variations. In addition, each generator for different noise input is able to generate diverse samples, indicating intra-class diversity.

5.5. Diverse Face Generation

Here we show diverse face generations (CelebA dataset) using MAD-GAN where we use DCGAN [23] as each of our three generators. Again, we use the same setting as provided in DCGAN. The high quality face generations are shown in the Fig. 9.

5.6. Unsupervised Representation Learning

Similar to DCGAN [23], we train our framework using SVHN dataset [21]. The trained discriminator is used to extract features. Using these features, we train an SVM for the classification task. For the MAD-GAN, with three generators, we obtained misclassification error of 17.5% which is almost 5% better than the results reported by DCGAN (22.48%). This clearly indicates that our framework is able to learn a better feature space in an unsupervised setting.

6. Conclusion

We presented a very simple and effective framework, Multi-Agent Diverse GAN (MAD-GAN), for generating diverse and meaningful samples. We showed the efficacy of our approach and compared it with various variants of GAN that it captures diverse modes while producing high quality samples. We presented a theoretical analysis of MAD-GAN with conditions for global optimality. Looking forward, an interesting future direction would be to estimate a priori the number of generators needed for a particular dataset. It is not clear how to do that given that we do not have access to the true data distribution. In addition, we would also like to theoretically understand the limiting cases that depend on the relationship between the number of generators and the complexity of the data distribution. Another interesting direction would be to exploit different generators such that their combinations can be used to capture diverse modes.

7. Acknowledgements

This work was supported by the EPSRC, ERC grant ERC-2012-AdG 321162-HELIOS, EPSRC grant Seebibyte EP/M013774/1 and EPSRC/MURI grant EP/N019474/1.

References

[1] M. Abadi and D. Andersen. Learning to protect communications with adversarial neural cryptography. *arXiv preprint arXiv:1610.06918*, 2016.

[2] M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations*, 2017.

[3] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. In *International Conference on Machine Learning*, 2017.

[4] S. Arora, R. Ge, Y. Liang, T. Ma, and Y. Zhang. Generalization and equilibrium in generative adversarial nets (gans). In *International Conference on Machine Learning*, 2017.

[5] D. Berthelot, T. Schumm, and L. Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

[6] C. Bishop. Pattern recognition and machine learning (information science and statistics), 1st edn. 2006. corr. 2nd printing edn. *Springer, New York*, 2007.

[7] T. Che, Y. Li, A. Jacob, Y. Bengio, and W. Li. Mode regularized generative adversarial networks. In *International Conference on Learning Representations*, 2017.

[8] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2016.

[9] I. Durugkar, I. Gemp, and S. Mahadevan. Generative multi-adversarial networks. In *International Conference on Learning Representations*, 2017.

[10] A. Ghosh, V. Kulharia, and V. Namboodiri. Message passing multi-agent gans. *arXiv preprint arXiv:1612.01294*, 2016.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014.

[12] P. Isola, J.-Y. Zhu, T. Zhou, and A. Efros. Image-to-image translation with conditional adversarial networks. In *Computer Vision and Pattern Recognition*, 2017.

[13] F. Juefei-Xu, V. N. Boddeti, and M. Savvides. Gang of gans: Generative adversarial networks with maximum margin ranking. *arXiv preprint arXiv:1704.04865*, 2017.

[14] D. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

[15] V. Kulharia, A. Ghosh, A. Mukerjee, V. Namboodiri, and M. Bansal. Contextual rnn-gans for abstract reasoning diagram generation. In *AAAI Conference on Artificial Intelligence*, 2017.

[16] S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 1951.

[17] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *Computer Vision and Pattern Recognition*, 2017.

[18] M.-Y. Liu and O. Tuzel. Coupled generative adversarial networks. In *Advances in neural information processing systems*, 2016.

[19] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. Unrolled generative adversarial networks. In *International Conference on Learning Representations*, 2017.

[20] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[21] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[22] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Computer Vision and Pattern Recognition*, 2016.

[23] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[24] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. In *International Conference on Machine Learning*, 2016.

[25] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, 2016.

[26] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *Advances In Neural Information Processing Systems*, 2016.

[27] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems*, 2016.

[28] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[29] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, 2016.

B
iSketchNFill Paper

Contributions:

- Arnab Ghosh came up with the ideas and conducted the experiments. This was in conjunction with weekly meetings with the mentors from Adobe: Richard Zhang, Oliver Wang and Eli Shechtman. Alexei Efros from UC Berkeley also helped in developing the core ideas. Philip Torr and Puneet Dokania from Oxford supervised the project.
- Richard Zhang and Puneet Dokania contributed heavily for the writing of the paper. It was reviewed and refined further by all the authors.

This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Interactive Sketch & Fill: Multiclass Sketch-to-Image Translation

Arnab Ghosh[1] Richard Zhang[2] Puneet K. Dokania[1]
Oliver Wang[2] Alexei A. Efros[2,3] Philip H. S. Torr[1] Eli Shechtman[2]

[1]University of Oxford [2]Adobe Research [3]UC Berkeley

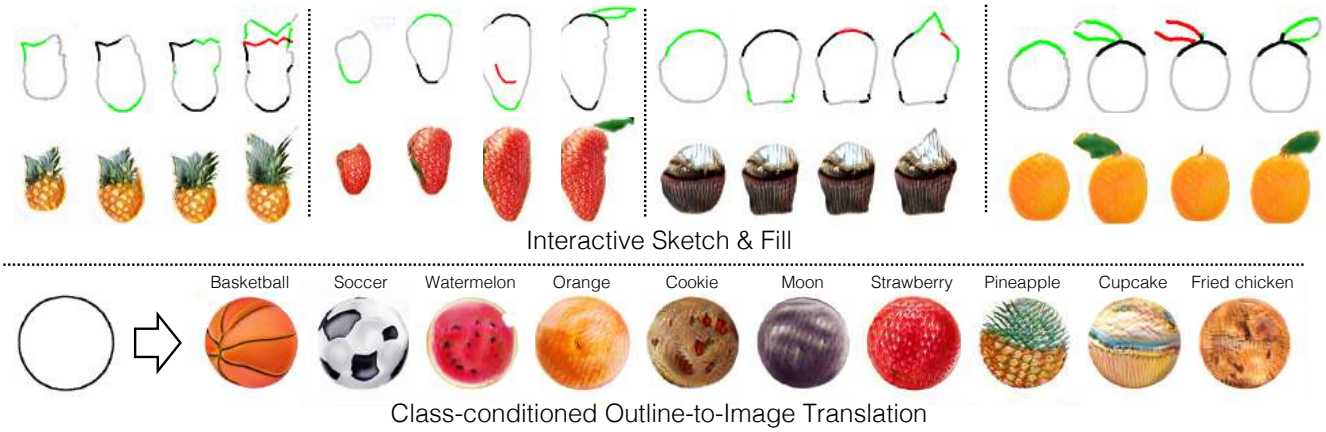


Figure 1: (**Top**) Given sparse user input (first row), our model estimates the complete shape and provides this as a recommendation to the user (shown in gray), along with the final synthesized object (second row). These estimates are updated as the user adds (green) or removes strokes (red) over time – previous edits are shown in black. (**Bottom**) This generation is class-conditioned, and our method is able to generate distinct multiple objects for the same outline (*e.g.* 'circle') by conditioning the generator on the object category.

Abstract

We propose an interactive GAN-based sketch-to-image translation method that helps novice users easily create images of simple objects. The user starts with a sparse sketch and a desired object category, and the network then recommends its plausible completion(s) and shows a corresponding synthesized image. This enables a feedback loop, where the user can edit the sketch based on the network's recommendations, while the network is able to better synthesize the image that the user might have in mind. In order to use a single model for a wide array of object classes, we introduce a gating-based approach for class conditioning, which allows us to generate distinct classes without feature mixing, from a single generator network.

1. Introduction

Conditional GAN-based image translation [25, 43, 61] models have shown remarkable success at taking an abstract user input, such as an edge map or a semantic segmentation map, and translating it to a real image. These methods run at interactive rates, and combining them with a user interface allows the user to quickly create fun (but usually, unrealistic) images. A few limitations prevent them from being used as a true interactive tool that assists the user in generating an image of an object they have in mind. First, the user is required to provide an entire abstract map as input (full edge or label map). This may prove difficult for many, as untrained practitioners generally struggle at free-hand drawing of accurate proportions of objects and their parts [6], 3D shapes and perspective [45]. It is much easier with current image translation methods to obtain realistic looking images by editing existing images [8, 40] than creating images from scratch. Second, current GAN-based image translation methods are limited to a single class of images. For example, switching from a cat to a dog requires loading (or storing in memory) a new model per class.

We propose a new GAN-based interactive image generation system for drawing objects that: 1) generates full images given only *sparse* and *partial* user strokes (or

sketches); 2) serves as a *recommender system* that suggests or helps the user *during* their creative process, in order to generate a desired image; and 3) uses a single conditional GAN for *multiple* image classes with an effective gating mechanism. Such a system allows for creative input to come from the user, while the challenging task of getting exact object proportions correct is left to the model, which constantly predicts a plausible completion of the user's sketch (Fig. 1).

We use sparse object outlines/sketches/simplified-edges instead of dense edge maps as the user input, as these are closer to the lines that novice users tend to draw [7]. Our model first completes the input, which could be partial outlines or edges, and then generates the image conditioned on the completed shape. There are several advantages to this two-stage approach. For one, we are able to give the artist feedback on the general object shape in our interactive interface (similar to ShadowDraw [31]), allowing them to quickly refine the completed shape until it is satisfactory. Second, we found this to work better than going directly from partial outlines to images, as the additional intermediate supervision on full outlines/sketches breaks the problem into two easier sub-problems – first recover the geometric properties of the object (shape, proportions) and then fill in the appearance (colors, textures).

For the second stage, multi-class conditional generation, we use a gating mechanism conditioned on the input class label. Briefly, gating allows the network to focus on the important parts (activations) of the network specific to the conditioning class. Such an approach allows for a clean separation of classes, enabling us to train a single generator and discriminator across *multiple* object classes.

To demonstrate the potential of our method as an interactive tool for stroke-based image generation, we collect a new image dataset of ten simple object classes (pineapple, soccer, basketball, etc.) with white background. In order to stress test our gating mechanism, six of the object classes have similar round outlines, so the model is truly conditioned on the class label and cannot figure out the class only from the stroke. Fig. 2 shows a short video of an interactive editing session using our system. Along with these simple objects, we also demonstrate the potential of our method on complicated ones such as faces and shoes.

2. Related Work

Interactive Generation Interactive interfaces for free-hand drawing go all the way back to Ivan Sutherland's Sketchpad [48]. The pre-deep work most related to us, ShadowDraw [31], introduced the concept of generating multiple shadows for novice users to be able to draw sketches. PhotoSketcher [13] introduces a retrieval based method for obtaining real images from sketches. More recently, deep recurrent networks have been used to generate sketches [18, 14]. Sketch-RNN [18] provides a completion of partial strokes, with the advantage of intermediate stroke information via the Quickdraw dataset at training time. SPIRAL [14] learns to generate digits and faces using a reinforcement learning approach. Zhu et al. [60] train a generative model, and an optimization-based interface to generate possible images, given color or edge constraints. The technique is limited to a single class and does not propose a recommendation for the completion of the shape. SketchyGAN [3] also aimed at generating multi-class images but lacks interactive capability. In contrast to the above, our method provides interactive prediction of the shape and appearance to the user and supports multiple object classes.

Generative Modeling Parametric modeling of an image distribution is a challenging problem. Classic approaches include autoencoders [21, 54] and Boltzmann machines [47]. More modern approaches include autoregressive models [12, 51], variational autoencoders (VAEs) [28], and generative adversarial networks (GANs). GANs and VAEs both learn mappings from a low-dimensional "latent" code, sampled stochastically, to a high-dimensional image through a feedforward pass of a network. GANs have been successful recently [9, 41, 1], and hybrid models feature both a learned mapping from image to latent space as well as adversarial training [10, 11, 30, 4].

Conditioned Image Generation The methods described above can be conditioned, either by a low-dimensional vector (such as an object class, or noise vector), a high-dimensional image, or both. Isola et al. [25] propose "pix2pix", establishing the general usefulness of conditional GANs for image-to-image translation tasks. However, they discover that obtaining multimodality by injecting a random noise vector is difficult, a result corroborated in [33, 38, 62]. This is an example of mode collapse [16], a phenomenon especially prevalent in image-to-image GANs, as the generator tends or ignore the low-dimensional latent code in favor of the high-dimensional image. Proposed solutions include layers which better condition the optimization, such as Spectral Normalization [58, 35], modifications to the loss function, such as WGAN [2, 17] or optimization procedure [20], or modeling proposals, such as MAD-GAN [15] and MUNIT [24]. One modeling approach is to add a predictor from the output to the conditioner, to discourage the model from ignoring the conditioner. This has been explored in the classification setting in Auxiliary-Classifier GAN (ACGAN) [36] and regression setting with InfoGAN [4] and ALI/BiGAN ("latent regressor" model) [11, 10], and is one half of BicycleGAN model [62]. We explore a complementary approach of architectural modification via gating.

Gating Mechanisms Residual networks [19], first intro-

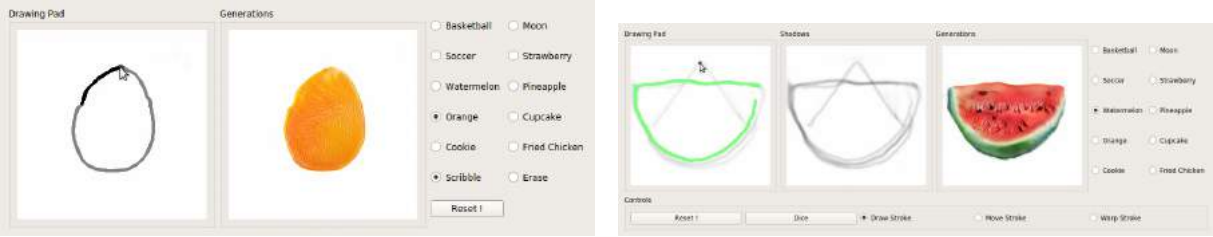


Figure 2: **Video of our interface** We can see two versions of our interface. The left side shows how a user can quickly generate multiple objects using a few strokes, while the right side shows the utility of multimodal completions where the user can quickly explore different possible shape generations while drawing. **Please view with Acrobat Reader.**

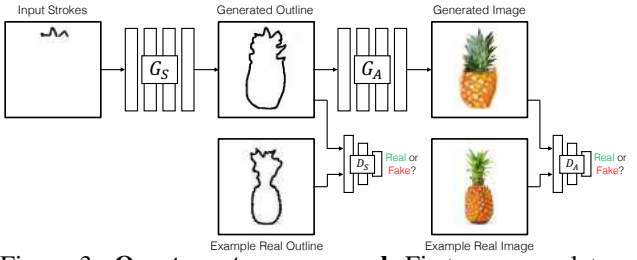


Figure 3: **Our two-stage approach** First, we complete a partial sketch using the shape generator G_S. Then we translate the completed sketch into an image using the appearance generator G_A. Both generators are trained with their respective discriminators D_S, and D_A.

duced for image classification [29], have made extremely deep networks viable to train. Veit et al. [53] find that the skip connection in the architecture enables test-time removal of blocks. Follow-up work [52] builds in block removal during training time, with the goal of subsets of blocks specializing to different categories. Inspired by these results, we propose the use of gating for image generation and provide a systematic analysis of gating mechanisms.

The adaptive instance normalization (AdaIn) layer has similarly been used in arbitrary style transfer [23] and image-to-image translation [24], and Feature-wise Linear Modulation (FiLM) [39]. Both methods scale and shift feature distributions, based on a high-dimensional conditioner, such as an image or natural language question. Gating also plays an important role in sequential models for natural language processing: LSTMs [22] and GRU [5]. Similarly, concurrent work [27], [37] use a AdaIN-style network to modulate the generator parameters.

3. Method

We decouple the problem of interactive image generation into two stages: object shape completion from sparse user sketches, and appearance synthesis from the completed shape. More specifically, as illustrated in Fig. 3 we use the Shape Generator G_S for the automatic shape (outline/sparse-sketch/simplified-edge) generation and the Appearance Generator G_A for generating the final image as well as the adversary discriminators D_S and D_A. Example usage is shown in our user interface in Fig. 2.

3.1. Shape completion

The shape completion network G_S should provide the user with a visualization of its completed shape(s), based on the user input, and should keep on updating the suggested shape(s) interactively. We take a data-driven approach for this whereby, to train the network, we simulate partial strokes (or inputs) by removing random square patches from the full outline/ full sparse sketch/ full simplified edges. The patches are of three sizes (64×64, 128×128, 192×192) and placed at a random location in the image of size 256×256 (see Fig. 5 for an example). To extend the technique beyond outlines and generate more human-like sketches, we adopt the multistage procedure depicted in Fig. 6. We refer to these generated sketches as "simplified edges". We automatically generate data in this manner, creating a dataset where for a given full outline/sketch or a simplified edge-map, 75 different inputs are created. The model, shown in Fig. 3, is based on the architecture used for non-image conditional generations in [34]. We modify the architecture such that the conditioning input is provided to the generator and discriminator at multiple scales as shown in Fig. 4. This makes the conditioning input an active part of the generation process and helps in producing multimodal completions.

3.2. Appearance synthesis

An ideal interactive sketch-to-image system should be able to generate multiple different image classes with a single generator. Beside memory and time considerations (avoiding loading/using a separate model per class, reducing overall memory), a single network can share features related to outline recognition and texture generation that are common across classes, which helps training with limited examples per class.

As we later show, class-conditioning by concatenation can fail to properly condition the network about the class information in current image translation networks [25, 62].

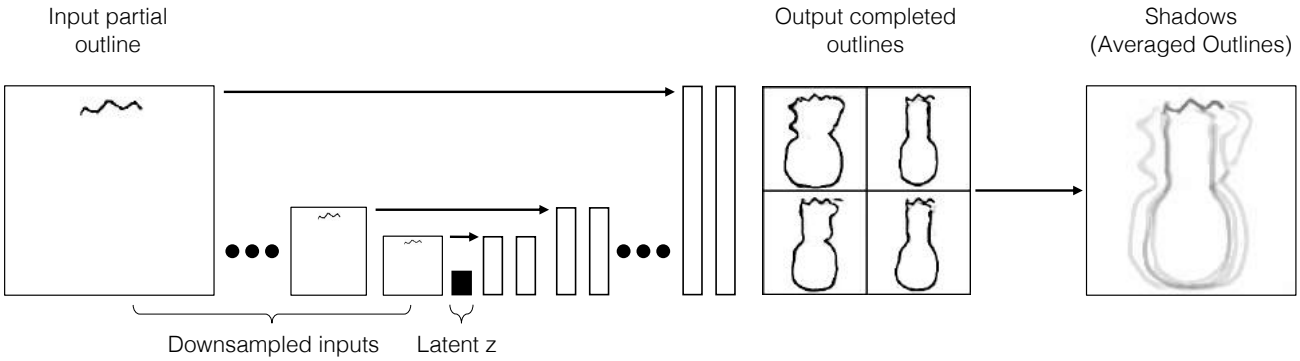
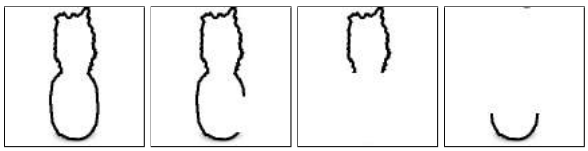


Figure 4: **First stage (Shape Generator)** To achieve multi-modal completions, the shape generator is designed using inspiration from non-image conditional model [34] with the conditioning input provided at multiple scales, so that the generator network doesn't ignore the partial stroke conditioning.



Outline Simulated Partial Inputs

Figure 5: **Simulated Inputs** Three sizes of occluders were used to simulate partial outlines.



Figure 6: **Simplified Edges** The 2nd edgemap is obtained using the technique of [25], while the 3rd is the intermediate edgemap using [32] and further simplified using [46] which looks closer to what a human would sketch.

To address this, we propose an effective soft gating mechanism, shown in Fig. 7. Conceptually, our network consists of a small external gating network that is conditioned on the object class (encoded as a 1-hot vector). The gating network outputs parameters that are used to modify the features of the main generator network. Given an input feature tensor X_l, "vanilla" ResNet [19] maps it to

$$X_{l+1} = X_l + \mathcal{H}_l(X_l). \tag{1}$$

Changes in resolution are obtained by upsampling before or downsampling after the residual block. Note that we omit l subscript from this point forward to reduce clutter. Our gating network augments this with a predicted scalar α for each layer of the network using a learned network $\mathcal{F}(\mathbf{y})$, where $\mathbf{y}$ is the conditioning vector:

$$X + \alpha\, \mathcal{H}(X), \text{where } \alpha \in [0, 1] \tag{2}$$

If the conditioning vector $\mathbf{y}$ has no use for a particular block, it can predict α close to zero and effectively switch off the layer. During training, blocks within the main network can transform the image in various ways, and $\mathcal{F}$ can modulate such that the most useful blocks are selected. Unlike previous feature map conditioning methods such as AdaIn [50], we apply gating to *both* the generator and discriminator. This enables the discriminator to select blocks which effectively judge whether generations are real or fake, conditioned on the class input. Some blocks can be shared across regions in the conditioning vector, whereas other blocks can specialize for a given class.

A more powerful method is to apply this weighting channel-wise using a vector $\boldsymbol{\alpha}$:

$$X + \boldsymbol{\alpha} \odot \mathcal{H}(X), \text{where } \boldsymbol{\alpha} \in [0, 1]^c, \tag{3}$$

where $\odot$ represents channel-wise multiplication. This allows specific channels to be switched "on" or "off", providing additional degrees of freedom. We found that this channelwise approach for gating provides the strongest results. AdaIn describes the case where an Instance Normalization [50] (IN) operation is applied before scaling and shifting the feature distribution. We constrain each element of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in $[-1, 1]$. We additionally explored incorporating a bias term after the soft-gating, either block-wise using a scalar $\beta \in [-1, 1]$ per layer, or channel-wise using a vector $\boldsymbol{\beta} \in [-1, 1]^c$ per layer but we found that they did not help much, and so we leave them out of our final model. Refer Fig. 8 for pictorial representation of various gatings.

Finally, we describe our network architecture, which utilizes the gated residual blocks described above. We base our architecture on the proposed residual **Encoder-Decoder** model from MUNIT [24]. This architecture is comprised of 3 `conv` layers, 8 residual blocks, and 3 `up-conv` layers. The residual blocks have 256 channels. First, we deepen the network, based on the principle that deeper networks have more valid disjoint, partially shared paths [53], and add 24 residual blocks. To enable the larger number of residual

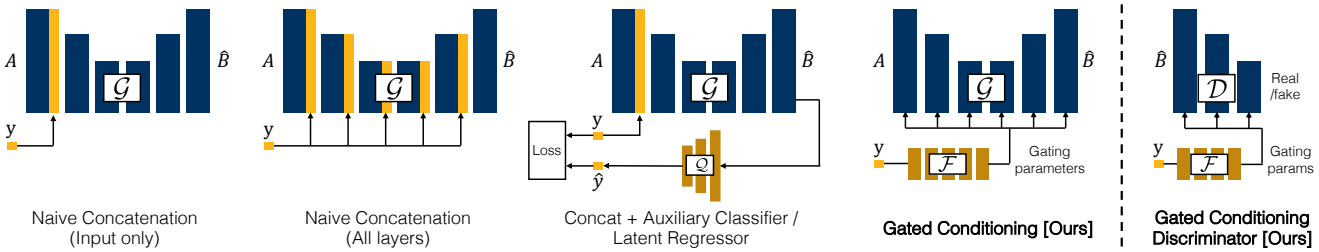


Figure 7: **Conditioning variants for the Appearance Generator** Our model uses gating on all the residual blocks of the generator and the discriminator, other forms of conditioning such as (naive concatenation in input only, all layers, AC-GAN like latent regressor [36]) are evaluated as well.

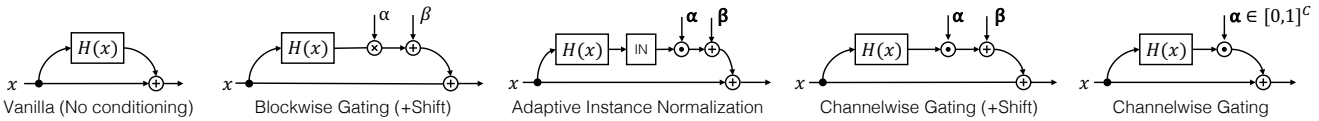


Figure 8: **Injecting conditioning with modified residual layers (Left)** A "vanilla" residual block without conditioning applies a residual modification to the input tensor. **(Mid-left)** The $\mathcal{H}(X)$ block is softly-gated by scalar parameter α and shift β. **(Mid)** Adaptive Instance Normalization [23] applies a channel-wise scaling and shifting after an instance normalization layer. **(Mid-right)** Channel-wise gating adds restrictions to the range of $\boldsymbol{\alpha}$. **(Right)** We find that channel-wise gating (without added bias) produces the best results empirically.

Trained task	FID
Faces	
Partial Simplified Edges → Image	383.02
Partial Simplified Edges → Simplified Edges → Image	374.67
Shoes	
Partial Simplified Edges → Image	170.45
Partial Simplified Edges → Simplified Edges → Image	154.32

Table 1: **Single-class generation, 2-stage vs 1-stage**. We evaluate the result quality from different task pipelines.

Trained task	Avg Acc
Partial edges → Image	73.12 %
Partial outline → Image	88.74 %
Partial outline → Full outline → Image [Ours]	**97.38%**

Table 2: **Multi-class generation, 2-stage vs 1-stage**. We evaluate the result quality from different task pipelines. Accuracy is computed by a fixed, pretrained classification network, on the resulting images.

blocks, we drastically reduce the width to 32 channels for every layer. We refer to this network as **SkinnyResNet**. Additionally, we found that modifying the downsampling and upsampling blocks to be residual connections as well improved results, and also enables us to apply gating to *all* blocks. When gating is used, the gate prediction network, $\mathcal{F}(\mathbf{y})$, is also designed using residual blocks. Additional architecture details are in the supplementary material.

4. Experiments

We first compare our 2 step approach for interactive image generation on existing datasets such as the UTZappos Shoes dataset [57] and CelebA-HQ [26]. State-of-the-art techniques such as pix2pixHD [55] are used to generate the final image from the autocompleted sketches. We finally evaluate our approach on a multi-class dataset that we collected to test our proposed gating mechanism.

4.1. Single Class Generation

Datasets We use the edges2shoes[25], CelebA-HQ[26] datasets to test our method on single class generation. We simplify the edges to attempt to more closely resemble how humans would draw strokes by first using the preprocessing code of [32] further reducing the strokes with a sketch simplification network [46].

Architecture We use the architecture described in Section 3.1 for shape completion. In this case, each dataset only contains a single class, so we can use an off-the-shelf network, such as pix2pixHD [56] for rendering.

Results As seen in Fig. 9, our 2 step technique allows us to complete the simplified edge maps from the partial strokes and also generate realistic images from the autocompleted simplified edges. Table 1 also demonstrates, across two datasets (faces and shoes), that using a 2 step procedure produces stronger results than mapping directly from the partial sketch to the completed image.

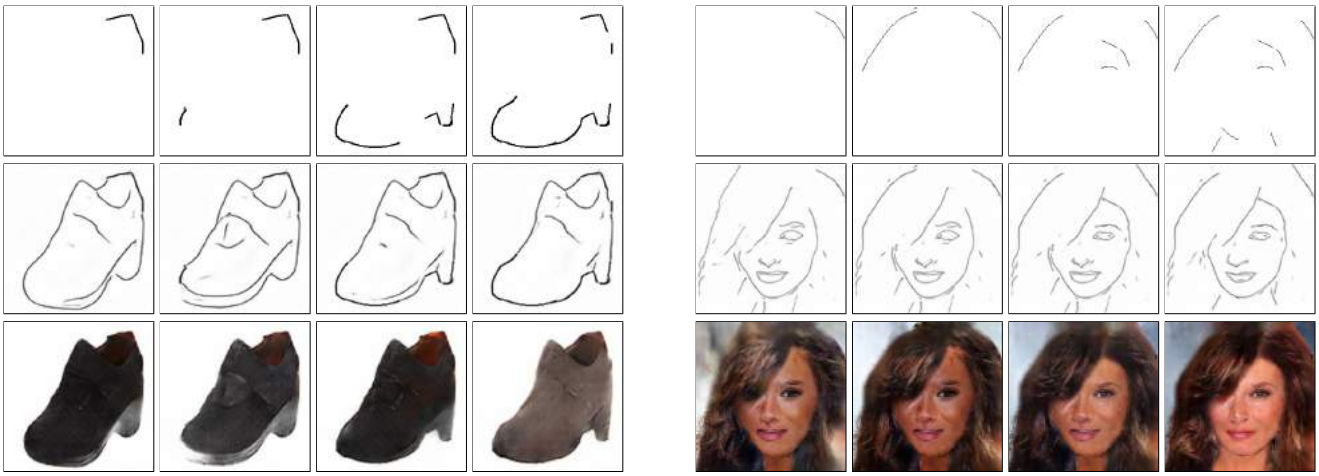


Figure 9: **Example Sketch & Fill Progression.** The **first row** represents the progressive addition of new strokes on the canvas, the **second row** shows the autocompleted sketch, and the **third row** is the final generated image. As the sparse strokes are changed by the user, the completed shape and generated image evolve as well. Note that changing a stroke locally produces coherent changes in other parts of the image.

Method	SkinnyResNet		EncDec	
	Class. Acc [%]	AMT Fool. Rate [%]	Class. Acc [%]	AMT Fool. Rate [%]
Ground truth	100.0	50.0	100.0	50.0
1 gen/class	***97.0***	17.7±1.46	–	–
Concat (In)	62.6	15.0±1.4	39.2	7.5±1.06
Concat (All)	64.5	15.3±1.41	51.4	5.4±0.88
Cat(In)+Aux-Class	65.6	14.5±1.5	–	–
Cat(All)+Aux-Class	67.0	19.7±1.42	–	–
BlockGate(+bias)	89.6	19.6±1.34	–	–
BlockGate	**99.6**	17.3±1.61	–	–
AdaIn	94.5	14.9±1.47	–	–
ChanGate(+bias)	94.1	14.8±1.43	–	–
ChanGate	***97.0***	**23.4±1.99**	92.7	14.1±1.48

Table 3: **Accuracy vs Realism on Multiclass Outline→Image task.** We measure generation accuracy with a pretrained network. We measure realism using the real vs. fake judges from AMT. Higher is better for both. Our SkinnyResNet architecture outperforms the Encoder-Decoder network, inspired by MUNIT [24]. We perform a thorough ablation on our architecture and find that channel-wise gating achieves high accuracy and higher realism.

4.2. Multi-Class Generation

Datasets To explore the efficacy of our full pipeline, we introduce a new outline dataset consisting of 200 images (150 train, 50 test) for each of 10 classes – basketball, chicken, cookie, cupcake, moon, orange, soccer, strawberry, watermelon and pineapple. All the images have a white background and were collected using search keywords on popular search engines. In each image, we obtain rough outlines for the image. We find the largest blob in the image after thresholding it into a black and white image. We fill the interior holes of the largest blob and obtain a smooth outline using the SavitzkyGolay filter [44].

Architecture For the shape completion, we use the architecture in Section 3.1. For class-conditioned image generation, test the gated architectures in Section 3.2.

Results In order to test the fidelity of the automatically completed shapes, we evaluate the accuracy of a trained classifier on being able to correctly label a particular generation. We first test in Table 2 that our 2 stage technique is better than 1 step generation. We evaluate the results on the multi-class outline to image generations on two axes: adherence to conditioning and realism. We first test the conditioning adherence – whether the network generates an image of the correct class. Off-the-shelf networks have been previously used to evaluate colorizations [59], street scenes [25, 56], and ImageNet generations [42]. We take a similar approach and fine-tune a pretrained InceptionV3 network [49] for our 10 classes. The generations are then tested with this network for classification accuracy. Results are presented in Table 3.

To judge the generation quality, we also perform a "Visual Turing test" using Amazon Mechanical Turk (AMT). Turkers are shown a real image, followed by a generated image, or vice versa, and asked to identify the fake. An algorithm which generates a realistic image will "fool" Turkers into choosing the incorrect image. We use the implementation from [59]. Results are presented in Table 3, and qualitative examples are shown in Fig. 10.

Gating Architectures We compare our proposed model to the residual **Encoder-Decoder** model [24]. In addition, we compare our proposed gating strategy and **SkinnyResNet** architecture to the following methods for conditional image generation:



Figure 10: **Conditioning injection comparison.** We show results across methods on the outline→image task using the **SkinnyResNet** architecture. Naive Concatenation **Concat** often confuses classes, such as oranges and basketballs, while gating mechanisms such as the **ChannelGate** method succeed. The gating method also improves results for the **EncoderDecoder** architecture.

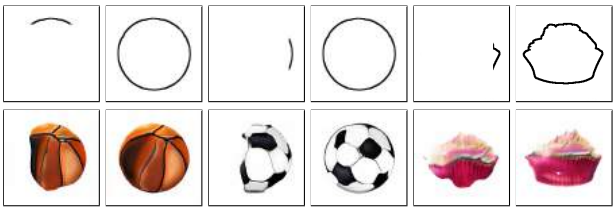


Figure 11: **Directly mapping from partial outline to image** Our proposed system uses a 2-stage approach, using a completed edge map as an intermediate. Here, we show results when directly mapping from the partial outline to the image. When the outline is well-defined, the network can generate realistic images. However, when the outline is sparse, the network struggles with the geometry.

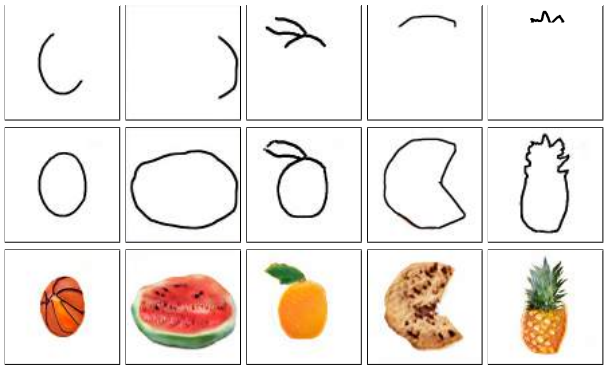


Figure 12: **Multiclass Sketch & Fill results** A few input strokes (first row) are enough to automatically complete the class specific outlines (second) and appearance (last).

- **Per-class**: a single generator for each category; this is the only test setting with *multiple* networks, all others train a single network
- **Concat (In)**: naive concatenation, input layer only
- **Concat (All)**: naive concatenation, all layers
- **Concat (In)+Aux-Class**: we add an auxiliary classifier, both for input-only and all layers settings
- **BlockGate(+Bias), BlockGate**: block-wise soft-gating, with and without a bias parameter
- **AdaIn**: Adaptive instance normalization
- **ChannelGate(+Bias), ChannelGate**: channel-wise soft-gating, with and without a bias parameter

Does naive concatenation effectively inject conditioning? In Fig. 10, we show a selected example from each of the 10 classes. The per-class baseline trivially adheres to the conditioning, as each class gets to have its own network. However, when a single network is trained to generate all classes, naive concatenation is unable to successfully inject class information, for either network and for either type of concatenation. For the **EncoderDecoder** network, basketballs, oranges, cupcakes, pineapples, and fried chicken are all confused with each other. For the **SkinnyResNet** network, oranges are generated instead of basketballs, and pineapples and fried chicken drumsticks are confused. As seen in Table 3, classification accuracy is slightly higher when concatenating all layers (64.5%) versus only the input layer (62.6%), but is low for both.

Does gating effectively inject conditioning? Using the proposed soft-gating, on the other hand, leads to successful generations. We test variants of soft-gating on the **SkinnyResNet**, and accuracy is dramatically improved, between 89.6% to 99.6%, comparable to using a single generator per class (97.0%). Among the gating mechanisms, we find that channel-wise multiplication generates the most realistic images, achieving an AMT fooling rate of 23.4%. Interestingly, the fooling rate is higher than the per-class generator of 17.7%. Qualitatively, we notice that per-class generators sometimes exhibits artifacts in the background, as seen in the generation of "moon". We hypothesize with the correct conditioning mechanism, the single generator across multiple classes has the benefit of seeing more training data and finding common elements across classes, such as clean, white backgrounds.

Is gating effective across architectures? As seen in Table 3, using channelwise gating instead of naive concatenation improves performance both accuracy and realism *across* architectures. For example, for the **EncoderDecoder** architecture, gating enables successful generation of the pineapple. Both quantitatively and qualitatively, results are better for our proposed **SkinnyResNet** architecture.

Do the generations generalize to unusual outlines? The training images consist of the outlines corresponding to the geometry of each class. However, an interesting test scenario is whether the technique generalizes to unseen shape and class combinations. In Fig. 1, we show that an input circle not only produces circular objects, such as a basketball, watermelon, and cookie, but also noncircular objects such as strawberry, pineapple, and cupcake. Note that both the pineapple crown and bottom are generated, even without any structural indication of these parts in the outline.

5. Discussion

We present a two-stage approach for interactive object generation, centered around the idea of a shape completion intermediary. This step both makes training more stable and also allows us to give coarse geometric feedback to the user, which they can choose to integrate as they desire.

Acknowledgements

AG, PKD, and PHST are supported by the ERC grant ERC-2012-AdG, EPSRC grant Seebibyte EP/M013774/1, EPSRC/MURI grant EP/N019474/1 and would also like to acknowledge the Royal Academy of Engineering and FiveAI. Part of the work was done while AG was an intern at Adobe.

References

[1] Martín Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *ICLR*, 2017. 2

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *ICML*, 2017. 2

[3] Wengling Chen and James Hays. Sketchygan: towards diverse and realistic sketch to image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9416–9425, 2018. 2

[4] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *NIPS*, 2016. 2

[5] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *EMNLP*, 2014. 3

[6] Dale J Cohen and Susan Bennett. Why can't most people draw what they see? *Journal of Experimental Psychology: Human Perception and Performance*, 23(3):609, 1997. 1

[7] Forrester Cole, Aleksey Golovinskiy, Alex Limpaecher, Heather Stoddart Barros, Adam Finkelstein, Thomas Funkhouser, and Szymon Rusinkiewicz. Where do people draw lines? *ACM Transactions on Graphics (TOG)*, 27(3):88, 2008. 2

[8] Tali Dekel, Chuang Gan, Dilip Krishnan, Ce Liu, and William T Freeman. Sparse, smart contours to represent and edit images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3511–3520, 2018. 1

[9] Emily L Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS*, 2015. 2

[10] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *ICLR*, 2017. 2

[11] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *ICLR*, 2017. 2

[12] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *ICCV*, 1999. 2

[13] Mathias Eitz, Ronald Richter, Kristian Hildebrand, Tamy Boubekeur, and Marc Alexa. Photosketcher: interactive sketch-based image synthesis. *IEEE Computer Graphics and Applications*, 31(6):56–66, 2011. 2

[14] Yaroslav Ganin, Tejas Kulkarni, Igor Babuschkin, SM Eslami, and Oriol Vinyals. Synthesizing programs for images using reinforced adversarial learning. *ICML*, 2018. 2

[15] Arnab Ghosh, Viveka Kulharia, Vinay Namboodiri, Philip H. S. Torr, and Puneet K Dokania. Multi-agent diverse generative adversarial networks. *CVPR*, 2018. 2

[16] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016. 2

[17] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *NIPS*, pages 5767–5777, 2017. 2

[18] David Ha and Douglas Eck. A neural representation of sketch drawings. *Conference on Neural Information Processing Systems*, 2017. 2

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2, 4

[20] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017. 2

[21] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. 2

[22] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 3

[23] Xun Huang and Serge J Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, pages 1510–1519, 2017. 3, 5

[24] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. *ECCV*, 2018. 2, 3, 4, 6

[25] P. Isola, J-Y. Zhu, T. Zhou, and A. Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017. 1, 2, 3, 4, 5, 6

[26] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *ICLR*, 2018. 5

[27] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *CVPR*, 2019. 3

[28] D. Kingma and M. Welling. Auto-encoding variational bayes. *ICLR*, 2014. 2

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 3

[30] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *ICML*, 2016. 2

[31] Yong Jae Lee, C Lawrence Zitnick, and Michael F Cohen. Shadowdraw: real-time user guidance for freehand drawing. In *ACM Transactions on Graphics (TOG)*, volume 30, page 27. ACM, 2011. 2

[32] Yijun Li, Chen Fang, Aaron Hertzmann, Eli Shechtman, and Ming-Hsuan Yang. Im2pencil: Controllable pencil illustration from photographs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1525–1534, 2019. 4, 5

[33] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. In *ICLR*, 2016. 2

[34] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? *ICML*, 2018. 3, 4

[35] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *ICLR*, 2018. 2

[36] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *ICML*, 2017. 2, 5

[37] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. *CVPR*, 2019. 3

[38] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. Efros. Context encoders: Feature learning by inpainting. *CVPR*, 2017. 2

[39] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. *AAAI*, 2018. 3

[40] Tiziano Portenier, Qiyang Hu, Attila Szabo, Siavash Arjomand Bigdeli, Paolo Favaro, and Matthias Zwicker. Faceshop: Deep sketch-based face image editing. *ACM Transactions on Graphics (TOG)*, 37(4):99, 2018. 1

[41] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *ICLR*, 2016. 2

[42] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *NIPS*, 2016. 6

[43] Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Scribbler: Controlling deep image synthesis with sketch and color. In *CVPR*, volume 2, 2017. 1

[44] Abraham Savitzky and Marcel JE Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639, 1964. 6

[45] Ryan Schmidt, Azam Khan, Gord Kurtenbach, and Karan Singh. On expert performance in 3d curve-drawing tasks. In *Proceedings of the 6th eurographics symposium on sketch-based interfaces and modeling*, pages 133–140. ACM, 2009. 1

[46] Edgar Simo-Serra, Satoshi Iizuka, Kazuma Sasaki, and Hiroshi Ishikawa. Learning to simplify: fully convolutional networks for rough sketch cleanup. *ACM Transactions on Graphics (TOG)*, 35(4):121, 2016. 4, 5

[47] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, DTIC Document, 1986. 2

[48] Ivan E. Sutherland. Sketch pad a man-machine graphical communication system. In *Proceedings of the SHARE Design Automation Workshop*, DAC '64, pages 6.329–6.346, New York, NY, USA, 1964. ACM. 2

[49] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, pages 2818–2826, 2016. 6

[50] D Ulyanov, A Vedaldi, and VS Lempitsky. Instance normalization: the missing ingredient for fast stylization. corr abs/1607.0 (2016). 4

[51] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *NIPS*, 2016. 2

[52] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. *ECCV*, 2018. 3

[53] Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In *NIPS*, pages 550–558, 2016. 3, 4

[54] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008. 2

[55] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 5

[56] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. *CVPR*, 2018. 5, 6

[57] Aron Yu and Kristen Grauman. Fine-grained visual comparisons with local learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 192–199, 2014. 5

[58] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *ICML*, 2019. 2

[59] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, pages 649–666. Springer, 2016. 6

[60] J. Zhu, P. Krähenbühl, E. Shechtman, and A. Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, 2016. 2

[61] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CVPR*, 2017. 1

[62] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. pages 465–476, 2017. 2, 3

C
DGPose Paper

Contributions:

- Rodrigo De Bem came up with the initial idea and set up the pipelines for the initial experimentation.
- Arnab Ghosh helped in conducting the GAN and VAE-GAN based experiments.
- All of the authors helped in writing and refining the manuscript.

International Journal of Computer Vision (2020) 128:1537–1563
https://doi.org/10.1007/s11263-020-01306-1

DGPose: Deep Generative Models for Human Body Analysis

Rodrigo de Bem[1,2] · Arnab Ghosh[1] · Thalaiyasingam Ajanthan[1,3] · Ondrej Miksik[1] · Adnane Boukhayma[1] · N. Siddharth[1] · Philip Torr[1]

Received: 13 November 2018 / Accepted: 7 February 2020 / Published online: 24 April 2020
© The Author(s) 2020

Abstract
Deep generative modelling for human body analysis is an emerging problem with many interesting applications. However, the latent space learned by such approaches is typically not interpretable, resulting in less flexibility. In this work, we present deep generative models for human body analysis in which the body pose and the visual appearance are disentangled. Such a disentanglement allows independent manipulation of pose and appearance, and hence enables applications such as pose-transfer without specific training for such a task. Our proposed models, the Conditional-DGPose and the Semi-DGPose, have different characteristics. In the first, body pose labels are taken as conditioners, from a fully-supervised training set. In the second, our structured semi-supervised approach allows for pose estimation to be performed by the model itself and relaxes the need for labelled data. Therefore, the Semi-DGPose aims for the joint *understanding* and *generation* of people in images. It is not only capable of mapping images to interpretable latent representations but also able to map these representations back to the image space. We compare our models with relevant baselines, the ClothNet-Body and the Pose Guided Person Generation networks, demonstrating their merits on the Human3.6M, ChictopiaPlus and DeepFashion benchmarks.

Keywords Deep generative models · Semi-supervised learning · Human pose estimation · Variational autoencoders · Generative adversarial networks

1 Introduction

Human body analysis has been a long-standing goal in computer vision, with many applications in human-machine interaction, health-care, shopping, sports, entertainment and gaming (Achilles et al. 2016; Moeslund et al. 2011; Seemann et al. 2004; Shotton et al. 2011; von Marcard et al. 2017). Popular approaches to this problem have focused on supervised learning of discriminative models (Bulat and Tzimiropoulos 2016; Cao et al. 2017; Chu et al. 2017; Wei et al. 2016), which map visual inputs (images or videos) to suitable abstract representations (e.g. human body pose). While these approaches do exceptionally well on their prescribed task, as evidenced by their performance on pose estimation benchmarks (Andriluka et al. 2014; Ionescu et al. 2014; Johnson and Everingham 2010), they fall short due to: (a) reliance on fully-labelled data, and (b) the inability to generate novel data from the abstractions.

The former is a fairly onerous shortcoming, particularly when one is dealing with real-world visual data, as it requires a substantial amount of human time and effort to annotate. Thus, being able to relax the reliance on labelled data is a

Communicated by Xavier Alameda-Pineda, Elisa Ricci, Albert Ali Salah, Nicu Sebe, Shuicheng Yan.

Rodrigo de Bem
rodrigo@robots.ox.ac.uk

Arnab Ghosh
arnabg@robots.ox.ac.uk

Thalaiyasingam Ajanthan
ajanthan@robots.ox.ac.uk

Ondrej Miksik
omiksik@robots.ox.ac.uk

Adnane Boukhayma
adnane@robots.ox.ac.uk

N. Siddharth
nsid@robots.ox.ac.uk

Philip Torr
phst@robots.ox.ac.uk

[1] Department of Engineering Science, University of Oxford, Oxford, UK

[2] Center of Computational Sciences, Federal University of Rio Grande, Rio Grande, Brazil

[3] Present Address: Australian National University, Canberra, Australia

highly desirable goal. The latter, states a rather significant limitation, the incapacity to manipulate abstractions directly with the aim of generating novel visual data. For instance, changes in the pose of an arm cannot be used for the generation of images or videos in which that arm is correspondingly displaced.

Generative models, in contrast to discriminative ones, enable the *analysis-by-synthesis* of the human body. With them, ideally, one could generate images of humans in diverse combinations of body poses and appearances, i.e. clothing, skin colours, hairstyles, and scenarios. This has many potential applications. For instance, it can be used for performance capture and reenactment of RGB videos, as already showcased for faces (Thies et al. 2016), and still incipient for human bodies (Balakrishnan et al. 2018; Chan et al. 2018). It can also be used to generate images in user-specified poses, to enhance and augment datasets with minimal annotation effort.

Recently, such approaches have been commonly formulated as deep generative models (DGMs) (Goodfellow et al. 2014; Kingma and Welling 2013; Rezende et al. 2014)—an extension of standard generative models that incorporate neural networks as flexible function approximators. These models are particularly effective in complex perceptual domains such as computer vision (Kulkarni et al. 2015), language (Massiceti et al. 2018), and robotics (Wang et al. 2017), effectively delegating bottom-up feature learning to neural networks, while simultaneously incorporating top-down probabilistic semantics into the model. They solve both the deficiencies of discriminative methods discussed above by (a) employing unsupervised learning, thereby removing the need for labels, and (b) embracing a fully generative modelling.

However, DGMs introduce a new problem—the learnt abstractions, or latent variables, are not human-interpretable. This lack of interpretability is a by-product of the unsupervised learning of representations from data. The learnt latent variables, usually represented as a smooth high-dimensional manifold, do not have the consistent semantic meaning as different sub-spaces in this manifold can encode arbitrary variations in the data. This is particularly unsuitable for our purposes as we would like to view and manipulate the latent variables, e.g. the body pose.

In order to ameliorate the issue mentioned above, while still eschewing reliance on fully-labelled data, we rely on a structured semi-supervised variational autoencoder (VAE) framework (Kingma et al. 2014; Siddharth et al. 2017). Here, the model structure is assumed to be partially specified, with consistent semantics imposed on some interpretable subset of the latent variables (e.g. pose), and the rest is left to be non-interpretable, although referred by us here as appearance. Weak (semi) supervision acts as a means to constrain the pose latent variables to actually encode the pose. This gives us the full complement of desirable features, allowing (a) semi-supervised learning, relaxing the need for labelled data, (b) generative modelling through stochastic computation graphs (Schulman et al. 2015), and (c) interpretable subset of latent variables defined through the model structure.

In this context, we present a structured semi-supervised VAEGAN architecture, the Semi-DGPose, in which we further extend structured semi-supervised models (Kingma et al. 2014; Siddharth et al. 2017) with a discriminator-based loss function from generative adversarial networks (GANs) (Goodfellow et al. 2014; Larsen et al. 2016), formulating it as a principled and unified probabilistic framework. To our knowledge, it is the first structured semi-supervised deep generative model of people directly learned in the *natural image* (or *natural scene*) space. This allows the method to directly learn the intricacies in the formation of natural (i.e. real) images. However, it is important to mention that natural images, in contrast to artificial visual stimuli (e.g. segmentation masks, binary masks, or pose vectors), have complex statistical structure and are much more challenging to parameterised (Geisler 2008; Kay et al. 2008; Simoncelli et al. 2001). Consequently, methods that work well with the latter may not succeed when tackling the former (Fei-Fei and Perona 2005; Krizhevsky et al. 2010). In contrast to previous work (Lassner et al. 2017; Ma et al. 2017, 2018; Siarohin et al. 2018; Walker et al. 2017), our model directly enables: (i) *semi-supervised pose estimation*; and (ii) *indirect pose-transfer* without specific training for such a task, both of which are tested and verified by experimental evidence.

Additionally, as an intermediate step in the investigation towards our main contribution, we propose a conditional-VAEGAN model, dubbed Conditional-DGPose. It is less distinct from previous art (Lassner et al. 2017; Ma et al. 2017), however, still differently from earlier work in the literature, it has: (i) allowed pose manipulation on extreme cases, e.g. by performing *cross-domain pose-transfer* and by *hallucinating* multiple people, in a variety of unseen or even unrealistic poses; and (ii) achieved state-of-the-art results on image reconstruction conditioned on pose, outperforming the closest related comparable baseline (Lassner et al. 2017). We illustrate some capabilities of our models in Fig. 1.

The present paper builds upon our previous approaches (de Bem et al. 2018, 2019) with further theoretical and technical details, evaluation, and discussion. Here, we present in full our comprehensive deep generative model framework for human body analysis in images. Along with an overview of VAEGAN models, this enables us to shed light on differences and similarities between conditional-VAEGANs and structured semi-supervised VAEGANs. More precisely, we provide additional evaluations of our Conditional-DGPose and Semi-DGPose models on the most relevant bench-

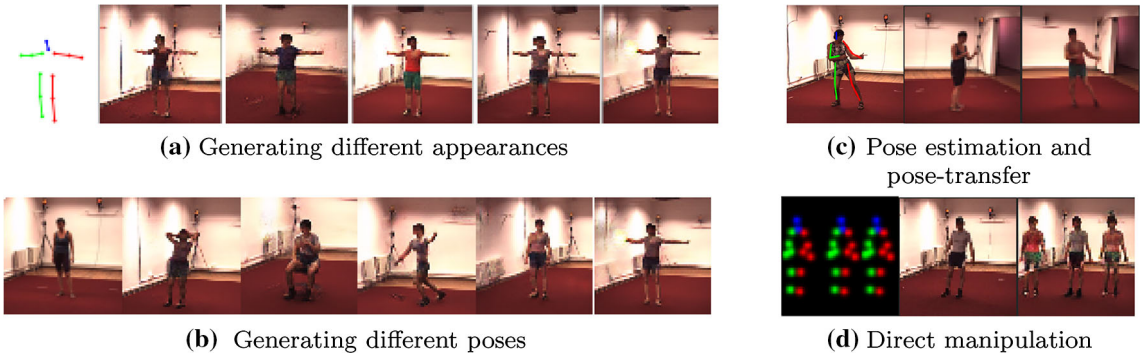


Fig. 1 Sampled results from our deep generative models for images of people. (**a**) For a given pose (first image), we show some samples of appearance. (**b**) For a given appearance (first image), samples of different poses. (**c**) For an estimated pose (first image) and a given appearance (second image), we show a generated sample combining the pose of the first image with the appearance of the second. (**d**) For manipulated poses (first image) and a given appearance (second image), it can *hallucinate* people in the scene

marks in the literature, the Human3.6M (Ionescu et al. 2014), the ChictopiaPlus (Lassner et al. 2017), and the DeepFashion (Liu et al. 2016) datasets. We also provide new qualitative and quantitative comparisons with the Pose Guided Person Generation (PG^2) baseline (Ma et al. 2017). The application of our models to real images and the results obtained are essential to show the relevance of interpretable and structured modelling. This emphasise the effectiveness of the proposals, despite the significant challenge of jointly aim for *understanding* and *generating* people in images. In summary, our main contributions are:

(i) a comprehensive framework for the joint *understanding* and *generation* of people in images, not only capable of mapping images to interpretable latent representations but also capable of mapping these representations back to the image space;
(ii) a real-world application of structured deep generative models of images, disentangling pose from appearance in the analysis of the human body;
(iii) a thorough quantitative and qualitative evaluation of the capabilities of our models; and
(iv) a demonstration of its principal utilities by performing semi-supervised pose estimation, pose-transfer and pose manipulation.

2 Related Work

2.1 Analysing Humans in Images: Overview

The analysis of people in visual data has been actively investigated as a computer vision and machine learning topic lately (Balakrishnan et al. 2018; Chan et al. 2018; Hattori et al. 2018, 2015; Rogez and Schmid 2016, 2018; Thies et al. 2016; Varol et al. 2017; Wang et al. 2018). Historically, the process of synthesising *virtual humans* (Hilton et al. 1999; Magnenat-Thalmann and Thalmann 2005, 2006) is a computer graphics undertake since its origins in the '60s, with Boeing's "first man" (Boeing 2018; Fetter 1982). Therefore, the geometric and photometric intricacies in the formation of digital images depicting people are well-known in computer graphics, as demonstrated by the existence of many commercial and academic specialised engines (3Lateral 2018; MakeHuman 2018; Massive Software 2018; Müller et al. 2018; Poser 2018; Unreal Engine 2018). Nonetheless, the unconstrained creation of truly realistic RGB images is still reasonably dependent upon manual intervention (Ian Spriggs 2018). Moreover, to produce accurate images of people is harder since humans seem to be very familiarised to corporal traits (e.g. faces) even since their early ages (MacDorman et al. 2009; Valenza et al. 1996).

Over time, the generation of humans in images was also embraced by the computer vision community. Aiming for less manual intervention, image-based techniques were successfully adopted on matters like rendering and modelling (Blanz et al. 1999; Borshukov et al. 2005; Ezzat and Poggio 1996; Kanade et al. 1997; Starck and Hilton 2007). For instance, a large body of work has relied on geometric 3D models for generating synthetic images of faces (Ichim et al. 2015; Pighin et al. 2006), bodies (Bogo et al. 2014; Starck et al. 2005), and hands (de La Gorce et al. 2011; Romero et al. 2017; Rosales et al. 2001). Despite that, to automatically synthesise artificial images indistinguishable from real ones may be considered as equivalent to succeed in a *visual Turing test* (Shan et al. 2013). Hence, a substantially complicated and consequently yet unsolved challenge (Fan et al. 2018).

Another line of approaches, following the machine learning methodologies closely, had modelled the image formation by designing and learning probabilistic generative

models (Enzweiler and Gavrila 2008; Fleuret et al. 2007; Fossati et al. 2007; Franco and Boyer 2005; Lee and Elgammal 2005; Wang et al. 2004; Yuille and Kersten 2006). However, it is highly complex and constrained due to intractable probability distributions and the high variability of latent factors. Often, simplifying assumptions are made in practice, such as independence between different factors of variation, leading to weak generative models that fail to capture statistical subtleties.

Recently, the advent of the deep generative models (DGMs) (Goodfellow et al. 2014; Kingma and Welling 2013; Rezende et al. 2014) somehow gathers the three lines of methods mentioned above. Bringing together characteristics from computer graphics, computer vision, and machine learning makes the DGMs a powerful *analysis-by-synthesis* framework. We discuss the DGM-based approaches related to our work in the following section.

2.2 Analysing Humans in Images with DGMs

Generally, in *classical DGMs*, such as standard VAEs and GANs, pose representation is non-interpretable and unsupervised, entangled with the visual appearance in the latent space. This is similarly employed by some *image-to-image translation networks*, however, in contrast to the relatively low-dimensional manifolds learned by the DGMs, in the latter case high-dimensional abstractions are learned and used strictly for direct mapping from and to the image space. On the other hand, *conditional DGMs* usually define part of the abstract data representation, i.e. body pose, to be an interpretable and observable random variable, while the rest of the representation (visual appearance) is kept non-interpretable and latent, still subjected to unsupervised learning. Finally, in *structured DGMs* approaches, as the Semi-DGPose, the latent space can be simultaneously composed by interpretable and non-interpretable random variables. In the former case, the variables may be fully or semi-supervised, while in the latter group they are still maintained unsupervised. Below, we describe related literature gathering the methods according to their adopted type of approach.

Image-to-image networks. Ma et al. (2017) introduce the Pose Guided Person Generation Network (PG^2), a two stage image-to-image translation model which is trained on pairs of images of the same person in different poses, scales and points of view. The authors admit the difficulty of generating poses and detailed appearance simultaneously in an end-to-end fashion. Their model, which is conditioned on images rather than poses, does not allow sampling, thus in its essence, it is not a generative model, which is again in contrast to our single-stage approaches. In a second proposal, Ma et al. (2018) present a GAN-based model for learning image embeddings of foreground, background and pose variables encoded as interpretable variables. The method is still limited to training and testing with cross-pose/scale pairs for pose-transfer, however, it allows sampling, differently from the PG^2. In contrast to our Semi-DGPose model, it is not capable of performing either pose estimation or semi-supervised learning, relying on off-the-shelf pose estimators to perform pose-transfer.

Recently, Esser et al. (2018) present a conditional image-to-image translation network based on the U-Net (Ronneberger et al. 2015). The model is conditioned on an appearance encoding obtained using a VAE architecture. It is more versatile than (Ma et al. 2017, 2018), although still not capable of producing either an interpretable encoding of pose (pose estimation) or performing semi-supervised learning. Similarly, Balakrishnan et al. (2018) also propose a U-Net-based approach. In this case, the authors make use of three U-Nets which tackle foreground segmentation and synthesis, as well as background synthesis. The model is trained with video sequences of the same person performing a limited set of activities. Therefore, it is limited to translating images of the same person to different poses. Other very recent approaches (Chan et al. 2018; Neverova et al. 2018) have to be explicitly trained for pose-transfer, i.e. using images pairs, and do not have the capability of predicting pose. This is in sharp contrast to our Semi-DGPose approach, in which we learn pose estimation, while pose-transfer is achieved as a by-product. In the method by Trumble et al. (2018), pose is estimated from multiple views, although it does not allow semi-supervised learning.

Rhodin et al. (2018) learn 3D pose estimation from multi-view images of the same person acquired from synchronised and calibrated cameras. In contrast to our approach, their method explicitly uses the rotation matrix between cameras during training for the unsupervised learning of a geometry-aware latent representation. From such representation, the 3D pose is estimated posteriorly with a shallow network. The authors do not define their method as a generative model, but as a 3D pose estimator, although it can perform novel viewpoint synthesis. Another work by Zanfir et al. (2018) focus uniquely on the specific task of appearance transfer, also based on 3D pose. In contrast, our closely related task of pose-transfer is just one among all the tasks our DGMs can perform (e.g. sampling, pose estimation, direct manipulation) employing only 2D pose representations. Lastly, Zhang et al. (2018) focus on a slightly different task. They propose the unsupervised discovery of 2D landmarks using optical flows from Human3.6M videos as a short-term self-supervision. Such landmarks are an intermediate representation of pose since they do not correspond explicitly to specific body parts. In contrast, we employ single still images using directly and explicitly interpretable pose representations.

Finally, it is essential to differentiate such image-to-image translation methods from our DGMs. The former depends upon input images at test time, while the latter effectively allow sampling from the latent structured representations learned during training. This subtle difference means that such structured representations are responsible for learning the underlying factors of variations in image generation, without relying on information from input images for generating outputs at test time.

Classical DGMs. Lassner et al. (2017) have proposed the ClothNet-full model, in which a VAE model is used to learn a latent representation of segmentation masks of people in given poses. The reconstructed masks are mapped back to the image space by an image-to-image translation module based on Isola et al. (2016). In contrast, we learn our generative models directly on the raw image data without the need for body parts segmentation. Moreover, pose is interpretable in both of our methods. Siarohin et al. (2018) propose a GAN model with skip connection in the generator and a discriminator conditioned on pose. Similarly to Ma et al. (2017), the model is restricted to pose-transfer on pairs of images of the same person. The body pose is always given to the model and non-interpretable in the learned latent encoding. Apart from this, Walker et al. (2017) proposed a hybrid architecture, associating a VAE and a GAN for forecasting future poses in a video. Here, a low-dimensional pose representation is learned using a VAE, and once the future poses are predicted, they are mapped to images using a GAN generator. Considering GAN based generative models, Tulyakov et al. (2018) present a GAN network that learns motion and content in two separate latent spaces in an unsupervised manner. However, it does not allow explicit manipulation over the human pose.

Conditional DGMs. Lassner et al. (2017) present a second model, the ClothNet-Body, which is a CVAE conditioned on human pose. This model is closely related to our Conditional-DGPose, but it also uses low-dimensional segmentation masks and an auxiliary image-to-image transfer network, based on Isola et al. (2016), to generate realistic images. Pumarola et al. (2018) propose an unsupervised image synthesis based on a conditional GAN method, yet it is also not capable of performing pose prediction.

In summary, there are methods in the literature closely related to our Conditional-DGPose, mainly due to its conditional nature. Although, to our knowledge, no other method gathers the capabilities of our Semi-DGPose as a *structured DGM*. The novelty in the Semi-DGPose largely relies on how the body pose is handled, differing it from related work. Moreover, the capacity for performing pose estimation, indirect pose-transfer, and semi-supervised learning, while aiming for joint *understanding* and *generation* of people in images is peculiar to our model. Following Larsen et al. (2016), we use a discriminator in our training to improve the quality of the generated images. However, in contrast to Larsen et al. (2016), the latent space of our approach is interpretable, which enables us to sample different poses and appearances.

3 Preliminaries

Deep generative models (DGMs) come in two broad flavours—Variational Autoencoders (VAEs) (Kingma and Welling 2013; Rezende et al. 2014), and Generative Adversarial Networks (GANs) (Goodfellow et al. 2014). In both cases, the goal is to learn a generative model $p_\theta(\mathbf{x}, \mathbf{z})$ over data $\mathbf{x}$ and latent variables $\mathbf{z}$, with parameters θ. Typically the model parameters θ are represented in the form of a neural network.

VAEs express an objective to learn the parameters θ that maximise the marginal likelihood (or evidence) of the model denoted as $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})dz$. They introduce a conditional probability density $q_\phi(\mathbf{z}|\mathbf{x})$ as an approximation to the unknown and intractable model posterior $p_\theta(\mathbf{z}|\mathbf{x})$, employing the variational principle in order to optimise a surrogate objective $\mathcal{L}(\phi, \theta; \mathbf{x})$, called the evidence lower bound (ELBO), as

$$\begin{aligned} \log\, p_\theta(\mathbf{x}) &\geq \mathcal{L}_{\text{VAE}}(\phi, \theta; \mathbf{x}) \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]. \end{aligned} \tag{1}$$

The conditional density $q_\phi(\mathbf{z}|\mathbf{x})$ is called the recognition or inference distribution, with parameters ϕ also represented in the form of a neural network. Lastly, VAEs also admit an extension to *conditional* generative models (CVAEs) (Sohn et al. 2015), simply by incorporating a conditioning variable $\mathbf{y}$, to derive

$$\begin{aligned} \log\, p_\theta(\mathbf{x}|\mathbf{y}) &\geq \mathcal{L}_{\text{CVAE}}(\phi, \theta; \mathbf{x}|\mathbf{y}) \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})}\left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z}|\mathbf{y})}{q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})}\right]. \end{aligned} \tag{2}$$

On the other hand, in the context of structured semi-supervised learning, one can factor the latent variables into unstructured or non-interpretable variables $\mathbf{z}$ and structured or interpretable variables $\mathbf{y}$ without loss of generality (Kingma et al. 2014; Siddharth et al. 2017). For learning in this framework, the objective can be expressed as the combination of supervised and unsupervised objectives. Let $\mathcal{D}_u$ and $\mathcal{D}_s$ denote the unlabelled and labelled subset of the dataset $\mathcal{D}$, and let the joint recognition network factorise as $q_\phi(\mathbf{y}, \mathbf{z}|\mathbf{x}) = q_\phi(\mathbf{y}|\mathbf{x})q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})$. Then, the combined objective summed over the entire dataset corresponds to

$$\mathcal{L}_{\text{SS}}(\theta, \phi; \mathcal{D}) = \sum_{\mathbf{x}_u \in \mathcal{D}_u} \mathcal{L}_u(\theta, \phi; \mathbf{x}_u) + \gamma \sum_{(\mathbf{x}_s, \mathbf{y}_s) \in \mathcal{D}_s} \mathcal{L}_s(\theta, \phi; \mathbf{x}_s, \mathbf{y}_s) \tag{3}$$

where $\mathcal{L}_u$ and $\mathcal{L}_s$ are defined as

$$\mathcal{L}_u(\theta, \phi; \mathbf{x}_u) = \mathcal{L}_{\text{VAE}}(\theta, \phi; \mathbf{x}_u), \text{ and} \tag{4}$$

$$\mathcal{L}_s(\theta, \phi; \mathbf{x}_s, \mathbf{y}_s) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_s, \mathbf{y}_s)} \left[\log \frac{p_\theta(\mathbf{x}_s, \mathbf{z}|\mathbf{y}_s)}{q_\phi(\mathbf{z}|\mathbf{x}_s, \mathbf{y}_s)} \right] + \alpha \log q_\phi(\mathbf{y}_s|\mathbf{x}_s), \tag{5}$$

respectively. Here, the hyper-parameter γ (Eq. 3) controls the relative weight between the supervised and unsupervised dataset sizes, and α (Eq. 5) controls the relative weight between generative and discriminative learning.

Note that by the factorisation of the generative model, VAEs necessitate the specification of an explicit likelihood function $p_\theta(\mathbf{x}|\mathbf{z})$, which can often be difficult. GANs, on the other hand, attempt to sidestep this requirement by learning a surrogate to the likelihood function, while avoiding the learning of a recognition distribution. Here, the generative model $p_\theta(\mathbf{x}, \mathbf{z})$, viewed as a mapping $G : \mathbf{z} \mapsto \mathbf{x}$, is setup in a two-player minimax game with a "discriminator" $D : \mathbf{x} \mapsto \{0, 1\}$, whose goal is to correctly identify if a data point $\mathbf{x}$ came from the generative model $p_\theta(\mathbf{x}, \mathbf{z})$ or the true data distribution $p(\mathbf{x})$. Such objective is defined as

$$\mathcal{L}_{\text{GAN}}(D, G) = \mathbb{E}_{p(\mathbf{x})} \left[\log D(\mathbf{x}) \right] + \mathbb{E}_{p_\theta(\mathbf{z})} \left[1 - \log D(G(\mathbf{z})) \right]. \tag{6}$$

In fact, in our structured model, generation is defined as a function of pose and appearance as $G(\mathbf{y}, \mathbf{z})$. Crucially, learning a customised approximation to the likelihood can result in a much higher quality of generated data, particularly for the visual domain (Karras et al. 2018).

A more recent family of DGMs, VAEGANs (Larsen et al. 2016), bring together these two different approaches into a single objective that combines both the VAE and GAN objectives directly as

$$\mathcal{L} = \mathcal{L}_{\text{VAE}} + \mathcal{L}_{\text{GAN}}. \tag{7}$$

This marries better the likelihood learning with the inference-distribution learning, providing a more flexible family of models.

4 Our Approach

As set out in the preliminaries (Sect. 3), we use the VAEGAN framework as the basis for our generative models (Larsen et al. 2016). Note that, in incorporating semi-supervised learning, the semi-supervised VAEGAN includes two distinct tasks. First, it involves learning a recognition network that can estimate pose $\mathbf{y}$ and *appearance* $\mathbf{z}$ for any given RGB image $\mathbf{x}$. Second, it involves learning a generative network that combines a given pose with an appearance to generate visual data (RGB image) corresponding to those variables.

From discriminative modelling, we know that the first task, i.e. predicting pose, is eminently plausible up to learning an appearance model. However, learning the full generative model is something that can be fraught with difficulties. For one, pose and appearance can exhibit a large degree of information imbalance—pose can be distilled into a set of (x, y) coordinates, whereas appearance can encode a vast swathe of information (e.g. texture, colour, shapes) about the given input.

Given a generative model that takes both appearance $\mathbf{z}$ and pose $\mathbf{y}$ as inputs to produce an RGB image $\mathbf{x}$, a reasonable first step can be just to evaluate the performance of a conditional generative model, where the conditioning variable is taken to be the interpretable pose $\mathbf{y}$. We refer to this setup as Conditional-DGPose, with reference to the fact that it is a conditional-VAEGAN model. Its lower bound is given by Eq. 2, and its final objective function is defined as

$$\mathcal{L} = \mathcal{L}_{\text{CVAE}} + \mathcal{L}_{\text{GAN}}, \tag{8}$$

in contrast to the standard VAEGAN objective (Eq. 7). Here, all data is "labelled" with pose, but the goals were: (i) primarily, to verify qualitatively if a low-dimensional conditioning variable would affect the conditional generative model; (ii) secondly, to evaluate the accuracy of the reconstructed images quantitatively w.r.t. the human body poses and the image quality.

Once verified through experiments that the conditional approach works, we could then proceed towards our structured semi-supervised VAEGAN, referred to as Semi-DGPose, as its main difference from the previous setup is that the encoding distribution is no longer conditioned on the pose, but instead predicts it as per Eq. 3–6. In contrast to the standard VAEGAN objective (Eq. 7), the structured semi-supervised VAEGAN final objective function is given by,

$$\mathcal{L} = \mathcal{L}_{\text{SS}} + \mathcal{L}_{\text{GAN}}. \tag{9}$$

We describe the details and implementations of our models in the rest of this section. Next, we start defining the adopted pose representations, which are common for both, the Conditional-DGPose and the Semi-DGPose architectures.

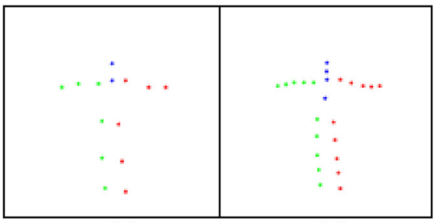


Fig. 2 Vector representation. (**a**) $J = 14$ joints which compose a 2D pose vector $\mathbf{y}_v \in \mathcal{R}^{2J}$. (**b**) An extended 2D vector composed by 24 body parts ($J = 14$ annotated joints, $R = 9$ intermediate points between joints and $B = 1$ central point), such that $\mathbf{y}_v \in \mathcal{R}^{2(J+R+B)}$

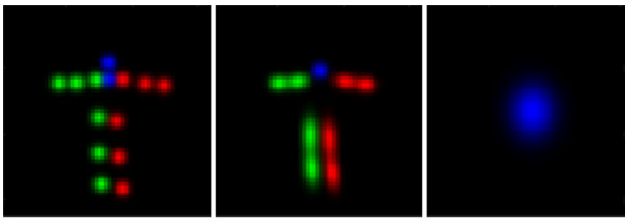


Fig. 3 Heatmap representation. Heatmaps superimposed corresponding to (**a**) $J = 14$ annotated joints, (**b**) $R = 9$ rigid parts, and (**c**) $B = 1$ whole body; such that $\mathbf{y}_h \in \mathcal{R}^{P\times H\times W}$. Right, left and central body parts are denoted by the colours green, blue and red, respectively, in the person-centric representation (Color figure online)

4.1 Pose Representation

In our DGMs, the random variable $\mathbf{y}$ corresponds to an abstraction of the human body pose. Therefore a suitable concrete representation must be adopted in the implementation of the models. As mentioned in our literature review, many methods which define a generative model in the *pose space* would simply encode J joints defining the body as a vector $\mathbf{y}_v$, such that $\mathbf{y}_v \in \mathcal{R}^{2J}$. Others employ extended versions of it, in which positions of R rigid parts and B whole body are derived from the annotated joints (Yang and Ramanan 2011), such that $\mathbf{y}_v \in \mathcal{R}^{2(J+R+B)}$. Both cases are illustrated in Fig. 2.

On the other hand, the mapping of 2D joints positions to heatmaps has shown to be very effective in several pose estimation approaches (Chu et al. 2017; Newell et al. 2016; Tompson et al. 2014; Wei et al. 2016). The Gaussian heatmaps represent the underlying probability distribution of body parts' locations. In our method, the heatmap representation $\mathbf{y}_h$ consists of P body elements, in a way that $\mathbf{y}_h \in \mathcal{R}^{P\times H\times W}$, where H and W are the heatmap height and width, respectively. In the simplest case $P = J$, however, as the set of joints is reasonably sparse, to cover the entire area of the bodies, joints, rigid parts and the whole body might be used as an extended case, in which $P = J + R + B$ (de Bem et al. 2018), as illustrated in Fig. 3. In this way, each body element p is represented using a 2D Gaussian around its centre $\boldsymbol{\mu}_p = (i_p, j_p)$, with diagonal covariance matrix $\Sigma_p = R_p \begin{bmatrix} \sigma_{p,i}^2 & 0 \\ 0 & \sigma_{p,j}^2 \end{bmatrix} R_p^\top$, computed as follows:

Joints. Since joints have a limited spatial extent, we follow previous approaches (Chu et al. 2017; Newell et al. 2016; Tompson et al. 2014; Wei et al. 2016) in modelling them as isotropic Gaussians that are centred at the ground-truth joint location and have a small standard deviation (e.g. $\sigma_{p,i} = \sigma_{p,j}$ = 1.5 pixel for a 64 × 64 heatmap).

Rigid Parts. The centre $\boldsymbol{\mu}_p$ of a rigid part p is defined as the mean point of the centres $\boldsymbol{\mu}_k$ and $\boldsymbol{\mu}_l$ of the joints it connects. We orient the Gaussian representing the rigid part to align its i axis with the line connecting $\boldsymbol{\mu}_k$ and $\boldsymbol{\mu}_l$. We define $\sigma_{p,i}$ to be proportional to $|\boldsymbol{\mu}_k - \boldsymbol{\mu}_l|$, and set $\sigma_{p,j} = \kappa_p \sigma_{p,i}$, where κ_p is a part-specific ratio, inspired by anthropometric measurements (NASA 1995).

Body. The body centre is defined to be the mean of the annotated joint centres. Principal component analysis (PCA) of the joint centres is used to obtain the orientation of the body in the image plane. We define $\sigma_{p,i}$ and $\sigma_{p,j}$ to be proportional to the distance between the extreme projections of the joint centres onto, respectively, the principal and secondary axes of variation.

In our both models, as detailed in the next sections, we make use of both forms of pose representation, taking advantage of their particular characteristics in each case. In the Conditional-DGPose, only the heatmap representation $\mathbf{y}_h$ is employed, since, as shown later in our experiments, it can be seamlessly concatenated to feature maps, helping on the generation of accurate output images. On the other hand, in the Semi-DGPose model, we additionally employ the vector-based form $\mathbf{y}_v$, as a way of maintaining a low-dimensional latent representation of pose.

4.2 DGPose Architectures

We have tested several variations of deep CNN architectures for implementing our models, culminating in our best performing ones, which are described here. All its modules are deep CNNs, and full implementation definitions are given in Appendix A and referred adequately in the text. Due to the generality of generative models, the architectures may be employed in different ways according to the aimed tasks. Thus, we describe separately training and test phases, dividing the latter into *reconstruction*, *pose-transfer*, *sampling* and *pose-estimation*, for both models. Thus, the Conditional-DGPose and the Semi-DGPose are described following.

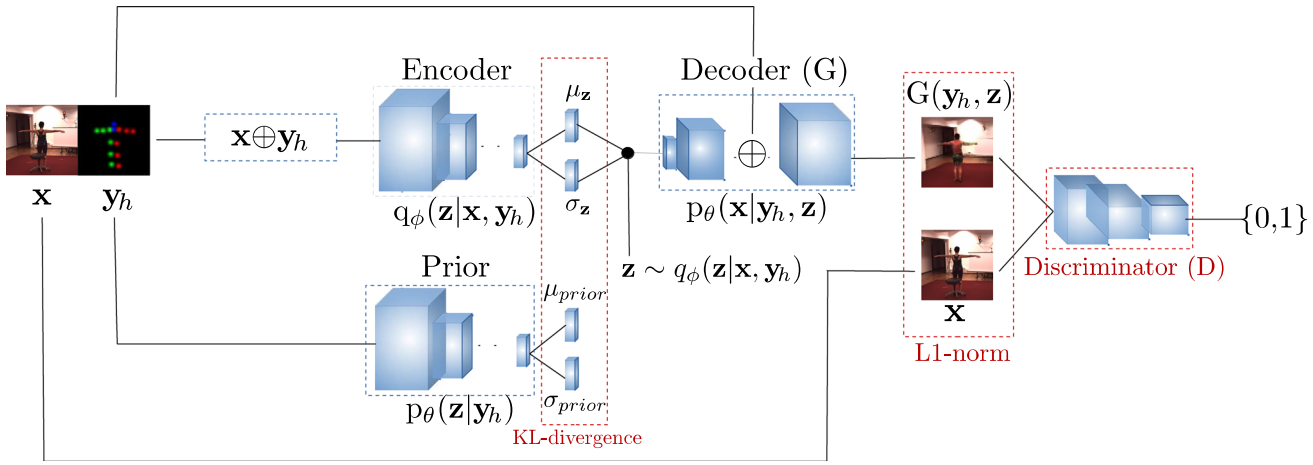


Fig. 4 Conditional-DGPose architecture. At the training, the Encoder receives $\mathbf{x} \oplus \mathbf{y}_h$ as input and learns the posterior $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y}_h)$. The Prior module receives $\mathbf{y}_h$ alone and learns the distribution $p_\theta(\mathbf{z}|\mathbf{y}_h)$. Appearance is sampled $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y}_h)$, using the reparametrization trick (Kingma and Welling 2013), and passed to the Decoder, as well as the conditioning pose $\mathbf{y}_h$, which is concatenated to the Decoder feature maps. The Decoder then generates a reconstructed image $G(\mathbf{y}_h, \mathbf{z})$. The loss function (see Eq. 8, Sect. 4) is composed by the following terms, highlighted in red: the L1-norm $L1(\mathbf{x}, G(\mathbf{y}_h, \mathbf{z}))$ which is computed between the original and the reconstructed image; the KL-divergence $\mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y}_h)||p_\theta(\mathbf{z}|\mathbf{y}_h)]$, which is used to regularise the posterior distribution; and the GAN Discriminator cross-entropy loss used to learn how to discern between real and generated images (Color figure online)

4.2.1 Conditional-DGPose

Our conditional-VAEGAN model learns the parameters of four deep CNN networks simultaneously: (i) a recognition network (Encoder), which estimates appearance $\mathbf{z}$ conditioned to pose $\mathbf{y}_h$ and to a given RGB image $\mathbf{x}$; (ii) a Prior network, which estimates appearance $\mathbf{z}$ conditioned to pose $\mathbf{y}_h$ alone; (iii) a generative network (Decoder), which combines appearance $\mathbf{z}$ and the conditioning pose $\mathbf{y}_h$, to generate corresponding RGB images $G(\mathbf{y}_h, \mathbf{z})$; and (iv) a Discriminator network, which differentiates between real images $\mathbf{x}$ and generated images $G(\mathbf{y}_h, \mathbf{z})$. Learning is pursued by the minimisation of the loss function $\mathcal{L} = \mathcal{L}_{\text{CVAE}} + \mathcal{L}_{\text{GAN}}$ (Eq. 8, Sect. 4), composed by the CVAE evidence lower bound (ELBO) $\mathcal{L}_{\text{CVAE}}$ and by the GAN cross-entropy discriminator loss $\mathcal{L}_{\text{GAN}}$. An overview of our model is shown in Fig. 4 and implementation details are provided in Table 6 (Appendix A). Below, we describe further the training and the test phases, dividing the latter into *reconstruction*, *pose-transfer* and *sampling*.

Training. Given an image $\mathbf{x}$, the corresponding heatmap labels (conditioning pose) are concatenated to it as per $\mathbf{x} \oplus \mathbf{y}_h$ (Encoder, Layer 1, Table 6). Then, the Encoder estimates the conditional posterior distribution $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y}_h)$. The heatmap labels $\mathbf{y}_h$ alone are the input of the Prior module, which estimates the distribution $p_\theta(\mathbf{z}|\mathbf{y}_h)$. Appearance is sampled from the posterior $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y}_h)$, using the reparametrisation trick (Kingma and Welling 2013). The sample $\mathbf{z}$, along with the conditioning pose $\mathbf{y}_h$ (Decoder, Layer 7, Table 6), are passed through the Decoder which generates a reconstructed image $G(\mathbf{y}_h, \mathbf{z})$. This reconstructed image, along with the real image $\mathbf{x}$, are still used as inputs for the Discriminator module, which learns how to discern between them. Finally, the overall loss function minimised during training is composed of the L1-norm reconstruction loss $L1(\mathbf{x}, G(\mathbf{y}_h, \mathbf{z}))$; the KL-divergence, which acts as a regulariser, between the posterior and the prior distributions, $\mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y}_h)|p_\theta(\mathbf{z}|\mathbf{y}_h)]$; and the cross-entropy Discriminator loss (Eq. 6, Sect. 3).

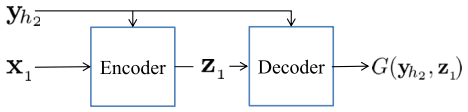


Fig. 5 Conditional-DGPose direct pose-transfer and manipulation at test time

Reconstruction and Direct Pose-transfer. At test time, when an image $\mathbf{x}_1$ and its corresponding pose $\mathbf{y}_{h_1}$ are given as input, the reconstructed image $G(\mathbf{y}_{h_1}, \mathbf{z}_1)$ is obtained as the Decoder output. However, if $\mathbf{x}_1$ is used as input along with a different pose $\mathbf{y}_{h_2}$, the person in the reconstructed image $G(\mathbf{y}_{h_2}, \mathbf{z}_1)$ will keep the appearance of $\mathbf{x}_1$, with the body pose defined by $\mathbf{y}_{h_2}$, as illustrated in Fig. 5. Similarly, as shown later in our experiments, the same procedure may be adopted to *directly manipulate* the reconstructed image, such as changing body size and aspect ratio, moving or suppressing body parts or even hallucinating multiple people.

Sampling. At test time, sampling is obtained when no RGB image is given as input. In this case, as illustrated in Fig. 6, only a conditioning pose $\mathbf{y}_h$ is given as the input of the Prior module, which defines $p_\theta(\mathbf{z}|\mathbf{y}_h)$. From this Prior distribution,

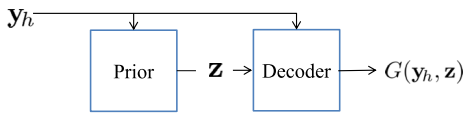


Fig. 6 Conditional-DGPose sampling at test time

the sampled appearance $\mathbf{z}$ and the conditioning pose $\mathbf{y}_h$ are passed to the Decoder network. In this manner, for a given pose, different appearances can be randomly created from the learned generative model.

4.2.2 Semi-DGPose

Differently from the Conditional-DGPose, our structured semi-supervised VAEGAN model (Fig. 7) learns the parameters of three deep CNN networks simultaneously: (i) a recognition network (Encoder), which estimates appearance $\mathbf{z}$ and pose $\mathbf{y}_v$ from a given RGB image $\mathbf{x}$; (ii) a generative network (Decoder), which combines appearance $\mathbf{z}$ and pose $\mathbf{y}_v$, to generate corresponding RGB images $G(\mathbf{y}_v, \mathbf{z})$; and (iii) a Discriminator network, which differentiates between real images $\mathbf{x}$ and generated images $G(\mathbf{y}_v, \mathbf{z})$. Learning is pursued by the minimisation of the loss function $\mathcal{L} = \mathcal{L}_{\text{SS}} + \mathcal{L}_{\text{GAN}}$ (Eq. 9, Sect. 4), composed by the structured semi-supervised VAE evidence lower bound (ELBO) $\mathcal{L}_{\text{SS}}$ and by the GAN cross-entropy discriminator loss $\mathcal{L}_{\text{GAN}}$. A fourth module, called Mapper, is introduced by us to overcome a peculiarity caused by the inclusion of pose in the latent space. Such a module, trained separately, is described next.

The Mapper Module. Our preliminary experiments with the Conditional-DGPose showed that heatmaps led to better quality reconstructions, in contrast to the vector-based representation. On the other hand, a low-dimensional representation is more suitable and desirable as a latent variable, since human pose lies in a low-dimensional manifold embedded in the high-dimensional image space (Elgammal and Lee 2004; Goodfellow et al. 2016). To cope with this mismatch, we introduce the Mapper module, which maps pose-vectors $\mathbf{y}_v$ to heatmaps $\mathbf{y}_h$. Ground-truth heatmaps are constructed from manually annotated 2D joints labels, using a simple weak annotation strategy (de Bem et al. 2018). The Mapper module is then trained to map 2D joints to heatmaps, minimising the L2-norm between predicted and ground-truth heatmaps. This module is trained separately with the same training hyper-parameters used for our full architecture, described later in Sect. 5.5. In the training of the full Semi-DGPose architecture, the Mapper module is integrated to it with its weights kept fixed, since the mapping function has been learned already. The Mapper allows us to keep a low-dimensional representation $\mathbf{y}_v$ in the latent space, at the same time that a dense high-dimensional "spatial" heatmap representation $\mathbf{y}_h$ facilitates the generation of accurate images by the Decoder. As it is fully differentiable, the module allows the gradients to be backpropagated normally from the Decoder to the Encoder, when it is required during the training of the full architecture.

In the rest of this section, we describe further the training and the test phases, dividing the latter into *reconstruction*, *indirect pose-transfer*, *sampling* and *pose estimation*. An overview of our model is shown in Fig. 7 and implementation details are provided in Table 7 (Appendix A).

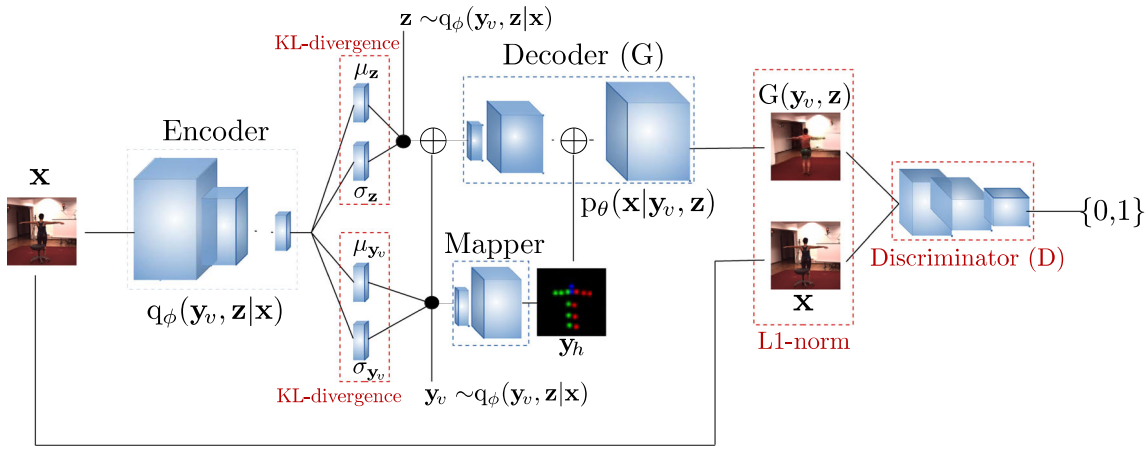


Fig. 7 Semi-DGPose architecture. At the training, the Encoder receives $\mathbf{x}$ as input and learns the posterior distribution $q_\phi(\mathbf{y}_v, \mathbf{z}|\mathbf{x})$. In the *unsupervised* routine, samples of appearance $\mathbf{z}$ and pose $\mathbf{y}_v$ are obtained using the reparametrisation trick (Kingma and Welling 2013). These samples are passed to the Decoder, which generates a reconstructed image $G(\mathbf{y}_v, \mathbf{z})$. The unsupervised loss function is composed by the following terms, highlighted in red: the L1-norm $L1(\mathbf{x}, G(\mathbf{y}_v, \mathbf{z}))$ between the original and the reconstructed images; the KL-divergence losses between the posterior distribution $q_\phi(\mathbf{y}_v, \mathbf{z}|\mathbf{x})$ and the weak priors $p(\mathbf{y}_v)$ and $p(\mathbf{z})$, which work as regularisers (see Eq. 4, Sect. 3); and the cross-entropy Discriminator loss (Eq. 6, Sect. 3). In the *supervised* routine (not shown above for simplicity), the only difference is that a regression loss between the estimated pose and the pose ground-truth label substitutes the KL-divergence over the pose posterior distribution (see Eq. 5, Sect. 3). In both, supervised and unsupervised training routines, the low-dimensional pose vector $\mathbf{y}_v$ is mapped to a heatmap representation $\mathbf{y}_h$ by the Mapper module and concatenated to the Decoder. Eq. 3 (Sect. 3) shows the overall loss function

Training. The terms of Eq. 3 (Sect. 3) correspond to two training routines which are alternately employed, according to the presence or absence of ground-truth labels.

In the *unsupervised case*, when no label is available, it is similar to the standard VAE (see Eq. 4, Sect. 3). Accurately, given the image $\mathbf{x}$, the Encoder estimates the posterior distribution $q_\phi(\mathbf{y}_v, \mathbf{z}|\mathbf{x})$, where both appearance $\mathbf{z}$ and pose $\mathbf{y}_v$ are assumed to be independent given the image $\mathbf{x}$. Then, pose $\mathbf{y}_v$ and appearance $\mathbf{z}$ are sampled from the posterior, using the reparametrization trick (Kingma and Welling 2013), and passed to the Decoder to generate a reconstructed image. Finally, the unsupervised loss function minimised during training is composed of the L1-norm reconstruction loss $L1(\mathbf{x}, G(\mathbf{y}_v, \mathbf{z}))$; the KL-divergences, which act as regularisers, between the posterior and the prior distributions, $\mathrm{KL}[q_\phi(\mathbf{y}_v|\mathbf{x})|p(\mathbf{y}_v)]$ and $\mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x})|p(\mathbf{z})]$; and the cross-entropy Discriminator loss (Eq. 6, Sect. 3).

In the *supervised case*, when the pose label is available, the KL-divergence between the posterior pose distribution and the pose prior, $\mathrm{KL}[q_\phi(\mathbf{y}_v|\mathbf{x})|p(\mathbf{y}_v)]$, is replaced with a regression loss between the estimated pose and the given label (see Eq. 5, Sect. 3). Now, only the appearance $\mathbf{z}$ is sampled from the posterior distribution and passed to the Decoder, along with the ground-truth pose label. Finally, the supervised loss function minimised during training is composed of the L1-norm reconstruction loss, the KL-divergence over the appearance distribution, the regression loss over the pose vector, and the cross-entropy Discriminator loss. In this case, gradients are not backpropagated from the Decoder to the Encoder, through the pose posterior distribution, since pose was not estimated.

In both *unsupervised* and *supervised* cases, the Mapper module, which is trained *offline*, is used to map the pose-vector $\mathbf{y}_v$ in the latent space to a dense heatmap representation $\mathbf{y}_h$, as illustrated in Fig. 7.

Reconstruction. At test time, only an image $\mathbf{x}$ is given as input, and the reconstructed image $G(\mathbf{y}_v, \mathbf{z})$ is obtained from the Decoder, as illustrated in Fig. 8. In the reconstruction process, *direct manipulation* of the pose representation $\mathbf{y}_v$ allows image generations with varying body poses and sizes while the appearance is kept the same.

Indirect Pose-transfer. Our method allows us to do *indirect pose-transfer* without specific training for such a task. As illustrated in Fig. 9, an image $\mathbf{x}_1$ is first passed through the Encoder network, from which the target pose $\mathbf{y}_{v_1}$ is estimated and kept. In the second step, another image $\mathbf{x}_2$ is propagated through the Encoder, from which the appearance encoding $\mathbf{z}_2$ is kept. Finally, $\mathbf{z}_2$ and $\mathbf{y}_{v_1}$ are jointly propagated through the Decoder, and an image $\mathbf{x}_3$ is reconstructed, containing a person in the pose $\mathbf{y}_{v_1}$ estimated from the first image, but with the appearance $\mathbf{z}_2$ defined by the second image. This is a novel application that our approach enables. In contrast to the prior art, our network neither relies on any external pose estimator nor on conditioning labels to perform pose-transfer.

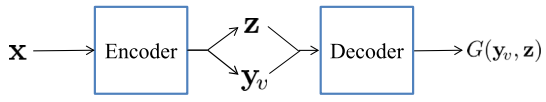


Fig. 8 Semi-DGPose reconstruction at test time

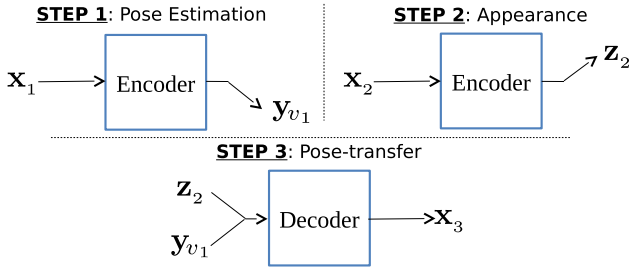


Fig. 9 Semi-DGPose indirect pose-transfer at test time

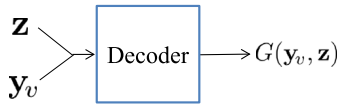


Fig. 10 Semi-DGPose sampling at test time

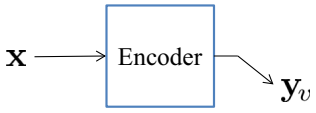


Fig. 11 Semi-DGPose pose estimation at test time

Sampling. When no image is given as input, we can jointly or separately sample pose $\mathbf{y}_v$ and appearance $\mathbf{z}$ from the posterior distribution. They may be sampled at the same time, or one may be kept fixed while the other distribution is sampled. In all cases, the encodings are passed through the Decoder network to generate a corresponding RGB image, as illustrated in Fig. 10.

Pose Estimation. One of the main differences between our approach and the prior art is the ability of our model to estimate human-body pose as well. In this case, as illustrated in Fig. 11, given an input image $\mathbf{x}$, it is possible to perform pose estimation by regressing to the pose representation vector $\mathbf{y}_v$. Thus, the appearance encoding $\mathbf{z}$ is disregarded, and the Decoder, Mapper, and Discriminator networks are not used.

5 Experiments and Results

We have performed a large number of experiments to evaluate our models. In this section, we present the datasets, metrics, and training hyper-parameters used in our work. Finally, quantitative and qualitative results show the effectiveness and novelty of our Conditional-DGPose and Semi-DGPose architectures.

5.1 Human3.6M Dataset

Human3.6M (Ionescu et al. 2014) is a widely used benchmark for human body analysis. It contains 3.6 million images acquired by recording 5 female and 6 male actors performing a diverse set of motions and poses corresponding to 15 activities, under 4 different viewpoints. We followed the standard protocol and used sequences of 2 out of 11 actors as our test set, while the rest of the data was used for training. We use a subset of 14 (out of 32) body joints represented by their (x, y) 2D image coordinates as our ground-truth data, neglecting minor body parts (e.g. fingers). Due to the high frequency of video acquisition (50 Hz), there is a considerable level of practically redundant images. Thus, out of images from all 4 cameras, we subsample frames in time, producing subsets for training and testing, with 317,989 and 1280 images, respectively. All the original images have a resolution of 1000×1000 pixels.

5.2 ChictopiaPlus Dataset

ChictopiaPlus (Lassner et al. 2017) is an extension of the Chictopia dataset (Liang et al. 2015). It augments the original per-pixel annotations for body parts with pose annotation (Insafutdinov et al. 2016), 3D shape (Loper et al. 2015), and facial segmentation. In contrast to the Human3.6M dataset, in which each actor always wears the same outfit, it contains 23,011 training, 2913 validation, and 2873 testing images of segmented people (without background) dressed in a great variety of clothes. All the images have an original resolution of 286×286 pixels.

5.3 DeepFashion Dataset

The DeepFashion dataset (In-shop Clothes Retrieval Benchmark) (Liu et al. 2016) consists of 52,712 images of people in a variety of clothing and poses. We follow Ma et al. (2017), using their joints' annotations obtained with an off-the-shelf pose estimator (Cao et al. 2017), and divide the dataset into training (44,950 images) and testing (6560 images) subsets. Images with wrong pose estimations were suppressed and all original images have 256×256 pixels. Importantly, we aim to learn a complete generative model of people in images, which is significantly more complex, compared to models focusing on a particular task, such as pose-transfer. For this reason, we use images individually in our training set, instead of employing pairs of images of the same person as in Ma et al. (2017) and Siarohin et al. (2018).

5.4 Metrics

Quantitative evaluation of generative models is inherently difficult (Theis et al. 2016). Since our models explicitly represent *appearance* and *body pose* as separate variables, we evaluate their performance w.r.t. three different aspects. (i) **Image quality** of reconstructions is evaluated using the standard Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) metrics (Wang et al. 2004). (ii) **Accuracy of the reconstructed poses** is evaluated using a protocol introduced by us as follow. To set a common ground for comparing an original test set, with a reconstructed one, we start using a well-established (discriminative) human pose estimator (Newell et al. 2016), and initially estimating all 2D poses in the original test set. In our protocol, we assume that such estimations are the *ground-truth* poses of the test set. Subsequently, we apply the same discriminative estimator over the reconstructed test images, produced by the trained generative models. Finally, we use the Percentage of Correct Keypoints (PCK) metric (Yang and Ramanan 2011), which computes the percentage of 2D joints correctly located by a pose estimator, given the *ground-truth* and a normalised distance threshold corresponding to the size of the person's torso. Thus, we assume that any degradation in the PCK metric is caused by imperfections on the reconstructed images, since a PCK score of 100% would correspond to having all the estimated joints, in the original and the reconstructed images, at the same locations, up to the distance threshold. We illustrate this metric in Fig. 12. (iii) **Accuracy of pose estimation**, obtained by the Semi-DGPose model, is measured using the PCK metric with *real* 2D annotated labels as ground-truths.

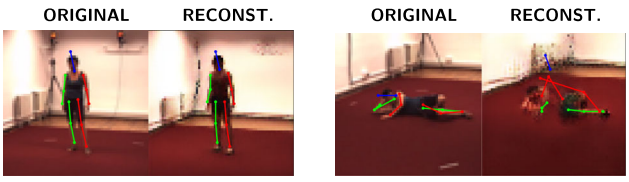


Fig. 12 Accuracy of the reconstructed poses. Samples illustrating best and worst pose reconstructions on the Human3.6M dataset. Each pair of images shows the pose estimation over the original image (left) and the reconstructed image (right). Lines connect the estimated joints for visualisation purposes. Right limbs, left limbs, and head are shown, respectively, by green, red and blue lines. (**a**) It illustrates the best reconstructed poses, with PCK@0.5 = 1.00. (**b**) It illustrates the worst reconstructed poses, with PCK@0.5 = 0.00. All images are 64×64 pixels (Color figure online)

5.5 Training

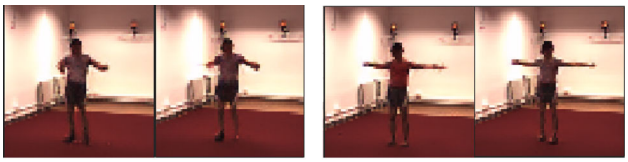
All models were trained with mini-batches consisting of 64 images. We used the Adam optimiser (Kingma and Ba 2015) with an initial learning rate set to 10^{-4}. The weight decay regulariser was set to 5×10^{-4}. Network weights were initialised randomly for fully-connected layers and with robust initialisation (He et al. 2015) for convolutional and transposed-convolutional layers. Except when stated differently, for all images and all models, we used a 64×64 pixels crop, centring the person of interest. We did not use any form of data augmentation or preprocessing except for image normalisation to zero mean and unit variance. All models were implemented in Caffe (Jia et al. 2014), and all experiments ran on an NVIDIA Titan X GPU.

5.6 Conditional-DGPose

As mentioned earlier (Sect. 4), the Conditional-DGPose is taken by us as an intermediate step in the investigation towards our Semi-DGPose model. To better evaluate and understand its capabilities, we start our experiments by validating it qualitatively with the Human3.6M benchmark, since this dataset is composed of images in a controlled environment. Initially, in Sect. 5.6.1, we evaluate different pose representations, with the best performance presented by the heatmap representation. In Sect. 5.6.2, we show the effectiveness of the Conditional-DPGose architecture, illustrating *reconstruction* and *sampling* tasks. Besides that, we particularly stress the effects of pose manipulation, by performing *pose-transfer* and *hallucinating* multiple people in a variety of unseen or even unrealistic poses, still on the Human3.6M dataset. After that, we present qualitative and quantitative results on the ChictopiaPlus dataset (Lassner et al. 2017). The Conditional-DGPose outperforms the closest related comparable baseline, the ClothNetBody (Lassner et al. 2017), achieving state-of-the-art results on the ChictopiaPlus. Finally, qualitative and quantitative experiments on the DeepFashion dataset (Liu et al. 2016) are shown. On this dataset, our baseline is the image-to-image translation architecture by Ma et al. (2017), which is trained on pairs of images showing the same person in different poses. Although our Conditional-DGPose method tackles a significantly more complex problem, i.e. learning a generative model and its latent representation in the high-dimensional image space, instead of mapping one image to another, it presents reasonable results in comparison with the ones from Ma et al. (2017).

5.6.1 Pose Representation

We perform experiments with the two pose representations mentioned in Sect. 4.1 and with their respective extensions.



(a) **(b)** **(c)** **(d)**

Fig. 13 Reconstructed images, obtained with each one of the four representations of human pose evaluated: (**a**) 2D vector, (**b**) 2D vector extended, (**c**) heatmaps and (**d**) heatmaps extended. We highlight the difficult for capturing the spatial extent of some body parts, particularly extremities far from the torso, when the vector representations are adopted. In this example, the use of joints' heatmaps is already sufficient to improve the reconstruction. However, the extended version (with rigid parts and body) turns the model more robust to more complex poses, since the 14 joints are fairly sparse

Table 1 Average reconstruction errors obtained with the Conditional-DGPose architecture using L1-norm for our validation set

Pose representation	L1-Norm
2D vector (14 joints)	14.52
2D vector extended (28 joints)	13.91
Heatmaps (14 joints)	13.55
Heatmaps extended	
⌞ (14 joints + 9 rigid parts + 1 whole body)	**13.41**

Best result is shown in bold

We executed end-to-end training with the Conditional-DGPose architecture, which converged in approximately 15 epochs. The qualitative evaluation was performed by the inspection of the reconstructed images, shown in Fig. 13. As can be observed, the vector representations, even the extended one, fail to capture some parts of the body. This problem is particularly evident concerning the extremities of the limbs. On the other hand, the additional heatmaps for rigid parts and whole body have shown a positive impact in the reconstructions. The quantitative measurements, shown in Table 1, support our qualitative evaluation. In all experiments, the heatmaps had the same dimension of the images (64×64).

5.6.2 Conditional-DGPose Results on Human3.6M

Initially, in Fig. 14, we show our heatmap pose representation along with reconstructions, to demonstrate that realistic images with accurate poses can be generated. Furthermore, we illustrate *sampling* in Fig. 15, in which the separation between pose and appearance is made evident by the independent change of each variable.

Next, we stress the pose-transfer and compositionality capabilities of the model, pushing it beyond what is usually done in related methods. Regarding *pose-transfer*, we demonstrate the capability of our model to learn pose and appearance as separate variables which allows direct control over the two at test time. To this end, we generate images

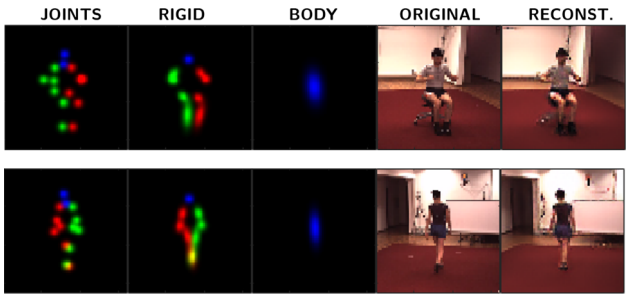


Fig. 14 Reconstructions on Human3.6M. From the left to right columns we have: joints, rigid parts and body heatmaps; original image and finally, the reconstructed image. In the heatmaps, right parts are shown in green, left parts in red and central parts in blue. Human3.6M images are 64 × 64 pixels (Color figure online)



Fig. 15 Sampling on Human3.6M. Results obtained by randomly changing pose and appearance independently

in which we maintain the appearance of the input image, yet the generated person is "moved" into the required target pose. The target pose may be composed manually, extracted from another image with an off-the-shelf pose estimator or provided interactively by a user. This is illustrated in Fig. 16, in which we employ target poses from the LSP dataset (Johnson and Everingham 2010), that have completely different poses in a drastically different environment compared to our training set. The quality of the generations shows that our generative model could disentangle pose and appearance and generate images with poses that do not exist in the training data.

Concerning manipulation, we show in Fig. 17 how our model can be used to "compose" images that have never been seen in the training data. For instance, we can generate images with multiple people in the same (replicated) pose simply by conditioning on a respective heatmap. In fact, we can go one step further and generate an image where all people are in the same pose, but one of them is, e.g. *shorter* and another *thinner*, as shown in Fig. 18a. In an extreme case, we can even generate "unreal" images containing only certain body parts (e.g. heads) or disconnecting them from the rest of the body, as in Figs. 18b and 18c, respectively. Note that the training dataset is composed of only single person images. Thus the model has never seen an image with multiple people or only some separate body parts. This demonstrates that the learned latent space of our model is indeed disentangled. To the best of our knowledge, this capability has not been demonstrated by any other work in the literature.



Fig. 16 Cross-domain pose-transfer on Human3.6M. Here we illustrate the *pose-transfer* capability of our Conditional-DGPose. On the leftmost column, we show test images from the LSP dataset (Johnson and Everingham 2010), along with their corresponding ground-truth 2D pose annotations, composed of 14 joints. These are taken as conditioners (*target-poses*) on our model for the generation of the reconstructions, shown from the third to the rightmost column. As can be observed, the *target-poses* are transferred to the output images, while the latter maintain their original appearances. We highlight the fact that neither the LSP images nor their poses were part of the training set

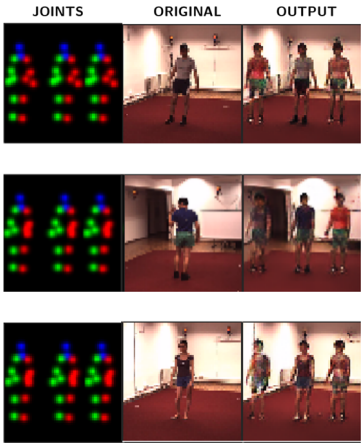


Fig. 17 Hallucinating multiple people on Human3.6M. The Conditional-DGPose model was trained with images containing only one person. The output images are generated keeping the appearance of the original images but conditioned to the manipulated heatmap pose representation (left). Heatmaps of rigid parts and whole body are not shown for simplicity

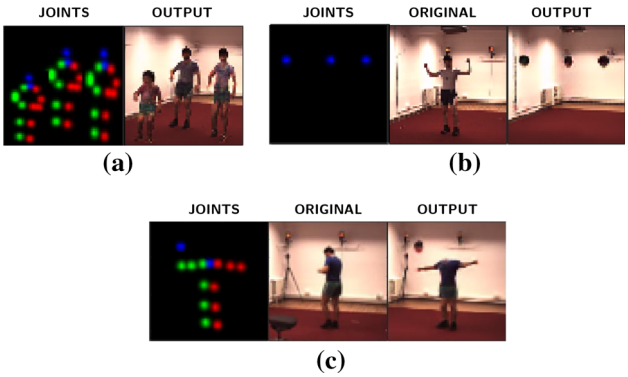


Fig. 18 Generating "unreal" images on Human3.6M. We illustrate the versatility of the model extrapolating the generation of images to unseen scenes. (**a**) Sampled image in which the pose representation in the centre was manually translated and scaled, producing two additional bodies: one shorter and chunkier (left) and one taller and thinner (right). (**b**) Reconstructed image in which all the body parts were suppressed, except the head. (**c**) Pose-transfer in which the position of the head was manually changed, disconnecting it from the rest of the body. Heatmaps of rigid parts and whole body are not shown for simplicity

5.6.3 Conditional-DGPose Results on ChictopiaPlus

We compare our method with Lassner et al. (2017), the closest related work from the literature. We employ the PSNR and the SSIM metrics to evaluate image quality, and the PCK metric to provide a quantitative evaluation of pose reconstructions, as described previously (see Sect. 5.4). In Table 2, we initially show that our method outperforms the ClothNet-body network (Lassner et al. 2017) regarding both, the PSNR and the SSIM metrics. Qualitative results are shown in Fig. 19. Moreover, our model reports 95.14% of accuracy, with PCK score at 0.5, and again outperforms (Lassner et al. 2017) by a large margin, which reports 70.89%. The overall PCK curve is shown in Fig. 20. Our results demonstrate the good quality of our reconstructions w.r.t. image quality and the human pose. The better performance, in comparison with Lassner et al. (2017), can be particularly noticed in the extremities of body limbs, which we hypothesise as a benefit of the single stage end-to-end Conditional-DGPose model, in contrast to the multiple stages of training and testing in Lassner et al. (2017).

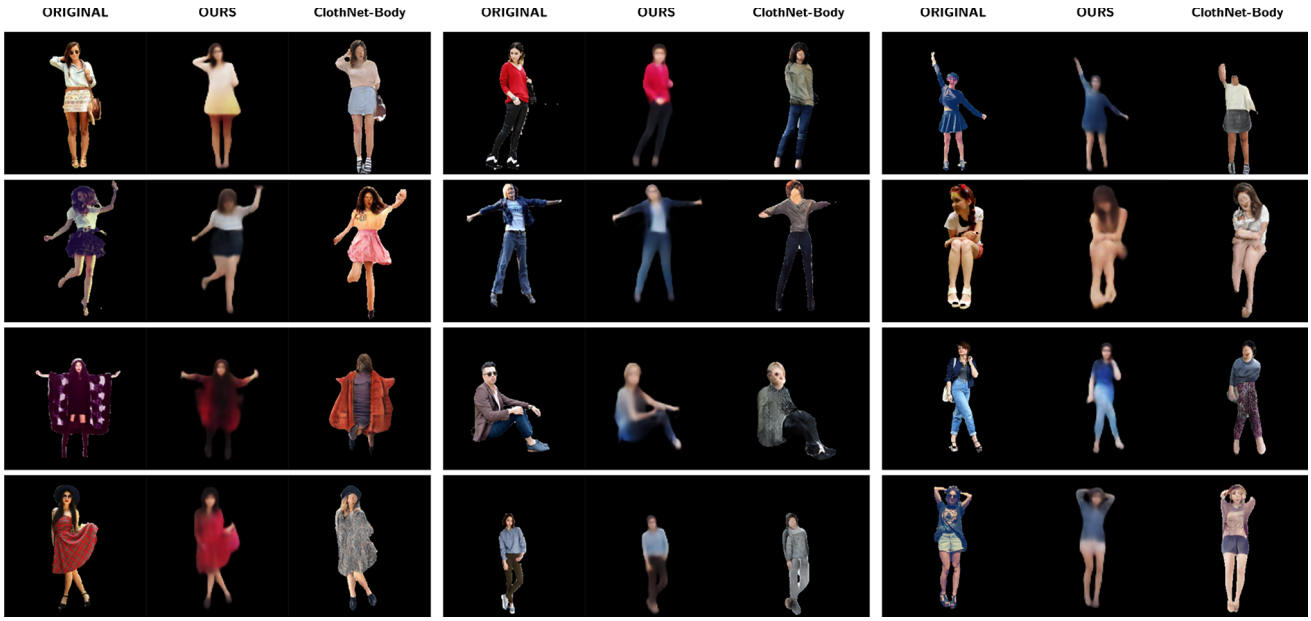


Fig. 19 Reconstructions on ChictopiaPlus. In each trio of images we have, respectively: original image (256 × 256), Conditional-DGPose and ClothNet-body (Lassner et al. 2017) reconstructions. Notice that the images generated by our model are much closer to the originals in terms of appearance (colors). Moreover, in general, the Conditional-DGPose captures the body parts' locations more accurately, resulting in better pose reconstructions (see Fig. 20). Best viewed if zoomed in digital version (Color figure online)

Table 2 Image quality on ChictopiaPlus

	PSNR	SSIM
Conditional-DGPose	**21.33**	**0.88**
ClothNet-body (Lassner et al. 2017)	16.89	0.82

Best result is shown in bold
Quantitative evaluation w.r.t. image quality, showing that our method outperforms (Lassner et al. 2017) considering both metrics, the PSNR and the SSIM

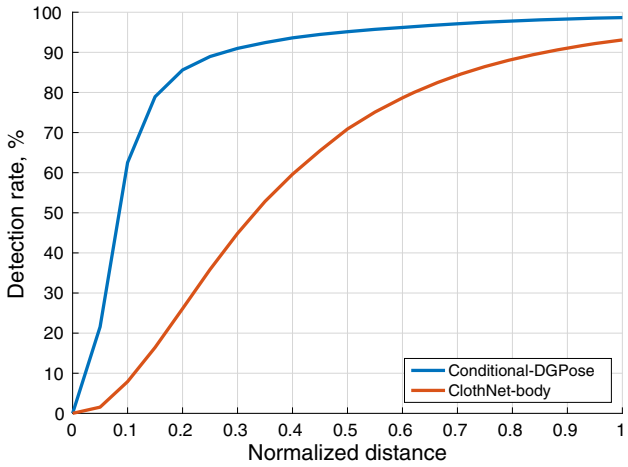


Fig. 20 Accuracy of Poses on ChictopiaPlus. The PCK scores over reconstructed images of our Conditional-DGPose (blue) significantly outperforms the ClothNet-body (Lassner et al. 2017) (red). Detection rate represents the percentage of joints correctly relocated in the reconstructions (Color figure online)

5.6.4 Conditional-DGPose Results on DeepFashion

Here we show qualitative and quantitative experiments on the DeepFashion dataset (Liu et al. 2016). The baseline on this dataset is the image-to-image pose guided generation (PG^2) by Ma et al. (2017). Thus, we use their same training and test sets. However, as our model is not an image-to-image translation architecture, we do not use pairs of images for training. Instead, we use individually 44,950 training images and 6560 test images.

Again, we employ the PSNR and the SSIM metrics to evaluate image quality, and the PCK metric to provide a quantitative evaluation of pose reconstructions, as described previously (see Sect.5.4). In Table 3, we initially show that even not being trained on images pairs and tackling the significantly more complex task of learning a generative model, instead of executing image-to-image translation, our method achieves scores only slightly below the ones by the PG^2 network on image reconstruction. A similar observation can be done regarding pose reconstruction, since our model reports 74.94% of accuracy, with PCK score at 0.5, against 78.27% from Ma et al. (2017). The overall PCK curve is shown in Fig. 21.

Table 3 Image quality on DeepFashion

	PSNR	SSIM
Conditional-DGPose	18.38	0.79
PG^2 (Ma et al. 2017)	**18.96**	**0.83**

Best result is shown in bold
Quantitative evaluation w.r.t. image quality, showing that our method presents a performance only slightly below the baseline (Ma et al. 2017), considering both metrics, the PSNR and the SSIM, despite the fact it tackles a significantly more complex task than image-to-image translation

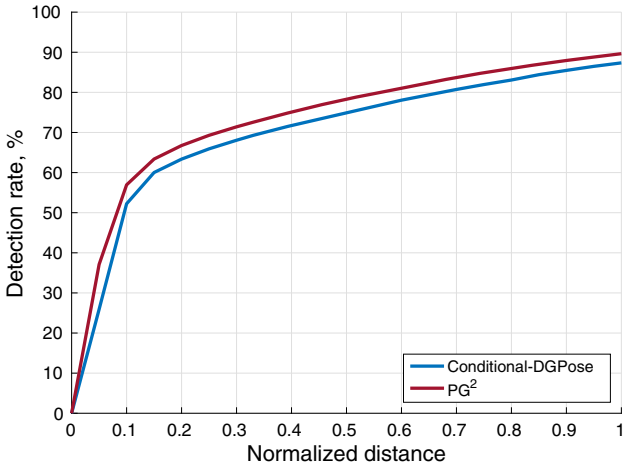


Fig. 21 Accuracy of Poses on DeepFashion. The PCK scores over reconstructed images of our Conditional-DGPose (blue) performs only slightly below the PG^2 network (Ma et al. 2017) (red), despite the fact it is tackling a significantly more complex problem than image-to-image translation. Detection rate represents the percentage of joints correctly relocated in the reconstructions (Color figure online)

Concretely, the learning of a full generative model, instead of image-to-image translation, allows for the execution of tasks, such as sampling from the learned latent space, which are just not feasible with architectures purely trained on image pairs. To illustrate this, in Fig. 22 we traverse the appearance manifold learned on the DeepFashion dataset. Using only our heatmap pose representation as input, for a given pose, we smoothly vary the values of the latent appearance representation, generating samples with different visual aspect for the same body posture. Such kind of direct sampling is not feasible with the PG^2 (Ma et al. 2017) architecture.

Finally, the Conditional-DGPose performs 3.06% and 4.82% worse than the PG^2 (Ma et al. 2017) regarding, respectively, the PSNR and the SSIM metrics (see Table 3). Despite that, it produces reasonable results in comparison with the ones from Ma et al. (2017). A qualitative evaluation is shown in Fig. 23.



Fig. 22 Conditional-DGPose Appearance Manifold. Illustration of the appearance manifold learned on the DeepFashion dataset. We smoothly traverse the manifold for a given pose, causing changes in the visual appearance of the person in the image. No image is used as input, only our heatmap pose representation, evidencing that a truly generative model of images was learned, in which pose and appearance are disentangled. Best viewed if zoomed in digital version

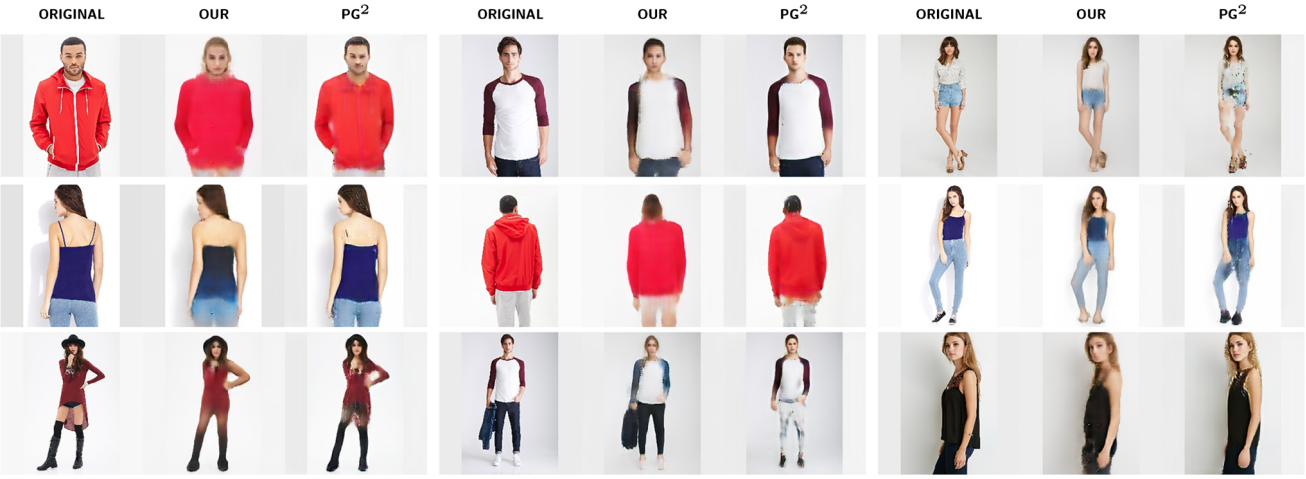


Fig. 23 Reconstructions on DeepFashion . In each trio of images, we have, respectively: original image, Conditional-DGPose and PG^2 (Ma et al. 2017) reconstructions. All images have 256×256 pixels. Although tackling a more complex task than (Ma et al. 2017), our results are still reasonable. Best viewed if zoomed in digital version

5.7 Semi-DGPose

Here, we initially evaluate our Semi-DGPose model on the Human3.6M (Ionescu et al. 2014) dataset. The Human3.6M is more suitable than both, the ChictopiaPlus and the DeepFashion, for pose estimation evaluations, since the former has joints' annotations obtained by an accurate motion capture system. While the two other datasets are augmented with 2D pose labels obtained using an *off-the-shelf* pose estimator, consequently resulting in more errors in the ground-truth annotations. We show quantitative and qualitative results, focusing particularly on the pose estimation and the *indirect pose-transfer* capabilities, described later in this section. Our experiments and results show the effectiveness of the Semi-DGPose method on the Human3.6M.

To show the generality of the model, we present additional results on the DeepFashion dataset. We now use our Conditional-DGPose architecture and the image-to-image translation network PG^2 (Ma et al. 2017) as baselines, despite to their relevant differences with the Semi-DGPose. However, to our knowledge, there are no closer related methods in the literature, i.e. that simultaneously pursue the *understanding* and the *generation* of people directly in the image space. Since our Conditional-DGPose method outperforms the ClothNet-body (Lassner et al. 2017) architecture, we do not carry out a direct comparison with the latter.

5.7.1 Semi-DGPose Results on Human3.6M

To evaluate the efficacy of our model, we perform a "relative" comparison. In other words, we first train our model with full supervision (i.e. all data points are labelled) to evaluate performance in an ideal case and then we train the model with other setups, using labels only for 75%, 50%, and 25% data points. Such an evaluation allows us to decouple the efficacy of the model itself and the semi-supervision to see how the gradual decrease in the level of supervision affects the final performance of the method on the same validation set.

With full supervision, we first cross-validated the hyper-parameter α which weights the regression loss (see Eq. 5, in Sect. 3) and found that $\alpha = 100$ yields the best results, as shown in Fig. 24a. Following Siddharth et al. (2017), we keep $\gamma = 1$ in all experiments (see Eq. 3, in Sect. 3). In Fig. 24b, we show reconstructed images along with the heatmap pose representation, which are realistic and comparable with the ones obtained with the Conditional-DGPose (see Fig. 14). *Direct manipulation*, when pose representation is changed during the reconstruction process while appearance is kept the same, is illustrated in Fig. 25. Still with full supervision, we show the pose estimation accuracy for different samples in Fig. 26. The Semi-DGPose achieves 93.85% PCK score, normalised at 0.5, in the fully-supervised setup (see Fig. 28). This pose estimation accuracy is on par with the state-of-

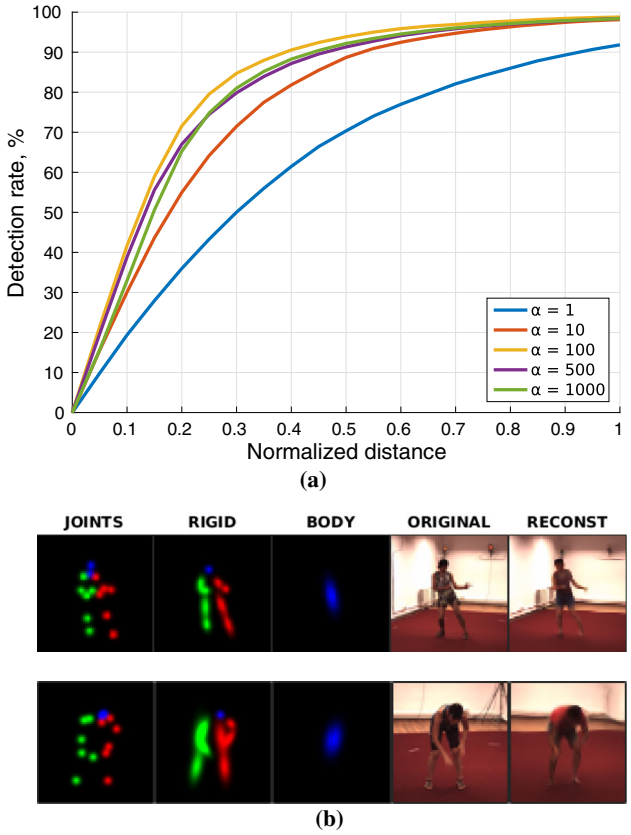


Fig. 24 (**a**) PCK scores for the cross-validation adjustment of the regression loss weight α. (**b**) Qualitative reconstructions with full supervision

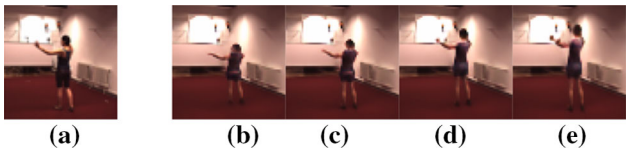


Fig. 25 Direct manipulation. Original image (**a**), followed by reconstructions in which the person's height was changed to a percentage of the original, as: (**b**) 80%, (**c**) 95%, (**d**) 105% and (**e**) 120%. The same procedure may be applied to produce different changes in the body size and aspect ratio

Table 4 Image quality on Human3.6M

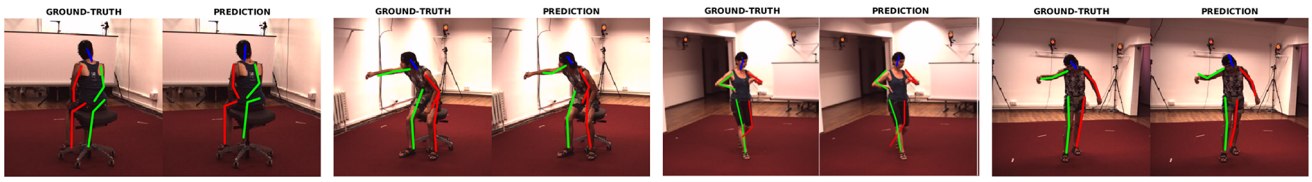
Level of supervision	PSNR	SSIM
100%	22.27	0.89
75%	21.49	0.87
50%	21.36	0.86
25%	20.06	0.83

Quantitative evaluations of the Semi-DGPose with different levels of supervision using the PSNR and SSIM metrics

the-art pose estimators on unconstrained images (Yang et al. 2017). However, since the Human3.6M was captured in a controlled environment, a standard (discriminative) pose estimator is expected to perform better.

Subsequently, we evaluate it across different levels of supervision, with the PSNR and SSIM metrics and show results in Table 4. In Fig. 27, we show reconstructed images obtained with such different levels. It allows us to observe how image quality is affected when we gradually reduce the availability of labels. Furthermore, we also evaluate the pose estimation accuracy with semi-supervision. The overall PCK curves corresponding to each percentage of supervision in the training set is shown in Fig. 28. Note that, even with only 25% of labels available, our model still obtains 88.35% PCK score, normalised at 0.5, showing the effectiveness of the semi-supervised approach. Qualitative samples are shown in Fig. 29. Again, aiming to illustrate how the gradual decrease of supervision in the training set affects the quality of pose estimation on the test images.

Concerning *indirect pose-transfer*, as both latent variables corresponding to pose and appearance can be inferred by the model's Encoder (recognition network) at test time, latent variables extracted from different images can be combined in a subsequent step, and employed together as inputs for the Decoder (generative network). The result of that is a generated image combining appearance and body pose, extracted from two different images. The process is done in three phases, as illustrated in Fig. 30. Firstly, the latent pose representation $\mathbf{y}_{v_1}$ is estimated from the first input image



(a) PCK=92.9% **(b)** PCK=100.0% **(c)** PCK=96.4% **(d)** PCK=100.0%

Fig. 26 Pose Estimation on Human3.6M. Pairs of ground-truth and predicted joints superimposed on the original images. Below each pair, we show the PCK score normalised at 0.5 times the torso size, as usual for the PCK metric. Such normalised distance explains the high scores despite the existence of minor differences between ground-truth and predicted positions. Results were obtained with 100% of supervision during training, and each pair correspond to one of the 4 cameras from the Human3.6M dataset

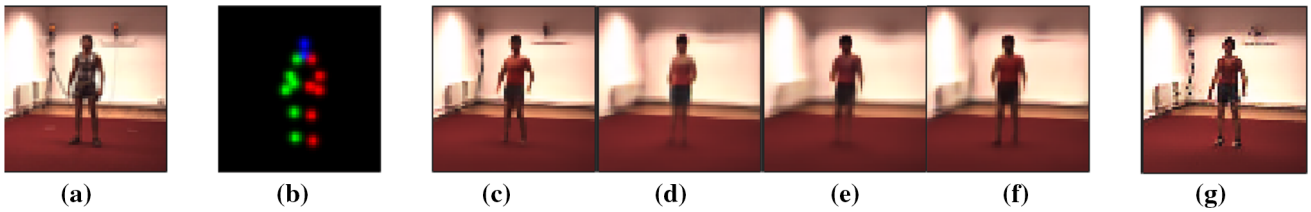


Fig. 27 Reconstructions on Human3.6M. (**a**) Original image. (**b**) Heatmap pose representation (rigid parts and body suppressed in the illustration for simplicity), followed by reconstructions with different levels of supervision: (**c**) 100%, (**d**) 75%, (**e**) 50%, (**f**) 25%, and (**g**) Conditional-DGPose

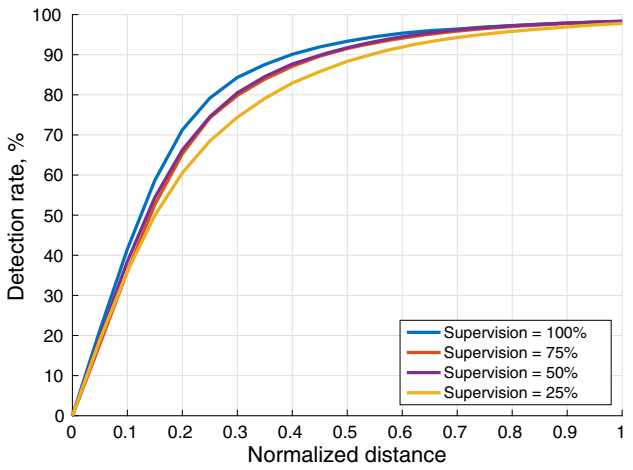


Fig. 28 Accuracy of Poses on Human3.6M. Quantitative evaluations of Semi-DGPose for different levels of supervision using the PCK scores. Note that, even with 25% supervision, our Semi-DGPose obtains 88.35% PCK score, normalised at 0.5

through the Encoder. Secondly, the latent *appearance* representation $\mathbf{z}_2$ is estimated from a second image, also through the Encoder. Lastly, $\mathbf{y}_{v_1}$ and $\mathbf{z}_2$ are propagated through the Decoder, and a new image is generated, combining body pose and appearance, respectively, from the first and second *encoded* images. We evaluate qualitatively the effects of semi-supervision over the indirect pose-transfer in Fig. 31.

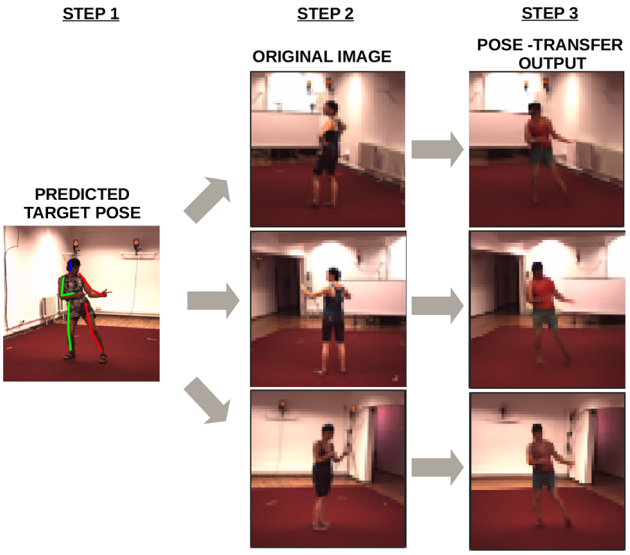


Fig. 30 Indirect pose-transfer on Human3.6M. **Step 1**: the latent target pose representation $\mathbf{y}_{v_1}$ is estimated (Encoder). **Step 2**: the image from which the latent *appearance* $\mathbf{z}_2$ is estimated (Encoder). **Step 3**: the output image generated as a combination of $\mathbf{y}_{v_1}$ and $\mathbf{z}_2$ (Decoder). The people's outfits in the output images are approximated to the ones in the original images. However, restricted by the low diversity of outfits observed in Human3.6M training data. Note that, to highlight the separation of appearance and pose, we chose the image on Step 1 to be from camera 2, while the original images are from cameras, 1, 3 and 4, respectively. As can be seen, the background scene is totally defined by the original images

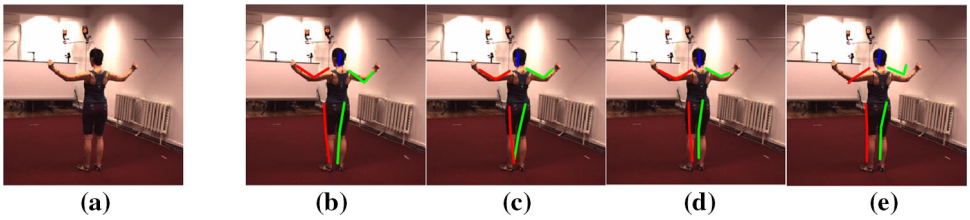


Fig. 29 Qualitative results of semi-supervised pose estimation. Original image (**a**), followed by predictions, over the original image, with: (**b**) 100%, (**c**) 75%, (**d**) 50% and (**e**) 25% of supervision. The figure aims to illustrate how the decrease in supervision affects pose estimation. The results are similar, yet it is possible to observe some important discrepancies. For instance, due to the shortage of labelled training data, the pose estimation result in (**e**) is worse than the one shown in (**b**), particularly regarding the location of arms' extremities

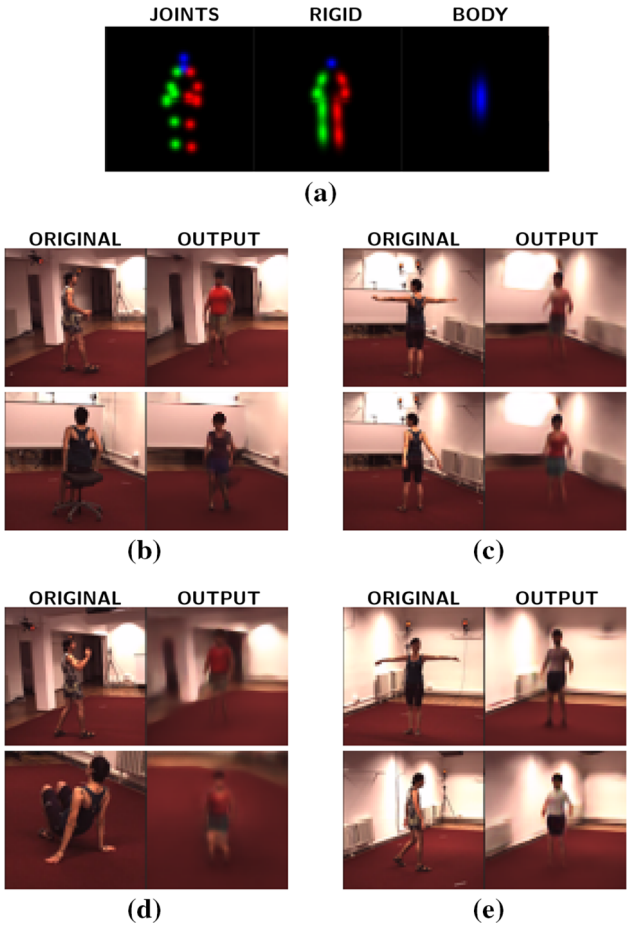


Fig. 31 Indirect Pose-transfer on Human3.6M. Qualitative results with different levels of supervision. (**a**) Heatmap representation of the target pose (i.e. after being processed by the Mapper module) used for all the subsequent results. Such results show pairs of original images and pose-transfer outputs obtained with the following levels of supervision: (**b**) 100%, (**c**) 75%, (**d**) 50%, and (**e**) 25%. In the pose-transfer outputs, appearance comes from the original images while the body posture is defined by the target pose

5.7.2 Semi-DGPose Results on DeepFashion

To show the generality of the Semi-DGPose, model we present additional results on the DeepFashion dataset, using our Conditional-DGPose architecture and the image-to-image translation network PG^2 (Ma et al. 2017) as baselines. The same hyper-parameters reported previously were used in training. In Table 5, we compare the image quality of reconstructions, while in Fig. 32 we show the comparison concerning the quality of pose reconstructions. Although the

Table 5 Image quality on DeepFashion

	PSNR	SSIM
Semi-DGPose	16.84	0.76
Conditional-DGPose	18.38	0.79
PG^2 (Ma et al. 2017)	**18.96**	**0.83**

Best result is shown in bold
Quantitative evaluation of Semi-DGPose using PSNR and SSIM measures comparing the image quality of reconstructions. The Semi-DGPose shows less accurate results, yet in contrast to the other methods, it performs a significantly more complex task, simultaneously executing pose estimation, and also allowing for semi-supervised learning

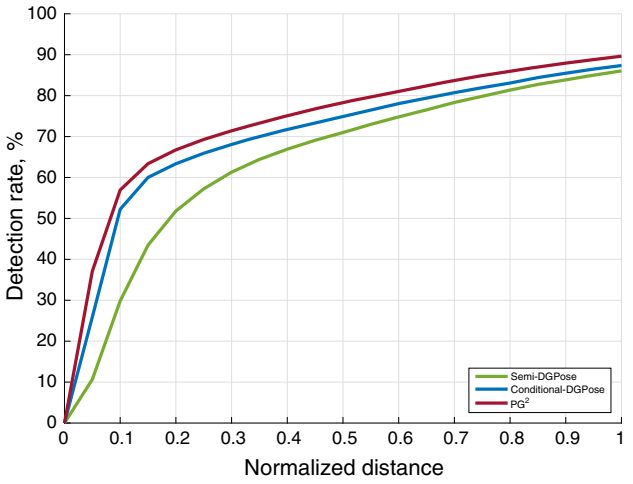


Fig. 32 Accuracy of Poses on DeepFashion. Quantitative evaluation of Semi-DGPose PCK scores over reconstructed poses. The Semi-DGPose (green) shows less accurate results, however, in contrast to the Conditional-DGPose (blue) and the PG^2 network (Ma et al. 2017) (red), it performs a significantly more complex task, simultaneously executing pose estimation and allowing for semi-supervised learning. Detection rate represents the percentage of joints correctly relocated in the reconstructions

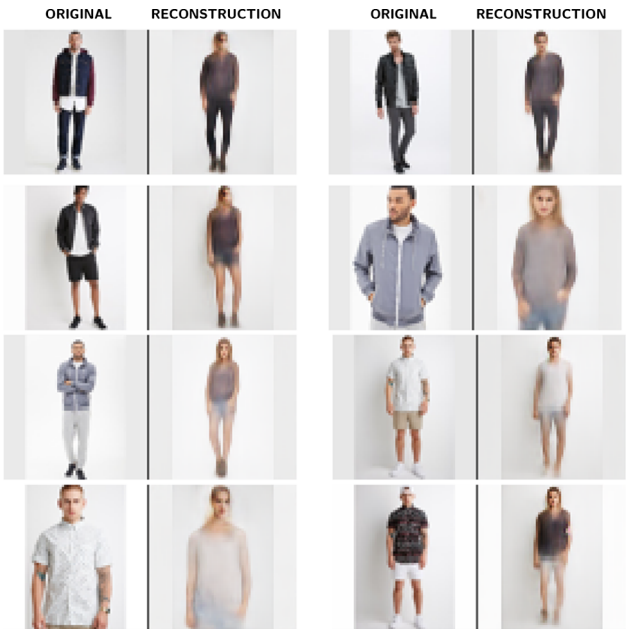


Fig. 33 Reconstructions on DeepFashion. The only input of the Semi-DGPose is the original image. At test time, as pose is estimated in the latent space, discrepancies between the original and reconstructed poses are more frequently observed, in comparison with the Conditional-DGPose. Best viewed if zoomed in digital version

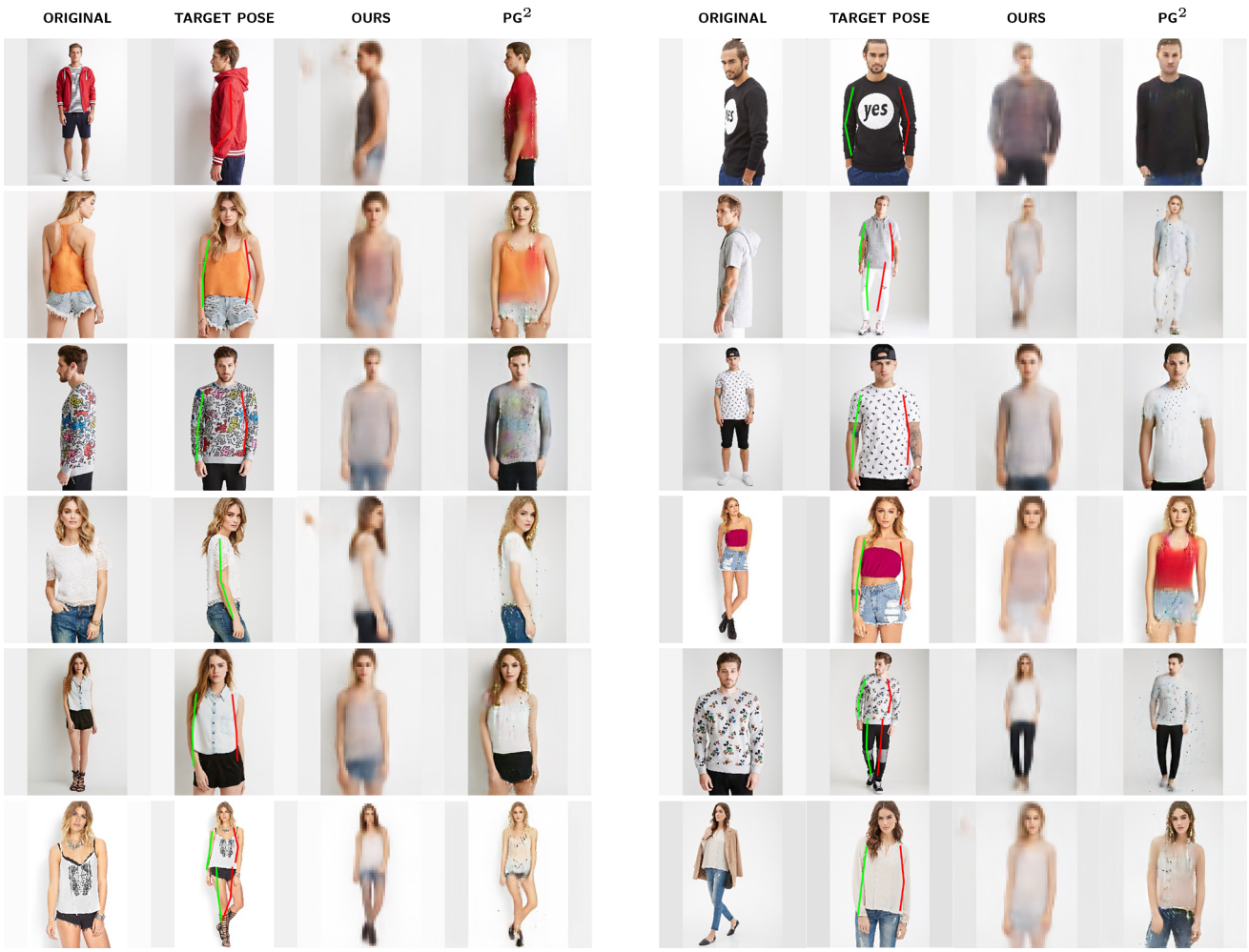


Fig. 34 Indirect pose-transfer in DeepFashion dataset. In each set of images, we have, respectively: the original image, the target image with the superimposed target pose predicted by the Semi-DGPose, the pose-transfer output from the Semi-DGPose and the pose-transfer output from PG^2 (Ma et al. 2017). Although tackling a more complex task than (Ma et al. 2017), which includes the prediction of pose, our results are still reasonable

Semi-DGPose presents less accurate results, it is important to highlight that it is also tackling the pose estimation task, which is not performed by either one of the other two methods, i.e. the Conditional-DGPose and the PG^2. To pursue, simultaneously, the *understanding*, i.e. estimation of pose and appearance in the latent space, and the *generation* of people directly in images, shows to be indeed a significantly more complex task. Nevertheless, the justification for seeking such a challenging goal, as mentioned before, mainly lie on its important capability of allowing for semi-supervised learning, that is not present in the comparable methods.

In Fig. 33, we show comparisons between input and reconstructed images. In some of the samples, we can observe small differences between the original and the reconstructed body postures, mainly regarding the positions of the limbs. This illustrates the higher complexity involved in simultaneously estimating pose and appearance in our latent space. For instance, inaccurate predictions of pose, performed by the Encoder, may have effects into the final reconstructed appearance, and vice-versa, when the latent representations are mapped back to the image space, by the Decoder. Such interdependency does not exist when pose is a given observable variable, as in the case of the conditional models or image-to-image translation networks.

Finally, we highlight *indirect pose-transfer* in the DeepFashion dataset, which is a distinctive capability of the Semi-DGPose, in comparison to related methods. In Fig. 34, we compare the indirect pose-transfer results, from our single-stage structured generative model, the Semi-DGPose, with the results from the image-to-image translation baseline, the PG^2 network (Ma et al. 2017). It is important to notice that our Semi-DGPose model was not trained specifically for pose-transfer, i.e. it was not trained on pairs of images. On the other hand, the PG^2 architecture is trained on pairs of images of the same person, in different poses, scales or point of views (first two images of each set in Fig. 34).

Moreover, in the Semi-DGPose the body pose is estimated by the Encoder network (illustrated in every second image of each set in Fig. 34), along with appearance, while in the PG^2 pose is given as an observable variable to the model. Despite such critical competitive disadvantages, we can observe that the Semi-DGPose produce reasonable results in comparison to the ones from PG^2. Lastly, it is crucial to call attention for the capabilities of our Semi-DGPose approach such as, interpretability of the latent space, pose estimation, sampling and semi-supervised learning, which are not jointly present in the PG^2 or in the related work from the literature. These features justify our approach for learning a deep generative model of people in images and, to our knowledge, significantly differentiate the Semi-DGPose model from prior art.

5.8 Limitations of the Models

Here, we discuss two important limitations common to the Conditional-DGPose and the Semi-DGPose. The first refers to the modelling of appearance in both models. As we mention in Sect. 1, our latent representation of appearance encodes all the visual information in the images (e.g. clothing, skin colours, hairstyles, and background) except for the body pose of the subjects. However, such a strategy does not separate the individual visual characteristics in the latent representation. In Fig. 22 (Sect. 5.6.4), we can observe that as the appearance manifold is traversed, the visual features gradually change altogether. A disentangled representation for appearance itself would be needed for allowing control over specific visual features. Another aspect concerning appearance regards limitations to approximate clothing "seen" few times or "unseen" during training. Interestingly, the extrapolation capabilities shown for unseen poses (see Fig. 18 in Sect. 5.6.2) is not observed for appearance. For example, in the Human3.6M dataset, the low diversity of subjects' outfits may eventually prevent the clothing in the reconstructed images to be precisely equal to the ones in the original images, as can be observed in Fig. 30 (Sect. 5.7.1). Other works in the literature refer to this same problem concerning the Human3.6M dataset, e.g. Rhodin et al. (2018).

The second relevant limitation refers to our pose representation. Aiming to investigate and explore the capabilities of simple body representations, we have worked only with 2D pose in our models. Such option turns our approaches

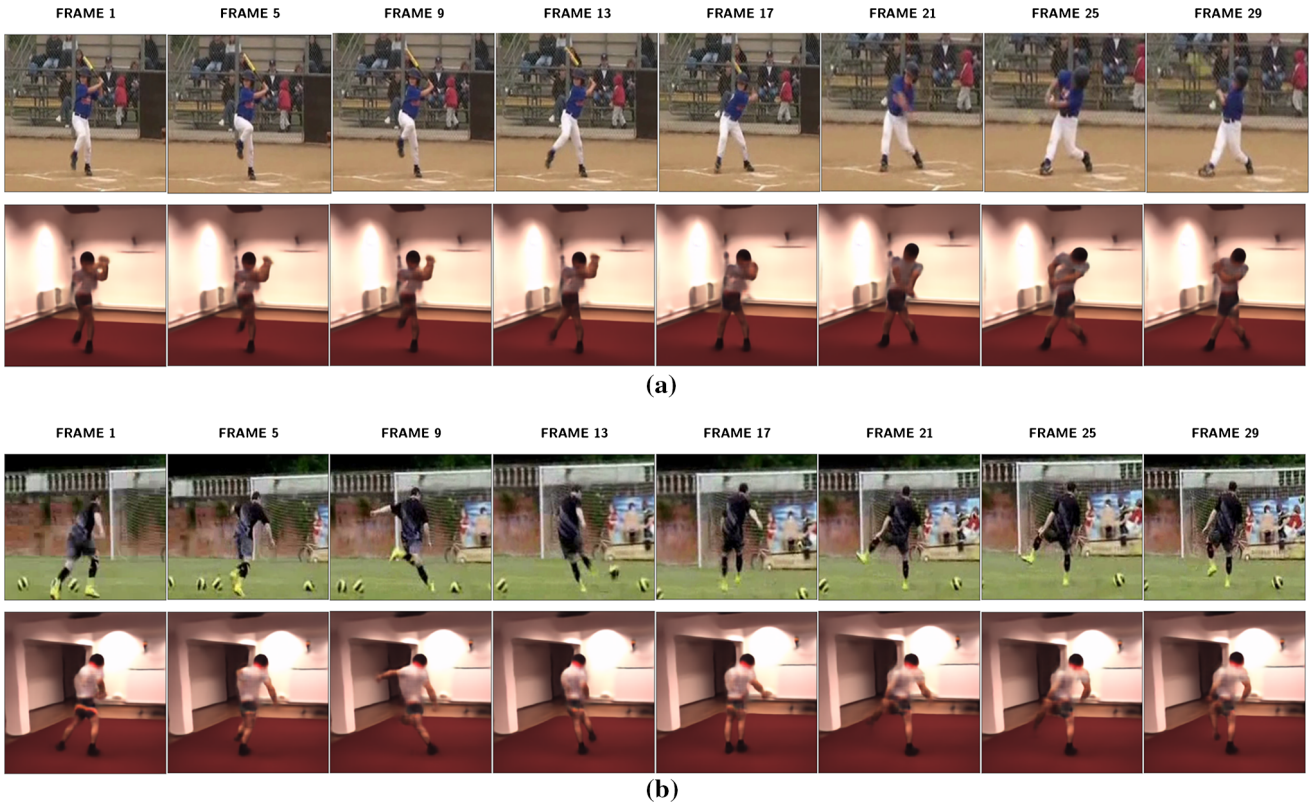


Fig. 35 Cross-domain pose-transfer over single images from short video sequences from the JHMDB dataset (Jhuang et al. 2013). (**a**) A sequence of frames shows a boy batting a ball while playing baseball (top row) and the correspondent pose-transfer outputs (bottom row). Mainly due to self-occlusion, some limbs appear blended. (**b**) A football player is kicking a ball towards the goal (top row) and the correspondent pose-transfer outputs (bottom row). Frames 5, 9, and 25 present important issues due to particular postures and self-occlusion of limbs. Best viewed if zoomed in digital version

more general since they are not dependent on 3D information (e.g. 3D models, camera calibration, or multi-view images). It allows, for example, their application on ordinary monocular images. Moreover, this strategy is also less susceptible to body shape variations in comparison to segmentations mask or 3D body meshes, which might not be directly transferable from one person to another. However, such simplicity creates some limitations. An important one concerns the lack of depth information in the body model. Despite the reasonable results obtained with single people in relatively "well-behaved" poses, the models might face difficulties in the presence of stronger self-occlusions associated with particular body postures. In the absence of depth, it is hard to infer, for instance, which one of two overlapping limbs is closer to the camera. Without such explicit information in the body representation, the correct reconstruction might present flaws.

To analyse such issues here, which are present in our both models, we have employed the Conditional-DGPose, trained on the Human3.6M dataset, to perform cross-domain pose-transfer over single images from short video sequences. Employing a sequence of frames allow us to observe how the performance of the model changes according to the concurrent presence of self-occlusion and different poses. In the current experiments, we have used short videos from the JHMDB dataset (Jhuang et al. 2013). Each "in-the-wild" video depicts a single person performing one activity. The dataset provides 2D pose annotations per frame for all videos. Such annotations are used as inputs for the Conditional-DGPose cross-domain pose-transfer. We crop the images maintaining the subjects centralised.

In Fig. 35a, a sequence of frames shows a boy batting a ball while playing baseball (top row) and the correspondent pose-transfer outputs (bottom row). Although the reenacted frames present the gist of the original sequence, already it is possible to observe that overlapping arms and legs appear to be blended in some of the output images (e.g. frames 1 and 5), making evident the problem we have mentioned earlier. Fig. 35b (top row) depicts a football player kicking a ball towards the goal. We call attention for frame 5, in which the self-occluded arm of the original subject turns the upper body of the reconstructed person wider. In frame 9, the concurrent overlapping legs and the unusual pose contribute for an ambiguous posture of the person in the output image, which might be facing forwards or backwards. The particular body pose in frame 25 provokes the misalignment of head, torso and arms in of the body in the output. Finally, even without a task-specific training, we believe that the use of a 3D body representation, which would explicitly encode depth, may be beneficial to mitigate the main issues mentioned above.

6 Conclusions

In this paper, we have presented a comprehensive deep generative model framework for human pose analysis in images. Our models are based on a principled VAEGAN approach and allow the disentanglement of body posture and visual appearance, aiming for the independent manipulation of such factors. With our conditional-VAEGAN model, the Conditional-DGPose, differently from previous art, we have taken such manipulation to extreme cases, e.g. by performing cross-domain *pose-transfer* and by hallucinating multiple people in a variety of unseen or even unrealistic poses. Moreover, we have achieved state-of-the-art results on image reconstruction conditioned on pose, *outperforming* the closest related comparable baseline (Lassner et al. 2017). With a single-stage structured semi-supervised VAEGAN architecture, the Semi-DGPose, we pursued the joint *understanding* and *generation* of people in images, not only mapping images to partially interpretable latent representations but also mapping these representations back to the image space. Importantly, such an approach simultaneously allows for reconstruction, direct manipulation, sampling, pose estimation, indirect pose-transfer, and semi-supervised learning. These joint capabilities differentiate the Semi-DGPose from other methods in the literature and demonstrate a real-world application of structured deep generative models with the highly relevant potential of being less dependable of fully-labelled data. We have systematically evaluated our methods on well-known benchmarks, the Human3.6M, the ChictopiaPlus, and the DeepFashion datasets, comparing our results with the closest related baseline methods in the literature (Lassner et al. 2017; Ma et al. 2017). Such results and comparisons highlight the novelty and effectiveness of our approaches and its capabilities, despite the significant challenge posed by our aimed goal. We believe that we have shown and reinforced the relevance of employing an interpretable and structured latent space, which allows for semi-supervised learning, as well as the importance of tackling the problem with single-stage end-to-end architectures.

Acknowledgements This work was supported by the ERC grant ERC-2012-AdG 321162-HELIOS, EPSRC grant Seebibyte EP/M013774/1 and EPSRC/MURI grant EP/N019474/1. We would also like to acknowledge the Royal Academy of Engineering and FiveAI. Rodrigo de Bem is a CAPES Foundation scholarship holder (Process no: 99999.013296/2013-02, Ministry of Education, Brazil).

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your

intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

A DGPose Architectures Details

Here, we provide implementation details of our both architectures considering the following inputs: images $\mathbf{x}$ (batch_size = 64, channels = 3, height = 64, width = 64) and heatmaps $\mathbf{y}_h$ (batch_size = 64, channels = 24, height = 64, width = 64). Regarding the heatmap labels, the channels correspond to: **(i)** 14 joints (head top, neck, right shoulder, right elbow, right wrist, right hip, right knee, right ankle, left shoulder, left elbow, left wrist, left hip, left knee, left ankle); **(ii)** 9 rigid parts (head, right upper arm, right lower arm, right upper leg, right lower leg, left upper arm, left lower arm, left upper leg, left lower leg); **(iii)** 1 whole body. Finally, in Tables 6 and 7, we show the full definition of both, the Conditional-DGPose and the Semi-DGPose, respectively.

Table 6 Conditional-DGPose architecture for 64 × 64 input images

Encoder	
Input: $\mathbf{x}$(batch_size = 64, channels = 3, height = 64, width = 64); $\mathbf{y}_h$(batch_size = 64, channels = 24, height = 64, width = 64)	
Layer	**Definition**
1	CONCAT($\mathbf{x}$, $\mathbf{y}_h$)
2	CONV-(N64, K7, S2, P1), LeakyReLU(0.01)
3	CONV-(N128, K3, S2, P1), BN, ReLU
4	CONV-(N256, K3, S2, P1), BN, ReLU
5	CONV-(N512, K3, S2, P1), BN, ReLU
6	CONV-(N512, K3, S2, P1), BN, ReLU
7	CONV-(N512, K3, S2, P1), BN, ReLU
8	RESIDUAL-(N512, K3, S1, P1)
9	RESIDUAL-(N512, K3, S1, P1)
10	RESIDUAL-(N512, K3, S1, P1)
11	RESIDUAL-(N512, K3, S1, P1), SIGMOID
$\mu_{\mathbf{z}}$	FC-(N100)
$\sigma_{\mathbf{z}}$	FC-(N100)
Prior	
Input: $\mathbf{y}_h$(batch_size = 64, channels = 24, height = 64, width = 64)	
Layer	**Definition**
1	CONV-(N128, K4, S2, P1), LeakyReLU(0.2)
2	CONV-(N256, K4, S2, P1), BN, LeakyReLU(0.2)
3	CONV-(N512, K4, S2, P1), BN, LeakyReLU(0.2)
4	CONV-(N1024, K4, S2, P1), BN, LeakyReLU(0.2)
5	CONV-(N100, K4, S1, P0), SIGMOID
μ_{prior}	FC-(N100)
σ_{prior}	FC-(N100)
Decoder	
Input: $\mathbf{z}$(batch_size = 64, channels = 100)	
Layer	**Definition**
1	RESHAPE$_{\text{(batch_size = 64, channels = 100, height = 1, width = 1)}}$
2	DECONV-(N512, K4, S1, P0), BN, LeakyReLU(0.2)
3	DECONV-(N256, K4, S2, P1), BN, LeakyReLU(0.2)
4	DECONV-(N128, K4, S2, P1), BN, LeakyReLU(0.2)
5	DECONV-(N64, K4, S2, P1), BN, LeakyReLU(0.2)
6	DECONV-(N128, K4, S2, P1), BN, LeakyReLU(0.2)
7	CONCAT(DECONV-6, $\mathbf{y}_h$)
8	CONV-(N512, K5, S1, P2), BN, LeakyReLU(0.2)
9	CONV-(N256, K5, S1, P2), BN, LeakyReLU(0.2)
10	CONV-(N128, K5, S1, P2), BN, LeakyReLU(0.2)
11	CONV-(N128, K5, S1, P2), BN, LeakyReLU(0.2)
G($\mathbf{y}_h$, $\mathbf{z}$)	CONV-(N3, K5, S1, P2), TANH
Discriminator	
Input: G($\mathbf{y}_h$, $\mathbf{z}$)(batch_size = 64, channels = 3, height = 64, width = 64); $\mathbf{x}$(batch_size = 64, channels = 3, height = 64, width = 64)	
Layer	**Definition**
1	CONV-(N64, K4, S2, P1), LeakyReLU(0.2)
2	CONV-(N128, K4, S2, P1), BN, LeakyReLU(0.2)
3	CONV-(N256, K4, S2, P1), BN, LeakyReLU(0.2)
4	CONV-(N512, K4, S2, P1), BN, LeakyReLU(0.2)
5	CONV-(N1, K4, S1, P0), SIGMOID

We use the following abbreviations: N for the number of kernels/neurons, K for kernel size, S for stride and P for zero padding. Concerning the layers, CONCAT means concatenation layer, CONV means convolutional layer, BN means batch normalization layer with running average coefficient $\beta = 0.9$ and learnable affine transformation, DECONV means transpose convolutional layer, FC means fully connected layer, SUM corresponds to element-wise sum layer and RESIDUAL denotes a residual block, detailed at Table 8. The additional layers can be clearly understood. Finally, particular parameters for specific layers are defined between parenthesis after the layers' names

Table 7 Semi-DGPose architecture for 64 × 64 input images

Encoder	
Input: $\mathbf{x}$(batch_size = 64, channels = 3, height = 64, width = 64)	
Layer	**Definition**
1	CONV-(N64, K7, S2, P1), LeakyReLU(0.01)
2	CONV-(N128, K3, S2, P1), BN, ReLU
3	CONV-(N256, K3, S2, P1), BN, ReLU
4	CONV-(N512, K3, S2, P1), BN, ReLU
5	CONV-(N512, K3, S2, P1), BN, ReLU
6	CONV-(N512, K3, S2, P1), BN, ReLU
7	RESIDUAL-(N512, K3, S1, P1)
8	RESIDUAL-(N512, K3, S1, P1)
9	RESIDUAL-(N512, K3, S1, P1)
10	RESIDUAL-(N512, K3, S1, P1), SIGMOID
$\mu_{\mathbf{z}}$	FC-(N100)
$\sigma_{\mathbf{z}}$	FC-(N100)
$\mu_{\mathbf{y}_v}$	FC-(N48)
$\sigma_{\mathbf{y}_v}$	FC-(N48)
Mapper	
Input: $\mathbf{y}_v$(batch_size = 64, channels = 48)	
Layer	**Definition**
1	$\text{RESHAPE}_{(\text{batch_size}=64,\ \text{channels}=48,\ \text{height}=1,\ \text{width}=1)}$
2	DECONV-(N512, K4, S1, P0), BN, LeakyReLU(0.2)
3	DECONV-(N256, K4, S2, P1), BN, LeakyReLU(0.2)
4	DECONV-(N128, K4, S2, P1), BN, LeakyReLU(0.2)
5	DECONV-(N64, K4, S2, P1), BN, LeakyReLU(0.2)
$\mathbf{y}_h$	DECONV-(N24, K4, S2, P1), SIGMOID
Decoder	
Input: $\mathbf{z}$(batch_size = 64, channels = 100); $\mathbf{y}_v$(batch_size = 64, channels = 48); $\mathbf{y}_h$(batch_size = 64, channels = 24, height = 64, width = 64)	
Layer	**Definition**
1	CONCAT($\mathbf{z}$, $\mathbf{y}_v$)
2	$\text{RESHAPE}_{(\text{batch_size}=64,\ \text{channels}=148,\ \text{height}=1,\ \text{width}=1)}$
3	DECONV-(N512, K4, S1, P0), BN, LeakyReLU(0.2)
4	DECONV-(N256, K4, S2, P1), BN, LeakyReLU(0.2)
5	DECONV-(N128, K4, S2, P1), BN, LeakyReLU(0.2)
6	DECONV-(N64, K4, S2, P1), BN, LeakyReLU(0.2)
7	DECONV-(N128, K4, S2, P1), BN, LeakyReLU(0.2)
8	CONCAT(DECONV-6, $\mathbf{y}_h$)
9	CONV-(N512, K5, S1, P2), BN, LeakyReLU(0.2)
10	CONV-(N256, K5, S1, P2), BN, LeakyReLU(0.2)
11	CONV-(N128, K5, S1, P2), BN, LeakyReLU(0.2)
12	CONV-(N128, K5, S1, P2), BN, LeakyReLU(0.2)
G($\mathbf{y}_v$, $\mathbf{z}$)	CONV-(N3, K5, S1, P2), TANH
Discriminator	
Input: G($\mathbf{y}_v$, $\mathbf{z}$)(batch_size = 64, channels = 3, height = 64, width = 64); $\mathbf{x}$(batch_size = 64, channels = 3, height = 64, width = 64)	
Layer	**Definition**
1	CONV-(N64, K4, S2, P1), LeakyReLU(0.2)
2	CONV-(N128, K4, S2, P1), BN, LeakyReLU(0.2)
3	CONV-(N256, K4, S2, P1), BN, LeakyReLU(0.2)
4	CONV-(N512, K4, S2, P1), BN, LeakyReLU(0.2)
5	CONV-(N1, K4, S1, P0), SIGMOID

We use the following abbreviations: N for the number of kernels/neurons, K for kernel size, S for stride and P for zero padding. Concerning the layers, CONCAT means concatenation layer, CONV means convolutional layer, BN means batch normalization layer with running average coefficient $\beta = 0.9$ and learnable affine transformation, DECONV means transpose convolutional layer, FC means fully connected layer, SUM corresponds to element-wise sum layer and RESIDUAL denotes a residual block, detailed at Table 8. The additional layers can be clearly understood. Finally, particular parameters for specific layers are defined between parenthesis after the layers' names

Table 8 Architecture of the residual block employed in the DGPose encoder

RESIDUAL Layer	
Input: *previous_layer_output*	
Layer	**Definition**
1	CONV-(N512, K3, S1, P1), BN, ReLU
2	CONV-(N512, K3, S2, P1), BN
3	SUM(CONV-2, *previous_layer_output*)

References

3Lateral: 3Lateral. (2018). http://www.3lateral.com/.

Achilles, F., Ichim, A. E., Coskun, H., Tombari, F., Noachtar, S., & Navab, N. (2016). Patient MoCap: Human pose estimation under blanket occlusion for hospital monitoring applications. In *MICCAI*.

Andriluka, M., Pishchulin, L., Gehler, P., & Schiele, B. (2014). 2d human pose estimation: New benchmark and state of the art analysis. In *CVPR*.

Balakrishnan, G., Zhao, A., Dalca, A. V., Durand, F., & Guttag, J. (2018). Synthesizing images of humans in unseen poses. In *CVPR*

de Bem, R., Arnab, A., Sapienza, M., Golodetz, S., & Torr, P. (2018) Deep fully-connected part-based models for human pose estimation. In *ACML*.

de Bem, R., Ghosh, A., Ajanthan, T., Miksik, O., Siddharth, N., & Torr, P. (2018). A semi-supervised deep generative model for human body analysis. In *ECCV (HBUGEN)*.

de Bem, R., Ghosh, A., Ajanthan, T., Siddharth, N., & Torr, P. (2019). A conditional deep generative model of people in natural images. In *WACV*.

Blanz, V., Vetter, T., et al. (1999). A morphable model for the synthesis of 3d faces. In *SIGGRAPH*.

Boeing: William Fetter's Boeing Man. (2018). https://secure.boeingimages.com/archive/William-Fetter-s-Boeing-Man-2F3XC5YCZNC.html.

Bogo, F., Romero, J., Loper, M., & Black, M. J. (2014). Faust: Dataset and evaluation for 3d mesh registration. In *CVPR* (pp. 3794–3801)

Borshukov, G., Piponi, D., Larsen, O., Lewis, J. P., & Tempelaar-lietz, C. (2005). Universal capture-image-based facial animation for "The Matrix Reloaded". In *SIGGRAPH*.

Bulat, A., & Tzimiropoulos, G. (2016). Human pose estimation via convolutional part heatmap regression. In *ECCV*.

Cao, Z., Simon, T., Wei, S. E., & Sheikh, Y. (2017). Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*.

Chan, C., Ginosar, S., Zhou, T., & Efros, A. A. (2018). Everybody dance now. arXiv preprint arXiv:1808.07371.

Chu, X., Yang, W., Ouyang, W., Ma, C., Yuille, A. L., & Wang, X. (2017). Multi-context attention for human pose estimation. In *CVPR*.

de La Gorce, M., Fleet, D. J., & Paragios, N. (2011). Model-based 3D hand pose estimation from monocular video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(9), 1793–1805.

Elgammal, A., & Lee, C. S. (2004). Inferring 3d body pose from silhouettes using activity manifold learning. In *CVPR*.

Enzweiler, M., & Gavrila, D. M. (2008). A mixed generative-discriminative framework for pedestrian classification. In *CVPR*.

Esser, P., Sutter, E., & Ommer, B. (2018). A variational u-net for conditional appearance and shape generation. In *CVPR*.

Ezzat, T., & Poggio, T. (1996). Facial analysis and synthesis using image-based models. In *FG*.

Fan, S., Ng, T. T., Koenig, B. L., Herberg, J. S., Jiang, M., Shen, Z., et al. (2018). Image visual realism: From human perception to machine computation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(9), 2180–2193.

Fei-Fei, L., & Perona, P. (2005). A Bayesian hierarchical model for learning natural scene categories. In *CVPR* (Vol. 2).

Fetter, W. A. (1982). A progression of human figures simulated by computer graphics. *IEEE Computer Graphics and Applications*, *2*(9), 9–13.

Fleuret, F., Berclaz, J., Lengagne, R., & Fua, P. (2007). Multicamera people tracking with a probabilistic occupancy map. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(2), 267–282.

Fossati, A., Dimitrijevic, M., Lepetit, V., & Fua, P. (2007). Bridging the gap between detection and tracking for 3d monocular video-based motion capture. In *CVPR* (pp. 1–8).

Franco, J. S., & Boyer, E. (2005). Fusion of multi-view silhouette cues using a space occupancy grid. In *ICCV* (pp. 1747–1753).

Geisler, W. S. (2008). Visual perception and the statistical properties of natural scenes. *Annual Review of Psychology*, *59*, 167–192.

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning*. Cambridge: MIT Press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In *NIPS*.

Hattori, H., Lee, N., Boddeti, V. N., Beainy, F., Kitani, K. M., & Kanade, T. (2018). Synthesizing a scene-specific pedestrian detector and pose estimator for static video surveillance. *International Journal of Computer Vision*, *126*(9), 1027–1044.

Hattori, H., Naresh Boddeti, V., Kitani, K. M., & Kanade, T. (2015). Learning scene-specific pedestrian detectors without real data. In *CVPR*.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*.

Hilton, A., Beresford, D., Gentils, T., Smith, R., & Sun, W. (1999). Virtual people: Capturing human models to populate virtual worlds. In *Proceedings of computer animation* (pp. 174–185).

Ian Spriggs. (2018). http://www.ianspriggs.com/.

Ichim, A. E., Bouaziz, S., & Pauly, M. (2015). Dynamic 3d avatar creation from hand-held video input. *ACM Transactions on Graphics (ToG)*, *34*(4), 45.

Insafutdinov, E., Pishchulin, L., Andres, B., Andriluka, M., & Schiele, B. (2016). Deepercut: A deeper, stronger, and faster multi-person pose estimation model. In *ECCV*.

Ionescu, C., Papava, D., Olaru, V., & Sminchisescu, C. (2014). Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(7), 1325–1339.

Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2016). Image-to-image translation with conditional adversarial networks. arXiv preprint arXiv:1611.07004.

Jhuang, H., Gall, J., Zuffi, S., Schmid, C., & Black, M. J. (2013). Towards understanding action recognition. In *ICCV*.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., & Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093.

Johnson, S., & Everingham, M. (2010). Clustered pose and nonlinear appearance models for human pose estimation. In *BMVC*.

Kanade, T., Rander, P., & Narayanan, P. J. (1997). Virtualized reality: Constructing virtual worlds from real scenes. *IEEE Multimedia*, *4*(1), 34–47.

Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2018). Progressive growing of gans for improved quality, stability, and variation. In *ICLR*.

Kay, K. N., Naselaris, T., Prenger, R. J., & Gallant, J. L. (2008). Identifying natural images from human brain activity. *Nature*, *452*(7185), 352.

Kingma, D., & Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR*.

Kingma, D. P., Mohamed, S., Rezende, D. J., & Welling, M. (2014). Semi-supervised learning with deep generative models. In *NIPS*.

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

Krizhevsky, A., Hinton, G., et al. (2010). Factored 3-way restricted Boltzmann machines for modeling natural images. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*.

Kulkarni, T. D., Whitney, W. F., Kohli, P., & Tenenbaum, J. (2015). Deep convolutional inverse graphics network. In *NIPS*.

Larsen, A. B. L., Sønderby, S. K., Larochelle, H., & Winther, O. (2016). Autoencoding beyond pixels using a learned similarity metric. In *ICML*.

Lassner, C., Pons-Moll, G., & Gehler, P. V. (2017). A generative model for people in clothing. In *ICCV*.

Lee, C. S., & Elgammal, A. (2005). Facial expression analysis using nonlinear decomposable generative models. In *International workshop on analysis and modeling of faces and gestures*.

Liang, X., Liu, S., Shen, X., Yang, J., Liu, L., Dong, J., Lin, L., & Yan, S. (2015). Deep human parsing with active template regression. In *TPAMI*.

Liu, Z., Luo, P., Qiu, S., Wang, X., & Tang, X. (2016). Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *CVPR*.

Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., & Black, M. J. (2015). SMPL: A skinned multi-person linear model. *ACM Transactions on Graphics (TOG)*, *34*(6), 1–16.

Ma, L., Jia, X., Sun, Q., Schiele, B., Tuytelaars, T., & Gool, L. V. (2017). Pose guided person image generation. In *NIPS*.

Ma, L., Sun, Q., Georgoulis, S., Van Gool, L., Schiele, B., & Fritz, M. (2018). Disentangled person image generation. In *CVPR*.

MacDorman, K. F., Green, R. D., Ho, C. C., & Koch, C. T. (2009). Too real for comfort? Uncanny responses to computer generated faces. *Computers in Human Behavior*, *25*(3), 695–710.

Magnenat-Thalmann, N., & Thalmann, D. (2005). Virtual humans: Thirty years of research, what next? *Visual Computer*, *21*(12), 997–1015.

Magnenat-Thalmann, N., & Thalmann, D. (2006). *Handbook of virtual humans*. Hoboken: Wiley.

MakeHuman. (2018). http://www.makehumancommunity.org/.

Massiceti, D., Siddharth, N., Dokania, P., & Torr, P. H. (2018). FlipDial: A generative model for two-way visual dialogue. In *CVPR*.

Massive Software. (2018). http://www.massivesoftware.com/.

Moeslund, T. B., Hilton, A., Krüger, V., & Sigal, L. (2011). *Visual analysis of humans*. Berlin: Springer.

Müller, M., Casser, V., Lahoud, J., Smith, N., & Ghanem, B. (2018). Sim4cv: A photo-realistic simulator for computer vision applications. *International Journal of Computer Vision*, *126*(9), 902–919.

NASA . (1995). Space flight human-system standard volume 1. Technical report NASA-STD-3001, National Aeronautics and Space Administration—NASA

Neverova, N., Alp Guler, R., & Kokkinos, I. (2018). Dense pose transfer. In *ECCV*.

Newell, A., Yang, K., & Deng, J. (2016). Stacked hourglass networks for human pose estimation. In *ECCV*.

Pighin, F., Hecker, J., Lischinski, D., Szeliski, R., & Salesin, D. H. (2006). Synthesizing realistic facial expressions from photographs. In *SIGGRAPH* (p. 19). ACM.

Poser. (2018). https://www.posersoftware.com/.

Pumarola, A., Agudo, A., Sanfeliu, A., & Moreno-Noguer, F. (2018). Unsupervised person image synthesis in arbitrary poses. In *CVPR*.

Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *ICML*.

Rhodin, H., Salzmann, M., & Fua, P. (2018). Unsupervised geometry-aware representation for 3d human pose estimation. In *ECCV*.

Rogez, G., & Schmid, C. (2016). Mocap-guided data augmentation for 3d pose estimation in the wild. In *NIPS*.

Rogez, G., & Schmid, C. (2018). Image-based synthesis for deep 3d human pose estimation. *International Journal of Computer Vision*, *126*(9), 993–1008.

Romero, J., Tzionas, D., & Black, M. J. (2017). Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics (ToG)*, *36*(6), 245.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*.

Rosales, R., Athitsos, V., Sigal, L., & Sclaroff, S. (2001). 3d hand pose reconstruction using specialized mappings. In *ICCV*.

Schulman, J., Heess, N., Weber, T., & Abbeel, P. (2015). Gradient estimation using stochastic computation graphs. In *NIPS*.

Seemann, E., Nickel, K., & Stiefelhagen, R. (2004). Head pose estimation using stereo vision for human–robot interaction. In *FG*.

Shan, Q., Adams, R., Curless, B., Furukawa, Y., & Seitz, S. M. (2013). The visual turing test for scene reconstruction. In *3DV*.

Shotton, J., Fitzgibbon, A. W., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., & Blake, A. (2011). Real-time human pose recognition in parts from single depth images. In *CVPR*.

Siarohin, A., Sangineto, E., Lathuilière, S., & Sebe, N. (2018). Deformable gans for pose-based human image generation. In *CVPR*.

Siddharth, N., Paige, B., Desmaison, A., van de Meent, J. W., Wood, F., Goodman, N. D., et al. (2017). Learning disentangled representations with semi-supervised deep generative models. *In*,. NIPS.

Simoncelli, E. P., & Olshausen, B. A. (2001). Natural image statistics and neural representation. *Annual Review of Neuroscience*, *24*(1), 1193–1216.

Sohn, K., Lee, H., & Yan, X.(2015). Learning structured output representation using deep conditional generative models. In *NIPS*.

Starck, J., & Hilton, A. (2007). Surface capture for performance-based animation. *IEEE Computer Graphics and Applications*, *27*(3), 21–31.

Starck, J., Miller, G., & Hilton, A. (2005). Video-based character animation. In *SIGGRAPH* (pp. 49–58). ACM

Theis, L., van den Oord, A., & Bethge, M. (2016). A note on the evaluation of generative models. In *ICLR*.

Thies, J., Zollhöfer, M., Stamminger, M., Theobalt, C., & Nießner, M. (2016). Face2face: Real-time face capture and reenactment of rgb videos. In *CVPR*.

Tompson, J., Jain, A., LeCun, Y., & Bregler, C. (2014). Joint training of a convolutional network and a graphical model for human pose estimation. In *NIPS*.

Trumble, M., Gilbert, A., Hilton, A., & Collomosse, J. (2018). Deep autoencoder for combined human pose estimation and body model upscaling. In *ECCV18*.

Tulyakov, S., Liu, M. Y., Yang, X., & Kautz, J. (2018). Mocogan: Decomposing motion and content for video generation. In *CVPR*.

Unreal Engine. (2018). https://www.unrealengine.com.

Valenza, E., Simion, F., Cassia, V. M., & Umiltà, C. (1996). Face preference at birth. *Journal of Experimental Psychology: Human perception and performance*, *22*(4), 892–903.

Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M. J., Laptev, I., & Schmid, C. (2017). Learning from synthetic humans. In *CVPR*.

von Marcard, T., Rosenhahn, B., Black, M., & Pons-Moll, G. (2017). Sparse inertial poser: Automatic 3d human pose estimation from sparse imus. Eurographics.

Walker, J., Marino, K., Gupta, A., & Hebert, M. (2017). The pose knows: Video forecasting by generating pose futures. In *ICCV*.

Wang, T. C., Liu, M. Y., Zhu, J. Y., Yakovenko, N., Tao, A., Kautz, J., Catanzaro, B. (2018). Video-to-video synthesis. In *NIPS*.

Wang, Y., Huang, X., Lee, C. S., Zhang, S., Li, Z., Samaras, D., et al. (2004). High resolution acquisition, learning and transfer of dynamic 3-d facial expressions. *Computer Graphics Forum*, *23*(3), 677–686.

Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, *13*(4), 600–612.

Wang, Z., Merel, J. S., Reed, S. E., de Freitas, N., Wayne, G., & Heess, N. (2017). Robust imitation of diverse behaviors. In *NIPS*.

Wei, S. E., Ramakrishna, V., Kanade, T., & Sheikh, Y. (2016). Convolutional pose machines. In *CVPR*.

Yang, W., Li, S., Ouyang, W., Li, H., & Wang, X. (2017). Learning feature pyramids for human pose estimation. In *ICCV*.
Yang, Y., & Ramanan, D. (2011). Articulated pose estimation with flexible mixtures-of-parts. In *CVPR*.
Yuille, A., & Kersten, D. (2006). Vision as bayesian inference: Analysis by synthesis? *Trends in Cognitive Sciences*, *10*(7), 301–308.
Zanfir, M., Popa, A. I., Zanfir, A., & Sminchisescu, C. (2018). Human appearance transfer. In *CVPR*.
Zhang, Y., Guo, Y., Jin, Y., Luo, Y., He, Z., & Lee, H. (2018). Unsupervised discovery of object landmarks as structural representations. In *CVPR*.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

D
STEER Paper

Contributions:

- Arnab Ghosh came up with the initial idea and set up the pipelines for the initial experimentation. This was in conjunction with weekly meeetings with Harkirat Singh Behl, Vinay Namboodiri, Emilien Dupont and Philip Torr.
- All of the authors helped in writing and refining the manuscript.

STEER: Simple Temporal Regularization For Neural ODEs

Arnab Ghosh
University of Oxford
arnabg@robots.ox.ac.uk

Harkirat Singh Behl
University of Oxford
harkirat@robots.ox.ac.uk

Emilien Dupont
University of Oxford
emilien.dupont@stats.ox.ac.uk

Philip H. S. Torr
University of Oxford
phst@robots.ox.ac.uk

Vinay Namboodiri
University of Bath
vpn22@bath.ac.uk

Abstract

Training Neural Ordinary Differential Equations (ODEs) is often computationally expensive. Indeed, computing the forward pass of such models involves solving an ODE which can become arbitrarily complex during training. Recent works have shown that regularizing the dynamics of the ODE can partially alleviate this. In this paper we propose a new regularization technique: randomly sampling the end time of the ODE during training. The proposed regularization is simple to implement, has negligible overhead and is effective across a wide variety of tasks. Further, the technique is orthogonal to several other methods proposed to regularize the dynamics of ODEs and as such can be used in conjunction with them. We show through experiments on normalizing flows, time series models and image recognition that the proposed regularization can significantly decrease training time and even improve performance over baseline models.

1 Introduction

Neural Ordinary Differential Equations (ODE) are an elegant approach for building continuous depth neural networks [4]. They offer various advantages over traditional neural networks, for example a constant memory cost as a function of depth and invertibility by construction. As they are inspired by dynamical systems, they can also take advantage of the vast amount of mathematical techniques developed in this field. Incorporating continuous depth also enables efficient density estimation [18] and modeling of irregularly sampled time series [46].

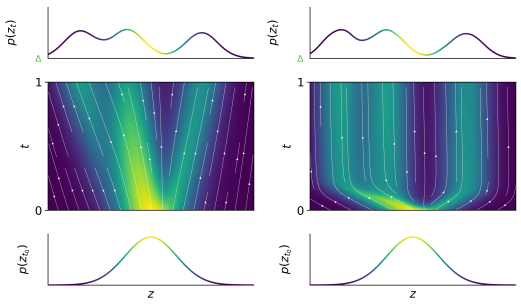


Figure 1: Flow fields of a standard Neural ODE (left) vs one equipped with STEER (right). These models transform a simple initial distribution $p(z_{t_0})$ to the target distribution $p(z_t)$. STEER learns simpler flows, which enable faster training.

The aforementioned advantages come at the expense of a significantly increased computational cost compared with traditional models like Residual Networks [21]. The training dynamics of Neural ODEs become more complicated as training progresses and the number of steps taken by the adaptive ODE solver (typically referred to as the number of function evaluations) can become prohibitively large [14]. In fact, several works have argued that this is one of the most

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

pressing limitations of Neural ODEs [4, 14, 16]. It is therefore important to design methods that can help reduce the computational cost of this tool.

A natural approach for tackling this issue is to explore regularization techniques that simplify the underlying dynamics for these models. Indeed, simpler dynamics have shown benefits in terms of faster convergence and a lower number of function evaluations both during training and testing [14]. [16] recently proposed an interesting technique for explicitly regularizing the dynamics based on ideas from optimal transport showing significant improvements in the effective convergence time. In this work, we propose a regularization technique in order to further accelerate training. Our regularization is orthogonal to previous approaches and can therefore be used in conjunction with them. Our technique is extremely simple: during training we randomly sample the upper limit of the ODE integration.

Specifically, the solver in Neural ODEs takes as input the integration limits (t_0, t_1). We propose to introduce regularization by randomly perturbing the end time parameter t_1 to be $t_1 \pm \delta$ where δ is a random variable. Empirically, we demonstrate that, in a wide variety of tasks, the proposed regularization achieves comparable performance to unregularized models at a significantly lower computational cost. We test our regularization technique on continuous normalizing flows [18], irregularly sampled time series [46] and image recognition tasks and show that it improves performance across all these tasks. It also performs better in certain instances of Stiff ODEs. Stiff ODEs are a widely studied topic in the field of engineering[6]. During training the dynamics learnt by the implicit model parametrized by the neural network can turn into an instance of a Stiff ODE. Techniques which could tackle the instances of Stiff ODE would prove to be instrumental as these implicit deep learning models mature.

2 Related Work

Connections between deep neural networks and dynamical systems were made as early as [31]. The adjoint sensitivity technique was explored in [44]. Some early libraries such as dolfin [15, 45], implemented the adjoint method for neural network frameworks. Previous works have shown the connection between residual networks and dynamical systems [47, 19, 36, 53, 51, 1]. By randomly dropping some blocks of a residual network, it was demonstrated that deep neural networks could be trained [23] or tested [50, 17] with stochastic depth, and that such a training procedure produces ensembles of shallower networks.

Neural ODEs [4] introduced a framework for training ODE models with backpropagation. Recent works [38, 58, 5, 25, 37, 9, 20, 40] have tried to analyze this framework theoretically and empirically. Neural ODEs were extended to work with flow based generative models [18]. In contrast to standard flow based models such as [29, 11, 43, 42], continuous normalizing flows do not put any restrictions on the model architecture. Furthermore, Neural ODEs have been applied in the context of irregularly sampled time series models [46], reinforcement learning [13] and for set modeling[34]. Neural ODEs have also been explored for harder tasks such as video generation [57, 27] and for modeling the evolving dynamics of graph neural networks [55]. Recent work has also extended Neural ODEs to stochastic differential equations [33, 35, 22, 41, 54, 26].

The methods most relevant to our work are [28, 16] and [14]. A regularization based on principles of optimal transport which simplifies the dynamics of continuous normalizing flow models, was introduced in [16] to make training faster. Equations which are easier to learn were analyzed in [28] whereby they used a differential surrogate for the time cost of standard numerical solvers, using higher-order derivatives of solution trajectories. Augmented Neural ODEs [14] introduced a simple modification to Neural ODEs by adding auxillary dimensions to the ODE function and showing how this simplified the dynamics. Our work is complementary to these methods, and we empirically demonstrate that our method can be used to further improve the performance of these techniques.

There is a huge treasure of work which deals with instability in training ODEs[8, 7]. Curtiss et al. [6] were one of the first works that dealt with solving stiff equations. It was observed in Curtiss and Hirschfelder [6] that explicit methods failed for the numerical solution of ordinary differential equations that model certain chemical reactions. It is very difficult to integrate "stiff" equations by ordinary numerical methods. Small errors are rapidly magnified if the equations are integrated. The other case might be that the errors oscillate rapidly. Implicit methods such as the second order Adams-Moulton (AM2) [52] are elegant techniques which deal with some of the problems of stiff ODEs. In the context of Neural ODEs[4] implicit computations reduces the efficiency of the solvers.

This effect is especially apparent in complex modeling tasks such as generative modeling using continuous normalizing flows[18] and irregularly sampled timeseries models[46]. Hence an analysis of simple scenarios of stiff equations which cause problems for standard Neural ODEs are discussed in the following sections.

3 Methodology

3.1 Neural ODEs

Neural ODEs can be interpreted as a continuous equivalent of Residual networks [21]. The transformation of a residual layer is $\mathbf{z}_{t+1} = \mathbf{z}_t + f_t(\mathbf{z}_t)$, where the representation at layer t is $\mathbf{z}_t$ and $f_t(\mathbf{z}_t)$ is a function parametrized by a neural network. The output dimension of the function $f_t(\mathbf{z}_t)$ should be the same as the input dimension to allow it to be added to the previous layer's features $\mathbf{z}_t$. This transformation can be rewritten as

$$\frac{\mathbf{z}_{t+\delta_t} - \mathbf{z}_t}{\delta_t} = f_t(\mathbf{z}_t), \quad \delta_t = 1, \tag{1}$$

which can be interpreted as finite difference approximation of the derivative. In contrast to traditional deep neural networks where $\delta_t = 1$ is fixed, Neural ODEs [4] introduced a continuous version by analysing Equation 1 in the limit $\delta_t \to 0$. The resulting equation becomes

$$\frac{d\mathbf{z}(t)}{dt} = f(\mathbf{z}(t), t). \tag{2}$$

To transform an input $\mathbf{x}$ to an output $\mathbf{y}$, Residual networks use a sequence of k functions to transform the representation. In an analogous manner, Neural ODEs evolve the representation from the initial time $t = t_0,\ \mathbf{z}(t_0) = \mathbf{x}$, to the final time $t = t_1,\ \mathbf{z}(t_1) = \mathbf{y}$. The representation $\mathbf{z}(t)$ at any given time t can then be obtained by solving Equation 2 using an off-the-shelf ODE solver with the function $f(\mathbf{z}(t), t)$ parameterized by the neural network. To update the parameters of the function $f(\mathbf{z}(t), t)$ such that $\mathbf{z}(t_1) \approx \mathbf{y}$ we backpropagate through the operations of the ODE solver to obtain the appropriate gradients.

3.2 Temporal Regularization

The formulation of a standard Neural ODE is given by

$$\mathbf{z}(t_1) = \mathbf{z}(t_0) + \int_{t_0}^{t_1} f(\mathbf{z}(t), t, \theta)dt = \text{ODESolve}(\mathbf{z}(t_0), f, t_0, t_1, \theta), \tag{3}$$

where $\mathbf{z}(t)$ is the latent representation at any time t. θ are the parameters of the function f and (t_0, t_1) are the limits of the integration. The initial condition of the ODE is set as $\mathbf{z}(t_0) = \mathbf{x}$.

Neural ODEs often take an excessively large time to train, especially in the case of continuous normalizing flows [18] . The dynamics become more complicated as training progresses thus leading to an increase in the number of function evaluations. In the context of Neural ODEs, the dynamics of the ODE and training time are intricately related. In practice, simpler dynamics [14, 16] often lead to faster training with fewer function evaluations. Regularization is commonly used in machine learning to ease the training of a model and improve its generalization. Some common instances include Batchnorm [24] and early stopping [56].

Neural ODEs continuously evolve the representation $\mathbf{z}(t)$ with time t. Thus in the context of neural ODEs, an alternative type of regularization is possible, namely temporal regularization. In this work, we propose a simple regularization technique that exploits this temporal property. We propose to perturb the final time t_1 for the integration of the Neural ODE in Equation 3 during training. At each training iteration, the final time t_1 for the integration is stochastically sampled. More formally, if we

assume $t_0 < t_1$, then our regularization can be formulated as

$$\begin{aligned} \mathbf{z}(t_1) &= \mathbf{z}(t_0) + \int_{t_0}^{T} f(\mathbf{z}(t), t, \theta)dt = \text{ODESolve}(\mathbf{z}(t_0), f, t_0, T, \theta), \\ T &\sim \text{Uniform}(t_1 - b, t_1 + b), \\ b &< t_1 - t_0 : \text{parameter controlling interval of uniform distribution.} \end{aligned} \tag{4}$$

We call the technique STEER, which stands for Simple Temporal Regularization. Since Neural ODEs model a continuous series of transformations, applying STEER is essentially equivalent to performing a variable number of transformations on the input $\mathbf{x}$ in each training iteration, thereby introducing a new form of stochasticity in the training process.

We observe that the proposed regularization effectively ensures convergence to the solution at time $t_1 - b$. The effective behavior of $\mathbf{z}(t)$ can be see in Fig. 2

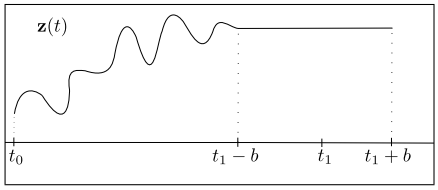


Figure 2: Effective behavior of Neural ODEs with STEER regularization. The solution is reached at a time $t_1 - b$ instead of time t_1.

3.3 Stiff ODEs

Stiff ODEs have been extensively researched in the field of differential equations [2]. Inspite of years of research a concrete definition of Stiff ODEs has not been agreed upon [49]. We highlight how some of the problems of Stiff ODEs could impact the training dynamics of Neural ODEs in specific scenarios. Certain discussions of Stiff ODEs define that the stiffness concerns the underlying mathematical form of the equation, integration time and a set of initial data.

In terms of the mathematical equation, stiffness is a problem that can arise when numerically solving ODEs. In such systems there is typically a fast and a slow transient component which causes adaptive ODE solvers to take exceedingly small steps. This increases the number of function evaluations and the time taken to integrate the function. Furthermore, in Neural ODEs, the function $\frac{d\mathbf{z}}{dt}$ is parameterized by a neural network which is updated during training, and could become stiff. This would be hard for a traditional ODE solver to integrate efficiently, thereby making this a limitation.

Stiffness ratio is one of the techniques used to characterize some forms of stiff equations. The stiffness ratio is used to characterize the stiffness for the general case of linear constant coefficient systems. These systems can be represented as

$$\frac{d\mathbf{z}}{dt} = \mathbf{A}\mathbf{z} + f(t), \tag{5}$$

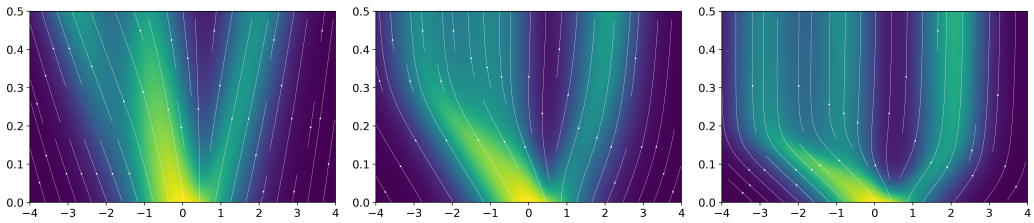
where $\mathbf{z}, f(t) \in \mathbb{R}^n$ and $\mathbf{A}$ is a $n \times n$ matrix with eigenvalues $\lambda_i \in \mathbb{C}, i = 1, 2, \ldots, n$ (assumed distinct) and corresponding eigenvectors $\mathbf{c_i} \in \mathbb{C}^n, i = 1, 2, \ldots, n$. The eigenvalues and eigenvectors can be complex. The general solution of Equation 9 takes the form

$$\mathbf{z}(t) = \sum_{i=1}^{n} \kappa_i \exp(\lambda_i t)\mathbf{c}_i + \mathbf{g}(t). \tag{6}$$

If λ_i is complex, the term $\exp(\lambda_i t)$ varies sinusoidally. The stiffness ratio is defined as $\frac{\max_{i=1\ldots n}|Re(\lambda_i)|}{\min_{i=1\ldots n}|Re(\lambda_i)|}$, where $Re(\lambda_i)$ denotes the real component of $\lambda_i \in \mathbb{C}$. Ordinarily, temporal regularization might fail in the face of stiffness. However, in our empirical experiments we observed that at least in the case of toy problems, STEER actually performed better. This analysis is provided in the next section.

4 Experiments

In this section, we demonstrate the effectiveness of the proposed method on several classes of problems. First, we conduct experiments on continuous normalizing flow based generative models,



$b = 0$ $b = 0.25$ $b = 0.375$

Figure 3: Plots showing the path taken by FFJORD with STEER regularization. STEER regularization ($b = 0.25$ and $b = 0.375$) performs the same transformation in shorter paths.

and compare against FFJORD [18] and RNODE [16]. Then, we conduct experiments on irregularly sampled time-series models, and compare against the models and baselines proposed in Latent ODEs [46]. We also conduct experiments on feedforward models for image recognition, and compare against Neural ODEs and Augmented Neural ODEs [14]. We finally evaluate STEER and standard Neural ODEs on a toy stiff ODE experiment.

4.1 Generative Models

Continuous normalizing flow based generative models [18, 4] provide an alternate perspective to generative modeling. Let $p(\mathbf{x})$ be the likelihood of a datapoint $\mathbf{x}$ and $p(\mathbf{z})$ be the likelihood of the latent $\mathbf{z}$. FFJORD [18] showed that under certain conditions the change in log probability also follows a differential equation $\frac{d\log(p(\mathbf{z}(t)))}{dt} = -\text{tr}\left(\frac{df}{d\mathbf{z}(t)}\right)$. Thus the total change in log probability can be obtained by integrating the equation which leads to

$$\log p(\mathbf{z}(t_1)) = \log p(\mathbf{z}(t_0)) - \int_{t_0}^{t_1} \text{tr}\left(\frac{df}{d\mathbf{z}(t)}\right) dt. \tag{7}$$

This allows to compute the point $\mathbf{z_0}$ which generates a given datapoint $\mathbf{x}$, and $\log p(\mathbf{x})$ under the model by solving the initial value problem:

$$\underbrace{\begin{bmatrix} \mathbf{z_0} \\ \log p(\mathbf{x}) - \log p_{\mathbf{z_0}}(\mathbf{z_0}) \end{bmatrix}}_{\text{solutions}} = \underbrace{\int_{T}^{t_0} \begin{bmatrix} f(\mathbf{z}(t), t; \theta) \\ -\text{tr}\left(\frac{df}{d\mathbf{z}(t)}\right) \end{bmatrix} dt}_{\text{dynamics}}, \qquad \underbrace{\begin{bmatrix} \mathbf{z}(T) \\ \log p(\mathbf{x}) - \log p(\mathbf{z}(T)) \end{bmatrix} = \begin{bmatrix} \mathbf{x} \\ \mathbf{0} \end{bmatrix}}_{\text{initial values}}. \tag{8}$$

The combined dynamics of $\mathbf{z}(t)$ and the log-density of the sample is integrated backwards in time from T to t_0. The log likelihood $\log p(\mathbf{x})$ can be computed by adding $\log p_{\mathbf{z_0}}(\mathbf{z_0})$ to the solution of Equation 8. A major drawback of continuous normalizing flow based models is extremely slow convergence. STEER regularization enables faster convergence by sampling $T \sim \text{Uniform}(t_1 - b, t_1 + b)$ at each iteration.

We focus on the task of density estimation of image datasets using the same architectures and experimental setup as FFJORD [18]. More details are provided in the supplementary material. The performance of the model is evaluated on multiple datasets, namely MNIST [32], CIFAR-10 [30] and ImageNet-32 [10], and the results are shown in Table 1. The performance is also analyzed in the simple scenario of a 1D Mixture of Gaussians as shown in Fig. 3.

It can be seen in Table 1 that models with STEER regularization converge much faster. In some cases, such as on CIFAR-10, our model is 30% faster than the state of the art.

Fig. 3 shows the evolution of the probability distribution from the latent distribution $p(\mathbf{z})$ to the data distribution $p(\mathbf{x})$ at different test times. It can be seen that the trajectories reach the final states at an earlier test time with STEER regularization.

4.2 Times Series Models

For time series models, the function is evaluated at multiple times $t_0, ..t_n$. We employ the regularization between every consecutive pair of points (t_i, t_{i+1}). More formally, if the time is split into

	MNIST		CIFAR-10		IMAGENET-32	
	BITS/DIM	TIME	BITS/DIM	TIME	BITS/DIM	TIME
FFJORD, VANILLA.	0.974	68.47	3.40	> 5 days	3.96	> 5 days
FFJORD RNODE. [16]	0.973	24.37	3.398	31.84	2.36	30.1
FFJORD, VANILLA. + STEER ($b = 0.5$)	0.974	51.21	3.40	86.34	3.84	> 5 days
FFJORD, VANILLA. + STEER ($b = 0.25$)	0.976	58.21	3.41	103.4	3.87	> 5 days
FFJORD RNODE + STEER ($b = 0.5$)	**0.971**	**16.32**	**3.397**	**22.24**	**2.35**	**24.9**
FFJORD RNODE + STEER ($b = 0.25$)	0.972	19.71	3.401	25.24	2.37	27.84

Table 1: Test accuracy and training time comparison on various image datasets. Unless mentioned as days, the time reported is in hours.

Model	**Accuracy**
Latent ODE (ODE enc)	0.846 ± 0.013
Latent ODE (RNN enc.)	0.835 ± 0.010
ODE-RNN	0.829 ± 0.016
Latent ODE (ODE enc, STEER)	$\mathbf{0.880 \pm 0.011}$
Latent ODE (RNN enc, STEER)	0.865 ± 0.021
ODE-RNN (STEER)	0.872 ± 0.012

Table 2: Per-time-point classification accuracies on the Human Activity dataset.

Model	**AUC**
Latent ODE (ODE enc)	0.829 ± 0.004
Latent ODE (RNN enc.)	0.781 ± 0.018
ODE-RNN	$\mathbf{0.833 \pm 0.009}$
Latent ODE (ODE enc, STEER)	0.810 ± 0.018
Latent ODE (RNN enc, STEER)	0.772 ± 0.014
ODE-RNN (STEER)	0.811 ± 0.007

Table 3: Per-sequence classification AUC on Physionet.

consecutive intervals $(t_0, t_1)...(t_{n-1}, t_n)$, we start with t_0 and the initialization $\mathbf{z}(t_0)$ and sample the end time as $T \sim \text{Uniform}(t_1 - b, t_1 + b)$. The resulting integration using the ODE solver gives us the latent representation $\mathbf{z}(t_1)$. We repeat the same procedure where the limits of integration are now (t_1, t_2). In this iteration t_1 is not altered while $T \sim \text{Uniform}(t_2 - b, t_2 + b)$ is sampled to obtain $\mathbf{z}(t_2)$ and so on. In the case of irregularly sampled time series models, the length of the interval $t_{i+1} - t_i$ can itself vary. The parameter b in this case is adaptive such that $b = |t_{i+1} - t_i| - \epsilon$. The ϵ parameter is added to avoid numerical instability issues. In the case of time series models the parameter b changes at each forward pass thus adding another source of stochasticity.

Latent ODEs [46] introduced irregularly sampled time series models in which Neural ODEs perform much better. Indeed these datasets break the basic assumptions of a standard RNN, namely that the data are not necessarily collected at regular time intervals. This is the case in several real life scenarios such as patient records or human activity recognition which are observed at irregularly sampled timepoints.

The Mujoco experiments consist of a physical simulation of the "Hopper" model from the Deepmind control suite. The initial condition uniquely determines the trajectory. It consists of $10,000$ sequences of 100 time points each. The Human Activity dataset [46] consists of sensor data fitted to individuals. The sensor data is collected using tags attached to the belt, chest and ankles of human performers. It results in 12 features in total. The task involves classifying each point as to which activity was being performed. Physionet [48] consists of time series data from initial 48 hours of ICU admissions. It consists of sparse measurements where only some of the features' measurements are taken at an instant. The measurement frequency is also irregular which makes the dataset quite challenging. To

	Interpolation (% Observed Pts.)				Extrapolation (% Observed Pts.)			
Model	10 %	20 %	30 %	50 %	10 %	20 %	30 %	50 %
Latent ODE (ODE enc)	0.360	0.295	0.300	0.285	1.441	1.400	1.175	1.258
Latent ODE (RNN enc.)	2.477	0.578	2.768	0.447	1.663	1.653	1.485	1.377
ODE-RNN	1.647	1.209	0.986	0.665	13.508	31.950	15.465	26.463
Latent ODE (ODE enc, STEER)	**0.230**	**0.235**	**0.246**	**0.210**	**1.01**	**0.98**	**1.01**	**1.04**
Latent ODE (RNN enc., STEER)	2.31	0.561	2.534	0.386	1.541	1.498	1.236	1.119
ODE-RNN (STEER)	1.528	1.09	0.796	0.603	11.508	29.34	14.23	23.528

Table 4: Test Mean Squared Error (MSE) ($\times 10^{-2}$) on the MuJoCo dataset.

	NODE	NODE(STEER)	ANODE	ANODE(STEER)
MNIST	$97.2\% \pm 0.2$	$98.3\% \pm 0.3$	$98.2\% \pm 0.1$	$\mathbf{98.6\% \pm 0.3}$
CIFAR-10	$53.7\% \pm 0.4$	$54.9\% \pm 0.5$	$60.6\% \pm 0.4$	$\mathbf{62.1\% \pm 0.2}$
SVHN	$81.0\% \pm 0.6$	$81.8\% \pm 0.3$	$83.5\% \pm 0.5$	$\mathbf{84.1 \pm 0.2}$

Table 5: Test accuracies on Feedforward models on various datsets.

predict mortality, binary classifiers were constructed by passing the hidden state to a 2 layer binary classifier.

We use the same architecture and experiment settings as used in Latent ODEs [46]. For the Mujoco experiment 15 and 30 latent dimensions were used for the generative and recognition models respectively. The ODE function had 500 units in each of the 3 layers. A 15 dimensional latent state was used for the Human Activity dataset. For the Physionet experiments the ODE function has 3 layers with 50 units each. The classifier consisted of 2 layers with 300 units each.

In the case of irregularly sampled timeseries with Latent ODEs, the number of function evaluations is not a major concern since the training time is quite small, so we compare only in terms of accuracy. It can be seen in Table 2 and Table 4 that STEER regularization consistently improves the performance of ODE based models. Table 4 demonstrates that the extrapolation results are much better with STEER regularization, indicating that it may help improve the generalization of the model. However in the case of Physionet Table 3, the results do not improve over standard Neural ODEs, although the performance remains very close.

4.3 Feedforward Models

In feedforward learning for classification and regression, the Neural ODE framework can be used to map the input data $\mathbf{x} = \mathbf{z}(t_0) \in \mathbb{R}^d$ to features $\mathbf{z}(t) \in \mathbb{R}^d$ at any given time t. For standard neural ODEs, one integrates in the range (t_0, t_1), and for STEER the integration is performed in the range (t_0, T) where $T \in \text{Uniform}(t_1 - b, t_1 + b)$. Once the features $\mathbf{z}(T)$ at the final time T are obtained, a linear layer $\mathbb{R}^d \to \mathbb{R}$ is used to map the features to the correct dimension for downstream tasks such as classification or regression.

We evaluate STEER on a similar set of experiments as performed in [14] on the MNIST [32], CIFAR-10 [30] and SVHN [39] datasets using convolutional architectures for $f(\mathbf{z}(t), t)$. We use the same architectures and experimental settings as used in [14] for both Neural ODEs (NODE) and Augmented Neural ODEs (ANODE). For the MNIST experiments, 92 filters were used for NODEs while 64 filters were used for ANODEs to compensate for 5 augmented dimensions. For the CIFAR-10 and SVHN experiments, 125 filters were used for NODEs while 64 filters were used for ANODEs with 10 augmented dimensions. Such a design was employed to ensure a roughly similar number of parameters for a fair comparison.

Table 5 demonstrates the performance of STEER regularization with $b = 0.99$. STEER provides a small boost in accuracy and performs consistently across all datasets.

4.4 A Toy Example of Stiff ODE

In this experiment, we analyze a toy example of stiff ODE from [3]. The aim of this experiment is to highlight the vulnerability of Neural ODEs to stiffness. The experimental setting for section 4.4 was similar to the one used in the official github repository [1] of Neural ODE. Instead of using a 2-D ODE we used a 1-D ODE to be able to better visualize and analyze the behavior. We used Eq 12 in the paper as mentioned in the numerical methods book by [3] on page 752. It was surprising that a Neural ODE was able to model a 2 dimensional model with ease while the stiff equation was extremely hard for the Neural ODE to model.

The ODE is given by

$$\frac{dy}{dt} = -1000y + 3000 - 2000e^{-t}, \tag{9}$$

[1] https://github.com/rtqichen/torchdiffeq/blob/master/examples/ode_demo.py

with the initial condition of $y(0) = 0$. The solution for the above stiff ODE is $y = 3 - 0.998e^{-1000t} - 2.002e^{-t}$. We use a simple architecture with a single hidden layer of dimension 500 for both the standard Neural ODE and the one with STEER. The effective input dimension is 2 resulting by concatenating y and t, and the output dimension is 1.

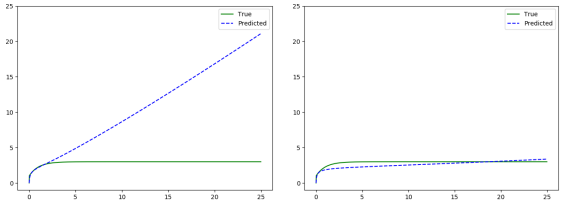


Figure 4: Standard Neural ODE struggles while STEER regularization helps to fit a stiff ODE with stiffness ratio $r = 1000$.

As shown in Fig. 4, the solution is initially dominated by the fast exponential term e^{-1000t}. After a short period ($t < 0.005$), this transient dies down and the solution is dictated by the slow exponential e^{-t}. The training interval $[0, 15]$ is broken down into small intervals of length $(t_1 - t_0) = 0.125$. We sample 1000 random initial points t_0 and use $y(t_0)$ as the initialization and then try to estimate the value of the function $y(t_1)$ at $t_1 = t_0 + 0.125$. It can be seen in Fig. 4 that standard Neural ODE doesn't show the same asymptotic behavior as the actual solution, whereas STEER regularization enables better asymptotic behavior. The minimum Mean Squared Error(MSE) loss over all the points of the trajectory in the test interval $[0, 25]$ for standard Neural ODE is 3.87, and with STEER regularization with $b = 0.124$ it is 0.52.

To further study the behaviour with varying stiffness ratio, the formula in Equation 9 is generalized as $\frac{dy}{dt} = -ry + 3r - 2re^{-t}$. The generalized equation has the same asymptotic behavior as the original. It also reaches a steady state at $y = 3$. The $-ry$ term would introduce a term of e^{-rt} and another term of e^{-t} would be present in the solution. Thus the stiffness ratio as described in Subsection 3.3 is exactly r. The plot in Fig. 5 shows that as the stiffness increases, the STEER regularization compares better in terms of MSE to models without the regularization. The experiments were performed using the dormand-prince [12] ODE solver.

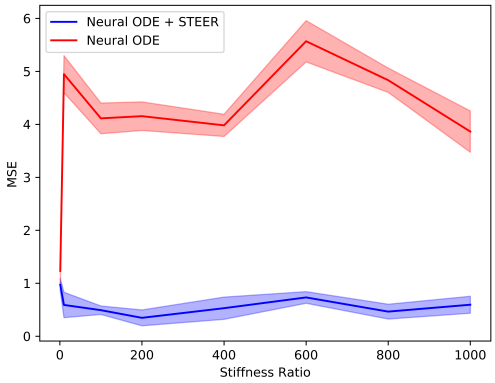


Figure 5: Model with STEER regularization achieves a much lower MSE on a Stiff ODE as the stiffness ratio increases.

While we do not promise to have proposed a solution for the difficult problem of Stiff-ODEs [3], we empirically observed that it gives improvement on toy examples. We want to highlight that further research is needed in this important area of Stiffness. We show analysis about the choice of b for our technique in the appendix.

4.5 A Toy example of a Timeseries Model:

In this experiment, we analyze a toy example of irregularly sampled timeseries models. The experimental setting was the same as in [46]. The models were tested on a toy dataset of periodic trajectories. The initial point was sampled from a standard Gaussian. Gaussian noise was added to the subsequent observations. Each trajectory has a 100 observations. A fixed subset of these points from each trajectory are sampled. The entire set of 100 observations are attempted to be reconstructed. In the original experiment by [46] the amplitude was the same throughout the trajectories while the frequency was varied for the $A.sin(\phi t)$ where ϕ is the frequency of the sine component. We made a simple change to the experimental setting whereby we also added a t component to the sampled periodic trajectories. Thus the final form of the trajectories being sampled are $A.sin(\phi t) + t$ where A is constant and ϕ can vary within the sampled trajectories. In this experimental setting we see that standard Neural ODEs start to produce trajectories which are qualitatively quite different from the actual sampled trajectories. STEER on the other hand produces trajectories which match the qualitative behavior of the sampled trajectories more closely.

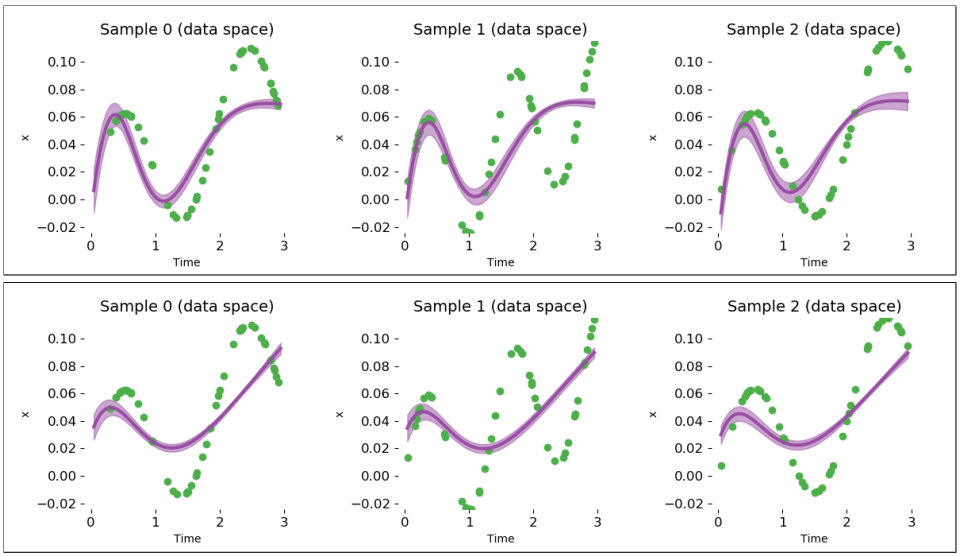


Figure 6: A simple addition of t to $A.sin(\phi t)$ changes the qualitative performance of Neural ODE vs STEER. The top row demonstrates the 3 sample trajectories that have been irregularly sampled using STEER, while the bottom row is with a standard Neural ODE. The green points indicate the irregularly sampled points, while the rest of the points are predicted using each trained model. The amplitude A of the model is the same across the various samples but the frequency ϕ is variable across the samples. As we see from the sampled trajectories STEER models the qualitative behavior of the trajectories better than standard Neural ODEs.

5 Conclusion

In this paper we introduced a simple temporal regularization method for Neural ODEs. The regularization consists of perturbing the final time of the integration of the ODE solver. The main advantage of our work is the empirical evidence showing the regularization results in simpler dynamics for Neural ODEs models on a wide variety of tasks. This is demonstrated for a specific stiff ODE that is challenging for a basic Neural ODE solver, faster continuous normalizing flow models and improved or comparable accuracy for time series models and deep feedforward networks. We further show that our approach is orthogonal to other regularization approaches and can be thus combined with them to further improve Neural ODEs.

6 Broader Impact

Neural ODEs operate under the broad umbrella of 'Implicit Methods in Deep Learning'. The last decade in deep learning was characterized by layer based deep neural networks. The limits of that framework are starting to approach whereby one can fit only so many layers in the GPU of a computer. Thus, alternate forms of computational frameworks are emerging whereby the function is not explicitly modeled. Rather a pseudo function is learnt from which the original function that needed to be approximated can be retrieved. This shows a different model of computation and previous work has shown signs that it is much more parameter efficient than resnet based deep neural networks.

Our work is in the broad area of Neural ODEs which provides continuous depth neural networks. This line of work would have a similar impact as most supervised deep learning techniques. However,current continuous depth neural networks show limitations that are improved through our proposed regularization scheme. Thus, our models converge faster during training by reducing the number of function evaluations and effective training time. The resulting trained models also require fewer function evaluations at inference, thus reducing the computation at test time too. Our research could potentially encourage the search for more such efficient techniques for training neural ODEs.

Although we have highlighted some drawbacks of existing Neural ODEs in the context of Stiff ODEs. We do not fully understand why our solution works better in the case of stiff ODEs. The issue of stiff ODEs has a rich history of research in engineering. Inevitably during training, there is a possibility that the dynamics learnt by the neural network might be an instance of a stiff equation. This cannot be prevented with the current formulation. As Neural ODEs become more stable and mature we need to understand the effects of stiffness. These models can be truly useful if we understand it from all possible points of failure.

Acknowledgments and Disclosure of Funding

Arnab Ghosh was funded by the University of Oxford through a studentship using combined corporate gifts from Microsoft and Technicolor. Harkirat Singh Behl was supported using a Tencent studentship through the University of Oxford. Emilien Dupont was funded by the University of Oxford through a Deepmind studentship. Philip H.S. Torr was supported by the Royal Academy of Engineering under the Research Chair and Senior Research Fellowships scheme, EPSRC/MURI grant EP/N019474/1 and FiveAI. Vinay Namboodiri was funded using Startup funding support from University of Bath.

References

[1] Avelin, B. and Nyström, K. (2019). Neural odes as the deep limit of resnets with constant weights. *arXiv preprint arXiv:1906.12183*.

[2] Brugnano, L., Mazzia, F., and Trigiante, D. (2011). Fifty years of stiffness. In *Recent Advances in Computational and Applied Mathematics*, pages 1–21. Springer.

[3] Chapra, S. C., Canale, R. P., et al. (2010). *Numerical methods for engineers*. Boston: McGraw-Hill Higher Education,.

[4] Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583.

[5] Cuchiero, C., Larsson, M., and Teichmann, J. (2019). Deep neural networks, generic universal interpolation, and controlled odes. *arXiv preprint arXiv:1908.07838*.

[6] Curtiss, C. F. and Hirschfelder, J. O. (1952). Integration of stiff equations. *Proceedings of the National Academy of Sciences of the United States of America*, 38(3):235.

[7] Dahlquist, G. (1976). Error analysis for a class of methods for stiff non-linear initial value problems. In *Numerical analysis*, pages 60–72. Springer.

[8] Dahlquist, G. (1985). 33 years of numerical instability, part i. *BIT Numerical Mathematics*, 25(1):188–204.

[9] Davis, J. Q., Choromanski, K., Varley, J., Lee, H., Slotine, J.-J., Likhosterov, V., Weller, A., Makadia, A., and Sindhwani, V. (2020). Time dependence in non-autonomous neural odes. *arXiv preprint arXiv:2005.01906*.

[10] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database.

[11] Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016). Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*.

[12] Dormand, J. R. and Prince, P. J. (1980). A family of embedded runge-kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26.

[13] Du, J., Futoma, J., and Doshi-Velez, F. (2020). Model-based reinforcement learning for semi-markov decision processes with neural odes. *arXiv preprint arXiv:2006.16210*.

[14] Dupont, E., Doucet, A., and Teh, Y. W. (2019). Augmented neural odes. In *Advances in Neural Information Processing Systems*, pages 3134–3144.

[15] Farrell, P. E., Ham, D. A., Funke, S. W., and Rognes, M. E. (2013). Automated derivation of the adjoint of high-level transient finite element programs. *SIAM Journal on Scientific Computing*, 35(4):C369–C393.

[16] Finlay, C., Jacobsen, J.-H., Nurbekyan, L., and Oberman, A. M. (2020). How to train your neural ode. In *International Conference in Machine Learning*.

[17] Ghosh, A., Zhang, R., Dokania, P. K., Wang, O., Efros, A. A., Torr, P. H., and Shechtman, E. (2019). Interactive sketch & fill: Multiclass sketch-to-image translation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1171–1180.

[18] Grathwohl, W., Chen, R. T., Bettencourt, J., Sutskever, I., and Duvenaud, D. (2018). Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*.

[19] Haber, E. and Ruthotto, L. (2017). Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004.

[20] Hanshu, Y., Jiawei, D., Vincent, T., and Jiashi, F. (2019). On robustness of neural ordinary differential equations. In *International Conference on Learning Representations*.

[21] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[22] Hodgkinson, L., van der Heide, C., Roosta, F., and Mahoney, M. W. (2020). Stochastic normalizing flows. *arXiv preprint arXiv:2002.09547*.

[23] Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. (2016). Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer.

[24] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

[25] Jabir, J.-F., Šiška, D., and Szpruch, Ł. (2019). Mean-field neural odes via relaxed optimal control. *arXiv preprint arXiv:1912.05475*.

[26] Jia, J. and Benson, A. R. (2019). Neural jump stochastic differential equations. In *Advances in Neural Information Processing Systems*, pages 9843–9854.

[27] Kanaa, D., Voleti, V., Kahou, S., and Pal, C. (2019). Simple video generation using neural odes.

[28] Kelly, J., Bettencourt, J., Johnson, M. J., and Duvenaud, D. (2020). Learning differential equations that are easy to solve. *arXiv preprint arXiv:2007.04504*.

[29] Kingma, D. P. and Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224.

[30] Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

[31] Lagaris, I. E., Likas, A., and Fotiadis, D. I. (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000.

[32] LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., and Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404.

[33] Li, X., Wong, T.-K. L., Chen, R. T., and Duvenaud, D. (2020a). Scalable gradients for stochastic differential equations. *arXiv preprint arXiv:2001.01328*.

[34] Li, Y., Yi, H., Bender, C. M., Shan, S., and Oliva, J. B. (2020b). Exchangeable neural ode for set modeling. *arXiv preprint arXiv:2008.02676*.

[35] Liu, X., Xiao, T., Si, S., Cao, Q., Kumar, S., and Hsieh, C.-J. (2019). Neural sde: Stabilizing neural ode networks with stochastic noise. *arXiv preprint arXiv:1906.02355*.

[36] Lu, Y., Zhong, A., Li, Q., and Dong, B. (2017). Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. *arXiv preprint arXiv:1710.10121*.

[37] Massaroli, S., Poli, M., Bin, M., Park, J., Yamashita, A., and Asama, H. (2020a). Stable neural flows. *arXiv preprint arXiv:2003.08063*.

[38] Massaroli, S., Poli, M., Park, J., Yamashita, A., and Asama, H. (2020b). Dissecting neural odes. *arXiv preprint arXiv:2002.08071*.

[39] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and N, A. Y. (2011). Reading digits in natural images with unsupervised feature learning.

[40] Norcliffe, A., Bodnar, C., Day, B., Simidjievski, N., and Liò, P. (2020). On second order behaviour in augmented neural odes. *arXiv preprint arXiv:2006.07220*.

[41] Oganesyan, V., Volokhova, A., and Vetrov, D. (2020). Stochasticity in neural odes: An empirical study. *arXiv preprint arXiv:2002.09779*.

[42] Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2019). Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*.

[43] Papamakarios, G., Pavlakou, T., and Murray, I. (2017). Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347.

[44] Pearlmutter, B. A. (1995). Gradient calculations for dynamic recurrent neural networks: A survey. *IEEE Transactions on Neural networks*, 6(5):1212–1228.

[45] Rackauckas, C. and Nie, Q. (2017). Differentialequations. jl–a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of Open Research Software*, 5(1).

[46] Rubanova, Y., Chen, T. Q., and Duvenaud, D. K. (2019). Latent ordinary differential equations for irregularly-sampled time series. In *Advances in Neural Information Processing Systems*, pages 5321–5331.

[47] Ruthotto, L. and Haber, E. (2019). Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, pages 1–13.

[48] Silva, I., Moody, G., Scott, D. J., Celi, L. A., and Mark, R. G. (2012). Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012. In *2012 Computing in Cardiology*, pages 245–248. IEEE.

[49] Söderlind, G., Jay, L., and Calvo, M. (2015). Stiffness 1952–2012: Sixty years in search of a definition. *BIT Numerical Mathematics*, 55(2):531–558.

[50] Veit, A., Wilber, M. J., and Belongie, S. (2016). Residual networks behave like ensembles of relatively shallow networks. In *NIPS*, pages 550–558.

[51] Wang, B., Shi, Z., and Osher, S. (2019). Resnets ensemble via the feynman-kac formalism to improve natural and robust accuracies. In *Advances in Neural Information Processing Systems*, pages 1655–1665.

[52] Wanner, G. and Hairer, E. (1996). *Solving ordinary differential equations II*. Springer Berlin Heidelberg.

[53] Weinan, E. (2017). A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11.

[54] Wu, H., Köhler, J., and Noé, F. (2020). Stochastic normalizing flows. *arXiv preprint arXiv:2002.06707*.

[55] Xhonneux, L.-P. A., Qu, M., and Tang, J. (2019). Continuous graph neural networks. *arXiv preprint arXiv:1912.00967*.

[56] Yao, Y., Rosasco, L., and Caponnetto, A. (2007). On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315.

[57] Yildiz, C., Heinonen, M., and Lahdesmaki, H. (2019). Ode2vae: Deep generative second order odes with bayesian neural networks. In *Advances in Neural Information Processing Systems*, pages 13412–13421.

[58] Zhang, H., Gao, X., Unterman, J., and Arodz, T. (2019). Approximation capabilities of neural ordinary differential equations. *arXiv preprint arXiv:1907.12998*.

Bibliography

[1] M. Abadi and D. Andersen. "Learning to Protect Communications with Adversarial Neural Cryptography". In: *arXiv preprint arXiv:1610.06918* (2016).

[2] AK. URL: https://twitter.com/ak92501/status/1408169728833691648.

[3] Guillaume Alain et al. "GSNs: generative stochastic networks". In: *Information and Inference: A Journal of the IMA* 5.2 (2016), pp. 210–249.

[4] M. Arjovsky and L. Bottou. *Towards Principled Methods For Training Generative Adversarial Networks*. Tech. rep. 2017.

[5] M. Arjovsky, S. Chintala, and L. Bottou. "Wasserstein gan". In: *ICML* (2017).

[6] Martín Arjovsky and Léon Bottou. "Towards Principled Methods for Training Generative Adversarial Networks". In: *ICLR*. 2017.

[7] S. Arora et al. "Generalization and Equilibrium in Generative Adversarial Nets (GANs)". In: *International Conference on Machine Learning*. 2017.

[8] Benny Avelin and Kaj Nyström. "Neural ODEs as the Deep Limit of ResNets with constant weights". In: *arXiv preprint arXiv:1906.12183* (2019).

[9] David Berthelot, Tom Schumm, and Luke Metz. "Began: Boundary equilibrium generative adversarial networks". In: *arXiv preprint arXiv:1703.10717* (2017).

[10] C Bishop. "Pattern Recognition and Machine Learning (Information Science and Statistics), 1st edn. 2006. corr. 2nd printing edn". In: *Springer, New York* (2007).

[11] Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large scale gan training for high fidelity natural image synthesis". In: *ICLR* (2019).

[12] Luigi Brugnano, Francesca Mazzia, and Donato Trigiante. "Fifty years of stiffness". In: *Recent Advances in Computational and Applied Mathematics*. Springer, 2011, pp. 1–21.

[13] Steven C Chapra, Raymond P Canale, et al. *Numerical methods for engineers*. Boston: McGraw-Hill Higher Education, 2010.

[14] T. Che et al. *Mode Regularized Generative Adversarial Networks*. Tech. rep. ICLR. 2017.

[15] Shu-Yu Chen et al. "DeepFaceDrawing: Deep Generation of Face Images from Sketches". In: *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2020)* 39.4 (2020), 72:1–72:16.

[16] Tian Qi Chen et al. “Neural ordinary differential equations”. In: *Advances in neural information processing systems*. 2018, pp. 6571–6583.

[17] Wengling Chen and James Hays. “SketchyGAN: towards diverse and realistic sketch to image synthesis”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 9416–9425.

[18] X. Chen et al. “Infogan: Interpretable representation learning by information maximizing generative adversarial nets”. In: *NIPS* (2016).

[19] Kyunghyun Cho et al. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *EMNLP* (2014).

[20] Dale J Cohen and Susan Bennett. “Why can’t most people draw what they see?” In: *Journal of Experimental Psychology: Human Perception and Performance* 23.3 (1997), p. 609.

[21] Forrester Cole et al. “Where do people draw lines?” In: *ACM Transactions on Graphics (TOG)* 27.3 (2008), p. 88.

[22] Katherine Crowson. URL: `https://github.com/crowsonkb/v-diffusion-pytorch`.

[23] Christa Cuchiero, Martin Larsson, and Josef Teichmann. “Deep neural networks, generic universal interpolation, and controlled ODEs”. In: *arXiv preprint arXiv:1908.07838* (2019).

[24] Charles Francis Curtiss and Joseph O Hirschfelder. “Integration of stiff equations”. In: *Proceedings of the National Academy of Sciences of the United States of America* 38.3 (1952), p. 235.

[25] Germund Dahlquist. “33 years of numerical instability, Part I”. In: *BIT Numerical Mathematics* 25.1 (1985), pp. 188–204.

[26] Germund Dahlquist. “Error analysis for a class of methods for stiff non-linear initial value problems”. In: *Numerical analysis*. Springer, 1976, pp. 60–72.

[27] Jared Quincy Davis et al. “Time Dependence in Non-Autonomous Neural ODEs”. In: *arXiv preprint arXiv:2005.01906* (2020).

[28] Tali Dekel et al. “Sparse, smart contours to represent and edit images”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3511–3520.

[29] J. Deng et al. “ImageNet: A Large-Scale Hierarchical Image Database”. In: 2009.

[30] Emily L Denton et al. “Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks”. In: *NIPS*. 2015.

[31] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. “Density estimation using real nvp”. In: *arXiv preprint arXiv:1605.08803* (2016).

[32] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. "Adversarial feature learning". In: *ICLR* (2017).

[33] John R Dormand and Peter J Prince. "A family of embedded Runge-Kutta formulae". In: *Journal of computational and applied mathematics* 6.1 (1980), pp. 19–26.

[34] Jianzhun Du, Joseph Futoma, and Finale Doshi-Velez. "Model-based Reinforcement Learning for Semi-Markov Decision Processes with Neural ODEs". In: *arXiv preprint arXiv:2006.16210* (2020).

[35] Vincent Dumoulin et al. "Adversarially learned inference". In: *ICLR* (2017).

[36] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. "Augmented neural odes". In: *Advances in Neural Information Processing Systems*. 2019, pp. 3134–3144.

[37] I. Durugkar, I. Gemp, and S. Mahadevan. "Generative Multi-Adversarial Networks". In: *ICLR* (2017).

[38] Alexei A Efros and Thomas K Leung. "Texture synthesis by non-parametric sampling". In: *ICCV*. 1999.

[39] Mathias Eitz et al. "Photosketcher: interactive sketch-based image synthesis". In: *IEEE Computer Graphics and Applications* 31.6 (2011), pp. 56–66.

[40] Robert K Englund. "The Origins of Script: Before Writing. Denise Schmandt-Besserat. In two volumes. Vol. 1, From Counting to Cuneiform. xviii, 269 pp., illus. 60.Vol.2,ACatalogofNearEasternTokens.xxxvi,416pp.,illus. 85. University of Texas Press, Austin, 1992." In: *Science* 260.5114 (1993), pp. 1670–1671.

[41] Patrick Esser, Robin Rombach, and Bjorn Ommer. "Taming transformers for high-resolution image synthesis". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 12873–12883.

[42] Patrick E Farrell et al. "Automated derivation of the adjoint of high-level transient finite element programs". In: *SIAM Journal on Scientific Computing* 35.4 (2013), pp. C369–C393.

[43] Chris Finlay et al. "How to train your neural ode". In: *International Conference in Machine Learning*. 2020.

[44] Yaroslav Ganin et al. "Synthesizing programs for images using reinforced adversarial learning". In: *ICML* (2018).

[45] A. Ghosh, V. Kulharia, and V. Namboodiri. "Message Passing Multi-Agent GANs". In: *arXiv preprint arXiv:1612.01294* (2016).

[46] A. Ghosh et al. "Contextual RNN-GANs for Abstract Reasoning Diagram Generation". In: 2017.

[47] Arnab Ghosh et al. "Interactive sketch & fill: Multiclass sketch-to-image translation". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 1171–1180.

[48] Arnab Ghosh et al. “Multi-agent diverse generative adversarial networks”. In: *CVPR* (2018).

[49] I. Goodfellow et al. “Generative adversarial nets”. In: *NIPS*. 2014.

[50] Ian Goodfellow. “NIPS 2016 tutorial: Generative adversarial networks”. In: *arXiv preprint arXiv:1701.00160* (2016).

[51] Ian Goodfellow et al. “Generative adversarial nets”. In: *NIPS*. 2014.

[52] Will Grathwohl et al. “Ffjord: Free-form continuous dynamics for scalable reversible generative models”. In: *arXiv preprint arXiv:1810.01367* (2018).

[53] Ishaan Gulrajani et al. “Improved training of wasserstein gans”. In: *NIPS*. 2017, pp. 5767–5777.

[54] David Ha and Douglas Eck. “A neural representation of sketch drawings”. In: *Conference on Neural Information Processing Systems* (2017).

[55] Eldad Haber and Lars Ruthotto. “Stable architectures for deep neural networks”. In: *Inverse Problems* 34.1 (2017), p. 014004.

[56] YAN Hanshu et al. “On Robustness of Neural Ordinary Differential Equations”. In: *International Conference on Learning Representations*. 2019.

[57] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[58] Martin Heusel et al. “Gans trained by a two time-scale update rule converge to a local nash equilibrium”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 6626–6637.

[59] Geoffrey E Hinton and Ruslan R Salakhutdinov. “Reducing the dimensionality of data with neural networks”. In: *Science* 313.5786 (2006), pp. 504–507.

[60] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[61] Liam Hodgkinson et al. “Stochastic Normalizing Flows”. In: *arXiv preprint arXiv:2002.09547* (2020).

[62] Gao Huang et al. “Deep networks with stochastic depth”. In: *European conference on computer vision*. Springer. 2016, pp. 646–661.

[63] Xun Huang and Serge J Belongie. “Arbitrary Style Transfer in Real-Time with Adaptive Instance Normalization.” In: *ICCV*. 2017, pp. 1510–1519.

[64] Xun Huang et al. “Multimodal Unsupervised Image-to-Image Translation”. In: *ECCV* (2018).

[65] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *arXiv preprint arXiv:1502.03167* (2015).

[66] P. Isola et al. “Image-to-Image Translation with Conditional Adversarial Networks”. In: *CVPR* (2017).

[67] Jean-François Jabir, David Šiška, and Łukasz Szpruch. “Mean-Field Neural ODEs via Relaxed Optimal Control”. In: *arXiv preprint arXiv:1912.05475* (2019).

[68] Junteng Jia and Austin R Benson. “Neural jump stochastic differential equations”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 9843–9854.

[69] T. Joachims, T. Finley, and C.-N. Yu. “Cutting-plane Training of Structural SVMs”. In: *Machine Learning* (2009).

[70] Felix Juefei-Xu, Vishnu Naresh Boddeti, and Marios Savvides. “Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking”. In: *arXiv preprint arXiv:1704.04865* (2017).

[71] David Kanaa et al. “Simple Video Generation using Neural ODEs”. In: (2019).

[72] Tero Karras, Samuli Laine, and Timo Aila. “A style-based generator architecture for generative adversarial networks”. In: *CVPR* (2019).

[73] Tero Karras et al. “Progressive Growing of GANs for Improved Quality, Stability, and Variation”. In: *ICLR*. 2018.

[74] Tero Karras et al. “Progressive growing of gans for improved quality, stability, and variation”. In: *ICLR* (2018).

[75] Jacob Kelly et al. “Learning differential equations that are easy to solve”. In: *arXiv preprint arXiv:2007.04504* (2020).

[76] D. Kingma and M. Welling. “Auto-encoding variational bayes”. In: *ICLR* (2014).

[77] Diederik P Kingma et al. “Semi-supervised learning with deep generative models”. In: *NIPS*. 2014.

[78] Durk P Kingma and Prafulla Dhariwal. “Glow: Generative flow with invertible 1x1 convolutions”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 10215–10224.

[79] Mario Klingemann. URL: `https://aiartists.org/mario-klingemann`.

[80] Aran Komatsuzaki. URL: `https://twitter.com/arankomatsuzaki/status/1399471244760649729`.

[81] Alex Krizhevsky, Geoffrey Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009).

[82] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *NIPS*. 2012, pp. 1097–1105.

[83] Solomon Kullback and Richard A Leibler. “On information and sufficiency”. In: *The annals of mathematical statistics* (1951).

[84] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. "Artificial neural networks for solving ordinary and partial differential equations". In: *IEEE transactions on neural networks* 9.5 (1998), pp. 987–1000.

[85] Anders Boesen Lindbo Larsen et al. "Autoencoding beyond pixels using a learned similarity metric". In: *ICML*. 2016.

[86] Anders Boesen Lindbo Larsen et al. "Autoencoding Beyond Pixels Using a Learned Similarity Metric". In: *ICML*. 2016.

[87] Christoph Lassner, Gerard Pons-Moll, and Peter V. Gehler. "A Generative Model for People in Clothing". In: *ICCV*. 2017.

[88] Yann LeCun et al. "Handwritten digit recognition with a back-propagation network". In: *Advances in neural information processing systems*. 1990, pp. 396–404.

[89] C. Ledig et al. "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network". In: *arXiv preprint arXiv:1609.04802* (2016).

[90] Yong Jae Lee, C Lawrence Zitnick, and Michael F Cohen. "Shadowdraw: real-time user guidance for freehand drawing". In: *ACM Transactions on Graphics (TOG)*. Vol. 30. 4. ACM. 2011, p. 27.

[91] Xuechen Li et al. "Scalable Gradients for Stochastic Differential Equations". In: *arXiv preprint arXiv:2001.01328* (2020).

[92] Yang Li et al. "Exchangeable Neural ODE for Set Modeling". In: *arXiv preprint arXiv:2008.02676* (2020).

[93] Yijun Li et al. "Im2Pencil: Controllable Pencil Illustration from Photographs". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1525–1534.

[94] M. Liu and O. Tuzel. "Coupled Generative Adversarial Networks". In: *NIPS* (2016).

[95] Xuanqing Liu et al. "Neural SDE: Stabilizing Neural ODE Networks with Stochastic Noise". In: *arXiv preprint arXiv:1906.02355* (2019).

[96] Andrew Lock. *On the recent origin of symbolically-mediated language and its implications for psychological science*. na, 1999.

[97] Yiping Lu et al. "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations". In: *arXiv preprint arXiv:1710.10121* (2017).

[98] Liqian Ma et al. "Pose Guided Person Image Generation". In: *NIPS*. 2017.

[99] Karl Marx and Friedrich Engels. *The communist manifesto*. Yale University Press, 2012.

[100] Stefano Massaroli et al. "Dissecting neural odes". In: *arXiv preprint arXiv:2002.08071* (2020).

[101] Stefano Massaroli et al. "Stable Neural Flows". In: *arXiv preprint arXiv:2003.08063* (2020).

[102] Michael Mathieu, Camille Couprie, and Yann LeCun. "Deep multi-scale video prediction beyond mean square error". In: *ICLR*. 2016.

[103] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. "Which training methods for GANs do actually converge?" In: *ICML* (2018).

[104] L. Metz et al. "Unrolled Generative Adversarial Networks". In: 2017.

[105] M. Mirza and S. Osindero. "Conditional generative adversarial nets". In: *arXiv preprint arXiv:1411.1784* (2014).

[106] Takeru Miyato et al. "Spectral normalization for generative adversarial networks". In: *ICLR* (2018).

[107] Y. Netzer et al. "Reading Digits in Natural Images with Unsupervised Feature Learning". In: 2011.

[108] Alexander Norcliffe et al. "On Second Order Behaviour in Augmented Neural ODEs". In: *arXiv preprint arXiv:2006.07220* (2020).

[109] Augustus Odena, Christopher Olah, and Jonathon Shlens. "Conditional image synthesis with auxiliary classifier gans". In: *ICML* (2017).

[110] Viktor Oganesyan, Alexandra Volokhova, and Dmitry Vetrov. "Stochasticity in Neural ODEs: An Empirical Study". In: *arXiv preprint arXiv:2002.09779* (2020).

[111] Aaron van den Oord et al. "Conditional image generation with pixelcnn decoders". In: *NIPS*. 2016.

[112] George Papamakarios, Theo Pavlakou, and Iain Murray. "Masked autoregressive flow for density estimation". In: *Advances in Neural Information Processing Systems*. 2017, pp. 2338–2347.

[113] George Papamakarios et al. "Normalizing Flows for Probabilistic Modeling and Inference". In: *arXiv preprint arXiv:1912.02762* (2019).

[114] Taesung Park et al. "Semantic Image Synthesis with Spatially-Adaptive Normalization". In: *CVPR* (2019).

[115] D. Pathak et al. "Context Encoders: Feature Learning by Inpainting". In: *CVPR* (2017).

[116] Barak A Pearlmutter. "Gradient calculations for dynamic recurrent neural networks: A survey". In: *IEEE Transactions on Neural networks* 6.5 (1995), pp. 1212–1228.

[117] Ethan Perez et al. "Film: Visual reasoning with a general conditioning layer". In: *AAAI* (2018).

[118] Tiziano Portenier et al. "Faceshop: Deep sketch-based face image editing". In: *ACM Transactions on Graphics (TOG)* 37.4 (2018), p. 99.

[119] Christopher Rackauckas and Qing Nie. "Differentialequations. jl–a performant and feature-rich ecosystem for solving differential equations in julia". In: *Journal of Open Research Software* 5.1 (2017).

[120] A. Radford, L. Metz, and S. Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *ICLR* (2016).

[121] Alec Radford et al. "Learning transferable visual models from natural language supervision". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8748–8763.

[122] Aditya Ramesh et al. "Zero-shot text-to-image generation". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8821–8831.

[123] S. Reed et al. "Generative adversarial text to image synthesis". In: *NIPS* (2016).

[124] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. "Stochastic backpropagation and approximate inference in deep generative models". In: *ICML*. 2014.

[125] Yulia Rubanova, Tian Qi Chen, and David K Duvenaud. "Latent Ordinary Differential Equations for Irregularly-Sampled Time Series". In: *Advances in Neural Information Processing Systems*. 2019, pp. 5321–5331.

[126] Lars Ruthotto and Eldad Haber. "Deep neural networks motivated by partial differential equations". In: *Journal of Mathematical Imaging and Vision* (2019), pp. 1–13.

[127] T. Salimans et al. "Improved techniques for training gans". In: *NIPS* (2016).

[128] Patsorn Sangkloy et al. "Scribbler: Controlling deep image synthesis with sketch and color". In: *CVPR*. Vol. 2. 2017.

[129] Abraham Savitzky and Marcel JE Golay. "Smoothing and differentiation of data by simplified least squares procedures." In: *Analytical chemistry* 36.8 (1964), pp. 1627–1639.

[130] Ryan Schmidt et al. "On expert performance in 3D curve-drawing tasks". In: *Proceedings of the 6th eurographics symposium on sketch-based interfaces and modeling*. ACM. 2009, pp. 133–140.

[131] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. "Singan: Learning a generative model from a single natural image". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 4570–4580.

[132] N Siddharth et al. "Learning Disentangled Representations with Semi-Supervised Deep Generative Models". In: *NIPS*. 2017.

[133] Ikaro Silva et al. "Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012". In: *2012 Computing in Cardiology*. IEEE. 2012, pp. 245–248.

[134] Edgar Simo-Serra et al. "Learning to simplify: fully convolutional networks for rough sketch cleanup". In: *ACM Transactions on Graphics (TOG)* 35.4 (2016), p. 121.

[135] Paul Smolensky. *Information processing in dynamical systems: Foundations of harmony theory*. Tech. rep. DTIC Document, 1986.

[136] Gustaf Söderlind, Laurent Jay, and Manuel Calvo. "Stiffness 1952–2012: Sixty years in search of a definition". In: *BIT Numerical Mathematics* 55.2 (2015), pp. 531–558.

[137] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. "Learning structured output representation using deep conditional generative models". In: 2015, pp. 3483–3491.

[138] Ivan E. Sutherland. "Sketch Pad a Man-machine Graphical Communication System". In: *Proceedings of the SHARE Design Automation Workshop*. DAC '64. New York, NY, USA: ACM, 1964, pp. 6.329–6.346. DOI: 10.1145/800265.810742. URL: http://doi.acm.org/10.1145/800265.810742.

[139] Christian Szegedy et al. "Rethinking the inception architecture for computer vision". In: *CVPR*. 2016, pp. 2818–2826.

[140] I. Tsochantaridis et al. "Support vector machine learning for interdependent and structured output spaces". In: 2004.

[141] D Ulyanov, A Vedaldi, and VS Lempitsky. *Instance normalization: the missing ingredient for fast stylization. CoRR abs/1607.0 (2016)*.

[142] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel recurrent neural networks". In: *International Conference on Machine Learning*. PMLR. 2016, pp. 1747–1756.

[143] Andreas Veit and Serge Belongie. "Convolutional Networks with Adaptive Inference Graphs". In: *ECCV* (2018).

[144] Andreas Veit, Michael J Wilber, and Serge Belongie. "Residual networks behave like ensembles of relatively shallow networks". In: *NIPS*. 2016, pp. 550–558.

[145] Pascal Vincent et al. "Extracting and composing robust features with denoising autoencoders". In: *ICML*. 2008.

[146] C. Vondrick, H. Pirsiavash, and A. Torralba. "Generating videos with scene dynamics". In: *NIPS* (2016).

[147] Bao Wang, Zuoqiang Shi, and Stanley Osher. "ResNets Ensemble via the Feynman-Kac Formalism to Improve Natural and Robust Accuracies". In: *Advances in Neural Information Processing Systems*. 2019, pp. 1655–1665.

[148] Ting-Chun Wang et al. "High-Resolution Image Synthesis and Semantic Manipulation With Conditional GANs". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[149] Ting-Chun Wang et al. "High-resolution image synthesis and semantic manipulation with conditional gans". In: *CVPR* (2018).

[150] Gerhard Wanner and Ernst Hairer. *Solving ordinary differential equations II*. Springer Berlin Heidelberg, 1996.

[151] E Weinan. "A proposal on machine learning via dynamical systems". In: *Communications in Mathematics and Statistics* 5.1 (2017), pp. 1–11.

[152] Hao Wu, Jonas Köhler, and Frank Noé. "Stochastic Normalizing Flows". In: *arXiv preprint arXiv:2002.06707* (2020).

[153] J. Wu et al. "Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling". In: *NIPS* (2016).

[154] Louis-Pascal AC Xhonneux, Meng Qu, and Jian Tang. "Continuous Graph Neural Networks". In: *arXiv preprint arXiv:1912.00967* (2019).

[155] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. "On early stopping in gradient descent learning". In: *Constructive Approximation* 26.2 (2007), pp. 289–315.

[156] Cagatay Yildiz, Markus Heinonen, and Harri Lahdesmaki. "ODE2VAE: Deep generative second order ODEs with Bayesian neural networks". In: *Advances in Neural Information Processing Systems*. 2019, pp. 13412–13421.

[157] Aron Yu and Kristen Grauman. "Fine-grained visual comparisons with local learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 192–199.

[158] Han Zhang et al. "Approximation capabilities of neural ordinary differential equations". In: *arXiv preprint arXiv:1907.12998* (2019).

[159] Han Zhang et al. "Self-Attention Generative Adversarial Networks". In: *ICML* (2019).

[160] Richard Zhang, Phillip Isola, and Alexei A Efros. "Colorful image colorization". In: *ECCV*. Springer. 2016, pp. 649–666.

[161] Bolei Zhou et al. "Places: A 10 million Image Database for Scene Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

[162] J. Zhu et al. "Generative visual manipulation on the natural image manifold". In: *ECCV*. 2016.

[163] Jun-Yan Zhu et al. "Toward multimodal image-to-image translation". In: 2017, pp. 465–476.

[164] Jun-Yan Zhu et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: *CVPR* (2017).