Influence of the Input Data on Learning Deep Representations

Sylvestre-Alvise Rebuffi
Wadham College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Supervisors: Prof. Andrea Vedaldi and Prof. Michael Osborne

Michaelmas 2020

To my parents, brothers, Angelika
and all the people I care for

Acknowledgements

I would like to thank first my supervisor Andrea Vedaldi for his patience and support during these four years. I am very grateful for the freedom of research he gave me and the many facets of research I learned from working with him.

I thank very much Christoph Lampert for the internship at IST Austria, which was the best start for my DPhil. I also thank Michael Osborne for his co-supervision as well as Andrew Zisserman, Pawan Kumar and Philip Torr for their great research feedbacks.

I specially thank my closest collaborators Sébastien Ehrhardt and Kai Han: it was a real pleasure to work with them. I also thank Hakan Bilen for the help at the start of my DPhil as well as Ruth Fong and Xu Ji for their patience, help and expertise.

I thank all the members of the VGG for the fun, the discussions and the enriching diversity, all packed in one overheated and overcooled floor.

I am very thankful to the teams of MathWorks Cambridge and Huawei Paris for the very nice experience there as intern. I gratefully acknowledge funding from MathWorks and ERC IDIU-638009.

I am very indebted to the excellent French educational system with the Classes Préparatoires (with warm thanks to Mathilde, Célia and Rémi) and the Grandes Écoles that taught me how to learn, study, work, research and, most importantly, take pleasure in Science.

Finally, I would like to express my warmest thanks to my better half Angelika, my brothers Jean-Baptiste, Louis-Sébastien, Pierre-Alexandre, and my parents Florence and Luigi.

Statement of Originality

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.

Sylvestre-Alvise Rebuffi, Wadham College

Abstract

This thesis studies, through the prism of image classification, the influence of the input data on learning deep representations. Indeed, data is abundant but also multifaceted: samples might come from a single or multiple domains, or they can also be more or less labelled. We show in this thesis that methods to learn deep network representations should be designed according to the type of input data at hand.

Our work covers different training data configurations. We start with the multiple visual domains setting. We tackle this challenge by modifying the network architecture with residual adapters, domain specific modules which lead to a high degree of parameter sharing across domains with better transfer learning performance than standard fine-tuning.

Second, we consider the settings where a part of the data is unlabelled. This thesis spans different levels of supervision but, in all cases, we promote the systematic use of self-supervision pre-training. First, we consider the semi-supervised setting with few labels per class for which we propose an alternating optimisation scheme to avoid overfitting the scarse labelled data. We then learn to discover new classes given some known classes by proposing a two-headed network which jointly learns to cluster the unlabelled data and to classify both the labelled and unlabelled samples. To this end, we propose a clustering loss which learns to connect pairs of samples based on their similarity at the feature level. We show that this loss, proposed for novel class discovery, can also be used to perform deep clustering by choosing a nearest neighbours based similarity.

From another perspective, we look at the influence of the data on the deep representations through the lens of saliency. To interpret which parts of the image contribute the most to the data representation, we introduce a unifying saliency framework based on the learning mechanism of convolutional neural networks. This framework can then be used to derive new or existing backpropagation based saliency methods.

Contents

Chapter 1

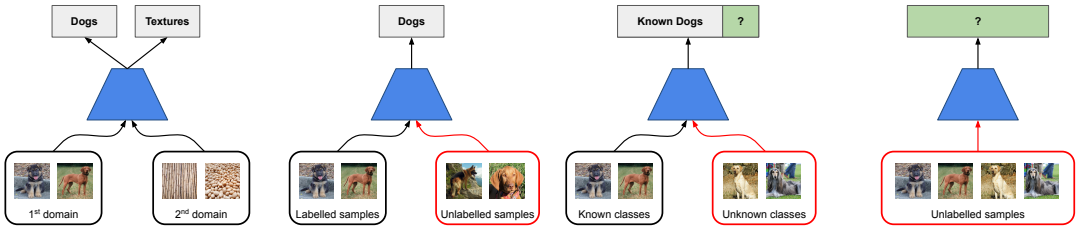
Introduction

1.1 Motivation

Since the advent of the deep learning era in computer vision [26, 83], representation learning moved from handcrafted approaches like SIFT [93] to using Convolutional Neural Networks (CNNs) [87] which provide representations optimised in an end-to-end manner for a target loss. After their initial success on image classification, deep neural networks have been successfully applied to a wide range of visual tasks such as semantic segmentation [90] or object detection [58] but also to other fields like audio [33] and text [144]. However, even for the standard image classification task, a CNN performance on the target test set will heavily depend on the input training data. Indeed, the test accuracy will depend on the number of labelled samples [9] as well as the quantity of extra unlabelled data [111]. Furthermore, a CNN will perform well only on the visual domain it has been trained on [15]. How does this variety of input data impact the design of deep representation methods?

In this thesis we propose to study this question through the prism of image classification in four data settings: (1) fully labelled data coming from diverse visual domains, (2) one visual domain but few labels per class, (3) novel class discovery given some known classes and (4) deep clustering with no training labels at all. These diverse input data configurations illustrated in Figure 1.1 correspond to real-life applications. In fact, data is abundant but might come from multiple sources and its annotation is a tedious and costly process, which might require labelling millions of images like for ImageNet [122] or expert knowledge like for medical datasets [77].

Next, to analyse the influence of the training data on the CNN representation from a different perspective, we also explore which spatial parts of the images contribute the most to the network representation. For this purpose, we introduce a principled



(a) Multi-domain (b) Semi-supervised (c) Class discovery (d) Clustering

Figure 1.1: **Input data configurations used in this thesis.** We cover diverse settings such as (a) multiple visual domains (chapters 3-4) or (b)(c)(d) different levels of supervision with the red frames indicating unlabelled data. We tackle: (b) semi-supervised learning with few labels per class (chapter 5), (c) novel class discovery given some known classes (chapter 6) and (d) deep clustering with no training labels (chapter 7).

framework based on the gradients of the weights to derive existing and new backpropagation saliency methods. The common objective of this thesis is to provide tools and techniques to both design representation learning methods according to the type of input data at hand and understand in which way the input data contributes to the network representation.

In this thesis we restrict ourselves to the image classification task but some of our proposed methods could also be transposed to other tasks like detection, or other modalities like text and speech. We organise this manuscript as follows. First, we introduce in Chapters 3 and 4 adaptive modules termed residual adapters which lead to a high degree of parameter sharing between visual domains. We then explore different levels of supervision: semi-supervised learning with few labels per class in Chapter 5, novel class discovery given some known classes in Chapter 6 and deep clustering with no training labels in Chapter 7. Finally, by reformulating saliency as the spatial contribution to the gradient of the network weights, we provide in Chapter 8 an interpretability tool to analyse which parts of the input image contribute the most to the data representation at the different stages of the network.

1.2 Thesis outline and contributions

In this section we present the contributions introduced in this thesis as well as the motivations behind them and the chapters to which they correspond.

1.2.1 Visual Decathlon Challenge

We start this thesis by studying the problem of image classification on multiple visual domains. To evaluate methods in this setting, we introduce in Chapter 3 a benchmark termed Visual Decathlon. The desired benchmark evaluation should indicate whether the tested model performs well on all of the considered visual domains. Thus, inspired by the scoring system of decathlon, we propose, on each dataset, to both normalise the score and enforce a minimum accuracy threshold to avoid domain sacrificing. To cover a wide range of visual domains, we select 10 classification datasets which vary in size (one large, five medium, four small) and in style (natural images, textures, digits, characters or videos encoded as dynamic images). Examples of the ten datasets can be seen in Figure 1.2.

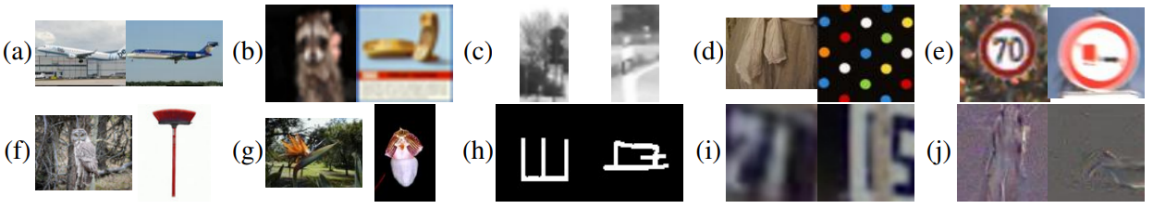


Figure 1.2: **Visual Decathlon.** We evaluate the classification performance of a single model on ten visual domains. We use the following datasets: (a) Aircraft [97], (b) CIFAR-100 [82], (c) Daimler Pedestrians [104], (d) Describable Textures [25], (e) German Traffic Signs [139], (f) ILSVRC (ImageNet) 2012 [122], (g) VGG-Flowers [108], (h) OmniGlot [85], (i) SVHN [105], (j) UCF101 Dynamic Images [136].

1.2.2 Introducing the residual adapters

In addition to the proposed benchmark for multi-domain image classification, our goal is to design a method achieving competitive classification performance with a high degree of parameter sharing across domains. Bilen et al. [15] proposed a shared single network with domain-specific classifiers and normalisation layers. However, we show using the Visual Decathlon evaluation metric that this approach performs worse than training a different model on each dataset. Inspired by this idea of domain-specific parametrisation of a shared backbone network, we introduce in Chapter 3 the residual

adapters, parameter efficient modules appended all along the backbone network to modulate its output.

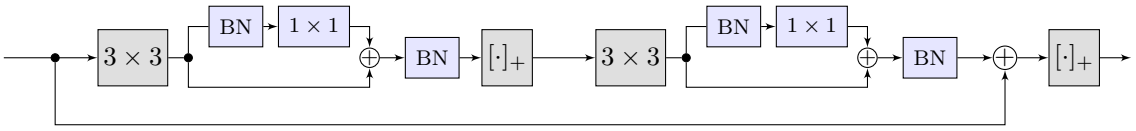
What we refer to as the backbone network is the 3×3 convolution layers of the network. In fact, after adding a first set of adapters to the randomly initialised network, the whole network (including the adapters) is pre-trained on the biggest dataset (ImageNet [122] in our case) and then we freeze all the 3×3 convolution layers in the network. When a new dataset needs to be trained, we replace the adapters and train the new adapters on the target dataset. To sum up, the frozen 3×3 filters are shared among domains and each dataset leverages these frozen filters by training its own set of adapters.

To design the adapters, we follow two main criteria: (1) few extra parameters compared to the size of the backbone network and (2) ability to control the model capacity depending on the size of the target dataset. To achieve the first criterion of small memory footprint, our adapters only contain 1×1 convolutions and batch normalisation layers. Thus, we make the assumption that the different domains do not need any other spatial filters than the pre-trained 3×3 convolution layers. To fulfill the second criterion, we control the capacity of the adapters through parameter shrinkage. More specifically, we choose a residual architecture for the adapters as shown in Figure 1.3. Indeed, thanks to the residual architecture increasing the shrinkage causes the adapters to tend towards the identity layer, thus reducing the model expressiveness. We empirically show that a higher shrinkage of the adapting parameters improves the classification performance on smaller datasets.

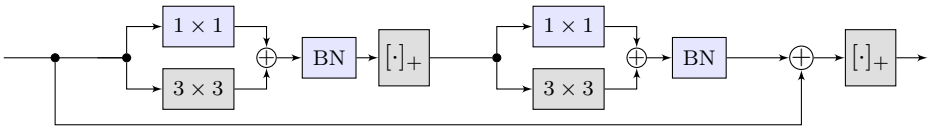
We evaluate our proposed adapters on the Visual Decathlon and show that they lead to similar or better results than independently fine-tuning a pre-trained model on each dataset. Beside parameter efficient transfer learning, our approach also allows to learn new domains sequentially without forgetting the old ones. In fact, adding a new dataset boils down to training a new set of domain-specific adapters without changing the shared filters.

1.2.3 Improving the residual adapters

The main downside of the adapters introduced in Chapter 3 is that they require to be included in the network before the pre-training step on ImageNet. Indeed, adding these adapters to a pre-trained off-the-shelf network leads to significantly lower transfer learning results. We address this issue in Chapter 4 where we propose plug-and-play parallel adapters which can be directly used with any pre-trained network. As shown in Figure 1.3, the parallel design differs from the series adapters of Chapter 3



(a) Series design



(b) Parallel design

Figure 1.3: **Residual adapters designs.** We illustrate how the adapters are integrated into the residual block of a ResNet. The adapters (in blue) are domain-specific while the shared parameters (in gray) are pre-trained on ImageNet and then frozen. The series design (a) is introduced in Chapter 3. The parallel design (b) proposed in Chapter 4 can directly be used with off-the-shelf networks.

by the position of the adapter, in series or in parallel, with respect to the shared 3×3 convolution. This small change increases the convergence speed when training on a target dataset and allows us to use publicly available pre-trained networks. Similarly to the series adapter, we can also control the capacity of the parallel adapter through parameter shrinkage thanks to its residual architecture. Another way to regularise the adapters consists in inserting channel dropout [149] just before the 1×1 convolution in the adapters, thus providing a significant performance gain when using wider networks such as WideResNet [164]. Finally, we empirically show that, contrary to the standard transfer learning practice [162] of freezing the early layers and fine-tuning only the last layers, it is beneficial to train adapters all along the network.

There are as many adapters as the number of domains so there is still some redundancy in the adapter parameters, which can be exploited for further adapter compression. To this end, we propose to stack together the 1×1 convolution filters of the different domains and then perform a truncated SVD on the resulting matrices. In this way, each 1×1 adapter convolution is decomposed into a domain-agnostic encoder followed by a domain-specific decoder. Finally, the joint encoders are frozen and the task-specific decoders are fined-tuned on each domain. On the Visual Decathlon the compressed adapters achieve better overall results than the parallel adapters with twice as less parameters. Interestingly, while they achieve similar performance on the medium-sized datasets, the compressed adapters obtain better results on the small datasets thanks to the cross-domain knowledge of the shared encoders.

1.2.4 Boosting low-supervision with self-supervision pre-training

In Chapters 3 and 4, we propose the residual adapters to harness the filters learned on another dataset. However, some settings such as clustering do not allow to use any external data. An alternative way to initialise the model without using any external data is to leverage the intrinsic knowledge contained in the dataset. Hence, similarly to what is commonly done in natural language processing, we propose the systematic use of self-supervision to pre-train the model. This is a common thread in Chapters 5 to 7 where we show that self-supervision pre-training can boost diverse low-supervision tasks such as semi-supervised classification in Chapter 5, novel class discovery in Chapter 6 and deep clustering in Chapter 7.

In these three chapters we adopt the same process: we first pre-train the network with self-supervision on the whole dataset and then we freeze the early layers of the pre-trained network. This pre-training and partial freezing are beneficial for low-supervision settings in two ways: (1) the self-supervision literature [57, 112] shows that such pre-training provides a better initialisation for fine-tuning on few samples than random initialisation and (2) freezing the early layers constrains the network representation by reducing the number of trainable parameters and prevents the network from exploiting low-level visual cues. In Chapter 5 we empirically show that applying self-supervision pre-training to existing semi-supervised methods consistently boost their performance. This performance increase is particularly significant in the extreme case with very few labels, where the existing semi-supervised methods fail without such pre-training. We make a similar observation in Chapter 7 as our deep clustering method also requires self-supervision pre-training. Finally, we show in Chapter 6 that the impact of self-supervision pre-training on our method for novel class discovery depends on the dataset at hand with, for example, more impact on SVHN than on CIFAR-10.

1.2.5 Semi-supervised image classification with scarce annotations

Another possible type of input data is the semi-supervised classification setting where samples come from a single dataset but with few labels per class. While our proposed domain-specific parametrisations suited the multiple visual datasets case, it is desirable to share the whole network across data for the semi-supervised setting. Hence,

an optimisation approach is more adapted here than an architecture approach. Indeed, standard semi-supervised techniques typically rely on the balance between a loss on the labelled data and another loss on the unlabelled samples. However, when optimising the two losses jointly, there is a risk of overfitting the labelled data as the labelled samples are much rarer than the unlabelled ones. Therefore, the balance between the two losses is particularly fragile in extreme cases with scarse annotations.

Based on this observation, we propose in Chapter 5 a semi-supervised method which is specifically designed to leverage self-supervised pre-training. Previous state-of-the-art methods jointly optimise the losses on the labelled and unlabelled data. This requires a careful choice of hyperparameter to balance these losses. Instead of this, we propose a two-phase training cycle which alternates between training on the labelled and unlabelled data. In the first phase, the classification layer of the network is trained on the labelled set and then we use this classifier to assign pseudo-labels to the unlabelled set. In the second phase, we re-initialise the network with the self-supervision weights and we train the network on the unlabelled set using the pseudo-labels assigned in the first phase. In this way, we disentangle the two losses by learning the classifier on the labelled data and the representation on the unlabelled data. Inspired by cross-validation, we further propose to sample at each training cycle a held-out set among the unlabelled data to avoid overfitting the pseudo-labels assigned in the first phase. Finally, we show in the experiments that our method also performs particularly well in semi-supervised transfer learning compared to existing techniques.

1.2.6 Discovering new classes given some known labelled classes

The setting in the previous subsection supposes that there are at least a few labels per class. In another low-supervision case called transfer clustering, the dataset is composed of two disjoint parts: a group of unknown classes whose samples are all unlabelled and another group of known labelled classes which should be used to improve the clustering performance on the unknown classes. After the architecture parametrisations for the multi-domain setting in Chapter 3-4 and the alternating optimisation for semi-supervised classification in Chapter 5, we propose in Chapter 6 a method which combines both architecture and optimisation designs by respectively using a two-headed network and a clustering loss based on the comparison of top rank statistics between pairs of samples. As this clustering loss is also a key component of Chapter 7, we will discuss it in the next subsection.

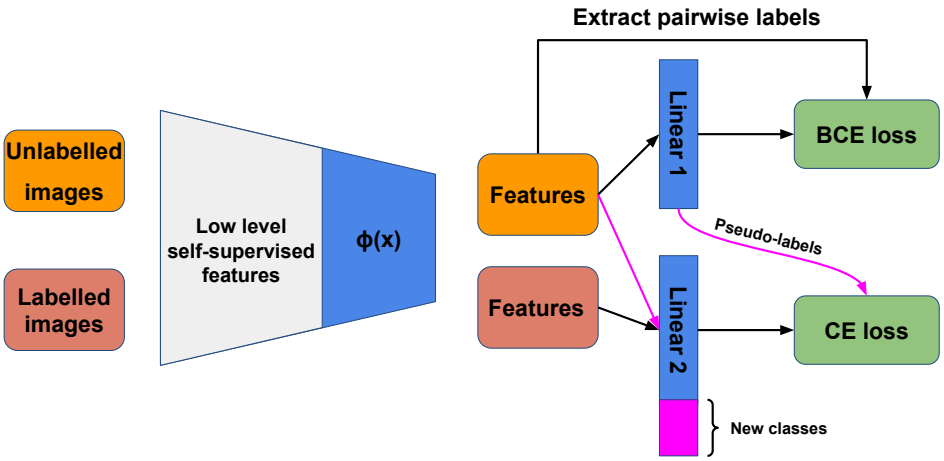


Figure 1.4: **Discovering new classes given some known labelled classes.** Each minibatch contains labelled and unlabelled images. The clustering of the unlabelled samples is learned in the top head by using a binary cross-entropy (BCE) based loss. The classification head is jointly trained on the labelled and unlabelled data with the cross-entropy (CE) loss. As indicated by the curved purple arrow, the classification pseudo-labels of the unlabelled samples are generated on-the-fly by using the cluster assignments from the clustering head .

While existing transfer clustering methods either do not use the unlabelled data for training like MCL [69] or use the labelled data just for the network initialisation like DTC [62], we propose to jointly train the network on the unlabelled and labelled data. In our method, the joint representation of the two-headed network is optimised to (1) cluster the unlabelled data in the first head and (2) perform image classification on both the unlabelled and labelled data in the second head. The pipeline of our method can be seen in Figure 1.4.

Another contribution is how we integrate the two-headed network into our algorithm. Indeed, our goal is to produce a positive interaction between the clustering and classification heads. We do so by using the class assignments from the clustering head as pseudo-labels for the the unlabelled samples in the classification head. In this way the classification head is trained jointly on the unknown and known classes, thus improving the feature representation. As the clustering pseudo-labels are extracted at the joint representation level, the improved representation might be beneficial for clustering. The other way around, a better clustering results in better pseudo-labels for the unlabelled samples in the classification head. We empirically show in an ablation study that this design improves the clustering of the unlabelled samples. Furthermore, it allows us to incrementally learn new classes without forgetting the

known classes contrary to methods like DTC which do not preserve the representation learned on the known classes.

1.2.7 Deep clustering by pairwise cluster predictions

Chapters 6 and 7 both rely on clustering as the deep clustering setting of Chapter 7 is the extreme case of Chapter 6, when there is no known labelled classes. We first propose in Chapter 6 a clustering loss based on pairwise cluster predictions, which we discuss and analyze more deeply in Chapter 7.

Our proposed clustering loss is a binary cross-entropy loss on the prediction whether or not a pair of samples belong to the same cluster. The pseudo-label for this loss is the minibatch adjacency matrix which is generated based on the comparisons between pairs of samples. Contrary to existing pseudo-labeling methods such as DEC [155] or DAC [21], we choose to compare pairs of samples at the data representation level rather than at the cluster prediction level. In Chapter 7 we empirically show that the choice of space where the pairwise pseudo-labels are extracted has a significant impact on the clustering performance.

Another key component of our method is how the pairwise pseudo-labels are generated. We propose to establish a connection between a pair of samples if their pairwise similarity is above a certain threshold. Hence, the choice of similarity is critical for good performance. Inspired by the observation made in representation analysis papers [96] that channels in the feature space encode for concepts, we devise in Chapter 6 a new similarity based on the feature channels learned on the labelled data. Our similarity compares which are the most activated channels for a pair of unlabelled samples. If both considered samples share the same top rank statistics, then the pseudo-label for this pair is positive. Thus, another role of the classification head in Chapter 6 is to transfer knowledge from the labelled data to the clustering head by learning a feature representation whose channels are used to compare unlabelled samples. Moreover, we compare in Chapter 7 different similarities such as cosine or symmetric SNE and we also introduce a similarity based on k-nearest neighbours.

Finally, as our clustering loss is based on a standard classification loss, we can combine it with efficient data composition techniques such as RICAP [145] or Mixup [167], which feed composite images to the network and use linear combinations of labels as targets. We show that such augmentation methods are essential to get good performance with our method, thus confirming the paramount importance of data augmentation in the low-supervision setting.

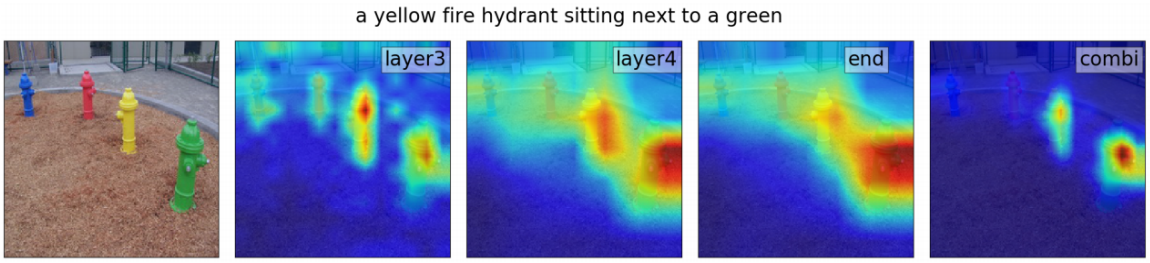


Figure 1.5: **Image captioning saliency.** We show saliency maps produced by our NormGrad method at different layers of a ResNet101. In the last image, combining saliency maps allows to highlight the most important parts of the image.

1.2.8 A framework for backpropagation saliency methods

Hitherto, we have seen how different types of input data such as multiple visual domains or low-supervision settings influence the design of representation learning techniques. From another perspective, we also consider saliency, that is, the problem of examining which parts of the input samples contribute the most to the data representation.

Staying consistent with the previous chapters, we approach saliency from a representation learning angle. In Chapter 8 we propose a unifying framework for backpropagation saliency methods by reframing saliency as the contribution of the spatial locations to the gradient of the model weights. Our framework exploits the fact that the gradient of spatially shared weights is the result of a summation over spatial locations. Therefore, the resulting saliency maps quantify the contribution of each pixel to the change in model weights. We extend this framework to second order gradients by introducing meta-saliency, namely adding an inner optimisation step within the loss used to produce the saliency map. In this way meta-saliency measures which pixels contribute the most to this inner training step.

This framework can be used to derive existing saliency methods such as gradient backpropagation [133] or linear approximation [110] but also to create new methods such as NormGrad which we introduce in Chapter 8. Finally, as the saliency methods derived from our framework can be applied at any layer of the model, we propose to combine saliency maps across layers as illustrated in Figure 1.5 to obtain a trade-off between the high resolution of saliency maps at early layers and the class discriminability at late layers. We show that the combination of saliency maps across layers results in a performance boost for quantitative benchmarks such as the Pointing Game [168].

1.3 Publications

Each publication is presented in an individual chapter as listed below (* indicates that authors contributed equally):

Chapter 3: Learning multiple visual domains with residual adapters.
Sylvestre-Alvise Rebuffi, Hakan Bilen and Andrea Vedaldi. *In Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

Chapter 4: Efficient parametrization of multi-domain deep neural networks.
Sylvestre-Alvise Rebuffi, Hakan Bilen and Andrea Vedaldi. *In Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Chapter 5: Semi-supervised learning with scarce annotations.
Sylvestre-Alvise Rebuffi*, Sebastien Ehrhardt*, Kai Han*, Andrea Vedaldi and Andrew Zisserman. *In Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020.

Chapter 6: Automatically Discovering and Learning New Visual Categories with Ranking Statistics.
Kai Han*, Sylvestre-Alvise Rebuffi*, Sebastien Ehrhardt*, Andrea Vedaldi and Andrew Zisserman. *In International Conference on Learning Representations (ICLR)*, 2020.

Chapter 7: LSD-C: Linearly Separable Deep Clusters.
Sylvestre-Alvise Rebuffi*, Sebastien Ehrhardt*, Kai Han*, Andrea Vedaldi and Andrew Zisserman. *In submission*, 2020.

Chapter 8: There and Back Again: Revisiting Backpropagation Saliency Methods.
Sylvestre-Alvise Rebuffi*, Ruth Fong*, Xu Ji* and Andrea Vedaldi. *In Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Chapter 2

Literature Review

Our contributions described in Chapters 3 to 8 can be linked to different subfields of computer vision. First, related to the residual adapters of Chapters 3-4 and the incremental learning component of our method in Chapter 6, we point out the different settings where tasks are trained in parallel or in sequence. Then we delve into the literature corresponding to the low-supervision settings covered from Chapter 5 to 7. Finally, we refer to saliency papers which are related to the framework we propose in Chapter 8.

2.1 Learning tasks sequentially or in parallel

Chapters 3 and 4 respectively introduce and improve residual adapters for multi-domain learning. The considered setup intersects with other subfields of computer vision like multi-task learning, life-long learning or domain adaption. In this section we will stress the differences and similarities between these settings that learn tasks in parallel or in sequence.

2.1.1 Learning tasks in parallel: multi-task learning

The goal of multi-task learning (MTL) is to propose models which learn different related tasks while sharing parameters and computation among them. MTL approaches resulted in performance gains on various problems such as landmark detection [172], object detection [14] or tracking [171]. Beyond computer vision, MTL has also been successfully applied to a wide variety of fields such as speech recognition [35], natural language processing [27] or drug discovery [117].

MTL approaches are based on diverse assumptions. A first line of research with non neural models assumes close relationships between the tasks. Based on this

assumption, Argyriou et al. [5] enforce a group-sparsity constraint through a mixed norm such that only a few features are used across all tasks. The assumption that all tasks are closely related can be relaxed by regrouping tasks into clusters through an implicit [43] or explicit [74] clustering constraint.

One of the first neural network approach in this area is the seminal paper by Caruana [20] that proposes a neural network architecture whose early layers are shared among tasks. This work empirically shows that parameter sharing acts as a regulariser and leads to better overall performance. Recently, more elaborate architectures [39, 118, 81] have been designed for MTL in computer vision. For example, we refer to UberNet [81] which trains five different tasks on Pascal VOC [42] thanks to its architecture design with skip and fusion layers as well as task-specific deviations. This idea of combining the ouputs of different stages of the network can also be found in HyperFace [118] and in the work of Doersch and Zisserman [39]. To avoid explicit network architecture search, Yang et al. [160] propose to learn how to share the network filters among tasks in an end-to-end manner by using a tensor factorization technique.

Kokkinos [81] experimentally shows that parameter sharing can also result in negative interference between tasks. Instead of designing architectures to address this problem, another line of work focuses on loss balancing to mitigate these negative tasks interference. For example, Kendall et al. [78] weight the segmentation, instance segmentation and depth prediction losses by including the task uncertainties in the likelihood. Similarly, GradNorm [24] achieves better performance on the same tasks by choosing the loss weights such that the norms of the gradients are balanced among tasks.

Finally, we stress the difference with multi-domain learning by pointing out that the tasks in standard MTL belong to a same input domain whereas multi-domain learning deals simultaneously with different visual domains like natural images, textures [25] or even more abstract visual inputs like the dynamic images obtained using the method described in [13].

2.1.2 Life-long learning

Life-long learning differs from standard MTL as the tasks are learned sequentially rather than in parallel. This sequential training leads to *catastrophic forgetting*, a problem identified in the connectionist networks community in the late 1980s by McCloskey et al. [101]. Catastrophic forgetting happens when a neural network is

trained on new incoming data which causes the network to overwrite the representation learned on previous data. This phenomenon was deeply studied in the 1990s neural network literature [4, 49] and a summary of these approaches can be found in Thrun's review [148]. With the advent of deep-learning this topic regained interest and the newly proposed methods can be grouped in two branches that already existed in the connectionist work.

One line of work proposes to first freeze the network trained on the old tasks and then to grow the network to train it on the new incoming tasks without modifying any prior knowledge. For example, in reinforcement learning, Rusu et al. [123] propose to horizontally grow the network by adding a new column of trainable layers for each new incoming task. More efficiently, Xiao et al. [154] devise a tree based model that can be incrementally increased when new classes arrive. Our residual adapters in Chapters 3-4 are related to this type of approach which avoids the catastrophic forgetting phenomenon by construction. One of the downside of these methods is that they require increasing resources as the number of tasks increase. We take this problem into account as our adapter parameters only represent a small fraction of the overall network parameters.

Besides this freezing and growing strategy, another way to avoid catastrophic forgetting consists in constraining representation learning on the new tasks with the knowledge from the previous tasks. EWC [80] proposes a regularisation term based on a weighted distance to the parameters learned on the previous tasks. By using the elements of the Fisher information matrix as weights, the authors avoid over-constraining parameters which matter less for the previous tasks. Instead of acting directly on the network parameters, Li and Hoiem [89] preserve the output of the previously trained network by distillation of the old model.

Class incremental learning is a subfield of life-long learning, where new classes arrive over time and are added to the same classification task. Furthermore, the setup described in iCaRL [120] allows a small memory of annotated samples from the previous classes. The methods proposed for this setting differ from one another in the way they use this small memory. For example, iCaRL proposes to use a nearest class mean classifier to prevent the classifier from being biased towards the new classes. Alternatively, GEM [92] projects the gradients of the new samples such that they do not increase the loss on the stored samples. In Chapter 6, our class incremental learning setup for novel class discovery differs from this setting as we allow the full storage of the old classes but we do not have access to the labels of the new classes.

2.1.3 Domain adaptation

When our residual adapters in Chapters 3-4 are trained sequentially, our work can be related to domain adaptation (DA). DA, introduced in [34], aims at transferring a model trained on a labelled source domain to an unlabelled target domain. This definition corresponds to the most common setting but some approaches also study the semi-supervised case with some labels available in the target domain. Csurka [29] did a review of the different DA settings and methods in computer vision.

There are two popular approaches in deep DA: (1) minimising a divergence criterion between the statistics of the source and target distributions and (2) using adversarial training with a domain confusion loss. The common objective of these two lines of work is to align the domain distributions, thus leading to a domain invariant representation.

Tzeng et al. [150] propose a regularisation term based on Maximum Mean Discrepancy (MMD), which penalises the difference between the means of the distributions in the feature space. Closer in spirit to our residual adapters, Long et al. [91] add a residual function to the classifier to account for changes in the target domain. To strengthen the distribution alignment constraint, Sun et al. [141] add second order statistics to the MMD regularisation term and DeepJDOT [12] chooses instead to match the source and target distributions by optimal transport with the Wasserstein metric.

DANN by Ganin et al. [51, 52] goes beyond matching distributions using metrics: this method adversarially trains a domain confusion loss by means of a gradient reversal layer to enforce that the source and target domains are indistinguishable. Several articles [16, 65] then extend this idea with generative adversarial networks to generate images from the source domain in the target domain.

DA differs from the multi-domain setting of Chapters 3-4 in two major aspects. First, in multi-domain learning the classes at hand can be different from one domain to another whereas in the DA setting the classes stay the same across domains. Second, the goal of DA is to maximise the performance in the target domain without explicitly trying to preserve the performance in the source domain while our Visual Decathlon metric in Chapter 3 is based on the classification performance on all of the domains.

2.1.4 Multi-domain learning

Compared to the previous settings, multi-domain learning can be done either in sequence or in parallel. Its goal is to learn a model that can achieve competitive

classification performance on diverse visual domains with a high degree of parameter sharing across domains.

In the first work on the topic Bilen et al. [15] take the differences between visual domains into account by using domain-specific batch and instance normalisation layers. With this approach, the model lacks capacity on some datasets so we introduce the residual adapters which allow to control the degree of adaptation based on the size of the target dataset. Similar adapters were concurrently proposed by Rosenfeld and Tsotsos in [121] but with a different non residual architecture. Another line of work focuses on reducing the memory usage of the adaptation parameters rather than the overall number of parameters. To this end, Mancini et al. [99] and Mallya et al. [98] modulate the network output by training binary masks on the network weights, thus achieving good adaptation performance on the Visual Decathlon with a small memory footprint.

More recently, adapters have been transposed to other subfields of computer vision. For example, for counting elements in different visual domains with papers from Lu et al. [94] and Marsden et al. [100]. They have also been used in natural language processing for parameter efficient transfer learning in the work of Houlsby et al. [66] and Artetxe et al. [6].

2.2 Self-supervised representation learning

Chapters 5 to 7 span diverse low-supervision settings such as semi-supervised classification, novel class discovery and deep clustering. In such cases, we propose the systematic use of self-supervision pre-training as we empirically show that it leads to a significant performance boost in various low-supervision settings. Hence, in this section we present the motivations behind self-supervision as well as the main approaches in the literature.

2.2.1 Motivation

Self-supervised learning is a form of unsupervised learning, where the supervisory signal is directly extracted from the data. In practice, self-supervision methods create and solve a pretext task whose labels can be extracted from the data. The intuition behind self-supervision is that solving the pretext task will require the network to learn a semantic representation which can be exploited for pre-training or regularising the model for a downstream task like classification or object detection. Self-supervised learning is motivated by the fact that annotating images is a tedious

and costly process while vast collections of unlabelled images are available. Until recently, self-supervision pre-training was not as commonly used in computer vision as it is in natural language processing like in GPT [116] or BERT [36]. A possible explanation is the success of ImageNet pre-training which is either used as a feature extractor or for fine-tuning on a target dataset. However, since 2018 there has been an increased interest about self-supervision in computer vision, which caused self-supervision representations to progressively close the gap with ImageNet pre-training and even surpass it on some datasets like in the BYOL paper [60].

2.2.2 Self-supervision methods

One of the first work about self-supervised convolutional neural networks is Exemplar-CNN [41]. This method proposes to create surrogate classes containing the augmented versions of a same image, called exemplars, obtained by applying diverse random transformations. In this case, the pretext task is to classify these exemplars and assign them to the correct surrogate class. Instead of using data augmentation to create a class from an image, RotNet [57] takes the opposite approach of considering the possible data augmentations as the target classes. As shown in Figure 2.1, RotNet [57] trains the network on the pretext task of predicting the rotation applied to the input image among four possible rotations: 0, 90, 180 and 270 degrees. In practice, RotNet is fast to train and easy to implement as it solves a 4-class classification problem. Furthermore, this method provides robust low-level filters as the authors show that fine-tuning only the last part of the network almost reaches the classification performance obtained by training the whole network with full supervision. For these reasons, we systematically choose RotNet when we use self-supervision pre-training in our work.

Another family of pretext tasks consists in predicting the relationships between patches of an image. This idea is first introduced in Doersch et al. [38] where a network is trained to predict the relative position between two random patches. A downside of this approach is that it requires strong data augmentations to avoid learning visual shortcut signals such as texture continuation or chromatic aberration. Noroozi and Favaro [109] extend this idea by solving a jigsaw puzzle with 9 patches: the network is trained to put the shuffled patches in their original order.

Before the work on CNNs, one of the first approaches on self-supervision for deep networks is the denoising autoencoder (dAE) of Vincent et al. [151], which learns to reconstruct the original image from a noisy version of it. Instead of corrupting the input data with noise, other publications propose to withhold some parts of the input

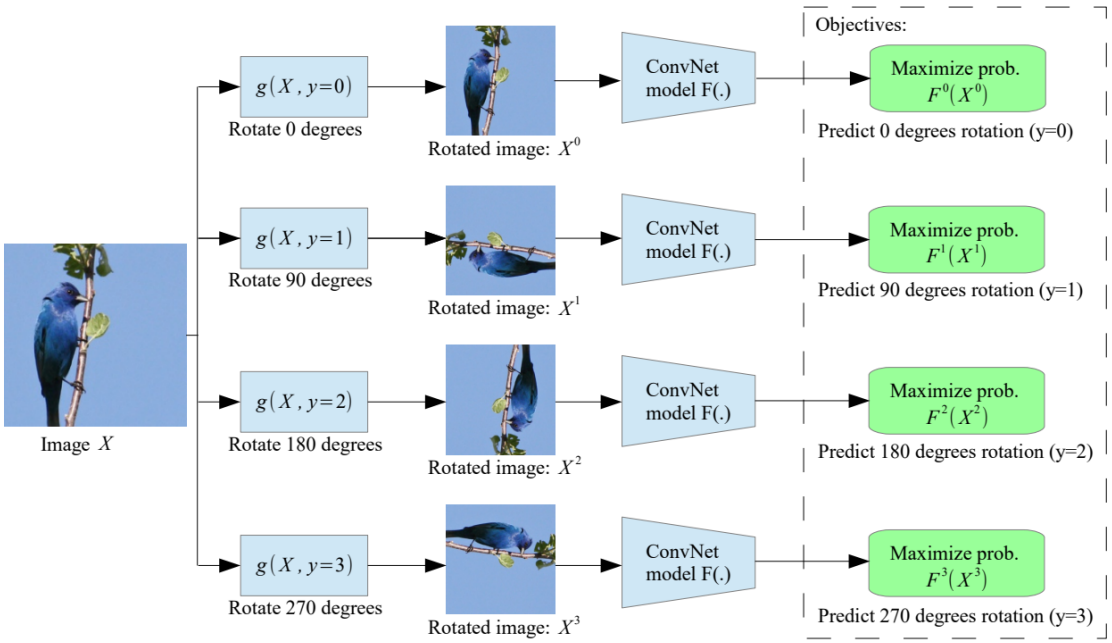


Figure 2.1: **RotNet** [57] solves the pretext task of predicting which rotation was applied to the input image among four possibilities: 0, 90, 180 and 270 degrees. To solve this task, the network has to learn semantic features from the dataset. This image is taken from Gidaris et al. [57].

signal and train the network to recover the missing content. For example, Pathak et al. [113] propose inpainting as pretext task and Zhang et al. [170] learn to predict a subset of hidden colour channels. In a similar spirit, several papers [169, 86] convert the input images to a grayscale representation and task the network to colour them using a classifier over colour bins. It should be noted that this colourisation technique requires loss balancing as some colour bins are more frequent than others. Finally, rather than manually perturbing the input images, an alternative solution would be to use adversarial training. Generative adversarial networks (GANs) [59] deeply improved image generation but they map latent variables to images whereas self-supervision requires the inverse mapping. Bidirectional GANs [40] adapt GANs to feature learning by considering the joint space of image and feature in the generation-discrimination mechanism.

More recently, compelling results have been obtained by using contrastive losses based on Noise Contrastive Estimation (NCE) [61]. The associated pretext task consists in identifying the target representation among a set of negative samples. For example, Contrastive Predictive Coding [112] considers a target patch against other negative patches coming from the same image or from other images of the minibatch.

Other variants of this approach, for example MoCo [63] or SimCLR [23], also optimise the same InfoNCE contrastive loss but they discriminate between representations of images rather than patches. While the core idea of MoCo and SimCLR is similar, it is interesting to note some of their practical specificities. For instance, MoCo compares the target representation with representations coming from the exponential moving average of the network. Also, SimCLR replaces the dot product between representations by a cosine similarity for the input of the contrastive loss.

MoCo and SimCLR learn to match augmented versions of a same image while the other images of the minibatch act as negative samples. Instead of considering images individually, it is possible to group them in clusters and use clustering as a pretext task to learn an unsupervised representation. This idea is studied in DeepCluster [19] and is then refined in the work of Asano et al. [7], which constrains the clusters to be non-empty by performing the cluster assignment step with optimal transport. Rather than assigning an image to a single cluster, Gidaris et al. [56] go even further by extracting a visual bag of words for each image and tasking the network to predict this bag of words.

Finally, some of these self-supervision methods are complementary. In fact, Doersch et al. [39] successfully combine pretext tasks such as relative position, colourisation and exemplar networks.

2.2.3 Using self-supervision for pre-training or regularisation

The most common use of self-supervision is to pre-train the network before fine-tuning it on a target task. Indeed, most of the papers mentioned in the previous subsection evaluate their methods on how their ImageNet self-supervised representations transfer to Pascal VOC classification, detection and segmentation [42].

Second, the pretext task and the target task can be learned simultaneously, in which case the self-supervision loss acts as a regulariser for the loss of the target task. This approach has contributed to improve performance in low-supervision settings such as few-shot learning with the work of Gidaris et al. [55], semi-supervised learning with S4L [166], or domain adaptation [142].

2.3 Semi-supervised image classification

In the standard definition of semi-supervised learning from the reference book by Chapelle et al. [22], the training dataset is composed of a small amount of labelled

data and a large amount of unlabelled data. Semi-supervised learning is related to many fields like speech [163], text [161, 102] or bioinformatics [130] but also to several subfields of computer vision such as tracking [147], domain adaptation [174] or active learning [132]. Here, we restrict the scope of this review to semi-supervised learning for image classification, which corresponds to the setting of Chapter 5. In most of the semi-supervised algorithms, the overall loss L_{tot} can be written as the sum of a cross-entropy loss on the labelled data L_{lab} and another loss on the unlabelled samples L_{unlab} such that $L_{\text{tot}} = L_{\text{lab}} + \lambda L_{\text{unlab}}$ with λ being a hyperparameter which balances the two losses. Depending on the chosen loss L_{unlab}, semi-supervised learning methods can be divided in two main categories, which we present in the next subsections. The first line of work, called self-labeling, assigns pseudo-labels to the unlabelled samples before using them in the cross-entropy loss L_{unlab}. The other main type of approach, called consistency methods, proposes unsupervised losses for L_{unlab}, which encourage related input images to produce similar output predictions. In our work we use self-labeling and a consistency loss in Chapter 5 for semi-supervised learning but we also add a consistency regularisation in our methods for novel class discovery in Chapter 6 and deep clustering in Chapter 7.

2.3.1 Self-labeling methods

Self-labeling is probably the earliest approach in semi-supervised learning and dates back to the 1960s with the work of Scudder [127], Fralick [48] and Agrawal [2]. Self-labeling methods assign pseudo-labels to the unlabelled samples based on the model predictions. This allows the model to be trained for classification on both the unlabelled and labelled data. These methods usually have a two-phase training: first the model is trained only on the labelled data, then the training set is progressively extended to the unlabelled samples.

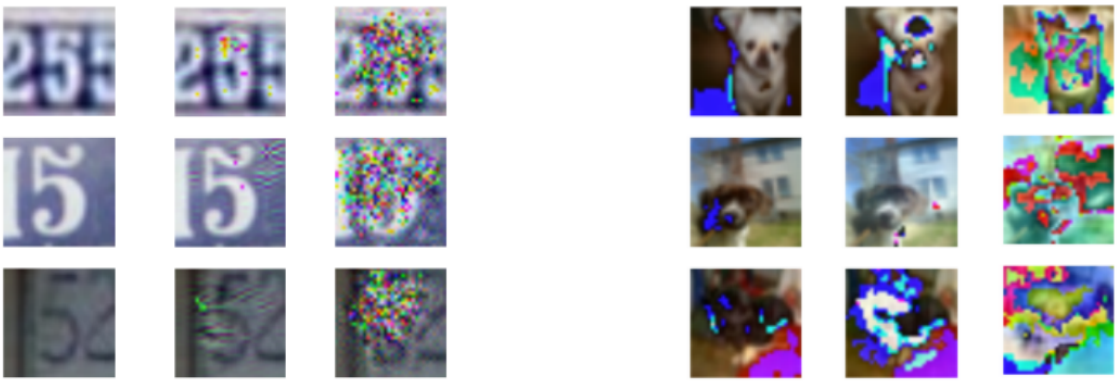
With the advent of deep learning, this idea reappeared with the work of Lee [88]. This method extracts one-hot pseudo-labels by taking the class with the highest prediction score. Furthermore, the two-phase training is implemented by using a ramp-up function for the coefficient which balances the losses on the labelled and unlabelled samples. The ramp-up function is small in the beginning such that the training occurs mainly on the labelled set. Then, the ramp-up coefficient linearly increases up to its final value. In this way, the loss on the unlabelled samples is progressively added to the total loss to avoid erasing the representation learned on the labelled data. Instead of extracting one-hot pseudo-labels, MixMatch [11] generates soft targets on the unlabelled samples. More precisely, for each unlabelled sample this

method computes the average prediction over several augmented views of the sample before sharpening this average prediction to obtain the soft target. This approach is inspired by the distillation idea [64] that soft targets are more informative for training than one-hot labels. FixMatch [135] proposes an alternative to the linear ramp-up function by optimising the cross-entropy loss on an unlabelled sample only if the network prediction score for this sample is above a fixed threshold. Hence, this filtering approach causes the unlabelled samples to be progressively added to the training.

The previous loss-balancing methods require a careful hyperparameter search to avoid overfitting the labelled set. Indeed, the learning curriculum depends on the choice of the confidence threshold for FixMatch and on the ramp-up function for the other methods. This motivates the design of our alternating optimisation method in Chapter 5, which does not require such hyperparameters. In a similar spirit, Iscen et al. [72] also propose an algorithm which alternates between two training phases: one on the labelled data and the other one on the unlabelled samples with assigned pseudo-labels. Compared to our method, they assign pseudo-labels to the unlabelled samples by label propagation on a nearest neighbour graph rather than directly extracting the labels from the network predictions. Finally, Jackson and Schulman [73] propose an alternative method for pseudo-label generation by reformulating it as the optimisation problem of aligning gradients on the labelled and unlabelled sets.

2.3.2 Consistency methods

Self-labeling methods aim at solving the classification problem on the labelled set and, by means of pseudo-labels, on the unlabelled data. Consistency methods take a different approach as they regularise the classification loss on the labelled data by encouraging the network outputs to be "consistent" on the unlabelled samples. The term "consistent" means that two related images should produce similar outputs when being fed to the same network or two related networks. In the first instances of this approach, Rasmus et al. [119] and Sajjadi et al. [124] propose as consistency loss the squared L_2 distance between the network outputs of perturbed copies of the same image. Instead of using the standard augmentation techniques to obtain the augmented views of the sample, VAT [103] applies adversarial perturbations which maximise the Kullback–Leibler divergence with the current model prediction. Examples of images produced by VAT [103] on CIFAR-10 [82] and SVHN [105] can be seen in Figure 2.2.



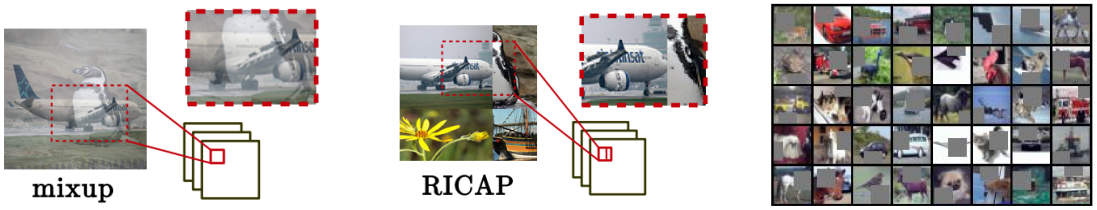
(a) SVHN (b) CIFAR-10

Figure 2.2: **Virtual adversarial training** [103] perturbs the input image to maximise the change in the model prediction. The consistency loss is then applied to the network outputs of the original image and its perturbed counterpart. These images are taken from Miyato et al. [103].

Consistency can also be applied to the outputs of related networks. In their Temporal Ensembling model Laine and Aila [84] encourage the consistency between current and past predictions of the network by optimising the squared distance between the output of the network and the average of the outputs at earlier epochs. In a variant of this approach, Mean Teacher [146] replaces the average of the outputs at earlier epochs by the output of an exponential moving average of the network over training iterations.

2.3.3 Importance of data augmentation in low-supervision settings

In Chapter 7, we empirically show that combining our proposed clustering loss with data composition techniques such as Mixup [167] or RICAP [145] provides a major boost in clustering performance. This observation concurs with the growing predominance of data augmentation in methods for low-supervision settings. The boosts obtained in such settings are typically much more substantial than in fully supervised problems. For example, MixMatch [11], a self-labeling method for semi-supervised learning, decreases its classification error on CIFAR-10 [82] from 39% to 12% when using Mixup [167] during training. In a follow-up method called ReMixMatch [10] the authors propose a strong augmentation technique termed CTAugment to further improve the results of their original MixMatch method. Similarly, UDA [156] improves the performance of consistency based methods by using RandAugment [31] to generate the augmented views of the images. Finally, FixMatch [135], the current



(a) Mixup [167] (b) RICAP [145] (c) Cutout [37]

Figure 2.3: **Illustration of data composition techniques.** (a) Mixup linearly interpolates between two images, (b) RICAP assembles random crops of four images and (c) Cutout randomly inserts grey squared patches. The images (a) and (b) are taken from [145] while (c) is taken from [37].

state-of-the-art method for semi-supervised classification, is very similar to the first self-labeling method in deep learning proposed by Lee [88] in 2013. As described above, one difference consists in replacing the loss-balancing ramp-up function by a filtering mechanism based on the prediction score. However, the most influential difference is the use of the strong augmentation techniques CTAugment [10] and RandAugment [31] when training on the pseudo-labels. Hence, competitive performance in semi-supervised learning highly depends on the use of strong data augmentation and how this augmentation is integrated into the algorithm. We describe below the augmentation techniques commonly used to improve the performance of low-supervision methods.

A first type of data augmentation artificially increases the number of training images by combining images. For example, Mixup [167] feeds convex combinations of pairs of images and their labels to the network. Instead of doing a linear interpolation between two images, RICAP [145] patches together random crops of four images. The label of the resulting image is the linear combination of the labels of the four images, weighted by the areas of their crops. Related to these techniques, Cutout [37] randomly applies grey squared patches to the training images. Its goal is to force the network to use the whole image during training rather than focusing on a part of the image. Examples of images augmented using these three techniques can be seen in Figure 2.3.

Another line of work is based on combining the image transformations available in the Python Imaging Library (PIL) which includes, among others, rotations, colour distortions or contrast changes. One of the reference method is AutoAugment [30] which uses reinforcement learning to learn how to compose these transformations. Note that this approach requires many labelled samples to provide the feedback needed for reinforcement learning. RandAugment [31], which randomly samples the transformation

magnitudes and thus does not require any labelled data, is more adapted to the semi-supervised learning setting. CTAugment [10], which is specifically developed for semi-supervised learning, is the middle point between the two previous approaches as it sets the magnitude ranges for the various transformations such that the model predictions on the few labelled samples are close to the true labels.

2.4 Deep clustering

In the previous section on semi-supervised learning, each class is partially labelled. In Chapter 6 we consider another low-supervision setting where a few classes are fully labelled and unlabelled data from new unknown classes is added to the problem. The goal in this setting is, given a group of known labelled classes, to cluster the unlabelled samples into groups corresponding to the new unknown classes. This is a special case of deep clustering with prior knowledge. In Chapter 7, we consider the general case of deep clustering where there are no labelled samples. In this section, we first describe the approaches for the general case of deep clustering followed by a review on the special case with prior known classes.

2.4.1 Clustering on extracted deep features

A first type of deep clustering approach applies classic clustering methods on top of features extracted by a deep network. These approaches, which cluster frozen deep features, usually correspond to the baselines in deep clustering papers. To obtain the deep embeddings, an autoencoder is first trained with a reconstruction loss and then the deep features are extracted at the end of the encoder.

The most commonly used clustering method on top of frozen deep embeddings is K-means [95] which alternates between assigning each sample to the closest centroid and updating each centroid with the mean of its assigned samples. Instead of K-means, Jiang et al. [76] compare their method VaDE with the combination of a Gaussian Mixture Model (GMM) [114] and a variational autoencoder. Moreover, Foget et al. [44] use deep embeddings with Affinity Propagation [50] or Hierarchical-DBSCAN [17] as baselines. Finally, Steinbach et al. [140] show that GMMs and K-means do not perform well on highly non-convex cluster structures and in high dimensional spaces. To address these problems, Nie et al. [107] and Yang et al. [159] combine embeddings with Spectral Clustering [106, 153, 129], which performs a low-dimension projection in the truncated eigenspace of the affinity matrix before applying K-means.

There are also some recent methods which are developed for extracted deep features. For example, FINCH [125] proposes to iteratively merge clusters by using the first nearest neighbour graph. However, most of the recent approaches jointly learn clustering and the network representation.

2.4.2 Jointly clustering and deep representation learning

The seminal method DEC [155] has set the trend for methods which combine clustering and deep representation learning. As there is no labelled data in the deep clustering setting, the network initialisation plays a major role in avoiding trivial clusters. Hence, DEC carefully initialises both the network by pre-training a stacked denoising autoencoder [152] and the classification layer with K-means centroids. Next, DEC fine-tunes the encoder by optimising the Kullback-Leibler divergence between the cluster prediction and a sharper version of it. We note that this fine-tuning step can be seen as a self-labeling method like the ones we reviewed for semi-supervised learning. Contrary to the two-step approach of DEC, methods such as DEPICT [54], DCN [157] and VaDE [76] propose to jointly train the clustering loss and the autoencoder reconstruction loss. The goal of this multi-task approach is to regularise the clustering loss with the reconstruction loss.

A key aspect in deep clustering methods is the design of the clustering loss. Indeed, the proposed loss should not only regroup samples into clusters but it also needs to be compatible with gradient backpropagation. Moreover, while DEC combines the Kullback-Leibler divergence with soft centroid assignments following a Student's t-distribution, other self-labeling methods choose to keep the standard image classification design with a cross-entropy loss and a linear classifier followed by a softmax layer. In such a case, these methods differ from one another in the way self-labeling is performed. For example, DEPICT uses a sharpened version of the prediction as pseudo-label whereas DeepCluster [19] alternates between a K-means clustering phase to assign pseudo-labels and a training phase using these assigned labels. The methods of Asano et al. [7], Genevay et al. [53] and Huang et al. [71] extend this two-phase alternating approach by using optimal transport to enforce uniform cluster sizes.

Also based on self-labeling, DAC [21] however differs from the previous methods as it performs clustering by comparing pairs of images rather than treating each sample independently. Instead of doing self-labeling on the network predictions, DAC does it on the cosine similarity between the predictions of the pairs of samples. Instead of comparing pairs of different samples, IMSAT [70] and IIC [75] perform clustering by maximising the mutual information between two augmented copies of the same image.

Cover et al. [28] underline that the mutual information can be written as the difference between the marginal entropy and the conditional entropy. Thus, increasing the mutual information corresponds to encouraging uniform cluster sizes and confident cluster assignments to respectively increase the marginal entropy and decrease the conditional entropy.

Like many of the deep clustering approaches, our proposed method relies on the principle of self-labeling. Some methods such as DEC or DEPICT treat samples independently while others consider pairs of samples like DAC, triplets like JULE [158] or augmented pairs of a same image such as IIC and IMSAT. Our approach is related to DAC as we also consider pairs of samples but with a different clustering loss.

2.4.3 Discovering new classes given some known labelled classes

Deep image clustering is an ill-posed problem. Indeed, the clustering losses defined above are less constraining than a fully supervised loss or a semi-supervised loss, so there can be many valid partitions of the dataset. One way to constrain the problem is to use prior knowledge like a set of labelled classes (see Chapter 6) to guide clustering by having some examples of what constitutes a class.

Throughout the rest of this subsection we consider the setting where the dataset is split into two distinct groups, one composed of labelled samples from known classes and the other one consisting of unlabelled samples from unknown classes. The goal in this setup called transfer clustering is, given the few labelled classes, to cluster the unlabelled data into new classes. A related but different setting is Open Set Recognition described by Scheirer et al. [126], which aims at separating the known classes from the unknown classes. The work of Fontanel et al. [47] mixes both transfer clustering and Open Set Recognition as it proposes to learn to separate known from unknown classes before clustering the data identified as coming from unknown classes.

In a first approach for transfer clustering, Hsu et al. [67, 68, 69] propose to learn on the labelled classes how to regroup pairs of samples into clusters. First, the authors use the ground truth labels to generate binary pairwise labels: a pair of samples has a positive label if both samples come from the same class. Then, using these binary pairwise labels, the model is trained to predict if pairs of samples belong to the same class. Finally, this pair prediction network is applied to the dataset split containing the unknown classes and the clusters are extracted from the network latent prediction space. A downside of this first approach is that the model is trained only on the labelled classes and does not use the unknown classes at training time. In

another line of work, Han et al. [62] modify the deep clustering method DEC [155] to adapt it to the transfer clustering setting. Their first main change is to initialise the network by pre-training it on the labelled classes instead of using an autoencoder. After initialising the centroid classifier with K-means on the unlabelled classes, the authors apply DEC on the unlabelled data and propose to update the pseudo-labels only after each epoch instead of generating them on-the-fly. The motivation behind this change compared to DEC is to avoid forgetting the knowledge learned on the labelled classes in the initialisation step.

We note that, contrary to our work, both of these approaches do not jointly use the labelled and unlabelled classes at training time. Furthermore, we introduce further goals for this setting such as not forgetting the known classes when discovering the new classes, or also integrating both groups of classes into a single classifier.

2.5 Saliency methods

The goal of saliency methods is to produce importance maps which highlight the parts of the input image contributing the most to the network representation at a target layer. In Chapter 8 we propose a unifying framework for backpropagation saliency methods based on the spatial contributions to the gradient of the weights. In this section we also present the other types of saliency methods to stress the differences between the existing approaches. Visualisations for some of the methods presented in this section can be seen in Figure 2.4. Finally, in the last subsection we present qualitative and quantitative evaluations for saliency methods.

2.5.1 Backpropagation based saliency methods

Backpropagation saliency methods regroup the approaches that rely on the backpropagation process that trains the network. These methods go through the following steps. In the forward pass, the input image is fed to the network and the activation map at the target layer is stored. Then, at the end of the network, the gradient of the loss (usually the logit of the target class or the cross-entropy loss) is backpropagated through the network and the gradient of the activation at the target layer is extracted. Finally, the saliency map is obtained by combining the previously stored activation and gradient maps.

One of the early work on the topic is the gradient method of Simonyan et al. [133], which visualises the gradient of the target logit with respect to the pixels of the input image. They obtain a saliency map from this gradient by taking the maximum

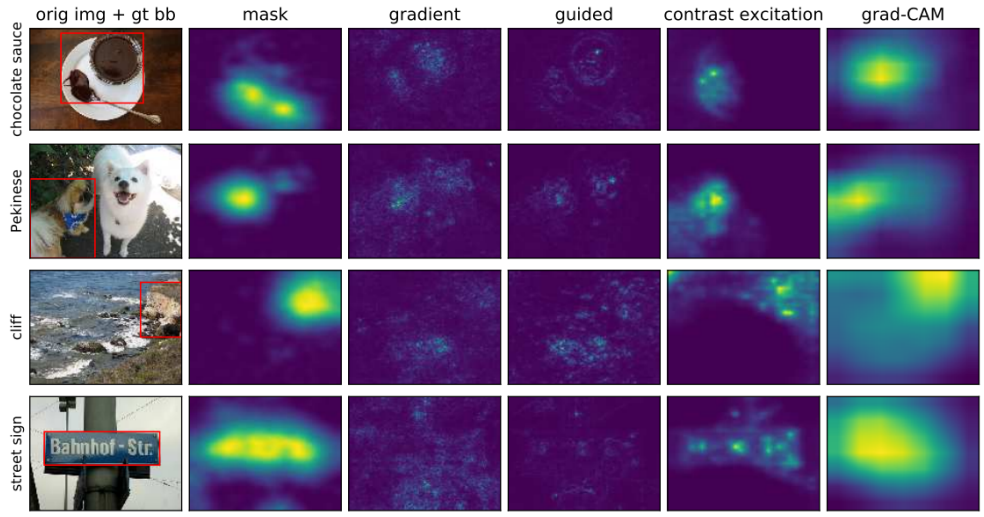


Figure 2.4: **Illustrations of some saliency methods.** The columns from left to right correspond to: the original image with its ground truth bounding box, perturbation masking from Fong et al. [46], the gradient method of Simonyan et al. [133], Guided Backpropagation [137], Excitation Backprop [168] and Grad-CAM [128]. These images are taken from Fong et al. [46].

across colour channels. The intuition behind this method is that the most salient pixel should be the one leading to the biggest change in the target class score. As the resulting saliency maps are noisy, some work combines this method with noise reduction techniques. We refer to SmoothGrad [134] which aggregates saliency maps computed on augmented views of the input image using a pixel-wise Gaussian noise. Similarly, Integrated Gradients [143] combines the saliency maps of linear interpolations between the input image and a black reference image.

Rather than considering the gradients with respect to the input image, CAM [173] exploits the activation maps at the end of the network. Indeed, the saliency map of CAM is the weighted sum of the activation at the target layer. The weights in this sum correspond to the weights of a linear classification layer trained on the spatial average of the activation. Hence, this method works well only at the last layers of the network, where the spatially averaged activation are class-discriminative enough. To avoid training a classification layer, Grad-CAM [128] replaces the classifier weights in the weighted sum by the spatial average of the gradient of the activation at the target layer. However, Grad-CAM still fails at earlier layers. To provide saliency maps at the different stages of the network, Kindermans et al. [79], Olah et al. [110] and Srinivas et al. [138] base their method on a linear approximation of the network. The resulting saliency map is the dot product between the activation and the gradient of the activation at the target layer.

In another line of work, some papers propose saliency methods which require to modify the backpropagation rules. As our proposed framework is based on the formula of the gradient of the weights, these methods do not fit in our framework. For example, based on the gradient method of [133], DeconvNet [165] and Guided Backpropagation [137] propose to reduce the noise in the saliency map by backpropagating only the positive gradients in the ReLU layers throughout the network. Excitation Backprop [168] goes even further and deeply modifies the backpropagation rules to compute in a top-down manner the winning probability of each neuron for their Winner-Take-All formulation. Moreover, other papers propose heuristic formulas to backpropagate saliency scores from the network output to the input pixels. For example, Bach et al. [8] backpropagate a quantity called layer-wise relevance which was proven in Ancona et al. [3] to be equivalent to the product of the gradient map and the activation map. We also refer to DeepLIFT [131] which proposes a saliency score based on the difference between activations.

2.5.2 Mask based saliency methods

Another family of saliency methods applies masks to the input of the model and derives saliency maps from the impact of these masks on the network output. The intuition behind these methods is that a spatial location is considered salient: (1) if preserving this region keeps the target class prediction unchanged or (2) if perturbing this region alters the target class prediction.

RISE [115] considers the preservation definition of saliency and applies random preserving masks (i.e. the image is perturbed except where the mask is applied) to several copies of the input image before feeding them to the network. The resulting saliency map is a weighted sum of these masks, where the weights correspond to the output scores of the target class. In the other case, using the perturbation definition of saliency, Fong et al. [46] apply perturbation masks (i.e. the image is preserved except where the mask is applied) to the input image and formulate the masking task as an optimisation problem where the mask is optimised to degrade the classification loss. The mask is also regularised using the total variation to enforce connected shapes and the L_1 norm to encourage smaller masks. Fong et al. [45] add a further regularisation term which constrains the mask area to a fixed size.

Moreover, masking methods are more computationally expensive than backpropagation saliency methods as the latter only require a few forward and backward passes whereas the former require hundreds of passes through the network for a single input image. Dabkowski et al. [32] address this problem by learning a network which

generates the masks rather than having to optimise them on each image. After an initial training phase, the network can then generate a saliency map with a single forward pass. Compared to backpropagation saliency methods, both the optimisation and network-generated masking approaches share the same downside of requiring carefully tuned hyperparameters to avoid producing trivial masks.

2.5.3 Evaluating saliency methods

Saliency methods produce nice visual interpretations of the network predictions. However, it would be misleading to evaluate these approaches with a human aesthetic judgement. Indeed, Adebayo et al. [1] show that some methods such as Guided Backpropagation [137] or DeconvNet [165], which were designed to improve the visual quality of saliency maps, fail a sanity check based on parameter randomisation. In fact, the failing methods keep producing the same saliency maps even when the network parameters are progressively re-initialised in a top-down manner. In such cases the saliency maps are invariant to the model and thus these approaches cannot be used to interpret or debug a trained model. In Chapter 8, we qualitatively assess the methods of our proposed framework and show that they pass this sanity check.

We also quantitatively evaluate our approach on the Pointing Game [168]. The goal of this benchmark is to assess the selectiveness of the saliency maps when localising a target object in a crowded environment. The evaluation consists in comparing the positions of the target object and the maximum point of the saliency map. If the positions are close enough, a hit is counted and the overall accuracy is obtained by dividing the number of hits by the number of pointing tests. An alternative to the Pointing Game is the task described in Cao et al. [18], whose goal is to predict bounding boxes for weakly-supervised localisation. The bounding boxes are obtained by first thresholding the saliency map and then fitting the tightest box which contains the remaining salient locations. Finally, the localisation is considered successful if the Intersection Over Union (IOU) between the predicted and the ground truth bounding boxes is above 0.5. However, weakly-supervised localisation has some downsides compared to the Pointing Game as the former requires to tweak the bounding box threshold for each method and this evaluation task does not suit saliency methods which aim at finding the smallest salient region such as the masking method of Fong et al. [46].

Chapter 3

Learning multiple visual domains with residual adapters

This work was presented as a Spotlight Presentation at the Conference on Neural Information Processing Systems (NeurIPS), 2017.

This paper tackles the problem of model sharing for multiple visual domains classification. It proposes residual adapters, parameter efficient modules which adapt a pre-trained deep network to the target domains. We also introduce a benchmark called *Visual Decathlon Challenge* which evaluates on ten different visual datasets the ability of parameter sharing methods to perform well across domains. On this benchmark we show that our proposed adapters perform similarly or better than standard fine-tuning with much fewer added weights.

Learning multiple visual domains with residual adapters

Sylvestre-Alvise Rebuffi[1] **Hakan Bilen**[1,2] **Andrea Vedaldi**[1]

[1] Visual Geometry Group
University of Oxford
{srebuffi,hbilen,vedaldi}@robots.ox.ac.uk

[2] School of Informatics
University of Edinburgh

Abstract

There is a growing interest in learning data representations that work well for many different types of problems and data. In this paper, we look in particular at the task of learning a single visual representation that can be successfully utilized in the analysis of very different types of images, from dog breeds to stop signs and digits. Inspired by recent work on learning networks that predict the parameters of another, we develop a tunable deep network architecture that, by means of adapter residual modules, can be steered on the fly to diverse visual domains. Our method achieves a high degree of parameter sharing while maintaining or even improving the accuracy of domain-specific representations. We also introduce the *Visual Decathlon Challenge*, a benchmark that evaluates the ability of representations to capture simultaneously ten very different visual domains and measures their ability to perform well uniformly.

1 Introduction

While research in machine learning is often directed at improving the performance of algorithms on specific tasks, there is a growing interest in developing methods that can tackle a large variety of different problems within a single model. In the case of perception, there are two distinct aspects of this challenge. The first one is to extract from a given image diverse information, such as image-level labels, semantic segments, object bounding boxes, object contours, occluding boundaries, vanishing points, etc. The second aspect is to model simultaneously many different visual domains, such as Internet images, characters, glyph, animal breeds, sketches, galaxies, planktons, etc (fig. 1).

In this work we explore the second challenge and look at how deep learning techniques can be used to learn *universal representations* [5], *i.e*. feature extractors that can work well in several different image domains. We refer to this problem as *multiple-domain learning* to distinguish it from the more generic multiple-task learning.

Multiple-domain learning contains in turn two sub-challenges. The first one is to develop algorithms that can *learn well from many domains*. If domains are learned sequentially, but this is not a requirement, this is reminiscent of *domain adaptation*. However, there are two important differences. First, in standard domain adaptation (*e.g*. [9]) the content of the images (e.g. "telephone") remains the same, and it is only the *style* of the images that changes (e.g. real life vs gallery image). Instead in our case a domain shift changes both style and content. Secondly, the difficulty is not just to adapt the model from one domain to another, but to do so while making sure that it still performs well on the original domain, *i.e*. to *learn without forgetting* [21].

The second challenge of multiple-domain learning, and our main concern in this paper, is to construct models that can represent compactly all the domains. Intuitively, even though images in different domains may look quite different (*e.g*. glyph *vs*. cats), low and mid-level visual primitives may still

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

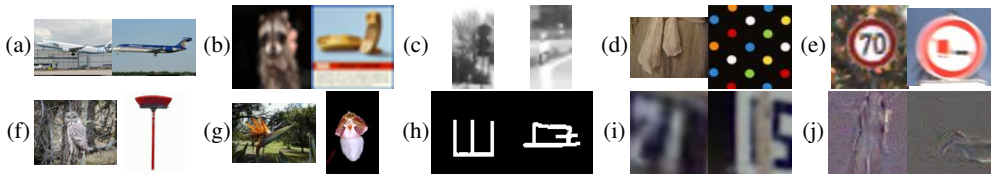


Figure 1: **Visual Decathlon.** We explore deep architectures that can learn simultaneously different tasks from very different visual domains. We experiment with ten representative ones: (a) Aircraft, (b) CIFAR-100, (c) Daimler Pedestrians, (d) Describable Textures, (e) German Traffic Signs, (f) ILSVRC (ImageNet) 2012, (g) VGG-Flowers, (h) OmniGlot, (i) SVHN, (j) UCF101 Dynamic Images.

be largely shareable. Sharing knowledge between domains should allow to learn compact multivalent representations. Provided that sufficient synergies between domains exist, multivalent representations may even work better than models trained individually on each domain (for a given amount of training data).

The primary contribution of this paper (section 3) is to introduce a design for **multivalent neural network architectures for multiple-domain learning** (section 3 fig. 2). The key idea is reconfigure a deep neural network on the fly to work on different domains as needed. Our construction is based on recent learning-to-learn methods that showed how the parameters of a deep network can be predicted from another [2, 16]. We show that these formulations are equivalent to packing the adaptation parameters in convolutional layers added to the network (section 3). The layers in the resulting parametric network are either domain-agnostic, hence shared between domains, or domain-specific, hence parametric. The domain-specific layers are changed based on the ground-truth domain of the input image, or based on an estimate of the latter obtained from an auxiliary network. In the latter configuration, our architecture is analogous to the *learnet* of [2].

Based on such general observations, we introduce in particular a **residual adapter module** and use it to parameterize the standard residual network architecture of [13]. The adapters contain a small fraction of the model parameters (less than 10%) enabling a high-degree of parameter sharing between domains. A similar architecture was concurrently proposed in [31], which also results in the possibility of learning new domains sequentially without forgetting. However, we also show a specific advantage of the residual adapter modules: the ability to modulate adaptation based on the size of the target dataset.

Our proposed architectures are thoroughly evaluated empirically (section 5). To this end, our second contribution is to introduce the **visual decathlon challenge** (fig. 1 and section 4), a new benchmark for multiple-domain learning in image recognition. The challenge consists in performing well simultaneously on ten very different visual classification problems, from ImageNet and SVHN to action classification and describable texture recognition. The evaluation metric, also inspired by the decathlon discipline, rewards models that perform better than strong baselines on all the domains simultaneously. A summary of our finding is contained in section 6.

2 Related Work

Our work touches on multi-task learning, learning without forgetting, domain adaptation, and other areas. However, our multiple-domain setup differs in ways that make most of the existing approaches not directly applicable to our problem.

Multi-task learning (MTL) looks at developing models that can address different tasks, such as detecting objects and segmenting images, while sharing information and computation among them. Earlier examples of this paradigm have focused on kernel methods [10, 1] and deep neural network (DNN) models [6]. In DNNs, a standard approach [6] is to share earlier layers of the network, training the tasks jointly by means of back-propagation. Caruana [6] shows that sharing network parameters between tasks is beneficial also as a form of regularization, putting additional constraints on the learned representation and thus improving it.

MTL in DNNs has been applied to various problems ranging from natural language processing [8, 22], speech recognition [14] to computer vision [41, 42, 4]. Collobert *et al*. [8] show that semi-supervised learning and multi-task learning can be combined in a DNN model to solve several language processing prediction tasks such as part-of-speech tags, chunks, named entity tags and semantic

roles. Huang *et al*. [14] propose a shared multilingual DNN which shares hidden layers across many languages. Liu *et al*. [22] combine multiple-domain classification and information retrieval for ranking web search with a DNN. Multi-task DNN models are also reported to achieve performance gains in computer vision problems such as object tracking [41], facial-landmark detection [42], object and part detection [4], a collection of low-level and high-level vision tasks [18]. The main focus of these works is learning a diverse set of tasks in the same visual domain. In contrast, our paper focuses on learning a representation from a diverse set of domains.

Our investigation is related to the recent paper of [5], which studied the "size" of the union of different visual domains measured in terms of the capacity of the model required to learn it. The authors propose to absorb different domain in a single neural network by tuning certain parameters in batch and instance normalization layers throughout the architecture; we show that our residual adapter modules, which include the latter as a special case, lead to far superior results.

Life-long learning. A particularly important aspect of MTL is the ability of learning multiple tasks sequentially, as in Never Ending Learning [25] and Life-long Learning [38]. Sequential learning typically suffers in fact from forgetting the older tasks, a phenomenon aptly referred to as "catastrophic forgetting" in [11]. Recent work in life-long learning try to address forgetting in two ways. The first one [37, 33] is to freeze the network parameters for the old tasks and learn a new task by adding extra parameters. The second one aims at preserving knowledge of the old tasks by retaining the response of the original network on the new task [21, 30], or by keeping the network parameters of the new task close to the original ones [17]. Our method can be considered as a hybrid of these two approaches, as it can be used to retain the knowledge of previous tasks exactly, while adding a small number of extra parameters for the new tasks.

Transfer learning. Sometimes one is interested in maximizing the performance of a model on a target domain. In this case, sequential learning can be used as a form of *initialization*[29]. This is very common in visual recognition, where most DNN are initialize on the ImageNet dataset and then *fine-tuned* on a target domain and task. Note, however, that this typically results in forgetting the original domain, a fact that we confirm in the experiments.

Domain adaptation. When domains are learned sequentially, our work can be related to domain adaptation. There is a vast literature in domain adaptation, including recent contributions in deep learning such as [12, 39] based on the idea of minimizing domain discrepancy. Long *et al*. [23] propose a deep network architecture for domain adaptation that can jointly learn adaptive classifiers and transferable features from labeled data in the source domain and unlabeled data in the target domain. There are two important differences with our work: First, in these cases different domains contain the same objects and is only the visual style that changes (*e.g*. webcam *vs*. DSLR), whereas in our case the object themselves change. Secondly, domain adaptation is a form of transfer learning, and, as the latter, is concerned with maximizing the performance on the target domain reagardless of potential forgetting.

3 Method

Our primary goal is to develop neural network architectures that can work well in a multiple-domain setting. Modern neural networks such as *residual networks* (ResNet [13]) are known to have very high capacity, and are therefore good candidates to learn from diverse data sources. Furthermore, even when domains look fairly different, they may still share a significant amount of low and mid-level visual patterns. Nevertheless, we show in the experiments (section 5) that learning a ResNet (or a similar model) directly from multiple domains may still not perform well.

In order to address this problem, we consider a *compact parametric family* of neural networks $\phi_\alpha : \mathcal{X} \to V$ indexed by parameters α. Concretely, $\mathcal{X} \subset \mathbb{R}^{H\times W\times 3}$ can be a space of RGB images and $V = \mathbb{R}^{H_v\times W_v\times C_v}$ a space of feature tensors. ϕ_α can then be obtained by taking all but the last classification layer of a standard ResNet model. The parametric feature extractors ϕ_α is then used to construct predictors for each domain d as $\Phi_d = \psi_d \circ \phi_{\alpha_d}$, where α_d are domain-specific parameters and $\psi_d(v) = \mathrm{softmax}(W_d v)$ is a domain-specific linear classifier $V \to \mathcal{Y}_d$ mapping features to image labels.

If α comprises *all* the parameters of the feature extractor ϕ_α, this approach reduces to learning independent models for each domain. On the contrary, our goal is to maximize parameter sharing, which we do below by introducing certain network parametrizations.

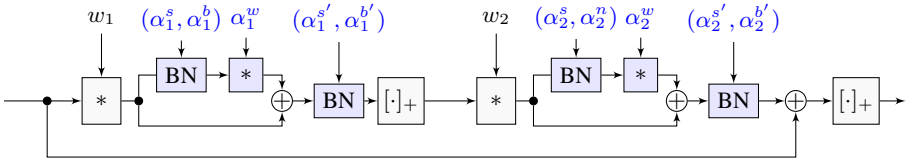


Figure 2: **Residual adapter modules.** The figure shows a standard residual module with the inclusion of adapter modules (in blue). The filter coefficients (w_1, w_2) are domain-agnostic and contains the vast majority of the model parameters; (α_1, α_2) contain instead a small number of domain-specific parameters.

3.1 Learning to learn and filter prediction

The problem of adapting a neural network dynamically to variations of the input data is similar to the one found in recent approaches to *learning to learn*. A few authors [34, 16, 2], in particular, have proposed to learn neural networks that predict, in a data-dependent manner, the parameters of another. Formally, we can write $\alpha_d = Ae_{d_x}$ where e_{d_x} is the indicator vector of the domain d_x of image x and A is a matrix whose columns are the parameter vectors α_d. As shown later, it is often easy to construct an auxiliary network that can predict d from x, so that the parameter $\alpha = \psi(x)$ can also be expressed as the output of a neural network. If d is known, then $\psi(x, d) = \alpha_d$ as before, and if not ψ can be constructed as suggested above or from scratch as done in [2].

The result of this construction is a network $\phi_{\psi(x)}(x)$ whose parameters are predicted by a second network $\psi(x)$. As noted in [2], while this construction is conceptually simple, its implementation is more subtle. Recall that the parameters w of a deep convolutional neural network consist primarily of the coefficients of the linear filters in the convolutional layers. If $w = \alpha$, then $\alpha = \psi(x)$ would need to predict millions of parameters (or to learn independent models when d is observed). The solution of [2] is to use a low-rank decomposition of the filters, where $w = \pi(w_0, \alpha)$ is a function of a filter basis w_0 and α is a small set of tunable parameters.

Here we build on the same idea, with some important extensions. First, we note that linearly parametrizing a filter bank is the same as *introducing a new, intermediate convolutional layer* in the network. Specifically, let $F_k \in \mathbb{R}^{H_f \times W_f \times C_f}$ be a basis of K filters of size $H_f \times W_f$ operating on C_f input feature channels. Given parameters $[\alpha_{tk}] \in \mathbb{R}^{T \times K}$, we can express a bank of T filters as linear combinations $G_t = \sum_{k=1}^K \alpha_{tk} F_k$. Applying the bank to a tensor x and using associativity and linearity of convolution results in $G * x = \sum_{k=1}^K \alpha_{:k}(F_k * x) = \alpha * F * x$ where we interpreted α as a $1 \times 1 \times T \times K$ filter bank. While [2] used a slightly different low-rank filter decomposition, their parametrization can also be seen as introducing additional filtering layers in the network.

An advantage of this parametrization is that it results in a useful decomposition, where part of the convolutional layers contain the domain-agnostic parameters F and the others contain the domain-specific ones α_d. As discussed in section 5, this is particularly useful to address the forgetting problem. In the next section we refine these ideas to obtain an effective parametrization of residual networks.

3.2 Residual adapter modules

As an example of parametric network, we propose to modify a standard residual network. Recall that a ResNet is a chain $g_m \circ \cdots \circ g_1$ of residual modules g_t. In the simplest variant of the model, each residual module g takes as input a tensor $\mathbb{R}^{H \times W \times C}$ and produces as output a tensor of the same size using $g(x; w) = x + ((w_2 * \cdot) \circ [\cdot]_+ \circ (w_1 * \cdot))(x)$. Here w_1 and w_2 are the coefficients of banks of small linear filters, $[z]_+ = \max\{0, z\}$ is the ReLU operator, $w * z$ is the convolution of z by the filter bank w, and $\circ$ denotes function composition. Note that, for the addition to make sense, filters must be configured such that the dimensions of the output of the last bank are the same as x.

Our goal is to parametrize the ResNet module. As suggested in the previous section, rather than changing the filter coefficients directly, we introduce additional parametric convolutional layers. In fact, we go one step beyond and make them small residual modules in their own right and call them

residual adapter modules (blue blocks in fig. 2). These modules have the form:

$$g(x;\alpha) = x + \alpha * x.$$

In order to limit the number of domain-specific parameters, α is selected to be a bank of 1×1 filters.

A major advantage of adopting a residual architecture for the adapter modules is that the adapters reduce to the identity function when their coefficients are zero. When learning the adapters on small domains, this provides a simple way of controlling over-fitting, resulting in substantially improved performance in some cases.

Batch normalization and scaling. Batch Normalization (BN) [15] is an important part of very deep neural networks. This module is usually inserted after convolutional layers in order to normalize their outputs and facilitate learning (fig. 2). The normalization operation is followed by rescaling and shift operations $s \odot x + b$, where (s, b) are learnable parameters. In our architecture, we incorporate the BN layers into the adapter modules (fig. 2). Furthermore, we add a BN module right before the adapter convolution layer.[1] Note that the BN scale and bias parameters are also dataset-dependent – as noted in the experiments, this alone provides a certain degree of model adaptation.

Domain-agnostic vs domain-specific parameters. If the residual module of fig. 2 is configured to process an input tensor with C feature channels, and if the domain-agnostic filters w_1, w_2 are of size $h \times h \times C$, then the model has $2(h^2C^2 + hC)$ domain-agnostic parameters (including biases in the convolutional layers) and $2(C^2 + 5C)$ domain-specific parameters.[2] Hence, there are approximately h^2 more domain-agnostic parameters than domain specific ones (usually $h^2 = 9$).

3.3 Sequential learning and avoiding forgetting

While in this paper we are not concerned with sequential learning, we have found it to be a good strategy to bootstrap a model when a large number of domains have to be learned. However, the most popular approach to sequential learning, *fine-tuning* (section 2), is often a poor choice for learning shared representations as it tends to quickly forget the original tasks.

The challenge in learning without forgetting is to maintain information about older tasks as new ones are learned (section 2). With respect to forgetting, our adapter modules are similar to the tower model [33] as they preserve the original model exactly: one can pre-train the domain-agnostic parameters w on a large domain such as ImageNet, and then fine-tune only the domain-specific parameters α_d for each new domain. Like the tower method, this preserves the original task exactly, but it is far less expensive as it does not require to introduce new feature channels for each new domain (a quadratic cost). Furthermore, the residual modules naturally reduce to the identity function when sufficient shrinking regularization is applied to the adapter weights α^w. This allows the adapter to be tuned depending on the availability of data for a target domain, sometimes significantly reducing overfitting.

4 Visual decathlon

In this section we introduce a new benchmark, called *visual decathlon*, to evaluate the performance of algorithms in multiple-domain learning. The goal of the benchmark is to assess whether a method can successfully learn to perform well in several different domains at the same time. We do so by choosing ten representative visual domains, from Internet images to characters, as well as by selecting an evaluation metric that rewards performing well on all tasks.

Datasets. The decathlon challenge combines ten well-known datasets from multiple visual domains: **FGVC-Aircraft Benchmark** [24] contains 10,000 images of aircraft, with 100 images for each of 100 different aircraft model variants such as Boeing 737-400, Airbus A310. **CIFAR100** [19] contains 60,000 32×32 colour images for 100 object categories. **Daimler Mono Pedestrian Classification Benchmark (DPed)** [26] consists of 50,000 grayscale pedestrian and non-pedestrian images, cropped and resized to 18×36 pixels. **Describable Texture Dataset (DTD)** [7] is a texture database, consisting of 5640 images, organized according to a list of 47 terms (categories) such as bubbly, cracked,

[1] While the bias and scale parameters of the latter can be incorporated in the following filter bank, we found it easier to leave them separated from the latter

[2] Including all bias and scaling vectors; $2(C^2 + 3C)$ if these are absorbed in the filter banks when possible.

marbled. **The German Traffic Sign Recognition (GTSR) Benchmark** [36] contains cropped images for 43 common traffic sign categories in different image resolutions. **Flowers102** [28] is a fine-grained classification task which contains 102 flower categories from the UK, each consisting of between 40 and 258 images. **ILSVRC12 (ImNet)** [32] is the largest dataset in our benchmark contains 1000 categories and 1.2 million images. **Omniglot** [20] consists of 1623 different handwritten characters from 50 different alphabets. Although the dataset is designed for one-shot learning, we use the dataset for standard multi-class classification task and include all the character categories in train and test splits. **The Street View House Numbers (SVHN)** [27] is a real-world digit recognition dataset with around 70,000 32×32 images. **UCF101** [35] is an action recognition dataset of realistic human action videos, collected from YouTube. It contains 13,320 videos for 101 action categories. In order to make this dataset compatible with our benchmark, we convert the videos into images by using the Dynamic Image encoding of [3] which summarizes each video into an image based on a ranking principle.

Challenge and evaluation. Each dataset $\mathcal{D}_d$, $d = 1, \ldots, 10$ is formed of pairs $(x, y) \in \mathcal{D}_d$ where x is an image and $y \in \{1, \ldots, C_d\} = \mathcal{Y}_d$ is a label. For each dataset, we specify a training, validation and test subsets. The goal is to train the best possible model to address all ten classification tasks using only the provided training and validation data (no external data is allowed). A model Φ is evaluated on the test data, where, given an image x and its ground-truth domain d_x label, it has to predict the corresponding label $y = \Phi(x, d_x) \in \mathcal{Y}_d$.

Performance is measured in terms of a single scalar score S determined as in the *decathlon* discipline. Performing well at this metric requires algorithms to perform well in all tasks, compared to a minimum level of baseline performance for each. In detail, S is computed as follows:

$$S = \sum_{d=1}^{10} \alpha_d \max\{0, E_d^{\text{max}} - E_d\}^{\gamma_d}, \qquad E_d = \frac{1}{|\mathcal{D}_d^{\text{test}}|} \sum_{(x,y) \in \mathcal{D}_d^{\text{test}}} \mathbf{1}_{\{y \neq \Phi(x,d)\}}. \tag{1}$$

where E_d is the average test error for each domain. E_d^{max} the baseline error (section 5), above which no points are scored. The exponent $\gamma_d \geq 1$ rewards more reductions of the classification error as this becomes close to zero and is set to $\gamma_d = 2$ for all domains. The coefficient α_d is set to $1,000\,(E_d^{\text{max}})^{-\gamma_d}$ so that a perfect result receives a score of 1,000 (10,000 in total).

Data preprocessing. Different domains contain a different set of image classes as well as a different number of images. In order to reduce the computational burden, all images have been resized isotropically to have a shorter side of 72 pixels. For some datasets such as ImageNet, this is a substantial reduction in resolution which makes training models much faster (but still sufficient to obtain excellent classification results with baseline models). For the datasets for which there exists training, validation, and test subsets, we keep the original splits. For the rest, we use 60%, 20% and 20% of the data for training, validation, and test respectively. For the ILSVRC12, since the test labels are not available, we use the original validation subset as the test subset and randomly sample a new validation set from their training split. We are planning to make the data and an evaluation server public soon.

5 Experiments

In this section we evaluate our method quantitatively against several baselines (section 5.1), investigate the ability of the proposed techniques to learn models for ten very diverse visual domains.

Implementation details. In all experiments we choose to use the powerful ResNets [13] as base architectures due to their remarkable performance. In particular, as a compromise of accuracy and speed, we chose the ResNet28 model [40] which consists of three blocks of four residual units. Each residual unit contains 3×3 convolutional, BN and ReLU modules (fig. 2). The network accepts 64×64 images as input, downscales the spatial dimensions by two at each block and ends with a global average pooling and a classifier layer followed by a softmax. We set the number of filters to $64, 128, 256$ for these blocks respectively. Each network is optimized to minimize its cross-entropy loss with stochastic gradient descent. The network is run for 80 epochs and the initial learning rate of 0.1 is lowered to 0.01 and then 0.001 gradually.

Model	#par.	ImNet	Airc.	C100	DPed	DTD	GTSR	Flwr	OGlt	SVHN	UCF	mean	S
# images		1.3m	7k	50k	30k	4k	40k	2k	26k	70k	9k		
Scratch	$10\times$	59.87	57.10	75.73	91.20	37.77	96.55	56.30	88.74	96.63	43.27	70.32	1625
Scratch+	$11\times$	59.67	59.59	76.08	92.45	39.63	96.90	56.66	88.74	96.78	44.17	71.07	1826
Feature extractor	$1\times$	59.67	23.31	63.11	80.33	45.37	68.16	73.69	58.79	43.54	26.80	54.28	544
Finetune	$10\times$	59.87	60.34	82.12	92.82	55.53	97.53	81.41	87.69	96.55	51.20	76.51	2500
LwF [21]	$10\times$	59.87	61.15	82.23	92.34	58.83	97.57	83.05	88.08	96.10	50.04	76.93	2515
BN adapt. [5]	$\sim 1\times$	59.87	43.05	78.62	92.07	51.60	95.82	74.14	84.83	94.10	43.51	71.76	1363
Res. adapt.	$2\times$	59.67	56.68	81.20	93.88	50.85	97.05	66.24	89.62	96.13	47.45	73.88	2118
Res. adapt. decay	$2\times$	59.67	61.87	81.20	93.88	57.13	97.57	81.67	89.62	96.13	50.12	76.89	2621
Res. adapt. finetune all	$2\times$	59.23	63.73	81.31	93.30	57.02	97.47	83.43	89.82	96.17	50.28	77.17	2643
Res. adapt. dom-pred	$2.5\times$	59.18	63.52	81.12	93.29	54.93	97.20	82.29	89.82	95.99	50.10	76.74	2503
Res. adapt. (large)	$\sim 12\times$	67.00	67.69	84.69	94.28	59.41	97.43	84.86	89.92	96.59	52.39	79.43	3131

Table 1: Multiple-domain networks. The figure reports the (top-1) classification accuracy (%) of different models on the decathlon tasks and final decathlon score (S). ImageNet is used to prime the network in every case, except for the networks trained from scratch. The model size is the number of parameters w.r.t. the baseline ResNet. The fully-finetuned model, written blue, is used as a baseline to compute the decathlon score.

Model	Airc.		C100		DPed		DTD		GTSR		Flwr		OGlt		SVHN		UCF	
Finetune	1.1	60.3	3.6	63.1	0.6	80.3	0.7	45.3	1.4	68.1	27.2	73.6	13.4	87.7	0.2	96.6	5.4	51.2
LwF [21] high lr	4.1	61.1	21.0	82.2	23.8	92.3	36.7	58.8	11.5	97.6	34.2	83.1	3.0	88.1	0.2	96.1	18.6	50.0
LwF [21] low lr	38.0	50.6	33.0	80.7	53.3	92.2	47.0	57.2	23.7	96.6	45.7	75.7	21.0	86.0	13.3	94.8	29.0	44.6
Res. adapt. finetune all	59.2	63.7	59.2	81.3	59.2	93.3	59.2	57.0	59.2	97.5	59.2	83.4	59.2	89.8	59.2	96.1	59.2	50.3

Table 2: Pairwise forgetting. Each pair of numbers report the top-1 accuracy (%) on the old task (ImageNet) and a new target task after the network is fully finetuned on the latter. We also show the performance of LwF when it is finetuned on the new task with a high and low learning rate, trading-off forgetting ImageNet and improving the results on the target domain. By comparison, we show the performance of tuning only the residual adapters, which by construction does not result in any performance loss in ImageNet while still achieving very good performance on each target task.

5.1 Results

There are two possible extremes. The first one is to learn ten independent models, one for each dataset, and the second one is to learn a single model where all feature extractor parameters are shared between the ten domains. We evaluate next different approaches to learn such models.

Pairwise learning. In the first experiment (table 1), we start by learning a ResNet model on ImageNet, and then use different techniques to extend it to the remaining nine tasks, one at a time. Depending on the method, this may produce an overall model comprising ten ResNet architectures, or just one ResNet with a few domain-specific parameters; thus we also report the total number of parameters used, where $1\times$ is the size of a single ResNet (excluding the last classification layer, which can never be shared).

As baselines, we evaluate four cases: i) learning an individual ResNet model from scratch for each task, ii) freezing all the parameters of the pre-trained network, using the network as feature extractor and only learn a linear classifier, iii) standard finetuning and iv) applying a reimplementation of the LwF technique of [21] that encourages the fine-tuned network to retain the responses of the original ImageNet model while learning the new task.

In terms of accuracy, learning from scratch performs poorly on small target datasets and, by learning 10 independent models, requires $10\times$ parameters in total. Freezing the ImageNet feature extraction is very efficient in terms of parameter sharing ($1\times$ parameters in total), preserves the original domain exactly, but generally performs very poorly on the target domain. Full fine-tuning leads to accurate results both for large and small datasets; however, it also forgets the ImageNet domain substantially (table 2), so it still requires learning 10 complete ResNet models for good overall performance.

When LwF is run as intended by the original authors [21], is still leads to a noticeable performance drop on the original task, even when learning just two domains (table 2), particularly if the target domain is very different from ImageNet (*e.g.* Omniglot and SVHN). Still, if one chooses a different trade-off point and allows the method to forget ImageNet more, it can function as a good regularizer that slightly outperforms vanilla fine-tuning overall (but still resulting in a $10\times$ model).

Next, we evaluate the effect of sharing the majority of parameters between tasks, whereas still allowing a small number of domain-specific parameters to change. First, we consider specializing only the BN layer scaling and bias parameters, which is equivalent to the approach of [5]. In this case, less than the 0.1% of the model parameters are domain-specific (for the ten domains, this results in a model with $1.01\times$ parameters overall). Hence the model is very similar to the one with the frozen feature extractor; nevertheless, the performances increase very substantially in most cases (e.g. $23.31\% \rightarrow 43.05\%$ accuracy on Aircraft).

As the next step, we introduce the **residual adapter modules**, which increase by 11% the number of parameters per domain, resulting in a $2\times$ model. In the pre-training phase, we first pretrain on ImageNet the network with the added modules. Then, we freeze the task agnostic parameters and train the task specific parameters on the different datasets. Differently from vanilla fine-tuning, there is no forgetting in this setting. While most of the parameters are shared, our method is either close or better than full fine-tuning. As a further control, we also train 10 models from scratch with the added parameters (denoted as Scratch+), but do not observe any noticeable performance gain in average, demonstrating that parameters sharing is highly beneficial. We also contrast learning the adapter modules with two values of weight decay (0.002 and 0.005) higher than the default 0.0005. These parameters are obtained after a coarse grid search using cross-validation for each dataset. Using higher decay significantly improves the performance on smaller datasets such as Flowers, whereas the smaller decay is best for larger datasets. This shows both the importance and utility of controlling overfitting in the adaptation process. In practice, there is an almost direct correspondence between the size of the data and which one of these values to use. The optimal decay can be selected via validation, but a rough choice can be performed by simply looking at the dataset size.

We also compare to another baseline where we only finetune the last two convolutional layers and freeze the others, which may be thought to be generic. This amounts to having a network with twice the number of total parameters in a vanilla ResNet which is equal to our proposed architecture. This model obtains 64.7% mean accuracy over ten datasets, which is significantly lower than our 73.9%, likely due to overfitting (controlling overfitting is one of the advantages of our technique).

Furthermore, we also assess the quality of our adapter without residual connections, which corresponds to the low rank filter parametrization of section 3.1; this approach achieves an accuracy of 70.3%, which is worse than our 73.9%. We also observe that this configuration requires notably more iterations to converge. Hence, the residual architecture for the adapters results in better performances, better control of overfitting, and a faster convergence.

End-to-end learning. So far, we have shown that our method, by learning only the adapter modules for each new domain, does not suffer from forgetting. However, for us sequential learning is just a scalable learning strategy. Here, we also show (table 1) that we can further improve the results by fine-tuning all the parameters of the network end-to-end on the ten tasks. We do so by sampling a batch from each dataset in a round robin fashion, allowing each domain to contribute to the shared parameters. A final pass is done on the adapter modules to take into account the change in the shared parameters.

Domain prediction. Up to now we assume that the domain of each image is given during test time for all the methods. If this is unavailable, it can be predicted on the fly by means of a small neural-network predictor. We train a light ResNet, which is composed three stacks of two residual networks, half deep as the original net, obtaining 99.8% accuracy in domain prediction, resulting in a barely noticeable drop in the overall multiple-domain challenge (see Res. adapt dom-pred in table 1). Note that similar performance drop would be observed for the other baselines.

Decathlon evaluation: overall performance. While so far we have looked at results on individual domain, the Decathlon score eq. (1) can be used to compare performance overall. As baseline error rates in eq. (1), we double the error rates of the fully finetuned networks on each task. In this manner, this $10\times$ model achieves a score of 2,500 points (over 10,000 possible ones, see eq. (1)). The last column of table 1 reports the scores achieved by the other architectures. As intended, the decathlon score favors the methods that perform well overall, emphasizes their *consistency* rather than just their average accuracy. For instance, although the Res. adapt. model (trained with single decay coefficient for all domains) performs well in terms of average accuracy (73.88%), its decathlon score (2118) is relatively low because the model performs poorly in DTD and Flowers. This also shows that, once the weight decays are configured properly, our model achieves superior performance (2643 points) to all the baselines using only $2\times$ the capacity of a single ResNet.

Finally we show that using a higher capacity ResNet28 ($12\times$, ResNet adapt. (large) in table 1), which is comparable to 10 independent networks, significantly improves our results and outperforms the finetuning baseline by 600 point in decathlon score. As a matter of fact, this model outperforms the state-of-the-art [40] (81.2%) by 3.5 points in CIFAR100. In other cases, our performances are in general in line to current state-of-the-art methods. When this is not the case, this is due to reduced image resolution (ImageNet, Flower) or due to the choice of a specific video representation in UCF (dynamic image).

6 Conclusions

As machine learning applications become more advanced and pervasive, building data representations that work well for multiple problems will become increasingly important. In this paper, we have introduced a simple architectural element, the residual adapter module, that allows compressing many visual domains in relatively small residual networks, with substantial parameter sharing between them. We have also shown that they allow addressing the forgetting problem, as well as adapting to target domain for which different amounts of training data are available. Finally, we have introduced a new multi-domain learning challenge, the Visual Decathlon, to allow a systematic comparison of algorithms for multiple-domain learning.

Acknowledgments: This work acknowledges the support of Mathworks/DTA DFR02620 and ERC 677195-IDIU.

References

[1] A. Argyriou, T. Evgeniou, and M Pontil. Multi-task feature learning. In *Proc. NIPS*, volume 19, page 41. MIT; 1998, 2007.

[2] L. Bertinetto, J. F. Henriques, J. Valmadre, P. Torr, and A. Vedaldi. Learning feed-forward one-shot learners. In *Proc. NIPS*, pages 523–531, 2016.

[3] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *Proc. CVPR*, 2016.

[4] H. Bilen and A. Vedaldi. Integrated perception with recurrent multi-task neural networks. In *Proc. NIPS*, 2016.

[5] H. Bilen and A. Vedaldi. Universal representations: The missing link between faces, text, planktons, and cat breeds. *arXiv preprint arXiv:1701.07275*, 2017.

[6] R. Caruana. Multitask learning. *Machine Learning*, 28, 1997.

[7] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *Proc. CVPR*, 2014.

[8] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *icml*, pages 160–167. ACM, 2008.

[9] H. Daumé III. Frustratingly easy domain adaptation. *ACL 2007*, page 256, 2007.

[10] T. Evgeniou and M. Pontil. Regularized multi–task learning. In *SIGKDD*, pages 109–117. ACM, 2004.

[11] R. M. French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[12] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. *Proc. ICML*, 2015.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *Proc. ECCV*, pages 630–645. Springer, 2016.

[14] J. T. Huang, J. Li, D. Yu, L. Deng, and Y. Gong. Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *ICASSP*, pages 7304–7308, 2013.

[15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, 2015.

[16] X. Jia, B. De Brabandere, T. Tuytelaars, and L. Gool. Dynamic filter networks. In *Proc. NIPS*, pages 667–675, 2016.

[17] J. Kirkpatrick, E. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *National Academy of Sciences*, 2017.

[18] I. Kokkinos. Ubernet: Training auniversal'convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. *Proc. CVPR*, 2017.

[19] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[20] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[21] Z. Li and D. Hoiem. Learning without forgetting. In *Proc. ECCV*, pages 614–629, 2016.

[22] X. Liu, J. Gao, X. He, L. Deng, K. Duh, and Y. Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *HLT-NAACL*, pages 912–921, 2015.

[23] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Unsupervised Domain Adaptation with Residual Transfer Networks. In *Proc. NIPS*, pages 136–144, 2016.

[24] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013.

[25] T. Mitchell. Never-ending learning. Technical report, DTIC Document, 2010.

[26] S. Munder and D. M. Gavrila. An experimental study on pedestrian classification. *PAMI*, 28(11):1863–1868, 2006.

[27] 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.

[28] M-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *ICCVGIP*, Dec 2008.

[29] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *CVPR DeepVision Workshop*, 2014.

[30] S. A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *Proc. CVPR*, 2017.

[31] Amir Rosenfeld and John K Tsotsos. Incremental learning through deep adaptation. *arXiv preprint arXiv:1705.04228*, 2017.

[32] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and K. Fei-Fei. Imagenet large scale visual recognition challenge, 2014.

[33] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[34] J. Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

[35] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[36] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32(0):323–332, 2012.

[37] A. V. Terekhov, G. Montone, and J. K. O'Regan. Knowledge transfer in deep block-modular neural networks. In *Biomimetic and Biohybrid Systems*, pages 268–279, 2015.

[38] S. Thrun. Lifelong learning algorithms. In *Learning to learn*, pages 181–209. Springer, 1998.

[39] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Simultaneous deep transfer across domains and tasks. In *Proc. CVPR*, pages 4068–4076, 2015.

[40] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[41] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Robust visual tracking via structured multi-task sparse learning. *IJCV*, 101(2):367–383, 2013.

[42] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Facial landmark detection by deep multi-task learning. In *Proc. ECCV*, 2014.

Chapter 4

Efficient parametrization of multi-domain deep neural networks

This work was presented as a Poster Presentation at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

This paper is a follow-up work of Chapter 3 where we further study the concept of residual adapters. We propose an improvement of these adapters by adopting a parallel design rather than adding the adapters in series. Compared to the series adapters, we show that the parallel adapters are plug-and-play and can be directly used with any off-the-shelf pre-trained network. We also introduce a cross-domain adapter compression which improves the classification performance while reducing the memory footprint of the adapters. Finally, we present some good practice guidelines, for example how to regularise the adapters depending on the dataset and network sizes, or that it is beneficial to add adapters all along the network instead of the transfer learning practice of freezing the early layers.

Efficient parametrization of multi-domain deep neural networks

Sylvestre-Alvise Rebuffi[1] Hakan Bilen[2] Andrea Vedaldi[1]

[1]Visual Geometry Group
University of Oxford
{srebuffi,vedaldi}@robots.ox.ac.uk

[2]School of Informatics
University of Edinburgh
hbilen@ed.ac.uk

Abstract

A practical limitation of deep neural networks is their high degree of specialization to a single task and visual domain. Recently, inspired by the successes of transfer learning, several authors have proposed to learn instead universal feature extractors that, used as the first stage of any deep network, work well for several tasks and domains simultaneously. Nevertheless, such universal features are still somewhat inferior to specialized networks.

To overcome this limitation, in this paper we propose to consider instead universal parametric families of neural networks*, which still contain specialized problem-specific models, but differing only by a small number of parameters. We study different designs for such parametrizations, including series and parallel residual adapters, joint adapter compression, and parameter allocations, and empirically identify the ones that yield the highest compression. We show that, in order to maximize performance, it is necessary to adapt both shallow and deep layers of a deep network, but the required changes are very small. We also show that these universal parametrization are very effective for transfer learning, where they outperform traditional fine-tuning techniques.*

1. Introduction

As deep neural networks continue to dramatically improve results in almost all traditional problems in computer vision, the interest of the community has started to shift towards more ambitious goals. One of them is to supersede the common paradigm of addressing different image understanding problems independently, using ad-hoc solutions and learning different and largely incompatible models for each of them. Just like the human brain is capable of addressing a very large number of different image analysis tasks, so it should be possible to develop models that address well and efficiently a variety of different computer vision problems, with better efficiency and generalization than individual networks.

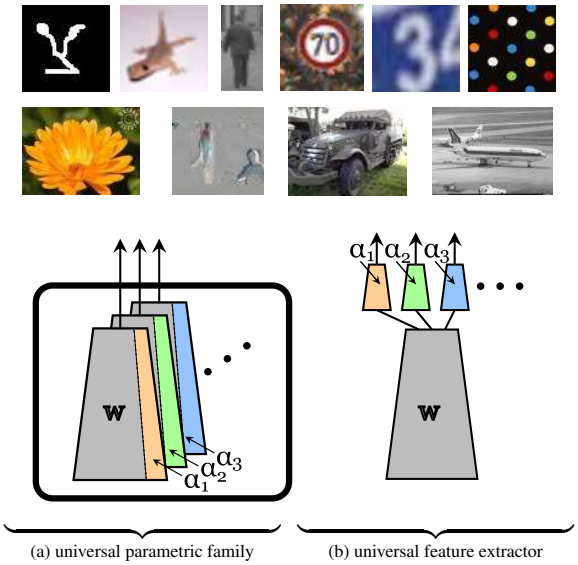


Figure 1: **Universal parametric network families.** We develop compact parametric families of neural networks (a) that can target very different visual domains, from ImageNet to stop signs and characters, while sharing the vast majority of their parameters $\mathbf{w}$. Domain-specific parameters α_t are isolated in small modular adapters that can be attached to an existing network to steer it non-disruptively to different domains and enable efficient model storage, transfer, and exchange, as well as transfer learning. Parametric families are shown empirically to be much more powerful than sharing a fixed universal feature extractor as in (b).

There are at least three aspects to this challenge. The first is to construct a *multi-task model* that can extract multiple types of information from an image, performing class/object detection and segmentation, boundary extraction, motion estimation, etc. [14]. The second is to develop a *multi-domain model* that can work well for many different visual domains, such as Internet images, scene text, medical images, satellite images, driving images, etc [3, 23]. The

third is to develop an *extensible model* that can evolve over time, reusing previously acquired knowledge to learn to process new tasks and domains efficiently, while at the same time avoiding to forget previously-acquired abilities [18].

Concerned with the second and third problem, several authors before us have framed this as the problem of learning a single universal first-stage to be shared among different deep networks (fig. 1.b). The idea is that early layers should process low-level and hence widely-applicable visual information. However, such universal feature extractors do not work quite as well as learning problem-specific networks, either from scratch or using transfer learning.

In this paper, we propose an alternative perspective. Instead of seeking a single, fixed first stage, we want to develop compact parametrizations for multi-domain networks (fig. 1.a). Consider a deep network $\Phi(\mathbf{x}; \mathbf{w}, \alpha)$ applied to an image $\mathbf{x}$, for example for image classification. We partition the network parameters in a universal vector $\mathbf{w}$, which is fixed and shared among all domains, and a parameter vector α, which is instead domain specific. We then seek architectures that: 1) can share the vast majority of their parameters, so that the size of α is a small fraction of the size of $\mathbf{w}$, and 2) can learn a new α for a new domain from a very small number of training examples. In other words, we would like to compress a family of domain-specific neural networks so that they can be exchanged and learned more efficiently.

While our method does not result in a single, universal neural network, as the parameters α are still domain-specific, finding architectures that afford a great degree of parameter sharing is an important step in this direction. There are also concrete practical benefits. First, universal families work better than the standard transfer learning approach of fine-tuning off-the-shelf models; hence, they may replace the latter strategy in numerous applications. Second, there are applications such as mobile devices that require running several different neural networks, which may incur a significant computational and energy overhead due simply to the need of swapping their parameters on a dedicated integrated circuit. One may face similar overheads when transmitting model parameters over a network, or storing them locally. Our approach makes storing, exchanging, and updating models much more efficient.

Related to our work, a few papers [23, 25] have proposed low-dimensional parametrizations of the filters in a neural network with good compression results. The paper of [23], in particular, proposed the idea of *residual adapters* to build networks with a high-degree of parameter sharing. In this work, we propose some important improvements over this basic module. First, we show that a simple change, where the topology of the adapter is *parallel* rather than series, results in major improvements across the board, in terms of overall accuracy, applicability to existing off-the-shelf network, and transfer learning. Second, we investigate which parts of typical networks require adaptation, and we show that often *both early and late layers* need to be adapted to obtain the best performance. Third, we experiment with different regularization strategies for the adapters such as dropout which proves highly beneficial when using a bigger pretrained network. Fourth, we introduce a cross-domain compression procedure for the adapters which allows to reduce significantly the numbers of adapter parameters. Most importantly, this compression contributes to multi-domain regularization resulting in improved overall performance thanks to information sharing among target datasets.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 describes our neural network parametrization and how it applies to state-of-the-art neural network architectures. Section 4 demonstrates empirically the power of our approach on standard datasets, setting in particular the new state of the art on the Visual Decathlon benchmark, as well as demonstrating excellent transfer learning capabilities. Finally, section 5 summarizes our findings.

2. Related Work

Our work intersects with various lines of research in multi-task learning, learning without forgetting, domain adaptation, and other areas.

Multi-task learning (MTL) aims at learning multiple related tasks simultaneously by sharing information and computation among them. Early work [5] in this area focuses on deep neural network (DNN) models which share weights in the earlier layers and use specialized ones in the later layers. It is shown in [5] that sharing parameters during training helps exploiting regularities present across tasks and improving the performance by constraining the learned representation. However this setting requires to manually design the network and decide which layers should be shared across multiple tasks. This paradigm is applied to various learning problems from natural language processing [6] and automated drug discovery [7] to speech recognition [12]. In computer vision, deep MTL models are applied to object tracking [34], facial-landmark detection [35], object and part detection [2], object detection and instance segmentation [10], a collection of low-level and high-level vision tasks [14]. Differently from our work, this line of research focuses on learning a diverse set of tasks in the same visual domain.

Multi-domain learning. Our method is most related to recent works [3, 23, 25] which aim at learning a single network to perform image classification tasks in a diverse set of domains. The main focus is to learn a single network that can represent compactly all the domains with minimal number of task specific parameters. To do so, Bilen and Vedaldi [3] propose to model different domains in a single neural network by sharing all core model parameters

except parameters in batch and instance normalization layers. Rebuffi *et al*. [23] extend [3] and propose a new parameterization of the standard residual network architecture that enables a high degree of parameter sharing between domains with a small increase ($< 10\%$) in the model parameters. The authors of [25] propose a parameter-efficient architecture that enables learning new domains sequentially without forgetting. We build our method on [23, 25] and significantly improve over them in terms of accuracy and compression ratio by introducing a novel and more compact adapter module, and a better regularization strategy.

Parameterized MTL. Another MTL approach [1, 32, 20] focuses on dynamically generating DNN weights given the task identity. Bertinetto *et al*. [1] propose a method to learn the parameters of a deep model from a single exemplar for one-shot classification. As a naive predicting of high dimensional weights is not feasible, the authors first obtain a low rank decomposition of filters and define the new network as a linear combination of the low-rank filters. Similarly, the authors of [32] propose a tensor factorization method that can realize automatic learning of end-to-end knowledge sharing in deep networks. Meyerson and Miikkulainen [20] propose a soft ordering approach, which dynamically computes to what extent each filter contributes to each tasks and thus how much is shared across different tasks. As a matter of fact, we also use a similar low rank decomposition technique to the one in [1]. However, the decomposition is used to design a more compact sharing across tasks.

Domain adaptation. There is a rich body of work in domain adaptation including the ones in deep learning such as [9, 31] that minimizes the domain discrepancy. The authors of [19] propose a deep network architecture that can jointly learn adaptive classifiers and transferable features from the source to target domain by modeling source classifier as sum of target classifier and a residual function. Bousmalis *et al*. [4] consider an explicit parameterization of domain-generic and domain-specific that learns to extract image representations from the partitioned subspaces. Li *et al*. [17] propose a meta-learning method that trains any given model to be more robust to domain shift. Our method differs to this group of work in two important aspects: First, in addition to domain change (*e.g*. DSLR *vs*. webcam), each domain contains a unique set of outputs (*i.e*. object categories) in our case. Second, domain adaptation typically aims to maximise performance on the target domain regardless of potential forgetting.

Life-long learning. Another important research direction in MTL is sequential learning of multiple tasks [21, 30]. While the key idea is to exploit the knowledge from the previous tasks, learning sequentially typically suffers from forgetting the previous tasks, a phenomenon referred as "catastrophic forgetting" in [8]. Recent work [29, 26] address this problem by freezing the network parameters for the old tasks and only updating the parameters of the new task which leads to a linear growth in the number of total parameters with the number of tasks. Another approach is to preserve the previous knowledge by retaining the response of the original network on the new task [18, 24]. The problem is also addressed by keeping the network parameters [13] and features [22] of the new task close to the original ones. Our method can also be related to both [26, 13], as it retains the knowledge of previous tasks perfectly, while adding a small number of extra parameters for the new tasks.

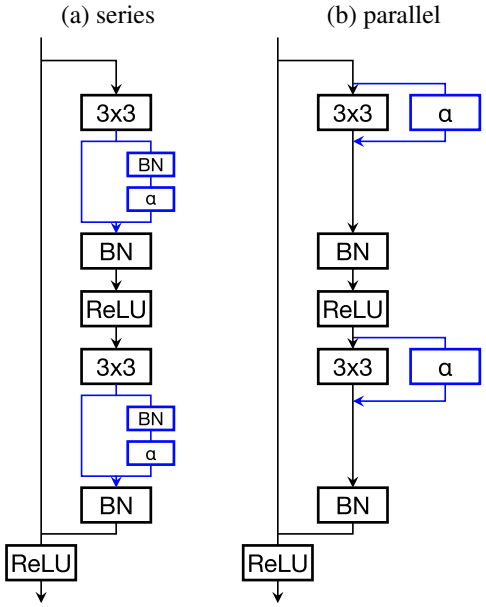


Figure 2: Series *vs* parallel residual adapters. (a) typical module of a residual network inclusive of batch normalization layers and residual adapters (in blue). (b) the same configuration, but with parallel adapters instead, resulting in a simpler network.

3. Method

This section describes different ways of constructing a parametric family of neural networks that can tackle multiple domains while sharing the vast majority of their parameters. Section 3.1 introduces a number of *adapter modules*. These modules attach to a standard deep neural network architecture such as ResNet [11] to steer it to different problems by means of a small number of adaptation parameters. Section 3.2 discusses different ways in which residual adapters can be injected in a standard neural network, section 3.3 how they can be regularized, and section 3.4 how the parameters can be further compressed.

3.1. Adapter modules

We begin by reviewing the recent adapter modules of [23] (section 3.1.1). We then discuss a number of alternative designs that, as shown empirically in section 4, perform significantly better (3.1.2). These modules are illustrated in fig. 2.

3.1.1 Series residual adapters

The residual adapter modules introduced by [23] consist of a 1×1 filter bank in parallel with a skip connection (fig. 2.a):

$$\mathbf{y} = \rho(\mathbf{x};\alpha) = \mathbf{x} + \mathrm{diag}_1(\alpha) * \mathbf{x}.$$

If the input tensor has shape $\mathbf{x} \in \mathbb{R}^{H\times W\times C}$, then $\alpha \in \mathbb{R}^{C\times C}$ has $O(C^2)$ parameters. Here we use the operator $\mathrm{diag}_L(A) \in \mathbb{R}^{L\times L\times C\times D}$ to reshape a matrix $A \in \mathbb{R}^{C\times D}$ in a bank of "diagonal" filters:

$$[\mathrm{diag}_L(A)]_{vucd} = \begin{cases} A_{dc}, & v = u = (L-1)/2+1, \\ 0, & \text{otherwise.} \end{cases}$$

This operator transforms the matrix A into a 1×1 filter bank embedded as the central element of a larger $L\times L$ filter bank by appending zeros around it (L is assumed to be odd).

An advantage of this relatively cumbersome notation is that we can rewrite the module as a single filter:

$$\rho(\mathbf{x};\alpha) = \mathrm{diag}_1(I+\alpha) * \mathbf{x}$$

The rationale for the additive parameterization is that the identity function is recovered if $\alpha = 0$. This is the case when a strong regularizer is applied on α during learning, shrinking the weights towards zero. In turn, this allow to easily control the adaptation strength, and thus generalization.

Residual adapters are installed in series with standard filter banks $\mathbf{f} \in \mathbb{R}^{L\times L\times C\times C}$ in the neural network. So for example a typical sequence is

$$\mathbf{z} = \rho(\mathbf{f} * \mathbf{x};\alpha) = (\mathrm{diag}_1(I+\alpha) * \mathbf{f}) * \mathbf{x}.$$

This can also be interpreted as a low-rank decomposition of a filter bank $\mathbf{g}$, using $\mathbf{f}$ as a basis:

$$\rho(\mathbf{f} * \mathbf{x};\alpha) = \mathbf{g} * \mathbf{x}, \quad [\mathbf{g}]_{vucd} = \sum_d (1+\alpha_{dc})[\mathbf{f}]_{vucd}.$$

This also means that the adapters can be "fused" with the convolutional layer $\mathbf{f}$ by computing $\mathbf{g}$ explicitly, with no added evaluation cost at test time. However, this operation is difficult to undo, preventing from retargeting the network to another problem, which may be inappropriate in certain applications.

Size of the adapters. In this configuration, the adapter parameters are a fraction $C^2/L^2C^2 = 1/L^2$ of the filter bank parameters. For example, for a 3×3 filter bank, $L = 3$ and the adapters are 9 times smaller.

Relationship to batch normalization. For learning, it is customary to inject batch normalization (BN) layers in architectures, especially of the very deep variety such as ResNet. Figure 2.(a) illustrates a complete residual module, inclusive of BN, ReLU, convolution, and adapter layers for the series configuration.

3.1.2 Parallel residual adapters

While in the previous section adapters are installed in series with existing filter banks $\mathbf{f}$, we propose here an alternative configuration in which adapters are connected in *parallel* instead (fig. 2.b):

$$\mathbf{y} = \mathbf{f} * \mathbf{x} + \mathrm{diag}_1(\alpha) * \mathbf{x} = (\mathbf{f} + \mathrm{diag}_L(\alpha)) * \mathbf{x}.$$

Parallel adapters can also be interpreted as a low-dimensional parametrization of a filter bank $\mathbf{g}$:

$$\rho(\mathbf{f} * \mathbf{x};\alpha) = \mathbf{g} * \mathbf{x},$$
$$[\mathbf{g}]_{vucd} = [\mathbf{f}]_{vucd} + \begin{cases} \alpha_{dc}, & v = u = (L-1)/2+1, \\ 0, & \text{otherwise.} \end{cases}$$

However, differently from series ones, in this case the decomposition is *affine*. The parameters $\mathbf{f}$ can be thought as a universal filter bank which is adjusted additively by modifying the "diagonal" elements of the filters based on α.

Like for the series residual adapters, at test time it is possible to "fuse" the adapters α and filters $\mathbf{f}$ by computing $\mathbf{g}$ explicitly. Differently from that case, however, this additive change can be easily undone to allow to retarget the network to a new task.

Size of the adapters. If $\mathbf{f} \in \mathbb{R}^{L\times L\times C\times C}$, then $\alpha \in \mathbb{R}^{C\times C}$ has the same dimensions as before, so parallel and series adapters have the same number of parameters. It also benefits from the same shrinking to identity property, as setting $\alpha = 0$ recovers $\mathbf{f}$.

Relationship to batch normalization. Just as for series adapters, injection in a neural network such as ResNet [11] requires to clarify the relationship between the adapters and other layers such as BN. This is illustrated in fig. 2.(b). Note that the parallel configuration is *significantly simpler*. For example, compared to the parallel adapters, it saves one BN layer per application.

Further discussion. For both residual and parallel adapters, the filters $\mathbf{g}$ are points in a certain low-dimensional affine subspace parameterized by α. However, for residual adapters the affine subspace is linear (passes through the origin) and its orientation is variable. For parallel adapters the subspace is affine and the orientation is fixed (given by coordinate axis along the "diagonal").

3.2. Network architecture

Having chosen a type of adapter modules, the next question is how they can be best applied to a deep neural network. Adapters may be applied throughout its depth, or more adaptation may be required at the shallower, intermediate, or deeper layers.

To explore these design strategies, we consider as baseline model ResNet [11] in the 26-layer configuration (suitable for medium-sized images). This network (section 3.2) is formed of 3 macro-blocks of convolutional layers, each outputting 64, 128 and 256 feature channels. Each macro-block contains 4 residual blocks each, each of which consists of two convolutional layers using 3×3 filters and a skip connection. The resolution of the data is halved from a macro-block to the next using average pooling. Note that, compared to other architecture such as AlexNet [16] and VGG16 [27], ResNet has a minimal fully-connected layer, meaning that abstraction is likely to increase more uniformly throughout the convolutional part of the network.

In order to experiment with different placements for the adapters, ResNet is broken down into three trunks: early, mid, and late, corresponding to the three macro blocks. Empirically (section 4), we apply the adapters to each stage individually, or to the three stages together. We also experiment with distributing adapters throughout the depth of the model, but skipping one every two, by adapting only the second convolutional layer in each residual block.

Note that the adapter dimensionality is determined by the number of channels in different layers of the architectures. Adapters applied to deeper layers are therefore bigger because the number of feature channels increases with depth. In section 3.4 we show how adapters can be further compressed.

3.3. Regularization: shrinkage *vs* dropout

One advantage of residual adapters is that they revert to the original neural network when α is zero. This is true for series adapter (as noted before) as well as for parallel adapters.

However, there are many alternative forms of regularization that apply to deep networks. For example, BN layers are noisy by construction, and are known to help regularize learning. Another well known method is dropout [28]. In the experiments, shrinkage is compared empirically against dropout, and the latter is shown to be necessary when using a bigger pretrained network. Note that, due to the additive nature of the adapter, dropout in this case is akin to injecting additive noise to the output of the network filters.

3.4. Cross-domain adapter compression

The size of a residual adapter is determined by the number of feature channels of the convolutional layer it is applied to. For deep layers in a neural network, where the number of channels C can be quite large, the number C^2 of adapter parameters can still be non-negligible.

In order to address this issue, we propose to further compress the adapters. A simple approach is to consider a *low rank decomposition* $\alpha = \beta\gamma^\top$ of the adapter matrix $\alpha \in \mathbb{R}^{C\times C}$, where $\beta, \gamma \in \mathbb{R}^{C\times K}$ and $K \ll C$. Such a decomposition can be obtained efficiently using the SVD to minimize the reconstruction residual $\|\alpha - \beta\gamma^\top\|_F$. After replacing α with β, γ, the latter are fine-tuned again on the target task to improve performance further. This scheme uses a fraction $2KC/C^2 = 2K/C$ of the parameters.

Better compression can be obtained by decomposing the adapters jointly for all domains. In order to do so, let $\alpha_1, \dots, \alpha_T \in \mathbb{R}^{C\times C}$ be domain-specific adapters for T tasks. After stacking these matrices, computing the SVD decomposition of the result, and retaining only the top K singular values, one gets:

$$\begin{bmatrix}\alpha_1 \dots \alpha_T\end{bmatrix} = U\Sigma V = \begin{bmatrix}U\\ \vdots\end{bmatrix}\begin{bmatrix}\bar\Sigma & \\ & \ddots\end{bmatrix}\begin{bmatrix}\bar V_1^\top & | & \bar V_T^\top\\ \vdots & & \vdots\end{bmatrix}$$

where $U, \bar V_t \in \mathbb{R}^{C\times K}$, $U^\top U = \sum_t \bar V_t^\top \bar V_t = I \in \mathbb{R}^{K\times K}$ and $\bar\Sigma \in \mathbb{R}^{K\times K}$ is diagonal. Setting $\beta = U\bar\Sigma$ and $\gamma_t = \bar V_t^\top$, we obtain the approximation:

$$\forall t = 1, \dots, T: \quad \alpha_t \approx \beta\gamma_t^\top \tag{1}$$

where $\beta, \gamma_t \in \mathbb{R}^{C\times K}$. In this case, β is shared between domains acting as a common metric and only the factors γ_t are fine-tuned to simplify optimization.

The total number of parameters in $(\beta, \gamma_1, \dots, \gamma_T)$ over the parameters in $(\alpha_1, \dots, \alpha_T)$ for a large number of tasks T is given by

$$\frac{TCK + CK}{TC^2} \to \frac{K}{C}.$$

In practice, we show that good results can be obtained by setting $K = C/2$, therefore with a $2\times$ reduction in the adapter parameters. Joint compression also allows target tasks to communicate and further share parameters (β), in contrast with [23, 25] where adapters are independent. We show that this results in a multi-task regularizer which allows each domain to further benefit from the knowledge of the others.

Finally, note that the parallel adapters can be seen as a parametrization of filters spanning a fixed coordinate subspace. Equation (1) provides a more efficient parametrization of the same subspace, resulting in a higher degree of parameter sharing.

4. Experiments

This section thoroughly assesses the proposed designs, including the topology and position of the residual adapters

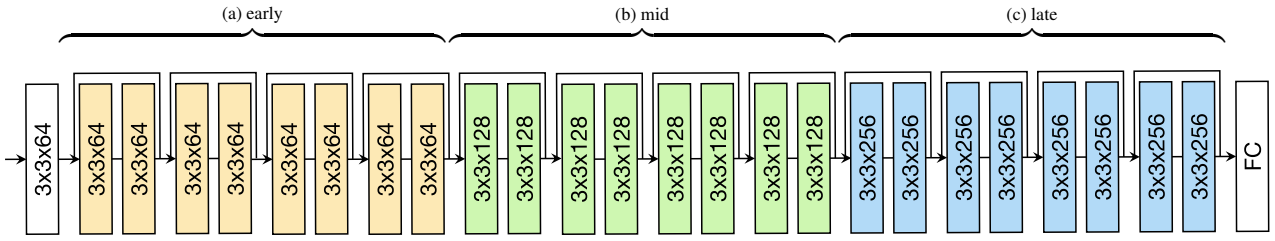


Figure 3: Adapter injection in ResNet-26. Following the scheme of fig. 2, adapters are added to each residual block (here given by a pair of convolutional blocks). We experiment with focusing adaptation on different segments of the network: early, mid, and late. adapters

and the regularization and compression strategies introduced in section 3. We evaluate these decisions quantitatively in multi-domain learning (section 4.1) and transfer learning scenarios (section 4.2). We share our code and models in `https://github.com/srebuffi/residual_adapters`.

4.1. Learning multiple domains

We first investigate the problem of learning multiple, visually-diverse domains using a parameterized neural network family. To this end, we use the recently-introduced Visual Decathlon benchmark [23]. This benchmark consists of 10 different well known datasets, from ImageNet, to OmniGlot (glyphs) and German Traffic Signs. In the benchmark, images are resized to a common resolution of roughly 72 pixels to accelerate evaluation. Furthermore, given the different nature and difficulty of the problems, results are reported both in terms of top-1 accuracy as well as using a "Decathlon score" that rebalances the different problems making them comparable [23] (see table 1).

Following [23], we first train the universal parameters $\mathbf{w}$ of the model using the ImageNet data with a 26 layer ResNet [11] via a stochastic gradient optimization with momentum and finally obtain 60.32% top-1 accuracy on 72 pixel resized validation set. As the first baseline, we fine-tune the pre-trained network for each dataset separately, denote it as "Finetuning" and report its performance in table 1. This standard procedure produces a strong baseline with competitive results, 76.9% mean accuracy and score 3096 (higher than the finetuning score 2500 of [23] due to improvements in the data augmentation process). However it requires ten times more parameter capacity than the base network, as it needs to train one network for each domain.

Parallel vs Series. Next, we compare different topologies for the adapter modules, series and parallel (see section 3.1). For both settings, we first freeze the weights of the pre-trained ImageNet model and learn only the adaptation parameters $\alpha_{1,\cdots,K}$ for each domain. Compared to fine-tuning, adding class-specific adapters lead to a modest increase ($2\times$ vs $10\times$) in total number of parameters. Despite their compactness, both approaches outperform the fine-tuning baseline, achieving similar or better accuracy over all datasets. This indicates that substantial parameter sharing is possible. We also see that the parallel configuration outperforms the series one (by 1 point in average accuracy and 250 decathlon points).

The parallel configuration has the key advantage of being *plug-and-play* whereas the series configuration of [23] requires the *adapters to be included* when ResNet is pre-trained on ImageNet. Indeed, adding them a-posteriori decreases performance substantially (-1.73 point in accuracy on average over 4 datasets). In contrast, parallel adapters can be appended to any pre-trained network, which allows them to be used with off-the-shelf models.

Location of residual adapters. Here we study the optimal placement strategy for the residual adapters throughout the network. As shown in fig. 3, the network is composed of three macro blocks, early, mid and late. In the first experiment, we apply the parallel residual adapters to each macro-block, skip the other two and report the results in table 1 as "Parallel (early,mid,late)". We observe that it is crucial to use the adapters in all the macro blocks as these 3 partial models perform significantly worse than the full model. Still, the adapters are most beneficial in the last block which suggests that, as expected, filters become more specialized and domain specific towards the end of network. We also investigate how the adapters should be distributed within each residual block. In the default setting, the adapters are applied at each of the two convolutional layers (see fig. 2). We evaluate the performance when it is applied to only the second convolutional layer which reduces the number of domain specific parameters by half. We observe that this results in a consistent drop in classification accuracy, suggesting that adapting each convolutional layer is beneficial.

Regularization. One of the challenges of training a single network for multiple tasks is to find an optimal training setting that can work when tasks differ in their difficulty level and number of training images. For instance, we observe in the preliminary experiments that training the adapter modules on the domains with fewer images per class such as Aircraft, DTD, Flowers datasets lead to overfitting on the training set after only a few iterations. To prevent this, we ap-

Model	#par.	ImNet	Airc.	C100	DPed	DTD	GTSR	Flwr	OGlt	SVHN	UCF	mean	S
# images		1.3m	7k	50k	30k	4k	40k	2k	26k	70k	9k		
Finetuning	10×	60.32	60.34	82.12	92.82	55.53	99.42	81.41	89.12	96.55	51.20	76.88	3096
Series Res. adapt.	2×	60.32	61.87	81.22	93.88	57.13	99.27	81.67	89.62	96.57	50.12	77.17	3159
Parallel Res. adapt.	2×	60.32	64.21	81.91	94.73	58.83	99.38	84.68	89.21	96.54	50.94	78.07	3412
Parallel (early)	2×	60.32	50.47	78.58	93.26	58.46	99.00	82.27	87.68	95.39	47.77	75.32	2610
Parallel (mid)	2×	60.32	57.88	79.25	94.24	56.65	98.85	83.43	88.47	95.96	48.98	76.40	2852
Parallel (late)	2×	60.32	61.06	80.58	94.02	57.87	99.19	84.68	89.06	96.30	50.94	77.40	3159
Parallel (half)	1.5×	60.32	61.15	81.24	94.36	58.40	98.85	84.76	88.69	96.19	49.99	77.40	3061
Parallel SVD	1.5×	60.32	66.04	81.86	94.23	57.82	99.24	85.74	89.25	96.62	52.50	78.36	3398
Rebuffi *et al*. [23]	2×	59.23	63.73	81.31	93.30	57.02	97.47	83.43	89.82	96.17	50.28	77.17	2643
Rosenfeld & Tsotsos [25]	2×	57.74	64.11	80.07	91.29	56.54	98.46	86.05	89.67	96.77	49.38	77.01	2851

Table 1: Reports the (top-1) classification accuracy (%) and decathlon overall score (S) of different models on the decathlon tasks [23]. The model size ("#par") is the number of parameters w.r.t. the vanilla network pretrained on ImageNet. Our best models use the parallel adapters and SVD, indicated as "Parallel SVD".

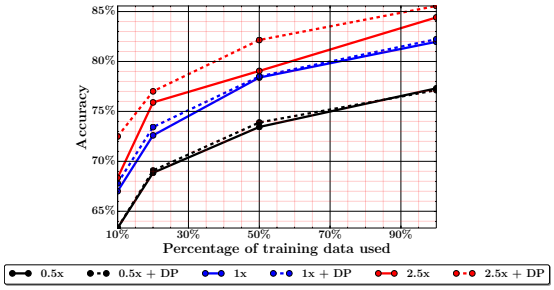
ply a stronger regularization by increasing the weight decay during training time. In particular, we group the datasets in terms of size of their training set as in [23] and assign a different weight decay value for each dataset *i.e*. higher weight decay for smaller datasets (0.002 for Aircraft, DTD, Flowers, 0.0005 for Omniglot, Pedestrian and UCF101 and 0.0001 for CIFAR100, GTSRB and SVHN). This forces a stronger regularization for smaller datasets such that the resulting network has to stay close to the pretrained network. In addition to shrinkage, we also evaluate the effect of another popular regularization strategy, dropout [28]. In this experiment, we apply dropout just before the second parallel adapter in each residual block as done in the standard WideResNet [33]. Figure 4a shows classification accuracies for parallel adapters (with and without dropout) used with pre-trained ResNet models with varying filter widths. We see that dropout needs a wider pretrained network (2.5×) to be effective and that the effect is significant no matter what the size of the training set with a state-of-the-art 85% accuracy using the full training set or an impressive 73% accuracy using only 50 images per class. Thus, dropout enables a better use of the adapters for high capacity pretrained networks even when few images per class are available.

Adapter compression. The size of each residual adapter is dictated by the number of filters in its corresponding convolutional layer. In most of the modern deep network architectures such as AlexNet [15], ResNet [11], the number of convolutional filters is designed to double after each block. This leads to significant increase in the adapter size at the later layers. While this is found to be beneficial for a generic network design, we speculate that the dimensionality of required residual modules can be reduced without any drop in classification performance, as some filter combinations can be useful for more than one domain. Thus, we assume that weights of adapter modules α for different domains are not linearly independent. To test our reasoning, we first take the pre-trained ResNet model and freeze all the weights and only learn domain specific parameters α. As described in section 3.4, we stack these weights, apply the SVD and only retain half of the original dimensionality that yields a further 50% reduction in parameters. Finally, we freeze β weights and fine-tune γ for each domain. We show in table 1 that this approach preserves the performance of the default parallel residual modules while having lower number of parameters (twice as less adapters parameters). As expected, the cross-domain compression acts as a multi-task regularizer and thus prevents from overfitting on small datasets. For example, we can point out the significant effect on Aircraft, VGG-Flowers or UCF 101 with respectively an improvement of 1.8, 1.1 and 1.5 accuracy points with less trainable parameters. For bigger datasets, the performances are preserved while reducing the number of parameters.

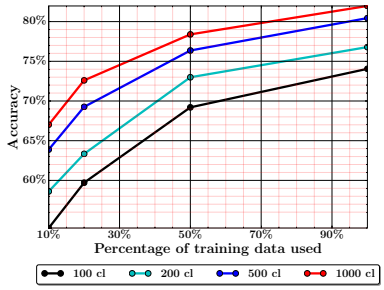
Comparison to the state-of-the-art. We also compare our method to the recent work [23, 25] that report results on the Decathlon dataset and show in table 1 that our method significantly outperforms both. Our approach is directly comparable to [23] as the same base network is used; in particular, "Series Res. adapt." in table 1 is our re-implementation of [23], which outperforms the original in terms of mean accuracy and Decathlon score (3159 vs 2643). Our final result (SVD) achieves a boost of 1.2% in classification accuracy and approximately 750 Decathlon points while employing only half of the additional parameters used in [23]. Similarly, we obtain a remarkable improvement over [25] (1.4% in classification accuracy and approximately 550 Decathlon points) with a significantly more compact architecture.

4.2. Transfer learning results

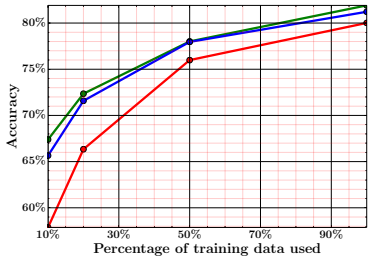
A desired property for a multiple domain learning method is the ability to learn a previously unseen domain



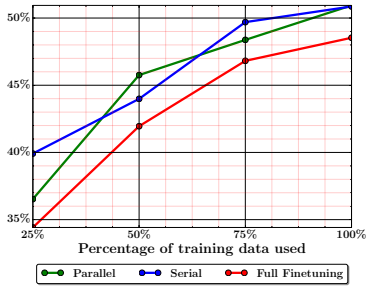
(a) Different pretrained network widths w/ and w/o Dropout



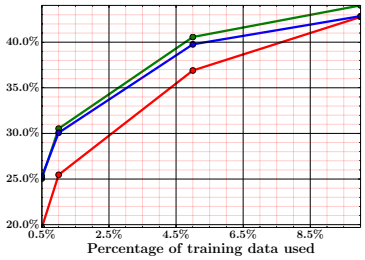
(b) Influence of number of pretrained classes



(c) CIFAR 100



(d) UCF 101



(e) MIT Places

Figure 4: (a,b) analyze on CIFAR100 the influence of pretrained network settings when combined with parallel adapters. (c,d,e) compare the performances of the different methods on 3 datasets in the Transfer Learning setting.

especially when training data is limited. To assess this quantitatively, we take the pretrained ResNet model on ImageNet and finetune by using the residual adapters on three datasets, UCF-101 ("small"), CIFAR-100 ("medium") and MIT Places 205 ("large") with more than 2 million images, all resized to 72 pixels. We train our method with varying the percent of training data and report the results in Figure 4. Both the parallel and series configurations clearly outperform finetuning, not only when there is fewer data available but also for the full size of CIFAR-100 and UCF101. Finetuning only outperforms our method when it is trained on the full training set of the MIT Places and obtains 51.13% compared to our 47.2% validation accuracy. As our method only updates the adapter parameters, finetuning can exploit the high capacity of the network. Hence, the parallel adapters compare very favorably to standard fine-tuning except for extremely large datasets. Series adapters are similar, but with the key difference that the parallel configuration can be applied to an off-the-shelf model a-posteriori. In short, parallel adapters are a simple strategy that can replace and outperform standard fine-tuning in almost every way.

4.3. Influence of the pre-training network

We discussed previously that dropout allows an efficient use of wider networks with the parallel adapters and fig. 4a shows that increasing the pretrained network size (from $0.5\times$ to $2.5\times$) helps even when amount of training data is limited. Here we also study how the pretrained network affects the performances of transfer learning when it is trained on a training set sampled from a smaller number of categories. We observe in fig. 4b that the classification accuracies on the target task decrease steadily if we pretrain the same network with less ImageNet classes. Thus a good network for transfer learning with adapters should be saturated with as many classes as possible during pretraining.

5. Conclusion

In this paper we have shown that it is possible to build universal parametric families of networks that can share parameters very efficiently among multiple domains. We have proposed and evaluated several design strategies for the design of such architectures. The best results were obtained by using parallel residual adapter modules distributed throughout a neural network architecture and further jointly rank-compressed. The resulting network families are very compact, resulting in substantial savings in terms of model storage, exchange, update, and transmission. They also significantly outperform recent alternatives in benchmarks such as Visual Decathlon. We have also showed that parallel adapter can replace traditional fine-tuning techniques, achieving far superior performance that those in almost all cases with no additional constraint or limitation.

Acknowledgments: This work acknowledges the support of Mathworks/DTA DFR02620, EPSRC SeeBiByte and ERC 677195-IDIU.

References

[1] L. Bertinetto, J. F. Henriques, J. Valmadre, P. Torr, and A. Vedaldi. Learning feed-forward one-shot learners. In *Proc. NIPS*, pages 523–531, 2016.

[2] H. Bilen and A. Vedaldi. Integrated perception with recurrent multi-task neural networks. In *Proc. NIPS*, 2016.

[3] H. Bilen and A. Vedaldi. Universal representations: The missing link between faces, text, planktons, and cat breeds. *arXiv preprint arXiv:1701.07275*, 2017.

[4] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. Domain separation networks. In *Proc. NIPS*, pages 343–351, 2016.

[5] R. Caruana. Multitask learning. *Machine Learning*, 28, 1997.

[6] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *icml*, pages 160–167. ACM, 2008.

[7] G. E. Dahl, N. Jaitly, and R. Salakhutdinov. Multi-task neural networks for qsar predictions. *arXiv preprint arXiv:1406.1231*, 2014.

[8] R. M. French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[9] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. *Proc. ICML*, 2015.

[10] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. *arXiv preprint arXiv:1703.06870*, 2017.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *Proc. ECCV*, pages 630–645. Springer, 2016.

[12] J. T. Huang, J. Li, D. Yu, L. Deng, and Y. Gong. Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *ICASSP*, pages 7304–7308, 2013.

[13] J. Kirkpatrick, E. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *National Academy of Sciences*, 2017.

[14] I. Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *Proc. CVPR*, 2017.

[15] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Technical report, University of Toronto*, 2009.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. NIPS*, pages 1097–1105, 2012.

[17] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales. Learning to generalize: Meta-learning for domain generalization. *arXiv preprint arXiv:1710.03463*, 2017.

[18] Z. Li and D. Hoiem. Learning without forgetting. In *Proc. ECCV*, pages 614–629, 2016.

[19] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Unsupervised Domain Adaptation with Residual Transfer Networks. In *Proc. NIPS*, pages 136–144, 2016.

[20] E. Meyerson and R. Miikkulainen. Beyond shared hierarchies: Deep multitask learning through soft layer ordering. *arXiv preprint arXiv:1711.00108*, 2017.

[21] T. Mitchell. Never-ending learning. Technical report, DTIC Document, 2010.

[22] A. Rannen, R. Aljundi, M. Blaschko, and T. Tuytelaars. Encoder based lifelong learning. In *Proc. ICCV*, pages 1320–1328, 2017.

[23] S. Rebuffi, H. Bilen, and A. Vedaldi. Learning multiple visual domains with residual adapters. In *Proc. NIPS*, 2017.

[24] S. A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *Proc. CVPR*, 2017.

[25] A. Rosenfeld and J. K. Tsotsos. Incremental learning through deep adaptation. *arXiv preprint arXiv:1705.04228*, 2017.

[26] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[29] A. V. Terekhov, G. Montone, and J. K. ORegan. Knowledge transfer in deep block-modular neural networks. In *Biomimetic and Biohybrid Systems*, pages 268–279, 2015.

[30] S. Thrun. Lifelong learning algorithms. In *Learning to learn*, pages 181–209. Springer, 1998.

[31] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Simultaneous deep transfer across domains and tasks. In *Proc. CVPR*, pages 4068–4076, 2015.

[32] Y. Yang and T. Hospedales. Deep Multi-task Representation Learning: A Tensor Factorisation Approach. In *ICLR*, 2016.

[33] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[34] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Robust visual tracking via structured multi-task sparse learning. *IJCV*, 101(2):367–383, 2013.

[35] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Facial landmark detection by deep multi-task learning. In *Proc. ECCV*, 2014.

Chapter 5

Semi-supervised learning with scarce annotations

This work was presented at the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2020. This work was previously submitted and rejected at ICCV 2019.

This paper tackles the problem of semi-supervised classification when very few labels per class are available. First, we propose to pre-train the network with self-supervision and freeze the early layers to respectively initialise the network without using any label and constrain the data representation. When using such a pre-training, existing methods can be competitive with as few as 10 labels per class. Second, we design a self-labeling approach that leverages the self-supervision pre-training. Our method combines an alternating optimisation scheme and the random subsampling of the unlabelled set to avoid overfitting the labelled data and the pseudo-labels. Finally, we show that our algorithm is also suited for other initialisations than self-supervision as it clearly outperforms existing methods in semi-supervised transfer learning from a pre-trained network.

Semi-Supervised Learning with Scarce Annotations

Sylvestre-Alvise Rebuffi* Sebastien Ehrhardt* Kai Han*
Andrea Vedaldi Andrew Zisserman
Visual Geometry Group, University of Oxford
{srebuffi,hyenal,khan,vedaldi,az}@robots.ox.ac.uk

Abstract

While semi-supervised learning (SSL) algorithms provide an efficient way to make use of both labelled and unlabelled data, they generally struggle when the number of annotated samples is very small. In this work, we consider the problem of SSL multi-class classification with very few labelled instances. We introduce two key ideas. The first is a simple but effective one: we leverage the power of transfer learning among different tasks and self-supervision to initialize a good representation of the data without making use of any label. The second idea is a new algorithm for SSL that can exploit well such a pre-trained representation. The algorithm works by alternating two phases, one fitting the labelled points and one fitting the unlabelled ones, with carefully-controlled information flow between them. The benefits are greatly reducing overfitting of the labelled data and avoiding issue with balancing labelled and unlabelled losses during training. We show empirically that this method can successfully train competitive models with as few as 10 labelled data points per class. More in general, we show that the idea of bootstrapping features using self-supervised learning always improves SSL on standard benchmarks. We show that our algorithm works increasingly well compared to other methods when refining from other tasks or datasets.

1. Introduction

The success of Deep Learning (DL) in computer vision comes at the cost of collecting large quantities of labelled data. In many applications, data collection is increasingly inexpensive, but data annotation still involves manual and thus expensive labor. Semi-supervised learning (SSL) can significantly reduce the cost of learning new models by using large datasets of which only a small proportion comes with manual labels.

When only some data samples are annotated, one can exploit the structure in the data to infer, implicitly or explicitly, the missing annotations. For example, the class of an image generally does not change if the camera is slightly shifted while maintaining the focus on the same point. The resulting viewpoint change can be approximated by a deformation or warp of the image. A change in illumination, which also generally leaves the image class unchanged, can instead be approximated by a linear transformation of the image range. Hence, an effective way of propagating labels is to enforce consistency in prediction: if one of these perturbations is applied to the image, then the prediction should remain the same.

A simple way to incorporate such invariance while training a deep neural network is to consider image augmentations such as rotations and flips and other perturbations such as Dropout [34], which is injected directly at the level of the features. Such perturbations are used as a form of regularization for supervised learning to avoid overfitting. The idea is that a perturbed image should maintain the same ground-truth label as the original image. In the SSL setting, where most labels are unknown, augmentation can be used to encourage the predictions of unlabelled data instances to be 'consistent', i.e. to remain stable across small variations of the input image, regardless of which specific label the data point takes. If data samples are dense enough, stability to local perturbations may be sufficient to propagate the known labels to the rest of the dataset. Similar ideas are used in many approaches to semi-supervised classification, including the recent works of [16, 21, 35]. Still, performance degrades quickly with a diminishing number of labelled instances per class [24].

We can explain this performance drop as follows. Perturbations can bridge some of the gaps between labelled data points, but the effect is ultimately only local. The ability to transport information across points that are farther apart (since there are fewer of them) depends mainly on the smoothness of the data representation. For example, the Euclidean distance is essentially meaningless in image space, so that transferring labels between images that are close as vectors of pixels is very ineffective. On the other hand, if

*Authors contributed equally

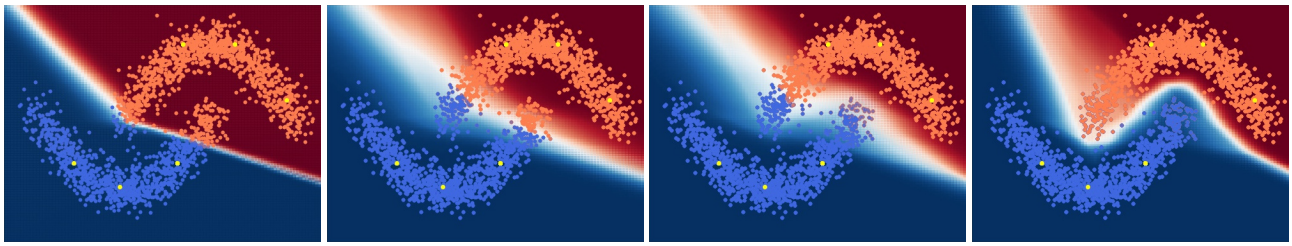


Figure 1. **Consistency loss with the ramp-up strategy.** The Π-model [16] is applied to the 'two moons' problem of [21] and various steps of the optimization are shown. At first, the learned network creates a hard border (first panel) based on the labelled anchor points (in yellow). Then the border is progressively stretched as the consistency loss is enforced more strongly (second and third panels). Overall, the Π-model works well to label connected components of the data space. (Darker regions are more confident; best viewed in color.)

images are first encoded via a representation function in a low dimensional space whose smoothness intrinsically captures some of the relevant invariances, then augmentation can be much more effective.

Unfortunately, simply training a representation such as a deep neural network on a semi-supervised dataset is unlikely to solve the problem. In fact, it is difficult to learn effectively from a label-deficient dataset with a small number of data points with known labels and a very large number of data points with unknown labels. In general, labelled points would work as "anchors": since their label is known with certainty, the neural network is forced to fit those points with confidence, so that the information extracted from them can be reliably propagated. However, when the ratio of labelled vs unlabelled points is very unbalanced, the most likely result is that the neural network would overfit the few labelled points, which would then cease to have an influence on the unlabelled points, thus preventing effective propagation.

We make several contributions to address such shortcomings. The first contribution is a simple but effective one: we propose to use self-supervision to bootstrap a good representation of the data before running our SSL algorithm. Self-supervision defines a pretext-task where a full set of labels can be generated. While these labels are unrelated to the task at hand, we exploit the ability of deep network to *transfer effectively between tasks* and re-use the pre-trained feature to initialize SSL. Note that the data are the same, it is only the task which is being transferred.

The other contribution is a new SSL algorithm that can leverage the bootstrapped representation well. The core idea is that, instead of fitting labelled and unlabelled data points simultaneously, we alternate between two phases of optimization. The information flow between the two phases is carefully controlled to minimize the risk of overfitting. Namely, during *phase one* we fit the labelled data, but only change the final layer of the representation. Then, the resulting classifier is used to generate pseudo-labels for *phase two*, where the unlabelled data are fitted. At this point, the representation is reset to initial state (from the pre-text task) and fine-tuned using the pseudo-labels only. Finally, the fine-tuned representation is passed back to *phase one* to retrain the classifier and obtain new-pseudo labels. In this way, the few available data points are only used to fit a small number of model parameters (a classifier layer only), whereas the large unlabelled dataset is used to fine-tune the representation. We show empirically that this greatly reduces overfitting.

We add other technical contributions to this basic scheme. The most significant in terms of final performance is inspired by cross-validation, Tri-Training [41] and weakly supervised localization [9], and amounts to split our dataset in different subsets, considering only part of the unlabelled data at each cycle of the alternate optimization above.

Empirically, we show that our method achieve close to *state-of-the-art* results in a large number of benchmark cases we test. Furthermore, in addition to showing that our method is effective as a standalone algorithm, we also show two other potential utilities of it. First it is able to refine results obtained from others SSL methods and second it works best among other SSL algorithm when transferring information from one dataset to another. Finally, further studies on architectures and self-supervised task were carried out to assess their importance in improving SSL accuracy.

2. Related work

There exist a vast number of classic works on SSL [4] across many disciplines [7, 31, 36]. In the classification context, one common approach is to optimise the conventional cross-entropy loss on the labelled data together with a regularisation term that propagates information to the unlabelled data. One popular form of 'regularisation' is the consistency of predictions to perturbation [1, 29]. [26] introduced a ladder network that minimises the reconstruction loss between the network outputs from a given sample and its stochastically-perturbed counterpart. [16] simplified the ladder network into two temporal methods: Π-Model and Temporal Ensembling. Both encourage the output of the network for each unlabelled instance to be as similar

as possible between different training epochs, by penalising the inconsistency between current network predictions and past predictions. More recently, the Mean Teacher [35] extended these methods by ensembling over the parameter space: instead of recalling past predictions they run an exponential moving average over the weights of the network to build a Teacher model. The Teacher network is then used to enforce consistency in predictions with the main model, called the Student. Mean Teacher was extended in [19] by a certainty-driven consistency loss that focuses on the unlabelled training samples whose labels are the most uncertain. Similarly, [21] considered instead adversarial perturbations that maximise the change in model prediction. [12] also used this idea together with gradient alignment between labelled and unlabelled data to further improve classification results.

SSL approaches alternative to perturbations include GANs [30, 33]. [6] introduced a memory-assisted deep neural network (MA-DNN) which uses an external memory module to maintain the category prototypes and provide guidance for learning with unlabelled data. Alternatively, [20] uses a label graph to refine the Mean Teacher's model predictions. When using graphs, label propagation [13, 18, 42] is also commonly used. [17] proposed to iteratively assign pseudo-labels for unlabelled data and pick the high-confidence assignments as training data in the subsequent learning steps, combined with the entropy loss to further regularise training. The idea of pseudo labelling is also used alongside with interpolation in [37]. Tri-training [41] and its deep network extension Tri-Net[5] use co-training [3] to generate three classifiers from three different portions of the data and let them label the unlabelled data. [24] shows that consistency is generally the most resilient method as the number of labelled instances decreases, but all such methods suffer significantly when this number is decreased past a certain point.

Transfer learning is not often used in SSL. [8] and [24] used a pre-trained network to fine-tune it on a restricted part of the labelled set. Closer to our approach, [40] also combined both SSL algorithms and transfer learning. However, their approach used a pre-trained network from ImageNet[28] to train SSL methods on non-standard datasets for SSL. Our work extends their study to transfer learning from self-supervision and propose a method that surpasses current SSL technique when transferring representation across dataset.

In addition, we notice very recent concurrent works [2, 32] showing promising results. However, none of them focuses on SSL with scarce annotations.

3. Method

We introduce a SSL algorithm that can work effectively when there are only very few annotated data points, including avoiding supervised pre-training on large labelled datasets. The algorithm starts by a *preparation phase* (section 3.1, line 1 in algorithm 1) that initializes the weights of the model using self-supervision, followed by alternating between two phases (section 3.2, lines 5-9 in algorithm 1), where in *phase one* a subset of the weights are fitted to the few available labels and in *phase two* the whole model is re-trained from scratch using pseudo-labels on all the data.

Algorithm 1 Proposed Alternative Optimisation Algorithm

1: **Preparation phase:**
2: Train a self-supervised method on the whole dataset and freeze the first blocks' weights. Replace the last layer with one dimensioned for the classification task. The trainable weights now form a network N_t
3: **Main Loop:**
4: **for** $i \in \{1, \ldots, N\}$ **do**
5: **Supervised-training**: Fine-tune N_t classification layer on the labelled subset with cross-entropy loss $\mathcal{L}_{labelled}$.
6: **Labels assignement**: Use N_t to assign a label y_i to each unlabelled sample x_i.
7: **Dataset split**: Create a training set T_i from a random split of the unlabelled data.
8: **Restart**: Reassign N_t to the weights extracted from the preparation phase.
9: **Unsupervised-training**: Train N_t on the unlabelled metaset T_i with consistency and pseudo labelling loss: $\mathcal{L}_{unlabelled} = 0.5\mathcal{L}_{temp} + 0.5\mathcal{L}_{pseudo}$.
10: **end for**

3.1. Transfer learning and self-supervision

SSL algorithms risk overfitting the available labelled data points, especially when there is ony a small number of them. Data augmentation and other forms of regularisation can help to some extent, but their effectiveness is limited. A more effective solution is to pre-train the model on a much larger dataset, adding to the information contained in the few labelled data points. Then transfer learning can be used to fine-tune the pre-trained model using the labelled data.

The most common form of pre-training is to use a large, external labelled dataset such as ImageNet. In this work, however, we focus primarily on the case in which labels are scarce, including in the pre-training phase. Furthermore, we wish to avoid the use of external data altogether.

In order to do so, instead of using an external labelled dataset for pre-training, we propose to use instead *self-supervision*. Self-supervision uses a pretext task defined on the available data, both labelled and unlabelled, to bootstrap the model. The advantage is that self-supervision does not require any label nor external data.

Empirically, we show that self-supervised pre-

initialization is very good. In particular, excellent performance can be obtained by freezing most of the model parameters to their pre-initialized values and use our SSL algorithm to fine-tune only a small subset of them.

3.2. Alternate optimisation

Given a pre-initialized model, most SSL algorithms fine-tune it using mini-batches containing a mix of labelled and unlabelled data. They use a sum of two losses, the first one enforcing the correct classification of the labelled samples and the second one enforcing a form of prediction consistency for the unlabelled samples. This consistency term usually captures the fact that neighbor data points are likely to have the same label. Often, a ramp-up is used for the consistency loss [16, 35], so that in the first iterations of training the model focuses on fitting the labelled data.

The effect of this ramp-up can be visualized using the 'two moons' of [21], a simple 2D toy clustering problem. In fig. 1, we apply the Π-model algorithm of [16] to this dataset. We see that, before the consistency loss is applied, the network learns a simple boundary that fits the labelled anchor points with tightly. Then, when the consistency loss is applied, the network stretches the boundary to satisfy the consistency criterion. This causes the classifier to become less certain near the decision boundary, but more accurate overall.

A drawback of this approach is that, since both losses are optimized jointly, they must be carefully balanced by the choice of appropriate hyper-parameters. This is particularly true when there are very few label samples, which causes a large imbalance between the two loss terms. Furthermore, since mini batches are generally evenly split between labelled and unlabelled samples, the few labelled samples will be seen very frequently by the optimizer, which will thus overfit them.

Inspired by incremental learning techniques that try to avoid overfitting on past exemplars while learning new classes [27], we propose to disentangle the two losses by alternating the optimisation of labelled and unlabelled data points while maintaining soft constraints between them. This method reduces the number of hyper-parameters and avoids strong overfitting of the labelled data while still extracting useful knowledge from the labelled set.

In practice, after the preparatory phase of section 3.2 (l.1 in algorithm 1), each task is learnt separately in an alternate fashion with a regularisation term that works as a soft constraint borrowing information from the other task.

***Phase one*: fitting the labelled data.** In the first part of our training, we optimise the cross-entropy loss on the labelled set $\mathcal{L}_{labelled}$ and train the model for a few epochs. In this part we only fine-tune the final classification layer of the network (l.5 in algorithm 1). This way the labelled samples benefit the most from the feature learnt during optimisation on the unlabelled set while not modifying the intermediate representation. We then use this trained network to assign soft or hard pseudo-labels to the unlabelled data. These pseudo-labels are used in the next phase to fit the unlabelled data (l.6 in algorithm 1).

***Phase two*: fitting the unlabelled data.** In the second phase we reset the model with the parameters learnt during the preparatory phase of section 3.2 (see l.8 in algorithm 1). Then, we fine-tune the whole architecture on the unlabelled set using a loss $\mathcal{L}_{unlabelled}$ that is a weighted average of a term $\mathcal{L}_{pseudo}$ fitting the pseudo-labels estimated in the first phase, and a second temporal consistency term $\mathcal{L}_{temp}$, borrowed from the Π-model of [16]. The idea of the second term is to match the probability distribution p_i^{t-1} obtained from the network for a sample x_i at epoch $t-1$ with the probability p_i^t assigned at iteration t to the same sample. We use the KL-divergence between p_i^{t-1} and p_i^t during *phase two*. (l.9 in algorithm 1). The loss $\mathcal{L}_{pseudo}$, besides being a soft constraint tying *phase two* to *phase one*, can also be viewed as another consistency condition [40] that also helps reduce the entropy on the final prediction.

Information flow. Note that the model is reinitialised every time *phase two* is entered. This is done to avoid getting stuck in local minima. Hence, the information that flows from *phase one* to *phase two* is only the pseudo-labels on the unlabelled data, whereas the information that flows from *phase two* back to *phase one* are the weights of the network fine-tuned on the unlabelled data.

While these choices may seem ad-hoc, they constitute the primary reason why our algorithm is able to work well in a low-labelled-data regime by avoiding overfitting. Thanks to this technique, the algorithm focuses on fitting one loss at a time while still allowing information to be exchanged between the two subtasks. We show in section 4 this approach robustly improve the quality of the overall classifier at every step.

3.3. Dataset split

One potential pitfall of alternating optimisation is the drift that may be induced by training for a long time on the unlabelled data. Inspired by cross-validation and co-training [3], we also suggest to split the dataset into several parts. However, differently from [3] we do not train different models from each split. Instead, at the beginning of each training cycle we randomly choose two-thirds of the unlabelled training data to learn from (l.7 in algorithm 1). The rest of the unlabelled data is held out to reduce the risk of overfitting to it in the current training cycle. This way, the held-out samples are more likely to have their labels swapped during pseudo-label generation in the next cy-

cle. Roughly two-thirds of this "fresher" held-out data will be used to fit the model in the next cycle. The benefits of this reshuffling procedure diminish as training nears completion, so all the data is used in the final few phases of training. In section 4 we demonstrate empirically the benefits of this approach.

4. Experiments

Datasets. Following [6] we mostly use standard image classification tasks suitably modified to the semi-supervised context. Each dataset is randomly split in two sets: one small set containing images with their corresponding labels evenly divided among classes and another set with the rest of the image without annotations. For each dataset we evaluated our method on the publicly available test set. We used the same datasets as [6]:

SVHN [23]. A Street View House Numbers dataset including 10 classes (0-9) of coloured digit images from Google Street View. The classification task is to recognise the central digit of each image. We use the format-2 version that provides cropped images sized at 32×32, and the standard 73,257/26,032 training/test data split. We centered and scaled all images RGB channels based on the extracted dataset statistics and augmented the data with random cropping.

CIFAR-10 [15]. A natural images dataset containing 50,000 training and 10,000 test image samples from 10 object classes. Images have a 32×32 resolution and are evenly divided among classes. We again pre-processed the data with centering and scaling and augmented the data with random cropping and flipping.

CIFAR-100 [15]. A dataset (with the same image size as CIFAR-10) containing 50,000/10,000 training/test images from 100 more fine-grained classes. We used the same preprocessing and data augmentation as CIFAR-10.

Baselines. We used Mean-Teacher [35] as the main baseline to experiment alongside with our proposed method. Additionally we also used MA-DNN [6] on a more restricted set of experiments. For both we used publicly available code provided by the authors[1].

Implementation details. Unless stated otherwise, we used the ResNet-18 [11] model with then average pooling and skip connections. We chose RotNet [8] as the self-supervision method as it is easy to implement and integrate in any pipeline. For all networks we trained this proxy task on 200 epochs with a step-wise decaying schedule. A more thorough study is done in section 4.6. All networks are trained using Nesterov accelerated gradient method [22]. We use the same hyper-parameters when training with and without transfer learning, which have every time been tuned on the specific task.

[1] https://github.com/CuriousAI/mean-teacher; https://github.com/yanbeic/semi-memory

4.1. Impact of self-supervision

We first compare Mean-Teacher and our algorithm on standard classification benchmarks. For every dataset we trained the network with 10/25/50/100 labels per class, with an extra experiment of 400 labels per class on CIFAR-10 as is common practice [6, 16, 35]. In every experiment we keep the same split of the data for each method. In the default case, models are trained from scratch. For every method we also used transfer learning from self-supervision, with only the last two blocks and the last classification layer used for semi-supervised training (Legends containing 'with T.L.'). Results are shown in fig. 2.

This experiment clearly demonstrates the importance of transfer learning as the number of labelled data per class decreases. For instance, on CIFAR-10, transfer learning from self-supervision used with semi-supervision makes an improvement of over 50 points in classification error for both our method and Mean Teacher. On SVHN and CIFAR-10 the performance from 100 labels per class to 10 labels per class decreases by less than 5 points in error. Performances on CIFAR-100 dropped by a larger amount when having fewer labels due to the higher complexity of the dataset, although the performance deteriorate noticeably more slowly when using self-supervision pre-training. Overall this means that self-supervision enables networks to train with competitive results in the extremely low data setting where standard SSL methods fail. We also emphasise that we only use the dataset provided and did not use any regularisation method and very limited data augmentation. This is particularly crucial and demonstrates that a carefully chosen self-supervision method can be sufficient to extract very good features which are sufficient to train good models with few labels.

For every dataset we see in fig. 2 that our method and Mean-Teacher obtain similar performance when they both use transfer from self-supervised networks. However our method needs a more careful initialisation as its performance dropped for models trained from scratch. This can easily be understood, as fine-tuning on the labelled set would be more efficient with either an increased number of labelled data or better intermediate representation. In the rest of the paper we call a 'good' representation of an image, a representation from which it is easier to discriminate different classes.

In addition, we show that self-supervision can also improve the state-of-the-art performance of a more recent SSL method. To this end we use MA-DNN [6] with the 10-layers model proposed in their paper. Again we used RotNet [8] as the self-supervision method and only fine-tune the last

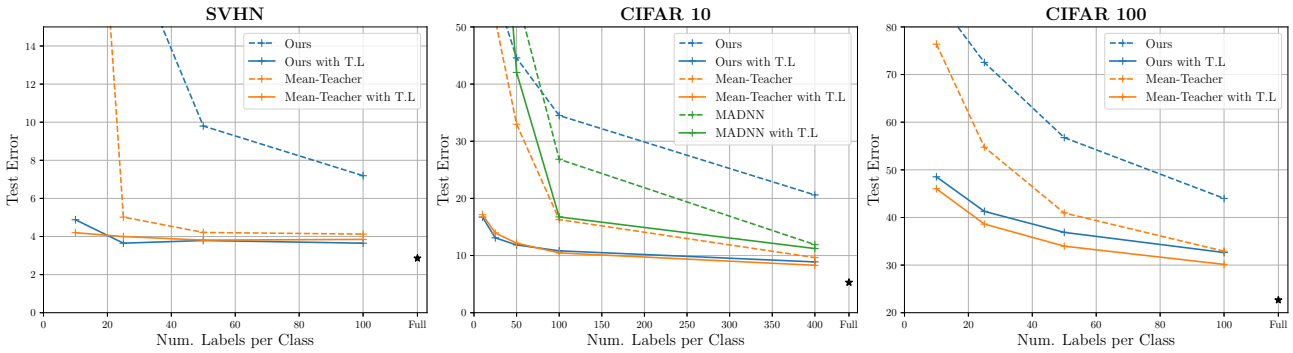


Figure 2. **Impact of Tranfer learning for SSL classification benchmark.** For each dataset we vary the number of labelled data per class in $10, 25, 50, 100$ with an additional experiment on CIFAR-10 with 400 labels per class. We note that for every method, models benefit from self-supervision. The 'with T.L' indicates a model pretrained with RotNet[8]. In addition we can see that our method is getting competitive results with Mean Teacher. The star point '*' denotes our network trained with supervision only on the full training set. The shown results are the average of 10 runs per setting.

block and linear layer on the SSL task. We see in fig. 2 that again, transfer from self-supervision consistently achieves the best performance, improving their *state-of-the-art* results on CIFAR-10 from 11.9% error to 11.1% using 400 labels per class. With the hyper-parameters given in their paper, we demonstrate again that under equivalent settings, self-supervision is beneficial to SSL. Finally we found that MA-DNN was more sensitive to the number of labels per class. This means that building a reliable template of every class requires more annotations, while we observed that consistency based methods are seemingly showing better robustness overall.

4.2. Ablation study

To assess the impact of every component of our method we also conducted an ablation study on the CIFAR-10 SSL task with 10 labels per class.

We first measure the gain of using random splitting of the unlabelled set through training cycles as opposed to using the whole set. We observe in fig. 3 that random dataset splitting allows much faster progress in the course of training. When only using pseudo-labelling on the unlabelled set and no temporal consistency, this technique also leads to a better optimum.

Then we analyse the effect of the consistency in the training cycle. The dashed curves correspond to cases where the temporal consistency is not used. It means that, when training on the unlabeled set, we only optimise the cross-entropy on the pseudo-labels previously assigned during the training phase on the labelled set. Interestingly, we also note that the performance is still increasing over cycles even without an explicit consistency loss. In fact, this corroborates the statement of [40], that the assigned pseudo-labels act as an implicit consistency across samples (instead of the explicit consistency across augmented versions of the same sample) which improves the representation for the next training phase on the labelled set.

Finally we can observe for the different settings that the performance is steadily increasing over cycles. The training indeed does not stall after a few cycles but instead keeps improving the feature representation over the course of alternating optimisation cycles. Alternating optimisation therefore refines the results every cycle with each optimisation helping the next one to reach a better optimum. In addition, we conduct another ablation experiment skipping re-initialization (line 8 in algorithm 1). Compared to the two-phase algorithm with re-initialization, the accuracy drops by 12 points on CIFAR-10 and even 39 points on CIFAR-100. When re-initialization is turned on, $10\times$ more samples switch labels every assignment cycle, thus avoiding local minima. To the best of our knowledge, other existing SSL

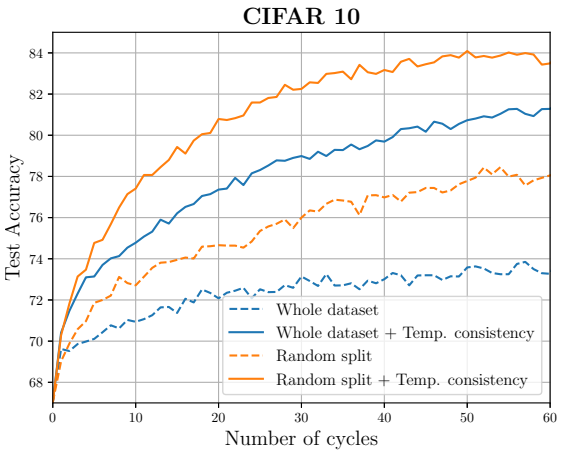


Figure 3. **Ablation study.** Starting from a pre-trained model, our alternating optimisation method makes steady progress over cycles. Dataset random splitting is also effectively speeding-up training. Default case uses $\mathcal{L}_{unlabelled} = \mathcal{L}_{pseudo}$. 'Temp. consistency' indicates that we use both $\mathcal{L}_{temp}$ and $\mathcal{L}_{pseudo}$.

	CIFAR-10	CIFAR-100	
Labels per class	10	10	25
MT	32.4	23.6	45.7
MT + Self-Sup	86.4	53.7	60.7
MT + Self-Sup + Refinement	**87.7**	**58.3**	**63.9**

Table 1. **Our method applied as a refinement of Mean Teacher ('MT').** For different numbers of labels per class our method always improve the testing accuracy by a big margin.

methods do not have any specific mechanism against representation forgetting when doing transfer learning.

4.3. A refining SSL algorithm

In fig. 2 we note that our alternating training method is particularly efficient when using a good representation rather than starting from scratch. Hence we hypothesise that our method could make an even bigger difference when starting from a better network representation. One way of doing it is to build our method on top of others in order to refine their results. To validate this hypothesis, we apply our method as a refinement on CIFAR-10 and CIFAR-100 datasets for networks pre-trained with Mean-Teacher [35]. It should be noted that, as the starting features are already competitive, we do not use the dataset random splitting in our training procedure and we choose soft labels assignment in l.6 of algorithm 1.

In table 1, we see that in every case our method used as refinement consistently improves the test accuracy. The synergy of alternating training is enabling the network to gain as much as 4.6 points in test accuracy on CIFAR-100 and 1.3 points on CIFAR-10. The refinement technique combined with self supervision allows us on CIFAR-10 to almost match previous *state-of-the-art* results with methods trained with 400 labels per class (11.9% reported in [6]) whereas we only used 10 labels. These results confirm our previous hypothesis and motivate us to see the impact of increasingly better representations in the next section.

4.4. Transfer learning from different tasks

So far we have only used the dataset at hand to extract our intermediate representation with self-supervised learning. In this section we use a different type of transfer learning used in [40]: we learn our representation on a different classification dataset. To this end, we use networks pretrained on ImageNet [28]. As is done in the other sections, we then fix the first two blocks of our ResNet-18 network and train the rest of the weights on the hardest semi-supervised learning task of 10 labels per class using either Mean Teacher or our alternating optimisation method (see section 3). We experimented with CIFAR-10 and CIFAR-100 with pretraining on a downsampled 32×32 version of ImageNet, and also MIT Indoor-67 [25], and 'Places10' where we only kept 10 classes from Places [39] using a pretraining on the standard version of ImageNet. Results are shown in table 2 where we perform 10 runs per setting. Fine-tuning on the full set is given as a reference.

	C-10	C-100	P-10	MIT-67
Fully Sup.	96.6	82.5	83.3	75.5
Labelled Set	82.5±1.0	62.8±0.6	66.4±1.7	55.1±0.8
M.T.[35]	88.0±1.5	63.3±0.7	72.8±1.2	58.8±1.3
Ours	**93.5±0.4**	**69.2±0.4**	**75.8±1.7**	**63.0±0.7**

Table 2. **Transfer Learning: Test accuracy on the target task.** Transfer is from training on ImageNet. We compare our method with Mean Teacher (M.T.) and supervision on the labelled set only ('Labelled set') for 10 labelled sample per class. Our method consistently outperforms both method on this task and achieves results nearly as good as the one obtained by training on the full labelled set.

We note that our method consistently outperforms Mean Teacher for such scenarios. This result corroborates our assumption in the previous section that our method performs increasingly well with better representation. In fact, as also noted in [24], since ImageNet has redundant classes with the datasets, we can consider the representation learnt from ImageNet as an upper bound. Thus it is not entirely surprising to see our method obtaining nearly as good results as if it were using the complete set with 93.12% accuracy compared to 96.56% when fine-tuning on the full labelled set on CIFAR-10. As our method aims to disentangle the two losses by using distillation to preserve the effect of the very reduced labelled set, its effect is magnified when using a representation as good as the one learnt on ImageNet. In contrast, techniques like Mean Teacher overfit on a small labelled set, which in turns will lead to the degradation of the representation coming from ImageNet.

4.5. Evaluating representation quality with SSL

In the context of SSL for images classification, practitioners usually use shallow networks. If some works [35, 37] also experimented with deeper and wider architectures, then those were generally applied when the number of labels at hand was large enough (4,000 on CIFAR-10, ∼100,000 on ImageNet). So far in this work, we have shown that it is possible to train a deeper architecture by using transfer-learning from self-supervision. In practice, what matters in our context is the quality of the extracted intermediate representation that we will train our model from.

In this section we experiment with two other architectures and use SSL to measure the quality of their representation. As is done in [14], we compare with one fully-convolutional model and a RevNet[10] model. We chose the fully convolutional 10-layers/3-blocks model from [16] which we named 'TempEns'. We provided our own RevNet implementation named 'RevNet-18' aimed to match the ResNet-18 we used in this work. More specifically, the

N	TempEns[16]	RevNet-18[10]	ResNet-18[11]
10	18.4±1.9	20.0±2.4	**16.4±2.6**
50	16.3±0.9	13.7±0.7	**11.9±0.4**
100	14.8±0.6	12.2±0.5	**10.8±0.4**
400	12.8±0.2	10.5±0.4	**8.9±0.2**
Full	6.7	5.8	5.3

Table 3. **Model variation.** Test error on CIFAR-10 with different number of labels per class 'N'. In all cases networks, were pre-trained with RotNet [8]. First two block are then frozen while the rest of the network is fine-tuned on the classification task. 'Full' indicates that we used the full training set with a fully supervised method. All methods uses the same hyper-parameters.

RevNet-18 has four main blocks and uses the same type of downsampling mechanism as our ResNet-18. We pre-trained all models with RotNet and trained our SSL method on CIFAR-10 with different numbers of annotated instances. For all architectures the first two blocks of the networks are fixed during SSL training, as we found it was giving best results for every architecture. We report results in table 3.

We found that the ResNet-18 has better performance over all methods. TempEns surprisingly remains very competitive for all cases with less weights trained than both methods, with good robustness to extreme cases. While RevNet is almost always able to reach ResNet-18 level of performances. With such sparse data, since RevNet has the most parameters to fine-tune it is thereby the most prone to overfit. Overall, the RevNet results slightly contrasts with the one found by [14], where a RevNet representation surpasses ResNet when training with rotation. However, we note that their methods of evaluation differs significantly from ours. These results emphasise that while K-means neighbors or full supervision can be used to evaluate self-supervised learning [14, 38], SSL could also be used as a way to evaluate the intermediate representation learnt from self-supervised learning algorithms.

4.6. Self-supervision performance vs classification accuracy

In general, different self-supervision tasks would lead to different intermediate representation. However one could also wonder how much solving the proxy task could help to find a representation that would be best suited for the classification task. Intuitively, heavy fine-tuning on the proxy task could lead to a more specialized network which would be harder to adapt to the classification task. To verify this, we studied the rotation accuracy impact on the final SSL performance. Our self-supervised method was trained for 200 epochs with a step-wise decaying learning rate starting from 0.1 at epochs 60, 120, and 160. We saved snapshot of the network before decaying the learning rate and measure the performance reached by our method on the CIFAR-100 classification task with 10 labelled instances per class. In addition, we also trained a RotNet for 500 epochs with a learning rate schedule directly extracted from [6]. We summarise the results in table 4.

	Self-supervised training stopped at epoch				FT
	60	120	150	200	
Rotation accuracy	74.1	83.5	85.9	86.1	**86.8**
Labelled set	37.6	40.4	**40.9**	40.7	40.4
SSL (our method)	46.5	50.2	50.4	50.7	**52.3**

Table 4. **Rotation accuracy vs Classification accuracy.** Correlation between rotation accuracy and final SSL classification accuracy on CIFAR-100 with 10 labels per class. 'FT' is fine-tuned model for 500 epochs. 'Labelled' set indicates fine-tuning on label set only.

Overall we found that fine-tuning on the proxy task was beneficial to the SSL algorithm. Interestingly, we note that the accuracy on the labelled set does not entirely correlate with the final SSL accuracy. While the model with best accuracy on the labelled set obtained 50.4% SSL accuracy, the fine-tuned model obtained an overall SSL accuracy of 52.3%. This also corroborates the results found in section 4.5, better accuracy on the complete or partial set does not necessarily mean a better final SSL accuracy. We believe this result also motivates a new method of self-supervised learning evaluation using SSL.

5. Conclusion and discussion

In this paper, we have pushed the limits of SSL to work with very few annotations. We achieved this goal with several innovations. We first proposed to leverage the power of transfer learning among different tasks and self-supervision to provide a good initial feature representation for SSL. We also proposed a new SSL algorithm that can exploit well such a pre-trained representation, achieving state-of-the-art results on a large number of public benchmarks. We showed that self-supervised learning can not only benefit our proposed SSL algorithm, but also existing state-of-the-art methods. Equipped with the representation learned from self-supervised learning, we found that the performance of state-of-the-art methods can be notably boosted, especially the extreme cases with sparsely annotated data. More generally, we found that SSL could be a new and sensitive downstream task for ranking self-supervised learning methods. Besides working as a standalone method showing superior performance, our method can also serve as an effective refinement method on a network pre-trained either with an existing SSL algorithm or another dataset.

Acknowledgments This work is supported by the EPSRC Programme Grant Seebibyte EP/M013774/1, Mathworks/DTA DFR02620, and ERC IDIU-638009.

References

[1] P. Bachman, O. Alsharif, and D. Precup. Learning with pseudo-ensembles. In *NeurIPS*, 2014. 2

[2] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *NeurIPS*, 20190. 3

[3] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Conference on Computational learning theory*, 1998. 3, 4

[4] O. Chapelle and A. Scholkopf, B.and Zien. *Semi-Supervised Learning*. MIT Press, 2006. 2

[5] D.-D. Chen et al. Tri-net for semi-supervised deep learning. In *IJCAI*, 2018. 3

[6] Y. Chen, X. Zhu, and S. Gong. Semi-supervised deep learning with memory. In *ECCV*, 2018. 3, 5, 7, 8

[7] R. Fergus, Y. Weiss, and A. Torralba. Semi-supervised learning in gigantic image collections. In *NeurIPS*, 2009. 2

[8] S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018. 3, 5, 6, 8

[9] Ramazan Gokberk Cinbis, Jakob Verbeek, and Cordelia Schmid. Multi-fold mil training for weakly supervised object localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2409–2416, 2014. 2

[10] A. N. Gomez et al. The reversible residual network: Backpropagation without storing activations. In *NeurIPS*, 2017. 7, 8

[11] K. He et al. Deep residual learning for image recognition. In *CVPR*, 2016. 5, 8

[12] J. Jackson, J.and Schulman. Semi-supervised learning by label gradient alignment. *ArXiv e-prints*, 2019. 3

[13] K. Kamnitsas et al. Semi-supervised learning via compact latent space clustering. In *ICML*, 2018. 3

[14] X. Kolesnikov, A. andZhai and L Beyer. Revisiting self-supervised visual representation learning. In *ArXiv e-prints*, 2019. 7, 8

[15] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Technical report*, 2009. 5

[16] S. Laine and T. Aila. Temporal ensembling for semi-supervised learning. *ICLR*, 2017. 1, 2, 4, 5, 7, 8

[17] D.-H. Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML Workshop on Challenges in Representation Learning*, 2013. 3

[18] Q. Li, X-M. Wu, and Z. Guan. Generalized label propagation methods for semi-supervised learning. *ICLR*, 2019. 3

[19] Y. Li, L. Liu, and R. Tan. Certainty-driven consistency loss for semi-supervised learning. In *ArXiv e-prints*, 2019. 3

[20] Y. Luo, J. Zhu, et al. Smooth neighbors on teacher graphs for semi-supervised learning. In *CVPR*, 2018. 3

[21] T. Miyato et al. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE TPAMI*, 2018. 1, 2, 3, 4

[22] Y. Nesterov. A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. akad. nauk Sssr*, volume 269, pages 543–547, 1983. 5

[23] Y Netzer et al. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011. 5

[24] A. Oliver et al. Realistic evaluation of deep semi-supervised learning algorithms. In *NeurIPS*, 2018. 1, 3, 7

[25] A. Quattoni and Antonio Torralba. Recognizing indoor scenes. In *CVPR*, 2009. 7

[26] A. Rasmus et al. Semi-supervised learning with ladder networks. In *NeurIPS*, 2015. 2

[27] S.-A. Rebuffi et al. icarl: Incremental classifier and representation learning. In *CVPR*, 2017. 4

[28] O. Russakovsky et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 2015. 3, 7

[29] M. Sajjadi, M. Javanmardi, and T. Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *NeurIPS*, 2016. 2

[30] T. Salimans et al. Improved techniques for training gans. In *NeurIPS*, 2016. 3

[31] M. Shi and B. Zhang. Semi-supervised learning improves gene expression-based prediction of cancer recurrence. *Bioinformatics*, 27(21):3017–3023, 2011. 2

[32] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *ArXiv e-prints*, 2020. 3

[33] J. Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *ArXiv e-prints*, 2015. 3

[34] N. Srivastava et al. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research (JMLR)*, 2014. 1

[35] A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*, 2017. 1, 3, 4, 5, 7

[36] A. Teichman and S.n Thrun. Tracking-based semi-supervised learning. *The International Journal of Robotics Research*, 31(7):804–818, 2012. 2

[37] Verma.V et al. Interpolation consistency training for semi-supervised learning. In *ArXiv e-prints*, 2019. 3, 7

[38] L. Zhang et al. Aet vs. aed: Unsupervised representation learning by auto-encoding transformations rather than data. In *CVPR*, 2019. 8

[39] B. Zhou et al. Learning deep features for scene recognition using places database. In *NeurIPS*, 2014. 7

[40] H. Zhou et al. When semi-supervised learning meets transfer learning: Training strategies, models and datasets. In *ArXiv e-prints*, 2018. 3, 4, 6, 7

[41] Z.-H. Zhou and M. Li. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge & Data Engineering*, 2005. 2, 3

[42] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002. 3

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	Semi-supervised learning with scarce annotations
Publication Status	X Published X Accepted for Publication □ Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style
Publication Details	Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2020. SA Rebuffi*, S Ehrhardt*, K Han*, A Vedaldi, A Zisserman. (* denotes equal contribution)

Student Confirmation

Student Name:	Sylvestre-Alvise REBUFFI		
Contribution to the Paper	• Conception of the idea (self-supervision initialization, alternate optimization) • Implementation and doing the experiments (ablations, refinement, transfer learning) • Writing the paper • Experiments for the rebuttal		
Signature 		Date	24/07/2020

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title:	Prof. Andrea Vedaldi
Supervisor comments	
Signature 	Date

This completed form should be included in the thesis, at the end of the relevant chapter.

Chapter 6

Automatically Discovering and Learning New Visual Categories with Ranking Statistics

This work was presented as a Poster Presentation at the International Conference on Learning Representations (ICLR), 2020.

The goal of this work is to cluster unlabelled samples from new unknown classes given some known labelled classes. Compared to the previous semi-supervised setting, some classes are fully labelled while others are fully unlabelled. First, as opposed to existing methods, we pre-train the early layers of the network with self-supervision rather than using the labelled classes. Second, we propose to jointly train the model on the labelled and unlabelled data by using a two-headed network where the first head clusters the unlabelled samples into new classes and the second head learns to classify the labelled and unlabelled samples. To transfer the knowledge from the classification head to the clustering head, we exploit the shared feature space to generate pairwise clustering pseudo-labels by comparing the most activated channels between pairs of unlabelled samples.

AUTOMATICALLY DISCOVERING AND LEARNING NEW VISUAL CATEGORIES WITH RANKING STATISTICS

Kai Han* **Sylvestre-Alvise Rebuffi*** **Sebastien Ehrhardt***
Andrea Vedaldi **Andrew Zisserman**
Visual Geometry Group, Department of Engineering Science, University of Oxford
{khan,srebuffi,hyenal,vedaldi,az}@robots.ox.ac.uk

ABSTRACT

We tackle the problem of discovering novel classes in an image collection given labelled examples of other classes. This setting is similar to semi-supervised learning, but significantly harder because there are no labelled examples for the new classes. The challenge, then, is to leverage the information contained in the labelled images in order to learn a general-purpose clustering model and use the latter to identify the new classes in the unlabelled data. In this work we address this problem by combining three ideas: (1) we suggest that the common approach of bootstrapping an image representation using the labeled data only introduces an unwanted bias, and that this can be avoided by using self-supervised learning to train the representation from scratch on the union of labelled and unlabelled data; (2) we use rank statistics to transfer the model's knowledge of the labelled classes to the problem of clustering the unlabelled images; and, (3) we train the data representation by optimizing a joint objective function on the labelled and unlabelled subsets of the data, improving both the supervised classification of the labelled data, and the clustering of the unlabelled data. We evaluate our approach on standard classification benchmarks and outperform current methods for novel category discovery by a significant margin.

1 INTRODUCTION

Modern machine learning systems can match or surpass human-level performance in tasks such as image classification (Deng et al., 2009), but at the cost of collecting large quantities of annotated training data. Semi-supervised learning (SSL) (Oliver et al., 2018) can alleviate this issue by mixing labelled with unlabelled data, which is usually much cheaper to obtain. However, these methods still require some annotations for each of the classes that one wishes to learn. We argue this is not always possible in real applications. For instance, consider the task of recognizing products in supermarkets. Thousands of new products are introduced in stores every week, and it would be very expensive to annotate them all. However, new products do not differ drastically from the existing ones, so it should be possible to discover them automatically as they arise in the data. Unfortunately, machines are still unable to effectively learn new classes without manual annotations.

In this paper, we thus consider the problem of discovering new visual classes automatically, assuming that a certain number of classes are already known by the model (Hsu et al., 2018; 2019; Han et al., 2019). This knowledge comes in the form of a *labelled dataset* of images for a certain set of classes. Given that this data is labelled, off-the-shelf supervised learning techniques can be used to train a very effective classifier for the known classes, particularly if Convolutional Neural Networks (CNNs) are employed. However, this does not mean that the learned features are useful as a representation of the *new classes*. Furthermore, even if the representation transfers well, one still has the problem of identifying the new classes in an unlabelled dataset, which is a clustering problem.

We tackle these problems by introducing a novel approach that combines three key ideas (section 2 and fig. 1). The first idea is to pre-train the image representation (a CNN) using all available images, both labelled and unlabelled, using a self-supervised learning objective. Crucially, this objective *does not* leverage the known labels, resulting in features that are much less biased towards the labelled

*indicates equal contribution

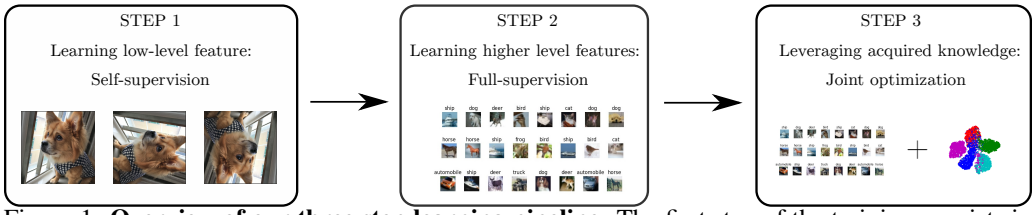


Figure 1: **Overview of our three step learning pipeline**. The first step of the training consists in learning an unbiased image representation via self-supervision using both labelled and unlabelled data, which learns well the early layers of the representation; in the second step, we fine-tune only the last few layers of the model using supervision on the labelled set; finally, the fine-tuned representation is used, via rank statistics, to induce clusters in the unlabelled data, while maintaining a good representation on the labelled set.

classes. Labels are used only after pre-training to learn a classifier specific to the labelled data as well as to fine-tune the deepest layers of the CNN, for which self-supervision is not as effective.

The second idea is a new approach to transfer the information contained in the labelled images to the problem of clustering the unlabelled ones. Information is transferred by sharing the same representation between labelled and unlabelled images, motivated by the fact that the new classes are often similar to the known ones. In more detail, pairs of unlabelled images are compared via their representation vectors. The comparison is done using robust rank statistics, by testing if two images share the same subset of k maximally activated representation components. This test is used to decide if two unlabelled images belong to the same (new) class or not, generating a set of noisy pairwise pseudo-labels. The pseudo-labels are then used to learn a similarity function for the unlabelled images.

The third and final idea is, after bootstrapping the representation, to optimise the model by minimizing a joint objective function, containing terms for both the labelled and unlabelled subsets, using respectively the given labels and the generated pseudo-labels, thus avoiding the forgetting issue that may arise with a sequential approach. A further boost is obtained by incorporating incremental learning of the discovered classes in the classification task, which allows information to flow between the labelled and unlabelled images.

We evaluate our method on several public benchmarks (section 3), outperforming by a large margin all existing techniques (section 4) that can be applied to this problem, demonstrating the effectiveness of our approach. We conclude the paper by summarizing our findings (section 5). Our code can be found at `http://www.robots.ox.ac.uk/~vgg/research/auto_novel`.

2 METHOD

Given an *unlabelled* dataset $D^u = \{x_i^u, i = 1, \dots, M\}$ of images $x_i^u \in \mathbb{R}^{3\times H\times W}$, our goal is to automatically cluster the images into a number of classes C^u, which we assume to be known *a priori*. We also assume to have a second *labelled* image dataset $D^l = \{(x_i^l, y_i^l), i = 1, \dots, N\}$ where $y_i^l \in \{1, \dots, C^l\}$ is the class label for image x_i^l. We also assume that the set of C^l labelled classes is disjoint from the set of C^u unlabelled ones. While the statistics of D^l and D^u thus differ, we hypothesize that a general notion of what constitutes a "good class" can be extracted from D^l and that the latter can be used to better cluster D^u.

We approach the problem by learning an image representation $\Phi : x \mapsto \Phi(x) \in \mathbb{R}^d$ in the form of a CNN. The goal of the representation is to help to recognize the known classes and to discover the new ones. In order to learn this representation, we combine three ideas, detailed in the next three sections.

2.1 SELF-SUPERVISED LEARNING

Given that we have a certain number of labelled images D^l at our disposal, the obvious idea is to use these labels to bootstrap the representation Φ by minimizing a standard supervised objective such as the cross-entropy loss. However, experiments show that this causes the representation to overly-specialize for the classes in D^l, providing a poor representation of the new classes in D^u.

Thus we resist the temptation of using the labels right away and use instead a self-supervised learning method to bootstrap the representation Φ. Self-supervised learning has been shown (Kolesnikov et al., 2019; Gidaris et al., 2018) to produce robust low-level features, especially for the first few layers of typical CNNs. It has the benefit that no data annotations are needed, and thus it can be applied to both labelled and unlabelled images during training. In this way, we achieve the key benefit of ensuring that the representation is initialized without being biased towards the labelled data.

In detail, we first pre-train our model Φ with self-supervision on the union of D^l and D^u (ignoring all labels). We use the RotNet (Gidaris et al., 2018) approach[1] due to its simplicity and efficacy, but any self-supervised method could be used instead. We then extend the pre-trained network Φ with a classification head $\eta^l : \mathbb{R}^d \to \mathbb{R}^{C^l}$ implemented as a single linear layer followed by a softmax layer. The function $\eta^l \circ \Phi$ is fine-tuned on the labelled dataset D^l in order to learn a classifier for the C^l known classes, this time using the labels y_i and optimizing the standard cross-entropy (CE) loss:

$$L_{\text{CE}} = -\frac{1}{N}\sum_{i=1}^{N} \log \eta^l_{y_i}(z^l_i) \tag{1}$$

where $z^l_i = \Phi(x^l_i) \in \mathbb{R}^d$ is the representation of image x^l_i. Only η^l and the last macro-block of Φ (section 3) are updated in order to avoid overfitting the representation to the labelled data.

2.2 Transfer learning via rank statistics

Once the representation Φ and the classifier η^l have been trained, we are ready to look for the new classes in D^u. Since the classes in D^u are unknown, we represent them by defining a relation among pairs of unlabelled images (x^u_i, x^u_j). The idea is that similar images should belong to the same (new) class, which we denote by the symbol $s_{ij} = 1$, while dissimilar ones should not, which we denote by $s_{ij} = 0$. The problem is then to obtain the labels s_{ij}.

Our assumption is that the new classes will have some degree of visual similarity with the known ones. Hence, the learned representation should be applicable to old and new classes equally well. As a consequence, we expect the descriptors $z^u_i = \Phi(x^u_i)$ and $z^u_j = \Phi(x^u_j)$ of two images x^u_i, x^u_j from the new classes to be close if they are from the same (new) class, and to be distinct otherwise.

Rather than comparing vectors z^u_i, z^u_j directly (e.g., by a scalar product), however, we use a more robust rank statistics. Specifically, we rank the values in vector z^u_i by magnitude. Then, if the rankings obtained for two unlabelled images x^u_i and x^u_j are the same, they are very likely to belong to the same (new) class, so we set $s_{ij} = 1$. Otherwise, we set $s_{ij} = 0$. In practice, it is too strict to require the two rankings to be identical if the dimension of z^u_i is high (otherwise we may end up with $s_{ij} = 0$ for all pairs $(i, j), i \neq j$). Therefore, we relax this requirement by only testing if the *sets* of the top-k ranked dimensions are the same (we use $k = 5$ in our experiments), i.e.:

$$s_{ij} = \mathbb{1}\left\{\text{top}_k(\Phi(x^u_i)) = \text{top}_k(\Phi(x^u_j))\right\}, \tag{2}$$

where $\text{top}_k : \mathbb{R}^d \to \mathcal{P}(\{1, \dots, d\})$ associates to a vector z the subset of indices $\{1, \dots, d\}$ of its top-k elements.

Once the labels s_{ij} have been obtained, we use them as pseudo-labels to train a comparison function for the unlabelled data. In order to do this, we apply a new head $\eta^u : \mathbb{R}^d \to \mathbb{R}^{C^u}$ to the image representation $z^u_i = \Phi(x^u_i)$ to extract a new descriptor vector $\eta^u(z^u_i)$ optimized for the unlabelled data. As in section 2.1, the head is composed of a linear layer followed by a softmax. Then, the inner product $\eta^u(z^u_i)^\top \eta^u(z^u_j)$ is used as a score for whether images x^u_i and x^u_j belong to the same class or not. Note that $\eta^u(z^u_i)$ is a normalized vector due to the softmax layer in η^u. This descriptor is trained by optimizing the *binary cross-entropy* (BCE) loss:

$$L_{\text{BCE}} = -\frac{1}{M^2}\sum_{i=1}^{M}\sum_{j=1}^{M}[s_{ij} \log \eta^u(z^u_i)^\top \eta^u(z^u_j) + (1 - s_{ij}) \log(1 - \eta^u(z^u_i)^\top \eta^u(z^u_j))]. \tag{3}$$

[1] We present to the network Φ randomly-rotated versions Rx of each image and task it with predicting R. The problem is formulated as a 4-way classification of the rotation angle, with angle in $\{0°, 90°, 180°, 270°\}$. The model $\eta \circ \Phi(Rx)$ is terminated by a single linear layer η with 4 outputs each scoring an hypothesis. The parameters of η and Φ are optimized by minimizing the cross-entropy loss on the rotation prediction.

Furthermore, we structure η^u in a particular manner: We set its output dimension to be equal to the number of new classes C^u. In this manner, we can use the index of the maximum element of each vector $\hat{y}_i^u = \text{argmax}_y[\eta^u \circ \Phi(x_i^u)]_y$ as prediction $\hat{y}_i^u$ for the class of image x_i^u (as opposed to assigning labels via a clustering method such as k-means).

2.3 JOINT TRAINING ON LABELLED AND UNLABELLED DATA

We now have two losses that involve the representation Φ: the CE loss L_{CE} for the labelled data D^l and the pairwise BCE loss L_{BCE} for the unlabelled data D^u. They both share the same image embedding Φ. This embedding can be trained sequentially, first on the labelled data, and then on the unlabelled data using the pseudo-labels obtained above. However, in this way the model will very likely forget the knowledge learned from the labelled data, which is known as *catastrophic forgetting* in incremental learning (Rebuffi et al., 2017; Lopez-Paz & Ranzato, 2017; Shmelkov et al., 2017; Aljundi et al., 2018).

In contrast, we jointly fine-tune our model using both losses at the same time. Note that most of the model Φ is frozen; we only fine-tune the last macro-block of Φ together with the two heads η^u and η^l. Importantly, as we fine-tune the model, the labels s_{ij} are changing at every epoch as the embedding η^l is updated. This in turn affects the rank statistics used to determine the labels s_{ij} as explained in section 2.2. This leads to a "moving target" phenomenon that can introduce some instability in learning the model. This potential issue is addressed in the next section.

2.4 ENFORCING PREDICTIONS TO BE CONSISTENT

In addition to the CE and BCE losses, we also introduce a consistency regularization term, which is used for both labelled and unlabelled data. In semi-supervised learning (Oliver et al., 2018; Tarvainen & Valpola, 2017; Laine & Aila, 2017), the idea of consistency is that the class predictions on an image x and on a randomly-transformed counterpart tx (for example an image rotation) should be the same. In our case, as will be shown in the experiments, consistency is very important to obtain good performance. One reason is that, as noted above, the pairwise pseudo-labels for the unlabelled data are subject to change on the fly during training. Indeed, for an image x_i^u and a randomly-transformed counterpart tx_i^u, if we do not enforce consistency, we can have $\text{top}_k(\Phi(x_i^u)) \neq \text{top}_k(\Phi(tx_i^u))$. According to eq. (2) defining s_{ij}, it could result in different s_{ij} for x_i^u depending on the data augmentation applied to the image. This variability of the ranking labels for a given pair could then confuse the training of the embedding.

Following the common practice in semi-supervised learning, we use the *Mean Squared Error* (MSE) as the consistency cost. This is given by:

$$L_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(\eta^l(z_i^l) - \eta^l(\hat{z}_i^l))^2 + \frac{1}{M}\sum_{i=1}^{M}(\eta^u(z_i^u) - \eta^u(\hat{z}_i^u))^2, \tag{4}$$

where $\hat{z}$ is the representation of tx.

The overall loss of our model can then be written as

$$L = L_{\text{CE}} + L_{\text{BCE}} + \omega(t)L_{\text{MSE}}, \tag{5}$$

where the coefficient $\omega(t)$ is a ramp-up function. This is widely used in semi-supervised learning (Laine & Aila, 2017; Tarvainen & Valpola, 2017). Following (Laine & Aila, 2017; Tarvainen & Valpola, 2017), we use the sigmoid-shaped function $\omega(t) = \lambda e^{-5(1-\frac{t}{T})^2}$, where t is current time step and T is the ramp-up length and $\lambda \in \mathbb{R}_+$.

2.5 INCREMENTAL LEARNING SCHEME

We also explore a setting analogous to incremental learning. In this approach, after tuning on the labelled set (end of section 2.1), we extend the head η^l to C^u new classes, so that $\eta^l : \mathbb{R}^d \to \mathbb{R}^{C^l+C^u}$. The head parameters for the new classes are initialized randomly. The model is then trained using the same loss eq. (5), but the cross-entropy part of the loss is evaluated on both labelled and unlabelled data D^l and D^u. Since the cross-entropy requires labels, for the unlabelled data we use the *pseudo-labels* $\hat{y}_i^u$, which are generated on-the-fly from the head η^u at each forward pass.

The advantage is that this approach *increments* η^l to discriminate both old and new classes, which is often desirable in applications. It also creates a feedback loop that causes the features z_i^u to be

	CIFAR-10	CIFAR-100	SVHN
Ours w/o Con	82.6±12.0%	61.8±3.6%	61.3±1.9%
Ours w/o CE	84.7±4.4%	58.4±2.7%	59.7±6.6%
Ours w/o BCE	26.2±2.0%	6.6±0.7%	24.5±0.5%
Ours w/o S.S.	89.4±1.4%	67.4±2.0%	72.9±5.0%
Ours full	**90.4±0.5%**	**73.2±2.1%**	**95.0±0.2%**
Ours w/ I.L.	**91.7±0.9%**	**75.2±4.2%**	**95.2±0.3%**

Table 1: **Ablation study.** "w/o Con." means without consistency constraints; "w/o CE" means without the cross entropy loss for training on labeled data. "w/o BCE" means without binary cross entropy loss for training on unlabeled data; "w/o S.S." means without self-supervision.

refined, which in turn generates better pseudo-labels $\hat{y}_i^u$ for D^u from the head η^u. In this manner, further improvements can be obtained by this cycle of positive interactions between the two heads during training.

3 EXPERIMENTS

3.1 DATA AND EXPERIMENTAL DETAILS

We evaluate our models on a variety of standard benchmark datasets: CIFAR-10 (Krizhevsky & Hinton, 2009), CIFAR-100 (Krizhevsky & Hinton, 2009), SVHN (Netzer et al., 2011), OmniGlot (Lake et al., 2015), and ImageNet (Deng et al., 2009). Following Han et al. (2019), we split these to have 5/20/5/654/30 classes respectively in the unlabelled set. In addition, for OmniGlot and ImageNet we use 20 and 3 different splits respectively, as in Han et al. (2019), and report average clustering accuracy. More details on the splits can be found in appendix A.

Evaluation metrics. We adopt *clustering accuracy* (ACC) to evaluate the clustering performance of our approach. The ACC is defined as follow:

$$\max_{g \in \mathrm{Sym}(L)} \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\left\{\overline{y}_i = g\left(y_i\right)\right\}, \tag{6}$$

where $\overline{y}_i$ and y_i denote the ground-truth label and clustering assignment for each data point $x_i^u \in D^u$ respectively, and $\mathrm{Sym}(L)$ is the group permutations of L elements (this discounts the fact that the cluster indices may not be in the same order as the ground-truth labels). Permutations are optimized using the Hungarian algorithm (Kuhn, 1955).

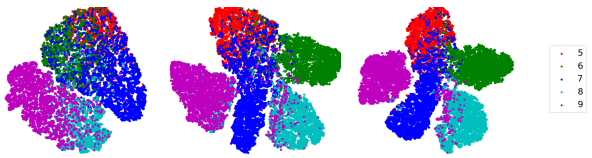
Implementation details. We use the ResNet-18 (He et al., 2016) architecture, except for OmniGlot for which we use a VGG-like network (Simonyan & Zisserman, 2015) with six layers to make our setting directly comparable to prior work. We use SGD with momentum (Sutskever et al., 2013) as optimizer for all but the OmniGlot dataset, for which we use Adam (Kingma & Ba, 2014). For all experiments we use a batch size of 128 and $k = 5$ which we found worked consistently well across datasets (see appendix D). More details about the hyper-parameters can be found in appendix B.

3.2 ABLATION STUDY

We validate the effectiveness of the components of our method by ablating them and measuring the resulting ACC on the unlabelled data. Note that, since the evaluation is restricted to the unlabelled data, we are solving a clustering problem. The same unlabelled data points are used for both training and testing, except that data augmentation (i.e. image transformations) is not applied when computing the cluster assignments. As can be seen in table 1, all components have a significant effect as removing any of them causes the performance to drop substantially. Among them, the BCE loss is by far the most important one, since removing it results in a dramatic drop of 40–60% absolute ACC points. For example, the full method has ACC 90.4% on CIFAR-10, while removing BCE causes the ACC to drop to 26.2%. This shows that that our rank-based embedding comparison can indeed generate reliable pairwise pseudo labels for the BCE loss. Without consistency, cross entropy, or self-supervision, the performance drops by a more modest but still significant 7.8%, 5.7% and 1.0% absolute ACC points, respectively, for CIFAR-10. It means that the consistency term plays a role as important as the cross-entropy term by preventing the "moving target" phenomenon described in section 2.4. Finally, by incorporating the discovered classes in the classification task, we get a further boost of 1.3%, 2.0% and 0.2% points on CIFAR-10, CIFAR-100 and SVHN respectively.

Table 2: **Novel category discovery results on CIFAR-10, CIFAR-100, and SVHN.** ACC on the unlabelled set. "w/ S.S." means with self-supervised learning.

No		CIFAR-10	CIFAR-100	SVHN
(1)	k-means (MacQueen, 1967)	65.5±0.0 %	56.6±1.6%	42.6%±0.0
(2)	KCL (Hsu et al., 2018)	66.5±3.9%	14.3±1.3%	21.4%±0.6
(3)	MCL (Hsu et al., 2019)	64.2±0.1%	21.3±3.4%	38.6%±10.8
(4)	DTC (Han et al., 2019)	87.5±0.3%	56.7±1.2%	60.9%±1.6
(5)	k-means (MacQueen, 1967) w/ S.S.	72.5±0.0%	56.3±1.7%	46.7±0.0%
(6)	KCL (Hsu et al., 2018) w/ S.S.	72.3±0.2%	42.1±1.8%	65.6±4.9%
(7)	MCL (Hsu et al., 2019) w/ S.S.	70.9±0.1%	21.5±2.3%	53.1±0.3%
(8)	DTC (Han et al., 2019) w/ S.S.	88.7±0.3%	67.3±1.2%	75.7±0.4%
(9)	Ours	**90.4±0.5%**	**73.2±2.1%**	**95.0±0.2%**
(10)	Ours w/ I.L.	**91.7±0.9%**	**75.2±4.2%**	**95.2±0.2%**



(a) init (b) epoch 30 (c) epoch 90

Figure 2: **Evolution of the t-SNE during the training of CIFAR-10.** Performed on unlabelled data (i.e., instances of dog, frog, horse, ship, truck). Colors of data points denote their ground-truth labels.

3.3 Novel category discovery

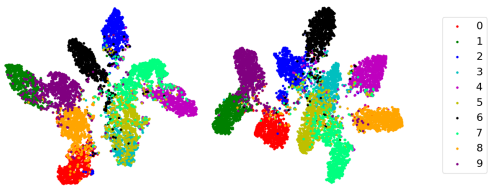
We compare our method to baselines and state-of-the-art methods for new class discovery, starting from CIFAR-10, CIFAR-100, and SVHN in table 2. The first baseline (row 5 in table 2) amounts to applying k-means (MacQueen, 1967) to the features extracted by the fine-tuned model (the second step in section 2.1), for which we use the k-means++ (Arthur & Vassilvitskii, 2007) initialization. The second baseline (row 1 in table 2) is similar, but uses as feature extractor a model trained from scratch using only the labelled images, which corresponds to a standard transfer learning setting. By comparing rows 1, 5 and 9 in table 2, we can see that our method substantially outperforms k-means. Next, we compare with the KCL (Hsu et al., 2018), MCL (Hsu et al., 2019) and DTC (Han et al., 2019) methods. By comparing rows 2–4 to 9, we see that our method outperforms these by a large margin. We also try to improve KCL, MCL and DTC by using the same self-supervised initialization we adopt (section 2.1), which indeed results in an improvement (rows 2–4 vs 6–8). However, their overall performance still lags behind ours by a large margin. For example, our method of section 2.4 achieves 95.0% ACC on SVHN, while "KCL w/ S.S.", "MCL w/ S.S." and "DTC w/ S.S." achieve only 65.6%, 53.1% and 75.7% ACC, respectively. Similar trends hold for CIFAR-10 and CIFAR-100. Finally, the incremental learning scheme of section 2.5 results in further improvements, as can be seen by comparing rows 9 and 10 of table 2.

In fig. 2, we show the evolution of the learned representation on the unlabelled data on CIFAR-10 using t-SNE (van der Maaten & Hinton, 2008). As can be seen, while the clusters overlap in the beginning, they become more and more separated as the training progresses, showing that our model can effectively discover novel visual categories without labels and learn meaningful embeddings for them.

We further compare our method to others on two more challenging datasets, OmniGlot and ImageNet, in table 3. For OmniGlot, results are averaged over the 20 alphabets in the *evaluation* set (see appendix A); for ImageNet, results are averaged over the three 30-class unlabelled sets used in (Hsu et al., 2018; 2019). Since we have a relatively larger number of labelled classes in these two datasets, we follow (Han et al., 2019) and use metric learning on the labelled classes to pre-train the feature extractor, instead of the self-supervised learning. We empirically found that self-supervision does not provide obvious gains for these two datasets. This is reasonable since the data in the labelled sets of these two datasets are rather diverse and abundant, so metric learning can provide good feature initialization as there is less class-specific bias due to the large number of pre-training classes. However, by comparing rows 1 and 5 in table 3, it is clear that metric learning alone is not sufficient for the task of novel category discovery. Our method substantially outperforms the k-means results obtained using the features from metric learning — by 11.9% and 10.6% on OmniGlot and ImageNet respectively. Our method also substantially outperforms the current state-of-the-art, achieving 89.1%

No		OmniGlot	ImageNet
(1)	k-means (MacQueen, 1967)	77.2%	71.9%
(2)	KCL (Hsu et al., 2018)	82.4%	73.8%
(3)	MCL (Hsu et al., 2019)	83.3%	74.4%
(4)	DTC (Han et al., 2019)	89.0%	78.3%
(5)	Ours	**89.1%**	**82.5%**

Table 3: **Novel category discovery results on OmniGlot and ImageNet.** ACC on the unlabelled set.



(a) Ours (b) + incr. learning

Figure 3: **t-SNE on CIFAR-10: impact of incremental Learning.** (a) representation on the labelled and (b) unlabelled CIFAR classes. Colors of data points denote their ground-truth labels. We observe a bigger overlap in (a) between the "old" class 3 and the "new" class 5 when not incorporating Incremental Learning.

and 82.5% ACC on OmniGlot and ImageNet respectively, compared with 89.0% and 78.8% of (Han et al., 2019), thus setting the new state-of-the-art.

3.4 Incremental Learning

Table 4: **Incremental Learning with the novel categories.** "old" refers to the ACC on the labelled classes while "new" refers to the unlabelled classes in the *testing set*. "all" indicates the whole testing set. It should be noted that the predictions are not restricted to their respective subset. Standard deviation can be found in appendix C.

	CIFAR-10			CIFAR-100			SVHN		
Classes	old	new	all	old	new	all	old	new	all
KCL w/ S.S.	79.4%	60.1%	69.8%	23.4%	29.4%	24.6%	90.3%	65.0%	81.0%
MCL w/ S.S.	81.4%	64.8%	73.1%	18.2%	18.0%	18.2%	94.0%	48.6%	77.2%
DTC w/ S.S.	58.7%	78.6%	68.7%	47.6%	49.1%	47.9%	90.5%	72.8%	84.0%
Ours w/ I.L.	**90.6%**	**88.8%**	**89.7%**	**71.2%**	**56.8%**	**68.3%**	**96.3%**	**96.1%**	**96.2%**

Here, we further evaluate our incremental scheme for novel category discovery as described in section 2.5. Methods for novel category discovery such as (Han et al., 2019; Hsu et al., 2019; 2018) focus on obtaining the highest clustering accuracy for the new unlabelled classes, but may forget the existing labelled classes in the process. In practice, forgetting is not desirable as the model should be able to recognize both old and new classes. Thus, we argue that the classification accuracy on the labelled classes should be assessed as well, as for any incremental learning setting. Note however that our setup differs substantially from standard incremental learning (Rebuffi et al., 2017; Lopez-Paz & Ranzato, 2017; Shmelkov et al., 2017; Aljundi et al., 2018) where every class is labelled and the focus is on using limited memory. In our case, we can store and access the original data without memory constraints, but the new classes are unlabelled, which is often encountered in applications.

By construction (section 2.5), our method learns the new classes on top of the old ones incrementally, out of the box. In order to compare to methods such as KCL, MCL and DTC that do not have this property, we proceed as follows. First, the method runs as usual to cluster the unlabelled portion of the data, thus obtaining pseudo-labels for it, and learning a feature extractor as a byproduct. Then, the feature extractor is used to compute features for both the labelled and unlabelled training data, and a linear classifier is trained using labels and pseudo-labels, jointly on all the classes, old and new.

We report in table 4 the performance of the resulting joint classifier networks *on the testing set* of each dataset (this is now entirely disjoint from the training set). Our method has similar performances on the old and new classes for CIFAR-10 and SVHN, as might be expected as the split between old and new classes is balanced. In comparison, the feature extractor learned by KCL and MCL works much better for the old classes (e.g., the accuracy discrepancy between old and new classes is 25.3% for KCL on SVHN). Conversely, DTC learns features that work better for the new classes, as shown by the poor performance for the old classes on CIFAR-10. Thus, KCL, MCL and DTC learn representations that are biased to either the old or new classes, resulting overall in suboptimal performance. In contrast, our method works well on both old and new classes; furthermore, it drastically outperforms existing methods on both.

4 RELATED WORK

Our work draws inspiration from semi-supervised learning, transfer learning, clustering, and zero-shot learning. We review below the most relevant contributions.

In semi-supervised learning (SSL) (Chapelle et al., 2006), a partially labelled training dataset is given and the objective is to learn a model that can propagate the labels from the labelled data to unlabelled data. Most SSL methods focus on the classification task where, usually both labelled and unlabelled points belong to the same set of classes. On the contrary, our goal is to handle the case where the unlabelled data classes differ from the labelled data. Oliver et al. (2018) summarizes the state-of-the-art SSL methods. Among them, the consistency-based methods appeared to be the most effective. Rasmus et al. (2015) propose a ladder network which is trained on both labelled and unlabelled data using a reconstruction loss. Laine & Aila (2017) simplifies this ladder network by enforcing prediction consistency between a data point and its augmented counterpart. As an alternative to data augmentation, they also consider a regularization method based on the exponential moving average (EMA) of the predictions. This idea is further improved by Tarvainen & Valpola (2017): instead of using the EMA of predictions, they propose to maintain the EMA of model parameters. The consistency is then measured between the predictions of the current model (student) and the predictions of the EMA model (teacher). More recently (and closer to our work) practitioners have also combined SSL with self-supervision(Rebuffi et al., 2019; Zhai et al., 2019) to leverage dataset with very few annotations.

Transfer learning (Pan & Yang, 2010; Weiss et al., 2016; Tan et al., 2018) is an effective way to reduce the amount of data annotations required to train a model by pre-training the model on an existing dataset. In image classification, for example, it is customary to start from a model pre-trained on the ImageNet (Deng et al., 2009) dataset. In most transfer learning settings, however, both the source data and the target data are fully annotated. In contrast, our goal is to transfer information from a labelled dataset to an unlabelled one.

Many classic (e.g., Aggarwal & Reddy (2013); MacQueen (1967); Comaniciu & Meer (1979); Ng et al. (2001)) and deep learning (e.g., Xie et al. (2016); Chang et al. (2017); Dizaji et al. (2017); Yang et al. (2017; 2016); Hsu et al. (2018; 2019)) clustering methods have been proposed to automatically partition an unlabelled data collection into different classes. However, this task is usually ill-posed as there are multiple, equally valid criteria to partition most datasets. We address this challenge by learning the appropriate criterion by using a labelled dataset, narrowing down what constitutes a proper class. We call this setting "transfer clustering".

To the best of our knowledge, the work most related to ours are (Hsu et al., 2018; 2019; Han et al., 2019). Han et al. (2019) also consider discovering new classes as a transfer clustering problem. They first learn a data embedding by using metric learning on the labelled data, and then fine-tune the embedding and learn the cluster assignments on the unlabelled data. In (Hsu et al., 2018; 2019), the authors introduce KCL and MCL clustering methods. In both, a similarity prediction network (SPN), also used in (Hsu et al., 2016), is first trained on a labelled dataset. Afterwards, the pre-trained SPN is used to provide binary pseudo labels for training the main model on an unlabelled dataset. The overall pipelines of the two methods are similar, but the losses differ: KCL uses a Kullback-Leibler divergence based contrastive loss equivalent to the BCE used in this paper (eq. (3)), and MCL uses the Meta Classification Likelihood loss. Zero-shot learning (ZSL) (Xian et al., 2018; Fu et al., 2018) can also be used to recognize new classes. However, differently from our work, ZSL also requires additional side information (e.g., class attributes) in addition to the raw images.

Finally, other works (Dean et al., 2013; Yagnik et al., 2011) discuss the application of rank statistics to measuring the similarity of vectors; however, to the best of our knowledge, we are the first to apply rank statistics to the task of novel category discovery using deep neural networks.

5 CONCLUSIONS

In this paper, we have looked at the problem of discovering new classes in an image collection, leveraging labels available for other, known classes. We have shown that this task can be addressed very successfully by a few new ideas. First, the use of self-supervised learning for bootstrapping the image representation trades off the representation quality with its generality, and for our problem this leads to a better solution overall. Second, we have shown that rank statistics are an effective method to compare noisy image descriptors, resulting in robust data clustering. Third, we have

shown that jointly optimizing both labelled recognition and unlabelled clustering in an incremental learning setup can reinforce the two tasks while avoiding forgetting. On standard benchmarks, the combination of these ideas results in much better performance than existing methods that solve the same task. Finally, for larger datasets with more classes and diverse data (e.g., ImageNet) we note that self-supervision can be bypassed as the pretraining on labelled data already provides a powerful enough representation. In such cases, we still show that the rank statistics for clustering gives drastic improvement over existing methods.

6 ACKNOWLEDGMENTS

This work is supported by the EPSRC Programme Grant Seebibyte EP/M013774/1, Mathworks/DTA DFR02620, and ERC IDIU-638009.

REFERENCES

Charu C. Aggarwal and Chandan K. Reddy. *Data Clustering: Algorithms and Applications*. CRC Press, 2013.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, 2018.

David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *ACM-SIAM symposium on Discrete algorithms*, 2007.

Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep adaptive image clustering. In *ICCV*, 2017.

Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. *Semi-Supervised Learning*. MIT Press, 2006.

Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE TPAMI*, 1979.

Thomas Dean, Mark A. Ruzon, Mark Segal, Jonathon Shlens, Sudheendra Vijayanarasimhan, and Jay Yagnik. Fast, accurate detection of 100,000 object classes on a single machine. In *CVPR*, 2013.

Jia Deng, Wei Dong, Richard Socher, Li-Jua Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *ICCV*, 2017.

Yanwei Fu, Tao Xiang, Yu-Gang Jiang, Xiangyang Xue, Leonid Sigal, and Shaogang Gong. Recent advances in zero-shot recognition: Toward data-efficient understanding of visual content. *IEEE Signal Processing Magazine*, 2018.

Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *ICLR*, 2018.

Kai Han, Andrea Vedaldi, and Andrew Zisserman. Learning to discover novel visual categories via deep transfer clustering. In *ICCV*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Yen-Chang Hsu, Zhaoyang Lv, and Zsolt Kira. Deep image category discovery using a transferred similarity function. *arxiv*, 2016.

Yen-Chang Hsu, Zhaoyang Lv, and Zsolt Kira. Learning to cluster in order to transfer across domains and tasks. In *ICLR*, 2018.

Yen-Chang Hsu, Zhaoyang Lv, Joel Schlosser, Phillip Odom, and Zsolt Kira. Multi-class classification without multi-class labels. In *ICLR*, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014.

Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. In *CVPR*, 2019.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Technical report*, 2009.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 1955.

Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *ICLR*, 2017.

Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 2015.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *NeurIPS*, 2017.

James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *NeurIPS*, 2001.

Avital Oliver, Augustus Odena, Colin Raffel, Ekin D. Cubuk, and Ian J. Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *NeurIPS*, 2018.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 2010.

Rasmus, Mathias M. Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *NeurIPS*, 2015.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 2017.

Sylvestre-Alvise Rebuffi, Sebastien Ehrhardt, Kai Han, Andrea Vedaldi, and Andrew Zisserman. Semi-supervised learning with scarce annotations. *arxiv*, 2019.

Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. Incremental learning of object detectors without catastrophic forgetting. In *ICCV*, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013.

Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *International Conference on Artificial Neural Networks*, 2018.

Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*, 2017.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 2008.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *NeurIPS*, 2016.

Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big Data*, 2016.

Yongqin Xian, Christoph H. Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning - a comprehensive evaluation of the good, the bad and the ugly. *IEEE TPAMI*, 2018.

Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, 2016.

Jay Yagnik, Dennis Strelow, David A. Ross, and Ruei sung Lin. The power of comparative reasoning. In *ICCV*, 2011.

Bo Yang, Xiao Fu, Nicholas D. Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *ICML*, 2017.

Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *CVPR*, 2016.

Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S4l: Self-supervised semi-supervised learning. 2019.

A DATASET SPLITS

For CIFAR-10 and SVHN we keep the labels of the five first categories (namely airplane, automobile, bird, cat, deer for CIFAR-10, 0–4 for SVHN) and keep the rest of the data as the unlabelled set. For CIFAR-100 we use the first 80 categories as labelled data while the rest are used for the unlabelled set. Following Hsu et al. (2018; 2019), for OmniGlot, each of the 20 alphabets in *evaluation* set (with 20–47 categories, 659 characters/class) is used as unlabelled data, and all the 30 alphabets in *background* set are used as labelled set (964 characters/class). For ImageNet, we follow Hsu et al. (2018; 2019) to use the 882/118 classes split proposed in Vinyals et al. (2016), and use the three 30-class subsets sampled from the 118 classes as unlabelled sets.

B IMPLEMENTATION DETAILS

In the first self-supervised training step, otherwise mentioned, we trained our model with the pretext task of rotation predictions (i.e., a four-class classification: 0°, 90°, 180°, and 270°) for 200 epochs and a step-wise decaying learning rate starting from 0.1 and divided by 5 at epochs 60, 120, and 160.

In the second step of our framework (i.e., supervised training using labelled data), we fine-tuned our model on the labelled set for 100 epochs and a step-wise decaying learning rate starting from 0.1 and halved every 10 epochs. From this step onward we fix the first three convolutional blocks of the model, and fine-tuned the last convolutional block together with the linear classifier.

Finally, in the last joint training step, we fine-tuned our model for 200/100/90 epochs for {CIFAR-10, CIFAR-100, SVHN}/OmniGlot/ImageNet, which was randomly sampled from the merged set of both labelled and unlabelled data. The initial learning rate was set to 0.1 for all datasets, and was decayed with a factor of 10 at the 170th/{30th, 60th} epoch for {CIFAR-10, CIFAR-100, SVHN}/ImageNet. The learning rate of 0.01 was kept fixed for OmniGlot. For the consistency regularization term, we used the ramp-up function as described in section 2.4 with $\lambda = \{5.0, 50.0, 50.0, 100.0, 10.0\}$, and $T = \{50, 150, 80, 1, 50\}$ for CIFAR-10, CIFAR-100, SVHN, OmniGlot, and ImageNet respectively.

In the incremental learning setting, all previous hyper parameters remain the same for our method. We only add a ramp-up on the cross entropy loss on unlabelled data. The ramp-up length is the same as the one used for eq. (4) and we use for all experiments a coefficient of 0.05. For all other methods we trained the classifier for 150 epochs with SGD with momentum and learning rate of 0.1 divided by 10 at epoch 50.

We implemented our method using PyTorch 1.1.0 and ran experiments on NVIDIA Tesla M40 GPUs. Following (Han et al., 2019), our results were averaged over 10 runs for all datasets, except for ImageNet which was averaged over the three 30-class subsets. In general, we found the results were stable. Our code is publicly available at `http://www.robots.ox.ac.uk/~vgg/research/auto_novel`.

C STANDARD DEVIATION OF INCREMENTAL LEARNING EXPERIMENT IN TABLE 4

Table 5: **Incremental Learning with the novel categories.** "old" refers to the standard deviation ACC on the labelled classes while "new" refers to the unlabelled classes in the *testing set*. "all" indicates the whole testing set. It should be noted that the predictions are not restricted to their respective subset.

	CIFAR-10			CIFAR-100			SVHN		
Classes	old	new	all	old	new	all	old	new	all
KCL w/ S.S.	0.6%	0.6%	0.1%	0.3%	0.3%	0.2%	0.3%	0.5%	0.1%
MCL w/ S.S.	0.4%	0.4%	0.1%	0.3%	0.1%	0.2%	0.2%	0.3%	0.1%
DTC w/ S.S.	0.6%	0.2%	0.3%	0.2%	0.2%	0.2%	0.3%	0.2%	0.1%
Ours w/ I.L.	0.2%	0.2%	0.1%	0.1%	0.3%	0.1%	0.1%	0.0%	0.1%

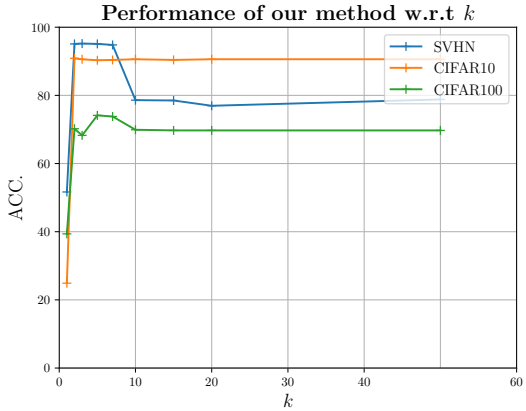


Figure 4: Performance evolution with respect to k. We report results for $k = \{1, 2, 3, 5, 7, 10, 15, 20, 50\}$.

D IMPACT OF k OVER RESULTS

We provide an additional study of the evolution of performances of our method with respect to k. We results on SVHN/CIFAR10/CIFAR100 in fig. 4. We found that $k = \{5, 7\}$ gave the best results overall. We also found that for all values of k except 1 results were in general stable.

E RESULTS WITH AN UNKNOWN NUMBER OF CLASSES

While in our work we assume the number of new classes C^u to be known a priori, this hypothesis can be restrictive in practice. Instead, one can estimate the number of classes in D^u using recent methods such as DTC (Han et al., 2019). In table 6 we compare ACC of KCL (Hsu et al., 2018), KCL (Hsu et al., 2019), DTC(Han et al., 2019) and our method on unlabelled splits of OmniGlot and ImageNet datasets with C^u computed from DTC. We note that our method again reaches the state-of-the-art on ImageNet and is on par with the state-of-the-art on OmniGlot.

Table 6: **Novel category discovery results with unknown** C^u.

No		OmniGlot	ImageNet
(1)	KCL (Hsu et al., 2018)	80.3%	71.4%
(2)	MCL (Hsu et al., 2019)	80.5%	72.9%
(3)	DTC (Han et al., 2019)	**87.0**%	77.6%
(4)	Ours	85.4%	**80.5**%

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	Automatically Discovering and Learning New Visual Categories with Ranking Statistics
Publication Status	X Published X Accepted for Publication □ Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style
Publication Details	The International Conference on Learning Representations (ICLR), 2020. K Han*, SA Rebuffi*, S Ehrhardt*, A Vedaldi, A Zisserman (* denotes equal contribution)

Student Confirmation

Student Name:	Sylvestre-Alvise REBUFFI		
Contribution to the Paper	• Conception of the idea (self-supervision initialization, incremental learning) • Implementation and doing the experiments (incremental learning) • Writing the paper • Experiments for the rebuttal and writing the rebuttal		
Signature		Date	24/07/2020

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title:	Prof. Andrea Vedaldi
Supervisor comments	
Signature 	Date

This completed form should be included in the thesis, at the end of the relevant chapter.

Chapter 7

LSD-C: Linearly Separable Deep Clusters

This work is currently in the review process of the Conference on Neural Information Processing Systems (NeurIPS), 2020.

This paper tackles the problem of deep clustering. Compared to the previous chapters, there are no labels available at training time. First, we pre-train the network with self-supervision to avoid overfitting obvious visual cues. Second, we propose to reformulate clustering as the binary classification problem of predicting whether a pair of samples belong to the same cluster. The pseudo-labels for this clustering loss are extracted at the feature level and are based on the similarity between pairs of samples. To this end, we propose a new similarity based on k-nearest neighbours. Finally, we show that data augmentation and the choice of similarity are essential for the competitive performance of our method.

LSD-C: Linearly Separable Deep Clusters

Sylvestre-Alvise Rebuffi* **Sebastien Ehrhardt*** **Kai Han***
Andrea Vedaldi **Andrew Zisserman**
Visual Geometry Group, Department of Engineering Science, University of Oxford
{srebuffi,hyenal,khan,vedaldi,az}@robots.ox.ac.uk

Abstract

We present LSD-C, a novel method to identify clusters in an unlabeled dataset. Our algorithm first establishes pairwise connections in the feature space between the samples of the minibatch based on a similarity metric. Then it regroups in clusters the connected samples and enforces a linear separation between clusters. This is achieved by using the pairwise connections as targets together with a binary cross-entropy loss on the predictions that the associated pairs of samples belong to the same cluster. This way, the feature representation of the network will evolve such that similar samples in this feature space will belong to the same linearly separated cluster. Our method draws inspiration from recent semi-supervised learning practice and proposes to combine our clustering algorithm with self-supervised pretraining and strong data augmentation. We show that our approach significantly outperforms competitors on popular public image benchmarks including CIFAR 10/100, STL 10 and MNIST, as well as the document classification dataset Reuters 10K. Our code is available at https://github.com/srebuffi/lsd-clusters.

1 Introduction

The need for large scale labelled datasets is a major obstacle to the applicability of deep learning to problems where labelled data cannot be easily obtained. Methods such as clustering, which are unsupervised and thus do not require any kind of data annotation, are in principle more easily applicable to new problems. Unfortunately, standard clustering algorithms [7, 10, 38, 41] usually do not operate effectively on raw data and require to design new data embeddings specifically for each new application. Thus, there is a significant interest in automatically learning an optimal embedding while clustering the data, a problem sometimes referred to as simultaneous data clustering and representation learning. Recent works have demonstrated this for challenging data such as images [27, 51] and text [28, 44]. However, most of these methods work with a constrained output space, which usually coincides with the space of discrete labels or classes being estimated, therefore forcing to work at the level of the semantic of the clusters directly.

In this paper, we relax this limitation by introducing a novel clustering method, *Linearly Separable Deep Clustering* (LSD-C). This method operates in the feature space computed by a deep network and builds on three ideas. First, the method extracts mini-batches of input samples and establishes pairwise pseudo labels (connections) for each pair of sample in the mini-batch. Differently from prior art, this is done in the space of features computed by the penultimate layer of the deep network instead of the final output layer, which maps data to discrete labels. From these pairwise labels, the method learns to regroup the connected samples into clusters by using a clustering loss which forces the clusters to be linearly separable. We empirically show in section 4.2 that this relaxation already significantly improves clustering performance.

*indicates equal contribution

Preprint. Under review.

Second, we initialize the model by means of a self-supervised representation learning technique. Prior work has shown that these techniques can produce features with excellent linear linear separability [4, 16, 22] that are particularly useful as initialization for downstream tasks such as semi-supervised and few-shot learning [14, 42, 54].

Third, we make use of very effective data combination techniques such as RICAP [49] and MixUp [55] to produce composite data samples and corresponding pseudo labels, which are then used at the pairwise comparison stage. In section 4 we show that training with such composite samples and pseudo labels greatly improves the performance of our method, and is in fact the key to good performance in some cases.

We comprehensively evaluate our method on popular image benchmarks including CIFAR 10/100, STL 10 and MNIST, as well as the document classification dataset Reuters 10K. Our method almost always outperforms competitors on all datasets, establishing new state-of-the-art clustering results. The rest of the paper is organized as follows. We first review the most relevant works in section 2. Next, we develop the details of our proposed method in section 3, followed by the experimental results, ablation studies and analysis in section 4. Our code is publicly available at `https://github.com/srebuffi/lsd-clusters`.

2 Related work

Deep clustering. Clustering has been a long-standing problem in the machine learning community, including well-known algorithms such as K-means [38], mean-shift [7], DBSCAN [10] or Gaussian Mixture models [41]. Furthermore it can also be combined with other techniques to achieve very diverse tasks like novel category discovery [21, 11] or semantic instance segmentation [9] among others. With the advances of deep learning, more and more learning-based methods have been introduced in the literature [12, 13, 19, 26, 28, 36, 45, 51, 52]). Among them, DEC [51] is one of the most promising method. It is a two stage method that jointly learns the feature embedding and cluster assignment. The model is first pretrained with an autoencoder using reconstruction loss, after which the model is trained by constructing a sharpened version of the soft cluster assignment as pseudo target. This method inspired a few following works such as IDEC [17] and DCED [18]. JULE [53] is a recurrent deep clustering framework that jointly learns the feature representation with an agglomerative clustering procedure, however it requires tuning a number of hyper-parameters, limiting its practical use. More recently, several methods have been proposed based on mutual information [5, 25, 27]. Among them, IIC [27] achieves the current state-of-the-art results on image clustering by maximizing the mutual information between two transformed counterparts of the same image. Closer to our work is the DAC [3] method, which considers clustering as a binary classification problem. By measuring the cosine similarity between predictions, pairwise pseudo labels are generated from the most confident positive or negative pairs. With the generated pairwise pseudo labels, the model can then be trained by a binary cross-entropy loss. DAC can learn the feature embedding as well as the cluster assignment in an end-to-end manner. Our work significantly differs from DAC as it generates pairwise predictions from a less constrained feature space using similarity techniques not limited to cosine distance.

Self-supervised representation learning. Self-supervised representation learning has recently attracted a lot of attention. Many effective self-supervised learning methods have been proposed in the literature [1, 2, 4, 15, 16, 22]. DeepCluster [2] learns feature representation by classification using the pseudo labels generated from K-means on the learned features in each training epoch. RotNet [16] randomly rotates an image, and learns to predict the applied rotations. Very recently, contrastive learning based methods MoCo [22] and SimCLR [4] have achieved the state-of-the-art self-supervised representation performance, surpassing the representation learnt using ImageNet labels. Self-supervised learning has been also applied in few-shot learning [14], semi-supervised learning [42, 54] and novel category discovery [20], which successfully boosts their performance. In this work we make use of the provably well-conditioned feature space learnt from self-supervised learning method to initialize our network and avoid degenerative cases.

Pairwise pseudo labeling. Pairwise similarity between pairs of sample has been widely used in the literature for dimension reduction or clustering (e.g., t-SNE [37], FINCH [44]). Several methods have shown the effectiveness of using pairwise similarity to provide pseudo labels on-the-fly to train deep convolutional neural networks. In [24], a binary classifier is trained to provide pairwise pseudo labels

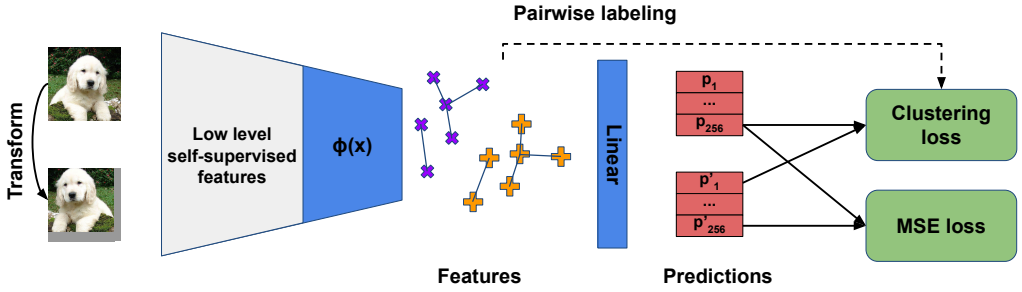


Figure 1: **Overview of LSD-C.** Pairwise labels are extracted at the feature level. They are then used in a clustering loss after the linear classifier. This way, the feature maps will evolve such that connected samples will be grouped in linearly separated clusters. The MSE loss acts a regularizer and enforces the consistency of the cluster predictions when data augmentation is applied.

to train a multi-class classifier. In [20], ranking statistics is used to obtain pairwise pseudo labels on-the-fly for the task of novel category discovery. In [44], the pairwise connection between data points by finding the nearest neighbour is used to cluster images using CNN features. In our method, we compute pairwise labels from a neural network embedding. This way we generate pseudo labels for each pair in each mini-batch and learn cluster assignment without any supervision.

3 Method

Our methods is divided into three stages: (i) self-supervised pre-training, (ii) pairwise connection and clustering, and (iii) data composition. We provide an overview of our pipeline in figure 1. Our method processes each input data batch x in two steps, by extracting features $\mathbf{f} = \Phi(x) \in \mathbb{R}^{N \times D}$ by means of a neural network Φ, followed by estimating posterior class probabilities $\mathbf{p} = \Psi(\mathbf{f}) \in \mathbb{R}^{N \times K}$ by means of a linear layer Ψ and softmax non-linearity. We use the symbol $\mathbf{p}' = \Phi(\Psi(x'))$ to denote the class predictions for the same mini-batch x' with data augmentation (random transformations) applied to it. We use the letters D, K and N to denote the feature space dimension, the number of clusters and the mini-batch size. We now detail each component of LSD-C.

3.1 Self-supervised pretraining

As noted in the introduction, traditional clustering methods require handcrafted or pretrained features. More recently, methods such as [27] have combined deep learning and clustering to learn features and clusters together; even so, these methods usually *still* require ad hoc pre-processing steps (e.g. pre-processing such as Sobel filtering [2, 27]) and extensive hyperparameter tuning. In our method we address this issue and avoid bad local minima in our clustering results by initializing our representation by means of self-supervised learning. In practice, this amounts to train our model on a pretext task (detailed in section 4) and then retain and freeze the earlier layers of the model when applying our clustering algorithm. As reported in [4, 16], the features obtained from self-supervised pre-training are linearly separable with respect to typical semantic image classes. This property is particularly desirable in our context and also motivates our major design choice: since the feature space of self-supervised pre-trained network is linearly separable, it is therefore easier to directly operate on it to discriminate between different clusters.

3.2 Pairwise labeling

A key idea in our method is the choice of space where pairwise the data connections are established: we extract pairwise labels at the level of the data representation rather than at the level of the class predictions. The latter is a common design choice, used in DAC [3] to establish pairwise connections between data points and in DEC [51] to match the current label posterior distribution to a sharper version of itself.

Table 1: **Pairwise labeling with adjacency matrices $A_{ij} = \mathbb{1}_{C_{ij}}$ based on different similarities.** τ is the thresholding hyperparameter for L_2, SNE and Cosine. The number of neighbours k is kNN's hyperparameter.

L_2 dist.	SNE	Cosine	kNN
$C_{ij} = \|\mathbf{f}_j - \mathbf{f}_i\|^2 < \tau$	$\frac{\exp(-\|\mathbf{f}_j - \mathbf{f}_i\|^2/T^2)}{H(Z_i, Z_j)} > \tau$	$\frac{\mathbf{f}_j^\top \mathbf{f}_i}{\|\mathbf{f}_j\|\|\mathbf{f}_i\|} > \tau$	$(j \in \text{kNN}(i)) \vee (i \in \text{kNN}(j))$

The collection of pairwise labels between samples in a mini-batch is given by the adjacency matrix A of an undirected graph whose nodes are the samples and whose edges encode their similarities. DAC [3] generates pseudo labels by checking if the output of the network is above or under certain thresholds. The method of [34] proceeds similarly in the semi-supervised setting. In our method, as we work instead at the feature space level, the pairwise labeling step is a separate process from class prediction and we are free to choose any similarity to establish our adjacency matrix A. We denote with $\mathbf{f}_i \in \mathbb{R}^D$ and $\mathbf{f}_j \in \mathbb{R}^D$ the feature vectors for samples i and j in a mini-batch, obtained from the penultimate layer of the neural network Φ. We also use the symbol $A_{ij} \in \{0, 1\}$ to denote the value of the adjacency matrix for the pair of samples (i, j). Next, we describe the different types of pairwise connections considered in this work and summarize them in table 1.

Cosine and L_2 similarity. Let $\tau \in \mathbb{R}^+$ be a threshold hyperparameter and define $C_{ij} = [\cos(\mathbf{f}_j, \mathbf{f}_i) > \tau]$ (cosine) or $C_{ij} = [\|\mathbf{f}_j - \mathbf{f}_i\|^2 < \tau]$ (Euclidean) where cos denotes the dot product between L_2-normalized vectors. We then define $A_{ij} = \mathbb{1}_{C_{ij}}$ where $\mathbb{1}$ is the indicator function. These definitions connect neighbor samples but do not account well for the local structure of the data. Indeed, it is not obvious that the cosine similarity or Euclidean distance would establish good data connections in feature space.

Symmetric SNE. A possible solution to alleviate the previous issue is to use the symmetric SNE similarity introduced in t-SNE [37]. This similarity is based on the conditional probability $p_{j|i}$ of picking j as neighbor of i under a Gaussian distribution assumption. We make a further assumption compared to [37] of an equal variance for every sample in order to speed up the computation of pairwise similarities and define:

$$p_{j|i} = \frac{\exp(-\|\mathbf{f}_j - \mathbf{f}_i\|^2/T^2)}{\sum_{k \neq i} \exp(-\|\mathbf{f}_k - \mathbf{f}_i\|^2/T^2)} = \frac{\exp(-\|\mathbf{f}_j - \mathbf{f}_i\|^2/T^2)}{Z_i}, \tag{1}$$

$$C_{ij} = \frac{p_{j|i} + p_{i|j}}{2} > \tau \quad \Longleftrightarrow \quad \frac{\exp(-\|\mathbf{f}_j - \mathbf{f}_i\|^2/T^2)}{H(Z_i, Z_j)} > \tau. \tag{2}$$

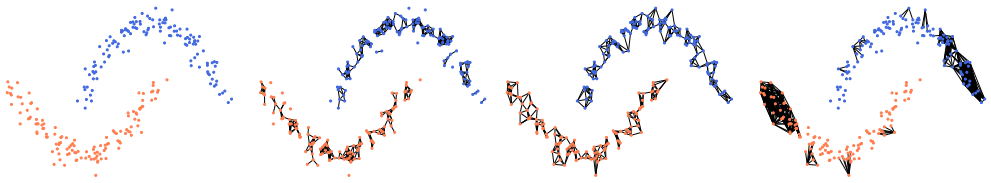
As shown in equation (1), we introduce a temperature hyperparameter $T \in \mathcal{R}^+$ and we call Z_i the partition function for sample i. Then the associated adjacency matrix in equation (2) can be written as a function of the L_2 distance between samples and, in the denominator, of the harmonic mean H of the partition functions. As a result, if sample i or j has many close neighbours, it will reduce the symmetric SNE similarity and possibly prevent a connection between samples i and j. Such a phenomenon is shown on the two moons toy dataset in figure 2.

k-nearest neighbors. We also propose a similarity based on k-nearest neighbours (kNN) [8] where the samples i and j are connected if i is in the k-nearest neighbours of j or if j is in the k-nearest neighbours of i. With this similarity, the hyperparameter is the minimum of neighbours k and not the threshold τ.

3.3 Clustering loss and data composition

Now that we have established pairwise connections between each pair of samples in the mini-batch, we will use the adjacency matrix as target for a binary cross-entropy loss. Denoting with $P(i = j)$ the probability that samples i and j belong to the same cluster, we wish to optimize the clustering loss:

$$L_{\text{clus}} = -\sum_{i,j} A_{ij} \log P(i = j) + (1 - A_{ij}) \log P(i \neq j). \tag{3}$$



(a) Raw data (b) L_2 dist. (c) kNN (d) SNE

Figure 2: **Pairwise connections on the two moons toy data.** From left to right. We apply our algorithm with different connection techniques on a toy dataset shown in (a) where each color represents a class. We use the different connections techniques of table 1 such that there are 650 undirected edges for each similarity. Compared to L_2 distance and SNE, kNN produces neighbourhoods of similar sizes and every sample is connected. SNE captures the local structure of the data: most of the connections are at the external tails of the moons where there are less points.

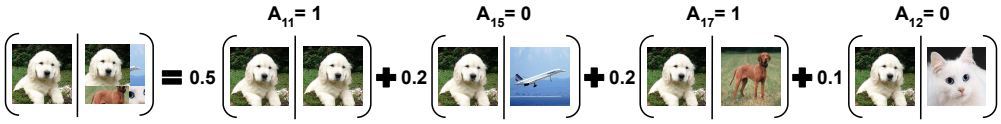


Figure 3: Illustration for equation (6) of a pairwise target between the "pure" image $i = 1$ and the composite image j with $\sigma(j) \in (1, 5, 7, 2)$. In this case, the resulting pairwise target equals 0.7.

The left term of this loss aims at maximizing the number of connected samples (i.e. $A_{ij} = 1$) within a cluster and the right term at minimizing the number of non-connected samples within it (namely, the edges of the complement of the similarity graph $1 - A_{ij} = 1$). Hence the second term prevents the formation of a single, large cluster that would contain all samples.

The next step is to model $P(i = j)$ by using the linear classifier predictions of samples i and j. As seen in equation (4), for a fixed number of clusters K, the probability of samples i and j belonging to the same cluster can be rewritten as a sum of probabilities over the possible clusters. For simplicity, we assume that samples i and j are independent. This way, the pairwise comparison between samples appear only at the loss level and we can thus use the standard forward and backward passes of deep neural networks where each sample is treated independently. By plugging equation (4) in equation (3) and by replacing $\mathbf{p}_j$ with $\mathbf{p}'_j$ to form pairwise comparisons between the mini-batch and its augmented version, we obtain our final clustering loss $\mathcal{L}_{\text{clus}}$:

$$P(i = j) = \sum_{k=1}^{K} P(i = k, j = k) = \sum_{k=1}^{K} P(i = k)P(j = k) = \mathbf{p}_i^\top \mathbf{p}_j, \tag{4}$$

$$L_{\text{clus}} = -\sum_{i,j} A_{ij} \log(\mathbf{p}_i^\top \mathbf{p}'_j) + (1 - A_{ij}) \log(1 - \mathbf{p}_i^\top \mathbf{p}'_j). \tag{5}$$

A similar loss is used in [24] but with supervised pairwise labels to transfer a multi-class classifier across tasks. It is also reminiscent of DAC [3], but differs from the latter because the DAC loss does not contain a dot product between probability vectors but between L_2 normalized probability vectors. Hence DAC optimizes a Bhattacharyya distance whereas we optimize a standard binary cross-entropy loss.

In practice L_{clus} can be used in combination with effective data augmentation techniques such as RICAP [49] and MixUp [55]. These methods combine the images from the minibatch and use a weighted combination of the labels of the original images as new target for the cross-entropy loss. We denote with σ permutation of the samples in the minibatch; RICAP and MixUp require 4 and 2 permutations respectively. RICAP creates a new minibatch of composite images by patching together random crops from the 4 permutations of the original minibatch, whereas MixUp produces a new minibatch by taking a linear combination with random weights from 2 permutations. The new target for a composite image is then obtained by taking a linear combination of the labels in the recombined images, weighted by area proportions in RICAP and the mixing weights in MixUp. These techniques were proposed for the standard supervised classification setting, so we adapt them here to clustering.

In order to do so, we propose to perform a pairwise labeling between the composite images and the raw original images. Both minibatches of original and composite images are fed to the network. Then, as illustrated in figure 3, the pairwise label between a composite image and a raw image is the linear combination of the pairwise labels between the components of both. To sum up, to obtain the pairwise labels between a minibatch and its composite version we just need to extract the adjacency matrix A of the minibatch and then do a linear combination of the adjacency matrix A with the different column permutations σ:

$$L_{\text{clus}} = -\sum_{\sigma} \sum_{i,j} w_{\sigma} A_{i\sigma(j)} \log(\mathbf{p}_i^{\top} \tilde{\mathbf{p}}_j) + (1 - w_{\sigma} A_{i\sigma(j)}) \log(1 - \mathbf{p}_i^{\top} \tilde{\mathbf{p}}_j) \tag{6}$$

Regarding the predicted probability of the 'pure' image i and the composite image j being in the same cluster, we take the dot product between their respective cluster predictions $\mathbf{p}_i$ and $\tilde{\mathbf{p}}_j$.

3.4 Overall loss

The overall loss we optimise is given by

$$L_{\text{tot}} = L_{\text{clus}}(\mathbf{f}, \mathbf{p}, \mathbf{p}') + L_{\text{cons}}(\mathbf{p}, \mathbf{p}'), \tag{7}$$

where

$$L_{\text{cons}} = \frac{\omega(t)}{KN} \sum_{i=1}^{N} \|\mathbf{p}_i - \mathbf{p}'_i\|^2, \tag{8}$$

and $\omega(t) = \lambda e^{-5(1-\frac{t}{T})^2}$ is the ramp-up function proposed in [32, 50] with t the current training step, T the ramp-up length and $\lambda \in \mathbb{R}_+$. L_{cons} is a consistency constraint which requires the model to produce the same prediction $\mathbf{p} \approx \mathbf{p}'$ for an image and an its augmented version. We use it in our method in a similar way as semi-supervised learning techniques [32, 39, 43, 50], i.e. as a regularizer to provide consistent predictions. This differs significantly from clustering methods like IIC [27] and IMSAT [25] where augmentations are used as a main clustering cue by maximizing the mutual information between different versions of an image. Instead, as commonly done in semi-supervised learning, we use the Mean Squared Error (MSE) between predictions as the consistency loss.

4 Experiments

Datasets. We conduct experiments on five popular benchmarks which we use to compare our method against recent state-of-the-art approaches whenever results are available. We use four image datasets and one text dataset to illustrate the versatility of our approach to different types of data. We use MNIST [33], CIFAR 10 [30], CIFAR 100-20 [30] and STL 10 [6] as image datasets. All these datasets cover a wide range of image varieties ranging from 28×28 pixels grey scale digits in MNIST to 96×96 higher resolution images from STL 10. CIFAR 100-20 is redesigned from original CIFAR 100 since we consider only the 20 meta classes for evaluation as common practice [27]. Finally we also evaluate our method on a text dataset, Reuters 10K [35]. Reuters 10K contains 10,000 English news labelled with 4 classes. Each news has 2,000 *tf-idf* features. For all datasets we suppose the number of classes to be known.

Experimental details. We use ResNet-18 [23] for all the datasets except two. For MNIST we use a model inspired from VGG-4 [46], described in [27] and for Reuters 10K we consider a simple DNN of dimension 2000–500–500–2000–4 described in [51]. We train with batch-size of 256 for all experiments. We use SGD optimizer with momentum [48] and weight decay set to 5×10^{-4} for every dataset except for Reuters 10K where we respectively use Adam [29] and decay of 2×10^{-3}. When comparing with other methods in table 2 and table 3, we run our method using 10 different seeds and report average and standard deviation on each dataset to measure the robustness of our method with respect to initialization. As it is common practice [27], we train and test the methods on the whole dataset (this is acceptable given that the method uses no supervision). Further experimental details about data augmentation and training are available in the appendix.

Evaluation metrics. We take the commonly used *clustering accuracy* (ACC) as evaluation metric. ACC is defined as

$$\max_{g \in \text{Sym}(K)} \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\left\{\overline{y}_i = g\left(y_i\right)\right\}, \tag{9}$$

Table 2: **Comparison with other methods.** Our method almost constantly reaches state-of-the-art performances by a large margin. Note that [27] report best results over all the heads while we report results over ten different initializations. This further shows that our method is overall stable and robust to initialization.

	K-means [38]	JULE [53]	DEC [51]	DAC [3]	IIC [27]	Ours
CIFAR 10	22.9	27.2	30.1	52.2	61.7	**81.7** $\pm$ 0.9
CIFAR 100-20	13.0	13.7	18.5	23.8	25.7	**42.3** $\pm$ 1.0
STL 10	19.2	27.7	35.9	47.0	59.6	**66.4** $\pm$ 3.2
MNIST	57.2	96.4	84.3	97.8	**99.2**	98.6 $\pm$ 0.5

Table 3: **Results on Reuters 10K.** Our method performs on average on par with state of the art. Note that for the best seed we reach state-of-the-art results of 83.5%.

	K-means [38]	IMSAT [25]	DEC [51]	VaDE [28]	FINCH [44]	Ours
Reuters 10K	52.4	71.9	72.2	79.8	**81.5**	79.0 $\pm$ 4.3

where $\overline{y}_i$ and y_i respectively denote the ground-truth class label and the clustering assignment obtained by our method for each sample in the dataset. $\mathrm{Sym}\,(K)$ is the group of permutations with K elements and following other clustering methods we use the Hungarian algorithm [31] to optimize the choice of permutation.

4.1 Results on standard benchmarks

We compare our method with the K-means [38] baseline and recent clustering methods. In table 2, we report results on image datasets. We use RotNet [16] self-supervised pre-training for each dataset on all the data available (*e.g* including the unlabelled set in STL-10). Our method significantly outperforms the others by a large margin. For example, our method achieves 81.5% on CIFAR 10, while the previous state-of-the-art method IIC [27] gives 61.7%. On CIFAR 10, our method also outperforms the leading semi-supervised learning technique FixMatch [47] which obtains 64.3% in its one label per class setting. Similarly, on CIFAR 100-20 and STL 10, our method outperforms other clustering approaches respectively by 14.7 and 6.8 points. On MNIST, our method and IIC both achieve a very low error rate around 1%.

These results clearly show the effectiveness of our approach. Unlike the previous state-of-the-art method IIC that requires to apply Sobel filtering and very large batch size during training, our method does not require such preprocessing and works with a common batch size. We also note that our method is robust to different initialization, with a maximum 3.2% of standard deviation across all datasets.

To analyse further the results on CIFAR 10, we can look at the confusion matrix resulting from our model's predictions.We note that most of the errors are due to the 'cat' and 'dog' classes being confused. If we retain only the confident samples with prediction above 0.9 (around 60% of the samples), the accuracy rises to 94%. We assume that the two classes 'cat' and 'dog' are are more difficult to discriminate due to their visual similarity.

In table 3, we also evaluate our method on the document classification dataset Reuters 10K to show its versatility. We compare with different approaches than in table 2 as clustering methods developed for text are seldom evaluated on image datasets like CIFAR and vice versa. Following existing approaches applied to Reuters 10K, we pretrain the deep neural network by training a denoising autoencoder on the dataset [28]. Our method works notably better than the K-means baseline, and is on par with the best results methods FINCH [44] and VaDE [28]. Most notably one run of our method established state-of-the-art results of 83.5%, 2 points above the current best model.

4.2 Ablation studies

In order to analyze the effects of the different components of our method, we conduct a three parts ablation study on CIFAR 10 and CIFAR 100-20. First, we compare the impact of different possible pairwise labeling methods in the feature space. Second, as one of our key contribution is to choose

Table 4: **Ablation study.** We analyse the effect of different pairwise labeling methods but also the impact of where the labeling is done (feature vs prediction space). We also show the paramount importance of data augmentation for clustering some datasets like CIFAR 10.

	Pairwise labeling				Using the pred. space			Data augmentation		
	L_2	Cosine	kNN	SNE	Cosine	kNN	SNE	RICAP	MixUp	None
CIFAR 10	70.2	81.1	**81.7**	81.5	63.7	64.7	**67.0**	**81.7**	75.3	53.7
CIFAR 100-20	26.1	34.4	**42.3**	40.4	20.4	**32.8**	30.4	**42.3**	37.1	35.4

the space where the pairwise labeling is performed, we test doing so at the level of features and predictions (*i.e.* after the linear classifier but before the softmax layer like DEC [51] or DAC [3]). Third, we analyse the importance of data augmentation in clustering raw images. Results are reported in table 4 and discussed next.

Pairwise similarity. We compare, in feature space, pairwise labeling methods based on L_2 distance, cosine similarity, kNN and symmetric SNE as described in table 1. For kNN, we set the number of neighbors k to 20 and 10 for CIFAR 10 and CIFAR 100-20 respectively. For the cosine similarity, we use respectively thresholds 0.9 and 0.95. For the L_2 distance, we ran a grid search between 0 and 2 to find an optimal threshold. For SNE, we set the threshold to 0.01 and the temperature to 1 and 0.5, for CIFAR 10 and CIFAR 100-20 respectively. Further details about the hyperparameters are available in the supplementary. We observe that kNN, SNE and cosine similarity perform very well on CIFAR 10 with values around 81%. It is interesting to note that cosine similarity performs noticeably worse than kNN and SNE on CIFAR 100-20 with around 6 points less. We also notice that L_2 distance performs consistently worse than the other labeling methods. We can conclude that kNN and SNE are the best labeling methods empirically with consistent performance on these two datasets.

Feature space embedding. Instead of using these labeling methods before the linear classifier, we apply them after it. In this case, our overall approach becomes more similar to standard pseudo-labeling methods such as [3, 34, 51], which aim to match the network predictions output with a 'sharper' version of it. We observe that the performance drops considerably for all labeling methods with an average decrease of 16.3 points for CIFAR 10 and 10.6 points for CIFAR 100-20. Hence, this shows empirically that where pseudo labeling is applied plays a major role in clustering effectiveness and that labeling at the feature space level is noticeably better than doing so at the prediction space level.

Data augmentation. We compare RICAP, MixUp, and the case without data composition (denoted as None). As can it can be seen in table 4, data composition is crucial for CIFAR 10 where RICAP and MixUp surpass None by respectively 28 and 22 points. On CIFAR 100-20, the differences are smaller but using data composition still brings a clear improvement with a 5.1 points increase when using RICAP. Interestingly, RICAP clearly outperforms MixUp in both cases.

5 Conclusions

We have proposed a novel deep clustering method, LSD-C. Our method establishes pairwise connections at the feature space level among different data points in a mini-batch. These on-the-fly pairwise connections are then used as targets by our loss to regroup samples into clusters. In this way, our method can effectively learn feature representation together with the cluster assignment. In addition, we also combine recent self-supervised representation learning with our clustering approach to bootstrap the representation before clustering begins. Finally, we adapt data composition techniques to the pairwise connections setting, resulting in a very large performance boost Our method substantially outperforms existing approaches in various public benchmarks, including CIFAR 10/100-20, STL 10, MNIST and Reuters 10K.

Broader Impact

Our method considers the task of unsupervised clustering from unlabeled data. We mainly consider two types of data: images and text document. While we make significant advances in terms of clustering accuracy compared to previous work, we believe the data we used to be at low risk since we consider datasets wide-spread around the community for sometimes decades.

While the data we used are not at risk we believe there is an inherent risk of misuse with clustering particularly when learnt from raw data. As any learning algorithm the clustering also depends on the data bias and could lead to misinformation or misinterpretation of results obtained from our model.

However we believe our method and clustering in general to be of interest for future years as it would reduce the need of heavy data annotations and processing.

6 Acknowledgments

We thank Kevin Scaman for his very useful comments. This work is supported by the EPSRC Programme Grant Seebibyte EP/M013774/1, Mathworks/DTA DFR02620, and ERC IDIU-638009.

References

[1] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *Proc. ICLR*, 2019.

[2] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proc. ECCV*, 2018.

[3] Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep adaptive image clustering. In *Proc. ICCV*, 2017.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv*, 2020.

[5] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Proc. NIPS*, 2016.

[6] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011.

[7] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *PAMI*, 1979.

[8] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 1967.

[9] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. *arXiv*, 2017.

[10] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, 1996.

[11] Dario Fontanel, Fabio Cermelli, Massimiliano Mancini, Samuel Rota Bulò, Elisa Ricci, and Barbara Caputo. Boosting deep open world recognition by clustering. *arXiv*, 2020.

[12] Aude Genevay, Gabriel Dulac-Arnold, and Jean-Philippe Vert. Differentiable deep clustering with cluster size constraints. *arXiv*, 2019.

[13] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proc. ICCV*, 2017.

[14] Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Boosting few-shot visual learning with self-supervision. In *Proc. ICCV*, 2019.

[15] Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Learning representations by predicting bags of visual words. *arXiv*, 2020.

[16] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *Proc. ICLR*, 2018.

[17] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep embedded clustering with local structure preservation. In *IJCAI*, 2017.

[18] Xifeng Guo, Xinwang Liu, En Zhu, and Jianping Yin. Deep clustering with convolutional autoencoders. In *International conference on neural information processing*, 2017.

[19] Philip Haeusser, Johannes Plapp, Vladimir Golkov, Elie Aljalbout, and Daniel Cremers. Associative deep clustering: Training a classification network with no labels. In *German Conference on Pattern Recognition*, 2018.

[20] Kai Han, Sylvestre-Alvise Rebuffi, Sebastien Ehrhardt, Andrea Vedaldi, and Andrew Zisserman. Automatically discovering and learning new visual categories with ranking statistics. In *Proc. ICLR*, 2020.

[21] Kai Han, Andrea Vedaldi, and Andrew Zisserman. Learning to discover novel visual categories via deep transfer clustering. In *Proc. ICCV*, 2019.

[22] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv*, 2019.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016.

[24] Yen-Chang Hsu, Zhaoyang Lv, Joel Schlosser, Phillip Odom, and Zsolt Kira. Multi-class classification without multi-class labels. In *Proc. ICLR*, 2019.

[25] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximizing self-augmented training. In *Proc. ICML*, 2017.

[26] Gabriel Huang, Hugo Larochelle, and Simon Lacoste-Julien. Centroid networks for few-shot clustering and unsupervised few-shot classification. *arXiv*, 2019.

[27] Xu Ji, João F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proc. ICCV*, 2019.

[28] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. *arXiv*, 2016.

[29] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2014.

[30] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Technical report*, 2009.

[31] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 1955.

[32] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *Proc. ICLR*, 2017.

[33] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[34] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, 2013.

[35] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 2004.

[36] Fengfu Li, Hong Qiao, and Bo Zhang. Discriminatively boosted image clustering with fully convolutional auto-encoders. *Pattern Recognition*, 83:161–173, 2018.

[37] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 2008.

[38] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[39] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[40] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Proc. NIPS*, 2019.

[41] Karl Pearson. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London Series A*, 1894.

[42] Sylvestre-Alvise Rebuffi, Sebastien Ehrhardt, Kai Han, Andrea Vedaldi, and Andrew Zisserman. Semi-supervised learning with scarce annotations. In *Proc. CVPR Workshop*, 2020.

[43] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in neural information processing systems*, 2016.

[44] Saquib Sarfraz, Vivek Sharma, and Rainer Stiefelhagen. Efficient parameter-free clustering using first neighbor relations. In *Proc. CVPR*, 2019.

[45] Uri Shaham, Kelly Stanton, Henry Li, Boaz Nadler, Ronen Basri, and Yuval Kluger. Spectralnet: Spectral clustering using deep neural networks. In *Proc. ICLR*, 2018.

[46] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv*, 2014.

[47] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv*, 2020.

[48] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proc. ICML*, 2013.

[49] Ryo Takahashi, Takashi Matsubara, and Kuniaki Uehara. Ricap: Random image cropping and patching data augmentation for deep cnns. In *Asian Conference on Machine Learning*, 2018.

[50] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Proc. NIPS*, 2017.

[51] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *Proc. ICML*, 2016.

[52] Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *Proc. ICML*. JMLR. org, 2017.

[53] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *Proc. CVPR*, 2016.

[54] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S4l: Self-supervised semi-supervised learning. In *Proc. ICCV*, 2019.

[55] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv*, 2017.

LSD-C: Linearly Separable Deep Clusters Supplementary Material

In this supplementary material, we provide our implementation details, the confusion matrices on CIFAR 10 using our method with kNN labeling and some additional ablation studies. We also include the code to run our method on CIFAR 10 together with the network pretrained with RotNet [16].

A Implementation details

Self-supervised pretraining. We train the RotNet [16] (i.e. predicting the rotation applied to the image among four possibilities: 0°, 90°, 180°, and 270°) on all datasets with the same configuration. Following the authors' released code, we train for 200 epochs using a step-wise learning rate starting at 0.1 which is then divided by 5 at epochs 60, 120, and 160.

Main LSD-C models. After the self-supervised pretraining step, following [20] we freeze the first three macro-blocks of the ResNet-18 [23] as the RotNet training provides robust early filters. We then train the last macro-block and the linear classifier using our clustering method. For all the experiments, we use a batch size of 256. We summarize in table 5 all the hyperparameters for the different datasets and labeling methods.

Table 5: **Hyperparameters.** Optimizer, ramp-up function and parameters of different labeling methods on different datasets.

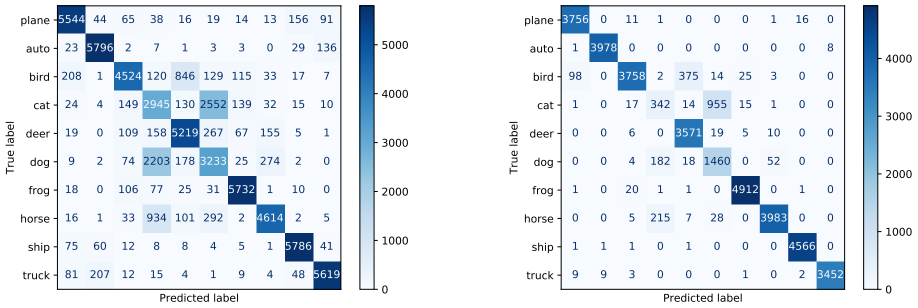
	Optimizer				Ramp-up		Cosine	SNE		kNN
	Type	Epochs	LR steps	LR init	λ	T	τ	τ	Temp	k
CIFAR 10	SGD	220	[140, 180]	0.1	5	100	0.9	0.01	1.0	20
CIFAR 100-20	SGD	200	170	0.1	25	150	0.95	0.01	0.5	10
STL 10	SGD	200	[140, 180]	0.1	5	50	-	0.01	0.5	-
MNIST	SGD	15	-	0.1	5	50	-	-	-	10
Reuters 10K	Adam	75	-	0.001	25	100	-	-	-	5

Data augmentation techniques. We showed in the main paper that data composition techniques like RICAP [49] and MixUp [55] are highly beneficial to our method. For RICAP, we follow the authors' instructions to sample the width and height of crops for each minibatch permutation by using a Beta(0.3, 0.3) distribution. Regarding MixUp, we note that using a Beta(0.3, 0.3) distribution for the mixing weight works better in our case than the Beta(1.0, 1.0) advised for CIFAR 10 in the MixUp paper. Furthermore, we have to decrease the weight decay to 10^{-4} to make MixUp work.

Miscellaneous. Our method is implemented with PyTorch 1.2.0 [40]. Our experiments were run on NVIDIA Tesla M40 GPUs and can run on a single GPU with 12 GB of RAM.

B Confusion matrices on CIFAR 10

In figure 4, we show some confusion matrices on CIFAR 10 to analyse how our clustering method performs on the different classes. We notice that there are 8 confident clusters with a very high clustering accuracy of 94.0% for confident samples. The "dog" and "cat" clusters are not well identified possibly due to a huge intra-class variation of the samples.



(a) All samples (b) Confident samples

Figure 4: **Confusion matrices on CIFAR 10 using our method with kNN labeling.** Figure 4a shows that most of the errors are due to the "cat" and "dog" classes. When taking the samples with prediction above 0.9 (60% of the samples) in Figure 4b, there are less than 2000 predictions on classes "cat" and "dog" whereas there are more than 3500 for each of the other classes. Our method manages to ignore the problematic classes when taking the confident samples. Indeed, the accuracy for confident samples is 94.0%.

C Additional ablation studies

We report in table 6 the results of some additional ablation studies to evaluate the impact of more components of our method. For example, we apply K-means [38] on the feature space of the pretrained RotNet model and we note very poor performance on CIFAR 10 and CIFAR 100-20. We can conclude that before training with our clustering loss, the desired clusters are not yet separated in the feature space. After training with our clustering loss, the clusters can be successfully separated. Moreover, if we only use the clustering loss and drop the consistency MSE loss, the performance decreases on both CIFAR 10 and CIFAR 100-20 by 1.5 and 1.3 points respectively, showing that the MSE provides a moderate but clear gain to our method. Finally, if we replace the linear classifier by a 2-layer classifier (i.e. this corresponds to a non-linear separation of clusters in the feature space), it results in a small improvement on CIFAR 10 but a clear decrease of 1.9 points on CIFAR 100-20. Hence using a linear classifier provides more consistent results across datasets.

Table 6: **Additional ablation studies.** From the first column, we observe that the desired clusters are not yet separated in the feature space after the RotNet pretraining. The second column shows that the MSE consistency loss provides a boost of more than 1 point to our method. Finally, we see that using a non-linear classifier harms the performance on CIFAR 100-20.

	K-means + RotNet	Ours (kNN)	Ours (kNN) w/o MSE	Ours (kNN) w/ non-lin.
CIFAR 10	14.3	81.7	80.2	82.0
CIFAR 100-20	9.1	40.5	39.2	38.6

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	LSD-C: Linearly Separable Deep Clusters
Publication Status	□Published □Accepted for Publication X Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style
Publication Details	Submitted to NeurIPS 2020. SA Rebuffi*, S Ehrhardt*, K Han*, A Vedaldi, A Zisserman. (* denotes equal contribution)

Student Confirmation

Student Name:	Sylvestre-Alvise REBUFFI		
Contribution to the Paper	• Conception of the idea (self-supervision initialization, kNN pairwise labeling, combining the loss with RICAP) • Implementation and doing the experiments (ablations, comparisons on CIFAR and Reuters) • Writing the paper • Preparing the supplementary material		
Signature		Date	24/07/2020

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title:	Prof. Andrea Vedaldi
Supervisor comments	
Signature 	Date

This completed form should be included in the thesis, at the end of the relevant chapter.

Chapter 8

There and Back Again: Revisiting Backpropagation Saliency Methods

This work was presented as a Poster Presentation at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

This paper presents a unifying framework for backpropagation saliency methods. This framework is based on the formula of the gradient of spatially shared weights. In addition, we propose to combine saliency maps at different layers to leverage the high resolution of early layers and the class discriminability of late layers. Finally, we introduce a meta-learning based saliency method to increase the sensitivity of the saliency map to the target class.

There and Back Again: Revisiting Backpropagation Saliency Methods

Sylvestre-Alvise Rebuffi * Ruth Fong* Xu Ji* Andrea Vedaldi
Visual Geometry Group, University of Oxford
{srebuffi, ruthfong, xuji, vedaldi}@robots.ox.ac.uk

Abstract

Saliency methods seek to explain the predictions of a model by producing an importance map across each input sample. A popular class of such methods is based on backpropagating a signal and analyzing the resulting gradient. Despite much research on such methods, relatively little work has been done to clarify the differences between such methods as well as the desiderata of these techniques. Thus, there is a need for rigorously understanding the relationships between different methods as well as their failure modes. In this work, we conduct a thorough analysis of backpropagation-based saliency methods and propose a single framework under which several such methods can be unified. As a result of our study, we make three additional contributions. First, we use our framework to propose NormGrad, a novel saliency method based on the spatial contribution of gradients of convolutional weights. Second, we combine saliency maps at different layers to test the ability of saliency methods to extract complementary information at different network levels (e.g. trading off spatial resolution and distinctiveness) and we explain why some methods fail at specific layers (e.g., Grad-CAM anywhere besides the last convolutional layer). Third, we introduce a class-sensitivity metric and a meta-learning inspired paradigm applicable to any saliency method for improving sensitivity to the output class being explained.

1. Introduction

The adoption of deep learning methods by high-risk applications, such as healthcare and automated driving, gives rise to a need for tools that help machine learning practitioners understand model behavior. Given the highly-parameterized, opaque nature of deep neural networks, developing such methods is non-trivial, and there are many possible approaches. In the basic case, even the predictions of the model itself, either unaltered or after being distilled into a simpler function [11, 3], can be used to shed light on its behaviour.

Saliency is the specific branch of interpretability concerned with determining not what the behaviour of a model is for whole input samples, but which parts of samples contribute the most to that behaviour. Thus by definition, determining saliency - or attribution - necessarily involves reversing the model's inference process in some manner [20]. Propagating a signal from the output layer of a neural network model back to the input layer is one way of explicitly achieving this.

The number of diverse works based on using signal backpropagation for interpretability in computer vision [37, 27, 4, 38] is testimony to the power of this simple principle. Typically, these techniques produce a heatmap for any given input image that ranks its pixels according to some metric of importance for the model's decision. Inspired by the work of [1], we propose to delve deeper into such methods by discussing some of the similarities, differences and potential improvements that can be illustrated.

We begin with presenting a framework that unifies several backpropagation-based saliency methods by dividing the process of generating a saliency map into two phases: *extraction* of the contribution of the gradient of network parameters at each spatial location in a particular network layer, and *aggregation* of such spatial information into a 2D heatmap. GradCAM [27], linear approximation [17] and gradient [29] can all be cast as such processes. Noting that no appropriate technique has yet been proposed for properly aggregating contributions from convolutional layers, we introduce NormGrad, which uses the Frobenius norm for aggregation. We introduce identity layers to allow for the generation of saliency heatmaps at all spatially-grounded layers in the network (i.e. even after ReLU), since NormGrad computes saliency given a parameterised network layer.

We conduct a thorough analysis of backpropagation-based saliency methods in general, with evaluation based on utilising saliency heatmaps for weak localisation. Notably, we conduct an investigation into simple techniques for combining saliency maps taken from different network layers - in contrast to the popular practice of computing maps at the input layer [29] - and find that using a weighted aver-

*indicates equal contribution

age of maps from all layers consistently improves performance for several saliency methods, compared to taking the single best layer. However, not all layers are equally important, as we discover that models optimised on datasets such as ImageNet [26] and PASCAL VOC [7] learn features that become increasingly *class insensitive* closer towards the input layer. This provides an explanation for why Grad-CAM [27] produces unmeaningful saliency heatmaps at certain layers earlier than the last convolutional layer, as the sensitivity of the gradient to class across spatial locations is eliminated by Grad-CAM's spatial gradient averaging, meaning such layers are devoid of class sensitive signals from which to form saliency heatmaps.

Finally, building off [20, 1], we introduce a novel metric for quantifying the class sensitivity of a saliency method, and present a meta-learning inspired paradigm that increases the class sensitivity of any method by adding an inner SGD step into the computation of the saliency heatmap.

2. Related work

Saliency methods. Our work focuses on backpropagation-based saliency methods; these techniques are computationally efficient as they only require one forward and backward pass through a network. One of the earliest methods was [29] which visualised the gradient at the input with respect to an output class being explained. Several authors have since proposed adaptations in order to improve the heatmap's visual quality. These include modifying the ReLU derivatives (Deconvnet [37], guided backprop [32]) and averaging over randomly perturbed inputs (SmoothGrad [31]) to produce masks with reduced noise. Several works have explored visualizing saliency at intermediate layers by combining information from activations and gradients, notably CAM [39], Grad-CAM [27], and linear approximation (a.k.a. gradient $\times$ input) [17]. Conservation-preserving methods (Excitation Backprop [38], LRP [4], and DeepLIFT [28]) modify the backward functions of network layers in order to "preserve" an attribution signal such that it sums to one at any point in the network. Reference-based methods average over attributions from multiple interpolations [33] between the input and a non-informative (e.g. black) reference input or use a single reference input with which to compare a backpropagated attribution signal [28].

Although we focus on backpropagation-based methods, another class of methods studies the effects that perturbations on the input induce on the output. [37] and [24] generate saliency maps by weighting input occlusion patterns by the induced changes in model output. [10, 9, 15] learn for saliency maps that maximally impact the model, and [5] trains a model to predict effective maps. LIME [25] learns linear weights that correspond to the effect of including or excluding (via perturbation) different image patches in an image. Perturbation-based approaches have also been used to perform object localisation [30, 35, 34].

Assessing and unifying saliency methods. A few works have studied if saliency methods have certain desired sensitivities (e.g., to specific model weights [1] or the output class being explained [20]) and if they are invariant to unmeaningful factors (e.g., constant shift in input intensity [16]). [20] showed that gradient [29], deconvnet [37], and guided backprop [32] tend to produce class insensitive heatmaps. [1] introduced sanity check metrics that measure how sensitive a saliency method is to model weights by reporting the correlation between a saliency map on a trained model vs. a partially randomized model.

Other works quantify the utility of saliency maps for weak localization [38, 10] and for impacting model predictions. [38] introduced Pointing Game, which measures the correlation between the maximal point extracted from a saliency map with pixel-level semantic labels. [10] extracts bounding boxes from saliency maps and measures their agreement with ground truth bounding boxes. [36] evaluates attribution methods using relative feature importance. [22] proposes a dataset designed for measuring visual explanation accuracy. [4, 24, 15] present variants of a perturbation-based evaluation metric that measures the impact of perturbing (or unperturbing for [15]) image patches in order of importance as given by a saliency map. However, these perturbed images are outside the training domain; [13] mitigates this by measuring the performance of classifiers re-trained on perturbed images (i.e., with 20% of pixels perturbed).

To our knowledge, the only work that has been done to unify saliency methods focuses primarily on "invasive" techniques that change backpropagation rules. The α-LRP variant [4] and Excitation Backprop [38] share the backpropagation rule, and DeepLIFT [28] is equivalent to LRP when 0 is used as the reference activation throughout a network. [19] unifies a few methods (e.g., LIME [25], LRP [4], DeepLIFT [28]) under the framework of additive feature attribution.

3. Method

Preliminaries. Consider a training set $\mathcal{D}$ of pairs $(\boldsymbol{x}, \boldsymbol{y})$ where $\boldsymbol{x} \in \mathbb{R}^{3\times H\times W}$ are (color) images and $\boldsymbol{y} \in \{1, \dots, C\}$ their labels. Furthermore, let $\boldsymbol{y} = \Phi_\theta(\boldsymbol{x})$ be the output of a neural network model whose parameters θ are optimized using the cross-entropy loss ℓ to predict labels from images.

3.1. Extract & Aggregate framework

In most methods, the saliency map is obtained as a function of the network activations, computed in a forward pass,

and information propagated from the output of the network back to its input using the backpropagation algorithm. While some methods modify backpropagation in some way, here we are interested in those, such as gradient, linear approximation, and all variants of our proposed NormGrad saliency method, that do not.

In order to explain these "non-invasive" methods, we suggest that their saliency maps can be interpreted as a measure of how much the corresponding pixels contribute to changing the model parameters during a training step. We then propose a principled two-phase framework capturing this idea. In the extraction phase, a method isolates the contribution to the gradient from each spatial location. We use the fact that the gradient of spatially shared weights can be written as the sum over spatial locations of a function of the activation gradients and input features. In the aggregation phase, each spatial summand is converted into a scalar using an aggregation function, thus resulting in a saliency map.

Algorithm 1 Extract & Aggregate

1. **Extract.** Compute spatial contributions to the summation of the gradient of the weights.
   - Choose a layer whose parameters are shared spatially (layers from table 1).
   - Alternatively, insert an identity layer (section 3.1.2) at the targeted location.
2. **Aggregate.** Transform these spatial contributions into saliency maps using an aggregation function.
   - Norm: NormGrad (ours)
   - Voting/Summing: linear approximation [17]
   - Filtering: GradCAM [27], selective NormGrad (ours)
   - Max: Gradient backprogation [29]

3.1.1 Phase 1: Extract

We first choose a target layer in the network at which we plan to compute a saliency map. Assuming that the network is a simple chain[1], we can write $L = \ell \circ \Phi = h \circ k_{\boldsymbol{w}} \circ q$, where $k_{\boldsymbol{w}}$ is the target layer parameterised by $\boldsymbol{w}$, h is the composition of all layers that follow it (including loss ℓ), and q is the composition of layers that precede it. Then, $\boldsymbol{x}^{in} = q(\boldsymbol{x}) \in \mathbb{R}^{K\times H\times W}$ denotes the input to the target layer, and its output is given by $\boldsymbol{x}^{out} = k_{\boldsymbol{w}}(\boldsymbol{x}^{in}) \in \mathbb{R}^{K'\times H'\times W'}$.

[1]Other topologies are treated in the same manner, but the notation is more complex.

Layer	Spatial contribution	Size at each location
Bias	$\boldsymbol{g}_u^{out}$	vector: K'
Scaling	$\boldsymbol{g}_u^{out} \odot \boldsymbol{x}_u^{in}$	vector: K'
Conv $N \times N$	$\boldsymbol{g}_u^{out} \boldsymbol{x}_{u,N\times N}^{in}{}^\top$	matrix: $K' \times N^2K$

Table 1. Formulae and sizes of the spatial contributions to the gradient of the weights for layers with spatially shared parameters. $\odot$ denotes the elementwise product and $\boldsymbol{x}_{u,N\times N}^{in}$ is the N^2K vector obtained by unfolding the $N \times N$ patch around the target location.

Convolutional layers with general filter shapes. For convolutions with an $N \times N$ kernel size, we can re-write the convolution using the matrix form:

$$\mathbf{X}^{out} = \mathbf{W}\mathbf{X}_{N\times N}^{in} \tag{1}$$

where $\mathbf{X}^{out} \in \mathbb{R}^{K'\times HW}$ and $\mathbf{W} \in \mathbb{R}^{K'\times N^2K}$ are the output and filter tensors reshaped as matrices and $\mathbf{X}_{N\times N}^{in} \in \mathbb{R}^{N^2K\times HW}$ is a matrix whose column $\boldsymbol{x}_{u,N\times N}^{in} \in \mathbb{R}^{N^2K}$ contains the unfolded patches at location u of the input tensor to which filters are applied.[2] Then, the gradient w.r.t. the filter weights W is given by

$$\frac{dL}{d\mathbf{W}} = \sum_{u\in\Omega} \frac{d}{d\mathbf{W}} \left\langle \boldsymbol{g}_u^{out}, \mathbf{W}\boldsymbol{x}_{u,N\times N}^{in} \right\rangle = \sum_{u\in\Omega} \boldsymbol{g}_u^{out} \boldsymbol{x}_{u,N\times N}^{in}{}^\top, \tag{2}$$

where $\boldsymbol{g}_u^{out} = dh/d\boldsymbol{x}_u^{out}$ is the gradient of the "head" of the network. Thuis, for the convolutional layer case, the spatial summand is an outer product of two vectors; thus, the spatial contribution at each location to the gradient of the weights is a matrix of size $K' \times N^2K$.

Other layer types. Besides convolutional layers, bias and scaling layers also share their parameters spatially. In modern architectures, these are typically used in batch normalization layers [14]. We denote $\boldsymbol{b}, \boldsymbol{\alpha} \in \mathbb{R}^K$ as the parameters for the bias and scaling layers respectively. They are defined respectively as follows:

$$x_{ku}^{out} = x_{ku}^{in} + b_k, \qquad x_{ku}^{out} = \alpha_k x_{ku}^{in}.$$

Then, the gradients w.r.t. parameters are given by

$$\frac{dL}{d\boldsymbol{b}} = \sum_{u\in\Omega} \boldsymbol{g}_u^{out}, \qquad \frac{dL}{d\boldsymbol{\alpha}} = \sum_{u\in\Omega} \boldsymbol{g}_u^{out} \odot \boldsymbol{x}_u^{in}. \tag{3}$$

where $\odot$ is the elementwise product. For these two types of layer, the spatial summand is a vector of size K, the number of channels. Table 1 summarizes the spatial contributions to the gradients for the different layers.

[2]This operation is called `im2row` in MATLAB or `unfold` in PyTorch.

3.1.2 Virtual identity layer

So far, we have only extracted spatial gradient contributions for layers that share parameters spatially. We will now extend our summand extraction technique to any location within a CNN by allowing the insertion of a virtual identity layer at the target location. This layer is a conceptual construction that we introduce to derive in a rigorous and unified manner the equations employed by various methods to compute saliency.

We are motivated by the following question: if we were to add an identity operator at a target location, how should this operator's parameters be changed? A virtual identity layer is a layer which shares its parameters spatially and is set to the identity. Hence, it could be any of the layer from table 1, like a bias or a scaling layer; then, $b_k = 0$ or $c_k = 1$ respectively for $k \in \{1, \dots, K\}$. It could also be a $N \times N$ convolutional layer with filter bank $\mathbf{W} \in \mathbb{R}^{K \times N^2 K}$ such that $w_{kk'ij} = \delta_{k,k'}\delta_{i,0}\delta_{j,0}$, where $\delta_{a,b}$ is the Kronecker delta function[3], for $(i, j) \in \{-\frac{N-1}{2}, \dots, \frac{N-1}{2}\}$.

Because of the nature of the identity, this layer does not change the information propagated in either the forward or backward direction. Conceptually, it is attached to the part of the model that one wishes to inspect as shown in fig. 1. This layer is never "physically" added to the model (i.e., the model is not modified); its "inclusion" or "exclusion" simply denotes whether we are using input or output activations ($\boldsymbol{x}^{in}$ or $\boldsymbol{x}^{out}$). Indeed the backpropagated gradient $\boldsymbol{g}^{out}$ at the output of the identity as shown in fig. 1 is the gradient that would have been at the output of the layer preceding the identity. This construction allows to examine activations and gradients at the same location (e.g., $\boldsymbol{x}^{out}$ and $\boldsymbol{g}^{out}$) as most existing saliency methods do. We now can use the formulae defined in section 3.1.1 when analyzing the gradient of the weights of this identity layer. For example, if we consider an identity scaling layer, the spatial contribution would be $\boldsymbol{g}_u^{out} \odot \boldsymbol{x}_u^{out}$. Or, for a $N \times N$ identity convolutional layer, it would be $\boldsymbol{g}_u^{out} \boldsymbol{x}_{u,N\times N}^{out}{}^{\top}$.

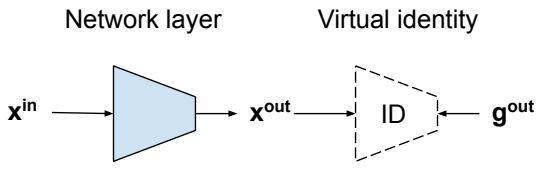


Figure 1. **Virtual identity.** Here, we visualize inserting an identity after a specific layer in the network for saliency computation purposes. The gradient coming from later stages of the network that is the gradient at output of the network layer, is now also the gradient at the output of the identity, $\boldsymbol{g}^{out}$.

[3]Kronecker delta function: $\delta_{a,b} = 1$ if $a = b$; otherwise, $\delta_{a,b} = 0$.

3.1.3 Phase 2: Aggregate

Following the extraction phase (section 3.1.1), we have the local contribution at each spatial location to the gradients of either an existing layer or a virtual identity layer. Each spatial location is associated with a vector or matrix (table 1). In this section, we describe different aggregation functions to map these vectors or matrices to a single scalar per spatial position. We drop the spatial location u in the notations.

Understanding existing saliency methods. One possible aggregation function is the sum of the elements in a vector. When combined with a virtual scaling identity layer (section 3.1.2), we obtain the linear approximation method [17]: $\sum_k g_k^{out} x_k^{out}$.The contributions from the scaling identity encode the result of channels changing (after the gradient is applied) at each location; thus, the sum aggregation function acts as a voting mechanism. The resulting saliency map highlights the locations that would be most impacted if following through with the channel updates.

Aggregation functions can also be combined with element-wise filtering functions (e.g., the absolute value function). Another aggregation function takes the maximum absolute value of the vector: $\mathrm{maxabs}(\boldsymbol{x}) = \max_k |x_k|$. If we combine a virtual bias identity layer in phase 1, which gives $\boldsymbol{g}^{out}$ as the spatial contribution, with the maxabs function for aggregation, we obtain at each spatial location the gradient [29] method: $\max_k |g_k^{out}|$.

As for CAM [39] and Grad-CAM [27], we cannot directly use the spatial contributions extracted at each location because they spatially average $\boldsymbol{g}^{out}$.[4] However, for architectures that use global average pooling followed after their convolutional layers (e.g., ResNet architectures), $\boldsymbol{g}_u^{out} = \bar{\boldsymbol{g}}^{out}$. Then, CAM and Grad-CAM and CAM can be viewed as combining a virtual scaling identity layer from phase 1 with summing and positive filtering (i.e., $\mathrm{filter}_+(x) = (x)_+$) functions for aggregation.

NormGrad. The sum and max functions have clear interpretations when using bias or scaling identity layers; however, they cannot be easily transported to convolutional identity layers as interactions between input channels can vary depending on the output channel and are represented as a matrix, not as a vector as are the case for scaling and bias layers. Thus, we would like to have an aggregation function that could be used to aggregate any type of spatial contribution, regardless of its shape.

Using the norm function satisfies this criterion, for example the L^2 norm when dealing with vectors and the Frobenius norm for matrices. We note that the matrices obtained at each location for convolutional layers are the outer

[4]Because CAM global average pooling + one fully connected layer, $\boldsymbol{g}^{out}$ is equal to the fully connected weights.

Phase 1	Phase 2	Saliency map
Bias IDL	Max	$\max_k g_k^{out}$
Scaling IDL	Sum	$\sum_k g_k^{out} x_k^{out}$
Scaling IDL	F + N	$\|(g^{out} \odot x^{out})_+\|$
Conv 1×1 IDL	Norm	$\|\boldsymbol{g}^{out}\|\|\boldsymbol{x}^{out}\|$
Real Conv 3×3	Norm	$\|\boldsymbol{g}^{out}\|\|\boldsymbol{x}^{in}_{3\times3}\|$

Table 2. **Combining layers and aggregation functions for saliency.** g_k^{out} and x_k^{out} are tensor slices for channel k and contain only spatial information. IDL denotes an identity layer. F + N is positive filtering + norm. From top to bottom, the rows correspond to the following saliency methods: 1., gradient, 2., linear approximation, 3., NormGrad selective, 4., NormGrad, and 5., NormGrad without the virtual identity trick.

products of two vectors. For such matrices, the Frobenius norm is equal to the product of the norms of the two vectors. For example, for an existing 1×1 convolutional layer, we consider the saliency map given by $\|\boldsymbol{g}^{out}\|\|\boldsymbol{x}^{in}\|$. We call this class of saliency methods derived by using norm as an aggregation function, "NormGrad".

Saliency maps from the NormGrad outlined above are not as class selective as other methods because they highlight the spatial locations that contribute the most to gradient of the weights, both positively and negatively. One way to introduce class selectivity is to use positive filtering before applying the norm. If we apply norm and positive filtering aggregation to a scaling identity layer, the resulting saliency map is given by $\|(\boldsymbol{g}^{out} \odot \boldsymbol{x}^{out})_+\|$. Throughout the rest of the paper, we call this variant "selective NormGrad".

4. Experiments

In this section we quantitatively and qualitatively evaluate the performance of a large number of backpropagation-based saliency methods. Code for our framework is released at http://github.com/srebuffi/revisiting_saliency. Additional experiments such as image captioning visualizations are included in the appendix.

Experimental set-up. Unless otherwise stated, saliency maps are generated on images from either the PASCAL VOC [7] 2007 test set or the ImageNet [6] 2012 validation set for either VGG16 [29] or Resnet50 [12]. For PASCAL VOC, we use a model pre-trained on ImageNet with fine-tuned fully connected layers on PASCAL VOC. We use [9]'s TorchRay interpretability package to generate saliency methods for all other saliency methods besides our NormGrad and meta-saliency methods as well as to evaluate saliency maps on [38]'s Pointing Game (see [38] for more details). For all correlation analysis, we compute the Spearman's correlation coefficient [21] between saliency maps that are upsampled to the input resolution: 224×224.

4.1. Justifying the virtual identity trick.

In order to justify our use of the virtual identity trick, we compare NormGrad saliency maps generated at 1×1 and 3×3 convolutional layers both with and without the virtual identity trick (4th and 5th rows respectively in table 2) for VGG16 and ResNet50. We first computed the correlations between saliency maps generating with and without the virtual identity trick. We found that the mean correlation across $N = 50k$ ImageNet validation images was over 95% across all layers we tested. We also evaluated their performance on the Pointing Game [38] and found that the mean absolute difference in pointing game accuracy was $0.53\% \pm 0.62\%$ across all layers we tested (see supp. for more details and full results). This empirically demonstrates that using the virtual identity trick closely approximates the behavior of calculating the spatial contributions for the original convolutional layers.

4.2. Combining saliency maps across layers.

Training linear classifiers on top of intermediate representations is a well-known method for evaluating the learned features of a network at different layers [2]. This suggests that saliency maps, too, may have varying levels of meaningfulness at different layers.

We explore this question by imposing several weighting methods for combining the layer-wise saliency maps of ResNet50 and VGG16 and measuring the resulting performance on PASCAL VOC Pointing Game on both the "difficult" and "all" image sets. To determine the weight γ_j for a given layer j out of J layers in a network we use:

1. **Feature spread**. Given the set of feature activations at layer j, $\boldsymbol{x}^i$ for $i \in M$ input images sampled uniformly across classes, compute the spatial mean $\bar{\boldsymbol{x}}^i = \sum_{u \in \Omega} \boldsymbol{x}^i_u$. $\gamma_j = \frac{1}{M} \sum_{i=1}^{M} |\bar{\boldsymbol{x}}^i - \bar{\boldsymbol{x}}^\mu|$ where $\bar{\boldsymbol{x}}^\mu = \frac{1}{M} \sum_{i=1}^{M} |\bar{\boldsymbol{x}}^i|$.
2. **Classification accuracy**. Given the set of feature activations at layer j, $\boldsymbol{x}^i$ for $i \in M$ input images sampled uniformly across classes, train a linear layer Ψ using image labels $\boldsymbol{y}^i$. $\gamma_j = \frac{1}{M} \sum_{i=1}^{M} \delta_{\Psi(\boldsymbol{x}^i), \boldsymbol{y}^i}$ where $\delta_{\Psi(\boldsymbol{x}^i), \boldsymbol{y}^i} = 1$ if $\Psi(\boldsymbol{x}^i) = \boldsymbol{y}^i$, 0 otherwise.
3. **Linear interpolation**. $\gamma_j = \frac{j}{J}$.
4. **Uniform**. $\gamma_j = \frac{1}{J}$.

To obtain a combined saliency map $\boldsymbol{M}$ from maps $\boldsymbol{m}_j$ from each layer j, the weights are normalised and applied either additively, $\boldsymbol{M} = \sum_{j=0}^{J} \gamma_j \cdot \boldsymbol{m}_j$, or with a product, $\boldsymbol{M} = \prod_{j=0}^{J} \boldsymbol{m}_j^{\gamma_j}$ (see fig. 2 for visualization).

Table 3 shows that weighted saliency maps produce the best overall performance in all four key test cases, which is surprising as there were only four weight schemes tested

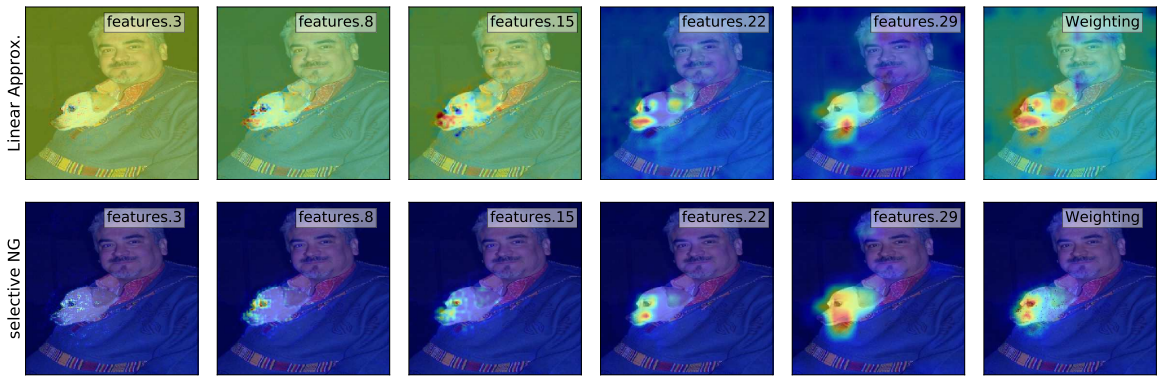


Figure 2. **Saliency maps at different network depths and as a weighted combination.** Linear approximation vs. selective NormGrad saliency maps of VGG16 on VOC2007. The first 5 images of each row correspond to different depths whereas the last one is a weighted product combination (using classification accuracy weights) of the first saliency maps. We observe that the weighted version produces more fined grained maps for both methods.

	All				Difficult			
	Resnet50		VGG16		Resnet50		VGG16	
	b.s.	b.w.	b.s.	b.w.	b.s.	b.w.	b.s.	b.w.
CEB	**90.7**	88.6	82.1	78.2	82.2	82.2	67.0	65.2
EB	84.5	83.1	77.5	75.7	71.5	71.3	57.8	56.1
GC	90.3	90.5	**86.6**	80.6	**82.3**	82.6	74.0	67.8
Gd	83.9	83.3	**86.6**	82.7	70.3	69.4	66.4	67.4
Gds	80.0	77.4	76.8	77.2	62.9	59.5	57.9	59.4
Gui	82.3	81.0	75.8	74.4	67.9	63.4	53.0	51.6
LA	90.2	**91.2**	86.4	**86.9**	81.9	**83.8**	**74.5**	**77.4**
NG	84.6	83.5	81.9	81.8	72.2	70.2	64.8	64.6
sNG	87.4	88.7	86.0	86.8	77.0	79.1	72.6	74.5

Table 3. **Pointing game results on VOC07.** b.s. and b.w. stand for best single layer and best weighted combination. (C)EB: (Contrastive) Excitation Backprop, GC: GradCAM, Gd(s): Gradient (sum), Gui: guided backprop, LA: linear approximation, (s)NG: (selective) NormGrad.

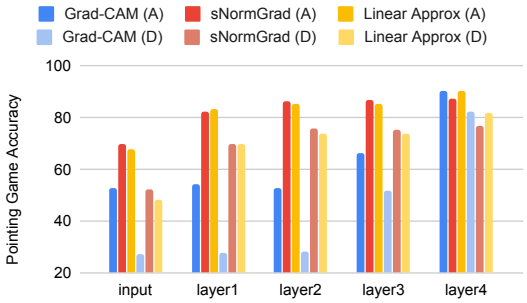


Figure 3. **Select Pointing Game results.** Results for ResNet50 on VOC07 at different network depths (A: all images; D: difficult subset). Grad-CAM performs worse at every layer except the last conv layer and lower than pointing at the center (all: 69.6%; diff: 42.4%) at most layers.

(in addition to best single layer), none of which were explicitly optimised for use with saliency maps. Our results strongly indicate that linear approximation in particular benefits from combining maps from different layers, and linear approximation with layer combination consistently produces the best performance overall and beats far more complex methods at weak localisation using a single forward-backward pass (see supp. for full results).

Note that the feature spread and classification accuracy metrics can both be used as indicators of class sensitivity (section 4.3). This is because if feature activations are uniform for images sampled across classes, it is not possible for them to be sensitive to - or predictive of - class, and the classification accuracy metric is an explicit quantisation of how easily features can be separated into classes. We observe from the computed weights that both metrics generally increase with layer depth (see supp.).

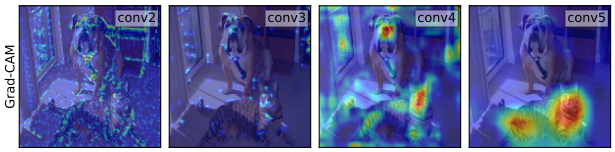


Figure 4. **Grad-CAM failure mode.** Grad-CAM saliency maps w.r.t. "tiger cat" at different depths of VGG16. Grad-CAM only works at the last conv layer (rightmost col).

Explanation of Grad-CAM failure mode. Figure 4 showed qualitatively that Grad-CAM does not produce meaningful saliency maps at any layer except the last convolutional layer, which is confirmed by Grad-CAM's Pointing Game results at earlier layers. Class sensitivity - as measured by our weighting metrics - increasing with layer depth offers an explanation for this drop in performance. Since

Grad-CAM spatially averages the backpropagated gradient before taking a product with activations, each pixel location in the heatmap receives the same gradient vector (across channels) *irrespective* of the image content contained within its receptive field. Thus, if the activation map used in the ensuing product is also not class selective - firing on both dogs and cats for example, fig. 4 - the saliency map cannot be. On the other hand, methods that do not spatially average gradients such as NormGrad (fig. 3) can rely on gradients that are free to vary across the heatmap with underlying class, increasing the class sensitivity of the resulting saliency map.

4.3. An explicit metric for class sensitivity

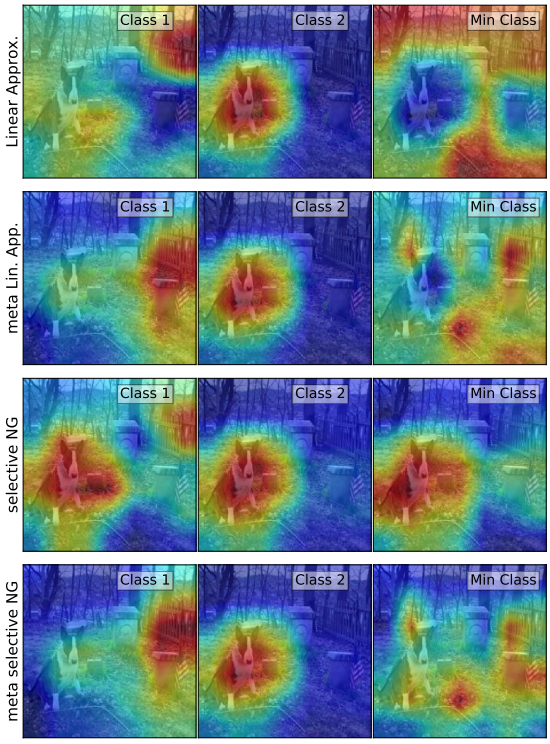


Figure 5. **Class sensitivity with and without meta-saliency.** Min class saliency maps that use meta-saliency (row 2 and 4, right col) are less informative than those that don't use meta-saliency (rows 1 and 3, right col). Class 1 is the ground truth class (fence), class 2 is the maximally predicted class (Cardigan Welsh corgi), min class is the minimally predicted class (black widow spider).

[20] qualitatively shows that early backprop-based methods (e.g., gradient, deconvnet, and guided backprop) are not sensitive to the output class being explained by showing that saliency maps generated w.r.t. different output classes and gradient signals appear visually indistinguishable. Thus, similar to [1], we introduce a sanity check to measure a saliency method's output class sensitivity. We compute the correlation between saliency maps w.r.t. to output class predicted with highest confidence (max class) and that predicted with lowest confidence (max and min class respectively) for $N = 1000$ ImageNet val. images (1 per class).

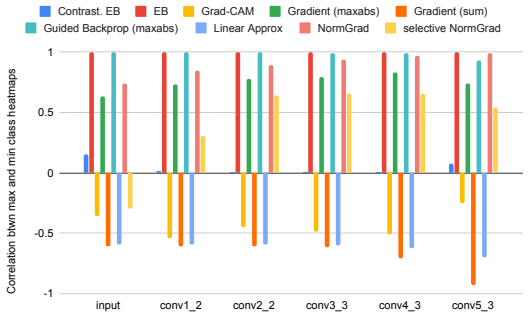


Figure 6. **Class sensitivity of saliency methods.** This plot shows the correlation between VGG16 saliency maps computed w.r.t. to the maximally and minimally predicted class (closer to zero is better).

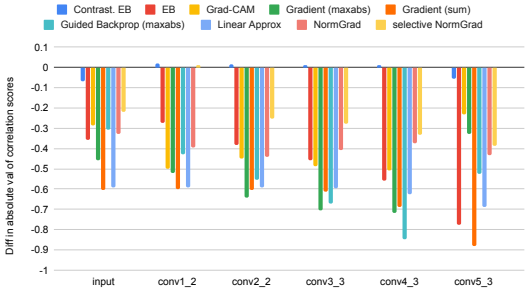


Figure 7. **Meta-saliency improves class sensitivity for all saliency methods.** Using meta-saliency yields weaker correlations between the saliency maps w.r.t. the maximally and minimally predicted output class compared to not using meta-saliency (lower is better).

We would expect saliency maps w.r.t. the max class to be visually salient while those w.r.t. to the min class to be uninformative (because the min class is not in the image). Thus, we desire the correlation scores to be close to zero.

Figure 6 shows results for various saliency methods. We observe that excitation backprop and guided backprop yield correlation scores close to 1 for all layers, while contrastive excitation backprop yields scores closest to 0. Furthermore, methods using sum aggregation (e.g., gradient [sum], linear approx, and Grad-CAM) have negative scores (i.e., their max-min-class saliency maps are anti-correlated). This is because sum aggregation acts as a voting mechanism; thus, these methods reflect the fact that the network has learned anti-correlated relationships between max and min classes.

4.4. Meta-saliency analysis

As a general method for improving the sensitivity of saliency heatmaps to the output class used to generate the

gradient, we propose to perform an inner SGD step before computing the gradients with respect to the loss. This way we can extend any saliency method to second order gradients. This is partly inspired by the inner step used in, for example, few shot learning [8] and architecture search [18]. We want to minimize:

$$L(\theta, x) = \ell(\theta - \epsilon \nabla_\theta \ell(\theta, x), x). \tag{4}$$

We take $\epsilon \ll 1$ to use a Taylor expansion of this loss at θ and we now have the resulting approximated loss:

$$L(\theta, x) \approx \ell(\theta, x) - \epsilon \|\nabla_\theta \ell(\theta, x)\|^2. \tag{5}$$

As done in the previous section, we can now take the gradient of the loss with respect to the parameters θ:

$$\nabla_\theta L(\theta, x) \approx \nabla_\theta \ell(\theta, x) - 2\epsilon \nabla^2_\theta \ell(\theta, x) \nabla_\theta \ell(\theta, x). \tag{6}$$

Using a finite difference scheme of step h as in [23], we can approximate the hessian-vector product by:

$$\nabla^2_\theta \ell(\theta, x) \nabla_\theta \ell(\theta, x) = \frac{\nabla_\theta \ell(\theta, x) - \nabla_{\theta^-} \ell(\theta^-, x)}{h} + O(h).$$

where $\theta^- = \theta - h \nabla_\theta \ell(\theta, x)$. We chose on purpose a backward finite difference such that two terms cancel each other when taking $h = 2\epsilon$ and we get:

$$\nabla_\theta L(\theta, x) \approx \nabla_{\theta'} \ell(\theta', x).$$

where $\theta' = \theta - 2\epsilon \nabla_\theta \ell(\theta, x)$ corresponds to one step of SGD of learning rate 2ϵ. We notice that if we take $\epsilon \to 0$, this formula boils back down to the original gradient of the weights without meta step. We further note that this meta saliency approach only requires one more forward-backward pass compared to usual saliency backpropagation methods.

Conversely, if we would like to get an importance map that highlights the degradation of the model's performance, we should add an inner step with gradient ascent within the loss. Hence by minimizing the resulting loss $-\ell(\theta + \epsilon \nabla_\theta \ell(\theta, x), x)$, we get the same formula for the gradients of the weights but with $\theta' = \theta + 2\epsilon \nabla_\theta \ell(\theta, x)$.

We hypothesize that applying meta-saliency to a saliency method should decrease correlation strength because allowing the network to update one SGD step in the direction of the min class should "destroy" the informativeness of the resulting saliency map. We use a learning rate $\epsilon = 0.001$ for the class sensitivity quantitative analysis. Figure 5 shows qualitatively that this appears to be the case: without meta-saliency, selective NormGrad and linear approximation yield max (class 2) and min class heatmaps that are highly positively and negatively correlated respectively. However, when meta-saliency is applied, the min class saliency map appears more random. Figure 7 shows results comparing the max-min class correlation scores with and without meta-saliency. These results demonstrate that meta-saliency decreases max-min class correlation strength for nearly all saliency methods and suggest that meta-saliency can increase the class sensitivity for any saliency method.

4.5. Model weights sensitivity

[1] shows that some saliency methods (e.g., Guided Backprop in particular) are not sensitive to model weights as they are randomized in a cascading fashion from the end to the beginning of the network. Figure 8 shows qualitatively that, by the late conv layers, saliency maps for linear approximation and selective NormGrad are effectively scrambled (top two rows). It also highlights that, because meta-saliency increases class selectivity and is allowed to take one SGD in the direction of the target class, it takes relatively longer (i.e., more network depth) to randomize a meta-saliency heatmap (bottom row and see appendix).

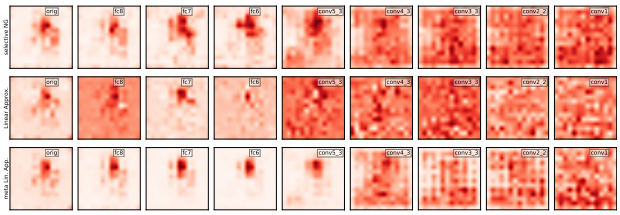


Figure 8. **Model weights sensitivity.** Sanity check by randomizing VGG16 model weights in a cascading fashion for the "Irish terrier" image from [1]. Top row: selective NormGrad, middle row: linear approximation, bottom row: linear approximation with meta-saliency (lower correlation with orig heatmap [leftmost col] is better). All methods look random after conv4_3. By comparing the last two rows at conv5_3, we see the that meta-saliency enforces more class sensitivity than the non-meta variant.

5. Conclusions

We introduced a principled framework based on the contribution of each spatial location to the weights' gradient. This framework unifies several existing backpropagation-based methods and allowed us to systematically explore the space of possible saliency methods. We use it for example to formulate NormGrad, a novel saliency method. We also studied how to combine saliency maps from different layers, discovering that it can consistently improve weak localization performance and produce high resolution maps. Finally, we introduced a class-sensitivity metric and proposed meta-saliency, a novel paradigm applicable to any existing method to improve sensitivity to the target class.

6. Acknowledgments

This work is supported by Mathworks/DTA, the Rhodes Trust (M.P.), EPSRC AIMS CDT and ERC 638009-IDIU.

References

[1] Julius Adebayo, Justin Gilmer, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Proc. NeurIPS*, 2018. 1, 2, 7, 8

[2] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv*, 2016. 5

[3] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Proc. NIPS*, 2014. 1

[4] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 2015. 1, 2

[5] Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. In *Proc. NIPS*, 2017. 2

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*, 2009. 5

[7] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 2015. 2, 5

[8] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. ICML*, 2017. 8

[9] Ruth Fong, Mandela Patrick, and Andrea Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *Proc. ICCV*, 2019. 2, 5

[10] Ruth Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proc. CVPR*, 2017. 2

[11] Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *arXiv*, 2017. 1

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016. 5

[13] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. Evaluating feature importance estimates. *arXiv*, 2018. 2

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv*, 2015. 3

[15] Andrei Kapishnikov, Tolga Bolukbasi, Fernanda Viégas, and Michael Terry. Xrai: Better attributions through regions. In *Proc. ICCV*, 2019. 2

[16] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. 2019. 2

[17] Pieter-Jan Kindermans, Kristof Schütt, Klaus-Robert Müller, and Sven Dähne. Investigating the influence of noise and distractors on the interpretation of neural networks. *arXiv*, 2016. 1, 2, 3, 4

[18] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv*, 2018. 8

[19] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proc. NIPS*, 2017. 2

[20] Aravindh Mahendran and Andrea Vedaldi. Salient deconvolutional networks. In *Proc. ECCV*, 2016. 1, 2, 7

[21] Leann Myers and Maria J Sirois. Spearman correlation coefficients, differences between. *Encyclopedia of statistical sciences*, 12, 2004. 5

[22] Jose Oramas, Kaili Wang, and Tinne Tuytelaars. Visual explanation by interpretation: Improving visual feedback capabilities of deep neural networks. In *Proc. ICLR*, 2019. 2

[23] Barak A. Pearlmutter. Fast exact multiplication by the hessian. *Neural Computation*, 1994. 8

[24] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. In *Proc. BMVC*, 2018. 2

[25] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proc. KDD*, 2016. 2

[26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015. 2

[27] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *Proc. ICCV*, 2017. 1, 2, 3, 4

[28] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proc. ICML*, 2017. 2

[29] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Proc. ICLR*, 2014. 1, 2, 3, 4, 5

[30] Krishna Kumar Singh and Yong Jae Lee. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. In *Proc. ICCV*, 2017. 2

[31] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv*, 2017. 2

[32] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv*, 2014. 2

[33] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proc. ICML*, 2017. 2

[34] Xiaolong Wang, Abhinav Shrivastava, and Abhinav Gupta. A-fast-rcnn: Hard positive generation via adversary for object detection. In *Proc. CVPR*, 2017. 2

[35] Yunchao Wei, Jiashi Feng, Xiaodan Liang, Ming-Ming Cheng, Yao Zhao, and Shuicheng Yan. Object region mining with adversarial erasing: A simple classification to semantic segmentation approach. In *Proc. CVPR*, 2017. 2

[36] Mengjiao Yang and Been Kim. Benchmarking attribution methods with relative feature importance. *arXiv*, 2019. 2

[37] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Proc. ECCV*, 2014. 1, 2

[38] Jianming Zhang, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. In *Proc. ECCV*, 2016. 1, 2, 5

[39] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proc. CVPR*, 2016. 2, 4

There and Back Again: Revisiting Backpropagation Saliency Methods Supplementary Material

A. Pointing Game details

We use [6]'s Pointing Game for quantitative evaluation of saliency methods. Saliency maps are computed with respect to every object class present in each image. If the maximally salient point for each class lands on the ground truth annotation for that object (within a threshold of 15 pixels), then a "hit" is recorded; otherwise, a "miss" is recorded. Pointing game accuracy is computed as the mean over per-class accuracies given by the following: $\frac{|\text{hits}|}{|\text{hits}+\text{misses}|}$.

We evaluate VGG16 and ResNet50 networks that have been trained on ImageNet and fine-tuned on PASCAL VOC and COCO. We evaluate on the PASCAL VOC 2007 test split ($N = 4952$ images) and COCO 2014 val split ($N \approx 50k$). We also show performance on the difficult subsets of the data provided by [6]; these are images for which the total area of the annotations (bounding boxes for PASCAL VOC and segmentation masks for COCO) for the given object class is less than 25% of the image size and for which there is at least one other object class present. We use [2]'s TorchRay library for evaluation; see [6] for more details.

B. Virtual identity trick correlations

We computed the correlation between saliency maps generated both with and without the virtual identity trick (paper section 4.1). The high correlation shown in Table 1, as well as the minimal difference in pointing game performance ($0.53\% \pm 0.62\%$) demonstrates that the identity trick closely approximates the behaviour of calculating the spatial contributions for the original convolutional layers.

Architecture	Layer	Correlation
ResNet50	layer1.0.conv1	99.48 ± 0.55
ResNet50	layer2.0.conv1	99.32 ± 0.35
ResNet50	layer3.0.conv1	99.16 ± 0.42
ResNet50	layer4.0.conv1	98.35 ± 1.05
VGG16	features.2	97.77 ± 1.34
VGG16	features.7	99.15 ± 0.42
VGG16	features.14	98.99 ± 0.57
VGG16	features.21	97.79 ± 1.42
VGG16	features.28	94.33 ± 3.09

Table 1. **Correlations between saliency maps generated with and without the virtual identity trick.** For both VGG16 and ResNet50, the high values of correlation across the network (min of 94.33) justify the use of the identity trick.

	ResNet50		VGG16	
	All	Difficult	All	Difficult
CEB	layer3	layer4	feat.29	feat.29
EB	layer4	layer4	feat.29	feat.29
GC	layer4	layer4	feat.29	feat.29
Gd	layer2	layer2	feat.29	feat.22
Gds	layer2	layer2	feat.8	feat.8
Gui	layer3	layer3	input	input
LA	layer4	layer4	feat.29	feat.29
NG	layer3	layer3	feat.22	feat.22
sNG	layer4	layer4	feat.29	feat.29

Table 2. **Best individual layer for the Pointing game on VOC07.** (C)EB: (Contrastive) Excitation Backprop, GC: GradCAM, Gd(s): Gradient (sum), Gui: guided backprop, LA: linear approximation, (s)NG: (selective) NormGrad.

C. Performance of combining saliency maps

As noted in paper section 4.2, feature spread and classification accuracy are both interpretable as measures of feature sensitivity with respect to the class of the input image. Figure 1 shows that both increase with network depth, with the exception of feature spread for VGG16, which decreases in the last two layers despite simultaneously increasing classification accuracy. We note that classification accuracy is a more reliable metric than feature spread as it directly codes for the separability of features with respect to class, whereas feature spread is susceptible to non-material differences in the absolute scale of activation values across layers. In the case of VGG16, this means the features in the last two layers differ less across images (and hence, classes) in absolute terms, but are nonetheless highly discriminative of class.

The performance gains of our weighting schemes over using the best individual layer are given in table 3. The best individual layers for computing maps are given in table 2, which is notable as in no case does the practice of computing saliency at the earliest layer produce the best performance.

D. Meta-saliency analysis

We hypothesized that meta vs. non-meta saliency maps should be less correlated for validation images than for training images. This is because the inner gradient step of meta-saliency should not be as impactful for a seen training image as for an unseen validation image.

For both NormGrad and selective NormGrad, we evaluated over time the average correlation between the impor-

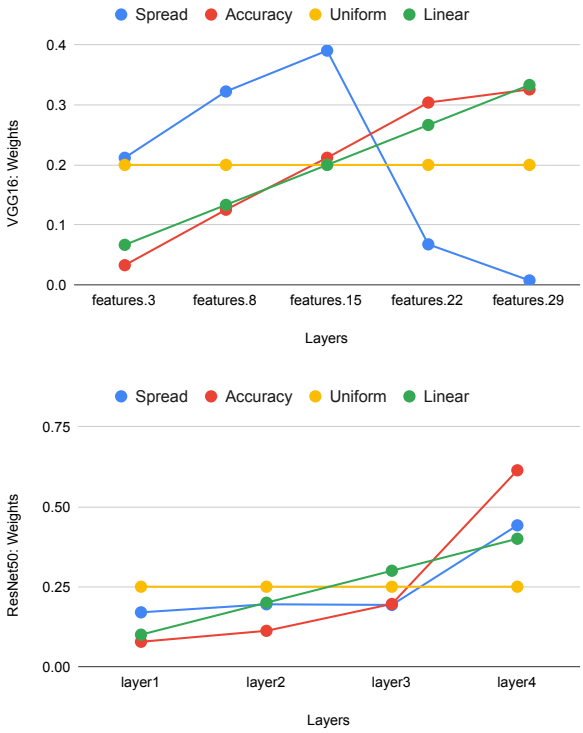


Figure 1. **Weights for the different weighting methods.** VGG16 on top and ResNet50 on the second row.

			ResNet50		VGG16	
			All	Difficult	All	Difficult
LA	+	accuracy	0.57	1.47	0.22	2.09
LA	+	spread	0.81	1.66	-3.01	-7.18
LA	+	linear	0.68	0.74	0.36	2.22
LA	+	uniform	0.58	1.29	-0.61	-0.20
LA	×	accuracy	0.87	**1.84**	0.42	**2.91**
LA	×	spread	**1.01**	1.78	-2.92	-6.80
LA	×	linear	0.52	0.72	**0.49**	2.87
LA	×	uniform	0.32	0.83	-0.25	0.52
sNG	+	accuracy	1.08	1.37	0.48	0.85
sNG	+	spread	0.94	1.40	-3.66	-6.43
sNG	+	linear	0.94	1.67	0.62	0.85
sNG	+	uniform	0.25	0.54	-0.93	-2.55
sNG	×	accuracy	**1.26**	1.65	**0.83**	**1.88**
sNG	×	spread	1.12	**2.09**	-3.13	-6.18
sNG	×	linear	0.63	0.95	0.72	0.62
sNG	×	uniform	0.89	1.79	-0.27	-1.76

Table 3. **Score gains compared to best individual layer performance for weighting methods with Linear Approximation (LA) and selective NormGrad (sNG) on the Pointing game on VOC07.** Paper showed that LA and sNG benefit from weighting methods. Here we can observe that using the weights as exponents in a product (lines with ×) is the most effective solution for both LA and sNG on ResNet50 and VGG16. Both the weightings using the features spread and the layer accuracy perform the best on ResNet50 but only the layer accuracy perform consistently across datasets and saliency methods.

tance maps with and without meta-saliency on the training and testing sets respectively. Figure 2 demonstrates that the correlation scores are indeed decreased for validation images.

E. Model weights sensitivity

In fig. 5, we show the effects of cascading randomization on linear approximation and selective NormGrad methods by using the same images as [1] ("junco", "corn" and "Irish terrier") to illustrate the model weights sensitivity.

Qualitatively, both methods - with and without meta-saliency - demonstrate the desired sensitivity to model weights as the saliency maps progressively lose focus from the object target as layer depth decreases. We also observe that using meta-saliency on top of the chosen base saliency method (shown on every other line of fig. 5) delays the degradation in saliency maps to earlier layers.

F. Image captioning

We show in this section a few examples of our proposed selective NormGrad method applied to the image captioning setting. We use a variant [4] of the original neuraltalk2 [3] with a ResNet101 as backbone network followed by the LSTM caption model. As in [5] we do a backward pass using the log probability of the generated caption as objective function. We apply selective NormGrad before the final global average pooling and at the ReLU layers just after the downsampling shortcuts of the third and fourth macro blocks. We also compare these saliency maps with the product combination map of these layers using a linear weighting scheme (see section 4.2 in the paper). We notice in fig. 6 that using a combination of layers produces a sharper saliency map than for individual layers.

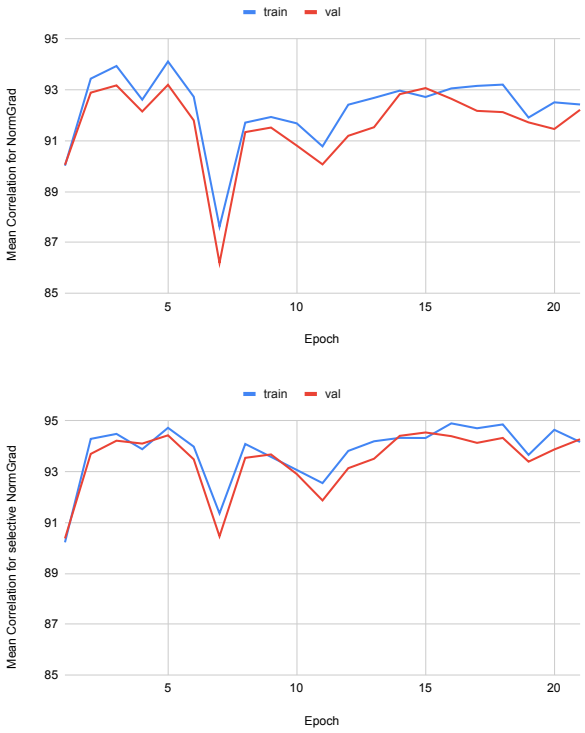


Figure 2. **Correlation between meta and non-meta saliency maps over time for train and val splits.** For both NormGrad (top) and selective NormGrad (bottom), the correlation is lower on the validation split.

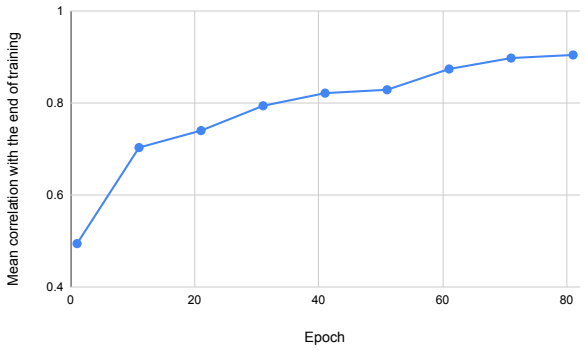


Figure 3. **Mean correlation on the val split between the meta-saliency maps of selective NormGrad at epoch t and the end of training.** The saliency maps stabilize at the end of the training.

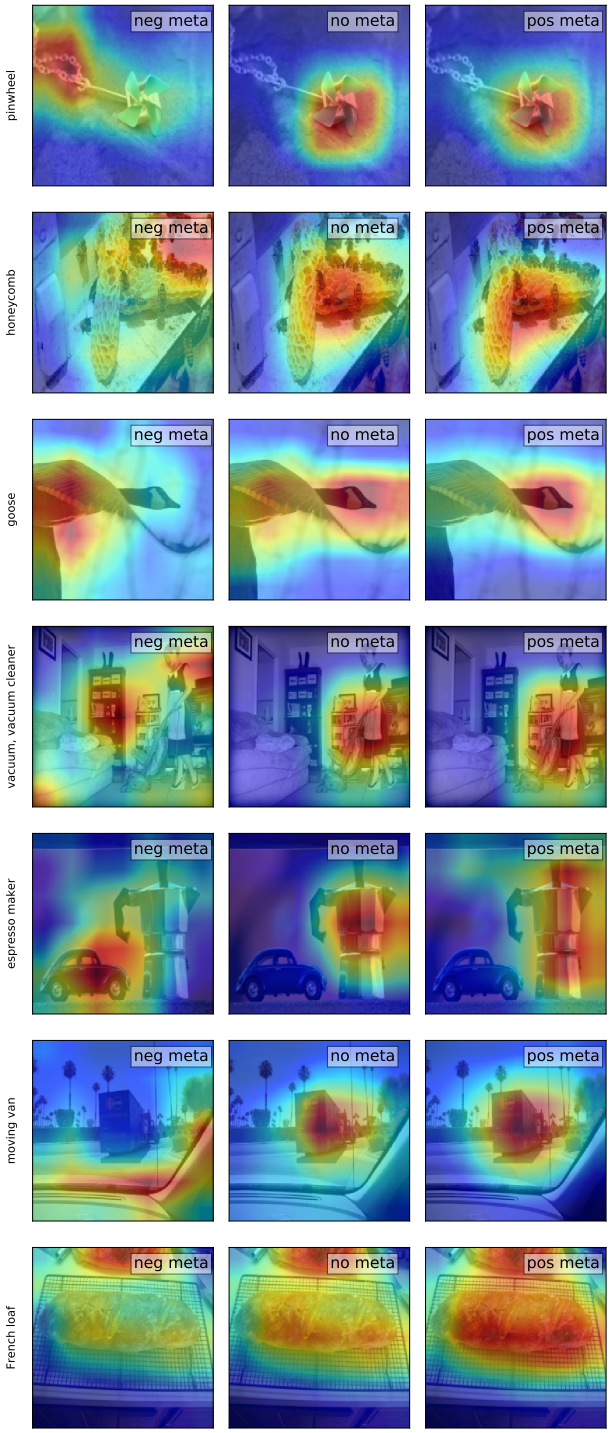


Figure 4. **Examples of positive and negative meta-saliency for selective NormGrad**. Negative meta-saliency (left images) corresponds to a gradient ascent inner step whereas a gradient descent step is used for the positive meta-saliency (images on the right). The center images do not use meta-saliency.

Figure 5. **Model weights sensitivity on VGG16.** Linear Approximation and selective NormGrad with and without meta-saliency.

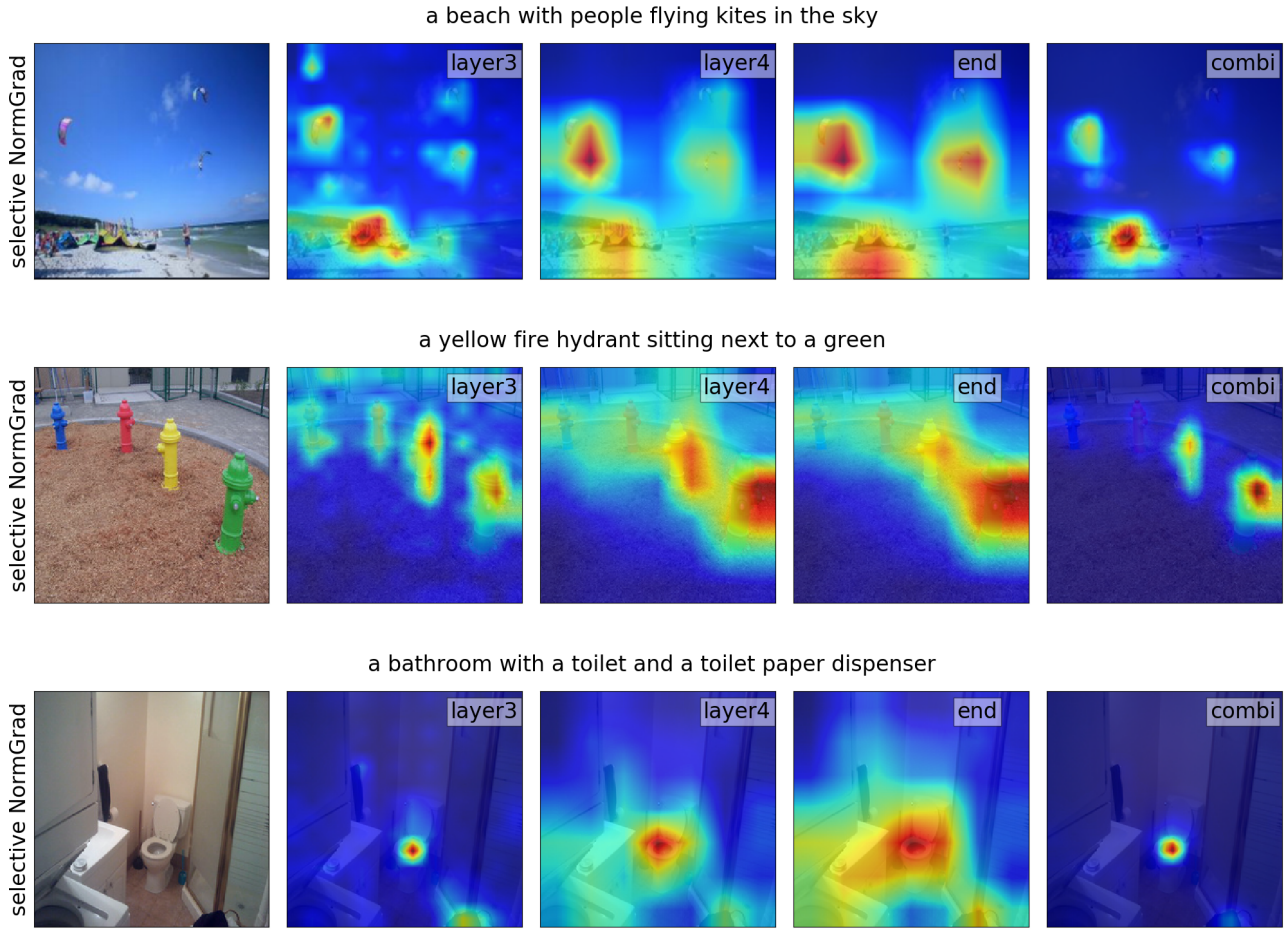


Figure 6. **Image captioning explanations.** Selective NormGrad is applied at different layers or combination of layers (last column) of a ResNet101. We observe that saliency maps at individual layers highlight a big part of the images. Using a combination of layers allows a clearer focus on the important parts of the images.

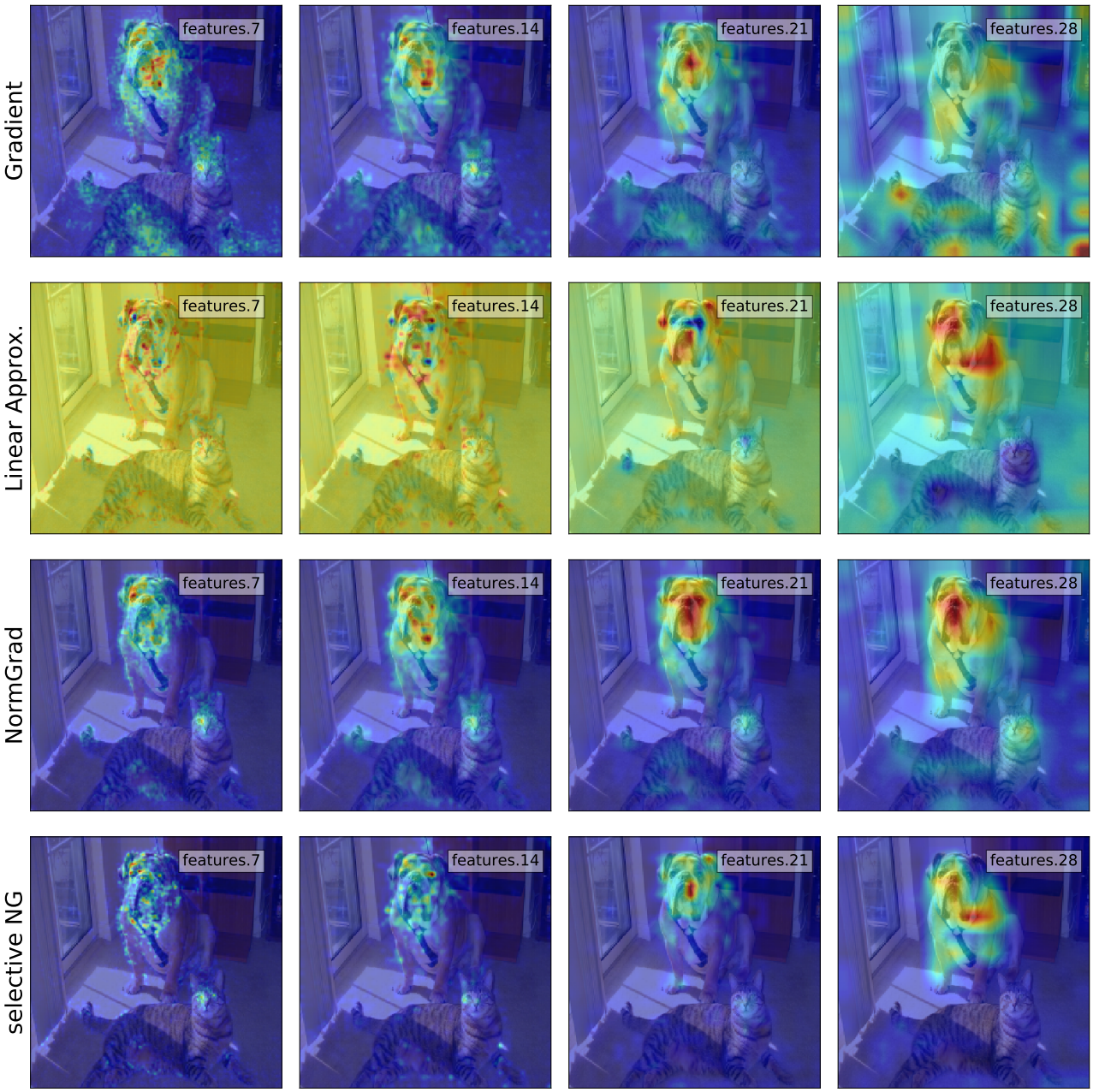


Figure 7. **Visualizations for some methods of the Extract & Aggregate framework at different layers of VGG16.** The gradient backprogation method (first row) works well at all stages except at the end of the network. Selective NormGrad (last row), NormGrad (third row) and Linear Approximation (second row) perform well across the network. Finally we can observe that selective NormGrad and Linear Approximation are more class selective than NormGrad as the non targeted "tiger cat" appears more in the third row.

References

[1] Julius Adebayo, Justin Gilmer, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Proc. NeurIPS*, 2018. 2

[2] Ruth Fong, Mandela Patrick, and Andrea Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *Proc. ICCV*, 2019. 1

[3] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proc. CVPR*, 2015. 2

[4] Ruotian Luo, Brian Price, Scott Cohen, and Gregory Shakhnarovich. Discriminability objective for training descriptive captions. *arXiv*, 2018. 2

[5] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *Proc. ICCV*, 2017. 2

[6] Jianming Zhang, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. In *Proc. ECCV*, 2016. 1

Statement of Authorship for joint/multi-authored papers for PGR thesis

To appear at the end of each thesis chapter submitted as an article/paper

The statement shall describe the candidate's and co-authors' independent research contributions in the thesis publications. For each publication there should exist a complete statement that is to be filled out and signed by the candidate and supervisor **(only required where there isn't already a statement of contribution within the paper itself).**

Title of Paper	There and Back Again: Revisiting Backpropagation Saliency Methods
Publication Status	X Published X Accepted for Publication □ Submitted for Publication □Unpublished and unsubmitted work written in a manuscript style
Publication Details	Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020. SA Rebuffi*, R Fong*, X Ji*, A Vedaldi. (* denotes equal contribution)

Student Confirmation

Student Name:	Sylvestre-Alvise REBUFFI		
Contribution to the Paper	• Conception of the idea (Extract & Aggregate framework, layer combination) • Implementation and doing the experiments • Writing the paper • Preparing the supplementary material • Writing the rebuttal		
Signature 		Date	24/07/2020

Supervisor Confirmation

By signing the Statement of Authorship, you are certifying that the candidate made a substantial contribution to the publication, and that the description described above is accurate.

Supervisor name and title:	Prof. Andrea Vedaldi
Supervisor comments	
Signature 	Date

This completed form should be included in the thesis, at the end of the relevant chapter.

Chapter 9

Conclusions

In this thesis we examined the influence of the input data on learning deep representations for image classification. First, we explored how the training data impacts the design of the model and its optimisation. Our work spanned diverse input data settings from fully labelled multiple visual domains to different types of low-supervision such as semi-supervised learning, novel class discovery and clustering. Some of our proposed methods act on the network architecture like the residual adapters in the multi-domain case or the two-headed network to jointly discover new classes and learn them with the old classes. We also focused on the optimisation side with the clustering loss based on pairwise cluster predictions or with the alternating optimisation scheme to avoid overfitting the scarse labelled data in the semi-supervised setting. Second, besides adapting our representation learning methods to the training data, we also delved into saliency to highlight which parts of the image are responsible for the network predictions. To this end, we introduced a framework based on the learning mechanism of convolutional neural networks to derive new or existing backpropagation saliency methods. This body of work raises certain questions and observations, which we discuss in the following section.

9.1 Discussion and future work

From the exploration of the different types of input data we observe that the design of our methods highly depends on how deep networks interact with data. This leads us to the question of the versatility and the limitations of deep neural networks. For example, deep neural networks have a very large amount of degrees of freedom, which poses an overfitting issue when there are few labelled data at hand. We addressed this problem by proposing the systematic use of self-supervision pre-training and early

layers freezing in settings such as semi-supervised learning, class discovery or clustering. Thus, such pre-training and freezing prevent the early layers of the network from overfitting trivial visual cues such as colours or edges.

Merging representations. In some cases the solution to a deep network limitation is not fully satisfactory. For example, in the multiple visual domains image classification case, network parametrisations such as our residual adapters or the masking techniques of [99, 98] naturally emerge from the sequential nature of the CNN architecture and modulate the output of previously learned filters. However, these adapting modules are specific to each dataset and do not explicitly capture relations between datasets. Hence, these approaches circumvent the key issue of merging several tasks and data representations in one fully shared network.

We examined this problem in Chapters 5 and 6. In fact, our novel class discovery method aims at inducing a positive interaction between two tasks: clustering new classes and learning all the classes together. In another instance, our alternating optimisation scheme addresses the sensitive balancing between the losses on the labelled and unlabelled set in the semi-supervised setting. Nevertheless, the problem of merging representations in one network and avoiding negative interference between tasks is an open question which could be a source of future work. Indeed, it touches several subfields of computer vision, where the network learns tasks either sequentially as in life-long learning or simultaneously as in multi-task and multi-domain learning.

Robustness. From our work on semi-supervised learning, novel class discovery and clustering, we noted the paramount importance of consistency in low-supervision settings. Hence, we could extend to other tasks this idea of enforcing consistent predictions across different augmented views of the same image. An example of future work could use consistency to improve the model robustness to domain changes. More specifically, it would be particularly useful for a domain adaptation setting where the only samples available at training time come from a single domain. In such a case, the choice of the data augmentation for the consistency could play a major role in the adaptation performance. Indeed, we observed the importance of the data augmentation choice in low-supervision settings.

Another possible way to improve the model robustness could be inspired by our meta-saliency idea. Indeed, instead of using for saliency the loss which contains an inner gradient ascent step, we could use it to adversarially perturb the network weights

during training. By construction, the network would thus be trained to be more robust to adversarial perturbations. We could evaluate this method on diverse settings such as robustness to adversarial attacks or domain changes.

Prediction uncertainty. Another topic closely related to robustness is prediction uncertainty. In fact, numerous real-life applications like medical diagnosis or self-driving cars require the ability to predict the failures of the model. In our deep clustering work we noted the efficacy of data composition techniques such as RICAP [145] or Mixup [167], which use linear interpolations of labels as targets for the classification loss. As these augmentations generate other targets in the probability simplex than the standard one-hot vector labels, the network is trained to produce mixed predictions corresponding to the weights of the different classes in the image. These more complex prediction vectors could be exploited in a future work on confidence calibration.

An alternative direction could be to use network parametrisations like our residual adapters on top of pre-trained filters. In fact, we noticed that such approaches obtain good generalisation performance with few trainable parameters whereas standard fine-tuning tends to overfit the training data. Indeed, when using the adapters, the predictions obtained on the training set do not tend towards one-hot vectors. Hence, as in the previous paragraph, these complex prediction vectors might be usable for confidence calibration. Furthermore, our work on residual adapters also got some attention in transfer learning for natural languague processing [66, 6], where adapters get similar or better accuracy than standard fine-tuning with faster convergence and less parameters. Hence, it would be interesting to study whether such adapters could be transposed to other domains like speech or even to other fields such as object tracking or reinforcement learning.

Defining classes. Recent semi-supervised methods [135, 10] show, by combining self-labeling with strong data augmentations, that a few labels are sufficient to almost match fully supervised classification performance. Therefore, a few labels can be sufficient to define classes. Note that defining classes becomes more challenging when there are no available labels. Popular self-supervision methods such as SimCLR [23], CPC [112] or MoCo [63] learn representations by using a contrastive loss between augmented views of the same sample. However, these approaches do not regroup samples into classes. They learn to discriminate between samples at the image

level, not at the class level and they still require labels to fine-tune the model for a classification task.

Hence, to jointly identify classes and learn the model representation, we proposed a clustering loss which establishes groups of samples based on the similarity between pairs of samples. Though, our method under-performs on classes with a high inner class variability such as the "dog" and "cat" classes in CIFAR-10. A possible explanation for this clustering issue is that there could be many valid partitionings of the data. Hence, a goal for a future work would be to devise a method which proposes different possible clusterings of the data. Furthermore, this method should ideally be able to motivate each of the proposed clusterings, for example by creating and extracting attributes to define the different classes.

Bibliography

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Proc. NIPS*, pages 9505–9515, 2018.

[2] A Agrawala. Learning with a probabilistic teacher. *IEEE Transactions on Information Theory*, 16(4):373–379, 1970.

[3] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv preprint arXiv:1711.06104*, 2017.

[4] Bernard Ans and Stéphane Rousset. Avoiding catastrophic forgetting by coupling two reverberating neural networks. *Comptes Rendus de l'Académie des Sciences-Series III-Sciences de la Vie*, 320(12):989–997, 1997.

[5] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. In *Proc. NIPS*, pages 41–48, 2007.

[6] Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. On the cross-lingual transferability of monolingual representations. *arXiv preprint*, 2019.

[7] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. *arXiv preprint*, 2019.

[8] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[9] Michele Banko and Eric Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th annual meeting of the Association for Computational Linguistics*, pages 26–33, 2001.

[10] David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. *arXiv preprint arXiv:1911.09785*, 2019.

[11] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Proc. NIPS*, pages 5049–5059, 2019.

[12] Bharath Bhushan Damodaran, Benjamin Kellenberger, Rémi Flamary, Devis Tuia, and Nicolas Courty. Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. In *Proc. ECCV*, pages 447–463, 2018.

[13] Hakan Bilen, Basura Fernando, Efstratios Gavves, Andrea Vedaldi, and Stephen Gould. Dynamic image networks for action recognition. In *Proc. CVPR*, pages 3034–3042, 2016.

[14] Hakan Bilen and Andrea Vedaldi. Integrated perception with recurrent multi-task neural networks. In *Proc. NIPS*, pages 235–243, 2016.

[15] Hakan Bilen and Andrea Vedaldi. Universal representations: The missing link between faces, text, planktons, and cat breeds. *arXiv preprint*, 2017.

[16] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Proc. CVPR*, pages 3722–3731, 2017.

[17] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013.

[18] Chunshui Cao, Xianming Liu, Yi Yang, Yinan Yu, Jiang Wang, Zilei Wang, Yongzhen Huang, Liang Wang, Chang Huang, Wei Xu, et al. Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks. In *Proc. ICCV*, pages 2956–2964, 2015.

[19] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proc. ECCV*.

[20] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[21] Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep adaptive image clustering. In *Proc. ICCV*, pages 5879–5887, 2017.

[22] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[23] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[24] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proc. ICML*, pages 794–803, 2018.

[25] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proc. CVPR*, pages 3606–3613, 2014.

[26] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Proc. CVPR*, pages 3642–3649. IEEE, 2012.

[27] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. ICML*, pages 160–167, 2008.

[28] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[29] Gabriela Csurka. A comprehensive survey on domain adaptation for visual applications. In *Domain adaptation in computer vision applications*, pages 1–35. Springer, 2017.

[30] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[31] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

[32] Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. In *Proc. NIPS*, pages 6967–6976, 2017.

[33] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on audio, speech, and language processing*, 20(1):30–42, 2011.

[34] Hal Daume III and Daniel Marcu. Domain adaptation for statistical classifiers. *Journal of artificial Intelligence research*, 26:101–126, 2006.

[35] Li Deng, Geoffrey Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications: An overview. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8599–8603. IEEE, 2013.

[36] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[37] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[38] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proc. ICCV*, pages 1422–1430, 2015.

[39] Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *Proc. ICCV*, pages 2051–2060, 2017.

[40] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

[41] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *Proc. NIPS*, pages 766–774, 2014.

[42] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

[43] Theodoros Evgeniou, Charles A Micchelli, and Massimiliano Pontil. Learning multiple tasks with kernel methods. *Journal of machine learning research*, 6(Apr):615–637, 2005.

[44] Sharon Fogel, Hadar Averbuch-Elor, Daniel Cohen-Or, and Jacob Goldberger. Clustering-driven deep embedding with pairwise constraints. *IEEE computer graphics and applications*, 39(4):16–27, 2019.

[45] Ruth Fong, Mandela Patrick, and Andrea Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *Proc. ICCV*, pages 2950–2958, 2019.

[46] Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proc. ICCV*, pages 3429–3437, 2017.

[47] Dario Fontanel, Fabio Cermelli, Massimiliano Mancini, Samuel Rota Bulò, Elisa Ricci, and Barbara Caputo. Boosting deep open world recognition by clustering. *arXiv preprint arXiv:2004.13849*, 2020.

[48] S Fralick. Learning to recognize patterns without a teacher. *IEEE Transactions on Information Theory*, 13(1):57–64, 1967.

[49] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[50] Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

[51] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proc. ICML*, pages 1180–1189, 2015.

[52] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[53] Aude Genevay, Gabriel Dulac-Arnold, and Jean-Philippe Vert. Differentiable deep clustering with cluster size constraints. *arXiv preprint arXiv:1910.09036*, 2019.

[54] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proc. ICCV*, pages 5736–5745, 2017.

[55] Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Boosting few-shot visual learning with self-supervision. In *Proc. ICCV*, pages 8059–8068, 2019.

[56] Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Learning representations by predicting bags of visual words. In *Proc. CVPR*, pages 6928–6938, 2020.

[57] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[58] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. CVPR*, pages 580–587, 2014.

[59] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proc. NIPS*, pages 2672–2680, 2014.

[60] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

[61] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010.

[62] Kai Han, Andrea Vedaldi, and Andrew Zisserman. Learning to discover novel visual categories via deep transfer clustering. In *Proc. ICCV*, pages 8401–8409, 2019.

[63] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proc. CVPR*, pages 9729–9738, 2020.

[64] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[65] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *Proc. ICML*, pages 1989–1998, 2018.

[66] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. *arXiv preprint arXiv:1902.00751*, 2019.

[67] Yen-Chang Hsu and Zsolt Kira. Neural network-based clustering using pairwise constraints. *arXiv preprint arXiv:1511.06321*, 2015.

[68] Yen-Chang Hsu, Zhaoyang Lv, and Zsolt Kira. Learning to cluster in order to transfer across domains and tasks. *arXiv preprint arXiv:1711.10125*, 2017.

[69] Yen-Chang Hsu, Zhaoyang Lv, Joel Schlosser, Phillip Odom, and Zsolt Kira. Multi-class classification without multi-class labels. *arXiv preprint arXiv:1901.00544*, 2019.

[70] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximizing self-augmented training. *arXiv preprint arXiv:1702.08720*, 2017.

[71] Gabriel Huang, Hugo Larochelle, and Simon Lacoste-Julien. Centroid networks for few-shot clustering and unsupervised few-shot classification. *arXiv preprint arXiv:1902.08605*, 3(7), 2019.

[72] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label propagation for deep semi-supervised learning. In *Proc. CVPR*, pages 5070–5079, 2019.

[73] Jacob Jackson and John Schulman. Semi-supervised learning by label gradient alignment. *arXiv preprint arXiv:1902.02336*, 2019.

[74] Laurent Jacob, Jean-philippe Vert, and Francis R Bach. Clustered multi-task learning: A convex formulation. In *Proc. NIPS*, pages 745–752, 2009.

[75] Xu Ji, João F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proc. ICCV*, pages 9865–9874, 2019.

[76] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. *arXiv preprint arXiv:1611.05148*, 2016.

[77] Alistair EW Johnson, Tom J Pollard, Nathaniel R Greenbaum, Matthew P Lungren, Chih-ying Deng, Yifan Peng, Zhiyong Lu, Roger G Mark, Seth J Berkowitz, and Steven Horng. Mimic-cxr-jpg, a large publicly available database of labeled chest radiographs. *arXiv preprint arXiv:1901.07042*, 2019.

[78] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proc. CVPR*, pages 7482–7491, 2018.

[79] Pieter-Jan Kindermans, Kristof Schütt, Klaus-Robert Müller, and Sven Dähne. Investigating the influence of noise and distractors on the interpretation of neural networks. *arXiv preprint arXiv:1611.07270*, 2016.

[80] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[81] Iasonas Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *Proc. CVPR*, pages 6129–6138, 2017.

[82] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

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

[84] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

[85] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 2015.

[86] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *Proc. ECCV*, pages 577–593. Springer, 2016.

[87] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[88] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, 2013.

[89] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[90] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proc. CVPR*, pages 3431–3440, 2015.

[91] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. In *Proc. NIPS*, pages 136–144, 2016.

[92] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Proc. NIPS*, pages 6467–6476, 2017.

[93] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[94] Erika Lu, Weidi Xie, and Andrew Zisserman. Class-agnostic counting. In *Asian conference on computer vision*, pages 669–684. Springer, 2018.

[95] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[96] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proc. CVPR*, pages 5188–5196, 2015.

[97] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[98] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proc. ECCV*, pages 67–82, 2018.

[99] Massimiliano Mancini, Elisa Ricci, Barbara Caputo, and Samuel Rota Bulò. Adding new tasks to a single network with weight transformations using binary masks. In *Proc. ECCV*, 2018.

[100] Mark Marsden, Kevin McGuinness, Suzanne Little, Ciara E Keogh, and Noel E O'Connor. People, penguins and petri dishes: Adapting object counting models to new visual domains and object types without forgetting. In *Proc. CVPR*, pages 8070–8079, 2018.

[101] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[102] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, 2006.

[103] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

[104] Stefan Munder and Dariu M Gavrila. An experimental study on pedestrian classification. *IEEE transactions on pattern analysis and machine intelligence*, 28(11):1863–1868, 2006.

[105] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[106] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Proc. NIPS*, pages 849–856, 2002.

[107] Feiping Nie, Zinan Zeng, Ivor W Tsang, Dong Xu, and Changshui Zhang. Spectral embedded clustering: A framework for in-sample and out-of-sample spectral clustering. *IEEE Transactions on Neural Networks*, 22(11):1796–1808, 2011.

[108] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008.

[109] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Proc. ECCV*, pages 69–84. Springer, 2016.

[110] Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. *Distill*, 3(3):e10, 2018.

[111] Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *Proc. NIPS*, pages 3235–3246, 2018.

[112] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[113] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proc. CVPR*, pages 2536–2544, 2016.

[114] Karl Pearson. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London. A*, 185:71–110, 1894.

[115] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. *arXiv preprint arXiv:1806.07421*, 2018.

[116] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.

[117] Bharath Ramsundar, Steven Kearnes, Patrick Riley, Dale Webster, David Konerding, and Vijay Pande. Massively multitask networks for drug discovery. *arXiv preprint arXiv:1502.02072*, 2015.

[118] Rajeev Ranjan, Vishal M Patel, and Rama Chellappa. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(1):121–135, 2017.

[119] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *Proc. NIPS*, pages 3546–3554, 2015.

[120] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proc. CVPR*, pages 2001–2010, 2017.

[121] Amir Rosenfeld and John K Tsotsos. Incremental learning through deep adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[122] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[123] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[124] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Proc. NIPS*, pages 1163–1171, 2016.

[125] Saquib Sarfraz, Vivek Sharma, and Rainer Stiefelhagen. Efficient parameter-free clustering using first neighbor relations. In *Proc. CVPR*, pages 8934–8943, 2019.

[126] Walter J Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E Boult. Toward open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1757–1772, 2012.

[127] H Scudder. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371, 1965.

[128] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proc. ICCV*, pages 618–626, 2017.

[129] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[130] Mingguang Shi and Bing Zhang. Semi-supervised learning improves gene expression-based prediction of cancer recurrence. *Bioinformatics*, 27(21):3017–3023, 2011.

[131] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. *arXiv preprint*, 2017.

[132] Oriane Siméoni, Mateusz Budnik, Yannis Avrithis, and Guillaume Gravier. Rethinking deep active learning: Using unlabeled data at model training. *arXiv preprint arXiv:1911.08177*, 2019.

[133] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[134] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[135] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*, 2020.

[136] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[137] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[138] Suraj Srinivas and François Fleuret. Full-gradient representation for neural network visualization. In *Proc. NIPS*, pages 4124–4133, 2019.

[139] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32:323–332, 2012.

[140] Michael Steinbach, Levent Ertöz, and Vipin Kumar. The challenges of clustering high dimensional data. In *New directions in statistical physics*, pages 273–309. Springer, 2004.

[141] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[142] Yu Sun, Eric Tzeng, Trevor Darrell, and Alexei A Efros. Unsupervised domain adaptation through self-supervision. *arXiv preprint arXiv:1909.11825*, 2019.

[143] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. *arXiv preprint arXiv:1703.01365*, 2017.

[144] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Proc. NIPS*, pages 3104–3112, 2014.

[145] Ryo Takahashi, Takashi Matsubara, and Kuniaki Uehara. Data augmentation using random image cropping and patching for deep cnns. *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[146] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Proc. NIPS*, pages 1195–1204, 2017.

[147] Alex Teichman and Sebastian Thrun. Tracking-based semi-supervised learning. *The International Journal of Robotics Research*, 31(7):804–818, 2012.

[148] Sebastian Thrun. Lifelong learning algorithms. In *Learning to learn*, pages 181–209. Springer, 1998.

[149] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. In *Proc. CVPR*, pages 648–656, 2015.

[150] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[151] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proc. ICML*, pages 1096–1103, 2008.

[152] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

[153] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[154] Tianjun Xiao, Jiaxing Zhang, Kuiyuan Yang, Yuxin Peng, and Zheng Zhang. Error-driven incremental learning in deep convolutional neural network for large-scale image classification. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 177–186, 2014.

[155] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *Proc. ICML*, pages 478–487, 2016.

[156] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*, 2019.

[157] Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *Proc. ICML*, pages 3861–3870, 2017.

[158] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *Proc. CVPR*, pages 5147–5156, 2016.

[159] Yi Yang, Dong Xu, Feiping Nie, Shuicheng Yan, and Yueting Zhuang. Image clustering using local discriminant models and global integration. *IEEE Transactions on Image Processing*, 19(10):2761–2773, 2010.

[160] Yongxin Yang and Timothy Hospedales. Deep multi-task representation learning: A tensor factorisation approach. In *Proc. ICLR*, 2016.

[161] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pages 189–196, 1995.

[162] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Proc. NIPS*, pages 3320–3328, 2014.

[163] Dong Yu, Balakrishnan Varadarajan, Li Deng, and Alex Acero. Active learning and semi-supervised learning for speech recognition: A unified framework using the global entropy reduction maximization criterion. *Computer Speech & Language*, 24(3):433–444, 2010.

[164] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[165] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Proc. ECCV*, pages 818–833. Springer, 2014.

[166] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S4l: Self-supervised semi-supervised learning. In *Proc. ICCV*, pages 1476–1485, 2019.

[167] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[168] Jianming Zhang, Sarah Adel Bargal, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. *International Journal of Computer Vision*, 126(10):1084–1102, 2018.

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

[170] Richard Zhang, Phillip Isola, and Alexei A Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proc. CVPR*, pages 1058–1067, 2017.

[171] Tianzhu Zhang, Bernard Ghanem, Si Liu, and Narendra Ahuja. Robust visual tracking via structured multi-task sparse learning. *International journal of computer vision*, 101(2):367–383, 2013.

[172] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *Proc. ECCV*, pages 94–108. Springer, 2014.

[173] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proc. CVPR*, pages 2921–2929, 2016.

[174] Yang Zou, Zhiding Yu, BVK Vijaya Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *Proceedings of the European conference on computer vision (ECCV)*, pages 289–305, 2018.