Discrete Representations of Continuous Data using Deep Learning and Clustering

Louis Mahon

Linacre College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2022

To the universe, for its magnificent maddening mystery

Abstract

The divide between continuous and discrete data is a fundamental one in computer science and mathematics, as well as related areas such as cognitive science. Historically, most of computing has operated in the discrete domain, but connectionism offers an alternative set of techniques for representing data with continuous vectors, an alterative which has come to the fore with the advent of deep learning over the past decade. This thesis explores techniques for converting continuous, high-dimensional data, of the sort processed so successfully by deep learning, to discrete compact representations, of the sort used by traditional computing. Each of the five main chapters introduces a novel technique that contributes towards this goal, but is also able to be read as a stand-alone piece of research. These techniques fall under deep learning and clustering, and, in keeping with representation learning in general, are mostly, though not entirely, in the unsupervised setting. Some chapters focus on deep learning or clustering separately as a means to form discrete representations of continuous data. Others explore how to combine both deep learning and clustering in a single end-to-end learning system. Such a combination itself involves the interface between continuous and discrete, as deep learning operates on the former, and clustering on the latter.

Being able to bridge the gap between the worlds of continuous and discrete also aligns with the original goal of AI to model human intelligence, as an important part of human cognition is the movement between the worlds of continuous and discrete. Our sensory input is largely continuous, but we represent it with a natural language and reasoning apparatus that is largely discrete. A machine that one day thinks and acts as a human will have to learn to do the same.

Contents

This thesis is based on the following papers, all of which I am first author of:

1. L. Mahon, E. Giunchiglia, B. Li, and T. Lukasiewicz (2020). "Knowledge graph extraction from videos". In: *Proceedings of The Ninth IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, pp. 25–32 (Chapter 6)

2. L. Mahon and T. Lukasiewicz (2021). "Selective pseudo-label clustering". In: *Proceedings of the Conference on Artificial Intelligence (Künstliche Intelligenz)*. Springer, pp. 158–178 (Chapter 3)

3. L. Mahon and T. Lukasiewicz (2022). "Efficient deep clustering of human activities and how to improve evaluation". In: *arxiv preprint arXiv:2209.08335* (Chapter 4)

4. L. Mahon and T. Lukasiewicz (2023a). "Hard Regularization to Prevent Collapse in Online Deep Clustering without Data Augmentation". In: *arXiv preprint arXiv:2303.16521* (Chapter 5)

5. L. Mahon and T. Lukasiewicz (2023b). "Minimum Description Length Clustering to Measure Meaningful Image Complexity". In: *Available at SSRN 4391368* (Chapter 7)

Other works, which are not part of this thesis, include

1. R. Sauri, L. Mahon, I. Russo, and M. Bitinis (2019). "Cross-dictionary linking at sense level with a double-layer classifier". In: *2nd Conference on Language, Data and Knowledge (LDK 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik

2. L. Mahon and C. Vogel (2022). "The Proof is in the Pudding: Using Automated Theorem Proving to Generate Cooking Recipes". In: *arXiv preprint arXiv:2203.02683*

3. L. Mahon, L. Shah, and T. Lukasiewicz (2023). "Correcting Flaws in Common Disentanglement Metrics". In: *arXiv preprint arXiv:2304.02335*

Acronyms

AE autoencoder

AI artificial intelligence

ARI adjusted Rand index

CNN convolutional neural network

DNN deep neural network

GAN generative adversarial network

GMM Gaussian mixture model

GRU gated recurrent unit

GT ground truth

HAR human activity recognition

HMM hidden Markov model

KL Kullback-Leibler (divergence)

NL natural language

NLP natural language processing

NMI normalized mutual information

RI Rand index

rloss reconstruction loss

Chapter 1

Introduction

1.1 Motivation

Over the past decade, deep learning has greatly advanced the field of artificial intelligence (AI). It has been especially effective at certain aspects of intelligence which, in humans, are referred to as perceptual, and which are performed rapidly and largely unconsciously. For example, processing an image to locate and recognize objects, or processing an audio signal of speech to detect word boundaries and identify words. Perceptual tasks can be distinguished from more abstract symbolic tasks such as playing chess, solving logic puzzles or assignment problems. Programming computers to perform these more abstract tasks is mostly associated with research in the 1970s and 1980s in areas such as logic programming, the semantic web, automated theorem proving and expert systems (Jackson, 1986). Some success was seen in symbolic tasks, such as super-human chess performance in IBM's Deep Blue, but the perceptual tasks were more resistant to progress. Prior to the advent of deep learning, the leading techniques in AI struggled with perception-level tasks, even as they thrived on tasks that, to humans, seem more difficult. This differential ability of AI was noted from at least 1988, when Moravec commented that it was easy for a computer to surpass humans in many intelligence tests, though impossible for them to achieve the perceptual ability of baby (Moravec, 1988), an observation that was later referred to as Moravec's paradox.

Modern deep learning has largely redressed the imbalance highlighted in Moravec's paradox. Now, computers show human-level performance in many narrow perceptual tasks such as object recognition (K. He, X. Zhang, S. Ren, and J. Sun, 2015), medical image diagnostics (OŃeil, Kascenas, Henry, Wyeth, Shepherd, Beveridge, Clunie, Sansom, Seduikyte Keith Muir, and Poole, 2018; Rajpurkar, Hannun, Haghpanahi, Bourn, and Ng, 2017) and emotion classification (Arriaga, Valdenegro-Toro, and Plöger, 2017). However, in the wake of this progress, there is a rift between the framework of discrete symbols that dominated AI research in the 1980s, and the framework of continuous vectors that underlies the gradient-based optimization of deep neural networks. In order to preserve and build off the deep learning revolution of the past decade, we must either replace symbolic systems with deep learning equivalents or we must create systems that unify the two approaches. The latter are often called hybrid systems (Smolensky, 1986; Minsky, 1991). This thesis

focuses on data representations at the interface in hybrid systems. Specifically, it explores how the continuous representations formed using deep learning might be converted into symbolic forms of the sort that can be processed by a rule-based system. In part, this is motivated by the need for such conversions in hybrid systems, but it is also motivated by comparisons with human cognition.

The contrast between deep learning and symbolic AI can be compared with Kahneman's famous System 1 and System 2 thinking in human psychology (Kahneman, 2011). System 1 is fast, requires minimal effort and often operates beneath conscious awareness. I might instantly and automatically recognize the face of a friend, but be unable to say what sort of cognitive processing led to this recognition, and struggle to articulate the aspects of the visual stimulus that make it recognizable. Some stimuli can affect even our thoughts and behaviour without entering consciousness at all, so-called subliminal stimuli, (Verwijmeren, Karremans, Stroebe, and Wigboldus, 2011; Vorberg, Mattler, Heinecke, Schmidt, and Schwarzbach, 2003). At other times, however, they do enter the field of consciousness, and it is there that they can be processed by System 2. The purview of System 2 is deliberate and deductive reasoning. It is slow, and it requires effort, but it can do things that System 1 cannot, such as solve moderate-to-difficult numerical problems, or analyze the validity of logical arguments. While this comparison can be a constructive one and has been made a number of times when discussing AI and deep learning (Bengio, Lecun, and G. Hinton, 2021; Nye, Tessler, Tenenbaum, and Lake, 2021; Bengio, 2017), the two distinctions, perception/symbolic in AI and System 1/System 2 in human cognition, do not align perfectly. System 1 also includes tasks such as simple arithmetic and reading short text snippets, which seem to fit more with symbolic manipulation and, conversely, certain perceptual tasks that are unfamiliar and require directed attention fall under System 2. Additionally, consciousness plays an important role in Kahneman's framework: System 1 is unconscious and System 2 is conscious, and it is hard to see how this can be applied to AI. For a hybrid AI system, it would be contrived to say that the deep learning component is unconscious and the symbolic component is conscious.

Another comparison that could be made to human cognition relates to learning. The topic of this thesis, namely converting continuous vector-based representations into symbolic representations, is analogous to the problem solved by the human mind when it learns to make sense of the rich world of sensations we are born into. Every second, we are confronted with an enormous quantity of sensory information[1], a dazzling bombardment tamed only by the imposed order of our cognitive structures. William James famously used the phrase 'buzzing, blooming confusion' to refer to the world of a baby who has yet to acquire much of this cognitive structure, and so is less shielded from the full extent of its input. Somehow or other, babies all find their way out of this confusion, and end up, as adults, with a relatively

[1]The figure of 11 million bits of unconscious information and 50 bits of conscious information is often quoted in popular science writing about the human brain, and occasionally even peer-reviewed articles (Driver, Svensson, Amato, and Pate, 1996; Orzan, Zara, and Purcarea, 2012; Gelter, 2003). The earliest reference I have found was to a popular science book from 1993 (Norretranders, Faerch-Jensen, and Wahlen, 1993). However, I have been unable to find a primary source that was not restricted to animal models and narrow artificial distributions of stimuli, and the figures are heavily dependent on the input distributions, because the information contents are calculated using the probability under these distributions. So, while the spirit behind these figures is correct, their direct veracity is questionable.

coherent understanding of the world. An essential component of this understanding is the representation of continuous sense impressions using discrete symbols. For example, we receive a certain continuous pattern of stimulation on the retina, and then conceive of and describe this pattern as a set of objects–i.e. as a set of instances of categories–where these objects are in particular locations and bear particular properties. These categories are clearly not innate, because many of them correspond to synthetic objects that did not exist in our evolutionary history. Instead, infants must learn these categories from the data presented to them via their senses. In the fields of neuroscience and psychology, this is known as concept learning, and is a long-established and actively researched area (Hunt, 1962; Zeithamova, Mack, Braunlich, T. Davis, Seger, Kesteren, and Wutz, 2019). Closely related to concept learning is word learning, as an infant learning to use a word is, at the same time, learning the semantic category that that word represents (Booth, Waxman, and Y. T. Huang, 2005; Waxman and Gelman, 2009).

As with the popular analogy of System 1 to deep learning and System 2 to symbolic AI systems, there are, it should be noted, complexities to infant learning that render the analogy imperfect. There is evidence that infants learn concepts not just from perceptually obvious features such as size and colour, but also use abstract properties such as agency and function (J. Yin and Csibra, 2015), and are guided not just by salient visual stimuli but also by the attentional focus of an adult social partner (Cleveland, Schug, and Striano, 2007). Additionally, they are biased towards learning certain objects, such as those with shape corresponding to human body parts (Ayzenberg and Lourenco, 2022). Nevertheless, research on infant learning often casts the problem as the learning of discrete objects from a complex and noisy continuous input, particularly in the context of early language acquisition (Bloom, 2002; Clerkin, Hart, Rehg, C. Yu, and Smith, 2017). That is, converting continuous representations of the sort produced by deep neural networks to discrete symbols of the sort that can have properties and relations to one another, is a rough blueprint for concept/word learning in infants. Put the other way around, the learning performed by an infant who acquires new concepts and words from their perceptual stimuli is a useful framework in which to place this thesis.

Machine learning is often divided into supervised and unsupervised. For human concept learning, it is not easy to make such a clear distinction. Much has been said about supervision in human learning, especially word learning (Jansen, Dupoux, Goldwater, M. Johnson, Church, Kenneth, Feldman, Hermansky, Metze, Rose, Clark, Pascal, McGraw, Varadarajan, Bennett, Chiu, Justin, Dunbar, Fourtassi, Harwath, C.-y. Lee, Norouzian, Atta, Peddinti, Richardson, Schatz, and Thomas, 2013; Billman and Knutson, 1996; Kuhl, 2004; Thompson, McKerchar, and Dancho, 2004), but there is no consensus on the role played by explicit instruction from parents or teachers. Also, the learning of a child in response to explicit instruction is certainly not identical to how a supervised machine learning model learns, because the child does not necessarily know what the label is and what the input is, or may not even understand the concept of labelling. Most of the work in this thesis is on unsupervised machine learning. Partly this is because of the practical advantages of not requiring labelled data, but it is also reflective of my own opinion that, of the two common settings in current machine learning, unsupervised is closer to human learning than is supervised.

In summary, the motivation for this thesis is twofold. Firstly, it is a practical one: to

allow machines to bridge the gap between neural systems that excel at perceptual tasks, and symbolic systems that excel at reasoning tasks, and to do so largely without labelled data. As outlined above, this means learning discrete representations of continuous data. Secondly, it is to learn these representations in a similar way to how they are learnt in humans, and so shed light on the origin of our own concepts. The complex, powerful and still mysterious display of learning that is concept formation underlies nearly everything we understand of the world, because nearly everything is understood through concepts we have learnt. What follows is an exploration of techniques that might one day allow the same learning, and perhaps even the same understanding, to occur in a machine.

1.2 Contributions

The work in this thesis can be grouped into three parts.

Deep Clustering The first concerns deep clustering, which is the combination of deep learning for dimensionality-reduction, with a clustering model to form discrete groups of the data. This work is discussed in Chapters 3, 4 and 5, based, respectively, on three conference papers, (Mahon and Lukasiewicz, 2021),(Mahon and Lukasiewicz, 2022) and (Mahon and Lukasiewicz, 2023a), all of which I am first author of. Each of these chapters proposes and analyzes new methods to advance the field of deep clustering. The first builds off a common deep clustering technique, called pseudo-label training, in which cluster labels are fed back to the deep learning component as training targets. I devise an improved method for training with these pseudo-labels, by using an ensemble of clustering models and training only on these points that receive the same label in all ensemble members. This removes noise from the labels and leads to greater accuracy. It still holds the SOTA clustering performance on MNIST and USPS. This work was presented at Informatik 2021. The second paper concerns the application of deep clustering to human activity recognition data from wearable sensors. I propose a new deep clustering HAR model, again based on refining the labels for pseudo-label training, but this time avoiding the need for an ensemble. This model outperforms prior works, where they exist. I also identify some ambiguities with existing evaluation of HAR clustering models, and demonstrate empirically that the results can be significantly affected by one of these ambiguities, namely whether a different model is used for each subject or a single model is used for all subjects. This work is accepted for publication in ACML 2022. The third paper on deep clustering concerns the online setting, in which cluster labels are assigned to each new data point as it is processed, rather than waiting until all points have been processed to assign any labels. Online deep clustering can easily suffer from collapse, in which all points are assigned to the same cluster. Prior methods combat this using data augmentation, by forcing the model to cluster positive pairs together and negative pairs separately. I propose a method that does not require data augmentation, but instead uses a Bayesian formulation to derive an optimization objective for assigning each batch of points. An important difference of my method from prior methods is that I regularize the hard cluster assignments, and encourage them to be uniform, whereas prior works focus on the soft cluster assignments, which I argue to be an inferior approach. In the

paper, I show the difference between hard and soft regularization, and why the former is preferable. This work is under review at ICLR 2022.

Logical Annotation Chapter 6 contains the second area of work, and is the only part of my research in the supervised setting. The discrete representation produced by clustering is very simple, each data point receives a single atomic label. Using a supervised model, with access to annotated data for training, allows us to pursue a more ambitious type of discrete representation, one with internal structure. Specifically, I propose a deep learning system that produces logical annotations of the input data, that is, a set of atoms formed from individuals, classes and relations, and tested this method on the task of annotating videos. For example, if the video was of a man folding a piece of paper, the system could produce the annotation 'fold(man,paper)', where the individuals, such as 'man' and 'paper', and the predicates such as 'fold' are part of a predefined vocabulary. Individuals are represented as a vectors, and predicates as MLPs. At inference, the system uses one multiclassifier component to predict which individuals are present, then runs the corresponding vectors through each MLP and, if the output is over some threshold, adds to the annotation the atom formed from applying that predicate to that individual or pair of individuals. As well as providing an interface between neural and symbolic systems, and being a key component in human cognition as argued above, the production of logical annotations offers some additional, practical advantages. Compared to natural language annotations, they are easier to translate, easier to evaluate, and they restrict our attention to the important semantic information we desire to describe the video contents instead of the difficult syntax of natural language (NL). They also allow interfacing with the semantic web and the inclusion of background knowledge. The paper discusses logical annotation at length and describes the proposed deep learning system. It was present at ICMLA 2020. Currently an MSc student, whom I am co-supervising, is extending this work.

Clustering to Measure Data Complexity Continuing towards more complex discrete representations, Chapter 7 describes the third part of this thesis: a method to represent an image as a hierarchy of clustered patches. Normally, image clustering refers to the task discussed in the above works, where a dataset of images is fed to a clustering model and each image receives a single atomic label. Another direction is to cluster the different parts of a single image. There is some prior work on clustering image patches, mostly in the context of image denoising. My contribution to this line of work is novel in two ways. Firstly, it uses internal image clustering in a novel way, namely to measure image complexity based on the entropy of local multisets of cluster labels. This way of measuring image complexity allows us to avoid being affected by noise, through the use of the minimum description length (MDL) principle. MDL says that we should choose the number of clusters, and which points are outliers, so as to minimize the number of bits needed to represent the entire dataset. The result is that, in very noisy images, most points are designated as outliers, the optimum number of clusters is low, and so the entropy of labels is low. Mine is the first complexity measure to assign white noise images low complexity, all existing methods assign them maximum or near-maximum complexity. The second novelty of my internal image clustering is that I form a hierarchy of patches. The cluster labels produced at one

level of the hierarchy are used as input to the level above. This allows the representation to capture semantics at different levels of locality. To demonstrate that the hierarchical representation is effective, I show the complexity, using the entropy of local multisets of cluster labels, at each level. Some images have lots of fine detail but no global structure, such as repeating, intricate patterns. Other images are low-resolution, or relatively uniform at a small scale, but overall depict something meaningful, such as a low-resolution MNIST digit. The complexity score produced by my method can correctly judge complexity at different scales, such as giving detailed repetitive patterns a high score locally but low score globally, and the opposite for MNIST digits. I then conduct a series of further experiments to validate the scores produced by my proposed complexity metric: the addition of Gaussian noise, the changing of image resolution, and the changing of the fractal dimension. Typically, images are represented by computers as an array of continuous pixel values, but humans tend to understand images as composed, at each location, of a particular type of thing, a certain colour or texture of pattern, which are then combined to form larger things. The representation formed by my clustered hierarchy of patches method is step towards a more human-like understanding.

1.3 Outline

The remainder of thesis is organized as follows:

Chapter 2 introduces general background material, on clustering, unsupervised deep learning and deep clustering. The background related to logical video annotation, and more detailed background on clustering and deep learning, are discussed in the relevant chapters.

Chapters 3, **4** and **5** contain, respectively, the threee works on deep clustering, based on material from (Mahon and Lukasiewicz, 2021), which refines pseudo-label training using an ensemble, (Mahon and Lukasiewicz, 2022), which refines pseudo-label training for human activity recognition and shows how to improve evaluation of HAR clustering, and (Mahon and Lukasiewicz, 2023a), which introduces a method to avoid collapse in online deep clustering.

Chapter 6 describes the supervised logical-annotation model. Based on material from (Mahon, Giunchiglia, B. Li, and Lukasiewicz, 2020), it introduces the task of logical annotation and my proposed model, and presents empirical results on two video annotation datasets.

Chapter 7 discusses the representation of an image as a hierarchy of patches via MDL clustering. It includes a detailed derivation of my method, a proof of correctness in the case of white noise (i.e. that white noise receives a low complexity score), and multiple experiments that demonstrate its ability to measure image complexity.

Finally, **Chapter 8** summarizes the findings and contributions of this thesis, and suggests directions for future work.

Chapter 2

Background

This chapter contains the background material from existing works that is necessary to understand the novel work presented in Chapters 3 to 7. It is divided into three parts. Section 2.1 gives a brief introduction to machine learning in general, and the two main classes of machine learning models considered in this thesis: deep learning and clustering. Section 2.2 concerns clustering; Section 2.3 concerns deep learning, and its relation to clustering.

2.1 Machine Learning, Deep Learning and Clustering

Machine learning is a general approach to building computational systems that leverage data in order to improve their performance. A more precise definition can come in various forms, such as a set of techniques for statistical inference and prediction (K. P. Murphy, 2012) or as the study of self-improving algorithms (Mitchell and Mitchell, 1997). The definition used here is that machine learning is an approach to programming in which, rather than specify the algorithm exactly as in conventional programming, some aspects of the algorithm are free to change, and are then selected so as to optimize performance on some data (referred to as training data). This definition is similar to Andrej Karpathy's concept of Software 2.0, referring to a new way to design software https://karpathy.medium.com/software-2-0-a64152b37c35. One might make different choices as to which parts of the algorithm are fixed and which are free to change based on the training data, as well as what is to be optimized on the training data. Each of these choices constitutes a machine learning model. It is useful to distinguish between certain classes of machine learning models as supervised vs unsupervised, based on the nature of the training data. Supervised models require training data consisting of input-output pairs, where the output specifies the desired output of the model given the corresponding input, e.g., inputs are images and outputs are labels specifying what the input image is of. This thesis concerns two classes of machine learning models: clustering models and neural networks. Clustering is an unsupervised task that aims to find a partition of the given data such that data points with the same subset are similar and data points within different subsets are different. Neural networks are a type of model originally inspired by the brain, consisting of a (normally large) number of neurons. Some neurons have weighted directed connections to others, and each neuron emits an output that depends on the weighted sum of the outputs of the neurons connected

to it. Deep learning refers to the study of deep neural networks, that is networks with many sequentially connected layers of neurons, and to neural networks more generally. The following material assumes knowledge of the basic concepts in deep learning. Readers wishing for an introduction to the subject are referred to LeCun, Bengio, and G. Hinton, 2015 and Goodfellow, Bengio, and Courville, 2016 for brief and detailed introductions, respectively.

2.2 Three Interpretations of Clustering

The technique of clustering is well-suited to pursuing the goal of this thesis, because it can take continuous data as input and produce a discrete cluster label as output. Quantifying the quality of a given clustering is not straightforward, because it can have different use cases and interpretations. This section outlines three different interpretations of clustering–similarity and difference, unsupervised classification, and compression–and discusses the metrics corresponding to each.

2.2.1 Clustering Interpretation 1: Similarity and Difference

By the standard definition, clustering is the search for a partition of the data such that points in the same subset are similar and points in different subsets are different. Several clustering metrics reflect this conception of clustering. For example, two common unsupervised clustering metrics are the Davies-Bouldin index (Davies and Bouldin, 1979) and the silhouette index (Rousseeuw, 1987). Both measure the quality of a given partition of a dataset S into clusters $C_1, \ldots, C_K$, based on a notion of distance $d : S \times S \to \mathbb{R}$. Specifically, the Davies-Bouldin index is defined as

$$\sum_{i=1}^{N} \max_{j \neq i} \frac{\sigma_i + \sigma_j}{d(c_i, c_j)}\,, \tag{2.1}$$

where c_i is the centroid of the cluster of the ith data point, σ_i is the diameter of the ith cluster, and N is the size of the dataset. It takes values in $[0, \infty)$, and lower is better.

The silhouette index, for a given point i is defined as

$$s(i) = \begin{cases} \frac{b(i)-a(i)}{\max\{a(i),b(i)\}}, & \text{if } a(i) < b(i) \\ 0, & \text{otherwise} \end{cases} \tag{2.2}$$

where $a(i)$ and $b(i)$ denote, respectively, how far point i is from its own cluster, and how far it is from the nearest neighbouring cluster:

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j) \tag{2.3}$$

$$b(i) = \min_{J \neq I} \frac{1}{|C_J|} \sum_{j \in J} d(i, j)\,. \tag{2.4}$$

The silhouette index for the entire clustered dataset is the mean silhouette index for each point: $\sum_{i \in S} s(i)$. It takes values in $[-1, 1]$, and higher is better.

Both of these metrics reward clusterings in which points within the same clustering are simliar, and penalize when points from different clusters are similar. They are both unsupervised metrics, also known as internal metrics, in that they do not make reference to any ground truth labels of the data.

There are also a number supervised metrics, which are in fact more commonly used in current research. One such supervised metric, which is also based on maximizing intra-cluster similarity, and minimizing inter-cluster similarity, is the Rand index (Rand, 1971). This considers all pairs of points, and measures what fraction were correctly placed either together or separately, that is, what fraction of points with the same ground truth label were put in the same cluster, and what fraction with different ground truth labels were put in different clusters. Calling points with the same ground-truth label similar, and those with different labels dissimilar, the Rand index measures the fraction of points the model correctly identified as being the same or different. Formally, the Rand index, Rand index (RI), is defined as

$$RI(X, y, \tilde{y}) = 1 - \frac{\sum_{i=1}^{N} \sum_{j>i}^{N} \mathbb{1}(y(x_i) = y(x_j)) \oplus \mathbb{1}(\tilde{y}(x_i) = \tilde{y}(x_j))}{\binom{N}{2}}, \tag{2.5}$$

where $y, \tilde{y}$ are, respectively, the ground truth and model label functions on the dataset $X = (x_i)_{1 \leq i \leq N}$, and $\oplus$ denotes XOR. Even for randomly assigned labels, some of these pairs will agree by chance. Often, an adjustment is added so that randomly assigned labels receive a score of zero in expectation. The proposed variant, referred to as the adjusted Rand index, was proposed in the original (Rand, 1971) as

$$ARI(X, y, \tilde{y}) = \frac{\sum_{i=i}^{N} \binom{n_{ij}}{2} - (\sum_{i=1}^{N} \binom{a_i}{2})(\sum_{i=1}^{N} \binom{b_i}{2}) / \binom{N}{2}}{1/2(\sum_{i=1}^{N} \binom{a_i}{2} + \sum_{i=1}^{N} \binom{a_i}{2}) - (\sum_{i=1}^{N} \binom{a_i}{2})(\sum_{i=1}^{N} \binom{b_i}{2}) / \binom{N}{2}}, \tag{2.6}$$

where

$$\begin{aligned} a_i &= |\{x \in X : y(x) = i\}| \\ b_i &= |\{x \in X : \tilde{y}(x) = i\}| \\ n_{ij} &= |\{x \in X : y(x) = i \wedge \tilde{y}(x) = j\}|. \end{aligned}$$

2.2.2 Clustering Interpretation 2: Unsupervised Classification

Another way to interpret clustering is as unsupervised classification. We can measure clustering accuracy as the fraction of points that receive the same cluster label as the ground truth label. Applied naively, however, this will not work as the order of labels used by the model might not be the same as the order used in the ground truth. For example, if we were clustering images of cats and dogs, then 'dog' might be represented as a '0' in the ground truth but as a '1' in the clustering model. If so, then we should count as correct those images that the ground truth labels '1' and the clustering model labels '0' or vice versa. What is needed is a translation function, that specifies the corresponding ground truth label for each

model label. Assuming the clustering model has learnt anything meaningful at all, the most likely tranlsation for a particular model label, is the ground truth label that it coincides with most often. Thus, the most appropriate translation is the one which renders the model most accurate. We may also wish to enforce that distinct model-labels are translated as distinct ground truth labels, so that if two different model labels m_i, m_j both coincide most with the same ground truth label g_k, then they cannot both be translated as g_k. If we assume that the number of distinct model labels, k, equals the number of distinct ground truth labels, which is often the case, as the user typically sets the number of clusters to be equal to the number of ground truth classes, then this gives rise to the following optimization problem

$$\underset{\sigma \in S_k}{\operatorname{argmax}} \left| \{x \in X : \sigma(\tilde{y}(x)) = y(x)\} \right|,$$

where S_k is the permutation group on k elements. Equivalently, we can express the problem in terms of the confusion matrix formed from the model labels and ground truth labels.

$$A = \begin{bmatrix} n_{11} \dots n_{1k} \\ \vdots \ddots \vdots \\ n_{k1} \dots n_{kk} \end{bmatrix},$$

where n_{ij} are defined as in (2.6). The problem then is to select k entries from A with maximal sum such that no two entries are in the same row or the same column. This is the linear sum assignment problem. The linear sum assignment problem can be described as having a set of jobs that need to be done and a set of workers each of which charges a different amount to perform each job. The problem is to assign each job to a different worker so as to minimize the total cost. Similar to A, the costs of assigning jobs to workers can be expressed by a cost matrix, whose (i, j)th entry is the cost of assigning job i to worker j. This is a minimization problem, but can be converted to a maximization problem by taking the negative of the cost matrix: the linear assignment problem on $-A$ is the same as the above problem of aligning cluster labels.

A solution to this problem was first proposed by Kuhn (1955), and refined in (Kuhn, 1956). Soon afterward, Munkres (1957) proved its correctness, and so the solution became known as the Kuhn-Munkres algorithm. The original algorithm from Kuhn and Munkres ran in $O(k^4)$, but was improved, separately, by Tomizawa (1971) and Edmonds and Karp (1972) to $O(k^3)$. It is the improved version that I use in various parts of the work from Chapters 3, 4 and 5, and that I now describe. The algorithm is based on the observation that subtracting a constant from any row or column does not change the optimal assignment, because all assignments must pick exactly one or the altered elements, so all will have their cost changed by the same amount. The first step in the algorithm is to subtract from each row, the minimum element in that row, and then do the same for each column. Then every row and every column will contain at least one zero. If we can assign k zeroes at this point, then we are done, because all entries are non-negative so the minimum possible assignment cost is zero. If we cannot, then the algorithm next draws horizontal and vertical lines through the matrix, drawing the minimum number needed to cover all zeroes. (See the original (Kuhn, 1955) for a description of the subroutine used to do this.) We can make exactly one assignment of zero for each line drawn, so we want to increase the number of

lines. This can be done by selecting the smallest uncovered element, subtracting it from each uncovered row and adding it to each covered column. This is equivalent to subtracting it from all uncovered elements and adding it to all doubly covered elements. Now the smallest uncovered entry has become zero, so we need to draw another line to cover it. It is possible that the new line also adds a second cover to some other zeroes, in which case the line previously covering those zeroes could possibly be removed and we would not actually have increased the number of lines. However, it is shown in (Munkres, 1957) that repeated application of this process will eventually be able to increase the number of lines. Thus the number of lines can continue to be increased until it equals k, at which point we can make an assignment of cost zero, which is minimal as all weights are non-negative. The changes to the matrix did not change the optimal assignment, so this is an optimal assignment to the original problem too.

For the case in which the number of model clusters $\tilde{k}$ is greater than the number of ground truth clusters k, the cost matrix instead will be rectangular $\tilde{k} \times k$ matrix. The above algorithm will not work on a rectangular matrix, because the minimum number of lines to cover all zeroes will never be less than $\min(\{\tilde{k}, k\})$, and we want to make $\max(\{\tilde{k}, k\})$ assignments. However, we can modify the matrix to make the algorithm applicable. As $\tilde{k} > k$, the matrix is tall and thin, but we can make it square as follows. First, we duplicate the matrix $\lceil \frac{\tilde{k}}{k} \rceil$ times horizontally, and then add $k\lceil \frac{\tilde{k}}{k} \rceil - \tilde{k}$ dummy rows at the bottom of all zeroes to make it square. The horizontal duplication is equivalent to allowing multiple model clusters to be assigned to the same ground truth cluster. The choice to duplicate the minimum number of times needed to allow all model clusters to be assigned, namely $\lceil \frac{\tilde{k}}{k} \rceil$ times, means requiring the same number of assignments (up to integer rounding) to each ground truth cluster. The addition of dummy rows is equivalent to including some extra model clusters, each of whose assignment cost is the same for ground truth clusters, and who, therefore, will not affect the other assignments.

The most common approach to using supervised metrics to measure clustering performance, and the one taken in the work presented here in Chapters 3, 4 and 5, is to manually set the number of clusters equal to the number of ground truth classes and compute a translation function from model-labels to ground truth labels using the Kuhn-Munkres algorithm. Supervised metrics, such as accuracy, precision, recall and micro-/macro-F1 can then be measured in the normal way. These are useful because they enable a direct comparison to classifiers, and reveal the size of the gap between supervised and unsupervised models at the specific task of classification.

2.2.3 Clustering Interpretation 3: Compression

A third way to measure clustering performance, and one especially relevant to the work of this thesis, is to treat clustering as compression. After clustering, we can replace each data point, which is often large and high-dimensional, as in the case of images, with a single integer from $0, \ldots, k-1$. For an accurate clustering, this cluster label should encode the maximum amount of useful information about the data point. That is, the cluster labels act as a compression code, and we want to judge how much information this compression code contains about the uncompressed original. Not all parts of the input are equally important,

however. Some regions of an image, for example, may be quite irrelevant to its semantic content and we do not want to penalize the model for failing to represent such regions. In fact, we would rather the model did not represent them as they are, semantically speaking, noise, not signal. Therefore, rather than measuring the mutual information between the cluster labels and the corresponding data points, a more common approach is to measure the mutual information between the cluster labels and the ground truth labels. The ground-truth labels are thought of as the ideal compression code, which contain most or all of the relevant information about the input, and we then measure the information-theoretic distance between them and the compression code resulting from the model labels.

Similarly to the Rand index above, the mutual information is more commonly used in a normalized form, so that the maximum possible value is 1. Mutual information can be written as the difference between the entropy and the conditional entropy

$$\begin{aligned} I(X;Y) =& H(X) - H(X|Y) \\ =& H(Y) - H(Y|X)\,, \end{aligned}$$

where H denotes entropy. As the entropy of any distribution is non-negative, the maximum value for $I(X;Y)$ is $\min(\{H(X), H(Y)\})$. Normalized mutual information (normalized mutual information (NMI)), then, divides by some average of $H(X)$ and $H(Y)$, which is $\geq \min(\{H(X), H(Y)\})$ so guarantees a result ≤ 1. Different variants use different averages, arithmetic mean, geometric mean, harmonic mean etc, and it is unsatisfactory that these all give slightly different results, but there is little reason to select one over the others. A case is made in (Strehl and Ghosh, 2002) that the geometric mean is to be preferred because the resulting expression

$$\frac{I(X;Y)}{\sqrt{H(X)H(Y)}}\,.$$

is analogous to the normalized inner product on a Hilbert space, and also, more specifically, to the Pearson correlation coefficient

$$\frac{\mathrm{Cov}(X;Y)}{\sqrt{\mathrm{Var}(X)\,\mathrm{Var}(Y)}}\,, \tag{2.7}$$

This argument is strengthened by there already being a congruence between mutual information and covariance in the fact that $I(X;X) = H(X)$ and $\mathrm{Cov}(X,X) = \mathrm{Var}(X)$.

2.2.4 Effect of Number of Clusters on Different Clustering Metrics

Here, I compare how NMI, ARI and ACC/F1 respond to changes in the number of predicted clusters.

NMI. A downside of NMI, is that random labellings still get a score > 0. By chance alone, there is likely to be some difference between the distribution of ground truth (GT) labels, and the conditional distribution of GT labels given the model label. As the mutual information can be equivalently expressed as the Kullback-Leibler (divergence) (KL) divergence between

these two distributions, it is likely to be strictly positive. Relatedly, NMI does not penalize for assigning different labels to points with the same ground truth label, it only penalizes for the opposite: assigning the same label to points with different ground truth labels. To illustrate this, take the extreme case of a model that assigns every point a different cluster label, i.e. has N clusters. This will have (unnormalized) mutual information of $\log k$ with the ground truth labels, where k is the number of ground truth clusters, because the entropy of the model labels is $\log N$, and their conditional entropy given the ground truth labels is $\log \frac{N}{k}$. Assuming the normalization from (2.7), this would give an NMI of

$$NMI = \frac{\log k}{\sqrt{H(Y) \log N}} ,$$

where $H(Y)$ is the entropy of the ground truth labels. As there are k, clusters in the ground truth clustering, $H(Y) \leq \log k$, so we can compute a lower bound on the NMI as

$$NMI = \frac{\log k}{\sqrt{\log N H(Y)}} \geq \frac{\log k}{\sqrt{\log N \log k}} = \sqrt{\frac{\log k}{\log N}} .$$

For many common values ranges of N and k, this lower bound is significantly greater than 0. For example, MNIST has 60000 data points of 10 different classes, giving

$$\sqrt{\frac{\log 10}{\log 60000}} \approx 0.457 .$$

This ability to increase NMI by artificially increasing the number of clusters is a disadvantage of the metric.

Supervised Metrics. Clustering accuracy, after alignment as discussed above, also has these undesirable properties, that random labellings score > 0, and that increasing the number of clusters in the model can further increase the score artificially. In fact, there the situation is even worse because giving each point a different cluster will result in perfect accuracy. Each model cluster has a single element, and so can be assigned the unique ground cluster that this element belongs to. The same is true for precision, recall and micro/macro-F1. All of these metrics will give a better and better score as the number of model clusters is increased, up to a perfect score for the N model clusters.

ARI. This is perhaps the most robust single metric, because it does not have these disadvantages. It can assign negative scores, and it assigns random clusterings a score of 0 in expectation. It also cannot be artificially improved by increasing the number of model clusters. If only one metric is to be used, I would therefore favour ARI. However, in practice we can normally consider other metrics too to capture different aspects of clustering, such as accuracy to compare to supervised models, and NMI to measure the relevant information lost in using the cluster labels as compression code.

2.3 Unsupervised Deep Learning

2.3.1 Unsupervised Deep Learning for Feature Extraction

As outlined in Section 1.1, this thesis is interested primarily, though not exclusively, in unsupervised learning, because of the greater similarities to human concept learning. Unsupervised deep learning, i.e. training neural networks without labelled data, has long been an area of interest to machine learning researchers. Early examples of unsupervised neural networks are Hopfield networks (Hopfield, 1982), restricted Boltzman machines (Smolensky, 1986) and deep belief networks (G. E. Hinton, Osindero, and Teh, 2006; G. E. Hinton, 2009).

A more recent class of unsupervised deep learning models are autoencoders (Kramer, 1991; G. E. Hinton and R. Zemel, 1993; R. S. Zemel and G. E. Hinton, 1994). Autoencoders are composed of an encoder which converts the input to a vector representation, often called the hidden or latent representation, and a decoder which then tries to reconstruct the input from the hidden representation. The error from the decoder in reconstructing the input can be backpropagated to the encoder too. Thus, unlike Hopfield networks, Boltzmann machines and Belief nets, autoencoders can be trained using gradient descent, as most modern deep learning models are. In order to prevent the networks from simply learning the identity function, the size of the latent vector is typically constrained to be smaller than that of the input. There are many variants of autoencoders, such as denoising autoencoders (Vincent, Larochelle, Lajoie, Bengio, Manzagol, and Bottou, 2010) which add noise to the input and train the decoder to reconstruct the original, unnoised version. Another version, which has attracted much attention in recent years, is variational autoencoders (VAEs) (Kingma and Welling, 2013). VAEs leverage the structure of an autoencoder to perform variational inference, where the latent vector, i.e. the output of the encoder, contains the latent variables. The encoder then represents the conditional distribution $p(z|x)$ for input x, and the decoder is treated as an approximation to the reversed conditional $q(x|z) \approx p(x|z)$. This allows a straightforward optimization to maximize the model evidence and simultaneously minimize the Kullback-Leibler divergence of $q(x|z)$ from $p(x|z)$. See (Goodfellow, Bengio, and Courville, 2016) Chapter 14, or (Bank, Koenigstein, and Giryes, 2020) for more on different varieties of autoencoders.

A common use-case for unsupervised deep learning has often been pretraining, followed by fine-tuning using supervised learning (Bengio, Lamblin, Popovici, and Larochelle, 2006). Soon after its introduction in 2010, the Imagenet Large-scale Visual Recognition Challenge (Russakovsky, J. Deng, Su, Krause, Satheesh, S. Ma, Z. Huang, Karpathy, Khosla, Bernstein, A. C. Berg, and Fei-Fei, 2015) became a very popular benchmark in computer vision, around which much of the field coalesced. Although a supervised benchmark, it was also the basis of evaluating many unsupervised learning techniques, by seeing how effective these techniques were in pretraining. In 2012, ILSVRC was won by a large margin by AlexNet, the first deep learning winner of the contest, and the beginning of the deep learning revoluation in computer vision. AlexNet was a purely supervised method, trained without any pretraining (Krizhevsky, Sutskever, and G. E. Hinton, 2012), which temporarily caused interest to shift away from unsupervised pretraining, and the string of landmark papers that followed AlexNet also trained from scratch (Simonyan and Zisserman, 2014; Szegedy, W. Liu, Jia, Sermanet, Reed, Anguelov, Erhan, Vanhoucke, and Rabinovich, 2015; Szegedy, Vanhoucke,

Ioffe, Shlens, and Wojna, 2016; K. He, X. Zhang, S. Ren, and J. Sun, 2016). However, in the past two to three years, unsupervised pretraining has become popular again, driven in part by the difficulty of obtaining sufficiently large labelled datasets, and is currently receiving much research attention (Caron, Bojanowski, Joulin, and Douze, 2018; Caron, Misra, Mairal, Goyal, Bojanowski, and Joulin, 2020; Grill, Strub, Altché, Tallec, Richemond, Buchatskaya, Doersch, Pires, Z. Guo, Azar, Piot, k. k. k., Munos, and Valko, 2020; J. Li, P. Zhou, Xiong, and Hoi, 2020; D. L. Chen and Dolan, 2011; J. Huang and S. Gong, 2021; Y. Cai, Z. Zhang, Y. Liu, Ghamisi, K. Li, X. Liu, and Z. Cai, 2021; Zbontar, Jing, Misra, LeCun, and Deny, 2021). The form of unsupervised learning covered in this thesis differs from these works in that it seeks a discrete representation, rather than a continuous vector representation, but the techniques used for the former are relevant to the latter, and will be discussed further below, particularly in Chapters 3 and 5.

Another area in which unsupervised deep learning has received much attention is natural language processing (natural language processing (NLP)), and there it has shown great success. Word vectors (Mikolov, K. Chen, Corrado, and Dean, 2013) have enabled DNNs to be trained on large quantities of unlabelled text, initially as neural language-models, and later as auto-regressive models (Devlin, M.-W. Chang, K. Lee, and Toutanova, 2018). However, despite being a rich area of unsupervised deep learning, NLP does not fit into the general theme of my DPhil research. It does not require converting high-dimensional data to discrete representations, because word tokens are already discrete and relatively low-dimensional. In fact, word vectors move in the opposite direction, converting discrete tokens into continuous vectors. Beyond a brief mention in Chapter 6, NLP is not discussed further in this thesis.

2.3.2 Deep Learning and Clustering

The goal of this thesis is to form discrete representations of continuous data. Clustering is a useful tool in this respect, because it takes continuous data as input and outputs discrete cluster labels. However, if the input is high-dimensional, the curse of dimensionality can make it difficult to apply clustering models. The curse of dimensionality (Bellman, 1966) refers to a number of properties of the notion of distance in Euclidean spaces that emerge as the dimension increases, and that interfere with the performance of methods designed for lower dimensions. The most problematic aspect of the curse of dimensionality for clustering is that, as dimension increases, distance becomes less meaningful as a discriminator between different pairs of points. Clustering is based on the relative similarity between pairs of data points, which, for real-valued data, is inferred from the distance between them; but for high-dimensional data, the distance between any two pairs of points tends to be roughly the same. More specifically, if we keep increasing the dimension n of the data X by continually adding new coordinates to all data points, then, for any reference point $x_0 \in X$,

$$\lim_{n\to\infty} \frac{\max_{x\in X} ||x - x_0|| - \min_{y\in X} ||y - x_0||}{\max_{x\in X} ||x - x_0||} = 0\,. \tag{2.8}$$

(Full proof in (Beyer, Goldstein, Ramakrishnan, and Shaft, 1999).) One solution is to reduce the dimensionality of the data before applying the clustering algorithm. Some works use non-neural dimensionality reduction techniques for clustering, (Tasoulis, Pavlidis, and

Roos, 2020; Song, H. Yang, Siadat, and Pechenizkiy, 2013). Some works have even shown improved results by projecting onto a random subset of dimensions (Boutsidis, Zouzias, Mahoney, and Drineas, 2014; Dasgupta, 2000; D. Niu, Dy, and Jordan, 2011). The recent investigation in (Allaoui, Kherfi, and Cheriet, 2020) of UMAP (McInnes, Healy, and Melville, 2018) to improve clustering is especially revealing.

Different from these non-neural dimensionality-reduction techniques, this thesis considers deep learning encoders for dimensionality reduction. Deep learning has proved highly successful at extracting meaningful lower-dimensional representations from high-dimensional data, especially in the case of perception-like data such as images and speech audio, which is just the sort of data that humans process and convert into abstract symbols such as words or objects. One of the main strengths of deep learning is that it can extract features hierarchically, each layer extracting features from the layer before, and so it can operate effectively with very high-dimensional data (LeCun, Bengio, and G. Hinton, 2015; Zeiler and Fergus, 2014; L. Deng and D. Yu, 2014).

Thus, with respect to the goal of this thesis, it is natural to combine the techniques of deep learning and clustering. The former can convert high-dimensional continuous representations into low-dimensional continuous representations, and the latter can convert low-dimensional continuous representations into discrete representations. First applying the deep neural network (DNN) to reduce the dimension, rather than clustering the high-dimensional data directly, allows the cluster model to avoid the curse of dimensionality. The question that then arises is how to train the deep learning encoder. Current deep learning models are trained using gradient descent with the gradient produced by a prediction error. Often this comes through comparing a predicted label to a ground truth label, but if we already had ground truth labels then we could use them to partition the dataset and would not need to cluster in the first place. Another approach is to first conduct unsupervised pretraining using one of the methods discussed in Section 2.3.1, which is based on using domain knowledge to construct clever loss functions in the absence of labels, mostly data augmentations. However, using these methods here would mean isolating the training of the encoer from the training of the clustering model, i.e. first pretraining the encoder and then freezing it during clustering. The approaches considered in this thesis instead integrate the two tasks, encoder training and clustering, so that information from the training of the clustering module can be fed back to the training of the encoder. This allows the encoder to specialize in creating clustering-friendly representations.

Two general methods for such integrated training are introduced here, and then further details are left to the relevant chapters below.

Differentiable Clustering Objective The first is to simply express the clustering objective as a differentiable function of the encoder parameters, and then optimize using gradient descent. The clustering objective can be expressed as decreasing the distance of each point from its centroid, and increasing the distance from other centroids, as in the Davies-Bouldin index in (2.1) or the silhouette index in (2.2). This can easily be expressed as a differentiable function of model parameters, but unfortunately, optimizing it with gradient descent admits a trivial solution where the encoder maps every input to the same point and every point is put in the same cluster, and so additional means need to be taken to avoid such a collapsed

state.

Pseudo-label Clustering The second integrated clustering objective is known as pseudo-label training, as introduced by Caron, Bojanowski, Joulin, and Douze (2018). Pseudo-label training begins with a randomly initialized convolutional neural network (CNN) encoder for an image dataset. The entire dataset of N data points is encoded to produce N feature vectors, and the feature vectors are then clustered. As the encoder has not been trained, these feature vectors will not be a precise representation of the inputs, but they will nevertheless contain some meaningful information due to the priors encapsulated by the CNN architecture and their suitability to image feature extraction (pseudo-label training was initially proposed for the image domain, and it is mostly there that it has been applied, but it can be adapted to other domains by changing the encoder suitably, e.g. (Krishna and Ganapathy, 2022; H. Ma, Z. Zhang, W. Li, and S. Lu, 2021)). Thus, the resulting clustering will be more accurate than chance. These cluster labels are then used as training targets to train the encoder by attaching a classification head and training as a classifier. After training the encoder for some number of epochs, the inputs are re-encoded and clustered again and the process is iterated.

These two methods, pseudo-label training and minimizing the clustering objective with gradient descent, are the subjects of the work discussed in Chapters 3, 4 and 5, where they are discussed in further detail.

Chapter 3

Selective Pseudo-label Clustering

Abstract

The most accurate existing methods for deep learning and clustering combine the training of the deep neural network (DNN) with the clustering objective, so that information from the clustering process can be used to update the DNN to produce better features for clustering. One problem with this approach is that the "pseudo-labels" produced by the clustering algorithm are noisy, and any errors that they contain will hurt the training of the DNN. In this chapter, I propose selective pseudo-label clustering, which uses an ensemble of autoencoders and clustering models to select only the most confident pseudo-labels for training the DNN. I formally prove the performance gains under certain conditions. Applied to the task of image clustering, the new approach achieves state-of-the-art performance on three popular image datasets. Code is available at https://github.com/Lou1sM/clustering.

3.1 Introduction

As outlined in Section 2.3, there are three common approaches to training the encoder in deep clustering models: unsupervised pretraining (then followed by a separate clustering phase), clustering-loss which allows the encoder and the clustering loss to be optimized jointly, and pseudo-label training. All three are discussed further in Section 3.2. The model I present in this Chapter, *selective pseudo-label clustering* (*SPC*), combines an unsupervised pretraining loss, namely reconstruction loss for an autoencoder, and pseudo-label training loss. It uses an ensemble to select the loss function for each data point, depending on how confident the predictions are of their cluster membership.

Ensemble learning is a method of function approximation where multiple approximating models are trained, and then the results are combined in some way. In order for an ensemble to confer a more accurate approximation, variance across the ensemble is required. If all individual approximators were identical, there would be no gain in combining them. For ensembles composed of DNNs, variance is ensured by the random initializations of the weights and stochasticity of the training dynamics. In the simplest case, the output of the ensemble is the average of each individual output (mean for regression and mode for classification) (Opitz and Maclin, 1997).

When applying an ensemble to clustering problems (referred to as consensus clustering; see (Boongoen and Iam-On, 2018) for a comprehensive discussion), the sets of cluster labels must be aligned across the ensemble. In my implementation, this is performed using the Kuhn-Munkres algorithm (Section 2.2) to translate each clustering into the ground truth labels. SPC considers a clustered data point to be confident if it received the same cluster label (after alignment) in each member of the ensemble. The intuition is that, due to random initializations and stochasticity of training, there is some non-zero degree of independence between the different sets of cluster labels, so the probability that all cluster labels are incorrect for a particular point is less than the probability that a single cluster label is incorrect.

My main contributions in this chapter are briefly summarized as follows.

- I describe a generally applicable deep clustering method (SPC), which treats cluster assignments as pseudo-labels, and introduces a novel technique to increase the accuracy of the pseudo-labels used for training. This produces a better feature extractor, and hence a more accurate clustering.

- I formally prove the advantages of SPC, given some simplifying assumptions. Specifically, I prove that my method does indeed increase the accuracy of the targets used for pseudo-label training, and that this increase in accuracy does indeed lead to a better clustering performance.

- I implement SPC for image clustering, with state-of-the-art performance on three popular image clustering datasets, and I present ablation studies on its main components. The code is available at https://github.com/Lou1sM/clustering.

The rest of this chapter is organized as follows. Section 3.2 gives an overview of related work. Sections 3.3 and 3.4 give a detailed description of SPC and a proof of

correctness, respectively. Section 3.5 presents and discusses the experimental results, including a comparison to existing image clustering models and ablation studies on the main components of SPC. Finally, Section 3.6 summarizes my contributions and gives an outlook on future work.

3.2 Related Work

One of the first deep image clustering models was (P. Huang, Y. Huang, W. Wang, and L. Wang, 2014). It trains an autoencoder (autoencoder (AE)) on reconstruction loss (rloss), and then clusters in the latent space, using additional loss terms to make the latent space more amenable to clustering.

In (B. Yang, Fu, Sidiropoulos, and Hong, 2017), the training of the encoder is integrated with the clustering. A second loss function is defined as the distance of each encoding to its assigned centroid. It then alternates between updating the encoder and clustering by k-means. A different differentiable loss is proposed in (J. Xie, R. Girshick, and Farhadi, 2016), based on a soft cluster assignment using Student's t-distribution. The method pretrains an AE on rloss, then, like (B. Yang, Fu, Sidiropoulos, and Hong, 2017), alternates between assigning clusters and training the encoder on cluster loss. Two slight modifications were made in later works: X. Guo, L. Gao, X. Liu, and J. Yin (2017) added the use of rloss after pretraining, and Ghasedi Dizaji, Herandi, C. Deng, W. Cai, and H. Huang (2017) added regularization to encourage equally-sized clusters.

B. Gao, Y. Yang, Gouk, and Hospedales (2020) replace this alternating optimization with a clustering loss that allows cluster centroids to be optimized directly by gradient descent. The result is an online clustering method, which clusters and trains the encoder simultaneously. All the other existing methods discussed in this chapter, along with the proposed method SPC are offline and alternate between training the encoder and clustering. Online deep clustering is the subject of Chapter 5.

Pseudo-label training, as described in Section 2.3, is introduced by Caron, Bojanowski, Joulin, and Douze (2018). Although this method had a substantial effect on the field of deep clustering and unsupervised deep learning in general, it is difficult to compare it quantitatively with other models, as the work does not report clustering accuracy directly.

Some methods are based on prior knowledge that certain data points should be assigned to the same cluster. Haeusser, Plapp, Golkov, Aljalbout, and Cremers (2018), for example, generate four transformed versions of each image using reflections, rotations, and brightness shifts, and then assign them to the same cluster as the original image. This teaches the clustering to be invariant to the specific transformations applied. Prior knowledge is also used by Abavisani and Patel (2018), to train on pairs of images at a time, specifically on two transformed versions of the same digit.

Generative adversarial networks (Goodfellow, Pouget-Abadie, Mirza, B. Xu, Warde-Farley, Ozair, Courville, and Bengio, 2014) (generative adversarial network (GAN)s) have produced impressive results in image synthesis (Karras, Laine, and Aila, 2019; Elgammal, B. Liu, Elhoseiny, and Mazzone, 2017; Brock, Donahue, and Simonyan, 2018), and can also be used for clustering. At the time of writing, the most accurate GAN-based image clustering models (Mukherjee, Asnani, E. Lin, and Kannan, 2019; F. Ding and Luo, 2019)

design a generator to sample from a latent space that is the concatenation of a multivariate normal vector and a categorical one-hot encoding vector, then recover latent vectors for the input images as in (Creswell and Bharath, 2018; Lipton and Tripathi, 2017), and cluster the latent vectors. A similar idea is employed in (Z. Jiang, Zheng, H. Tan, Tang, and H. Zhou, 2016), though not in an adversarial setting. For more details on GAN-based clustering, see (P. Zhou, Hou, and Feng, 2018; W. Zhao, S. Wang, Z. Xie, J. Shi, and C. Xu, 2018; F. Ding and Luo, 2019; J. Liang, J. Yang, H.-Y. Lee, K. Wang, and M.-H. Yang, 2018; Y. Wang, L. Zhang, Nie, X. Li, Z. Chen, and F. Wang, 2019) and the references therein.

Adversarial training is used for regularization by Mrabah, Bouguessa, and Ksantini (2019). This is based on prior work from the same authors (Mrabah, N. M. Khan, Ksantini, and Lachiri, 2019). Conflicted data points are identified as those whose maximum probability across all clusters is less than some threshold, or whose max and next-to-max are within some threshold of each other. Pseudo-label training is then performed on the unconflicted points only. A similar threshold-based filtering method is employed by J. Chang, L. Wang, G. Meng, S. Xiang, and C. Pan (2017).

A final model to consider is that of McConville, Santos-Rodriguez, Piechocki, and Craddock (2019), which uses a second round (i.e., after the DNN) of dimensionality reduction via UMAP (McInnes, Healy, and Melville, 2018), before clustering. Many of the above regularizations of the latent space aim to encourage some form of locality. UMAP preserves local structure, so it is expected to take the place of such regularizations, also shown empirically in (McConville, Santos-Rodriguez, Piechocki, and Craddock, 2019).

3.3 Method

Pseudo-label training is an effective deep clustering method, but training on only partially accurate pseudo-labels can hurt the encoder's ability to extract relevant features. Selective pseudo-label clustering (SPC) addresses this problem by selecting only the most confident pseudo-labels for training, using the four steps shown in Fig. 3.1.

1. Train K autoencoders in parallel.
2. Cluster in the latent space of each, to obtain K sets of pseudo-labels.
3. Select for pseudo-label training, those points are those that received the same label in all K sets of pseudo-labels, after the labellings have been aligned using the Kuhn-Munkres algorithm.
4. Train on the selected pseudo-labels. Go back to (2).

Training ends when the number of agreed points stops increasing. Then, each data point is assigned its most common cluster label across the (aligned) ensemble.

3.3.1 Formal Description

Given a dataset $\mathcal{X} \subseteq \mathbb{R}^n$ of size N with C ground truth clusters, let $(f_j)_{1\leq j\leq K}$, $f_j : \mathbb{R}^n \to \mathbb{R}^m$, and $(g_j)_{1\leq j\leq K}$, $g_j : \mathbb{R}^m \to \mathbb{R}^n$ be the K encoders and decoders, respectively. Let

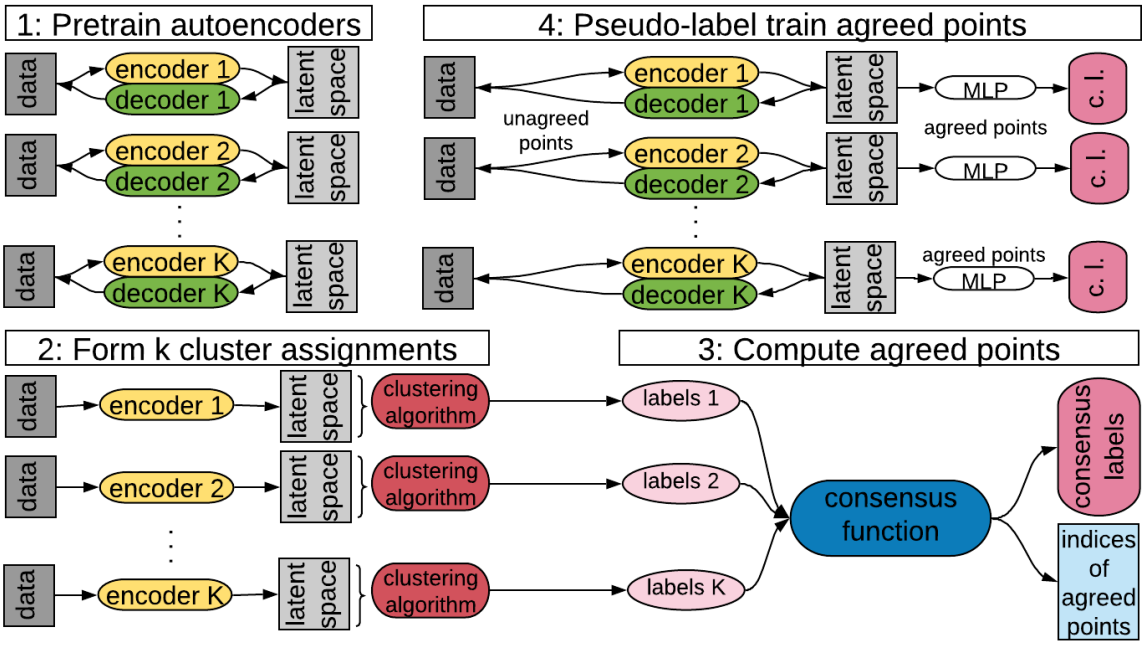

Figure 3.1: The complete SPC method. (1) Pretrain autoencoders. (2) Perform multiple clusterings independently. (3) Identify agreed points as those that receive the same label in all ensemble members. (4) Perform pseudo-label training on agreed points and autoencoder training on unagreed points. Steps (2)–(4) are looped until the number of agreed points stops increasing.

$\psi : \mathbb{R}^{N \times m} \to \{0, \ldots, C-1\}^N$ be the clustering function, which takes the N encoded data points as input, and returns a cluster label for each. I refer to the output of ψ as a labelling. Let $\Gamma : \{0, \ldots, C-1\}^{K \times N} \to \{0, \ldots, C-1\}^N \times \{0, 1\}^N$ be the consensus function, which aggregates K different labellings of $\mathcal{X}$ into a single labelling, and also returns a Boolean vector indicating agreement. Then,

$$(c_1, \ldots, c_N), (a_1, \ldots, a_N) = \Gamma(\psi(f_1(\mathcal{X})) \circ \cdots \circ \psi(f_K(\mathcal{X}))), \tag{3.1}$$

where $(c_1, \ldots, c_N)$ are the consensus labels, and $a_i = 1$ if the i-th data point received the same cluster label (after alignment) in all labellings, and 0 otherwise. The consensus function is the ensemble mode average, c_i is the cluster label that was most commonly assigned to the i-th data point.

Define K pseudo-classifiers $(h_j)_{1 \leq j \leq K}, h_j : \mathbb{R}^m \to \mathbb{R}^C$, and let

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} \begin{cases} CE(h_j(f_j(x_i)), c_i) & a_i = 1 \\ ||g_j(f_j(x_i)) - x_i|| & \text{otherwise,} \end{cases} \tag{3.2}$$

where CE denotes categorical cross-entropy:

$$CE : \mathbb{R}^C \times \{0, \ldots, C-1\} \to \mathbb{R}$$
$$CE(x, n) = -\log(x[n])\,.$$

Note that, because the model only pseudo-label trains on agreed points, training each encoder on the consensus labels $c_1, \ldots, c_N$ is equivalent to training each encoder on its own labels.

Algorithm 1 Training algorithm for SPC

for $j = 1, \ldots, K$ **do**
  Update parameters of f_j and g_j using autoencoder reconstruction
end for
while number of agreed points increases **do**
  compute $(c_1, \ldots, c_N), (a_1, \ldots, a_N)$ as in (3.1)
  for $j = 1, \ldots, K$ **do**
    Update parameters of f_j and h_j to minimize (3.2)
  end for
end while

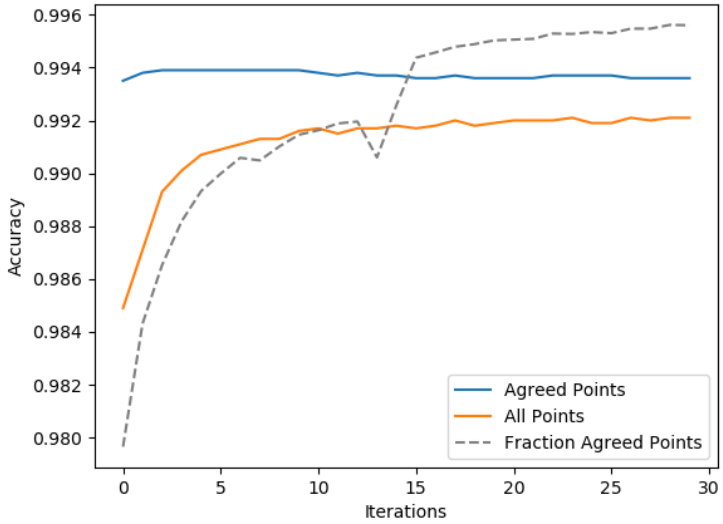

Figure 3.2: Iterations of (2)–(4) in Figure 3.1 on MNIST.

For agreed points, all ensemble labels, and hence the consensus labels, are the same up to reordering.
First, I pretrain the autoencoders, then compute $(c_1, \ldots, c_N), (a_1, \ldots, a_N)$ and minimize $\mathcal{L}$, recompute, and iterate until the number of agreed points stops increasing. The method is summarized in Algorithm 1.

Figure 3.2 shows the training dynamics. Agreed points are those that receive the same cluster label in all members of the ensemble. As expected, the agreed points' accuracy is higher than the accuracy on all points. Initially, the agreed points will not include those that are difficult to cluster correctly, such as an MNIST digit 3 that looks like a digit 5. Some ensemble members will cluster it with the 3's and some with the 5's. The training process aims to make these difficult points into agreed points, thus increasing the fraction of agreed points, without decreasing the agreed points' accuracy. Figure 3.2 shows that this aim is achieved. As more points become agreed (black dotted line), the total accuracy approaches the agreed accuracy. The agreed accuracy remains high, decreasing only very slightly (blue line). The result is that the total accuracy increases (orange line). Training continues until the number of agreed points plateaus.

3.3.2 Implementation Details

Encoders are stacks of convolutional and batch norm layers; decoders of transpose convolutional layers. Decoders have a $tanh$ activation on their output layer, all other layers use leaky ReLU with parameter 0.3. The MLP pseudo-classifier has a hidden layer of size 25. The latent space of the autoencoders has size 50 for MNIST and FashionMNIST, and 20 for smaller USPS. I inject noise from a multivariate normal into the latent space as a simple form of regularization. As suggested by (H. Zhao, Gallo, Frosio, and Kautz, 2015), the reconstruction loss is ℓ_1. The architectures are the same across the ensemble, diversity comes from random initialization and training dynamics.

The clustering function (ψ above) is a composition of UMAP (McInnes, Healy, and Melville, 2018) and either HDBSCAN (McInnes, Healy, and Astels, 2017) or a Gaussian mixture model (GMM), both HDBSCAN and GMM are reported below. As in previous works, the number of clusters is set to the ground truth. UMAP uses the parameters suggested in the clustering documentation, $n\_neighbours$ is 30 for MNIST and scaled in proportion to the dataset size for the others. HDBSCAN uses all default parameters. I cut the linkage tree at a level that gives the correct number of clusters. On the rare occasions when no such cut can be found, the clustering is excluded from the ensemble. The GMM uses all default parameters.

Consensus labels are taken as the most common across the ensemble, after alignment with the Hungarian algorithm (called the "direct" method in (Boongoen and Iam-On, 2018)).

3.4 Proof of Correctness

Proving correctness requires proving that the expected accuracy of the agreed pseudo-labels is higher than that of all pseudo-labels, and that training with more accurate pseudo-labels makes the latent vectors easier to cluster correctly.

3.4.1 Agreed Pseudo-Labels are More Accurate

Given that each member of the ensemble is initialized independently at random, and undergoes different stochastic training dynamics, we can assume that each cluster assignment contains some unique information. Formally, there is strictly positive conditional mutual information between any one assignment in the ensemble and the true cluster labels, conditioned on all the other assignments in the ensemble. From this assumption, the reasoning proceeds as follows.

Choose an arbitrary data point x_0 and cluster c_0. Let X be a random variable, indicating the true cluster of x_0, given that $n \geq 0$ members of the ensemble have assigned it to c_0, and other assignments are unknown. Thus, the event $X = c_0$ is the event that x_0 is correctly clustered. Let Y be a Boolean random variable indicating that the $(n+1)$-th member of the ensemble also assigns it to c_0. We want to show that the probability of x_0 being correctly clustered increases if $Y = c_0$, i.e. that an additional ensemble member assigning to c_0 improves the chances that the ensemble as a whole, which has so far unanimously assigned to c_0, is correct.

Assume that, if n ensemble members have assigned x_0 to c_0, and other assignments are unknown, then x_0 belongs to c_0 with probability at least $1/C$ and belongs to all other clusters with equal probability, i.e.,

$$\begin{aligned} p(X = c_0) &= t \\ \forall c \neq c_0, p(X = c) &= (1-t)/(C-1)\,, \end{aligned}$$

for some $1/C \leq t \leq 1$.

It can then be established that the entropy $H(X)$ is a strictly decreasing function of t, as follows.

Proposition 1. *Given a categorical random variable X of the form*

$$\begin{aligned} p(X = c_0) &= t \\ \forall c \neq c_0, p(X = c) &= \frac{1-t}{C-1}\,, \end{aligned}$$

for some $1/C \leq t \leq 1$, the entropy $H(X)$ is a strictly decreasing function of t.

Proof.

$$\begin{aligned} H(X) &= -\,t \log t - (1-t) \log \frac{1-t}{C-1} \\ \frac{d(H(X))}{dt} &= -\log t - 1 - \frac{1}{1-t} + \log \frac{1}{C-1} + \\ &\quad + \frac{t}{1-t} + \log 1 - t \\ &= -\,2 - \log t - \log C - 1 + \log t - 1 \\ &= -\,2 - \log \left(\frac{t}{1-t}(C-1) \right). \end{aligned}$$

The argument to the $\log$ is clearly an increasing function of t for $t \in (0,1)$. Therefore, for $1/C \leq t < 1$, it is lower-bounded by setting $t = 1/C$. This gives

$$\begin{aligned} \frac{d(H(X))}{dt} &\leq -2 - \log \left(\frac{1/C}{1-1/C}(C-1) \right) \\ &< -\log \left(\frac{1/C}{1-1/C}(C-1) \right) = -\log 1 = 0\,. \end{aligned}$$

The derivative is always strictly negative with respect to t, so, as a function of t, $H(X)$ is always strictly decreasing. □

This shows that the above assumption on conditional mutual information, written $I(X;Y) > 0$, is equivalent, in the case that $Y = c_0$, to an increase in t. More explicitly, $I(X;Y) > 0$ if and only if $H(X|Y) < H(X)$ if and only if (by Proposition 1) t increases on learning Y. Finally, observe that t increasing is equivalent to $p(X = c_0|Y) > p(X = c_0)$. This establishes that the accuracy of the agreed labels is an increasing function of ensemble size. Standard pseudo-label training uses $n = 1$, whereas SPC uses $n > 1$ and so results in more accurate pseudo-labels for training.

3.4.2 Increased Pseudo-Label Accuracy Improves Clustering

Although it is intuitive that training on more accurate pseudo-labels makes the clustering task easier, and although it has been implicitly assumed in previous works using pseudo-label training (Caron, Bojanowski, Joulin, and Douze, 2018; Mrabah, N. M. Khan, Ksantini, and Lachiri, 2019), a formal proof has not been provided. I prove the claim in a simplified case. This validates both the general strategy of pseudo-label training, and my strategy of training on only the most accurate pseudo-labels.

Let $\mathcal{D}$ be a dataset of i.i.d. points from a distribution over $\mathcal{S} \in \mathbb{R}^n$, where $\mathcal{S}$ contains C true clusters $c_1, \ldots, c_C$. Let T be the random variable defined by the identity function on S and $f : \mathcal{S} \to \mathbb{R}^m$, an encoding function parametrized by θ, whose output is a random variable X. The task is to recover the true clusters conditional on X, and we are interested in choosing θ such that this task is as easy as possible. Pseudo-label training applies a second function $h : \mathbb{R}^m \to \{0, \ldots, C-1\}$ and trains the composition $h \circ f : \mathbb{R}^n \to \{0, \ldots, C-1\}$ using gradient descent, with cluster assignments as pseudo-labels. The claim is that an increased pseudo-label accuracy facilitates a better choice of θ.

To formalize "easy", recall the definition of clustering as a partition that minimizes intra-cluster variance and maximizes inter-cluster variance. We want the same property to hold of the random variable X. Let $y : \mathcal{D} \to \{0, \ldots, C-1\}$ be the true cluster assignment function and Y the corresponding random variable, then ease of recovering the true clusters is captured by a high value of d, where

$$d = \mathrm{Var}(\mathbb{E}[X|Y]) - \mathbb{E}[\mathrm{Var}(X|Y)]\,.$$

High d means that a large fraction of the variance of X is accounted for by cluster assignment, as, by Eve's law, we can decompose:

$$\mathrm{Var}\,(X) = \mathbb{E}[\mathrm{Var}(X|Y)] + \mathrm{Var}(\mathbb{E}[X|Y])\,, \tag{3.3}$$

In the following, we assume that f and g are linear, $C = 2$, $h \circ f(\mathcal{D}) \subseteq (0, 1)$, and $\mathbb{E}[T] = \vec{0}$. The proof proceeds by expressing the value of d in terms of expected distances between encoded points after a training step with correct labels and with incorrect labels, and hence proving that the value is greater in the former case. I show that the expectation is greater in each coordinate, from which the claim follows by linearity of expectation.

Lemma 1. *Let $x, x' \in \mathcal{D}$ be two data points, and consider the expected squared distance between their encodings under f. Let u_{same} and u_{diff} denote the value of this difference after a gradient descent update in which both labels are the same and after a step in which both labels are different, respectively. Then, $u_{same} < u_{diff}$.*

Proof. If $w \in \mathbb{R}^m$ and $w' \in \mathbb{R}$ are, respectively, the vector of weights mapping the input to the i-th coordinate of the latent space, and the scalar mapping the i-th coordinate of the latent space to the output, then the expected squared distance in the i-th coordinate of the latent vectors before the gradient descent update is

$$\mathop{\mathbb{E}}_{x,x' \sim T}[(w^T x - w^T x')^2] = \mathop{\mathbb{E}}_{x,x' \sim T}[(w^T (x - x'))^2]\,.$$

When the two labels are the same, assume w.l.o.g. that $y = y' = 0$. Then, with step size η , the update for w and the following expected squared difference u_{same} is

$$\begin{aligned} w &\leftarrow w - \eta(w'(x + x')) \\ u_{same} &= \mathop{\mathbb{E}}_{x,x' \sim T}[((w - \eta w'(x + x'))^T (x - x'))^2] \\ &= \mathop{\mathbb{E}}_{x,x' \sim T}[(w^T(x - x') - \eta w'(||x||^2 - ||x'||^2))^2] \,. \end{aligned}$$

When the two labels are different, assume w.l.o.g. that $y = 0$, $y' = 1$, giving

$$\begin{aligned} w &\leftarrow w - \eta(w'(x - x')) \\ u_{diff} &= \mathop{\mathbb{E}}_{x,x' \sim T}[((w - \eta(w'(x - x'))^T)(x - x')])^2] = \\ &\mathop{\mathbb{E}}_{x,x' \sim T}[(w^T(x - x') - \eta w' ||x - x'||^2)^2] \,. \end{aligned}$$

Decomposing u_{same} according to the definition of variance (as the expectation of the square minus the square of the expectation) gives

$$\begin{aligned} &\mathop{\mathbb{E}}_{x,x' \sim T}[w^T(x - x') - \eta w'(||x||^2 - ||x'||^2)]^2 + \\ &\quad \mathrm{Var}(w^T(x - x') - \eta w'(||x||^2 - ||x'||^2)) \,. \end{aligned}$$

The expectation term equals 0, as

$$\begin{aligned} &w^T \mathop{\mathbb{E}}_{x,x' \sim T}[(x - x')] - \eta w' \mathop{\mathbb{E}}_{x,x' \sim T}[(||x||^2 - ||x'||^2)] = \\ &w^T(\mathbb{E}[T] - \mathbb{E}[T]) - \eta w'(\mathbb{E}[||T||^2] - \mathbb{E}[||T||^2]) = 0 \,. \end{aligned}$$

By symmetry, we can replace covariances involving x' with the same involving x. The remaining term can then be rearranged to give

$$\begin{aligned} u_{same} &= 2 \,\mathrm{Var}(w^T x - \eta w' ||x||^2) \\ &= 2 w^T Cov(T) w + 2\eta w' Var(||x||^2) - 4 \,\mathrm{Cov}(w^T x, \eta w' ||x||^2) \,. \end{aligned}$$

Now rewrite u_{diff}. Decomposing as above gives

$$\begin{aligned} &\mathop{\mathbb{E}}_{x,x' \sim T}[w^T(x - x') - \eta w'(||x - x'||^2)]^2 + \\ &\quad \mathrm{Var}(w^T(x - x') - \eta w'(||x - x'||^2)) \,, \end{aligned}$$

and here the expectation term does not equal 0:

$$\begin{aligned} &(w^T \mathop{\mathbb{E}}_{x,x' \sim T}[(x - x')] - \eta w' \mathop{\mathbb{E}}_{x,x' \sim T}[(||x - x'||^2)])^2 = \\ &\quad (\eta w')^2 \mathop{\mathbb{E}}_{x,x' \sim T}[||x - x'||^2]^2 \,. \end{aligned}$$

The variance term can be expanded to give:

$$\begin{gathered}\operatorname{Var}(w^T(x-x') - \eta w'(||x-x'||^2)) = \\ 2w^T \operatorname{Cov}(T)w + 2\eta w' \operatorname{Var}(||x-x'||^2) - \\ 4 \operatorname{Cov}(w^T x, \eta w' ||x-x'||^2)\,.\end{gathered}$$

By comparing terms, we can see that this expression is at least as large as u_{same}. First, consider the covariance terms.

Claim. $\operatorname{Cov}(w^T x, \eta w' ||x-x'||^2) = \operatorname{Cov}(w^T x, \eta w' ||x||^2)$.

$$\begin{aligned}
&\operatorname{Cov}(w^T x, \eta w' ||x-x'||^2) \\
&= \mathbb{E}[w^T x \eta w' ||x-x'||^2] - \mathbb{E}[w^T x]\mathbb{E}[\eta w' ||x-x'||^2] \\
&= \eta w' \mathbb{E}[w^T x ||x-x'||^2] - 0\mathbb{E}[\eta w' ||x-x'||^2] \\
&= \eta w' \mathbb{E}[w^T x ||x-x'||^2] \\
&= \eta w' \mathbb{E}[w^T x \sum_k x^2 - 2xx' + x'^2] \\
&= \eta w' \sum_k \mathbb{E}[w^T x x_k^2] - 2\mathbb{E}[w^T x x_k]\mathbb{E}[x'] + \mathbb{E}[w^T x]\mathbb{E}[x'^2] \\
&= \eta w' \sum_k \mathbb{E}[w^T x x_k^2] - 2\mathbb{E}[w^T x x_k]\vec{0} + 0\mathbb{E}[x'^2] \\
&= \eta w' \sum_k \mathbb{E}[w^T x x_k^2] \\
&= \eta w' \mathbb{E}[w^T x \sum_k x_k^2] \\
&= \eta w' \mathbb{E}[w^T x ||x||^2] \\
&= \eta w' \mathbb{E}[w^T x ||x||^2] - 0\mathbb{E}[\eta w' ||x||^2] \\
&= \mathbb{E}[w^T x \eta w' ||x||^2] - \mathbb{E}[w^T x]\mathbb{E}[\eta w' ||x||^2] \\
&= \operatorname{Cov}(w^T x, \eta w' ||x||^2)\,.
\end{aligned}$$

So, we see the covariance terms are equal.

Next, compare the second variance terms

Claim. $\operatorname{Var}(||x-x'||^2) \geq \operatorname{Var}(||x||^2)$.

$$
\begin{aligned}
&\operatorname{Var}(||x - x'||^2) \\
&= \operatorname{Var}\left(\sum_{k=0}^{nz}(x)_k^2 + (x')_k^2 - 2(x)_k(x')_k\right) \\
&= \operatorname{Var}\left(\sum_{k=0}^{nz}(x)_k^2\right) + \operatorname{Var}\left(\sum_{k=0}^{nz}(x')_k^2\right) + 2\operatorname{Var}\left(\sum_{k=0}^{nz} x_k x'_k\right) \\
&= 2\operatorname{Var}\left(\sum_{k=0}^{nz}(x)_k^2\right) + 2\operatorname{Var}\left(\sum_{k=0}^{nz} x_k x'_k\right) \\
&= 2(\operatorname{Var}(||x||^2) + \operatorname{Var}(x^T x')) \\
&\geq \operatorname{Var}(||x||^2)\,.
\end{aligned}
$$

Assuming that the data are not all identical, this implies that u_{diff} is strictly greater than u_{same}.

$$
\begin{aligned}
&u_{diff} - u_{same} \\
&= (\eta w')^2 \mathop{\mathbb{E}}_{x,x'\sim T}[||x - x'||^2]^2 + 2w^T \operatorname{Cov}(T)w + \\
&\quad + 2\eta w' \operatorname{Var}(||x - x'||^2) - 4\operatorname{Cov}(w^T x, \eta w'||x - x'||^2) - \\
&\quad - ((2w^T \operatorname{Cov}(T)w + 2\eta w' Var(||x||^2) \\
&\quad - 4\operatorname{Cov}(w^T x, \eta w'||x||^2))) \\
&= (\eta w')^2 \mathop{\mathbb{E}}_{x,x'\sim T}[||x - x'||^2]^2 + \\
&\quad + 2\eta w' \left(\operatorname{Var}(||x - x'||^2) - \operatorname{Var}(||x||^2)\right) - \\
&\quad - 4\left(\operatorname{Cov}(w^T x, \eta w'||x - x'||^2) - \operatorname{Cov}(w^T x, \eta w'||x||^2)\right) \\
&= (\eta w')^2 \mathop{\mathbb{E}}_{x,x'\sim T}[||x - x'||^2]^2 + \\
&\quad + 2\eta w' \left(\operatorname{Var}(||x - x'||^2) - \operatorname{Var}(||x||^2)\right) \\
&\geq (\eta w')^2 \mathop{\mathbb{E}}_{x,x'\sim T}[||x - x'||^2]^2 > 0\,.
\end{aligned}
$$

□

Lemma 2. *Let z be a third data point, $z \in \mathcal{D}, z \neq x, x'$, and consider the expected squared distance of the encodings, under f, of x and z. Let v_{same} and v_{diff} denote, respectively, the value of this difference after a gradient descent update with two of the same labels, and with two different labels. Then, $v_{same} = v_{diff}$.*

Proof. $v_{diff} - v_{same}$

$$
\begin{aligned}
&= \mathbb{E}[(w^T(x-z) - w'(x-x')(x-z))^2] - \\
&\quad - \mathbb{E}[(w^T(x-z) - w'(x+x')(x-z))^2] \\
&= \mathbb{E}[(w^T(x-z) - w'(x-x')^T(x-z))^2 - \\
&\quad - (w^T(x-z) - w'(x+x')^T(x-z))^2] \\
&= \mathbb{E}[(w^T(x-z) - w'(x-x')^T(x-z) + \\
&\quad + w^T(x-z) - w'(x+x')^T(x-z)) \\
&\quad (w^T(x-z) - w'(x-x')^T(x-z) - \\
&\quad - w^T(x-z) - w'(x+x')^T(x-z))] \\
&= \mathbb{E}[(2w^T(x-z) - w'(x-z)^T(x-x'+x+x')) \\
&\quad (-w'(x-z)^T(x-x'-x-x'))] \\
&= \mathbb{E}[(2w^T(x-z) - 2w'(x-z)^T(x))(2w'(x-z)^T(x'))] \\
&= 2\mathbb{E}[(w^T(x-z) - w'(x-z)^T(x))w'(x-z)^T]\mathbb{E}[x'] \\
&= 2\mathbb{E}[(w^T(x-z) - w'(x-z)(x))w'(x-z)^T]\vec{0} = 0\,.
\end{aligned}
$$

□

Lemma 3. *Let s and r denote, respectively, the expected squared distance between the encodings, under f, of two points in the same cluster and between two points in different clusters. Then, there exist $\lambda_1, \lambda_2 > 0$ whose values do not depend on the parameters of f, such that $d = \lambda_1 r - \lambda_2 s$.*

Proof. Let C denote the number of clusters. For simplicity, assume that the clusters are equally sized. The argument can easily be generalized to clusters of arbitrary sizes. Then

$$
\begin{aligned}
\mathrm{Var}(X) =& \tfrac{1}{2} \mathop{\mathbb{E}}_{x,x' \sim T}[w^T(x-x')^2] \\
=& \tfrac{1}{2}(\mathop{\mathbb{E}}_{x,x' \sim T}[w^T(x-x')^2 | y(x) = y(x')] P(y(x) = y(x')) + \\
& \mathop{\mathbb{E}}_{x,x' \sim T}[w^T(x-x')^2 | y(x) \neq y(x')] P(y(x) \neq y(x')) \\
=& \tfrac{1}{2}(sP(y(x) = y(x')) + rP(y(x) \neq y(x')) \\
=& \frac{1}{2}\left(s\frac{1}{C} + r\frac{C-1}{C}\right)\,.
\end{aligned}
$$

Noting that $s = 2\mathbb{E}[\mathrm{Var}(X|Y)]$, and using Eve's law, we have

$$
\begin{aligned}
d =& \mathrm{Var}(X) - s \\
=& \frac{1}{2}\left(s\frac{1}{C} + r\frac{C-1}{C}\right) - s \\
=& \frac{C-1}{2C}r - \frac{2C-1}{2C}s\,.
\end{aligned}
$$

□

Definition 4. Let $\tilde{y} : \mathcal{D} \to \{0, \ldots, C-1\}$ be the pseudo-label assignment function. For $d_i, d_j \in \mathcal{D}$, the pseudo-labels are *pairwise correct* iff $y(x_i) = y(x_j)$ and $\tilde{y}(x_i) = \tilde{y}(x_j)$, or $y(x_i) \neq y(x_j)$ and $\tilde{y}(x_i) \neq \tilde{y}(x_j)$.

Note that more pairwise correct labels corresponds to a more accurate clustering, as measured with the Rand Index, as described in Section 2.2.

Theorem 5. *Let d_T and d_F denote, respectively, the value of d after a gradient descent step from two pairwise correct labels and from two pairwise incorrect labels, and let $x, x' \in \mathcal{D}$ as before. Then, $d_T > d_F$.*

Proof. Let r_T, s_T, and r_F, s_F be, respectively, the values of r and s after a gradient descent step from two pairwise correct labels and from two pairwise incorrect labels. Consider the two cases $y(x) = y(x')$ and $y(x) \neq y(x')$.

If $y(x) = y(x')$, then Lemma 2 means that the expected distance of the encodings of x and x' to any data point from another cluster is unchanged by whether the update was from points with the same or with different labels. Similarly, the distance between any two other points is unchanged by whether the update was from points with the same or with different labels. This establishes that $r_T = r_F$. As for the intra-cluster variance, it is smaller after the update with the same labels than with different labels. Lemma 1 shows that the expected distance between the encodings of the two points themselves is smaller if the labels were the same, and the same argument as above shows that all other expected distances within clusters are unchanged. This means that $s_T < s_F$, and so, by Lemma 3, $d_T > d_F$.

If $y(x) \neq y(x')$, then Lemma 2 means that the expected distance of the encodings of x and any data point from the same cluster is unchanged by whether the update was from points with the same or with different labels (and the same for x'). Similarly, the distance between any two other points is unchanged by whether the update was from points with the same or with different labels. This establishes that $s_T = s_F$. As for the *inter*-cluster variance, it is larger after the update with the same labels than with different labels. Lemma 1 shows that the expected distance between the encodings of the two points themselves is larger if the labels were different, and the same argument as above shows that all other expected distances within clusters are unchanged. This shows that $r_T > r_F$ and so, again by Lemma 3 $d_T > d_F$. □

The fraction of pairwise correct pairs is one measure of accuracy (Rand Index). Thus, training with more accurate pseudo-labels facilitates better clustering.

3.5 Experimental Results

Following previous works, I measure accuracy and NMI, as described in Section 2.2. I report on two handwritten digits datasets, MNIST (size 70000) (LeCun, Bottou, Bengio, and Haffner, 1998) and USPS (size 9298) (Hull, 1994), and FashionMNIST (size 70000) (Xiao, Rasul, and Vollgraf, 2017) of clothing items. The existing methods I compare to are all described in Section 3.2. Table 3.1 gives an overview of these methods, and compares them with respect to their loss functions, use of data augmentation and dimensionality reduction.

Hyperparameter Choice The main relevant hyper-parameters in the presented method are the ensemble size, the number of clusters, and the parameters of the UMAP dimension reduction algorithm. The effect of different choices of ensemble size is explored experimentally in Section 3.5.2. The number of clusters is a required parameter for one of the two clustering algorithms we report results on, the GMM, but not the other, HDBSCAN. I observed empirically that HDBSCAN was consistently able to find the correct number of clusters for MNIST and USPS, but was less consistent for FashionMNIST. In the main results reported in Table 3.2, I fix the HDBSCAN number of clusters, and the GMM number of clusters, to the number of ground truth classes. This is also the approach in the methods of Chapters 4 and 5. Automatically determining the correct number of clusters is generally a difficult problem, and all the existing works from Table 3.2 also set it manually to the number of ground truth classes. The choice of UMAP parameters follows the recommendation in the UMAP documentation, to use minimum distnace=0 and use a large value for n_neighbours (here = 60). As reported in the documentation, a small value of minimum distance increases data density, and a high value of n_neighbours captures more global structure of the data, both of which are useful for clustering. I do not explore tuning these parameters because I want to demonstrate the strong performance of the proposed method on the recommended defaults.

model name	main loss	auxiliary loss	data-augmentation	dim. reduction
DynAE	reconstruction	locality-preserving	yes	none
ADC	reconstruction	locality-preserving	yes	none
DDC	Cauchy-Schwarz divergence of cluster distributions	collapse-prevention	yes	none
n2d	reconstruction	none	no	UMAP
DLS	ClusterGAN loss	reconstruction	no	none
DEPICT	reconstruction	collapse-prevention	no	none
DMSC	reconstruction	self-expressive loss	yes	none
ClusterGAN	generative adversarial	reverse reconstruction	no	none
VADE	evidence lower bound	none	no	none
IDEC	reconstruction	KL to normalized t-distribution	no	none
DEC	reconstruction	KL to normalized t-distribution	no	none
CKM	reconstruction	k-means	no	none
DCN	reconstruction	k-means	no	none

† means each data point can be expressed as a sparse linear combination of other data points

Table 3.1: Taxonomy of existing deep clustering methods.

3.5.1 Main Results

Table 3.2 shows the central tendency for five runs and the best single run. Results are shown for two different clustering algorithms: a Gaussian mixture model and the more advanced HDBSCAN (McInnes, Healy, and Astels, 2017). Both perform similarly, showing

	MNIST		USPS		FashionMNIST	
	ACC	NMI	ACC	NMI	ACC	NMI
SPC-best	**99.21**	**97.49**	**98.44**	**95.44**	*67.94*	**73.48**
SPC	**99.03 (.1)**	**97.04 (.25)**	**98.40 (.94)**	**95.42 (.15)**	65.58 (2.09)	**72.09 (1.28)**
SPC-GMM	**99.05 (.2)**	**97.10 (.47)**	**98.18 (.14)**	**94.93 (.32)**	65.03 (1.54)	**69.51 (1.21)**
DynAE (Mrabah, N. M. Khan, Ksantini, and Lachiri, 2019)†	*98.7*	*96.4*	*98.1*	94.8	59.1	64.2
ADC (Mrabah, Bouguessa, and Ksantini, 2019)†	98.6	96.1	*98.1*	94.8	58.6	66.2
DDC (Y. Ren, N. Wang, M. Li, and Z. Xu, 2020)†	98.5	96.1	97.0	*95.3*	57.0	63.2
n2d (McConville, Santos-Rodriguez, Piechocki, and Craddock, 2019)	97.9	94.2	95.8	90.0	67.2	*68.4*
DLS (F. Ding and Luo, 2019)	97.5	93.6	-	-	**69.3**	66.9
JULE (J. Yang, Parikh, and Batra, 2016)	96.4	91.3	95.0	91.3	56.3*	60.8*
DEPICT (Ghasedi Dizaji, Herandi, C. Deng, W. Cai, and H. Huang, 2017)	96.5	91.7	96.4	92.7	39.2*	39.2*
DMSC (Abavisani and Patel, 2018)†	95.15	92.09	95.15	92.09	-	-
ClusterGAN (Mukherjee, Asnani, E. Lin, and Kannan, 2019)	95	89	-	-	63	64
VADE (Z. Jiang, Zheng, H. Tan, Tang, and H. Zhou, 2016)	94.5	87.6*	56.6*	51.2*	57.8*	63.0*
IDEC (X. Guo, L. Gao, X. Liu, and J. Yin, 2017)	88.06	86.72	76.05	78.46	52.9*	55.7*
CKM (B. Gao, Y. Yang, Gouk, and Hospedales, 2020)	85.4	81.4	72.1	70.7	-	-
DEC (J. Xie, R. Girshick, and Farhadi, 2016)	84.3	83.4*	76.2*	76.7*	51.8*	54.6*
DCN (B. Yang, Fu, Sidiropoulos, and Hong, 2017)	83	81	68.8*	68.3*	50.1*	55.8*

† =uses data augmentation *=results taken from (Mrabah, N. M. Khan, Ksantini, and Lachiri, 2019)

Table 3.2: Accuracy and NMI of SPC compared to other top-performing image clustering models. The best results are in bold, and the second-best are emphasized. I report the mean and standard deviation (in parentheses) for five runs.

robustness to clustering algorithm choice. SPC-GMM performs slightly worse on USPS and FashionMNIST (though within the margin of error), suggesting that HDBSCAN may cope better with the more complex images in FashionMNIST and the smaller dataset in USPS. In Table 3.2, 'SPC' uses HDBSCAN.

SPC (using either clustering algorithm) outperforms all existing approaches for both metrics on MNIST and USPS, and for NMI on FashionMNIST. The disparity between the two metrics, and between HDBSCAN and GMM, on FashionMNIST is due to the variance in cluster size. Many of the errors are lumped into one large cluster, and this hurts accuracy more than NMI, because being in this large cluster still conveys some information about what the ground truth cluster label is. See Section 3.5.4 for more details.

The most accurate existing methods use data augmentation. This is to be expected, given the well-established success of data augmentation in supervised learning (G. E. Hinton, Srivastava, Krizhevsky, Sutskever, and Salakhutdinov, 2012). More specifically, (X. Guo, E. Zhu, X. Liu, and J. Yin, 2018) have shown empirically that adding data augmentation to deep image clustering models improves performance in virtually all cases. Here, its effect is especially evident on the smaller dataset, USPS. For example, on MNIST, n2d (McConville, Santos-Rodriguez, Piechocki, and Craddock, 2019) (which does not use data augmentation) is only 0.6 and 1.9 behind DDC (Y. Ren, N. Wang, M. Li, and Z. Xu, 2020), which does on ACC and NMI, respectively, but is 1.2 and 5.3 behind on USPS. SPC could easily be extended to include data augmentation, and even without using it, outperforms models that do.

3.5.2 Ablation Studies

Table 3.3 shows the effect of removing each component of my model. All settings use HDBSCAN. Particularly relevant are rows 2 and 3. As described in Section 3.3, I produce

	MNIST		USPS		FashionMNIST	
	ACC	NMI	ACC	NMI	ACC	NMI
SPC	**99.03 (.1)**	**97.04 (.25)**	**98.40 (.94)**	**95.42 (.15)**	**65.58 (2.09)**	**72.09 (1.28)**
A1	98.01 (.04)	94.46 (.11)	97.03 (.65)	92.43 (1.29)	63.12 (.16)	70.59 (.01)
A2	98.18 (.05)	94.86 (.09)	97.31 (.89)	92.99 (1.84)	60.60 (4.45)	68.77 (.48)
A3	98.02 (.19)	94.45 (.43)	95.85 (.80)	89.77 (1.65)	59.23 (3.58)	67.09 (3.77)
A4	97.88 (.72)	94.8 (.85)	87.49 (7.93)	82.68 (2.6)	61.2 (4.28)	67.28 (1.72)
A5	96.17 (.26)	91.07 (.23)	87.00 (8.88)	80.79 (7.43)	55.29 (3.54)	66.07 (1.04)
A6	70.24	77.42	70.46	71.11	42.08	49.22

Table 3.3: Ablation results, central tendency for three runs. A1=w/o label filtering; A2=w/o label sharing; A3=w/o ensemble; A4=pseudo-label training only; A5=UMAP+AE; A6=UMAP. Both A1 and A2 train on all data points. The former trains each member of the ensemble on their own labels, and the latter uses the consensus labels. A3 sets ensemble size to 1.

	MNIST		USPS		FashionMNIST	
	ACC	NMI	ACC	NMI	ACC	NMI
25	98.48	95.60	97.70	93.82	67.67	73.25
20	98.49	95.64	97.87	94.21	67.52	73.13
15	99.03 (.10)	97.04 (.25)	98.40 (.94)	95.42 (.15)	65.58 (2.09)	72.09 (1.28)
12	98.82	96.54	98.20	95.02	67.77	73.13
10	98.78	96.42	98.39	95.47	62.93	69.89
8	98.75	96.32	98.41	95.44	67.45	71.99
6	98.61	95.90	98.40	95.39	63.84	70.62
5	98.56	95.82	98.30	95.19	67.91	73.46
4	98.47	95.60	98.27	95.18	67.90	73.38
3	98.44	95.50	98.15	94.84	63.36	70.88
2	98.27	95.07	97.98	94.40	62.9	70.41
1	98.02 (.19)	94.45 (.43)	95.85 (.80)	89.77 (1.65)	59.23 (3.58)	67.09 (3.77)

Table 3.4: Ablation studies on the size of the ensemble.

multiple labellings of the dataset and select for pseudo-label training only those data points that received the same label in all labellings. I perform two different ablations on this method: A1 and A2. Both use all data points for training, but A1 trains each ensemble on all data points using the labels computed in that ensemble member, and A2 uses the consensus labels. At inference, both use consensus labels. The significant drop in accuracy in both settings demonstrates that the strong performance of SPC is not just due to the application of an ensemble to existing methods, but rather to the novel method of label selection.

It is interesting to observe that A1 performs worse than A2 on MNIST and USPS. Combining approximations in an ensemble has long been observed to give higher expected accuracy (Clemen, 1989; Pearlmutter and Rosenfeld, 1991; Perrone, 1993; Breiman, 1996), so the training targets would be more accurate in A1 than in A2. I hypothesize that the reason this fails to translate to improved clustering is a reduction in ensemble variance. On MNIST and USPS, high accuracy across the ensemble means high agreement. Giving the

same training signal for every data point reduces variance further. Especially, compared with A2, the reduction is greatest on incorrectly clustered data points, because most incorrectly clustered data points are non-agreed points, and as argued in (Kittler, Hatef, Duin, and Matas, 1998), high ensemble variance in the errors is important for performance.

A4 clusters in the latent space of one untrained encoder and then pseudo-label trains (essentially the method in (Caron, Bojanowski, Joulin, and Douze, 2018)). It performs significantly worse than SPC, showing the value of the decoder, and of SPC's label selection technique.

A3 omits the ensemble entirely. Comparing with A2 again shows that the ensemble itself only produces a small improvement. Alongside SPC's label selection method, the improvement is much greater.

3.5.3 Ensemble Size

The number of autoencoders in the ensemble, K in the terminology of Section 3.3.1, is a hyperparameter. I add the concatenation of all latent spaces as an additional element. Table 3.4 shows the performance for smaller ensemble sizes. In MNIST and USPS, where the variance is reasonably small, there is a discernible trend of the performance increasing with K, then plateauing and starting to decrease. For FashionMNIST, where the variance is higher, the pattern is less clear. For all three datasets, however, we can see a significant difference between an ensemble of size two and an ensemble of size one (i.e., no ensemble). I hypothesize that the decrease for $K = 20, 25$ is due to a decrease in the number of agreed points, and so fewer pseudo-labels to train the encoders.

3.5.4 Cluster Sizes

The results in 3.2 report the central tendency of five different training runs for each dataset. Tables 3.5, 3.6, and 3.7 show the sizes of the clusters predicted by SPC for one randomly selected run out of these five. On MNIST and USPS, where the accuracy of SPC is $> 98\%$, the predicted sizes are close to the true sizes. On FashionMNIST, where the accuracy is $\sim 65\%$, there is a much greater variance. This accounts for the discrepancy in ACC and NMI for FashionMNIST. Most of the errors are put into one large cluster, specifically the cluster that was aligned to 'coat' is over three times larger than it should be. This hurts accuracy more than NMI, because the incorrect data points in the 'coat' cluster count for zero when calculating the accuracy, but they are not randomly distributed among the other classes, so the conditional entropy of a data point that was mis-clustered as a coat is $< \log(10)$. Actually, most of the mistakes in the 'coat' cluster are pullovers or shirts, and almost none of them are, for example, boots or tops, meaning the entropy is quite low. Comparing the cluster sizes for SPC-HDBSCAN and SPC-GMM also accounts for the differences across ACC and NMI between these two settings on FashionMNIST: SPC-GMM produces more uniformly-sized clusters, so the difference between ACC and NMI is smaller.

	Zero	One	Two	Three	Four	Five	Six	Seven	Eight	Nine
HDBSCAN	6923	7878	6979	7095	6802	6290	6911	7384	6776	6962
GMM	6942	6958	6791	7885	6976	7096	7350	6294	6906	6802
Ground Truth	7000	7000	7000	7000	7000	7000	7000	7000	7000	7000

Table 3.5: Sizes of predicted clusters for MNIST.

	Zero	One	Two	Three	Four	Five	Six	Seven	Eight	Nine
HDBSCAN	1565	1272	933	819	856	706	833	787	693	834
GMM	1271	834	785	833	690	835	862	930	699	1559
Ground Truth	1553	1269	929	824	852	716	834	792	708	821

Table 3.6: Sizes of predicted clusters for USPS.

	Top	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Boot
HDBSCAN	7411	6755	56	6591	21333	6046	3173	5666	3711	9258
GMM	6700	3111	16379	6807	6753	9127	7389	4482	8814	438
Ground Truth	7000	7000	7000	7000	7000	7000	7000	7000	7000	7000

Table 3.7: Sizes of predicted clusters for FashionMNIST.

3.6 Summary

This chapter has presented a deep clustering model, called selective pseudo-label clustering (SPC). SPC employs pseudo-label training, which alternates between clustering features extracted by a DNN, and treating these clusters as labels to train the DNN. I have improved this framework with a novel technique for preventing the DNN from learning noise. The method is formally sound and achieves state-of-the-art performance on three popular image clustering datasets. Ablation studies have demonstrated that the high accuracy is not merely the result of applying an ensemble to existing techniques, but rather is due to SPC's novel filtering method.

SPC is a direct combination, in a single model, of the two families of techniques explored in this thesis: deep learning and clustering. SPC takes as input a set of continuous, high-dimensional data points in the form of images, and assigns each one a cluster label. This can be conceived of as forming a simple discrete representation, in the form of the cluster label, of the continuous input data. Chapters 4 and 5 will further develop the potential of deep learning and clustering, combined in a single model, to produce simple discrete representations of continuous data.

Chapter 4

Human Activity Recognition Clustering

Abstract

This chapter describes a deep clustering model applied to the task of human activity recognition (HAR). There has been much recent research on HAR due to the proliferation of wearable sensors in watches and phones, and the advances of deep learning methods, which avoid the need to manually extract features from raw sensor signals. A significant disadvantage of deep learning applied to HAR is the need for manually labelled training data, which is especially difficult to obtain for HAR datasets. Progress is starting to be made in the unsupervised setting, in the form of deep HAR clustering models, which can assign labels to data without having been given any labels to train on, but there are problems with evaluating deep HAR clustering models, which makes assessing the field and devising new methods difficult. In this chapter, I highlight problems with how deep HAR clustering models are evaluated, describing these problems in detail and conducting careful experiments to explicate the effect that they can have on results. I then discuss solutions to these problems, and suggest standard evaluation settings for future deep HAR clustering models. Additionally, I present a new deep clustering model for HAR. When tested under my proposed settings, my model performs better than (or on par with) existing models, while also being more efficient and better able to scale to more complex datasets by avoiding the need for an autoencoder. Code for the evaluation method and the clustering method is available at https://github.com/Lou1sM/\ac{HAR}.

4.1 Introduction

Human activity recognition (HAR), the task of automatically determining the activity that a person is performing based on recorded data, has a number of important applications. It is of interest to healthcare research, as it can provide a direct measure of exercise frequency and intensity. The World Health Organization lists inactivity as the fourth leading risk factor for mortality, and estimates that over 30% of adults are insufficiently active.[1] However, such estimations are difficult. Self-report does not give a reliable measure of exercise, as patients tend to significantly over-report (McConnell, Turakhia, Harrington, King, and Ashley, 2018), so being able to directly monitor human activity is desirable. HAR is also used in wearable sports technology. Sports watches, for example, provide users with a breakdown of how much time they spend sitting, standing, and walking. Globally, the wearable technology market was valued at $41bn in 2019, and it is forecast to grow to $114bn by 2028.[2]

The recorded data on which human activity recognition is based can come from three different types of device: video recorders, ambient sensors, and wearable sensors. (See (T. Singh and Vishwakarma, 2019), (Demrozi, Pravadelli, Bihorac, and Rashidi, 2020), and (Y. Wang, Cang, and H. Yu, 2019), respectively, for surveys of video, ambient sensor, and wearable sensor HAR.) This data is then input to a recognition model, to infer the activity being performed. If video recorders are used, then the task is one of computer vision, if ambient or wearable sensors are used, then the task is a form of signal processing. The difference between ambient and wearable sensors is that the former stays at a fixed location in the environment, and the latter is attached to the human performing the activity. In this chapter, I focus primarily on HAR from wearable sensors. There are two types of wearable sensors, accelerometers, which measure acceleration in three spatial dimensions, and gyroscopes, which measure orientation and angular momentum.

Raw sensor data cannot always be easily interpreted by human inspection, which has two important consequences. Firstly, it can make feature engineering difficult. For example, in the case of gyroscope readings, we do not know, a priori, what the relevant differences are between the signals for certain activities, especially those that are similar, such as walking upstairs vs. downstairs. While engineered features have been used with some success (see Section 4.2), these are mostly statistical features, rather than features that leverage domain knowledge. Deep learning, which can learn to automatically extract features, is therefore an attractive approach to HAR. The second important consequence is that HAR data is very difficult to label. Labelled data is always more expensive and time-consuming to obtain than unlabelled data, but this is especially the case for sensor-based HAR data, because humans cannot provide annotations just by looking at the sensor readings. Instead, annotators must directly observe a subject, which requires taking them into the lab, or be given a video of the performed actions, which requires subjects to remember to film themselves while using the sensors outside the lab. There is therefore a need for models that can operate without labelled data, as has been noted in two recent survey papers (J. Wang, Y. Chen, Hao, X. Peng, and L. Hu, 2019; K. Chen, D. Zhang, L. Yao, B. Guo, Z. Yu, and Y. Liu, 2021). This is one reason HAR clustering is of value. If it was solved very accurately, so that all instances

[1] https://www.who.int/data/gho/indicator-metadata-registry/imr-details/3416

[2] https://www.grandviewresearch.com/industry-analysis/wearable-technology-market

of the same activity were clustered together and all instances of different activities were clustered separately, then, the HAR classification problem would also have been solved, and its solution would not have required any labelled data at all. Once the cluster labels had been aligned with activity names such as 'walking', we could simply equate clusters with classes. Another advantage of HAR clustering is that, even in the absence of a very accurate solution, it can shed light on the most appropriate set of classes into which activities should be partitioned. What the most appropriate set is, is not always obvious. For example, some datasets distinguish between 'walking' and 'fast walking', while some others just use a single class 'walking'; similarly for 'running' and 'jogging'. If clustering shows there to be a significant difference between walking and fast walking, this is evidence that such a distinction is warranted. This use is not explored further here, though it has been in previous works (Mejia-Ricart, Helling, and Olmsted, 2017).

For these reasons, HAR clustering has received significant research attention. This chapter focuses on deep HAR clustering, i.e., clustering HAR data using a deep neural network for feature extraction. Recent years have seen some works applying deep learning to HAR clustering (McConville, Santos-Rodriguez, Piechocki, and Craddock, 2021; Sheng and Huber, 2020; H. Ma, Z. Zhang, W. Li, and S. Lu, 2021). However, progress has been obstructed in deep HAR clustering, and HAR clustering more generally, by a lack of agreed evaluation standards. Different works test on different datasets, many of them private, and some crucial details are left out when describing the exact evaluation settings. In particular, the distinction between subject-dependent and subject-independent clustering is often not made explicit, even though it greatly affects the results. These problems are discussed in detail, and explicated experimentally, in Section 4.3.2.

As well as highlighting these problems and describing more rigorous evaluation settings that can address them, I propose a new deep HAR clustering model and test it under these settings, showing that it outperforms existing methods (insofar as a comparison can be made, with respect to the above points). I then present ablation studies on its main components, and a computational cost analysis, showing the execution time of each component. Mine is the first deep HAR clustering model not to require the reconstruction loss of an autoencoder, making it more efficient and better able to scale to more complex datasets. Below is a brief summary of my contributions.

- I explicate differences in evaluation procedures for deep HAR clustering, and show empirically that these differences can affect performance metrics. While my main focus is on deep HAR clustering, much of the analysis, including the important distinction between subject-dependence and subject-independence, applies to HAR clustering in general.

- I discuss suitable evaluation settings to use for HAR clustering. Adoption of these recommendations by future works will enable a direct comparison and benchmarking of deep HAR clustering models (and HAR clustering models more generally), and thus accelerate progress in the field.

- I describe a streamlined and scalable deep HAR clustering model. On six public datasets, this model performs better than or on par with existing models (insofar as

they can be compared). I also present ablation studies showing the contributions of its components.

The rest of this chapter is organized as follows. Section 4.2 provides an overview of related work. Section 4.3 explicates the shortcomings of existing evaluation methods for HAR clustering. In Section 4.4, I describe my new method for HAR. Section 4.5 then presents the results of my method under my proposed evaluation settings, and Section 4.6 summarizes my work.

4.2 Related Work

Machine learning has been identified as a promising approach to HAR since at least 2000 (Hongeng, Bremond, and Nevatia, 2000), with early works using, e.g., naive Bayes (Tapia, Intille, and Larson, 2004), support vector machines (Z. He and L. Jin, 2009), and generalized discriminant analysis (A. M. Khan, Y.-K. Lee, S.-Y. Lee, and Kim, 2010). These machine learning algorithms require feature engineering, and in the case of HAR, this is commonly done by taking statistical quantities such as mean and higher moments of the raw signal, in both the time and frequency domains. These can be combined with more bespoke features (M. Zhang and Sawchuk, 2011; Z. He and L. Jin, 2009).

Deep learning is a form of machine learning that does not require hand-crafted features, but rather can learn to extract features from a raw input (LeCun, Bengio, and G. Hinton, 2015; G. E. Hinton, Srivastava, Krizhevsky, Sutskever, and Salakhutdinov, 2012; Zeiler and Fergus, 2014) Applied to HAR, not only does deep learning avoid the need for feature engineering, but has also been shown to achieve better accuracy than feature-engineered models (Alsheikh, Selim, Niyato, Doyle, S. Lin, and H.-P. Tan, 2016; Ferrari, Micucci, Mobilio, and Napoletano, 2019). Network architectures include convolutional neural networks (CNNs) (M. Zeng, L. T. Nguyen, B. Yu, Mengshoel, J. Zhu, P. Wu, and J. Zhang, 2014; J. Yang, M. N. Nguyen, San, X. L. Li, and Krishnaswamy, 2015; Y. Chen and Xue, 2015; W. Jiang and Z. Yin, 2015; Ronao and S.-B. Cho, 2015) and recurrent neural networks (RNNs) (Murad and Pyun, 2017; M. Inoue, S. Inoue, and Nishida, 2018; D. Singh, Merdivan, Psychoula, Kropf, Hanke, Geist, and Holzinger, 2017). See (Hammerla, Halloran, and Plötz, 2016) for an empirical comparison of CNNs and RNNs for HAR. See (K. Chen, D. Zhang, L. Yao, B. Guo, Z. Yu, and Y. Liu, 2021; J. Wang, Y. Chen, Hao, X. Peng, and L. Hu, 2019) for summaries of recent deep HAR models.

There has been increasing interest in reducing the supervision needed for HAR. The first efforts in this direction were semi-supervised methods that trained on unlabelled data alongside labelled data (F. Li and Dustdar, 2011), that used unlabelled data for pretraining (Y. Li, D. Shi, B. Ding, and D. Liu, 2014; Alsheikh, Selim, Niyato, Doyle, S. Lin, and H.-P. Tan, 2016), or that investigated the optimality of the label-set by comparing to cluster-labels (Mejia-Ricart, Helling, and Olmsted, 2017). However, training these semi-supervised models still requires some labelled data. A recent survey on HAR described the need for fully unsupervised models as urgent due to the difficulty obtaining labels (J. Wang, Y. Chen, Hao, X. Peng, and L. Hu, 2019). Another (K. Chen, D. Zhang, L. Yao, B. Guo, Z. Yu, and Y. Liu, 2021) discussed the advantages of unsupervised HAR models but notes a disadvantage

that most are restricted to feature extraction and cannot provide labels. Deep clustering models can redress this problem by interpreting cluster membership as labels.

As is the case for supervised models, most HAR clustering models begin with a feature extraction stage. While some works apply clustering algorithms directly to the raw sensor signal (Trabelsi, Mohammed, Chamroukhi, Oukhellou, and Amirat, 2013), the most common approach is to first extract features from the sensor signal, and then cluster the extracted features. Initial HAR clustering models performed feature extraction by computing statistical quantities (Kwon, K. Kang, and Bae, 2014; I. P. Machado, Gomes, Gamboa, Paixão, and Costa, 2015; Y. Lu, Y. Wei, L. Liu, J. Zhong, L. Sun, and Y. Liu, 2017). H. He, Y. Tan, and J. Huang (2017) replace statistical features with the discrete wavelet packet transform (R. X. Gao and R. Yan, 2010) followed by principal components analysis for dimensionality reduction. Clustering is performed using fuzzy c-means with a novel initialization method based on cosine similarity. Another HAR clustering model is proposed by Sheng and Huber (2020), which includes a deep autoencoder in the feature extraction stage with two additional loss terms to encourage locality and temporal consistency. Statistical feature extraction is dispensed with completely by H. Ma, Z. Zhang, W. Li, and S. Lu (2021), replaced with a CNN-BiLSTM autoencoder plus pseudo-label training (Caron, Bojanowski, Joulin, and Douze, 2018). McConville, Santos-Rodriguez, Piechocki, and Craddock (2021) combine a deep autoencoder with UMAP (McInnes, Healy, and Melville, 2018), for dimensionality reduction, followed by clustering using a GMM.

4.3 Problems with Existing Literature

I identify three problems with the existing field of HAR clustering, shown with respect to existing works in Table 4.1.

- Different works often report on different datasets using different metrics. Many works report results on their own new dataset, and so cannot compare to prior works. Additionally, the datasets are often private.
- The exact evaluation criteria are unclear. There are multiple ways of evaluating the performance of a model, which can give significantly different results. Of particular importance is whether each subjects' data is clustered individually or whether all data is clustered together.
- Code is not released, making reproducibility difficult or impossible.

These problems make it hard for new researchers in the field to assess the best existing models from which to build, and difficult for them to determine whether they have improved over these existing models. Consequently, progress is held back.

In the following sections, I address these issues. Section 4.3.1 describes my choices of datasets on which to evaluate performance, and the reasoning behind these choices. Section 4.3.2 discusses the effect of different evaluation settings on performance metrics, and demonstrates these effects empirically, by showing that the same model can produce significantly different results in different evaluation settings.

4.3.1 Datasets

I select six suitable wearable-sensor HAR datasets for measuring clustering performance: Physical Activity Monitoring (PAMAP2) (Reiss and Stricker, 2012), Human Activity Recognition using Smartphones (UCI-Sm) (Anguita, Ghio, Oneto, Parra, and Reyes-Ortiz, 2013), WISDM-v1 (Kwapisz, Weiss, and S. A. Moore, 2011), WISDM-watch (Weiss, Yoneda, and Hayajneh, 2019), Realistic Sensor Displacement (REALDISP) (under the 'ideal placement' setting) (Banos, Damas, Pomares, Rojas, Toth, and Amft, 2012), and Heterogeneous Human Activity Recognition (Stisen, Blunck, Bhattacharya, Prentow, Kjærgaard, Dey, Sonne, and Jensen, 2015). The details of subjects, activities, and sensors for each dataset are shown in Table 4.3. There are three reasons for selecting these particular datasets:

- They are all easily accessible in the UCI repository.
- They have been used by some previous works, and so enable comparison. Although, as shown in Table 4.1, there is not as much consistency in the use of datasets as would be ideal.
- They vary in number of activities, number of data points, number of subjects, and number of sensor channels, helping to measure generalization ability. UCI-Sm and WISDM-v1 are smaller datasets, with only a few channels and activity classes. The other four datasets are more complex. WISDM-watch is unique in having a large number of users, 51, and HHAR is unique in having a large number of data points. REALDISP is a large dataset with many sensors channels. This set of datasets thus tests a model's performance in a range of settings, from a small to a large number of users, from a small to a large number of clusters and from simple hardware to many wearable sensors with a rich array sensor channels.

4.3.2 Ambiguous Evaluation Settings

I identify two ambiguities in how HAR clustering models are evaluated, subject-dependent vs. subject-independent, and window-wise vs. point-wise. The former refers to whether clustering was performed on all subjects' data at once (subject-independent) or on each subjects' data separately (subject-dependent). For supervised models, specification of the train-test split can disambiguate subject-dependence vs. independence, by specifying that, e.g., data for users X, Y, Z was used for testing. Clustering, however, does not typically use a train-test split, and the question of subject-dependence vs. independence is almost always unclear (see Table 4.1). The latter refers to whether data points are taken to be the sliding window or each time point. Each time point has one label, but training collates multiple time points into windows. (The window size is 512 in all experiments.) There is ambiguity as to whether data points should be taken to be the windows or the time points. This type of ambiguity can be present in supervised models too.

I empirically investigate the effect of these two factors, by training and testing a simplified version of my proposed model (described in Section 4.4) under the four resulting settings. The results are displayed in Table 4.2.

Table 4.1: Previous HAR clustering models relative to the evaluation criteria outlined in Section 4.3. There are a number of different datasets and metrics, but none releases their code, and almost all are ambiguous as to subject independence (indicated S-dep below.)

Name	Datasets	Metrics	Code Released	S-Dep
(Kwon, K. Kang, and Bae, 2014)	own (private)	ACC, NMI	no	unlcear †
(Trabelsi, Mohammed, Chamroukhi, Oukhellou, and Amirat, 2013)	own (private)	ACC, precision, recall	no	unclear
(Y. Lu, Y. Wei, L. Liu, J. Zhong, L. Sun, and Y. Liu, 2017)	own (private)	ACC, precision, recall, specificity, ARI, FM-index	no	unclear
(I. P. Machado, Gomes, Gamboa, Paixão, and Costa, 2015)	own (private)	ACC	no	both
(H. He, Y. Tan, and J. Huang, 2017)	WISDM-v1	RI, ARI	no	unclear‡
(Sheng and Huber, 2020)	PAMAP2, SBHAR, REALDISP	ACC, ARI, NMI	no	unclear
(H. He, Y. Tan, and W. Zhang, 2018)	DSAD	RI, ARI, FM-index	no	yes
(Jothi, 2018)	UC other*	ARI, Jacard-index, Silhouette-index	no	unclear
(Mohmed, Lotfi, Langensiepen, and Pourabdollah, 2018)	own (private)	other (graphical)	no	n/a
(H. Ma, Z. Zhang, W. Li, and S. Lu, 2021)	HAR, MotionSense,[1] MobiAct,[2] own (private)	precision, recall, F1, NMI	no	unclear
Dobbins and Rawassizadeh (2018)	HHAR	silhouette-index	no	unclear *
ours	PAMAP2, UCI-Sm, WISDM-v1, WISDM-watch, REALDISP, HHAR	ACC,ARI,NMI,F1	yes	yes

[1](Altun, Barshan, and Tunçel, 2010)
[2](Malekzadeh, Clegg, Cavallaro, and Haddadi, 2018)
† mentions different cluster sizes for different subjects, implying subject-dependence
‡displays confusion matrix referring to one subject only
* discusses subject heterogeneity within activity classes, implying subject-independence

Columns one and two are subject-dependent. They train a separate clustering model on each subject and average the results across these separate models, weighting each subject by their number of data points. Columns three and four are subject-independent. They train a single clustering model and cluster all subjects' data at once. The subject-dependent models outperform the subject-independent models by a large margin. This is in keeping with results from the supervised domain, where it has been noted that training on some data from the test subject significantly improves performance (Reiss and Stricker, 2012; Suh, Rey, and Lukowicz, 2021), which suggests the existence of subject-specific features in how activities are performed. The large difference in results between these two settings, evidenced in Table 4.2, means that it is crucial that models specify which they are using. Moreover, the two are fundamentally different tasks, one discovering patterns in the activity signal of a specific user, and the other learning generalized activities, independent of who is performing them.

Columns one and three in Table 4.2 are window-wise, while columns two and four are point-wise. The former treats each window as a data point whose label is the most commonly occurring label across all time points in that window. The latter treats each time point as a data point. Using overlapping windows means that each time point appears in multiple windows. In order to produce a single predicted label for each time point, the window-wise

Table 4.2: Evaluation under the four settings corresponding to the two ambiguities described in Section 4.3.2. Window-wise vs. point-wise does not affect results, but subject-dependent vs. subject-independent does. The subject-dependent settings performs substantially better across all datasets and metrics. It is therefore essential that HAR clustering works specify whether they are testing in the subject-dependent or subject-independent setting.

		window-wise subject-dependent	point-wise subject-dependent	window-wise subject-independent	point-wise subject-independent
PAMAP	ACC	66.28	66.27	48.30	47.35
	NMI	64.95	64.80	46.61	48.14
	ARI	50.57	50.54	30.44	31.31
	F1	65.63	65.57	45.26	45.73
UCI-Sm	ACC	50.57	50.73	38.95	38.93
	NMI	56.36	56.77	35.59	35.57
	ARI	39.57	39.50	23.94	23.92
	F1	46.74	46.88	35.32	35.28
WISDM-v1	ACC	72.15	72.33	50.14	50.04
	NMI	69.44	69.40	38.78	38.80
	ARI	60.05	60.03	33.07	33.06
	F1	64.88	64.95	38.91	38.88
WISDM-watch	ACC	78.40	78.48	25.58	25.56
	NMI	84.68	84.71	28.38	28.36
	ARI	72.22	72.17	12.60	12.59
	F1	77.61	77.67	25.32	25.31
REALDISP	ACC	89.60	89.60	51.37	51.36
	NMI	93.87	93.97	71.80	71.79
	ARI	88.60	88.53	43.91	43.87
	F1	84.36	84.38	47.00	46.99
HHAR	ACC	53.80	53.80	47.93	48.25
	NMI	51.62	51.62	38.42	38.52
	ARI	38.14	38.07	25.60	25.78
	F1	52.57	52.48	49.08	49.36

setting takes the most commonly occurring label across the multiple windows that contain that time point. (I also explored other means of combining these multiple labels, with very similar results.) There is essentially no difference between window-wise and point-wise evaluation so, although existing works are ambiguous as to which is being employed, this ambiguity does not prevent a clear assessment of performance. The results presented in Section 4.5 are all in the point-wise setting.

4.3.3 Subject-dependence in Real-World Applications

Given the large difference between subject-dependent and subject-independent clustering models, it is worth considering how these two different settings might be reflected in real-world applications. There are two questions that arise: what is the cause of the greater accuracy of subject-dependent clustering? and which setting should be used when deploying a HAR model?

To the first question, the improved accuracy in subject-dependent clustering is clearly suggestive of individual differences within the same activity, e.g. person A walking produces a different sensor reading to person B walking. This is consistent with recent physiology

Table 4.3: Information on each of the datasets on which I report results. Accel = 3d accelerometer, gyro = 3d gyroscope, and magneto = 3d magnetometer. Total time points = the sum of time points across all users, after discarding those without labels and those with missing data.

Name	Date Released	Number of Activities	Number of subjects	Sensors	Channels	Total Time Points
PAMAP2	2012	12	9	3 x (accel, gyro, magneto)	51	1921431
UCI-Sm	2013	6	30	phone accel and gyro	6	71968
WISDM-v1	2011	5	36	phone accel	3	1085363
WISDM-watch	2019	18	51	phone and watch accel and gyro	12	3635842
REALDISP	2012	33	17	9 x (accel,gyro, magneto, orient)	117	669618
HHAR	2015	6	9	2 x (accel, gyro)	12	11279265

research, showing that human biomechanics is highly individual specific, with the differences even persisting reliably across time (Horst, Mildner, and Schöllhorn, 2017; Hug, Vogel, Tucker, Dorel, Deschamps, Le Carpentier, and Lacourpaille, 2019; Horst, Lapuschkin, Samek, Müller, and Schöllhorn, 2019). There can also be differences in the distribution of activities performed by different users. For example, consider a user who works in a tall office building and regularly takes the stairs. The cluster model is much more likely to discover "stair walking" (or "up/downstairs walking") for this user than for one who barely climbs stairs at all in a typical week. There is survey evidence indicating individual differences in stair usage, even among those in the same building (Kinsey, Galea, and Lawrence, 2012). In a subject-independent model, both of these users would have to share the same set of clusters, and this would be a disadvantage to at least one of them. If there was no cluster corresponding to stair-walking, then for the first user, the cluster for walking would have to include all these instances of stair-walking, which could have significantly different features from flat walking. This would make the walking cluster difficult to learn. If there was a cluster for stair-walking then, from the perspective of the second user's data, it would be an unneeded extra cluster which could only contain false positives. A third factor that likely contributes to the improved accuracy in subject-dependent models is the individual differences in sensor placement. Many studies have shown a significant effect of sensor placement on the detection of activities (Kunze and Lukowicz, 2008; Kunze and Lukowicz, 2014; Banos, Toth, Damas, Pomares, and Rojas, 2014). One user might wear their watch on their dominant hand, another on their non-dominant hand; or one user might carry their phone in a side pocket, another in their back pocket. These differences in placement would produce different sensor readings. That different users place sensors differently was verified by (Banos and Tóth, 2014).

To the question of whether real-world applications should use subject-dependent or subject-independent models, the answer will vary depending on the goals of the application. There are advantages and disadvantages to each. The obvious advantage of subject-dependence is the improved accuracy. A user of a sports watch, for example, would benefit from a model that was trained on only her data, not anyone else's, as this model would be better able to detect which activity she is performing. In general, when the data for

each individual is considered separately, then subject-dependent clustering may be superior because of the improved accuracy. However, when trying to compare or collate data from multiple users, subject-independence would likely be preferred, as otherwise there would not be a clear correspondence between the cluster labels for one user and those for another. If, for example, one wanted to determine how much time the average person of a given demographic spends walking per week, they would need to know that walking meant the same thing for each user, but if subject-dependent clustering had been used, this could be called into question. One user might just have walking, another have brisk walking and leisurely walking, another have stairs walking and flat walking, another treadmill walking and outdoor walking etc., and it would not be obvious how to align these different clusters. If subject-independent clustering had been used, this would not be an issue.

4.4 Proposed Deep HAR Clustering Method

The proposed model for deep HAR clustering is based on the technique of pseudo-label training, extended with a novel method for selecting points on which to pseudo-label train. Pseudo-label training allows the encoder weights to be iteratively refined with the clustering model (see Section 2.3 for a description). However, the pseudo-labels are noisy and often incorrect. As explored in Chapter 3, filtering out the least confident label while pseudo-label training can improve performance. In this chapter, I propose a novel method of filtering the pseudo-labels. At each iteration, the encoder is trained only on those data points that received the same cluster label as they did in the previous iteration. I also implement a graded label filtering, by double-weighting the training updates for the points that received the same cluster label in every iteration so far. Formally, let X be the space of data points. Elements of X are windowed sequences of sensor readings of length 512. The encoder network f_θ and clustering model g are then represented by the following functions

$$f_\theta : X \to Z \,,$$
$$g : Z \to \{0, \ldots, K-1\} \,,$$

where the latent space Z is of lower dimension than X, θ denotes the network parameters, and K is the user-defined number of clusters. Let T be the total number of training iterations, and $\theta_j, 1 \leq j \leq T$ be the value of the network parameters at iteration j. Then, I define the full deep clustering model at iteration j as

$$C_j \coloneqq g \circ f_{\theta_j} \,,$$
$$C_j : X \to \{0, \ldots, K-1\} \,.$$

The encoder loss at iteration $j+1$ is then given by

$$\mathcal{L}_j = \sum\nolimits_{M_j} CE(h(f_\theta(x_i)), C_j(x_i)) + \sum\nolimits_{S_j} CE(h(f_\theta(x_i)), C_j(x_i)) \,,$$

where $h : Z \to (0,1)^K$ is the softmax classifier used for pseudo-label training (it is discarded after training), CE is the categorical cross-entropy loss,

$$S_j \coloneqq \{x \in X | C_j(x) = C_{j-1}(x)\} \,,$$

$$M_j \coloneqq \{x \in X | \forall 1 \leq k < j, C_j(x) = C_k(x)\} \,.$$

Here, S_j and M_j correspond to $SemiMask$ and $Mask$ in Algorithm 2, respectively, and allow for my proposed method of graded label filtering. Note that $M_j \subset S_j$.

Existing deep HAR clustering models all require an autoencoder to generate feature vectors for clustering (Sheng and Huber, 2020; H. Ma, Z. Zhang, W. Li, and S. Lu, 2021; McConville, Santos-Rodriguez, Piechocki, and Craddock, 2021). This effectively doubles the time and space requirements, as a decoder must be trained in conjunction with the encoder. Autoencoders can also present some problems for clustering, as they learn to reconstruct every detail of the input, including irrelevant features and noise. This has been well-documented in the case of image clustering, and becomes a greater problem the larger the input is; see (Mrabah, N. M. Khan, Ksantini, and Lachiri, 2020) and the references therein for a full discussion. My method, in contrast, uses a single streamlined loss which does not require a decoder, and thus can scale better to richer datasets with more sensors, both in terms of computational costs and accuracy. This is supported by the results from Section 4.5.

Before clustering the latent space, UMAP is applied (uniform manifold approximation (McInnes, Healy, and Melville, 2018)), as a second round of dimensionality reduction, reducing the latent dimension from 32 to 2. Clustering is performed with a hidden Markov model (HMM) to capture temporal consistency. As with previous HAR clustering models, the number of clusters is set manually to be equal to the number of classes in the dataset. The label filtering method described above identifies, each time it is run, a subset of confident labels. It can thus be repeated a number of times. If, in any of the repetitions, a data point received a confident label, then the output of the model for that data point is its most recent confident label. Otherwise, the output of the model is the label from the final repetition. This process is iterated until the set of points that have ever received a confident label stops increasing.

A final feature of my model is that, for the five smaller datasets, I reduce the step size from 100 (as is standard) to 5, to increase the number of data points. This reduction provides 20 times more data, which improves the training of the encoder, see Table 4.7. However HHAR, having by far the most data points, but comparatively few sensors (12, compared to, e.g., 51 for PAMAP or 117 for REALDISP), does not require additional data, so I keep the original step size of 100. The entire method is described in Algorithm 2.

To evaluate the four settings discussed in Section 4.3.2 and presented in Table 4.2, I use a pared-down version of the main model. This pared-down version differs from the above in that it removes UMAP and label filtering, does not repeat to find more points with agreed labels, and does not decrease the step size. That is, it simply encodes the data using the same convolutional architecture as the main model, with a step size of 100, clusters the encodings using a HMM, and then performs pseudo-label training.

Algorithm 2 Training algorithm

$X \leftarrow$ data;
$Mask, SemiMask \leftarrow X$;
$Final \leftarrow$ empty hash table, will store the final cluster labels for the dataset;
while $|Final|$ increases **do**
  $f \leftarrow$ encoder, randomly initialized
  for $i = 1, \ldots, 10$ **do**
    $Z_1 \leftarrow f(X)$
    $Z_2 \leftarrow UMAP(Z_1)$
    $C, P \leftarrow HMM(Z_2)$
    ▷ C and P are hash tables, $C[x]$ and $P[x]$ store, respectively, the cluster labels
    ▷ and probabilities assigned for data point x under the hidden Markov model (HMM) clustering model
    for $x \in X$ **do**
      if $i > 1$ and $c(x) \neq c'(x)$ **then**
        remove x from $Mask$ and $SemiMask$
      else
        add x to $SemiMask$
      end if
    end for
    for $epoch = 1, \ldots, 5$ **do**
      for $x \in X$ **do**
        if $x \in Mask$ **then**
          train on $x, c(x)$
        else if $x \in SemiMask$ **then**
          train on $x, c(x)$, weighted by 0.5
        end if
      end for
      $c'(x) \leftarrow c(x)$
    end for
  end for
  for $x \in Mask$ **do**
    $Final[x] \leftarrow c(x)$
  end for
end while
▷ For all data points that were not in $Mask$ in any iteration, use the label from the
▷ last iteration
for $x \in X \setminus Final$ **do**
  $Final[x] \leftarrow c(x)$
end for

Table 4.4: Comparison with (McConville, Santos-Rodriguez, Piechocki, and Craddock, 2021) and various supervised methods. The supervised methods are (Dua, S. N. S, and V. B. S, 2021) on PAMAP, UCI-Sm, and WISDM-v1, (Aljarrah and Ali, 2021) on REALDISP, and (Qin, Y. Zhang, S. Meng, Qin, and Choo, 2020) on HHAR.

		ours	n2d	supervised
PAMAP	ACC	**86.3**	86.0	95.3
	NMI	**88.4**	85.4	-
	ARI	**81.4**	78.3	-
	F1	80.1	**83.7**	95.2
UCI-Smartphone	ACC	**65.9**	52.1	96.2
	NMI	**68.6**	55.8	-
	ARI	**56.3**	38.7	-
	F1	**64.4**	48.5	96.2
WISDM-v1	ACC	**75.3**	70.5	97.2
	NMI	**76.0**	69.1	-
	ARI	**65.5**	58.1	-
	F1	**68.9**	63.4	97.2
WISDM-watch	ACC	**91.7**	84.8	-
	NMI	**93.8**	88.9	-
	ARI	**88.1**	79.0	-
	F1	**92.4**	84.5	-
REALDISP	ACC	**91.0**	80.3	99.8
	NMI	**95.2**	90.4	-
	ARI	**88.9**	79.0	-
	F1	**88.0**	72.5	99.8
HHAR	ACC	**62.3**	59.7	96.6
	NMI	**67.9**	60.8	-
	ARI	**50.5**	46.0	-
	F1	59.0	**59.3**	96.6
UCI-full-feats	ACC	**65.5**	64.9	-
	NMI	59.3	**67.0**	-
	ARI	46.3	**55.1**	-
	F1	64.5	**70.4**	-

Table 4.5: Comparison of my method with that of (Sheng and Huber, 2020). Mine performs better on both datasets and all metrics, with a more significant difference on the more complex dataset, REALDISP.

		ours	S20
PAMAP	ACC	**86.3**	85.4
	NMI	**88.4**	87.3
	ARI	**81.4**	80.2
REALDISP	ACC	**91.0**	68.1
	NMI	**95.2**	60.5
	ARI	**88.9**	80.4

Table 4.6: Comparison of my method with that of (H. Ma, Z. Zhang, W. Li, and S. Lu, 2021) on the HHAR dataset. I achieve a significantly higher NMI but a lower F1.

	HHAR	
	F1	NMI
ours	**67.9**	59.0
M21	55.0	**65.9**

4.5 Experimental Evaluation

Metrics. As discussed in Section 2.2, there are several possible choices of metrics for measuring clustering performance, because the objective is not as clearly defined as in most supervised problems. Therefore, it is useful to report multiple metrics, to capture different interpretations of the clustering objective. However, comparing different models is only possible if they report the same set of metrics, and this requires a standard set of metrics. For HAR clustering, I therefore propose (and use) the following four: clustering accuracy (ACC), adjusted Rand index (ARI), normalized mutual information (NMI), and macro-F1, each as defined in Section 2.2. It was stated in Section 2.2 that supervised metrics enable a direct comparison with supervised models, and this is exemplified in Table 4.4.

Table 4.7: Ablation studies on the main components of my model.

		ours	no UMAP	no label-filter	GMM	step 100	net. dim.
PAMAP	ACC	**86.3**	56.0	73.3	78.7	81.6	65.06
	NMI	**88.4**	52.9	76.8	81.0	81.5	64.32
	ARI	**81.4**	39.8	62.72	69.7	72.4	50.11
	F1	**80.1**	52.9	71.4	76.7	78.4	63.69
UCI-Sm	ACC	**65.9**	49.8	60.8	58.7	62.5	55.63
	NMI	**68.6**	52.7	63.8	62.4	67.2	56.38
	ARI	**56.3**	36.9	49.3	47.1	52.1	41.71
	F1	**64.4**	44.8	58.9	58.3	60.6	54.52
WISDM-v1	ACC	**75.3**	68.8	68.6	71.5	69.9	65.72
	NMI	**76.0**	63.2	70.2	74.6	69.2	61.2
	ARI	**65.5**	55.1	56.0	60.9	55.9	51.15
	F1	**68.9**	58.6	60.4	63.7	64.9	57.29
WISDM watch	ACC	**91.7**	69.1	87.8	89.2	88.8	62.73
	NMI	**93.8**	80.8	90.5	92.1	92.6	70.31
	ARI	**88.1**	63.6	81.8	85.0	85.5	51.58
	F1	**92.4**	67.0	88.0	90.4	89.5	61.9
REALDISP	ACC	**91.0**	82.4	82.7	87.7	76.8	51.07
	NMI	**95.2**	91.0	92.1	93.0	91.5	68.73
	ARI	**88.9**	82.7	79.9	84.5	79.1	46.53
	F1	**88.0**	76.0	78.8	84.6	81.2	49.3
HHAR	ACC	**62.3**	57.9	58.5	61.2	-	64.84
	NMI	**67.9**	58.4	61.0	66.7	-	65.32
	ARI	**50.5**	45.8	45.4	50.2	-	52.3
	F1	**59.0**	56.4	54.9	58.6	-	63.68

Network Architecture and Training Parameters. The encoder network contains four convolutional layers, with batchnorm and max-pooling of size 2 after each. The filter sizes and strides for the convolutional layers are $(50, 2)$, $(40, 2)$, $(7, 1)$, $(4, 1)$, and the number of filters per layer are $4, 8, 16, 32$. In every layer, the convolutional filters are 1D with weight sharing across sensor channels. After the convolutional layers, all channels are combined with a fully connected layer (with input size $32 \times$ the number of sensor channels, and

output size 32). Weights are updated by Adam (Kingma and Ba, 2014) with learning rate 1e-3, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and weight decay of 0. The latent space has dimension 32. UMAP uses two components, minimum distance 0 and n_neighbours 60. This follows the recommendation in the UMAP documentation, to use minimum distnace=0 and use a large value for n_neighbours. As reported in the documentation, a small value of minimum distance increases data density, and a high value of n_neighbours captures more global structure of the data, both of which are useful for clustering. I do not explore tuning these parameters because I want to demonstrate the strong performance of the proposed method on the recommended defaults. Clustering is performed by HMM, whose emission probabilities are Gaussian distributions with no restrictions on the covariance matrices. I did not explore restricting the covariance matrices, as the main advantage of doing so would be to reduce run time for clustering, and this is not significant in the overall run time (see Table 4.8. The transition probabilities are set to $1 - p$ on the diagonal and $\frac{p}{K-1}$ on the off-diagonal, where p is the fraction of time points in the dataset that are followed by the same action. This is chosen because it agrees with the average true probability of transitioning to a new activity. I observed that this makes little difference compared to random initialization. The window size is 512 for all datasets. I do not explore alternative values for the step size. For information on the step size, see Section 4.4. As is done by the methods of Chapter 3 and 5, the number of clusters in the clustering models is set to the number of ground truth classes. Automatically determining the correct number of clusters is generally a difficult problem, and all the existing works from Table 4.4, 4.5 and 4.6 also set it manually to the number of ground truth classes.

The cluster labels are then used for pseudo-label training as first proposed in (Caron, Bojanowski, Joulin, and Douze, 2018). The softmax classifier used for pseudo-label training is a multi-layer perceptron (MLP) with a single hidden layer of 250 units. After the encoder has been pseudo-label trained, I cluster again. Training alternates ten times between clustering and pseudo-label training the encoder on the cluster labels, training for five epochs each iteration.

4.5.1 Comparisons with Prior Work.

Tables 4.4, 4.5 and 4.6 compare, respectively, the performance of my model to that of (McConville, Santos-Rodriguez, Piechocki, and Craddock, 2021), (Sheng and Huber, 2020) and (H. Ma, Z. Zhang, W. Li, and S. Lu, 2021). All evaluations are in the pointwise subject-dependent setting. For the latter two, comparison is only possible on certain datasets and metrics, as the authors do not report on all the datasets and metrics that I do, and they do not release their code. I attempted to reimplement their methods using the details in the respective papers, but the implementation details were insufficient and I was unable to, and where I contacted the authors to ask for access to their code, I received no response. The reasons for testing on these six datasets are given in Section 4.3.1. Table 4.4 also compares to recent supervised models, to give an indication of the gap between them and clustering models. The supervised models used are shown in the table caption.

I outperform (McConville, Santos-Rodriguez, Piechocki, and Craddock, 2021) on almost all datasets and metrics. The most significant difference is on the most complex dataset, REALDISP (see Table 4.3). This supports the claim that my method is better able to scale

to complex datasets because it avoids the need for an autoencoder. For comparison with McConville, Santos-Rodriguez, Piechocki, and Craddock (2021), I report on both UCI-Sm, which contains the raw sensor signal, and on UCI-feat, which contains statistical features for each window. (Both datasets and their details are available on the UCI repository.) The figures reported by (McConville, Santos-Rodriguez, Piechocki, and Craddock, 2021) on UCI-feat are significantly higher than I obtained by running their code, 80.1 and 68.3 for ACC and NMI, respectively. This difference could be due to their reported results using a larger number of clusters than the ground truth, which tends to increase ACC significantly while keeping NMI similar. For fair comparison, I fixed the number of clusters to the ground truth for all methods.

I outperform (Sheng and Huber, 2020) on both PAMAP and REALDISP. The margin is larger for REALDISP, again suggesting that my approach is better able to leverage the more complex information in REALDISP's 117 channels, because it does not use an autoencoder. I outperform (H. Ma, Z. Zhang, W. Li, and S. Lu, 2021) on NMI but not F1. Partly, this could be due to their reporting micro-F1 (not specified but implied), which tends to be higher. My micro-F1 score on HHAR is closer at 62.30. This difference between metrics also highlights the value of reporting multiple metrics to capture different aspects of performance.

4.5.2 Ablation Studies.

Table 4.7 shows the results of removing each of the components of my model.

UMAP. For UMAP, there are two different ablation settings. The first removes UMAP and clusters the unreduced latent vectors of size 32, the second removes UMAP and replaces it with an additional network for dimensionality reduction. The additional network is a fully-connected MLP with one hidden layer of size 256, and output size 2 so that it reduces to the same dimensionality as UMAP did. Comparing the main model with the first UMAP ablation setting shows that UMAP gives an improved performance across all metrics and datasets, over having no dimensionality reduction. This is consistent with its use in image clustering models (Allaoui, Kherfi, and Cheriet, 2020). Comparison with the net. dim. setting allows us to investigate further how UMAP improves clustering performance. Comparing net. dim. with the main model, there is, in general, a drop in performance. This drop is larger on datasets with more sensors. Results on HHAR are comparable to using UMAP, likely because, as well as only 12 sensor channels, HHAR has the most data with which the additional network can train. Results on all other datasets are worse. PAMAP and, especially, REALDISP, show a significant drop. This suggests that, for simple datasets, most of the improvement with UMAP is due to it reducing dimension, rather than exactly how it reduces dimension, but for complex datasets with more information to fit into the reduced dimensions, the manner of dimension reduction is more important.

Label Filtering. Label filtering, the technique of removing likely-incorrect labels from pseudo-label training, so that the less noisy filtered labels can facilitate better training of the encoder, has been shown to improve clustering performance in prior works (J. Chang, L. Wang, G. Meng, S. Xiang, and C. Pan, 2017; Mrabah, N. M. Khan, Ksantini, and Lachiri,

2020) and in the work from Chapter 3. Here, my proposed method of label-filtering is also effective, significantly increasing all metrics across all datasets.

HMM for Clustering. Clustering with a HMM instead of a GMM markedly improves performance on PAMAP, UCI-Sm, and WISDM-v1. On WISDM-watch and REALDISP, where the metrics are already high, the improvement, though still significant, is less substantial. This suggests that, as the sensor data become richer, the further benefit of temporal information offers less improvement. (The GMM has full covariance matrices, convergence threshold of $1e-3$, maximum EM steps of 100, is initialized with k-means, and takes the best clustering from five initializations.)

Reduced Step Size. The smaller step size, which produces more data points to train the encoder, is effective at improving performance on all datasets. It is most effective on the two datasets with the most sensor channels, PAMAP and REALDISP, as they require larger networks, and hence more training data.

4.5.3 Computational Costs.

The computational costs are important to measure for HAR models, as use cases are often on constrained hardware, such as phones and watches. Table 4.8 shows the execution time (in sec) of my model, for each of the three phases of training: pseudo-label training the CNN, UMAP, and clustering. All runs use a single 12GB NVIDIA TITAN GPU. As expected, training and clustering are slower for datasets with more channels and more classes. UMAP is unaffected by either. At the time of writing, UMAP is only supported on CPU. However, a GPU implementation is expected soon, which will give a substantial speed-up. The run time is also affected by the number of iterations until convergence, as described in Section 4.4, with PAMAP and UCI-Sm taking the most iterations.

Table 4.8: Run time, in seconds, of my model, broken down by the three phases of training. The training time is the time taken to pseudo-label train the encoder CNN.

	Train	UMAP	Cluster	Total	Per Point
PAMAP	13616	81986	10056	109523	5.70e-2
UCI-Sm	4750	700	1700	7236	1.00e-1
WISDM-v1	1896	18252	639	20787	1.90e-2
WISDM-watch	22790	130580	40490	194843	5.36e-2
REALDISP	3303	12858	14892	31053	4.9e-1
HHAR	25386	16983	1137	43506	3.9e-3

4.6 Summary

In this chapter, I articulated and discussed the shortcomings of current evaluation procedures for HAR clustering models. I noted a lack of consistency in previous works' reporting

of results, and conducted experiments to show that a common ambiguity in evaluation, namely subject dependence, can significantly alter the results. I then discussed superior evaluation alternatives. Additionally, I introduced a new deep clustering model for HAR. Tested on six public datasets, under my proposed settings, it performs better than or on par with existing works, while also being more scalable and efficient by avoiding the need for an autoencoder. Finally, I presented ablation studies on, and measured the run-time of, each of my model's components. The material in this chapter can serve as a guide for future efforts in deep clustering of human activities, by articulating the requirements for comprehensive evaluation, and by detailing a number of effective techniques with respect to the new proposed model.

Similar to the work of Chapter 3, the model presented in this chapter combines deep learning and clustering in a single model, and can form a simple discrete representation, in the form of the cluster label, of the continuous input data. A difference from Chapter 3 is that the HAR data used here is temporally extended and assigning cluster labels to each activity also involves, implicitly, determining when one action ends and another begins. That the hidden Markov model is able to leverage this temporal extension is likely part of the reason for its improvement of performance. The deep clustering model that is the subject of Chapter 5 will also include a sort of temporality, this time in the learning of the model itself.

Chapter 5

Online Hard Clustering

Abstract

Image clustering has improved greatly in recent years, and is now not far behind supervised models in terms of classification accuracy on some popular image datasets. Top performing models employ pseudo-label training to provide a training signal to an encoding network (as described in Section 2.2 and used in the models from Chapter 3 and 4), and so need to alternate between clustering and training, which is costly computationally, and limited to settings where all data are available in advance. Online methods have also been proposed, which avoid these drawbacks by jointly updating the encoder and cluster parameters with respect to the clustering objective. However, this setting can easily reach the collapsed solution where the encoder maps all images to the same point and all data points are put into a single cluster. Existing models have employed various techniques to avoid this problem, most of which require data augmentation or which aim to make the average soft assignment across the dataset the same for each cluster. The method I propose in this chapter does not require data augmentation, but, unlike existing data-augmentation-free methods, I propose to take the hard assignments as a measure of collapse, and address it by regularizing the hard assignments. Using a Bayesian framework, I derive an intuitive optimization objective that can be straightforwardly included in the training of the encoder network. I test this empirically on four image datasets. First I show that it consistently avoids collapse more robustly than other methods. I also measure the resulting clustering accuracy, and show it to be higher than that of existing data-augmentation-free methods.I then conduct further experiments and analysis to support the claim that regularizing hard assignments is a better approach.

5.1 Introduction

Unsupervised computer vision models can be broadly divided into clustering models, which aim to place each data point into a cluster, so as to maximize similarity within clusters and minimize it between clusters, and contrastive learning models, which train a model without labels by encouraging representations for certain data points (positive pairs) to be similar, and those for others (negative pairs) to be dissimilar. While some methods propose to leverage contrastive learning for clustering performance (J. Li, P. Zhou, Xiong, and Hoi, 2020; Y. Cai, Z. Zhang, Y. Liu, Ghamisi, K. Li, X. Liu, and Z. Cai, 2021; Caron, Misra, Mairal, Goyal, Bojanowski, and Joulin, 2020), the two are fundamentally different objectives. The former returns a partition of the dataset, equivalently a discrete label for each point, and the latter returns a, normally continuous, feature vector for each data point. Contrastive learning requires some prior knowledge about the data in order to be able to select which data points should be similar or different. Often, this prior knowledge comes in the form of data augmentation, where two augmented versions of the same data point serve as positive pairs, and this requires knowing the appropriate semantic symmetries used for augmentation, e.g., reflection or brightness shift. Contrastive learning can also require a large number of negative pairs, if there are a large number of semantically meaningful axes of variation in the data. Clustering, on the other hand, can be performed even on datasets with no prior knowledge, and does not require the generation of negative pairs.

Deep clustering, as described in Section 2.3, refers to the combination of deep learning and clustering, where the data are first encoded with a deep neural network to a feature space, and then clustering is performed in the feature space. Deep clustering models (and clustering models more generally) can be classified as offline or online. Offline models process the entire dataset, and then assign all cluster labels at once. Online models, by contrast, assign a cluster label to each data point, or each batch, as it is processed. Offline methods tend to produce more accurate clusterings, as they can leverage information from later data points when assigning cluster labels to earlier data points. However, they are more expensive to train, as they must alternate between encoding the entire dataset/training the encoder for some number of epochs, and clustering the encoded data. The deep clustering models from Chapters 3 and 4 were both offline models, and they both required this alternation between clustering and training the encoder. Online models, on the other hand, can jointly encode and cluster so are less computationally expensive. They are also more versatile, being applicable in real-world settings where new data is constantly becoming available, as opposed to offline methods, which are limited to having a fixed, pre-defined dataset.

The disadvantage of online methods is that they are more difficult to train. A particular difficulty is that they run the risk of producing a degenerate solution where the large majority of data points are concentrated in a small number of clusters. In the extreme case, all points are placed into the same cluster. I refer to this as partition collapse. For example, an intuitive way to formulate a training procedure for online deep clustering is to define a clustering objective, and then for each new data point, update both the encoder network and the clustering parameters with respect to this objective. (The clustering parameters are the parameters of the algorithm used to cluster the encodings, e.g., the centroids in K-means.) However, this training objective is trivially minimized by the encoder network mapping all data points to the same point in feature space, where this point is equal to one of the

cluster centroids (i.e., total partition collapse). Several techniques have been proposed to avoid partition collapse, but they are often ad hoc and lack a rigorous technical foundation. Additionally, they take the soft assignments as a measure of partition collapse and propose to regularize the soft assignments to make them more uniform. In this chapter, I argue that soft assignments are not an accurate measure of partition collapse, and that we should instead focus on hard assignments.

I propose a method to regularize hard assignments. Specifically, I consider the problem of how to optimally assign each data point in a batch to the most appropriate cluster. I express this problem probabilistically in a Bayesian framework, where the partition support element is captured by a uniform prior across clusters. I then use this expression to derive a precise optimization objective, which I also show to be equivalent, up to a small error term, to the objective of maximizing the mutual information between batch indices and cluster assignments across a batch. This objective itself is too slow to solve exactly, but I devise a greedy approximation algorithm that can be implemented straightforwardly and results in an intuitive method for fully online clustering, which I term combinatorial assignment. Tested on four popular image clustering datasets, this method achieves a superior clustering performance and better partition support than existing methods. Finally, I analyze the role of hard vs. soft cluster assignments in my partition support method, and in previous methods, and make the case that regularizing hard assignments is a more effective approach.

My contributions in this chapter are briefly summarized below.

- I articulate a clear Bayesian framework of the problem of hard assignments in online deep clustering models.
- I use this framework to derive an optimization objective and prove that it is approximately equivalent to an information-theoretic framework that maximizes the mutual information of the batch indices with the cluster assignments in a batch.
- I devise a greedy algorithm to approximately solve this optimization objective.
- I show empirically that the resulting method significantly outperforms existing partition support methods, both in terms of avoiding partition collapse and in terms of resulting clustering accuracy.
- I conduct a further analysis of the performance of different partition support methods, which justifies my choice to focus on hard assignments.

The rest of this chapter is organized as follows. Section 5.2 gives an overview of related work, while Section 5.3 lays the theoretical foundations of my proposed method, including the comparison with entropy maximization, and describes a proposed greedy algorithm for optimizing the resulting objective. Section 5.4 reports empirical results and analysis, and finally Section 5.5 summarizes my findings.

5.2 Related Work

The first deep clustering models for images used autoencoders. After training the autoencoder to reconstruct the input, the decoder could be discarded, and the encoder used to

extract feature vectors which were fed to a clustering algorithm (B. Yang, Fu, Sidiropoulos, and Hong, 2017; P. Huang, Y. Huang, W. Wang, and L. Wang, 2014; J. Xie, R. Girshick, and Farhadi, 2016). Autoencoders are trained on reconstruction loss in pixel space, and while this can be successful on simple datasets such as MNIST, where images from the same class have significant pixel overlap, it does not transfer to more realistic datasets where, e.g., two different images of a cat could have very different pixel values but we still desire their feature vectors to be similar. Therefore, autoencoder-based clustering is limited for complex images.

An alternative method, known as pseudo-label training, was proposed in (Caron, Bojanowski, Joulin, and Douze, 2018), described in Section 2.3 and used in Chapters 3 and 4. Most recent deep clustering methods employ a version of pseudo-label training (J. Wu, Long, F. Wang, Qian, C. Li, Z. Lin, and Zha, 2019; Mrabah, N. M. Khan, Ksantini, and Lachiri, 2019; Van Gansbeke, Vandenhende, Georgoulis, Proesmans, and Van Gool, 2020; Mahon and Lukasiewicz, 2021; C. Niu, Shan, and G. Wang, 2021). The strongest performance has been obtained by those methods that carefully select and refine the pseudo-labels they use for training, e.g., my model from Chapter 3 is essentially human-level performance on simple datasets, and (C. Niu, Shan, and G. Wang, 2021) is close to supervised methods on more complex datasets. The disadvantage of pseudo-label training is that it is slow to alternate back and forth between computing pseudo-labels and using them as targets for training. Also, it is almost always required to be offline, because the pseudo-labels are computed using an offline clustering algorithm such as k-means, so cannot operate in a real-world setting in which new data is continuously becoming available.

Online deep clustering methods have also been proposed, which cluster each new data point as it arrives. The difficulty in this setting is the training of the encoder, as there is a risk of partition collapse, where (almost) every data point is placed in the same cluster. Contrastive learning, and those clustering models that employ it as a part of the training procedure, is much more resistant to partition collapse, because the negative pairs are encouraged to be represented differently. This can mean that they are to be placed in different clusters (J. Huang and S. Gong, 2021), or just that the encodings should be far apart (D. Zhang, Nan, X. Wei, S. Li, H. Zhu, K. McKeown, Nallapati, Arnold, and B. Xiang, 2021). Either approach helps resist all points having the same representation and cluster label. BYOL (Grill, Strub, Altché, Tallec, Richemond, Buchatskaya, Doersch, Pires, Z. Guo, Azar, Piot, k. k. k., Munos, and Valko, 2020) performs representation learning without negative pairs. Instead an "online" network is trained to predict the output of a "target" network, where the target network is a slow-moving average of the online network.

Among online cluster models, several partition support methods have also been proposed. Kulshreshtha and Guha (2018) avoids the problem entirely by freezing the encoder during clustering, but this requires pretraining it on a separate task. B. Gao, Y. Yang, Gouk, and Hospedales (2020) proposes an online autoencoder-based clustering model, where the reconstruction loss helps to avoid partition collapse, but this suffers from the limitation described above. Zhan, J. Xie, Z. Liu, Ong, and Loy (2020) continually reweights the loss function to encourage the assignment to smaller clusters. Additionally, when clusters decrease below a certain threshold, they are deleted, and the largest cluster is split in two using K-means.

The approach of H. Zhong, C. Chen, Z. Jin, and Hua (2020) is to minimize the sum

of squares of the marginal probability (i.e., soft assignment) of each cluster, marginalized over each training batch. Decomposing the expectation of the square, we see that this also involves minimizing the variance. However, the main method that the authors use is based on contrastive learning, so they do not fully rely on sum-of-squares minimization to avoid partition collapse.

A similar idea, employed by J. Li, P. Zhou, Xiong, and Hoi (2020), W. Hu, Miyato, Tokui, Matsumoto, and Sugiyama (2017), C. Niu, Shan, and G. Wang (2021), and C. Niu, J. Zhang, G. Wang, and J. Liang (2020), is to maximize the marginal entropy of cluster probabilities (i.e., soft assignments), marginalized over each training batch. For an input distribution X with corresponding soft assigned labels Y, both approximated over a batch, an extra term $-H(Y)$ is added to the loss function. As the entropy of a multinomial is maximized at the uniform distribution, this encourages more equally sized clusters. Entropy maximization has the advantage of a solid formal interpretation as part of the maximization of the mutual information between data and cluster assignments: by maximizing $H(Y)$ but minimizing $H(Y|X)$ (the latter is minimized explictly by W. Hu, Miyato, Tokui, Matsumoto, and Sugiyama (2017) and implicitly via contrastive learning by J. Li, P. Zhou, Xiong, and Hoi (2020)), we are maximizing

$$I(X;Y) = H(Y) - H(Y|X)\,.$$

However, the marginal entropy term tends to only be partially successful at preventing partition collapse, often the entire dataset is still put into only a small number of clusters. As well as being confirmed in my experiments in Section 5.4, this empirical weakness of entropy maximization for avoiding partition collapse is reported by W. Hu, Miyato, Tokui, Matsumoto, and Sugiyama (2017), and better results are found by explicitly constraining the soft cluster assignments using non-linear programming. J. Li, P. Zhou, Xiong, and Hoi (2020) do not rely entirely on entropy maximization, because they employ contrastive learning, which (as explained above) also helps to avoid partition collapse. Thus, while theoretically sound, entropy maximization is not sufficiently effective empirically.

Another approach to avoiding partition collapse is to directly impose a constraint on cluster assignments. Asano, Rupprecht, and Vedaldi (2019) proposed to constrain the soft cluster assignments to be marginally uniform across the dataset, i.e., each cluster has the same number of points assigned to it. This was originally an offline method, as the entire dataset was labelled at once, but it has been adapted to the online setting by Caron, Misra, Mairal, Goyal, Bojanowski, and Joulin (2020) and Kumar, Haresh, Ahmed, Konin, Zia, and Tran (2021) by enforcing uniformity on each training batch separately. This is effective in preventing partition collapse, but the constraint that each batch contains equal numbers of each class is too strict, as noted by Kumar, Haresh, Ahmed, Konin, Zia, and Tran (2021). A given batch is very unlikely to be exactly uniform across classes, so the ground-truth distribution of classes will almost certainly violate this constraint. This excessive strictness is unsatisfying theoretically, as well as hurting performance empirically.

5.3 Method

5.3.1 Problem Formulation

In this section, I present my proposed method for online deep clustering.

We want to make hard assignments of cluster labels. That is, if the encoding network is $f_{\theta_1} : \mathbb{R}^n \to \mathbb{R}^m$, the clustering model $g_{\theta_2} : \mathbb{R}^m \to \{1, \ldots, K\}$, and the batch size is N, then we seek a function of the form $\Gamma : \mathbb{R}^{N \times m} \to \{1, \ldots, K\}^N$. The difference between Γ and g_{θ_2} is that the former is used during training to assign an entire batch, while the latter is used at inference time and can assign each point individually. It is possible to use the clustering model itself independently on each point to assign batch labels during training, though this is not advisable, as discussed below.

If we have a method for batchwise assignment, then the training objective can be formulated as minimizing the distance of points from the centroids of their assigned clusters. This objective can be used to train both the encoder network and the clustering model:

$$\underset{\theta_1, \theta_2}{\operatorname{argmin}} \sum_{i=1}^{N} ||f_{\theta_1}(x_i), \mu_{k_i}||^2 , \tag{5.1}$$

where x is the input batch, $k_i = \Gamma(f_{\theta_1}(x))_i$ is the cluster label assigned to the encoding of x_i under f, and μ_k is the kth cluster centroid.

Finding a suitable assignment function Γ is non-trivial. Of course, Γ should take into account the distance of the feature vector to each centroid, and be more likely to assign it to the clusters whose centroids are closer. However, using only this rule, and assigning every vector to its closest centroid, we allow (5.1) to be trivially minimized by f_{θ_1} picking one centroid and mapping every input to it: $\forall x, f_{\theta_1}(x) = c_k$, for some $k \in \{1, \ldots, K\}$. I use the term "partition support" to refer to any method for avoiding this pathological state. Several partition support methods were discussed in Section 5.2, and are compared to my proposed method empirically in Section 5.4. I refer to my proposed method as combinatorial assignment, because it assigns labels using a prior over the combination of such labels under a multinoulli distribution.

5.3.2 Combination Assignment

Combinatorial assignment is based on a Bayesian formulation of the online clustering problem. Let $\mathcal{D}$ be the data distribution, $X \sim \mathcal{D}$ be a sampled batch of data, Z be the random variable defined by applying the feature extraction to X (i.e., Z is the output of a deep encoder network), and let Y be the assigned cluster labels. Here, we consider the neural network to be fixed, and we are just interested in the best hard assignment of cluster labels, given the extracted features. That is, we want to determine the values of Y with the maximum probability under the a posteriori distribution $p(Y|Z)$. We assume a uniform prior over K cluster labels

$$p(Y = k) = \frac{1}{K}, \text{ for } k \in \{1, \ldots, K\} ,$$

so, the prior probability of a batch containing exactly n_k labels for each cluster $k \in \{1, \ldots, K\}$ is

$$\left(\frac{1}{K}\right)^N \frac{N!}{\prod_{k=1}^K n_k!}\,, \tag{5.2}$$

where $N = \sum_{k=1}^K n_k$ is the batch size.

We then model each cluster as a multivariate normal distribution in feature space, so that the likelihood is given by

$$p(Z = z_1, \ldots, z_N | Y = k_1, \ldots, k_N) = \tag{5.3}$$
$$\prod_{i=1}^N \frac{\exp(-\frac{1}{2}(z_i - \mu_{k_i})\Sigma_{k_i}^{-1}(z_i - \mu_{k_i}))}{\sqrt{(2\pi)^d |\Sigma_{k_i}|}}\,,$$

where d is the dimension of the feature space, and μ_k and Σ_k are the centroid and covariance matrix of the kth cluster, respectively. If we further assume each cluster is spherical, with the same isotropic variance across all clusters, i.e., $\Sigma_k = \sigma^2 I,$ for $k \in \{1, \ldots, K\}$, then (5.3.2) simplifies to

$$\prod_{i=1}^N \frac{\exp(-\frac{1}{2\sigma^2}||z_i - \mu_{k_i}||^2)}{\sqrt{(2\pi\sigma)^d}}\,,$$

and the full a posteriori is

$$p(Y|Z) \propto P(Y)P(Z|Y) = \left(\frac{1}{K}\right)^N \frac{N!}{\prod_{k=1}^K n_k!} \prod_{i=1}^N \frac{\exp(-\frac{1}{2\sigma^2}||z_i - \mu_{k_i}||^2)}{\sqrt{(2\pi\sigma)^d}}$$
$$\propto \frac{\prod_{i=1}^N \exp(-\frac{1}{2\sigma^2}||z_i - \mu_{k_i}||^2)}{\prod_{k=1}^K n_k!}\,,$$

where we drop the constants that are independent of Y. Then, we obtain an optimization objective by minimizing the corresponding negative log-likelihood as follows

$$\begin{aligned}
\operatorname*{argmax}_Y p(Y|Z) &\propto \operatorname*{argmax}_Y p(Y)P(Z|Y) = \\
&= \operatorname*{argmax}_Y \log(\prod_{i=1}^N \exp(-\tfrac{1}{2\sigma^2}||z_i - \mu_{k_i}||^2)) - \log(\prod_{k=1}^K n_k!) = \\
&= \operatorname*{argmax}_Y \sum_{i=1}^N -\tfrac{1}{2\sigma^2}||z_i - \mu_{k_i}||^2 - \sum_{k=1}^K \log(n_k!) = \\
&= \operatorname*{argmin}_Y \sum_{i=1}^N ||z_i - \mu_{k_i}||^2 + 2\sigma \sum_{k=1}^K \log(n_k!)\,.
\end{aligned} \tag{5.4}$$

In matrix notation, the objective is

$$\operatorname*{argmin}_{Q \in \mathcal{B}^{N \times K}} \langle Q, \tilde{Q} \rangle + 2\sigma \log(\mathbb{1}_N^T Q!)\mathbb{1}_K \tag{5.5}$$
$$\text{subject to } Q\mathbb{1}_K = \mathbb{1}_M\,,$$

where $\tilde{Q}$ is the matrix of unnormalized log probabilities, i.e., $\tilde{Q}_{i,j} = ||z_i - \mu_j||^2$, $\langle \cdot, \cdot \rangle$ denotes the Frobenius inner product, $\mathbb{1}_a$ is an a-dimensional vector of all ones, and $\log$ and factorial are applied to each element in the K-dimensional vector $\mathbb{1}_N^T Q$. The requirement that Q be a Boolean matrix specifies that we seek hard assignments, and the constraint $Q\mathbb{1}_K = \mathbb{1}_M$ specifies that the rows of Q are one-hot vectors, i.e., each latent vector is assigned to exactly one cluster.

5.3.3 Solving the Optimization Problem

Unfortunately, it is too slow to solve (5.5) exactly. However, to motivate my assignment method, we can consider the simpler problem of assigning the last data point in a batch, given that the rest have already been assigned. That is, we maximize the conditional probability of the Nth assignment in a batch, conditioned on the $N-1$ previous assignments:

$$\begin{aligned}
&\operatorname*{argmax}_{k_N=1,\dots,K} p(y_N = k | y_1 = k_1, \dots, y_{N-1} = k_{N-1}; Z) = \\
&\operatorname*{argmax}_{k_N=1,\dots,K} \frac{p(y_1 = k_1, \dots, y_N = k_N | Z)}{p(y_1 = k_1, \dots, y_{N-1} = k_{N-1} | Z)} = \\
&\operatorname*{argmax}_{k_N=1,\dots,K} \log p(y_1 = k_1, \dots, y_N = k_N | Z) - \\
&\quad - \log p(y_1 = k_1, \dots, y_{N-1} = k_{N-1} | Z) = \\
&\operatorname*{argmax}_{k_N=1,\dots,K} -\sum_{i=1}^{N} ||z_i - \mu_{k_i}||^2 + 2\sigma \sum_{k=1}^{K} \log(n'_k!) + \\
&\quad + \sum_{i=1}^{N-1} ||z_i - \mu_{k_i}||^2 - 2\sigma \sum_{k=1}^{K} \log(n_k!) = \\
&\operatorname*{argmin}_{k_N=1,\dots,K} \sum_{i=1}^{N} ||z_i - \mu_{k_i}||^2 - \sum_{i=1}^{N-1} ||z_i - \mu_{k_i}||^2 + \\
&\quad + 2\sigma (\sum_{k=1}^{K} \log(n'_k!) - \sum_{k=1}^{K} \log(n_k!)) = \\
&\operatorname*{argmin}_{k_N=1,\dots,K} ||z_N - \mu_{k_N}||^2 + 2\sigma (\sum_{k=1}^{K} \log(n_k!) - \sum_{k=1}^{K} \log(n_k!')) \,, \qquad (5.6)
\end{aligned}$$

where n_k is the number of points assigned to cluster k before the Nth assignment, and n'_k is the number assigned to the kth cluster after all assignments have been made. This means that

$$n'_k = \begin{cases} n_k + 1 & k = k_N \\ n_k & \text{otherwise} \,. \end{cases}$$

Thus, (5.6) becomes

$$\begin{aligned}\underset{k_N=1,\ldots,K}{\operatorname{argmin}} \; ||z_N - \mu_{k_N}||^2 + 2\sigma \log(n_{k_N} + 1!) - \log(n_{k_N}!) = \\ \underset{k_N=1,\ldots,K}{\operatorname{argmin}} \; ||z_N - \mu_{k_N}||^2 + 2\sigma \log(n_{k_N} + 1)\,. \end{aligned} \tag{5.7}$$

To approximately solve (5.5), I employ a greedy algorithm that iteratively solves (5.7) with respect to the most confident assignment possible. That is, at each iteration, select the pair (i, k) (corresponding to (N, k_N) above) for which (5.7) is smallest, with i ranging over the indices of still-unassigned points, and k ranging over all clusters, and assign the ith point to cluster k.

5.3.4 Interpretation

Here, I show that the proposed approximation algorithm from Section 5.3.3 is closely equivalent to maximizing the mutual information between the assigned cluster labels and the batch index i, where $1 \leq i \leq N$ and N is the batch size. Specifically, I show that the greedy algorithm that iteratively solves (5.7) can be interpreted as (a close approximation to) iteratively making whatever assignment will maximize the mutual information between the batch index i and the cluster labels. First note that, because my proposed model makes hard assignments, the entropy of cluster labels given the batch index is automatically zero, and so recalling that

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)\,,$$

we see that the mutual information of the batch index i and the cluster labels equals the entropy of cluster labels. Below, I show that my proposed method, up to small approximation error, maximizes the entropy of cluster labels and, hence, the mutual information of cluster labels and the batch indices in each batch.

Lemma 6. *Let $X \in \mathbb{R}^{N\times K}$ be the matrix of already-made assignments in the current batch, and let $H^{(k)}$ be the marginal entropy after the new hard assignment is made to cluster k. Then*

$$H^{(k)} - H^{(k')} \approx \frac{1}{N+1}\left(\log(x_{k'} + 1) - \log(x_{k'} + 1)\right)\,. \tag{5.8}$$

Proof. Let $X \in \mathbb{R}^{N\times K}$ be the matrix of already-made assignments in the current batch after N points have been assigned, so that H, the current marginal entropy of X, is given by:

$$H = -\sum_{j=1}^{K}\left(\frac{1}{N}\sum_{i=1}^{N} x_{ij}\right)\log\left(\frac{1}{N}\sum_{i=1}^{N} x_{ij}\right)\,.$$

To simplify notation, let $x_j = \sum_{i=1}^{N} x_{ij}$. Then $H^{(k)}$, the marginal entropy after the new hard

assignment is made to cluster k, is given by

$$
\begin{aligned}
H^{(k)} =& -\frac{x_k+1}{N+1}\log\frac{x_k+1}{N+1} - \sum_{j=1,j\neq k}^{K}\frac{x_j}{N+1}\log\frac{x_j}{N+1} \\
=& \frac{-1}{N+1}\left((x_k+1)\log(x_k+1) + \sum_{j=1,j\neq k}^{K} x_j(\log x_j - \log(N+1))\right) \\
=& \frac{-1}{N+1}\left((x_k+1)\log(x_k+1) + \sum_{j=1,j\neq k}^{K} x_j\log x_j - \sum_{j=1,j\neq k}^{K} x_j\log(N+1)\right) \\
=& \frac{-1}{N+1}\left((x_k+1)\log(x_k+1) + \sum_{j=1,j\neq k}^{K} x_j\log x_j - N\log(N+1)\right) \\
=& \frac{-1}{N+1}\left((x_k+1)\log(x_k+1) + \sum_{j=1,j\neq k}^{K} x_j\log x_j\right) + \frac{N}{N+1}\log(N+1)
\end{aligned}
$$

Now, consider the difference $H^{(k)} - H^{(k')}$ between the entropy after making assignment k vs. after making a different assignment k'.

$$
\begin{aligned}
=& \frac{-1}{N+1}\left((x_k+1)\log(x_k+1) + \sum_{j=1,j\neq k}^{K} x_j\log x_j\right) - \left((x_{k'}+1)\log(x_{k'}+1) + \sum_{j=1,j\neq k'}^{K} x_j\log x_j\right) = \\
=& \frac{-1}{N+1}\left((x_k+1)\log(x_k+1) - (x_{k'}+1)\log(x_{k'}+1)\right) + \left(\sum_{j=1,j\neq k}^{K} x_j\log x_j - \sum_{j=1,j\neq k'}^{K} x_j\log x_j\right) = \\
=& \frac{-1}{N+1}\big((x_k+1)\log(x_k+1) - (x_{k'}+1)\log(x_{k'}+1)\big) + (x_{k'}\log x_{k'} - x_k\log x_k) = \\
=& \frac{-1}{N+1}\left(((x_k+1)\log(x_k+1) - x_k\log x_k) - ((x_{k'}+1)\log(x_{k'}+1) - x_{k'}\log x_{k'})\right) \\
\approx& \frac{-1}{N+1}\left((\log(x_k+1) + \frac{x_k}{x_k+1}) - (\log(x_{k'}+1) + \frac{x_{k'}}{x_{k'}+1})\right) \\
=& \frac{-1}{N+1}\left(\log(x_k+1) - \log(x_{k'}+1) - \frac{x_k - x_{k'}}{(x_k+1)(x_{k'}+1)}\right) \\
=& \frac{1}{N+1}\left(\log(x_{k'}+1) - \log(x_k+1) - \frac{x_{k'} - x_k}{(x_k+1)(x_{k'}+1)}\right) \\
=& \frac{1}{N+1}\left(\log(x_{k'}+1) - \log(x_k+1)\right) - \frac{1}{N+1}\left(\frac{x_{k'} - x_k}{(x_k+1)(x_{k'}+1)}\right)
\end{aligned}
$$

where the fourth last line uses the fact that $\log n \approx H_n$ to make the substitution

$$x_k\log x_k \approx x_k(\log(x_k+1) - \frac{x_k}{x_k+1})\,,$$

and similarly for $x_{k'}$. Note that the term $\frac{1}{N+1}\frac{x_k - x_{k'}}{(x_k+1)(x_{k'}+1)}$ is 0 in expectation and has absolute value $\leq \frac{N}{(N+1)^2}$. If we drop this small error term, then we get

$$H^{(k)} - H^{(k')} \approx \frac{1}{N+1}\left(\log(x_{k'}+1) - \log(x_k+1)\right), \tag{5.9}$$

as desired. □

Lemma 7. *Assume that N data points in a batch have already been assigned. Let $\mathcal{L}(k)$ be the batch likelihood, under the a K-component Gaussian mixture model with isotropic variance σ^2 (as described in Section 5.3.2), after the $(N+1)$th data point is assigned to cluster k. Let H^k be, as above, the entropy of cluster sizes after the $(N+1)$th data point has been assigned to cluster k. Then maximizing the objective $\log \mathcal{L}(k) + \lambda H^k$ with respect to the $(N+1)$th cluster assignment gives the following optimization problem*

$$\underset{k\in\{1,\ldots,K\}}{\operatorname{argmin}} \; ||z-\mu_k||^2 + \frac{2\lambda\sigma^2}{N+1}\log(x_k+1)$$

Proof. Maximizing $\log \mathcal{L}(k) + \lambda H^k$ with respect to the $(N+1)$th cluster assignment means we prefer to assign to cluster k over cluster k' if and only if

$$\log \mathcal{L}(k) + \lambda H^k > \log \mathcal{L}(k') + \lambda H^{k'} \iff \tag{5.10}$$

$$\log \mathcal{L}(k) - \log \mathcal{L}(k') > \lambda H^{k'} - \lambda H^k \iff \tag{5.11}$$

$$-\log \mathcal{L}(k) - (-\log \mathcal{L}(k')) < \lambda H^k - \lambda H^{k'} \iff \tag{5.12}$$

$$\log \mathcal{L}(k) - \log \mathcal{L}(k') < \frac{\lambda}{N+1}\left(\log(x_{k'}+1) - \log(x_k+1)\right), \tag{5.13}$$

where the last line uses lemma 6. Let z be the encoding of the $(N+1)$th point, as in Section 5.3.2. Then

$$\begin{aligned} -\log \mathcal{L}(k) &= -\log\left(\frac{\exp(-\frac{1}{2\sigma^2}||z-\mu_k||^2)}{\sqrt{2\pi\sigma^d}}\right) \\ &= \tfrac{1}{2}\log\left(2\pi\sigma^d\right) + \frac{1}{2\sigma^2}||z-\mu_k||^2 . \end{aligned}$$

Subbing this into (5.13), we get

$$\left(\tfrac{1}{2}\log\left(2\pi\sigma^d\right) + \frac{1}{2\sigma^2}||z-\mu_k||^2\right) - \left(\tfrac{1}{2}\log\left(2\pi\sigma^d\right) + \frac{1}{2\sigma^2}||z-\mu_{k'}||^2\right) <$$
$$\frac{1}{N+1}\left(\log(x_{k'}+1) - \log(x_k+1)\right) \iff$$
$$\frac{1}{2\sigma^2}(||z-\mu_k||^2 - ||z-\mu_{k'}||^2) < \frac{\lambda}{N+1}\left(\log(x_{k'}+1) - \log(x_k+1)\right) \iff$$
$$\frac{1}{2\sigma^2}||z-\mu_k||^2 + \frac{\lambda}{N+1}\log\left(x_k+1\right) < \frac{1}{2\sigma^2}||z-\mu_k||^2 + \frac{\lambda}{N+1}\log\left(x_k+1\right) \iff$$
$$||z-\mu_k||^2 + \frac{2\lambda\sigma^2}{N+1}\log\left(x_k+1\right) < ||z-\mu_{k'}||^2 + \frac{2\lambda\sigma^2}{N+1}\log\left(x_{k'}+1\right).$$

Choosing pairwise between all k, k' as per (5.3.4) is equivalent to choosing k so as to minimize

$$||z - \mu_k||^2 + \frac{2\lambda\sigma^2}{N+1} \log (x_k + 1) .$$

□

Remark 8. This shows that my proposed method closely approximates a maximization of the entropy of cluster labels. There is some similarity to those methods, discussed in Section 5.2, that use an additional loss term to encourage greater entropy of soft assignments in each batch, but the important difference here is that we are maximizing the entropy of hard assignments.

Theorem 9. *Assume that N data points in a batch have already been assigned. Let $\mathcal{L}(k)$ be the batch likelihood, under the a K-component Gaussian mixture model with isotropic variance σ^2 (as described in Section 5.3.2), after the $(N+1)$th data point is assigned to cluster k. Let C and B be, respectively, random variables indicating the cluster assignments in the batch and the batch indices. Then, the method presented in Section 5.3.3 is equivalent, up to a small error term, to maximizing*

$$\log \mathcal{L}(k) + \lambda I(C; B) ,$$

for some $\lambda \in \mathbb{R}$ that does not depend on the cluster assignments in the batch.

Proof. By Lemma 6, the method in (5.4) is equivalent to maximizing

$$\log \mathcal{L}(k) + \lambda H^k , \tag{5.14}$$

for $\lambda = \frac{1}{N+1}$. The mutual information $I(C; B)$ can be expressed in terms of entropy as

$$I(C; B) = H(C) - H(C|B) .$$

Moreover, we are making hard assignments so, given the cluster index, the distribution over cluster labels has all the probability on one cluster and has zero entropy. This means

$$I(C; B) = H(C) - H(C|B) = H(C) - 0 = H(C) .$$

Subbing this into (5.14), the result follows. □

5.3.5 Training Procedure

Our model comprises an encoder network f_{θ_1} and a set of cluster centroids $\theta_2 = \{\mu_1, \ldots, \mu_K\}$. To train on an input batch, the raw data is first encoded using f_{θ_1}, then the combinatorial assignment method of Section 5.3.3 is employed to produce cluster assignments, and these assignments are used to minimize (5.4) with respect to both θ_1 and θ_2. At inference time, combinatorial assignment is not used. Instead, each point is simply assigned to the cluster with the nearest centroid. This means that the resulting clustering model can assign points individually, and is not restricted to assigning cluster labels batchwise. The full methods for training and inference are described by the functions TrainOnBatch and PredictDataPoint, respectively, in Algorithm 3.

Algorithm 3 (Combinatorial assignment): During training, cluster labels are assigned batchwise, with partition support provided by a uniform prior across clusters. During inference, cluster labels are assigned pointwise, without any explicit partition support.

$f_{\theta_1} \leftarrow$ encoder network
$\theta_2 = \mu_1, \ldots, \mu_K \leftarrow$ centroids for each of the K clusters
$\sigma \leftarrow$ isotropic variance of all clusters
function ASSIGNBATCH(Z)
$counts \leftarrow K$-dimensional array, intially all 0s, $counts[k]$ will hold the number of times the kth cluster appears in the batch
$isAssigned \leftarrow N$-dimensional Boolean array, intially all False ▷ will indicate whether the each data point in the batch has been assigned to a cluster yet
$D \leftarrow N \times K$ matrix, where $D_{ij} = ||Z_i - \mu_j||^2$
for r=1,...,N **do** ▷ make each assignment iteratively via (5.7)
$\tilde{D} \leftarrow N \times K$ matrix, where $\tilde{D}_{ij} = D_{ij} + 2\sigma \log(counts[i] + 1)$
$(a, k) \leftarrow$ argmin $\{\tilde{D}_{ij} : \neg isAssigned[i]\}$
$counts[k] \leftarrow counts[k] + 1$
$isAssigned[a] \leftarrow True$
$loss \leftarrow loss + D_{a,k}$
end for
return $loss$
end function
function TRAINONBATCH(X)
$Z \leftarrow f_{\theta_1}(X)$, encodings for a batch of N data points
$loss \leftarrow AssignBatch(Z)$
take a gradient descent step on $loss$ with respect to θ_1, θ_2
end function
function PREDICTDATAPOINT(x)
$z \leftarrow f_\theta(x)$ encodings for a batch of N data points
$assignment = \text{argmin}_{j=1,\ldots,K} ||z - \mu_j||^2$
return assignment
end function

5.4 Experimental Evaluation

5.4.1 Datasets and Metrics

I report results on four popular image clustering datasets: CIFAR 10, CIFAR 100, Fashion-MNIST and STL. I use the standard clustering metrics of ACC, NMI and ARI, defined in Section 2.2 and also used in Chapter 4. In addition to these metrics, I report the variance in the size of predicted clusters (normalized across datasets by dividing by the mean cluster size). All four datasets have uniform numbers of ground-truth classes, so ideally this variance will be close to zero. A high variance is indicative of partition collapse.

5.4.2 Cluster sizes and clustering accuracy

Table 5.1 compares my proposed method of partition support with three existing methods: sum of squares minimization, denoted "SS", the Sinkhorn-Knopp algorithm for optimal transport, denoted "SK", and marginal entropy maximization, denoted "Ent" (see Section 5.2 for details). To make further explicit the phenomenon of partition collapse, I also include a model without any partition support. My proposed method significantly outperforms others across all datasets and metrics.

Observe that the unregularized model exhibits total partition collapse in all experiments, placing all points in the same cluster and consequently achieving a cluster performance no better than random guessing. The performance of SS is not much better. In many cases, it too suffers total partition collapse. By making a slight change to the author's original method, I could actually significantly improve its results (see appendix), but it was still unreliable and less accurate than my proposed method. The other two existing partition support methods do a reasonable job of avoiding partition collapse. However, entropy maximization occasionally also reaches the state with all points in the same cluster (this is consistent with previous literature, e.g., (W. Hu, Miyato, Tokui, Matsumoto, and Sugiyama, 2017)). The Sinkhorn-Knopp method is more reliable, but by far the most uniform cluster sizes are produced by my proposed method. Recall that I do not employ the assignment algorithm at inference time, instead each point is simply assigned to the cluster with the nearest centroid. This shows that the cluster centroids from my method are well-distributed around the data manifold, each capturing a sizeable subset of the data even when the explicit support is removed. Together, these figures show that

1. some form of partition support is necessary to learn anything meaningful,

2. my proposed method of combinatorial assignment is better at avoiding partition collapse than previous methods, and

3. this leads to my proposed model producing a better clustering performance.

Note that these experiments are not designed to hunt after state-of-the-art scores, but rather to demonstrate the effect on performance of this one component of an online clustering model: the partition support method, which is also shown to be an essential component. I do not use data augmentation, despite its well-documented improvement on image clustering (X. Guo, E. Zhu, X. Liu, and J. Yin, 2018), do not perform hyperparameter tuning, and use a relatively simple architecture for all datasets. This is the same for all methods being compared. The network consists of two convolutional layers with filter sizes 6 and 16, and ReLU activations, followed by a fully connected layer to a latent space of dimension 128. Training uses Adam, with default parameters of learning rate=1e-3, $\beta_1 = 0.9$, $\beta_2 = 0.99$. Batch size is 256. We follow previous works in setting K, the number of clusters, to the number of ground truth classes. This is the same for all methods being compared. The network consists of two convolutional layers with filter sizes 6 and 16, and ReLU activations, followed by a fully connected layer to a latent space of dimension 128. Training uses Adam, with learning rate 1e-3, $\beta_1 = 0.9$, $\beta_2 = 0.99$, and batch size 256. We follow previous works in setting K, the number of clusters, to the number of ground truth classes, as in methods from Chapter 3 and 4.

Table 5.1: Effect, on cluster size and clustering performance, of my proposed method compared to two existing methods of preventing partition collapse. "CA" refers to my proposed method of combinatorial assignment, "SK" refers to Sinkhorn-Knopp regularization, as proposed by (Asano, Rupprecht, and Vedaldi, 2019), "Ent" refers to entropy maximization, as used by (W. Hu, Miyato, Tokui, Matsumoto, and Sugiyama, 2017) and others, "SS" is the sum of squares minimization proposed in (H. Zhong, C. Chen, Z. Jin, and Hua, 2020), and "No-Reg" is the model without any partition support component. Along with standard cluster metrics, I report the variance in cluster size (denoted "CVar") to indicate the extent of partition collapse. All figures are the mean of 5 runs, with std dev in parentheses. Best results are in bold.

		CA	SK	Ent	SS	No Reg
Cifar10	Acc	**22.7 (2.07)**	16.7 (0.36)	18.6 (1.36)	11.8 (1.72)	10.0 (0.00)
	NMI	**10.1 (1.68)**	3.8 (0.68)	8.7 (2.63)	1.1 (1.15)	0.0 (0.00)
	ARI	**5.8 (0.93)**	2.7 (0.60)	5.7 (1.76)	0.3 (0.38)	0.0 (0.00)
	CVar	**425 (136)**	8931 (3634)	16240 (2243)	43542 (11424)	54000 (0)
Cifar100	Acc	**6.4 (0.22)**	2.6 (0.21)	2.4 (0.17)	1.2 (0.22)	1.0 (0.00)
	NMI	**13.2 (0.37)**	5.3 (0.41)	6.3 (0.73)	0.6 (1.05)	0.0 (0.00)
	ARI	**1.7 (0.14)**	0.3 (0.04)	0.5 (0.10)	0.0 (0.04)	0.0 (0.00)
	CVar	**1280 (156)**	13451 (4430)	25156 (3283)	55405 (3743)	59400 (0)
FashionMNIST	Acc	**54.5 (6.96)**	25.1 (2.80)	25.5 (5.51)	10.0 (0.04)	10.0 (0.00)
	NMI	**53.2 (4.23)**	17.7 (2.08)	20.9 (10.12)	0.0 (0.04)	0.0 (0.00)
	ARI	**39.1 (6.29)**	9.2 (1.27)	10.6 (5.37)	0.0 (0.00)	0.0 (0.00)
	CVar	**386 (51)**	7087 (3530)	14559 (3203)	53950 (98)	54000 (0)
STL	Acc	**23.5 (1.42)**	15.2 (1.03)	10.9 (1.76)	10.1 (0.20)	10.0 (0.00)
	NMI	**13.7 (1.33)**	2.8 (0.53)	1.3 (2.56)	0.0 (0.08)	0.0 (0.00)
	ARI	**7.1 (0.70)**	1.1 (0.37)	0.2 (0.32)	0.0 (0.00)	0.0 (0.00)
	CVar	**217 (21)**	2319 (466)	11320 (751)	11317 (765)	11700 (0)

5.4.3 A Closer Look at Hard vs. Soft Assignments

A key element of my proposed method is that I regulate the hard cluster assignments, whereas previous methods regulate soft cluster assignments. This is a fundamentally different objective, and it is possible for a batch of assignments to be close to partition collapse with respect to one but not to the other. For example, assume there are only three clusters and a batch size of four, and let $h, s : \mathbb{R}^{4\times 3} \to \mathbb{R}^3$ be the functions that compute the marginal hard and soft cluster distributions for a given matrix of batch assignment probabilities. Then, consider the following two such matrices

$$D_1 = \begin{bmatrix} .98 & .01 & .01 \\ .98 & .01 & .01 \\ .49 & .50 & .01 \\ .49 & .01 & .50 \end{bmatrix} \; D_2 = \begin{bmatrix} .34 & .33 & .33 \\ .34 & .33 & .33 \\ .34 & .33 & .33 \\ .34 & .33 & .33 \end{bmatrix},$$

which have marginal hard and soft assignments, and resulting marginal hard and soft entropy as follows:

$$s(D_1) = \big[.74, .13, .13\big] \qquad H(h(D_1)) = 1.10 \tag{5.15}$$

$$h(D_1) = \big[.5, .25, .25\big] \qquad H(s(D_1)) = 1.50 \tag{5.16}$$

$$s(D_2) = \big[.34., .33, .33\big] \qquad H(s(D_2)) = 1.58 \tag{5.17}$$

$$h(D_2) = \big[1, 0, 0\big] \qquad H(h(D_2)) = 0\,. \tag{5.18}$$

The matrix D_1 has a lower soft entropy than D_2 but a much higher hard entropy. Indeed, despite having nearly maximum soft entropy, D_2 has zero hard entropy, suggestive of partition collapse if this was repeated across the dataset.

The same discrepancy between the uniformity of hard and soft assignments is shown in Figure 5.1. This figure uses an idealized case with five clusters and batch size of four, and shows how different batch assignments could have different degrees of soft and hard uniformity. The top pane shows assignments from a model that is totally collapsed, where almost all the probability mass is consistently placed on the same cluster, meaning both the marginal soft assigments and the marginal hard assignments are very far from uniform. The second pane shows assignments of the sort that could be produced by soft assignment regularization. The marginal soft assignments are close to uniform, but the argmax remains the same for every data point, meaning every data point is still assigned to the same cluster and so the model is still collapsed. The third pane shows assignments from a model with hard-assignment regularization, of the sort applied by my proposed method from Section 5.4.2. 5.3.2. Here, the argmax varies across the batch, as desired, and so this model is not collapsed. Note that there is less soft-assignment uniformity here than in the second pane, which shows that, not only are uniform soft assignments insufficient for producing uniform hard assignments, they are also unnecessary. Of course, there is some correlation between both, uniformity of soft assignments will tend to coincide with uniformity in hard assignments, but it is possible that the two become quite disconnected, with one very high and the other very low. This supports my claim that measuring the uniformity in soft assignments is not a good measure of collapse, and that regularization to encourage uniformity in soft assignments is not a good method of preventing collapse.

The next question is whether this possibility obtains in practice, whether methods to encourage soft assignment uniformity lead to assignments like that of matrix D_2 in (5.15) or the second pane in Figure 5.1. To investigate this, I examine the cluster assignments produced by the methods from Section 5.4.2, and measure the variance of cluster sizes and entropy of both hard and soft marginal assignment probabilities marginalized across the dataset. A collapsed model would indicate high variance in cluster sizes, and low entropy of marginal assignment probabilities; both are measures of collapse. To measure the variance for soft assignments, I simply take, for each cluster, the sum of all partial assignments to that cluster as the total 'soft cluster size' in the batch, and then compute the variance of these sizes across all clusters. The results are shown in Table 5.2.

We are interested in whether these measures of collapse can be significantly different for the hard and soft assignments, and the results show that indeed they can. The difference is most striking for SS. There, the soft entropy is often close to the maximum value (equal to the logarithm of the number of clusters), but the hard entropy is consistently close to zero.

Figure 5.1: Effects of different regularization on the assignments for an idealized batch of size 4, with 5 clusters. Top: with no regularization the model collapses, it places most of the probability mass on the same cluster for each data point. Middle: soft regularization forces the soft marginal assignments to be close to uniform, but still the argmax is the same for every data point, so all points are assigned to the same cluster and the model is also collapsed. Bottom: regularizing the hard assignments causes the argmax to change. It allows the marginal soft assignment probabilities to deviate from uniform, and instead encourages uniformity in the number assigned to each cluster. This model avoids collapse. To make the diagram more readable, marginal assignments are not to scale with each batch assignment, as denoted by (rel.).

Table 5.2: Comparison of the variance and entropy for both hard and soft marginal assignments. Previous methods regularize the soft assignments, but this does not transfer well to the hard assignments. Ours (CA) is the only method that produces low variance and high entropy for both hard and soft assignments.

	Cifar10				Cifar100				FashionMNIST				STL			
	HVar	SVar	HEnt	SEnt	HVar	SVar	HEnt	SEnt	HVar	SVar	HEnt	SEnt	HVar	SVar	HEnt	SEnt
CA	**425**	407	**3.3**	3.3	**1280**	190	**5.4**	6.5	**386**	368	**3.3**	3.3	**217**	194	**3.2**	3.2
SK	8931	**0.0**	2.5	**3.3**	13451	**0.0**	3.5	**6.6**	7087	**0.0**	2.7	**3.3**	2319	**0.0**	2.3	**3.3**
Ent	16240	13141	1.8	2.1	25156	4118	1.6	4.6	14559	10052	1.9	2.3	11320	3380	0.1	2.0
SS	43542	339	0.9	2.6	55405	121	0.2	6.5	53950	376	0.0	3.1	11317	524	0.1	3.1

This shows that this form of regularization produces batch assignment probabilities similar to matrix D_2 from (5.15) or the second pane from Figure 5.1, where the probabilities for each data point are squeezed to be close to one another, without much change in the order of highest to lowest, and in particular the argmax.

A similar discrepancy is found in SK, which produces near-perfect uniformity in the soft assignments, with a variance of zero and the maximum possible entropy (up to rounding) on each dataset. This is because it is (a close approximation to) a hard constraint problem. However, SK's hard assignments still show significant variability, and markedly lower entropy than the soft assignments. This suggests that applying the SK algorithm also produces batch assignment probabilities somewhat similar to D_2 above. My proposed method, on the other hand, explicitly forces the argmax to be more evenly distributed during training and, as Table 5.2 shows, this transfers to the testing setting as well. (Recall that, during testing, the model simply assigns each point to the cluster with the nearest centroid.) The hard entropy from my method is only slightly lower than the soft entropy, and is consistently higher than that of the other three methods.

Thus, the mean of the soft assignments across a batch does not contain sufficient information to determine if the clustering model is learning a meaningful partition, and regularizing this quantity is not an optimal way to prevent partition collapse. Rather, the argmax for each point is also important, and this information is lost if we look only at the mean across a batch.

This finding does not contradict that of Caron, Misra, Mairal, Goyal, Bojanowski, and Joulin (2020), who report better results using soft assignments as training labels. I show that *regularizing*, i.e., encouraging uniformity of, hard assignments facillitates better training, whereas Caron, Misra, Mairal, Goyal, Bojanowski, and Joulin (2020), on the other hand, have found better results from *training* on soft labels. I also explored training the Sinkhorn-Knopp regularized model using soft assignments as training targets, but the results were slightly worse than using hard targets. My own method performs hard batch assignments by definition, so there is no soft training setting to consider.

Relation to Discrete vs Continuous Representations From the wider perspective of the goal of this thesis, the difference between collapsed hard assignments and collapsed soft assignments is particularly interesting. The hard assignments, which are cluster labels, are a form of discrete representation, as discussed in Section 2.2. The soft assignments can also be considered a form of representation, this time a continuous representation, where the feature

specified by each coordinate is the relative affinity to the corresponding cluster. When we use the soft assignments to choose which cluster a point belongs to, that is, when we make cluster assignments in the usual way, we are converting from the continuous representation to the discrete representation. The findings of this chapter suggest that the conversion is a significant one, that some important properties of the representation may change when this line is crossed. Specifically, we saw how marginal uniformity across a batch can be completely lost. Existing methods for online clustering attempt to avoid collapse by ensuring uniformity before the transition to discrete has been made. While still on the continuous side, the optimization is simpler computationally. For 'Ent' and 'SS', gradient-based optimization is used, and for 'SK', the variant of the Sinkhorn-Knopp algorithm used specifically exploits the relaxation of the assignment problem to the continuous polytope. The hope with these methods is to ensure some desired property of the continuous representation (in this case, a non-collapsed partition), where the optimization is easier, and then maintain this property after converting to a discrete representation, where it is actually desired. However, the results here show that these attempts to, as it were, have one's cake and eat it, are overly ambitious. The desired property, marginal uniformity across the batch, can be, and often is, largely lost after converting to the discrete. It seems, with respect to presently existing methods, that we must do the harder work of optimizing the hard assignments (discrete representation) directly, if that is what we want optimized.

5.5 Summary

In this chapter, I presented a method to prevent partition collapse in online deep clustering. I framed probabilistically the problem of deciding which clusters to assign a batch of data points to, given the cluster centroids and features of the data points, and I used this framing to derive a concise optimization objective for making hard cluster assignments to a batch of data. I then described an algorithm to approximately solve this optimization problem and demonstrated empirically on four datasets that this method outperforms existing methods, both in better preventing partition collapse and in leading to better clustering performance. Finally, I analyzed how the cluster assignment distribution is affected by my proposed partition support method and previous comparable methods. The analysis suggests that regularizing the soft assignments, as is done by existing works, is not sufficient to redress the problem of partition collapse, and that a better approach is to regularize the hard assignments, as is done by my proposed method.

This is the last of the chapters of this thesis to discuss a deep clustering model. Chapter 6 concerns deep learning only, not clustering, and Chapter 7 concerns clustering only, not deep learning. The work in this chapter differed from the previous two in that it works in the online setting, which presents extra challenges over the offline setting, but also has some practical advantages, such as being faster and more widely applicable. Additionally, to return to the motivation outlined in Section 1.1, online clustering is more true to the goal of learning discrete representations in a similar way to how humans learn them. An intelligent machine, whatever techniques it is built out of, will have to be able to form some partial understanding of the data it encounters before it has seen every single data point, and so a move towards online deep clustering is a step in the right direction.

Chapter 6

Logical Annotation using Deep Learning

Abstract

This chapter concerns the use of deep learning to extract knowledge graph annotations from videos. Nearly all existing techniques for automated video annotation (or captioning) describe videos using natural language sentences. However, this has several shortcomings: (i) it is very hard to then further use the generated natural language annotations in automated data processing, (ii) generating natural language annotations requires solving the hard subtask of generating semantically precise and syntactically correct natural language sentences, which is actually unrelated to the task of video annotation, (iii) it is difficult to quantitatively measure performance, as standard metrics (e.g., accuracy and F1-score) are inapplicable, and (iv) annotations are language-specific. Hence, in this chapter, I propose producing a description in the form of a knowledge graph of the contents of a given video. Since no datasets exist for this task, I also include a method to automatically generate them, starting from datasets where videos are annotated with natural language. I then describe a deep-learning model for knowledge graph extraction from videos, and report results on MSVD* and MSR-VTT*, two datasets obtained from MSVD and MSR-VTT using my method. Code for the annotation method and the creation of datasets is available at https://github.com/Lou1sM/video_annotation.

6.1 Introduction

Recent progress in deep learning shows exciting potential for visual understanding. Promising results have been produced in the tasks of video annotation (or captioning) using natural language (J. Zhang and Y. Peng, 2019; L. Gao, Z. Guo, H. Zhang, X. Xu, and H. T. Shen, 2017; L. Zhou, Y. Zhou, Corso, Socher, and Xiong, 2018) and video question answering (Z. Zhao, Q. Yang, D. Cai, X. He, and Zhuang, 2017; K. Zeng, T. Chen, Chuang, Liao, Niebles, and M. Sun, 2017; Lei, L. Yu, Bansal, and T. L. Berg, 2018).

However, annotations written in natural language (NL) are not ideal, for several reasons. Firstly, it is difficult to use NL sentences in subsequent automated data processing tasks. One example of such data processing is database search. If we have a database of videos annotated with NL and want to search for all videos depicting a man, we cannot simply perform string matching on the annotations. This would, e.g., produce a false positive for the annotation "sailors man the ship", and a false negative for "a firefighter puts on his coat". Another example of automated data processing is logical inference. For example, if a NL annotation model correctly identifies that there is a man throwing a ball in an image, it cannot then conclude that there is a male throwing a ball in the image. That is, a logical inference rule of the form $\forall x\ man(x) \Rightarrow male(x)$ cannot easily be applied. Automated inference in NL is known to be a difficult problem (Bowman, Angeli, Potts, and Manning, 2015; Belinkov, Poliak, Shieber, Van Durme, and Rush, 2019). Again, string matching would not provide a solution, as the above examples attest. A second drawback of NL annotations is that they require more work than just understanding the contents to be annotated. Traditionally, text generation has been divided into a semantic stage of determining what to say, and then a realization stage of determining how to say it (K. R. McKeown, 1985; Levelt, 1993). The former corresponds to understanding the contents being annotated and is the task that we are interested in, but deep learning NL annotation models must, at the same time, learn to perform the latter, which means learning the language's complex meaning and grammatical structure. This is irrelevant to the goal of annotation and so needlessly makes the task more difficult. Thirdly, it is not easy to interpret and quantitatively measure the performance of NL annotations. The agreement between a ground-truth and the predicted sentence cannot be measured by standard metrics, such as accuracy and F1-score. Instead, they require specially designed ones such as, BLEU (Papineni, Roukos, Ward, and W.-J. Zhu, 2002), METEOR (Banerjee and A. Lavie, 2005), and LEPOR (Han, Wong, and Chao, 2012). Recognizing the imperfection of each of these, results typically report scores on multiple metrics, none of which have a simple and intuitive interpretation. This makes it difficult to evaluate how well a model is performing. A fourth problem is that NL annotations are specific to a single language. A model trained to produce annotations in English cannot produce annotations in German, and it is non-trivial to translate the English annotations into German ones. This is a particular drawback for low-resource languages.

Replacing NL annotations with knowledge graphs avoids all the above problems. A knowledge graph (KG) specifies the *individuals* (i.e., the objects) present in the video and the facts that hold true of these individuals. A *fact* expresses a relation between two individuals (e.g., $fold(person, paper)$) or an attribute of an individual (e.g., $white(paper)$). A visual representation of a KG is given in Figure 1, where nodes representing individuals are depicted in red, nodes representing attributes are depicted in blue, and nodes representing

relations are depicted in green. A KG can be represented visually or as a set of facts; see Figure 1 (7) and (5), respectively. KGs are machine-readable, so enable automated data processing, such as searching for all videos depicting a man or applying the inference rule $\forall x\ man(x) \Rightarrow male(x)$. They are determined only by the semantics of the video contents, and do not involve a semantically and syntactically complex natural language structure. They can be evaluated using standard machine-learning metrics, such as accuracy and F1-score, which makes it easy to interpret performance. For each data point, the ground truth and the prediction are sets of facts; so, e.g., the F1-score can be computed by counting the facts that appear in both as the true positives, those that appear in the former only as false negatives, etc. Finally, each fact, and hence each KG, can be trivially translated by translating one component at a time. For example, we can translate $white(paper)$ to German by translating $white$ to $weiss$ and $paper$ to $Papier$, giving $weiss(Papier)$. (As discussed in Sections 6.2 and 6.5, the components of each fact denote particular senses of words, so word-sense disambiguation is not an issue.) For these reasons, I propose to produce annotations using KGs rather than NL.

These advantages have begun to be recognized in neighboring areas. The entire task of open information extraction is motivated by use cases that can only be met by a structured representation of information, and not by the same information expressed in natural language. In computer vision, researchers have started to argue for the importance of basic visual reasoning (J. Johnson, Hariharan, Maaten, Fei-Fei, Zitnick, and R. B. Girshick, 2016) and the ability to leverage external knowledge (P. Wang, Q. Wu, C. Shen, A. R. Dick, and Hengel, 2016). A number of works aim to extract a structured description from an input image to reason about the identified objects (P. Wang, Q. Wu, C. Shen, A. Dick, and Van Den Henge, 2017). In the video understanding field, there have been only preliminary attempts (Vasile and Lukasiewicz, 2018), though, recently, (Curtis, Awad, Rajput, and Soboroff, 2020) has sketched some planned future work to manually create a dataset of videos annotated with facts.

In this Chapter, I describe a method for automatically generating datasets of videos and corresponding KGs. The method uses a rule-based semantic parser, based on the Stanford parser (Qi, Dozat, Y. Zhang, and Manning, 2018). The generated datasets are sufficient for the development of KG extraction models. Additionally, I introduce one such model, trained and tested on two generated datasets, which achieves a superior performance to (Vasile and Lukasiewicz, 2018), the only existing work to attempt the same task (F1-score of 14.0 vs. 6.1). I also compare to an artificial baseline, conduct ablation experiments that demonstrate the efficacy of each of the model's components, and present qualitative results. This model enjoys all the advantages described above: it facilitates automated data processing, it does not require modelling complex NL syntax and semantics, its performance is easy to evaluate, and the resulting annotations are easy to translate.

Example 10. *Figure 6.1 shows an example where the input is a video of someone folding paper. Given this video, my model detects the contained individuals, namely, person, man, paper, and origami. Then, it predicts which facts are true of these individuals: demonstrate(person), fold(person, paper), white(paper), and fold(man, origami). As an example of automated data processing, it may then exploit an ontology to predict more facts that are true for the video, e.g., the rule* $\forall x, y\ (fold(x, y) \Rightarrow change(x, y))$ *to infer the fact*

change(person, paper). The corresponding KG is shown in (7). ◁

The main contributions of this chapter are summarized briefly as follows.

- I propose the task of extracting KGs from videos. I provide arguments for the superiority of this approach to NL annotations, and support these arguments empirically.
- I describe and make available a method for automatically generating datasets of videos and corresponding KGs, and describe its application to generate two datasets. The corresponding code is available at https://github.com/Lou1sM/video_annotation.
- I introduce a model for KG extraction, trained and tested on the above two datasets. This outperforms the existing work to attempt the same task. I also compare to an artificial baseline and conduct ablation studies on the main components of the model.

The rest of this chapter is organized as follows. In Section 6.2, I present my method for automatically generating datasets. In Section 6.3, I describe my KG extraction model, and then report experimental results in Section 6.4. Section 6.5 provides a further discussion, and Section 6.6 summarizes my main contributions and gives an outlook on future work.

(1) MSR-VTT ground truth:
a person folds a piece of white paper

(2) Predicted caption from Oli2019:
a person is folding a piece of paper

(3) MSR-VTT* ground truth:
fold(*person, piece*)

(4) Predicted individuals:
person, man, paper, origami

(5) Predicted facts:
demonstrate(*person*), *fold*(*person, paper*)

(6) Some inferred facts:
change(*person, paper*), *fold*(*male, origami*)
white(*paper*), *fold*(*man, origami*)

(7) Visual representation of the predicted knowledge graph:

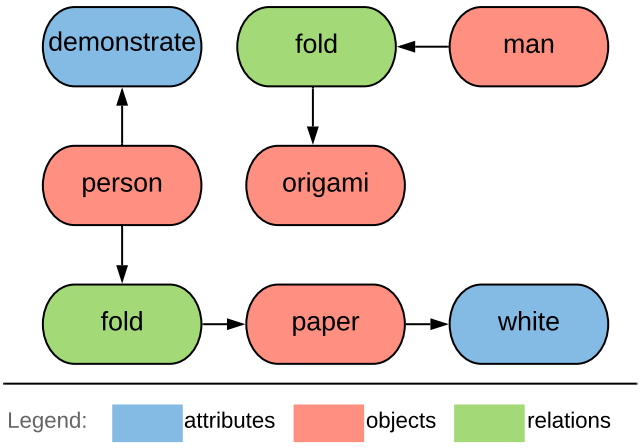

Figure 6.1: The first frame from a video in MSR-VTT*, with (1) the ground-truth natural language annotation in MSR-VTT, (2) the natural language annotation produced by (Olivastri, G. Singh, and Cuzzolin, 2019) (referred to as 'Oli2019'), (3) the ground-truth set of facts in MSR-VTT*, (4) the individuals predicted by my system, (5) the facts predicted by my system, and (6) some further inferences that can be made on top of these facts.

6.2 Dataset Generation

In the absence of appropriate datasets, I devise an automated method to generate them. There are several NL-annotated datasets, which I refer to as video-captioning datasets. Our method begins with these datasets and converts the NL annotations into sets of facts (which are equivalent to KGs). I now describe this method with reference to its application to two well-known video-captioning datasets: MSVD (D. L. Chen and Dolan, 2011) and MSR-VTT (J. Xu, Mei, T. Yao, and Rui, 2016). I denote the generated datasets MSVD* and MSR-VTT*, respectively, and Section 6.4 reports the results of our proposed model on these datasets. The dataset generation code is freely available at https://github.com/Lou1sM/video_annotation. To create the datasets MSVD* and MSR-VTT*, as described in Section 6.2, I use a semantic parser to convert the natural language annotations into logical annotations. There are four steps to this conversion: parsing individual sentences, linking entities to the ontology (in our case, WordNet (G. A. Miller, Beckwith, Fellbaum, Gross, and K. J. Miller, 1990)), merging the multiple annotations of each video and filtering out uncommon entities. They are described, respectively, in the following three sections.

6.2.1 Parsing Natural Language Sentences

In both MSVD and MSR-VTT, the training examples are composed of a video and a NL sentence describing the contents of that video. I parse these sentences using a rule-based parser based on the Stanford NLP syntactic parser (Qi, Dozat, Y. Zhang, and Manning, 2018). This produces a dependency parse of the sentence, which identifies the part of speech for each word, and the syntactic relations that hold between them, and represents them with a grammatical dependency tree. In a grammatical dependency tree, a single word is identified as the root, and all other words are marked depending on their syntactic relation to their parent in the tree. The information produced by the Stanford Parser, for our purposes, can be thought of as marking each word in the input sentence with (i) a part of speech, (ii) another word in the sentence designated as the parent (unless this word is the root), and (iii) a grammatical dependency relation to the parent. An example of such a relation is 'nsubj', which means this word occupies the subject position of its parent, its parent being a verb. Further discussion, including a list of all possible such dependencies, can be found in (Droganova and Zeman, 2019; Rademaker and Tyers, 2019) and online at https://universaldependencies.org/introduction.html. An example dependency parse is shown in Figure 6.2.

I then apply a sequence of rules to extract the semantic information from the sentence, and use it to form a KG, which is composed of a set of facts. These desired facts each contain a predicate and the corresponding arguments: $\langle subject, predicate, object \rangle$ triples in the case of binary predicates, and $\langle subject, predicate \rangle$ pairs in the case of unary predicates (see Figs. 6.4 and 6.5). These are extracted from the dependency parse as per Algorithm 4. From this syntactic parse, atoms are formed by Algorithm 5.

Algorithm 4 ExtractRootAtom

Inputs: grammatical dependence tree of a natural language sentence
Output: a single atom if one can be found, else null
$s \leftarrow$ the input natural language sentence
root $\leftarrow$ the word designated as root
if there is a word w in s with dependency marking 'nsubj' to root **then**
    $subj \leftarrow w$
else
    return null
end if
if root is a verb **then**
    if there is a word w' in s with dependency marking 'obj' to root **then**
        return $root(subj, w')$
    else
        return $root(subj)$
    end if
end if
if root is an adjective **then**
    if there is a word c in s with dependency marking 'cop' to root **then**
        return $root(subj)$
    else
        return null
    end if
end if
if root is a noun **then**
    if there is a word c in s with dependency marking 'cop' to root and a word p with dependency marking 'case' to root and with part of speech 'ADP' **then**
        return $p(arg1, root)$
    end if
end if

Algorithm 5 ExtractAllAtoms

Inputs: a natural language sentence
Output: a list of atoms
$s \leftarrow$ input natural language sentence
extractedAtoms $\leftarrow$ []
$root \leftarrow$ the word in sentence with dependency marking 'root'
if there is a word w with dependency marking 'obj' to root **then**
    append ExtractRootAtom(s, $root$, w) to extractedAtoms
else
    return null
end if
for each word $conjWord$ in s with dependency relation 'cc' **do**
    $subClauseRoot \leftarrow$ parent of $conjWord$
    $subClauseSubj \leftarrow$ parent of $subClauseRoot$
    $subClauseAtom \leftarrow$ ExtractRootAtom(s,
                $subClauseRoot$,$subClauseSubj$)
    append $subClauseAtom$ to extractedAtoms
end for
for each word $modWord$ in s with dependency relation 'amod' to to some other word $targetWord$ **do**
    append $modWord(targetWord)$ to extractedAtoms
end for
for each word $compWord$ in sentence with dependency relation 'compound' to some other word $targetWord$ **do**
    if $compWord$ appears immediately before $targetWord$ in s **then**
        replace all instances of $targetWord$ in
        extractedAtoms with the concatenation
        $compWord \cdot targetWord$
    else if $compWord$ appears immediately after $targetWord$ in s **then**
        replace all instances of $targetWord$ in
        extractedAtoms with the concatenation
        $targetWord \cdot compWord$
    end if
end for
return extractedAtoms

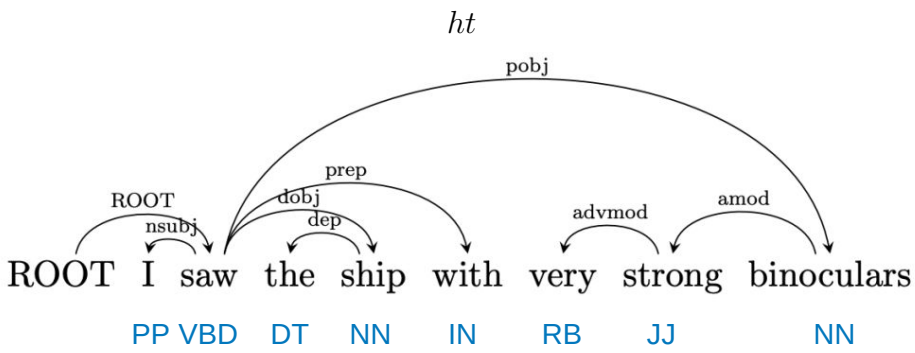

Figure 6.2: An example dependency parse of the sentence 'I saw the ship with very strong binoculars'. The root is 'saw', and all other words in the tree are marked for their syntactic relation to their parent. Parts of speech are shown in blue, using the Penn Treebank tagset, which is the tagset used by the Stanford parser.

6.2.2 Linking to an Ontology

The next step is to link all predicates, and individuals to entities in an ontology. The ontology that I use here is WordNet (G. A. Miller, Beckwith, Fellbaum, Gross, and K. J. Miller, 1990). Specifically, I need to identify, for each predicate and object in the logical annotation, the corresponding entity from the ontology. The linking method is context-sensitive, i.e., for a given word, it finds the most suitable WordNet synset using the surrounding words in the NL annotation. This amounts to word-sense disambiguation. In WordNet, words are broken up, by sense, into different synsets. E.g., the word 'bank' has one synset corresponding to the financial institution and another corresponding to the bank of a river. To identify the correct synset for each word w in our logical annotations, I compute the doc2vec (Le and Mikolov, 2014) representation of the original natural language sentence from which w was taken, and the doc2vec representation of the definition of each WordNet synset containing the word w, and I pick the synset which has the most similar vector representation.

Linking to an ontology formalizes the vocabulary and allows the application of inference rules to augment the information produced directly by our annotation system. It also means that the components of our KGs correspond to particular senses of words. For example, "bank" referring to the financial institution, and "bank" referring to the side of a river would be linked to different WordNet synsets, and so would appear as different items in a KG.

Formally,

$$syn = \underset{\{syn' | w \in syn'\}}{\text{argmax}} \; doc2vec(def(syn'))^T doc2vec(s),$$

where $def(syn')$ denotes the definition of synset syn', and s is the original natural language annotation sentence from which w was taken.

A logical annotation is a set of atoms. I merge all the logical annotations for each video by simply taking the union of the constituent atoms.

Table 6.1: Numbers of distinct predicates and individuals that were included in our final datasets. Predicates and individuals were included if they appeared at least 50 times. The final column shows the number of data points that remained with non-empty annotations even after infrequently occurring words were excluded.

	Num Training Examples	Num Individuals	Num Attributes	Num Relations	Num Facts	Num Non-empty Training Examples
MSVD*	1970	122	48	69	117	1800
MSR-VTT*	10000	372	235	113	348	9802

6.2.3 Filtering the Vocabulary

Words that appear only a small number of times across the dataset would be difficult to reach a good performance on. Therefore, I only consider ontology elements (i.e., objects and predicates) that appear a sufficient number of times in our dataset. Specifically, I remove all elements that appear in fewer than 50 data points, after the merging of captions in the previous step. Additionally, I exclude the verbs "take", "do", "be", and "have". These are semantically weak verbs, which normally function as copulas or other syntactic operators, and do not convey relevant information about the video. For example, the parse of the sentence "a woman is standing at the top of the stairs" should not include the atom $be(woman)$.

6.2.4 Adding Negatives

The method described so far produces only non-negated facts, from here on referred to as T. This means a model could learn to simply predict every potential fact to be true. To prevent this, I use the local closed-world assumption (Dong, Gabrilovich, Heitz, Horn, Lao, K. Murphy, Strohmann, S. Sun, and W. Zhang, 2014) to create a set of negated facts, from here on referred to as F, which the model must also learn to predict. For each fact in the description, I obtain a corrupted version by replacing the predicate with a different predicate from the vocabulary. For example, the fact $fold(person, section)$, which appears in Figure 6.1, could be corrupted to $\neg throw(person, section)$. This negated fact is then added to F.

6.2.5 Merging Logical Annotations

As a final step, I merge all KGs for each video by taking the union of the facts of their parses. In MSVD and MSR-VTT, each video appears with multiple captions, and each produces a separate training example. In MSVD* and MSR-VTT*, each video appears in only one training example. After exclusion and merging, the numbers of distinct individuals, distinct predicates, and data points with at least one fact or negated fact are as reported in Table 6.1.

6.3 Proposed Model

In this section, I describe a deep learning model (illustrated in Figure 6.3) for extracting KGs from videos. I first describe a general method for extracting a KG from an arbitrary input, and then detail how this method applies to the case where the inputs are videos.

6.3.1 General Knowledge Graph Extraction Model

Given a set of possible inputs X and a knowledge base KB (of ontological domain knowledge about X), let the vocabulary consist of the set P of all predicates that appear in KB and the set A of all individuals that appear in KB. Then, the neural architecture for KG extraction consists of the following four components:

1. an encoder $f\colon X \to Z$,
2. a multi-classifier $g\colon Z \to (0,1)^{|A|}$,
3. a set of multilayer perceptrons (MLPs): $\{m_p \mid p \in P\}$,
4. a set of trainable individual vectors: $\{v_a \mid a \in A\}$,

where Z is the space of extracted feature vectors.

The data for training the above neural architecture consists of 4-tuples (x, c, T, F), where $x \in X$ is the input to be annotated, $c \subseteq A$ is the set of individuals that are present in the input, and T (resp., F) is a set of facts (resp., negated facts) containing these individuals.

Example 11. *Consider again the video in Figure 6.1. The training tuple associated with the video is*

$$(x, \{person, piece\}, \{fold(person, piece)\}, F)\ ,$$

where x represents the video itself, and F is a set of negated facts that does not contain ¬fold(person, piece). An example of a negated fact that belongs to F is ¬throw(person, piece). ◁

During training, I compute $f(x) = e \in Z$ and feed this feature vector to the multi-classifier to predict the individuals present, $\hat{c} = g(e)$. Then, I compute a binary cross-entropy loss (BCE) for each individual in the vocabulary, giving

$$\mathcal{L}_c = \textstyle\sum_{j=0}^{|A|} \mathcal{L}_{bin}(\hat{c}_j, c_j)\ ,$$

where $c_j \in \{0,1\}$ and $\hat{c}_j \in (0,1)$ are the ground truth and prediction as to whether the jth individual in the vocabulary is present, respectively, and $\mathcal{L}_{bin}$ represents BCE. We also produce predictions for the facts and negated facts in T and F by selecting the individual vectors corresponding to the individuals in the ground truth, and feeding these vectors (along with the video encoding) to the predicate MLPs corresponding to the predicates in the ground truth. That is, for each fact $t \in T$ with $t = p(a, b)$, we compute a prediction $\hat{t} = m_p(e, v_a, v_b)$, and similarly for negated facts. A naive brute-force method would be to run every predicate MLP on every individual vector. This would require $|P| \times |A|$ forward

passes for every video. Selecting only the present individuals reduces this to $|P| \times n$, where n is the number of individuals in the input video, which is typically less than three. In theory, the number of predicted individuals could be as large as the vocabulary of individuals, in which case there would be no speed up, but in practice it is always much lower, givining a significant speed-up. Formally, the worst case complexity is $O(k_1|P|.|A|^2 + k_2)$, where k_1 is the time for a forward and backward pass of each predicate MLP, and k_2 is the time for a forward and backward pass of the encoder and multiclassifer. In practice, however, the number of individuals predicted to be present is less than three, and it is never observed to be more than five. We can therefore treat it as a constant, giving asymptotic complexity $O(k_1|P| + k_2)$.

The vectors v_a and v_b are initialized to the corresponding word2vec word vectors (Le and Mikolov, 2014). For example, v_{man} is initialized to the word2vec vector for the word "man". These vectors are updated during training along with the network weights.

The training data contain many more negated facts than facts. To counteract this imbalance, I weigh the loss for facts and negated facts inversely to how many there are. The prediction loss is again calculated using BCE, this time applied to each fact (resp., negated fact). Formally,

$$\mathcal{L}_p = -\tfrac{1}{2|T|} \textstyle\sum_{t \in T} \log(\tilde{t}) - \tfrac{1}{2|F|} \sum_{f \in F} \log(1 - \tilde{f}) \,.$$

The total backpropagated loss is then a simple summation:

$$\mathcal{L} = \mathcal{L}_c + \mathcal{L}_p \,.$$

At inference time, given an input x, the model first computes the feature vector (step (i)), then thresholds the outputs of the multi-classifier to obtain the set I of predicted individuals (step (ii)):

$$\hat{I} = \{a \in A \mid g(f(x))_a > 0.5\} \,.$$

As the ground truth set of individuals is no longer available, the model instead selects the *predicted* individuals and passes them to the predicate-MLPs (step (iii)). This again avoids having to run all predicate-MLPs on all individuals. Specifically, to predict the unary relations that hold of these individuals, I run all predicate-MLPs in the vocabulary on all individual vectors corresponding to individuals in $\hat{I}$, and to predict the binary relations, I do the same on all pairs of individuals in $\hat{I}$. The total set $\hat{T}$ of predicted facts is the union of the two:

$$\begin{aligned}
U &= \{p(a) \mid p \in P, a \in \hat{I}, m_p(a) > 0.5\}, \\
B &= \{p(a,b) \mid p \in P, a, b \in \hat{I}, m_p(a,b) > 0.5\}, \\
\hat{T} &= U \cup B.
\end{aligned}$$

Now, as an example of automated information processing, I can apply the additional knowledge from KB to the predicted facts $\hat{T}$ to obtain an additional set of inferred facts T':

$$\begin{aligned}
U' &= \{p(a) \mid p \in P, a \in \hat{I}, KB \cup \hat{T} \models p(a)\}, \\
B' &= \{p(a,b) \mid p \in P, a, b \in \hat{I}, KB \cup \hat{T} \models p(a,b)\}, \\
T' &= U' \cup B'.
\end{aligned}$$

The annotation can then be updated with these inferences (step (iv)):

$$\hat{T} \leftarrow \hat{T} \cup T'.$$

Example 12. *Consider again the video in Figure 6.1, and the schema of the model in Figure 6.3. Following the steps in Figure 6.3, the KG for the video in Figure 6.1 is produced in four steps.*

1. *The input video x is mapped into a fixed-length feature vector e by the encoder f.*

2. *The encoding e is fed into a multi-classifier MLP g, which returns the probability that each individual in the vocabulary is present in x. The probabilities are then thresholded at 0.5, and to give the set*

$$I = \{person, man, paper, origami\}.$$

3. *Consider the individual $paper$. The encoding e, together with the vector v_{paper} is passed through all the MLPs, and a fact is returned only when the prediction of the MLP is greater than 0.5, such as $white(paper)$. This is then repeated for each predicted individual, and each ordered pair of predicted individuals, producing*

$$\begin{aligned}\hat{T} = \{&demonstrating(man), white(paper),\\ &fold(person, paper), fold(man, origami)\}.\end{aligned}$$

4. *All facts that can be inferred from KB and the predicted facts are used to augment the KG extracted from the video. Examples of such inferred facts are $change(person, paper)$ and $fold(male, origami)$.* ◁

6.3.2 Application to Video Extraction

The above method can be used to extract a KG from any unstructured data X. For example, X may contain sequences of word embeddings representing text, spectrograms representing audio, or image tensors representing still images. In each case, one only has to choose an encoder f, so that $f(x)$ is a fixed-length feature vector for each $x \in X$. I now describe how this method is applied when the input domain is videos.

Our encoder, f, is composed of a convolutional neural network, followed by a recurrent neural network. Given a video $x = x_1, x_2, \ldots, x_n$, I first process each video frame x_i using a pre-trained frozen copy of VGG19 (Simonyan and Zisserman, 2014) and take as ith frame encoding, ζ_i, the output of the 3rd last layer. (This is a standard approach taken by others in the field; e.g., (Y. Pan, T. Yao, H. Li, and Mei, 2017; Y. Pan, Mei, T. Yao, H. Li, and Rui, 2016; X. Zhang, K. Gao, Y. Zhang, D. Zhang, J. Li, and Tian, 2017).) The sequence of frame encodings $\zeta_1, \ldots, \zeta_n$ is then passed through a gated recurrent unit (GRU) (K. Cho, Merrienboer, Bahdanau, and Bengio, 2014) to produce a sequence $\overline{\zeta}_1, \ldots, \overline{\zeta}_n$. As a second stream, I also compute the feature vector from a frozen copy of the I3D network (Carreira and Zisserman, 2017). The final output of the encoder is the concatenation of this I3D

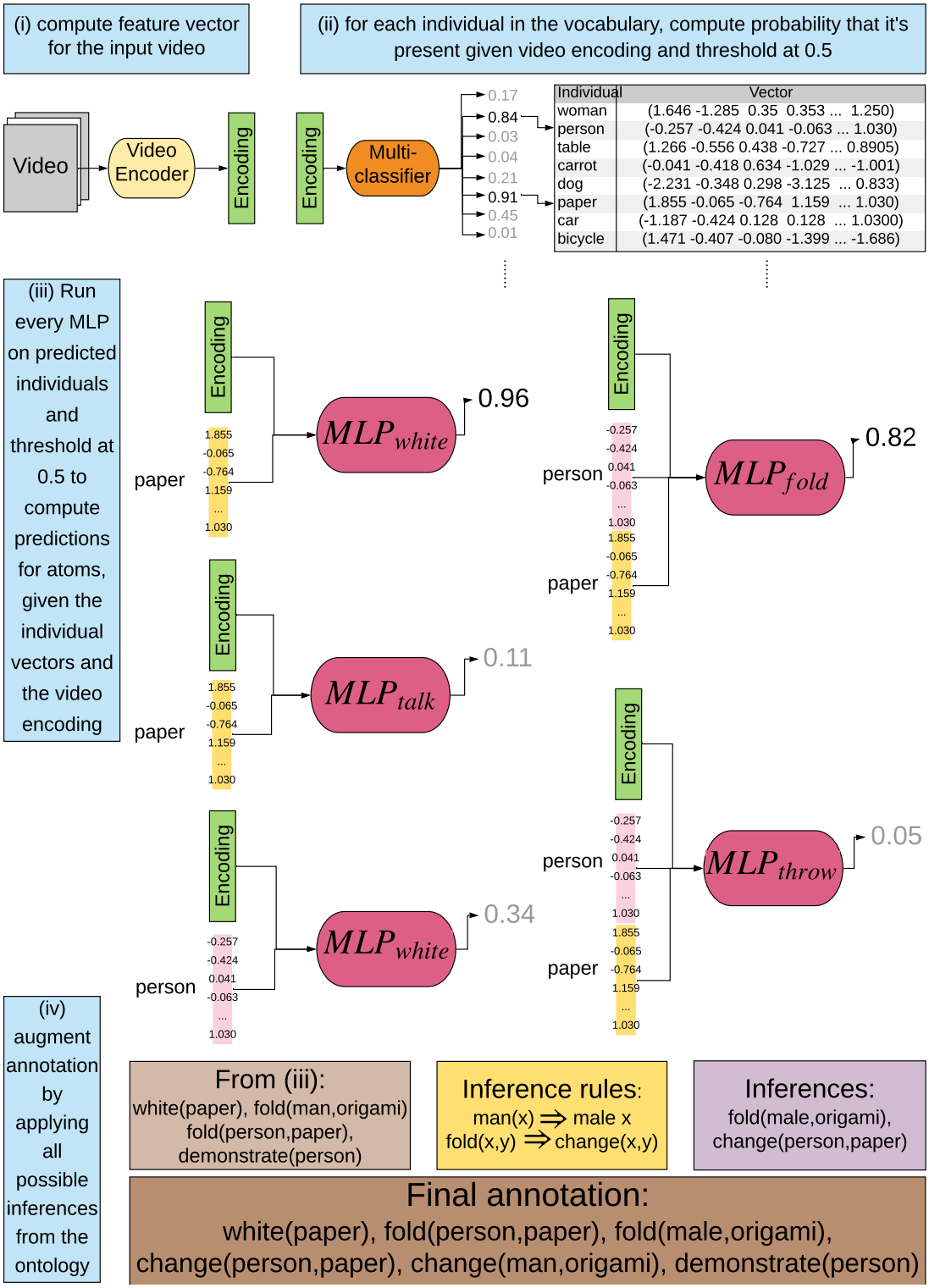

Figure 6.3: Depiction of how the neural architecture processes a video of a man folding a piece of paper (as in Figure 6.1). The four stages are: (i) encode the video, (ii) predict individuals, (iii) predict facts, and (iv) add facts inferred from KB.

feature vector and a weighted sum of the $\overline{\zeta}_i$'s, weighted by a learnable n-dimensional vector w. Using vector notation, with the VGG network applied along the first dimension, the encoder f is defined by:

$$f(x) = w.GRU(VGG(x)) \cdot I3D(x)\,. \tag{6.1}$$

The MLPs for each predicate have one hidden layer, as does the multi-classifier. The input size of the multi-classifier is the size of the video encoding, denoted $dim(f(x))$. The predicate-MLPs have input size $dim(f(x)) + D$ in the case of unary relations and $dim(f(x))+2D$ in the case of binary relations, where D is the size of the individuals' vectors (300 in this case). Using the encoder defined in (6.1), $dim(f(x)) = 4096 + 1024 = 5120$. I restrict attention to unary and binary predicates, but the framework can naturally be extended to include higher-order predicates.

6.4 Experimental Results

In this section, I provide quantitative experimental results for the two datasets MSVD* and MSR-VTT* and compare these results to existing baselines. I also report on ablation studies for our neural architecture and give some further qualitative results.

6.4.1 Quantitative Results

Tables 6.2 and 6.3 display our performance on the two generated datasets MSVD* and MSR-VTT*, respectively, reporting both overall accuracy and F1-score. For each training example, these datasets contain many more negated facts than facts (see Section 6.2 for details). This is the reason for such a large difference between accuracy and F1-score: the predictions are dominated by true negatives, which increase the overall accuracy but have no effect on the F1-score. Accuracy and F1-score are harsh metrics, as they pertain to whole facts. An annotation would achieve 0 accuracy if it contained correct individuals and correct predicates, but related them in the wrong way, e.g., $fold(paper, person)$ or $carry(person, paper)$ in Figure 6.1. All results are taken from a held-out test set, using the train/val/test splits defined in the original MSVD and MSR-VTT datasets.

There are limited existing baselines (only (Vasile and Lukasiewicz, 2018)) for the task of KG extraction from videos. To benchmark our performance, I also compare to an artificial baseline composed of the video-captioning network from (Olivastri, G. Singh, and Cuzzolin, 2019), and the semantic parser used to create our datasets (described in Section 6.2).

Our model significantly outperforms the previous work by Vasile and Lukasiewicz (2018), and performs on par with the video-captioning baseline. Importantly, it shows the best positive accuracy, which is the most difficult metric to score highly on.

The metrics, both for the baseline and for our network, are lower on MSR-VTT*. This is consistent with other reported results in the literature. For example (Olivastri, G. Singh, and Cuzzolin, 2019; J. Xu, T. Yao, Y. Zhang, and Mei, 2017; X. Zhang, K. Gao, Y. Zhang, D. Zhang, J. Li, and Tian, 2017; Y. Chen, S. Wang, W. Zhang, and Q. Huang, 2018; B. Wang, L. Ma, W. Zhang, and W. Liu, 2018) all report lower video captioning performance on MSR-VTT than on MSVD. A likely reason is the greater vocabulary size of MSR-VTT:

Table 6.2: Results on the MSVD* video and derived KGs, as described in Section 6.2 (best results in bold).

	F1-score	Positive Accuracy	Negative Accuracy	Total Accuracy
Our Model	**13.99**	**12.65**	99.20	22.16
Baseline	13.5	7.53	**99.96**	**25.55**
V&L 2018	6.11	3.36	-	-

Table 6.3: Results on the MSR-VTT* video and derived KGs, as described in Section 6.2 (best results in bold).

	F1-score	Positive Accuracy	Negative Accuracy	Total Accuracy
Our Model	8.90	**7.33**	99.48	59.19
Baseline	**11.83**	6.76	**99.96**	**83.01**

29,316 vs. 13,010 for MSVD (figures taken from (J. Xu, Mei, T. Yao, and Rui, 2016)). Although I exclude all but the most common individuals, a greater vocabulary size in the original NL captions would still make our task more difficult. It means that more facts will be excluded, and so there will be more annotations with missing information, which do not, therefore, fully describe the input video.

6.4.2 Ablation Studies

To investigate the contribution of each part of our neural architecture, I perform ablation studies on the encoder f, the predicate MLPs, and the individual vector ($\{m_p \mid p \in P\}$ and $\{V_a \mid a \in A\}$ in the terminology of Section 6.3).

The results for MSVD* and MSR-VTT* are shown in Tables 6.4 and 6.5, respectively. In the "without encoder" setting, the video encoding feature vector is replaced with a randomly generated vector. This tests whether the system is actually using information from the video or is just predicting common individuals and predicates, e.g., $talk(man)$. That the results in this setting are worse shows that information from the video is indeed being used. In the "single MLP decoder" setting, one MLP is used for all predicates, in contrast to the main system in which each predicate has its own MLP. This is to test whether predicate-specific information is being used, e.g., whether the system makes meaningfully different predictions for $throws(woman, ball)$ and $sees(woman, ball)$. Similarly, in the single-individual decoder setting, one vector is used for all individuals, in contrast to the main system in which each individual has its own vector. This tests if individual-specific information has been learned by the individual vectors. Both decoder ablation settings produce worse results, which shows that the predicate-specific MLP and individual-specific vectors have learned some meaningful semantics of their respective predicates and individuals. Note that the results suffer more in these ablation settings on MSR-VTT* than on MSVD*. This is to be expected, as MSR-VTT* contains a larger vocabulary, which allows for more semantic differentiation between different predicates and between different

Table 6.4: Ablation results for MSVD* (best results in bold).

	F1	Positive Accuracy	Negative Accuracy	Total Accuracy
Our Model	**13.99**	**12.65**	99.20	22.16
Single MLP decoder	13.75	12.5	98.45	19.04
Single individual vector decoder	10.72	9.91	**99.32**	**22.50**
Without encoder	8.87	9.77	97.49	15.60

Table 6.5: Ablation results for MSR-VTT* (best results in bold).

	F1	Positive Accuracy	Negative Accuracy	Total Accuracy
Our Model	**8.90**	**7.33**	**99.48**	**59.19**
Single MLP decoder	6.16	6.75	99.15	49.09
Single individual vector decoder	3.84	3.94	99.51	58.72
Without encoder	5.00	5.36	99.19	49.64

individuals.

6.4.3 Qualitative Results

To further evaluate the quality of our KG extractions, I present some manual inspections of images and the predicted facts. Figures 6.4 and 6.5 show first frame and predicted facts from MSVD* and MSRVTT*, respectively. (Figure 6.1 shows another from MSR-VTT*.)

Having predicted the facts present in the annotation itself, we can then infer additional facts from KB and the facts generated by the video annotation network, which is one of the advantages of our approach of using KGs. The class man is a subclass of $person$ and $male$ in WordNet, and this inference can be applied to all facts mentioning a man, producing $stand(male)$ and $stand(person)$ in Figs. 6.4 and 6.5. If one wanted to determine how many videos in a database depicted at least one person, or how many depict males, or how many depict males sitting vs. standing, this would not be possible by merely annotating the videos with NL sentences. It is, on the other hand, possible with our system, because of the inferences that can be made on top of a KG. Finally, the qualitative examples show the limitations imposed by the smaller vocabulary size in MSVD*. As discussed in Section 6.2, I exclude individuals and relations that appear fewer than 50 times across the dataset. This means that there is sometimes insufficient material to fully describe a video. For example, the man in the video shown in Figures 6.4 and 6.5 is eating a banana, and there was one MSVD caption that expressed this. However, $banana$ appears fewer than 50 times, so it is excluded from our training data, and the model cannot predict $eat(man, banana)$. (Recall from Section 6.2 that there are multiple annotations for each video, and I merge all of them in our dataset. The annotation shown in Figures 6.4 and 6.5 is one that was not excluded.)

Note also, that despite producing a coherent and accurate natural language sentence, the video captioning system misses some *direct* information present in the video, that is,

(1) MSVD ground truth:
a man is standing at the bus stop

(2) Predicted caption from (Olivastri, G. Singh, and Cuzzolin, 2019):
a man is talking on the phone

---

(3) MSVD* ground truth:
$stand(man)$

(4) Predicted individuals:
man

(5) Predicted facts:
$stand(man)$, $sit(man)$

(6) Some inferred facts:
$stand(person)$, $stand(male)$

Figure 6.4: The first frame from a video in the MSVD* test set, with the ground-truth caption, the caption produced by (Olivastri, G. Singh, and Cuzzolin, 2019), the facts predicted by our system, and some further inferences made using KB and these facts. Compared with a video captioning system, our output can be used to infer additional information about the video, such as the fact that it contains a male.

(1) MSR-VTT ground truth:
a man is driving a car

(2) Predicted caption from (Olivastri, G. Singh, and Cuzzolin, 2019):
a man is driving a car

---

(3) MSR-VTT* ground truth:
$drive(man, car)$

(4) Predicted individuals:
man, car, $person$

(5) Predicted facts:
$drive(man, car), drive(person, car)$

(6) Some inferred facts:
$drive(man, vehicle)$

Figure 6.5: The second frame (first is very unclear) from a video in the MSR-VTT* test set, with the ground-truth caption, the caption produced by (Olivastri, G. Singh, and Cuzzolin, 2019), the facts predicted by our system, and some further inferred facts. Although the video captioning model produces a sentence similar to the ground truth, it cannot be used for inference and so misses the additional inferred information.

information which could not even be inferred, such as the paper being white in Figure 6.1. Because the video captioning model is trained to mimic natural language sentences, there is a strong incentive to produce grammatical sentences. This encourages the production of short, simple sentences, a problem noted by (H. Yu, J. Wang, Z. Huang, Y. Yang, and W. Xu, 2016) among others, which can omit some information from the video. Our outputs are not bound by English grammar, and so are not biased towards short, simple annotations.

6.5 Further Discussion

In this section, I further discuss the advantages of generating KGs compared to NL annotations with reference to the output of my model. I also examine my model as compared to the video-captioning baseline.

6.5.1 Value of KGs

One advantage of KGs is that they enable the application of automated data processing. An example is the application of inference rules to infer additional facts, which is illustrated by the results shown in Figs. 6.4 and 6.5. The facts predicted by our model (3) allow the inference of further information (6), whereas the NL annotation produced by (Olivastri, G. Singh, and Cuzzolin, 2019) does not, even when it is exactly correct, as in the case of the MSR-VTT video. As well as producing additional information about the video contents, these inferences are especially useful for producing abstract classes and relations, such as $male$ in Figures 6.4 and 6.5 or $change$ Figure 6.1. Other examples of automated data processing are query answering and search. If we are interested in videos that depict a male changing some object in some way, as happens in the video shown in Figure 6.1, then using the extracted KGs (including further inferences), we can simply return the videos where, for some individual x, the fact $change(male, x)$ has been predicted. In contrast, NL annotations do not enable us to perform such a search. We cannot simply replace all occurrences of man, boy, guy etc. with $male$, and all occurrences of $fold$, $open$, $close$ etc. with $change$ and then count the annotations that contain both $male$ and $change$. This would produce false positives with annotations like "a man is handed a folded sheet of paper". In such a sentence, although $male$ and $change$ would both be present, they are not connected in the way that we are interested in. The sentence does not describe a scenario in which a male is changing something, even though it contains the words man and $fold$. It is non-trivial to represent the information encoded in a NL sentence as a set of individuals and relations that hold between members of this set. To apply such a translation, a semantic parser would need to be employed, and this would then essentially be the method of the video-captioning baseline, and the demerits of this approach are discussed below. String matching in NL is not even able to detect the individuals that are present. The sentence "a person folds a boy's school uniform" could describe a video in which no boy is present. My annotations, on the other hand, represent information in the semantically structured format of a KG, from which information such as the above can be easily read off. Finally, observe that the set of facts (equivalent to a KG) shown in Figures 6.4 and 6.5 could be easily represented in a language other than English. The individuals and predicates in the vocabulary correspond to particular

senses of words, so disambiguating polysemous words is not required. To translate, e.g., the fact $drive(man, car)$ to German, we could simply look up the translation for the senses of each of these three words: drive $\rightarrow$ fahren, man $\rightarrow$ Mann, and car $\rightarrow$ Auto. Translating one item at a time, we then get $fahren(Mann, Auto)$. In contrast, translating the NL sentence would require both the disambiguation of polysemous words, and the extraction of English syntactic features, and their expression in German.

6.5.2 Comparison with Video-Captioning Baseline

One possible method of producing KGs, and hence the advantages discussed above, is to first produce an NL annotation and then semantically parse this NL sentence to convert it to a set of facts. This is the method employed in the video-captioning baseline, to which I compare my performance in Section 6.4. The quantitative comparison, across the multiple metrics and datasets, shows that it performs about equally well to my model. However, there are three reasons to believe that the latter is more promising for future progress. Firstly, the video-captioning baseline must solve two problems to produce an annotation: the problem of annotating videos with NL, and the problem of semantically parsing NL. This means it has two potential sources of error. However, errors in the second stage (that of semantic parsing) are not properly penalized in the results in Tables 6.2 and 6.3, because the baseline uses the same semantic parser as was used to create the dataset. Therefore, if the video-captioning network correctly predicts the original NL caption, the result, after parsing, will automatically be correct according to the generated dataset. This will happen even if there was a mistake in the semantic parse. For example, in Figure 6.1, the NL annotation predicted by (Olivastri, G. Singh, and Cuzzolin, 2019) is "a person is folding a piece of paper", which the parser then, incorrectly, parses as $fold(person, piece)$ (instead of $fold(person, paper)$). However, this same incorrect parse was made when creating the dataset, so the ground truth in MSR-VTT* is also $fold(person, piece)$, and the baseline is marked correct in this prediction. If there existed a dataset of videos with human-made KGs, the baseline would still require a semantic parser, and its accuracy would be limited by the parsing accuracy, whereas my model would not.

A second advantage of my model is that a user can restrict the vocabulary of the annotations when desired. One may, on occasion, only be interested in a certain set of individuals, and one could easily modify the network to only output annotations concerning these individuals. If, for example, one were only interested in determining whether a video contained any animals and what those animals were doing, they could remove all facts from the generated dataset other than those pertaining to animals and then train my model on the modified dataset. This would be a simpler task and be expected to give higher accuracy. The video-captioning baseline, however, would still output an NL sentence as an intermediary step, and it would not be simple to apply the same restriction to this space of NL sentences. One may think that the vocabulary of the output language model in the video-captioning network could be restricted to only a certain, animal-related vocabulary. However, many words that have one meaning denoting an animal have also other meanings, e.g., "fly" or "bear". Again, this highlights the complexities of working with NL and so the undesirability of using it as an intermediary step to produce a KG.

Thirdly, the captioning and parsing approach is only possible where there already exist

networks mapping the input domain to NL. This is true for videos, but need not be true in general. My method, on the other hand, can be applied to any input domain, as long as a suitable encoder is used.

6.6 Summary

This chapter has proposed the task of KG extraction from videos, where a KG is composed of a set of facts that describe properties of holding of individuals and relations holding between individuals. I have provided arguments and empirical support for the advantages of KGs over NL annotations. No datasets currently exist of videos and corresponding KGs, so I have proposed a method to generate them from existing datasets of videos and NL annotations. Further, I have introduced a deep learning model for KG extraction, and evaluated its performance both qualitatively and quantitatively on two generated datasets.

Knowledge graphs are perhaps the most direct kind of discrete semantic representation. When we view a semantically rich piece of data, as a video or image, we come to understand it, in the most basic sense, by answering two questions: what things/objects are depicted? and what properties do they have/how do they relate to each other? The discrete representations formed by the deep clustering methods from Chapters 3, 4 and 5 all answered the former, by labelling each data point as a certain type of thing in the form of a cluster label, but were unable to answer the latter. Knowledge graphs, on the other hand, can answer both. The extra ability of the model from this chapter comes at the cost of dependence on labelled training data, whereas methods in previous chapters were unsupervised. As might be expected, adding supervision to the training, despite having practical disadvantages and being arguably a poor model of human learning (as briefly discussed in Section 1.1), does enable a richer discrete representation to be learnt.

Chapter 7

Hierarchical Clustering to Measure Data Complexity

Abstract

This chapter presents a method to quantify the complexity of images. Being able to quantify the complexity of data is an important question in machine learning, computer science, and data science. In the case of image data, a number of methods have been proposed. However, existing methods are based only on the degree of variation across the image, and cannot distinguish meaningful content from noise. In particular, existing methods assign a very high complexity to white noise images, despite such images containing no meaningful information. In this chapter, I present a method to measure the complexity of images through hierarchical clustering of patches. Beginning with individual pixels, I assemble a hierarchy of patches, each composed of cluster labels from the level below. The complexity is the sum, across all levels, of the entropy of cluster labels inside all patches on that level. The result is that the image is represented as a hierarchical structure of cluster labels. Clustering is performed using the minimum description length principle (MDL), which is a theoretically sound means of determining the number of clusters. I test against existing methods on seven different sets of images, four from public image datasets and three synthetic, and show that mine is the only method that can assign an accurate measure of complexity to all images considered. Every other method measures white noise as highly complex, while my method gives it zero complexity. I also prove theoretically that the method will assign low complexity to white noise, by showing that the MDL-optimal number of clusters for white noise is one. I then present ablation studies showing the contribution of the components of my method, and how it performs when the inputs are modified in certain ways, including the addition of Gaussian noise and the lowering of the resolution.

7.1 Introduction

There is unavoidable subjectivity in trying to quantify the notion of complexity. This is always the case when defining a new metric. I cannot begin the investigation of a complexity metric by defining what I take complexity to mean, that would be to put the cart before the horse. Inevitably, the investigation involves exploring what complexity is, not just how to measure it, that is, the definition of complexity and the specification of a complexity metric are two sides of the same thing, the latter is really an instantiation of the former. For example, if one defines complex images as those in which there is high variation between all the pixel values, then it is natural to use the entropy of pixel values as a complexity metric; or if one defines complex images as those in which nearby pixel intensities tend to be very different from each other, then another metric is the obvious choice (grey-level co-occurrence matrix, which is discussed in Section 7.2). This renders unavailable the standard blueprint for applied machine learning research of showing that a novel method outperforms existing methods on some quantifiable task or benchmark, because the field does not have such a benchmark for measuring image complexity. What we do have is a vague idea of what complexity is, vague but still powerful and important. The task is to translate this vague idea into something computable.

7.1.1 Assembly Theory

Many existing techniques for quantifying and measuring image complexity (discussed further in Section 7.2) are based on measuring intricacy, the idea being that the more intricate it is and the more dissimilar its parts, the more complex it is. This is relatively easy to measure, but it is incomplete for two reasons. Firstly, and most importantly, it does not distinguish between meaningful intricacy (i.e., signal) and meaningless intricacy (i.e., noise). Using intricacy as a measure of complexity means that a white-noise image, where the pixel values are chosen independently at random, is measured as highly, perhaps even maximally, complex, because there is a high degree of difference between neighbouring pixels. While measuring a white noise image as highly complex may be the desired outcome for some cases, it does not agree with our judgements of complexity, and often we would like to measure complexity in a way that is not affected by noise. A second limitation of existing methods is that they cannot capture the fact that images can have a different complexity at different scales. A blurry photograph of a complex scene, for example, is locally simple but globally complex, while a finely-detailed but repetitive pattern is the opposite.

Rather than model complexity as a high degree of variation, an alternative conception is that complex things are those which take a large number of steps to assemble. Quantifying complexity based on the assembly process is the approach taken in the theory of assembly pathways (Cronin, Krasnogor, B. Davis G., Alexander, Robertson, Steinke, Schroeder, Khlobystov, G. Cooper, Gardner, Siepmann, Whitaker, and Marsh, 2006; Marshall, D. Moore, Murray, S. I. Walker, and Cronin, 2019), originally for the purpose of quantifying the complexity of molecules to aid in the search for extraterrestrial life (Marshall, Mathis, Carrick, Keenan, G. J. T. Cooper, Graham, Craven, Gromski, D. G. Moore, S. Walker, and Cronin, 2021; Schwieterman, Kiang, Parenteau, Harman, DasSarma, Fisher, Arney, Hartnett, Reinhard, Olson, Meadows, Cockell, S. I. Walker, Grenfell, Hegde, Rugheimer,

R. Hu, and Lyons, 2018). The pathway assembly index of an object is the minimum number of combinations needed to produce it from simple parts, where repeated components can be reused without adding to the count. For example, applied to measuring string complexity, the string 'complexity' has an assembly index of ten, because it contains no repeated subcomponents, whereas 'abracadabra' has an assembly index of only seven, because the sub-string 'abra' can be reused. The method presented here for measuring image complexity is inspired by assembly theory, treating an image as a hierarchy of patches, where each patch is a discrete unit represented by its cluster label and composed of its sub-patches. An image can be thought of as being built out of pixel types (the type again represented by the cluster label) local groups of pixels are combined to form patches, groups of neighbouring patches are combined to form super-patches etc. Compared to the original application with structured data such as strings and molecules, working with images is significantly different in some important ways, and presents a different set of challenges, which are addressed below.

7.1.2 Clustering of Image Patches

The most obvious difference between images and molecules (or strings) is that the former are continuous and the latter discrete. The floating-point numbers representing pixel intensities do not allow an image to be understood as a set of components and sub-components. In order to discretize the structure of the image in this way, I employ clustering, at increasing scales. For the first scale, I cluster the pixel values and replace them with their cluster index. For higher scales, I cluster the multisets of cluster indices in each patch across the image. This is a similar idea to that used by convolutional neural networks, which also process an image patch-wise and hierarchically. The difference is that they extract from each patch a feature vector, whereas my method extracts a cluster index. At each scale, I compute the entropy of the multisets of cluster indices across the image. The total complexity score is the sum of this entropy at each scale. I can also examine the entropy for individual scales to get an indication of the local vs. global complexity in the image: low scales (i.e., low levels in the hierarchy) measure local complexity, whereas higher scales capture more global structure (as shown in Section 7.5.4). In this framework, an image becomes a structured object, namely a hierarchy of patches, each composed of the subpatches it contains. The use of clustering to form a hierarchy of patches in this way allows the application of pathway assembly theory, and also the quantification of complexity at different scales.

7.1.3 Minimum Description Length

At each scale, the cluster indices produced depend on K, which is the number of clusters in clustering the model. I choose K in a sound way by employing the minimum description length (MDL) principle (Rissanen, 1983). MDL says that we should choose the model that can completely represent the given data in the fewest number of bits. Clustering can be interpreted as compression, where I encode each point by its cluster index, along with the residual error of how it differs from the centroid of that cluster. Treating each cluster as a probability distribution, and employing the Kraft-McMillan inequality, I see that the residual error for a point x under the cluster probability distribution p can be represented

using $-\log p(x)$ bits. Representing the data under the clustering model takes $-\sum_x \log p(x)$ bits, plus the number of bits to represent the cluster indices and the model itself. Increasing K reduces the average residual error, but increases the size of the indices and the model itself. By MDL, I choose K so as to minimize the total size. MDL is a key component in filtering out noise from my complexity measure. In white noise images, where there is no meaningful or consistent pattern between different points, MDL finds only one cluster, because the small reduction in residual error from encoding more is not worth the extra cost, so the image ends up with a very low complexity score. As well as showing this empirically, I prove it theoretically. Specifically, I prove that, when modelling clusters as normal distributions, for a clustering model fit on independent and uniformly random pixel values, i.e. white noise, the optimal number of clusters found by MDL is 1.

7.1.4 Contributions

The contributions of this chapter are briefly summarized below.

- I propose a novel measure of image complexity, which has a clear theoretical interpretation and is rigorously grounded in information theory.
- I test my method empirically on seven image datasets, four public and three synthetic datasets that I created. I show that my method performs as desired in distinguishing images from different datasets. In particular, my method is able to correctly assign a low complexity to white noise, in contrast to existing methods, which assign it a high complexity.
- I support these results theoretically by proving that, given normally distributed clusters, MDL will always find just a single cluster when the clustering model is fit on white noise.
- I conduct a further set of experiments, showing how my method can measure complexity at different scales in the image, how it performs when Gaussian noise is added to the image or the resolution of the image is reduced, and how it responds to increasing fractal dimension of a fractal image.

The rest of this chapter is organized as follows. Section 7.2 gives an overview of related work. Section 7.3 describes my method, Section 7.4 contains the proof of correctness on white noise, and Section 7.5 presents my empirical evaluation. Finally, Section 7.6 summarizes my findings and suggests directions for future work.

7.2 Related Work

Image complexity measures are used in a number of different tasks: remote sensing (Falconer, 2004; W. Sun, G. Xu, P. Gong, and S. Liang, 2006; X. Yang and C. Zhou, 2000), automatic target recognition (Peters and Strickland, 1990; X.-T. Wang, W.-c. Ma, K. Zhang, and J. Yan, 2018), psychometrics (Forsythe, Mulhern, and Sawey, 2008), user interface design (Stickel, Ebner, and Holzinger, 2010), and measuring aesthetic properties of art

(Forsythe, Nadal, Sheehy, Cela-Conde, and Sawey, 2011; Carballal, Fernandez-Lozano, Rodriguez-Fernandez, I. Santos, and Romero, 2020). Existing works fall into one of several approaches.

Fractal dimension is a property of curves, which in some sense measures their complexity (B. Mandelbrot, 1967; Falconer, 2004). It can be applied to an image by first binarizing with a threshold, then taking the boundary between white and black pixels as a curve and computing its Minkowski-Bouligand dimension. Lam, Qiu, Quattrochi, and Emerson (2002) explores the use of fractal dimension to measure the complexity of satellite images, and W. Sun, G. Xu, P. Gong, and S. Liang (2006) considers the application to remote sensing images more generally. Both also contain a detailed account of methods that use fractal dimension for image complexity. Forsythe, Nadal, Sheehy, Cela-Conde, and Sawey (2011) compare fractal dimension against human judgements of the complexity and beauty of visual art.

File compression ratio is the ratio between the size of a compressed file under a chosen compression algorithm, and the size of the uncompressed original. Marin and Leder (2013) measures image complexity using the file compression ratio, under two compression algorithms: GIF, which is lossy, and TIFF, which is lossless. The compression ratio was compared to human judgements of complexity, on the International Affective Picture System. It is also used as a complexity measure in Forsythe, Nadal, Sheehy, Cela-Conde, and Sawey (2011) and by P. Machado, Romero, Nadal, A. Santos, Correia, and Carballal (2015). The former investigates the ability of JPEG-ratio, GIF-ratio, and a novel 'perimeter detection' method to predict human judgements of complexity in visual art. The latter explores various combinations of compression algorithms with automated edge detection, and compares the results to human judgements of complexity. The authors find the best results using Sobel and Canny filters, followed by JPEG compression.

Carballal, Fernandez-Lozano, Rodriguez-Fernandez, I. Santos, and Romero (2020) test the accuracy of various supervised machine learning models of complexity by annotating art and non-art images with human judgements of complexity, then regressing these annotations using an ML algorithm that includes feature selection. This was repeated a number of times, and the accuracy of a given feature was taken to be the fraction of times it was selected by the feature selection algorithm.

An alternative method is to use the **gradient of pixel intensities** across the image. This is the approach taken by Redies, Amirshahi, Koch, and Denzler (2012). The gradient is computed separately for each of the RGB channels, and the gradient at a pixel is taken to be the maximum across the three channels. The average gradient across the entire image is then taken as a measure of complexity. This is again applied to quantifying aesthetic judgement of visual art, this time as part of the Birkhoff-like measure (Birkhoff, 1933), which characterizes beauty as the ratio of order and complexity.

A final method to consider is **the Fourier transform**, as used by T. M. Khan, Naqvi, and Meijering (2021). The idea is that the more high-frequency components present in the power spectrum, the more complex the image. The authors investigate using both the mean and the median of the power spectrum. The best results are found for the median power spectrum. The application in this case is guiding neural architecture search, the claim being that one should first measure the complexity of a given image dataset, and then use the result to inform architecture design.

7.3 Method

7.3.1 Minimum Description Length Patch Clustering

Our measure of complexity uses a form of clustering based on description length, i.e., the number of bits needed to specify the given data. Description length is relative to an encoding scheme, and via the Kraft-MacMillan inequality, this corresponds to a probability distribution. Specifically, the Kraft-MacMillan inequality says that, under the optimal encoding scheme (optimal in the sense of being shortest on average) of a probability distribution $p(\cdot)$, the description length of a point x is $-\log p(x)$. In our case, the probability distribution is modelled using a Gaussian mixture model (GMM). The description length is therefore relative to the means $\mu = (\mu_i)_{1 \leq i \leq K}$ and the covariances $\Sigma = (\Sigma_i)_{1 \leq i \leq K}$ of this GMM. The probability of a point x under its assigned component of the mixture model (μ, Σ) is given by

$$p(x, \mu, \Sigma) = \max_{1 \leq k \leq K} \frac{\exp(-\frac{1}{2}(x - \mu_k)\Sigma_k^{-1}(x - \mu_k))}{\sqrt{(2\pi)^d |\Sigma_k|}}, \tag{7.1}$$

where μ_k and σ_k are, respectively, the mean and covariance of the kth component, and d is the dimensionality of the data. Specifying x under p requires first indexing the cluster to which x belongs and then encoding x under the probability distribution of that cluster, which I refer to as the residual error. The latter was just shown to take $-\log p(x, \mu, \Sigma)$ bits. Similarly, the length of the former depends on the encoding scheme for, and equivalently the probability distribution over, the indices $1, \ldots, K$, which can be taken empirically from the data. Specifically, the length of encoding which cluster x belongs to is $-\log n_k/N$, where k is the index of the cluster that it belongs to, n_k is the number of points belonging to cluster k, and N is the total number of data points.

7.3.1.1 Differential Description Length

Because the multivariate normal distributions composing the GMM are continuous probability density functions (pdf), instead of probability mass functions as in the discrete case, it is possible that $p(x, \mu, \Sigma) > 1$. Note that this is always the case for pdfs, e.g. the univariate Normal distribution

$$\mathcal{N}(\mu, \tfrac{1}{5\sqrt{2\pi}})$$

has value 5 at $x = \mu$. In these cases where the pdf is greater than 1, the Kraft-MacMillan inequality would seem to suggest that the corresponding encoding scheme can represent x with a strictly negative number of bits, which of course is not possible. The apparent contradiction is resolved by making explicit the precision with which x is to be encoded. Completely specifying any real number is not possible with a finite number of bits, instead one can only specify an extended region $D_x \subset \mathbb{R}^n$, which contains x. The number of required bits is then determined by the probability mass inside D_x, which is given by

$$p_m(D_x, \mu, \Sigma) = \int_{D_x} p(z, \mu, \Sigma) dz. \tag{7.2}$$

Let ϵ be the coordinate-wise precision for specifying x, i.e., set D_x to be a hypercube of side-length ϵ. The probability mass in D_x is then approximated as $p(x, \mu, \Sigma)\epsilon^d$, giving description length

$$-d \log \epsilon - \log(p(x, \mu, \Sigma) + \log K \,. \tag{7.3}$$

The additional term $-d \log \epsilon$ will be higher for smaller ϵ, and will always increase the total description length to be positive even if $-\log(p(x, \mu, \Sigma) < 0$. That it will be large enough to counterbalance $-\log(p(x, \mu, \Sigma)$ is clear from observing that the probability mass in (7.2) is never greater than 1.

Note that the additional $-d \log \epsilon$ term is independent of the pdf itself. Thus, it can be ignored when using MDL and comparing different pdfs (which correspond to different fit clustering models). That is, when invoking the MDL principle, It is sufficient to look only at the term remaining after the $-d \log \epsilon$ term has been removed:

$$-\log(p(x, \mu, \Sigma) + \log K \,. \tag{7.4}$$

I refer to this remaining quantity as the differential description length. Specifically, I define the differential description length (DDL) to be the negative logarithm of the probability density. It is the continuous analogue of the description length, just as differential entropy is the continuous analogue of entropy. Similar to differential entropy, DDL can be negative. This happens precisely when the probability density is greater than 1, as just discussed. DDL should not be thought of as the description length of that point in bits. Rather, it is related to the description length as follows: for a point x with DDL D, the number of bits required to specify it to a precision ϵ is $\max(\{0, -D - d \log \epsilon\})$. The max is required to account for the case where the region specified by the precision ϵ is larger than the interval in which we already know x to lie. For example, if we assume a priori that x is uniformly distributed on $[0, 1]$, in which case all points have DDL 0 under the prior distribution, and then we try to specify to precision 2, we will end up with

$$-D - d \log \epsilon = 0 - \log 2 = -1 \,.$$

Including the max means that, in such cases, computing the required number of bits gives the correct result of 0.

7.3.1.2 Determining Outliers

As well as choosing the number of clusters (see Section 7.3.1.3), the minimum description length (MDL) principle, can determine which points are outliers with respect to the given model. An outlier can be defined as one that takes more bits to specify under the model than it does to specify directly, independently of the model. We can always specify (up to finite precision ϵ) any point directly using the same discretizing reasoning as above. First, restrict attention to some bounded region of $\mathbb{R}^n$, which is large enough that we can assume it will contain all values the data could have. There are several reasonable choices for such a bounded set: the range of values that can be specified using a standard 32-bit float or the hyperrectangle whose sides are the coordinate-wise ranges across the dataset of patches. I find that the exact choice does not affect results. In my implementation, I choose the hypercube whose sides, in each dimension, run from the minimum to the maximum values

across all dimensions in the dataset. Once this bounded region is specified, partition it into a set of small regions–hypercubes with side-length ϵ–and then specify a point x by indexing the unique region that contains x. The number of possible regions is

$$\left(\frac{a_{max} - a_{min}}{\epsilon}\right)^d ,$$

where d is the dimensionality of the data, and a_{max} and a_{min} are the maximum and minimum values, respectively, that appear anywhere in the image. The number of bits to specify a point directly is then

$$\log\left(\frac{a_{max} - a_{min}}{\epsilon}\right)^d = -d\log\epsilon + d\log(a_{max} - a_{min}). \tag{7.5}$$

Here is where we can ignore the precision value ϵ, because it will appear equally in both description length under the model and the description length from indexing the hypercube. Instead, we can use the differential description length. The indexing of the ϵ hypercube in (7.5), is equivalent, when using the differential description length, of using a uniform prior on $[a_{min}, a_{max}]^d$. Under such a distribution, the DDL of any point is $d\log(a_{max} - a_{min})$. Comparing to the DDL under the model, as in (7.4), a point is an outlier iff

$$-\log(p(x,\mu,\Sigma) + \log K > d\log(a_{max} - a_{min}) \iff \tag{7.6}$$

$$-\log(p(x,\mu,\Sigma) + \log\frac{N}{n_k} > d\log(a_{max} - a_{min}) \iff \tag{7.7}$$

$$\iff p(x,\mu,\Sigma)\frac{n_k}{K} < (a_{max} - a_{min})^{-d}, \tag{7.8}$$

where, as above, n_k is the number of points assigned to the same cluster as x. I can then define the total DDL of x, where x can be specified either directly or using the encoding scheme from the model, as

$$D(x,\mu,\Sigma) = \min\left(d\log(a_{max} - a_{min}), -\log(p(x,\mu,\Sigma) + \log\frac{N}{n_k}\right) . \tag{7.9}$$

7.3.1.3 Determining the Number of Clusters

For a given set of points $X = (x_i)_{1\le i\le N}$, the DDL of the entire set is the sum of the DDL of all its points

$$D(X) = \sum_{i=1}^{N} d(x_i) . \tag{7.10}$$

The description length of X under the GMM depends on the number of clusters in the GMM, and using the MDL principle, we can determine the optimum number of clusters as that which produces the smallest DDL.

Let $\mu(X,K), \Sigma(X,K)$ denote the values of μ and Σ with K components, which maximize the probability of X:

$$\mu(X,K), \sigma(X,K) = \underset{\mu,\Sigma}{\operatorname{argmax}} \prod_{x\in X} p(x,\mu,\Sigma) . \tag{7.11}$$

Finding these optimal parameters means fitting the GMM to the dataset X, and can be performed with the usual expectation-maximization algorithm. Denote by $D(X, K)$ the DDL of X under the optimal encoding corresponding to this fit GMM. The value of $D(X, K)$ is the description length of the model itself plus the DDL of X under the model. The former requires specifying the means and covariances for each component, and so has description length

$$D(X, K) + D(K)\,. \tag{7.12}$$

$$D(K) := Kd \log\left(a_{max} - a_{min}\right) + Kd^2 \log\left(a_{max} - a_{min}\right)\,, \tag{7.13}$$

where ϵ is, as above, the chosen precision with which to represent real numbers. By the MDL principle, the optimal number of clusters K^* is that which minimizes the total description length:

$$K^* = \underset{1 \leq K \leq |X|}{\operatorname{argmin}}\ D(X, K) + D(K)\,. \tag{7.14}$$

Note that one only needs to consider values of K up to the size of the dataset, as adding more clusters beyond that point can only increase the total description length. In practice, I test only values up to 8, as fitting GMMs with many clusters becomes expensive and, in my experiments, does not change results. In my experiments, I use diagonal covariance matrices in the GMM, so d^2 is changed to d in (7.13).

7.3.2 Hierarchical Patch Entropy

The method described in this section is depicted graphically in Figure 7.1. At each level of the hierarchy, we begin with a 3d tensor X of shape (H, W, C) and will cluster the vectors of the last dimension; on the first level, this means clustering 3d vectors specifying the colour intensities for each of the three colour channels at each point. Before clustering, the model computes K^* as in (7.14), then clusters the last-dimension vectors of X using a mixture model with K^* components. From this clustering, we can form the 2d tensor A, of shape (H, W) whose (i, j)th entry is the cluster index of the (i, j)th pixel in X, and B, the 3d tensor of shape $(H - m + 1, W - m + 1, K^*)$ whose (i, j, k)th entry is the count of how many times the kth cluster appears in the $m \times m$ patch beginning at (i, j) in A. The patch size m is a user-set parameter. I refer to the vector at location i, j in B as the signature of the (i, j)th patch. My measure of entropy at this level is the entropy of all signatures that appear in B. Strictly speaking, what I compute is the entropy of the categorical distribution of the frequencies of all signatures that appear in B. I refer to this simply as entropy.

To measure complexity at a larger scale, I repeat the above procedure, this time beginning with B instead of X. Let subscripts denote the level of the hierarchy, so that A_i and B_i are the tensors formed, as just described, on the ith level of the hierarchy. Then we can say that B_i contains the signatures (i.e. counts vectors) of the patches in A_i, and A_i contains the MDL-cluster indices of the last-dimension vectors in B_{i-1}. To begin the iteration, B_0 is set to X, the input image.

The present implementation computes up to B_4, and uses larger patch sizes for each level: 4, 8, 16, and 32. Note, however, that this is not the same as simply clustering larger patches

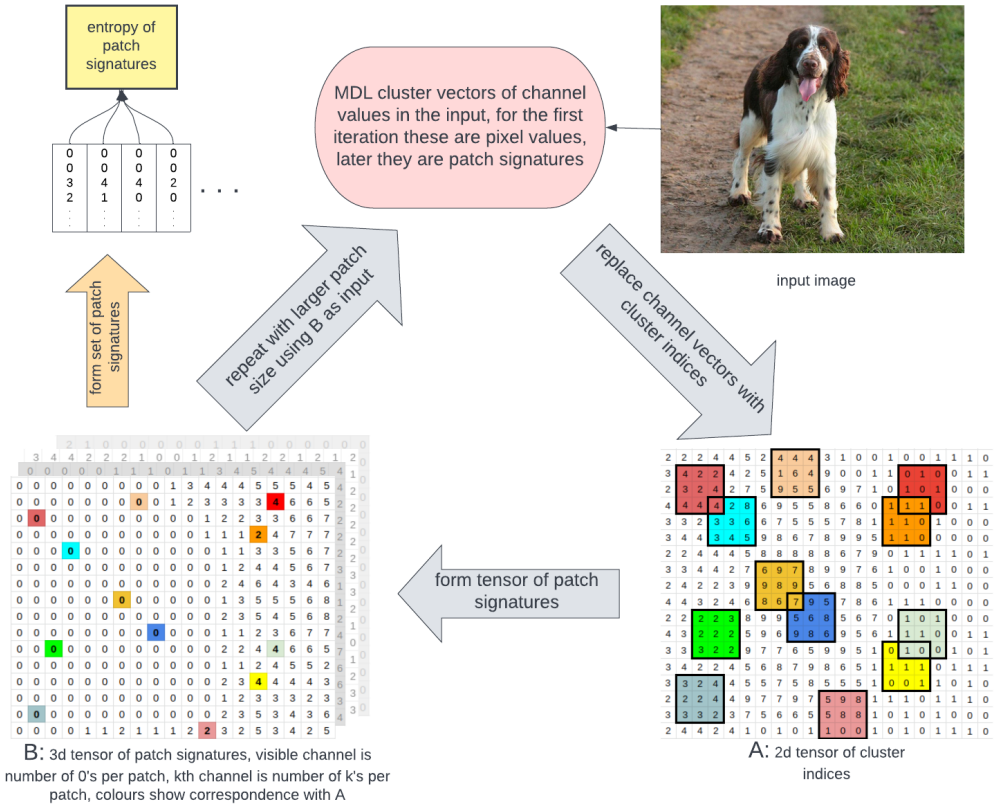

Figure 7.1: Method for computing the entropy of patch signatures as a measure of complexity. Each patch signature is the multiset of MDL cluster indices that appear there.

of an image. What is clustered at each level is the cluster indices from the level below, so is quite different from the input image. The full method is described in Algorithm 6.

7.3.3 Worked Example

This section contains a worked example on a randomly chosen image from ImageNet, shown in Figure 7.2. The steps of my method are enumerated for each of the four layers of the hierarchy. This shows how the final complexity score is arrived at. At each level i, the model

1. performs MDL clustering on the set of array elements B_{i-1}, and assigns each a cluster label, to form A_i (on the first level, this is an image array of pixels, i.e. each pixel gets a cluster label)

2. forms B_i out of patch signatures of multisets of labels in each patch of A_i

Layer 1

Number of points to be clustered (pixels): 50000
Number of components found by MDL, as per (7.14): 8
Assign each pixel a label from $0, \ldots, 7$, and form patch signatures as multisets of labels inside all patches of size 4×4.

Algorithm 6 Algorithm for computing the complexity of an image.

function MDL_CLUSTER(D)
  $best\_DL \leftarrow \infty$
  $A \leftarrow$ cluster indices of MDL of D, initialized randomly
  for $K \in \{1, \ldots, K\_max\}$ **do** ▷ find the K that minimizes description length (DL)
    fit a GMM with K components to D
    $DL \leftarrow$ differential description length of D under this fit GMM, as per (7.13)
    if DL < best_DL **then**
      $A \leftarrow$ cluster indices of D under this fit GMM
      $best\_DL \leftarrow DL$
    end if
  end for
  return A
end function
function ENTROPY(S) ▷ compute entropy of the empirical distribution
  $bin\_counts \leftarrow$ hash table whose keys are the unique elements in S, and whose values are the number of times that element occurs in S
  return $-\sum_{b \in bin\_counts} \frac{bin\_counts[x]}{|S|} \log \frac{bin\_counts[x]}{|S|}$
end function
function COMPUTE_PATCH_SIGNATURES(X,m)
  $A \leftarrow$ MDL_CLUSTER(X)
  $B \leftarrow$ multisets of cluster indices appearing in all $m \times m$ patches of A (including overlapping)
  return B
end function
function COMPLEXITY(X,scales)
  $total\_complexity \leftarrow 0$
  for $m \in scales$ **do** ▷ take the sum of complexities on each level
    $X \leftarrow$ COMPUTE_PATCH_SIGNATURES(X, m)
    $total\_complexity \leftarrow total\_complexity +$ ENTROPY(X)
  end for
  return $total\_complexity$
end function

ht

Figure 7.2: Example of a relatively high-resolution real-world image from imagenet. ID: n03000684_30692.

This results in 48216 different patch signatures, of 1801 different unique values.
Entropy of categorical distribution of patch signatures: **6.42**

Layer 2

Number of points to be clustered: 48216
Number of components found by MDL, as per (7.14): 8
Assign each point a label from $0, \ldots, 7$, and form patch signatures as multisets of labels inside all patches of size 8×8.
This results in 44744 different patch signatures, of 3278 different unique values.
Entropy of categorical distribution of patch signatures: **7.32**

Layer 3

Number of points to be clustered: 44744
Number of components found by MDL, as per (7.14): 8
Assign each point a label from $0, \ldots, 7$, and form patch signatures as multisets of labels inside all patches of size 16×16.
This results in 38184 different patch signatures, of 7358 different unique values.
Entropy of categorical distribution of patch signatures: **11.07**

Layer 4

Number of points to be clustered: 38184
Number of components found by MDL, as per (7.14): 8
Assign each point a label from $0, \ldots, 7$, and form patch signatures as multisets of labels inside all patches of size 32×32.
This results in 26600 different patch signatures, of 7279 different unique values.
Entropy of categorical distribution of patch signatures: **12.73**

Total complexity: $6.42 + 7.32 + 11.07 + 12.73 = \mathbf{37.53}$
More worked examples for different images can be found in the appendix.

7.4 Proof of correctness on white noise

In this section, I prove that the expected DDL of white noise under a GMM is a monotonically increasing function of the number of components in the GMM. This means that the MDL principle will select just a single cluster for white noise. Having just a single cluster is the reason that my method assigns white noise very low complexity, because the entropy of a categorical distribution with just one category is zero.

Lemma 13. *When clustering white noise on $[0, 1]^m$, with a k-component GMM, the radius r of each cluster is approximated by*

$$\frac{1}{\sqrt{3}\sqrt[m]{k}}\left(\frac{2}{\sqrt{3}}\right)^{1/m}. \tag{7.15}$$

Proof. For balls of radius r, the distance along each coordinate axis between their centres will be $2r$ for one of the dimensions and $2r\frac{\sqrt{3}}{2}$ for all other dimensions. This is because the centres of each 3 touching balls will form the vertices of an equilateral triangle with side length $2r$, which will then have height $2r\frac{\sqrt{3}}{2}$. Thus, the number of balls of radius r that can fit inside each axis is $\frac{1}{2r}$ for the first axis and $\frac{1}{2r}\frac{2}{\sqrt{3}}$ for all other axes. The total number of balls that can fit inside $[0, 1]^m$ is, therefore

$$\frac{1}{(2r)^m}\left(\frac{2}{\sqrt{3}}\right)^{m-1}.$$

Conversely, given that the GMM will have k clusters, we can approximate the radius of each cluster using

$$\begin{aligned}
\frac{1}{(2r)^m}\left(\frac{2}{\sqrt{3}}\right)^{m-1} &= k \\
(2r)^m &= \frac{1}{k}\left(\frac{2}{\sqrt{3}}\right)^{m-1} \\
2r &= \frac{1}{\sqrt[m]{k}}\left(\frac{2}{\sqrt{3}}\right)^{\frac{m-1}{m}} \\
r &= \frac{1}{\sqrt{3}\sqrt[m]{k}}\left(\frac{\sqrt{3}}{2}\right)^{1/m}
\end{aligned}$$

□

Proposition 2. *For uniformly distributed points in an m-dimensional hyperball of radius r, the pdf of the distance of a point from the centre is given by*

$$p(x) = m\frac{x^{m-1}}{r^m}$$

Proof. The probability density is clearly proportional to x^{m-1}, thus we need only to find the constant c such that the pdf is appropriately normalized.

$$\begin{aligned} c\int_0^r x^{m-1}dx &= 1 \\ c\frac{x^m}{m}\Big|_0^r &= 1 \\ c\frac{r^m}{m} &= 1 \\ c &= \frac{m}{r^m}\,. \end{aligned}$$

□

Lemma 14. *When clustering white noise in $[0,1]^m$ dimensions, with a k-component GMM, the expected squared distance of a point from the centroid of its cluster, denoted a, is given by*

$$a = \frac{m}{3(m+2)}\left(\frac{\sqrt{3}}{2}\right)^{2/m}\frac{1}{k^{2/m}}\,, \tag{7.16}$$

Proof. Using Proposition 2, the expected squared distance from the centroid can be calculated directly from the pdf as

$$\begin{aligned} a &= \frac{m}{r^m}\int_0^r x^{m+1}dx \\ a &= \frac{m}{r^m}\frac{x^{m+2}}{m+2}\Big|_0^r = 1 \\ a &= r^2\frac{m}{m+2} \end{aligned}$$

Substituting r from Lemma 13 we get

$$a = \frac{m}{3(m+2)}\left(\frac{\sqrt{3}}{2}\right)^{2/m}\frac{1}{k^{2/m}}\,,$$

as desired. □

Lemma 15. *When clustering white noise in $[0,1]^m$ dimensions, with a k-component GMM, the fit covariance matrix will Σ will satisfy $\Sigma = \sigma I$, where*

$$\sigma = \frac{1}{3(m+2)}\left(\frac{\sqrt{3}}{2}\right)^{2/m}\frac{1}{k^{2/m}}$$

Proof. It is clear that Σ will be of the form σI for some σ, as the clusters will all be identical by symmetry (up to approximation at the boundary of the hyperbox, but this is small for more than a few clusters).

Next, observe that the expected squared distance of a point from the centroid of its cluster is, by linearity of expectation, equal to the sum of the expected squared distances in each coordinate, i.e. $m\sigma$. Then, by Lemma 14, we get

$$\sigma = \frac{1}{3(m+2)} \left(\frac{\sqrt{3}}{2}\right)^{2/m} \frac{1}{k^{2/m}}$$

as desired. □

Proposition 3. *The DDL of a point under the distribution of its cluster depends only on its distance from the centroid of its cluster.*

Proof. The probability density of a point z under a cluster with centroid μ is

$$\begin{aligned} p(z) =& \frac{1}{\sqrt{2\pi|\Sigma|}} \exp\left(\frac{-1}{2}(z-\mu)^T\Sigma^{-1}(z-\mu)\right) \\ =& \frac{1}{\sqrt{2\pi|\Sigma|}} \exp\left(\frac{-|(z-\mu)|^2}{2\sigma}\right). \end{aligned}$$

Thus, the DDL under the cluster distribution, which we denote $\bar{D}$, is given by

$$\begin{aligned} &-\ln\left(\frac{1}{\sqrt{2\pi|\Sigma|}} \exp\left(\frac{-|(z-\mu)|^2}{2\sigma}\right)\right) \\ &= \frac{1}{2}\ln\left(2\pi|\Sigma|\right) + \frac{|(z-\mu)|^2}{2\sigma}. \end{aligned}$$

In what follows, we will use the function $f(x)$ to denote the DDL under the cluster distribution of a point a distance x from its centroid, where

$$f(x) = \frac{1}{2}\ln\left(2\pi|\Sigma|\right) + \frac{x^2}{2\sigma}.$$

□

Definition 16. *Denote as outliers, those points with greater DDL under their cluster distribution than under the prior distribution.*

Lemma 17. *When clustering white noise, so that the prior distribution is $[0,1]^m$, a point is an outlier if and only if it is greater than a distance d from is centroid, where $d = \sqrt{\sigma \ln \frac{1}{2\pi|\Sigma|}}$.*

Proof. The DDL of a point treated as an outlier on the uniform m-box $[0,1]^m$ is 0, because treating as an outlier means using the prior distribution, which is uniform $p(x) = 1$, giving

DDL of $\ln 1 = 0$. Thus, a point a distance x from its centroid is not an outlier if and only if its DDL under the distribution of its cluster is strictly negative:

$$\begin{aligned} DDL(x) &< 0 \\ \frac{1}{2}\ln\left(2\pi|\Sigma|\right) + \frac{x^2}{2\sigma} &< 0 \\ \frac{x^2}{\sigma} &< -\ln\left(2\pi|\Sigma|\right) \\ x^2 &< \sigma \ln\frac{1}{2\pi|\Sigma|} \\ x &< \sqrt{\sigma \ln\frac{1}{2\pi|\Sigma|}}\,. \end{aligned}$$

We refer to this distance d as the inlier radius. □

Lemma 18. *When clustering white noise in m dimensions using a GMM with k components, some points will be classed as outliers if and only if k satisfies*

$$k < \frac{2e\sqrt{\pi}}{\sqrt{3}}\left(\frac{e}{3(m+2)}\right)^{m/2}$$

Proof. A point will be an outlier if and only if it is within the cluster, i.e. with a distance r from its centroid, but outsider the inlier radius d. This is possible if and only if

$$\begin{aligned} d &< r \\ \sqrt{\sigma \ln\frac{1}{2\pi|\Sigma|}} &< \frac{1}{\sqrt{3}\sqrt[m]{k}}\left(\frac{\sqrt{3}}{2}\right)^{1/m} \\ \sigma \ln\frac{1}{2\pi|\Sigma|} &< \frac{1}{3k^{2/m}}\left(\frac{\sqrt{3}}{2}\right)^{2/m}\,. \end{aligned}$$

As $\Sigma = \sigma I$, we can sub in $|\Sigma| = \sigma^m$, and also sub in σ from Lemma 15:

$$\frac{1}{3(m+2)}\left(\frac{\sqrt{3}}{2}\right)^{2/m}\frac{1}{k^{2/m}}\ln\frac{1}{2\pi\sigma^m} < \frac{1}{3k^{2/m}}\left(\frac{\sqrt{3}}{2}\right)^{2/m}$$

$$\frac{1}{(m+2)}\ln\frac{1}{2\pi\sigma^m} < 1$$

$$\ln\frac{1}{2\pi\sigma^m} < m+2$$

$$\ln\frac{1}{\sigma^m} < m+2+\ln 2(\pi)$$

$$m\ln\frac{1}{\sigma} < m+2+\ln 2(\pi)$$

$$\ln\frac{1}{\sigma} < 1+\frac{2+\ln 2(\pi)}{m}$$

$$\sigma > \frac{1}{e^{1+\frac{2+\ln 2(\pi)}{m}}}$$

$$\frac{1}{3(m+2)}\left(\frac{\sqrt{3}}{2}\right)^{2/m}\frac{1}{k^{\frac{2m-2}{m}}} > \frac{1}{\sqrt[m]{\pi}e^{1+\frac{2}{m}}}$$

$$\frac{\sqrt[m]{\pi}e^{1+\frac{2}{m}}}{3(m+2)}\left(\frac{\sqrt{3}}{2}\right)^{2/m} > k^{2/m}$$

$$k^{2/m} < \frac{\sqrt[m]{\pi}e^{1+\frac{2}{m}}}{3(m+2)}\left(\frac{\sqrt{3}}{2}\right)^{2/m}$$

$$k < \frac{\sqrt{\pi}e^{\frac{m}{2}+1}}{(3(m+2))^{m/2}}\left(\frac{2}{\sqrt{3}}\right)$$

$$k < \frac{\sqrt{\pi}e^{\frac{m}{2}+1}}{(3(m+2))^{m/2}}\frac{\sqrt{3}}{2}$$

$$k < \frac{\sqrt{\pi}e\sqrt{3}}{2}\frac{e^{m/2}}{(3(m+2))^{m/2}}$$

$$k < \frac{e\sqrt{3\pi}}{2}\left(\frac{e}{3(m+2)}\right)^{m/2}$$

□

Lemma 19. *For a k-component GMM fit on m-dimensional white noise, the expected DDL of a point is*

$$\frac{m}{r^m}\int_0^x x^{m-1}\min(\{0, f(x)\})dx + \ln k\,.$$

with $f(x)$ defined as in Proposition 3.

Proof. The DDL of a point can be decomposed as the number of bits needed to specify which cluster the point belongs to, plus the DDL of the point under that cluster. The former is equal to $\ln k$, because each cluster will, by symmetry, be equally sized. The latter, we denote $\bar{D}$, and is given by

$$\bar{D} = \int_0^r p(r) \min(\{0, f(x)\}) \,,$$

where $p(r)$ is the probability density function of the distance of a point from its centroid and $f(x)$ specifies the DDL of a point in terms of the distance from its centroid.

Substituting for $p(x)$ from Proposition 2 gives

$$\bar{D} = \frac{m}{r^m} \int_0^x x^{m-1} \min(\{0, f(x)\}) dx \,.$$

□

Lemma 20. *When clustering white noise on $[0,1]^m$ with a k-component GMM, for $k \leq d$, the expected DDL is an increasing function of k.*

Proof. When some points are outliers, we have, using Lemma 19

$$\begin{aligned}
\bar{D} &= \frac{m}{r^m} \int_0^r x^{m-1} \min(\{0, f(x)\}) dx \\
&= \frac{m}{r^m} \int_{\{x \in [0,r] | f(x) < 0\}} r^{m-1} f(x) dx \,.
\end{aligned}$$

By Lemma 17, $\{x \in [0,r] | f(x) < 0\} = [0,d)$, giving

$$\bar{D} = \frac{m}{r^m} \int_0^d x^{m-1} f(x) dx$$

Substituting $f(x)$ from Proposition 3 and integrating:

$$\begin{aligned}
\bar{D} &= \frac{m}{r^m} \int_0^d x^{m-1} \left(\frac{1}{2} \ln (2\pi|\Sigma|) + \frac{x^2}{2\sigma} \right) dx \\
&= \frac{m}{r^m} \int_0^d \frac{1}{2} \ln (2\pi|\Sigma|) x^{m-1} + \frac{x^{m+1}}{2\sigma} dx \\
&= \frac{m}{r^m} \left(d^m \frac{1}{2m} \ln (2\pi|\Sigma|) + \frac{d^{m+2}}{2\sigma(m+2)} \right) \\
&= \frac{d^m}{2r^m} \left(\ln (2\pi|\Sigma|) + d^2 \frac{m}{\sigma(m+2)} \right) \,.
\end{aligned}$$

Substituting d from Lemma 17:

$$\begin{aligned}
\bar{D} &= \frac{(\sigma \ln \frac{1}{2\pi|\Sigma|})^{m/2}}{2r^m}\left(\ln(2\pi|\Sigma|) + (\sigma \ln \frac{1}{2\pi|\Sigma|})\frac{m}{\sigma(m+2)}\right) \\
&= \frac{(\sigma \ln \frac{1}{2\pi|\Sigma|})^{m/2}}{2r^m}\left(\ln(2\pi|\Sigma|) + (\ln \frac{1}{2\pi|\Sigma|})\frac{m}{m+2}\right) \\
&= \frac{(\sigma \ln \frac{1}{2\pi|\Sigma|})^{m/2}}{2r^m}\left(\ln \frac{1}{2\pi|\Sigma|}\frac{-2}{m+2}\right) \\
&= \frac{-\sigma^{m/2}}{r^m(m+2)}\left(\ln \frac{1}{2\pi|\Sigma|}\right)^{\frac{m}{2}+1}.
\end{aligned}$$

Substituting σ from Lemma 15:

$$\begin{aligned}
\bar{D} &= -\left(\frac{1}{3(m+2)}\left(\frac{\sqrt{3}}{2}\right)^{2/m}\frac{1}{k^{2/m}}\right)^{m/2}\frac{1}{r^m(m+2)}\left(\ln \frac{1}{2\pi|\Sigma|}\right)^{\frac{m}{2}+1} \\
&= -\left(\frac{1}{(3(m+2))^{m/2}}\left(\frac{\sqrt{3}}{2}\right)\frac{1}{k}\right)\frac{1}{r^m(m+2)}\left(\ln \frac{1}{2\pi|\Sigma|}\right)^{\frac{m}{2}+1} \\
&= -\left(\frac{1}{(3(m+2))^{m/2}}\right)\left(\frac{\sqrt{3}}{2}\right)\frac{1}{kr^m(m+2)}\left(\ln \frac{1}{2\pi|\Sigma|}\right)^{\frac{m}{2}+1} \\
&= \frac{-3\sqrt{3}}{r^m 2(3)^{m/2}k(m+2)}\left(\frac{1}{(m+2)}\ln \frac{1}{2\pi|\Sigma|}\right)^{\frac{m}{2}+1}.
\end{aligned}$$

Substituting r from Lemma 13:

$$\begin{aligned}
\bar{D} &= -\left(\sqrt{3}\sqrt[m]{k}\left(\frac{2}{\sqrt{3}}\right)^{1/m}\right)^m \frac{-3\sqrt{3}}{2(3)^{m/2}k(m+2)}\left(\frac{1}{(m+2)}\ln \frac{1}{2\pi|\Sigma|}\right)^{\frac{m}{2}+1} \\
\bar{D} &= -(\sqrt{3})^m k\frac{2}{\sqrt{3}}\frac{-3\sqrt{3}}{2(3)^{m/2}k(m+2)}\left(\frac{1}{(m+2)}\ln \frac{1}{2\pi|\Sigma|}\right)^{\frac{m}{2}+1} \\
\bar{D} &= \frac{-3}{(m+2)^{\frac{m}{2}+2}}\left(\ln \frac{1}{2\pi|\Sigma|}\right)^{\frac{m}{2}+1}.
\end{aligned}$$

Substituting $\Sigma = \sigma I$, with σ from Lemma 15:

$$\begin{aligned}
\bar{D} &= \frac{-3}{(m+2)^{\frac{m}{2}+2}} \left(\ln \frac{1}{2\pi} + \ln \frac{1}{|\Sigma|} \right)^{\frac{m}{2}+1} \\
&= \frac{-3}{(m+2)^{\frac{m}{2}+2}} \left(\ln \frac{1}{2\pi} + \ln \frac{1}{\sigma^m} \right)^{\frac{m}{2}+1} \\
&= \frac{-3}{(m+2)^{\frac{m}{2}+2}} \left(\ln \frac{1}{2\pi} + \ln \left(3^m (m+2)^m \frac{4}{3} k^2 \right) \right)^{\frac{m}{2}+1} \\
&= \frac{-3}{(m+2)^{\frac{m}{2}+2}} \left((m) \ln 3 + m \ln (m+2) + \ln \tfrac{4}{3} + 2 \ln k - \ln 2\pi \right)^{m/2} \\
&= \frac{-3}{(m+2)^{\frac{m}{2}+2}} \left(m \ln (3m+6) + 2 \ln k + \ln \tfrac{2}{3\pi} \right)^{\frac{m}{2}+1} \\
&= \frac{-3}{(m+2)^{\frac{m}{2}+2}} \left(m \ln (3m+6) + 2 \ln k + \ln \tfrac{2}{3\pi} \right)^{\frac{m}{2}+1}
\end{aligned}$$

Differentiating, with respect to k, the total DDL of $\bar{D} + \ln k$:

$$\begin{aligned}
&\frac{d}{dk} \left(\frac{-3}{(m+2)^{\frac{m}{2}+2}} (m \ln (3m+6) + \ln \tfrac{2}{3\pi} + 2 \ln k)^{\frac{m}{2}+1} + \ln k \right) \\
&= \frac{-3}{(m+2)^{\frac{m}{2}+2}} \frac{d}{dk} \left((m \ln (3m+6) + \ln \tfrac{2}{3\pi} + 2 \ln k)^{\frac{m}{2}+1} \right) + \frac{1}{k} \\
&= \frac{-3}{(m+2)^{\frac{m}{2}+2}} \left((\frac{m}{2}+1)(m \ln (3m+6) + \ln \tfrac{2}{3\pi} + 2 \ln k)^{m/2} \frac{2}{k} \right) + \frac{1}{k} \\
&= \frac{-3}{(m+2)^{\frac{m}{2}+2}} \frac{(m+2)}{k} \left(m \ln (3m+6) + \ln \tfrac{2}{3\pi} + 2 \ln k \right)^{m/2} + \frac{1}{k} \\
&= \frac{-3}{k(m+2)^{\frac{m}{2}+1}} \left(m \ln (3m+6) + \ln \tfrac{2}{3\pi} + 2 \ln k \right)^{m/2} + \frac{1}{k} \,.
\end{aligned}$$

We want to show this derivative is positive:

$$\begin{aligned}
&\frac{-3}{k(m+2)^{\frac{m}{2}+1}}^{\frac{m}{2}+2} \left(m \ln (3m+6) + \ln \tfrac{2}{3\pi} + 2 \ln k \right)^{m/2} + \frac{1}{k} > 0 \\
&\iff \frac{1}{k} > \frac{3}{k(m+2)^{\frac{m}{2}+1}} \left(m \ln (3m+6) + \ln \tfrac{2}{3\pi} + 2 \ln k \right)^{m/2} \\
&\iff \frac{3}{(m+2)^{\frac{m}{2}+1}} \left(m \ln (3m+6) + \ln \tfrac{2}{3\pi} + 2 \ln k \right)^{m/2} < 1
\end{aligned}$$

As, by assumption, some points are outliers, we can use the upper bound on k from Lemma

7.4:

$$
\begin{aligned}
&\frac{3}{(m+2)^{\frac{m}{2}+1}}\left(m\ln(3m+6)+\ln\tfrac{2}{3\pi}+2\ln k\right)^{m/2}\\
&=\frac{3}{(m+2)^{\frac{m}{2}+1}}\left(m\ln(3m+6)+\ln\tfrac{2}{3\pi}+\ln k^2\right)^{m/2}\\
&\le\frac{3}{(m+2)^{\frac{m}{2}+1}}\left(m\ln(3m+6)+\ln\tfrac{2}{3\pi}+\ln\left(\frac{4\pi e^2}{3}\left(\frac{e}{3(m+2)}\right)^m\right)\right)^{m/2}\\
&=\frac{3}{(m+2)^{\frac{m}{2}+1}}\left(m\ln(3m+6)+\ln\tfrac{2}{3\pi}+2+\ln\tfrac{4\pi}{3}+m-m\ln(3m+6)\right)^{m/2}\\
&=\frac{3}{(m+2)^{\frac{m}{2}+1}}\left(2+\ln\tfrac{2}{3\pi}+\ln\tfrac{4\pi}{3}+m\right)^{m/2}\\
&=\frac{3}{(m+2)^{\frac{m}{2}+1}}\left(m+2+\ln\tfrac{8}{9}\right)^{m/2}\\
&<\frac{3}{(m+2)^{\frac{m}{2}+1}}(m+2)^{m/2}\\
&=\frac{3}{m+2}\le 1\,,
\end{aligned}
$$

where the last inequality holds because the dimension m, is ≥ 1. □

Lemma 21. *When clustering white noise on $[0,1]^m$ with a k-component GMM, for $k > d$, the expected DDL is independent of k.*

Proof. When no points are outliers, $f(x) = g(x)$ for all x, so, using Lemma 19

$$
\begin{aligned}
\bar{D} &= \frac{m}{r^m}\int_0^r x^{m-1}f(x)dx\\
&= \frac{m}{r^m}\int_0^r x^{m-1}\left(\frac{1}{2}\ln(2\pi|\Sigma|)+\frac{x^2}{2\sigma}\right)dx\\
&= \frac{m}{r^m}\left(\frac{r^m}{2m}\ln(2\pi|\Sigma|)+\frac{r^{m+2}}{2\sigma(m+2)}\right)\\
&= \frac{1}{2}\ln(2\pi|\Sigma|)+\frac{r^2 m}{\sigma(m+2)}\,.
\end{aligned}
$$

Substituting r from Lemma 13:

$$
\bar{D} = \frac{1}{2}\left(\ln(2\pi|\Sigma|)+\frac{1}{3k^{2/m}}\left(\frac{\sqrt{3}}{2}\right)^{2/m}\frac{m}{\sigma(m+2)}\right)dx
$$

Substituting σ from Lemma 15:

$$\bar{D} = \frac{1}{2}\left(\ln\left(2\pi|\Sigma|\right) + \frac{1}{3k^{2/m}}\left(\frac{\sqrt{3}}{2}\right)^{2/m} 3(m+2)\left(\frac{2}{\sqrt{3}}\right)^{2/m} k^{2/m}\frac{m}{m+2}\right) dx$$

$$\bar{D} = \frac{1}{2}\left(\ln\left(2\pi|\Sigma|\right) + m\right) .$$

Substituting $\Sigma = \sigma I$, with σ from Lemma 15:

$$\begin{aligned}
\bar{D} &= \frac{1}{2}\left(\ln\left(2\pi|\Sigma|\right) + m\right) \\
&= \frac{1}{2}\left(\ln\left(2\pi\right) + m\ln\sigma + m\right) \\
&= \frac{1}{2}\left(\ln\left(2\pi\right) + m\ln\left(\frac{1}{3(m+2)}\left(\frac{\sqrt{3}}{2}\right)^{2/m}\frac{1}{k^{2/m}}\right) + m\right) \\
&= \frac{1}{2}\left(\ln\left(2\pi\right) - m\ln\left(3(m+2)\left(\frac{2}{\sqrt{3}}\right)^{2/m} k^{2/m}\right) + m\right) \\
&= \frac{1}{2}\left(\ln\left(2\pi\right) - m\ln\left(3m+6\right) + 2\ln\tfrac{2}{\sqrt{3}} - 2\ln k + m\right) \\
&= \frac{1}{2}\left(\ln\frac{4\pi}{\sqrt{3}} + m\ln\left(1+\frac{1}{3m+6}\right) - 2\ln k + m\right)
\end{aligned}$$

The full DDL is then

$$\begin{aligned}
&\bar{D} + \ln k \\
&= \frac{1}{2}\left(\ln\frac{4\pi}{\sqrt{3}} + m\ln\left(1+\frac{1}{3m+6}\right) - 2\ln k + m\right) + \ln k \\
&= \frac{1}{2}\left(\ln\frac{4\pi}{\sqrt{3}} + m\ln\left(1+\frac{1}{3m+6}\right) + m\right)
\end{aligned}$$

□

Theorem 22. *When clustering white noise in $[0,1]^m$, using a GMM with k components, the expected DDL of a point is a monotonically increasing function of k.*

Proof. By Lemma 20, the expected DDL is strictly increasing in k up to $k = d$. By Lemma 21, the expected DDL is constant in k for $k > d$. □

7.5 Experimental Evaluation

It is difficult to assess the performance of an image complexity measure empirically. Some works gather human subjective judgements on a particular distribution of images (e.g.,

European renaissance paintings) and report accuracy/correlation, often also training a supervised model on these human judgements (P. Machado, Romero, Nadal, A. Santos, Correia, and Carballal, 2015; Nagle and N. Lavie, 2020). Aside from the practical difficulties of running these psychological studies, evaluating a model on a single distribution does not give a rounded indication of its accuracy, it is unclear how such models will perform when presented with a more diverse set of images. Additionally, collecting human judgements of complexity in this way may not be reliable, they have been shown to be influenced by the presentation of the image as well as cognitive factors such as visual working memory (Sherman, Lim, Grabowecky, and Suzuki, 2013), and show high inter-subject variability (Madrid-Herrera, Chacon-Murguia, Posada-Urrutia, and Ramirez-Quintana, 2019). There is also EEG evidence suggesting that humans use different cognitive processes to judge an image's complexity depending on its degree of naturalness/familiarity (Nicolae and Ivanovici, 2020). I instead evaluate this method with a number of different experiments that, together, show that it assigns complexity scores in a coherent and consistent way, and that it accords with our intuitive understanding of complexity.

Firstly, I present the scores produced by my method for a diverse set of images of different types, taken from different datasets, both public and synthetic datasets that I create, and compare these scores to those produced by existing complexity metrics. Comparing sets/types of images, rather than individual images, has the advantage of reducing subjectivity. One can say with reasonable objectivity that ImageNet images are more complex than MNIST images, whereas trying to compare the complexity of two different renaissance paintings may involve more subjectivity. This experiment shows that the scores produced by my method match our intuitive notion of complexity on this diverse set of images much more closely than do the scores of existing complexity metrics.

Then, after presenting ablation studies, I investigate the distribution of complexity across different levels of the hierarchy, and show that these agree with the different scales of complexity in the different types of images, e.g. fine-detailed repetitive textures receive high complexity on the low levels of the hierarchy but lower scores on the higher levels, compared to globally structured images such as natural scenes from ImageNet.

Next, I show the effect of adding Gaussian noise and of lowering the resolution of images. A small amount of noise or reduction in resolution does not change the content of the image and so should not have a significant effect on the complexity score. For larger reductions in image quality, we would expect a gradual decline in complexity as the information in the image becomes increasingly obscured. This is exactly the case for my method. Its scores are largely unchanged by small quality degradations (addition of noise or reduction in resolution), and then show a steady decline with increasing degradation. As my method so effectively assigns low complexity to white noise images, it is particularly notable that it remains robust to a small/moderate amount of Gaussian noise.

Finally, I present the scores produced by my method on a fractal image, as the fractal dimension is varied. Again, the results are in line with our intuition about the type of complexity expressed by fractal dimension: higher fractal dimensions get a higher complexity score, but this is largely concentrated on the lower, more local levels.

ht

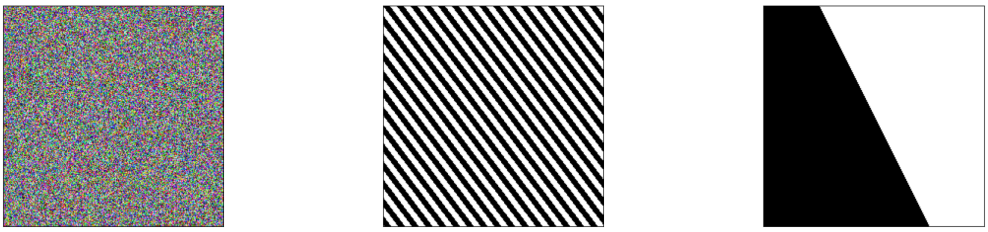

Figure 7.3: Examples of images from the synthetic datasets I create. Left: Rand dataset; Middle: Stripes dataset; Right: Halves dataset.

7.5.1 Datasets

I present the average score of my method on seven different sets of images, four popular image datasets and three synthetic datasets that I created:

1. **ImageNet** is a dataset with high complexity, depicting real-world objects in context.

2. **CIFAR** also shows real-world objects in context but of a much lower resolution, 32×32 vs. approximately 224×224 for ImageNet.

3. **MNIST** depicts low-resolution greyscale digits. Its images are simple in that they can be represented exactly with a small number of bits, but still have meaningful semantic content.

4. **DTD2** is a dataset I created by manually searching through the Describable Textures Dataset (Cimpoi, Maji, Kokkinos, Mohamed, and Vedaldi, 2014) for all images of fine-detailed repeating textures (full list in appendix).

5. **Stripes** is a synthetic dataset I create of greyscale images of stripes of varying thickness and orientation. The thickness of the lines, in pixels, is sampled uniformly at random from $[3, 10]$, and the slope of the lines is sampled uniformly at random from $[-0.5, -1.5]$. It is sufficient to consider negative slopes only as my method, and all methods that I compare to, are invariant to reflections, so the striped images with slope in $[0.5, 1.5]$ would receive identical scores to those in $[-0.5, -1.5]$. Note that my method is not necessarily invariant to rotations, because it is based on square, axis-aligned patches of pixels. The same is true of the fractal dimension computed with the Minkowski-Bouligand dimension (i.e., the fractal dimension), as it uses a box-counting method. An example of an image from Stripes images are shown in Figure 7.3.

6. **Halves** is a synthetic dataset I create of greyscale images of half-black and half-white. These images have one half entirely black and the other entirely white, with the dividing line at various angles. As with Stripes, the slope of this dividing line is sampled

uniformly at random from $[-0.5, -1.5]$. An example of an image from Halves are shown in Figure 7.3.

7. **Rand** is a synthetic dataset I create of white noise images, i.e., images with independent random pixel values. Their values are sampled uniformly at random from $[0, 1]$, independently for each location and each of three colour channels. An example of an image from Rand images are shown in Figure 7.3.

For DTD2, I find 341 suitable images. For all other datasets, I use 500 randomly sampled images and report the average for each image type. The list of exact images used in DTD and ImageNet, along with further examples of images from the synthetic datasets, can be found in the appendix. All images are resized to 224×224. The GMMs used for clustering are initialized with k-means, use diagonal covariance matrices, have tolerance $1e-3$, and are capped at 100 iterations.

7.5.2 Comparison with Existing Methods

Table 7.1 compares my method to seven others: 'khan2021' (T. M. Khan, Naqvi, and Meijering, 2021), 'machado2015' (P. Machado, Romero, Nadal, A. Santos, Correia, and Carballal, 2015), and 'redies2012' (Redies, Amirshahi, Koch, and Denzler, 2012) are as described in Section 7.2; 'entropy' converts the image to greyscale, discretizes the values into 256 bins, and then computes the Shannon entropy of the bin counts; 'fractal dim.' converts the image to greyscale, then binarizes it to 0 or 1, and computes the fractal dimension of the resulting shape using the box-counting method; 'jpg-ratio' measures the ratio of the JPEG-compressed file size to that of the original; and 'GLCM' computes the the average entropy of the grey-level co-occurrence matrix, at offsets 1, 4, 8, 16, and 32 (see Sebastian V., Unnikrishnan, and Balakrishnan (2012) for an account of GLCM in image complexity).

The most striking result is that my method assigns zero complexity to white-noise images, while every other method assigns them high complexity, with many assigning maximum complexity. White noise images are not at all meaningful or interesting to humans, and it is a significant finding that my method is the first to reflect this. It suggests that, while existing methods are based only on the variation across the image, my method is able to measure the degree of *meaningful* variation, i.e. it is able to distinguish signal from noise. With respect to real-world applications of image complexity metrics, being able to distinguish signal from noise is especially relevant to remote sensing, where images often become corrupted by noise due to the sensing equipment or various post-processing steps (Chioukh, Boutayeb, Deslandes, and K. Wu, 2014; Narayanan, Ponnappan, and Reichenbach, 2003; Landgrebe and Malaret, 1986). Much work has been done to reduce noise in remote sensing images and to improve the robustness of image processing methods to noise (Y. Chang, L. Yan, T. Wu, and S. Zhong, 2016; Rasti, Scheunders, Ghamisi, Licciardi, and Chanussot, 2018; W. Huang, S. Zhang, and H. H. Wang, 2020; Duan, X. Kang, S. Li, and Ghamisi, 2019; C. Chen, W. Li, Tramel, Cui, Prasad, and Fowler, 2014).

The only two existing methods not to measure white noise as maximally complex are 'machado2015' and 'redies20212', though they still give it a high score. Instead, they give

Table 7.1: Comparison of my method with existing methods. The figures for each dataset tare the mean across all images from that dataset, with std. dev. in parentheses. All methods are normalized, so the maximum complexity score that they assign is 1. Ours is the only method that does not assign white noise images a high complexity, and gives the most reasonable results on all other datasets.

	Dataset						
	ImageNet	**CIFAR**	**DTD2**	**MNIST**	**Stripes**	**Halves**	**white-noise**
ours	1.00 (.14)	0.92 (.07)	0.72 (.38)	0.63 (.1)	0.44 (.13)	0.33 (.01)	0.00 (.00)
khan2021	0.08 (.05)	0.02 (.01)	0.07 (.06)	0.00 (.00)	0.00 (.00)	0.00 (.00)	1.00 (.00)
machado2015	0.43 (.12)	0.29 (.04)	0.73 (.15)	0.37 (.07)	1.00 (.04)	0.12 (.00)	0.87 (.02)
redies2012	0.19 (.08)	0.07 (.02)	0.32 (.17)	0.05 (.01)	1.00 (.31)	0.02 (.00)	0.89 (.00)
entropy	0.93 (.11)	0.93 (.09)	0.87 (.13)	0.31 (.06)	0.13 (.00)	0.13 (.00)	1.00 (.00)
fractal dim.	0.74 (.09)	0.61 (.07)	0.86 (.16)	0.45 (.06)	0.99 (.02)	0.45 (.02)	1.00 (.00)
jpg-ratio	0.36 (.13)	0.15 (.02)	0.49 (.15)	0.1 (.01)	0.96 (.03)	0.09 (.00)	1.00 (.00)
GLCM	0.99 (.10)	0.84 (.06)	0.99 (.13)	0.27 (.07)	0.11 (.02)	0.09 (.00)	1.00 (.00)

their max score to Stripes. This is also undesirable, because the simple repeating black and white stripes are not intuitively complex or meaningful either. These methods are both based on gradients (see Section 7.2), and the stripes produce a sharp gradient at every transition from black to white, which is likely the reason for these high scores. Stripes is also given a high score by the fractal dimension and JPEG-ratio methods, both assigning it only slightly less than white noise and significantly more than any other dataset, including ImageNet. The method of T. M. Khan, Naqvi, and Meijering (2021) ('khan2021') is difficult to interpret at all, because it assigns such a high complexity score to the white noise that, after normalizing, all other datasets end up close to zero, with three being equal to zero. Recall that this method takes the median of the Fourier transform coefficients, so equals zero if over half of the coefficients are zero. Perhaps surprisingly, the relatively simple methods of entropy and GLCM entropy do a reasonable job of distinguishing real-world images from synthetic images and MNIST, compared to the more bespoke methods. However, they cannot detect a significant difference between ImageNet, CIFAR, and DTD, assigning all three very similar scores. In contrast, my method agrees much more closely with the intuitive notion of complexity: it assigns the highest complexity to ImageNet; it puts CIFAR ahead of DTD2 even though the latter is of higher resolution and has a complex texture, which shows that it recognizes CIFAR to have more semantically meaningful content; and it assigns MNIST a reasonably high complexity, despite it being the smallest in terms of file size, again showing that it can recognize global structure. Even aside from the white noise, no method but ours correctly places the remaining six datasets in order of complexity (left-to-right, as they appear in Table 7.1). This highlights the superior ability of my method to capture meaningful complexity across a variety of image types.

7.5.3 Ablation Studies

Table 7.2 shows the effect of removing two key components of my method. In 'no mdl', I fix the number of clusters to five for all images, rather than using the minimum description

Table 7.2: Effect of removing two main components of my method. In 'no mdl', clustering is performed without MDL, instead simply fixing the number of clusters to 5 for all images and all scales. In 'no patch', I compute the entropy of the clusters themselves rather than of the patch signatures.

	Dataset						
	ImageNet	**CIFAR**	**DTD2**	**MNIST**	**Stripes**	**Halves**	**white-noise**
main	1.00 (.14)	0.92 (.07)	0.72 (.38)	0.63 (.1)	0.44 (.13)	0.33 (.01)	0.00 (.00)
no mdl	0.70 (.06)	0.70 (.06)	0.92 (.11)	0.47 (.1)	0.38 (.04)	0.27 (.01)	1.00 (.00)
no patch	0.98 (.11)	1.00 (.04)	0.62 (.31)	0.65 (.1)	0.79 (.09)	0.53 (.01)	0.00 (.00)

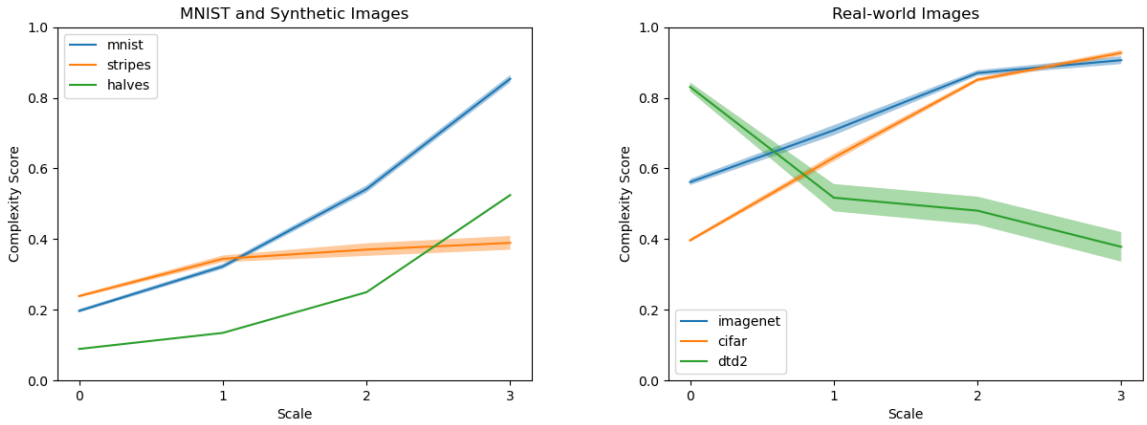

Figure 7.4: My complexity measure for different scales. The x-axis depicts patch size, on a log scale. Plots show mean score for all images of that type. Shaded regions are std dev from batches of 25 images.

length principle. This results in the same problem that existing methods suffer from: white noise is mistaken for high complexity and receives the maximum score. Also, 'no mdl' scores DTD2 too highly, showing that the method is not responding to global structure. In 'no patch', I take the entropy not of patch signatures, but of individual points in the array, i.e., of A rather than B in the terminology of Section 7.3.2. (Patch signatures are still used for the iteration step.) This setting still performs reasonably well, but it gives too high a score to Stripes and a higher score to CIFAR than to ImageNet.

7.5.4 Complexity at Different Scales

The results from Section 7.5.2 suggest that, unlike existing methods, which focus only on detailed textures, ours is able to recognize complexity at a global level. Figure 7.4 provides further support for this claim by showing the breakdown of my complexity measure at the four different scales (that is, four different levels of the hierarchy; see Section 7.3.2). Smaller scales respond to local complexity, and as the process is iterated to larger scales, global structure can be detected.

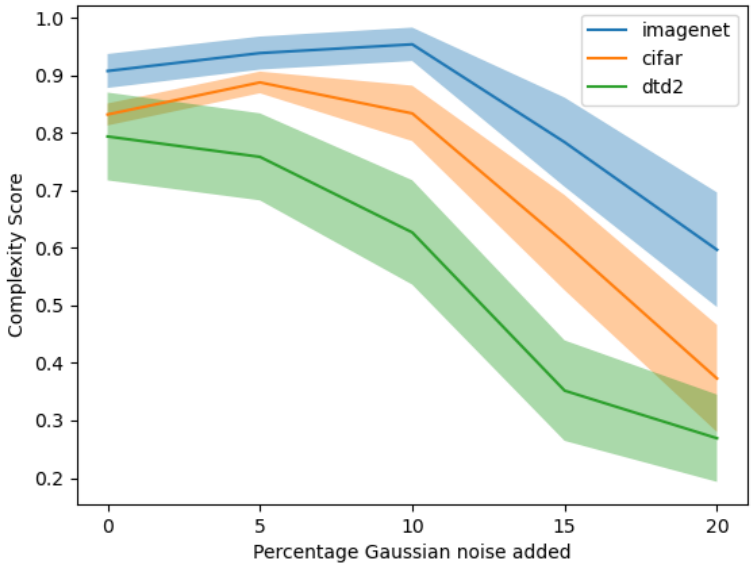

Figure 7.5: My complexity measure with different amounts of Gaussian noise added. Shaded regions are std from batches of 25 images.

The first plot shows MNIST and the synthetic images. While MNIST has a similar local complexity score to Stripes, it has a much higher global complexity score, indicating that the more meaningful global structure in MNIST images can be detected. Halves, which is almost uniform locally but shows some variation globally, is given a very low local complexity but a small amount of global complexity. The second plot compares real-world images. CIFAR has the lowest local complexity because it is low resolution, having been resized from 32×32 so neighbouring pixels are all similar, but this does not affect its global complexity, which is as high as that of Imagenet. DTD2, on the other hand, has the highest local complexity, because it depicts detailed textures, but the lowest global complexity, because the textures are uniform across different regions of the images.

7.5.5 Adding Gaussian noise

As my method so consistently assigns zero complexity to white noise, one may wonder whether it just searches for randomness in the image, and assigns zero if it finds any. To check this, I progressively add Gaussian noise to the three real-world datasets. The results are shown in Figure 7.5. Noise is sampled independently from a standard Normal distribution for each pixel, and a fraction of this noise is added to the image. Up until 10%, the scores are largely unchanged (DTD drops slightly), and then the scores for all three datasets steadily decrease with further noise. If the method was simply assigning low complexity in response to any randomness in the image, then we would see a sharp decline as soon as a small amount of noise is added. The results suggest that the method is instead responding to the amount of meaningful content in the image. A gradual decline in complexity is precisely what we would expect as the image quality deteriorates.

Table 7.3: Comparison, on the scores produced by my method, of downsampling ImageNet to 32×32. Taken from 100 randomly sampled images of the 500 used for the main results in Table 7.1.

	Level 1	Level 2	Level 3	Level 4	Total
full resolution	7.70 (1.75)	9.71 (1.99)	11.93 (1.75)	12.43 (1.98)	**41.77** (5.78)
low resolution	5.60 (0.73)	9.07 (1.43)	11.92 (1.14)	12.72 (0.58)	**39.03** (3.40)

7.5.6 Effect of Low Resolution

To investigate how much my method is affected by the resolution of the input image, I apply it to a downsampled ImageNet. I randomly select 100 of the 500 ImageNet images used for my main experiment and convert them to resolution 32×32. Table 7.3 shows the results of my method on these downsampled images and compares to the full-sized ImageNet images, which are roughly 256×256. There is a slight drop on the lower levels of the hierarchy, which corresponds to the greater uniformity at the local scale in the blurry, low-resolution images. The scores at the higher levels are essentially identical, and overall the scores are almost the same for the downsampled images as for the full-resolution images. This shows my method to be robust to changes in resolution, responding more to the contents of the image than to the resolution it is depicted at.

7.5.7 Scores for Varying Fractal Dimension

Fractal dimension can roughly be defined as the detail in a shape or curve expressed as an exponent of its scale. See B. Mandelbrot (1967) and B. B. Mandelbrot and B. B. Mandelbrot (1982) for the original introduction of the terms and early discussion, and see Edgar (2007) for more recent discussion including different options a precise definition. In this section, I test the scores produced by my complexity metric on images of varying fractal dimension, from the dataset "Color Fractal Images with Independent RGB Color Components" Ivanovici and Richard (2010). This is a small dataset of nine high-resolution colour images, which are essentially the same except that they differ in fractal dimension. The images are generated using the midpoint displacement algorithm, which iteratively increases the fractal dimension of a piecewise-linear curve (i.e. a joined sequence of straight line segments), by slightly moving the midpoint of each piece. The dataset begins with a straight line and iterates until the fractal dimension is a certain value. The values for the different images range from 1.1 to 1.9 in increments of 0.1. This is repeated independently for each of the three colour channels. The resulting images are shown in Figure 7.6.

It is generally thought that a higher fractal dimension indicates greater complexity, and so it is interesting to see whether my method is able to reflect this. Table 7.4 shows the scores produced by my method for each of the nine images in "Color Fractal Images with Independent RGB Color Components", averaged over five runs, with the clustering at each level using 10 different GMM initializations and keeping the one with the highest data likelihood. The table shows the total complexity score, and the complexity score for each level. Looking at the total scores, there is a clear trend of increasing complexity scores for

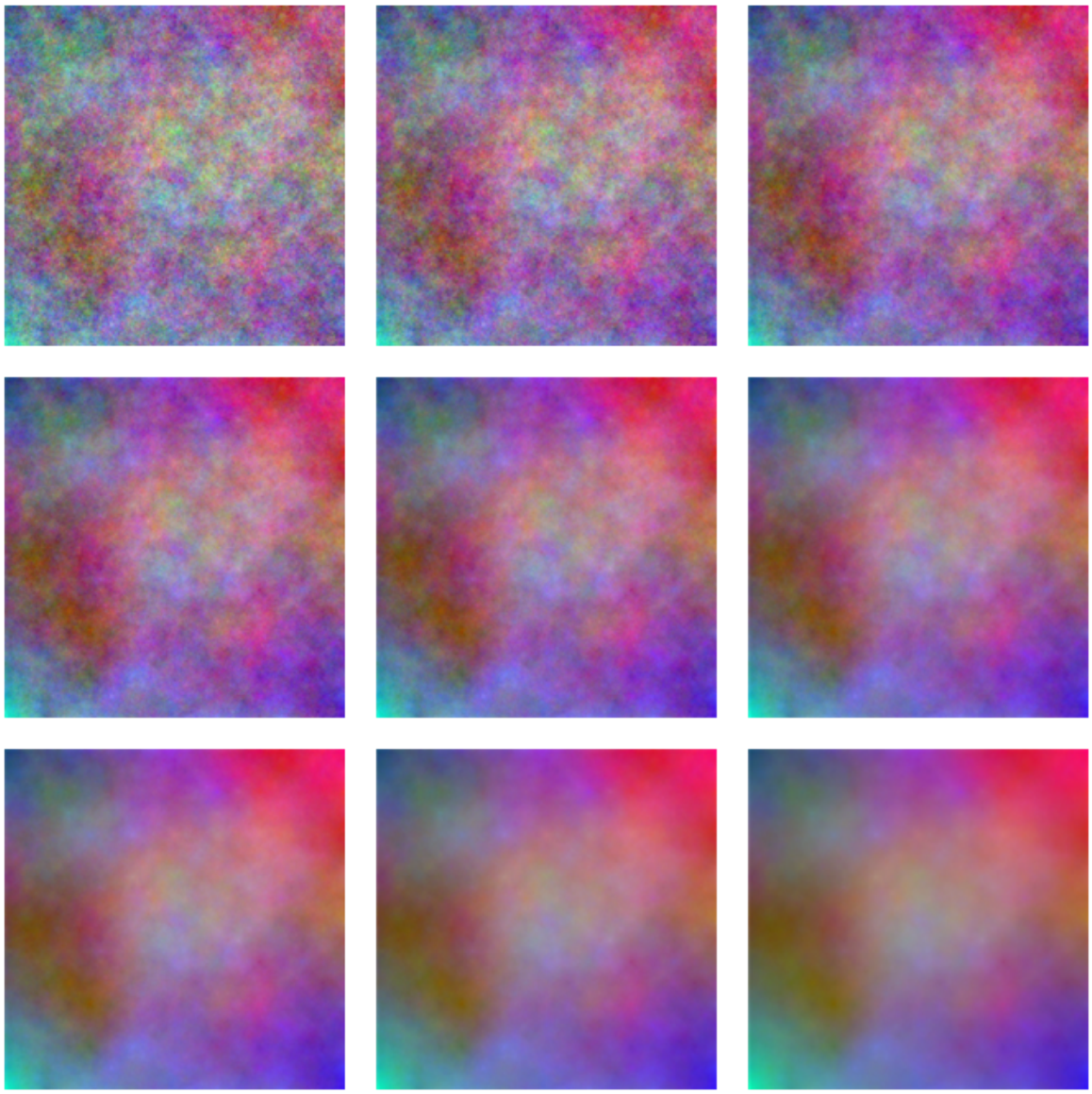

Figure 7.6: The nine images from the “Color Fractal Images with Independent RGB Color Components” dataset. The fractal dimension for the top-left images is 1.9, and then decreases in increments of 0.1 to the bottom-right image.

Table 7.4: Scores for fractal images of increasing fractal dimension, mean from 5 runs with std in parentheses. The clustering at each level in each run uses 10 different GMM initializations and keeps the one with the highest data likelihood.

	Total	Level 1	Level 2	Level 3	Level 4
fract-dim 1.1	32.96 (0.15)	3.79 (0.21)	6.21 (0.27)	10.42 (0.46)	12.54 (0.16)
fract-dim 1.2	34.39 (0.67)	4.02 (0.23)	6.73 (0.33)	11.22 (0.16)	12.41 (0.24)
fract-dim 1.3	34.95 (0.19)	4.24 (0.02)	6.95 (0.21)	11.06 (0.05)	12.70 (0.00)
fract-dim 1.4	36.02 (0.46)	4.60 (0.06)	7.60 (0.55)	11.58 (0.39)	12.24 (0.31)
fract-dim 1.5	37.67 (0.48)	5.31 (0.05)	8.34 (0.39)	11.63 (0.15)	12.39 (0.24)
fract-dim 1.6	39.64 (0.37)	6.27 (0.18)	9.36 (0.27)	11.56 (0.34)	12.44 (0.14)
fract-dim 1.7	41.45 (0.30)	7.35 (0.04)	9.61 (0.18)	12.10 (0.27)	12.38 (0.14)
fract-dim 1.8	44.94 (0.20)	8.96 (0.02)	10.81 (0.15)	12.59 (0.13)	12.58 (0.05)
fract-dim 1.9	47.75 (0.14)	10.75 (0.07)	11.84 (0.25)	12.75 (0.25)	12.42 (0.29)

increasing fractal dimension, showing that the method can detect the sort of complexity expressed in fractal dimension. Looking at the breakdown of this total across the four levels of the hierarchy, we see that the effect of increased fractal dimension is greater for lower levels. Level 1 increases from 3.79 for fractal dimension 1.1 to 10.75 for fractal dimension 1.9, whereas Level 4 shows no systematic increase at all. The same information is shown graphically in Figure 7.7. This makes sense, as the images in Figure 7.6 are very similar at a more global level, e.g. they all have a patch of pink/red in the top-right corner and a patch of purple/blue in the bottom-right corner. The difference across images is within the same patch location.

This shows that, not only is my method able to reliably detect an increase in fractal dimension by assigning a higher complexity score, but it distributes the increase with fractal dimension over the four levels of the hierarchy in the correct way. There is a significant increase at the most local level and progressively smaller increases at higher levels.

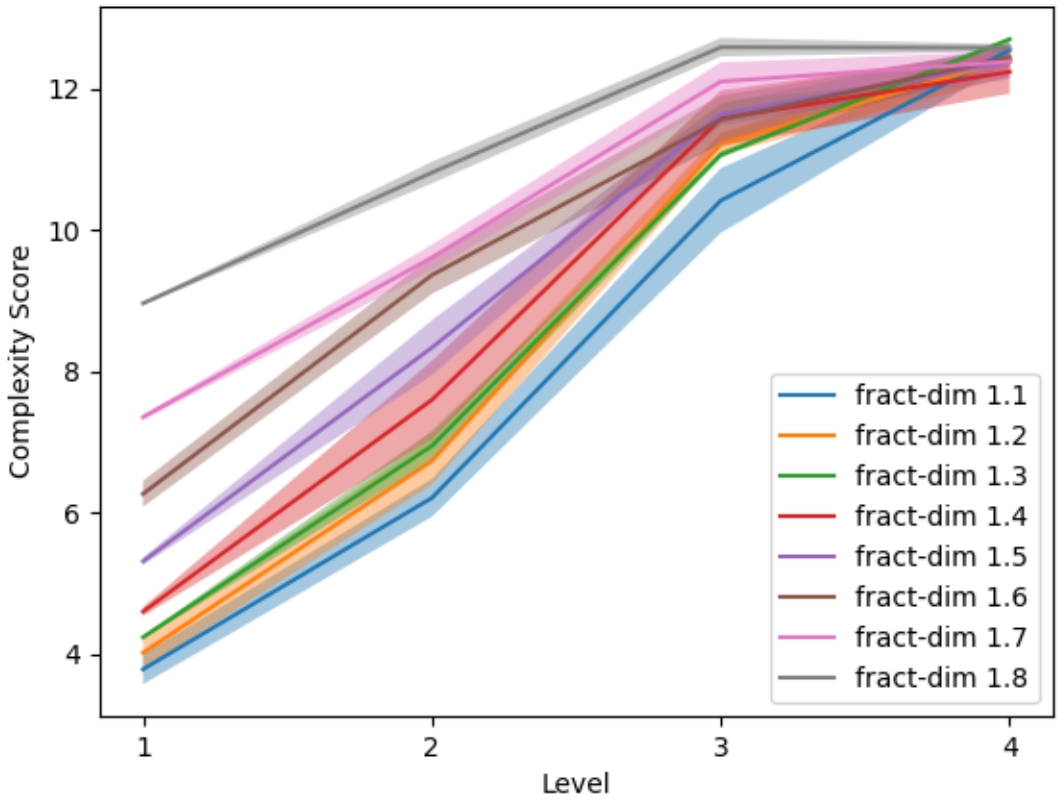

Figure 7.7: Trend of increasing complexity score for increasing fractal dimension, broken down by level of the hierarchy. Lower levels show a greater increasing trend, with no increase at all by the time we get to the highest, most global level.

7.6 Summary

This chapter presented a new method for measuring image complexity. Inspired by assembly theory from computational chemistry, it treats 'complex' as meaning 'takes a large number steps to build'. It uses clustering to analyse an image as being built out of a hierarchy of patches, with each patch composed of the cluster indices of its sub-patches. Clustering is performed with the minimum description length principle to distinguish signal from noise. I gave a detailed derivation of my method, and then presented experimental evaluation showing that it performs better than existing measures of image complexity. It assigns zero complexity to white noise, in contrast to existing methods which all assign white noise very high complexity. This result is also supported theoretically with a proof that white noise contains only one cluster, as judged by MDL, which immediately implies that my method will assign it very low complexity. I then presented ablation studies and a further set of experiments showing that it can accurately capture complexity at different scales, that it is robust to small-moderate degradations in quality, either from the addition of Gaussian noise or from a reduction in resolution, and that it can accurately reflect increasing fractal dimension in fractal images.

Previous chapters explored image clustering where a dataset of images was clustered and each image received a single label specifying its contents. The work of this chapter clusters the different parts of a single image. The result is that the cluster labels form a discrete representation of the continuous input image, a representation which, unlike those formed by clustering a set of images and assigning each a single label, has an internal structure. This was the key step in being able to apply assembly theory to the problem of measuring image complexity, as assembly theory was designed for objects composed of discrete units which fit together in a structured way, namely molecules composed of atoms.

As well as the practical benefit of being able to apply assembly theory, the representation of an image as a hierarchy of discrete units reflects the way in which humans view a scene. For example, an image of a bicycle we see as composed of smaller objects such as the wheels, frame and pedals, and each of those as composed of smaller objects still such as spokes, tires, cross-bars etc. Thus, the work presented in this chapter is in line with both motivations from Section 1.1, in allowing techniques from the discrete world to apply to a continuous object, and in moving our computational representations of data towards those of humans.

Chapter 8

Conclusion

This thesis has explored the conversion of continuous, typically high-dimensional data, to discrete representations. The motivation was twofold: firstly, bridging the gap between the set of data-processing techniques that apply to continuous data, largely dominated by deep learning, and those that apply to discrete data; and secondly, working towards modelling and representing data in a more human-like way. The techniques used to this end were from the areas of deep learning and clustering. Five novel pieces of research, discussed in Chapters 3 to 7, were presented, each presented new techniques/a new approach to a problem, and advanced the state of the art.

Chapters 3, 4 and 5 presented a deep clustering method, methods which combined deep learning and clustering in a single model. Chapters 3 and 4 devised new ways of refining the technique of pseudo-label training, by training only on a subset of points, in Chapter 3 this meant those points that receive the same label by all members of a clustering ensemble, and in Chapter 4, this meant those points that received the same label two iterations in a row. Chapter 5 presented an online clustering method, which avoided the problem of collapse by a novel Bayesian formulation of the cluster assignment problem, and through the novel decision to focus on hard cluster assignments instead of soft cluster assignments. The deep clustering models of these three chapters allowed the high-dimensional continuous input data to be represented by a single atomic unit, namely a cluster label.

Chapter 6 presented a deep learning model to annotate videos with knowledge graphs, which represent the image content in terms of individuals, properties and relations. The task itself is novel: most of the field is focussing on natural language annotations, which I argue to be an inferior choice for image annotation. The architecture of the deep learning model used to solve this task is also novel.

Chapter 7 presented a clustering method to measure the complexity of images. The method forms a hierarchy of cluster labels, where the labels at each level of the hierarchy are formed by clustering patches of labels at the level below. While some existing works have clustered patches of an image, none have formed a hierarchical structure like this. Additionally, none have used internal clustering as part of a complexity metric. The hierarchy of clustered patches allows the development of an accurate image complexity metric, as demonstrated through extensive experiments, by applying a technique originally developed for computational chemistry. A particular benefit of my method is that, unlike all existing image complexity metrics, it does not assign very high complexity to white noise, in fact it

assigns it close to zero. I prove this theoretically, as well as demonstrating it experimentally.

Each of these areas suggests directions for future work. For Selective Pseudo-label Clustering (Chapter 3), the idea of filtering out unconfident pseudo-labels was clearly justified, both theoretically and empirically. A natural next step would be to explore other means of filtering, which may produce even better results. Additionally, the filtering could be graded, rather than binary. That is, rather than exclude some points from training entirely, we could train on all points but put a higher weight on more confident points. Some of these extensions are taken up by the work in Chapter 4, but there is still potential to improve both the filtering and gradation mechanisms.

The most direct future work suggested by Chapter 4 itself is the development of a benchmark for human activity recognition clustering. In this chapter, I identified a number of problems with existing evaluation of HAR clustering, but, although I did suggest solutions, they could be made more concrete and easily actionable by future researchers. This would further help to focus the field around comparable, tangible targets and thus accelerate progress.

Chapter 5 includes a number of possibilities for future extensions. One is to test performance on naturally online data, such as the data gathered by a robot navigating an environment or a data stream from digital communications. Such data sources can only be clustered in an online fashion. Although my method would be applicable to such data sources, this has not been tested empirically. Another extension is to measure, and perhaps improve, robustness to catastrophic forgetting. The data used to train and test the method currently is shuffled in the usual fashion, but some naturally online data would have non-uniform distributions of some classes. Maybe one class is seen many times at the beginning of training, then not at all in the middle, and a small bit again towards the end. We would like to measure if my model can still learn to correctly cluster such a class.

Future work from Chapter 6 includes exploring knowledge graph extraction from other input domains. Of particular interest is the application to text, where our model would perform a task akin to open information extraction. Another extension is to manually construct a dataset designed specifically for the purpose of KG extraction, rather than using those generated by our automated method. There is also scope for improving the architecture. Currently, it selects candidate individuals, and then checks (with the predicate MLPs) all facts containing these individuals. An improvement would be to explore how to also select candidate predicates, and so further refine the candidate facts that are checked. This extension has just been conducted by an MSc student, under my co-supervision.

The method presented in Chapter 7 is quite a new one, without much similar prior work, and so there are many directions for future work. The method itself could be explored further: we could investigate the nature of the clusters it discovers, and whether they correspond to visual textures salient to humans; we could see how this varies over the hierarchy; and we could explore how the method performs with a different backbone clustering model (compared to the current choice of a Gaussian mixture model) and with different hyperparameter choices. As with the work from Chapter 6, it would also be interesting to apply this method to other input domains. For example, audio signals, like images, exhibit varying degrees of complexity, and the hope would be that my method (adapted to the new domain) can identify speech or music, both of which are structured through the combination of discrete units, as high complexity, and ambient noise or white

noise, neither of which have meaningful discrete components, as low complexity.

In addition to the work specific to each chapter, one may also consider the way forward for the original goal of this thesis. The methods I have presented, though successful in the specific tasks they target, may seem to fall short of modelling our own ability to form discrete representations of continuous data. Humans are impressively accomplished in this respect, our state-of-the-art machines still do not really come close. Despite the hype and media attention on artificial intelligence research, this division of continuous from discrete constitutes at least one important bridge it still must cross. A truly intelligent machine will need to process video-like visual data (c.f. Chapter 6) and other data sources such as physical sensors (c.f. Chapter 4), in real-time (c.f. Chapter 5), and convert it to a discrete representation that is compositional (c.f. Chapter 7) and rich enough to enable reasoning and interfacing with a sophisticated memory system (c.f. Chatper 6). Humans integrate all of these abilities, to a very a high accuracy, into a single system, along with others not touched in this thesis at all: motor control, planning, theory of mind, language acquisition and use, and perhaps others that I or even everyone has failed to identify as important. As well as the many technical things that I have learned in undertaking the preceding research, I have been reminded time and again, as I hope the reader will be reminded, of just how ambitious our target is. Fully solving this puzzle will take several major breakthroughs and hundreds, or perhaps thousands, more PhD theses. What I hope to have contributed is some small pieces of what the solution will end up being, some small steps towards the perennial and still tantalizingly elusive goal of artificial intelligence. We cannot even tell how far away this future is, but I am still a believer that one day it will arrive, and we will achieve the collective expression, through computation, of our own intellect.

Bibliography

Abavisani, M. and Patel, V. M. (2018). "Deep multimodal subspace clustering networks". In: *IEEE Journal of Selected Topics in Signal Processing* 12.6, pp. 1601–1614.

Aljarrah, A. A. and Ali, A. H. (2021). "Human Activity Recognition by Deep Convolution Neural Networks and Principal Component Analysis". In: *Further Advances in Internet of Things in Biomedical and Cyber Physical Systems*, p. 111.

Allaoui, M., Kherfi, M. L., and Cheriet, A. (2020). "Considerably improving clustering algorithms using UMAP dimensionality reduction technique: a comparative study". In: *Proceedings of the International Conference on Image and Signal Processing*. Springer, pp. 317–325.

Alsheikh, M. A., Selim, A., Niyato, D., Doyle, L., Lin, S., and Tan, H.-P. (2016). "Deep activity recognition models with triaxial accelerometers". In: *Workshop at the Thirtieth AAAI Conference on Artificial Intelligence*.

Altun, K., Barshan, B., and Tunçel, O. (2010). "Comparative study on classifying human activities with miniature inertial and magnetic sensors". In: *Pattern Recognition* 43.10, pp. 3605–3620.

Anguita, D., Ghio, A., Oneto, L., Parra, X., and Reyes-Ortiz, J. L. (2013). "A public domain dataset for human activity recognition using smartphones." In: *Proceedings of the 21st European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. Vol. 3, p. 3.

Arriaga, O., Valdenegro-Toro, M., and Plöger, P. (2017). "Real-time convolutional neural networks for emotion and gender classification". In: *arXiv preprint ArXiv:1710.07557*.

Asano, Y. M., Rupprecht, C., and Vedaldi, A. (2019). "Self-labelling via simultaneous clustering and representation learning". In: *arXiv preprint ArXiv:1911.05371*.

Ayzenberg, V. and Lourenco, S. (2022). "Perception of an object's global shape is best described by a model of skeletal structure in human infants". In: *ELife* 11, e74943.

Banerjee, S. and Lavie, A. (2005). "METEOR: An Automatic metric for mt evaluation with improved correlation with human judgments". In: *ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pp. 65–72.

Bank, D., Koenigstein, N., and Giryes, R. (2020). "Autoencoders". In: *arXiv preprint ArXiv:2003.05991*.

Banos, O., Damas, M., Pomares, H., Rojas, I., Toth, M. A., and Amft, O. (2012). "A benchmark dataset to evaluate sensor displacement in activity recognition". In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pp. 1026–1035.

Banos, O., Toth, M. A., Damas, M., Pomares, H., and Rojas, I. (2014). "Dealing with the effects of sensor displacement in wearable activity recognition". In: *Sensors* 14.6, pp. 9995–10023.

Banos, O. and Tóth, M. A. (2014). "Realistic sensor displacement benchmark dataset". In: *Dataset Manual*, pp. 1–4.

Belinkov, Y., Poliak, A., Shieber, S. M., Van Durme, B., and Rush, A. M. (2019). "Don't take the premise for granted: Mitigating artifacts in natural language inference". In: *arXiv preprint arXiv:1907.04380*.

Bellman, R. (1966). "Dynamic programming". In: *Science* 153.3731, pp. 34–37.

Bengio, Y. (2017). "The consciousness prior". In: *arXiv preprint ArXiv:1709.08568*.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2006). "Greedy layer-wise training of deep networks". In: *Advances in Neural Information Processing Systems* 19.

Bengio, Y., Lecun, Y., and Hinton, G. (2021). "Deep learning for AI". In: *Communications of the ACM* 64.7, pp. 58–65.

Beyer, K., Goldstein, J., Ramakrishnan, R., and Shaft, U. (1999). "When is "nearest neighbor" meaningful?" In: *Proceedings of the International Conference on Database Theory*. Springer, pp. 217–235.

Billman, D. and Knutson, J. (1996). "Unsupervised concept learning and value systematicitiy: A complex whole aids learning the parts." In: *Journal of Experimental Psychology: Learning, Memory, and Cognition* 22.2, p. 458.

Birkhoff, G. D. (1933). *Aesthetic Measure*. Harvard University Press, Cambridge.

Bloom, P. (2002). *How Children Learn the Meanings of Words*. MIT press.

Boongoen, T. and Iam-On, N. (2018). "Cluster ensembles: A survey of approaches with recent extensions and applications". In: *Computer Science Review* 28, pp. 1–25.

Booth, A. E., Waxman, S. R., and Huang, Y. T. (2005). "Conceptual information permeates word learning in infancy." In: *Developmental Psychology* 41.3, p. 491.

Boutsidis, C., Zouzias, A., Mahoney, M. W., and Drineas, P. (2014). "Randomized dimensionality reduction for k-means clustering". In: *IEEE Transactions on Information Theory* 61.2, pp. 1045–1062.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). "A large annotated corpus for learning natural language inference". In: *arXiv preprint ArXiv:1508.05326*.

Breiman, L. (1996). "Bagging predictors". In: *Machine Learning* 24.2, pp. 123–140.

Brock, A., Donahue, J., and Simonyan, K. (2018). "Large scale GAN training for high fidelity natural image synthesis". In: *ArXiv:1809.11096*.

Cai, Y., Zhang, Z., Liu, Y., Ghamisi, P., Li, K., Liu, X., and Cai, Z. (2021). "Large-scale hyperspectral image clustering using contrastive learning". In: *arXiv preprint ArXiv:2111.07945*.

Carballal, A., Fernandez-Lozano, C., Rodriguez-Fernandez, N., Santos, I., and Romero, J. (2020). "Comparison of outlier-tolerant models for measuring visual complexity". In: *Entropy* 22.4, p. 488.

Caron, M., Bojanowski, P., Joulin, A., and Douze, M. (2018). "Deep clustering for unsupervised learning of visual features". In: *Proceedings of the European Conference on Computer Vision*, pp. 132–149.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. (2020). "Unsupervised learning of visual features by contrasting cluster assignments". In: *arXiv preprint ArXiv:2006.09882*.

Carreira, J. and Zisserman, A. (2017). "Quo vadis, action recognition? A new model and the kinetics dataset". In: *Proceedings of the IEEE / CVF Computer Vision and Pattern Recognition Conference*, pp. 6299–6308.

Chang, J., Wang, L., Meng, G., Xiang, S., and Pan, C. (2017). "Deep adaptive image clustering". In: *Proceedings of the International Conference on Computer Vision*, pp. 5879–5887.

Chang, Y., Yan, L., Wu, T., and Zhong, S. (2016). "Remote sensing image stripe noise removal: From image decomposition perspective". In: *IEEE Transactions on Geoscience and Remote Sensing* 54.12, pp. 7018–7031.

Chen, C., Li, W., Tramel, E. W., Cui, M., Prasad, S., and Fowler, J. E. (2014). "Spectral–spatial preprocessing using multihypothesis prediction for noise-robust hyperspectral image classification". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 7.4, pp. 1047–1059.

Chen, D. L. and Dolan, W. B. (2011). "Collecting highly parallel data for paraphrase evaluation". In: *HTM*, pp. 190–200.

Chen, K., Zhang, D., Yao, L., Guo, B., Yu, Z., and Liu, Y. (2021). "Deep learning for sensor-based human activity recognition: overview, challenges, and opportunities". In: *ACM Computing Surveys (CSUR)* 54.4, pp. 1–40.

Chen, Y., Wang, S., Zhang, W., and Huang, Q. (2018). "Less is more: Picking informative frames for video captioning". In: *Proceedings of the European Conference on Computer Vision*, pp. 358–373.

Chen, Y. and Xue, Y. (2015). "A deep learning approach to human activity recognition based on single accelerometer". In: *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, pp. 1488–1492.

Chioukh, L., Boutayeb, H., Deslandes, D., and Wu, K. (2014). "Noise and sensitivity of harmonic radar architecture for remote sensing and detection of vital signs". In: *IEEE Transactions on Microwave Theory and Techniques* 62.9, pp. 1847–1855.

Cho, K., Merrienboer, B. van, Bahdanau, D., and Bengio, Y. (2014). "On the properties of neural machine translation: encoder-decoder approaches". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 103–111.

Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., and Vedaldi, A. (2014). "Describing textures in the wild". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3606–3613.

Clemen, R. T. (1989). "Combining forecasts: A review and annotated bibliography". In: *International Journal of Forecasting* 5.4, pp. 559–583.

Clerkin, E. M., Hart, E., Rehg, J. M., Yu, C., and Smith, L. B. (2017). "Real-world visual statistics and infants' first-learned object names". In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 372.1711, p. 20160055.

Cleveland, A., Schug, M., and Striano, T. (2007). "Joint attention and object learning in 5-and 7-month-old infants". In: *Infant and Child Development: An International Journal of Research and Practice* 16.3, pp. 295–306.

Creswell, A. and Bharath, A. A. (2018). “Inverting the generator of a generative adversarial network”. In: *IEEE Trans. Neural Netw. Learn. Syst.* 30.7, pp. 1967–1974.

Cronin, L., Krasnogor, N., Davis G, B., Alexander, C., Robertson, N., Steinke, J., Schroeder, S., Khlobystov, A., Cooper, G., Gardner, P., Siepmann, P., Whitaker, B., and Marsh, D. (2006). “The imitation game — A computational chemical approach to recognizing life”. In: *Nature Biotechnology* 24.10, pp. 1203–1206.

Curtis, K., Awad, G., Rajput, S., and Soboroff, I. (2020). “HLVU: A New challenge to test deep understanding of movies the way humans do”. In: *ICMR*, pp. 355–361.

Experiments with random projection. Uncertainty in Artificial Intelligence: Proceedings of the Sixteenth Conference (UAI-2000)(pp. 143–151) (2000). Morgan Kaufmann.

Davies, D. and Bouldin, D. (1979). “A cluster separation measure”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2, pp. 224–227.

Demrozi, F., Pravadelli, G., Bihorac, A., and Rashidi, P. (2020). “Human activity recognition using inertial, physiological and environmental sensors: A comprehensive survey”. In: *IEEE Access* 8, pp. 210816–210836.

Deng, L. and Yu, D. (2014). “Deep learning: methods and applications”. In: *Foundations and Trends® in Signal Processing* 7.3–4, pp. 197–387.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint ArXiv:1810.04805*.

Ding, F. and Luo, F. (2019). “Clustering by directly disentangling latent space”. In: *ArXiv:1911.05210*.

Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., and Zhang, W. (2014). “Knowledge Vault: A web-scale approach to probabilistic knowledge fusion”. In: *Proceedings of the ACM SIGKDD*, pp. 601–610.

Driver, M. J., Svensson, K., Amato, R. P., and Pate, L. E. (1996). “A human-information-processing approach to strategic change: Altering managerial decision styles”. In: *International Studies of Management & Organization* 26.1, pp. 41–58.

Droganova, K. and Zeman, D. (2019). “Towards Deep Universal Dependencies”. In: *Proceedings of the Fifth International Conference on Dependency Linguistics (Depling, SyntaxFest 2019)*, pp. 144–152.

Dua, N., S, S. N., and S, V. B. (2021). “Multi-input CNN-GRU based human activity recognition using wearable sensors”. In: *Computing*, pp. 1–18.

Duan, P., Kang, X., Li, S., and Ghamisi, P. (2019). “Noise-robust hyperspectral image classification via multi-scale total variation”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 12.6, pp. 1948–1962.

Edgar, G. (2007). *Measure, Topology, and Fractal Geometry*. Springer Science & Business Media.

Edmonds, J. and Karp, R. M. (1972). “Theoretical improvements in algorithmic efficiency for network flow problems”. In: *Journal of the ACM (JACM)* 19.2, pp. 248–264.

Elgammal, A., Liu, B., Elhoseiny, M., and Mazzone, M. (2017). “CAN: Creative adversarial networks, generating art by learning about styles and deviating from style norms”. In: *ArXiv:1706.07068*.

Falconer, K. (2004). *Fractal Geometry: Mathematical Foundations and Applications*. John Wiley & Sons.

Ferrari, A., Micucci, D., Mobilio, M., and Napoletano, P. (2019). “Hand-crafted features vs residual networks for human activities recognition using accelerometer”. In: *Proceedings*

of the IEEE 23rd International Symposium on Consumer Technologies (ISCT). IEEE, pp. 153–156.

Forsythe, A., Mulhern, G., and Sawey, M. (2008). "Confounds in pictorial sets: The role of complexity and familiarity in basic-level picture processing". In: *Behavior Research Methods* 40.1, pp. 116–129.

Forsythe, A., Nadal, M., Sheehy, N., Cela-Conde, C. J., and Sawey, M. (2011). "Predicting beauty: Fractal dimension and visual complexity in art". In: *British Journal of Psychology* 102.1, pp. 49–70.

Gao, B., Yang, Y., Gouk, H., and Hospedales, T. M. (2020). "Deep clusteringwith concrete k-means". In: *Proceedings of the 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 4252–4256.

Gao, L., Guo, Z., Zhang, H., Xu, X., and Shen, H. T. (2017). "Video captioning with attention-based LSTM and semantic consistency". In: *IEEE T. Multimedia* 19.9, pp. 2045–2055.

Gao, R. X. and Yan, R. (2010). *Wavelets: Theory and Applications for Manufacturing*. Springer Science & Business Media.

Gelter, H. (2003). "Why is reflective thinking uncommon". In: *Reflective Practice* 4.3, pp. 337–344.

Ghasedi Dizaji, K., Herandi, A., Deng, C., Cai, W., and Huang, H. (2017). "Deep Clustering via Joint Convolutional Autoencoder Embedding and Relative Entropy Minimization". In: *Proceedings of the International Conference on Computer Vision*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). "Generative adversarial nets". In: *Proceedings of the Twenty-eighth Conference on Neural Information Processing Systems*, pp. 2672–2680.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Pires Avila, B., Guo, Z., Azar Gheshlaghi, M., Piot, B., k. kavukcuoglu koray, Munos, R., and Valko, M. (2020). "Bootstrap your own latent-a new approach to self-supervised learning". In: *Advances in Neural Information Processing Systems* 33, pp. 21271–21284.

Guo, X., Gao, L., Liu, X., and Yin, J. (2017). "Improved deep embedded clustering with local structure preservation." In: *Proceedings of the Twenty-eight International Joint Conference on Artificial Intelligence*, pp. 1753–1759.

Guo, X., Zhu, E., Liu, X., and Yin, J. (2018). "Deep embedded clustering with data augmentation". In: *Proceedings of the Asian Conference on Machine Learning*, pp. 550–565.

Haeusser, P., Plapp, J., Golkov, V., Aljalbout, E., and Cremers, D. (2018). "Associative deep clustering: Training a classification network with no labels". In: *Proceedings of the Conference on Pattern Recognition*. Springer, pp. 18–32.

Hammerla, N. Y., Halloran, S., and Plötz, T. (2016). "Deep, convolutional, and recurrent models for human activity recognition using wearables". In: *arXiv preprint ArXiv:1604.08880*.

Han, A. L., Wong, D. F., and Chao, L. S. (2012). "LEPOR: A robust evaluation metric for machine translation with augmented factors". In: *Proceedings of the COLING, Posters*, pp. 441–450.

He, H., Tan, Y., and Huang, J. (2017). "Unsupervised classification of smartphone activities signals using wavelet packet transform and half-cosine fuzzy clustering". In: *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, pp. 1–6.

He, H., Tan, Y., and Zhang, W. (2018). "A wavelet tensor fuzzy clustering scheme for multi-sensor human activity recognition". In: *Engineering Applications of Artificial Intelligence* 70, pp. 109–122.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778.

He, Z. and Jin, L. (2009). "Activity recognition from acceleration data based on discrete consine transform and SVM". In: *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*. IEEE, pp. 5041–5044.

Hinton, G. E. (2009). "Deep belief networks". In: *Scholarpedia* 4.5, p. 5947.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). "A fast learning algorithm for deep belief nets". In: *Neural Computation* 18.7, pp. 1527–1554.

Hinton, G. E. and Zemel, R. (1993). "Autoencoders, minimum description length and Helmholtz free energy". In: *Advances in Neural Information Processing Systems* 6.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). "Improving neural networks by preventing co-adaptation of feature detectors". In: *ArXiv:1207.0580*.

Hongeng, S., Bremond, F., and Nevatia, R. (2000). "Representation and optimal recognition of human activities". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Vol. 1. IEEE, pp. 818–825.

Hopfield, J. J. (1982). "Neural networks and physical systems with emergent collective computational abilities." In: *Proceedings of the National Academy of Sciences* 79.8, pp. 2554–2558.

Horst, F., Lapuschkin, S., Samek, W., Müller, K.-R., and Schöllhorn, W. I. (2019). "Explaining the unique nature of individual gait patterns with deep learning". In: *Scientific Reports* 9.1, pp. 1–13.

Horst, F., Mildner, M., and Schöllhorn, W. I. (2017). "One-year persistence of individual gait patterns identified in a follow-up study–A call for individualised diagnose and therapy". In: *Gait & Posture* 58, pp. 476–480.

Hu, W., Miyato, T., Tokui, S., Matsumoto, E., and Sugiyama, M. (2017). "Learning discrete representations via information maximizing self-augmented training". In: *Proceedings of the International Conference on Machine Learning*. PMLR, pp. 1558–1567.

Huang, J. and Gong, S. (2021). "Deep clustering by semantic contrastive learning". In: *arXiv preprint ArXiv:2103.02662*.

Huang, P., Huang, Y., Wang, W., and Wang, L. (2014). "Deep embedding network for clustering". In: *Proceedings of the Eigteenth International Conference on Pattern Recognition*. IEEE, pp. 1532–1537.

Huang, W., Zhang, S., and Wang, H. H. (2020). "Efficient GAN-based remote sensing image change detection under noise conditions". In: *Proceedings of the International Conference on Image Processing and Capsule Networks*. Springer, pp. 1–8.

Hug, F., Vogel, C., Tucker, K., Dorel, S., Deschamps, T., Le Carpentier, É., and Lacourpaille, L. (2019). "Individuals have unique muscle activation signatures as revealed during gait and pedaling". In: *Journal of Applied Physiology* 127.4, pp. 1165–1174.

Hull, J. J. (1994). "A database for handwritten text recognition research". In: *TPAMI* 16.5, pp. 550–554.

Hunt, E. B. (1962). "Concept learning: An information processing problem." In.

Inoue, M., Inoue, S., and Nishida, T. (2018). "Deep recurrent neural network for mobile human activity recognition with high throughput". In: *Artificial Life and Robotics* 23.2, pp. 173–185.

Ivanovici, M. and Richard, N. (2010). "Fractal dimension of color fractal images". In: *IEEE Transactions on Image Processing* 20.1, pp. 227–235.

Jackson, P. (1986). "Introduction to expert systems". In.

Jansen, A., Dupoux, E., Goldwater, S., Johnson, M., Church, K., Kenneth, S., Feldman, N., Hermansky, H., Metze, F., Rose, R., Clark, S., Pascal, M., McGraw, I., Varadarajan, B., Bennett, E., Chiu, B., Justin, B., Dunbar, E., Fourtassi, A., Harwath, D., Lee, C.-y., Norouzian, L., Atta, K., Peddinti, V., Richardson, R., Schatz, T., and Thomas, S. (2013). "A summary of the 2012 JHU CLSP workshop on zero resource speech technologies and models of early language acquisition". In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, pp. 8111–8115.

Jiang, W. and Yin, Z. (2015). "Human activity recognition using wearable sensors by deep convolutional neural networks". In: *Proceedings of the 23rd ACM International Conference on Multimedia*, pp. 1307–1310.

Jiang, Z., Zheng, Y., Tan, H., Tang, B., and Zhou, H. (2016). "Variational deep embedding: An unsupervised and generative approach to clustering". In: *ArXiv:1611.05148*.

Johnson, J., Hariharan, B., Maaten, L. v. d., Fei-Fei, L., Zitnick, C. L., and Girshick, R. B. (2016). "CLEVR: A Diagnostic dataset for compositional language and elementary visual reasoning". In: *Proceedings of the IEEE / CVF Computer Vision and Pattern Recognition Conference*, pp. 1988–1997.

Jothi, R. (2018). "Clustering time-series data generated by smart devices for human activity recognition". In: *Proceedings of the International Conference on Intelligent Systems Design and Applications*. Springer, pp. 708–716.

Kahneman, D. (2011). *Thinking, Fast and Slow*. Macmillan.

Karras, T., Laine, S., and Aila, T. (2019). "A Style-Based Generator Architecture for Generative Adversarial Networks". In: *Proceedings of the IEEE / CVF Computer Vision and Pattern Recognition Conference*.

Khan, A. M., Lee, Y.-K., Lee, S.-Y., and Kim, T.-S. (2010). "Human activity recognition via an accelerometer-enabled-smartphone using kernel discriminant analysis". In: *Proceedings of the Fifth International Conference on Future Information Technology*. IEEE, pp. 1–6.

Khan, T. M., Naqvi, S. S., and Meijering, E. (2021). "Leveraging image complexity in macro-level neural network design for medical image segmentation". In: *arXiv preprint ArXiv:2112.11065*.

Kingma, D. P. and Ba, J. (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint ArXiv:1412.6980*.

Kingma, D. P. and Welling, M. (2013). “Auto-encoding variational bayes”. In: *arXiv preprint ArXiv:1312.6114*.

Kinsey, M., Galea, E., and Lawrence, P. (2012). “Human factors associated with the selection of lifts/elevators or stairs in emergency and normal usage conditions”. In: *Fire Technology* 48.1, pp. 3–26.

Kittler, J., Hatef, M., Duin, R. P., and Matas, J. (1998). “On combining classifiers”. In: *TPAMI* 20.3, pp. 226–239.

Kramer, M. A. (1991). “Nonlinear principal component analysis using autoassociative neural networks”. In: *AIChE Journal* 37.2, pp. 233–243.

Krishna, V. and Ganapathy, S. (2022). “Self supervised representation learning with deep clustering for acoustic unit discovery from raw speech”. In: *Proceedings of the 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 3268–3272.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). “Imagenet classification with deep convolutional neural networks”. In: *Advances in Neural Information Processing Systems* 25.

Kuhl, P. K. (2004). “Early language acquisition: cracking the speech code”. In: *Nature Reviews Neuroscience* 5.11, pp. 831–843.

Kuhn, H. W. (1955). “The Hungarian method for the assignment problem”. In: *Naval Research Logistics Quarterly* 2.1-2, pp. 83–97.

Kuhn, H. W. (1956). “Variants of the Hungarian method for assignment problems”. In: *Naval Research Logistics Quarterly* 3.4, pp. 253–258.

Kulshreshtha, P. and Guha, T. (2018). “An online algorithm for constrained face clustering in videos”. In: *Proceedings of the Twenty-fifth IEEE International Conference on Image Processing (ICIP)*. IEEE, pp. 2670–2674.

Kumar, S., Haresh, S., Ahmed, A., Konin, A., Zia, M. Z., and Tran, Q.-H. (2021). “Unsupervised Activity Segmentation by Joint Representation Learning and Online Clustering”. In: *arXiv preprint ArXiv:2105.13353*.

Kunze, K. and Lukowicz, P. (2008). “Dealing with sensor displacement in motion-based on-body activity recognition systems”. In: *Proceedings of the 10th International Conference on Ubiquitous Computing*, pp. 20–29.

Kunze, K. and Lukowicz, P. (2014). “Sensor placement variations in wearable activity recognition”. In: *IEEE Pervasive Computing* 13.4, pp. 32–41.

Kwapisz, J. R., Weiss, G. M., and Moore, S. A. (2011). “Activity recognition using cell phone accelerometers”. In: *ACM SigKDD Explorations Newsletter* 12.2, pp. 74–82.

Kwon, Y., Kang, K., and Bae, C. (2014). “Unsupervised learning for human activity recognition using smartphone sensors”. In: *Expert Systems with Applications* 41.14, pp. 6067–6074.

Lam, N. S.-N., Qiu, H.-l., Quattrochi, D. A., and Emerson, C. W. (2002). “An evaluation of fractal methods for characterizing image complexity”. In: *Cartography and Geographic Information Science* 29.1, pp. 25–35.

Landgrebe, D. A. and Malaret, E. (1986). “Noise in remote-sensing systems: The effect on classification error”. In: *IEEE Transactions on Geoscience and Remote Sensing* 2, pp. 294–300.

Le, Q. and Mikolov, T. (2014). “Distributed representations of sentences and documents”. In: *Proceedings of the Thirty-first International Conference on Machine Learning*, pp. 1188–1196.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). “Deep learning”. In: *Nature* 521.7553, pp. 436–444.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

Lei, J., Yu, L., Bansal, M., and Berg, T. L. (2018). “TVQA: Localized, compositional video question answering”. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 1369–1379.

Levelt, W. J. (1993). *Speaking: From Intention to Articulation*. Vol. 1. MIT press.

Li, F. and Dustdar, S. (2011). “Incorporating unsupervised learning in activity recognition”. In: *Workshop at the Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Li, J., Zhou, P., Xiong, C., and Hoi, S. C. (2020). “Prototypical contrastive learning of unsupervised representations”. In: *arXiv preprint ArXiv:2005.04966*.

Li, Y., Shi, D., Ding, B., and Liu, D. (2014). “Unsupervised feature learning for human activity recognition using smartphone sensors”. In: *Proceedings of the Intelligence and Knowledge Exploration*. Springer, pp. 99–107.

Liang, J., Yang, J., Lee, H.-Y., Wang, K., and Yang, M.-H. (2018). “Sub-GAN: An Unsupervised generative model via subspaces”. In: *Proceedings of the European Conference on Computer Vision*.

Lipton, Z. C. and Tripathi, S. (2017). “Precise recovery of latent vectors from generative adversarial networks”. In: *ArXiv:1702.04782*.

Lu, Y., Wei, Y., Liu, L., Zhong, J., Sun, L., and Liu, Y. (2017). “Towards unsupervised physical activity recognition using smartphone accelerometers”. In: *Multimedia Tools and Applications* 76.8, pp. 10701–10719.

Ma, H., Zhang, Z., Li, W., and Lu, S. (2021). “Unsupervised Human Activity Representation Learning with Multi-task Deep Clustering”. In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5.1, pp. 1–25.

Machado, I. P., Gomes, A. L., Gamboa, H., Paixão, V., and Costa, R. M. (2015). “Human activity data discovery from triaxial accelerometer sensor: Non-supervised learning sensitivity to feature extraction parametrization”. In: *Information Processing & Management* 51.2, pp. 204–214.

Machado, P., Romero, J., Nadal, M., Santos, A., Correia, J., and Carballal, A. (2015). “Computerized measures of visual complexity”. In: *Acta Psychologica* 160, pp. 43–57.

Madrid-Herrera, L., Chacon-Murguia, M. I., Posada-Urrutia, D. A., and Ramirez-Quintana, J. A. (2019). “Human image complexity analysis using a fuzzy inference system”. In: *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, pp. 1–6.

Mahon, L., Giunchiglia, E., Li, B., and Lukasiewicz, T. (2020). “Knowledge graph extraction from videos”. In: *Proceedings of The Ninth IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, pp. 25–32.

Mahon, L. and Lukasiewicz, T. (2021). “Selective pseudo-label clustering”. In: *Proceedings of the Conference on Artificial Intelligence (Künstliche Intelligenz)*. Springer, pp. 158–178.

Mahon, L. and Lukasiewicz, T. (2022). “Efficient deep clustering of human activities and how to improve evaluation”. In: *arxiv preprint arXiv:2209.08335*.
Mahon, L. and Lukasiewicz, T. (2023a). “Hard Regularization to Prevent Collapse in Online Deep Clustering without Data Augmentation”. In: *arXiv preprint arXiv:2303.16521*.
Mahon, L. and Lukasiewicz, T. (2023b). “Minimum Description Length Clustering to Measure Meaningful Image Complexity”. In: *Available at SSRN 4391368*.
Mahon, L., Shah, L., and Lukasiewicz, T. (2023). “Correcting Flaws in Common Disentanglement Metrics”. In: *arXiv preprint arXiv:2304.02335*.
Mahon, L. and Vogel, C. (2022). “The Proof is in the Pudding: Using Automated Theorem Proving to Generate Cooking Recipes”. In: *arXiv preprint arXiv:2203.02683*.
Malekzadeh, M., Clegg, R. G., Cavallaro, A., and Haddadi, H. (2018). “Protecting sensory data against sensitive inferences”. In: *Proceedings of the 1st Workshop on Privacy by Design in Distributed Systems*, pp. 1–6.
Mandelbrot, B. (1967). “How long is the coast of Britain? Statistical self-similarity and fractional dimension”. In: *Science* 156.3775, pp. 636–638.
Mandelbrot, B. B. and Mandelbrot, B. B. (1982). *The Fractal Geometry of Nature*. Vol. 1. WH freeman New York.
Marin, M. M. and Leder, H. (2013). “Examining complexity across domains: Relating subjective and objective measures of affective environmental scenes, paintings and music”. In: *PloS One* 8.8, e72412.
Marshall, S. M., Mathis, C., Carrick, E., Keenan, G., Cooper, G. J. T., Graham, H., Craven, M., Gromski, P. S., Moore, D. G., Walker, S., and Cronin, L. (2021). “Identifying molecules as biosignatures with assembly theory and mass spectrometry”. In: *Nature Communications* 12.1, pp. 1–9.
Marshall, S. M., Moore, D., Murray, A. R. G., Walker, S. I., and Cronin, L. (2019). “Quantifying the pathways to life using assembly spaces”. In: *arXiv preprint ArXiv:1907.04649*.
McConnell, M., Turakhia, M., Harrington, R., King, A., and Ashley, E. (2018). “Mobile health advances in physical activity, fitness, and atrial fibrillation: moving hearts”. In: *Journal of the American College of Cardiology* 71.23, pp. 2691–2701.
McConville, R., Santos-Rodriguez, R., Piechocki, R. J., and Craddock, I. (2019). “N2d:(Not too) deep clustering via clustering the local manifold of an autoencoded embedding”. In: *ArXiv:1908.05968*.
McConville, R., Santos-Rodriguez, R., Piechocki, R. J., and Craddock, I. (2021). “N2d:(not too) deep clustering via clustering the local manifold of an autoencoded embedding”. In: *Proceedings of the Twenty-fifth International Conference on Pattern Recognition (ICPR)*. IEEE, pp. 5145–5152.
McInnes, L., Healy, J., and Astels, S. (2017). “HDBSCAN: Hierarchical density based clustering”. In: *Journal of Open Source Software* 2.11, p. 205.
McInnes, L., Healy, J., and Melville, J. (2018). “UMAP: Uniform manifold approximation and projection for dimension reduction”. In: *ArXiv:1802.03426*.
McKeown, K. R. (1985). “Discourse strategies for generating natural-language text”. In: *Artificial Intelligence* 27.1, pp. 1–41.
Mejia-Ricart, L. F., Helling, P., and Olmsted, A. (2017). “Evaluate action primitives for human activity recognition using unsupervised learning approach”. In: *Proceedings of*

the Twelfth International Conference for Internet Technology and Secured Transactions (ICITST). IEEE, pp. 186–188.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). "Efficient estimation of word representations in vector space". In: *arXiv preprint ArXiv:1301.3781*.

Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K. J. (1990). "Introduction to WordNet: An on-line lexical database". In: *International Journal of Lexicography* 3.4, pp. 235–244.

Minsky, M. L. (1991). "Logical versus analogical or symbolic versus connectionist or neat versus scruffy". In: *AI Magazine* 12.2, pp. 34–34.

Mitchell, T. M. and Mitchell, T. M. (1997). *Machine learning*. Vol. 1. 9. McGraw-hill New York.

Mohmed, G., Lotfi, A., Langensiepen, C., and Pourabdollah, A. (2018). "Clustering-based fuzzy finite state machine for human activity recognition". In: *Proceedings of the UK Workshop on Computational Intelligence*. Springer, pp. 264–275.

Moravec, H. (1988). *Mind Children: The Future of Robot and Human Intelligence*. Harvard University Press.

Mrabah, N., Bouguessa, M., and Ksantini, R. (2019). "Adversarial deep embedded clustering: On a better trade-off between feature randomness and feature drift". In: *ArXiv:1909.11832*.

Mrabah, N., Khan, N. M., Ksantini, R., and Lachiri, Z. (2019). "Deep Clustering with a dynamic autoencoder: From reconstruction towards centroids construction". In: *ArXiv:1901.07752*.

Mrabah, N., Khan, N. M., Ksantini, R., and Lachiri, Z. (2020). "Deep clustering with a dynamic autoencoder: From reconstruction towards centroids construction". In: *Neural Networks* 130, pp. 206–228.

Mukherjee, S., Asnani, H., Lin, E., and Kannan, S. (2019). "Clustergan: Latent space clustering in generative adversarial networks". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01, pp. 4610–4617.

Munkres, J. (1957). "Algorithms for the assignment and transportation problems". In: *Journal of the Society for Industrial and Applied Mathematics* 5.1, pp. 32–38.

Murad, A. and Pyun, J.-Y. (2017). "Deep recurrent neural networks for human activity recognition". In: *Sensors* 17.11, p. 2556.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Nagle, F. and Lavie, N. (2020). "Predicting human complexity perception of real-world scenes". In: *Royal Society Open Science* 7.5, p. 191487.

Narayanan, R. M., Ponnappan, S. K., and Reichenbach, S. E. (2003). "Effects of noise on the information content of remote sensing images". In: *Geocarto International* 18.2, pp. 15–26.

Nicolae, I. E. and Ivanovici, M. (2020). "Preparatory experiments regarding human brain perception and reasoning of image complexity for synthetic color fractal and natural texture images via EEG". In: *Applied Sciences* 11.1, p. 164.

Niu, C., Shan, H., and Wang, G. (2021). "Spice: Semantic pseudo-labeling for image clustering". In: *arXiv preprint ArXiv:2103.09382*.

Niu, C., Zhang, J., Wang, G., and Liang, J. (2020). "Gatcluster: Self-supervised gaussian-attention network for image clustering". In: *Proceedings of the European Conference on Computer Vision*. Springer, pp. 735–751.

Niu, D., Dy, J., and Jordan, M. I. (2011). “Dimensionality reduction for spectral clustering”. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, pp. 552–560.

Norretranders, T., Faerch-Jensen, H., and Wahlen, J. (1993). *Märk Världen: En Bok Om Vetenskap Och Intuition*. Bonnier Alba.

Nye, M., Tessler, M., Tenenbaum, J., and Lake, B. M. (2021). “Improving coherence and consistency in neural sequence models with dual-system, neuro-symbolic reasoning”. In: *Advances in Neural Information Processing Systems* 34.

Olivastri, S., Singh, G., and Cuzzolin, F. (2019). “An end-to-end baseline for video captioning”. In: *arXiv preprint ArXiv:1904.02628*.

OŃeil, A. Q., Kascenas, A., Henry, J., Wyeth, D., Shepherd, M., Beveridge, E., Clunie, L., Sansom, C., Seduikyte Keith Muir, E., and Poole, I. (2018). “Attaining human-level performance with atlas location autocontext for anatomical landmark detection in 3D CT data”. In: *Workshop at the European Conference on Computer Vision*, p. 5.

Opitz, D. W. and Maclin, R. F. (1997). “An empirical evaluation of bagging and boosting for artificial neural networks”. In: *Proceedings of the ICNN*. Vol. 3. IEEE, pp. 1401–1405.

Orzan, G., Zara, I., and Purcarea, V. (2012). “Neuromarketing techniques in pharmaceutical drugs advertising. A discussion and agenda for future research”. In: *Journal of Medicine and Life* 5.4, p. 428.

Pan, Y., Mei, T., Yao, T., Li, H., and Rui, Y. (2016). “Jointly modeling embedding and translation to bridge video and language”. In: *Proceedings of the IEEE / CVF Computer Vision and Pattern Recognition Conference*, pp. 4594–4602.

Pan, Y., Yao, T., Li, H., and Mei, T. (2017). “Video captioning with transferred semantic attributes”. In: *Proceedings of the IEEE / CVF Computer Vision and Pattern Recognition Conference*, pp. 6504–6512.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). “BLEU: A method for automatic evaluation of machine translation”. In: *Proceedings of the Association for Computationl Linguistics*, pp. 311–318.

Pearlmutter, B. A. and Rosenfeld, R. (1991). “Chaitin-Kolmogorov complexity and generalization in neural networks”. In: *Proceedings of the Fifth Conference on Neural Information Processing Systems*, pp. 925–931.

Perrone, M. P. (1993). “Improving regression estimation: Averaging methods for variance reduction with extensions to general convex measure optimization”. PhD thesis.

Peters, R. A. and Strickland, R. N. (1990). “Image complexity metrics for automatic target recognizers”. In: *Proceedings of the Automatic Target Recognizer System and Technology Conference*. Citeseer, pp. 1–17.

Qi, P., Dozat, T., Zhang, Y., and Manning, C. D. (2018). “Universal dependency parsing from scratch”. In: *Proceedings of the CoNLL*, pp. 160–170.

Qin, Z., Zhang, Y., Meng, S., Qin, Z., and Choo, K.-K. R. (2020). “Imaging and fusing time series for wearable sensor-based human activity recognition”. In: *Information Fusion* 53, pp. 80–87.

Rademaker, A. and Tyers, F. (2019). *Proceedings of the 3rd Workshop on Universal Dependencies (UDW, SyntaxFest 2019)*.

Rajpurkar, P., Hannun, A. Y., Haghpanahi, M., Bourn, C., and Ng, A. Y. (2017). "Cardiologist-level arrhythmia detection with convolutional neural networks". In: *arXiv preprint ArXiv:1707.01836*.

Rand, W. M. (1971). "Objective criteria for the evaluation of clustering methods". In: *Journal of the American Statistical Association* 66.336, pp. 846–850.

Rasti, B., Scheunders, P., Ghamisi, P., Licciardi, G., and Chanussot, J. (2018). "Noise reduction in hyperspectral imagery: Overview and application". In: *Remote Sensing* 10.3, p. 482.

Redies, C., Amirshahi, S. A., Koch, M., and Denzler, J. (2012). "PHOG-derived aesthetic measures applied to color photographs of artworks, natural scenes and objects". In: *Proceedings of the European Conference on Computer Vision*. Springer, pp. 522–531.

Reiss, A. and Stricker, D. (2012). "Introducing a new benchmarked dataset for activity monitoring". In: *Proceedings of the Sixteenth International Symposium on Wearable Computers*. IEEE, pp. 108–109.

Ren, Y., Wang, N., Li, M., and Xu, Z. (2020). "Deep density-based image clustering". In: *Knowledge-Based Systems*, p. 105841.

Rissanen, J. (1983). "A universal prior for integers and estimation by minimum description length". In: *The Annals of Statistics* 11.2, pp. 416–431.

Ronao, C. A. and Cho, S.-B. (2015). "Deep convolutional neural networks for human activity recognition with smartphone sensors". In: *Proceedings of the International Conference on Neural Information Processing*. Springer, pp. 46–53.

Rousseeuw, P. J. (1987). "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis". In: *Journal of Computational and Applied Mathematics* 20, pp. 53–65.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). "Imagenet large scale visual recognition challenge". In: *International Journal of Computer Vision* 115.3, pp. 211–252.

Sauri, R., Mahon, L., Russo, I., and Bitinis, M. (2019). "Cross-dictionary linking at sense level with a double-layer classifier". In: *2nd Conference on Language, Data and Knowledge (LDK 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Schwieterman, E. W., Kiang, N. Y., Parenteau, M. N., Harman, C. E., DasSarma, S., Fisher, T. M., Arney, G. N., Hartnett, H. E., Reinhard, C. T., Olson, S. L., Meadows, V. S., Cockell, C. S., Walker, S. I., Grenfell, J. L., Hegde, S., Rugheimer, S., Hu, R., and Lyons, T. W. (2018). "Exoplanet biosignatures: A review of remotely detectable signs of life". In: *Astrobiology* 18.6, pp. 663–708.

Sebastian V., B., Unnikrishnan, A., and Balakrishnan, K. (2012). "Gray level co-occurrence matrices: generalisation and some new features". In: *arXiv preprint ArXiv:1205.4831*.

Sheng, T. and Huber, M. (2020). "Unsupervised embedding learning for human activity recognition using wearable sensor data". In: *Proceedings of the Thirty-Third International Flairs Conference*.

Sherman, A., Lim, S. Y., Grabowecky, M., and Suzuki, S. (2013). "Visual-object working memory affects aesthetic judgments". In: *Journal of Vision* 13.9, pp. 1308–1308.

Simonyan, K. and Zisserman, A. (2014). "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint ArXiv:1409.1556*.

Singh, D., Merdivan, E., Psychoula, I., Kropf, J., Hanke, S., Geist, M., and Holzinger, A. (2017). "Human activity recognition using recurrent neural networks". In: *Proceedings of the International Cross-domain Conference for Machine Learning and Knowledge Extraction*. Springer, pp. 267–274.

Singh, T. and Vishwakarma, D. K. (2019). "Human activity recognition in video benchmarks: A survey". In: *Advances in Signal Processing and Communication*, pp. 247–259.

Smolensky, P. (1986). *Information processing in dynamical systems: Foundations of harmony theory*. Tech. rep. Colorado Univ at Boulder Dept of Computer Science.

Song, M., Yang, H., Siadat, S. H., and Pechenizkiy, M. (2013). "A comparative study of dimensionality reduction techniques to enhance trace clustering performances". In: *Expert Systems with Applications* 40.9, pp. 3722–3737.

Stickel, C., Ebner, M., and Holzinger, A. (2010). "The XAOS metric–understanding visual complexity as measure of usability". In: *Proceedings of ths Symposium of the Austrian HCI and Usability Engineering Group*. Springer, pp. 278–290.

Stisen, A., Blunck, H., Bhattacharya, S., Prentow, T. S., Kjærgaard, M. B., Dey, A., Sonne, T., and Jensen, M. M. (2015). "Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition". In: *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pp. 127–140.

Strehl, A. and Ghosh, J. (2002). "Cluster ensembles—a knowledge reuse framework for combining multiple partitions". In: *Journal of Machine Learning Research* 3.Dec, pp. 583–617.

Suh, S., Rey, V. F., and Lukowicz, P. (2021). "Adversarial deep feature extraction network for user independent human activity recognition". In: *arXiv preprint ArXiv:2110.12163*.

Sun, W., Xu, G., Gong, P., and Liang, S. (2006). "Fractal analysis of remotely sensed images: A review of methods and applications". In: *International Journal of Remote Sensing* 27.22, pp. 4963–4990.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). "Going deeper with convolutions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826.

Tapia, E., Intille, S., and Larson, K. (2004). "Activity recognition in the home using simple and ubiquitous sensors". In: *Proceedings of the International Conference on Pervasive Computing*. Springer, pp. 158–175.

Tasoulis, S., Pavlidis, N. G., and Roos, T. (2020). "Nonlinear dimensionality reduction for clustering". In: *Pattern Recognition* 107, p. 107508.

Thompson, R. H., McKerchar, P. M., and Dancho, K. A. (2004). "The effects of delayed physical prompts and reinforcement on infant sign language acquisition". In: *Journal of Applied Behavior Analysis* 37.3, pp. 379–383.

Tomizawa, N. (1971). "On some techniques useful for solution of transportation network problems". In: *Networks* 1.2, pp. 173–194.

Trabelsi, D., Mohammed, S., Chamroukhi, F., Oukhellou, L., and Amirat, Y. (2013). "An unsupervised approach for automatic activity recognition based on hidden Markov

model regression". In: *IEEE Transactions on Automation Science and Engineering* 10.3, pp. 829–835.

Van Gansbeke, W., Vandenhende, S., Georgoulis, S., Proesmans, M., and Van Gool, L. (2020). "Scan: Learning to classify images without labels". In: *Proceedings of the European Conference on Computer Vision*. Springer, pp. 268–285.

Vasile, D. and Lukasiewicz, T. (2018). "Learning structured video descriptions: Automated video knowledge extraction for video understanding tasks". In: *On the Move to Meaningful Internet Systems. OTM 2018 Conferences*, pp. 315–332.

Verwijmeren, T., Karremans, J. C., Stroebe, W., and Wigboldus, D. H. (2011). "The workings and limits of subliminal advertising: The role of habits". In: *Journal of Consumer Psychology* 21.2, pp. 206–213.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A., and Bottou, L. (2010). "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." In: *Journal of Machine Learning Research* 11.12.

Vorberg, D., Mattler, U., Heinecke, A., Schmidt, T., and Schwarzbach, J. (2003). "Different time courses for visual perception and action priming". In: *Proceedings of the National Academy of Sciences* 100.10, pp. 6275–6280.

Wang, B., Ma, L., Zhang, W., and Liu, W. (2018). "Reconstruction network for video captioning". In: *Proceedings of the IEEE / CVF Computer Vision and Pattern Recognition Conference*, pp. 7622–7631.

Wang, J., Chen, Y., Hao, S., Peng, X., and Hu, L. (2019). "Deep learning for sensor-based activity recognition: A survey". In: *Pattern Recognition Letters* 119, pp. 3–11.

Wang, P., Wu, Q., Shen, C., Dick, A., and Van Den Henge, A. (2017). "Explicit knowledge-based reasoning for visual question answering". In: *Proceedings of the Twenty-eighth International Joint Conference on Artificial Intelligence*, pp. 1290–1296. ISBN: 978-0-9992411-0-3.

Wang, P., Wu, Q., Shen, C., Dick, A. R., and Hengel, A. v. d. (2016). "FVQA: Fact-Based Visual Question Answering". In: *IEEE TPAMI* 40, pp. 2413–2427.

Wang, X.-T., Ma, W.-c., Zhang, K., and Yan, J. (2018). "Complexity metric of infrared image for automatic target recognition". In: *Proceedings of the Third International Conference on Computational Intelligence and Applications (ICCIA)*. IEEE, pp. 175–180.

Wang, Y., Cang, S., and Yu, H. (2019). "A survey on wearable sensor modality centred human activity recognition in health care". In: *Expert Systems with Applications* 137, pp. 167–190.

Wang, Y., Zhang, L., Nie, F., Li, X., Chen, Z., and Wang, F. (2019). "WeGAN: Deep Image Hashing with Weighted Generative Adversarial Networks". In: *IEEE Trans. Multimed.*

Waxman, S. R. and Gelman, S. A. (2009). "Early word-learning entails reference, not merely associations". In: *Trends in Cognitive Sciences* 13.6, pp. 258–263.

Weiss, G. M., Yoneda, K., and Hayajneh, T. (2019). "Smartphone and smartwatch-based biometrics using activities of daily living". In: *IEEE Access* 7, pp. 133190–133202.

Wu, J., Long, K., Wang, F., Qian, C., Li, C., Lin, Z., and Zha, H. (2019). "Deep comprehensive correlation mining for image clustering". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8150–8159.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms". In: *ArXiv:1708.07747*.

Xie, J., Girshick, R., and Farhadi, A. (2016). "Unsupervised deep embedding for clustering analysis". In: *Proceedings of the Thirty-third International Conference on Machine Learning*, pp. 478–487.

Xu, J., Mei, T., Yao, T., and Rui, Y. (2016). "MSR-VTT: A large video description dataset for bridging video and language". In: *Proceedings of the IEEE / CVF Computer Vision and Pattern Recognition Conference*, pp. 5288–5296.

Xu, J., Yao, T., Zhang, Y., and Mei, T. (2017). "Learning multimodal attention LSTM networks for video captioning". In: *Proceedings of the ACM*, pp. 537–545.

Yang, B., Fu, X., Sidiropoulos, N. D., and Hong, M. (2017). "Towards k-means-friendly spaces: Simultaneous deep learning and clustering". In: *Proceedings of the Thirty-fourth International Conference on Machine Learning*. JMLR.org, pp. 3861–3870.

Yang, J., Nguyen, M. N., San, P. P., Li, X. L., and Krishnaswamy, S. (2015). "Deep convolutional neural networks on multichannel time series for human activity recognition". In: *Proceedings of the Twenty-fourth International Joint Conference on Artificial Intelligence*.

Yang, J., Parikh, D., and Batra, D. (2016). "Joint unsupervised learning of deep representations and image clusters". In: *Proceedings of the IEEE / CVF Computer Vision and Pattern Recognition Conference*, pp. 5147–5156.

Yang, X. and Zhou, C. (2000). "Analysis of the complexity of remote sensing image and its role on image classification". In: *Proceedings of the 2000. IEEE 2000 International Geoscience and Remote Sensing Symposium. Taking the Pulse of the Planet: The Role of Remote Sensing in Managing the Environment. Proceedings (Cat. No. 00CH37120)*. Vol. 5. IEEE, pp. 2179–2181.

Yin, J. and Csibra, G. (2015). "Concept-based word learning in human infants". In: *Psychological Science* 26.8, pp. 1316–1324.

Yu, H., Wang, J., Huang, Z., Yang, Y., and Xu, W. (2016). "Video paragraph captioning using hierarchical recurrent neural networks". In: *Proceedings of the IEEE / CVF Computer Vision and Pattern Recognition Conference*, pp. 4584–4593.

Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. (2021). "Barlow twins: Self-supervised learning via redundancy reduction". In: *Proceedings of the International Conference on Machine Learning*. PMLR, pp. 12310–12320.

Zeiler, M. D. and Fergus, R. (2014). "Visualizing and understanding convolutional networks". In: *Proceedings of the European Conference on Computer Vision*. Springer, pp. 818–833.

Zeithamova, D., Mack, M. L., Braunlich, K., Davis, T., Seger, C. A., Kesteren, M. T. van, and Wutz, A. (2019). "Brain mechanisms of concept learning". In: *Journal of Neuroscience* 39.42, pp. 8259–8266.

Zemel, R. S. and Hinton, G. E. (1994). "Developing population codes by minimizing description length". In: *Proceedings of the Eigth Conference on Neural Information Processing Systems*, pp. 11–18.

Zeng, K., Chen, T., Chuang, C., Liao, Y., Niebles, J. C., and Sun, M. (2017). "Leveraging Video Descriptions to Learn Video Question Answering". In: *Proceedings of the Thirty-first AAAI Conference on Artificial Intelligence*, pp. 4334–4340.

Zeng, M., Nguyen, L. T., Yu, B., Mengshoel, O. J., Zhu, J., Wu, P., and Zhang, J. (2014). "Convolutional neural networks for human activity recognition using mobile sensors". In:

Proceedings of the Sixth International Conference on Mobile Computing, Applications and Services. IEEE, pp. 197–205.

Zhan, X., Xie, J., Liu, Z., Ong, Y.-S., and Loy, C. C. (2020). "Online deep clustering for unsupervised representation learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6688–6697.

Zhang, D., Nan, F., Wei, X., Li, S., Zhu, H., McKeown, K., Nallapati, R., Arnold, A., and Xiang, B. (2021). "Supporting clustering with contrastive learning". In: *arXiv preprint ArXiv:2103.12953*.

Zhang, J. and Peng, Y. (2019). "Hierarchical Vision-Language Alignment for Video Captioning". In: *MultiMedia Modeling*. Springer, pp. 42–54. ISBN: 978-3-030-05710-7.

Zhang, M. and Sawchuk, A. A. (2011). "A feature selection-based framework for human activity recognition using wearable multimodal sensors." In: *Proceedings of the Sixth EAI International Conference on Body Area Networks*, pp. 92–98.

Zhang, X., Gao, K., Zhang, Y., Zhang, D., Li, J., and Tian, Q. (2017). "Task-driven dynamic fusion: Reducing ambiguity in video description". In: *Proceedings of the IEEE / CVF Computer Vision and Pattern Recognition Conference*, pp. 3713–3721.

Zhao, H., Gallo, O., Frosio, I., and Kautz, J. (2015). "Loss functions for neural networks for image processing". In: *ArXiv:1511.08861*.

Zhao, W., Wang, S., Xie, Z., Shi, J., and Xu, C. (2018). "GAN-EM: GAN based EM learning framework". In: *ArXiv:1812.00335*.

Zhao, Z., Yang, Q., Cai, D., He, X., and Zhuang, Y. (2017). "Video question answering via hierachical spatio-temporal attenion networks". In: *Proceedings of the Twenty-sixth International Joint Conference on Artificial Intelligence*, pp. 3518–3524.

Zhong, H., Chen, C., Jin, Z., and Hua, X.-S. (2020). "Deep robust clustering by contrastive learning". In: *arXiv preprint ArXiv:2008.03030*.

Zhou, L., Zhou, Y., Corso, J. J., Socher, R., and Xiong, C. (2018). "End-to-end dense video captioning with masked transformer". In: *Proceedings of the IEEE / CVF Computer Vision and Pattern Recognition Conference*, pp. 8739–8748.

Zhou, P., Hou, Y., and Feng, J. (2018). "Deep adversarial subspace clustering". In: *Proceedings of the IEEE / CVF Computer Vision and Pattern Recognition Conference*.

Appendix A

Online Hard Clustering Appendices

A.1 Modified Variance Maximization

As discussed in Section 5.4, one of the methods we compare to is that proposed by H. Zhong, C. Chen, Z. Jin, and Hua, 2020, which minimizes the sum of squares of the marginal soft assignments across a batch. The expectation of the square (and hence the sum of squares) can be decomposed as the square of the expectation plus the variance. Minimizing the sum of squares can then help to combat partition collapse as it involves minimizing the variance. However, empirically we find this method not to perform well, see Table 5.1. Here, we show that a simply modified version of this method performs better than the original, though still less well than our method. Results are presented in Table A.1.

The modification is to just minimize variance directly, rather than via sum of squares. Note that this may be equivalent in some formulations, if the probability of membership across clusters for a single data point is normalized to sum to 1. Then the expectation of the sum of memberships is 1, because it is 1 deterministically, so the square of the expectation is also 1 deterministically and, in particular, is independent of the cluster assignments. This means that minimizing the sum of squares with respect to cluster assignments, is identical to minimizing variance with respect to cluster assignments. In our model this is true. The probability of membership depends only on the distance to the cluster centroids, and is conditionally independent across clusters, given the cluster centroids. Details are not given in H. Zhong, C. Chen, Z. Jin, and Hua, 2020 as to whether this holds in their method.

		CA	Var	VarM
Cifar10	Acc	**22.7 (2.07)**	11.8 (1.72)	20.8 (0.67)
	NMI	10.1 (1.68)	1.1 (1.15)	**11.2 (0.65)**
	ARI	5.8 (0.93)	0.3 (0.38)	**7.3 (0.50)**
	HVar	**425 (136)**	43542 (11424)	11515 (1524)
	SVar	407 (135)	**339 (92)**	10028 (1407)
	HEnt	**3.3 (0.01)**	0.9 (1.22)	1.3 (0.09)
	SEnt	**3.3 (0.01)**	2.6 (0.97)	1.5 (0.08)
Cifar100	Acc	**6.4 (0.22)**	1.2 (0.22)	1.0 (0.00)
	NMI	**13.2 (0.37)**	0.6 (1.05)	0.0 (0.00)
	ARI	**1.7 (0.14)**	0.0 (0.04)	0.0 (0.00)
	HVar	**1280 (156)**	55405 (3743)	59400 (0)
	SVar	190 (21)	**121 (62)**	12580 (391)
	HEnt	**5.4 (0.11)**	0.2 (0.17)	0.0 (0.00)
	SEnt	6.5 (0.17)	**6.5 (0.06)**	3.1 (0.03)
FashionMNIST	Acc	**54.5 (6.96)**	10.0 (0.04)	37.4 (2.53)
	NMI	**53.2 (4.23)**	0.0 (0.04)	42.8 (2.08)
	ARI	**39.1 (6.29)**	0.0 (0.00)	27.1 (1.78)
	HVar	**386 (51)**	53950 (98)	8550 (1001)
	SVar	**368 (40)**	376 (172)	6072 (3066)
	HEnt	**3.3 (0.01)**	0.0 (0.01)	1.5 (0.07)
	SEnt	**3.3 (0.00)**	3.1 (0.40)	1.6 (0.06)
STL	Acc	**23.5 (1.42)**	10.1 (0.20)	22.6 (1.52)
	NMI	**13.7 (1.33)**	0.0 (0.08)	11.4 (1.70)
	ARI	7.1 (0.70)	0.0 (0.00)	**7.1 (1.16)**
	HVar	**217 (21)**	11317 (765)	1321 (384)
	SVar	**194 (17)**	524 (247)	949 (303)
	HEnt	**3.2 (0.02)**	0.1 (0.16)	1.7 (0.13)
	SEnt	**3.2 (0.01)**	3.1 (0.12)	1.9 (0.10)

Table A.1: Comparison between the modified variance minimization method, denoted 'VarM', the original variance minimization method from H. Zhong, C. Chen, Z. Jin, and Hua, 2020, denoted 'Var', and our method, denoted 'CA'.

Appendix B

Hierarchical Clustering to Measure Data Complexity Appendices

B.1 Worked Examples

Section 7.3.3 presented a single worked example of my image complexity metric on a single, randomly chosen image from ImageNet. This section contains a larger set of worked examples, with several images from ImageNet and others from Cifar, MNIST, and DTD2.

B.1.1 Imagenet Chainsaw Image

Figure B.1: Example of a relatively high resolution real-world image from imagenet. ID: n03000684_30692.

Layer 1

Num points to be clustered (pixels): 50000
Num components found by MDL: 8
Assign each pixel a label from $0, \ldots, 7$, and form patch signatures as multisets of labels

inside all patches of size 4×4.
Num patch signatures: 48216
Num unique patch signatures: 1801
Entropy of categorical distribution of patch signatures: **6.42**

Layer 2

Num points to be clustered: 48216
Num components found by MDL: 8
Assign each point a label from $0, \ldots, 7$, and form patch signatures as multisets of labels inside all patches of size 8×8.
Num patch signatures: 44744
Num unique patch signatures: 3278
Entropy of categorical distribution of patch signatures: **7.32**

Layer 3

Num points to be clustered: 44744
Num components found by MDL: 8
Assign each point a label from $0, \ldots, 7$, and form patch signatures as multisets of labels inside all patches of size 16×16.
Num patch signatures: 38184
Num unique patch signatures: 7358
Entropy of categorical distribution of patch signatures: **11.07**

Layer 4

Num points to be clustered: 38184
Num components found by MDL: 8
Assign each point a label from $0, \ldots, 7$, and form patch signatures as multisets of labels inside all patches of size 32×32.
Num patch signatures: 26600
Num unique patch signatures: 7279
Entropy of categorical distribution of patch signatures: **12.73**

Total complexity: $6.42 + 7.32 + 11.07 + 12.73 = \mathbf{37.53}$

B.1.2 Imagenet Hang-Gliding Image

Figure B.2: Example of a relatively high resolution real-world image from imagenet. ID: n03888257_2122.

Layer 1

Num points to be clustered (pixels): 50112
Num components found by MDL: 8
Assign each pixel a label from $0, \ldots, 7$, and form patch signatures as multisets of labels inside all patches of size 4×4.
Num patch signatures: 48316
Num unique patch signatures: 1001
Entropy of categorical distribution of patch signatures: **4.55**

Layer 2

Num points to be clustered: 48316
Num components found by MDL: 8
Assign each point a label from $0, \ldots, 7$, and form patch signatures as multisets of labels inside all patches of size 8×8.
Num patch signatures: 44820
Num unique patch signatures: 5875
Entropy of categorical distribution of patch signatures: **9.44**

Layer 3

Num points to be clustered: 44820
Num components found by MDL: 8
Assign each pixel a label from $0, \ldots, 7$, and form patch signatures as multisets of labels inside all patches of size 16×16.
Num patch signatures: 38212
Num unique patch signatures: 7788
Entropy of categorical distribution of patch signatures: **11.52**

Layer 4

Num points to be clustered: 38212
Num components found by MDL: 8
Assign each pixel a label from $0, \ldots, 7$, and form patch signatures as multisets of labels inside all patches of size 32×32.
Num patch signatures: 26532
Num unique patch signatures: 7505
Entropy of categorical distribution of patch signatures: **12.77**

Total complexity: $4.55 + 9.44 + 11.52 + 12.77 = \mathbf{38.28}$

B.1.3 Cifar Cat Image

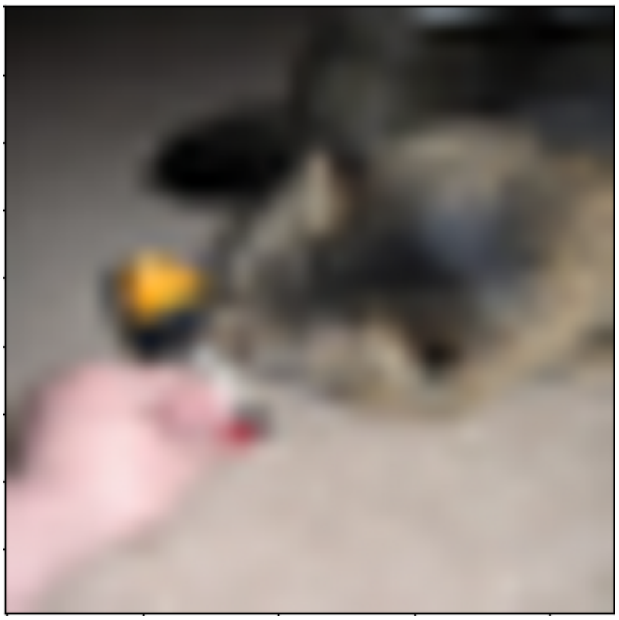

Figure B.3: Example of a low-resolution, real-world image from CIFAR10.

Layer 1

Num points to be clustered (pixels): 50176
Num components found by MDL: 8
Assign each pixel a label from $0, \ldots, 7$, and form patch signatures as multisets of labels inside all patches of size 4×4.
Num patch signatures: 48400
Num unique patch signatures: 231
Entropy of categorical distribution of patch signatures: **4.54**

Layer 2

Num points to be clustered: 48400
Num components found by MDL: 8
Assign each point a label from $0, \ldots, 7$, and form patch signatures as multisets of labels inside all patches of size 8×8.
Num patch signatures: 44944
Num unique patch signatures: 2355
Entropy of categorical distribution of patch signatures: **7.56**

Layer 3

Num points to be clustered: 44944
Num components found by MDL: 8
Assign each pixel a label from $0, \ldots, 7$, and form patch signatures as multisets of labels inside all patches of size 16×16.
Num patch signatures: 36416

Num unique patch signatures: 6374
Entropy of categorical distribution of patch signatures: **10.61**

Layer 4

Num points to be clustered: 38212
Num components found by MDL: 8
Assign each pixel a label from $0, \ldots, 7$, and form patch signatures as multisets of labels inside all patches of size 32×32.
Num patch signatures: 26896
Num unique patch signatures: 7336
Entropy of categorical distribution of patch signatures: **12.59**

Total complexity: $4.54 + 7.56 + 10.61 + 12.59 = \mathbf{35.31}$

B.1.4 MNIST Four Image

Figure B.4: Example of a low-resolution, greyscale handwritten digit from MNIST.

Layer 1

Num points to be clustered (pixels): 50176
Num components found by MDL: 8
Assign each pixel a label from $0, \ldots, 7$, and form patch signatures as multisets of labels inside all patches of size 4×4.
Num patch signatures: 48400
Num unique patch signatures: 417
Entropy of categorical distribution of patch signatures: **3.17**

Layer 2

Num points to be clustered: 48400
Num components found by MDL: 8
Assign each point a label from $0, \ldots, 7$, and form patch signatures as multisets of labels inside all patches of size 8×8.
Num patch signatures: 44944
Num unique patch signatures: 1776
Entropy of categorical distribution of patch signatures: **4.84**

Layer 3

Num points to be clustered: 44944
Num components found by MDL: 8
Assign each pixel a label from $0, \ldots, 7$, and form patch signatures as multisets of labels inside all patches of size 16×16.
Num patch signatures: 36416

Num unique patch signatures: 5499
Entropy of categorical distribution of patch signatures: **8.36**

Layer 4

Num points to be clustered: 38212
Num components found by MDL: 8
Assign each pixel a label from $0, \ldots, 7$, and form patch signatures as multisets of labels inside all patches of size 32×32.
Num patch signatures: 26896
Num unique patch signatures: 6730
Entropy of categorical distribution of patch signatures: **12.09**

Total complexity: $3.17 + 4.84 + 8.36 + 12.09 = \mathbf{28.46}$

B.1.5 DTD Fine Woven Texture

Figure B.5: Example of a detailed, high-resolution, repetitive pattern from DTD2.

Layer 1

Num points to be clustered (pixels): 50290
Num components found by MDL: 8
Assign each pixel a label from $0, \ldots, 7$, and form patch signatures as multisets of labels inside all patches of size 4×4.
Num patch signatures: 48400
Num unique patch signatures: 7568
Entropy of categorical distribution of patch signatures: **12.16**

Layer 2

Num points to be clustered: 48400
Num components found by MDL: 8
Assign each point a label from $0, \ldots, 7$, and form patch signatures as multisets of labels inside all patches of size 8×8.
Num patch signatures: 44944
Num unique patch signatures: 13356
Entropy of categorical distribution of patch signatures: **13.70**

Layer 3

Num points to be clustered: 44944
Num components found by MDL: 1
Assign each pixel a label from $0, \ldots, 7$, and form patch signatures as multisets of labels inside all patches of size 16×16.
Num patch signatures: 36416
Num unique patch signatures: 1

Entropy of categorical distribution of patch signatures: **0**

Layer 4

All patch signatures are identical because only one cluster was found at the previous level. So the entropy is **0**.

Total complexity: $12.16 + 13.70 + 0 + 0 = \mathbf{25.87}$

B.2 Datasets: Further Details

B.2.1 Synthetic Datasets

As described in Section 7.5, I created three synthetic datasets to help test my image complexity metric on a variety of images. The experimental results I report use 500 images sampled from these synthetic datasets. Here, I give the full details for the creation of these datasets. Code will also be released on publication.

Stripes These images depict a repeated striped black-and-white pattern. The thickness of the lines, in pixels, is sampled uniformly at random from $[3, 10]$, and the slope of the lines is sampled uniformly at random from $[-0.5, -1.5]$. It is sufficient to consider negative slopes only as our method, and all methods that I compare to, are invariant to reflections, so the striped images with slope in $[0.5, 1.5]$ would receive identical scores to those in $[-0.5, -1.5]$. Note that our method is not necessarily invariant to rotations, because it is based on square, axis-aligned patches of pixels. The same is true of the fractal dimension computed with the Minkowski-Bouligand dimension (i.e., the fractal dimension), as it uses a box-counting method. Examples of Stripes images are shown in Figure B.6.

Halves These images have one half entirely black and the other entirely white, with the dividing line being at various angles. As with Stripes, the slope of this dividing line is sampled uniformly at random from $[-0.5, -1.5]$. Examples are shown in Figure B.7.

Rand These images are white noise. Their values are sampled uniformly at random from $[0, 1]$, independently for each location and each of three colour channels. Examples are shown in Figure B.8.

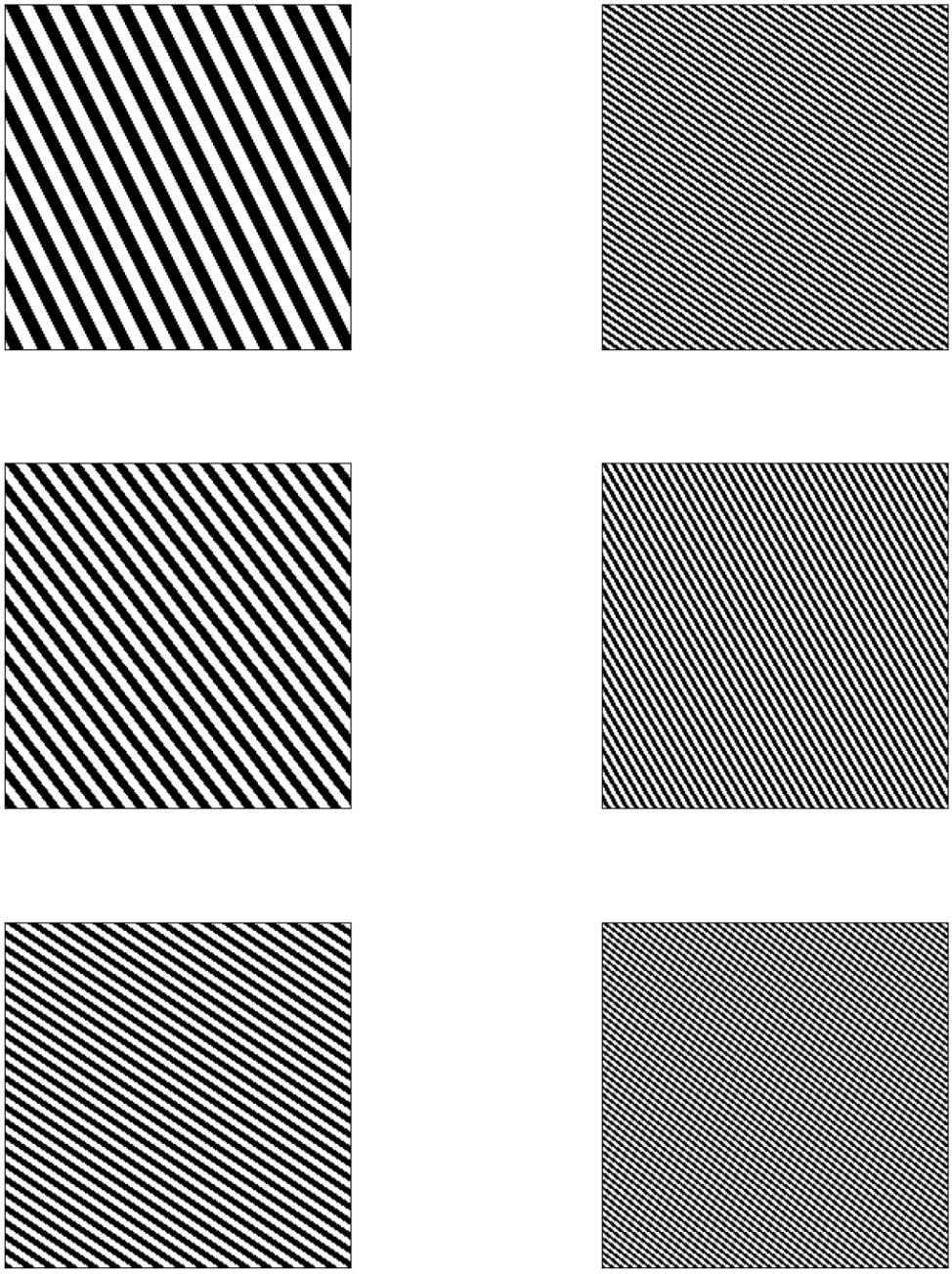

Figure B.6: Examples of images from our synthetic Stripes dataset. Most existing methods assign theses images a high complexity. Ours assigns them low, but non-zero complexity.

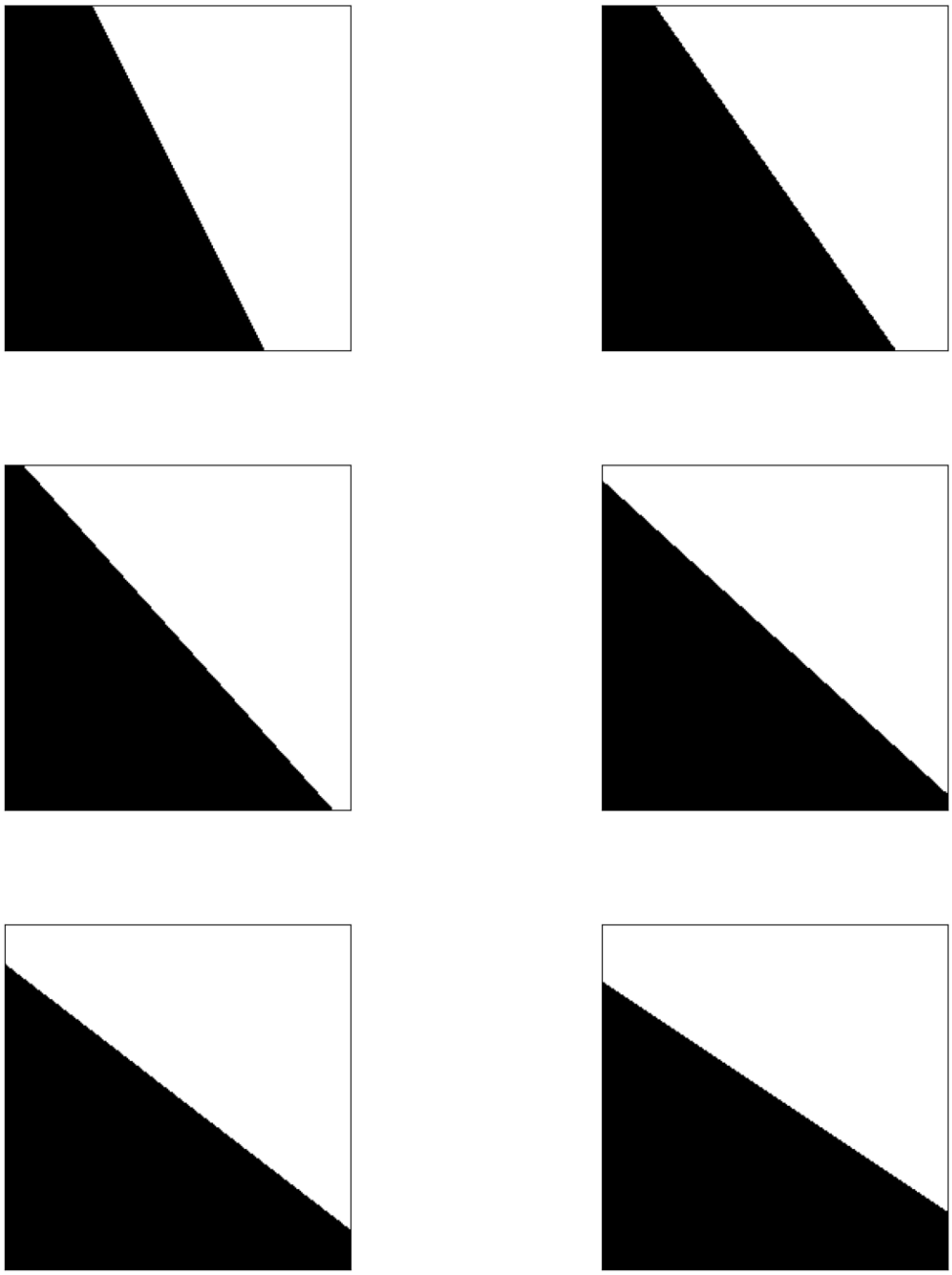

Figure B.7: Examples of images from our synthetic Halves dataset. Our method assigns these low complexity as do existing methods. However, when we break the method down by scale, as discussed in Section 4.2, we see that it assigns some complexity at a high scale, more so than, e.g., Stripes, because there is some difference between different parts at a high scale, whereas in Stripes, both halves of each image are the same.

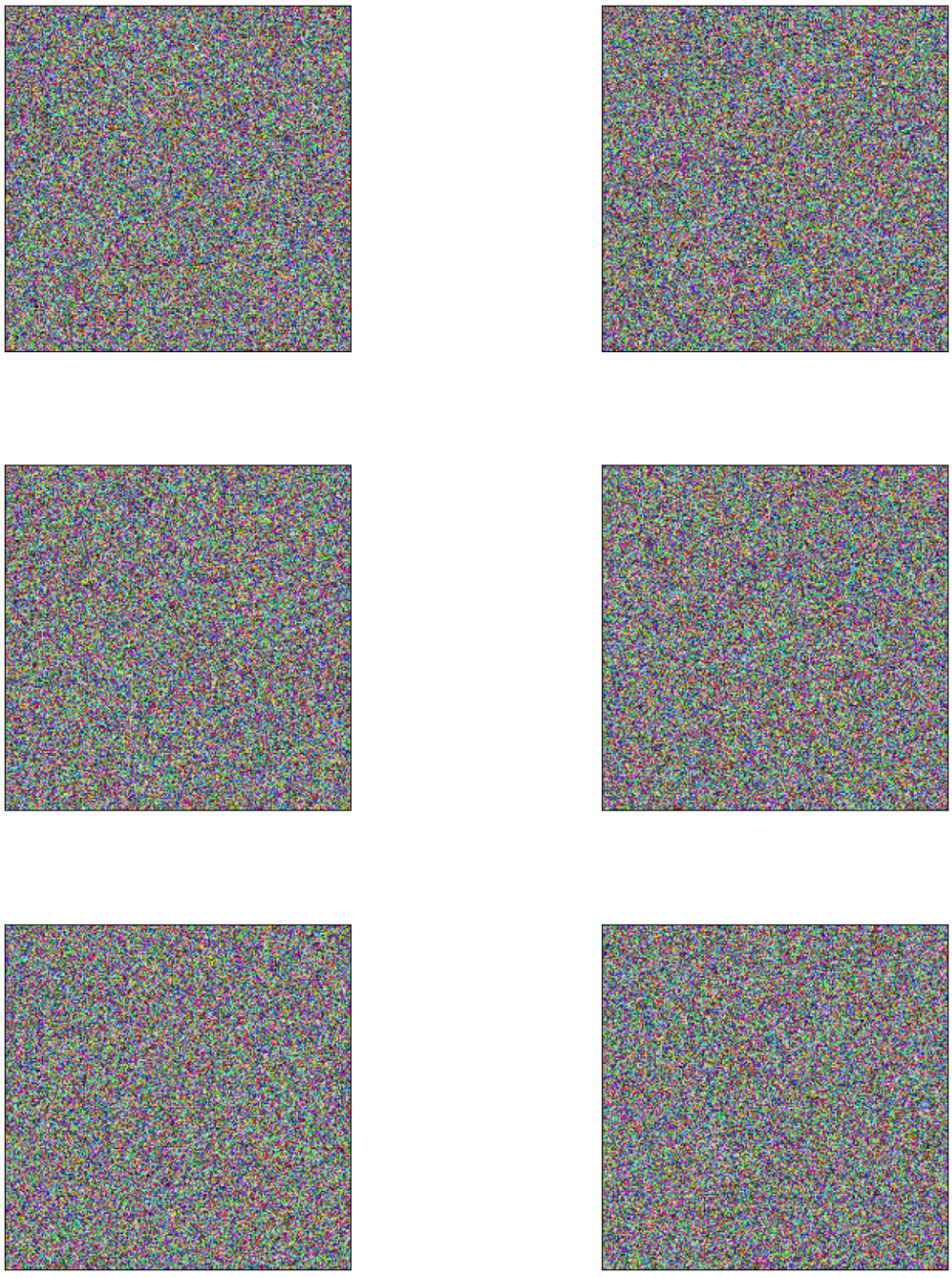

Figure B.8: Examples of the white noise images I used in the Rand dataset. Existing complexity measures assign these images a very high complexity. Our method, in contrast, gives them all a zero complexity.

B.2.2 Existing Datasets

For the experiments in Section 4, I randomly sample 500 images from each of ImageNet, CiFAR, and MNIST. For DTD2, we manually search through all 5640 images in the original Describable Textures Dataset Cimpoi, Maji, Kokkinos, Mohamed, and Vedaldi, 2014, and we find 341 images with fine detailed but repetitive textures. This section contains further information on the images used for each dataset.

ImageNet All images are from the Imagenette subset of ImageNet. The list of labels that we present is taken from the Imagenette labels, available at https://s3.amazonaws. com/fast-ai-imageclas/imagenette2.tgz. Most of the images IDs are of the form <wordnet-synset-id> - <index-within-synset>. Some instead use the class label from the 2012 version of ILSVRC.

- n03394916 - 50642.
- n02979186 - 10250.
- n03028079 - 14492.
- n03888257 - 36631.
- n03445777 - 3291.
- n03425413 - 14940.
- n03417042 - 19472.
- n03000684 - 9440.
- n01440764 - 16090.
- ILSVRC2012-00023440.
- n03394916 - 39102.
- n02979186 - 24592.
- n03028079 - 25712.
- n03888257 - 16542.
- n03445777 - 12262.
- n03425413 - 17220.
- n03417042 - 3351.
- n03000684 - 32351.
- n01440764 - 21191.
- n02102040 - 1782.
- n03394916 - 46700.
- n02979186 - 17680.
- n03028079 - 7422.
- n03888257 - 25150.
- n03445777 - 5582.
- n03425413 - 8801.
- n03417042 - 25411.
- n03000684 - 20052.
- n01440764 - 5361.
- n02102040 - 2930.
- n03394916 - 43381.
- n02979186 - 20620.
- n03028079 - 16811.
- n03888257 - 21201.
- n03445777 - 11171.
- n03425413 - 13581.
- n03417042 - 6420.
- ILSVRC2012-00045501.
- n01440764 - 16192.
- n02102040 - 5890.

- n03394916 - 43532.
- n02979186 - 9910.
- ILSVRC2012-00004912.
- n03888257 - 22330.
- n03445777 - 10762.
- n03425413 - 11061.
- n03417042 - 9620.
- n03000684 - 16872.
- n01440764 - 6812.
- n02102040 - 3452.
- n03394916 - 32870.
- n02979186 - 23650.
- n03028079 - 27781.
- n03888257 - 20300.
- n03445777 - 6091.
- n03425413 - 20371.
- n03417042 - 3771.
- n03000684 - 10992.
- n01440764 - 13842.
- n02102040 - 6851.
- n03394916 - 11582.
- n02979186 - 18720.
- n03028079 - 29062.
- n03888257 - 18441.
- n03445777 - 520.
- n03425413 - 21180.
- n03417042 - 4072.
- n03000684 - 9452.
- n01440764 - 14150.
- n02102040 - 652.
- n03394916 - 36000.
- n02979186 - 1810.
- n03028079 - 4612.
- n03888257 - 7921.
- n03445777 - 5932.
- n03425413 - 12711.
- n03417042 - 4462.
- n03000684 - 661.
- n01440764 - 9152.
- n02102040 - 7942.
- n03394916 - 36172.
- n02979186 - 13740.
- n03028079 - 9920.
- n03888257 - 36390.
- n03445777 - 6162.
- n03425413 - 12951.
- n03417042 - 2150.
- n03000684 - 10690.
- n01440764 - 14342.
- n02102040 - 5942.
- n03394916 - 43422.
- n02979186 - 13442.
- n03028079 - 34051.
- n03888257 - 46870.
- n03445777 - 13480.

- n03425413 - 20751.
- n03417042 - 5920.
- n03000684 - 18020.
- n01440764 - 7982.
- n02102040 - 182.
- n03394916 - 42721.
- n02979186 - 11971.
- n03028079 - 10191.
- n03888257 - 19580.
- n03445777 - 1390.
- n03425413 - 13862.
- n03417042 - 18582.
- ILSVRC2012-00045940.
- n01440764 - 1561.
- n02102040 - 4090.
- n03394916 - 33380.
- n02979186 - 2002.
- n03028079 - 9682.
- n03888257 - 9770.
- n03445777 - 11162.
- n03425413 - 15321.
- n03417042 - 14000.
- n03000684 - 15441.
- n01440764 - 16051.
- n02102040 - 6552.
- n03394916 - 59361.
- n02979186 - 9811.
- n03028079 - 29942.
- n03888257 - 20352.
- n03445777 - 2611.
- n03425413 - 11180.
- n03417042 - 26782.
- n03000684 - 7222.
- n01440764 - 19302.
- ILSVRC2012-00036282.
- n03394916 - 62451.
- n02979186 - 140.
- n03028079 - 49281.
- n03888257 - 14530.
- n03445777 - 5240.
- n03425413 - 21730.
- n03417042 - 12790.
- n03000684 - 13402.
- n01440764 - 4360.
- n02102040 - 352.
- n03394916 - 46672.
- n02979186 - 1542.
- n03028079 - 15392.
- n03888257 - 10680.
- n03445777 - 17492.
- n03425413 - 16220.
- n03417042 - 7080.
- n03000684 - 16291.
- n01440764 - 2921.
- n02102040 - 8061.

- n03394916 - 30072.
- n02979186 - 5321.
- n03028079 - 17690.
- n03888257 - 70632.
- n03445777 - 9572.
- n03425413 - 1672.
- n03417042 - 4761.
- n03000684 - 18591.
- n01440764 - 8030.
- n02102040 - 5641.
- n03394916 - 50730.
- n02979186 - 8861.
- ILSVRC2012-00016542.
- n03888257 - 3651.
- n03445777 - 5312.
- n03425413 - 21362.
- n03417042 - 8822.
- n03000684 - 19272.
- n01440764 - 6421.
- n02102040 - 960.
- n03394916 - 26422.
- n02979186 - 3260.
- n03028079 - 6110.
- n03888257 - 33021.
- n03445777 - 15810.
- n03425413 - 8661.
- n03417042 - 21361.
- n03000684 - 2820.
- n01440764 - 650.
- n02102040 - 1791.
- n03394916 - 52191.
- n02979186 - 14630.
- n03028079 - 6722.
- n03888257 - 142.
- n03445777 - 11150.
- n03425413 - 20500.
- n03417042 - 27630.
- n03000684 - 15521.
- n01440764 - 6130.
- n02102040 - 491.
- n03394916 - 71910.
- n02979186 - 8092.
- n03028079 - 5942.
- n03888257 - 11222.
- n03445777 - 2530.
- n03425413 - 602.
- n03417042 - 5221.
- n03000684 - 1970.
- n01440764 - 13702.
- n02102040 - 3450.
- n03394916 - 35320.
- n02979186 - 16142.
- n03028079 - 14992.
- n03888257 - 37362.
- n03445777 - 6042.

- n03425413 - 12712.
- n03417042 - 26850.
- n03000684 - 180.
- n01440764 - 12881.
- n02102040 - 4111.
- n03394916 - 16601.
- n02979186 - 4511.
- n03028079 - 5432.
- n03888257 - 64711.
- n03445777 - 7711.
- n03425413 - 17212.
- n03417042 - 5510.
- n03000684 - 19890.
- n01440764 - 27422.
- n02102040 - 651.
- n03394916 - 54570.
- n02979186 - 11.
- n03028079 - 16731.
- n03888257 - 13410.
- n03445777 - 7090.
- n03425413 - 13970.
- n03417042 - 27862.
- n03000684 - 2340.
- n01440764 - 3782.
- n02102040 - 290.
- n03394916 - 59430.
- n02979186 - 26820.
- n03028079 - 3600.
- n03888257 - 12401.
- n03445777 - 1750.
- n03425413 - 14302.
- n03417042 - 28552.
- n03000684 - 11511.
- n01440764 - 20451.
- n02102040 - 371.
- n03394916 - 47852.
- n02979186 - 3472.
- n03028079 - 8572.
- n03888257 - 14901.
- n03445777 - 3301.
- n03425413 - 14510.
- n03417042 - 2141.
- n03000684 - 31112.
- n01440764 - 2102.
- n02102040 - 2572.
- n03394916 - 38680.
- n02979186 - 1200.
- n03028079 - 17922.
- n03888257 - 15382.
- n03445777 - 13462.
- n03425413 - 20121.
- n03417042 - 15592.
- n03000684 - 31721.
- n01440764 - 32420.
- n02102040 - 1830.

- n03394916 - 35811.
- n02979186 - 12072.
- n03028079 - 46322.
- n03888257 - 28581.
- n03445777 - 602.
- n03425413 - 32871.
- n03417042 - 18042.
- n03000684 - 6220.
- n01440764 - 17501.
- n02102040 - 7392.
- n03394916 - 36361.
- n02979186 - 22761.
- n03028079 - 24471.
- n03888257 - 13790.
- n03445777 - 7930.
- n03425413 - 21040.
- n03417042 - 1330.
- n03000684 - 1542.
- n01440764 - 8302.
- n02102040 - 6081.
- n03394916 - 27071.
- n02979186 - 5781.
- ILSVRC2012-00034021.
- n03888257 - 38102.
- n03445777 - 16321.
- n03425413 - 20562.
- n03417042 - 4560.
- n03000684 - 6471.
- n01440764 - 762.
- n02102040 - 2110.
- n03394916 - 44882.
- n02979186 - 5481.
- n03028079 - 9220.
- n03888257 - 19211.
- n03445777 - 14301.
- n03425413 - 19050.
- n03417042 - 6691.
- n03000684 - 2972.
- n01440764 - 10040.
- n02102040 - 430.
- n03394916 - 46391.
- n02979186 - 13281.
- n03028079 - 16820.
- n03888257 - 30712.
- n03445777 - 14232.
- n03425413 - 21562.
- n03417042 - 29412.
- n03000684 - 13182.
- n01440764 - 10852.
- n02102040 - 5101.
- n03394916 - 29940.
- n02979186 - 2841.
- n03028079 - 23280.
- n03888257 - 23192.
- n03445777 - 2041.

- n03425413 - 14570.
- n03417042 - 20280.
- n03000684 - 8411.
- n01440764 - 7492.
- n02102040 - 6532.
- n03394916 - 28590.
- n02979186 - 560.
- n03028079 - 38692.
- n03888257 - 23571.
- n03445777 - 13680.
- ILSVRC2012-00000732.
- n03417042 - 18551.
- n03000684 - 34440.
- n01440764 - 522.
- ILSVRC2012-00008162.
- n03394916 - 1091.
- n02979186 - 10151.
- n03028079 - 12802.
- n03888257 - 171.
- n03445777 - 7670.
- n03425413 - 21202.
- n03417042 - 9601.
- ILSVRC2012-00029211.
- n01440764 - 5432.
- n02102040 - 4732.
- n03394916 - 292.
- n02979186 - 5460.
- n03028079 - 3700.
- n03888257 - 35800.
- n03445777 - 9921.
- n03425413 - 21911.
- n03417042 - 5090.
- n03000684 - 19211.
- n01440764 - 8601.
- n02102040 - 7841.
- n03394916 - 27932.
- n02979186 - 3161.
- n03028079 - 29012.
- n03888257 - 17340.
- n03445777 - 10782.
- n03425413 - 11161.
- n03417042 - 29722.
- n03000684 - 1490.
- n01440764 - 4962.
- n02102040 - 7792.
- n03394916 - 47110.
- n02979186 - 16952.
- n03028079 - 28242.
- n03888257 - 29762.
- n03445777 - 230.
- n03425413 - 3021.
- n03417042 - 10462.
- n03000684 - 2060.
- n01440764 - 6301.
- n02102040 - 2480.

- n03394916 - 44580.
- n02979186 - 20362.
- n03028079 - 3492.
- n03888257 - 30412.
- n03445777 - 13831.
- n03425413 - 20301.
- n03417042 - 10280.
- n03000684 - 16861.
- n01440764 - 9212.
- n02102040 - 6152.
- n03394916 - 34332.
- n02979186 - 14251.
- n03028079 - 9320.
- n03888257 - 35890.
- n03445777 - 5131.
- n03425413 - 16221.
- n03417042 - 5381.
- n03000684 - 3470.
- n01440764 - 8142.
- n02102040 - 762.
- n03394916 - 51071.
- n02979186 - 20160.
- n03028079 - 25542.
- n03888257 - 57010.
- n03445777 - 261.
- n03425413 - 7731.
- n03417042 - 3821.
- ILSVRC2012-00047060.
- n01440764 - 12971.
- n02102040 - 1300.
- n03394916 - 7292.
- n02979186 - 23362.
- n03028079 - 10020.
- ILSVRC2012-00038942.
- n03445777 - 11690.
- n03425413 - 13100.
- n03417042 - 6811.
- n03000684 - 20762.
- n01440764 - 11350.
- n02102040 - 1822.
- n03394916 - 33012.
- n02979186 - 1061.
- n03028079 - 16660.
- n03888257 - 38200.
- n03445777 - 10671.
- n03425413 - 6772.
- n03417042 - 1492.
- n03000684 - 24991.
- n01440764 - 7462.
- n02102040 - 362.
- n03394916 - 26802.
- n02979186 - 3530.
- n03028079 - 80.
- n03888257 - 66102.
- n03445777 - 8192.

- ILSVRC2012-00035211.
- n03417042 - 10300.
- n03000684 - 16072.
- n01440764 - 8451.
- n02102040 - 3260.
- ILSVRC2012-00025761.
- n02979186 - 5031.
- n03028079 - 10241.
- n03888257 - 12400.
- n03445777 - 6201.
- n03425413 - 260.
- n03417042 - 2062.
- n03000684 - 27850.
- n01440764 - 9491.
- n02102040 - 821.
- n03394916 - 32340.
- n02979186 - 1932.
- n03028079 - 26291.
- n03888257 - 9552.
- n03445777 - 101.
- n03425413 - 1792.
- n03417042 - 18152.
- n03000684 - 5041.
- n01440764 - 4980.
- n02102040 - 3532.
- n03394916 - 6742.
- n02979186 - 22882.
- n03028079 - 25462.
- n03888257 - 8381.
- n03445777 - 5382.
- n03425413 - 13232.
- n03417042 - 9170.
- n03000684 - 17330.
- n01440764 - 6361.
- n02102040 - 142.
- n03394916 - 51161.
- n02979186 - 15931.
- n03028079 - 28662.
- n03888257 - 12070.
- n03445777 - 10401.
- n03425413 - 4511.
- n03417042 - 1601.
- n03000684 - 10212.
- n01440764 - 7752.
- n02102040 - 1110.
- n03394916 - 38212.
- n02979186 - 1621.
- n03028079 - 2060.
- n03888257 - 7610.
- n03445777 - 7902.
- n03425413 - 21211.
- n03417042 - 3390.
- n03000684 - 11821.
- n01440764 - 8221.
- n02102040 - 350.

- n03394916 - 37321.
- n02979186 - 2312.
- n03028079 - 16501.
- n03888257 - 11081.
- n03445777 - 471.
- n03425413 - 24461.
- n03417042 - 6272.
- n03000684 - 6460.
- n01440764 - 7160.
- n02102040 - 3112.

- n03394916 - 33221.
- n02979186 - 15972.
- n03028079 - 9112.
- n03888257 - 7130.
- n03445777 - 8861.
- n03425413 - 14552.
- n03417042 - 2960.
- n03000684 - 5231.
- n01440764 - 16072.
- n02102040 - 672.

DTD2 The Describable Textures Dataset contains 47 classes, grouped according to texture: bumpy, dotted, lined, veined etc. The image ids below are of the form <class-id> - <index-within-class>. Unsurprisingly, most of the suitable images, i.e., those with detailed repeating textures, are from classes such as 'woven', 'grid', or 'wrinkled'. The number of images of each class is given below.

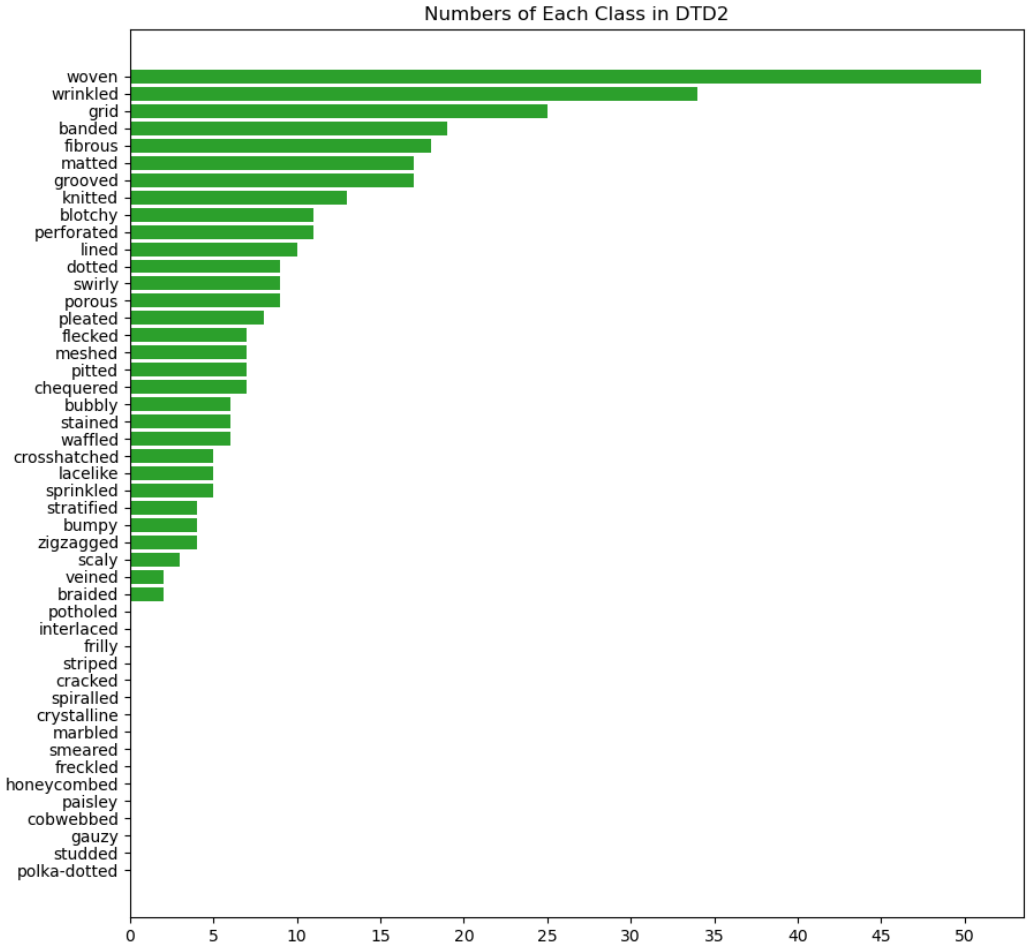

Figure B.9: Number of images of each class from DTD that we select to be part of our curated dataset, DTD2. We select images that show a detailed, repeating pattern, meaning some types of textures are much more likely to be selected.

- perforated - 0074.
- meshed - 0164.
- dotted - 0164.
- sprinkled - 0066.
- porous - 0117.
- woven - 0106.
- knitted - 0185.
- crosshatched - 0081.
- pleated - 0163.
- banded - 0046.
- wrinkled - 0129.
- banded - 0055.

- braided - 0008.
- grooved - 0119.
- dotted - 0131.
- wrinkled - 0015.
- bumpy - 0098.
- woven - 0004.
- zigzagged - 0109.
- matted - 0136.
- stratified - 0174.
- grooved - 0051.
- perforated - 0014.
- grooved - 0089.
- woven - 0025.
- wrinkled - 0106.
- lined - 0159.
- banded - 0008.
- matted - 0070.
- lined - 0109.
- dotted - 0192.
- fibrous - 0138.
- matted - 0150.
- pleated - 0142.
- grid - 0088.
- blotchy - 0038.
- chequered - 0043.
- banded - 0081.
- wrinkled - 0063.
- waffled - 0156.
- grid - 0073.
- grid - 0016.
- lacelike - 0078.
- matted - 0071.
- chequered - 0050.
- wrinkled - 0132.
- porous - 0149.
- stained - 0119.
- knitted - 0118.
- pleated - 0116.
- stained - 0066.
- knitted - 0144.
- chequered - 0062.
- grooved - 0085.
- blotchy - 0091.
- knitted - 0150.
- grid - 0124.
- pleated - 0168.
- zigzagged - 0133.
- grooved - 0058.
- zigzagged - 0008.
- stained - 0132.
- blotchy - 0088.
- bumpy - 0067.
- grid - 0049.
- woven - 0062.
- blotchy - 0059.

- matted - 0069.
- lined - 0133.
- woven - 0075.
- bubbly - 0097.
- matted - 0073.
- porous - 0151.
- blotchy - 0083.
- chequered - 0052.
- wrinkled - 0041.
- lacelike - 0096.
- matted - 0085.
- fibrous - 0150.
- banded - 0122.
- waffled - 0124.
- fibrous - 0164.
- grid - 0083.
- fibrous - 0193.
- dotted - 0060.
- meshed - 0176.
- woven - 0043.
- woven - 0088.
- stained - 0090.
- wrinkled - 0045.
- pleated - 0090.
- zigzagged - 0085.
- veined - 0135.
- dotted - 0132.
- stratified - 0046.
- woven - 0061.
- woven - 0028.
- swirly - 0074.
- matted - 0065.
- sprinkled - 0065.
- waffled - 0068.
- grooved - 0048.
- perforated - 0066.
- grid - 0022.
- woven - 0053.
- porous - 0152.
- fibrous - 0160.
- woven - 0055.
- matted - 0148.
- pitted - 0134.
- flecked - 0060.
- lacelike - 0020.
- grid - 0052.
- woven - 0067.
- knitted - 0192.
- flecked - 0053.
- chequered - 0054.
- chequered - 0088.
- lined - 0027.
- stained - 0030.
- knitted - 0146.
- grid - 0078.

- blotchy - 0070.
- swirly - 0060.
- perforated - 0057.
- porous - 0098.
- wrinkled - 0087.
- blotchy - 0096.
- grooved - 0081.
- wrinkled - 0039.
- lined - 0041.
- flecked - 0126.
- lined - 0038.
- dotted - 0135.
- pitted - 0036.
- wrinkled - 0103.
- wrinkled - 0034.
- grid - 0050.
- bubbly - 0083.
- woven - 0001.
- knitted - 0116.
- pleated - 0069.
- wrinkled - 0043.
- woven - 0059.
- knitted - 0079.
- matted - 0128.
- lacelike - 0017.
- fibrous - 0165.
- wrinkled - 0088.
- grid - 0129.
- blotchy - 0090.
- wrinkled - 0017.
- sprinkled - 0038.
- woven - 0032.
- flecked - 0074.
- woven - 0029.
- knitted - 0130.
- crosshatched - 0092.
- lacelike - 0065.
- knitted - 0141.
- grid - 0011.
- porous - 0099.
- woven - 0039.
- woven - 0113.
- fibrous - 0211.
- sprinkled - 0067.
- wrinkled - 0125.
- crosshatched - 0093.
- dotted - 0154.
- woven - 0130.
- veined - 0075.
- meshed - 0181.
- fibrous - 0103.
- fibrous - 0183.
- woven - 0082.
- woven - 0099.
- perforated - 0041.

- grid - 0099.
- grooved - 0084.
- meshed - 0162.
- wrinkled - 0036.
- banded - 0147.
- porous - 0157.
- wrinkled - 0108.
- dotted - 0185.
- grid - 0089.
- grid - 0101.
- woven - 0048.
- grid - 0066.
- bumpy - 0190.
- matted - 0166.
- woven - 0104.
- waffled - 0171.
- wrinkled - 0040.
- flecked - 0135.
- swirly - 0151.
- stratified - 0115.
- perforated - 0045.
- woven - 0026.
- fibrous - 0111.
- swirly - 0065.
- perforated - 0026.
- banded - 0107.
- woven - 0068.
- banded - 0037.
- fibrous - 0204.
- wrinkled - 0026.
- waffled - 0178.
- woven - 0108.
- grooved - 0164.
- woven - 0021.
- fibrous - 0127.
- banded - 0141.
- scaly - 0131.
- woven - 0123.
- braided - 0167.
- woven - 0046.
- grooved - 0057.
- perforated - 0024.
- swirly - 0137.
- grid - 0081.
- bubbly - 0118.
- grooved - 0108.
- wrinkled - 0079.
- flecked - 0003.
- fibrous - 0120.
- wrinkled - 0114.
- woven - 0083.
- fibrous - 0110.
- wrinkled - 0067.
- lined - 0169.
- wrinkled - 0025.

- wrinkled - 0021.
- wrinkled - 0013.
- dotted - 0041.
- woven - 0049.
- lined - 0076.
- scaly - 0122.
- grid - 0059.
- waffled - 0081.
- matted - 0117.
- fibrous - 0101.
- stained - 0075.
- woven - 0036.
- wrinkled - 0086.
- wrinkled - 0084.
- banded - 0047.
- banded - 0068.
- matted - 0155.
- perforated - 0080.
- pitted - 0078.
- pitted - 0008.
- fibrous - 0089.
- sprinkled - 0068.
- woven - 0084.
- grooved - 0093.
- woven - 0056.
- pitted - 0064.
- wrinkled - 0046.
- woven - 0109.
- banded - 0086.
- grid - 0084.
- grid - 0116.
- woven - 0038.
- pleated - 0082.
- bumpy - 0140.
- wrinkled - 0111.
- matted - 0115.
- fibrous - 0096.
- woven - 0093.
- swirly - 0144.
- banded - 0059.
- lined - 0141.
- woven - 0003.
- banded - 0114.
- woven - 0002.
- woven - 0127.
- knitted - 0126.
- banded - 0115.
- woven - 0051.
- lined - 0166.
- bubbly - 0084.
- flecked - 0165.
- wrinkled - 0105.
- woven - 0114.
- woven - 0126.
- grooved - 0063.

- wrinkled - 0083.
- grooved - 0088.
- grid - 0032.
- wrinkled - 0065.
- grid - 0067.
- perforated - 0016.
- meshed - 0108.
- blotchy - 0082.
- grooved - 0045.
- swirly - 0147.
- grooved - 0068.
- woven - 0065.
- grid - 0085.
- blotchy - 0089.
- wrinkled - 0085.
- woven - 0107.
- stratified - 0100.
- fibrous - 0201.
- scaly - 0137.
- woven - 0071.
- perforated - 0119.
- swirly - 0138.
- grooved - 0083.
- matted - 0072.
- grid - 0093.
- chequered - 0093.
- knitted - 0098.
- matted - 0084.
- pitted - 0010.
- porous - 0053.
- matted - 0129.
- woven - 0066.
- crosshatched - 0109.
- wrinkled - 0033.
- pitted - 0157.
- porous - 0142.
- woven - 0092.
- crosshatched - 0116.
- swirly - 0159.
- grid - 0082.
- banded - 0099.
- meshed - 0112.
- knitted - 0155.
- woven - 0112.
- meshed - 0161.
- banded - 0061.
- bubbly - 0055.
- woven - 0063.
- bubbly - 0096.
- blotchy - 0041.
- banded - 0002.
- woven - 0007.
- banded - 0090.
- pleated - 0094.

Table B.1: Counts of each class in our random sample of 500 CIFAR images.

class	airplane	car	bird	cat	deer	dog	frog	horse	ship	truck
count	66	66	48	48	48	47	49	40	48	40

Table B.2: Counts of each class in our random sample of 500 MNIST images.

class	0	1	2	3	4	5	6	7	8	9
count	47	61	57	52	47	45	41	50	53	47

CIFAR10 and MNIST For CIFAR and MNIST, we report the distribution of classes in our random sample of 500, in Tables B.1 and B.2, respectively.