THE UNIVERSITY *of* EDINBURGH

Learning reliable representations when proxy objectives fail

Luke Nicholas Darlow

Doctor of Philosophy

Institute for Adaptive and Neural Computation
School of Informatics
University of Edinburgh

2022

Abstract

Representation learning involves using an objective to learn a mapping from data space to a representation space. When the downstream task for which a mapping must be learned is unknown, or is too costly to cast as an objective, we must rely on proxy objectives for learning. In this Thesis I focus on representation learning for images, and address three cases where proxy objectives fail to produce a mapping that performs well on the downstream tasks.

When learning neural network mappings from image space to a discrete hash space for fast content-based image retrieval, a proxy objective is needed which captures the requirement for relevant responses to be nearer to the hash of any query than irrelevant ones. At the same time, it is important to ensure an even distribution of image hashes across the whole hash space for efficient information use and high discrimination. Proxy objectives fail when they do not meet these requirements. I propose composing hash codes in two parts. First a standard classifier is used to predict class labels that are converted to a binary representation for state-of-the-art performance on the image retrieval task. Second, a binary deep decision tree layer (DDTL) is used to model further intra-class differences and produce approximately evenly distributed hash codes. The DDTL requires no discretisation during learning and produces hash codes that enable better discrimination between data in the same class when compared to previous methods, while remaining robust to real-world augmentations in the data space.

In the scenario where we require a neural network to partition the data into clusters that correspond well with ground-truth labels, a proxy objective is needed to define how these clusters are formed. One such proxy objective involves maximising the mutual information between cluster assignments made by a neural network from multiple views. In this context, views are different augmentations of the same image and the cluster assignments are the representations computed by a neural network. I demonstrate that this proxy objective produces parameters for the neural network that are sub-optimal in that a better set of parameters can be found using the same objective and a different training method. I introduce deep hierarchical object grouping (DHOG) as a method to learn a hierarchy (in the sense of easy-to-hard orderings, not structure) of solutions to the proxy objective and show how this improves performance on the downstream task.

When there are features in the training data from which it is easier to compute class predictions (e.g., background colour), when compared to features for which it is

relatively more difficult to compute class predictions (e.g., digit type), standard classification objectives (e.g., cross-entropy) fail to produce robust classifiers. The problem is that if a model learns to rely on 'easy' features it will also ignore 'complex' features (easy versus complex are purely relative in this case). I introduce latent adversarial debiasing (LAD) to decouple easy features from the class labels by first modelling the underlying structure of the training data as a latent representation using a vector-quantised variational autoencoder, and then I use a gradient-based procedure to adjust the features in this representation to confuse the predictions of a constrained classifier trained to predict class labels from the same representation. The adjusted representations of the data are then decoded to produce an augmented training dataset that can be used for training in a standard manner.

I show in the aforementioned scenarios that proxy objectives can fail and demonstrate that alternative approaches can mitigate against the associated failures. I suggest an analytic approach to understanding the limits of proxy objectives for every use case in order to make the adjustments to the data or the objectives and ensure good performance on downstream tasks.

Lay summary

For neural networks to solve problems they often need to be taught using training examples. Unfortunately, it can be extremely difficult to collect or construct enough such examples. There may, however, be related 'proxy' problems for which it is easier to construct training examples. While it is common practice to use proxy problems as tools for teaching neural networks, there is a trade-off: neural networks tend to find shortcut solutions that could render proxy problems unusable.

In this thesis I dealt with three use-cases where training using proxy problems fail. First, when learning to map images to an extremely compact and discrete space for fast and efficient comparisons between images, widely used proxies fail for teaching neural networks to capture sufficient detail from images. My solution to this is to restructure the discrete space into a tree structure and amend the proxy problem to encourage a neural network into spreading its training data over the entire tree structure. The result is a neural network that maps images to a compact, yet sufficiently descriptive, space because the new proxy problem now requires better use of this space in order to be solved.

Second, the proxy problem of grouping images into clusters is insufficient because neural networks tend to group images based on easy features in the data (e.g., colour) as opposed to more complex features (e.g., object type). I deal with this by encouraging a neural network to learn many diverse solutions to the proxy problem and show that this improves its overall performance on the real problem it must later solve.

Third, when the proxy problem does not pay heed to differences in the data collection processes during training and test time, neural networks can tend to rely on easy-to-compute features in the training data that might not be present at test time. I show how we can intentionally model and then remove these easy-to-compute features from the training data and thereby improve performance at test time.

Table of Contents

Acknowledgements

Thank you to my supervisor Prof. Amos Storkey who met with me years ago, before I was a student here, and encouraged, helped, and guided me. You have been very kind to me, while still driving me to be a better scientist, researcher, and writer. Thank you to my co-supervisor Prof. Chris Williams for always providing insight and reference into those areas of research I never knew, and to Dr Vaishak Belle for excellent insight during my yearly reviews. This thesis was supported by the EPSRC Centre for Doctoral Training in Data Science, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1) and the University of Edinburgh.

I am so very grateful to all the important people in my life who have enabled me to undertake this journey. Foremost is my wonderful, loving, thoughtful wife Piette, who has supported me in every conceivable way. Thank you for being kind and gentle when I needed gentleness, and for pushing and driving me to be stronger and better when I needed driving. I am truly eternally grateful.

Thank you to my family who has always supported and believed in me. I always feel like making you all proud, and I am so grateful for the grace with which you handled me being here, so far from home, doing abstract academic things over these past years. I am so thankful for the love and pride emanating off my mother, my father, my sister, and my mother in law.

For my friends, who have always believed in me and made me feel special and loved, I am so grateful to have you in my life. My friend Arnold has always thought far too highly of me, and for that I have always pushed myself. I am here, in part, due to his belief in me. To Etienne and Patricia, who have been such magnificent friends these past few years, you two are such a huge part of why the pandemic never crippled my mental health; thank you. Thank you Etienne for those many hours of thoughtful discussions on my (sometimes strange) machine learning ideas, and for always being fully committed to listening. Finally, thank you to my friends in the CDT in Data Science were always up for a cup of coffee and a discussion around a whiteboard, and helped drive many good ideas to completion, while also keeping bad ideas from gaining traction.

Chapter 1

Introduction

The field of *Representation Learning* is seen to represent a fundamental challenge for machine learning [Bengio et al., 2013a]. The general problem of *learning representations* is that of finding a map $f : X \to Z$ from a data space X to some representation space Z that somehow *simplifies* the data and maintains or improves the *usefulness* of the data. Typically *usefulness* is understood as the ability to achieve good performance on some downstream task or set of such tasks through only using information from the representation. A downstream task is typically an evaluable prediction task, but one that is not necessarily known when learning the map f. Likewise *simplification* is understood as enabling good performance on the downstream task to be achieved more easily (e.g., using smaller or simpler models, less data or with less training) than when trying to achieve that task directly from the data space X.

Simplification by representation learning might also involve information compression – i.e., f maps the data to a representation that has lower dimensionality than the data space. When learning to predict a lower dimension target (e.g., class labels) information compression is expected [Shwartz-Ziv and Tishby, 2017, Tishby and Zaslavsky, 2015].

Learning a representation typically involves defining an objective function L for a good representation, parameterising the map f as a f_θ (differentiable w.r.t. θ), and then using $\nabla_\theta L$ in an optimisation of that objective [Gori and Tesi, 1992, Swirszcz et al., 2016]. Non-convexity of the objective means there is a potential of optimising to a local optimum rather than a global one.

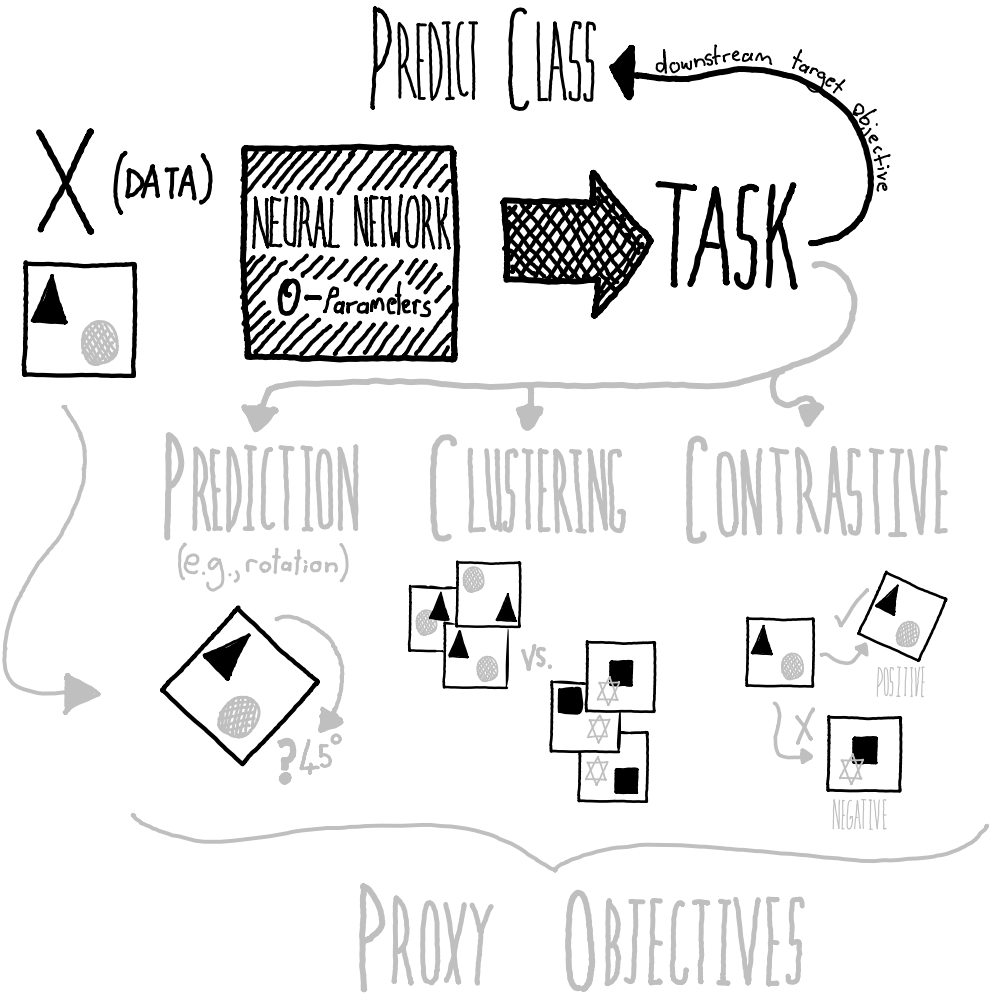


Figure 1.1: Visual depiction of proxy objectives. These can take many forms, but typically involve some proxy 'task' with which we train the underlying neural network in the hope that it performs well on the desired downstream task. Proxy objectives often involve altering the input data in some manner (e.g., with augmentations for images).

1.1 Proxy objectives

When the downstream tasks are not known while f is being optimised, it is not possible to formulate an objective function based on the performance on that task. In such a setting, a *proxy objective* is formed instead, and the representation map is learnt via optimising the proxy objective. A proxy objective usually takes the form of a loss function with which we can train a neural network even when we do not have sufficient labelled data or information to train this network on the desired downstream objective (e.g., classification). Figure 1.1 gives a visual depiction of proxy objectives, where the underlying neural network parameters, θ, are optimised using proxy objectives in the

hope that this network will eventually perform well on the desired downstream task.

However there can be a mismatch between the proxy objective and the true objective, and I argue that this mismatch can have far reaching consequences. The implications of such a mismatch is the primary topic of this Thesis. This issue has become increasingly important in the self-supervised learning domain [Chen et al., 2021, Li et al., 2020b, Robinson et al., 2021, Wang et al., 2021], but is also prevalent when training data causes model bias [Zhang et al., 2018, Das et al., 2018], or when learning discrete representations for which gradient-based optimisation methods for neural networks are ill-suited [Dubey, 2021].

1.2 Failure of proxy objectives

A large neural network can outperform most other model types on many predictive (e.g., classification) and generative (e.g., image generation) tasks given enough data, compute, and a reasonable choice of hyper-parameters [Sun et al., 2017, Sejnowski, 2020, Fabbri and Moro, 2018, Aggarwal, 2018]. Nonetheless, large neural networks also have many failure modes. For example, they are not immediately interpretable [Alvarez Melis and Jaakkola, 2018, Zhang et al., 2021b], they require long and data-inefficient optimisation to fit [Sun et al., 2017, Ying, 2019], and they can produce non-robust representations [Goodfellow et al., 2015, Geirhos et al., 2019]. Moreover, proxy objectives are not necessarily sufficient for learning useful representations because they might not fully capture the properties required by the downstream task(s).

In the work described in this Thesis, I consider a common failure of neural networks: learning settles in local optima characterised by the reliance of *easy-to-compute* or *easy-to-learn* features. This happens because: (1) proxy objectives are unable to fully capture or describe the properties required of a representation; (2) proxy objectives can sometimes be optimised for or satisfied in a number of ways; and (3) proxy objectives can result in compression of relevant information (to the downstream task) because this information may not be relevant to optimise the proxy objective.

A neural network with parameter values stuck in this type of local optima will be poorly matched to downstream tasks. A better local optima may exist – one that requires learning abstract features that have better utility for downstream use – but proxy objective failure prevents the learning process for the parameters from reaching it. The following are examples of specific instances where proxy objectives fail.

1. Proxy objectives are insufficient at fully describing the task at hand. This could be because the task is ill-posed or that fully describing the task is far too costly because of the constraints imposed by the user. In such cases, information compression in the learned representation [Shwartz-Ziv and Tishby, 2017] will tend to discard any additional information that is not captured by the proxy objective. This additional information might be useful for the downstream task. Therefore, this phenomenon is called **over-compression** – it is an unwanted consequence of 'irrelevant' information compression [Darlow and Storkey, 2020, Shwartz-Ziv and Tishby, 2017], where optimising the proxy objective discards information that is useful for the downstream task. In Chapter 3 the task is to provide a hash function suitable for image retrieval, and I focus on methods for learning such a semantic hash function. Class labels are sufficient to optimise supervised retrieval performance metrics (e.g., mean average precision – see Section 3.2.2), but insufficient at ensuring high-coverage hash functions. The result of learning hash functions with supervision alone is a constrained set of hash codes that are overly consistent within each class. For example, when failure occurs every hash code for an image of a cat is nearly identical, thereby disabling comparisons between cats. I expand on this in Section 2.5.7 and provide evidence in Section 3.2.3. Existing methods fail to produce hash codes with high-coverage of the available hash space [Knuth, 1997, Cichelli, 1980, Dietzfelbinger et al., Luo et al., 2021] (see Section 3.2.3), which in turn leads to reduced retrieval performance. I mitigate against this failure in Chapter 3 by learning a binary deep decision tree to evenly partition the training images for high-coverage hash codes.

2. Greedy gradient-based optimisation fails to find a local optima of the proxy objective that also results in good performance on downstream tasks. Often proxy objectives are ill-defined in that they have many local minima that are quantitatively similar (e.g., the loss is similar) but which rely on different features in the data (e.g., one could rely on colour or object type). In Chapter 4 the task is clustering using a neural network and the proxy objective is to maximise the mutual information between cluster assignments from different augmentations of the same image. I demonstrate that such a proxy objective finds parameters that are sub-optimal during learning, in that there are better parameter settings that can be learned by a different training method targeting the same objective.

My approach is to sequentially **expand the knowledge base of the model** into a hierarchy of solutions by minimising mutual information between successive clusterings.

3. Even if the proxy objective is fully descriptive of the downstream objective and optimisation is not affected by the aforementioned local optima issue, optimisation can result in a representation that is not robust to the types of distributional shift we might reasonably expect. In Chapter 5 I detail a setup where a model learns to rely on easy-to-compute spurious signals from extraneous variables, and ignores other predictive information that transfers better to the test setting. To deal with easy-to-compute spurious signals I propose the following process: (1) model the data as a latent representation using a vector quantised variational autoencoder [Van Den Oord et al., 2017]; (2) fit a shallow neural network that predicts the class label to this representation; (3) compute gradients of the latent representation with respect to the shallow network's predictions, and apply these changes to confuse the shallow network, effectively disassociating the spurious features and the class labels; and (4) use the resulting decoded (via the same autoencoder) images as training data in a standard classification training setup.

The remainder of this Thesis is arranged as follows. In Section 2 I provide the technical background and literature reviews for work relevant to later chapters. In Chapter 3 I present the deep decision tree layer for improving information efficiency of deep semantic hash functions. In Chapter 4 I present deep hierarchical object grouping for learning a diverse set of solutions that optimise the deep clustering objective for improved downstream classification performance. In Chapter 5 I present latent adversarial debiasing as a technique to remove spurious signals from training data to improve robustness at test time. Chapter 6 concludes this work with a summary and discusses implications and directions for future work.

Chapter 2

Technical background and literature review

This chapter provides a technical background of concepts and literature review of existing work that is relevant to the remainder of this Thesis. In Section 2.1 I argue the importance of representation learning, briefly describe the neural network architecture used through this Thesis as a backbone that produces representations in Section 2.1.1. In Section 2.1.2 I delineate the main differences between supervised and unsupervised learning.

In Section 2.2 I describe how proxy objectives are widely used for representation learning and give specific examples for: information maximisation (Section 2.2.1), contrastive learning (Section 2.2.2), and deep clustering (Section 2.2.3). I also review the relevant literature for each of these examples while explaining them.

In Section 2.3 I give the technical background as to why proxy objectives can result in sub-optimal solutions. I explain the phenomenon of over-compression in Section 2.3.1 and explain how it leads to sub-optimal representations when the training objective fails to capture the full requirements of a downstream task. I use deep clustering (Section 2.3.2) as an example where the proxy objective fails to prevent learning representations computed from easy features (e.g., colour) instead of complex features (e.g., object type). I also describe how neural network bias (Section 2.3.3) is an example where the proxy objective fails to delineate between easy-to-compute features and complex features, each of which are informative of the class label in the training data but not at test time.

In Section 2.4 I review the literature associated with the use of decision trees with neural networks – this is relevant for Chapter 3. In Section 2.5 I review the literature related to learning deep semantic hash functions – this is relevant for Chapter 3

2.1 Representation learning

Learning a mapping of data to representation space is seen as a fundamental challenge in machine learning (see Chapter 1). Different representations of data can disentangle or compress factors present in the data [Bengio et al., 2013a]. What makes a representation 'good' is a question of its relevance for a downstream task. For example, if the downstream task is object detection, a representation that disentangles factors such that objects are simple to detect with a small model (e.g., a linear projection) is useful. Important questions that have driven deep learning of representations are: (1) is this representation amenable to solving the downstream task? and (2) does learning on this representation enable generalisation to reasonable unseen data and/or tasks?

Mechanisms such as standardisation, whitening, normalisation, and various kinds of transformations are widely used on data before fitting many classes of models. These mechanisms can be seen as mappings of the data space that help improve the learning or generalisation of models. For example, standardisation is the process of scaling individual features in the data such that they have zero mean and unit variance. When a model relies internally on a distance measure between data points to make predictions, standardisation can improve performance. Dimensionality reduction (e.g., principle component analysis (PCA) [Abdi and Williams, 2010]) and kernel methods are ways of representing or comparing data that often result in improved model performance or generalisation. Hand crafted features for images, such as the scale-invariant feature transform [Lowe, 1999] or variations of histogram analysis, were widely used in computer vision before end-to-end feature learning using neural networks began to dominate.

Better representations of image data for computer vision tasks are typically those that capture high-level semantic information, such as object types, as opposed to low-level information (such as pixel colours) or mid-level information (such as textures). Neural networks [Rosenblatt, 1958, LeCun et al., 2015] are widely and successfully used for image problems problems because they are capable of modelling and synthesising complex non-linear functions at multiple scales, thereby enabling learning representations that are expressive of abstract semantic variation in the data.

Representation learning for image data using neural networks is a wide-reaching domain because of the ubiquity of image data and the difficulty associated with developing performant representations. The question central to this Thesis is: how can we learn data mappings that *perform well on downstream tasks* when *proxy training*

objectives fail? I discuss the backbone neural network we use to produce representations throughout this Thesis in the next section. In Section 2.1.2 I discuss the two main learning paradigms – supervised and unsupervised learning – both of which are explored to varying degrees in later chapters.

2.1.1 The ResNet backbone

A neural network consists of many densely interconnected computing nodes (neurons). These computing nodes are parameterised and are typically set up in a 'feed-forward' mode, such that they process data in a single direction only. Feed-forward processing via multiple densely connected compute layers, each of which is usually passed through a non-linear function, produces sequentially arranged *latent representations* of the data. These representations are often called activations or latent features. Optimising the parameters of a neural network is typically done using gradient-based approaches [Rumelhart et al., 1986, Kingma and Ba, 2015] that involve computing the gradient of the loss with respect to the parameters. This loss depends on the proxy objective. For image data, convolutional neural networks (CNNs) [LeCun et al., 1998] are ubiquitous as they enable translation invariant feature detection to map data to representations that demonstrably generalise better to unseen data and tasks.

The strength of CNNs is partly owing to weight sharing, since a convolution involves computing features via a small kernel applied to many local regions in the data/representation. Therefore, the features produced via the convolution of a kernel are invariant to the relative position of those features in the data/representation. Sharing weights like this improves the compute and learning efficiency of CNNs – small local regions can be viewed as different data points, effectively expanding the dataset. Nonetheless, neural network depth (number of compute layers) and capacity (which is related to the total number of parameters) still strongly dictates how powerful a CNN can be and issues do arise when training very deep neural networks [Xiao et al., 2018]. Notably, vanishing gradients [Hochreiter, 1998] occur during backpropagation where the computation of the gradient with respect to the weights gets vanishingly small for layers closer to the inputs. Batch normalisation [Ioffe and Szegedy, 2015] and skip connections [He et al., 2016a,b] were innovations that enabled learning very deep neural networks. These are the core technical components to the backbone models – residual neural networks (ResNets) – I use throughout this Thesis.

Architecture choice also impacts the generalisation capability of a neural network.

A large model will have a tendency to overfit without sufficient regularisation [Krogh and Hertz, 1991, Kingma and Ba, 2015, DeVries and Taylor, 2017, Kingma and Ba, 2015, Zhang et al., 2021a]. In this Thesis I use standard approaches to identify and mitigate against overfitting: cross-validation, data augmentation, and weight decay [Loshchilov and Hutter, 2019]. Cross-validation is a re-sampling method where different subsets of the data are used for training and testing in order to estimate the performance of a model on various splits of the data. Data augmentation involves applying realistic changes to the image data (e.g., colour/lighting changes or horizontal flips) as a pseudo expansion of the training data. Weight decay adds a quadratic penalty on the weights of a neural network, resulting in relatively smaller weights and better generalising neural networks.

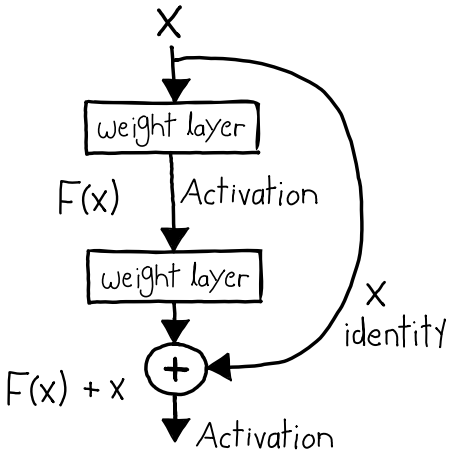


Figure 2.1: A residual compute 'block' introduced by He et al. [2016a]. The skip connection enables the weight layers to model a residual function, pass information forward without computing anything, and pass gradient information efficiently during backpropagation.

The ResNet He et al. [2016a] introduced the idea of residual connections for neural networks. Residual connections enable learning much deeper networks that generalise well to unseen data and which do not suffer from gradient degradation issues such as vanishing gradients. They reformulated the standard neural network layer to learn a residual function as opposed to the standard unreferenced function. Figure 2.1 illustrates the residual connection method in the form of a basic compute unit called a residual block. The residual block consists of: (1) multiple weight layers, usually convolutions; (2) activation functions; and (3) a skip connection. Intuitively, the resid-

ual connection helps to pass information forward through the network and also to pass gradients backward through the network during optimisation. Batch Normalisation [Ioffe and Szegedy, 2015] is applied before each activation function – it is a layer-wise technique that centres and re-scales the input on a minibatch basis to improve training stability and model performance. The standard variant of the ResNet consists of stacked residual compute blocks (Figure 2.1). While many architectural advances have been explored since the standard ResNet variant was introduced [Xie et al., 2017, Huang et al., 2017, Zagoruyko and Komodakis, 2016, Hu et al., 2018], the standard variant is used through this Thesis.

2.1.2 Supervised versus unsupervised learning

The main difference between supervised and unsupervised learning is the presence of target data. **Supervised learning** involves fitting a model that maps input data X to predict some target quantity Y. This target data can be continuous (for regression) or categorical (for classification). Supervised dimension reduction also incorporates target data (e.g., linear discriminant analysis [Izenman, 2013]).

Unsupervised learning, on the other hand, works by modelling the inherent structure in the input data without any pre-specified target data. The differences between various unsupervised learning methods are a consequence of the design choices, limits, and algorithms of those methods. For example, different cluster analysis methods (e.g., k-means [Lloyd, 1982] versus spectral clustering [Ng et al., 2001]) find substantially different clusters in data because of the underlying prior assumptions being made by algorithms used to compute cluster assignments. Unsupervised dimensionality reduction [Van Der Maaten et al., 2009], cluster analysis [Driver and Kroeber, 1932], manifold learning [Cayton, 2005], novelty detection, and density estimation are examples of unsupervised methods. Unsupervised methods can be assessed using supervised information (e.g., ground truth class labels to assess clustering).

Self-supervised learning [Ando et al., 2005] is a widely used approach for training neural networks to map image data to useful representations. The key to self-supervised learning is incorporating *low-cost* prior knowledge in a way that enables learning representations that perform well on downstream tasks. It is expensive (in terms of both money and time) to label many images. In contrast, defining self-supervised objectives can be relatively inexpensive; e.g., image augmentations are inexpensive to define or choose and they can be used to drive self-supervised repre-

sentation learning. Creating proxy objectives (Section 2.2) for image data is relatively easy (compared to tabular data or time-series data, for example) because of how easy it is to generate reasonable transformations of an image and to define appropriate losses that leverage these transformations. Context prediction [Doersch et al., 2015], information maximisation of different views of an image [van den Oord et al., 2018], and rotation prediction [Gidaris et al., 2018] are proxy objectives where additional knowledge about image data is used to drive self-supervised representation learning.

In Section 2.2 I discuss several proxy objectives that leverage self-supervision. I identify and discuss how neural networks prefer to model easy solutions to these proxy objectives in Section 2.3, and discuss three focus examples that typify this issue. These examples are explored and addressed in Chapters 3, 4, and 5.

2.2 Proxy objectives

A *proxy objective* is a training objective constructed using data and additional (usually inexpensive) prior knowledge about the data, such that this prior knowledge is not directly informative of the downstream task. Therefore, an objective is a 'proxy' when there is a disconnect between the training objective and the downstream task. Modern proxy objectives are usually formed by altering the input data in some automatic fashion (e.g., via image rotation) followed by predicting some automatically generated target value (e.g., the rotation angle). The intuition is that a useful representation must be learned to predict these quantities.

Formulating proxy objectives that result in good performance on downstream tasks can be challenging. Self-supervised learning has become popular for image data because of well-formed proxy objectives that can be optimised for using powerful neural networks. The impact of recent self-supervised methods (e.g., SimCLR [Chen et al., 2020a]) for common downstream tasks (e.g., object classification) is high – the downstream performance almost matches fully supervised training. Ericsson et al. [2021] studied how well self-supervised learning models transfer, and found that there was not a single method that dominated overall. I cover three broad areas that use proxy objectives in Sections 2.2.1, 2.2.2, and 2.2.3.

2.2.1 Information maximisation

The *infoMAX* principle involves using mutual information (MI) maximisation for representation learning [Linsker, 1988, Tschannen et al., 2019]. Self-supervised methods that use the *infoMAX* principle must define 'views' on the data to produce pseudo random variables as input to a MI maximisation procedure. It is challenging to define views in a reasonable fashion such that learning using the infoMAX principle results in good downstream performance. Furthermore, computing MI can be intractable and its estimation is not straightforward [Tschannen et al., 2019] or always possible.

Contrastive predictive coding [van den Oord et al., 2018] (CPC) models a 2D latent space using an autoregressive model and defines a predictive setup to maximise MI between distinct spatial locations. Deep InfoMAX [Hjelm et al., 2019] (DIM) does not maximise MI across a set of data augmentations, but instead uses mutual information neural estimation [Belghazi et al., 2018] and negative sampling to balance maximising MI between global representations and local representations (e.g., a crop of an image). Augmented multiscale Deep InfoMAX [Bachman et al., 2019] (AMDIM) incorporates MI maximisation across data augmentations and multiscale comparisons. Deep comprehensive correlation mining [Wu et al., 2019] (DCCM) constructs a sample correlation graph for pseudo-labels and maximises the MI between augmentations, and the MI between local and global features for each augmentation.

2.2.2 Contrastive learning

Contrastive learning is a popular method that is also based on the infoMAX principle. The proxy objective in this case involves mapping different views of the same data (e.g., image augmentations) to similar representations, while simultaneously encouraging representations from different data to have dissimilar representations. 'Contrasting' involves computing a loss that drives representations from positive sample pairs to be similar and representations from negative sample pairs to be dissimilar. Wang and Isola [2020] presented an informative perspective to understanding contrastive learning: that it is a balance between (1) maximising alignment between positive pairs of data, and (2) maximising the uniformity of the distribution of (normalised) representations on the unit hypersphere. Tian et al. [2020] suggested that good views for contrastive learning fulfil the criterion that they retain the minimal information necessary to perform the downstream task, and showed that the optimal views are necessarily dependent on the downstream task.

Instance discrimination Instance discrimination involves learning a mapping of image data such that each individual training image, and the same image processed using various augmentations, can be distinguished from all other images [Wu et al., 2018b]. This is essentially a N-way classification problem, where N is the number of training data points. The authors acknowledged that a parametric classification approach would require individual weight vectors per data point to act as a 'class prototype', which is infeasible for large datasets. Instead, they replaced these prototypes with the L2-normalised representations of each image, and kept an updated working memory of these representations during training. By sampling different data augmentations throughout training, the resultant representation must enable discrimination of training data while being invariant to the augmentations used. It is the use of data augmentations in this way that posits instance discrimination in the gamut of contrastive learning.

SimCLR SimCLR is framework for applying contrastive learning to image data [Chen et al., 2020a]. It operates in a similar manner as instance discrimination, and even uses a similar loss function. The main difference is that contrasting is done on a large minibatch basis as opposed to using a memory bank, which results in a better learning signal: the contrastive loss does not rely on out-of-date representations from a slowly updating memory bank. A disadvantage of contrasting on a minibatch basis is that the proxy objective is dependent on minibatch size, and downstream task performance is impacted by different choices of minibatch size. A number of architecture and learning adjustments were also included by the authors to facilitate the success of SimCLR. They emphasised the importance of choosing appropriate data augmentations, showing how stronger augmentations (i.e., those that alter the original image to a greater degree) can form a more challenging proxy objective that results in better downstream task performance. A sufficiently large neural network (i.e., a ResNet $4\times$ wider than usual) was able to achieve results comparable with supervised learning on ImageNet. An updated version of the framework [Chen et al., 2020b] further improved downstream task performance.

Momentum contrast One major drawback of SimCLR is that large minibatch sizes are required for good downstream performance. Momentum contrast (MoCo) [He et al., 2020] was designed to alleviate this issue by using a queue for encoding earlier minibatches. The solution was two-fold: (1) a momentum-based moving average

encoder – a slowly updated copy of the neural network – and (2) a queue mechanism that kept track of the most current momentum encoded representations. The queue can be large in order to retain a collection of diverse negative samples, thus alleviating the need for large minibatch sizes. The proxy objective in this case involves contrasting between the current encoded representations and their momentum encoded counterparts against all other momentum encoded counterparts.

Bootstrap your own latent Bootstrap your own latent (BYOL) [Grill et al., 2020] takes a different approach to mitigating the need for negative samples. BYOL used two networks: an online and a target network. The target network was updated similarly to MoCo using a moving average of the online network. Two augmentations are computed for a given image and processed by both online and target networks. The resultant representation from the online network is projected to produce a 'prediction' (of the same dimensionality as the representation). The target for this prediction is the representation computed by the target network (but without the final projection stage). One might expect that this setup admits collapsed solutions, where the network yields constant representations. The authors hypothesised that a combination of a slow-moving average for the target network and the prediction projection for the online network were enough to prevent collapsed solutions. Even without negative samples, this proxy objective still involves producing similar representations of an image under reasonable data augmentations.

2.2.3 Clustering

Clustering methods are designed to group data into two or more clusters. Cluster analysis was first used in the fields of anthropology and psychology to help draw conclusions about natural patterns in data [Driver and Kroeber, 1932]. Xu and Wunsch [2005] presented a survey of clustering methods that existed before deep learning became popular. The SK-Learn documentation [SKLearn, 2022] also provides an excellent overview of popular methods with corresponding visualisation of clustering results.

Early clustering methods had two key **features**: (1) they found natural patterns directly in the data, as opposed to building representations that are amenable to clustering (as is done with modern deep learning methods); and (2) the choice of data preprocessing (e.g., standardisation and whitening) and distance metrics had a large impact on the cluster analysis.

Perhaps the most well known clustering method is k-means clustering, first proposed by MacQueen [1967], and on which several modern deep-learning-based clustering methods are based [Fard et al., 2020, Wang et al., 2014a]. k-means aims to group data such that each datum belongs to the cluster with the nearest mean (of all member data). Fuzzy c-means [Dunn, 1973] extends this idea using a soft cluster attribution such that any datum can belong to more than one cluster.

Hierarchical clustering aims to group data into a hierarchy of clusters using either agglomerative or divisive methods. Agglomerative hierarchical clustering is a bottom-up approach where each datum starts in its own cluster and clusters are merged up the hierarchy. Divisive hierarchical clustering is a top-down approach where all data starts in a single cluster which is then split down the hierarchy. While divisive methods less widely used than agglomerative methods, divisive hierarchical clustering is closely related to the work presented in Chapter 3.

Spectral clustering uses dimensionality reduction by computed the eigenvalue decomposition of the data before performing clustering (with k-means, for example). Affinity propagation uses a complex message passing approach to cluster data, and is useful because it does not pre-require a specification of the number of clusters, unlike other methods. Affinity propagation results in exemplar data for each cluster. Density-based spatial clustering of applications (DBSCAN) operates under the assumption that clusters of data exist in high-density regions in space that are separated by low-density regions. A major advantage of DBSCAN over simpler methods like k-means is that the clusters can be any shape (whereas k-means assumes clusters are convex shaped).

What was a feature is now a challenge Finding natural patterns in data is useful, particularly for a user who seeks to understand the presence or absence of features that define a dataset. This advantage, however, can become an issue for neural networks because of how powerful and flexible they are. A neural network can learn to construct a representation that minimises a clustering objective by relying on simple features (e.g., colour) while suppressing other features (e.g., object type). When this representation needs to be useful for other downstream tasks, it becomes challenging to constrain neural networks from building representations that satisfy the clustering objective but which do not generalise well.

Deep clustering Deep clustering proxy objectives enable learning the parameters of a neural network such that the mapping it computes is a discrete labelling of data.

Therefore, it is typically possible to measure the performance of such representations directly using ground truth labels. Conversely, continuous and unconstrained representations (produced by contrastive learning, for example) require additional learning to decode (e.g., logistic regression if the downstream task is classification). Tschannen et al. [2019] identified that the decoding setup impacted significantly downstream performance, and experiments done for Deep InfoMax [Hjelm et al., 2019] showed how selecting representations closer to the input data (i.e., the outputs of shallower layers in the network) could benefit some downstream tasks. Chapter 4 was written before some of the following methods existed – I have partitioned the following discussion accordingly.

Pre-DHOG Deep embedding for clustering (DEC) [Xie et al., 2016] jointly learns an embedding suited to clustering and a clustering itself. They argued that the notion of distance in the feature space is crucial to a clustering objective. DEC used no contrastive-learning component to encourage an augmentation invariant representation space. Instead, it used a self-reinforcing mechanism to iteratively increase model confidence in the predictions it made. Without any signal to pull different views to the same predictions (via a contrastive component, for example), this method is inherently limited: there is no mechanism to drive the neural network to form clusters that are invariant to differences in views (e.g., colour differences). This is in some sense a failure of the DEC objective since it lacks the necessary delimiting information (e.g., image augmentations) that might drive a neural network to learn a useful clustering.

Yang et al. [2016] proposed joint unsupervised learning of deep representations and image clusters (JULE). JULE adapted an agglomerative clustering approach [Rokach and Maimon, 2005] to deep clustering. Agglomerative clustering is a variant of hierarchical clustering where clusters are recursively merged to form larger clusters. JULE was designed to increase the affinity of the neural network's output representation toward agglomeration. The authors argued that JULE performs agglomerative clustering in the forward pass and representation learning in the backward pass. As with DEC, JULE does not utilise data augmentations and is therefore limited in that it can only model structures inherent in the data and does not rely on additional information for representation learning.

One could argue that there exists a learnable partitioning of the data that is informative of the downstream task (in this case matching clusters to a ground truth labelling by a one-to-one mapping of cluster to class), and by extension argue that the use of

self-supervised learning methods is unnecessary – we simply need to find the 'right' partitioning. Why do DEC and JULE fail to find a partitioning that performs as well as partitions found by methods that also use contrastive methods (discussion to follow) to drive learning? There are no constraints on either the DEC or JULE objectives to drive learning clusterings of the data that are related to high-level semantic information.

One approach to constrain a representation is to use an autoencoder architecture to learn a bottleneck representation from which cluster assignments are computed. This setup ensures sufficient information retention such that the representation can be decoded back into image space, and thereby avoids cluster degeneracy (i.e., mapping all images to the same class). Ghasedi Dizaji et al. [2017] jointly optimised for image decoding and minimised relative entropy between latent factors in the representation. Fard et al. [2020] applied k-means in the representation space and defined a loss that reinforced the clusters found. The proxy objective for such methods can be seen as finding abstract features that form clusters and also enable decoding into the image space.

Deep adaptive clustering [Chang et al., 2017] (DAC) used a binary pseudo-labelling approach to perform clustering, where a positive binary label denoted a positive pair (i.e., both images in the same cluster). Cosine similarities of representations were computed between all images for each minibatch and an adaptive threshold parameter was learned such that the learned pseudo-labels produced pairs. The representation was constrained to tend toward a one-hot representation. A quadratic constraint was imposed on the representation to prevent collapsed predictions, where all images are mapped to the same cluster. Another mechanism that could be used to deal with collapsed solutions is to use a standard clustering algorithm, such as k-means, to iteratively group on learned features. This approach was used by DeepCluster [Caron et al., 2018]. Both DAC and DeepCluster used a pseudo-labelling proxy objective to reinforce the groups inherent in the structure of the data. The inadequacy of these types of proxy objectives is made apparent by the additional constraints they require to prevent collapsed solutions.

Associative deep clustering (ADC) [Haeusser et al., 2018] leveraged the idea that associations in the representation space could be used as a learning signal. What this means is that there are natural consistencies due to the structure of data that are present in representation space and that these can be latched onto and used to drive learning. They introduced centroids of the same size as the representation space and devised an objective to learn both the representation and centroids simultaneously. ADC used

four data augmentations per image and defined a loss that pulled their subsequent representations into the same clusters (in a similar fashion to contrastive learning).

Invariant information clustering (IIC) [Ji et al., 2019] used a neural network to predict a probability distribution over clusters for images, and maximised the mutual information (MI) between these predictions and others computed from differently augmented images. They effectively avoid collapsed solutions because MI maximisation implicitly targets marginal entropy and a collapsed solution would have lower marginal entropy than one that was not collapsed. IIC was used as the base method for DHOG in Chapter 4. That said, DHOG can be applied to any neural network that yields a distribution over clusters and is invariant to the underlying deep clustering method. I chose IIC when writing DHOG because it was the state-of-the-art deep image clustering method at the time. We will now discuss methods introduced after DHOG.

Post-DHOG Semantic clustering by adopting nearest neighbours (SCAN) [Van Gansbeke et al., 2020] used a two step approach: (1) a self-supervised method to learn a semantically meaningful representation; and (2) a clustering method that uses the nearest neighbours from these representations as drivers for learning. They tested instance discrimination [Wu et al., 2018b] and rotation prediction [Gidaris et al., 2018] as methods for learning the underlying representation. The strength of SCAN is directly related to the use of a self-supervised method.

Multi-model deep clustering (MMDC) [Shiran and Weinshall, 2021] involved training a neural network to align the learned representation with target points from a predefined Gaussian mixture model. Similar to SCAN, MMDC required an underlying self-supervised method (rotation prediction in this case) to learn useful representations of images. MMDC alternated between self-supervised and clustering objectives on a per epoch basis. Cluster allocation was determined by representation alignment with the Gaussian mixtures.

Matching priors and conditionals for clustering (MPCC) [Astorga et al., 2020] extended the autoencoder approach [Ghasedi Dizaji et al., 2017, Fard et al., 2020] using principles from generative adversarial networks (GANs) [Goodfellow et al., 2014]. The bottleneck latent representation was optimised to: (1) enable convincing decoded images (when assessed by a discriminator model as per the GAN framework); and (2) partition the data into clusters. Gaussian attention network for image clustering (GATCluster) used a number of self-supervised tasks to satisfy their constraints for a clustering algorithm and optimised toward one-hot encoding in cluster allocation prob-

abilities in order to ensure higher inter-cluster distances.

Prototypical contrastive learning (PCL) [Li et al., 2021] used an expectation maximisation (EM) setup to learn cluster centroids in an effort to bridge the gap between contrastive learning and clustering. The E-step involved applying k-means clustering to momentum encoded (in the same fashion as with MoCO) representations to produce k centroids (called 'prototypes' in this paper). The M-step was interpreted as maximising the log-likelihood of representations' alignment with their nearest centroids. The loss used for the M-step was similar to the contrastive loss used for SimCLR: alignment between a representation and its closest centroid was maximised and alignment between the same representation and all other centroids was minimised. An important caveat is that their method *also included the standard SimCLR contrastive loss to 'bootstrap' clustering*. Therefore, the proxy objective for PCL is similar to SimCLR but also includes a centroid approximation step.

The swapping assignments between multiple views (SwAV) method [Caron et al., 2020] also operates by maximising alignment between representations and centroids. SwAV is 'online' in the sense that it does not require an expectation step to compute cluster centroids (while PCL does), but instead learns centroids as model parameters. SwAV works by first processing two augmentations of a single image to produce corresponding representations (R_1 and R_2). The alignment between these representations and all centroids is then computed to select the closest centroids (C_1 and C_2). Opposite representations and centroids are then paired (C_1 with R_2; C_2 with R_1) and alignment between these pairs is maximised using a standard contrastive loss, where negative samples are other centroids instead of representations from other data. SwAV enables learning cluster assignments and centroids in an end-to-end fashion.

2.3 Proxy objective failure modes

In this section I identify and describe the tendency of neural networks to rely on easy-to-compute features in the data when trained using proxy objectives. What is meant by easy-to-compute features is situation dependent. For example, an easy feature for clustering might be average colour, while a complex feature might be object type. Intuitively, average colour is simple to compute, while object type might require a large neural network to compute. The tendency of neural networks to learn a mapping that relies on easy features exposes the failure of proxy objectives to mitigate against such reliance.

The failure of proxy objectives used in contrastive learning has recently been discussed by Chen et al. [2021] and Li et al. [2020b] – the problem is referred to as feature suppression in these works. They argued that the inclusion of stronger data augmentation to the contrastive learning framework was the implicit mechanism to mitigate against reliance on 'irrelevant features'. Wang et al. [2021] used the information bottleneck principle as a basis to explain that the 'minimal sufficient representation' for contrastive learning compresses away information that might be relevant for downstream tasks, and improved downstream performance by explicitly increasing the mutual information between the learned representation and input data. Li et al. [2020b] also retained information by decoding a representation to predict the input. Robinson et al. [2021] explored how well contrastive learning avoids shortcut solutions under different strengths of image augmentations, and proposed an implicit feature modification scheme to drive contrastive learning toward capturing a greater variety of features.

The concept of a proxy objective need not be limited to self-supervised learning. A standard classification objective (e.g., cross-entropy between predictions and ground truth labels) can also be cast as a proxy objective. In this case, any difference between the proxy objective and the downstream task is owing to differences between train and test data, instead of differences between the training loss downstream evaluation. Casting common training objectives as proxy objectives helps us to characterise their failure modes.

Jo and Bengio [2017] identified a possible cause for the sensitivity of high performing CNNs to imperceptible adversarial perturbations in the data [Szegedy et al., 2013]: that these neural networks learned to detect objects by computing low-level image statistics instead of detecting high-level abstract features. Earlier research provided evidence that such statistics were often sufficient for scene categorisation [Torralba and Oliva, 2003]. The proxy objective (which is the standard supervised classification objective in this case) failed in that it could not delineated between detection via statistics versus detection via abstract features – I address a similar issue in Chapter 5. Geirhos et al. [2019] studied how convolutional neural networks relied on texture as opposed to shape, even when shape reliance better optimised the proxy objective.

2.3.1 Over-compression

One perspective used to explain why neural networks with many parameters generalise well to unseen data is the information bottleneck (IB) principle [Tishby and Zaslavsky,

2015, Shwartz-Ziv and Tishby, 2017]. It posits that the internal representations produced by a neural network learn to compress *task-irrelevant information* while retaining maximal mutual information with the targets. This compression happens gradually after an initial information increase stage. Shwartz-Ziv and Tishby [2017] credited this to a diffusion process and claimed that the stochastic nature of stochastic gradient descent (SGD) optimisation produced this diffusion process. Prior to this Thesis I explored the applicability of the IB principle to a ResNet [Darlow and Storkey, 2020] and showed that compression did indeed occur in a real-world image classification setup. Elad et al. [2019] also validated the IB principle in a layer-wise learning setup by employing an objective function derived from the IB bottleneck principle to learn the internal representations of a neural network. They also explored the relationship between weight decay and the IB principle and did not observe compression without weight decay.

In some circumstances over-compression can be countered with suitable regularisation techniques like weight decay [Krogh and Hertz, 1991], L2-normalisation, or dropout [Srivastava et al., 2014]. The downstream task of mapping images to semantic hash codes for image retrieval (Chapter 3) is an example where regularisation does not suffice. Exisiting methods tends to produce hash functions that use only a small portion of the available hash space, meaning that images from the same class often get mapped to identical hash codes. This mapping is therefore insufficient for comparing images in the same class. This manifestation of over-compression results in reduced retrieval performance and poor coverage of the hash space.

2.3.2 Clustering local minima

Modern clustering methods usually incorporate additional constraints in the form of self-supervision to prevent forming partitionings of the data that perform poorly on downstream tasks (see Section 2.2.3). Self-supervision is used because clustering objectives alone do not necessarily enable learning neural network parameters that compute partitionings according to abstract features. There is no reason to expect a clustering objective to encourage partitioning by object type (an abstract feature) instead of partitioning by brightness (a low-level feature), for example [Grimmer and King, 2011].

I describe and demonstrate in Chapter 4 that greedy SGD optimisation results in neural network parameters that correspond to a sub-optimal local minima of the clus-

tering objective. Sub-optimality is two-fold here: (1) the clusters computed by the neural network are *non-robust* in that they do not correspond well with ground truth class labels; and (2) there is a relatively better local minima of the same clustering objective. Self-supervision can improve robustness, but it is nevertheless a symptomatic treatment of the underlying issue: neural networks trained using SGD will tend to rely on easy-to-compute features when available.

I approach this issue differently in Chapter 4. Intuitively, I first let the neural network learn the 'easy' solutions to the clustering objective (those that rely on non-robust low-level features), and then let it build a sequence of additional solutions that each have low MI with earlier solutions. The result is a set of solutions to the clustering objective that are arranged in a hierarchy of complexity (easy-to-hard). This approach has similarities with curriculum learning [Bengio et al., 2009]. I show that later (in the hierarchy) solutions tend to be more robust in that they perform better on the downstream task.

2.3.3 Neural network bias

Neural network bias is another problem that results when a proxy objective fails. When there is more than one way to map from images to target predictions, neural networks will prefer to learn a mapping from easy-to-compute features as opposed to a mapping from complex features (Chapter 5). 'Easy' versus 'complex' is relative: brightness is easy while shape is complex, for example. Both mappings correspond to different parameters of the neural network, each of which in turn corresponds to a distinct local minima of the loss surface. It is well-known that highly non-linear neural networks can have different parameter configurations that correspond to similar local minima of a loss surface [Swirszcz et al., 2016]. The issue is that when fundamentally different mappings can be learned, the training objective fails to provide any information as to which of these mappings should be preferred.

There are situations where spurious correlations exist in the data, such that the training objective can be optimised by relying on this correlation instead of the causal signal. For example, if all aeroplanes in a object classification training dataset contain blue sky in the background, a neural network can learn to map blue sky to the aeroplane class. This correlation is spurious when it prevents the neural network from learning to detect the shape of an aeroplane. Over-reliance on spurious signals result in non-robust classifiers that perform poorly at test time.

In Chapter 5 I demonstrate neural network bias by modifying training images with easy-to-compute features that correlate with target classes. It is a common assumption in this field that spurious correlations are characterised by easy-to-compute features [Nam et al., 2020, Bras et al., 2020]. My solution is to first capture the spurious correlation with a 'simple' classifier attached to a latent representation of the data, and then to remove this correlation using a quantised latent adversarial walk. I then decode the augmented latent representation to the original image space and train a standard classifier with the decoded images.

2.4 Decision trees and neural networks

Decision trees for regression and classification have a long history [Loh, 2011, Criminisi and Shotton, 2013]. They are non-parametric models that are designed to infer decisions on a target variable by learning decision rules from features in data. As such, the structure and rules depend on the data itself. Fitting decision trees can result in overly complex structures that do not generalise well. Decision trees are particularly useful when interpretability is a concern – they are white-box models where individual decisions can be interpreted.

Hehn and Hamprecht [2018] proposed an expectation maximisation method for optimising probabilistic decision trees in an end-to-end fashion. An advantage of this approach was that during optimisation any given data point could be mapped through the full decision tree, while an annealing mechanism recovers a favourable property of decision trees – that a datum only passes through a subset of nodes. This is similar to the method I present in Chapter 3, but does not use stochastic gradient descent nor a neural network to parameterise the decision tree. Norouzi et al. [2015] found that the problem of finding optimal decision trees was related to structured prediction with latent variables, and consequently developed and presented an efficient and non-greedy optimisation technique.

Richmond et al. [2015] explored the relationship between stacked decision forests and CNNs. These decision forests were used as an initialisation strategy for CNNs such that only a small amount of labelled data was required for fine-tuning these networks. McGill and Perona [2017] explored the idea of dynamic routing for training neural networks. They hypothesised that complex decisions could be broken down into less complex sub-decisions, and used inspiration from methods such as random forests that use dynamic routing during inference, in order to amend neural network training (and

proposed three separate methods to do so).

Little research has been done to partner decision trees with neural networks, likely because these models are at odds with each other. The main use-case of decision trees is interpretability, while neural networks are black-box models. Yang et al. [2018] combined decision trees and neural networks for tabular data and showed how these trees 'self-prune' at both split and feature levels. Wang et al. [2014b] introduced the concept of neural-backed decision trees, where they constructed decision trees by running hierarchical agglomerative clustering on the pre-trained final weights of a classifier network.

Unlike standard decision trees, deep decision trees require some parameterisation of the decision structure, where decisions are not directly made on the input features. Chapter 3 presents a different take on deep decision trees by imposing a parametric decision tree structure apriori. The objective is then to learn the parameters of that decision structure using contrastive learning.

2.5 Deep semantic hashing

This section covers the literature related to deep semantic hashing (relevant to Chapter 3). Conventional hashing is the process of mapping data of arbitrary size to a fixed size hash code. A good hash function should minimise collisions of data in the hash space. Contrary to this, locality sensitive hashing (LSH) [Slaney and Casey, 2008] is when hash functions produce hash codes such that semantic distances in the data space are preserved in the hash space (usually using hamming distance). For example, cats are semantically similar and the hamming distance between hashes of different cat images should be close, but the hamming distance between hashes from cat images versus dog images should be comparatively large. LSH has applications in nearest neighbour search and data clustering. There is no single approach to LSH, but all approaches rely on the idea of mapping similar data to similar hash codes. Balancing between the requirements of (1) capturing semantic similarity in hash codes, and (2) minimal total collisions is challenging.

2.5.1 Semantic hashing

Images present a challenge for hashing since they are high-dimensional and contain semantic content that is usually equivariant to many spatial, lighting, scale, and textu-

ral changes. To deal with this, features can be computed from images prior to hashing [Deselaers et al., 2008]. How these features are computed is crucial to the performance of downstream tasks that might use the hash codes. Local binary patterns [Ojala et al., 1994], localised wavelet patterns [Mallat, 1996], and other hand-engineered descriptors were invented or repurposed for hash-based image retrieval [Zhou et al., 2017, Lowe, 1999, Ojala et al., 2002, Dubey et al., 2014, Jacob et al., 2014, Song and Tao, 2009, Jégou et al., 2011, Murala et al., 2012, Dubey et al., 2015, Gong et al., 2012, Fan and Hung, 2014, Dubey et al., 2016, Chakraborty et al., 2016]. Features from pretrained neural networks can be used to construct hashes [Sun et al., 2015, Babenko et al., 2014]. Training deep neural networks as semantic hash functions has been investigated for over a decade (see Section 2.5.2)

Before deep learning was applied to semantic hashing, iterative quantisation (ITQ) [Gong et al., 2012] was the leading technique to transform images into semantic hash codes. ITQ was formulated as an optimisation to find the best rotation of zero-centred data onto the binary hyper-cube. It is related to spectral clustering [Ng et al., 2001] and can be applied after dimension reduction techniques.

2.5.2 Deep learning for semantic hashing

Dubey [2021] presented a comprehensive survey on image retrieval methods using deep learning for the 2011 – 2020 decade. They largely focused on hash-based methods and argued that hash-based image retrieval is superior in retrieval quality and efficiency. They outlined a number of approaches to learning (supervised, unsupervised, and semi-supervised), architecture types, modalities, assessment criteria, and hash descriptor types (discrete versus continuous). In light of the advances enabled by deep neural networks, they pointed out that a retrieval system must: (1) be discriminative in that supervision information is well-utilised and hashes capture class distinctions; (2) be robust to distribution shift in the data (e.g., lighting and pose changes); and (3) enable fast image search by way of maximally compact hashes.

Autoencoders were among the first models to be used for learning deep hash functions [Kang et al., 2012, Krizhevsky and Hinton, 2011]. An important assumption is that the latent codes produced by (bottleneck or denoising) autoencoders should capture semantic features even without supervision. Babenko et al. [2014] used the latent vectors produced by a pretrained neural network, and a PCA-compressed variant thereof, as features to perform image retrieval.

2.5.3 Pairwise and triplet losses

Pairwise and triplet losses have been widely used for many years, providing improvements in a variety of fields. One such example is in face recognition, where Schroff et al. [2015] used a triplet loss to train a neural network to produce an efficient (128-byte) representation of a face.

Some of the first end-to-end deep hash function learning was accomplished using variations on the triplet loss [Chechik et al., 2010]. Training using a triplet loss involves minimising the distance from a reference data point (called an anchor) to positive samples and maximising the distances from the anchor to negative samples. Wang et al. [2014c] applied a triplet loss and a triplet sampling mechanism to train a neural network that produced a continuous representation for image retrieval. Xia et al. [2014] proposed learning a convolutional neural network as a hash function (CNNH). They separated the problem into: (1) approximate hash function learning by minimising the reconstruction error between a class label-based similarity matrix and hash code approximations computed by the network, and (2) supervised fine-tuning of the neural network such that classes could be predicted from the hash codes. Lai et al. [2015] extended this work into a single stage process (they called it DNNH) using a triplet loss approach to directly learn the hash function. DNNH required using a piece-wise threshold function to approximate quantisation.

Zhang et al. [2015] also used a triplet-based approach called deep regularised similarity comparison hashing (DRSCH). Weightings were applied on bits during learning to enable code-length manipulation by truncating low weighted bits. Li et al. [2016] introduced deep pairwise-supervised hashing (DPSH) as one of the first methods to integrate a pairwise loss into an end-to-end deep hash function learning framework. Training using a pairwise loss involves minimising the distance between sampled positive pairs and maximising the distance between sampled negative pairs. DPSH was extended to use a triplet loss – this is called deep triplet-supervised hashing (DTSH) [Wang et al., 2016]. Pairwise correlation discrete hashing (PCDH) [Chen and Lu, 2020] used a standard supervised objective along with a pairwise loss, where pairs were sampled according to their vicinity in representation space. Deep discrete supervised hashing (DDSH) [Jiang et al., 2018] used a pairwise loss to directly guide both the discrete coding procedure and feature learning.

2.5.4 Quantisation

Supervised discrete hashing (SDH) [Shen et al., 2015] introduced the hash as an auxiliary variable prior to linear classification and showed how a variant of greedy gradient descent (i.e., greedy in the sense of learning the hash function bit by bit) could be used to circumvent the non-differentiable binarisation. The deep hashing network (DHN) Zhu et al. [2016] was the outcome of exploring the effect of quantisation error. DHN used a loss that better matched to the hamming distance typically measured in image retrieval. Husain and Bober [2016] developed a feature aggregation approach that used local CNN features to build descriptors. Deep hashing, and the extension called supervised deep hashing (called SDH2, for convenience) [Lu et al., 2017] aimed to learn a hash function that maximised the variance of learned binary codes in an effort to improve the distribution of the resultant hash codes.

Zhong et al. [2016] proposed a simple solution to learning a deep hash function: apply a sigmoid activation function to the penultimate representation of a standard classification setup and discretise the output after learning. They achieved impressive retrieval results but also showed that the resultant hash codes were not well distributed over the hash space. Wu et al. [2017] argued that multi-label setups are most indicative of a hash function's performance and that retrieval metrics do not measure fine-grained semantic similarity. They endeavoured to make the best use of pairwise similarities by leveraging multi-label datasets and showed superior performance when measured using the Jaccard index [Jaccard, 1912].

HashNet [Cao et al., 2017] asserted that learning a deep hash function is made challenging because of the need for gradient approximations. Gradient approximations are needed because quantisation (e.g., taking the sign as output activation) is a non-differential procedure and therefore incompatible with gradient-based optimisation methods. They proposed an approach that involved annealing a hyperbolic tan function toward a binary output as training progresses. Additionally, they proposed an adjustment to a pairwise binary cross-entropy loss that accounted for degrees of similarity often found in multi-object, scene, and retrieval datasets (e.g., NUS-wide[Chua et al., 2009]). They showed how HashNet was well distributed according to a 2D t-distributed stochastic neighbour embedding (t-SNE) [Van der Maaten and Hinton, 2008]. Nonetheless, they never directly measured the distribution of the resultant hash codes as we do in Chapter 3. Gordo et al. [2017] explored how imbalanced data was a contributing factor to poor performance in image retrieval. Deep supervised discrete

hashing (DSDH) [Li et al., 2017] instead learned directly on discrete hash codes and proposed a greedy discrete cyclic coordinate descent method to approximate gradients for gradient descent.

Deep product quantisation (DPQ) [Klein and Wolf, 2019] used ideas from product quantisation [Jegou et al., 2010] to produce separate components of hash codes. These components were concatenated to form the full hash codes. Deep positional aware hashing (DPAH) [Wang et al., 2020] used learnable class-specific hash codes as pseudo targets for learning. The authors argued that it was a lack of 'global' information (e.g., a class-specific hash code for each class) that caused poor downstream performance. They also introduced a Kurtosis loss that penalised long-tailedness of the continuous representation immediately prior to binarisation. In effect, the difference between the continuous representation and binary codes was diminished. Deep hashing with anchor graph (DHAG) [Chen et al., 2019] leveraged information from the full training dataset to build a measure of 'global' similarity to learn a deep hash function. The authors argued that minibatching resulted in an inefficient use of global similarity information, and this inefficiency reduced downstream retrieval performance. They proposed the use of an anchor graph as a means to alleviate this. The core idea behind an anchor graph is to use a small number of representative data points as anchor points, such that the difference between any two data points can be computed implicitly using the anchor points.

Deep supervised hashing (DSH) [Liu et al., 2016] is an earlier work that proposed regularisation on the continuous representation such that it better matched its quantised counterpart. Semantics-preserving deep hashing (SPDH) [Yang et al., 2017] proposed a three-part loss function including (1) cross-entropy, (2) a discretisation criterion, and (3) an entropy maximisation term. This issue of hash code quality is raised in Section 2.5.7 – SPDH is an example that failed to test hash code distribution even though they proposed a mechanism to encourage a good distribution of hash codes. The neurons merging layer (NML) [Fu et al., 2019] was introduced to reduce redundancy between bits in a hash function. Reducing redundancy led to improved hash efficiency and better downstream performance

Deep Weibull hashing with maximum mean discrepancy quantisation (DWH) [Feng et al., 2021a] used a Weibull distribution [Rinne, 2008] to enable better control the of the distance between hash codes of positive pairs in the triplet loss framework. Maximum mean discrepancy was used to reduce the difference between the continuous and discrete hash codes. The authors of deep fisher hashing (DFH) [Li et al., 2019]

argued that working in the hash space was essential to ensure that the hash codes remained separable at inference. Deep balanced discrete hashing (DBDH) [Zheng et al., 2020] used the straight-through estimator [Bengio et al., 2013b] to operate directly on a binary space for optimisation. Deep polarised networks (DPN) [Fan et al., 2020] imposed a polarisation loss on the continuous output space to encourage binary-like outputs.

Morgado et al. [2021] introduced a different kind of solution to the quantisation problem. Their method started by pre-determining class-specific sets of weights to be applied on an embedding space (to be learned later) such that maximal discrimination between classes was achieved. To do this, they used a classical optimisation technique known as sphere packing [Tammes, 1930] to find maximally discriminating weights and ideas from ITQ [Gong et al., 2012] to rotate the weight space into its most binary solution. These weights – called hash-consistent large margin (HCLM) proxies – were then frozen to train the rest of the network.

Su et al. [2018] argued that it was advantageous to avoid continuous approximations of discrete hash codes, and proposed a gradient passing method for back propagation to target the hash space directly. They called this Greedy Hash because discretisation can be thought of as a greedy process. Policy gradient deep hashing (PGDH) [Yuan et al., 2018] addressed the non-differentiability of discretisation by recasting it as sampling with a stochastic policy. By lending ideas from reinforcement learning they were able to maximise the expectation of rewards for preserving similarity directly in hash space. Discrepancy minimising deep hashing (DMDH) [Chen et al., 2018] used Taylor series expansions of the training objective to resolve discrepancies between continuous representations and their discretisations. They effectively minimised this discrepancy by gradually increasing the weight of higher order terms in the Taylor expansion.

2.5.5 Targeting properties in Hamming space

Cao et al. [2018] argued that Hamming space retrieval [Norouzi et al., 2012] is a necessary consideration since it is more time-efficient than linear scan or approximate nearest neighbour search. The Hamming space retrieval algorithm operates on binary hashes and has sub-linear run-time when the hash codes are approximately uniformly distributed. Hamming retrieval returns all data within a given hamming radius by way of hash table lookup. A ‘Hamming ball of radius 2’ around a query point refers to all

data with 2 or fewer bit-differences in hash codes. Deep Cauchy hashing (DCH) [Cao et al., 2018] used a probability transformation based on the Cauchy distribution. Their method penalised more strongly similar data points if they had a Hamming distance greater than the pre-set required Hamming radius (typically ≤ 2 since it is expensive for larger hamming balls). Concentrated hashing with neighbourhood embedding (CHNE) [Morgado et al., 2021] used the same probability distribution as DCH and argued that classification performance of the hash function was also important.

Maximum-margin Hamming hashing (MMHH) [Kang et al., 2019] accounted directly for a Hamming radius by explicitly characterising the Hamming ball with a max-margin t-distribution. Hu et al. [2022] presented boundary aware hashing (BAH) that used a contrastive loss and re-balanced the alignment and uniformity components according to whether pairs were inside or outside the Hamming ball (and whether they were positive or negative pairs). Wu et al. [2018a] proposed deep index-compatible hashing (DICH) to learn deep hash functions that used the multi-index Hamming ranking (a further refinement of Hamming space retrieval) [Greene et al., 1994].

Zhang et al. [2019] argued that multi-label datasets (e.g., NUS-wide) should be treated differently to single-label datasets, and that 'similarity' can be defined by the number of shared labels in the multi-label setup. Their method, improved deep hashing network (IDHN), leveraged multi-label similarity for a refined pairwise weighting to improve retrieval performance.

2.5.6 Alternative learning paradigms

Deep incremental hashing network (DIHN) argued that new incoming classes posed a significant challenge for deep hash functions, and used ideas from online learning [Carliner, 2004] to tackle this issue. Deep spherical quantisation (DSQ) [Eghbali and Tahvildari, 2019] used ideas from multi-codebook optimisation [Martinez et al., 2018] to partition the hash learning procedure into (1) supervision, (2) quantisation, and (3) dictionary learning. The 'spherical' component of DSQ is because of L2-normalisation (i.e., a projection onto the unit hypersphere) prior to quantisation. Multi-granularity feature learning hashing (MFLH) [Feng et al., 2021b] combined global and local information to build hash codes and also proposed an approximate discretisation procedure that enabled gradient flow and reduced quantisation error.

Deep progressive hashing (DPH) [Bai et al., 2019] used saliency information to build recursively aggregated deep hash functions. A saliency map enables 'zoom-

ing' into regions with more salient information. DPH used a graph long short-term memory (LSTM) compute module to aggregate features at multiple levels of detail. The motivation for this approach was the observation that humans conduct an implicit 'nonsaliency-to-saliency' attention scheme when analysing image data. Xu et al. [2019] proposed DHA, a method that shifted, scaled, and adjusted the similarity loss in order to account for (1) gradient degradation when learning discrete representations, and (2) differences in similarity for different pairs. Multi-level supervised hashing (MLSH) [Ng et al., 2020] found that constructing the hash codes from multiple resolution scales in a CNN was advantageous.

Just-maximising-likelihood hashing (JMLH) [Shen et al., 2019] proposed a comparatively simple method for learning deep hash functions. JMLH used the deep variational bottleneck (VIB) method [Alemi et al., 2017] as a theoretical basis to bound the MI in the hash space between images and their class labels. They used a reparameterisation trick along with gradient estimation to enable end-to-end learning. The VIB approach naturally yields a regularisation term – the KL-divergence between the prior (pre-selected) and the learned posterior, where this learned posterior is the discrete output of the target representation in the neural network. As it happens, this regularisation amounts to improving the entropy of the learned binary encoding, which means that it should result in well-distributed hash codes. Nonetheless, hash code distribution was never adequately measured.

Asymmetric deep supervised hashing (ADSH2) [Jiang and Li, 2018] used only the query images to train the neural network because this was argued to be a more realistic setup. The hash codes for the database were learned directly too, but this data was never used to learn the neural network. Weighted multi-deep ranking supervised hashing (MDRSH) [Li et al., 2020a] used multiple hashes, ranked according to mean average precision on held-out data computed after learning, to improve retrieval performance. Zhang et al. [2016] used a layer-wise learning technique to train very deep supervised Hashing (VDSH) networks.

Deep spatial attention hashing (DSAH) [Ge et al., 2019] used a spatial transformer network (STN) [Jaderberg et al., 2015] to better localise salient information. The STN approach was combined with a standard network to produce local and global representations, respectively. Deep transfer hashing (DTH) [Zhai et al., 2020] used ideas from another area of machine learning called knowledge distillation [Hinton et al., 2015]. First, a teacher network was designed with a latent continuous representation of the same number of dimensions as the desired hash space. The teacher network was

trained by minimising cross-entropy between predictions and class labels. The continuous representation was then discretised and used as the target for a student network to emulate.

2.5.7 The question of hash quality

Zhong et al. [2016] inadvertently highlighted a pervasive issue in the field of supervised hash learning: **the distribution of hash codes is seldom assessed**. They evidenced how hashes from similar samples were nearly identical. HCLM proxies were even designed such that all images in the same class map to the same hash. One measure of quality of a deep semantic hash function is how well the computed hash codes capture semantic similarity (measured using standard supervised retrieval metrics – see Chapter 3). The other, often ignored, measure, is how efficient the hash function is at using the available hash space. The implications of efficiency on generalisation, transfer, and usefulness are explored in Chapter 3.

Some of the works discussed in this section mention the importance of hash quality and proposed means to deal with it, but never measured the distribution of hash codes, nor how well they transfer to other (similar) datasets (we do this in Chapter 3). For example, SPDH included a loss that encouraged hash codes with equal numbers of 0's and 1's. This approach is insufficient at distributing the hash codes over the available space as it is measured on a point-wise basis, as opposed to across all data or a mini-batch. SDH2 aimed to maximise the variance of hash codes but never measured their distribution, while the NML aimed to reduce bit redundancy but never evidenced any performance gains. There are only a few earlier works where the transferability and distribution of the learned hash codes was measured or accounted for – e.g., t-SNE for HashNet and DWH.

2.6 Debiasing and adversarial learning

The following literature is relevant to the work in Chapter 5 regarding latent adversarial debiasing (LAD).

Neural networks lack robustness to human imperceptible perturbation called adversarial examples [Goodfellow et al., 2015, Madry et al., 2018], other semantic transformations such as rotations or translations [Kanbak et al., 2018, Engstrom et al., 2019, Hendrycks and Dietterich, 2019], and more broadly to a domain shift in the input dis-

tribution [Gulrajani and Lopez-Paz, 2021]. However, most relevant to this work is the observation that neural networks tend to rely on easy-to-learn biases or features that do not generalise outside the training distribution [Geirhos et al., 2020]. In image classification, this is exemplified by the reliance of neural networks on the high frequency components in the image [Jo and Bengio, 2017]. In natural language processing, neural networks were shown to latch onto statistical cues such as the present or absence of individual words [McCoy et al., 2019].

Adversarial learning is a set of techniques particularly useful for improving the robustness of deep neural networks [Goodfellow et al., 2015, Madry et al., 2018]. Predominantly, prior work focused on applying adversarial learning to improving the robustness to small perturbations bounded in ℓ_p norm.

Adversarial learning was also applied to de-biasing models [Goel et al., 2021, Qiu et al., 2019, Zhang et al., 2018, Beutel et al., 2017, Edwards and Storkey, 2016, Arjovsky et al., 2019, Stachura et al., 2020]. Zhang et al. [2018], Beutel et al. [2017], Edwards and Storkey [2016] remove the information about the bias from the input or latent representation. Arjovsky et al. [2019] use adversarial learning to reduce reliance of a neural network on features that change between different domains. In contrast, LAD does not require annotated data on the presence of a bias.

Data augmentation has also been shown to be effective in improving robustness of deep neural networks to semantically meaningful perturbations [Fawzi et al., 2016, Hendrycks et al., 2020]. LAD can be seen as an automatic method for creating such augmentations.

LAD is most closely related to unsupervised methods to de-biasing models [Nam et al., 2020, Bras et al., 2020, Gowal et al., 2020, Bahng et al., 2020], but substantially differ in the assumptions made. Gowal et al. [2020] assume access to a disentangled representation with an identified factor that is not causally related to the label. Then they use a decoder to produce augmented images by mixing the spurious factor between different pairs of images. Specifically, they train StyleGAN [Karras et al., 2018], and use its first stage representation as the spurious factor. Their method is hence limited to image classification scenarios in which features identified by StyleGAN correspond to the learned biases. Bahng et al. [2020] require providing a biased model that heavily relies on the bias information in the dataset (e.g., a CNN that has small receptive fields that bias the model towards textural information). They train a debiased classifier by regularising its representation to be statistically independent from the representation learned by the biased model.

Similarly to Nam et al. [2020] and Bras et al. [2020], LAD hinges on a relaxed assumption that the spurious signal is easier-to-learn than the causal signal. Nam et al. [2020] assumes that the first examples cause learning to be biased, and trains a second network that has an intentionally high loss on these examples. Bras et al. [2020] filters out examples that can be classified using a simple linear model. These approaches are inspired by the phenomenon that neural networks first prioritise learning a consistent subset of *easy-to-learn* examples [Swayamdipta et al., 2020, Arpit et al., 2017].

In contrast to Nam et al. [2020], Bras et al. [2020] we do not assume that a subset of examples is free of bias. Instead, I modify all images using a gradient walk on the underlying image manifold to produce new images where spurious signals are decoupled from target information. I compare directly to Nam et al. [2020] and Bahng et al. [2020] and show markedly improved generalisation performance. These were the only available methods at the time of experimenting with LAD.

Finally, LAD is related to Minderer et al. [2020]. Similarly to LAD, they reduce the reliance of training on easy-to-learn features by performing an adversarial walk in the latent space of an autoencoder. The key difference is that they use the method to improve self-supervised learning, in which they note the self-supervised objective can be too easily optimised by relying on shortcut (easy-to-learn) features.

Chapter 3

Deep decision tree layer: learning efficient discrete representations

The focus of this chapter is on learning compact and efficient discrete representations. Given a fixed capacity *compact* representation, efficiency here means making full use of the capacity of that representation. Discrete representations are of particular interest because their capacity is known and efficiency is measurable by how well a neural network that maps images to representations makes use of this available capacity. Compact discrete representations can be used as hash codes to enable fast image retrieval because of the low-cost of comparing hash codes. Training neural networks as deep semantic hash functions has been researched for over a decade [Dubey, 2021] (see Section 2.5 for a literature review). A semantic hash function for images takes an image as input and yields a binary vector that represents semantic information from the image (e.g., object type). A deep semantic hash function is a neural network trained to map images to such semantic hash codes. Deep semantic hash functions are used for fast content-based image retrieval (CBIR). CBIR produces an ordering of a database of images according to a query image from most similar to most dissimilar, where the differences between the query and database hash codes inform this ordering.

Mapping images to compact and discrete representations is challenging because of the high degree of compression needed to achieve this mapping – images are high-dimensional and typically contain complex semantic content that is difficult to summarise succinctly into small discrete representations. Furthermore, working with discrete representations presents an additional challenge when training neural networks: optimisation of a neural network loss to learn parameters is usually done by gradient-based methods; discretisation operations (e.g., taking the sign of a value) are non-differentiable and would result in non-differentiable loss functions and thus are incom-

patible with gradient-based methods. Some earlier methods included a quantisation loss (e.g., deep Cauchy hashing [Cao et al., 2018]) to pull continuous representations closer to their discrete counterparts. Forcing continuous representations to be close to their discrete counterparts necessitates discarding potentially useful information.

The problem: inefficient discrete representations I demonstrate in this chapter that there is a pervasive problem with learning deep semantic hash functions: objectives proposed by a number of earlier methods fail to produce hash functions that make efficient use of the available hash space. Practically this means that a large number of images are mapped to a much smaller number of hash codes, even when the hash space is large enough for a one-to-one mapping from training images to hash codes – see Section 3.2.3. This proxy objective failure is one of over-compression (see Section 2.3.1) that occurs because (1) training with class labels typically encourages a high degree of compression, and (2) the aforementioned additional compression that occurs when discretising continuous representations.

A solution to over-compression: the DDTL I consider the usual setting of leveraging a labelled dataset to learn hash codes Dubey [2021]. To overcome the issue of *over-compression* and improve hash code coverage for better information efficiency, I compose the hash code from a *supervised* portion and an *unsupervised* portion. Both portions are computed from a shared backbone, but with the former learnt using a classification head and a standard supervised objective, and the latter using a novel model structure – a deep decision tree layer (DDTL) – and a contrastive learning mechanism. The supervised component is constructed to be a minimal binary representation that captures fully the class label(s). For the unsupervised component, the DDTL provides a binary tree partitioning of the image space; each image is mapped to a binary vector that describes traversal through the tree (from root to leaf). This *decision path* builds the unsupervised portion of the hash. An advantage of our approach is that discretisation operations are never used during learning, mitigating one of the challenges associated with learning discrete representations. The DDTL enables:

1. efficient and ordered (in terms of bit-importance – see Section 3.1.1) semantic hash codes that are robust to realistic changes in input images (e.g., lighting changes, see Section 3.3.2);

2. better transfer to more complex or challenging datasets because of improved

information retention (Section 3.2.5);

3. image retrievals that respect fine-grained intra-class differences (Section 3.2.4);

4. neural network interpretability and dataset exploration by way of decision tree analysis (Section 3.4)

The DDTL matches or improves state-of-the-art mean average precision (*mAP*) on CIFAR-10 (with $mAP = 0.953 \pm 0.002$), ImageNet100 (with $mAP = 0.804 \pm 0.002$ without pretraining and $mAP = 0.872$ with pretraining to match setups with the previous state-of-the-art), and comparably performance on NUS-wide-21 (with $mAP = 0.897 \pm 0.002$). Regarding hash function efficiency, several methods from the literature are compared to the DDTL in Section 3.2.3, and I show that the DDTL is the only method that performs well at the task of retrieval and that has low collisions in the hash space. A new baseline is proposed in Section 3.3.1 that combines a classifier (for accurate supervised retrievals) and repeated PCA on the deepest latent representation of this classifier. This baseline is designed to be directly comparable to the DDTL in order to understand which is more robust to realistic changes in the image space (Section 3.3.2). Furthermore, a DDTL trained on CIFAR-100-20 (coarse grained super-class labels, 20 classes) matches the underlying fine-grained labelling (100 classes) with an accuracy of $42.13 \pm 0.39\%$ – see Section 3.2.5.

This chapter is organised as follows: Section 3.1 describes the DDTL in terms of architecture design and objective functions, and explains how the bit-ordering of the resultant hash codes is related to levels of semantic similarity; Section 3.2 gives the experimental results of comparing supervised retrieval performance of DDTL to earlier methods; Section 3.2.3 measures the efficiency and quality of hash codes from several methods, including the DDTL; Section 3.3 provides a number of ablation studies, including experiments in the fully unsupervised training setup in Section 3.3.4 and the supervised-only setup in Section 3.3.5; and Section 3.4 demonstrates how decision paths can be used for interpretability and dataset exploration.

3.1 Method: deep decision tree layer

Figure 3.1 gives an overview of the DDTL. Consider a (directed) binary tree T of depth d, directed from the root to leaves. Let $i = 1, 2, \ldots, 2^d - 1$ index the nodes in the binary tree from the root to the leaves. Let $k = 1, 2, \ldots, 2^{d-1}$ index the leaf nodes of the tree.

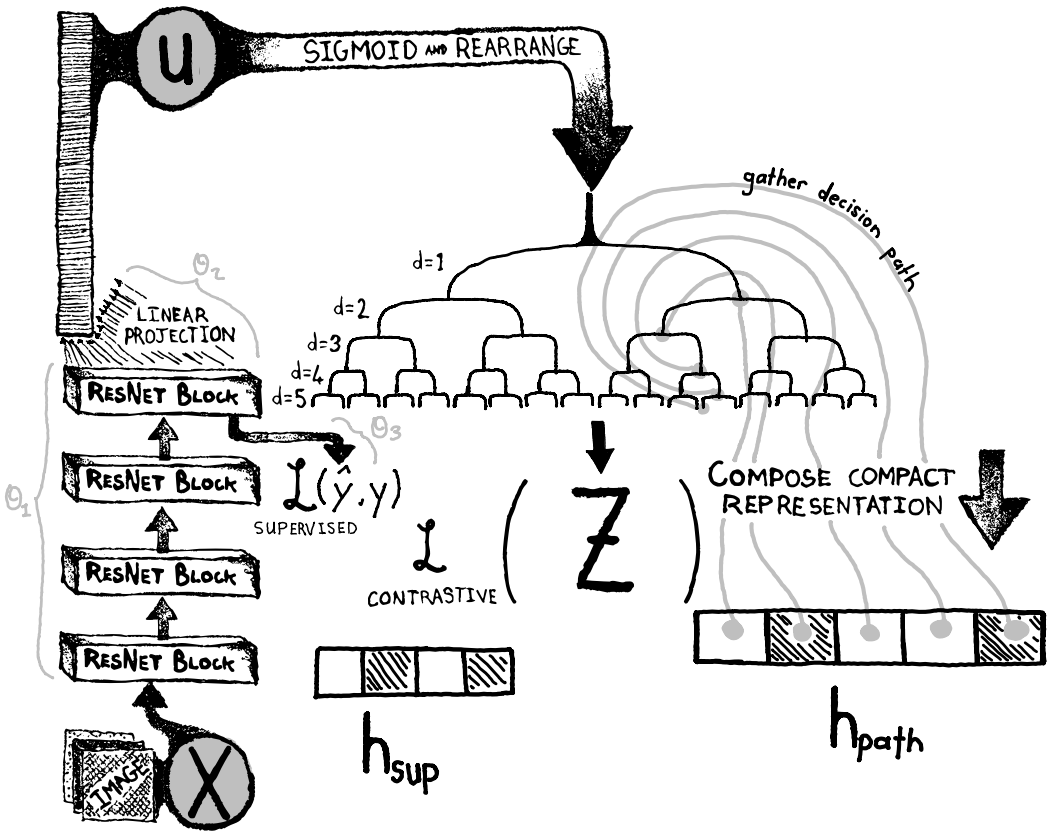


Figure 3.1: Deep Decision Tree Layer overview. An image is processed through the backbone neural network (a ResNet-34 in this case) to a standard sized representation. Two linear projections are then applied to (1) predict the class label(s), and (2) parameterise the decision tree. The latter is a vector that is then rearranged to compute the probabilities in the decision tree. The hash code is composed by concatenating the class prediction (in binary) and the decision path through the tree (see Figure 3.2).

We will consider an intermediate representation vector $\boldsymbol{u}$ with length $2^d - 1$, i.e. each element in $\boldsymbol{u}$ corresponds to a non-leaf node in the tree T; these will come to determine choice probabilities in the tree, effectively treating the tree as a decision tree. We will also consider a final representation vector $\boldsymbol{z}$ with length 2^{d-1}, i.e. each element in $\boldsymbol{z}$ corresponds to a leaf node of the tree T.

Now, for each leaf node k, let $path(k)$ denote the set of all nodes i in T on the path from the root to the leaf k (exclusive of 1 and inclusive of the leaf). Let $Pa(i)$ denote the parent of node i in the tree. Let $s_i \in (0,1)$ denote whether i is the left child ($s_i = 0$) or right child ($s_i = 1$) of $Pa(i)$ in binary tree T.

By way of further notation, let $\sigma(r) = 1/(1+\exp(-r))$ denote the sigmoid function, and overload the vector function $\boldsymbol{\sigma}$ to be the sigmoid vector function for scalar

arguments (returning a two-vector of probabilities) or a softmax vector function for (non-scalar) vector arguments:

$$\sigma_j(\mathbf{r}) = \frac{\exp(r_j)}{\sum_{j'} \exp(r_{j'})} \tag{3.1}$$

$$\boldsymbol{\sigma}(r) = (\sigma(r), 1-\sigma(r))^T. \tag{3.2}$$

Additionally, we define a thresholding function as:

$$H(r) = \begin{cases} 1, & r > 0.5 \\ 0, & r \leq 0.5 \end{cases} \tag{3.3}$$

Model definition With the notation defined, we can progress to define the model. Given an image x, parameterise the vector $\boldsymbol{u}$ using

$$\boldsymbol{u} = \boldsymbol{g}_1(\boldsymbol{f}(x, \boldsymbol{\theta}_1), \boldsymbol{\theta}_2), \tag{3.4}$$

where $\boldsymbol{f}(x, \boldsymbol{\theta}_1)$ is a neural network with parameters collected as $\boldsymbol{\theta}_1$, and $\boldsymbol{g}_1$ is linear projection with parameters collected as $\boldsymbol{\theta}_2$. All these parameters need to be estimated.

For each element $k = 1, 2, \ldots, 2^{d-1}$ of the vector $\boldsymbol{z}$, define

$$z_k = \prod_{i \in path(k)} \sigma(u_{Pa(i)})^{s_i} (1 - \sigma(u_{Pa(i)}))^{1-s_i}. \tag{3.5}$$

This equation defines $\boldsymbol{z}$ as the product of choice probabilities that are used to make each decision to eventually arrive at each leaf node in the decision tree.

Unsupervised hash In order to enable extracting a binary vector from the decision tree for a given image, we must first define a choice vector

$$c_k = \prod_{i \in path(k)} H(\sigma(u_{Pa(i)}))^{s_i} (1 - H(\sigma(u_{Pa(i)})))^{1-s_i} \tag{3.6}$$

to enable the selection of a single path $path(k^*)$, where $k^* = \arg\max(\boldsymbol{c})$, that effectively describes the decision path (see right hand side of Figure 3.2). Having defined $path(k^*)$ we can traverse this path from root to leaf to define a binarised hash

$$\boldsymbol{h}_{path} = (s_1, s_2, \ldots, s_{d-1}). \tag{3.7}$$

Supervised If there is supervised information about a particular classification of the images, a second linear projection $\boldsymbol{g}_2$ that yields an m-way class prediction can also be included. This prediction can either be used for single-label (i.e., one label per image) or multi-label (i.e., one or more labels per image) classification. Both of these setups are common in the literature (Section 2.5).

Single-label classification For single-label classification the class probability vector is computed using the m-way softmax function,

$$\hat{\boldsymbol{y}}_{single} = \boldsymbol{\sigma}(\boldsymbol{g}_2(\boldsymbol{f}(x, \boldsymbol{\theta}_1), \boldsymbol{\theta}_3)) \tag{3.8}$$

with parameters collected as $\boldsymbol{\theta}_3$. To compute supervised component of the hash code we need to first compute the class prediction $\hat{y}^* = \arg\max(\hat{\boldsymbol{y}}_{single})$. Following this we can convert $\hat{y}^*$ into a binary vector of length $\lfloor \log_2(m) \rfloor + 1$:

$$\boldsymbol{h}_{single} = \left(\lfloor \frac{\hat{y}^*}{2^0} \rfloor \bmod 2, \lfloor \frac{\hat{y}^*}{2^1} \rfloor \bmod 2, \ldots, \lfloor \frac{\hat{y}^*}{2^{\lfloor \log_2(m) \rfloor + 1}} \rfloor \bmod 2 \right), \tag{3.9}$$

where $\lfloor \, \rfloor$ is integer division. $\boldsymbol{h}_{single}$ is essentially the binary encoding of $\hat{y}^*$.

Multi-label classification For multi-label classification the class probability vector is computed using the sigmoid function applied to each element of the linear projection,

$$\hat{\boldsymbol{y}}_{multi} = (\sigma(g_{2,1}(\boldsymbol{f}(x, \boldsymbol{\theta}_1), \boldsymbol{\theta}_3)), \sigma(g_{2,2}(\boldsymbol{f}(x, \boldsymbol{\theta}_1), \boldsymbol{\theta}_3)), \ldots, \sigma(g_{2,m}(\boldsymbol{f}(x, \boldsymbol{\theta}_1), \boldsymbol{\theta}_3))). \tag{3.10}$$

The notation $g_{2,m}$ refers to the m^{th} element of the linear projection $\boldsymbol{g}_2$. To compute the supervised component of the hash code for multi-label classification we can apply the thresholding function to produce a likely-sparse vector

$$\boldsymbol{h}_{multi} = (H(\hat{y}_{multi,1}), H(\hat{y}_{multi.2}), \ldots, H(\hat{y}_{multi,m})), \tag{3.11}$$

that will be 1 at an index where the class is predicted as 'present' with a probability greater than 0.5, otherwise it will be 0 at that index.

Combined hash codes The supervised component of the hash code is therefore selected depending on classification setting as follows:

$$\boldsymbol{h}_{sup} = \begin{cases} \boldsymbol{h}_{single}, & \text{if single-label} \\ \boldsymbol{h}_{multi}, & \text{otherwise.} \end{cases} \tag{3.12}$$

The full hash code is the concatenation of both components (see Figure 3.2):

$$\boldsymbol{h} = \boldsymbol{h}_{sup} \oplus \boldsymbol{h}_{path} \tag{3.13}$$

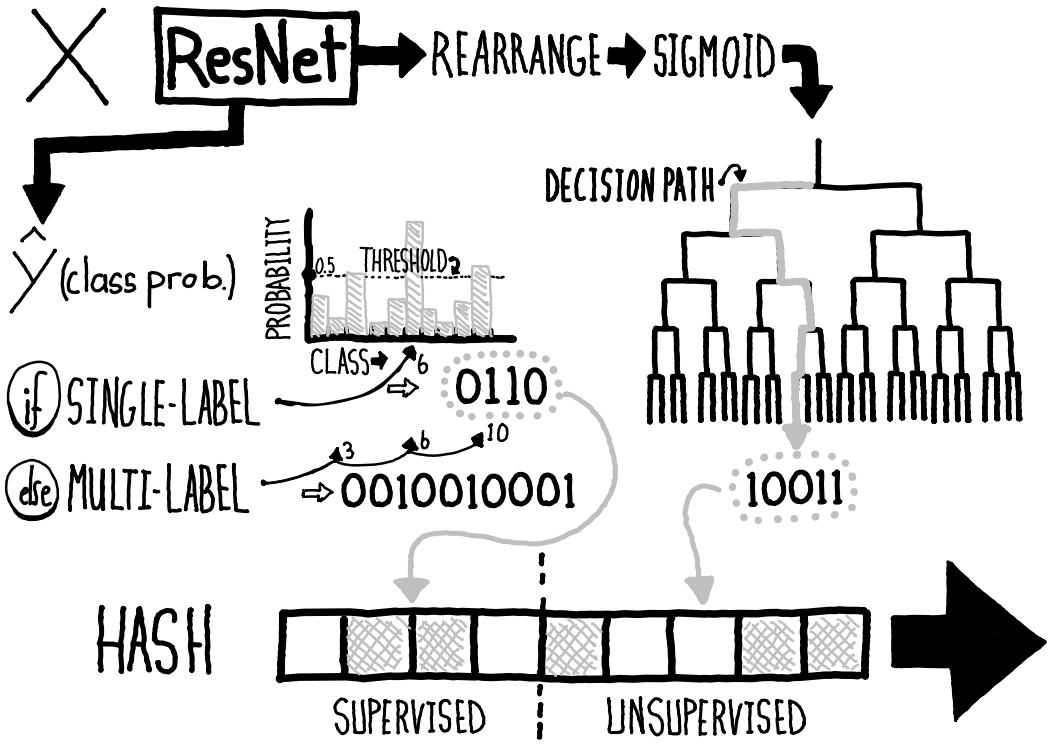


Figure 3.2: Construction of full hash code from an image. The supervised component of the hash code is constructed by transforming either single-label or multi-label class probabilities ($\hat{\boldsymbol{y}}$) into a binary vector. The unsupervised component is the decision path (see Figure 3.3 for more details).

3.1.1 Building hash codes in two parts

Figure 3.2 shows, for a given image, how a hash code is constructed using the class predictions, $\hat{\boldsymbol{y}}$, and the decision path through the tree, $path(k^*)$. Figure 3.3 shows an alternative intuitive perspective of how the decision path is extracted for an image. The goal of the decision tree is not to model the class labels (that is accomplished succinctly using $\hat{\boldsymbol{y}}$) but rather to capture additional class-unrelated information inherent in the images.

Relationship to divisive hierarchical clustering Since $path(k^*)$ describes which nodes an image is allocated to at every depth of the decision tree, the collection of decision paths for multiple images can be viewed as a divisive hierarchical clustering mechanism [Johnson, 1967]: at each level of the decision tree images that followed the same path are clustered together. Note that the discretisation needed to build the full hash code is not used during learning, meaning that no quantisation loss is required

and no gradient propagation tricks are necessary to stabilise learning as is needed by some earlier methods. The full hash code is then used for the downstream CBIR task.

An ordered hash code The concatenated hash output from the DDTL – Equation 3.13 and Figure 3.2 – is ordered. The first component of the hash captures the class information. The second component is the decision path that captures information to distinguish between images from the same class. The decision path is also ordered from root to leaf. The implication of this structure is that retrievals can be ordered to various degrees of granularity, ranging from class-level all the way to orderings that use the final decisions in the tree.

A sorted hash enables lexicographical ordering when performing CBIR. In practise this means sorting the retrieved data according to bit-wise differences between query and database images from left to right in their hashes. This type of lexicographical retrieval ordering enables a fast sub-linear time search, where each subsequent bit is only compared between query and a subset of database images, where this subset matches the query on all previous bits. The next sections describes how the DDTL is trained.

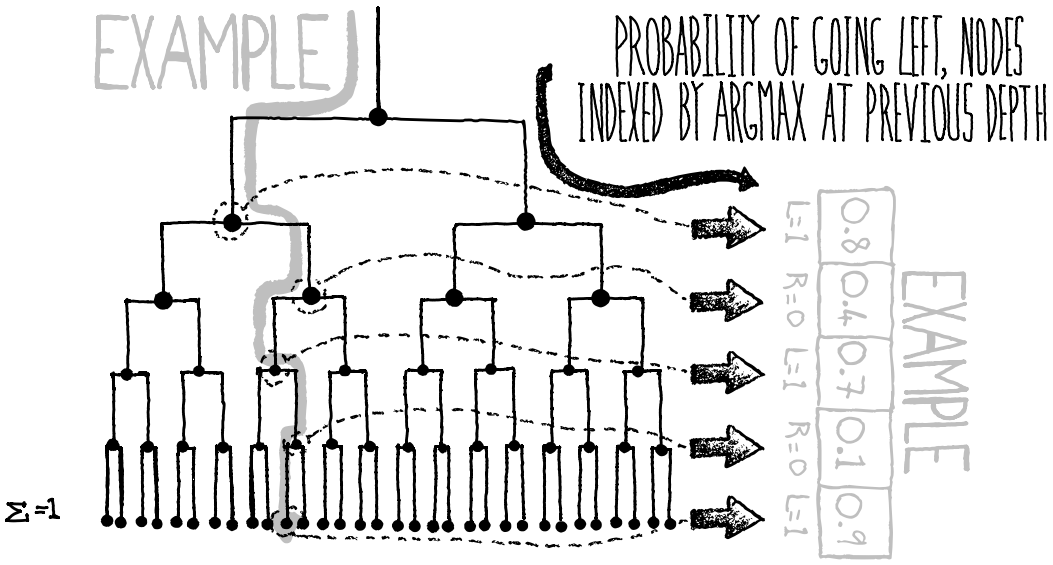


Figure 3.3: An alternative perspective of how the decision path constructed is constructed using decisions down the tree. The decision at each depth is the $\arg\max$ between the probability of going left versus the probability of going right.

3.1.2 Training

The parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3)$ require two losses to optimise: a supervised loss to learn the mapping to $\hat{\boldsymbol{y}}$ and a contrastive loss to learn the mapping to $\boldsymbol{z}$. The supervised loss between prediction $\hat{\boldsymbol{y}}$ and true label $\boldsymbol{y}$ is denoted by $\mathcal{L}_{\text{supervised}}$. This supervised loss is cross-entropy for single-label classification and an asymmetric multi-label loss proposed by Ben-Baruch et al. [2021] for multi-label classification.

Computing the contrastive loss involves first computing the cosine similarity between two $1D$ representations. The cosine similarity is the dot product between L2-normalised vector representations, $\boldsymbol{u}$ and $\boldsymbol{v}$ (placeholder names, not to be confused with $\boldsymbol{u}$ in Equation 3.4), and is defined by:

$$\text{sim}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^\intercal \boldsymbol{v} / \|\boldsymbol{u}\| \|\boldsymbol{v}\| . \tag{3.14}$$

The contrastive loss is then defined between a positive pair of representations (a, b):

$$l_{a,b}(\tau) = -\log \frac{\exp\left(\text{sim}(\boldsymbol{z}_a, \boldsymbol{z}_b)/\tau\right)}{\sum_{c=1}^{2N} \mathbb{1}_{[c \neq a]} \exp\left(\text{sim}(\boldsymbol{z}_a, \boldsymbol{z}_c)/\tau\right)}, \tag{3.15}$$

where $\mathbb{1}_{[c \neq a]} \in \{0, 1\}$ is an indicator function that is 1 *iff* $c \neq a$, τ is known as the 'temperature', and N is the minibatch size. Positive pairs (a, b) are representations computed from the same source image under two different augmentations. Negative pairs are constructed using all other images (two augmentations each) into $2(N-1)$ pairs, hence the $2N$ in the denominator. Following SimCLR [Chen et al., 2020a] the full contrastive loss is computed over all positive pairs, both (a, b) and (b, a), for each minibatch and aggregated into

$$\mathcal{L}_{\text{contrastive}}(\boldsymbol{z}, \tau) = \frac{1}{2N} \sum_{pairs} \left[l_{a,b}(\tau) + l_{b,a}(\tau) \right] . \tag{3.16}$$

Supervised and contrastive losses are combined for an element in the minibatch:

$$\mathcal{L}_{\text{DDTL}}(\hat{\boldsymbol{y}}, \boldsymbol{y}, \boldsymbol{z}, \tau) = \alpha \cdot \mathcal{L}_{\text{supervised}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) + \beta \cdot \mathcal{L}_{\text{contrastive}}(\boldsymbol{z}, \tau), \tag{3.17}$$

where α and β are loss scaling hyper-parameters. The contrastive loss is computed separately per-class in each minibatch and averaged over the classes and minibatch. This ensures the decision tree is making decisions according to class-unrelated features, instead of reinforcing the supervised cross-entropy signal. Early experiments evidenced that this resulted in more uniformly distributed hash codes.

For clarity, the contrastive loss in Equation 3.16 is applied for each separately separately in order to learn features that enable the unsupervised hash code in Equation

3.7. In some sense this loss is not strictly unsupervised because the class labels inform the negative samples for contrasting. Tian et al. [2021] proposed to 'divide and contrast' and applied contrastive learning to clustered subsets of data in order to improve results in challenging datasets. Khosla et al. [2020] also evidenced how contrastive learning can be adapted to use class labels and reported improved performance on a ResNet-200.

The DDTL parameters $\boldsymbol{\theta}$ are then optimised using a gradient based method to minimise the loss:

$$\arg\min_{\boldsymbol{\theta}} \mathcal{L}_{\text{DDTL}} \tag{3.18}$$

3.2 Experiments

Experiments were conducted on three datasets commonly used to assess deep semantic hash-based CBIR: CIFAR-10 [Krizhevsky et al., 2009], ImageNet [Deng et al., 2009] constrained to 100 randomly sampled classes (called ImageNet100), and NUS-wide-21 [Chua et al., 2009] constrained to the 21 most common classes.

3.2.1 DDTL training parameters

The neural network backbone for the DDTL was set to be a ResNet-34 (i.e., a standard ResNet architecture with 34 layers – see Section 2.1.1). The full model (backbone and two linear projections – see Section 3.1) was trained from scratch without any pretraining (unlike many existing methods – see Section 3.2.2). The Adam optimiser [Kingma and Ba, 2015] was used for optimisation. A cosine annealing learning rate schedule was applied to an initial learning rate of 0.001. For CIFAR-10: the model was trained for 500 epochs with a batch size of 512. For NUS-wide-21: images were resized to square images with sides of length 224, a batch size of 256 was used, and the model was trained for 500 epochs. For ImageNet100: a random square crop with sides of length 224 was taken from the input images, a batch size of 256 was used, and the model was trained for 200 epochs. The temperature τ for the contrastive loss was set to 1 for all experiments. The only hyper-parameters for the DDTL are the loss balancing factors α and β for supervision and contrastive losses, respectively. Cross-validation on CIFAR-10 was carried out to determine values of $\alpha = 1.0$ and $\beta = 0.1$. When error bars are given these are the standard deviation over three different random initialisations of the network parameters.

Results for the DDTL are compared to many existing methods in Section 3.2.2. Several methods (for which code was available or could be re-implemented) were trained from scratch using the same backbone architecture and training settings as the DDTL – analysis of the resultant hash functions is given in Section 3.2.3. Retrievals are given in Section 3.2.4. Results from several ablation studies are given in Section 3.3. Demonstrations of interpretability and dataset exploration are given in Section 3.4.

3.2.2 Comparison of the DDTL to other methods

Experimental methodology differs throughout the literature described in Section 2.5. The two most common experimental setups are:

1. *Partial* – starting with backbones pretrained on ImageNet, small subsets of images are used for training and retrieval. For CIFAR-10 and NUS-wide-21, 100 images per class are sampled as the query set, 500 images per class are sampled for training, and the remaining data is used as the retrieval database. For ImageNet100 the test set is the query set, 100 images per class are sampled for training, and the remaining data is used as the retrieval database.

2. *Full* – no sub-sampling of datasets is done in this setup, meaning that the test data of all three datasets are used as query sets, and the training data is used for training and as the databases. Pretraining is also not consistent.

The partial setup was dominant for earlier methods because of the computation cost associated with learning. Compute restrictions have since become less of a concern because of improvements to deep learning frameworks and hardware advances. Using a pretrained network makes it difficult to assess clearly the true capabilities of a method. For the DDTL all experiments were carried out on the full experimental setup without pretraining to enable a clearer evaluation of whether the DDTL can learn robust and efficient discrete representations.

Supervised retrieval assessment metric The metric used to compare retrieval performance is mean average precision (*mAP*). It is computed by evaluating the average precision over all images in a query set Q. The mAP_k refers to the *mAP* considering only the top k (called the rank threshold) retrieved images for each query image. It is defined as:

$$mAP_k = \mathbb{E}_Q \left[\frac{\sum_{k=1}^{K} P_k \delta(k)}{\sum_{k=1}^{K} \delta(k)} \right], \tag{3.19}$$

where P_k is the precision $\left(\frac{|\text{true neighbours} \cap \text{all retrievals}|}{|\text{all retrievals}|}\right)$ at rank threshold k. $\delta(k) = 1$ if the k^{th} retrieved result is a true neighbour of the query (they are of the same class for CIFAR-10 and ImageNet100 or they share at least one class for NUS-wide-21), otherwise $\delta(k) = 0$. For example, a perfect mAP_{1000} would be equal to 1 and would mean that the top $k = 1000$ retrieved images are true neighbours for every query image.

Tables 3.1, 3.2, and 3.3 give the *mAP* supervised retrieval results from the methods discussed in the literature review (Section 2.5) and gives the performance of the DDTL in the final rows. These tables are segmented according to the experimental setups for a more fair comparison. Unfortunately there is a broad disagreement in earlier literature regarding experimental setups, particularly when considering pre-training.

DDTL outperformed all other existing methods (from Section 2.5) at the supervised CBIR task for CIFAR-10 with a mAP_{5000} of 0.953 ± 0.002. For NUS-wide-21 DDTL achieved a mAP_{5000} of 0.897 ± 0.002. For ImageNet100 DDTL achieved a mAP_{1000} of 0.804 ± 0.002. The closest competitors on NUS-wide-21 and ImageNet100 were DTH and DPQ, respectively. Both of these methods used pretraining on a ResNet-50 model, meaning that it is impossible to compare directly the DDTL to these methods as they have access to more data via pretraining.

That said, an additional experiment on ImageNet100 was undertaken for a clearer comparison: the same pretrained ResNet-50 (used for DPQ) backbone was used for the DDTL and training was undertaken as before. This setup achieved a mAP_{1000} of 0.872 at 12 bits (an improvement of 0.028 over DPQ at 16 bits). The DDTL is the only method within the gamut of high-performing methods on large image datasets that does not require pretraining to achieve state-of-the-art supervised CBIR results.

Owing to the two-stage setup for the DDTL (Section 3.1) only 4 bits are needed for CIFAR-10 (since 4 bits enable 16 unique hash codes), 21 bits for NUS-wide-21 (since this is a 21-way multi-label setup), and 7 bits for ImageNet100 (since 7 bits enable 128 unique hash codes) to achieve the supervised CBIR results given in this section. The results for the DDTL listed in Tables 3.1, 3.2, and 3.3 are in the columns corresponding to the length of the supervised part of the hash code because the decision path is not needed for these results. In each case a decision tree of depth 12 was used to build the rest of the hash codes; the full informational capacities are 16 bits, 33 bits, and 19 bits, for CIFAR-10, NUS-wide-21, and ImageNet100, respectively.

Method	**CIFAR-10** *mAP*						*(using n bits)*
	4	12	16	24	32	48	64
Experimental setup: partial							
PCDH	-	-	0.843	0.856	0.861	0.868	0.875
DDSH	-	0.769	-	0.829	0.835	0.819	-
CNNH	-	0.465	-	0.521	0.521	0.532	-
DNNH	-	0.552	-	0.566	0.558	0.581	-
DPSH	-	0.713	-	0.727	0.744	0.757	-
DTSH	-	0.710	-	0.750	0.765	0.774	-
DHN	-	0.555	-	0.594	0.603	0.621	-
DSDH	-	0.740	-	0.786	0.801	0.820	-
DCH	-	-	0.790	-	0.798	0.807	0.794
CHNE	-	-	0.942	-	0.938	0.947	-
MMHH	-	-	0.792	-	0.818	0.825	0.819
BAH	-	-	0.850	-	0.843	0.841	0.840
GreedyHash	-	0.774	-	0.795	0.810	0.822	-
PGDH	-	-	0.736	-	0.741	0.747	0.762
DMDH	-	-	0.703	-	0.719	0.732	0.737
DPH	-	0.698	-	0.729	0.749	0.755	-
DHA	-	-	0.652	-	0.681	0.690	0.699
JMLH	-	-	0.805	-	0.841	-	0.837
DHAG	-	0.934	-	0.933	0.934	0.932	-
NML	-	0.786	-	0.813	0.821	0.828	-
DWH	-	0.726	-	0.741	0.755	0.765	-
SDH2	-	-	0.310	-	0.359	-	0.385
DPQ	-	0.744	-	0.745	0.755	0.760	-
MLSH	-	-	0.667	0.697	0.719	0.738	0.748
MFLH	-	0.726	-	0.758	0.771	0.781	-
ADSH2	-	0.847	-	0.906	0.918	0.926	-
DSHC	-	0.740	-	0.786	0.801	0.820	-
DSAH	-	0.822	-	0.841	0.845	0.849	-
DFH	-	0.803	-	0.825	0.831	0.844	-
DBDH	-	-	0.730	0.736	0.743	0.748	0.751
DPN	-	-	0.825	-	0.838	-	0.830
Experimental setup: full							
DSHC	-	0.935	-	0.940	0.939	0.939	-
DRSCH	-	-	0.615	0.622	0.629	0.631	0.633
DSAH	-	-	0.941	0.945	0.942	0.944	-
DTH	-	0.921	-	0.933	0.937	0.949	-
MDRSH	-	-	0.769	0.799	0.809	0.813	0.815
HCLM	-	-	0.945	0.947	0.949	0.950	-
DSH	-	0.616	-	0.651	0.661	0.676	-
DICH	-	-	-	-	0.8839	0.9047	0.9082
DSQ	-	-	0.7212	-	0.7346	0.7418	0.7589
ADSH	-	0.754	-	0.780	0.786	0.795	-
GreedyHash	-	-	0.942	0.943	0.943	0.944	-
DSDH	-	-	0.935	0.940	0.939	0.939	-
SDH	-	-	-	-	-	-	0.455
DDTL	0.953 ± 0.002	-	-	-	-	-	-

Table 3.1: Retrieval results for CIFAR-10. With the exception of the DDTL, all of these results were extracted from the literature (Section 2.5). Commonly mAP_{5000} and occasionally mAP_{50000} is measured in the literature; I listed mAP_{5000} for the DDTL. The rows are segmented according to experimental setup for simplified comparison. Error bars are standard deviation over 3 runs.

Method	**NUS-wide-21** *mAP*						*(using n bits)*
	12	16	21	24	32	48	64
Experimental setup: partial							
PCDH	-	0.810	-	0.826	0.832	0.836	0.839
DDSH	0.791	-	-	0.815	0.821	0.827	-
CNNH	0.623	-	-	0.630	0.629	0.625	-
DNNH	0.674	-	-	0.697	0.713	0.715	-
DPSH	0.794	-	-	0.822	0.838	0.851	-
DTSH	0.773	-	-	0.808	0.812	0.824	-
DHN	0.708	-	-	0.735	0.748	0.758	-
HashNet	-	0.662	-	-	0.699	0.711	0.716
DSDH	0.776	-	-	0.808	0.820	0.829	-
DCH	-	0.740	-	-	0.772	0.769	0.712
CHNE	-	-	-	0.751	0.723	-	0.622
MMHH	-	0.772	-	-	0.799	0.789	0.755
BAH	-	0.775	-	-	0.804	0.794	0.783
PGDH	-	0.761	-	-	0.780	0.786	0.792
DMDH	-	0.751	-	-	0.781	0.787	0.789
ADSH	0.780	-	-	0.808	0.815	0.823	-
DPH	0.770	-	-	0.784	0.790	0.786	-
DHA	-	0.669	-	-	0.706	0.721	0.727
JMLH	-	0.795	-	-	0.818	-	0.820
DPAH	-	0.816	-	-	0.827	0.835	0.828
DHAG	0.760	-	-	0.789	0.793	0.802	-
NML	0.801	-	-	0.824	0.832	0.840	-
DWH	0.794	-	-	0.819	0.828	0.835	-
MLSH	-	0.643	-	0.646	0.670	0.687	0.709
MFLH	0.782	-	-	0.814	0.817	0.824	-
ADSH2	0.857	-	-	0.894	0.901	0.907	-
MDRSH	-	0.701	-	0.719	0.724	0.737	0.741
DSHC	0.776	-	-	0.808	0.820	0.829	-
DSAH	0.885	-	-	0.902	0.898	0.906	-
DFH	0.795	-	-	0.823	0.833	0.842	-
DPN	-	0.847	-	-	0.859	-	0.863
IDHN	0.730	-	-	0.759	-	0.769	-
Experimental setup: full							
DSDH	-	0.815	-	0.814	0.820	0.821	-
DICH	-	-	-	-	0.768	0.769	0.774
DSQ	-	0.779	-	-	0.790	0.792	0.799
DSH	0.548	-	-	0.551	0.558	0.562	-
HCLM	-	0.814	-	0.825	0.830	0.835	-
DSHC	0.815	-	-	0.814	0.820	0.821	-
DSAH	-	0.834	-	0.856	0.883	0.901	-
DTH	0.876	-	-	0.900	0.907	0.917	-
DRSCH	-	0.618	-	0.622	0.623	0.628	0.641
DDTL	-	-	0.897 ± 0.002	-	-	-	-

Table 3.2: Supervised image retrieval results for NUS-wide-21. With the exception of the DDTL (final row), all of these results were extracted from the literature (Section 2.5). Commonly mAP_{5000} and occasionally mAP_{50000} is measured in the literature; I listed mAP_{5000} for the DDTL. The rows are segmented according to experimental setup for simplified comparison. Error bars are standard deviation over 3 runs.

Method	**ImageNet100** mAP_{1000} *(using n bits)*					
	7	16	24	32	48	64
Experimental setup: partial						
SDH	-	-	-	-	-	0.650
HashNet	-	0.506	-	0.631	0.663	0.684
CHNE	-	0.590	0.625	-	-	-
BAH	-	0.706	-	0.726	0.724	0.725
GreedyHash	-	0.625	-	0.662	0.682	0.688
PGDH -	0.518	-	0.653	0.707	0.716	
DMDH	-	0.513	-	0.612	0.673	0.692
DSQ	-	0.577	-	0.654	0.680	0.694
JMLH	-	0.668	-	0.714	-	0.727
DPAH	-	0.652	-	0.700	0.715	0.714
DWH	-	0.626	-	0.699	0.717	0.724
DPQ	-	0.844	-	0.879	-	0.874
DFH	-	0.590	-	0.697	-	0.747
DPN	-	0.684	-	0.740	-	0.756
Experimental setup: full						
HCLM	-	-	-	0.230	0.290	0.321
DDTL	0.804 ± 0.002	-	-	-	-	-

Table 3.3: Supervised image retrieval results for ImageNet100. With the exception of the DDTL (final row), all of these results were extracted from the literature (Section 2.5). The rows are segmented according to experimental setup for simplified comparison. Columns correspond to different numbers of bits in the hash codes. Error bars are standard deviation over 3 runs.

3.2.3 Hash code quality

Evaluating a deep semantic hash function must be two-fold. First, supervised CBIR performance is important. However, this can be optimised simply using a standard classifier for both single-label and multi-label classification setups (as shown by the performance of the DDTL in the previous section). Second, a good distribution of the resultant hash codes is an important requirement of a hash function: hash functions should minimise collisions in the hash space for high-coverage. The latter is almost always ignored in the literature because **it is difficult to measure whether the coverage of a hash function respects useful semantics in the input images**.

Well-distributed hash codes do not necessarily capture useful semantic information. Nonetheless, in order to maximise efficiency and thereby avoid over-compression (Section 2.3.1), we ideally require a deep semantic hash function to minimise collisions within each class. An ideal semantic hash function produces hash codes that: (1) have no inter-class collisions (i.e., have a perfect *mAP* score), and (2) are evenly distributed over the available hash space. This section measures the degree to which several methods, including the DDTL, meet these requirements. For the following experiments, all

the existing methods were configured with the same backbone network and training in the same way as the DDTL on CIFAR-10. The methods tested in this section are limited to those with available implementations.

Supervised and retrieval metrics

Method	mAP_{5000}	mAP_{50000}	Accuracy CIFAR-10	Accuracy CIFAR-100 (transfer)	
			Nearest N.	Log. reg	Nearest N.
Greedy hash	0.946 ± 0.001	0.954 ± 0.001	94.30 ± 0.15	26.29 ± 1.35	7.67 ± 0.53
DCH	0.951 ± 0.002	0.959 ± 0.001	94.68 ± 0.13	31.98 ± 0.81	5.83 ± 0.27
DHN	0.769 ± 0.035	0.736 ± 0.029	80.13 ± 3.51	7.54 ± 0.29	5.10 ± 0.22
DPN	0.891 ± 0.005	0.872 ± 0.007	91.84 ± 0.27	10.85 ± 0.58	5.17 ± 0.14
DPSH	0.536 ± 0.080	0.531 ± 0.076	55.99 ± 9.04	7.27 ± 1.16	4.42 ± 0.26
DSDH	0.443 ± 0.004	0.378 ± 0.024	49.44 ± 0.21	5.96 ± 0.39	4.33 ± 0.16
DSH	0.704 ± 0.092	0.636 ± 0.103	78.16 ± 7.45	6.77 ± 0.05	4.85 ± 0.33
DTSH	0.927 ± 0.001	0.934 ± 0.001	93.23 ± 0.13	20.39 ± 0.77	5.71 ± 0.29
Hashnet	0.744 ± 0.005	0.746 ± 0.005	75.54 ± 0.42	21.60 ± 0.63	5.08 ± 0.01
IDHN	0.691 ± 0.117	0.656 ± 0.123	75.68 ± 9.42	11.28 ± 1.46	5.25 ± 0.15
DDTL ($\beta = 0.1$)	0.947 ± 0.003	0.953 ± 0.002	94.72 ± 0.28	55.20 ± 0.54	10.64 ± 0.33

Table 3.4: Supervised retrieval metrics on methods for which code was available. Each method was run using the full experimental setup for CIFAR-10 and trained from scratch on the same backbone (ResNet-34) and hyper-parameters as DDTL. The nearest neighbour accuracies were computed using the 16-bit hash codes and hamming distance. The logistic regression-based accuracy on CIFAR-100 measures the transfer performance of the 512-dimensional continuous representation yielded by the ResNet-34 backbone network. Error bars are standard deviation over three seeded runs.

Supervised results are given in Table 3.4, including mAP_{5000} and mAP_{50000}, nearest neighbour accuracy on CIFAR-10, transfer to CIFAR-100 using both logistic regression (on the final continuous representation from the ResNet-34 output) and nearest neighbour assessments. All nearest neighbours were determined using the hamming distance on hash codes. The hash code length was set to 16 bits for each method. While DCH marginally outperformed the DDTL on *mAP* scores (columns 2 and 3), the DDTL produced hash codes that transferred better than any other method (10.64% on CIFAR-100 using nearest neighbour, final column; DCH achieved only 5.83%). This evidences that the decision tree component of the DDTL acts as a mechanism to

retain information that is also useful for transfer. All error bars are standard deviation over the three seeded runs.

Hash performance

Method	Entropy (bits)	$100 * \frac{\sum \text{unique hash codes}}{\text{num training data}}$ (%)	Average collision count
Greedy hash	4.06 ± 0.05	1.74 ± 0.10	4298 ± 40
DCH	3.39 ± 0.00	0.37 ± 0.02	4942 ± 3
DHN	4.13 ± 0.15	0.63 ± 0.03	4632 ± 543
DPN	3.55 ± 0.00	0.48 ± 0.07	4778 ± 2
DPSH	3.35 ± 0.76	0.47 ± 0.12	9525 ± 5623
DSDH	4.58 ± 0.39	0.61 ± 0.21	5756 ± 1578
DSH	5.11 ± 0.90	2.56 ± 0.80	3388 ± 756
DTSH	3.46 ± 0.02	0.15 ± 0.01	4848 ± 22
Hashnet	3.22 ± 0.04	0.58 ± 0.08	6594 ± 65
IDHN	5.43 ± 0.71	1.82 ± 0.33	2788 ± 701
DDTL ($\beta = 0.1$)	14.37 ± 0.00	51.29 ± 0.15	3 ± 0

Table 3.5: Hash codes quality for 16 bit hashes trained on CIFAR-10. Measurements were taken using the training set. The entropy is measured by computing the number of training images mapped to each available bin in the hash space. Column 3 measures the ratio of unique hash codes to training data. Error bars are standard deviation over three seeded runs.

Table 3.5 lists several hash quality metrics measured on the hash codes from the tested methods. It is clear that DDTL distributes hash codes over the available hash space far better than any other method tested here. For instance, consider column 3 that gives the ratio of unique hash codes to training images. DDTL yielded on average of over 25000 unique hash codes for the CIFAR-10 training data (of which there are 50000 images). The closest competitor in this case was DSH with an average of 1280 unique hash codes. However, DSH only achieved a $mAP_{5000} = 0.794 \pm 0.092$ and a $mAP_{50000} = 0.636 \pm 0.103$, meaning that the better distribution of DSH (compared to other existing methods) came at the cost of poor supervised performance. Consider the collision rates in column 4, where the DDTL has an average collision rate of 3. This is 929 $\times$ better than IDHN, the closest competitor. Again, IDHN also evidenced poor performance on the retrieval task. The hash codes computed by the DDTL enable distinction between training images at a far finer scale than any other method tested here, but that also enable good performance on the supervised retrieval task.

Why does DDTL produce efficient binary representations? The contrastive loss used to train the DDTL is known to result in an approximately uniform projection of the training images by the neural network onto the unit hypersphere [Wang and Isola, 2020]. Since the input representations to the contrastive loss are the leaf probabilities after processing images through the DDTL, the distribution of images over leaf nodes in the tree will be approximately uniform after training. This approach is unsupervised and therefore does not suffer from the high-compression owing to training using class labels alone.

Evenly distributed hash codes are not necessarily *usefully* distributed. While the evidence in Table 3.5 sets DDTL apart from other methods, achieving well-distributed hash codes can be trivial (see Section 3.3.1). Do the evenly distributed hash codes from the DDTL capture any useful semantic meaning beyond class labels? This is a challenging question that is closely related to assessing unsupervised clustering – there is no ground truth by which to measure the relevance of any semantic information in the hash codes. Grimmer and King [2011] discussed the difficulty of knowing apriori which partition of a data is 'the best' partition. Choosing a good clustering is often subjective and user or use-case specific when there are no known ground-truth labels. I aim to address this concern regarding the DDTL hash codes in the following sections. Section 3.2.4 gives retrievals for a qualitative analysis. Section 3.2.5 details the results of an experiment where the DDTL was trained on a coarse labelling of CIFAR-100 (called CIFAR-100-20, collected into 20 'super-classes') and tested on the original fine labelling of CIFAR-100. A number of ablation studies are discussed in Section 3.3 to further evidence the utility of the DDTL.

3.2.4 Retrievals

Figures 3.4 and 3.5 are retrieval results on CIFAR-10 for all classes and for a single class, respectively. Image attributes such as shape, background, texture, and orientation seem to be distinguishing features of the retrieved images, although this assessment is subjective. As an example, the horses in Figure 3.5 can be distinguished according to the presence or absence of a rider, background characteristics (e.g., grass), or portion of horse in the image (e.g., upper or full body). The red highlight in Figure 3.4 shows an incorrect retrieval (a truck was predicted as a cat) and the yellow highlight shows an interesting finding: the test and train sets of CIFAR-10 contain near-identical copies.

Figure 3.6 gives retrievals on Imagenet100, showing that the DDTL can scale to

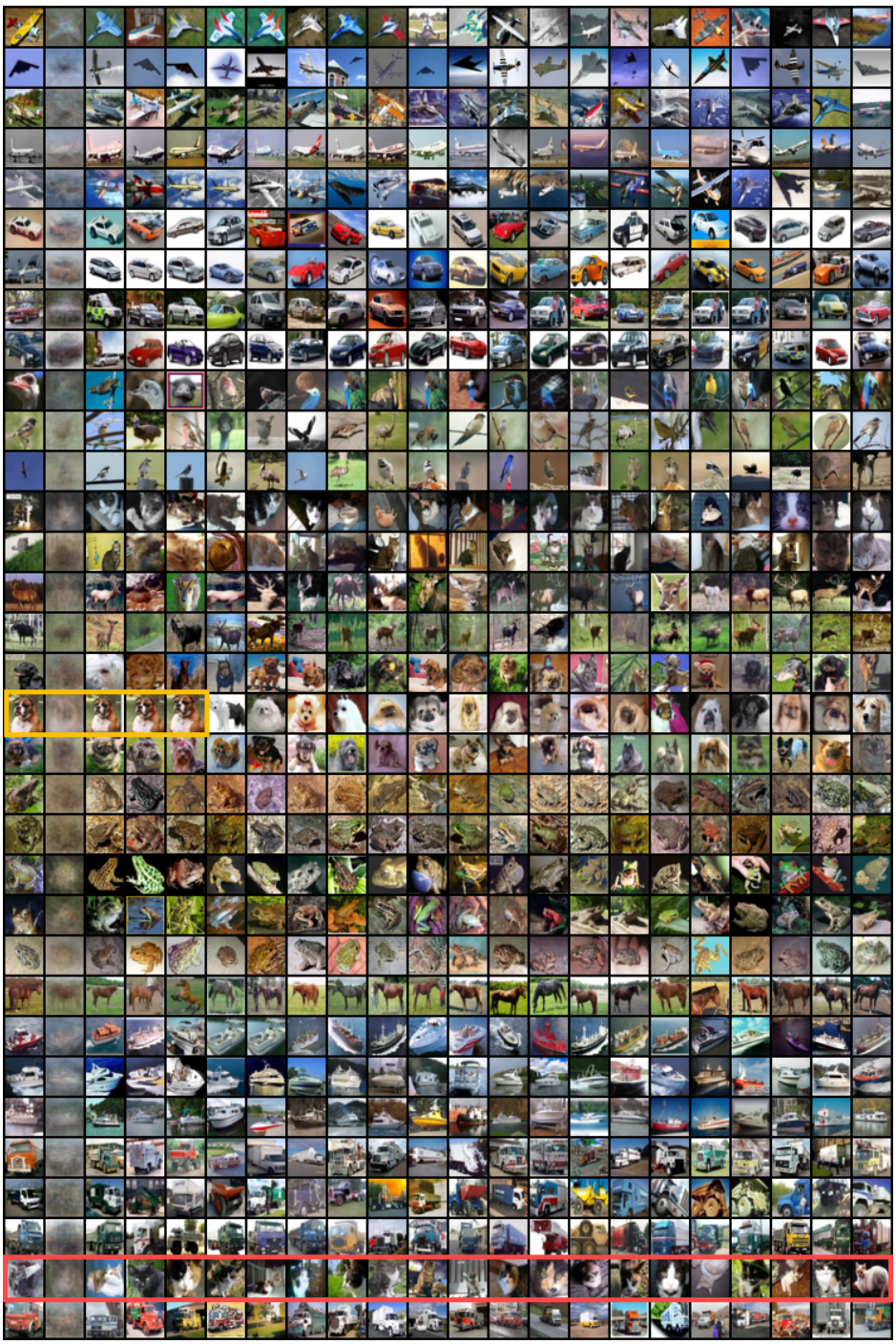


Figure 3.4: Image retrievals on CIFAR-10. The query images are in the leftmost column, followed by the average of retrieved images. The retrieved images are ordered from left to right according to the lexicographical ordering that respects the tree hierarchy (see Section 3.1).

high-resolution large datasets and still enable a fine granularity for retrieval. Multiple retrievals per class are intentionally shown to demonstrate the difference between queries from the same class. For example, the final two rows contain the same breed of dog, where a distinguishing factor in the retrievals is the number of dogs (multiple in the first case and singular in the second case).

3.2.5 Coarse to fine-grained label transfer

For this experiment the DDTL was trained on CIFAR-100-20: a coarse-grained grouping of CIFAR-100 into 20 super-classes. The hypothesis was that the decision tree would split images according to semantic content that corresponds to the fine-grained CIFAR-100 classes. This experiment was designed to assess whether the decision path from the DDTL has value when measured using a known fine-grained labelling. Accuracy was measured on the CIFAR-100 test set when using the decision tree directly as a classifier (by counting the number of CIFAR-100 training images in each class for each leaf nodes of the tree to assign class predictions to leaves). The test accuracy was 42.13 ± 0.39 (averaged over three seeded runs). This means that over 42% of the leaves modelled by the decision tree correspond well with the ground truth CIFAR-100 labels.

3.3 Ablations

Several ablation studies are detailed in this section. A comparison baseline that uses a classifier and repeated principle component analysis is tested in Section 3.3.1. The robustness of decision paths computed by the DDTL is tested in Section 3.3.2. The impact of fitting the decision tree after representation learning is explored in Section 3.3.3.

3.3.1 A comparison baseline

A simple baseline can be built that also uses a multi-stage hash construction (see Section 3.1). To do so, a standard classifier can be trained to meet the supervised requirement for good CBIR and an unsupervised method can be used to mimic the tree structure of the DDTL. To these ends, a ResNet-34 was trained using the same setup as the DDTL using CIFAR-10. Training included optimising a contrastive loss applied to the continuous representation from the ResNet-34 backbone (prior to a linear projection to class predictions). The supervised and contrastive losses were balanced with

Figure 3.5: Extended image retrievals on CIFAR-10. These are ordered in the same fashion as Figure 3.4 but show more retrievals for these two specific classes (boat and horse) in order to show the high distinction enabled by the DDTL hash codes.

Figure 3.6: Image retrievals on Imagenet100. The query images are in the leftmost column, followed by the average of retrieved images. The retrieved images are ordered from left to right according to the lexicographical ordering that respects the tree hierarchy (see Section 3.1).

the same α and β as was used to train the DDTL. To ensure a well-distributed hash code that is comparable to the DDTL, repeated principle component analysis (rPCA) was applied to the same continuous representation. rPCA operates as follows:

1. Apply PCA to the continuous representations from the ResNet-34 for all the training images.
2. Split the training images into two equally sized subsets along the first principle component (analogous to left versus right decisions for the DDTL decision path).
3. For each split apply PCA to the same continuous representations but only on the training images from this split.
4. Repeat splitting until a desired depth is reached.

Each split of rPCA is analogous to a single decision in the decision path of the DDTL. This baseline is called CrPCA (Classifier with repeated PCA).

Table 3.6 lists the results for CrPCA and the DDTL. It is evident that the repeated PCA strategy was unable to produce a hash as well distributed as DDTL, yet better than the methods listed in Table 3.5. This is because the DDTL is trained such that the decision tree spreads each class over the unit hypersphere [Wang and Isola, 2020], while the singular values of CrPCA are computed over the entire training dataset. This relative poor performance of the CrPCA baseline is likely due to the dominance of class-relevant directions in the representation to which PCA was applied.

Hash performance

Method	Entropy (bits)	$100 * \frac{\sum \text{unique hash codes}}{\text{num training data}}$ (%)	Average collision count
CrPCA	11.67 ± 0.54	8.53 ± 0.54	23 ± 16
DDTL ($\beta = 0.1$)	14.37 ± 0.00	51.29 ± 0.15	3 ± 0

Table 3.6: Hash codes quality for 16 bit hashes, comparing the CrPCA baseline to DDTL. Measurements were taken using the training set. Error bars are standard deviation over three seeded runs.

The transfer performance of CrPCA was also evaluated. Two accuracies were computed (for comparison to the final two columns in Table 3.4): (1) test accuracy using logistic regression on the continuous representation output of the ResNet-34, and (2)

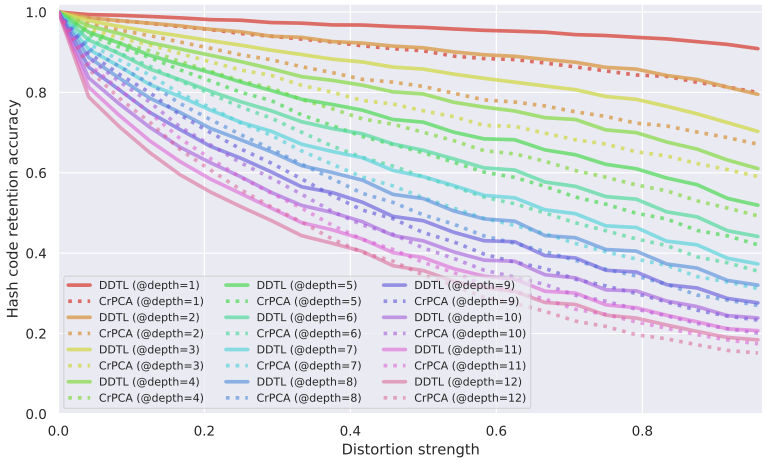
nearest neighbour accuracy using the hash codes and hamming distance. The test accuracies were $4.77 \pm 0.06\%$ and $50.84 \pm 0.55\%$, respectively: $\sim 5.9\%$ and $\sim 4.4\%$ worse than the DDTL. The following section tests the robustness of both the DDTL and CrPCA approaches to realistic changes in the input images.

3.3.2 Robustness to changes in the input images

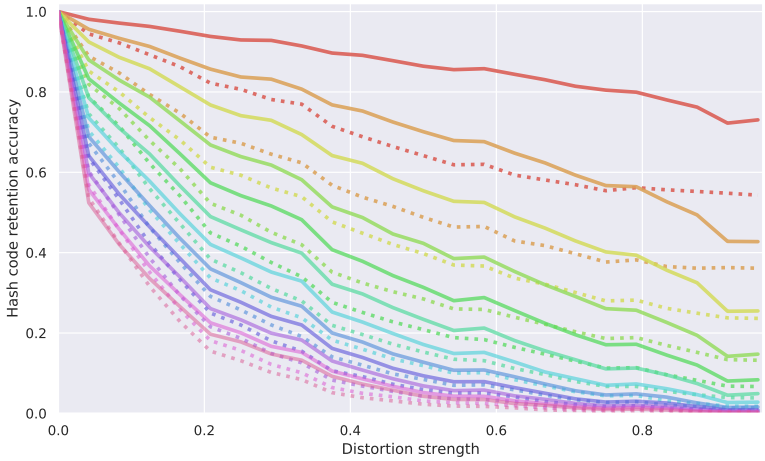
If the hash codes computed by a deep semantic hash function were found to be robust to realistic changes in input images, these hash codes would be related to high-level semantics that are not altered by said changes. To test this sort of robustness for the DDTL and the CrPCA baseline, unaltered test images were first processed to determine ground-truth hash codes for each method. The test images were then altered at various degrees of colour distortion or occlusion. The hash codes were recomputed for each level of distortion. Since each of these methods computes hash functions using tree structures, it is expected that the bits corresponding to the deepest decisions will be relatively more susceptible to change than bits corresponding to shallower decisions, because errors in the decisions will propagate from root to leaf.

Figure 3.7 shows the 'hash code retention accuracy' for both the DDTL and CrPCA under increasing degrees of image distortion strength, for (a) colour and (b) occlusion distortions. This metric measures the percentage of images that map to the same hash codes after distortion. Colour distortions use the built-in colour jitter functionality of the PyTorch framework [Paszke et al., 2019] that alters the brightness, contrast, and hue of images. Occlusion refers to the removal (by zero masking) of a portion of an image. Depth in this figure refers to the level of truncation applied to the hash codes: at a tree depth of 5 a hash code is truncated to length 5, keeping the leftmost bits. This is done to measure robustness at all levels of the trees. As expected, the bits corresponding to deeper decisions in the trees were less robust to changes in the input images.

While CrPCA was relatively more robust to lower levels of colour distortion at deeper tree levels (some of the dashed lines are above the solid lines in Figure 3.7 (a) at lower distortion strengths), DDTL was more robust to higher colour distortion levels and almost always more robust to occlusion.



(a) Colour distortion



(b) Occlusion

Figure 3.7: Retention of hash codes under increasing distortion strengths, measured as accuracy with respect to the hash codes under no distortion. Robustness to colour changes (random changes in brightness, contrast, saturation, and hue) is shown in (a) and robustness to occlusions (i.e., zero-masking regions of increasing size) is shown in (b). The solid lines are computed using DDTL and the dashed lines are computed using the baseline described in Section 3.3.1. 'Depth' refers to at which index the hash codes are truncated (e.g., a hash code of length 5 is truncated to length 5).

3.3.3 Fitting the decision tree after learning

For this experiment the backbone network was trained without the decision tree component but with an additional unsupervised contrastive loss applied to the continuous representation output by the backbone. It is expected that this would generate a map-

ping to a representation with similar properties to the corresponding representation in the backbone of the DDTL. To evaluate this hypothesis, the decision tree component of the DDTL was trained alongside a standard contrastive learning setup (to learn the mapping to a continuous representation prior to linear projection for the DDTL) but without propagating gradients after the decision tree parameters.

Fitting the decision tree separately resulted in similar retrieval performance to the DDTL setup, with a mAP_{5000} of 0.945 ± 0.002 and a mAP_{50000} of 0.951 ± 0.002. Regarding the distribution of the resultant hash codes, this setup yielded an entropy of 14.39 ± 0.01, a unique hash codes to training images ratio of 51.45 ± 0.07, and an

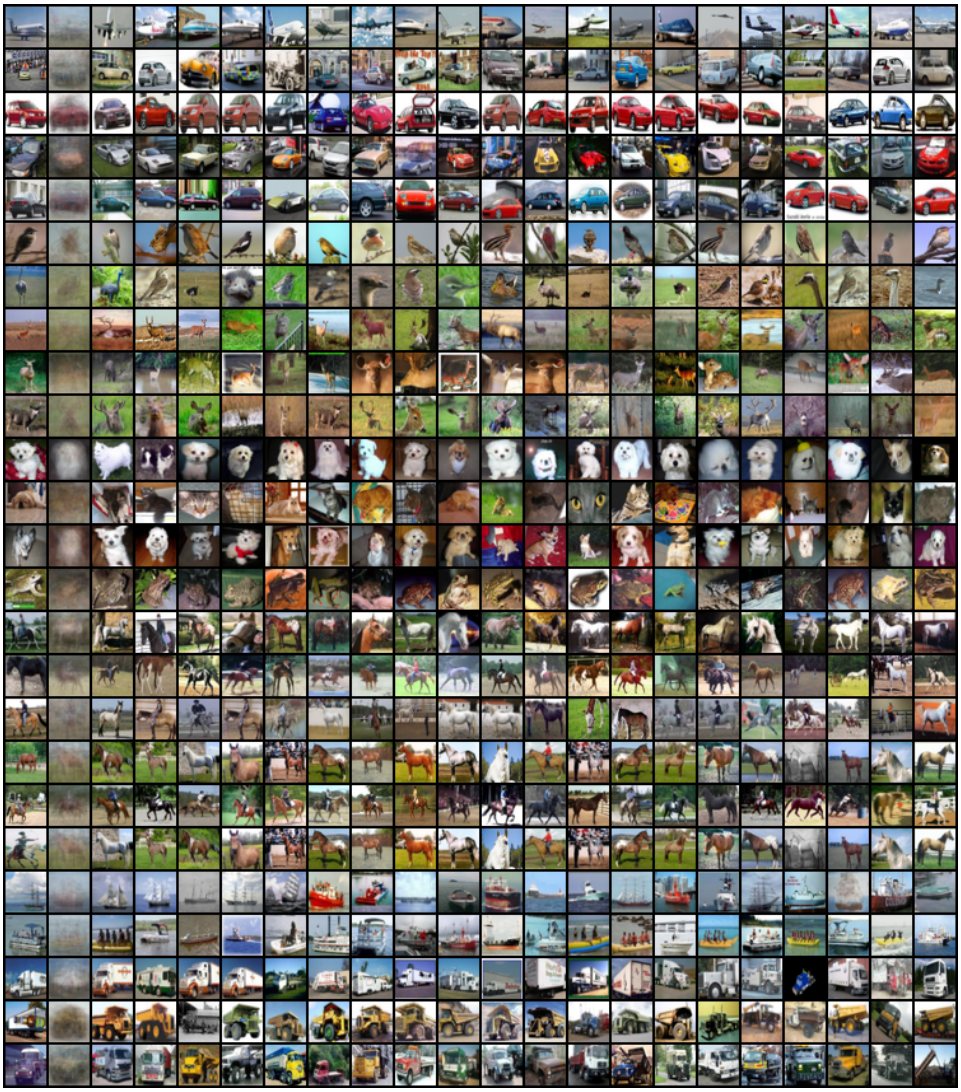


Figure 3.8: Retrievals when training the decision tree alongside a representation learning setup.

average collision count of 3.

Figure 3.8 shows CIFAR-10 retrievals when fitting the decision tree alongside representation learning. The DDTL can be applied alongside representation learning without the need to learn the decision tree parameters in an end-to-end fashion, provided a contrastive loss is also used.

3.3.4 Unsupervised setup

For the fully unsupervised setting the DDTL was trained with $\alpha = 0$ and $\beta = 1$. This equates to training using only a contrastive loss on the leaf probabilities of the decision tree. In this setup a stronger data augmentation scheme was used to match the scheme used by SimCLR [Chen et al., 2020a]. Logistic regression was applied to the continuous representation output by the ResNet backbone after learning, as is standard practise in contrastive learning. A test accuracy of 85.04% on CIFAR-10 was achieved.

3.3.5 Supervised only setup

In this setting the DDTL was trained with $\alpha = 1$ and $\beta = 0$ (i.e., no contrastive loss), yielding a $mAP_{5000} = 0.955 \pm 0.151$. It is expected that the supervised retrieval performance is high here, owing to the relationship between retrieval and classification. However, a collision rate of 5000 for CIFAR-10 (with 50000 images in the training dataset) means that training without the contrastive loss hinders utility of the resultant representation. Further, transfer to CIFAR-100 was worse, with a nearest neighbour accuracy of 4.77 ± 0.064 ($\pm$6% worse than the setting in Table 3.4), and a logistic regression accuracy of 50.84 ± 0.55 ($\pm$5% worse than the setting in Table 3.4).

3.4 Demonstration of decision tree utility

Figure 3.9 demonstrates how the decision tree can be used to understand the mistakes made by the classification network, enabling insight into what might have gone wrong. Further research needs to be done to fully leverage the decision tree as this assessment is purely subjective. Of particular interest is the frog example that looks remarkably like a truck (note the topmost image to the left of the frog, for example). This shows how fine-grained comparison between images can help users understand how a neural network might fail.

Figure 3.9: Interpretation by way of tree neighbour comparison. Each query image (centre) was incorrectly classified by the DDTL. The left and right retrievals are computed by using the predicted classes and true classes, respectively. This fine-grained analysis gives insight into similarities between instances in different classes: the aircraft in the first row is evidently mistaken for antlers, the truck in the second row has a similar shape and colour set to frogs, the third row shows how images from birds on the ground and deer are similar in quality, and the fourth row evidences how the classification detects a 'sail' shape instead of a bird.

Figures 3.10 and 3.11 demonstrates how the decision tree can be used for dataset exploration, for all classes and a single class, respectively. For example, it is immediately evident via this visualisation that there are many training images in CIFAR-10 that are nearly identical. This view on the data also helps to contextualise what features in the images the decision tree component of the DDTL is leveraging to minimise the contrastive loss.

3.5 Conclusion

In this chapter I presented the deep decision tree layer (DDTL) for learning efficient and robust discrete representations of images for use in the content-based image retrieval task. I identified a pervasive issue of over-compression when learning deep semantic hash functions owing to (1) high-compression supervised objectives and (2) discretisation operations that also compress information away. To address this issue I proposed building hash codes in two parts. First, a class prediction provided the minimal hash codes for high-performance supervised image retrieval. Second, a binary decision tree was learned that models further intra-class differences to enable distinction between images from the same class. Concatenation of the output of these two components yields ordered and evenly distributed hash codes for images. Neither component required discretisation during learning, meaning that compression because of discretisation was effectively mitigated against.

The DDTL achieved state-of-the-art results on supervised hash-based image retrieval metrics on three widely used datasets, with a mAP_{5000} of 0.953 ± 0.002 for CIFAR-10, a mAP_{5000} of 0.897 ± 0.002 for NUS-wide-21, and a mAP_{1000} of 0.804 ± 0.002 for ImageNet100. Experiments with consistent neural network backbones and training hyper-parameters were undertaken to assess how well the DDTL compared to a number of existing methods with regard to learning a 'good quality' semantic hash function. Good quality here means that a hash function produces hash codes that enable high performance on supervised retrieval metrics while also being evenly distributed over the available hash space. The DDTL was shown to: (1) be amongst the top performing methods regarding the supervised retrieval task; (2) transfer better to unseen data (training on CIFAR-10 with transfer to CIFAR-100); and (3) yield far more evenly distributed hash codes with minimal collisions.

Further experiments showed that the DDTL produced hash codes that were robust to realistic changes in the input images (e.g., colour/lighting changes) when compared to a strong baseline that used a supervised classifier paired with repeated principle component analysis for comparable hash code construction. The DDTL was also trained on the CIFAR-100-20 super-classes to test whether the decision tree was informative of the fine-grained CIFAR-100 classes: over 42% of the time the leaves in the tree corresponded well with ground-truth test set labels. Examples of (1) interpretability by way of the decision tree output and (2) dataset exploration were also given.

Future work will entail making better use of the tree structure to build model in-

terpretation approaches, and making use of ensembles of independent decision trees to improve both supervised and unsupervised image retrieval.

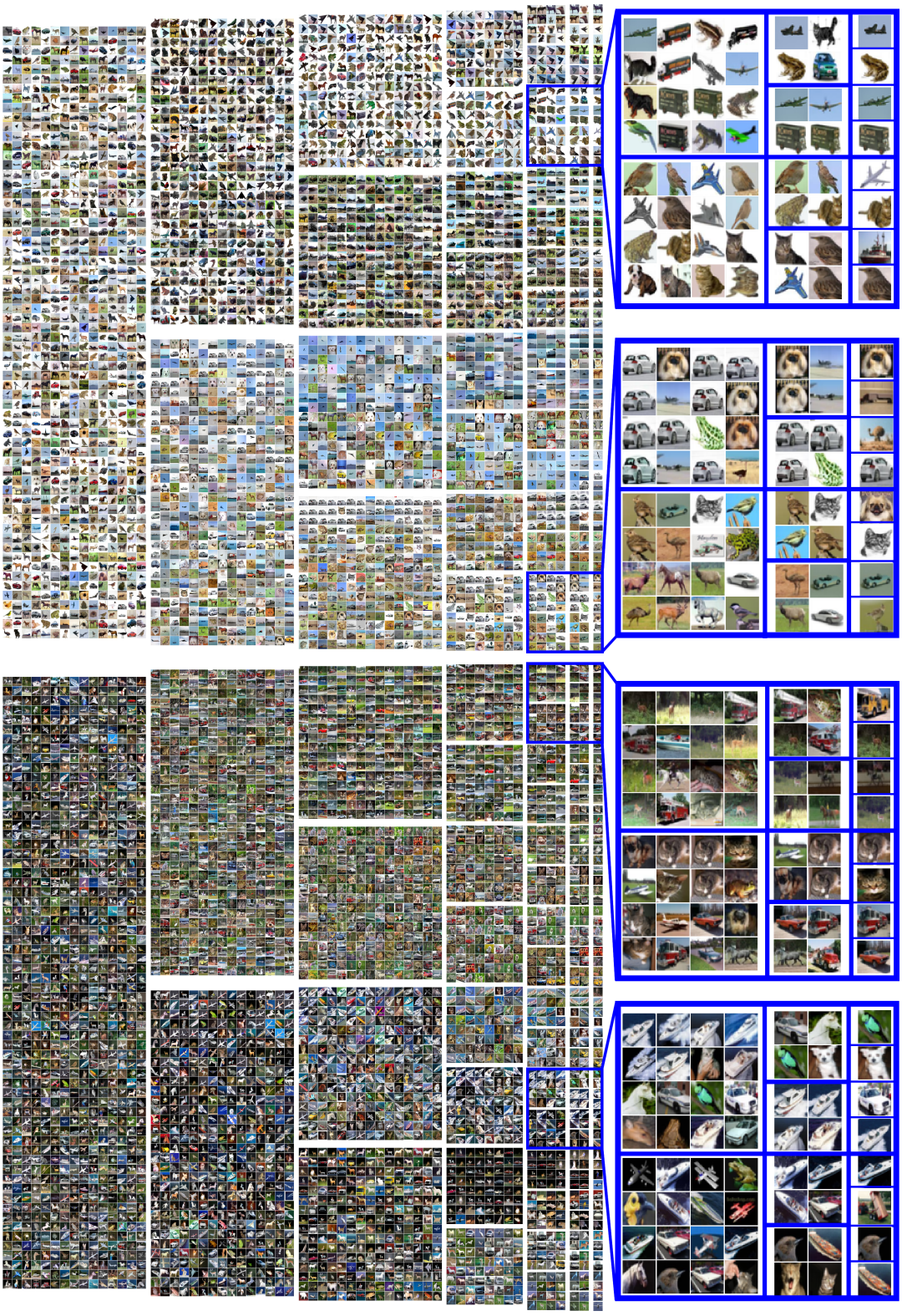


Figure 3.10: Decision tree visualisation for dataset exploration on all classes from CIFAR-10. Only 7 layers of the 12-bit tree could be shown owing to space constraints. Different features determine clusters at different parts of the tree. For example, object angle and background colour are distinguishing features in the lower right area. Latter regions are zoomed for legibility. The blue lines within the zoomed section separate images within unique nodes.

Figure 3.11: Decision tree visualisation for dataset exploration on the vehicle class from CIFAR-10. Only 7 layers of the 12-bit tree could be shown owing to space constraints. This type of exploration enables fast understanding of an image dataset. For example, note the near identical repetition of many vehicle types – pixel difference would not be informative in this case because the images are slightly different but nonetheless seem to share the same source. Latter regions are zoomed for legibility. The blue lines within the zoomed section separate images within unique nodes.

Chapter 4

Deep hierarchical object grouping

4.1 Overview

In this chapter I propose Deep Hierarchical Object Grouping (DHOG), a method to improve the optima to which a deep clustering objective converges. The deep clustering objective (see Section 2.2.3) is chosen as an example of a modern approach to mutual information (MI) maximisation between data augmentations (see Section 2.2.1). This chapter focuses on a particular mode of proxy objective failure in the field of deep clustering: where greedy SGD optimisation results in neural network parameters that correspond to demonstrably sub-optimal local minima of the clustering objective – see Section 2.3.2. Neural networks with these sub-optimal parameters typically produce worse performance on the downstream task than those with more optimal parameters. The downstream task in this case is matching cluster allocations with ground truth labels of the images.

DHOG is designed to account for, and circumvent, the tendency of neural networks to get stuck in sub-optimal local optima of the clustering objective. Learning for DHOG involves simultaneously optimising for many *diverse* local optima of the underlying objective. Each solution to the objective is captured by a compute 'head' that yields a probability over cluster assignments. Since each of these heads define probability distributions over clusters we can minimise the pairwise MI between all heads such that they maximise coverage of the solution space in a way that improves performance on the downstream task.

DHOG is a multi-head approach that is designed specifically to model and account for poor local minima of the clustering objective. I demonstrate that requiring DHOG to account for multiple local optima of the clustering objective leads to some compute heads that better optimise this objective, thus mitigating against proxy objective failure

(see Section 2.3.2). DHOG yields accuracy improvements on the downstream task when trained and evaluated on the CIFAR-10, CIFAR-100-20, and SVHN datasets.

4.2 Introduction

It is very expensive to label a dataset with respect to a particular task. Consider the alternative where a user, instead of labelling a dataset, specifies a simple set of class-preserving transformations or 'augmentations'. For example, lighting changes will not change a dog into a cat. Is it possible to learn a model that produces a useful representation by leveraging a set of such augmentations? This representation would need to be good at capturing salient information about the data, and enable downstream tasks to be done efficiently. If the representation were a discrete labelling which groups the dataset into clusters, an obvious choice of downstream task is unsupervised clustering. Ideally the clusters should match direct labelling, without ever having been learnt on explicitly labelled data.

Using data augmentations to drive unsupervised representation learning for images has been explored by a number of authors [Dosovitskiy et al., 2015, Bachman et al., 2019, Chang et al., 2017, Wu et al., 2019, Ji et al., 2019, Cubuk et al., 2020]. These approaches typically involve learning neural networks that map augmentations of the same image to similar representations. For more information, literature related to MI maximisation for representation learning is detailed in Section 2.2.1, relevant contrastive learning approaches are discussed in Section 2.2.2, and deep clustering approaches are discussed in Section 2.2.3.

A number of earlier works target maximising mutual information (MI) between augmentations [van den Oord et al., 2018, Hjelm et al., 2019, Wu et al., 2019, Ji et al., 2019, Bachman et al., 2019]. Targeting high MI between representations computed from distinct augmentations enables learning representations that perform well on challenging downstream tasks (e.g., object detection or classification). We are interested in a particularly parsimonious representation: a discrete labelling of the data. This labelling can be seen as a clustering [Ji et al., 2019] procedure, where MI can be computed and assessment can be done directly using the learned labelling, as opposed to via an auxiliary model trained posthoc.

One widespread and well-known issue with methods that maximise MI between augmentations is that these methods are strongly dependent on the augmentation strategy chosen. While it is fairly straightforward to define what is considered 'strong'

augmentations for images that demonstrably improve performance [Chen et al., 2020b, Tian et al., 2020, Wang and Qi, 2021], this is not true for all data types or domains. I propose in this Thesis that the underlying issue – neural networks prefer easy-to-compute solutions – can be alleviated or circumvented using techniques such as DHOG, as opposed to defining stricter and stronger data augmentations.

4.2.1 Sub-optimal mutual information maximisation

I show that the MI objective is not maximised effectively in existing work due to:

1. **Greedy optimisation algorithms** used to train neural networks, such as stochastic gradient descent (SGD), which potentially target local optima; and
2. The use of a limited set of data augmentations that can result in the existence of multiple **local optima to the MI maximisation** objective.

SGD is greedy in the sense that early-found high-gradient features can dominate and so networks will tend to learn easier-to-compute locally-optimal representations (for example, one that can be computed using fewer neural network layers) over those that depend on complex features. I demonstrate this phenomenon in the next chapter using a toy problem (see Section 5.3.2). Neural network solutions are difficult to assess and characterise absolutely because of the non-linear computation and their large size. Instead we can compare two locally-optimal representations by assessing (1) how well they optimise for the training objective and (2) how well they perform on the downstream assessment task(s).

By way of example, in natural images, average colour is easy-to-compute, whereas object type is not. If an augmentation policy tends to preserves average colour, then a reasonable neural network mapping need only compute colour to obtain high MI between image representations from different augmentations. A mapping like this is sub-optimal in the sense that *hypothetical higher MI optima exists* that also captures semantic information, assuming the model has sufficient capacity to learn and represent this. The conceivable existence of many such local optima coupled with greedy optimisation presents the challenge I tackle in this chapter: *how can we leverage powerful image augmentation-driven MI objectives while avoiding greedily-found local optima*?

Dealing with greedy solutions Heuristic approaches to dealing with easy, greedily found solutions are viable and widely implemented. One such example is to apply

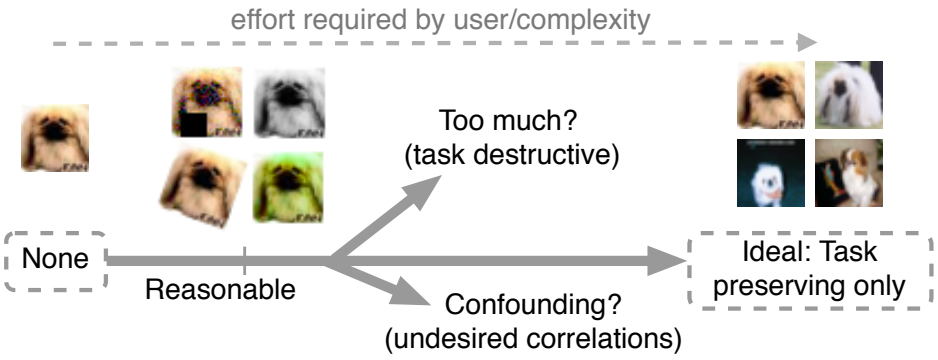


Figure 4.1: A spectrum of augmentations. Doing no augmentations (left) requires zero effort by the user. Ideal augmentations (right) preserve only task relevant information (object class, here), and effectively requires full labelling.

Sobel edge-detection [Caron et al., 2018, Ji et al., 2019] to images as a pre-processing step. This has been suggested to remove/alter the features in images that may cause trivial representations to be learned. However, taking this approach is a symptomatic treatment and not a solution. In the work presented in this chapter, I acknowledge that greedy SGD can cause neural networks to get stuck in local optima of the MI maximisation objective because of the limits of data augmentations (see Figure 4.1). Instead of trying to prevent a greedy solution, DHOG enables a model to learn a representation that relies on easy-to-compute features using a deep clustering objective, but *also requires this model to learn an additional representation* such that there is low MI between them. We can then extend this principle by adding representations, each time requiring the latest representations to be distinct from all previous representations.

Greedy optimisation finds sub-optimal minima The issue of sub-optimal augmentations is exacerbated by how we train neural networks. Greedy optimisation algorithms, such as SGD, will tend to prefer easier-to-compute valid solutions to the objective over those that depend on complex features in the data (see Section 2.3.2). These solutions are stable local optima because of the insufficiency of the predefined data augmentations. It should be noted that this is not a problem with the methods defined using augmentations but rather a shortcoming of augmentations as a learning driver: the better the augmentation strategy, the more expensive they are to produce (see Figure 4.1; the ideal augmentation for matching clusters with ground-truth classes requires the class labels themselves). SGD always runs the risk of getting stuck in these sub-optimal local optima.

Downstream task: clustering The focus of this chapter is on learning representations that result in higher MI between representations from differently augmented images. Assessment is on the downstream task of matching predicted clusters to ground-truth classes. There are two important advantages to clustering methods, namely: (1) they typically yield, or can be made to yield, a distribution of clusters for a given image, meaning that they are amenable to information theoretic analysis and methods; and (2) they offer a direct comparison because they require *no labels for learning a mapping from the learned representation to class labels*. Training labels are only required to assign specific clusters to appropriate classes and no learning is done using these.

4.2.2 Contributions

Learning a set of representations by encouraging them to have low MI, while still maximising the original augmentation-driven MI objective for each representation, is the core idea behind deep hierarchical object grouping (DHOG). DHOG defines a mechanism to produce a set of hierarchically-ordered solutions (in the sense of easy-to-hard orderings, not tree structures). DHOG is able to better maximise the original MI objective between augmentations since each representation must correspond to a unique local optima. The contributions of this chapter are as follows.

1. A demonstration that current methods do not effectively maximise the MI objective[1] because greedy SGD typically results in sub-optimal local optima.

2. To mitigate against this problem I propose DHOG as a robust neural network image grouping method to learn diverse and hierarchically arranged *sets of discrete image labellings* (Section 4.3). DHOG explicitly models, accounts for, and avoids spurious local optima, requiring only simple data augmentations, and needing no Sobel edge detection.

3. I show a marked improvement over the baseline method used for clustering (invariant information clustering [Ji et al., 2019]) which was the state-of-the-art when creating DHOG. Assessments are carried out using standard benchmarks in end-to-end image clustering for CIFAR-10, CIFAR-100-20 (a 20-way class grouping of CIFAR-100), SVHN, and STL-10. A new accuracy benchmark is also set on CINIC-10.

[1] I find and measure higher MI solutions using DHOG, and hence show that such better solutions exist.

To be clear, DHOG still learns to map data augmentations to similar representations as this is imperative to the learning process. The difference is that DHOG enables a number of intentionally distinct discrete data labellings to be learned, arranged hierarchically in terms of source feature complexity.

4.3 Method

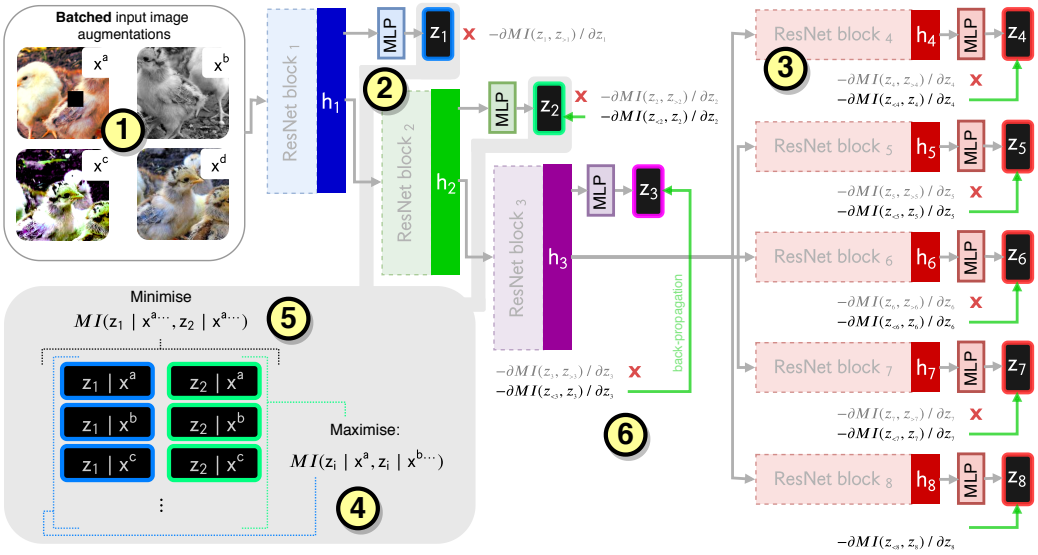


Figure 4.2: DHOG architecture. The skeleton is a ResNet18 [He et al., 2016b]. ① Augmentations of each image, $x^{a\ldots d}$, are separately processed by the network. ② Each shallow ResNet block $(1\ldots3)$ constitutes shared computation for deeper blocks, while also computing separate probability vectors, $z_1 \ldots z_3$. Each z_i is viewed as the probability for each outcome of the random variable c_i that makes a discrete labelling choice. The notation $\mathrm{MI}(z, z')$ between probability vectors is shorthand for $\mathrm{MI}(c, c')$ between random variables. ③ The final ResNet block is repeated $k-3$ times ($k=8$ here) to compute further $z_{>3}$. ④ The network is trained by maximising the MI between allocations c_i from *all data augmentations*, and ⑤ separately for each node i, minimising the MI between c_i and $c_{<i}$ for the *same data augmentation*. ⑥ This is implemented by stopping gradients such that they are *not back-propagated* for later computation paths (red crosses).

Figure 4.2 shows the DHOG architecture. DHOG is an approach for obtaining jointly trained multi-level representations as discrete labellings, arranged in a simple-

to-complex hierarchy, and computed by separate 'heads'. A head is an unit that computes a multivariate class probability vector. By requiring low MI between heads, a diversity of solutions to the MI maximisation objective can be found. The head that best maximises MI between augmentations typically aligns better with a ground truth task that also relies on complex features that augmentations are designed to preserve.

Figure 4.2 demonstrates the DHOG architecture and training principles. There are shared model weights (②: ResNet blocks 1, 2, and 3) and head-specific weights (the MLP layers and ③: ResNet blocks 4 to 8). For the sake of brevity, we abuse notation and use $\mathrm{MI}(\mathsf{z},\mathsf{z}')$ between labelling probability vectors as an overloaded shorthand for the mutual information $\mathrm{MI}(c,c')$ between the labelling random variables c and c' that have probability vectors z and z' respectively.

Any branch of the DHOG architecture (① to any z_i) can be regarded as a single neural network. These are trained to maximise the MI between the label variables at each head for different augmentations; i.e., between label variables with probability vectors $\mathsf{z}_i(\mathsf{x})$ and $\mathsf{z}_i(\mathsf{x}')$ for augmentations x and x'. Four augmentations are shown at ①. The MI is maximised pairwise between all pairs, at ④. This process can be considered *pulling* the mapped representations together.

Following IIC [Ji et al., 2019], we compute the MI directly from the label probability vectors within a minibatch. Let $\mathbf{z}_i, \mathbf{z}'_i$ denote the random probability vectors at head i associated with sampling a data item and its augmentations, and passing those through the network. Then we can compute the mutual MI between labels associated with each augmentation using

$$\begin{aligned}\mathrm{MI}_{aug}(c_i,c'_i) = \mathrm{Tr}(E[\mathbf{z}_i(\mathbf{z}'_i)^T]^T \log(E[\mathbf{z}_i(\mathbf{z}'_i)^T])) \\ - E[\mathbf{z}_i^T]\log E[\mathbf{z}_i] - E[(\mathbf{z}'_i)^T]\log E[\mathbf{z}'_i],\end{aligned} \tag{4.1}$$

where Tr is the matrix trace, logarithms are computed element-wise, and expectations are over data samples and augmentations of each sample. In practice we compute an empirical estimate of this MI based on samples from a minibatch.

4.3.1 Distinct heads

Each clustering head in DHOG is encouraged to compute unique solutions via *cross-head MI minimisation*. For a minibatch of images, the labelling from any head is optimised to have low MI with other heads' labellings. We assume multiple viable labellings because of natural patterns in the data. By encouraging low MI between heads, these must capture different patterns in the data.

Simple concepts (brightness, colour, etc.) are axes of variation that are reasonable and easy to group by. Groupings according to complex features (e.g., object type) typically require more processing and greedy optimisation may not discover these groupings without explicit encouragement. Unfortunately, the easier-to-compute groupings typically correspond poorly to downstream tasks. Without a mechanism to explore viable patterns in the data, greedy optimisation will avoid finding them.

Cross-head MI minimisation DHOG addresses *sub-optimal MI maximisation* (see Section 2.3.2) by encouraging unique solutions at sequential heads ($z_1 \dots z_8$ in Figure 4.2), which rely on different features. Let $\mathbf{z}_i, \mathbf{z}_j$ denote the random probability vectors from two heads. We can *minimise* the MI across heads:

$$\begin{aligned} \mathrm{MI}_{head}(c_i, c_j) = \mathrm{Tr}(E[\mathbf{z}_i \mathbf{z}_j^T]^T \log(E[\mathbf{z}_i \mathbf{z}_j^T])) \\ - E[\mathbf{z}_i^T] \log(E[\mathbf{z}_i]) - E[(\mathbf{z}_j)^T] \log(E[\mathbf{z}_j]). \end{aligned} \tag{4.2}$$

Logarithms are element-wise, and expectations are over the data and augmentations. Note $\mathbf{z}_i$ and $\mathbf{z}_j$ are each computed from the *same data augmentation*. We estimate this from each minibatch sample. This process can be thought of as *pushing* the heads apart. We note that the Tr operator is commutative – the hierarchical arrangement is accomplished through gradient stopping.

Hierarchical arrangement Requiring k heads (where $k = 8$ here) to produce unique representations is not necessarily the optimal method to account for sub-optimal MI maximisation. Instead, a simple-to-complex hierarchy structure to the heads is encouraged, defined according to cross-head comparisons made using Equation 4.2. The hierarchy enables a reference mechanism to produce diverse labellings of the data.

Figure 4.2 shows 8 heads, 3 of which are computed from early residual blocks of the network. The hierarchical arrangement is induced by only updating head-specific weights according to comparisons made with earlier heads. In practice this is done by stopping the appropriate gradients – ⑥ and all red crosses. For example, when computing the MI between $z_{i=6}$ and those using $z_{i\neq 6}$, gradient back-propagation is allowed when $i < 6$ but not when $i > 6$. In other words, when learning to produce $z_{i=6}$, the network is encouraged to produce a head that is distinct from heads 'lower' on the hierarchy. Extending this concept for $i = 1 \dots 8$ gives rise to the hierarchy (in terms of complexity). Initial experiments showed that if this routine was ignored, the gains were reduced.

4.3.2 Objective

The part of the objective producing high MI representations by 'pulling' together discrete labellings from augmentations is Equation 4.1 normalised over k heads:

$$\mathrm{MI}_{pull} = \frac{1}{k}\sum_{i=0}^{k} MI_{aug}(c_i, c_i'). \tag{4.3}$$

The quantity used to 'push' heads apart is Equation 4.2 normalised per head:

$$\mathrm{MI}_{push} = \sum_{i=1}^{k} \frac{\sum_{\substack{j=1 \\ j\neq i}}^{i} MI_{head}(c_i, c_j)}{i}, \tag{4.4}$$

where each cross-head MI term is scaled by the head index, i, since that directly tracks the number of comparisons made for each head. i scales up the hierarchy, such that the total MI_{head} associated with any head is scaled according to *the number of comparisons*. Scaling ensures that head-specific weight updates are all equally important. The final optimisation objective is:

$$\theta^* = \arg\max_{\theta} \mathrm{MI}_{pull} - \alpha \mathrm{MI}_{push}, \tag{4.5}$$

where θ are the network parameters, α is a hyper-parameter we call the *cross-head MI-minimisation coefficient*. For an ablation study we set $\alpha = 0$ in Section 4.4.

4.3.3 Design and training choices

The DHOG architecture (Figure 4.2) is based on a ResNet-18 backbone, where each residual block has two layers (with a skip connection over these). Blocks 1 to 3 have 64, 128, and 256 output units, respectively. Each parallel final block (4 to 8, here) have 512 units. Each MLP has a single hidden layer of width 200. Early experiments showed that entire block repetition was important to enable sufficient model flexibility. Similar to IIC [Ji et al., 2019] we used four data augmentation repeats with a batch size of 220.

DHOG maximises MI between discrete labellings from different data augmentations. This is equivalent to a clustering and is similar to IIC. There are, however, key differences. In our experiments:

- We train for **1000 epochs** with a cosine annealing learning rate schedule.
- We **do not use sobel edge-detection** or any other preprocessing as a fixed processing step.

- We make use of the fast auto-augment CIFAR-10 data augmentation strategy (for all tested datasets) found by [Lim et al., 2019]. We then randomly apply (with $p = 0.5$) grayscale and take random square crops of sizes 64 and 20 pixels for STL-10 and other datasets, respectively.

The choice of data augmentation is important, and we acknowledge that for a fair comparison to IIC the same augmentation strategy must be used. The ablation of any DHOG-specific loss (when $\alpha = 0$) largely recreates the IIC approach but with network and head structure matched to DHOG; this enables a fair comparison between an IIC and DHOG approach.

Since STL-10 has much more unlabelled data of a similar but broader distribution than the training data, the idea of 'overclustering' was used by Ji et al. [2019]; they used more clusters than the number of classes (70 versus 10 in this case). We repeat each head with an overclustering head that does not play a part in the cross-head MI minimisation. The filter widths are doubled for STL-10. We interspersed the training data evenly and regularly through the minibatches.

To determine the DHOG cross-head MI-minimisation coefficient, α, we carried out a non-exhaustive hyper parameter search using only CIFAR-10 images (without the labels), assessing performance on a held out validation set sampled from the training data. This did not use the evaluation data.

Assessment Once learned, the optimal head can be identified either using the highest MI, or a small set of labelled data. Alternatively all heads can be used according to some posthoc selection for a downstream task. In this chapter the head that maximises the normalised mutual information on the training data is chosen. *This is then fully unsupervised*, as with the head selection protocol of IIC. We also give results for the best posthoc head to show the potential for downstream analysis.

4.4 Experiments

The datasets used for assessment were CIFAR-10 [Krizhevsky et al., 2009], CIFAR-100-20 (CIFAR-100 [Krizhevsky et al., 2009] where classes are grouped into 20 superclasses), CINIC-10 [Darlow et al., 2018] (an extension of CIFAR-10 using images from Imagenet [Deng et al., 2009] of similar classes), street view house numbers [Netzer et al., 2011] (SVHN), and STL-10 [Coates et al., 2011]. For CINIC-10 only the standard training set of 90000 images (without labels) was used for training.

Table 4.1 gives the accuracy, normalised mutual information (NMI), and the adjusted rand index (ARI) between remapped assignments and classification targets. Before assessment a labelling-to-class remapping was computed using the training data and the Hungarian method [Kuhn, 1955]. The results listed for DHOG were computed using 3 seeded runs, where the error bars are standard deviation over these runs. IIC was state of the art over the course of development of DHOG, and DHOG still provides an alternative approach to other concurrent methods. Importantly, no Sobel edge-detection was used for any of our experiments, including when we set $\alpha = 0$.

We used an unsupervised posthoc head selection using $\text{NMI}(c_i, c_i')$ – *which corresponds directly to the original MI objective*. The selected heads almost always corresponded with the head that maxmimised $\text{NMI}(c_i, y)$, where y are class labels. DHOG produces data groupings that:

1. **Better maximise the widely used MI objective** and therefore is an effective mechanism for dealing with sub-optimal MI optimization owing to greedy SGD, as discussed in this work. Table 4.2 gives the NMI with and without DHOG (controlled by α) to confirm this.

2. Correspond better with the challenging **underlying object classification test objective**.

The advantage of a hierarchical ordering is evident when considering the ablation study: with ($\alpha = 0.05$) and without ($\alpha = 0$) cross-head MI minimisation. Figure 4.3 (a) and (b) are accuracy versus head curves, showing that without cross-head MI minimisation later heads converge to similar solutions. The confusion matrices in Figure 4.4 (b) show the classes the final learned network confuses in CIFAR-10. Compare this to the confusion matrix in Figure 4.4 (a) where $\alpha = 0$ and note the greater prevalence of cross-class confusion.

A claim throughout this chapter is that greedy training of neural networks can result in sub-optimal MI maximisation. Table 4.2 shows that for all datasets except STL-10 (for which further experiments are needed) DHOG resulted in a better MI result, thereby directly improving the training objective.

4.4.1 Cluster Visualisation

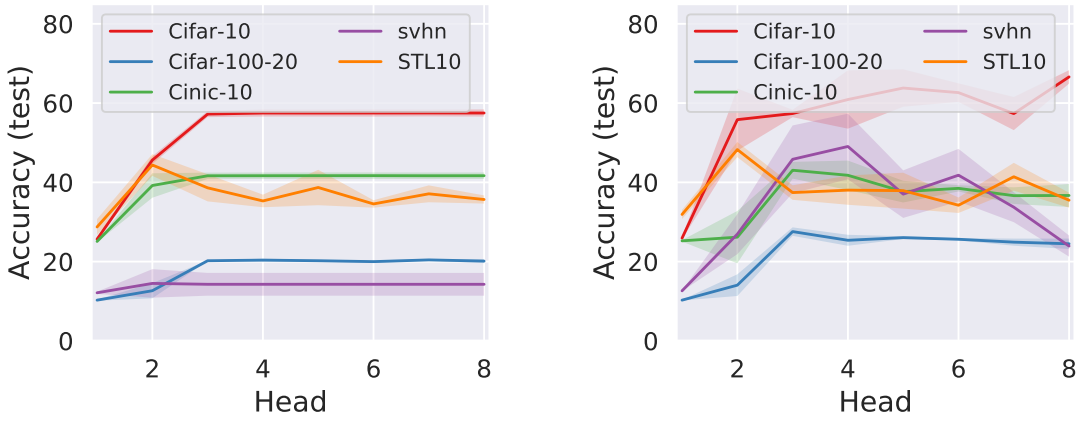
In order to aid understanding of the types of solutions that form with respect to the complexity hierarchy discussed throughout this section, we can visualise the consis-

	Method	Accuracy	NMI(c, y)	ARI
CIFAR-10	Cartesian K-means	22.89	0.0871	0.0487
	JULE	27.15	0.1923	0.1377
	DEC †	30.1	-	-
	DAC	52.18	0.3956	0.3059
	DeepCluster †	37.40	-	-
	ADC	29.3 ± 1.5	-	-
	DCCM	62.3	0.496	0.408
	IIC †	61.7	-	-
	SCAN	87.6 ± 0.4	0.787 ± 0.005	0.758 ± 0.007
	MMDC	82.0 ± 1.9	0.703 ± 0.011	-
	MPCC	64.25 ± 5.31	-	-
	GATCluster	61.0	0.475	0.402
	DHOG ($\alpha = 0$, ablation)	57.49 ± 0.8929	0.5022 ± 0.0054	0.4010 ± 0.0091
	DHOG, unsup. ($\alpha = 0.05$)	66.61 ± 1.699	0.5854 ± 0.0080	0.4916 ± 0.0160
	DHOG, best ($\alpha = 0.05$)	66.61 ± 1.699	0.5854 ± 0.0080	0.4916 ± 0.0160
CIFAR-100-20	DAC †	23.8	-	-
	DeepCluster †	18.9	-	-
	ADC	16.0	-	-
	IIC †	25.7	-	-
	SCAN	45.9 ± 2.7	0.468 ± 0.013	0.301 ± 0.021
	MPCC	35.21 ± 1.69	-	-
	GATCluster	28.1	0.215	0.116
	DHOG ($\alpha = 0$, ablation)	20.22 ± 0.2584	0.1880 ± 0.0019	0.0846 ± 0.0026
	DHOG, unsup. ($\alpha = 0.05$)	26.05 ± 0.3519	0.2579 ± 0.0086	0.1177 ± 0.0063
	DHOG, best ($\alpha = 0.05$)	27.57 ± 1.069	0.2687 ± 0.0061	0.1224 ± 0.0091
CINIC-10	K-means on pixels	20.80 ± 0.8550	0.0378 ± 0.0001	0.0205 ± 0.0007
	DHOG ($\alpha = 0$, ablation)	41.66 ± 0.8273	0.3276 ± 0.0084	0.2108 ± 0.0034
	DHOG, unsup. ($\alpha = 0.05$)	37.65 ± 2.7373	0.3317 ± 0.0096	0.1993 ± 0.0030
	DHOG, best ($\alpha = 0.05$)	43.06 ± 2.1105	0.3725 ± 0.0075	0.2396 ± 0.0087
SVHN	ADC	38.6 ± 4.1	-	-
	DHOG ($\alpha = 0$, ablation)	14.27 ± 2.8784	0.0298 ± 0.0321	0.0209 ± 0.0237
	DHOG, unsup. ($\alpha = 0.05$)	45.81 ± 8.5427	0.4859 ± 0.1229	0.3686 ± 0.1296
	DHOG, best ($\alpha = 0.05$)	49.05 ± 8.2717	0.4658 ± 0.0556	0.3848 ± 0.0557
STL-10	JULE †	27.7	-	-
	DEC	35.90	-	-
	DAC	46.99	0.3656	0.2565
	DeepCluster †	33.40	-	-
	ADC	47.8 ± 2.7	-	-
	DCCM	48.2	0.376	0.262
	IIC †	59.6	-	-
	SCAN	76.7 ± 1.9	0.680 ± 0.012	0.616 ± 0.018
	MMDC	69.4 ± 1.3	0.593 ± 0.005	-
	GATCluster	58.3	0.446	0.363
	DHOG ($\alpha = 0$, ablation)	38.70 ± 4.4696	0.3878 ± 0.0331	0.2412 ± 0.0265
	DHOG, unsup. ($\alpha = 0.05$)	48.27 ± 1.915	0.4127 ± 0.0171	0.2723 ± 0.0119
	DHOG, best ($\alpha = 0.05$)	48.27 ± 1.915	0.4127 ± 0.0171	0.2723 ± 0.0119

Table 4.1: Test set results on all dataset. Results with † are from [Ji et al., 2019]. $\mathrm{NMI}(c, y)$ is between remapped predicted label assignments and class targets. Ablation is DHOG with $\alpha = 0.0$, which is most similar to IIC. Our method is DHOG with $\alpha = 0.05$, highlighted in blue. We give results for the head chosen for the best $\mathrm{NMI}(z, z')$ and the head chosen for the best $\mathrm{NMI}(c, y)$.

Dataset	**NMI**(c, c'), $\alpha = 0$	**NMI**(c, c'), $\alpha = 0.05$
CIFAR-10	0.7292 ± 0.0066	0.7994 ± 0.0081
CIFAR-100-20	0.6104 ± 0.0098	0.6506 ± 0.0040
CINIC-10	0.6408 ± 0.0015	0.6991 ± 0.0044
SVHN	0.6337 ± 0.0085	0.7265 ± 0.0093
STL-10	0.6713 ± 0.0175	0.6610 ± 0.0084

Table 4.2: NMI between representations of augmentations for the training dataset, evidencing that cross-head MI minimisation does indeed aid in avoiding local optima of this objective function.



(a) Without DHOG training

(b) With DHOG training

Figure 4.3: Accuracy per head. DHOG causes heads to learn distinct solutions.

tency of clusters in terms of average images and top-probability examples. Table 4.3 shows the average images in clusters as well as top-probability examples. Note how for the earlier head the average images pay heed to very similar pixel values, while for the later head the low-level pixel values have less influence.

4.5 Conclusion

I presented deep hierarchical object grouping (DHOG): a method that improves the capability of a deep clustering neural network by modelling simultaneously multiple local optima to the training objective. Learning a good representation of an image using data augmentations is limited by the user, who chooses the set of plausible data augmentations but who is also unable to cost-effectively define an ideal set of augmen-

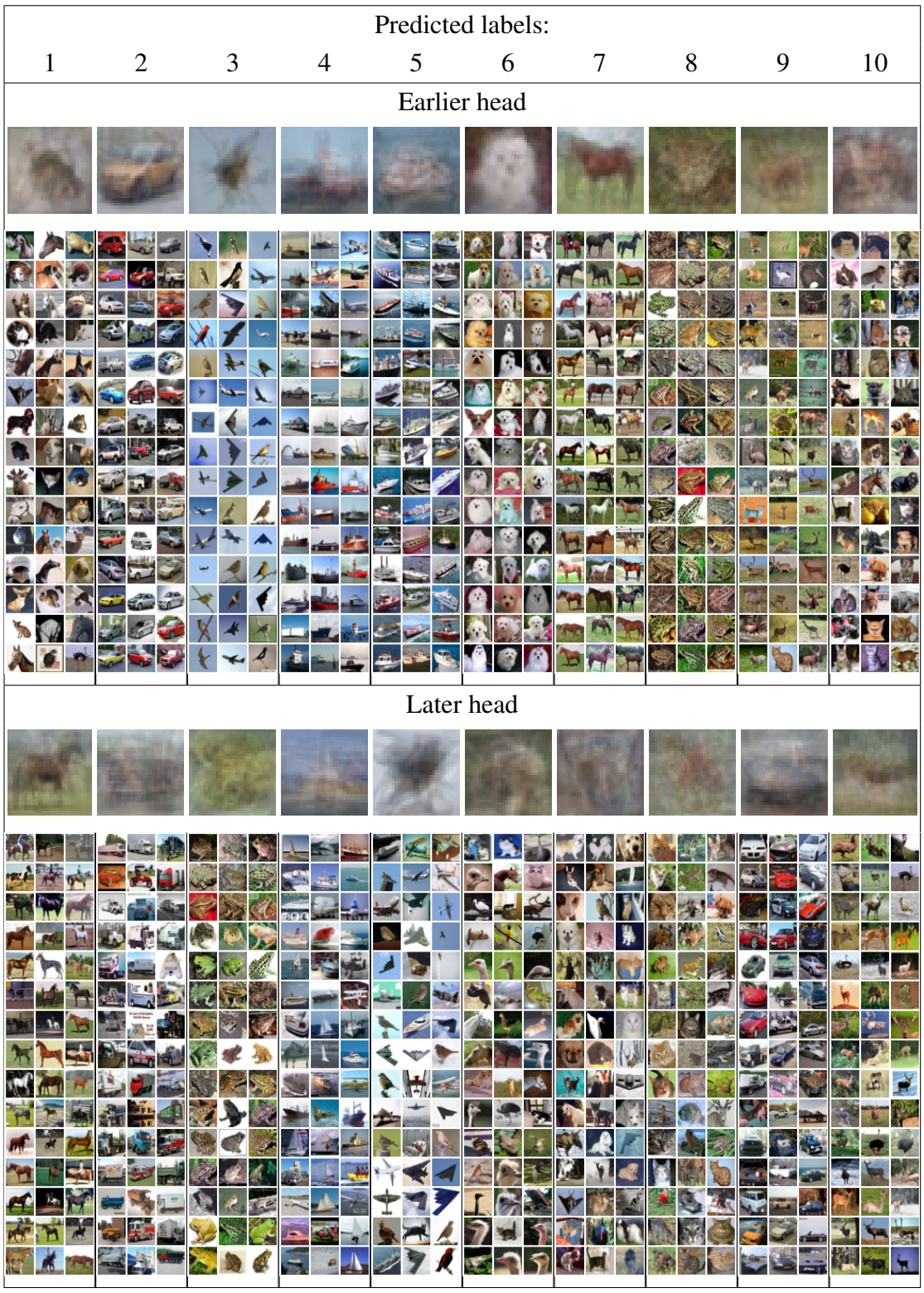
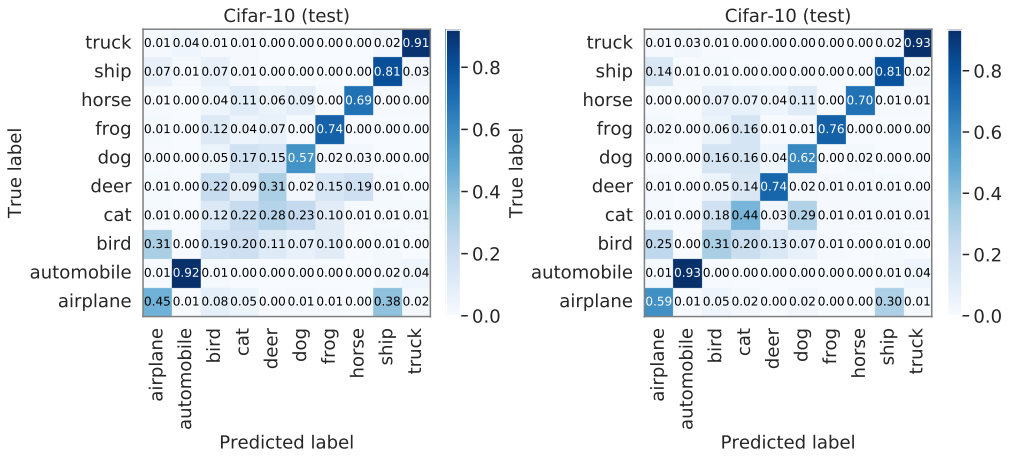


Table 4.3: Images that yielded the top probability for each discrete label for an early, $i = 2$, and late, $i = 8$, head, taken from a single DHOG run on Cifar-10. The average image for the top 10 is also shown. Note particularly the those images grouped by the early head are less diverse than those grouped by the later head. The label associated largely with frogs (8 for the earlier head and 3 for the later head) exemplifies this well.



(a) Without DHOG training (b) With DHOG training

Figure 4.4: Confusion matrices from the same seed (a) without and (b) with DHOG cross-head MI minimisation. These networks struggle with distinguishing natural objects (birds, cats, deer, etc.), although DHOG does improve this.

tations. I argued and showed that learning using greedy optimisation typically causes models to get stuck in local optima, since the data augmentations fail to fully describe the sought after invariances to all task-irrelevant information.

DHOG learns a simple-to-complex ordered hierarchy of representations to prevent a neural network from getting stuck in a sub-optimal local optima of the clustering objective. DHOG works by minimising mutual information between these representations such that those later in the hierarchy are encouraged to produce unique and independent discrete labellings of the data (w.r.t. earlier representations). Therefore, later heads avoid becoming stuck in the same local optima of the original mutual information objective (between augmentations, applied separately to each head). Our tests showed that DHOG resulted in an improvement over the baseline on CIFAR-10, CIFAR-100-20, and SVHN, without using preprocessing such as Sobel edge detection, and a consistent improvement of the underlying MI objective.

Subsequent to the work done in this chapter developments have been made in this line of work by a variety of authors. I detail several advances at the end of Section 2.2.3.

Chapter 5

Latent adversarial debiasing

The work in this chapter was completed with Dr Stanisław Jastrzębski [Darlow et al., 2020], who helped with discussions about the latent adversarial walk and to collect and summarise the relevant literature. My contributions constitute the majority of this work, including the core idea, implementation, and experiments.

5.1 Overview

In any predictive modelling problem, it is possible for bias to be introduced when an underlying causal variable ends up being strongly correlated with other variables only as a result of the training data collection procedure. These other variables are often called *extraneous variables* as their connection with any dependent variable we wish to predict is not there in the environment, but only induced by data collection. This sort of bias is called *collider bias*, and is a harmful form of sample selection bias that, in this chapter, I demonstrate neural networks are ill-equipped to handle. Model training can build dependence on the spurious signal, which results in model failure at test time when that dependence is not present. An example of this is given in Section 5.2.1.

Typically we do not observe dependent or extraneous variables directly. Rather we observe an image that might express signals from both variables at the same time. In the situation where the *dependence on spurious signals from the extraneous variable is easy-to-learn*, deep neural networks will latch onto this and the resulting model will generalise poorly to in-the-wild test scenarios. We argue in this chapter that the cause of failure is a combination of the deep structure of neural networks and the greedy gradient-driven learning process used – one that prefers easy-to-compute signals when available. We show it is possible to mitigate against this by generating bias-decoupled training data using latent adversarial debiasing (LAD), even when the spurious signal is

present in 100% of the training data. By training neural networks on these adversarial examples, we can improve their generalisation in collider bias settings. Experiments show state-of-the-art performance of LAD in label-free debiasing with gains of 76.12% on background coloured MNIST, 35.47% on foreground coloured MNIST, and 8.27% on corrupted CIFAR-10.

5.2 Introduction

Invariably, in real-world machine learning settings, training and test sets are different. This general phenomenon has become known as *dataset shift* [Caron et al., 2018]. Yet there are many causes for such shifts [Storkey, 2009]. One common scenario is *sample selection bias* [Heckman, 1979] where the process of curating a training dataset differs from the process by which data arrives during deployment. This issue is ubiquitous; even standard machine learning benchmarks (e.g. ImageNet) contain images selected for the clarity with which a class is represented, a clarity missing in many real applications, where for example you might see an object occluded.

One pernicious form of sample selection bias is *collider bias*. This is illustrated and characterised in Figure 5.1. Consider two variables: a *causal variable* that determines the target, and what we will call an *extraneous variable*, that is not directly related to the target. In collider bias, these two variables that are, for the most part, independent in the test scenario, become co-dependent in the training sample because of the restrictive way the training data is selected. Collider bias can cause a predictive algorithm to mistakenly target information from features associated with the extraneous variable rather than the causal variable; such features then do not generalise to the test scenario.

Even if the causal signal has higher information content than the spurious signal it may be harder to discover. This is usually due to extraneous variables being easier to disentangle from images than causal variables – a highly linear spurious signal, for example can be discovered before a highly non-linear causal signal. When this is combined with the greediness of neural network learning, it can mean the causal signal is simply never used. It is precisely this scenario that is the topic of this chapter.

I argue that collider bias can be a pervasive cause of non-robustness in deep neural networks. A main contribution of this work is the demonstration of a specific approach to mitigate the situation: latent adversarial debiasing (LAD), which pushes a network to recognise all sources of information for a problem by augmenting training using adversarially perturbed latent representations. In these latent representations, easy-to-

learn spurious signals are decoupled from the classification targets, forcing networks to also learn information from the causal signal.

5.2.1 Collider bias

Consider a toy classification problem: distinguishing dog images from cat images. A collected training dataset for dogs and cats may be biased: people take pictures of dogs outside while they take them for a walk, e.g., in a field, and take pictures of cats in their homes, e.g., on a sofa. Yet both cats and dogs go inside and outside.

It might be critical in the test setting to distinguish between dogs and cats each presented in indoor and outdoor settings. In this training dataset the simple feature of the background colour in the image is an extraneous variable that may overshadow the causal variable when learning a neural network to detect the difference in appearance between dogs and cats.[1]

This easy-to-learn bias-inducing signal is a **spurious signal**, as opposed to the true **causal signal**. A model relying on the causal signal will generalise well, while a model relying on the spurious signal will generalise poorly, because that dependence is not present at test time. I propose a solution that hinges on the assumption that spurious signals are typically *easier-to-compute* in that their gradients during learning are stronger than the gradients of causal signals (see Section 5.3.1). A similar assumption was also made in related earlier work [Nam et al., 2020, Bahng et al., 2020, Bras et al., 2020, Minderer et al., 2020]. I hypothesise that a specific form of adversarial examples can be used to augment training data such that spurious signal is decoupled from the causal signal. Adversarial data can be generated by gradient descent in the latent space of an autoencoder-like model to produce augmented training data. The effect of training on this data is a reduced association between spurious and causal signals, requiring a model to rely on the causal signal. LAD shows marked improvement over state-of-the-art, without relying on any presence of bias-free data (as opposed to Nam et al. [2020], Bras et al. [2020] and Bahng et al. [2020]).

[1]The Neural Network Tank Urban Legend (`https://www.gwern.net/Tanks`) is, in fact, another example of collider bias failure in neural networks. Or at least, it would be, if it had ever happened.

5.3 Problem Definition

Consider the probabilistic graphical model in Figure 5.1. In this setup there are multiple underlying unobserved variables[2], e.g. z_1 and z_2, a binary sample selection variable, s, and observed variables x and y. Here z_1 and z_2 correspond to causal and extraneous variables that influence the observed data. While for most problems we do not have access to this distribution, $P_{\mathcal{D}}(x,y)$ for data $\mathcal{D}$, representative empirical samples are typically available. Setting s to a value (e.g., without loss of generality, $s = 1$) implies the data is subject to a particular selection criterion which restricts the data to samples from $P(z_1, z_2|s)$. Ideally, we want a model to generalise across all possible values of s; that is to the unconditioned case $P(z_1, z_2)$.

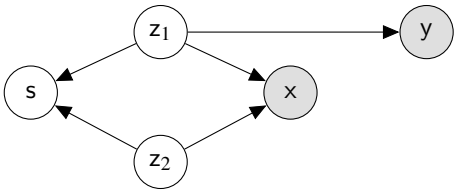


Figure 5.1: Graphical model for collider bias. z_1, z_2 are latent information sources. The target y is causally dependent on z_1 but not z_2. But x contains information from both latent sources. s is a binary sample selection variable. Setting s implies data is subject to some selection criterion.

When selecting or collecting data for a problem we inevitably introduce a *sampling bias*, effectively setting s to some value, which can be thought of as a form of rejection sampling. For example, all the times a user implicitly chooses *not* to take a photo, or not to include a photo in a particular collection. In this way sampling bias couples the underlying variables of the observed data in a manner that neural networks are ill-equipped to handle.

We have included the sampling mechanism, s, explicitly in Figure 5.1. Conditioning on s is what actually causes the association between underlying variables. In both cases the secondary signal does not cause the target and is therefore a **spurious signal**.

5.3.1 Spurious signals have high-gradient learning signals

Crucial to our insight into the tendency of neural networks to rely on easy-to-learn spurious signals in the training data is understanding how the gradient of the training

[2]Regarding notation, we are using x to denote a random variable and x to denote an observation thereof.

loss with respect to these signals evolves over learning. To this end I present a toy problem, called the one-pixel problem [3] in Section 5.3.2, and track the average gradient ratio (*GR*) to determine the relative reliance on causal (z_+) and spurious (z_-) variables:

$$GR(\mathsf{z}_+, \mathsf{z}_-) = \frac{1}{N}\sum_{i}^{N}\left[\left\|\frac{\partial L}{\partial z_{+,i}}\right\|_2 \Big/ \left\|\frac{\partial L}{\partial z_{-,i}}\right\|_2\right], \tag{5.1}$$

where $\|\cdot\|_2$ is the L2 Norm. The notation $z_{\cdot,i}$ refers to an observation i of the variable. When $GR = 1$ the learning process is not favouring either signal. When $GR < 1$ the learning process is favouring the spurious signal and when $GR > 1$ it is favouring the causal signal. As this quantity is impossible to compute when we do not have direct access to the causal variable, we have focused here on a toy problem to enable us to assess this.

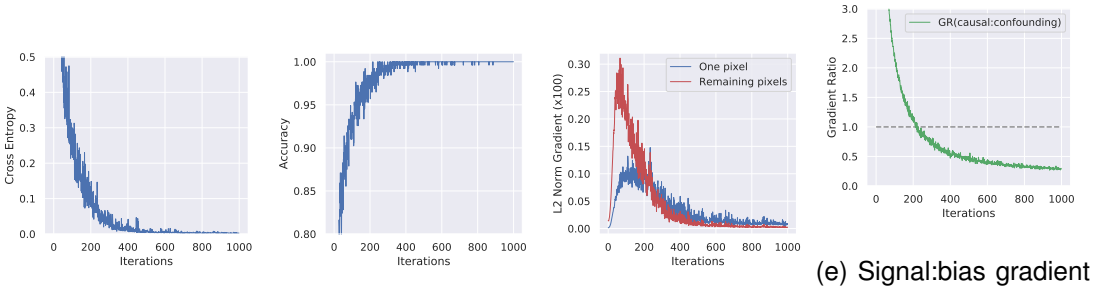
5.3.2 One-Pixel problem

The one-pixel problem is a bias-reliance demonstration that can be constructed simply using any image dataset for classification: all that it requires is setting the k^{th} pixel of the first row of each training image to a pre-selected value (where k is the class index) – see Figure 5.2.

Figure 5.2 tracks the gradient ratio between gradients with respect to the causal signal and those with respect to the spurious signal, $GR(\mathsf{z}_+, \mathsf{z}_-)$. This is measured as the ratio of the L2 norm of gradients w.r.t. the image pixels (all but the k pixels of the one-pixel problem) and the L2 norm of gradients w.r.t. the k pixels encoding class information, over 1000 minibatch iterations. This is simple to compute for the one-pixel problem as there are distinct pixels associated with each signal. At initialisation the gradients are stronger over the image space (the causal signal), but become dominated by the gradients over the k-pixel space (spurious) after only 230 iterations. This demonstration serves to show that neural networks prefer easy-to-compute spurious signals and that gradient information is the mechanism that preference takes.

[3] Identification of this issue was a result of discussions within in the research group with Harrison Edwards in 2016.

(a) Training images



(b) Cross entropy loss (c) Training accuracy (d) L2 norm gradients (e) Signal:bias gradient ratio, Equation 5.1

Figure 5.2: One-pixel problem on MNIST. The class information is encoded as a spurious signal in the first k pixels of the first row in each image (a). We show the training loss (b), training accuracy (b), l2 norm of gradients on the informative pixel and the remaining image (c), and the ratio of said gradients (d). In this case the test accuracy is no better than random chance.

5.4 Prior Work

5.5 LAD: Latent Adversarial Debiasing

In this work, I consider countering collider bias in neural networks by augmenting the training data to artificially disassociate spurious signals and causal signals. Three components are required to achieve this:

- A latent representation, h, of the underlying data manifold, modelled by a neural network mapping g, such that the spurious and causal signals are approximately disentangled and accessible.
- A biased classifier, f, trained to predict classes y from observations h.
- A method to remove spurious signals from training examples using both f and g.

In the rest of this section I describe each component. Note that similar but more restrictive assumptions were made by prior works [Nam et al., 2020, Bras et al., 2020,

Bahng et al., 2020, Karras et al., 2018]. Perhaps most importantly, we assume f relies on the easy-to-learn spurious signal when trained on h observations. f is also constrained to ensure this is the case.

5.5.1 Manifold Access

We assume the data distribution is conditional on a set of underlying variables resulting in the distributions: $p(x \mid z_1, z_2)$ and $p(y \mid z_1)$ (see Figure 5.1). This corresponds to an underlying low-dimensional manifold that dictates the space of plausible images in the data.

We need access to (an approximation of) the latent manifold parameterised in a way that the easy-to-compute spurious signal is disentangled from the causal signal. This helps ensure that when we train a classifier f, it learns to rely on the information that induces bias, and allows us to alter or remove this information.

Stutz et al. [2019] demonstrated a means of producing on-manifold adversarial examples by training class-specific variational autoencoder generative adversarial network (VAEGAN) hybrid models [Larsen et al., 2016] for each class in the data. Via an adversarial walk on the approximated manifold space, Stutz et al. [2019] were able to generate adversarial images with plausible deviations from the originals.

VQ-VAE: quantised latent space To satisfy the above desiderata, we can use a vector quantised-variational autoencoder (VQ-VAE) [Van Den Oord et al., 2017]. This model enables learning of a discrete (quantised) latent representation, where the number and size of the discrete codes are chosen prior to learning. Where Stutz et al. [2019] learned VAEGAN models for each class to constrain the changes such that they remain approximately on-manifold, the quantisation constraint of VQ-VAE offers a similar effect. We use the quantisation mechanism directly in the latent adversarial walk to project gradient-based changes onto the manifold. Early experimentation with standard autoencoders and VAEs evidenced that a strong constraint in the adversarial walk was paramount to downstream performance.

The VQ-VAE is effectively an encoder decoder structure (See Figure 5.3) with a quantised latent space. The straight-through gradient trick [Bengio et al., 2013b] is used to enable learning given the quantised space. Consider that the original image

can be reconstructed as:

$$\begin{aligned}\hat{x} &= g(x) \\ &= \text{dec}(h), h = \text{enc}(x),\end{aligned} \tag{5.2}$$

where x and $\hat{x}$ are the input and reconstructed images, respectively, $g(\cdot)$ is the VQ-VAE, (dec, enc) are the decoder and encoder components thereof, and h is the latent representation for x.

Classification from bias We can then attach a classifier f to the latent space h in order to approximate the decision boundary associated with the easy-to-learn spurious signal:

$$\hat{y} = f(h), \tag{5.3}$$

where $\hat{y}$ is a class prediction, and train it using standard SGD to minimise the cross-entropy loss. **While h will contain both spurious and causal signals, it is the tendency of f to latch onto easy-to-learn features that enables LAD to work**. In the following section I discuss how to use the two models $f(\cdot)$ and $g(\cdot)$ to traverse the latent space h, thereby augmenting the training data such that resultant decoded images are observations where the causal and extraneous variables are decoupled.

LAD bears resemblance to the work by Gowal et al. [2020], who assume that the lowest level representation learned by StyleGAN is not causally linked to the label. We relax this assumption in the sense that we only require our latent variable model to disentangle the underlying causal variables. We will rely on a (simple) classifier (f) and a gradient-based latent adversarial walk to decouple the spurious and causal signals.

5.5.2 Latent adversarial walk

While a number of adversarial attack example generation algorithms exist [Szegedy et al., 2013, Madry et al., 2018, Goodfellow et al., 2015], these aim at producing imperceptible changes to an image such that a target classifier makes an incorrect, yet often highly confident, prediction on that image. Consider the family of white-box adversarial attacks [Madry et al., 2018] that maximise the training loss:

$$\max_{\delta} \mathcal{L}_{ce}(f(x+\delta), y) \tag{5.4}$$

where $\mathcal{L}_{ce}$ is the cross-entropy loss and δ is computed via projected gradient descent. δ is constrained to ensure perceptual similarity between x and $x+\delta$. This is an optimisation process on the image space: an **adversarial walk** to systematically alter the

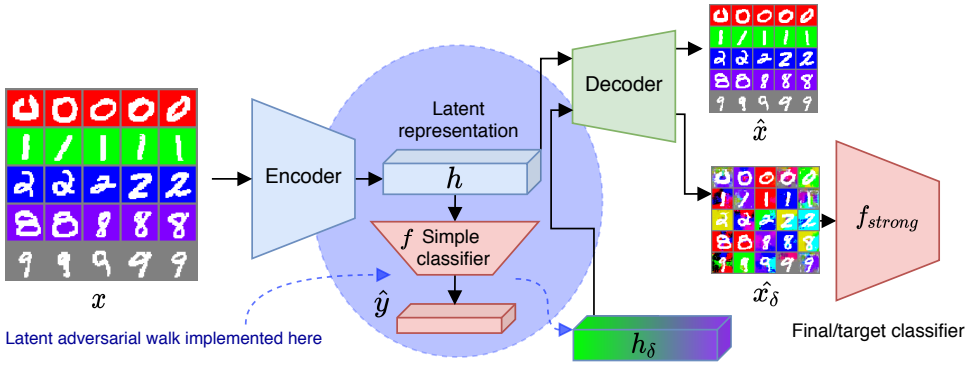


Figure 5.3: Model setup. The VQ-VAE is the encoder decoder structure, tasked with learning quantised latent representation, h. The 'simple' classifier, f, is learned using (h, y) as input-target pairs, where y is the class label. The latent adversarial walk (blue circle) alters h such that f produces high entropy (maximally uncertain) class predictions. The altered representation, h_δ, can then be decoded into x_δ where the easy-to-learn spurious signal and causal signal are decoupled. The ultimate aim is to train an additional classifier, f_{strong} using this augmented data for improved generalisation in collider bias settings.

image and maximise the training loss. In our case, however, we perform the adversarial walk on the latent space with the objective of maximising the entropy of the predictive probability:

$$\max_{\delta} H(f(\text{quantise}(h+\delta, G))), \tag{5.5}$$

where H is the entropy of the class probabilities f computes, the quantise$(\cdot)$ function is the quantisation mechanism of VQ-VAE with learned latent codes. We compute δ as the standardised partial gradient to preserve the strength of the changes for any example:

$$\delta = \alpha \cdot \left(\frac{\frac{\partial H}{\partial h} - \text{mean}(\frac{\partial H}{\partial h})}{\text{std}(\frac{\partial H}{\partial h})} \right), \tag{5.6}$$

where mean$(\cdot)$ and std$(\cdot)$ are the mean and standard deviation computed over the entire gradient vector for each image, and α is a hyper-parameter dictating the step size of the walk. Standardisation helps ensure the steps taken by the gradient walk are approximately equal in length.

Algorithm 1 details the **quantisation-constrained entropy-targeted adversarial walk** (QE-walk) used for LAD. The end-goal here is to produce an altered latent representation, $h_\delta = \text{advwalk}(h, , f)$.

Algorithm 1: Quantisation-constrained entropy-targeted adversarial walk.

Input : $f(\cdot)$, y, h, α, $steps$, $g(\cdot)$

Output: Adjusted latent representation, h_δ

$h_\delta \leftarrow h$ `// initialise` h_δ `to` h

for $i \leftarrow 1$ **to** $steps$ **do**

  $\hat{y} \leftarrow f(h_\delta)$ `// compute prediction`

  $H \leftarrow entropy(\hat{y})$;

  backprop to maximise H;

  compute δ;

  $h_\delta \leftarrow h_{delta} + \alpha \cdot \delta$;

  $h_\delta \leftarrow \text{quantise}(g, h_\delta)$;

end

5.5.3 Post-walk classification

Once the latent representation has been adjusted such that the simple classifier, f, outputs high entropy probabilities (i.e., it is unsure which class to predict for h), we can then decode the new representation using the VQ-VAE decoder to construct a new image:

$$\hat{x_\delta} = \text{dec}(h_\delta). \tag{5.7}$$

The goal is that this new image should contain very little information related to what f used to classify. Since f is constrained and we assume it will latch onto the easy-to-learn spurious signals, this gives us a way of intentionally augmenting data to remove these signals. We can then use x_δ to learn an additional classifier in a standard fashion. We call this final debiased classifier f_{strong}.

5.6 Experiments

I explored three datasets of increasing difficulty to assess the relative merit of LAD. For comparison with earlier works, I tested LAD using two variants of coloured MNIST [LeCun et al., 1998] (background and foreground) in Sections 5.6.2 and 5.6.3. Although seemingly similar, these two variants of MNIST differ in the level of entanglement between image shape and colour. For the background-coloured variant, the colour is largely independent of the image shape, while for the foreground-coloured variant, the colour always occurs with shape. Following Nam et al. [2020], I also

tested LAD using the corrupted CIFAR-10 dataset [Krizhevsky et al., 2009].

I consider two forms of assessment. The first is the **independent** setting, denoted 'cross-bias' by Bahng et al. [2020], where the spurious signal is independent of the causal signal during test (e.g., colours are sampled independently at random during test for coloured MNIST datasets). The second is called the **conditioned** setting, where the sample selection is changed so the spurious signal is held constant during the test (e.g., the original MNIST test set with black a background and white foreground). These cover two different test-scenarios where the spurious signal has no influence in the test setting.

Bias ratio For comparison to earlier works the concept of bias ratio needs to be introduced. This is the ratio of biased to unbiased training data. For example, using the concepts defined in this chapter, a bias ratio of 90% would mean that 9 out of 10 training observations would be affected by sample selection bias such that the underlying causal and extraneous variables are coupled. The remaining training samples will not exhibit this coupling.

Unlike earlier works, I chose to consider the circumstance where the different spurious signals are pervasive. Therefore I did not assume any training data was free from bias. The idea behind LAD is to handle the fundamental issue of easy-to-compute spurious signals, instead of leveraging small amounts of bias-free data. Since earlier works do not consider training data that is never free from bias, I also compared LAD on the settings they target, ensuring the proportion of biased data is listed consistently for earlier works.

5.6.1 Implementation Details

For all datasets a ResNet-20 [He et al., 2016b] was used for the final f_{strong} classifier, and a single hidden-layer multi-layer perceptron (with a width of 100 units) was used for f. For both MNIST datasets, the VQ-VAEs were trained with 20 discrete latent codes of length 64. For corrupted CIFAR-10 the VQ-VAE was trained with 2056 discrete latent codes of length 64.

Regarding the QE-walk: 20 steps with $\alpha = 0.1$ was used for background coloured MNIST; 7 steps with $\alpha = 0.1$ was used for foreground coloured MNIST; and 4 steps with $\alpha = 0.07$ was used for corrupted CIFAR-10. These settings were determined using a brief hyper-parameter search and cross-validation. The results given in the

Dataset	Method	Bias ratio	Accuracy (independent)	Accuracy (conditioned)
BG coloured MNIST	LAD	100%	**98.82 ± 0.039 %**	95.35 ± 0.32%
	Vanilla	100%	0.00 ± 0.00%	10.64 ± 0.65%
	ReBias	99%	88.1%	-
	ReBias	99.9%	22.7%	-
FG coloured MNIST	LAD	100%	**98.86 ± 0.10%**	98.34 ± 0.40%
	Vanilla	100%	0.01 ± 0.01%	9.90 +- 0.13
	LfF	95%	85.39 ± 0.94%	-
	LfF	99%	74.01 ± 2.21%	-
	LfF	99.5%	63.39 ± 1.97%	-
Corrupted CIFAR-10	LAD	100%	**39.93 ± 0.62%**	51.89 ± 0.32%
	Vanilla	100%	12.16 ± 0.16	21.49 ± 0.16
	LfF	95%	**59.95±0.16**%	-
	LfF	99%	41.37 ± 2.34%	-
	LfF	99.5%	31.66 ± 1.18%	-

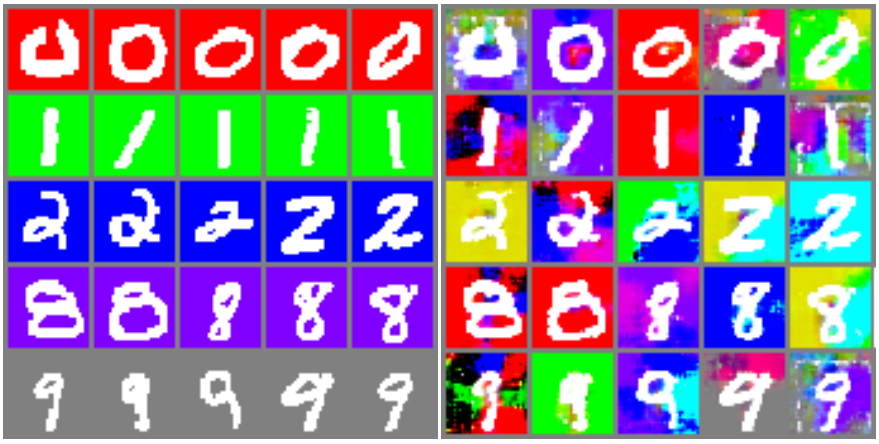
Table 5.1: Test accuracy on all datasets for two test conditions: the independent case and the conditioned case. The vanilla method is simply a f_{strong} (c.f. Figure 5.1) model trained on the original data. While we include results from ReBias [Bahng et al., 2020] and LfF [Nam et al., 2020], their methods are not directly comparable because they assume the training set contains some percentage of unbiased data.

following sections were computed on the held-out test data. Regarding training data augmentation, random crops with padding size of 4 was used for all datasets, random affine transformations were applied to the MNIST images (with limits of: $rotation = 15\ degrees$, $scale = [0.8, 1.1]$, and $shear = 15\ degrees$), and random horizontal flips were applied to the corrupted CIFAR-10 images.

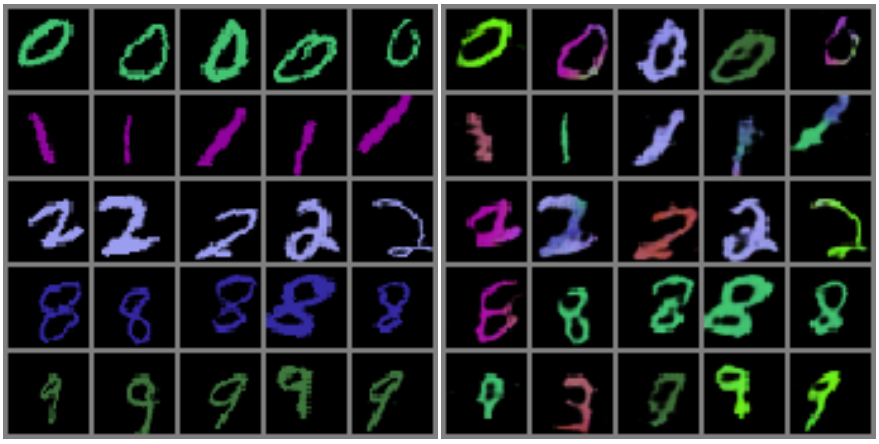
5.6.2 Background coloured MNIST

To produce this dataset we used the code provided [4] by the authors of ReBias [Bahng et al., 2020] and compare to their results in Table 5.1. It is evident that ReBias is strongly dependant on using a small portion of unbiased data: at a bias ratio of 99%, they achieved 88.1% test accuracy on an unbiased test set but only 22.6% at a bias ratio of 99.9%. LAD achieved 98.82% on this dataset at 100% bias ratio, approaching what is achievable on standard MNIST.

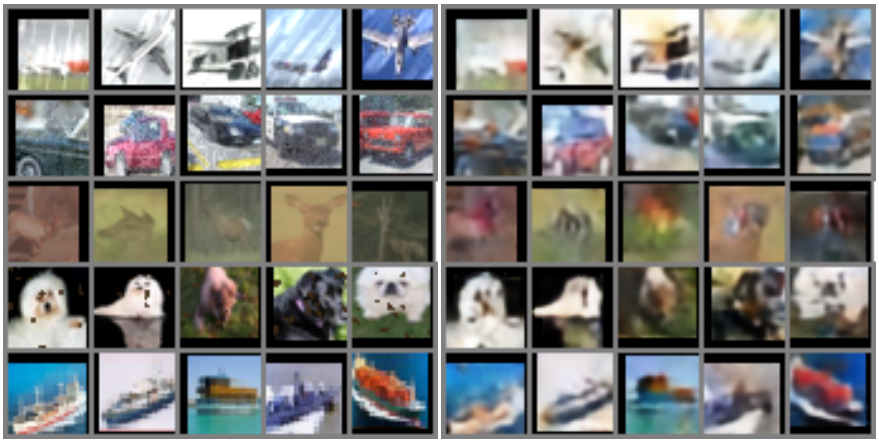
[4] https://github.com/clovaai/rebias



(a) x: BG coloured MNIST (b) $\hat{x_\delta}$: LAD reconstruction



(c) x: FG coloured MNIST (d) $\hat{x_\delta}$: LAD reconstruction



(e) x: corrupted CIFAR-10 (f) $\hat{x_\delta}$: LAD reconstruction

Figure 5.4: Training data examples for (a) background coloured MNIST, (c) foreground colorued MNIST, and (e) corrupted CIFAR-10, with corresponding LAD reconstructions for each dataset in (b), (d), and (f), respectively.

The efficacy of LAD is clearly evident by these results. Figure 5.4 (a) and (b) shows the input data and LAD reconstructions, respectively. LAD is clearly able to augment the colour information such that it is decoupled from the classification targets.

5.6.3 Foreground coloured MNIST

To generate this data the protocol in LfF [Nam et al., 2020] was followed. Ten colours were randomly chosen for each class. For each image an RGB colour is sampled (which is correlated to the class for training data – see Section 5.3) from $\mathcal{N} \sim (RGB, 0.005)$: Gaussian noise is added to the selected mean colour with a standard deviation of 0.005.

Similar to background coloured MNIST, LAD exceeds the state-of-the-art at the time of testing, even though we consider 100% biased training data: the closest comparison of LfF achieves a test accuracy of 63.39% at 99.5% bias while we achieve 98.86% test accuracy. Again, these results are approaching what is achievable on the standard MNIST dataset. However, a close inspection of Figure 5.4 (c) and (d) will evidence that the alterations owing to LAD do begin to affect the the digit shape – note particularly the digit 9 in Figure 5.4 (d). Compare this to (b) where the altered data leaves the digit shape almost entirely unchanged. This difference is owing to the level of entanglement between spurious and true signal. Foreground coloured MNIST has the extraneous variable (colour) overlayed on the causal signal (digit shape) instead of as a static background. Nonetheless, the test accuracies are almost identical.

5.6.4 Corrupted CIFAR-10

Next I consider a constructed dataset where the bias and signal variables are far more entangled. Following Nam et al. [2020], a variant of corrupted CIFAR-10 was constructed with corruptions: Snow, Frost, Fog, Brightness, Contrast, Spatter, Elastic, JPEG, Pixelate, Saturate. During training these correlate with each class such that the causal and spurious signals are coupled. Using corruptions to benchmark neural network robustness is not new – Hendrycks and Dietterich [2019], and using them as a class-informative signal yields an extremely challenging dataset.

Not only are corruptions often nuanced, they are also destructive, meaning that distentangling these from the underlying image is not always possible. Blurring, elastic distortion, JPEG compression, and pixelation are all examples non-reversible corruptions. While some (like contrast or brightness adjustment) are easier to alter, it is understandable that earlier work was only able to achieve 31.66% test accuracy at a

high bias ratio. We were able to achieve 39.93% test accuracy at a bias ratio of 100%. While LfF can achieve 59.95% on corrupted CIFAR-10, this requires a relatively low bias ratio of 95%, once more evidencing the reliance of earlier works on bias-free training data. Note that the reconstructions in Figure 5.4 (f) are blurry, highlighting the need for an improved encoder-decoder model.

5.7 Discussion and Conclusion

Neural networks tend to focus on easy-to-learn features, should those features be sufficiently informative of the target. In this chapter I showed that this problematic behaviour means that neural networks are ill-equipped to handle a form of sample selection bias known as collider bias. The process of collecting and curating training data can often create a scenario where test data differs substantially from training data. When the training data contains a spurious signal (such as lighting conditions), neural networks will generalise poorly. We argue that it is the deep structure of neural networks, combined with the gradient-driven learning process used that amplifies their dependence on easy-to-learn spurious signals.

I presented LAD, a method to produce latent adversarial examples that specifically target the easy-to-learn spurious signals in the data manifold. Using a VQ-VAE to approximate the data manifold corresponding to causal and spurious signals, the tendency of neural networks to latch on to easy-to-learn features could be leveraged to define an appropriate adversarial walk on this manifold. Decoding the adjusted latent manifold back to the image space yielded augmented data where the effect of spurious signals were largely mitigated against. A classifier trained on this new data generalises better. I evidenced substantial test accuracy gains of 76.12% on background coloured MNIST, 35.47% on foreground coloured MNIST, and 8.27% on corrupted CIFAR-10, even when 100% of the training data contained spurious signals.

Since LAD does not require any data without spurious signals, it can be seen as a step toward solving the broad issue that neural networks latch on to easier-to-learn features. While I focused on constructed datasets that demonstrated effectively the problem at hand, extending the ideas and solutions presented in this chapter to broader notions of dataset bias and neural network robustness is a natural progression and is planned for future work.

Chapter 6

Conclusion

Learning good representations of images is important to meet the demands made by downstream tasks, yet challenging because of the high-dimensionality and complex composition of images, and the cost of labelling data for specific tasks. Proxy objectives are widely used when the downstream task is unknown apriori, or when it is too costly or too difficult to label data. For example in self-supervised learning, proxy objectives are cost-constrained such that they do not use classification labels, while the downstream task is classification. The recent success of self-supervised learning [Chen et al., 2020a, Radford et al., 2021] shows that good proxy objectives can enable learning useful and robust representations of images.

However, proxy objective failure can occur when there is a disconnection between training objective and downstream task, such that the learned representations fail to meet the requirements for good performance on the downstream task. In the work described in this Thesis, I considered a common failure where neural networks learned via proxy objective tend to settle in local optima of the objective that are characterised by easy-to-learn features. This is a failure of proxy objectives to adequately describe representational requirements (such as invariance to changes in image colours, for example) or constrain learning (to not rely on easy features in the data, for example), resulting in neural networks that fit adequately the training objective but which fail to generalise to the downstream task(s).

Deep decision tree layer In Chapter 3 I focused on learning efficient compact discrete representations. This type of representation is of particular interest because information efficiency is measurable for fixed capacity discrete representations. Semantic hash codes are compact discrete representations used for the task of fast content-based image retrieval because comparing hash codes is low-cost. Yet it is challenging to use

neural networks to learn deep semantic hash functions because the mapping from image to hash space is highly compressive and involves a non-differential discretisation operation that is incompatible with gradient-based optimisation methods. I showed that common objectives in the literature can also have the tendency to **over-compress**; that is they lose more information than they should in forming hash codes, resulting in reduced retrieval performance and inefficient representations.

I presented the deep decision tree layer to overcome the issue of over-compression and improve the information efficiency of hash codes. To do this, hash codes were composed from a supervised portion and an unsupervised portion, with the former learnt using a classification head and a standard supervised objective, and the latter using a novel deep decision tree layer and a contrastive learning mechanism. The supervised portion was constructed to be a minimal binary representation that full captures the class label(s). The decision path from root to leaf through the decision tree builds the unsupervised portion. Training the decision tree using an unsupervised contrastive objective ensured an approximately even partitioning of images over the leaves in the tree, which in turn resulted in hash codes with good coverage of the available hash space. I provided evidence of: state-of-the-art image retrieval performance on CIFAR-10 and ImageNet100, and comparable performance on NUS-wide; better transfer to more complex datasets (CIFAR-10 to CIFAR-100, and CIFAR-100-20 to CIFAR-100); hash code robustness to distortions in the image space; image retrievals that could be precisely arranged by increasing relevancy with the query image, leveraging both high-level (e.g., class labels) and low-level (e.g., texture or colour) abstract features; and decision tree use for interpretability and dataset exploration.

Deep hierarchical object grouping In Chapter 4 I focused on the proxy objective of maximising mutual information between cluster assignments from differently augmented images. This proxy objective is ill-defined because it has multiple quantitatively similar local optima (e.g., they have similar losses), but which rely on substantially different features in the data (e.g., one could rely on average colour while another could rely on object type). Greedy gradient-based optimisation methods fail to find local optima that also result in good performance on the downstream task of clustering. Instead, neural networks optimised to map differently augmented images to high mutual information representations tend to learn mappings that **favour reliance on easy-to-compute features** in the data, when available. While stronger image augmentations (e.g., Sobel edge-detection or small image crops) make unavailable easy-to-compute

features, they also tend to destroy potentially useful semantic information in an image and are only a symptomatic treatment of the underlying proxy objective failure.

I presented deep hierarchical object grouping (DHOG) to address this proxy objective failure. Instead of trying to prevent a neural network from getting stuck in a bad local optima by imposing increasingly destructive image augmentations, I proposed to sequentially expand the knowledge base of the neural network into a hierarchy of solutions (i.e., clusterings) that optimise the proxy objective. By minimising the mutual information between cluster assignments from different compute heads, a neural network can be encouraged to learn diverse clustering solutions. These solutions are arranged in a hierarchy of complexity (from 'simple' to 'complex') by applying compute heads at different depths in the network, each of which yields cluster allocations. Gradient stopping was used to ensure the hierarchical arrangement, such that earlier compute heads were always unaffected by the solutions found by later compute heads. When it was developed DHOG improved performance on the state-of-the-art image deep clustering method while also using less destructive image augmentations.

Latent adversarial debiasing In Chapter 5 I showed how a pernicious form of sample selection bias, called collider bias, can result in training images with entangled features from both causal and extraneous unobserved underlying variables. In such a setup the presence of extraneous variables can result in training images with **spurious signals on which a dependence is easy-to-learn** when compared to a dependence on causal signals. This becomes a problem for learning neural networks for classification when it is easier to compute class predictions from spurious signals. I showed that a standard training objective failed to produce a robust classifier for three datasets from the literature constructed with spurious signals; e.g., MNIST where the background colours are correlated with the class in only the training images but not at test time.

I introduced latent adversarial debiasing (LAD) as a technique to decouple the spurious signals from the class labels in the training data. LAD works by first modelling the training images using a vector-quantised variational autoencoder in order to produce latent representations for the images. A simple classifier is then trained using the latent representations as input. Given an image, the relationship between the spurious signal and the class is then removed using a gradient walk in the latent space that targets high entropy in the simple classifier's predictions, with the vector-quantisation ensuring minimum deviation from the underlying data manifold. The adjusted latent space can then be decoded for this image, resulting in a new image that retains the

relationship between the causal signal and class but for which the spurious signal is not informative of the class. A different neural network trained on the altered images is substantially more robustness at test time.

6.1 Implications and direction for future work

What I have come to learn throughout this work is that there are many, often hidden, examples of proxy objective failure, for which circumvention is often challenging. The notion of a proxy objective is broad enough to encompass many different training paradigms and objectives, and it enables a perspective on representation learning that brings to light the tendency of neural networks to find simple, non-robust, or poorly generalising solutions to the problems users wish them to solve. This justifies a more pragmatic approach to questioning whether existing training objectives are sufficient at preventing non-performant neural networks. Instead of investing more money, compute, or time into training large neural networks, perhaps we should first make sure that the proxy objectives we use do not fail.

I would like to combine the ideas from DHOG and DDTL to create new deep decision forests, where each tree partitions the data in a unique way, and where the forest can be trained to maximise coverage of the space of possible binary tree partitionings. There has also been interesting recent work in comparing distributions [Feydy et al., 2019] that might enable better comparisons between clusters allocations or decision tree leaf probabilities, but this remains to be explored. Using gradient walks to produce neural network-dependent image augmentations is of interest to me. I explored this for a time during this work, with the goal of 'augmentation-free' self-supervised learning, but the degree of success was not sufficient to warrant inclusion in this Thesis. Nonetheless, combining the gradient walk from LAD with self-supervised learning in an online framework may be a particularly interesting direction for future work.

My original plan was to look into layer-wise training of neural networks toward the idea of 'instant neural networks'. In some sense the objectives used in layer-wise learning fail because they are usually intended for end-to-end training, and the features they produce are ill-suited to further downstream processing. Therefore, mitigating proxy objective failure is relevant for layer-wise learning and should still be explored. Finally, the notion of 'exploration' in deep reinforcement learning is intuitively similar to the ideas of finding unique solutions for optimising objectives, similar to what I did for DHOG, which warrants further exploration.

Bibliography

Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459, 2010.

Charu C Aggarwal. Neural networks and deep learning. *Springer*, 10:978–3, 2018.

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. In *Proceedings of the 5th International Conference on Learning Representations*, 2017.

David Alvarez Melis and Tommi Jaakkola. Towards robust interpretability with self-explaining neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Rie Kubota Ando, Tong Zhang, and Peter Bartlett. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(11), 2005.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv e-prints*, 2019.

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. In *Proceedings of the International Conference on Machine Learning*, 2017.

Nicolás Astorga, Pablo Huijse, Pavlos Protopapas, and Pablo Estévez. MPCC: Matching priors and conditionals for clustering. In *Proceedings of the European Conference on Computer Vision*, 2020.

Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. Neural codes for image retrieval. In *Proceedings of the European Conference on Computer Vision*, pages 584–599. Springer, 2014.

Philip Bachman, Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Hyojin Bahng, Sanghyuk Chun, Sangdoo Yun, Jaegul Choo, and Seong Joon Oh. Learning de-biased representations with biased representations. In *Proceedings of the International Conference on Machine Learning*, 2020.

Jiale Bai, Bingbing Ni, Minsi Wang, Zefan Li, Shuo Cheng, Xiaokang Yang, Chuanping Hu, and Wen Gao. Deep progressive hashing for image retrieval. *IEEE Transactions on Multimedia*, 21(12):3178–3193, 2019.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R Devon Hjelm. MINE: mutual information neural estimation. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 531–540. PMLR, 2018.

Emanuel Ben-Baruch, Tal Ridnik, Nadav Zamir, Asaf Noy, Itamar Friedman, Matan Protter, and Lihi Zelnik-Manor. Asymmetric loss for multi-label classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 82–91, 2021.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48, 2009.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013a.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013b.

Alex Beutel, Jilin Chen, Zhe Zhao, and Ed H. Chi. Data decisions and theoretical implications when adversarially learning fair representations. In *Proceedings of Fairness, Accountability, and Transparency in Machine Learning*, 2017.

Ronan Le Bras, Swabha Swayamdipta, Chandra Bhagavatula, Rowan Zellers, Matthew E. Peters, Ashish Sabharwal, and Yejin Choi. Adversarial filters of dataset biases. In *Proceedings of the International Conference on Machine Learning*, 2020.

Yue Cao, Mingsheng Long, Bin Liu, and Jianmin Wang. Deep cauchy hashing for hamming space retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1229–1237, 2018.

Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S Yu. Hashnet: Deep learning to hash by continuation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5608–5617, 2017.

Saul Carliner. *An overview of online learning*. Human Resource Development, 2004.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision*, pages 132–149, 2018.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Lawrence Cayton. Algorithms for manifold learning. Technical report, University of California, 2005.

Soumendu Chakraborty, Satish Kumar Singh, and Pavan Chakraborty. Local gradient hexa pattern: A descriptor for face recognition and retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(1):171–180, 2016.

Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep adaptive image clustering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5879–5887, 2017.

Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11(3), 2010.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the International Conference on Machine Learning*, pages 1597–1607, 2020a.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. In *Advances in Neural Information Processing Systems*, volume 33, 2020b.

Ting Chen, Calvin Luo, and Lala Li. Intriguing properties of contrastive losses. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Yaxiong Chen and Xiaoqiang Lu. Deep discrete hashing with pairwise correlation learning. *Neurocomputing*, 385:111–121, 2020.

Yudong Chen, Zhihui Lai, Yujuan Ding, Kaiyi Lin, and Wai Keung Wong. Deep supervised hashing with anchor graph. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9796–9804, 2019.

Zhixiang Chen, Xin Yuan, Jiwen Lu, Qi Tian, and Jie Zhou. Deep hashing via discrepancy minimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6838–6847, 2018.

Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. NUS-WIDE: a real-world web image database from national university of singapore. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, pages 1–9, 2009.

Richard J Cichelli. Minimal perfect hash functions made simple. *Communications of the ACM*, 23(1):17–19, 1980.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pages 215–223, 2011.

Antonio Criminisi and Jamie Shotton. *Decision forests for computer vision and medical image analysis*. Springer Science & Business Media, 2013.

Ekin Dogus Cubuk, Barret Zoph, Jon Shlens, and Quoc Le. RandAugment: Practical data augmentation with no separate search. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Luke Darlow, Stanisław Jastrzębski, and Amos Storkey. Latent adversarial debiasing: Mitigating collider bias in deep neural networks. *arXiv preprint arXiv:2011.11486*, 2020.

Luke N Darlow, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. CINIC-10 is not ImageNet or CIFAR-10. *arXiv preprint arXiv:1810.03505*, 2018.

Luke Nicholas Darlow and Amos Storkey. What information does a ResNet compress? *arXiv preprint arXiv:2003.06254*, 2020.

Abhijit Das, Antitza Dantcheva, and Francois Bremond. Mitigating bias in gender, age and ethnicity classification: a multi-task convolution neural network approach. In *Proceedings of the European Conference on Computer Vision Workshops*, 2018.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

Thomas Deselaers, Daniel Keysers, and Hermann Ney. Features for image retrieval: an experimental comparison. *Information Retrieval*, 11(2):77–107, 2008.

Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Martin Dietzfelbinger, Anna Karlin, Kurt Mehlhorn, Friedhelm Meyer Auf Der Heide, Hans Rohnert, and Robert E Tarjan. Dynamic perfect hashing: Upper and lower bounds. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 524–531.

Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015.

Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(9):1734–1747, 2015.

Harold Edson Driver and Alfred Louis Kroeber. *Quantitative expression of cultural relationships*, volume 31. Berkeley: University of California Press, 1932.

Shiv Ram Dubey. A decade survey of content based image retrieval using deep learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(5):2687–2704, 2021.

Shiv Ram Dubey, Satish Kumar Singh, and Rajat Kumar Singh. Rotation and illumination invariant interleaved intensity order-based local descriptor. *IEEE Transactions on Image Processing*, 23(12):5323–5333, 2014.

Shiv Ram Dubey, Satish Kumar Singh, and Rajat Kumar Singh. Local wavelet pattern: a new feature descriptor for image retrieval in medical CT databases. *IEEE Transactions on Image Processing*, 24(12):5892–5903, 2015.

Shiv Ram Dubey, Satish Kumar Singh, and Rajat Kumar Singh. Multichannel decoded local binary patterns for content-based image retrieval. *IEEE Transactions on Image Processing*, 25(9):4018–4032, 2016.

J. C. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57, 1973.

Harrison Edwards and Amos J. Storkey. Censoring representations with an adversary. In *Proceedings of International Conference on Learning Representations*, 2016.

Sepehr Eghbali and Ladan Tahvildari. Deep spherical quantization for image search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11690–11699, 2019.

Adar Elad, Doron Haviv, Yochai Blau, and Tomer Michaeli. Direct validation of the information bottleneck principle for deep nets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019.

Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. Exploring the landscape of spatial robustness. In *Proceedings of International Conference on Machine Learning*, pages 1802–1811, 2019.

Linus Ericsson, Henry Gouk, and Timothy M Hospedales. How well do self-supervised models transfer? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5414–5423, 2021.

Mirco Fabbri and Gianluca Moro. Dow jones trading with deep learning: The unreasonable effectiveness of recurrent neural networks. In *Data*, pages 142–153, 2018.

Kuo-Chin Fan and Tsung-Yung Hung. A novel local pattern descriptor – local vector pattern in high-order derivative space for face recognition. *IEEE Transactions on Image Processing*, 23(7):2877–2891, 2014.

Lixin Fan, KamWoh Ng, Ce Ju, Tianyu Zhang, and Chee Seng Chan. Deep polarized network for supervised learning of accurate binary hashing codes. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 825–831, 2020.

Maziar Moradi Fard, Thibaut Thonet, and Eric Gaussier. Deep k-means: Jointly clustering with k-means and learning representations. *Pattern Recognition Letters*, 138: 185–192, 2020.

A. Fawzi, H. Samulowitz, D. Turaga, and P. Frossard. Adaptive data augmentation for image classification. In *Proceedings of IEEE International Conference on Image Processing*, 2016.

Hao Feng, Nian Wang, and Jun Tang. Deep weibull hashing with maximum mean discrepancy quantization for image retrieval. *Neurocomputing*, 464:95–106, 2021a.

Hao Feng, Nian Wang, Jun Tang, Jie Chen, and Feng Chen. Multi-granularity feature learning network for deep hashing. *Neurocomputing*, 423:274–283, 2021b.

Jean Feydy, Thibault Séjourné, François-Xavier Vialard, Shun-ichi Amari, Alain Trouvé, and Gabriel Peyré. Interpolating between optimal transport and mmd using sinkhorn divergences. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, pages 2681–2690. PMLR, 2019.

Chaoyou Fu, Liangchen Song, Xiang Wu, Guoli Wang, and Ran He. Neurons merging layer: Towards progressive redundancy reduction for deep supervised hashing. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 2322–2328, 2019.

Lin-Wei Ge, Jun Zhang, Yi Xia, Peng Chen, Bing Wang, and Chun-Hou Zheng. Deep spatial attention hashing network for image retrieval. *Journal of Visual Communication and Image Representation*, 63:102577, 2019.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *Proceedings of the 7th International Conference on Learning Representations*, 2019.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2:665–673, 2020.

Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5736–5745, 2017.

Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *Proceedings of the International Conference on Learning Representations*, 2018.

Karan Goel, Albert Gu, Yixuan Li, and Christopher Ré. Model patching: Closing the subgroup performance gap with data augmentation. In *Proceedings of the International Conference on Learning Representations*, 2021.

Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12): 2916–2929, 2012.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, volume 27, 2014.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proceedings of the International Conference on Learning Representations*, 2015.

Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus. End-to-end learning of deep visual representations for image retrieval. *International Journal of Computer Vision*, 124(2):237–254, 2017.

Marco Gori and Alberto Tesi. On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(1):76–86, 1992.

Sven Gowal, Chongli Qin, Po-Sen Huang, Taylan Cemgil, Krishnamurthy Dvijotham, Timothy Mann, and Pushmeet Kohli. Achieving robustness in the wild via adversarial mixing with disentangled representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Dan Greene, Michal Parnas, and Frances Yao. Multi-index hashing for information retrieval. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 722–731. IEEE, 1994.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Justin Grimmer and Gary King. General purpose computer-assisted clustering and conceptualization. *Proceedings of the National Academy of Sciences*, 108(7):2643–2650, 2011.

Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *Proceedings of the International Conference on Learning Representations*, 2021.

Philip Haeusser, Johannes Plapp, Vladimir Golkov, Elie Aljalbout, and Daniel Cremers. Associative deep clustering: training a classification network with no labels. In *Proceedings of the German Conference on Pattern Recognition*, pages 18–32. Springer, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Proceedings of the European Conference on Computer Vision*, pages 630–645. Springer, 2016b.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

J. J. Heckman. Sample selection bias as a specification error. *Econometrica*, 47:153–162, 1979.

Thomas M Hehn and Fred A Hamprecht. End-to-end learning of deterministic decision trees. In *proceedings of German conference on pattern recognition*, pages 612–627. Springer, 2018.

Dan Hendrycks and Thomas G. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *Proceedings of International Conference on Learning Representations*, 2019.

Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. AugMix: A simple data processing method to improve robustness and uncertainty. In *Proceedings of the International Conference on Learning Representations*, 2020.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *Proceedings of the International Conference on Learning Representations*, 2019.

Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7132–7141, 2018.

Wenjin Hu, Yukun Chen, Lifang Wu, Ge Shi, and Meng Jian. Boundary-aware hashing for hamming space retrieval. *Applied Sciences*, 12(1):508, 2022.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017.

Syed Sameed Husain and Miroslaw Bober. Improving large-scale image retrieval through robust aggregation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9):1783–1796, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning*, pages 448–456. PMLR, 2015.

Alan Julian Izenman. Linear discriminant analysis. In *Modern multivariate statistical techniques*, pages 237–280. Springer, 2013.

Paul Jaccard. The distribution of the flora in the alpine zone. 1. *New Phytologist*, 11 (2):37–50, 1912.

I Jeena Jacob, KG Srinivasagan, and Kalyanakumar Jayapriya. Local oppugnant color texture pattern for image retrieval system. *Pattern Recognition Letters*, 42:72–78, 2014.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2010.

Hervé Jégou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Pérez, and Cordelia Schmid. Aggregating local image descriptors into compact codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1704–1716, 2011.

Xu Ji, João F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9865–9874, 2019.

Qing-Yuan Jiang and Wu-Jun Li. Asymmetric deep supervised hashing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Qing-Yuan Jiang, Xue Cui, and Wu-Jun Li. Deep discrete supervised hashing. *IEEE Transactions on Image Processing*, 27(12):5996–6009, 2018.

Jason Jo and Yoshua Bengio. Measuring the tendency of CNNs to learn surface statistical regularities. *arXiv e-prints*, art. arXiv:1711.11561, November 2017.

Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

Can Kanbak, Seyed-Mohsen Moosavi-Dezfooli, and P. Frossard. Geometric robustness of deep networks: Analysis and improvement. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

Rong Kang, Yue Cao, Mingsheng Long, Jianmin Wang, and Philip S Yu. Maximum-margin hamming hashing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8252–8261, 2019.

Yoonseop Kang, Saehoon Kim, and Seungjin Choi. Deep learning to hash with multiple representations. In *Proceedings of the 12th International Conference on Data Mining*, pages 930–935. IEEE, 2012.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, 2018.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 18661–18673, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2015.

Benjamin Klein and Lior Wolf. End-to-end supervised product quantization for image search and retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5041–5050, 2019.

Donald Ervin Knuth. *The art of computer programming*, volume 3. Pearson Education, 1997.

Alex Krizhevsky and Geoffrey E Hinton. Using very deep autoencoders for content-based image retrieval. In *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, volume 1, page 2. Citeseer, 2011.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Anders Krogh and John Hertz. A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems*, volume 4, 1991.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3270–3278, 2015.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *Proceedings of the International Conference on Machine Learning*, 2016.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

Jiayong Li, Wing WY Ng, Xing Tian, Sam Kwong, and Hui Wang. Weighted multi-deep ranking supervised hashing for efficient image retrieval. *International Journal of Machine Learning and Cybernetics*, 11(4):883–897, 2020a.

Junnan Li, Pan Zhou, Caiming Xiong, and Steven CH Hoi. Prototypical contrastive learning of unsupervised representations. In *Proceedings of the International Conference on Learning Representations*, 2021.

Qi Li, Zhenan Sun, Ran He, and Tieniu Tan. Deep supervised discrete hashing. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Tianhong Li, Lijie Fan, Yuan Yuan, Hao He, Yonglong Tian, Rogerio Feris, Piotr Indyk, and Dina Katabi. Addressing feature suppression in unsupervised visual representations. *arXiv e-prints*, page arXiv:2012.09962, 2020b.

Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang. Feature learning based deep supervised hashing with pairwise labels. In *Proceedings of the Joint Conference on Artificial Intelligence*, 2016.

Yunqiang Li, Wenjie Pei, Jan van Gemert, et al. Push for quantization: Deep fisher hashing. In *Proceedings of the 30th British Machine Vision Conference*, 2019.

Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast AutoAugment. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Ralph Linsker. Self-organization in a perceptual network. *Computer*, 21(3):105–117, 1988.

Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. Deep supervised hashing for fast image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2064–2072, 2016.

Stuart Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

Wei-Yin Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1:14–23, 2011.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of the International Conference on Learning Representations*, 2019.

David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the 7th IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157. IEEE, 1999.

Jiwen Lu, Venice Erin Liong, and Jie Zhou. Deep hashing for scalable image search. *IEEE Transactions on Image Processing*, 26(5):2352–2367, 2017.

Xiao Luo, Daqing Wu, Zeyu Ma, Chong Chen, Huasong Zhong, Minghua Deng, Jianqiang Huang, and Xian-sheng Hua. CIMON: Towards high-quality hash codes. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 2021.

J MacQueen. Classification and analysis of multivariate observations. In *Berkeley Symposium of Mathematics, Statistics, and Probability*, pages 281–297, 1967.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *Proceedings of International Conference on Learning Representations*, 2018.

Stephane Mallat. Wavelets for a vision. *Proceedings of the IEEE*, 84(4):604–614, 1996.

Julieta Martinez, Shobhit Zakhmi, Holger H Hoos, and James J Little. LSQ++: Lower running time and higher recall in multi-codebook quantization. In *Proceedings of the European Conference on Computer Vision*, pages 491–506, 2018.

R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Mason McGill and Pietro Perona. Deciding how to decide: Dynamic routing in artificial neural networks. In *Proceedings of the International Conference on Machine Learning*, pages 2363–2372. PMLR, 2017.

Matthias Minderer, Olivier Bachem, Neil Houlsby, and Michael Tschannen. Automatic shortcut removal for self-supervised representation learning. In *Proceedings of the International Conference on Machine Learning*, 2020.

Pedro Morgado, Yunsheng Li, Jose Costa Pereira, Mohammad Saberian, and Nuno Vasconcelos. Deep hashing with hash-consistent large margin proxy embeddings. *International Journal of Computer Vision*, 129(2):419–438, 2021.

Subrahmanyam Murala, RP Maheshwari, and R Balasubramanian. Local tetra patterns: a new feature descriptor for content-based image retrieval. *IEEE Transactions on Image Processing*, 21(5):2874–2886, 2012.

Junhyun Nam, Hyuntak Cha, Sungsoo Ahn, Jaeho Lee, and Jinwoo Shin. Learning from failure: De-biasing classifier from biased classifier. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

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

Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

Wing WY Ng, Jiayong Li, Xing Tian, Hui Wang, Sam Kwong, and Jonathan Wallace. Multi-level supervised hashing with deep features for efficient image retrieval. *Neurocomputing*, 399:171–182, 2020.

Mohammad Norouzi, Ali Punjani, and David J Fleet. Fast search in hamming space with multi-index hashing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3108–3115. IEEE, 2012.

Mohammad Norouzi, Maxwell Collins, Matthew A Johnson, David J Fleet, and Pushmeet Kohli. Efficient non-greedy optimization of decision trees. In *Advances in neural information processing systems*, volume 28, 2015.

Timo Ojala, Matti Pietikainen, and David Harwood. Performance evaluation of texture measures with classification based on kullback discrimination of distributions. In *Proceedings of the 12th International Conference on Pattern Recognition*, volume 1, pages 582–585. IEEE, 1994.

Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Haonan Qiu, Chaowei Xiao, Lei Yang, Xinchen Yan, Honglak Lee, and Bo Li. SemanticAdv: Generating adversarial examples via attribute-conditional image editing. In *Proceedings of the European Conference on Computer Vision*, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *Proceedings of the International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

David L Richmond, Dagmar Kainmueller, Michael Y Yang, Eugene W Myers, and Carsten Rother. Mapping auto-context decision forests to deep convnets for semantic segmentation. *arXiv preprint arXiv:1507.07583*, 2015.

Horst Rinne. *The Weibull distribution: a handbook*. Chapman and Hall/CRC, 2008.

Joshua Robinson, Li Sun, Ke Yu, Kayhan Batmanghelich, Stefanie Jegelka, and Suvrit Sra. Can contrastive learning avoid shortcut solutions? In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Lior Rokach and Oded Maimon. Clustering methods. In *Data Mining and Knowledge Discovery Handbook*, pages 321–352. Springer, 2005.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386, 1958.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

Terrence J Sejnowski. The unreasonable effectiveness of deep learning in artificial intelligence. *Proceedings of the National Academy of Sciences*, 117(48):30033–30038, 2020.

Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. Supervised discrete hashing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 37–45, 2015.

Yuming Shen, Jie Qin, Jiaxin Chen, Li Liu, Fan Zhu, and Ziyi Shen. Embarrassingly simple binary representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019.

Guy Shiran and Daphna Weinshall. Multi-modal deep clustering: Unsupervised partitioning of images. In *Proceedings of the 25th International Conference on Pattern Recognition*, pages 4728–4735. IEEE, 2021.

Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

SKLearn. Clustering, 2022. URL `https://scikit-learn.org/stable/modules/clustering.html`.

Malcolm Slaney and Michael Casey. Locality-sensitive hashing for finding nearest neighbors. *IEEE Signal Processing Magazine*, 25(2):128–131, 2008.

Dongjin Song and Dacheng Tao. Biologically inspired feature manifold for scene classification. *IEEE Transactions on Image Processing*, 19(1):174–184, 2009.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Damian Stachura, Christopher Galias, and Konrad Zolna. Leakage-robust classifier via mask-enhanced training. In *The Student Abstract Track of the 34th AAAI Conference on Artificial Intelligence*, 2020.

Amos J. Storkey. When training and test sets are different: Characterizing learning transfer. *Dataset Shift in Machine Learning*, pages 3–8, 2009.

David Stutz, Matthias Hein, and Bernt Schiele. Disentangling adversarial robustness and generalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Shupeng Su, Chao Zhang, Kai Han, and Yonghong Tian. Greedy hash: Towards fast optimization for accurate hash coding in CNN. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 843–852, 2017.

Shaoyan Sun, Wengang Zhou, Qi Tian, and Houqiang Li. Scalable object retrieval with compact image representation from generic object regions. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 12(2):1–21, 2015.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2020.

Grzegorz Swirszcz, Wojciech Marian Czarnecki, and Razvan Pascanu. Local minima in training of neural networks. *arXiv preprint arXiv:1611.06310*, 2016.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Pieter Merkus Lambertus Tammes. On the origin of number and arrangement of the places of exit on the surface of pollen-grains. *Recueil des travaux botaniques néerlandais*, 27(1):1–84, 1930.

Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Yonglong Tian, Olivier J Henaff, and Aäron van den Oord. Divide and contrast: Self-supervised learning from uncurated data. In *Proceedings of the International Conference on Computer Vision*, pages 10063–10074, 2021.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *Proceedings of the IEEE Information Theory Workshop*, pages 1–5. IEEE, 2015.

Antonio Torralba and Aude Oliva. Statistics of natural image categories. *Network: Computation in Neural Systems*, 14(3):391, 2003.

Michael Tschannen, Josip Djolonga, Paul K Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. In *Proceedings of the International Conference on Learning Representations*, 2019.

Aaron Van Den Oord, Oriol Vinyals, and others. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11), 2008.

Laurens Van Der Maaten, Eric Postma, Jaap Van den Herik, et al. Dimensionality reduction: a comparative. *Journal of Machine Learning Research*, 10(66-71):13, 2009.

Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. SCAN: Learning to classify images without labels. In *Proceedings of the European Conference on Computer Vision*, 2020.

Haoqing Wang, Xun Guo, Zhi-Hong Deng, and Yan Lu. Rethinking minimal sufficient representation in contrastive learning. In *Proceedings of the International Conference on Learning Representations*, 2021.

Jianfeng Wang, Jingdong Wang, Jingkuan Song, Xin-Shun Xu, Heng Tao Shen, and Shipeng Li. Optimized cartesian *k*-means. *IEEE Transactions on Knowledge and Data Engineering*, 27(1):180–192, 2014a.

Jianfeng Wang, Jingdong Wang, Jingkuan Song, Xin-Shun Xu, Heng Tao Shen, and Shipeng Li. Optimized cartesian *k*-means. *IEEE Transactions on Knowledge and Data Engineering*, 27(1), 2014b.

Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE Conference on Computer Cision and Pattern Recognition*, pages 1386–1393, 2014c.

Ruikui Wang, Ruiping Wang, Shishi Qiao, Shiguang Shan, and Xilin Chen. Deep position-aware hashing for semantic continuous image retrieval. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2493–2502, 2020.

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *Proceedings of the International Conference on Machine Learning*, pages 9929–9939. PMLR, 2020.

Xiao Wang and Guo-Jun Qi. Contrastive learning with stronger augmentations. *arXiv preprint arXiv:2104.07713*, 2021.

Xiaofang Wang, Yi Shi, and Kris M Kitani. Deep supervised hashing with triplet labels. In *Proceedings of the Asian Conference on Computer Vision*, pages 70–84. Springer, 2016.

Dayan Wu, Zheng Lin, Bo Li, Mingzhen Ye, and Weiping Wang. Deep supervised hashing for multi-label and large-scale image retrieval. In *Proceedings of the ACM International Conference on Multimedia Retrieval*, pages 150–158, 2017.

Dayan Wu, Jing Liu, Bo Li, and Weiping Wang. Deep index-compatible hashing for fast image retrieval. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 1–6. IEEE, 2018a.

Jianlong Wu, Keyu Long, Fei Wang, Chen Qian, Cheng Li, Zhouchen Lin, and Hongbin Zha. Deep comprehensive correlation mining for image clustering. In *Proceedings of the International Conference on Computer Vision*, 2019.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2018b.

Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. Supervised hashing for image retrieval via image representation learning. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, 2014.

Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks. In *Proceedings of the International Conference on Machine Learning*, pages 5393–5402. PMLR, 2018.

Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *Proceedings of the International Conference on Machine Learning*, pages 478–487, 2016.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1492–1500, 2017.

Jiehao Xu, Chengyu Guo, Qingjie Liu, Jie Qin, Yunhong Wang, and Li Liu. DHA: supervised deep learning to hash with an adaptive loss function. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019.

Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005.

Huei-Fang Yang, Kevin Lin, and Chu-Song Chen. Supervised learning of semantics-preserving hash via deep convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(2):437–451, 2017.

Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2016.

Yongxin Yang, Irene Garcia Morillo, and Timothy M Hospedales. Deep neural decision trees. *arXiv preprint arXiv:1806.06988*, 2018.

Xue Ying. An overview of overfitting and its solutions. In *Journal of Physics: Conference Series*, volume 1168. IOP Publishing, 2019.

Xin Yuan, Liangliang Ren, Jiwen Lu, and Jie Zhou. Relaxation-free deep hashing via policy gradient. In *Proceedings of the European Conference on Computer Vision*, pages 134–150, 2018.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Proceedings of the British Machine Vision Conference*, 2016.

Hongjia Zhai, Shenqi Lai, Hanyang Jin, Xueming Qian, and Tao Mei. Deep transfer hashing for image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(2):742–753, 2020.

Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, 2018.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021a.

Ruimao Zhang, Liang Lin, Rui Zhang, Wangmeng Zuo, and Lei Zhang. Bit-scalable deep hashing with regularized similarity learning for image retrieval and person reidentification. *IEEE Transactions on Image Processing*, 24(12):4766–4779, 2015.

Yu Zhang, Peter Tiňo, Aleš Leonardis, and Ke Tang. A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2021b.

Zheng Zhang, Qin Zou, Yuewei Lin, Long Chen, and Song Wang. Improved deep hashing with soft pairwise similarity for multi-label image retrieval. *IEEE Transactions on Multimedia*, 22(2):540–553, 2019.

Ziming Zhang, Yuting Chen, and Venkatesh Saligrama. Efficient training of very deep neural networks for supervised hashing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1487–1495, 2016.

Xiangtao Zheng, Yichao Zhang, and Xiaoqiang Lu. Deep balanced discrete hashing for image retrieval. *Neurocomputing*, 403:224–236, 2020.

Guoqiang Zhong, Hui Xu, Pan Yang, Sijiang Wang, and Junyu Dong. Deep hashing learning networks. In *Proceedings of the International Joint Conference on Neural Networks*, pages 2236–2243. IEEE, 2016.

Wengang Zhou, Houqiang Li, and Qi Tian. Recent advance in content-based image retrieval: A literature survey. *arXiv preprint arXiv:1706.06064*, 2017.

Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. Deep hashing network for efficient similarity retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.