Minimizing Computational Resources for Deep Machine Learning: A Compression and Neural Architecture Search Perspective for Image Classification and Object Detection

POYSER, MATTHEW

Minimizing Computational Resources for Deep Machine Learning: A Compression and Neural Architecture Search Perspective for Image Classification and Object Detection

Matt Poyser

A thesis presented for the degree of

Doctor of Philosophy

Department of Computer Science

Durham University

United Kingdom

November 2, 2023

Abstract

Computational resources represent a significant bottleneck across all current deep learning computer vision approaches. Image and video data storage requirements for training deep neural networks have led to the widespread use of image and video compression, the use of which naturally impacts the performance of neural network architectures during both training and inference. The prevalence of deep neural networks deployed on edge devices necessitates efficient network architecture design, while training neural networks requires significant time and computational resources, despite the acceleration of both hardware and software developments within the field of artificial intelligence (AI). This thesis addresses these challenges in order to minimize computational resource requirements across the entire end-to-end deep learning pipeline. We determine the extent to which data compression impacts neural network architecture performance, and by how much this performance can be recovered by retraining neural networks with compressed data. The thesis then focuses on the *accessibility* of the deployment of neural architecture search (NAS) to facilitate automatic network architecture generation for image classification suited to resource-constrained environments. A combined hard example mining and curriculum learning strategy is developed to minimize the image data processed during a given training epoch within the NAS search phase, without diminishing performance. We demonstrate the capability of the proposed framework across all gradient-based, reinforcement learning, and evolutionary NAS approaches, and a simple but effective method to extend the approach to the prediction-based NAS paradigm. The hard example mining approach within the proposed NAS framework depends upon the effectiveness of an autoencoder to regulate the latent space such that similar images have similar feature embeddings. This thesis conducts a thorough investigation to satisfy this constraint within the context of image classification. Based upon the success of the overall proposed NAS framework, we subsequently extend the approach towards object detection. Despite the resultant multi-label domain presenting a more difficult challenge for hard example mining, we propose an extension to the autoencoder to capture the additional object location information encoded within the training labels. The generation of an implicit attention layer within the autoencoder network sufficiently improves its capability to enforce similar images to have similar embeddings, thus successfully transferring the proposed NAS approach to object detection. Finally, the thesis demonstrates the resilience to compression of the general two-stage NAS approach upon which our proposed NAS framework is based.

Declaration

The work in this thesis is based on research carried out within the Department of Computer Science at Durham University, UK. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all the authors own work unless referenced to the contrary in the text.

Acknowledgements

I would firstly like to give my sincerest gratitude and appreciation to my supervisor Professor Toby Breckon, without whom this thesis would be nothing. He has shown me unlimited kindness throughout the time we have known each other, and provided help and support beyond measure. I could not have asked for a better supervisor.

I am very grateful to my friends and colleagues who have helped me and supported me during my time at Durham, and particular thanks to Brian and Seyma for being wonderful during our collaborations together.

I would also like to say thank you to my Mum, Dad, siblings and niblings, for their undending love and support, even when they had no clue what I was talking about. Thank you from the bottom of my heart for getting me to where I am today. Also thank you to Hedgy, for always being there with a smile on his face.

Above all, I would like to thank the love of my life, Vicky, for absolute everything ever, and supporting me every second. She was always willing to help however she could, even while completing her own PhD. This thesis would not be complete without her love.

Contents

List of Figures

List of Tables

Introduction

Computer vision plays a considerable role within our society, upon which a wide array of public infastructure relies. As such, the development of computer vision applications in a time and resource efficient manner is paramount. Over the past decade, it is commonplace to approach such challenges with the use of neural networks [19]. Indeed, with the resurgence of deep learning [20, 21, 22], high performance can and has been achieved across a multitude of challenge domains, not limited to computer vision. Generally, these networks require time in the order of hours or even days to train, for instance state of the art image classification networks (ResNet [18, 23], ConvNeXt [24]), general adversarial networks (GAN [25], StyleGAN [26]), and transformers (ViT [27], DETR [28]). Furthermore, large datasets, upwards of thousands of images, are required for training to achieve convergence [29, 30]. To this end, we dedicate this thesis towards determining the best methods available to minimize deployment challenges associated with training such applications. The first part of the thesis approaches this challenge by levying image compression to reduce data storage requirements such that perfomance of neural networks are uninhibited, without introducing additional complications (Chapter 3). We then focus on processes by which lightweight models can be generated, trained, and deployed within resource constrained environments (Chapter 4 and 5). Specifically, we consider Neural Architecture Search and its accessibility and feasibility given time constraints. Finally, we consider both the union of compression and the two-stage NAS paradigm, drawing parallels to the findings in Chapter 3 where possible (Chapter 6).

1.1 Motivation

With the rise of Convolutional Neural Networks (CNN [31, 32]), the world of machine learning for computer vision has been transformed over the past few decades. Indeed, automated deep learning approaches are rapidly approaching human performance across a multitude of application domains, not limited to image classification, object detection and tracking, pixel-wise image segmentation, and human action recognition [33]. Nevertheless, the data-driven training approach of such systems necessitates several orders of magnitude

of data. For supervised computer vision, this involves images labeled with respect to the vision task at hand.

Consequent to the evolution of big data [34], collection of or immediate access to public data has never been easier. However, the data storage requirements that follow such data acquisition is a constant battle. Commonplace image or video collections, for instance CCTV operations, UAV footage, facial recognition databases, and video sharing websites, depend upon tens of thousands of images or videos. One of the most wide-spread solutions to address this problem is to utilize image and video compression algorithms such as JPEG [35] or MPEG / H.264 [36, 37]. By minimizing the *imperceptible* information within an image, its storage requirements can be reduced. Fig. 1 illustrates the perceptible effect of heavy JPEG compression, introducing 'blocky' compression artifacts when quantized pixel information is decoded. While the differences between compressed and uncompressed data is often imperceptible to the human eye, especially at low compression rates, the neural networks employed within deep learning approaches operate at pixel level granularity and are thus negatively impacted by image compression. Network performance deterioration manifests not only directly in terms of accuracy, but also in certain application domains by introducing racial bias [38, 39, 40, 41]. As such, methods to circumvent this performance degredation are necessary.

Furthermore, modern deep-learning approaches for computer vision tasks are becoming increasingly complex to deal with the challenge at hand for two reasons. Firstly, the environment within which they are employed is becoming increasingly difficult; occlusion is more prevalent as scenes become more cluttered; lighting is more challenging, and thus the architectures required to effectively process a given image are more complex. Secondly, the task itself is more difficult. The question being asked has evolved from *what* is in the image (Image Classification) to *where* is it in the image (Object Detection), to even what *will* it do (Human Action Recognition / Tracking). Furthermore, we must ask these same questions while solving additional problems such as multi-view correspondence or scene reconstruction. Unsurpisingly, the complexity of deployed architectures have evolved to address these issues, whether that is the depth of the network or the mathematical operations in a given network layer, etc. In turn, power and storage requirements to run the model, and the instability during training, and subsequently the training duration, have also increased.

Consequently, we might seek to generate relatively simple networks compared to existing state of the art methods without loss in performance. This is especially important within public infrastructure, where it is infeasible to replace existing hardware, and thus constraints on resource requirements are prevalent. Considerable attention has been paid to this notion, for instance hand-crafting efficient networks (SqueezeNet [42] and the MobileNet family [43, 44, 45]). Similarly, pruning [46, 47], network quantisation [48, 49], and knowledge distillation from large teacher networks to smaller networks [50, 51, 52] are well researched fields that serve to reduce the size of a given network (usually) after it has been trained.

With the generation of networks that adhere to specific constraints in mind, we turn to neural archi-

(a) before (b) after

(c) before (d) after

Figure 1: Example of images before and after heavy JPEG compression (10% of original size), taken from the UCF101 [1] dataset

tecture search (NAS) to generate network architectures in an automated fashion. Imposing an architecture complexity constraint for instance is a natural addition to the NAS search procedure, yielding network architectures that are optimal with respect to both the task (classification/detection), as well as resource limitations. However, NAS approaches themselves are notoriously expensive in terms of time and memory resource complexity. Moreover, existing NAS literature has a predominant focus (not unreasonably) on the architectures that are generated. The work presented in this thesis addresses the aforementioned resource complexity whilst simultaneously identifying a promising new direction for NAS research. We formulate an overall understanding of deep learning approaches through both compression and NAS *accessibility* within the context of resource minimization.

1.2 Thesis Contributions

The main contributions of the thesis are as follows:

- An up-to-date and systematic collation of contemporary Neural Architecture Search approaches within their respective challenge domains, adopting consistent and accessible terminology (Chapter 2)
- A comprehensive evaluation of different CNN architecture performance (in terms of both overall accuracy and racial bias) with respect to JPEG/H.264 compression algorithms (Chapter 3).
- A novel *efficient* train/test cycle pipeline incorporating curriculum learning and hard example mining, through feature similarity representation in latent space as proxy for image hardness, optimized for CNN NAS evolutionary / reinforcement learning / gradient-based frameworks for image classification (Chapter 4).
- A novel hard example mining approach for data selection within prediction-based image classification NAS frameworks (Chapter 4).
- A gradient-based NAS framework extending prior work towards multi-label image classification and object detection challenges, capitalizing on object detection labels to improve our hard example mining approach via an implicit attention layer (Chapter 5).
- The union of the commonplace NAS (DARTS [53]) and Faster-RCNN [11] Object Detection approaches yielding efficient networks that illustrate a promising direction for future research (Chapter 5).
- An extensive evaluation of our proposed NAS approach under compression (Chapter 6).

1.3 Publications

The work contained within this thesis has been previously published in the following peer-review publications by the author, and is used in the chapters as indicated below:

- **On the impact of lossy image and video compression on the performance of deep convolutional neural network architectures**, M. Poyser, A. Atapour-Abarghouei, and T. Breckon, in 25th International Conference on Pattern Recognition (ICPR2020). IEEE, September 2020. (Contributing to Chapter 3)
- **Does lossy image compression affect racial bias within face recognition?**, Seyma Yucer, Matt Poyser, Noura Al Moubayed, and Toby P. Breckon in International Joint Conference on Biometrics, October 2022. (Contributing to Chapter 3)

1.4 Scope and Structure

With deployment challenges divided into two key approaches, it follows to divide this thesis similarly. Chapter 2 firstly reviews the existing literature for the impacts of image compression on neural networks and racial bias, and secondly identifies and evaluates existing NAS strategies, which is split into the dominant scene-understanding challenges within the field (image classification, object detection, and image segmentation). Further, a review on curriculum learning and hard example mining techniques is conducted to identify potential solutions for deployment within a NAS framework.

Chapter 3 explores the image compression domain. The chapter details the extent of impacts of image compression upon several neural network architectures, and hypothesises the cause of their respective differences in performance. By retraining networks with compressed imagery, it is evident by how much the problem can be circumvented across each architecture type. Additional racial bias introduced within compression is then discussed, and how retraining networks is insufficient to ameliorate it. Instead, omission of chroma subsampling within compression is proposed and evaluated.

Chapter 4 conducts a primary investigation into the use of NAS for image classification problems, and proposes a novel training pipeline for evolutionary, reinforcement learning, and gradient-based NAS approaches that incorporates curriculum learning and hard example mining to minimize training costs in terms of both time and power consumption. Further, it is shown that a similar hard example mining approach is sufficient to streamline prediction-based NAS.

Chapter 5 extends the algorithm presented in chapter 4 to multi-label and object detection challenges. It is evident that the clustering techniques adopted by the previous method are well suited to extension towards object detection through additional encoding techniques that act as an efficient attention layer. A further object detection architecture generation strategy is then introduced that is best able to harnass the benefits of NAS and our existing curriculum learning and hard example mining training algorithm.

Chapter 6 investigates the impact of compression upon the NAS methodology illustrated in the previous chapters, drawing parallels with the findings from Chapter 3 where possible.

Literature Review

This section introduces prior literature related to either (i) compression, (ii) Neural Architecture Search, or (iii) curriculum learning and hard example mining with respect to application of CNN towards efficient real-world deployment.

2.1 Compression

While the impact of compression algorithms within deployed infrastructure is well-considered, we constrain our study to its impacts on the application of CNN. In this respect, prior work is limited in scope and diversity [54, 55, 56, 57]. Dodge et al. [54] analyze the performance of now seminal CNN image classification architectures (AlexNet [58], VGG [59] and InceptionV1 [60]) performance under JPEG [35] compression and other distortion methods. They find that these architectures are resilient to compression artifacts (performance drops only for JPEG quality < 10) and contrast changes, but under-perform when noise and blur are introduced.

Similarly, Zanjani et al. [61] consider the impact of JPEG 2000 compression [62] on CNN, and whether retraining the network on lossy compressed imagery would afford better resultant model performance. They identify similar performance from the retrained model on higher quality images but are able to achieve up to as much as 59% performance increase on low quality images.

Rather than image compression, Yeo et al. [55] compare different block sizes and group-of-pictures (GOP) sizes within MPEG [36] compression against Human Action Recognition (HAR). They determine that both smaller blocks and smaller groups increase performance. Furthermore, B frames introduce propagation errors in computing block texture, and should be avoided within the compression process. Tom et al. [63] add that there is a near-linear relationship between HAR performance and the number of motion vectors (MV) corrupted within H.264[37] video data, with performance levelling off when 75% of MV are corrupted. Klare and Burge [56], however, demonstrate that there is a non-linear relationship between face recognition performance and bit rate within H.264 video data, with sudden performance degradation around 128kbps

(CRF). These contrasting results therefore demonstrate the need to investigate compression quality across multiple challenge domains, whose respective model architectures might have different resilience to lossy compression artifacts.

Multiple authors have developed impressive architectures trained on compressed data, indicating both the potential and need for in-depth investigation within the compressed domain. Zhuang and Lai [57] demonstrate that acceptable face detection performance can be obtained from H.264 video data, while Wang and Chang [64] use the DCT coefficients from MPEG compression [36] to directly locate face regions. The same authors even achieve accurate face tracking results in [65], still within the compressed video domain.

Prior research additionally examines the impact of image quality on overall facial recognition performance [38, 39, 40], but there is little invesigation into either its effect on racial bias (a known factor to consider) or specifically the impact of compression. Hernandez-Ortega et al. propose FaceQnet [66], optimized for face biometrics, to analyse the impact of compression and other quality factors present within an image, and thus evaluate image quality. Majumdar et al. [41] identify bias towards specific racial and gender subgroups stemming from image distortion, that influences network attention towards non-discriminative image regions. However, they overlook the impact of compression on racial bias.

2.1.1 Information Theory

When reformulating compression within the context of information theory, it is prudent to consider the seminal work of Shannon et al. [67, 68], which presents the communication of a message between two sources. The amount of *information* within that message can be measured in isolation from the semantic value of the message. Using this measurement of information, a theoretical minimum bound can be established concerning how many bits (code rate) are required to encode the message such that it can be perfectly reconstructed with negliglible loss in information (source coding theorem) and in the presence of noise (noisy channel coding theorem). Arithmetic coding is a form of entropy coding, which approaches the lower bound declared by Shannon's source coding theorem [69]. Huffman coding, used within JPEG and MPEG image compression algorithms approximates arithmetic coding with an efficient algorithmic formulation [70].

Formally, let us consider a message as a stream of independent and identically distributed (i.i.d) random variables. The entropy $H(X)$ of a variable X which takes values in the alphabet $\mathcal{X}$ is:

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x). \tag{1}$$

This result is commonly interpreted as the number of bits required to encode X, and the coding portion of image compression algorithms directly follows on from it. This result can be extended to two random

variables X and Y. Conditional entropy $H(Y|X)$ can be interpreted as the amount of information needed to describe Y if X is known:

$$H(Y|X) = -\sum_{x\in\mathcal{X}} p(x) \sum_{y\in\mathcal{Y}} p(y|x) \log p(y|x). \tag{2}$$

We may use the mutual information

$$I(X;Y) = H(Y) - H(Y|X) \tag{3}$$

to consider how far knowledge of X may reduce the uncertainty (information) about Y.

These findings have many practical use cases within contemporary deep machine learning tasks. For instance, VAE [15, 71] objective function uses kullbeck-leibler divergence, upon which mutual information $I(X;Y)$ is formulated, to minimize information loss within feature representations. Alternatively, plagiarism detection can be considered a reformulation of shared information [72], and classification networks in general can be considered optimising towards features with high information gain (i.e. mutual information) [73, 74]. Statistical dependencies and mutual information between variables or images can inform image matching and registration [75], and information theoretic approaches to optimal segregation and minimum description length can drive image segmentation [75, 76].

Building upon this, the nervous system is comprised of biological neurons, which act as an information channel between different parts of the body. During this communication, input signals may be lost due to noise interference, and it is therefore biologically efficient to send information in spikes rather than as a continuous signal [77], as the amplitude of a spiked signal does not decay as quickly. The two parts of the body are connected by a series of neurons, and in this context, the target alphabet for the neuron channel can still be represented as a 0 or a 1. The message must cross gaps between neurons (synapses), and if enough voltage is received by the postsynaptic neuron, the neuron will 'fire' and the resulting 1 signal will be propogated to the next information channel. Calculating the amount of information conveyed by a neuron (neural coding capacity) — the maximum output entropy — reveals the reliability of stimulus-response functions [78].

Delving deeper into biological models, an understanding of the human retina may be used to inform information theory within compression algorithms. The human eye signficantly adapts to luminance in order to function well at both high and low light levels. Photoreceptor cells (rods and cones) directly receive sensory image information, while ganglion cells communicate this information from these photoreceptor cells onwards to the rest of the body[1]. The functionality of a ganglion cell can be reformulated as a convolution

[1]For simplicity we ignore the intermediary bipolar cells that transmit information between the photoreceptor and ganglion

layer. Let us consider the convolution operator $\circledast$. The convolutional layer (photoreceptor cell) z is simply:

$$z = w \circledast x, \tag{4}$$

where x is the weights (connection strength between photorceptors and the ganglion cell [77]) and

$$x = s + \xi, \tag{5}$$

where ξ is the photoreceptor cell noise, encapsulating variations in luminance.

Stone [77] shows that a ganglion cell maximises the information in its output and emphasises that ganglion cells are an efficient communication channel. A ganglion cell receptive field is an image filter; some spatial frequencies affect it more than others (controlled by the weights, i.e. a convolution layer). Under high luminance conditions, the receptive field acts as a band-pass filter, while under low luminance conditions the receptive field acts as low-pass filter. When we apply the implications of this adaptation to information theory and its practical uses, we may inform the impact of luminance within the JPEG algorithm on deep learning based models[2]. It is possible that the work of Stone [77] and others informed the design of the JPEG algorithm to keep luminance information.

Informed by this information theoretic perspective, a given deep learning based model can learn to maximise information, searching for features with high information gain such that it is invariant to luminance, and can retain high performance rates despite extreme lighting conditions. At first glance, this appears to be an interesting result, as the JPEG compression algorithm explicitly transforms an image to the YUV space, based on human perceptions of light and colour. As is explained in more detail in Chapter 3 however, luminance information is not explicitly subsampled within the compression algorithm. Correlating the well-established findings of Stone with our work does not present an immediate advantage, as Stone offers no explicit insight into the capacity of a given machine learning based model to learn in the presence of discarded chroma information, nor in fact how reduced chroma information impacts human perception.

Examining the literature together, there are two key questions that must be answered before we deploy compression strategies. We must firstly consider by *how much* can data be compressed? Secondly, how can we incorporate compression without introducing racial bias?

cells.

[2]Noting that "information-theoretic models are not designed to mimic physiological receptive fields; their only objective is to maximise information throughput." [77]

2.2 Curriculum Learning and Hard Example Mining

Here we consider various curriculum learning and hard example mining strategies levied for data selection within neural network optimization. The process by which we present data to the deep learning model is paramount; it is not uncommon for models to be trained for thousands of epochs, and thus any superfluous data within the dataset will have a magnified negative impact on training speed. To address this, hard example mining [79], attempts to identify hard images within the global dataset. By only considering images that make a significant contribution to training (i.e. model optimisation), usually those that considerably reduce loss in any given epoch, we can not only (a) sample from a minimal dataset and therefore minimize duration of a training epoch, but (b) reduce the number of iterations required for model convergence, as the contribution of each image sample is maximised in every iteration.

Similarly, the images sampled by the model in any given training iteration can be controlled via curriculum learning [80] and self-paced learning. Contrary to hard example mining, where commonly only a subset of the global dataset will be considered during the entire training process, curriculum learning or self-paced learning enforces the initial iterations to sample one fraction of the global dataset, and subsequent iterations to sample from different fractions, until the entire global dataset is considered. Generally, curriculum learning introduces harder images (pre-defined by prior knowledge) as training progresses, while self-paced learning determines the current model performance as feedback to the controller, to determine which images to sample next.

Graves et al. [81] posit the need for a surrogate measure of learning progress to inform the curriculum of when to swap data, rather than model accuracy. They suggest two measures; loss, which they further differentiate into prediction gain (instantaneous loss for a sample), and gradient prediction gain (which measures the magnitude of the gradient descent vector) among many others; and complexity gain, also further split into various measures that consider how the model complexity increases over time, such as gradient variational complexity (derived from the direction of gradient descent). They identify prediction gain to present the most efficient method of informing learning progress when the model is learning to maximise log likelihood, and gradient variational complexity for variational inference training.

Hachoen and Weinshall [82] suggest instead to use a transfer scoring function to generate the curriculum. The scoring function ranks images within the dataset by difficulty through testing either the same model (pre-trained without curriculum learning), or a different model. Harder images are introduced to the model over time. Weinshall et al. [83] further evolve this process, to consider image difficulty in relation to task difficulty (e.g. fine detail differentiation is harder than coarse detail differentiation, which can for instance be trivially approximated with hierarchical datasets). Shrivastava et al. [84] on the other hand in their

hard example mining paper, rank the images in order of difficulty at train time, to dynamically generate a mini-curriculum at each iteration.

Kumar et al. [85], in their work on self paced learning, instead monitor image difficulty as either the negative log-likelihood for expectation-maximisation [86] or the upper bound on risk for latent structural support vector machines. Jiang et al. [87] incorporate both self paced learning and curriculum learning into a single framework. That is, the curriculum is pre-defined by some expert, but takes into account the feedback from the model (the learner) when selecting which images to propose to the network during training.

Finally, Matiisen et al. [88] introduce "mastery" into the curriculum learning framework. Mastery can simply be reaching an accuracy threshold, identified by prior expert knowledge. The model is presented with images from a global dataset, but with more probability of sampling images from the current curriculum fragment. As the model masters said fragment, the probability of sampling these images decreases, while probability of sampling the next curriculum fragment increases.

If we consider these studies altogether, it is evident that curriculum learning and hard example mining both greatly benefit the deep learning optimisation process, and the combination of the two more-so. We therefore uniquely propose to employ such methods within NAS, specifically, levying *mastery* from [88] in tandem with our own hard example mining approach reminiscent of the *'instructor-student collaborative'* learning paradigm [87]. Moreover, the work of Cazenavette et al. [50] build upon well explored dataset distillation techniques [89, 51, 90, 91]. By optimizing the $l2$ loss of the *parameters* of a network trained on only 50 images per class, compared to optimal network parameters (i.e. parameters induced by training with 5000 images per class), they are able to achieve reasonable performance (71.5% on CIFAR-10). On this basis, we can deduce that training on a fraction of images yields a promising direction of research, to which our method pertains without such loss in performance.

2.3 Neural Architecture Search

Recent acceleration within the deep learning domain [22] naturally follows the increased availability of public datasets that stems from the emergence of big data. Unsurprisingly, the complexity of the proposed network architectures is also increasing. As such, manually searching through this architecture space is less and less feasible, and we must rely on domain expertise to identify suitable networks for a given application. With this, Neural Architecture Search (NAS) has emerged, which automatically traverses the architecture search space for a given task, and generates models that are competitive alongside hand-crafted state-of-the-art architectures.

Recent NAS capability within the image classification domain is demonstrably powerful [92], with gener-

ated convolutional neural network (CNN) models achieving accuracies of 97.57% (CIFAR-10 [93]) and 76.2% (ImageNet [29]), comparable to leading image classification performance [24]. However, there is relatively little development in the way of NAS outside of pure CNN generation (e.g. transformer [94] and generative adversarial networks [25]), for which hand-crafted network architectures perform so well. Similarly, within the computer vision domain, consideration of NAS beyond image classification is under-developed; NAS for object detection and image segmentation architectures for example, receive less interest than their hand-crafted counterparts. To this end, we present a comprehensive review over recent NAS advancements, to best facilitate further insight and research in this area. We build upon existing, although now dated surveys [95, 96, 97], that fail to consider NAS for computer vision outside of CNN image classification.

Furthermore, as NAS rapidly evolves within several distinct domains, ambiguities and inconsistencies have arisen in several methodological descriptions. Consequently, it is increasingly difficult for incoming researchers to the NAS domain to meaningfully engage with the field, and clearly explain any proposed methodology with reasonable reproducibility. With this shortcoming in mind, we adopt a common terminology (following that of TuNAS [98]), to improve understanding and reproducibility of future NAS research. On this basis, our comprehensive review presents the following aspects:

- a systematic review, that to our knowledge is the first comprehensive NAS survey, of image classification, object detection, and image segmentation domains to date.
- a novel overview and taxonomy that for the first time uses consistent terminology over *all* contemporary NAS literature, resolving previous ambiguities and inconsistencies emanating from the original NAS works.
- analysis upon the NAS literature offering insights into promising future research directions.

2.3.1 Review Organization

We first divide this NAS review by computer vision task into image classification, object detection and image segmentation NAS methodologies. Subsequently, image classification is subdivided by NAS strategy into three key sub-areas: weight-sharing, gradient-based, and prediction-based, each of which correspond to a NAS search speed-up strategy. In all cases, we prioritize explaining how a given NAS method fits into the evolution of the NAS domain, while maintaining consistent terminology, to the best advantage of incoming researchers.

2.4 Neural Architecture Search: A Quick Overview

Rather than conventional hand-crafting of a neural network architecture, wherein layer operations are hand-selected and defined at each layer, NAS seeks to automatically generate the network architecture best suited to a given task, given a set of available operations. An overview of the general NAS process is illustrated in Fig. 2, which can be split into two key stages. First, the *search phase* involves traversing all architectures within the *search space*. Once the top performing architecture (or top-k architectures) is identified (termed *final searched architecture(s)*), it is retrained from scratch (*evaluation phase*).

With the rise of NAS, a multitude of recent literature has addressed the scalability challenge which occurs due to the resultant large search space of all potential neural network architectures and their respective training costs. The seminal work of Zoph et al. [99] demonstrates the capability of recurrent neural networks (RNN) with reinforcement learning to generate network architectures automatically, with their network architecture outperforming existing hand-crafted state-of-the-art network architectures both in accuracy and speed for both image classification and language modelling tasks.

Since this influential paper [99], interest and research in NAS has accelerated [95]. Reinforcement learning [100, 101, 102, 103] methods, as well as evolutionary [104, 105, 106, 107] approaches have since been developed. Notably, MnasNet [102] adopts a reinforcement learning search strategy for both image classification and object detection, incentivised towards minimizing inference latency. Unmistakeably however, the training cost incurred by NAS techniques remains their foremost problem. More recently, and with more success in this regard, gradient and predictor based approaches have been developed [53, 108, 109, 110], often in conjunction with weight-sharing techniques to improve convergence rate [111, 112, 113], notably by eliminating the need to train each architecture in the search space separately.

As such, we limit the literature covered by this survey to more recent NAS solutions for image scene analysis, where the training cost falls within a reasonable computational time limit. Moreover, several NAS *architecture* datasets exist that facilitate validation of the NAS framework performance, rather than their generated networks. Among these, NAS-Bench-101 [114] is notable, containing 5 million trained and evaluated models. NAS-Bench-201 [115] and NATS-Bench [116] are also available for evaluating architectures size and topology, with fixed architecture search space but more diagnostic information compared to NAS-Bench-101. Tables 1/2 provide a fair comparison where possible across common NAS application domains. In all cases, however, it is important to adhere to best practices when producing a NAS pipeline [117]. Radosavovic et al. [118] demonstrate that the manner in which the search space is constructed is critical.

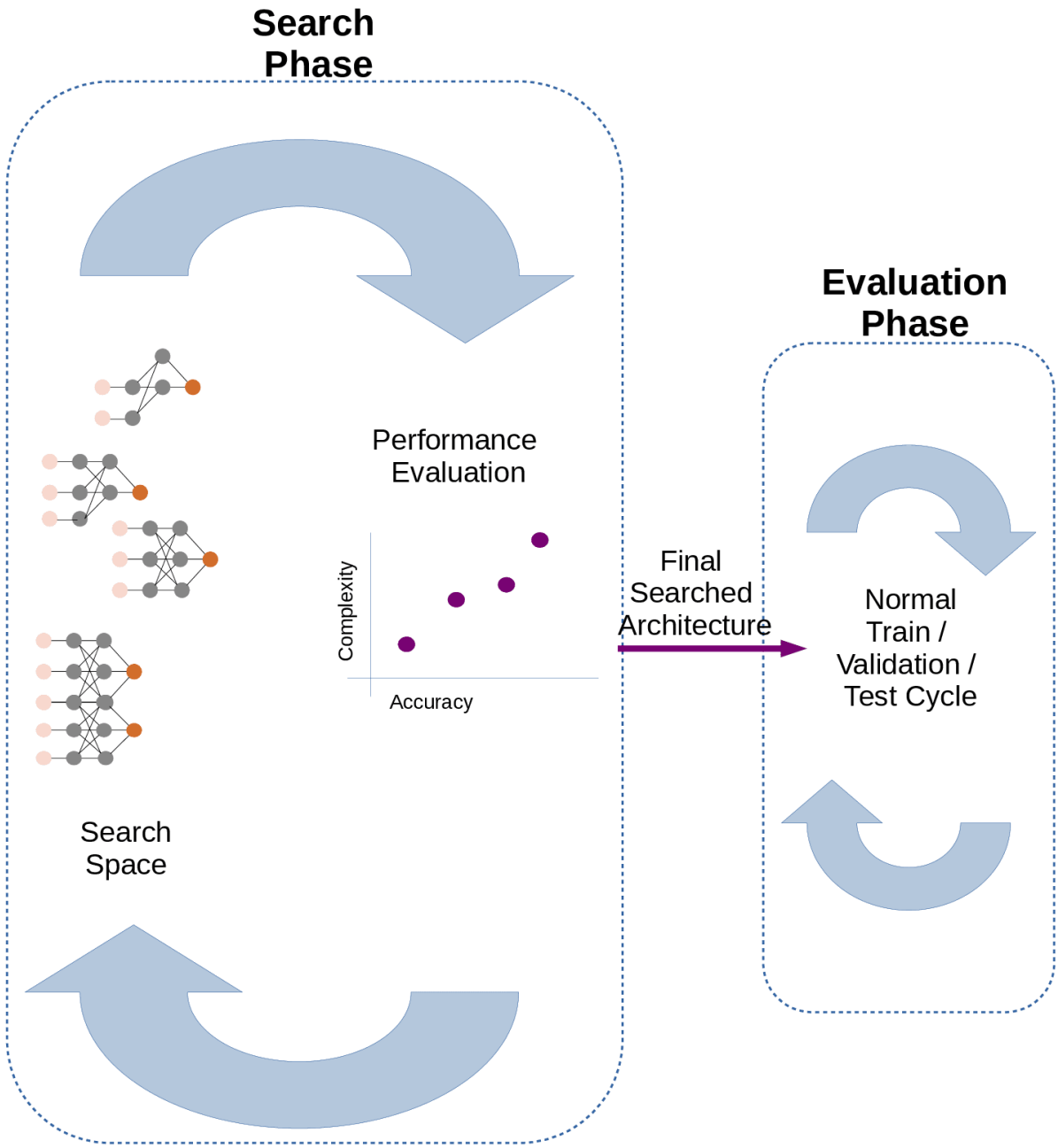


Figure 2: An overview visualization of the NAS process consistent across reinforcement learning, gradient-based and prediction-based approaches

2.5 Image Classification

Image classification is the primary challenge domain in which NAS operates within the computer vision field and for which existing literature is most comprehensive. To provide an overview of these approaches, we first introduce the general weight-sharing concept [111], and subsequently the gradient-based paradigm which employs weight-sharing in a highly efficient manner [53]. Finally, we introduce the prediction NAS paradigm [119], which to some extent circumvents training iterations (and by extension, the need for weight-sharing) entirely.

2.5.1 Weight-Sharing

Weight-sharing approaches, first proposed in ENAS [111], reduce training time through transfer learning of weights learnt for previously sampled architectures. In general, via the use of this technique, only a single network needs to be trained to convergence. Commonly, across all weight-sharing NAS approaches, this single network represents the *entire* architecture search space, and is referred to as a "super-network." Subsequent sampled network architectures during NAS search phase thereafter inherit initialisation weights from this super-network. They require few or zero training epochs before their performance is sufficiently evaluated and ranked. As such, the total search phase cost of NAS is drastically minimized. Due to the fact that only the super-network is trained to convergence during the NAS search phase, we refer to this approach as a *one-shot* method [3]. In general, one-shot methods aim to rank architecture performance using their shared weights [120] relative to *each other* rather than their absolute performance. Once the highest-ranked architecture is determined during the search phase, it is retrained and then fully optimised for the given task.

Here it is helpful to interpret a neural network as a directed acyclic graph (DAG). Each layer of the network can be considered a node, and the possible operations within a layer (convolution, max-pooling, etc.) are edges between nodes (Fig. 3a).

In their influential paper [111], ENAS first contructs a single DAG to represent the entire search space. First they fix the weights of the NAS controller (which determines which nodes and operations are sampled from the architecture search space). Sampling individual cells (child architectures) from the DAG (super-network), they update the weights of a given child network architecture, and thus the super-network after processing one minibatch, until one entire pass of the dataset has been completed. Next, they again sample child network architectures, but this time their weights are fixed (using transfer learning from the

[3] "Weight-sharing" is a broad term to denote how different architectures considered by a given NAS methodology do not use independent weights. "One-shot" (equivalently "super-network" under our terminology) methods necessarily employ weight-sharing but not vice-versa.

previously learned weights) and instead train the controller using reinforcement learning. They proceed to alternate training the child network architectures and controller for several iterations, at which point the best-performing model is sampled, initalised with random weights, and retrained from scratch without the use of transfer learning weights.

The subsequent development of SNAS [112] replaces the Markov Decision Process assumption implicitly adopted by ENAS with a differentiable reward function, positing that this change ameliorates delayed reward. Consequently, structural decisions to the architecture that arise from reinforcement learning are more appropriately rewarded, and thus architecture search hence becomes more efficient.

Following the success of weight-sharing in [111], CAS [121] develops a NAS learning paradigm whereby the cell structure evolves when trained on new datasets and domains, without loss in performance on the previous dataset. They introduce constraints on the learned weights such that the 'knowledge' is projected in an orthogonal direction, and thus does not affect prior knowledge related to that of the previous dataset/domain. They further extend their work to generate a network architecture better generalizable for multi-task learning.

CNAS [113] employs curriculum learning within the NAS search phase. The search space is divided into a series of smaller architecture spaces, where the number of searched operations are gradually increased. Rather than being immediately difficult and unwieldy, as with conventional one-shot methods [111], satisfying the objective function within reinforcement learning becomes progressively more difficult as the search space widens. Overall difficulty is thus more tractable.

ProxylessNAS [122] demonstrates that subsampling only a single path through the DAG at each iteration is effective. Binary gates are introduced (in a differentiable manner) to simulate a mask for a given activation path during a given training iteration. Consequently, training is more stable and can be performed directly on large datasets such as ImageNet. Previous methods were only able to achieve meaningful performance on such datasets by first training on a smaller, proxy dataset. Two variants, ProxylessNAS-R and ProxylessNas-G are presented, which use reinforcement learning and gradient-based methods to traverse the architecture search space respectively. Similarly, NASP [123] use an approach derived from the proximal algorithm [124] to enforce that only one operation is updated with each iteration, drastically improving convergence rate.

TuNAS [98] acknowledges that only if the weights are sufficiently trained will the promising sections of the search space be accurately identified. To help ensure this, the authors build upon [111, 122] and propose two methods, firstly: 'filter warm-up', whereby smaller filter sizes can be simulated by randomly masking out tensors from a bigger filter size output, thus reducing the need to train architectures that only differ in filter size. Secondly, 'operation warm-up', that can improve performance when we must choose a single operation from our operation search space. Here, we instead sample p possible operations where p is linearly decreased from 100% (all operations) to 0 (our controller is unaffected by warm-up) during the first 25% of epochs in

the search phase. Further, the authors propose a novel reward function, as well as more aggressive weight sharing. These, in-tandem with the warm-up methods above, enable i) larger search space and ii) more accurate network evaluation while still avoiding the need to train each sampled child network architecture from scratch as per other weight-sharing methods.

BigNAS [125] further considers the two-stage NAS process, focusing on training the super-network in such as manner that the accuracy of a given sampled sub-network architecture with inherited shared weights performs well with respect to its absolute performance were it retrained from scratch. To achieve this, five key methods are introduced to the conventional super-network training process, not limited to regularization and novel initialization strategy. Throughout this review we refer to this potential difference in *inherited* and *absolute* performance as the *optimization-gap* (see Section 2.5.2.1). With a sufficiently trained super-network, BigNAS demonstrates that a simple grid-search strategy is then enough to traverse the architecture search space efficiently, where network architectures can be sampled under memory or latency constraints.

AttentiveNas [126] improves upon BigNAS search strategy in a Pareto-aware fashion, where the best Pareto architectures are those that achieve better accuracy than every other architecture in the search space with the same or less computational consumption (and the worst Pareto architectures are dominated in performance by all other architectures with the same computational cost). Sampled candidate network architectures are only trained if they lie on the best or worst Pareto front. With this strategy, it is trivial to impose a computational limit on the generated network architectures.

Stage-Wise NAS [127] acknowledges that architectures can be divided into different 'stages.' The importance of a stage can then be determined, with fewer layers attributed to stages of lower importance. During the search phase, the depth of the network architecture is progressively increased. Weights are transferred from a pre-trained ResNet architecture that is necessarily of larger depth than the network architecture in a given training iteration. As such, the authors were able to reduce the volume of architectures searched by one order of magnitude, while retaining state-of-the-art performance on CIFAR-10.

ANASOD [128] reduces the architecture search space without loss in performance via approximate operation distribution encoding. On the basis that there is little difference in performance between architectures with only slight differences, the architectures sharing a distribution of operations map to the same encoding. Adopting conventional NAS methods (indeed ANASOD can be deployed upon most existing NAS strategies) that search over these distribution encodings rather than the entire operation search space makes NAS optimisation more tractable.

Converse to existing NAS solutions, GLiT [129] employs a one-shot NAS approach to optimize *transformer* architecture for image classification. Building upon a Multi-Head Attention (MHA) block as the basis for the search space, a locality module is further introduced such that each searched MHA cell has a varying

distribution of convolution-based locality modules (capturing local information within an image) and self-attention modules (capturing global information within an image). Adopting SPOS [130], GLiT divides the search space into disjointly searching for a) optimal distribution of local and global sub-modules and b) detailed architecture of modules given optimal distribution in (a). Resultantly, the vast search space encompassing transformer network architectures can be efficiently searched without compromising memory requirements.

NEAS [131] adopts a similar strategy whereby a super-network is trained and sampled by an evolutionary algorithm. Where GLiT adopts SPOS to divide the search space, NEAS instead shrinks the search space by first computing architecture similarity and operator quality (in practice approximated by the mean output of the ensemble architectures). Contrastingly with GLiT, NEAS does not consider transformer architectures within its search space. The worst performing ensemble architectures with respect to diversity and quality are then dropped. In tandem with sharing weights between the lowest layers of the ensemble networks, the best *ensemble* of classifiers can then be searched for under reasonable time complexity.

PAD-NAS [132] also adopts one-shot training in conjunction with evolutionary search based upon NSGA-II [133]. Here however, the pareto-optimal architectures (and the next best, according to nondomination rank [133]) with respect to accuracy and latency are identified. Operations that are not prevalent within these best architectures are pruned, thus reducing the architecture space during the search phase.

It should be noted that weight sharing techniques do not necessitate one-shot methodology. Indeed, BONAS [134] identifies the sensitivity to network initizialization of one-shot approaches, as well as high memory requirements. To this end, a Graph Convolutional Network is employed for Bayesian Optimization to identify similar network architectures. Weights are shared amongst these similar network architectures so that they can be trained simultaneously.

2.5.2 Gradient-Based

Until now, we have considered weight sharing techniques that optimize the architecture topology without updating the weights inherited from the super-network. Differentiable approaches [53, 135, 108] build upon the weight-sharing technique through application of stochastic gradient descent and other well-used deep learning techniques by relaxing the search space such that it is continuous. Consequently, convergence rate of the architecture is drastically improved. In general, this approach proffers the fastest architecture search without significant performance impact, but at the expense of a high GPU-memory intensity.

DARTS [53] constructs a shallow super-network in which each layer is in fact a softmax over all possible operations within the architecture search space (Fig. 3a), to allow traversal of the search space with gradient descent (Fig. 3b). Once the super-network is trained (Fig. 3c), they extract the top-k best-performing

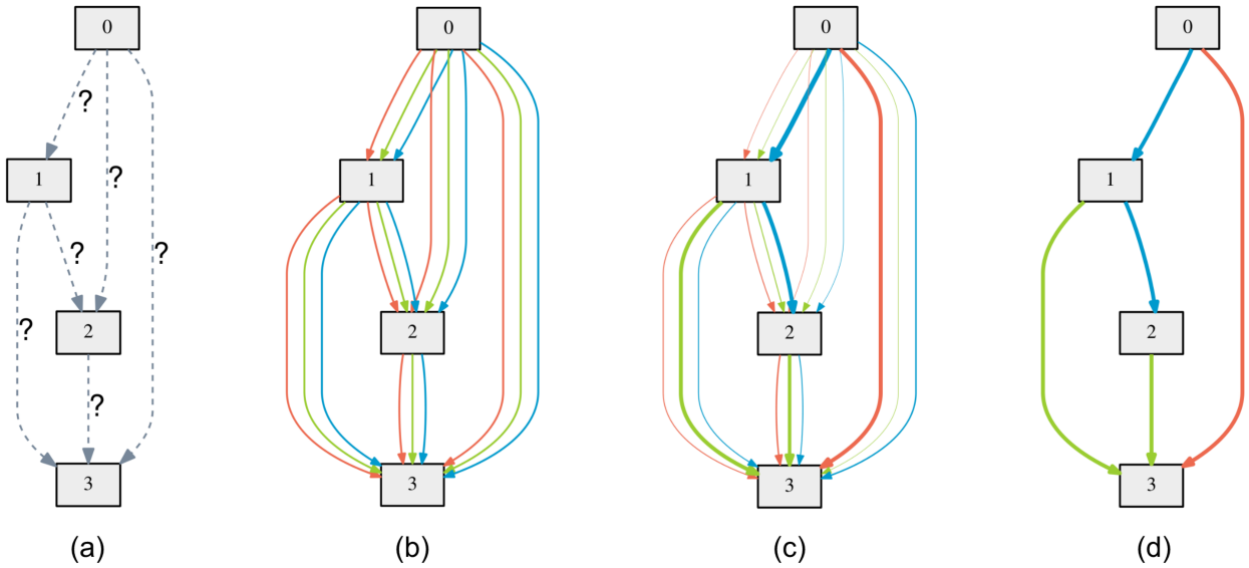


Figure 3: DARTS search phase pipeline. (a) The architecture search space can be realized as a Directed Acyclic Graph (DAG): edges between nodes represent possible operations (e.g. convolution, max-pooling, etc.). (b) A layer is a softmax over all possible operations within the search space. (c) Using SGD, this super-network is trained with respect to both which operations perform best, and the weights of the operations themselves. (d) A final searched network is selected from the top performing operations at each layer.

operations at each layer, and 'evaluate' a deep neural network under the resultant restricted architecture search space to produce a final optimized model (Fig. 3d).

2.5.2.1 DARTS-Like NAS approaches

P-DARTS [108] further capitalizes on the DARTS method, and minimizes the gap in performance between search results and their respective 'evaluation' scenarios (*denoted optimization-gap [134, 136]*), by progressively growing the NAS network depth during the search phase. The authors further propose two approximation techniques to alleviate the prevalence of skip-connections that occur due to instability and difficulty of training a deep network [137], operation-level dropout after skip connections, as well as a cap on the number of skip connections that can occur in the final architecture. They achieve over 1% lower error rate while affording an order of magnitude lower search time than DARTS, and two orders of magnitude lower than its predecessors.

PC-DARTS [138] corroborates that the weight-free operations (skip-connections, max-pooling) are prevalent within NAS-generated architectures since they increase training stability in early NAS iterations. To alleviate this, the authors suggest an alternative approach, whereby only a few of the available operations are considered at each epoch. However, this in turn introduces further instability into the training process, which they demonstrate can be ameliorated by applying edge normalization (with negligible computational

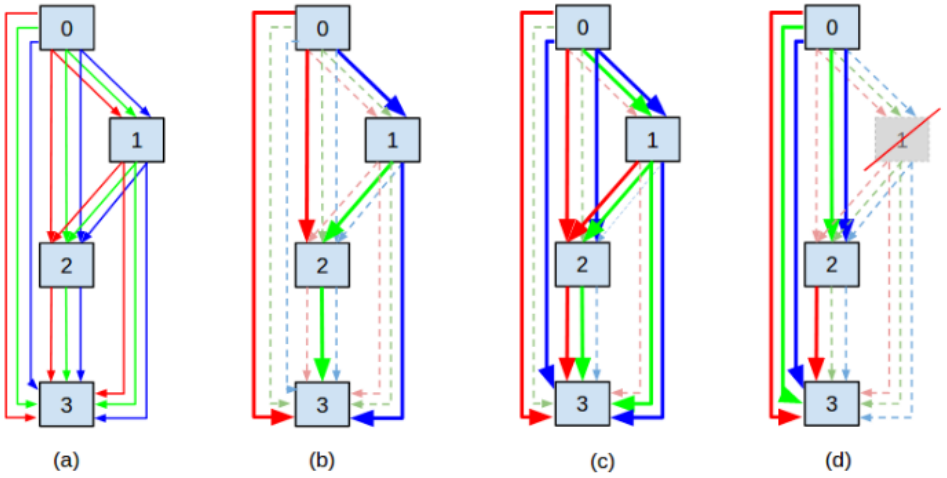


Figure 4: Different NAS architecture configurations as presented by Wu et al. [2]. (a) A NAS super-network where each coloured edge denotes a single operation, and all possible operations are considered between nodes through continuous relaxation (softmax layer). (b) Single-Path architecture; exactly one operation is selected in the final searched architecture. (c) Multi-Path architecture; exactly n operations are considered and aggregated between each and every node (for some pre-defined n). (d) Mixed-Path architecture; no limitations on operation number between given nodes in the final searched architecture.

overhead).

I-DARTS [139] builds upon DARTS by instead considering a softmax operation before operations rather than after. This removes the restriction upon the final model to consider at most one operation between each layer, thereby widening the search space, without reducing convergence rate. We can perceive such an approach as Mixed-Path in that it does not require a single operation between each node (Single-Path, [53, 122]), or multiple (but consistent number of) operations between each node (Multi-Path, [140, 141]), see Fig. 4. Wu et al. [2] adopt an alternative Mixed-Path approach. Rather than use softmax to relax the super-network graph into a continuous DAG, each intermediate node output is computed as a scaled linear combination of the *feature maps* of the previous nodes. By using Sparse Group Lasso regularization [142], there is implicit node and operation selection, in that the sparse regularizer may conclude that as many or as little of the nodes and operations contribute to performance, and thus there is no rigid constraint on the node or path structure. E-DNAS [143] further introduces flexibility within the proposed architecture by explicitly searching for the optimal kernel size as well as the weights of convolutional layers.

ISTA-NAS [136] adopts an alternative strategy compared to [2] towards fulfilling the sparsity constraint. By projecting the continuous relaxation of operations onto the sparse constraint, its LASSO formulation [144] can be solved with the ISTA [145] algorithm. Consquently, ISTA-NAS enables the same size (width, depth, batch-size) super-network to be used in both the NAS search and evaluation phase, due to the sparse

and more efficiently encoded super-network. In turn, the optimization-gap problem is minimized.

2.5.2.2 Addressing DARTS-Like Strategy Drawbacks

Despite the multitude of developments directly upon DARTS, it is not without its drawbacks [134, 146], and has received heavy criticism. To some extent, such problems can be minimized by careful super-network training schemes, including additional batch normalization, prevention of over-regularization, and reduced dropout [120]. FairNAS [147] formally identifies weaknesses in training a one-shot model in general, and proposes strict fairness (all single paths through the super-network are attributed equal optimization). Their work is deployable atop all two-stage NAS strategies, which they choose to demonstrate within an evolutionary search strategy based on NSGA-II [133].

The predominant problem with DARTS however, is perhaps the optimization-gap problem, in which searched architecture performance does not necessarily correlate with its performance after being re-trained during the NAS evaluation phase. One obvious reason for this is one-shot methods are ranking networks relative to each other rather than their absolute performance [120]. Further, and as previously discussed in [108, 136], the super-network architecture in the NAS search phase generally differs considerably to a derived sub-network architecture in the NAS evaluation phase. In fact, many NAS approaches only search for a cell structure during the search phase (owing to available computational resources), which is then stacked before being retrained during the NAS evaluation phase.

Yu et al. [146] identify the importance of random seed during the search phase. Indeed some search strategies, especially DARTS, perform worse than random with respect to some seeds. The authors' findings align with [120] in that the *ranking* of architectures yields a poor reflection of their performance after the evaluation phase. Further, weight sharing is detrimental to the NAS search phase. Finally, good performance from architectures can be attributed to the heavy search space restrictions, such that even random search over the space yields high performing architectures.

DOTS [148] shows with rank correlation analysis that a given searched cell within DARTS can be sub-optimal, as the joint search for operation and topology does not necessarily lead to the most suitable outcome. Indeed, DARTS derives topology from the best performing operation, but there is no guarantee that this operation should be retained in the toplogy at all. To this end, operation search and topology search are decoupled, yielding more optimal final searched topology.

Pi-NAS [149] addresses the optimization-gap prevalent within gradient-based NAS methods in an alternative fashion, considering the image inputs to the NAS solution rather than directly addressing NAS operation and topology selection strategies. By introducing negative samples to a training iteration, thereby drawing from the benefits of the thoroughly researched contrastive learning domain, correct loss descent

can be ensured and better guarantee accurate architecture ranking. Further, a given input image is augmented four times to be passed through separate super-network paths to yield better architecture ranking. As such, the optimization-gap is minimized as final searched architectures better resemble their standalone performance.

EnTranNAS-DST [150] further addresses the optimization-gap by representing non-derived connections in the final searched model as zero-weighted connections. As such, propogation of the super-network in the NAS search phase is the same as propogation of the final searched architecture in the NAS evaluation phase, eliminating the gap between the two.

Landmark Regularization [151] adopts an alternative approach to ameliorating the optimization-gap. Prior to super-network training, randomly sampled standalone architectures (landmarks) are trained to convergence. During training of the super-network, the performance of the sub-network architectures are preserved through an additional regularization term within the loss function wherein performance divergence from known landmark performance is minimized. To minimize harmful impact of the regularizer during early training iterations, the regularizer is only introduced after a sufficient warm-up period.

Shapley-NAS [92] reconsiders the derivation of the final searched architecture given a super-network trained by DARTS. Given that there is often a complex relationship between operations, simply selecting the strongest operation at each edge is flawed. Instead, with a single forward pass of the architecture, a Shapley score [152, 153] can be approximated with the Monte-Carlo algorithm for each operation to better quantify its contribution. Moreover, employing Shapley score in place of gradient descent better trains the super-network during the search phase.

2.5.2.3 Further Gradient-Based NAS developments

TAS [154] adopts a gradient-based search strategy to search for optimal size (network width and depth) instead of topology. Superfluous channel inputs (determining network width) and layers (network depth) are pruned. By designing a novel loss function, complex architectures are penalized, and the best performing network with respect to both accuracy and complexity is identified. Using knowledge transfer with a KD algorithm [155], the searched pruned network architecture inherits weights from the trained super-network.

DNA [156] posits that weight sharing may lead to a poor evaluation of an architecture (a potentially powerful architecture could be attributed a weak evaluation since it inherits inappropriate weights). Therefore, they divide the architecture search space into blocks with similar architectures, and weights are shared only within the blocks. Distribution Consistent NAS [157] adopts a comparable strategy, whereby architectures sharing at least one operator are iteratively sampled and their weights updated. The (layer-wise) architecture space is divided into clusters of architectures sharing operators at a given layer, via K-means

clustering. The super-network can thus be jointly optimized not only with respect to its parameters but also topological structure (i.e. between which architectures the weights can be shared [157]).

BossNAS [158] employs block-wise NAS similar to [156] within a CNN-transformer hybrid network. By constructing a searchable cell that can simulate both convolution and transformer network architectures, and a fabric [159] consisting of several such cells that optionally can halve the spatial resolution, the searched architecture resembles either conventional CNN, transformer, or a mixture of the two.

SETN [160] also adopts the super-network weight sharing method, but only one path through the super-network is considered and optimized during a given training iteration. The method by which the path is sampled is paramount here. Random sampling [161, 120] can lead to unnecessary consideration of poorly performing network architectures. Alternatively, sampling may expose the "Matthew effect." Quickly converging architectures appear as better sampling candidates: they consist of fewer convolution layers, and thus perform poorly overall after retraining from scratch [162, 163] (for instance a surplus of skip-connections at the expense of convolution layers). Consequently, the best performing network architectures when fully trained may be ignored. SETN adopts a stochastic operation and input selection strategy that avoids the Matthew effect (again inheriting weights optimized from a super-network), while simultaneously adopting an evaluator to minimize the selection of poorly performing architectures.

Zela et al. [162] similarly identify the prevalence of skip connections within DARTS-generated architectures, which they attribute to exploding eigenvalues during the NAS search phase. By increasing $l2$ regularization when the dominant eigenvalue exceeds a threshold, DARTS performance was increased across the board.

GDAS [164] samples one architecture of the super-network at each training iteration in an attempt to i) reduce the memory requirement during training and ii) increase efficiency, and by extension, convergence rate of the network. Furthermore, the authors suggest that searching for the best reduction cell can be ignored during the NAS process, since they can be effectively hand-crafted and contribute less to overall network architecture performance. They claim to be able to produce state-of-the-art results in a fraction of the time, but acknowledge that without re-implementing existing methods and evaluating them on the same experimental setup, the results are not necessarily fair. NSAS [165] adopts an alternative approach to CAS (Section 2.5.1) to prevent 'forgetting' prior knowledge (the previous network architecture performance reduces under weights learned by a new architecture), through introducing a novel loss function that penalizes such an occurrence. The NSAS solution is interleaved within the existing GDAS framework, denoted GDAS-NSAS.

Yan et al. [166] employ a variational graph isomorphism autoencoder before traversing the architecture search space. They conclude that this autoencoder out-performs state-of-the-art autoencoders [15, 167] and

best captures the local structure information of neural network architectures such that similar structures cluster better in the latent space. Traversing the search space such that the next most similar, unevaluated network architecture is evaluated in the next iteration, they are able to smooth the NAS search phase, leading to better overall performance.

NetAdaptV2 [168] introduces channel-level bypass connections (CBC) to simulate removal of filters for a given layer. In this way, all output channels of a given layer can be bypassed to simulate removal of the entire layer, and thus equate searching network width and depth. Levying CBC in conjunction with ordered dropout, whereby several sub-network architectures can be trained in a single forward-backward pass, a super-network can be efficiently trained. Final architectures are then efficiently derived from the trained super-network with respect to their latency constraints (or other search metrics).

BMTAS [169] employs a NAS pipeline within a differentiable search space best adapted for multi-task learning. Through masking (to simulate training one sub-network architecture at a time) and a novel resource-aware objective function, their pipeline formulates and traverses the search space in such a manner that promotes general purpose features (operations) within the final NAS-generated architecture.

SMASH [161] uses an auxiliary HyperNetwork [170] network to generate the weights of the network architecture itself. A super-network is generated to encompass the architecture search space. Much like conventional gradient-based NAS solutions, an architecture is sampled from the super-network. However, where its weights would normally be inherited directly, instead they are generated by a HyperNetwork trained a-priori.

FBNet [171] employs a differentiable NAS strategy wherein cells at different network architecture depths are searched from different architecture spaces, across expansion rate, kernel size, and group number (for group convolution). Further, they demonstrate that optimizing network latency is a superior measurement towards generating small and fast networks than optimizing FLOPs. Indeed, their generated network architectures outperform MobileNetV2 [44] with respect to size and speed, and achieve better accuracy and lower latency than even MnasNet [102], a leading efficient convolutional neural network.

Simon et al. [172] adapt DARTS such that convolutional layers have an additional noise injection module. Weights associated with this module learn how much noise to inject into a given input such that DARTS succesfully trains in the presence of label noise. Indeed, the results indicate that in the presence of noise, the modified DARTS method achieves superior performance, without performance degredation when input data is clean.

SVD-NAS [173] propose an algorithm to optimize the search for low latency network architectures via substitution of architecture layers with those optimized for FLOPs (low-rank approximation). The results are presented for gradient-based NAS frameworks, but can be deployed alongside any two-stage NAS approach.

2.5.3 Performance Prediction

An alternative method to reduce NAS training speed is to forgo training to completion entirely and instead *predict* how well a given network architecture will perform from its behaviour after minimal training. PNAS [119] first trains all network architectures in the search space composed of one block B1 (where a block is itself a shallow network), and trains a surrogate predictor network based on the performance of said network architectures. Progressively more complex network architectures are constructed by expanding each block in B_1 with each block in B_2 (for a total of $|B_1| \times |B_2|$ network architectures). Rather than train this latter, large set of network architectures, the surrogate predictor is employed to evaluate their performance. The K-best evaluated network architectures are then trained, and the process is repeated until network architectures of sufficient complexity are generated. In this regard, the predictor network guides the search through the architecture space.

NAO [174] employs an encoder to map a given neural network architecture into a continuous embedded space. Using an auxiliary predictor network to predict network architecture accuracy from its continuous representation, gradient ascent can be applied to determine the best (embedded) network architecture. Finally, a decoder network is used to extract a generated network architecture from its continuous representation. Despite employing a gradient-based search strategy, we include NAO within this section as an architecture sampled during the search phase is not evaluated in a conventional manner (i.e. network propogation with images), but by an auxiliary predictor network.

Wen et al. [175] train their own (graph convolutional based) predictor regression model, wherein N architectures from the NAS-Bench-101 search space are sampled, along with their valiation accuracies. Indeed, their network converges faster and more accurately than Regularized Evolution [176], the best identified predictor adopted by NAS-Bench-101 [114]. The regression model is further trained on the ProxlessNAS [122] search space, yielding competitive models for ImageNet.

MdeNAS [109] posits that network architectures that perform well after minimal iterations perform well after convergence. As such, there is no benefit in training each network architecture to completion for evaluation purposes during the search phase. Indeed, they demonstrate this hypothesis, while presenting their NAS pipeline within a multinomial distribution framework, achieving state-of-the-art results 6.0x faster than concurrent (non-performance-prediction-based) NAS methodology.

Baker et al. [177] formulates architecture performance prediction within a Bayesian framework. They train the predictor network upon both features (architecture parameters and hyperparameters) and time-series validation accuracy data (i.e. validation accuracy of a given network at several different epochs, for many networks), as well as first and second order validation accuracy differences. They train an ensemble of

sequential regression models where each successive model uses an additional point from the time-series data. They demonstrate that their final predictor network is well suited to determine whether a given partially trained network architecture is worth terminating or continuing training, and therefore sufficient for fast hyperparameter optimization algorithms such as Hyperband [178].

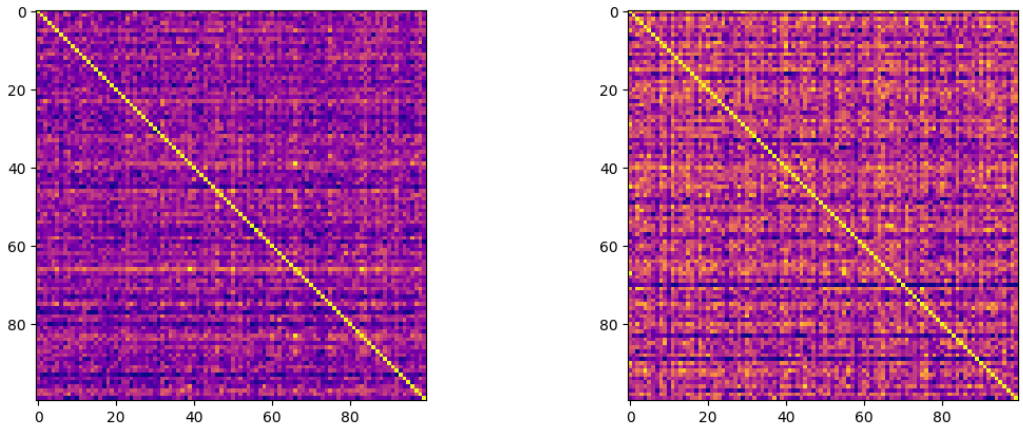
GBDT-NAS [179] employs a gradient boosting decision tree (GBDT) in order to predict the performance of the neural network architecture during the search phase. They further corroborate that pruning the search space into a smaller, but well performing space allows the NAS controller to sample the best architectures with higher probability [118]. Levying the ability of GBDT to identify the importance and contribution of a feature (i.e. an operation), they are able to prune architectures that employ operations with poor performance contribution.

NASWOT (NAS Without Training) [110] predicts the performance of network architectures without any auxiliary models. By examining the network architecture performance after being trained on a single minibatch, they are able to accurately predict its performance after full training. The local linear maps of network archcitectures that perform best will be independent across data point samples. Equivalently, a well performing model must be able to distinguish between the local linear operators associated with each data point in order to model a complex target function; a poor performing network architecture's operators will 'activate' similarly for different images in a minibatch, and thus the image inputs are difficult to disentangle (their respective activation matrix will appear denser - Fig. 39). Their pipeline is able to achieve near-state-of-the-art accuracy in seconds (rather than hours).

ReNAS [180] encodes a given architecture into a feature tensor representing an adjacency matrix of the operations between given nodes. A predictor is trained to map feature tensors to architecture performance, preserving ranking of different architectures rather than MSE loss between a given architecture and its performance, since the relative performance between two architectures is more important during the search phase than their absolute performance.

RMI [3] reformulates the NAS search phase as an operation selection challenge for a given edge in an architecture. In turn, this edge can be represented as a one-hot vector, enabling representation of an architecture as a matrix, for input to a random forest. Architectures are derived by the forest, through an iterative selection-update process using a novel RMI score based off mutual information and approximated by Hilbert-Schmidt Independence Criterion (HSIC) [181]. Once a sufficient number of architectures have been generated, the average (mode) operation for each edge is chosen for the final architecture (Fig. 6).

Of course, performance prediction strategies are not without their limitations. Mok et al. [182] suggest that several prediction-based strategies are inherently flawed. Estimating network performance at initialization often employs the neural tangent kernel (NTK), for which Frobenius Norm (utilized by RMI [3]) and



(a) (b)

Figure 5: Visualization of the ability of a generated network to distinguish between given image inputs. Row i, column j corresponds to the hamming distance between the binary codes representing the activation pattern of the ReLU operations of the given neural network architecture, induced by image i and image j. The matrix is normalized such that the distance between the codes induced by identical images (the diagonal) is 1. High performing network architectures (a) therefore have fewer off-diagonal elements.

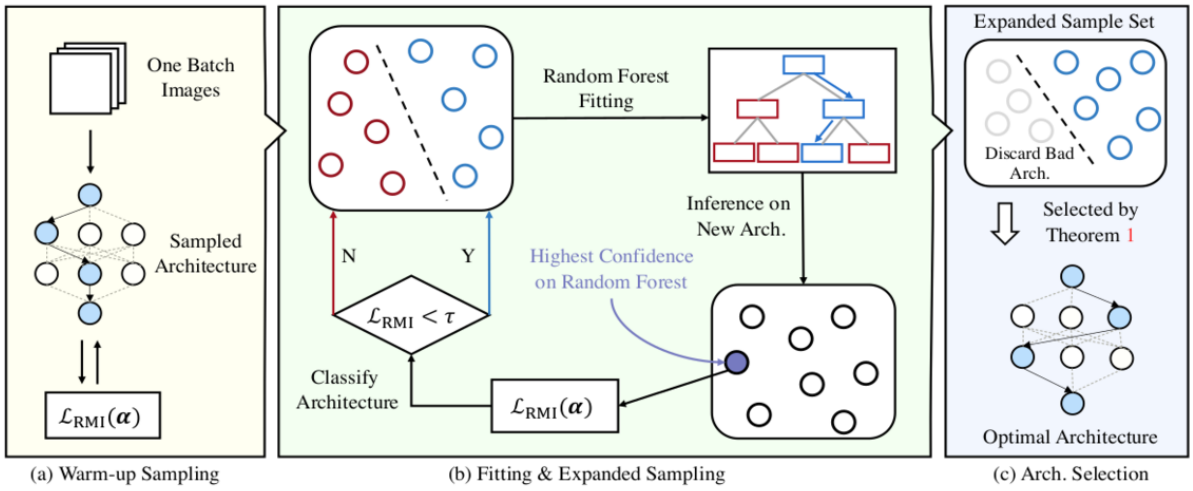


Figure 6: Three step NAS process from RMI [3] paper. (a) RMI score is used to classify good and bad architectures from the search space, additionally warming-up the random forest. (b) Architectures are selected according to random forest confidence. The architecture performance is estimated via RMI score, both classifying the architecture as good or bad, and for training the forest. (c) The most common operation at each edge from the best architectures is selected to generate the final architecture.

other common techniques are based. They demonstrate that modern DNN violate assumptions necessary to adopt NTK, which evolve non-linearly during training.

FreeRea [183] also acknowledge the limitations that NTK methods yield, and build upon the earlier genetic REA [184] algorithm by independently mutating parent cells and then uniformly sampling the resul-

tant cell genes from one parent. Such mutation and crossover operation strategy better explores the network architecture space. By adopting a modified Synflow [185] approach to evaluate the summed contribution of network architecture weights, wherein the weights are scaled down logarithmically (termed 'LogSynflow'), FreeRea assigns a more appropriate fitness score to a given network architecture cell. Additionally, architectures with skip connections are in fact *encouraged* to yield practical deep network architecture training.

2.5.3.1 Bayesian Optimization

In general, bayesian optimization (BO) considers a function $f(x)$ that is complex or unknown (thus behaving as a "black box"). In the context of NAS, we can denote f as the performance of a given architecture x. To optimize f, we require some kernel k that considers the distance between two inputs (x, x_1). Furthermore, we require an acquisition function $a(f, k, x)$, a measure of an expected loss of evaluating f at x, given a kernel function k. For clarity, let us consider the architectures x, x_1, where $k(x, x_1)$ is very small (i.e. the architectures are similar). If $f(x)$ is high (i.e. architecture x performs well), we would be wise to compute $f(x_1)$, as architecture x_1 is likely to perform well. Had $f(x)$ been low (i.e architecture x performs poorly), we should instead compute $f(x_2)$, for some alternative architecture x_2. There is litte benefit in computing $f(x_1)$ as it will be similar to $f(x)$, while computing $f(x_2)$ enables better exploration of the entire search space. Provided a is more easily computable than f, a BO approach to architecture selection via maximising a should be efficient.

Indeed, NASBOT [186] first adopted BO strategy for NAS, utilizing expected improvement as the acquisition function. Further, they define k as the OTMANN distance, a measure of the structural similarities between two architectures, weighted by their computational contribution to the network as a whole. This distance is shown to be computed efficiently via optimal transport algorithm [187].

Auto-Keras [188] adopts an alternative BO configuration, using upper-confidence bound acquisition function, with edit-distance as kernel definition, solved by an approximate dynamic programming algorithm that can be minimized under an equivalent bipartite graph matching problem. BayesNAS [189] adopt entropy-based acquisition function with incorporated hierarchical ARD prior [190].

BANANAS [191] identifies the drawbacks with rudimentary BO strategy, given the resouce intensive distance function computation. Instead they propose use of a predictor network to negate use of a distance function entirely. Consider an architecture encoding as a binary mask of the entire search space, where there is a 1 if that path (series of operations from input to output) exists in the architecture. Given such a path-encoding representation of an architecture, a neural network can predict its performance. Taking an ensemble of m predictors, the mean and standard deviation of the m predictions for an input architecture can be computed, and utilized within BO as a relatively reliable uncertainty measurement compared to previous

BO strategies, improving subsequent uncertainty callibration. The authors evaluated the performance of the framework across a range of configurations, and determined upper-confidence bound to be the highest performing acquisition function, in conjunction with a mutation optimization strategy.

2.6 Object Detection

In general, NAS towards more complex tasks than image classification, such as detection and segmentation is less studied. Further still, investigation of NAS within such complex task domains is generally restricted to searching for backbone architectures, rather than the end-to-end detection or segmentation architecture. DetNAS [192] identify the unsuitability of older NAS strategies (notably non-gradient-based) towards searching for detection backbone architectures due to the level of granularity required, and thus the necessity to pretrain architectures on ImageNet. As such they propose DetNAS which, much like gradient-based NAS strategies, generates a super-network which only requires pretraining on ImageNet once [130]. Converse to gradient-based strategies however, only one path is sampled during each iteration, and thus proposed architectures have entirely independent weights. Furthermore, super-network training and search space traversal is decoupled, allowing convergence to be achieved by an evolutionary algorithm rather than gradient-based.

SpineNet [4] employs a reinforcement learning (RL) NAS strategy to determine backbone architectures for object detectors. They posit however, that common leading scale-decreased backbones (e.g. ResNet [18]) may be unsuitable for detection architectures due to the loss of spatial information within down-sampling. This information may not be fully recovered by subsequent decoder networks, including FPN [193]. As such, generated architectures contain a (fixed, scale-decreased) stem followed by a learned scale-permuted network consisting of several blocks. Each block need not necessarily connect to a subsequent block corresponding to the next lowest resolution (scale-decreasing). Instead, blocks can connect to blocks of varying resolution, using nearest neighbour interpolation (upsampling) or stride 2×2 convolutions (downsampling) between blocks, as necessary (see Fig. 7).

NATS [194] considers a gradient-based NAS approach for object detection backbone architectures. In order to achieve the level of granularity required, lest found backbone architectures generate too coarse features [192], NATS further decomposes the search space beyond path-level strategies to the channel-level. Each channel at each operation is assigned its own parameter, allowing the channel search space to be continuous for gradient search.

Conversely to backbone search, [195, 196] consider the FPN network architecture as the search space within their NAS frameworks. NAS-FPN [195] employs reinforcement learning to iterate over the FPN search space in their framework. They propose a 'general' FPN-block, whereby two feature layers are sampled, and

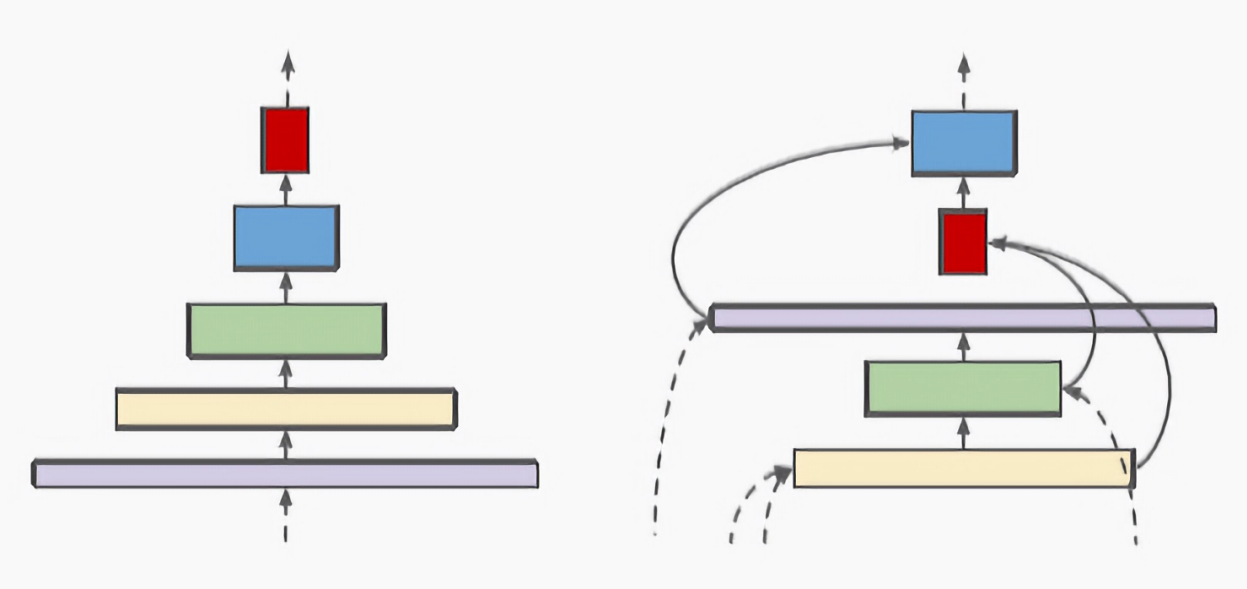


Figure 7: Scale-decreasing vs Scale-permuted network from SpineNet [4] paper. The width of the block indicates feature resolution and the height indicates feature dimension

then combined (with either sum or global pooling) to generate a new feature layer. The input layers, binary combination operation, and output layer resolution are determined by the NAS RNN RL controller. The output layer is inserted back into the initial stack to be sampled itself by the controller. The authors further propose a simple but effective strategy to realise the accuracy-speed tradeoff, whereby the FPN architecture can be stacked since its input and output are feature layers of identical scales.

Auto-FPN [196] opts for a gradient-based NAS framework to generate detector architectures. Similar to NAS-FPN [195], FPN network architecture is generalized within the search space, but the generalization is further extendable to PANet [197] and SSD [198] style pyramidal network architectures. Furthermore, the authors consider a head cell within the search space, to optimize classification and bounding box regression. Their Auto-FPN network architecture yields less accurate results than the concurrent work [195], but with a fraction of the resources required during training.

Drawing from both object detection NAS paradigms, NAS-FCOS [199] benefits from searching for both a competitive FPN as well as bounding box regressor head. Generating network architectures based upon the FCOS [200] anchor-free network architecture, they are able to achieve state-of-the-art performance. They construct the FPN search space as a sequence of blocks, where each block is constructed as an aggregated selection of operations performed upon two sampled feature layers. The highest most pyramid feature layers are obtained via 3×3 stride-2 convolutions on the preceding pyramid layers, consistent with FCOS. The regressor head search space is defined as a sequence of six operations, drawing from the same pool as the FPN search space, as well as the standard convolutions incorporated within FCOS and RetinaNet [201]. In

keeping with FCOS design and general NAS procedure, Group Normalization [202] is also used in place of Batch Normalization. First, the FPN search space is traversed while the regressor head is frozen. Following the discovery of a high performance searched FPN architecture, the regressor head is generated. The top 10 searched head architectures are then selected for full training to determine the best single FCOS-based network architecture.

OPANAS [203] applies the NAS strategy to searching for optimal FPN architecture within visual object detection. Representing a node as a feature map, and edges between nodes as possible information paths (top-down, bottom-up, scale-equalizing, fusing-splitting, skip-connect and none), an FPN super-network can be constructed as a DAG akin to commonplace NAS solutions. As such, the optimal aggregation of information paths can be derived from a trained super-network through an evolutionary algorithm.

2.7 Image Segmentation

The image segmentation domain poses a new style of problem compared to previous vision-based challenges, namely capturing long-range dependencies between features for dense (pixel-wise) prediction [204, 205]. Common solutions include scale image pyramids [206, 207, 208], encoder-decoder networks [209, 210, 211] and atrous convolution resampling [212, 213, 214].

DPC [204] constructs a novel search space for dense prediction, encapsulating both spatial pyramid pooling and atrous seperable convolutions, thereby capturing the aforementioned multi-scale contexts. With a random sampling search strategy, they are among the first to adapt NAS towards image segmentation, outperforming hand-crafted architectures for scene parsing, person part segmentation and semantic image segmentation.

Auto-DeepLab employs a different strategy wherein gradient-based search is adopted to find cells optimized for dense prediction. In addition to searching for optimal convolutional fabric [215] cells, the hierarchical network level search space is also traversed. High level spatial resolutions are thereby preserved as the inter-connectedness of a searched cell is not pre-defined, but explicitly searched for (Fig. 8).

DCNAS [205] builds upon the trellis search space [5], constructing a densely connected search space. By using a fusion module that efficiently aggregates semantic information between layers, the resource-intensity during search is minimized such that a given architecture can be searched for in a proxyless fashion; the dataset the final architecture is trained on is used during the search phase, ameliorating any contradiction between final and searched architecture performance (i.e. bridging the *optimization-gap*).

EDNAS [143] present a multi-task scene understanding (image segmentation, depth prediction, and surface normal estimation) NAS algorithm that focuses on the generation of network architectures optimal

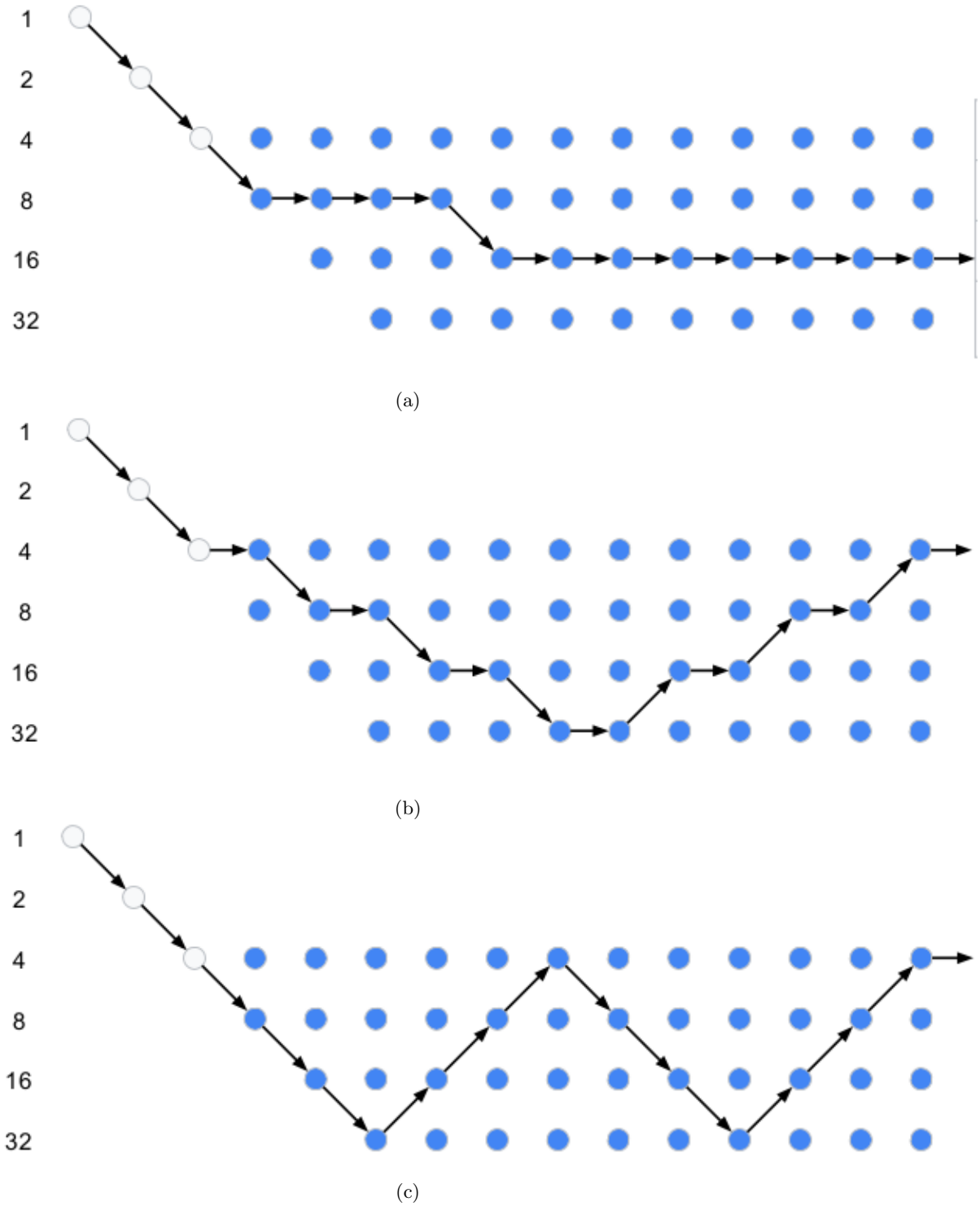


Figure 8: Examples from Auto-DeepLab [5] paper of the different network architectures that can be captured by their search space. Spatial resolution only ever doubles, halves, or remains unchanged in a given layer. Maximum downsample rate is 32. (a) DeepLabv3 [6]. (b) Encoder-decoder architecture, successfully deployed within semantic segmentation by Conv-Deconv [7]. (c) Stacked hourglass [8] architecture.

with respect to latency for given hardware. By designing a search space best suited to the placement of Inverted Bottleneck [44] blocks within an EfficientNet [216] backbone, an evolutionary search algorithm [184] can find architectures optimized for edge platforms.

2.8 Discussion

Clearly, and has been previously mentioned, there is a definitive bias towards NAS for image classification over object detection and image segmentation. This can be attributed to the complexity of image classification architecture, which until recently, could be more easily trained end-to-end. As such, these CNN architectures are well suited for architecture search within the NAS pipeline. While extending CNN architecture search to backbones to object detection and image segmentation is possible, the result is not as impressive. Indeed, suitable backbone architecture for these problems is only half the challenge. However, with the rise of object detection transformer architecture, which achieves very high performance on common datasets, and whose modularity is well suited towards NAS, one can expect this phenomenon to disappear. It is unsurprising therefore, that neural architecture search for transformers is receiving increased popularity in recent literature [129, 158]. We further note that implementing the architecture space for NAS in line with the modifications implemented within ConvNeXt architecture [24], capable of state-of-the-art results in both image classification and object detection, is a strong candidate for future research within the NAS domain.

The focus on architecture space *traversal* is evident. Conversely, even in the case of image classification, generation of resource-efficient architecture is limited, for which NAS is so well-suited. Moreover, where such strategies do exist [126, 129, 154, 168, 119], only [129] considers architecture beyond pure CNN or domains other than image classification. Since NAS can ultimately be reduced to a ranking of architectures, and thus introducing resource constraints into the ranking is both sufficient and efficient, this pattern is unfounded and there is much benefit to be gained here.

Finally, we note that dataset optimization is hardly considered within NAS frameworks, with the exception of Pi-NAS [149]. Hard example mining and curriculum learning, prevalent within conventional network training , receive no attention within NAS (excluding CNAS [113] which utilizes a curriculum for preparing the architecture space rather than the dataset). Considering a given dataset is often iterated over more times within NAS than manual training, there is no justification for this.

Reference	Technique							Top-1 Acc (CIFAR-10) (%)	Params (CIFAR-10)	Top-1 Acc (ImageNet) (%)	Params (ImageNet)
	Evolutionary Algorithms	Reinforcement Learning	Gradient Based	Prediction	Weight Sharing	Sample Based	One Shot				
Zoph & Le[99]		✓				✓		96.35	37.4M	n/a	n/a
MnasNet [102]		✓				✓		n/a	n/a	76.7M	5.2M
ENAS [111]		✓			✓		✓	97.11	4.6M	n/a	n/a
SNAS [112]		✓			✓		✓	97.02	2.9M	72.7	4.3M
CAS [121]		✓			✓		✓	n/a	n/a	n/a	n/a
CNAS [113]		✓			✓		✓	97.40	3.7M	75.40	5.3M
ProxylessNAS-R [122]		✓			✓		✓	97.70	5.8M	74.6	n/a
ProxylessNAS-G [122]			✓		✓		✓	97.92	5.7M	74.2	n/a
NASP [123]			✓		✓		✓	97.56	7.4M	73.7	9.5M
TuNAS [98]		✓			✓		✓	n/a	n/a	75.0	n/a
BigNAS-S [125]					✓		✓	n/a	n/a	76.5	4.5M
BigNAS-M [125]					✓		✓	n/a	n/a	78.9	5.5M
BigNAS-L [125]					✓		✓	n/a	n/a	79.5	6.4M
BigNAS-XL [125]					✓		✓	n/a	n/a	80.9	9.5M
AttentiveNAS (largest) [126]					✓		✓	n/a	n/a	80.7	n/a
Stage-Wise NAS [127]					✓	✓		95.68	7.27M	n/a	n/a
GLiT-Tiny [129]	✓				✓		✓	n/a	n/a	76.3	7.2M
GLiT-Small [129]	✓				✓		✓	n/a	n/a	80.5	24.6M
GLiT-Base [129]	✓				✓		✓	n/a	n/a	82.3	96.1M
NEAS [131]	✓				✓		✓	n/a	n/a	80.0	n/a
PAD-NAS [132]	✓				✓		✓	n/a	n/a	76.1	4.7M
BONAS [134]	✓				✓			97.57	3.3M	74.6	4.8M
DARTS [53]			✓		✓		✓	97.24	3.3M	73.3	4.7M
P-DARTS [108]			✓		✓		✓	97.50	3.5M	75.9	5.4
PC-DARTS [138]			✓		✓		✓	97.43	3.6M	75.8	5.3M
I-DARTS [139]			✓		✓		✓	97.63	3.8M	75.7	n/a
Wu et al. [2]			✓		✓		✓	97.50	3.5M	75.33	5.7M
E-DNAS [143]			✓		✓		✓	n/a	n/a	76.9	5.9M
ISTA-NAS [136]			✓		✓		✓	97.64	3.37M	76.0	5.65M
FairNAS [147]	✓				✓		✓	98.2	n/a	77.5	5.9M
DOTS [148]			✓		✓		✓	97.51	3.5M	76.0	5.3M
Pi-NAS [149]			✓		✓		✓	n/a	n/a	81.60	27.1M
EnTranNAS [150]			✓		✓		✓	97.78	7.68M	75.70	7.2M
EnTranNAS-DST [150]			✓		✓		✓	97.52	3.20M	76.20	7.0M
Landmark Regularization:SPOS [151]			✓		✓		✓	n/a	n/a	67.38	4.77M
Landmark Regularization:GDAS [151]			✓		✓		✓	n/a	n/a	68.82	5.07M
Landmark Regularization:NAO [151]			✓		✓		✓	n/a	n/a	68.89	4.49M
Shapley-NAS [92]					✓		✓	97.57	3.6	76.1	5.4
TAS [154]			✓		✓		✓	94.00	n/a	76.20	n/a
DNA [156]			✓		✓		✓	98.30	n/a	78.40	6.4M
Distribution Consistent [157]	✓				✓		✓	n/a	n/a	79.50	n/a
BossNAS [158]			✓		✓		✓	n/a	n/a	82.5	n/a
SETN [160]			✓		✓		✓	97.31	4.6M	74.3	5.4
R-DARTS($l2$) [162]			✓		✓		✓	97.49	n/a	n/a	n/a
GDAS [164]			✓		✓		✓	97.07	3.4M	74	5.3
GDAS-NSAS [165]			✓		✓		✓	97.27	3.54M	n/a	n/a
unsupervised: DARTS [166]			✓		✓		✓	97.44	3.6M	n/a	n/a
NetAdaptV2 [168]			✓		✓		✓	n/a	n/a	77.0	n/a
BMTAS [169]			✓		✓		✓	n/a	n/a	n/a	n/a
SMASH [161]			✓		✓		✓	94.47	4.6M	61.38	16.2M
FBNet-C [171]			✓		✓		✓	n/a	n/a	74.9	5.5M
PNAS [119]				✓				96.59	3.2M	74.2	5.1M
PNAS-Large [119]				✓				n/a	n/a	82.9	86.1M
NAO [174]			✓			✓		96.82	10.6M	74.3	11.35M
MdeNAS [109]				✓				97.45	3.61M	74.5	6.1M
GBDT-NAS [179]				✓				n/a	n/a	76.6	5.7M
NASWOT [110]				✓				n/a	n/a	n/a	n/a
ReNAS [180]				✓				n/a	n/a	n/a	n/a
NASBOT [186]				✓				91.31	n/a	n/a	n/a
Auto-Keras [188]				✓				96.40	n/a	n/a	n/a
BayesNAS [189]				✓				97.59	3.4M	73.5	3.9M
BANANAS [191]				✓				n/a	n/a	n/a	n/a
								COCO			
								AP		Params	
MnasNet [102] †		✓						23.0		4.9M	
DetNAS [192] †	✓				✓		✓	42.0		n/a	
SpineNET-49S [4] †		✓				✓		41.5		12M	
SpineNET-190 [4] †		✓				✓		52.1		163.6M	
NATS [194] †			✓		✓		✓	38.4		n/a	
NAS-FPN (AmoebaNet Backbone) [195] †		✓				✓		48.4		166.5M	
Auto-FPN [196] †			✓		✓		✓	44.3		n/a	
NAS-FCOS [199] †		✓				✓		46.1		89.4M	
OPANAS [203] †	✓				✓		✓	41.6		29.8M	
								mIOU (cityscapes)			
DPC [204] ‡								82.7			
Auto-DeepLab [5] ‡			✓		✓		✓	82.1			
DCNAS [205] ‡			✓		✓		✓	84.3			
EDNAS [143]	✓							n/a			

Table 1: Overview of NAS approaches, their performance, and the general methodology employed: Evolutionary Algorithms (EA), Reinforcement Learning (RL), Gradient-Based (GB), Weight-Sharing (WS) and Prediction (Pred). † entails object detection. ‡ entails instance segmentation. Otherwise, reported results are for Image Classification. Results correspond to the results reported in their respective original paper, even when subsequent papers report higher performance or results generated using more comparable computational resources.

NAS search algorithm	CIFAR-10	ImageNet-16-120
I-DARTS [139]	93.76	41.44
FairNAS [147]	93.23 ± 0.18	42.19 ± 0.31
Pi-NAS [149]	93.83 ± 0.00	n/a
Landmark Regularization:SPOS [151]	93.41 ± 0.43	n/a
Landmark Regularization:GDAS [151]	94.32 ± 0.28	n/a
Landmark Regularization:NAO [151]	93.53 ± 0.43	n/a
Shapley-NAS [92]	94.37 ± 0.00	46.85 ± 0.12
Distribution Constrained [157]	94.29 ± 0.07	46.41 ± 0.14
BossNAS [158] †	93.29	n/a
unsupervised: DARTS [166]	94.18 ± 0.24	46.27 ± 0.37
NASWOT [110]	92.96 ± 0.81	44.44 ± 2.10
ReNAS [180]	93.99 ± 0.25	45.97 ± 0.49
RMI [3]	94.28 ± 0.10	46.34 ± 0.00
FreeRea [183]	94.36 ± 0.00	46.34 ± 0.00

Table 2: Some literature report the NAS search phase prediction performance in place of or as well as final searched architecture performance. In these cases we present the findings on NAS-Bench-201 on CIFAR-10 and ImageNet-16-120. † denotes experiments conducted on NATS-Bench, the successor to NAS-Bench-201.

2.9 Conclusion

Within the compression domain, the limited studies open the door only slightly on the very question - what is the generalized impact of compression on varying deep neural network architectures? Here we consider multiple CNN variants spanning region-based, encoder-decoder and GAN architectures in addition to a wide range of target tasks spanning both discrete and regressive outputs. From our observations, we aim to form generalized conclusions on the hitherto unknown relationship between (lossy) image input to target function outputs within the domain of contemporary CNN approaches.

It is similarly important to determine suitable NAS approaches within the same train-test development cycle. There has been clear advancement of NAS towards faster generation and training of networks, but relatively little with respect to generation of *small, efficient* architectures. More notably however, there is little research towards NAS speed-up approaches within the context of data minimisation. Minimizing the data processed within NAS search phase algorithms would allow NAS to be deployed within constrained development environments. Furthermore, such a solution would be framework-agnostic, and could utilize NAS improvements proposed within existing and future literature in conjunction with its own strategy.

Finally, it is evident that curriculum learning and hard example mining are both valid methods of

improving the neural network training process. In fact, combining the two methods yields superior results. We propose therefore to deploy such methods within NAS, specifically, levying mastery from [24] in tandem with our own hard example mining approach reminiscent of the instructor-student collaborative learning paradigm [14]. Moreover, it is necessary to constrain our approach to efficient methods sufficient for training within online environments.

3

Compression

3.1 Introduction

Image compression is in *de facto* use within environments relying upon efficient image and video transmission and storage such as security surveillance systems within our transportation infrastructure and our daily use of mobile devices. However, the use of the commonplace lossy compression techniques, such as JPEG [35] and MPEG [36] to lower the storage/transmission overheads for such smart cameras leads to reduced image quality that is either noticeable or commonly undetectable to the human observer.

Indeed psychosomatic considerations form the basis for the image compression domain. In short, the human eye is able to see due to photoreceptor cells in the retina. These cells can be divided into approximately 120 million rods best adapated to scotopic vision (dim-light conditions), but only 6 million cones that are sensitive to colour [217]. This imbalance of cells leads to a signficantly greater sensitivity to high frequency information and movement compared to colour information.

We define the JPEG [35] algorithm following the ISO/IEC 10918 still image compression standard. Images are converted to the $YCrCb$ colour space to distinguish between chrominance and luminance. Chroma components are subsampled given their relative imperceptibility over luma components, and can be safely reduced. The primary component of JPEG then uses the DCT algorithm [218] to transform both chrominance and luminance components from the spatial domain to the frequency domain. High frequency components, which the human eye is less able to perceive, can be discarded at relatively high rates such that a decoded JPEG image is imperceptibly different, and from which the term 'lossy' compression is derived. This process is termed quantization, where the frequency components are translated to a set of integers (and information is discarded by rounding to zero). The resultant information can be efficiently entropy encoded since most information is in the upper left corner of the frequency space (thus maximizing the impact of run-length encoding and Huffman coding [70] during entropy encoding).

We define the H.264 [37] algorithm following the H.264/MPEG-4 AVC video compression standard. The algorithm uses an integer approximation for the DCT transformation for a given video frame (operating on

Figure 9: Image taken from [9]. A single change to the pixel values can have a significant impact on the output of the kernel.

4x4 patches compared to the 8x8 employed by JPEG), followed by entropy encoding, much like the JPEG standard. In addition however, H.264 applies motion estimation to remove temporal redundacies between given frames. Objects within an image frame are isolated so that their motion vectors compared to reference frames (usually the previous or successive frame) can be captured, whilst the rest of the frame remains unchanged and can be trivially encoded. Compared to previous iterations of the algorithm (MPEG [36], H.262 [219], H.263 [220]), H.264 employs several reference frames, not just those immediately adjacent, to better capture motion vectors. This allows identification and efficient encoding of periodic motion, alternating scenes within a video, and motion under occlusion.

Considering both algorithms remove pixel information during the quantization process, it follows that CNN performance will be impacted, and this forms the basis of the well-established domain of adversarial attacks [221]. Whilst the images may be imperceptibly different to human observation, the convolution kernels are necessarily fine-grained to discern any distinction [222, 223]. It is easy to show that a single pixel value change can have significant output on kernel output (see Fig. 9). It follows that learning kernel weights can be impacted by using training images with different compression levels.

With the recent rise of deep CNN [31, 32] for video analytics across a broad range of image-based detection applications, a primary consideration for classification and prediction tasks is the empirical trade-off between the performance of these approaches and the level of lossy compression that can be afforded within such practical system deployments (for storage/transmission). This is of particular interest as CNN are themselves

known to contain lossy compression architectures - removing redundant image information to facilitate both effective feature extraction [31, 32] and retaining an ability for full or partial image reconstruction from their internals [15, 224, 225].

Prior work on this topic [54, 55, 56, 57] largely focuses on the use of compressed imagery within the train and test cycle of deep neural network development for specific tasks. However, relatively few studies investigate the impact upon CNN task performance with respect to differing levels of compression applied to the input imagery at inference (deployment) time. Moreover, while racial bias within facial recognition is a prominent area of research within computer vision [38, 39, 40, 41], minimal consideration has been given to racial bias introduced during the image acquisition stage, within which compression resides.

This chapter investigates whether (a) existing pre-trained CNN models exhibit linear degradation in performance as image quality is impacted by the use of lossy compression and (b) whether training CNN models on such compressed imagery thus improves performance under such conditions. In contrast to prior work, we investigate these aspects across multiple CNN architectures and domains spanning segmentation (SegNet, [10]), human pose estimation (OpenPose, [12]), object recognition (R-CNN, [11]), human action recognition (dual-stream, [13]), and depth estimation (GAN, [226]). Furthermore, we determine within which domains compression is most impactful to performance and thus where image quality is most pertinent to deployable CNN model performance. Finally, we discuss the additional racial bias deployment challenge that compression introduces, and measures that can be taken to reduce it.

3.2 Architecture Representation

To determine how much lossy image compression is viable within CNN architectures before performance is significantly impacted we must study a range of second generation tasks, beyond simple and holistic image classification, requiring more complex CNN output granularity. We examine five CNN architectural variants across five different challenge domains, emulating the dataset and evaluation metrics characterized in their respective originating study in each case as closely as possible. Inference models processing images were tested six times, with a JPEG quality parameter in the set $\{5, 10, 15, 50, 75, 95\}$, whilst video-based models were tested with H.264 CRF compression parameters in the set $\{23, 25, 30, 40, 50\}$. Each model is then retrained with imagery compressed at each of the five higher levels of lossy compression to determine whether resilience to compression could be improved, and how much compression we can afford before a significant impact on performance is observed. Our methodology for each of our representative challenge domains is outlined in the following sections: semantic segmentation (Section: 3.2.1), depth estimation (Section: 3.2.2), object detection (Section: 3.2.3), human pose estimation (Section: 3.2.4), and human action recognition (Section:

3.2.5). Finally, we consider facial recognition and the additional respective racial bias challenge (Section: 3.5). In all cases it is experimentally analysed that the change in loss is lower than some small δ between two epochs and thus the stated number of epochs is sufficient for the model to have converged. In all cases, performance is evaluated using the standard benchmarks [1, 30, 227, 228, 229] such that it is comparable with the literature. During retraining of the networks, training images are equally sampled from the lowest compression rate (JPEG Paramter 95 / CRF 23) and the compression level in question. All train/validation splits are as presented in the respective literature [1, 30, 227, 228, 229]. We assume that train/test sets are identically distributed, but in some cases drawn from the same global distribution as the training data. This is standard practice in computer vision [230, 231], such as in the autonomous driving domain [232, 233].

3.2.1 Semantic Segmentation

Pixel-wise Segmantic segmentation involves assigning each pixel in an image (Fig. 10a, above) its respective class label (Fig. 10a, below). SegNet [10] uses an encoder-decoder neural network architecture followed by a pixel-wise classification layer to approach this challenge.

Implementing SegNet from [234], we evaluate global accuracy (percentage of pixels correctly classified), mean class accuracy (mean prediction accuracy over each class), and mean intersection over union (mIoU) against compressed imagery from the Cityscapes dataset [229]. When retraining the network, we use 33000 epochs, with hyperparameters from [10]: a batch size of 12, fixed learning rate (η) of 0.1, and momentum (β) of 0.9.

3.2.2 Depth Estimation

In order to evaluate GAN architecture performance under compression, we need a task decoupled from reconstructing high quality output, to which compression would be clearly detrimental. One such example is computing the depth map of a scene (Fig. 11a, below) from monocular image sequences (Fig. 11a, above).

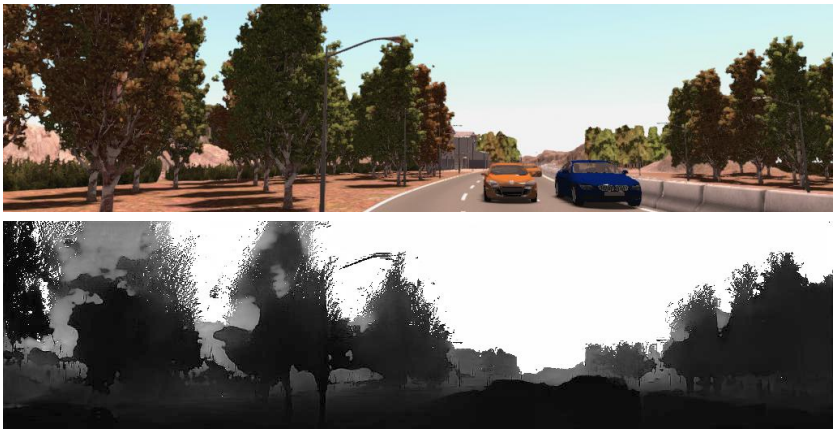
Using a simplified network from [226], we evaluate RMSE performance of the GAN against the Synthia dataset presented in [228]. We employ $\eta = 0.0001$ and batch size 10 over 10 epochs (long enough for the simplified network to converge).

(a) JPEG compression level: 95

(b) JPEG compression level: 15

(c) JPEG compression level: 10

Figure 10: Results of pre-trained SegNet model [10] on a JPEG image under different compression levels (original RGB image above, computed segmentation map below)



(a) JPEG compression level: 95

(b) JPEG compression level: 15

(c) JPEG compression level: 10

Figure 11: Results of pre-trained GAN model on a JPEG image under different compression levels (RGB image above, computed depth map below)

(a) JPEG compression level: 95

(b) JPEG compression level: 15

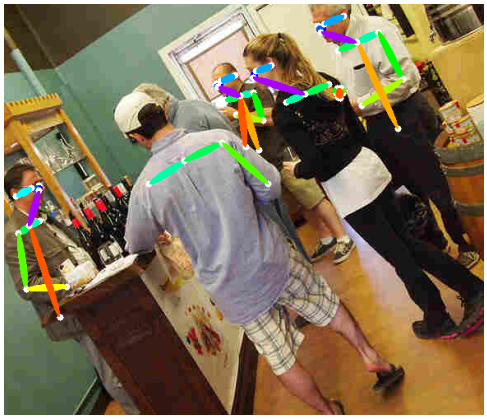
(c) JPEG compression level: 10

Figure 12: Results of pre-trained FasterRCNN model [11] on a JPEG image under different compression levels

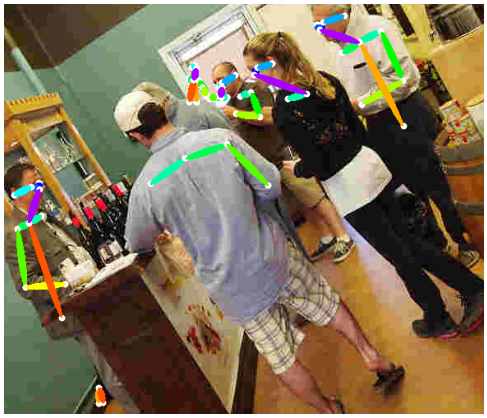
3.2.3 Object Detection

In object detection, we must locate and classify foreground objects within a scene (as opposed to semantic segmentation, which classifies each pixel), and compute the confidence of each classification (Fig. 12a). We evaluate mAP of the Detectron FasterRCNN [11] implementation [235] against the Pascal VOC 2007 dataset [227], over mIoU with threshold 0.5:0.95. When training the network, we use $\eta = 0.001$ and weight decay of 0.0005 over 60000 epochs as specified in [11].

(a) JPEG compression level: 95



(b) JPEG compression level: 15



(c) JPEG compression level: 10

Figure 13: Results of pre-trained OpenPose model [12] on a JPEG image under different compression levels

3.2.4 Human Pose Estimation

Human Pose Estimation involves computing (and overlaying) the skeletal position of people detected within a scene (Fig. 13a). Recent work uses part affinity fields to map body parts to individuals, thus distinguishing between visually similar features.

Using OpenPose [12] we compute the skeletal overlay of detected people in images from the COCO dataset [30]. We evaluate with mean average precision (mAP), over 10 object key-point similarity (OKS) thresholds, where OKS represents IoU scaled over person size. When retraining the network, we use $\eta = 0.001$, and a batch size of 8 over 40 epochs, with hyperparameters from [236].

(a) H.264 CRF value 23

(b) H.264 CRF value 30

(c) H.264 CRF value 40

Figure 14: One frame taken from a video input to the Two-Stream CNN model [13] under different H.264 compression rates

3.2.5 Human Action Recognition

To classify a single human action - from a handstand to knitting - with a reasonable level of accuracy, we must inspect spatial information from each frame, and temporal information across the entire video sequence.

We implement the dual-stream model from [13]; recognising human activity by fusing spatial and temporal predictions from the UCF101 video dataset presented in [1] (see Fig. 14 for example frames, dramatically deteriorating in quality as H.264 CRF value is increased). To train the temporal stream, we pass 20 frames randomly sampled from the pre-computed stack of optical flow images. Across both streams, we use hyper-parameters presented in [13], apart from fixed learning rate as per the chosen implementation [237].

Compression Rate	global ACC	mean ACC	mIoU
100% (JPEG Parameter 95)	0.910 ± 0.001	0.525 ± 0.012	0.446 ± 0.008
40% (JPEG Parameter 75)	0.908 ± 0.001	0.522 ± 0.011	0.442 ± 0.007
25% (JPEG Parameter 50)	0.904 ± 0.0005	0.516 ± 0.009	0.435 ± 0.005
15% (JPEG Parameter 15)	0.814 ± 0.004	0.459 ± 0.0008	0.338 ± 0.0016
10% (JPEG Parameter 10)	0.775 ± 0.017	0.423 ± 0.002	0.299 ± 0.003
7% (JPEG Parameter 5)	0.758 ± 0.024	0.358 ± 0.007	0.258 ± 0.008

(a) after testing a pre-trained SegNet model [10] on compressed imagery

Compression Rate	global ACC	mean ACC	mIoU
100% (JPEG Parameter 95)	0.910 ± 0.0010	0.525 ± 0.012	0.446 ± 0.008
40% (JPEG Parameter 75)	0.909 ± 0.0009	0.514 ± 0.008	0.440 ± 0.006
25% (JPEG Parameter 50)	0.908 ± 0.0005	0.511 ± 0.007	0.436 ± 0.005
15% (JPEG Parameter 15)	0.902 ± 0.0005	0.493 ± 0.002	0.419 ± 0.001
10% (JPEG Parameter 10)	0.896 ± 0.0005	0.493 ± 0.011	0.414 ± 0.007
7% (JPEG Parameter 5)	0.880 ± 0.0008	0.457 ± 0.009	0.382 ± 0.006

(b) after retraining a SegNet model [10] with compressed imagery

Table 3: **Segmentation:** Global accuracy, mean class accuracy and mIoU at varying compression rates. Compression rate indicates the approximate final file size compared to using JPEG Parameter 95.

Compression Rate	Abs. Rel.	Sq. Rel.	RMSE
100% (JPEG Parameter 95)	0.0120 ± 0.0010	0.0047 ± 0.0007	0.0646 ± 0.005
40% (JPEG Parameter 75)	0.0130 ± 0.0010	0.0048 ± 0.0008	0.0649 ± 0.005
25% (JPEG Parameter 50)	0.0135 ± 0.0010	0.0048 ± 0.0008	0.0652 ± 0.005
15% (JPEG Parameter 15)	0.0159 ± 0.0010	0.0050 ± 0.0009	0.0669 ± 0.006
10% (JPEG Parameter 10)	0.0204 ± 0.0009	0.0051 ± 0.0008	0.0679 ± 0.005
7% (JPEG Parameter 5)	0.0292 ± 0.0008	0.0067 ± 0.0006	0.0786 ± 0.003

(a) after testing a pre-trained GAN model for monocular depth estimation [226] on compressed imagery

Compression Rate	Abs. Rel.	Sq. Rel.	RMSE
100% (JPEG Parameter 95)	0.0120 ± 0.0010	0.0047 ± 0.00070	0.0646 ± 0.0050
40% (JPEG Parameter 75)	0.0111 ± 0.0002	0.0036 ± 0.00009	0.0564 ± 0.0006
25% (JPEG Parameter 50)	0.0107 ± 0.0003	0.0030 ± 0.00010	0.0514 ± 0.0010
15% (JPEG Parameter 15)	0.0121 ± 0.0002	0.0032 ± 0.00010	0.0540 ± 0.0010
10% (JPEG Parameter 10)	0.0149 ± 0.0004	0.0031 ± 0.00004	0.0525 ± 0.0005
7% (JPEG Parameter 5)	0.0156 ± 0.0004	0.0037 ± 0.00040	0.0581 ± 0.0030

(b) retraining a GAN model for monocular depth estimation [226] with compressed imagery

Table 4: **Depth Estimation:** Absolute Relative, Squared Relative, and Root Mean Squared Error at varying compression rates (lower, better)

Compression Rate	mAP
100% (JPEG Parameter 95)	0.704 ± 0.0005
40% (JPEG Parameter 75)	0.688 ± 0.0010
25% (JPEG Parameter 50)	0.669 ± 0.0020
15% (JPEG Parameter 15)	0.537 ± 0.0050
10% (JPEG Parameter 10)	0.434 ± 0.0060
7% (JPEG Parameter 5)	0.185 ± 0.0030

(a) after testing a pre-trained FasterRCNN model [11] on compressed imagery

Compression Rate	mAP
100% (JPEG Parameter 95)	0.704 ± 0.0005
40% (JPEG Parameter 75)	0.694 ± 0.0010
25% (JPEG Parameter 50)	0.696 ± 0.0030
15% (JPEG Parameter 15)	0.647 ± 0.0005
10% (JPEG Parameter 10)	0.627 ± 0.0005
7% (JPEG Parameter 5)	0.559 ± 0.0010

(b) retraining a FasterRCNN model [11] with compressed imagery

Table 5: **Object Detection:** Mean average precision at varying compression rates

3.3 Evaluation

In this section, we contrast the performance of the considered CNN architectures under their respective evaluation metrics before and after retraining. From this, we can determine how much we can safely compress the imagery while maintaining acceptable performance. We then propose possible explanations for the variations in resilience of the network architectures to image compression.

3.4 Semantic Segmentation

From results presented in Table 3 we can observe that the impact of lossy compression (Table 3a) is minimal, indicating high resilience to compression within the network. At the highest (most compressed) compression level, we see global accuracy reduce by 17%, down to 75.8%, while affording almost as little as 95% less storage cost on average per input image. However, at these heaviest compression rates, the compression artifacts introduced can lead to false labelling. This is particularly prominent where there are varying levels of lighting, affecting even plain roads (Fig. 10c). Subsequently, from Table 3b we can see that retraining the network further minimizes performance loss, especially minimizing false labelling of regions. At a JPEG compression level of 5, performance loss is reduced to 3.3%, resulting in global accuracy narrowly dropping below 0.9. Such resilience may stem from the up-sampling by the pooling layers within the decoder pipeline, which are innately capable of recovering information that has been lost during compression, but further investigation is left to future work.

3.4.1 Depth Estimation

Analyzing the results in Table 4, it is evident that lossy compression markedly diminishes root mean squared error (RMSE) performance of depth estimation when heavy compression rates are employed (Table 4a). At a JPEG compression level of 15, RMSE has not increased by more than 3.6%, but at a JPEG compression level of 10 and lower, performance begins to dramatically decline (in keeping with that of [54]). However, by retraining the network at the same compression level that is employed during testing (Table 4b), performance loss can be thoroughly constrained. Even at a JPEG compression level of 5, RMSE can be constrained to under 0.0600, improving performance by as much as 26% over the pre-trained network. Other performance measures demonstrate the same trend.

This performance is surprising: we might expect that RMSE would increase (thus lowering performance) after training on compressed imagery, since the GAN generates low quality imagery as the textures and features used to calculate depth estimation are lost, and is therefore unable to improve depth estimation performance. It is possible that it exceeds our expectation due to the encoder-decoder pipeline within the

Compression Rate	mAP
100% (JPEG Parameter 95)	0.730 ± 0.020
40% (JPEG Parameter 75)	0.707 ± 0.020
25% (JPEG Parameter 50)	0.670 ± 0.020
15% (JPEG Parameter 15)	0.419 ± 0.006
10% (JPEG Parameter 10)	0.332 ± 0.010
7% (JPEG Parameter 5)	0.100 ± 0.001

(a) after testing a pre-trained OpenPose model [12] on compressed imagery

Compression Rate	mAP
100% (JPEG Parameter 95)	0.730 ± 0.02
40% (JPEG Parameter 75)	0.734 ± 0.03
25% (JPEG Parameter 50)	0.707 ± 0.03
15% (JPEG Parameter 15)	0.681 ± 0.03
10% (JPEG Parameter 10)	0.621 ± 0.03
7% (JPEG Parameter 5)	0.428 ± 0.03

(b) after retraining an OpenPose model [12] with compressed imagery

Table 6: **Human Pose Estimation:** Mean average precision at varying compression rates

estimation process, which is also employed in the SegNet architecture, and thereby shares its compression resilience.

3.4.2 Object Detection

From Table 5, we can again discern that performance degrades rapidly at high lossy compression levels (JPEG compression level of 15 or less, see Table 5a). Applying a JPEG compression level of 15 leads to a 23.6% drop, down to mAP of 0.537, whilst a JPEG compression level of 5 causes mAP to drop by as much as 73.7%. Furthermore, with higher compression rates, fewer objects are detected, and their classification confidence also falls (Fig. 12c). Their classification accuracy remains unhindered, however. When the network is retrained on imagery lossily compressed at the same level, performance is noticeably improved (Table 5b). A significant performance drop only occurs at JPEG compression level of 5 to a JPEG compression level of 15. In fact, the retrained network is able to maintain an mAP above 0.6 even at a JPEG compression level of 10; reducing performance degradation to only 10.8%, while affording a lossy compression rate almost 10-fold higher in terms of reduced image storage requirements.

3.4.3 Human Pose Estimation

Results in Table 6 once again illustrate that lossy image compression (Table 6a) dramatically impacts performance at high rates. Similar to object detection, performance considerably lowers at 15% compression rate, in this case with performance falling by 42.6% to 0.419 mAP. Qualitatively, the network computes precisely located skeletal positions at higher compression rates, but detects and locates fewer joints (Fig. 13b). With high levels of compression (Fig. 13c), the false positive rate increases, and limbs are falsely detected and located. It is likely that optimizing the detection confidence threshold required of joints before computing their location, and thereby maximizing limb detection while minimizing false positives increases performance, especially during high compression. With a retrained network (Table 6b), a compression rate of 15% can be safely achieved before performance degradation exceeds 10%.

Compression Rate	Top-1 Spatial	Top-1 Motion	Top-1 Fusion
100% (H.264 CRF 23)	79.3022 ± 0.43	70.4898 ± 0.37	83.9585 ± 0.41
75% (H.264 CRF 25)	78.0768 ± 0.72	44.1825 ± 0.74	72.8730 ± 0.73
40% (H.264 CRF 30)	78.4779 ± 0.02	37.5198 ± 0.16	72.3729 ± 0.14
10% (H.264 CRF 40)	73.2753 ± 1.30	37.8465 ± 1.11	69.6203 ± 1.26
5% (H.264 CRF 50)	42.8359 ± 1.36	14.0867 ± 1.24	40.1377 ± 1.34

(a) after testing a pre-trained HAR model [13] on video data with varying H.264 CRF encoding values

Compression Rate	Top-1 Spatial	Top-1 Motion	Top-1 Fusion
100% (H.264 CRF 23)	79.3022 ± 0.43	70.4898 ± 0.37	83.9585 ± 0.41
75% (H.264 CRF 25)	79.1175 ± 0.56	39.2092 ± 0.51	72.2916 ± 0.53
40% (H.264 CRF 30)	78.6978 ± 0.21	33.9061 ± 0.41	70.9565 ± 0.38
10% (H.264 CRF 40)	75.1371 ± 0.81	10.2450 ± 0.99	66.2427 ± 0.88
5% (H.264 CRF 50)	61.9860 ± 0.53	7.7500 ± 1.02	55.5079 ± 0.72

(b) after retraining a HAR model [13] with on video data with varying H.264 CRF encoding values

Table 7: **Human Action Recognition:** Top-1 accuracy for each stream at varying compression rates

While impressive, the results are relatively insubstantial compared to those of other architectures, such as SegNet (Section 3.4, Table 3). The difference can perhaps be attributed to the double prediction task within the pose estimation network. Inaccuracies stemming from the lower quality images are not just propagated but multiplied through the network, as the architecture must simultaneously predict both detection confidence maps and the affinity fields for association encodings.

3.4.4 Human Action Recognition

From results presented in Table 7, it is evident that the impact of lossy compression (Table 7a) dramatically increases when we apply CRF factor 50. Conversely to all other examined architectures, we can see from Table 7b that retraining the network in fact *decreases* performance.

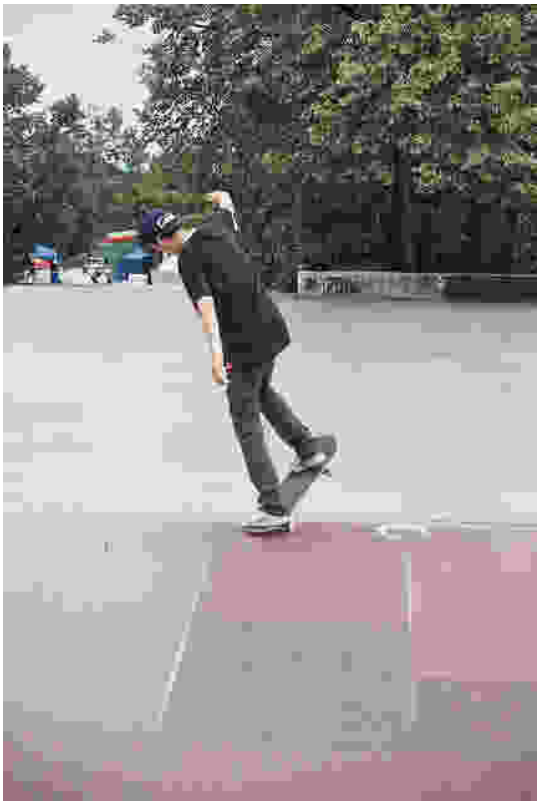
At first glance, we might expect similar performance to pose detection as with the two stream network for human action recognition, as the errors introduced by compression artifacts propagate through both streams in the network. However, the spatial and motion streams are not trained in tandem. Whilst the spatial stream remains resilient, once again due to the up-sampling within the architecture (Section 3.4), the motion stream is almost entirely unable to learn from compressed imagery. As such, retraining the network on compressed imagery in fact reduces overall performance (aside from when using CRF 50, as the spatial stream improvement outweighs the motion stream degradation). Future work may reveal whether better performance might be achieved by retraining just the spatial stream network on compressed imagery, and fusing its predictions with a motion stream trained only on uncompressed imagery.

3.4.5 Cross task analysis

Up until now, we have compared the impact of compression on the performance of models trained with and without compressed images. To form a better understanding of how different models perform we must investigate how different data and task spaces are affected by compression. For instance, it is easy for humans to estimate the pose for many of the images in the COCO training set, especially if there are only one or two people, even when highly compressed (Figure 15 (top)). More cluttered scenes present a more difficult task however, where people in the background are difficult to distinguish from each other, let alone their limbs. Facial details for the figure sitting down with a hat in the back in Figure 15 (bottom) are lost, and the bent right arm is very difficult to locate without access to the less compressed image for assistance. However, several examples also exist where it is easy for humans to estimate the human pose given contextual clues of the behaviour of the person, such as the people *sitting* in Figure 15 (middle). However, the discontinuities introduced by JPEG break the topology of the skeleton in this instance and the left arm of the person sitting on the right of the image can be missed by a keypoint detection model. The human skeleton prior encoded into the end to end OpenPose algorithm pipeline encourages correct association of limbs to people in cluttered scenes, but does not help the model detect partially hidden limbs. Predicting the bounding box of such a person however remains very possible for a detection model.

That does not mean to say that bounding box information can be easily recovered in the majority of situations however. In many cluttered images, the JPEG artifacts produce unclear boundaries within images, such as the height of the chair in the left of Figure 16 and its legs. The important features to succesfully classify and locate objects are removed due to JPEG artifacts. This effect can be demonstrated via the use of AblationCAM [238] to visualise what image regions are most paid attention to during inference. At low compression rates, the model attends to the relevant regions to locate the chair and table in the image, but is unable to extract relevant features from the image at high compression (Figure 17). In the same regard, optical flow is adversely affected by loss of such object boundary information. In contrast, segmentation and depth estimation is less sensitive to the aforementioned discontinuities that can affect pose estimation, or the lost boundary information. Even under high compression rates, instance boundaries are more disinguishable (Figure 18). JPEG disproportionately affects high frequency fidelity information. Object detection, pose estimation, and human action recognition are more affected than scene segmentation and depth estimation, which are less reliant on high frequency information. As compression increases, the discriminability between objects, actions and limbs reduces as images are represented with a smaller set of quantized vectors as a proxy for local image representation. Therefore, discriminability of localized information is reduced and tasks that rely on those details, such as position of joints or detail on objects, begin to fail first.

(a) JPEG compression level: 95



(b) JPEG compression level: 5

(c) JPEG compression level: 95

(d) JPEG compression level: 5

(e) JPEG compression level: 95

(f) JPEG compression level: 5

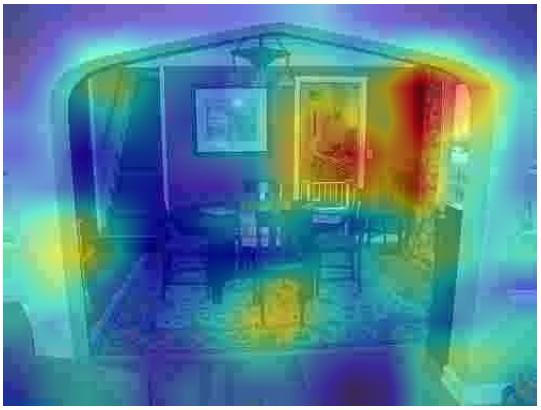
Figure 15: Qualitative examples of images in the OpenPose validation set under low (left) and high (right) compression.

(a) JPEG compression level: 95

(b) JPEG compression level: 5

Figure 16: Qualitative examples of images in the VOC dataset under low (left) and high (right) compression.

(a) JPEG compression level: 95



(b) JPEG compression level: 5

Figure 17: Visualisation of regions attended by a pretrained model in the VOC dataset under low (left) and high (right) compression.

(a) JPEG compression level: 95

(b) JPEG compression level: 5

Figure 18: Qualitative examples of images in the Cityscape validation set under low (left) and high (right) compression.

3.5 Racial Bias within Compression

In this section, we move away from considering overall network performance, but instead towards non-uniform performance across the network, and corresponding racial bias. [4]

During lossy compression, a given image colour space assumes $YCrCb$ format. Considering the relative insensitivity of the human eye to colour changes compared to brightness, information can be more heavily discarded from Cr and Cb channels than Y (luminance) channels. The most common chroma subsampling methods are presented in Fig. 19. Indeed, the JPEG algorithm by default applies 4:2:0 chroma subsampling alongside additional information removal steps. Given this colour space reduction, it is unsurprising that racial bias is introduced within compression given differences in skin tone.

Let us consider the uncompressed training data distribution $\mathbf{x} \sim p_{\mathrm{u}}$ and compressed training data distribution $\mathbf{x} \sim p_{\mathrm{c}}$. Assuming independence between channels, we can compute the entropy of an image from the sum of the entropy of its channels:

$$H(\mathbf{x}) = H(\mathbf{x}_Y) + H(\mathbf{x}_{Cr}) + H(\mathbf{x}_{Cb}), \tag{6}$$

where $H(\mathbf{x})$ corresponds to the entropy or information in a given image, and $H(\mathbf{x}_Y)$, $H(\mathbf{x}_{Cr})$ and $H(\mathbf{x}_{Cb})$ correspond to the entropy of the Y, Cr, and Cb channels respectively. Further, we can define $p_{\mathrm{c}} = \Psi(p_{\mathrm{u}})$, where Ψ denotes all compression functions. Indeed, for lossless compression functions, we have that $H(\mathbf{x}) = H(\Psi_{\mathrm{lossless}}(\mathbf{x}))$.

However, chroma subsampling reduces the number of samples in the spatial dimension by a factor of either 2 (4:2:2) or 4 (4:2:0) such that the entropy of the Cr and Cb channels is thus reduced under chroma subsampling (Equation 7 and 8):

$$H(\Psi_{\mathrm{JPEG}}(\mathbf{x}_{Cr})) < H(\mathbf{x}_{Cr}), \tag{7}$$

$$H(\Psi_{\mathrm{JPEG}}(\mathbf{x}_{Cb})) < H(\mathbf{x}_{Cb}), \tag{8}$$

where Ψ_{JPEG} denotes the JPEG compression function utilizing chroma subsampling. Therefore, from Equation 6, overall information in an image is reduced when that image is compressed with chroma subsampling within JPEG (Equation 9)

$$H(\Psi_{\mathrm{JPEG}}(\mathbf{x})) < H(\mathbf{x}). \tag{9}$$

We can reformulate this behaviour under the Nyquist theorem [239, 240] which states in general that a

[4] This section was produced in collaboration with Seyma Yucer

continuous signal of frequency B can be perfectly reconstructed with a discrete sampling rate greater than or equal to $2B$. In the context of images as 2D signals, for a line of pixels of length $2b$ pixels the highest spatial frequency that can be recovered is a spatial frequency of b. By halving the number of pixels in the spatial dimension, we halve the spatial frequency that can be reconstructed and hence the Nyquist rate in either the x or y dimension (4:2:2), or both (4:2:0), is reduced by a factor of 2 in the $\mathbf{x} \sim p_{\mathrm{Cr_c}}$ and $\mathbf{x} \sim p_{\mathrm{Cb_c}}$ distributions under subsampling conditions. If we assume a uniform distribution of relevant face recognition features, it is clear that the reconstructable frequencies that relate to racial phenotypes is similarly halved under the subsampling conditions. Of course, whilst we cannot calculate which spatial frequencies are relevant to face recognition performance, it has been shown that progressively reducing the high spatial frequencies in an image via blurring increases the prevalence of racial bias on such tasks [41]. From this, we can conclude that chroma subsampling reduces the high spatial frequency information across two subdistributions of the image ($\mathbf{x} \sim p_{\mathrm{Cr_c}}$ and $\mathbf{x} \sim p_{\mathrm{Cb_c}}$). As a result, JPEG compression negatively impacts fairness in face recognition tasks.

To evaluate the impact of compression on racial bias, we deploy ArcFace [241] with a ResNet50 [18] embedding network. We train the network with the BUPT-Balanced benchmark dataset [242] that consists of 28000 faces distributed evenly between African, Asian, Indian, and Caucasian ethnic groups. Using RFW [243] test dataset, we consider both network accuracy and false matching rate between selected phenotype pairings [16] (see Fig. 20 axes). A high false matching rate between a given phenotype pairing suggests the facial recognition network is more likely to mismatch a given face from one of the phenotypes in the pair with the other. We once again compare the performance of ArcFace on images compressed with JPEG quality parameter in the set {5,10,15,50,75,95}, trained with and without data from the same compression levels. In addition, we compare performance on data with and without chroma subsampling.

3.5.1 Racial Bias Results and Discussion

In general, ArcFace performance deteriorates significantly only with high compression (JPEG compression levels of 15 and below), aligning with earlier findings (Section 3.3). However, ArcFace performance across different racial groups increases non-uniformly with the introduction of chroma subsampling within JPEG compression.

From Table 8, it is clear to what extent the ArcFace network face recognition accuracy deteriorates during inference stage using compressed data. Without lossy compression, ArcFace achieves 94.76% accuracy. Minimal performance degredataion is observed with compression levels higher than 15 (not shown). By compression level 15, accuracy drops by 7.9% down to 87.31%. At compression level 5, accuracy is only 66.47%. However, by retraining the network on compressed image data, ArcFace displays significantly

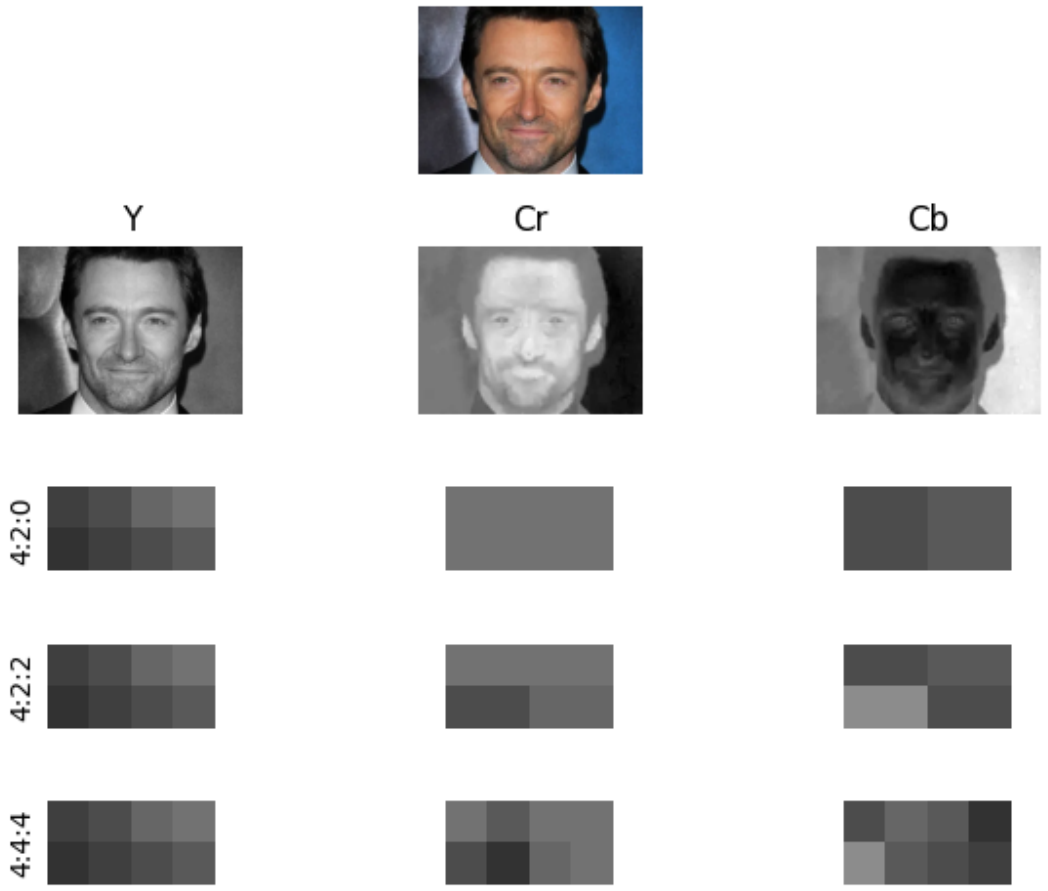


Figure 19: Chroma subsampling operation on different rates (4:2:0, 4:2:2, 4:4:4). Subsampling rate determines how many pixel values will be shared within a block. We display 4:2:2 with horizontal sampling but vertical sampling is sometimes used.

Attribute Name	Non-Compressed Training Set JPEG Compression Level			Compressed Training Set JPEG Compression Level			Original
	15	10	5	15	10	5	
Curly Hair	82.37	75.80	59.53	87.20	82.90	73.27	93.15
Full Lips	83.55	77.03	61.37	87.97	83.62	75.30	93.38
Monolid Eye	83.43	77.28	63.18	87.62	85.10	76.95	93.30
Type 5	85.98	80.17	60.32	90.22	87.03	76.97	94.85
Type 6	86.55	79.35	61.75	90.02	86.20	77.72	94.82
Black Hair	85.13	79.97	65.83	89.55	86.87	77.92	93.73
Wide Nose	85.53	79.97	63.15	89.57	86.78	78.33	93.98
Other Eye	86.65	81.10	65.28	89.57	87.43	78.55	94.38
Type 4	87.72	83.47	67.28	89.67	87.45	79.23	94.07
Type 1	86.88	84.72	72.43	89.87	88.21	79.57	92.86
Straight Hair	86.70	81.98	66.15	89.43	86.28	79.65	94.12
Narrow Nose	86.30	80.07	66.73	89.63	87.20	79.77	94.43
Type 3	86.07	81.03	67.05	89.48	86.80	79.93	93.98
Small Lips	87.28	82.03	67.53	90.63	87.97	81.22	94.37
Wavy Hair	89.05	84.63	69.53	92.17	89.33	82.73	95.83
Brown Hair	88.40	83.33	67.32	91.85	89.03	82.80	95.15
Bald Hair	90.43	85.93	67.62	93.07	90.37	83.13	96.55
Red Hair	90.57	84.97	71.20	92.49	89.98	84.89	96.91
Type 2	89.98	85.98	68.45	94.27	91.58	85.93	96.33
Gray Hair	92.47	88.83	72.60	94.35	91.93	86.75	96.55
Blonde Hair	92.50	88.52	71.55	94.83	93.40	87.85	97.15
Mean Accuracy	87.31	82.20	66.47	90.64	87.88	80.40	94.76
STD	2.76	3.58	3.85	2.18	2.61	3.81	1.31

Table 8: ArcFace accuracy on RFW benchmark test dataset using uncompressed (left) and compressed (right) training image data. Attribute-based pairings are selected as per [16].

higher resilience to compression. Performance only drops by 4.3% at compression level 15, and by 15.2% at compression level 5 compared to 29.9% without retraining.

More interestingly however, standard deviation (as a measure of non-uniform performance and hence bias) significantly increases with both compressed and non-compressed training data. Indeed, performance degradation is most apparent at heavy compression for Type 5 and Type 6 (darker skin tone), Full Lip, and Monolid Eye phenotype categories. For instance, with JPEG compression level 5, ArcFace performance on Type 5 skin tone decreases by just over 36% to only 60.32%. In contrast, for lighter skin tones, performance degradation is comparatively smaller (the lowest performance drop across skin tone types 1-3 can be observed for type 2 at 28.9%, down to 68.45% accuracy). Training on compressed data has minimal impact on standard deviation and thus ameliorating racial bias.

The same pattern can be observed by directly comparing JPEG compression with and without chroma subsampling (Fig. 20, 21). Values represent $FMR_{original} - FMR_5$, i.e. the FMR difference between original images and those compressed with JPEG quality parameter 5. Values closer to 0 indicate better performance (fewer mismatches). Equivalently, dark red cells indicate poorest performance while darkest

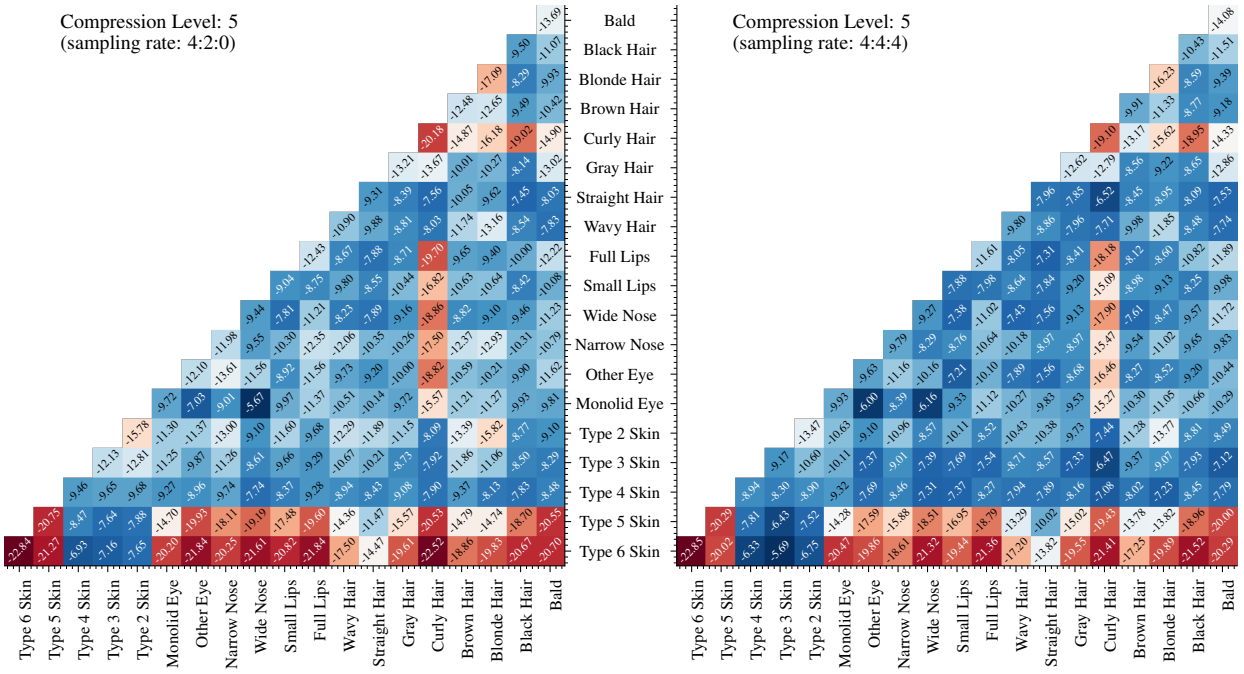


Figure 20: ArcFace performance tested with compressed (JPEG compression level 5) images in RFW test dataset, trained with uncompressed images from BUPT (balanced) dataset

blue cells indicate strongest performance. Training ArcFace with non-compressed data yields distinctly non-uniform performance with respect to FMR of cross-attribute based pairings (Fig. 20). Under chroma subsampling 4:2:0, the worst performance can be observed for mismatches between Type 6 skin tones (-22.84 FMR). By removing chroma subsampling however, ArcFace generally achieves superior FMR. Type 5 $\leftrightarrow$ Type 6 pair increases from -21.27 to -20.02 FMR, while Curly Hair $\leftrightarrow$ Type 5 pair increases from -20.53 to -19.43 FMR, for instance. Training ArcFace with compressed data once again partially recovers FMR performance, evident in that there are more blue / dark blue cells and fewer dark red cells (Fig. 21). However, retraining the network does little by way of minimizing the non-uniformity between phenotypes. There is still a disproportionately higher FMR for racial phenotypes with dark skin tones compared to lighter tones. FMR for Type 5 $\leftrightarrow$ Type 6 decreases to -6.15 FMR, over 6 $\times$ worse than FMR for Type 3 $\leftrightarrow$ Type 6 (Fig. 21).

3.6 Conclusion

This chapter investigates the impact of lossy image compression with respect to both racial bias and a multitude of existing deep CNN architectures. We consider how much compression can be achieved while maintaining acceptable CNN model performance, and to what extent performance degradation and racial bias can be ameliorated by retraining the networks with compressed imagery.

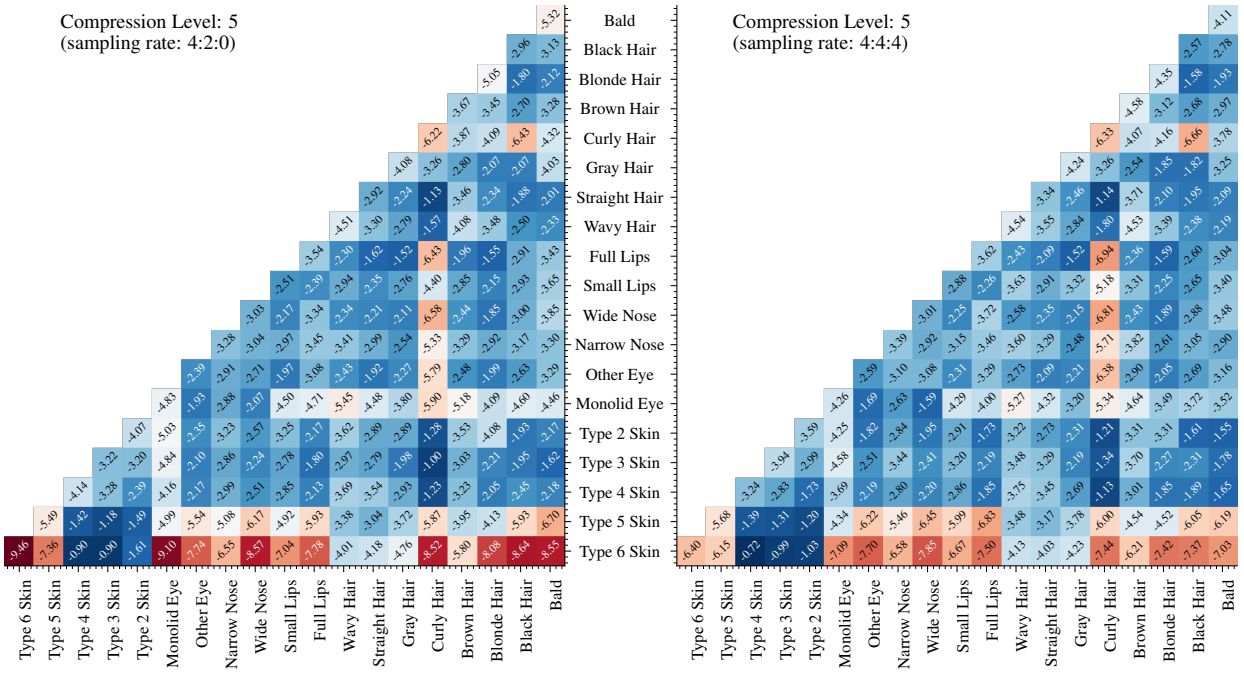


Figure 21: ArcFace performance tested with compressed (JPEG compression level 5) images in RFW test dataset, retrained with compressed (JPEG compression level 5) images from BUPT (balanced) dataset

Across all challenges, retraining the network on compressed imagery recovers overall performance to a certain degree. This chapter brings to attention in particular, however, that in very prevalent and so far unexamined network architectures, we can afford to compress imagery at extremely high rates. Segmentation and depth estimation in particular demonstrate resilience against even very significant compression, both by employing an encoder-decoder pipeline. By using retrained models, compression can safely reach as high as 85% across all domains. In doing so, current storage costs can be markedly diminished before performance is noticeably impacted. Hyper parameter optimization of the retrained model can assumedly capitalize on this even further, and in certain domains, such as segmentation, we can already afford to reduce to a twentieth of the original storage cost. It should be noted however, that even a 1 or 2% performance loss may be unacceptable in safety critical operations, such as depth estimation for vehicular visual odometry.

However, compression is not without its drawbacks, and introduces non-uniform performance degradation, negatively impacting racial groups in the context of facial recognition. Retraining the network only partially limits this behaviour, and does little to reduce any performance degredation that is exacerbated for darker skin tone, monolid eye, wide nose, full lips, and curly hair phenotypes. Increasing retention of colour information via removal of chroma subsampling during compression ameliorates this behaviour up to a point.

Nevertheless, we can suggest that lossy image compression is potentially viable as a data augmentation

technique within RCNN [11] and pose estimation [12] architectures, which receive only mild performance degradation. Networks employing an encoder-decoder architecture (SegNet [10], GAN [226]) would only notably benefit from very significant levels of image compression for data augmentation. However, human action recognition networks, or sub-networks in the case of the two stream approach [13], that consider motion input will not readily benefit from image compression as a data augmentation technique, since they appear unable to learn under such training conditions.

Future work will investigate whether performance is improved by retraining the network with more heavily or lightly compressed imagery than at testing, or even a variety of compression levels. Furthermore, evaluating performance of compressed networks such as MobileNet[43] against compressed imagery would be pertinent, as such light network architectures are prevalent amidst compressed imagery application domains.

4

Dynamic Data Selection: Minimizing Training Costs for Neural Architecture Search within Image Classification

4.1 Introduction

Following the emergence of big data and the ever-increasing public availability of datasets, each with tens of thousands of data points each, research within the deep learning domain is accelerating [22]. Consequently, there are two key factors that need to be addressed. Firstly, the process by which we present data to the deep learning model is paramount; it is not uncommon for models to be trained for thousands of epochs, and thus any superfluous data within the dataset will have a magnified negative impact on training speed. To address this, we propose to utilize a combination of hard example mining [79], curriculum learning [80] and self-paced learning [82] described in Chapter 2.

The second challenge arising from data accessibility is the evolution of the architecture search space. As research within the domain continues, newer, and often more complex network architectures are presented. To overcome this notion, Neural Architecture Search (NAS) has emerged, which automatically traverses the architecture search space for a given task, and generates models that are competitive alongside hand-crafted state-of-the-art models. In many cases, NAS offers several key advantages over conventional hand crafted neural networks, including generating smaller models [154] with lower latency [168, 171, 173]. We can divide the NAS domain into evolutionary, reinforcement-learning, prediction-based and gradient-based NAS frameworks, of which this chapter primarily considers the latter.

As such, we propose a strategy that incorporates a novel combined hard example mining and curriculum learning approach to enable Dynamic Data Selection (DDS) within a generalised NAS framework, denoted DDS-NAS. DDS-NAS can thus be used in conjunction with existing NAS frameworks to generate small or low latency models, retaining the key advantages that NAS possesses over conventional hand crafting of

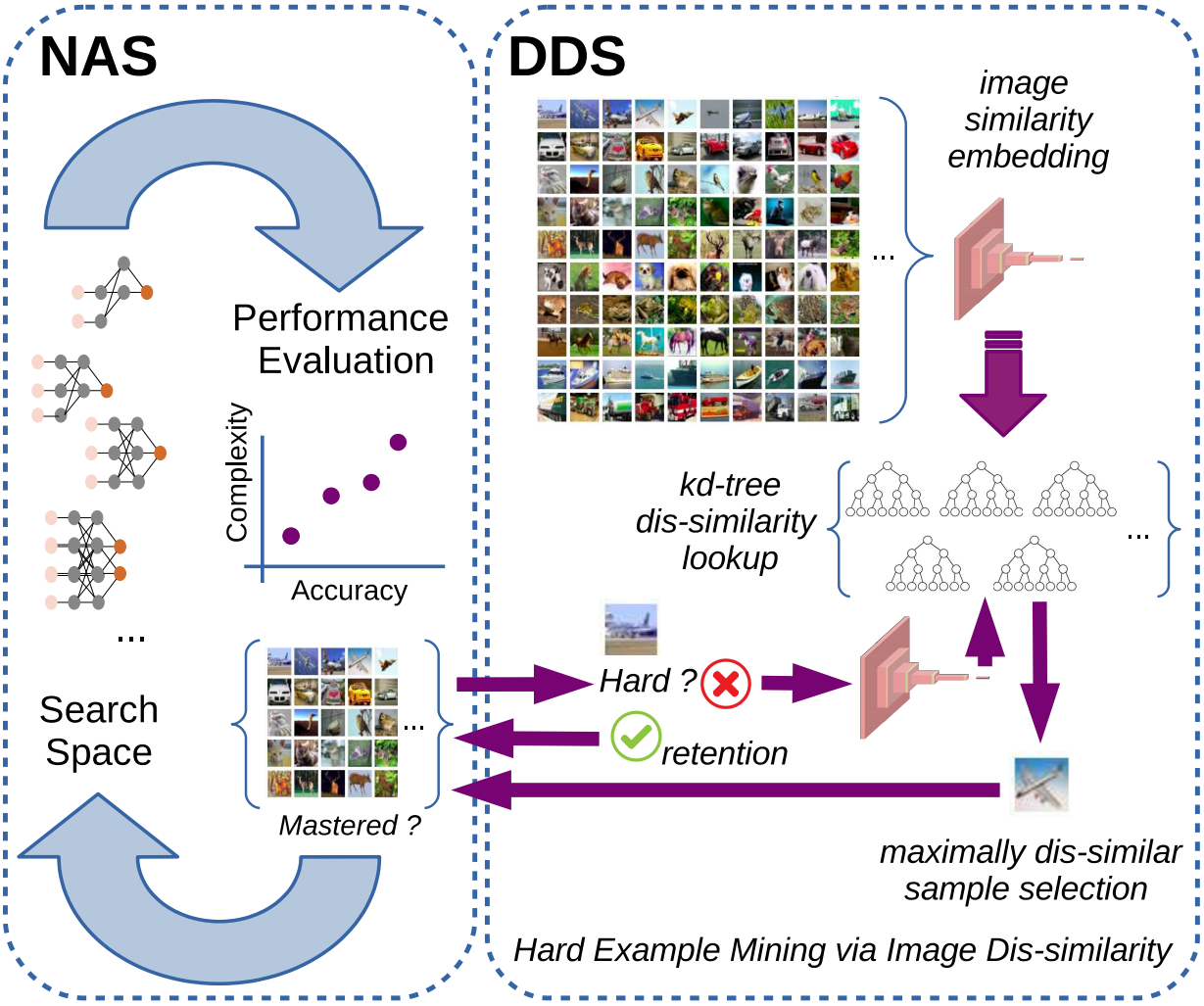


Figure 22: Overview of the DDS-NAS search phase. After a given training iteration, we determine whether a sufficient percentage of the data in the current subset was correctly classified, according to some a priori *mastery* threshold. If the subset has been mastered, we reformulate it dynamically. *Hard* images in the current subset are retained, according to some a priori hardness threshold, while easy images are replaced with the most different image from the same class. To determine the most different image, we employ an (approximate) furthest-neighbour *kd*-tree whereby each image is represented by the auto-encoded representation of its features within the latent space.

neural network architectures. By using image similarity as a proxy metric for image difficulty (on an easy to hard performance axis), we can select *hard* images for processing within a given NAS training iteration in logarithmic time, without compromising image diversity (Fig. 22). This process allows us to significantly improve the NAS search phase speed. Whilst this thesis specifically addresses image datasets, there is no reason not to apply identical techniques to other application domains such natural language processing (NLP).

Furthermore, we identify a new perspective to address and ameliorate the 'optimization gap' [134, 136] prevalent within gradient-based NAS literature, wherein the performance difference between the NAS model (super-network) and searched network architectures results in a performance drop.

4.2 Proposed Approach

In this section, we detail the process by which our proposed DDS-NAS training strategy dynamically samples the dataset in an online fashion within the NAS cycle (Figure 22). DDS-NAS is subsequently deployed across three leading contemporary NAS frameworks (DARTS [53], P-DARTS [108], TAS [154]).

Firstly, we define some key terms that we subsequently refer to throughout our discussion:

- *hard*: a given example within the dataset at the current NAS training cycle iteration is defined as being *hard* if the output of the current model correlates poorly with the ground truth label for this example and hence contributes significantly to the current loss value for the model (i.e. it is either mis-classified or classified with a low confidence score in the context of image classification).
- *easy*: as the converse of *hard* where for a given example the output of the current model correlates strongly with the ground truth label for this example and hence contributes less significantly to the current loss value for the model (i.e. correctly classified with a high confidence score in the context of image classification).
- *mastery*: a measure of when a given *a priori* performance threshold is reached on the current data subset such that the number of *easy* examples in the dataset is high with regard to the current model.

4.2.1 Curriculum Learning within NAS

To formulate an unbiased subset of the global dataset, we use a Hard Example Mining process detailed in Section 4.2.2. At every training iteration within the NAS search phase, we present such a subset to the NAS model. Following the success of [88], we in fact present the same subset until it has been *mastered*, according to some *a priori* mastery threshold (see Section 4.3.1). Only when the NAS model masters a subset do we sample a new set of examples from the global dataset. If the *mastery* threshold is very low, this subset

of data will change often. If the mastery threshold is very high, a given subset is presented to the NAS model for several successive iterations, and a smaller portion of the global dataset is sampled throughout the entire training process. Akin to the restriction with P-DARTS [108] whereby only network parameters (i.e. weights) are updated and not architectural parameters within the first 10 training epochs, we similarly restrict DDS-NAS from resampling the dataset in this way for the first 10 epochs of NAS training.

4.2.2 Dynamic Data Selection

In order to both improve convergence rate (and as a byproduct minimize the data subset used in each NAS iteration), without performance degradation, and facilitate efficient inter-iteration dataset re-sampling we require a low-overhead process by which we can dynamically select new data examples.

From the initial NAS training iteration, and the immediate subsequent iterations thereafter, model performance can be considered near-random[5]. As such, we necessarily depend upon a re-sampling process independent of model performance, and hence propose the use of dataset example similarity as an alternative measure to relative *hard*-ness between samples. The intuition is that a model will perform poorly on examples with greater dissimilarity to those upon which it has already been trained.

Given the need to perform efficient *one-to-many* feature distance comparisons via an online approach, we construct a series of efficient furthest-neighbour *kd*-tree structures from the chosen N-dimensional feature representations of each example in our global dataset. In order to maintain a balanced data subset in the presence of dynamic re-selection we construct one such *kd*-tree structure per class label in the dataset, resulting in m trees for m dataset classes. In this way we can facilitate *like-for-like* class-aware resampling and hence maintain dataset balance throughout the NAS training cycle.

In order to enable efficient look-up within our *kd*-tree structure, we require a sufficiently low dimension N of our feature representation such that the approximate furthest neighbour algorithm does not collapse [245]. As the dimensionality of image data is high (i.e. $N = 28 \times 28$ in case of MNIST, and larger for more complex datasets), we instead propose using an additional autoencoder architecture to construct an image similarity embedding with a much lower dimension ($N = 8$ for easier MNIST and Fashion-MNIST datasets, $N = 32$ for CIFAR-10).

In general, we find that contemporary state-of-the-art autoencoder architectures [246, 247, 248] employ skip-connections between the encoder and decoder sub-networks to facilitate improved image reconstruction. However, in this instance, such skip connections are detrimental to the performance of the encoder network in terms of constructing an encoding at the bottle-neck of the encoder-decoder architecture (our embedding) that maximally captures the highest level of feature detail within itself. On this basis, we employ the

[5]noting that Deep Image Priors [244] indicate that untrained model performance in fact correlates to architecture design.

proven autoencoder architecture from GANomaly [224] on the basis it is one of the most successful encoder-decoder architectures employed for encoded image discrimination, predating the wider move to the use of skip-connections in the field [247].

We require that the use of this encoder architecture results in a compact feature embedding that retains the property of spatial similarity such that similar images have similar embeddings within the latent space and vice versa. Of course, this property must not come at the expense of image reconstructability. Otherwise, we cannot be confident that a given embedding represents a given image. In other words, there would be no correlation between embedding space dissimilarity and image space dissimilarity. Given reconstructability but without similar images clustering within the embedding space, we cannot guarantee that the correlation is *strong*. [6]

To enforce these properties, we discovered that contractive loss [14] is sufficient for easier datasets, while harder datasets require a combined triplet margin ranking loss with MSE reconstruction loss, weighted via Kendall Loss [252]. Subsequently, we can thus order images by their dissimilarity within our furthest neighbour *kd*-tree structures. We provide a comprehensive overview of the impact of autoencoder training strategies in Section 4.2.2.1.

During a given NAS training iteration, we measure the *hard*-ness of a each example image in the current data subset based on cross-entropy loss following our earlier definition of *hard* and *easy* examples. To subsequently update our data subset in a dynamic manner, we first retain the images that are *hard* when averaged across the most recent epochs, according to some *a priori hard*-ness threshold (see Section 4.3.1). Secondly, by selecting the *kd*-tree from our set that is associated to the class label of each image in the current data subset that is below the *hard*-ness threshold (i.e. the *easy* images), we can then identify the most dissimilar image of the same class in the global training set in $O(log(n))$ time and use this to replace the *easy* image within the data subset. This dynamically updated training data subset will then be used for the next NAS training iteration.

Our overall pipeline is presented as follows: once the previous data subset of data has been *mastered* by iterative NAS training, we dynamically formulate a new balanced subset of the global training dataset based on (a) the retention of images that are considered *hard*, and (b) the replacement of images that are considered *easy* with dissimilar images of the same class to retain dataset balance (Fig. 22).

[6] We use the commonly accepted terminology 'cluster' to refer to the grouping and separation of points, as can be found in the literature [249, 250, 241], and according to the dictionary of computer vision [251].

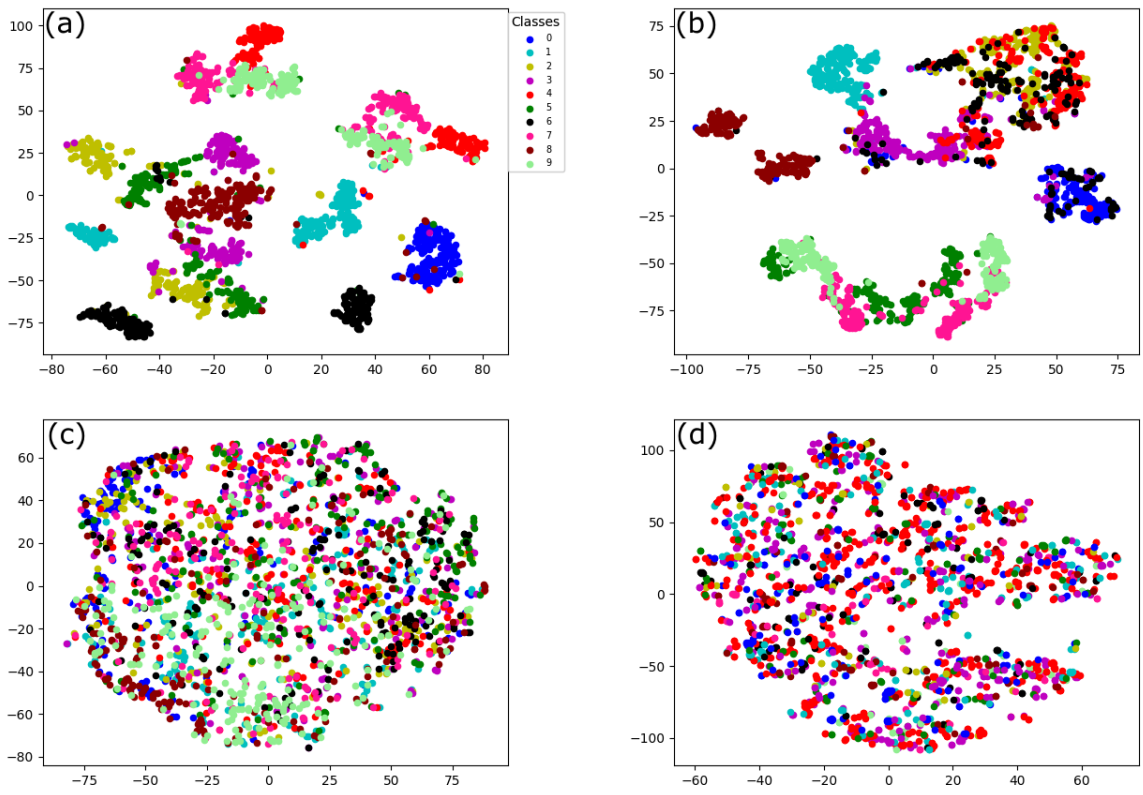


Figure 23: TSNE visualization of clustering of autoencoded image feature representation within latent space using contractive loss [14]. Our autoencoder preserves the property that similar images have similar encodings for MNIST (a) and Fashion-MNIST (b). Our autoencoder is unable to achieve the required clustering capability for CIFAR-10. Moreover, our compact embedding is unsuitable for fine-grained image classification such as FGVC-Aircraft (d), which is a known limitation of autoencoders.

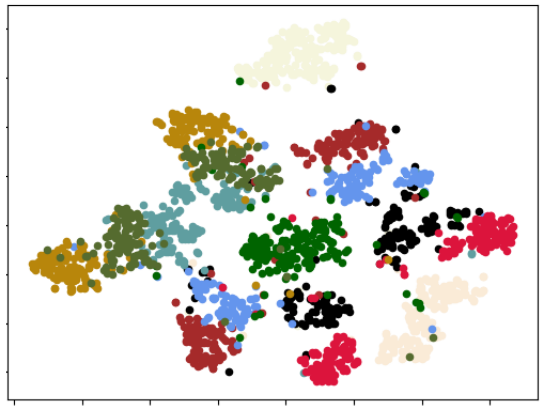
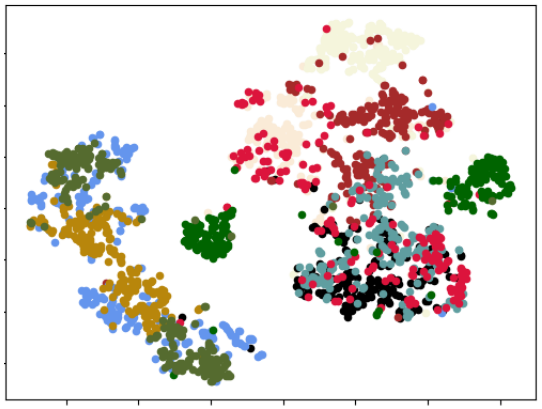
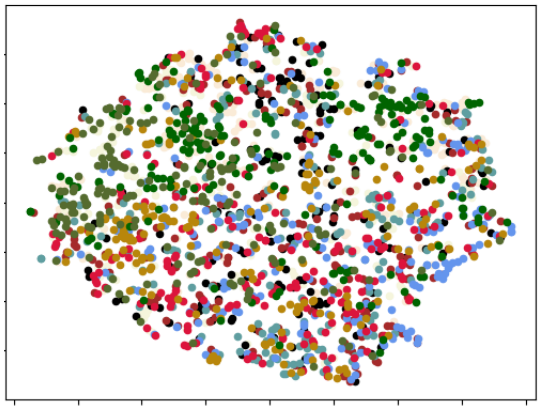
Config-uration	MNIST	Fashion-MNIST	CIFAR-10
Variational Autoencoder			

Table 9: Clustering capability of a Variational Autoencoder with various image datasets.

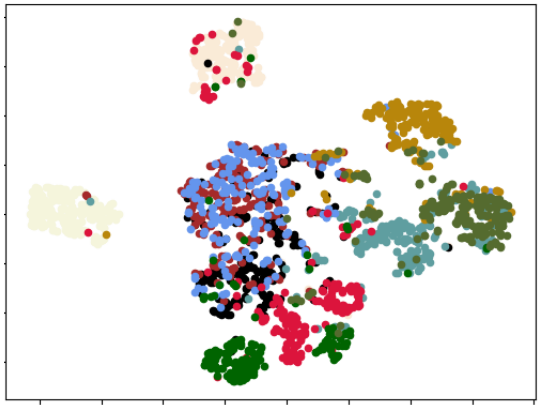
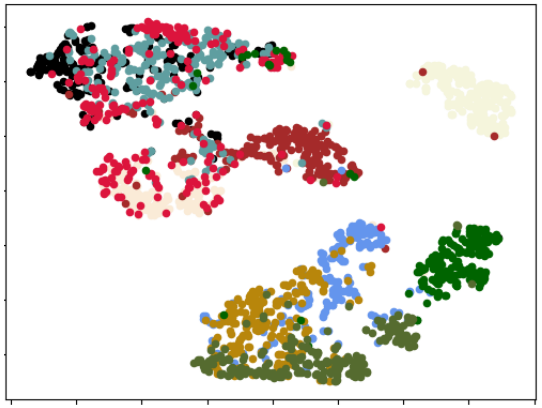
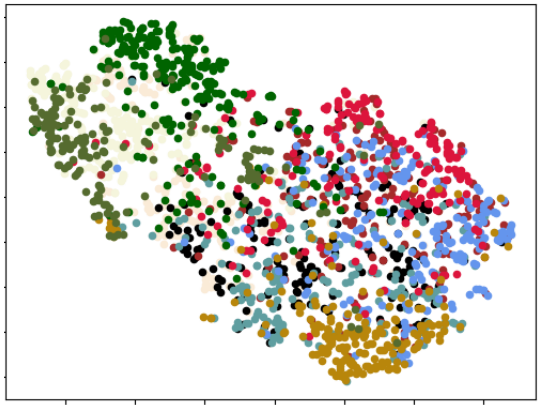
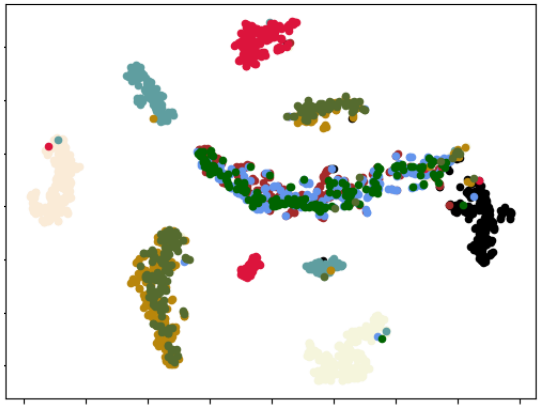
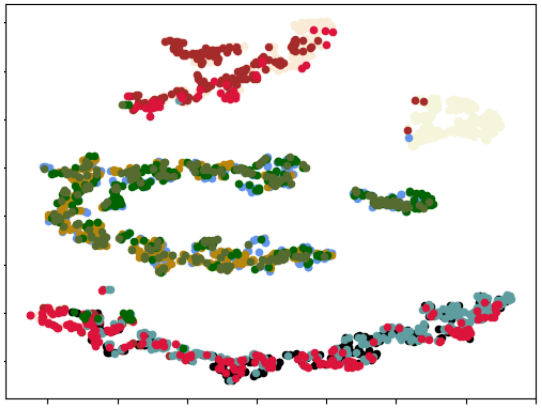
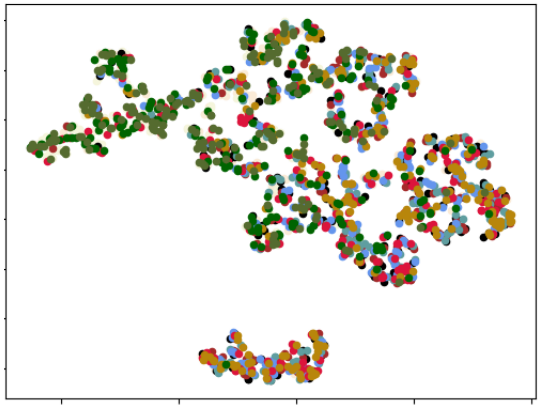
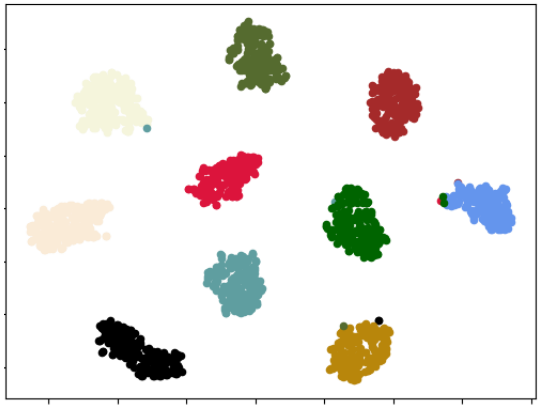
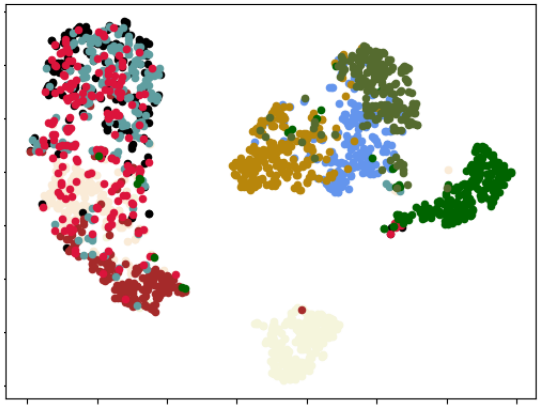
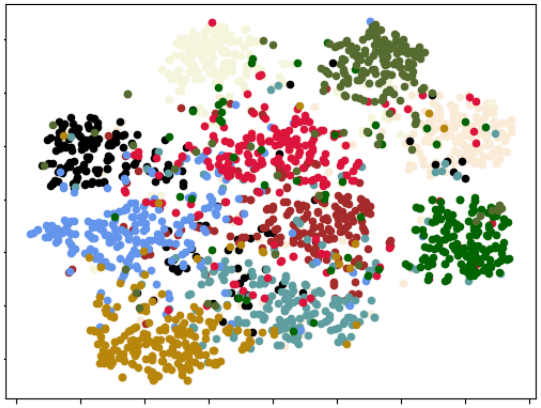
Config-uration	MNIST	Fashion-MNIST	CIFAR-10
Triplet Ranking Loss			
Cosine Embedding Ranking Loss			
Circle Ranking Loss			

Table 10: Clustering capability of tripet, cosine, and circle ranking loss, with various image datasets.

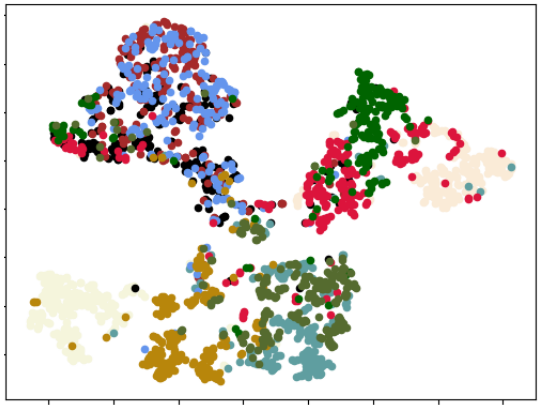
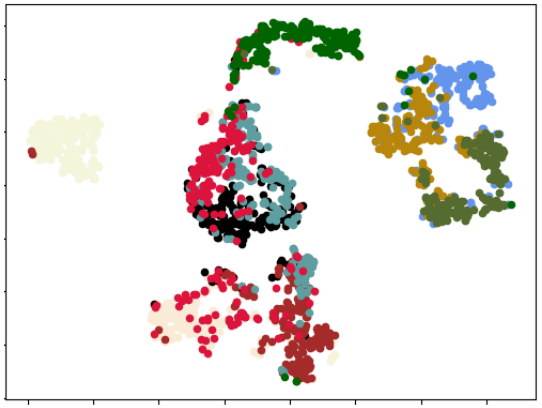
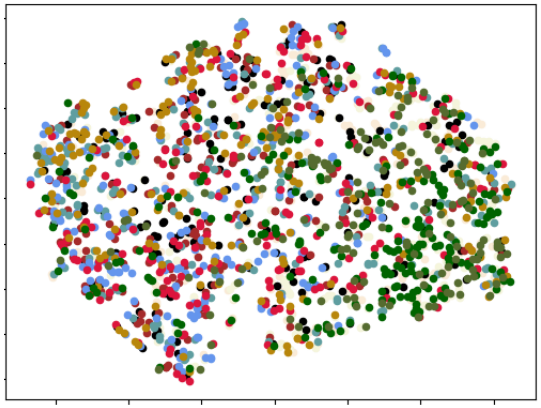
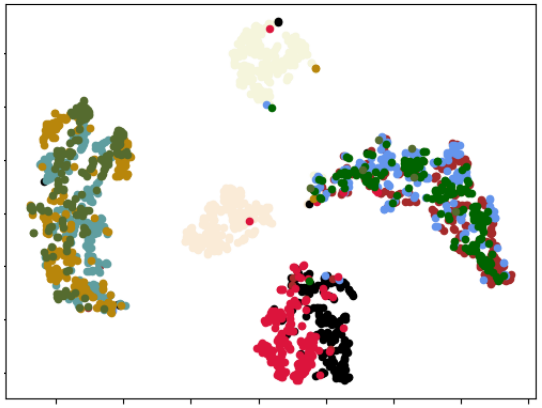
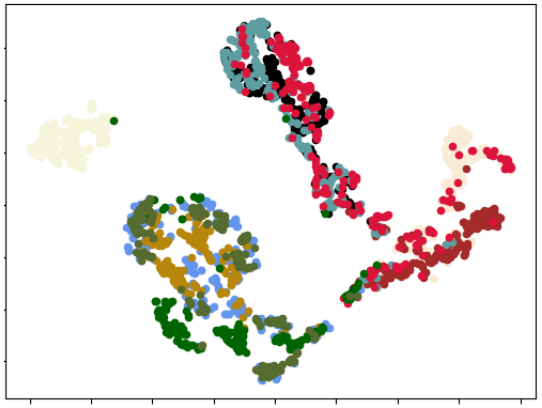
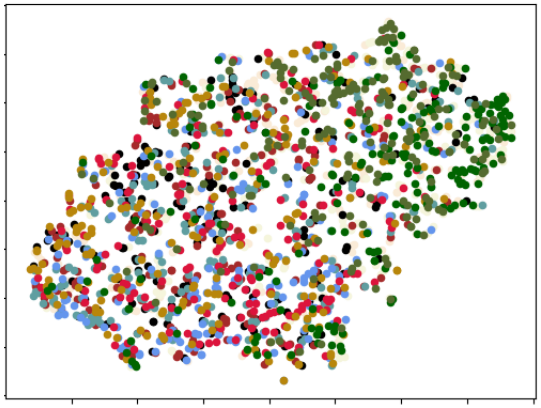
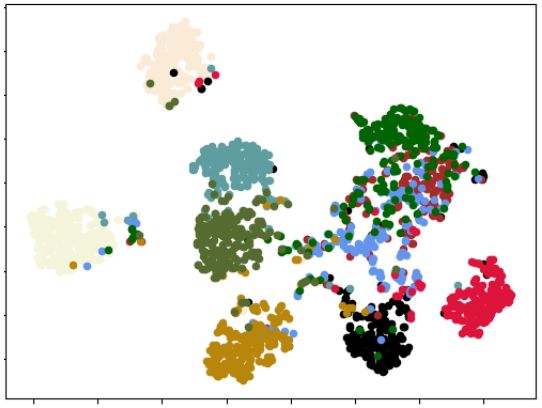
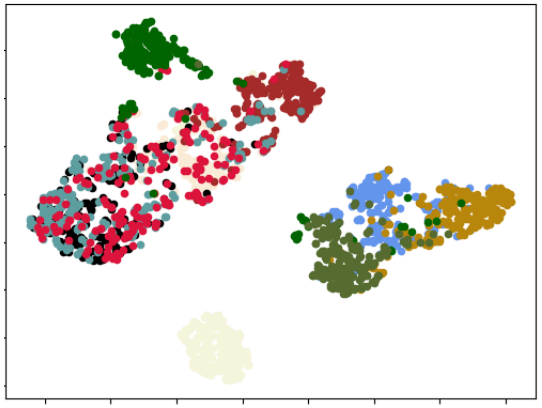
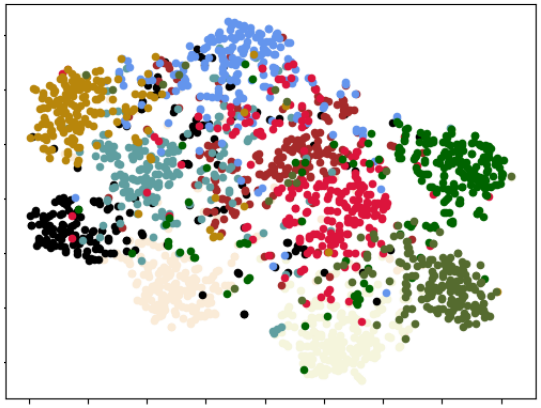
Config-uration	MNIST	Fashion-MNIST	CIFAR-10
Combined Triplet Ranking Loss and MSE Reconstruction Loss			
Combined Cosine Ranking Loss and MSE Reconstruction Loss			
Combined Circle Ranking Loss and MSE Reconstruction Loss			

Table 11: Clustering capability of tripet, cosine, and circle ranking loss, combined with MSE loss, with various image datasets.

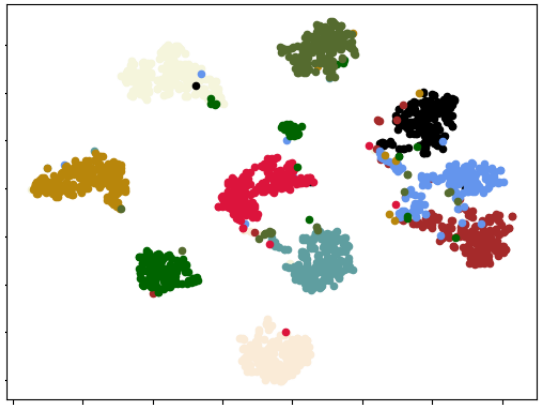
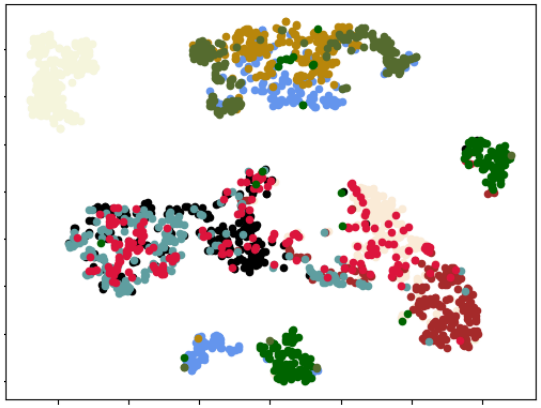
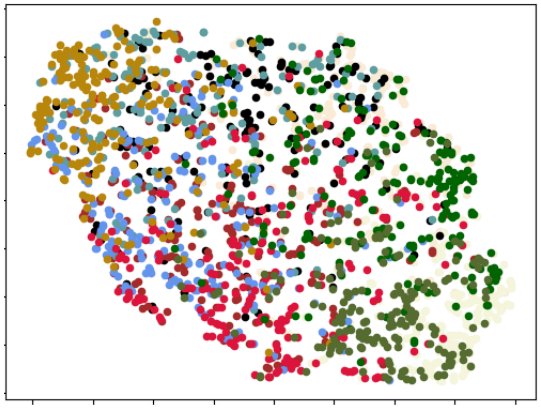
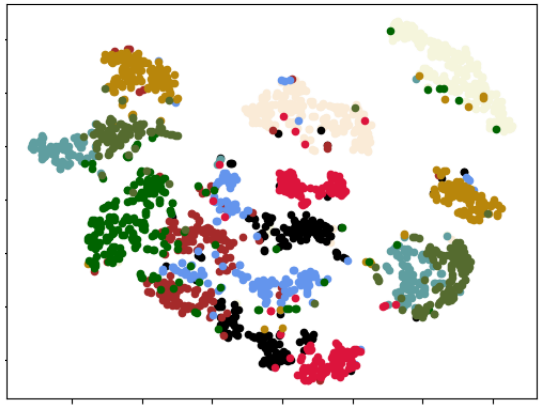
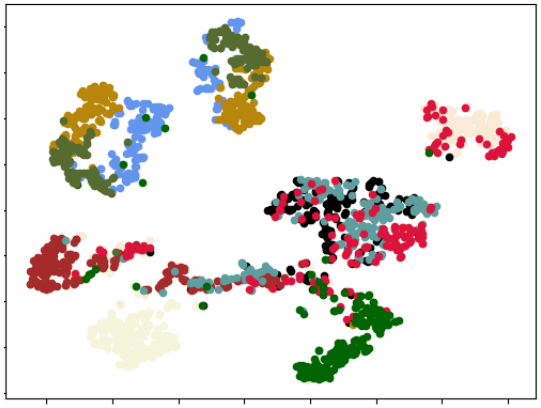
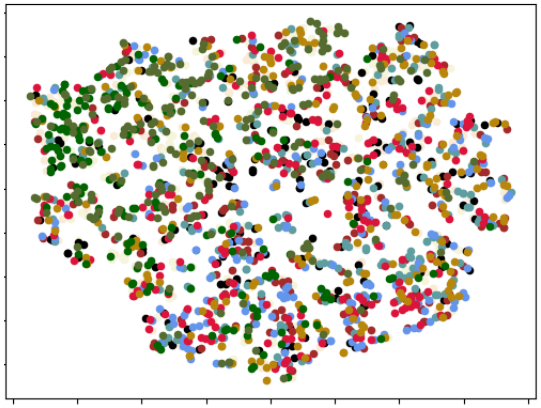
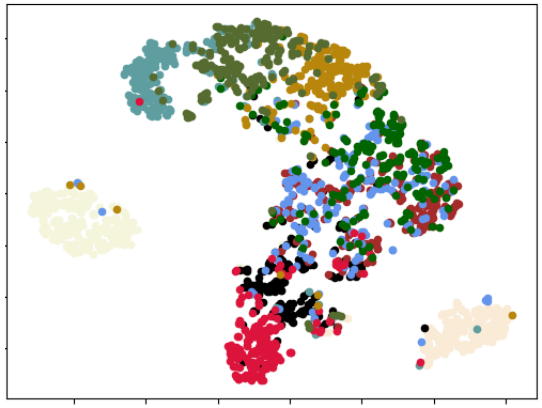
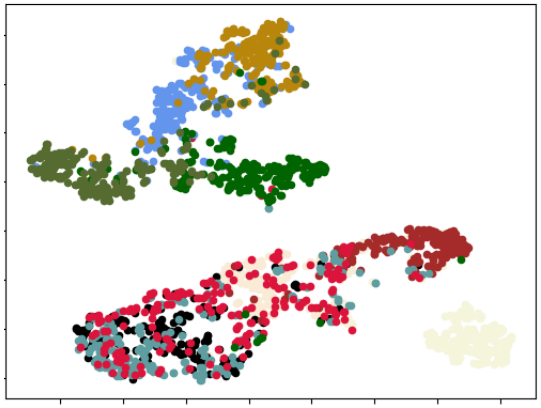
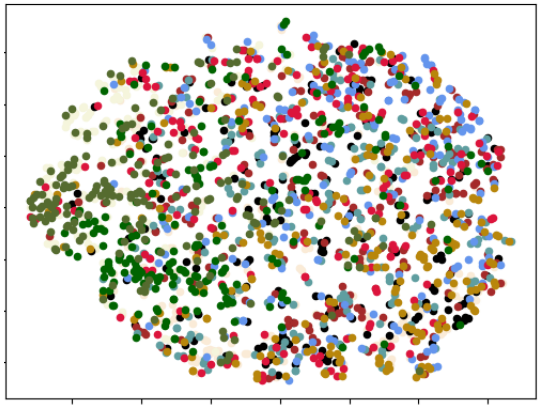
Config-uration	MNIST	Fashion-MNIST	CIFAR-10
Combined Triplet Ranking Loss and MSE Reconstruction Loss, Within Kendall Loss			
Combined Cosine Ranking Loss and MSE Reconstruction Loss, Within Kendall Loss			
Combined Circle Ranking Loss and MSE Reconstruction Loss, Within Kendall Loss			

Table 12: Clustering capability of tripet, cosine, and circle ranking loss, combined with MSE loss, within Kendall loss, with various image datasets.

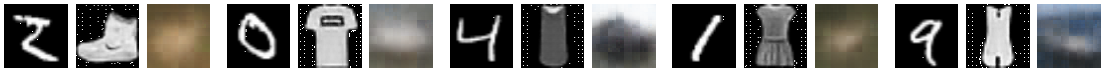


Figure 24: (Alternating) reconstructed MNIST, Fashion-MNIST and CIFAR-10 images with contractive autoencoder

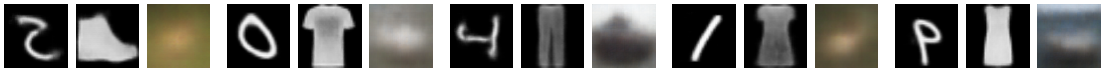


Figure 25: Reconstructed images with Variational autoencoder [15]

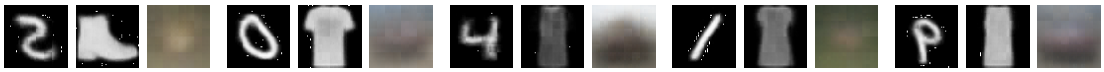


Figure 26: Reconstructed images with combined triplet ranking loss and MSE reconstruction loss

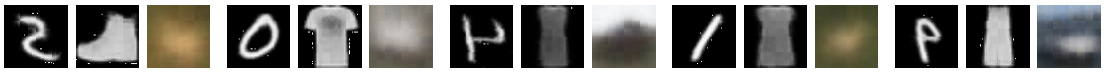


Figure 27: Reconstructed images with combined cosine embedding ranking loss and MSE reconstruction loss

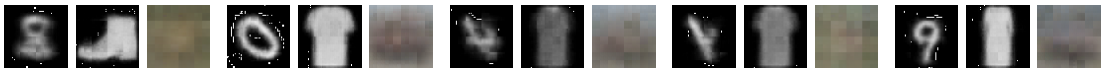


Figure 28: Reconstructed images with combined circle ranking loss and MSE reconstruction loss

Combined contractive loss and MSE reconstruction loss

Combined triplet ranking loss and MSE reconstruction loss

Combined cosine embedding loss and MSE reconstruction loss

Combined circle loss and MSE reconstruction loss

Figure 29: Reconstructed CIFAR-10 images with various loss functions, with autoencoder latent space dimension size $nz = 32$

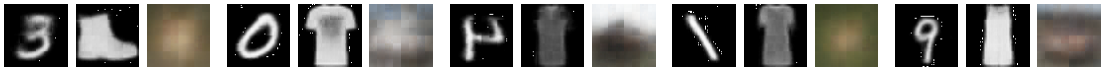


Figure 31: Reconstructed images with combined triplet ranking loss and MSE reconstruction loss, within Kendall Loss

Combined contractive loss and MSE reconstruction loss

Combined triplet ranking loss and MSE reconstruction loss

Combined cosine embedding loss and MSE reconstruction loss

Combined circle loss and MSE reconstruction loss

Figure 30: Reconstructed CIFAR-10 images with various loss functions, with autoencoder latent space dimension size $nz = 16$

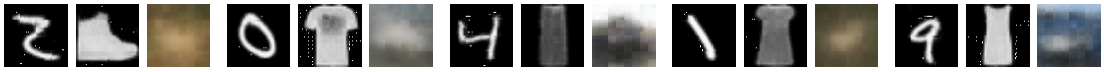


Figure 32: Reconstructed images with combined cosine embedding loss and MSE reconstruction loss, within Kendall Loss

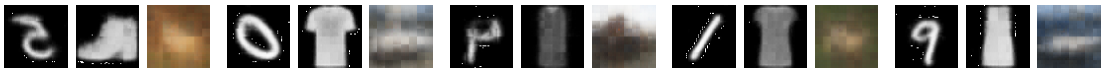


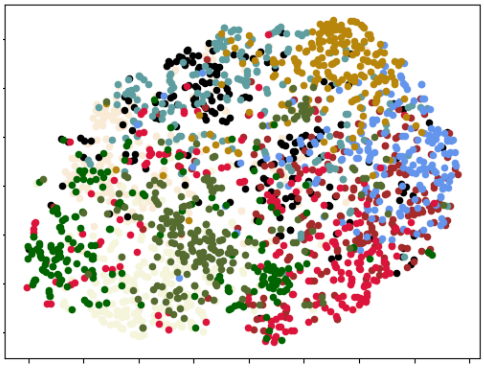
Figure 33: Reconstructed images with combined circle loss and MSE reconstruction loss, within Kendall Loss

Triplet ranking loss

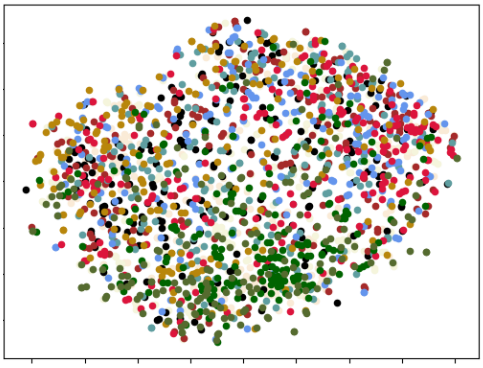
Cosine embedded ranking loss

Circle ranking loss

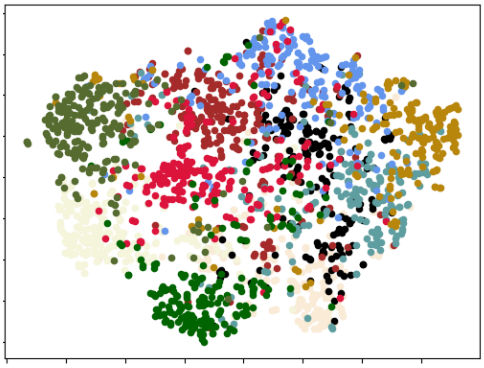
Figure 34: Reconstructed CIFAR-10 images with combined ranking loss and MSE reconstruction loss, within Kendall Loss; $nz = 32$



(a) Triplet ranking loss



(b) Cosine embedding loss



(c) Circle loss

Figure 35: TSNE visualization of clustering with combined ranking loss and MSE reconstruction loss, within Kendall Loss; $nz = 32$

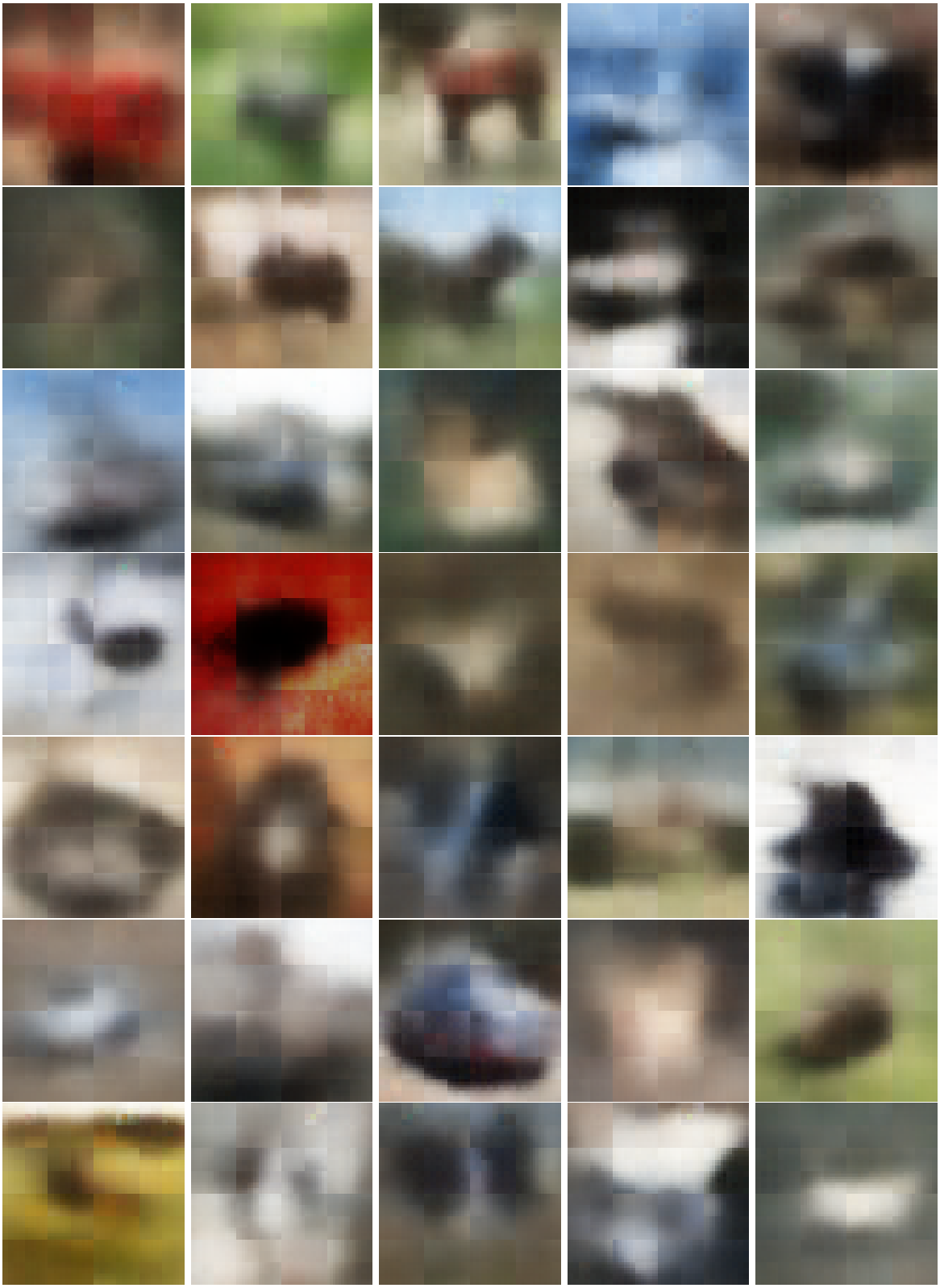


Figure 36: Additional reconstructed CIFAR-10 images with combined triplet ranking loss and MSE reconstruction loss, within Kendall Loss; $nz = 32$. The last tanh activation layer can introduce artifacts during reconstruction with GANomaly when a low embedding dimension is used. For visualisation purposes we simply employ min-max normalization to prevent this.

4.2.2.1 Enforcing Similarity within Latent Space

Our approach necessitates the high-dimensional feature representation of images to be encoded within the lower dimensional latent space. Consequently, we require that similar images also have similar encodings. To enforce this property, we qualitatively evalute the loss function adopted by state of the art autoencoder and generative adversarial networks whose primary task is image reconstruction. Using TSNE [249], we visualize the proximity of encoded images within the latent space, where each class is assigned a different colour, across several challenge datasets. To generate each displayed visualization, we adopt TSNE perplexity 30, with 1000 iterations, for a subset (2000 images) of a given dataset.

Auto-encoder training seeks to minimize the reconstruction error on a training set, typically through use of MSE or cross-entropy loss [15, 253, 246, 254, 255]. Contractive Autoencoders [14] propose an additional requirement during training, where autoencoder sensitivity to an input is penalized to encourage encoding of robust features within the latent space. Sensitivity can be represented as the sum of the squares of all partial derivatives of the extracted features with respect to input dimensions [14]. Penalizing this term discourages sensitivity, and thus contracts neighbourhoods within the feature space. In other words, small first derivatives correspond to flatness of the latent space and by extension, invariance. Contraction of this input space *of neighbourhoods* when projected into the latent space, as opposed to simply a global scaling (contraction of all samples uniformally) corresponds to an image similarity embedding, which is exactly what we require.

VAE [15] autoencoders encode inputs as distributions over the latent space rather than single points. VAE employs reconstruction loss along side (Kullback-Leibler) regularizer to measure divergence of a given distribution (in Gaussian form) from a standard Gaussian. In addition to *efficient* lower-bound likelihood estimation via this Kullback-Leibler Divergence, the latent space is contracted and regularized such that i) a given embedding space distribution pertains to a meaningful decoded output, but more importantly ii) similar image embeddings are sufficiently close, pertaining precisely to our requirements. Indeed there is considerable precedence for the adoption of VAE autoencoders, given the recent success of VAE-GAN generative networks [254, 71, 256, 257]. We do not consider the modifications such GAN architectures make to the VAE loss function however, given their dependency on critic loss, unique to GAN architecture and consequently unsuitable for training our autoencoder network.

VQ-VAE [253] instead utilize discrete encodings, rendering the regularization term from the continuous VAE loss formulation obsolete as the encoded codes are limited in number (and additional regularization is encapsulated within the autoencoder network itself). More recently, VQ-VAE-2 [246] adopts the VQ-VAE model [253] with additional hierarchy of vector quantized codes, such that local and global information are

encoded within separate hierarchical codes. The latent space representation of a given image is therefore modelled by several distinct encodings, which are considered in tandem during image reconstruction. Owing to our requirement that an encoding is as succint as possible for fast image lookup within a *kd*-tree, several separate encodings, and thus the VQ-VAE-2 approach, is inappropriate.

Considering the deployment of the GANomaly [224] encoder-decoder architecture within our encoder training regime comparison, it is natural to also consider use of the second encoder within the overall original GANomaly pipeline. The decoded images are passed through an additional encoder (with different parameterization) to generate an additional latent feature representation. By introducing an additional loss term that minimizes differences between the two latent feature representations, the embedding space is regularized in a manner reminiscent of Kullback-Leiber divergence within VAE. However, the results were unsatisfactory without the presence of adversarial loss as per the original formulation.

Additonally, there exists an entire research domain dedicated to training networks with our criteria in mind [258, 259]. Contrastive learning can generally be summarized as learning a similarity ranking between given images, before even considering higher order tasks such as image classification, object detection, or instance segmentation. In turn, this encourages learning the higher-level features within an image. Furthermore, by *ranking* images with respect to their similarity, clustering within the latent space is directly encouraged. We note that the formulation of the *kd*-tree (Section 4.2.2) within Euclidean space for image dissimilarity comparison necessitates use of the Euclidean distance metric within these ranking losses.

Indeed, the seminal contrastive loss [258] work directly achieves input data ranking by minimizing the embedding distance for within-class input pairs, while maximising the distance for between-class pairs. Triplet loss [250] adopts a similar approach in which triplet inputs are considered, necessarily consisting of an anchor input, a within-class (positive) input and a between-class (negative) input. Embedding distance between the anchor input and positive input should be minimized, while distance to the negative input is maximised.

Circle loss [260] demonstrate that attributing equal weighting to maximising within-class similarity and minimizing between-class similarity is inappropriate within ranking losses. Further, it is similarly inefficient to equally penalize similarity score regardless of how far it is from optimal. Therefore, within-class and between-class scores are not only independently weighted, but implemented as linear functions so that more optimal similarity scores are weighted less, and vice-versa (yielding a circular decision boundary). As well as yielding better results for image reconstruction, this technique linearly distinguishes between-class similarity, in turn encouraging latent space image representation clustering.

4.2.2.2 Evaluating Autoencoder Training

The effectiveness of our learnt feature embedding is illustrated in Figure 23-34 for various example datasets. To consider our autoencoder satisfactory, we require firstly that it learns an image embedding and a decoder that can reconstruct the original image. While the reconstruction does not have to be perfect, the original classes must at least be distinguishable, and somewhat recognisable by human observation. With this, we can be confident that the learnt latent space in fact represents a given image. Secondly, we require that images from the same class have similar encodings within the latent space. To reiterate, given the second criteria but not the first, we cannot be confident that a given embedding represents a given image. As such, using its embedding in place of the image itself does not correlate with its dissimilarity from other images. In other words, there is no correlation between embedding space dissimilarity and image space dissimilarity. Given the first criteria but not the second, we cannot guarantee that the correlation is *strong*. Provided these two criteria are met, image dissimilarity can be efficiently computed within our *kd*-tree (Section 4.2.2).

For the simpler dataset tasks (MNIST, FashionMNIST), contractive loss is sufficient; clustering between classes is distinct (Fig. 23a, 23b), while images are reconstructed satisfactorily (Fig. 24). However, for the more complex CIFAR-10 class (likely owing to the move to RGB-space), the clustering is insufficient (Fig. 23c). Moreover, the autoencoder does not yield reasonable reconstruction (Fig. 24) in this case.

VAE is unable to improve upon the poor clustering of CIFAR-10 images, despite adopting Kullback-Leibler divergence regularizer. While reconstruction shows relative promise to other methods (although blurry), we opt for alternative measures that can be more easily modified to generate reasonably clustering capability.

By adopting contrastive learning, more promising clustering results were achieved. Triplet loss achieves sufficient clustering (Table 10), owing to the *ranking* of image inputs. Cosine embedding loss certainly improves upon the clustering capability that contractive loss offers, but is inferior when compared to triplet loss (Table 10). Similar images are close within embedded space, in that classes are clustered close to each other. However, other classes reside in the same TSNE 2D embedding space. Although this is likely attributable to the imperfect TSNE visualization approach, it is nontheless suggestive that dissimilar images are also *somewhat* clustered closely. Adopting either ranking loss on its own does not enable reconstruction. To this end, adopting a combined ranking and reconstruction loss is necessary. In many cases, adopting a combined loss causes the clustering to deteriorate beyond satisfactory thresholds (Table 11). However, the reconstruction is present (Fig. 26, 27). Although by no means of sufficient quality for CIFAR-10, the MNIST and Fashion-MNIST class is adequately identifiable.

Circle ranking loss achieves near-perfect clustering for MNIST, and sufficient clustering for the other

two datasets (Table 10). Combining MSE reconstruction loss with circle ranking loss certainly improves reconstuction quality, in so far as MNIST and Fashion-MNIST classes are distinguishable (Fig. 28), but lack clarity compared to triplet and contrastive ranking loss counterparts. CIFAR-10 reconstruction quality is inferior with combined circle ranking and MSE compared to the other two ranking losses, and remains insufficient. Clustering quality on the other hand does not deteriorate to the same degree with the additional MSE loss term compared to alternative ranking metrics. This would suggest that under a combined ranking and reconstruction loss approach, circle loss introduces an inherently higher weighting compared to either triplet or cosine embedding loss, and one that MSE reconstruction is unable to surmount.

4.2.2.3 Evaluating Autoencoder Training Further

Up to this point, we have only considered a naive combination of at most two loss measures (ranking and reconstruction losses). However, the suprising reduction in performance that circle loss achieves would suggest a weighted combination of losses is appropriate. Furthermore, until now reconstruction quality for CIFAR-10 has been very poor.

Owing to this lack of reconstruction quality for the harder CIFAR-10 dataset across all ranking loss approaches, we increase the dimension of the embedding space to 32. These additional 24 dimensions are sufficient for the latent representation to capture the features of the input images, such that the decoder can reconstruct them to a reasonable degree (Fig. 29). The finer-grained details become more apparent as the objects within a given image begin to have shape. Moreover, we deem 32 to be the optimal latent space dimension; there is no need to attribute more memory to the latent space since we can already distinguish between classes for the reconstructed images. However, dimensions between 8 and 32 do not yield sufficient reconstruction quality (Fig. 30). The clustering ability across all reported configurations ($nz \in 8, 16, 32$) remains unaffected and insufficient. As such, we still require an alternative training strategy that enhances contrastive learning towards better reconstruction ability. Contractive loss may yield sufficient reconstruction quality (with $N = 32$), but its respective clustering ability is impossibly far from the required quality. [7]

The previous results for combined ranking and reconstruction loss indicated a promising direction. However, no such weighting exists where the clustering potential of ranking loss and reconstruction quality of MSE loss can be realised. As such, we introduce Kendall Loss [252] to dynamically learn a weighting between the two during training. The good TSNE visualization results (Table 12) illustrate that Kendall Loss generally learns a weighting biased towards ranking loss. As such, reconstruction quality once again deteriorates for lower embedding dimension sizes beyond satisfactory thresholds ($N = 8$: Fig. 31,32). For larger dimension

[7] Additionally, a combined ranking and MSE reconstruction loss term *as well as* a contractive loss term does not grant any further performance improvement.

sizes ($N = 32$) however, the quality remains suitable (Fig. 34). In fact, with this combined triplet ranking loss and MSE reconstruction loss in tandem with Kendall loss, we achieve appropriate reconstruction and clustering quality (Fig. 34,35).

Although reconstruction ability *appears* insufficient, we argue that it is infact *enough*. As will be illustrated within the ablation studies of this chapter (Sec. 4.4.2.2), reconstruction ability is inconsequential if clustering ability is insufficient. Moreover, adopting an autoencoder within DDS-NAS with the reconstruction capability illustrated in Fig. 34 yields very good empirical results (Sec. 4.4.1). Further, additional reconstructed images by our final proposed autoencoder better illustrate its capacity to produce identifiable images (Fig 36).

Finally, a combined circle loss and MSE reconstruction loss under Kendall loss surpasses triplet and contrastive ranking loss counterparts, with respect to both clustering abilty (Table 12) and reconstruction quality (Fig. 33). However, our DDS-NAS approach requires only that performance is sufficient in these two regards. Given training costs associated with circle loss implementations are higher than triplet loss, there is no benefit in selecting it for MNIST, Fashion-MNIST, and CIFAR-10 datasets for use within DDS-NAS.

We additionally note the precedence for and importance of hard triplet mining during triplet learning commonly employed within object detection tasks [250]. Considering we adopt triplet learning only to facilitate clustering, rather than for triplet ranking within the loss term for an image classification or object detection task itself, we can safely omit triplet mining. Consequently, we can avoid any drawbacks that hard triplet mining introduces, for instance mislabelled data dominating hard positive or negative triplets [250].

4.3 Experimental Setup

We detail our experimental setup for DDS-NAS deployment across the Differentiable Architecture Search (Darts [53]), Progressive DARTS (P-DARTS [108]) and Network Pruning via Transformable Architecture Search (TAS [154]) NAS frameworks. This setup is used to demonstrate performance of our proposed approach with several image classification datasets.

4.3.1 NAS Configuration

Unless otherwise stated, all employed NAS frameworks adopt the same common configuration using Adam optimisation [261] with initial learning rate $lr = 3e^{-4}$, weight decay $wd = 1e^{-3}$, and momentums $\beta_1 = 0.5$ and $\beta_2 = 0.999$ (P-DARTS uses $lr = 6e^{-4}, wd = 1e^{-3}$, TAS uses $lr = 1e^{-4}$). For weight optimization for the NAS derived architectures themselves, we use an SGD optimizer with $wd = 3e^{-4}$, and momentum $\beta = 0.9$ (P-DARTS uses $wd = 5e^{-4}$). Additionally, for DARTS we employ a Cyclic Learning Rate Scheduler with

base $lr = 0.001$, max $lr = 0.01$, and step size up = step size down = 10. We set $lr = 0.01$ when the previous dynamically selected data subset is mastered, and an updated data subset is introduced. Therefore, the updated data subset is learned quickly before being subsequently 'fine-tuned' as with the previous subset. Such an approach finds precedence in SGDR [262], wherein learning rate is periodically reset to a higher value before learning rate decay is reapplied. P-DARTS and TAS both adopt Cosine Annealing Learning Rate Scheduler with $lr = 2.5e^{-2}$ and $lr = 0.1e^{-2}$ respectively. We select the ResNet-110 architecture for the TAS *kd*-teacher training. The models are implemented using PyTorch [263] (v1.6.0, Python 3.6.9). For fair comparison, the DDS-NAS framework always adopts the same training augmentations as the original implementation, specified in the respective implementations ([53, 264], [108, 265], [154, 266]).

Performance of DDS-NAS deployed across each NAS framework is presented in terms of both Top-1 accuracy and parameter count (complexity) of the optimal NAS generated architecture, together with the computational effort of the NAS search phase (in GPU days) across all three datasets.

Experimentation indicates that our NAS framework is generally insensitive to *a priori* thresholds that do not need to be exhaustively searched. A subset-size of 100 is sufficient for the easier MNIST [267] and Fashion-MNIST [268] tasks, and 1000 for CIFAR-10 [93]. Adopting a high hardness threshold (*hard*-ness threshold > 0.8) across all datasets and all NAS strategies enables the searched network architecture to formulate a thorough feature representation for image classification. The best network architectures are discovered with a mastery threshold ≈ 0.5. P-DARTS and TAS learn deep representations for images more slowly compared to DARTS. This can be attributed to the additional tasks alongside reducing classification loss wherein P-DARTS progressively restricts the search space while increasing architecture depth, and TAS minimizes for network architecture complexity. Conversely, DARTS can afford a lower mastery threshold (≈ 0.15) for the easier MNIST and Fashion-MNIST tasks, but the performance gain is marginal. All presented results use the same hardness (0.85) and mastery (0.5) thresholds to ensure fairness.

4.3.2 Hard Example Mining

The GANomaly autoencoder [224] used to encode the images into their latent space representation is trained with Contractive Loss [14] for 30 epochs, with $bs = 8$, and Adam optimizer with momentums $\beta_1 = 0.9$ and $\beta_2 = 0.999$, $wd = 0$, $lr = 1e^{-3}$. For the more complex CIFAR-10 task, the autoencoder is instead trained with combined triplet margin loss [250] and MSE reconstruction loss, weighted under Kendall Loss [252].

Dataset	NAS Approach	Top-1 Accuracy (%) ↑ DARTS / P-DARTS / TAS	Params (M) ↓ DARTS / P-DARTS / TAS	Search Cost (GPU Days) ↓ DARTS / P-DARTS / TAS
MNIST	Original	99.75 / 99.26 / 99.27†	0.66 / 3.68 / 1.00	0.51 / 1.89 / 0.28
	DDS-NAS	99.78 / 99.17 / 99.30†	0.75 / 3.51 / 0.81	0.030 / 0.070 / 0.021
Fashion MNIST	Original	95.33 / 93.42 / 95.09†	3.27 / 4.04 / 0.94	0.63 / 1.98 / 0.27
	DDS-NAS	95.48 / 93.04 / 95.08†	3.44 / 4.23 / 0.83	0.030 / 0.078 / 0.031
CIFAR-10	Original	97.17 / 96.50 / 93.89	3.16 / 3.43 / 0.85	1.78 / 0.65 / 0.26
	DDS-NAS	96.57 / 95.07 / 93.12	3.72 / 4.13 / 1.06	0.36 / 0.095 / 0.040
	Shapley-NAS [92]	96.96	3.60	0.36
	SNAS [112]	97.15	2.85	1.83
	DenseNet [17]	94.23	7.0	–
	DenseNet+ [17]	95.90	7.0	–

Table 13: Accuracy, memory footprint and (search-phase) training cost of final generated model from DDS-NAS deployed upon DARTS, P-DARTS, and TAS, compared to their original implementations and others. † indicates results prior to *kd*-teacher training owing to lack of available teacher model for MNIST and Fashion-MNIST datasets. DenseNet(+) [17] refers to training configurations with and without the widely-adopted augmentation methods [18].

4.4 Evaluation

Having validated the feature representation embedding that underpins our dynamic data selection via hard example mining (Section 4.2.2.2 / Figure 23-31), we present out evaluation in terms of DDS-NAS comparison to contemporary state-of-the-art approaches (Section 4.4.1), supporting ablation studies (Section 4.4.2).

4.4.1 Neural Architecture Search

Table 13 presents the performance obtained by the final model generated by DDS-NAS with respect to each dataset under consideration. Across all cases our generated models offer performance competitive with the state of the art, with minimal to no impact on generated model size. Moreover, across all cases, we substantially lower the computational efforts required for NAS (0.07 GPU days compared to 1.89 in the case of P-DARTS for MNIST, 27× quicker)[8]. Since we can determine a replacement image for our dynamic subset in average case $O(log(n))$ time, we are able to reduce the search phase training cost by one order of magnitude over state of the art results.

[8]and still an order of magnitude faster even after factoring in the time taken to train the autoencoder

Dataset	NAS Approach	Top-1 Accuracy (%) ↑ DARTS / P-DARTS / TAS	Params (M) ↓ DARTS / P-DARTS / TAS
CIFAR-100	Original	81.33 / 80.58 / 71.72	2.75 / 3.49 / 1.15
	DDS-NAS	82.64 / 75.45 / 70.47	3.80 / 4.24 / 1.15
	Shapley-NAS [92]	83.42	3.66
	DenseNet [17]	76.21	7.0
	DenseNet+ [17]	79.80	7.0
ImageNet	Original	73.30 / 75.72 / -	4.51 / 4.94 / -
	DDS-NAS	76.26 / 75.63 / -	10.06 / 5.68 / -
	Shapley-NAS [92]	75.52	5.14
	DenseNet [17]	74.98	7.0

Table 14: Accuracy (Top-1), and memory footprint of final searched models from CIFAR-10 transferred to CIFAR-100 and ImageNet. DenseNet(+) [17] refers to training configurations with and without the widely-adopted augmentation methods [18].

As ever, result reproduction remains an ongoing significant problem [269], one that is only magnified within the complexity of the NAS domain [270]. As far as we can manage, the hyperparameters, dataset, and development environment we have used to train on ImageNet [29] remain consistent with the original papers. In the case of TAS [154] however, the evaluation results achieved by the network architecture searched by their respective original algorithm are considerably lower than those reported in the original literature. While using fewer, hard, mined data images selected by our DDS-NAS approach yields an architecture that transfers better to ImageNet, few conclusions can be drawn and we omit the results for clarity. Nevertheless, without loss in performance, our hard example mining method yields discriminative architectures that can be transferred to CIFAR-100 [93] and ImageNet [29] (Table 14). Evidently, using hard mined data yields architectures at least as transferable to more complex classifiation tasks. Furthermore, it is possible that by using only select, hard mined data during the NAS search phase (via DDS-NAS), we can better discriminate between the best performing architectures in the architecture search space, thus yielding architectures even more generalizable to complex tasks. Further work is required to confirm this behaviour, however. Furthermore, whilst our technique is demonstrated upon commonplace NAS approaches (DARTS, P-DARTS, TAS) it could equally be deployed on top of more recent advancements [92, 151, 112], further minimizing any difference in performance.

Dataset	NAS Approach	Top-1 Accuracy (Search Phase) (%) ↑ DARTS / P-DARTS / TAS	Top-1 Accuracy (Final) (%) ↑ DARTS / P-DARTS / TAS	Params (M) ↓ DARTS / P-DARTS / TAS
MNIST	DDS-NAS	94.00 / 78.89 / 44.89	99.78 / 99.17 / 99.30	0.75 / 3.51 / 0.81
	Original framework with dataset size 100	78.28 / 70.81 / 39.24	94.43 / 98.69 / 99.18	0.70 / 4.54 / 0.53
	DDS-NAS with untrained autoencoder	92.28 / 78.76 / 51.96	95.28 / 98.78 / 99.21	0.75 / 3.11 / 1.05
Fashion-MNIST	DDS-NAS	72.92 / 65.71 / 32.71	95.48 / 93.04 / 95.08	3.44 / 4.23 / 0.83
	Original framework with dataset size 100	56.27 / 58.16 / 35.66	91.69 / 90.03 / 94.61	3.47 / 4.69 / 0.46
	DDS-NAS with untrained autoencoder	69.49 / 64.47 / 39.12	92.04 / 91.52 / 94.87	3.48 / 3.92 / 0.93
CIFAR-10	DDS-NAS	56.00 / 22.14 / 29.88	96.57 / 95.07 / 93.12	3.72 / 4.13 / 1.06
	Original framework with dataset size 1000	51.02 / 41.70 / 23.81	88.58 / 85.74 / 90.83	3.55 / 4.04 / 0.32
	DDS-NAS with untrained autoencoder	51.10 / 46.59 / 28.21	88.90 / 88.96 / 91.72	3.64 / 4.25 / 0.83

Table 15: Ablation Studies: accuracy and memory footprint of models generated by DDS-NAS; models generated by the original framework with limited data (equivalent to removing hard example mining and curriculum learning); and models generated by DDS-NAS with untrained autoencoder (equivalent to removing hard example mining).

4.4.2 Ablation Studies

To validate our proposed approach we compare the performance of DDS-NAS to selected NAS frameworks both: (a) without *dynamic data selection* in order to ablate the contribution of our combined hard example mining and curriculum learning strategy; and (b) with an untrained autoencoder in order to ablate the contribution of the image dissimilarity based hard example mining strategy.

4.4.2.1 Without Dynamic Data Selection

For each dataset, we employ all three original implementations (DARTS [53], P-DARTS [108], TAS [154]), but with a subset of the data at each training iteration. *This is equivalent to omitting both hard example mining and curriculum learning.* We use the same volume of data as adopted by DDS-NAS: 100 randomly selected images for MNIST and Fashion-MNIST, and 1000 for CIFAR-10. Subsequently, we can determine the impact of our curriculum learning and hard example mining pipeline. Comparing the first and second row of the results for each dataset presented in Table 15, it is evident that DDS-NAS achieves substantially improved accuracy while yielding fractionally larger architectures in some cases. This behaviour is exhibited in MNIST, where the original DARTS framework achieves only 78.28% accuracy after the search phase, and 94.43% accuracy after fine-tuning the stacked searched cell (compared to 94.00% and 99.78% respectively

Reconstruction	✓	✓	×	×
Clustering	✓	×	✓	×
Top-1 Accuracy (%) ↑	96.57	95.29	94.94	88.90

Table 16: Accuracy of DDS-NAS-DARTS employing autoencoders with different capabilities on CIFAR-10.

for DDS-NAS-DARTS). This performance difference is further highlighted with both the other datasets, and other frameworks; the final performance of the original P-DARTS implementation falls behind DDS-NAS across all datasets (85.74% compared to 95.07% for CIFAR-10, for instance). Interestingly, with hard example mining and curriculum learning omitted in this manner, TAS generates smaller models (0.32M compared to 1.06M for CIFAR-10), but often at the expense of accuracy.

4.4.2.2 Untrained Autoencoder

We ablate the autoencoder derived feature embedding within our hard example mining method by replacing the DDS-NAS autoencoder with one that is untrained, and thus unable to determine the most dissimilar images from our current data subset used for training. *This can be considered as a process equivalent to curriculum learning without hard example mining*, as the images are effectively randomly sampled. This time, we compare the first and third row for each dataset in Table 15. Evidently, the models generated by DDS-NAS with an untrained autoencoder are significantly worse (for instance 92.04% compared to 95.48% upon Fashion-MNIST by DDS-NAS-DARTS). On this basis, we can therefore conclude that DDS-NAS necessarily requires a suitable hard example mining approach, for which our image similarity strategy is sufficient.

Furthermore, an autoencoder that achieves good reconstruction but mediocre clustering of embedded features is inadequate for DDS-NAS (Fig. 37, Table 16). Bad clustering and thus ineffective hard example mining directly yields inferior classification accuracy (95.29%) compared to hard example mining with good clustering (96.57%). Similarly, sufficient clustering but poor reconstruction is detrimental to DDS-NAS (94.94%). Lack of both properties yields significantly worse performance (88.90%), wherein there is no correlation between embedding space dissimilarity and image space dissimilarity at all.

By comparing row two (neither hard example mining nor curriculum learning) and row three (curriculum learning but not hard example mining) for each dataset in Table 15, it is clear that our curriculum learning methodology is *somewhat* effective even without incorporated hard example mining. DDS-NAS performance with an untrained autoencoder exceeds that of the original framework with limited data in all cases (88.58% compared to 88.90% for CIFAR-10 with DARTS, 90.03% compared to 91.52% for Fashion-MNIST with P-DARTS).

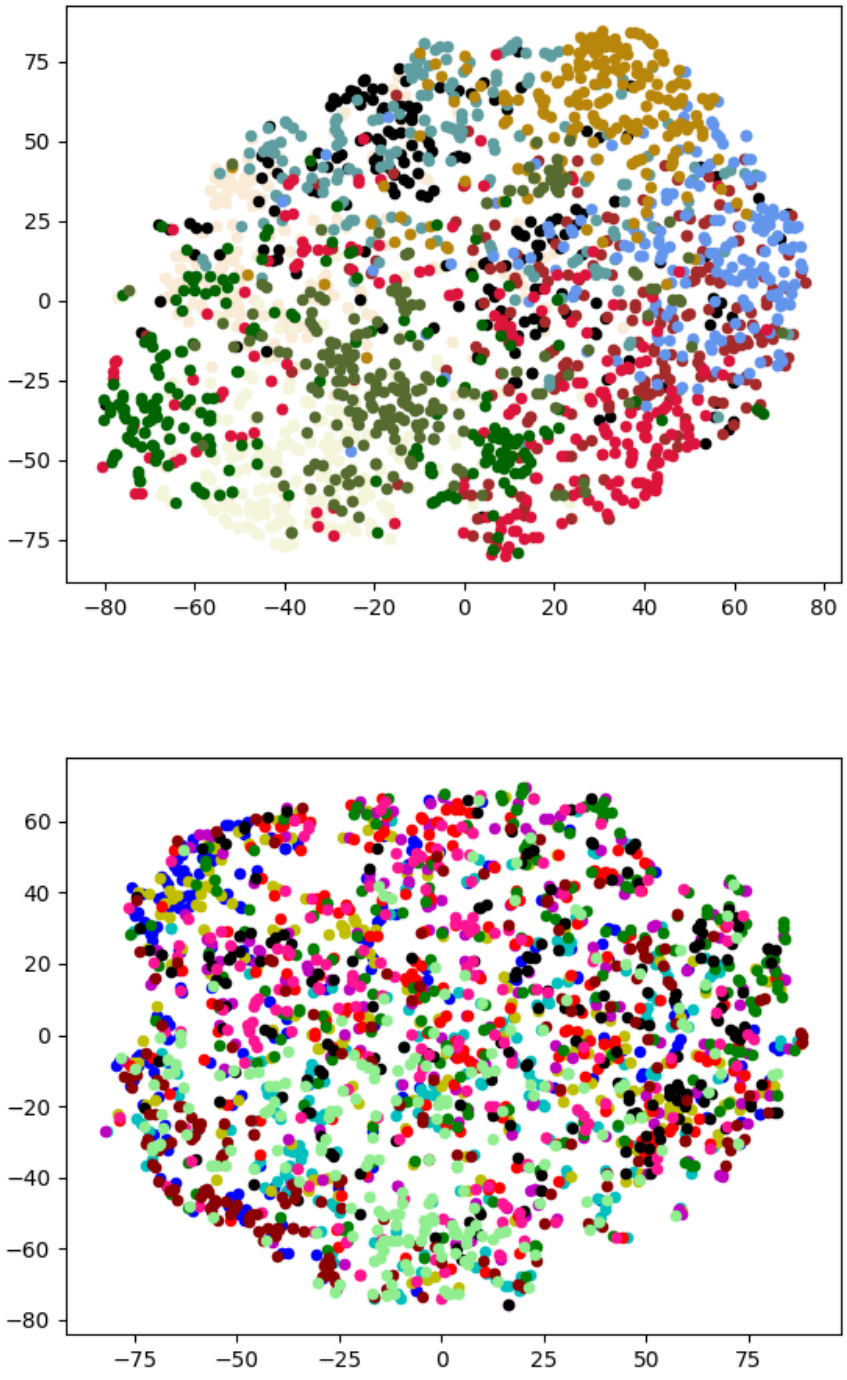


Figure 37: TSNE visualization of clustering of autoencoded CIFAR-10 image feature representation within the latent space. Training with triplet margin loss with Kendall loss achieves good clustering (above). Training with contractive loss achieves poor clustering (below).

4.4.3 Investigating Dynamic Dataset Sampling and Convergence

The impact of generating a representative subset of the dataset can be formulated in terms of gradient noisiness. Let us consider the DDS-NAS representative subset of data at a given epoch $x \sim p_{\text{rep}}$, compared to a NAS approach which has no such dataset sampling $x \sim p_{\text{data}}$. During gradient-descent in NAS search, if the gradients are more aligned with the direction towards the minimum of the loss function in the parameter space, then the super-network model will converge more quickly. Conversely, noisier gradients are not in the direction of the minimum, and convergence is slower. We expect that subsampling the dataset via DDS-NAS yields less noisy gradients, and thus the NAS super-network loss converges more quickly. Formally, at batch update n, we expect the loss L for a super-network model parameterized by θ for the two different distributions to be given by Equations 10 and 11:

$$\mathbb{E}_{x \sim p_{\text{rep}}}[\Delta L(n, x)] > \mathbb{E}_{x \sim p_{\text{data}}}[\Delta L(n, x)], \tag{10}$$

$$\mathbb{E}_{x \sim p_{\text{rep}}}[L(\theta_{n-1}, x) - L(\theta_n, x)] > \mathbb{E}_{x \sim p_{\text{data}}}[L(\theta_{n-1}, x) - L(\theta_n, x)]. \tag{11}$$

Let us consider the standard deviation of the loss of the model across a given batch as a measurement for the noisiness of the gradient update. It is clear from Figure 38 that the standard deviation of the loss across the batch decreases more quickly for DDS-NAS-DARTS than the original DARTS implementation. Representative subsampling of the data introduces fewer noisy gradients during NAS search, enabling the NAS super-network to converge more quickly without loss in performance.

The standard deviation in the earlier iterations during DDS-NAS-DARTS does not decrease, as there are no dataset-updates during the first 10 epochs (see Section 4.2.1). After this point, the standard deviation can be seen to fluctuate with each dataset update (rising during the first few iterations after each dataset update), but decreasing overall. The standard deviation has not decreased for the original DARTS implementation before DDS-NAS-DARTS search has completed in its entirety.

4.5 Extending to Prediction-Based NAS

The majority of NAS strategy for image classification can be considered iterative, by which we mean NAS training at each epoch considers all or part of a given dataset. The NAS framework trains for successive epochs until the model converges. Prediction-based NAS often breaks this paradigm, for instance not training until convergence [109], or even training for a single mini-batch [110]. In this section, we demonstrate that our hard example mining approximation is useful even under extreme prediction-based NAS strategies [110].

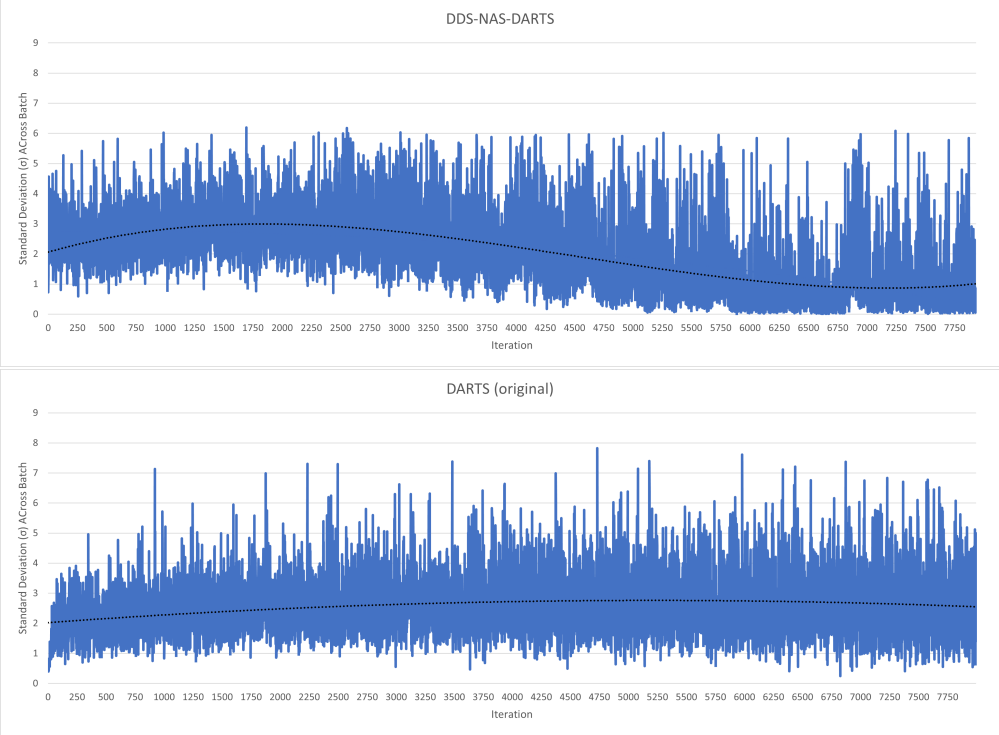


Figure 38: Comparison of the standard deviation across the batch for DDS-NAS-DARTS (above) and the original DARTS method (below) during NAS search on CIFAR-10. Standard deviation decreases for DDS-NAS-DARTS with each dataset update. The black line represents a third degree polynomial curve of best fit.

4.5.1 Scoring Network Architectures after a training mini-batch

Following the methodology to display 'entanglement' of images [110], the benefits of DDS-NAS are apparent (Figure 39). For a given input image, a binary code can be formulated that encompasses which ReLU operation layers are active during the forward pass of the network. The hamming distance between the two binary codes (row i, column j) induced by two input images (image i, image j) will be smaller for poorly performing network architectures. The intuition is that poorly performing network architectures are less able to distinguish between two different images. As such, there are fewer off-diagonal elements for better performing network architectures; network architectures generated by DDS-NAS (Fig. 39, left) show lower (i.e. more blue/purple off-diagonal coloured) off-diagonal entanglement matrices than network architectures generated with an untrained autoencoder (i.e. trained with random images - Fig. 39, middle). This is suggestive that our hard exampling mining approach encourages a deeper feature representation learning, as we are enforcing the network to train from a wide variety of only *hard* images. In turn this would suggest that DDS-NAS encourages the generation of architectures that are best able to distinguish between different inputs.

Furthermore, the DDS-NAS-DARTS entanglement matrix for a given searched model (Fig. 39, left) shows lower (i.e. more blue/purple) off-diagonal entanglement than those for the original DARTS implementation (Fig. 39, right). On the other hand, network architecture entanglement performance from images used during training by both NASWOT [110] and the original DARTS implementation are sampled randomly and exhibit patterns with a similar hue. This indicates that the searched model by the original DARTS implementation has less *potential* to achieve high final performance. DARTS suffers from the optimization gap problem, and fine-tuning the stacked searched cell architecture yields poorer performance than expected. DDS-NAS however enforces deeper representation learning; the stacked searched cells perform best when trained to convergence, ameliorating the optimization gap problem. Indeed these findings form the basis for Neural Architecture Search without Training [110] whose intermediary results we hence corroborate.

4.5.2 Evaluating Predictive Strength

Let us now consider the NASWOT [110] prediction strategy as a means to propose network architectures. Using the score described in Section 4.5.1, we can rank a given architecture in an insignificant time frame. We consider a random mini-batch of images, and assign a score to the network architecture given the image entanglement. We repeat this process 50 times for each architecture in a given architecture space (NasBench-101, NasBench-201, NDS-ResNet). We use the images obtained after the final dynamic dataset update of our DDS-NAS framework to formulate an alternative mined dataset.

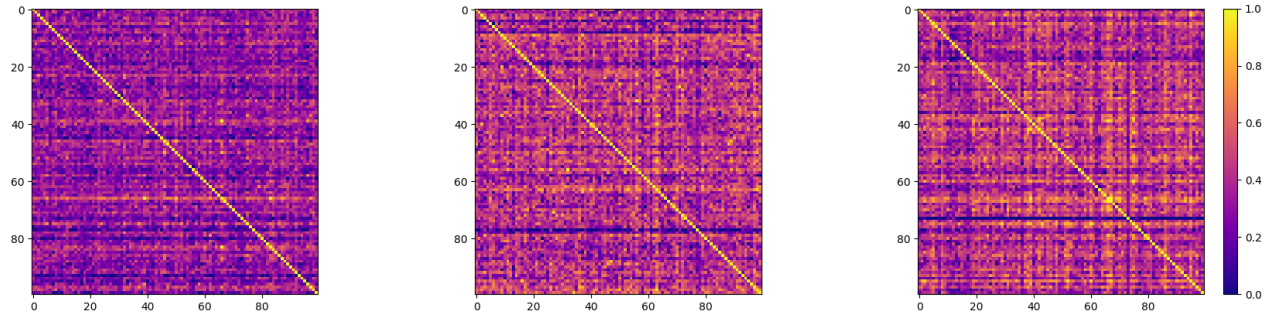
Across a multitude of architecture spaces, and mini-batch sizes, it is clear that instead adopting *hard* mined images yields better architectures (Table 17) than a random data mini-batch. Unsurprisingly, larger architecture spaces yield a larger difference between random and mined images. NASWOT performance gains with mined images is thus most apparent with the NASBench-101 dataset with a mini-batch size of 32, in which searching with mined images yields an architecture with 93.30% accuracy, while random images yields only 90.71% accuracy. Moreover, mined images consistently yield tighter bounds on the searched network architecture performance when we calculate average searched network architecture accuracy across several runs, as well as higher average score. We finally note that using a small mini-batch size marginally deteriorates architecture search performance when using random images, but using mined images ameliorates this behaviour entirely, offering an even faster NAS search speed.

These results call into question the findings from NASWOT that the selected images for a mini-batch have minimal impact on architecture ranking. The authors posit that NASWOT score "captures a property of the network architecture, rather than something data-specific." We propose a more nuanced conclusion, that hard images in a mini-batch better enable NASWOT score to capture network architecture properties.

Using *hard* mined images better distinguishes the best performing networks. At a cost, overall correlation

Mini-batch Size	Sampler	NASBench-101	NASBench-201	NDS-ResNet
32	Mined	$93.30 \pm 0.03\%$	$92.50 \pm 0.21\%$	$93.12 \pm 0.41\%$
	Random	$90.71 \pm 1.18\%$	$91.84 \pm 2.47\%$	$93.02 \pm 0.30\%$
64	Mined	$93.43 \pm 0.00\%$	$92.50 \pm 0.21\%$	$93.05 \pm 0.48\%$
	Random	$91.88 \pm 0.00\%$	$91.91 \pm 2.54\%$	$93.04 \pm 0.49\%$
128	Mined	$93.23 \pm 0.06\%$	$92.50 \pm 0.21\%$	$93.00 \pm 0.29\%$
	Random	$90.89 \pm 2.73\%$	$91.93 \pm 2.56\%$	$93.01 \pm 0.39\%$

Table 17: mean±std searched architecture accuracy (%) with NASWOT using mined and randomly selected images, across a multitude of architecture search spaces and mini-batch sizes. Mined images yield a better performing architecture.



(a) Network architecture generated by DDS-NAS-DARTS

(b) Network architecture generated with random images

(c) Network architecture generated by original DARTS implementation

Figure 39: Visualization of the ability of a generated network architecture to distinguish between given image inputs. Row i, column j corresponds to the hamming distance between the binary codes representing the activation pattern of the ReLU operations of the given neural network architecture, induced by image i and image j. The matrix is normalized such that the similarity between the codes induced by identical images (the diagonal) is 1. High performing network architectures (left) therefore have fewer off-diagonal elements appearing more blue/purple off-diagonal in our visualisation.

between attributed entanglement score drops (for instance $\tau = 0.596$ using random CIFAR-10 images, compared to $\tau = 0.592$ with mined CIFAR-10 images), as the majority of the search space consists of weaker architectures that are less able to distinguish *hard* images. However, in most cases we want to find the best overall architecture, and mined images are well suited to this task. If however, high correlation for all architectures was preferred (for instance to identify the top n architectures for some high value of n, or because only a subset of weak architectures are sampled due to a threshold on the size of the model), using images mined from an earlier DDS-NAS iteration would likely achieve even better results. This is left as a direction for future work, however.

4.6 Conclusion

To conclude, we propose DDS-NAS: a novel NAS framework capable of reducing the time required for the NAS search phase by one order of magnitude. By employing image similarity as a basis for hard example mining, and thus (online) dynamic data sub-selection, DDS-NAS yields models that remain competitive towards accuracy and memory costs upon common image datasets. Further, DDS-NAS is demonstrated to be deployable upon several NAS approaches, and similarly extendable to all existing evolutionary, reinforcement-learning or gradient-based NAS approaches. DDS-NAS can even incorporate NAS search phase techniques that are deployed alongside rather than in place of existing NAS approaches [151, 128].

Following the success of our approach, we posit that only a fraction of commonly used image datasets contribute to learning. As such, additional analysis of these datasets is necessary. Moreover, a more comprehensive investigation into the autoencoder architecture employed within our hard example mining method may yield better results (and thus reduce the volume of contributing images within a dataset further still). Specifically, we require an autoencoder that can generate similar embeddings for similar images even within the fine-grain classification domain. Alternative measures of image similarity such as hashing, may yield similar improvements. In the same vein, deep metric learning [259, 271, 272] literature largely pertains to our requirements, although existing approaches utilize too high a feature representation dimension to be incorporated into the DDS-NAS approach. Nevertheless, even a glimpse of image similarity as a metric within hard example mining has proven extremely effective. We thus introduce several new avenues for improvement, particularly alongside NAS frameworks, and demonstrate that network architecture topology design should not necessarily be the sole consideration for future NAS solutions.

5

Extending DDS-NAS to Multi-label Classification and Object Detection

5.1 Introduction

With the plethora of object detection challenges arising within the computer vision domain, efficient and/or real-time performance is paramount. Additionally, the complexity of modern architectures increases in correlation with the task. Indeed, state-of-the-art object detection necessitates deep convolutional neural networks [24] or transformer networks [94], with parameters several orders of magnitude larger than state of the art classification models. Following the emergence of big data and subsequent public availability of object detection datasets, each consisting of tens of thousands of images [30, 29], models are commonly trained for thousands of epochs to form a sufficient understanding across the entire dataset. This is in addition to the thousands of epochs the backbone feature extraction (image classification) networks are trained on, upon which convolutional object detection architectures are constructed.

Moreover, more and more devices of limited capability, not least mobile phones, UAV and embedded systems, receive and process real-time images under a range of qualities. As such, compressed image processing and generation of lightweight deep learning architectures is a primary focus. With this in mind, we revisit Neural Architecture Search (NAS) to overcome these two notions in tandem, which automatically traverses a given architecture search space, and generates models that are competitive alongside hand crafted state-of-the-art models. Considering the automated manner in which NAS considers architectures, it is trivial to impose constraints within the NAS pipeline such that only lightweight models are generated.

On this basis, we deploy our DDS-NAS framework (Chapter 4) within object detection, under a two-stage (Faster-RCNN [11]) object detection network. The main contributions of this chapter are as follows:

– a novel framework, DDS-NAS-MULTI, that reformulates the DDS-NAS-DARTS framework (Chapter 4) in order to minimize the training duration of a given epoch within NAS for multi-label image classification and object detection, demonstrated to be effective for the commonplace NAS DARTS

([53]) approach.

– an efficient and novel approach for hard example mining within the multi-label image domain, that considers image dissimilarity as an alternative metric to hardness, and employs an autoencoder architecture that enforces an image similarity embedding in latent space, yielding efficient dissimilar image look-up from a *kd*-tree structure.

– generation of models that are intrinsically robust to biased datasets, whilst retaining competitive, near state of the art accuracy over common object detection benchmarks.

5.2 Proposed Approach

In this section we detail the necessary modifications to our DDS-NAS approach for deployment within multi-label domains. Defining our architecture space within neural architecture search (Section 5.2.1) for multi-label image classification and object detection represents the most significant alteration from the previous proposed DDS-NAS solution. Secondly, our hard example mining approach must retain efficiency within the *multi-label* object detection domain (Section 5.2.2). Finally, we require our autoencoder to be suitably trained upon multi-label datasets such that our *kd*-tree lookup within hard example mining yields suitable images (Section 5.2.3).

5.2.1 Neural Architecture Searcch

We deploy our DDS-NAS framework upon the commonplace convolutional object detection Faster-RCNN [11] architecture. We replace the feature extractor backbone with a searchable image classification network architecture akin to the DARTS [53] search space. Prior literature within NAS for object detection indicates an overwhelming research into the *architecture* space. We propose however that *efficient* training under a relatively simple backbone architecture search space can yield equally promising results. In fact, our results directly contradict previous work that suggest altering the backbone architecture is insufficient for generating optimal models [196]. Reconsidering the NAS search phase from a data perspective rather than architecture perspective is a worthwhile and relevant challenge.

To appropriately reformulate DDS-NAS, we consider two strategies. The first approach replaces the final DDS-NAS classification layer with the RPN head and ROI head from Faster-RCNN (DDS-NAS-Faster). We update the weights corresponding to *architecture* optimization with the loss propagated through the *entire* object detection network. Due to instability in joint training of the architecture and supernetwork weights within DARTS, exacerbated by the relative complexity of training an object detection network compared to image classification, we introduce Kendall Loss [252] to learn the optimal weighting between

box regression loss, classification loss, and region proposal network loss. In practice however, searching on a given object detection dataset within the context of an object detection task requires significant memory resources. Instead, we find that searching on a given object detection dataset within the context of multi-label classification is sufficient to generate an effective backbone network architecture that can be later used for object detection during the NAS evaluation stage. Moreover, this strategy (DDS-NAS-MULTI) is enough to demonstrate the effectiveness of our proposed hard example mining autoencoder modifications (Section 5.2.3).

5.2.2 Hard Example Mining

The primary consideration within multi-label domains is that we train our network architecture with multi-class, *multi-label* images. Multiple (different) classes may exist in a given image. In Chapter 4, we adopted a single *kd*-tree for each class to enable efficient class-aware image lookup. This approach is not possible within object detection (or even multi-label image classification). Instead, we propose embedding class labels within the latent space to have implicit class-balance approximation. The most dissimilar image in a dataset will likely be that of a different class. By explicitly encoding class labels within the embedding, our *kd*-tree lookup will yield an image with different classes to the input. We propose that if the class labels are sufficiently embedded during autoencoder training, we will require only one *kd*-tree that sufficiently maintains class balance during lookup. Section 5.2.3 explains this process in more detail.

The second consideration pertains to calculation of image *hard*-ness. The *hard*-ness of an image within image classification can be recovered from its classification confidence. Indeed we keep this approach during the search phase within DDS-NAS-MULTI.

Within DDS-NAS-Faster and object detection however, we are not just classifying the class of the images, but also where the objects within the images lie. Rather than trivially extracting classification scores from the final output of the network architecture, we must identify classification score on a per-image basis during the box prediction step within the object detector (after matching between predictions and ground truth). Given DDS-NAS-Faster is grounded within the two stage Faster-RCNN network, box prediction follows feature extraction. As such, classification loss of the feature extractor is a reasonable estimation of the performance of the searchable part of our network architecture, given the backbone feature extractor contains the only searchable layers. Our rationale is that conventional object detection training first pretrains the (image classification) feature extractor backbone on ImageNet [29], followed by joint training of the backbone and object detection head. Similarly, our NAS framework searches for the best (backbone) architectures with image classification loss as an estimator for our hard example mining approach. The final searched backbone architecture and object detection head can then be jointly trained end-to-end during the NAS evaluation

stage.

5.2.3 Training the Autoencoder

Following the work in Chapter 4, we still require our autoencoder to construct an embedding space wherein images can suitably reconstructed. In the context of object detection, we informally define this such that classes present within a given multi-label image must be reconstructed to a qualitatively comparable degree to the single-label images in Chapter 4. Therefore, without prior knowledge of the multi-label classes, sufficiently reconstructed objects in an image may in fact not be classifiable to the human eye. Further, similar images must be relatively close in the embedding space. Moreover, the input image dimensionality commonly employed within object detection is high (for instance, 300×300 [198], 640×640 [273], and even higher with Faster-RCNN [11]). Consequently, compared to the autoencoder presented in Chapter 4 for image classification, the dimension size by which our autoencoder must reduce the input image is proportionally much higher. Finally, it needs to encode class labels within the image embedding (Section 5.2.2).

With this final criterion in mind, we directly provide the object bounding boxes (in the form of an instance segmentation label mask) to the encoder during training. Trivially extending the Kendall Loss combination for an additional classification loss, the decoder is forced to recover object labels as well as appearance from the latent image representation, satisfying this requirement. This behaviour is illustrated in Tables 18 and 19. We first calculate the percentage of images a given class appears in the COCO [30] dataset (Default Distribution). We then calculate the (absolute) difference in class distribution from an initial, randomly sampled subset of images, and the Default Distribution. Finally, we calculate the absolute difference between distributions across every epoch where a new data subset was generated (because the mastery value on the previous subset exceeded the pre-defined mastery threshold).

The class-aware autoencoder trained with the additional fourth input channel (Table 18) yields an average update closer to the overall distribution across the COCO dataset (14.32 absolute difference compared to 18.94 after random initialization). With each data subset update, the difference gets closer to 0, as the class-aware autoencoder actively attempts to maintain a balance across the classes. Without the additional channel, distribution in fact becomes less representative (Table 19) than random (19.46 compared to 18.49). In this case, the hard example mining approach selects harder images without any effort to maintain dataset balance.

Class	Default Distribution	Initial Random Sample	Average Update
0	0	0	0
1	18.55	0.36	0.65
2	1.3	0.18	0.24
3	3.33	0.38	0.54
4	1.59	0.5	0.05
5	0.51	0.12	0.01
6	0.8	0.22	0.06
7	0.58	0.09	0.03
8	1.96	0.48	0.26
9	1.49	0.4	0.02
10	1.63	0.23	0.16
11	0.58	0.12	0.11
12	0	0	0
13	0.54	0.06	0.11
14	0.33	0.08	0.01
15	1.23	0.17	0.24
16	0.83	0.29	0.6
17	1.74	0.19	0.27
18	1.23	0.1	0.49
19	0.91	0.04	0.17
20	0.51	0.23	0.08
21	0.72	0.12	0.03
22	1.3	0.49	0.28
23	0.11	0.03	0.11
24	0.76	0.02	0.11
25	0.83	0.23	0.07
26	0	0	0
27	1.78	0.34	0.42
28	1.01	0.38	0.19
29	0	0	0
30	0	0	0
31	1.2	0.66	0.49
32	0.65	0.09	0.25
33	0.65	0.16	0.26
34	0.54	0.09	0.03
35	0.58	0.19	0.03
36	0.51	0.16	0.22
37	1.59	0.36	0.62
38	0.51	0.19	0.28
39	0.72	0.02	0.25
40	0.72	0.16	0.29
41	1.05	0.35	0.19
42	0.87	0.24	0.26
43	1.05	0.21	0.23
44	2.93	0.65	0.2
45	0	0	0
46	0.43	0.06	0
47	2.5	0.56	0.71
48	0.72	0.44	0.04
49	1.05	0.14	0.19
50	1.27	0.7	0.09
51	2.03	0.67	0.01
52	0.94	0.54	0.24
53	0.4	0.05	0.03
54	0.29	0.31	0.14
55	0.69	0.08	0.33
56	0.51	0.16	0.06
57	0.25	0.03	0.11
58	0.65	0.33	0.21
59	1.05	0.24	0.26
60	0.62	0.15	0.01
61	0.4	0.16	0.01
62	3.37	0.35	0.11
63	0.98	0.35	0.26
64	1.49	0.4	0.02
65	0.87	0.25	0.17
66	0	0	0
67	2.79	0.51	0.04
68	0	0	0
69	0	0	0
70	1.96	0.84	0.02
71	0	0	0
72	1.41	0.11	0.08
73	0.87	0.1	0.05
74	0.69	0.3	0.04
75	0.36	0.11	0.18
76	0.65	0.23	0.03
77	1.34	0.08	0.16
78	0.43	0.17	0.04
79	0.83	0.12	0.14
80	0	0	0
81	1.99	0.08	0.41
82	0.8	0.03	0.24
83	0	0	0
84	1.78	0.01	0.02
85	1.49	0.09	0.2
86	1.09	0.17	0.24
87	0.07	0.18	0.11
88	0.65	0.16	0.07
89	0.04	0	0.03
90	0.51	0.3	0.31
Total	100	18.94	14.32

Table 18: Class distribution from hard example mining employing an autoencoder trained *with* the fourth concatenated channel. Initial Random Sample and Average Update illustrate the (absolute) difference to the default COCO distribution across the entire dataset.

Class	Default Distribution	Initial Random Sample	Average Update
0	0	0	0
1	18.55	0.21	0.93
2	1.3	0.31	0.54
3	3.33	0.09	0.35
4	1.59	0.41	0.04
5	0.51	0.7	0.73
6	0.8	0.29	0.07
7	0.58	0.1	0.26
8	1.96	0.34	0.11
9	1.49	0.46	0.36
10	1.63	0.31	0
11	0.58	0.07	0.18
12	0	0	0
13	0.54	0.14	0.29
14	0.33	0.04	0
15	1.23	0.68	0.15
16	0.83	0.49	0.7
17	1.74	0.05	0.5
18	1.23	0.28	0.04
19	0.91	0.29	0.37
20	0.51	0.08	0
21	0.72	0.05	0.08
22	1.3	0.45	0.65
23	0.11	0.18	0.04
24	0.76	0.21	0.22
25	0.83	0.27	0.26
26	0	0	0
27	1.78	0.97	0.44
28	1.01	0.13	0.01
29	0	0	0
30	0	0	0
31	1.2	0.09	0.47
32	0.65	0.23	0.08
33	0.65	0.27	0.07
34	0.54	0.2	0.14
35	0.58	0.16	0.66
36	0.51	0.04	0.18
37	1.59	0.23	0.54
38	0.51	0.34	0.14
39	0.72	0.09	0.1
40	0.72	0.05	0.18
41	1.05	0.31	0.29
42	0.87	0.01	0.18
43	1.05	0.31	0.18
44	2.93	0.54	0.06
45	0	0	0
46	0.43	0.23	0.33
47	2.5	0.07	0.12
48	0.72	0.53	0.12
49	1.05	0.27	0.15
50	1.27	0.09	0.11
51	2.03	0.27	0.48
52	0.94	0.09	0.1
53	0.4	0.07	0.14
54	0.29	0.33	0.36
55	0.69	0.14	0.25
56	0.51	0.74	0.47
57	0.25	0.23	0.37
58	0.65	0.1	0.29
59	1.05	0.09	0.14
60	0.62	0.08	0.22
61	0.4	0.56	0.36
62	3.37	0.36	0.37
63	0.98	0.27	0.25
64	1.49	0.27	0.14
65	0.87	0.23	0.33
66	0	0	0
67	2.79	0.44	0.48
68	0	0	0
69	0	0	0
70	1.96	0.06	0.54
71	0	0	0
72	1.41	0.16	0.03
73	0.87	0.12	0.44
74	0.69	0.36	0.07
75	0.36	0.12	0.26
76	0.65	0.03	0.14
77	1.34	0.16	0.15
78	0.43	0.34	0.14
79	0.83	0.09	0.21
80	0	0	0
81	1.99	0.23	0.32
82	0.8	0.18	0.18
83	0	0	0
84	1.78	0.13	0.11
85	1.49	0.09	0.14
86	1.09	0.1	0.07
87	0.07	0.04	0.08
88	0.65	0.06	0.26
89	0.04	0	0
90	0.51	0.29	0.15
Total	100	18.49	19.46

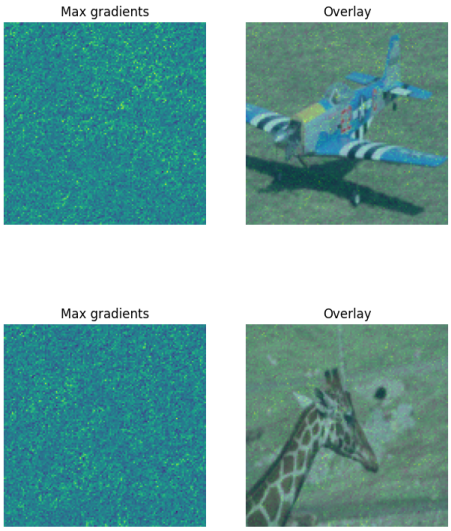
Table 19: Class distribution from hard example mining employing an autoencoder trained *without* the fourth concatenated channel. Initial Random Sample and Average Update illustrate the (absolute) difference to the default COCO distribution across the entire dataset.

Additionally, embedding object position information within the latent image (since labels are provided as an instance segmentation mask) acts as an attention map for autoencoder training (Fig. 40). The autoencoder network focuses on regions of a given image that correlate with objects (Fig. 40 (d)). Without this additional fourth channel during training, network attention is distributed more evenly throughout an image, often on regions without any objects at all (Fig. 40 (b)). We argue that this property yields relatively detailed reconstruction for objects within an image compared to the background, to which the first criterion pertains precisely. Indeed, reconstruction quality of the objects is better with this configuration (Fig. 43) compared to without (Fig. 42). Although background reconstruction quality is superior without the implicit attention layer, the distinction between foreground and background is relatively blurred (particularly apparent in the image of the bird). The image regions that enable *class-aware* encoding and image *hard*-ness (i.e. the objects) are all that are required to be reconstructed.

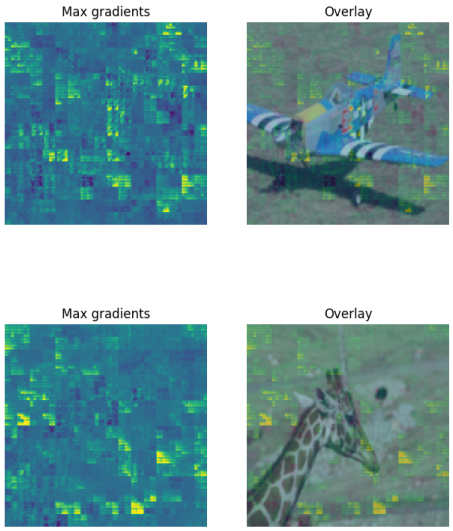
Inspired by the success of RetinaNet [201] within object detection, we additionally try replacing our classification loss term under Kendall Loss with Focal Loss [201]. By preventing easy negative examples from dominating cross entropy calculation, RetinaNet is able to focus on hard samples during training. We had hoped the same success would be attained within our reconstruction training. Easily classifiable objects would be penalized during training, forcing the encoder to focus on smaller or more occluded objects within an image. Instead however, reconstruction quality deteriorates (Fig 44). Neither foreground nor background details are reconstructed. In fact, there is very little disinguishabilty between the two. We can attribute this behaviour to the unsuitability of Focal Loss to image reconstruction. Larger objects within an image are easier to classify, and more easily detectable during object detection. The same objects are not necessarily more reconstructable however, as they have more pixels and fine-grained details that require encoding.

To achieve the necessary clustering capability to the same degree as Chapter 4, we once again exercise contrastive learning, which achieved great succcess within image classification. We consider Circle Loss to be most suitable. Where we previously discarded any superior performance it delivered in favour of speed during training, the more difficult object detection datasets instead necessitate adopting any and all possible performance improvements. However, conventional contrastive learning requires modifications before it can be applied to *multi-label* image datasets. Positive and negative classes do not make sense when there is more than one label per image. To this end, we propose two strategies to define positive and negative samples for a given anchor image.

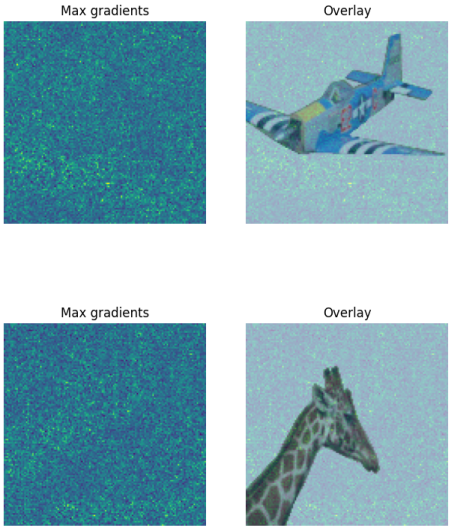
In both cases, we construct a similarity matrix for all images in the dataset prior to training. Under our first system, we adopt Jaccard Index (size of intersection divided by union) of the labels within an image (taking into consideration the common scenario wherein multiple objects of the same class exist within a given image). For the second system, we levy the autoencoder described earlier within this section.



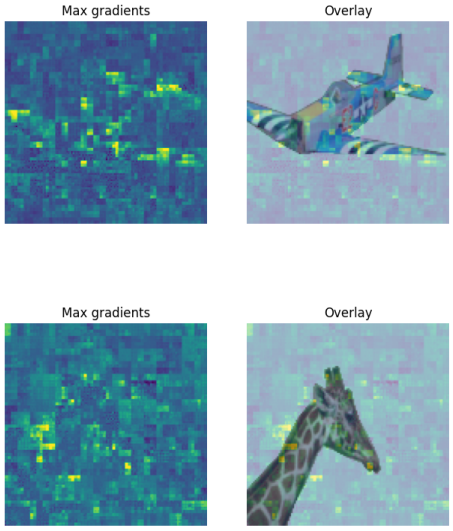
(a) Activations prior to training, without the implicit attention layer



(b) Activations during final epoch of training, without the implicit attention layer



(c) Activations prior to training, with the implicit attention layer



(d) Activations during final epoch of training, with the implicit attention layer

Figure 40: Prior to training, the highest gradients within the encoder network are evenly distributed throughout the input image (a,c). By adopting an implicit attention layer, our encoder acquires higher gradients from the image regions which contain foreground objects (d) compared to training with input images with three channels (b). This indicates that the network focuses on encoding these regions.

Considering the contractive loss term and implicit attention layer weighted under Kendall Loss, it is already *somewhat* able to identify similar images. Consequently, we are able to use the Euclidean distance between the embeddings of two given encoded images to construct the similarity matrix.[9] In both systems, we then determine a suitable threshold such that a percentage of images are considered positive to a given anchor image. Additionally, under both systems we pretrain the autoencoder in the same manner as defined earlier (contractive loss with an additional implicit attention layer), before training with Circle Loss.

Indeed, appropriate selection of triplets is critical to the learning process [250, 274]. Unsurprisingly therefore, different triplet selection strategies yield varied results. In fact, the second system (using an auxiliary autoencoder) loses the majority of its reconstruction ability during training under Circle Loss (Fig. 46), indicating that euclidean distance between images generates a very weak decision boundary between images. The ineffective Circle Loss term thus dominates training under Kendall Loss, such that any distinction between background and foreground objects is lost during training, and reconstructed images appear excessively blurred.

Triplet generation under the first system (using Jaccard Index) on the other hand yields very promising results. Foreground classes remain distinct after reconstruction (Fig. 45), suggesting more appropriate weighting towards reconstruction (MSE) loss (we know from Sec. 4.2.2.2 that Circle Loss does not contribute towards reconstruction ability). Fine grained details that are irrelevant with regards to image dissimilarity (such as tie colour, Fig. 45, left), are discarded. Only the information critical for image dissimilarity are retained within the small embedding space. We determine $nz = 64$ sufficiently large to be able to reconstruct only the objects within an image, given their quality appears qualitatively similar to images reconstructed by the final autoencoder in Chapter 4. Although this dimension is higher than within the DDS-NAS image classification counterpart, this is a reasonable allowance considering the higher image input size for object detection tasks compared to image classification.

Our final objective function for training the pretrained autoencoder is as follows:

$$\mathcal{L} = \mathcal{K}_1\mathcal{L}_{\text{recon}} + \mathcal{K}_2\mathcal{L}_{\text{triplet}} + \mathcal{K}_3\mathcal{L}_{\text{classification}}, \tag{12}$$

where $\mathcal{L}_{\text{recon}}$ denotes MSE loss:

$$\mathcal{L}_{\text{recon}} = \sum_{i=1}^{D}(x_i - y_i)^2, \tag{13}$$

L_{triplet} denotes triplet margin loss, with d representing the euclidean distance function, a, p and n representing

[9] Noting this second implementation yields significantly faster triplet generation, but this is a property we can safely disregard

anchor, positive and negative samples respectively, and m representing the margin (1.0):

$$L_{\text{triplet}} = \max(d(a,p) - d(a,n) + m, 0), \tag{14}$$

$\mathcal{L}_{\text{classification}}$ denotes (multi-label) binary cross entropy classification loss

$$\mathcal{L}_{\text{classification}} = -(b \log(q) + (1-b) \log(1-q)), \tag{15}$$

and where $\mathcal{K}_i$ denotes weighting of the ith loss term learned via Kendall Loss [252].

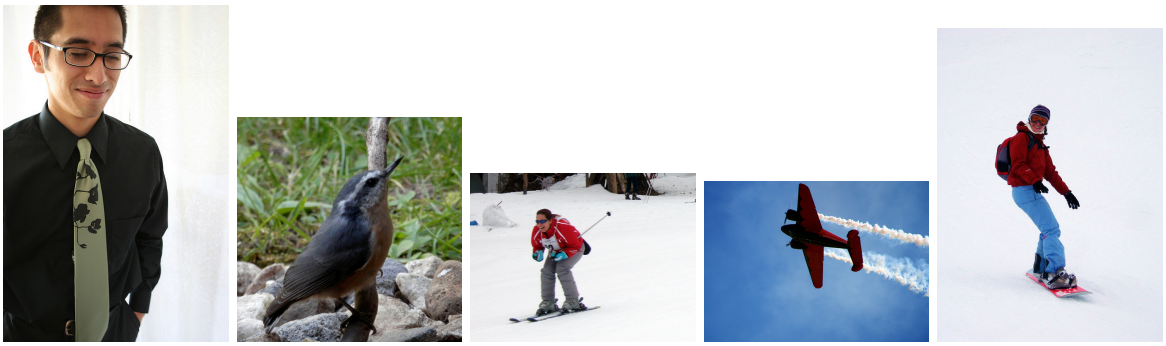


Figure 41: Ground truth images

Figure 42: Reconstructed images with a contractive autoencoder, ablating the use of the additional concatenated channel

Figure 43: Reconstructed images with a contractive autoencoder with the additional concatenated channel. Performance is qualitatively superior to Fig. 42.

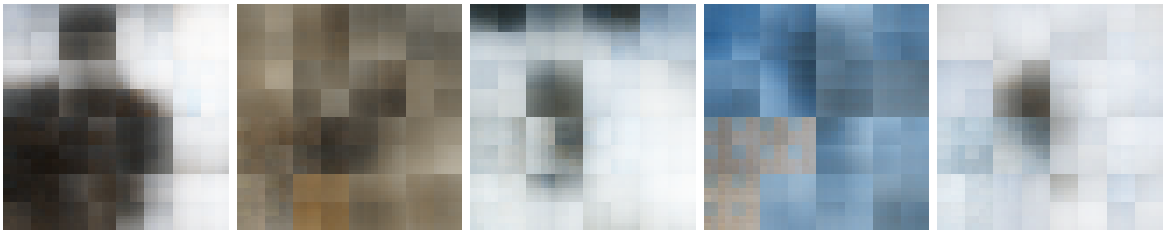


Figure 44: Reconstructed images with a contractive autoencoder with the additional concatenated channel and focal loss. Focal loss offers no benefit (compare with Fig. 42).

Figure 45: Reconstructed images with combined triplet ranking loss (system 1) and MSE reconstruction loss, within Kendall Loss. Performance is qualitatively superior to system 2 (Fig. 46).

Figure 46: Reconstructed images with combined triplet ranking loss (system 2) and MSE reconstruction loss, within Kendall Loss. Performance is qualitatively inferior to system 1 (Fig. 45).

5.3 Experimental Setup

To train DDS-NAS-MULTI, we adopt the same configuration as that of DDS-NAS during the search phase, with the exception of Cosine Annealing Learning Rate Scheduler in place of Cyclic Learning Rate Scheduler, with $lr = 0.025$ in both cases. We use image input size of 224×224. During the NAS evaluation stage, we use input images with a minimum size of 800×800, and maximum size of 1200×1200. Aspect ratio is retained during augmentation. We use $lr = 0.001$ for the first 100 epochs and fine tune with $lr = 0.0001$ for 50 epochs. For the autoencoders, we use input sizes equivalent to the respective DDS-NAS input image size, and latent space dimension $nz = 64$, epochs $= 100$.

5.4 Evaluation

Having validated the modifications to our Hard Example Mining approach necessitated by the move to object detection (Section 5.2.2-5.2.3 / Fig. 40-45), we present the performance of the architectures searched with the

DDS-NAS-MULTI framework (Table 20). For comparison, we present Faster-RCNN results (with a VGG-16 [59] backbone), in addition to the network architecture searched on CIFAR-10 in Chapter 4 transferred to COCO during the NAS evaluation stage.

Performance of both NAS searched architectures are comparable with the baseline Faster-RCNN results (within absolute 2%), indicating that NAS has potential within the object detection domain. Even architectures searched on the CIFAR-10 image classification dataset are generalized enough to be transferred to the COCO detection challenge. These results are aligned with the findings of Chapter 4, in that our DDS-NAS-DARTS framework is capable of distinguishing between high performance network architectures with high granularity, yielding a final network architecture that is sufficient for the challenging ImageNet and COCO tasks.

Searching for network architectures directly on the COCO dataset yields comparably performing architectures. Indeed, our DDS-NAS-MULTI searched architecture resembles the performance of DDS-NAS-DARTS transferred to COCO, indicating that our modifications to the autoencoder during the hard example mining process are sufficient to regulate the feature embedding space for multi-label images, such that similar images have similar encodings.

While we have compared to Faster-RCNN results as presented in the original literature, more recent results [18] improve upon this performance by introducing more effective backbone architectures. Our results are comparatively lacking compared to results found in the literature, but this does not represent a fair comparison. We expect that increasing computational resources to align the training batch size to commonplace training configurations (a batch size of at least 256) would signficantly improve this performance.[10] More importantly, we recall that the object detection performance of NAS approaches from the literature review in Chapter 2 is also higher. Future work utilizing these sophisticated backbone architecture search approaches [192, 199] in tandem with our DDS-NAS-MULTI approach would almost certainly exhibit even better performance. Indeed Chapter 4 illustrates the framework-agnostic characteristics of our NAS methodology that would allow such a combined approach. Nevertheless, our results are sufficient to demonstrate the effectiveness of the modified autoencoder training approach that DDS-NAS-MULTI depends on.

Furthermore, our DDS-NAS-MULTI approach offers one key advantage, in that it can search for architectures on a given object detection dataset directly. For practical real world applications, this has two benefits:

- The searched backbone architecture does not need to be trained on an auxiliary training dataset such as ImageNet to offer comparable performance.

[10] Our results indicate that training a Faster-RCNN network under our constrained resources significantly degrades performance compared to the original literature. Our results are better than the baseline in this regard.

NAS Approach	Mean Average Precision (%) ↑	
	$mAP_{0.5:0.95}$	$mAP_{0.5}$
Faster-RCNN	21.2	41.5
DDS-NAS-DARTS	19.8	35.5
DDS-NAS-MULTI	19.6	35.0

Table 20: Mean Average Precision of DDS-NAS-DARTS and DDS-NAS-MULTI compared to the Faster-RCNN baseline on MS-COCO val dataset.

- Overall end-to-end architecture search and training is considerably faster.

These two benefits represent a significant improvement aligned with the overall aims of our thesis to minimize deployment challenges within real world applications.

5.5 Conclusion

To conclude, modifying DDS-NAS to a multi-label domain represents a significant challenge that has been succesfully overcome, while retaining a class-aware hard example mining approach. In fact, the presence of object location availability during training can be utilized to enable a better image feature embedding representation for the hard example mining process, which we verify through illustrative visualizations. As such, we have shown that the proposed DDS-NAS-MULTI framework can directly search for network backbone feature extraction architectures without requiring an auxiliary training image dataset.

Moreover, access to additional training resources would likely yield even better network architectures given the potential to move to the DDS-NAS-Faster framework. Furthermore, and as alluded to in Section 5.2.3, hard example mining is already a prevalent point of discussion within object detection. The use of Focal Loss has seen widespread use within one-stage object detection networks [201, 273, 275], to prevent easy classification samples from dominating training loss. It is possible our hard example mining approach to some extent captures this same idea. In addition, recent work presents Commu [276] for metric learning within multi-label domains, exactly pertaining to the requirements of our autoencoder training. We leave these two research areas as possible directions for future research.

6

DDS-NAS under Compression

6.1 Introduction

While our DDS-NAS framework has been extensively evaluated on common image classification and object detection datasets, real-world applications offer unforeseen challenges. In practice, labelling datasets is not perfect. Noisy labelling negatively impacts supervised and semi-supervised approaches within computer vision, upon which our DDS-NAS approach is deployed. Class imbalance is more common, as data collection techniques are not perfect or restricted in terms of time and money. Moreover, occlusion and clutter is often more prevalent than in the datasets we have considered up until now. Additionally, compression artifacts impact different network architectures to different extents (see Chapter 3). Real-world datasets often change hands several times before reaching AI developers. Data collection is often outsourced to third parties with better infrastructure to either run the necessary simulations or collect data in the wild. Owing to the thousands of images that constitute computer vision data, it is not unusual to delegate labelling to third parties. Furthermore, ethical considerations must be upheld during data collection, requiring the data to be passed between multiple individuals or companies to ensure the proper regulations are adhered to. This may even include additional data processing such as image blurring or modifying for sensitive information retention, for instance. All these factors may introduce an unknown compression rate for a given image either due to company regulations or the image processing algorithms deployed. As such, understanding model performance with respect to varying compression rates is all the more important in order to guarantee expected performance, and to this end, we consider the performance of the DDS-NAS approach within compression.

6.2 Approach

To evaluate DDS-NAS with regards to compression, we utilize the same procedures employed in Chapter 3; adopting compression rates from the set $\{5, 10, 15, 50, 75.95\}$. We consider the same datasets in Chapter 4; MNIST [267], Fashion-MNIST [268], CIFAR-10 [93] and CIFAR-100 [93].

We analyse the impact of compression under three different scenarios, framing DDS-NAS as the concatenation of three principal components: an autoencoder (for hard example mining), NAS search phase (searching for the optimal network architecture), and NAS evaluation phase (training the searched network architecture from scratch). We evaluate the impact of compression at each of these three stages, in reverse order to best determine the most significantly effected DDS-NAS stage. We omit compression impact comparisons between DDS-NAS and regular two-stage NAS frameworks on the basis that the performance is identical. Dynamic data selection does not improve or degrade compression resilience.

6.2.1 DDS-NAS Evaluation Stage

NAS evaluation stage immediately follows NAS search. A given searched cell is stacked to yield a final image classification architecture. The NAS evaluation phase retrains this model from scratch. The image classifier network architecture most closely resembles the object detection (Faster-RCNN [11]) network architecture we evaluated in chapter 3, which demonstrated reasonable compression resilience especially after providing compressed training data. Lacking either the two-stream network architecture employed within human action recognition, or the dual-prediction task of human pose estimation, we expect a given image classifier, retrained on compressed images, to deliver the same compression resilience.

We adopt the same configuration for NAS evaluation as employed in Chapter 4. We use an SGD optimizer with $lr = 0.01$, momentum $\beta = 0.9$, and weight decay $wd = 3e^{-4}$. We train the network architecture for 300 epochs with a batch size of 32. Finally, we use the cells discovered by DDS-NAS in Chapter 4, with 4 layers for MNIST, 14 layers for Fashion-MNIST, and 12 layers for CIFAR-10. For CIFAR-100 we use the network architecture discovered under CIFAR-10.

6.2.2 DDS-NAS Search and Evaluation stage

The NAS search phase under DARTS (and our corresponding proposed DDS-NAS-DARTS) searches for the best classification cell architecture, in tandem with optimizing the super-network weights. Dual task learning has already been hypothesised to contribute to the low resilience of human pose estimation under compression (Section 3.4.3), suggestive of low compression resilience within NAS search as well.

We use the same hardness and mastery thresholds identified within Chapter 4 within our DDS-NAS framework, incorporating the same autoencoder (i.e. trained on uncompressed data). The same configuration for NAS search under DARTS as employed in Chapter 4 is used (Sec. 4.3.1). We search for the optimal cell architecture on each compressed dataset (each yielding six different cell architectures), and perform NAS evaluation stage on each final searched cell with the respective compressed data.

6.2.3 Retraining the Autoencoder

In order to determine the impact of compression on overall DDS-NAS performance after retraining the autoencoder, we consider two performance evaluation methods. We qualitatively evaluate autoencoder clustering ability when trained with the compressed dataset. We do not provide qualitative evaluation of the autoencoder with respect to reconstruction ability. In this regard, the autoencoder network is completely resilient. This is a fairly meaningless result however, considering we are reconstructing images with compression artifacts from a small embedding space. Reconstruction with uncompressed data via an autoencoder already yields pixelated images (as seen in Chapter 4).

For MNIST, Fashion-MNIST, and CIFAR-10 datasets, we adopt the same autoencoder as that employed by our DDS-NAS framework. For the grayscale datasets, this requires a contractive autoencoder. For CIFAR-10, we train the autoencoder with triplet ranking loss and MSE reconstruction loss, weighted via Kendall Loss. We adopt the same autoencoder training hyperparameters described in Chapter 4 (Sec. 4.3.2). As DDS-NAS CIFAR-100 final performance is determined after NAS evaluation stage using a network architecture searched with CIFAR-10, we do not evaluate the impact of compression on the autoencoder with the CIFAR-100 dataset.

Our selected autoencoder most closely resembles the SegNet architecture used in depth estimation (Sec. 3.2.2), which has been shown to be resilient to compression artifacts. We might expect this behaviour to be repeated in our autoencoder given the upsampling within the encoder-decoder architecture. However, the additional clustering required of our encoder complicates this matter.

6.3 Evaluation

In this section, we display the performance of DDS-NAS under compression applied to each DDS-NAS component (autoencoder, NAS search, and NAS evaluation, Table 21). We determine the impact of compression on a given component on overall network architecture performance, but also in contrast to the successive component, to better inform us of compression impact on the component in question. Finally, we propose possible explanations for any compression impact, where possible correlating performance with network architectures identified in Chapter 3.

6.3.1 DDS-NAS Evaluation Stage

In general, we can observe the expected trend that accuracy correlates with compression level. However, compression has very little impact on training performance of searched architectures on the MNIST dataset (red column). The lowest compression rate (95) achieves an accuracy of 99.65%, compared to 99.58% under

Dataset	JPEG Parameter Compression Level	Top-1 Accuracy (%) NAS Search Phase ↑		Top-1 Accuracy (%)	Top-1 Accuracy (%)
		Search Phase	Final Accuracy	NAS Evaluation Phase ↑	No Retraining ↑
MNIST	95	93.09	99.67	99.65	99.81
	75	89.68	99.69	99.66	99.79
	50	90.86	99.72	99.70	99.68
	15	91.50	99.70	99.62	99.57
	10	91.60	99.62	99.59	99.28
	5	90.90	99.60	99.58	77.93
Fashion-MNIST	95	59.20	92.28	94.15	94.17
	75	56.02	91.23	93.71	91.48
	50	55.63	92.04	93.18	88.46
	15	53.55	90.75	92.01	82.87
	10	53.13	89.18	91.57	78.72
	5	55.44	89.59	89.82	71.76
CIFAR-10	95	57.05	95.44	96.42	95.84
	75	44.31	92.48	93.42	87.99
	50	64.36	90.60	91.60	80.16
	15	50.87	84.34	85.58	54.09
	10	54.89	79.98	81.48	42.92
	5	37.69	71.94	72.68	26.13
CIFAR-100	95	-	77.59	80.63	77.86
	75	-	71.07	74.17	66.87
	50	-	67.27	70.09	57.73
	15	-	57.97	60.55	33.62
	10	-	53.31	55.94	22.81
	5	-	43.11	44.80	9.60

Table 21: The impacts of lossy image compression on the DDS-NAS search and evaluation phase

the highest compression rate (JPEG parameter 5). We can attribute this behaviour to the easiness of the dataset itself; compression artifacts have very little impact on the classifiability of the numbers (Fig.

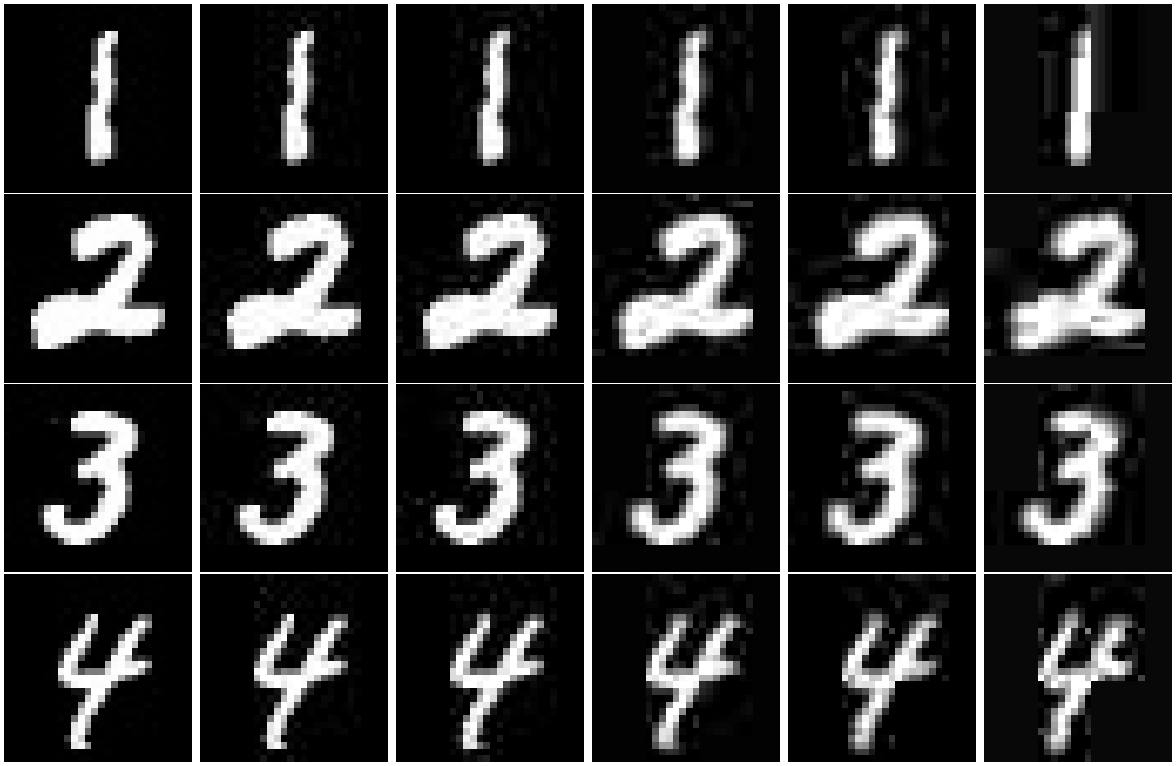


Figure 47: MNIST images under increasingly heavy compression rates (left: 95, right: 5)

Compression Level	Top-1 Accuracy (%) NAS Search Phase ↑	Top-1 Accuracy (%) NAS Evaluation Phase↑
95	90.69	92.02
75	89.93	91.02
50	89.09	89.93
15	84.01	84.64
10	80.62	81.50
5	71.65	71.31

Table 22: The impacts of lossy image compression during the NAS search and evaluation phase when using images with varying compression levels in the CIFAR-10 training set.

47). Any differences in DDS-NAS performance can be attributed to randomness within the NAS evaluation stage. Indeed, by comparing the NAS evaluation stage with the searched architecture trained on compressed data (red column) and uncompressed data (light blue column), we can observe a negligible performance difference. In many cases, retraining the network architecture offers little benefit. A significant difference can only be observed at the heaviest compression rate (JPEG parameter 5), where MNIST performance drops surprisingly low (given the ease with which images can be classified by a human at this compression level) down to 77.93%, and re-training the network architecture with data compressed at this level fully recovers network architecture performance (99.58%).

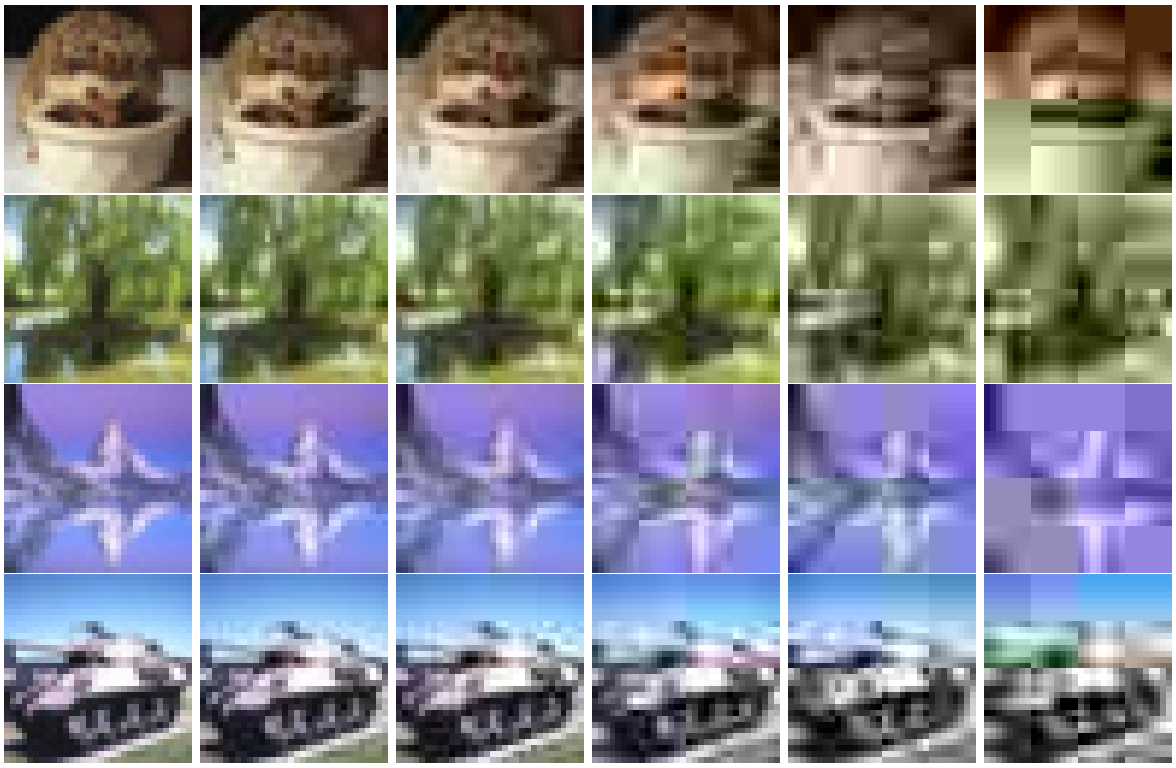


Figure 48: CIFAR-100 images under increasingly heavy compression rates (left: 95, right: 5). Examples include a hedgehog, a willow tree, a mountain scene, and a tank. Visible artifacts at high JPEG parameters are introduced during image resizing rather than compression (indeed JPEG parameter 95 introduces no visible compression artifacts).

For the harder tasks, the expected trend is more obvious. Fashion-MNIST performance difference is indiscernable until a JPEG compression level of 15, wherein accuracy drops by 2.27%. By JPEG compression level 5, performance has dropped by 4.6%, down to 89.82% Top-1 Accuracy (red column).

For CIFAR-10, accuracy drop is more considerable still. Even low compression rates receive performance degredation (3.1% drop at JPEG compression level 75 in the red column, for instance). At JPEG compression level 15, performance drops by 11.2%, and by level 5, it has dropped by 24.6% down to only 72.68% accuracy. The same rate of decline is observable for CIFAR-100, where there is a substantial drop in performance at JPEG compression level 15. The perceptible differences between images from CIFAR-10 and CIFAR-100 are much more apparent (Fig. 48), and their respective performance degradation follows. Of course, the performance is far higher than when we test without retraining the model on compressed images (light blue column).

The right most column in Table 22 can be directly compared to the identically coloured column in Table 21 (second from the right), which uses the same searched architecture. Each image in the training set is compressed at one randomly selected compression rate from the same JPEG Parameter set as before ({5,10,15,50,75,95}). It is evident that such a training setup has a detrimental impact on performance, especially when small amounts of compressions are used in the test dataset (a drop from 96.42% to 92.02% when using compression level 95 for instance). However, it is not unrealistic in real world scenarios to only have access to training data at a range of compression levels. If we compare the performance in Table 22 (red column) to the right most column in Table 21 (light blue), it is clear that performance is significantly improved. Unsurprisingly, if it is possible to train at the same compression level as will be used during testing, performance is recovered the most. Training with varying compression levels still recovers performance to some degree.

In all cases beyond the trivial MNIST setting, retraining a searched network architecture with compressed data (NAS evaluation phase) recovers network architecture performance, aligning with the findings of Chapter 3.

6.3.2 DDS-NAS Search and Evaluation Stage

By searching with compressed image data, we can once again observe that overall network architecture performance drops with higher compression rates. By comparing the dark blue and red columns (Table 21), MNIST performance is evidently unaffected when applying the NAS search phase on compressed image data. Meanwhile, Fashion-MNIST results marginally degrade compared to the easy MNIST task, where at high compression levels introducing compressed data to the NAS search phase causes performance to drop from 94.15% down to 92.28% at compression level 95 (dark blue vs. red columns). This behaviour is less apparent

at low compression levels (e.g. 89.82% down only to 89.59% at compression level 5).

CIFAR-10 results however indicate that performance drops considerably with compression levels of 15 or lower. The final accuracy is uniformly lower than performance obtained by DDS-NAS searched with uncompressed data, but where the evaluation stage uses compressed data (dark blue vs. red). At compression level 75, performance drops from 93.42% to 92.48%, and at compression level 5, there is a similarly marginal drop (72.68% down to 71.94%). However, if we make the same comparisons (dark blue vs. red) with CIFAR-100 data, a more interesting behaviour is observable. With low compression (JPEG quality level 95), performance drops by 3.9% (80.63% down to 77.59%). Similar degradation occurs at every other compression level. This would suggest that searching on compressed data generates less generalizable cell architectures; CIFAR-10 final performance is relatively unaffected but CIFAR-100 performance with the same network architecture is worse. The same behaviour can be seen by comparing the two columns in Table 22. Performing NAS search with varying compression levels in the dataset generates worse performing architectures than architectures searched with uncompressed data.

At the very least, searching on compressed data is *somewhat* harmful. Network architecture performance having searched on compressed data (dark blue column) is *uniformly* worse than searching on uncompressed data (red column). Performance degredation at a given compression level on a given dataset is however no worse than performance at a different compression level on the same dataset. CIFAR-10 performance is at most 1.5% percentage points lower when searching with compressed data than searching with uncompressed data. Given the dual learning task during NAS search (simultaneous architecture and network weight optimization), inaccuracies in training introduced by compression artifacts are exacerbated. Once again, these findings align with those of Chapter 3, where the dual prediction task within the human pose estimation network architecture induced lower performance accuracy when training with compression compared to other network architectures.

However, this behaviour is not an optimization gap problem. Stacking the network architecture cell searched with compressed data does not yield inferior final network architecture performance after the NAS evaluation stage (Table 23). NAS evaluation stage performance on uncompressed image data does not depend on the compression level at which a network architecture cell was searched; all searched architectures achieve roughly the same accuracy (between 95.39% and 95.60%). Performance differences between searched cells are only noticable when used during the NAS evaluation stage with compressed data (dark blue column, Table 21). Therefore, any differences in final network architecture performance between searching with and without compressed image data (dark blue vs. red columns, Table 21) must stem from the NAS search phase itself rather than the optimization gap problem induced from stacking a searched cell.

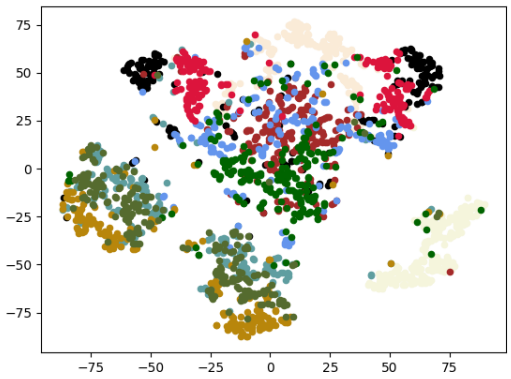
Compression Level During Search	Final Searched Performance
95	95.56 ± 0.10
75	95.54 ± 0.05
50	95.60 ± 0.12
15	95.59 ± 0.06
10	95.39 ± 0.18
5	95.57 ± 0.21

Table 23: The impacts of lossy image compression on the NAS optimization gap, with CIFAR-10 data. The NAS search phase uses compressed image data but the stacked cell is evaluated with uncompressed data

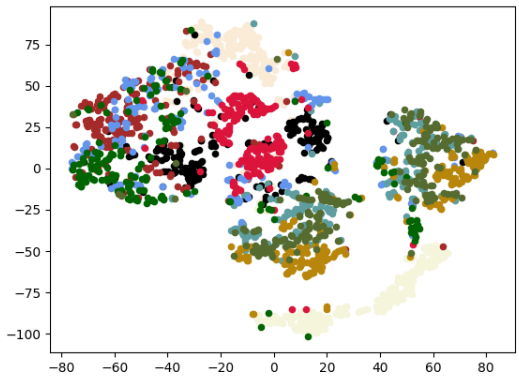
6.3.3 Retraining the Autoencoder

Generally, autoencoder training with compressed data exhibits reasonable resilience to compression. Across all compression levels, an autoencoder retrained with compressed data remains able to achieve sufficient clustering for MNIST and Fashion-MNIST datasets (Fig. 49, 50). However, for CIFAR-10, clustering ability deteriorates rapidly (Fig. 51). Compression introduces artifacts such that the autoencoder does not embed discriminative features, reminiscent of prior literature [41]. As such, despite employing contrastive learning, the autoencoder is unable to effectively *rank* the images. We therefore do not present results with the autoencoder trained with compressed data as results are inline with the ablation studies with bad autoencoders (Sec. 4.4.2.2), and yield no further insights.

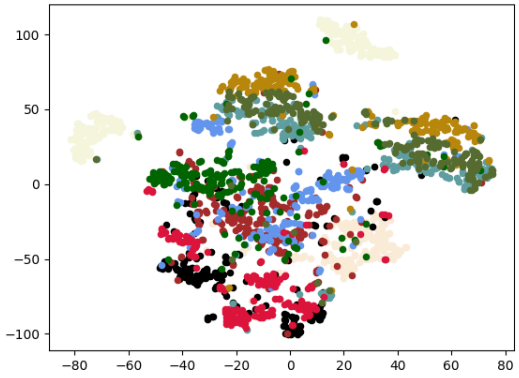
Interestingly however, we have already observed that using an autoencoder trained on the uncompressed images is sufficient for the DDS-NAS search phase (Sec. 6.2.3). Indeed, the autoencoder trained on the original CIFAR-10 uncompressed images is able to cluster the corresponding compressed images (Fig. 52); its ranking ability has not been hampered from re-training. Only at the heaviest compression rate (JPEG quality parameter 5) does the clustering deteriorate. Even at this level however, clustering between classes exists, albeit less distinctly than with lighter compression. The same behaviour can be expressed quantatively (Table. 24), where using the autoencoder trained on the original CIFAR-10 uncompressed images exhibits greater separation between the centroids of clusters, at all compression levels (higher minimum, maximum, and mean distances between centroids).



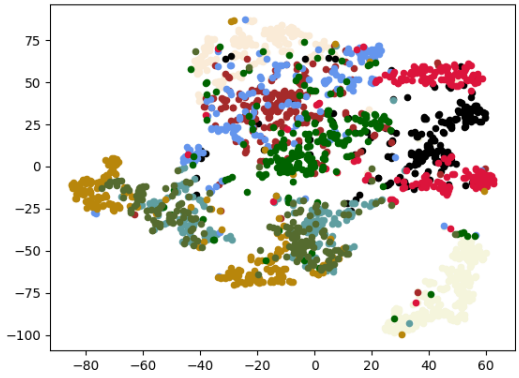
JPEG Compression Level 95



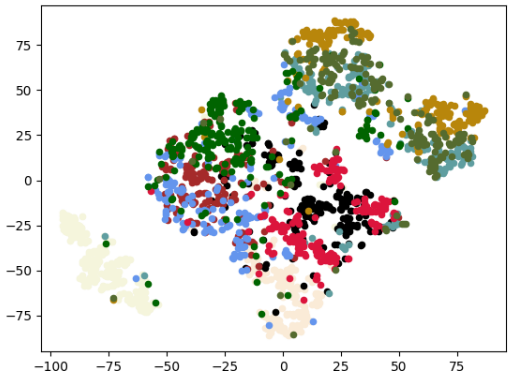
JPEG Compression Level 75



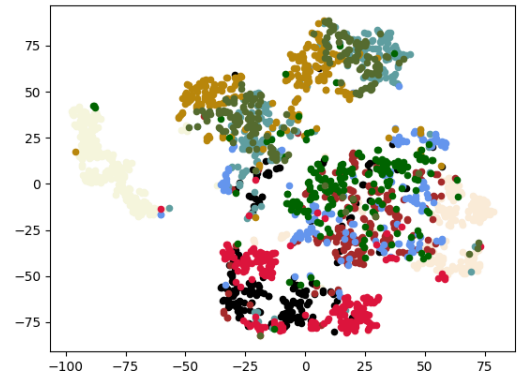
JPEG Compression Level 50



JPEG Compression Level 15

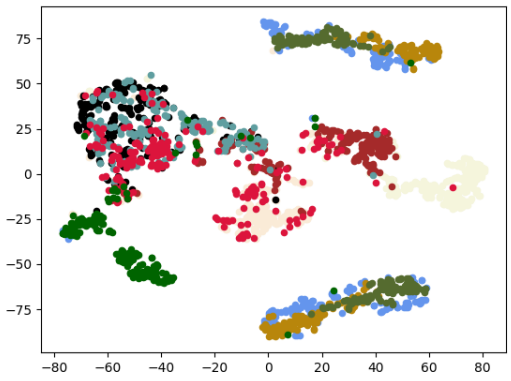


JPEG Compression Level 10

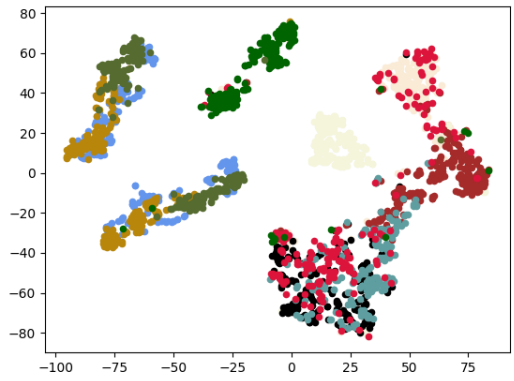


JPEG Compression Level 5

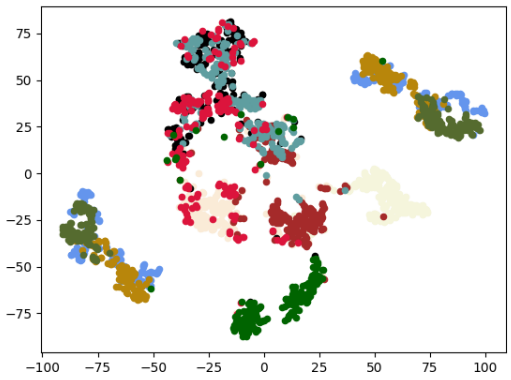
Figure 49: Autoencoder clustering trained with contractive loss on MNIST data compressed at six different rates



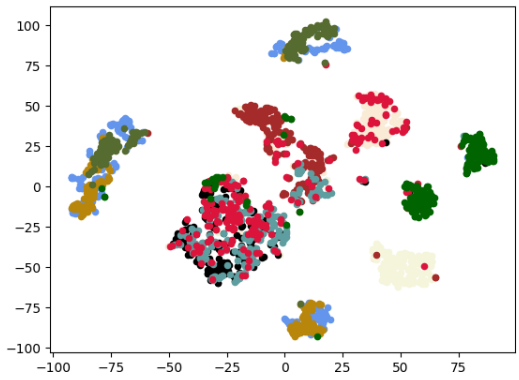
JPEG Compression Level 95



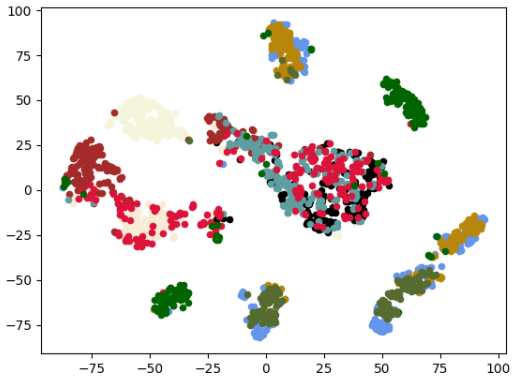
JPEG Compression Level 75



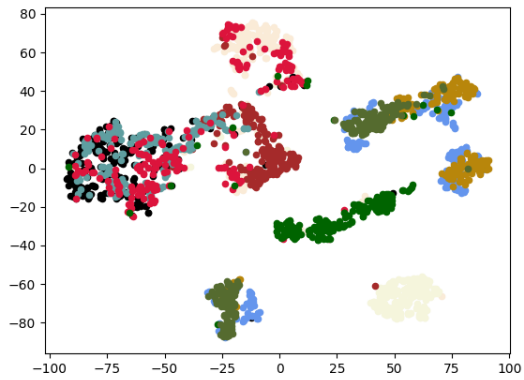
JPEG Compression Level 50



JPEG Compression Level 15

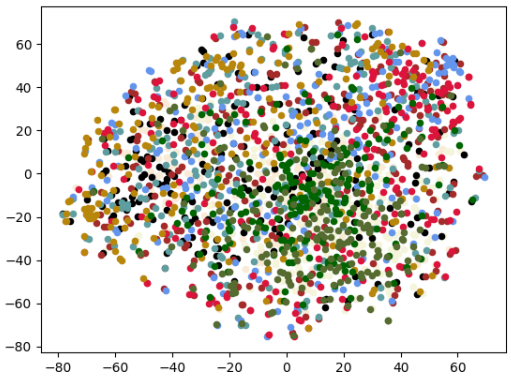


JPEG Compression Level 10

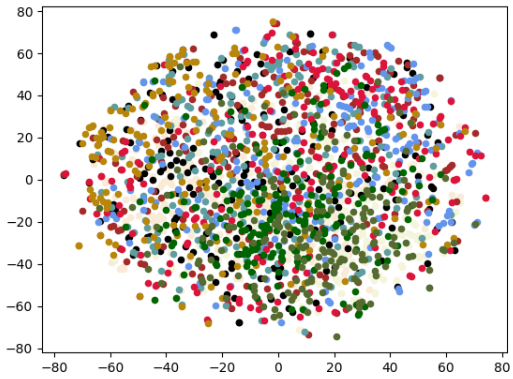


JPEG Compression Level 5

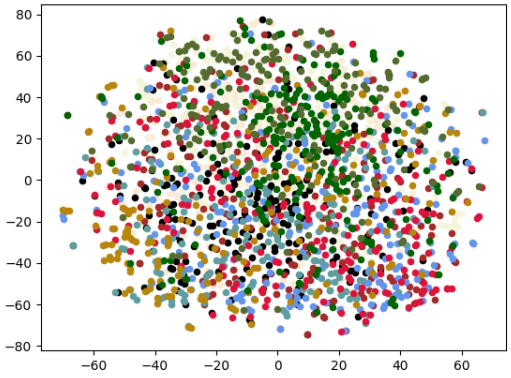
Figure 50: Autoencoder clustering trained with contractive loss on Fashion-MNIST data compressed at six different rates



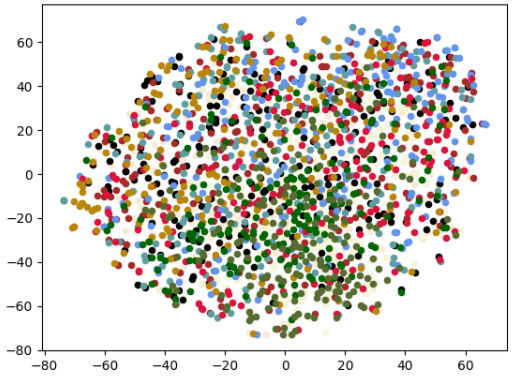
JPEG Compression Level 95



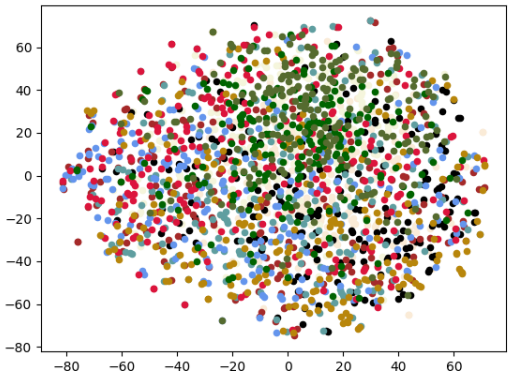
JPEG Compression Level 75



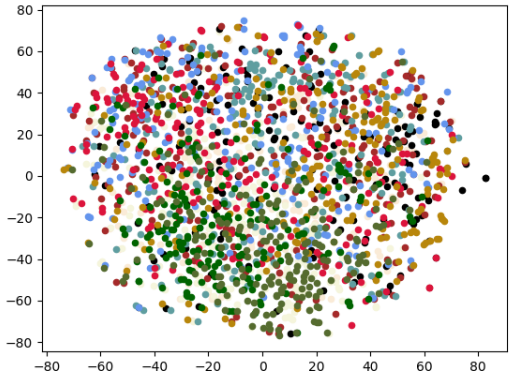
JPEG Compression Level 50



JPEG Compression Level 15

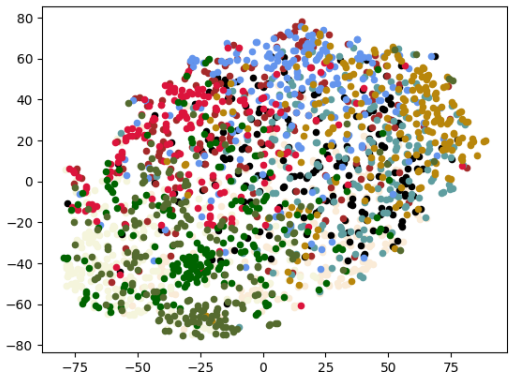


JPEG Compression Level 10

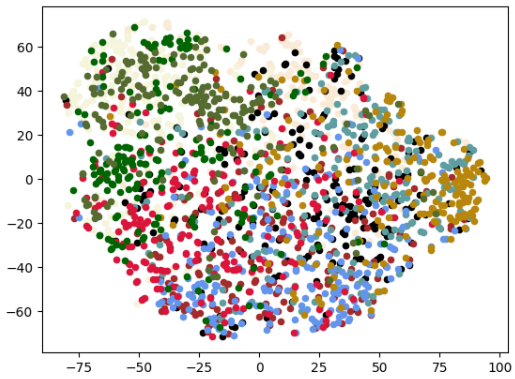


JPEG Compression Level 5

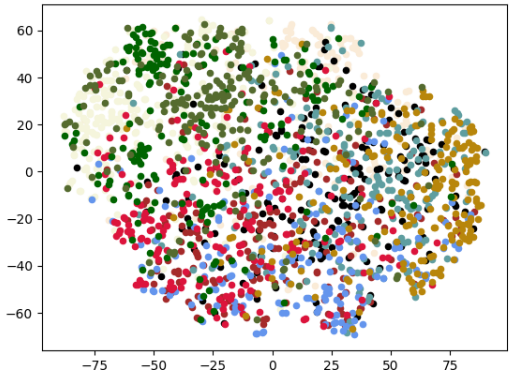
Figure 51: Autoencoder clustering trained with triplet ranking loss and MSE reconstruction loss (weighted via Kendall Loss) on CIFAR-10 data compressed at six different rates



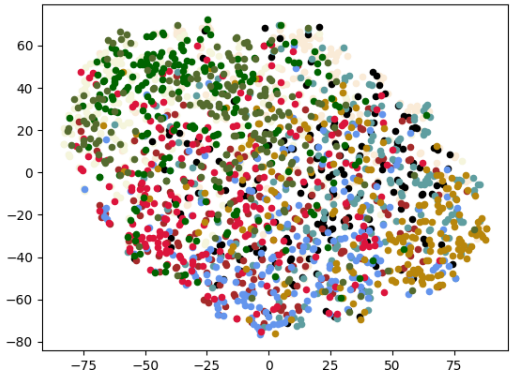
JPEG Compression Level 95



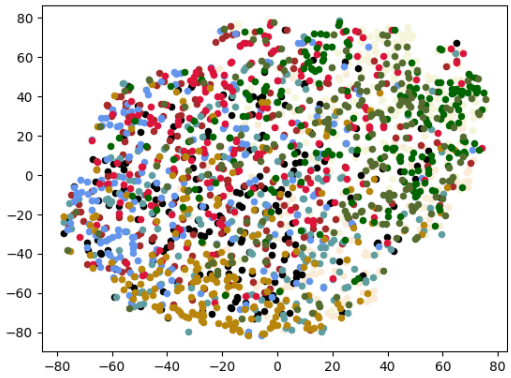
JPEG Compression Level 75



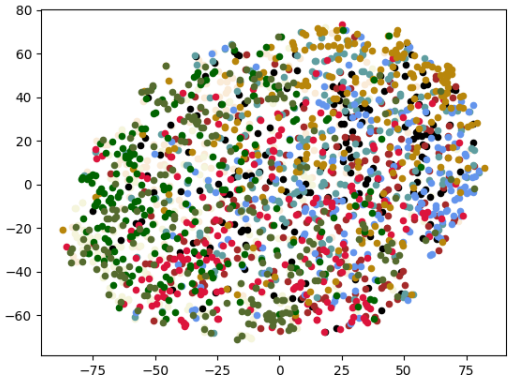
JPEG Compression Level 50



JPEG Compression Level 15



JPEG Compression Level 10



JPEG Compression Level 5

Figure 52: Autoencoder clustering trained with triplet ranking loss and MSE reconstruction loss (weighted via Kendall Loss) on uncompressed CIFAR-10 data, evaluated on CIFAR-10 compressed at six different rates

	Centroids Calculated from Mean			Centroids Calculated from Median		
Compression Level	Min	Max	Mean	Min	Max	Mean
95 (uncompressed)	264.13	4511.85	1849.61	363.01	5840.70	2510.52
95 (compressed)	77.61	1406.52	530.24	78.06	2002.90	80.83
75 (uncompressed)	250.97	4257.21	1671.91	328.43	5847.24	2421.12
75 (compressed)	133.66	1986.87	785.99	162.19	2231.08	887.82
50 (uncompressed)	197.43	4189.13	1661.49	258.08	5843.39	2314.83
50 (compressed)	52.75	1600.03	597.80	115.83	2736.34	1096.60
15 (uncompressed)	194.03	3704.38	1588.54	207.03	4940.62	2069.28
15 (compressed)	87.34	1282.32	478.40	118.04	2022.57	801.19
10 (uncompressed)	178.33	2987.64	1281.75	238.04	4035.60	1795.25
10 (compressed)	53.27	1118.21	431.53	64.82	1644.25	713.18
5 (uncompressed)	157.51	2764.08	1129.16	226.20	3997.81	1732.60
5 (compressed)	40.67	1124.42	431.31	93.47	1785.21	736.49

Table 24: The squared minimum, maximum and mean distances between the centroids of each cluster for the autoencoders trained with and without compressed images at each compression level. Centroids are calculated both from the mean and median of each point in the cluster.

6.4 Conclusion

With the findings across all DDS-NAS stages in mind, we recommend training the autoencoder and conducting the NAS search stage with uncompressed data if possible. Given the dependency of DDS-NAS on the autoencoder ability to cluster, for which training with compressed data is harmful, using compressed data results in significant overall performance degredation. Using compressed data during the search phase is to a lesser extent also harmful, likely owing to the multi-task training objective. Therefore, to obtain the best performance within the compressed image domain, only using compressed data during the NAS evaluation stage is recommended.

It is unfair to conclude that NAS is not resilient to compressed data however, despite the *relative* drawbacks of using compressed images during the NAS search phase. In general, both the NAS search phase and the overall two-stage gradient-based NAS approach exhibit considerable resilience to compression compared to several of the network architectures identified in Chapter 3. Indeed incorporating compressed data into the NAS search phase improves performance compared to using no compressed data at any stage (Table 21: peach vs. light blue). Given gradient-based NAS (and there is no reason to believe the same conclusions would not extend to evolutionary and reinforcement-learning approaches) can be divided into a search and evaluation phase, we simply recommend a more nuanced approach for optimal results wherein compression is only utilized during certain components of the overall training pipeline. We leave the investigation into the impacts of compression on reinforcement learning, evolutionary approaches, and prediction-based approaches to NAS to future research.

Conclusion

Deployment within real world scenarios is an incredibly complex challenge that presents many issues to be addressed. One primary consideration in contemporary computer vision AI approaches is data acquisition and maintenace thereof. Image data compression is consequently ubiqituous to AI development pipelines. Secondly, neural architecture search has received growing attention in recent years towards generating efficient network architectures without compromising final network accuracy. However, its critical associated long search phase is a major drawback within current NAS solutions. To this end, this thesis addresses these two deployment challenges in tandem, providing answers to the following research questions:

- How much does compression impact neural network performance, and by *how much* can this be recovered?
- Can we improve the NAS search phase speed purely from the perspective of input data?

By answering these questions, we can improve the AI / Machine Learning development pipeline at multiple stages, with minimal overhead from modifying existing approaches. Chapter 3 addresses this first research question, providing empirical evidence not only that compression impacts both accuracy and (where relevant) racial bias of neural network architectures, but also by precisely how much the performance degradation can be recovered.

The latter half of the thesis provides a solution to the second research question, demonstrated to be agnostic to and deployable alongside existing NAS frameworks. Drawing inspiriation from hard exampling mining [79, 84] and curriculum learning [80, 88, 82], we provide a novel dynamic dataset reformulation strategy to minimize data processed by any gradient-based neural network training strategy, but best suited to NAS. We identify the key contributions of this thesis in the following section.

7.1 Contributions

Chapter 3 conducts an extensive investigation into the impacts of compression across SegNet segmentation, monocular depth estimation GAN, two-stream human action recognition, OpenPose pose estimation, and

Faster-RCNN object detection network architecture, upon which some of the most well received recent literature (e.g. Stable Diffusion [255], DALLE-2 [277], Yolov7 [273]) is based. The impact of compression on these aforementioned architectures both before and after retraining the network with compressed image data is determined. Empirical evidence indicates that retraining most network architectures on compressed data largely recovers network performance even with compression rates as high as 15%. Furthermore, network architectures with upsampling layers exhibit higher resilience to compression (performance of a retrained SegNet network degrades by only 3.5% during inference at a compression rate of 5%).

Chapter 3 additionally identifies to what extent compression affects the ArcNet facial recognition network architecture, not just in terms of overall accuracy but also with respect to racial bias. Again, network performance on compressed images during inference is recovered if the network is retrained with compressed data. However, there is a non-uniform degradation across racial phenotypes at high compression levels, which can only be ameliorated to a limited extent by retraining with compressed images. Instead, we demonstrate that chroma-subsampling during JPEG compression is the primary cause of this performance loss and removing that component ameliorates the degradation.

Chapter 2, comprising the first literature review for contemporary NAS solutions under a unified and consistent terminology, highlights the scarcity of framework-agnostic NAS solutions that do not consider architecture generation methodology as a means to overcome the arduous NAS search phase. To this end, Chapter 4 approaches the issues in NAS deployment from a data perspective, presenting an end-to-end NAS solution (DDS-NAS) that achieves a NAS search phase speed up of one order of magnitude without loss in searched network architecture performance, and can be deployed alongside any existing NAS framework. By adopting a novel combined hard example mining and curriculum learning approach to dataset sampling, the volume of images processed in a given NAS search epoch is minimized without introducing class imbalance. To realise a fast dataset reformulation at a given epoch based on the curriculum, we are the first to consider latent space image representation disimilarity to mine appropriately *hard* images. Chapter 4 thus presents a thorough benchmark across autoencoder architectures that predate adopting skip-connections, in regards to latent space clustering and reconstruction capabilities.

To demonstrate the effectiveness of a hard exampling mining approach across all NAS methodology (and thus the suitability of a solution which only modifies the *data* presented to NAS), we incorporate *hard* mined data into a prediction-based NAS framework. In doing so, we further validate our DDS-NAS approach, and are able to consistently seek a network architecture with higher accuracy.

Chapter 5 extends our DDS-NAS approach to the multi-label image domain. Building upon the hard example mining approached in the preceding chapter, we incorporate an additional fourth channel into the autoencoder input layer. By training the autoencoder to encode the fourth channel image segmentation mask

within the latent space, the latent image representation encodes class label. Our solution can consequently be deployed within multi-label domains such as object detection. Moreover, the addition of the fourth channel acts as an attention layer for the encoder, improving overall capacity to capture the distinguishable regions of a given input image critical to our image-disimilarity-based *hard* mining objective.

We finally tie the two primary focuses of the thesis together in Chapter 6, in which we determine the impacts of compression on the commonplace two-stage NAS framework, and draw parallels with the findings of Chapter 3 where possible. Overall this thesis demonstrates the importance of data-driven approaches to common real-world deployment challenges. We produce an entirely novel approach for this task termed DDS-NAS, and comprehensively validate each of its components. With minimal modification to existing network architecture, by only altering the data presented to a given neural network, we can substantially reduce the impacts of compression on overall network performance, racial bias, and achieve a demonstrable NAS search phase speed up. Of course these results cannot be obtained freely, and the limitations of our work are addressed in Section 7.2.

7.2 Limitations and Future Work

While our thesis makes a substantial start to approaching current deployment issues via modification of data, it is not without its limitations. This section identifies such cases and presents possible directions for future research to expand on our approach.

7.2.1 Verifiability

We firstly acknowledge the complexity of our DDS-NAS solution, given the intertwining of several component features. The delicate relationship between the curriculum learning and hard exampling mining approach via latent space dissimilarity image representation, especially after extending to the multi-label domain, necessitates considerable verification. We have already visualized how the GANomaly autoencoder draws its attention to salient image regions after including an additional fourth instance segmentation input channel. Further visualisation to explain how this behaviour leads to a *class-aware* latent image represention would be particularly prudent, and could indicate potential architectural changes to better capture distinguishing features within the latent space. Majumdar et al. [41] occlude different facial features for a given input image for a facial recognition network to determine where the network architecture concentrates under varying levels of augmentation. Adopting the same or similar explainable AI approach (e.g. Grad-CAM [278]) could inform how the network encodes class labels.

Additionally, our image dissimilarity based *hard* example mining approach affords the generation of cell

architectures capable of high accuracy when stacked even though less image data is processed during the NAS search phase. The need to produce architectures that can disentangle *hard* images thus makes intuitive sense why DDS-NAS helps bridge the optimization gap compared to random image sampling within DARTS and other gradient-based NAS frameworks. However, additional verification beyond our current empirical evidence would better explain why this happens. In general, our work only touches the domain of AI explainability, which provides several techniques that our complex solution would benefit from greatly.

7.2.2 Scope

The proposed DDS-NAS solution utilizes a latent-representation based approach to *hard* example mining, in turn providing the first step to a significant contribution to reducing the NAS search phase duration. However, there already exists a considerable wealth of literature that calculates image similiarty. There exist several measures that consider pixel-wise absolute difference between images, including MSE and PSNR. These metrics are still prevalent within modern machine learning solutions, such as image reconstruction [15, 225, 279]. However, these measures are unable to capture perceived difference. Compression artifacts and noise for instance, introduce considerable changes at a pixel level but little perceptible difference to the human visual system. To this end, SSIM [280] extracts structural information from an image, from which perceived image differences are a reasonable approximation, rather than quantifying visibility of a given difference.

Perceptual score [281] as a metric for image similarity adopts the same philosophy. By considering Euclidean distance between the *perceptual features* (i.e. output of the penultimate layer of a pre-trained VGG [59] network), we necessarily quantify the differences between *structural* information, as high-dimensional deep features from the VGG network encode portions of the image with meaning (assuming the network is sufficiently trained). As eloquently stated, "a good feature is a good feature" [282]: features that are effective for image classification and detection are sufficient for perceptual image similarity judgements. While our novel approach can be efficiently incorporated into a *kd*-tree structure to compute image dissimilarity, further work can be done to compare it to existing methodology.

References

[1] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," *arXiv e-prints*, 2012.

[2] Y. Wu, A. Liu, Z. Huang, S. Zhang, and L. V. Gool, "Neural architecture search as sparse supernet," in *Thirty-Fifth AAAI Conference on Artificial Intelligence*, February 2021, pp. 10 379–10 387.

[3] X. Zheng, X. Fei, L. Zhang, C. Wu, F. Chao, J. Liu, W. Zeng, Y. Tian, and R. Ji, "Neural architecture search with representation mutual information," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2022, pp. 11 912–11 921.

[4] X. Du, T.-Y. Lin, P. Jin, G. Ghiasi, M. Tan, Y. Cui, Q. V. Le, and X. Song, "Spinenet: Learning scale-permuted backbone for recognition and localization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 589–11 598.

[5] C. Liu, L. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei, "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation."

[6] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv e-prints*, 2017.

[7] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *2015 IEEE International Conference on Computer Vision*, December 2015, pp. 1520–1528.

[8] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *Proceedings of the European Conference on Computer Vision*, vol. 9912, October 2016, pp. 483–499.

[9] F. Amherd and E. Rodriguez, "Heatmap-based Object Detection and Tracking with a Fully Convolutional Neural Network," *arXiv e-prints*, Jan. 2021.

[10] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, 2017.

[11] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*, December 2015, pp. 91–99.

[12] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1302–1310.

[13] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems*, Dec. 2014, pp. 568–576.

[14] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proceedings of the 28th International Conference on Machine Learning*. Omnipress, Jun. 2011, pp. 833–840.

[15] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations*, Apr. 2014.

[16] S. Yucer, F. Tektas, N. A. Moubayed, and T. P. Breckon, "Measuring hidden bias within face recognition via racial phenotypes," in *Winter Conference on Applications of Computer Vision*, January 2022, pp. 3202–3211.

[17] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2261–2269.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[19] R. Szeliski, *Computer Vision - Algorithms and Applications, Second Edition*, ser. Texts in Computer Science. Springer, 2022.

[20] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278 – 2324, 12 1998.

[21] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, p. 15271554, July 2006.

[22] A. Voulodimos, N. Doulamis, A. Doulamis, E. Protopapadakis, and D. Andina, "Deep learning for computer vision: A brief review," *Intell. Neuroscience*, vol. 2018, Jan. 2018.

[23] Y. You, Z. Zhang, C.-J. Hsieh, J. Demmel, and K. Keutzer, "Imagenet training in minutes," in *Proceedings of the 47th International Conference on Parallel Processing*, 2018.

[24] Z. Liu, H. Mao, C. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2022, pp. 11 966–11 976.

[25] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, Dec. 2014, pp. 2672–2680.

[26] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4396–4405.

[27] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[28] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proceedings of the European Conference on Computer Vision*, August 2020, p. 213229.

[29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, p. 211252, Dec. 2015.

[30] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," in *Computer Vision - ECCV 2014 - 13th European Conference*, ser. Lecture Notes in Computer Science, vol. 8693, September 2014, pp. 740–755.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *International Conference on Neural Information Processing Systems - Volume 1*. Curran Associates Inc., 2012, pp. 1097–1105.

[32] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–44, May 2015.

[33] L. Liu, W. Ouyang, X. Wang, P. W. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *Int. J. Comput. Vis.*, vol. 128, no. 2, pp. 261–318, 2020.

[34] I. Lee, "Big data: Dimensions, evolution, impacts, and challenges," *Business Horizons*, vol. 60, no. 3, pp. 293–303, 2017.

[35] G. K. Wallace, "The JPEG still picture compression standard," *Commun. ACM*, vol. 34, no. 4, pp. 30–44, Apr. 1991.

[36] D. Le Gall, "MPEG: A video compression standard for multimedia applications," *Commun. ACM*, vol. 34, no. 4, pp. 46–58, Apr. 1991.

[37] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Cir. and Sys. for Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[38] S. Karahan, M. K. Yildirim, K. Kirtaç, F. S. Rende, G. Butun, and H. K. Ekenel, "How image degradations affect deep cnn-based face recognition?" in *International Conference of the Biometrics Special Interest Group*, ser. LNI, vol. P-260, September 2016, pp. 313–320.

[39] F. Yang, Q. Zhang, M. Wang, and G. Qiu, "Quality classified image analysis with application to face detection and recognition," in *24th International Conference on Pattern Recognition*, August 2018, pp. 2863–2868.

[40] P. Terhörst, J. N. Kolf, N. Damer, F. Kirchbuchner, and A. Kuijper, "Face quality estimation and its correlation to demographic and non-demographic bias in face recognition," in *International Joint Conference on Biometrics*. IEEE, September 2020, pp. 1–11.

[41] P. Majumdar, S. Mittal, R. Singh, and M. Vatsa, "Unravelling the effect of image distortions for biased prediction of pre-trained face recognition models," in *International Conference on Computer Vision Workshops*, 2021, pp. 3779–3788.

[42] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size," *arXiv e-prints*, 2016.

[43] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *Computing Research Repository*, August 2017.

[44] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 4510–4520.

[45] A. Howard, M. Sandler, B. Chen, W. Wang, L.-C. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le, "Searching for mobilenetv3," in *International Conference on Computer Vision*, 2019, pp. 1314–1324.

[46] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems*, December 2015, p. 11351143.

[47] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," in *5th International Conference on Learning Representations*.

[48] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, no. 1, p. 68696898, jan 2017.

[49] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave gaussian quantization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, July 2017, pp. 5406–5414.

[50] G. Cazenavette, T. Wang, A. Torralba, A. A. Efros, and J.-Y. Zhu, "Dataset distillation by matching training trajectories," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2022, pp. 10 718–10 727.

[51] T. Nguyen, Z. Chen, and J. Lee, "Dataset meta-learning from kernel ridge-regression," in *9th International Conference on Learning Representations*, May 2021.

[52] C. Pham, T. Hoang, and T.-T. Do, "Collaborative multi-teacher knowledge distillation for learning low bit-width deep neural networks," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, January 2023, pp. 6435–6443.

[53] H. Liu, K. Simonyan, and Y. Yang, "DARTS: differentiable architecture search," in *7th International Conference on Learning Representations*, May 2019.

[54] S. F. Dodge and L. J. Karam, "Understanding how image quality affects deep neural networks," in *Eighth International Conference on Quality of Multimedia Experience*, June 2016, pp. 1–6.

[55] C. Yeo, P. Ahammad, K. Ramchandran, and S. S. Sastry, "High-speed action recognition and localization in compressed domain videos," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 8, pp. 1006–1015, Aug. 2008.

[56] B. Klare and M. Burge, "Assessment of H.264 video compression on automated face recognition performance in surveillance and mobile video scenarios," *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 7667, Apr. 2010.

[57] S.-S. Zhuang and S.-H. Lai, "Face detection directly from H.264 compressed video with convolutional neural network," in *International Conference on Image Processing*, Dec. 2009, pp. 2485 – 2488.

[58] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *International Conference on Neural Information Processing Systems - Volume 1*. Curran Associates Inc., 2012, pp. 1097–1105.

[59] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations*, May 2015.

[60] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *arXiv e-prints*, 2014.

[61] F. G. Zanjani, S. Zinger, B. Piepers, S. Mahmoudpour, P. Schelkens, and P. H. N. de With, "Impact of JPEG 2000 compression on deep convolutional neural networks for metastatic cancer detection in histopathological images," *Journal of Medical Imaging*, vol. 6, no. 2, pp. 1 – 9, 2019.

[62] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG 2000 still image compression standard," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 36–58, Sep. 2001.

[63] M. Tom, R. V. Babu, and R. G. Praveen, "Compressed domain human action recognition in H.264/AVC video streams," *Multimedia Tools Appl.*, vol. 74, no. 21, pp. 9323–9338, Nov. 2015.

[64] H. Wang and S.-F. Chang, "A highly efficient system for automatic face region detection in mpeg video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 4, pp. 615–628, Aug. 1997.

[65] H. Wang, H. S. Stone, and S.-F. Chang, "FaceTrack: tracking and summarizing faces from compressed video," in *Multimedia Storage and Archiving Systems IV*, vol. 3846, International Society for Optics and Photonics. SPIE, 1999, pp. 222 – 234.

[66] J. Hernandez-Ortega, J. Galbally, J. Fiérrez, and L. Beslay, "Biometric quality: Review and application to face recognition with faceqnet," *arXiv e-prints*, 2020.

[67] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication*, Urbana, IL, 1949.

[68] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948.

[69] G. G. L. Jr., "An introduction to arithmetic coding," *IBM J. Res. Dev.*, vol. 28, no. 2, pp. 135–149, 1984.

[70] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[71] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," in *Proceedings of the 33nd International Conference on Machine Learning*, ser. JMLR Workshop and Conference Proceedings, vol. 48. JMLR.org, Jun. 2016, pp. 1558–1566.

[72] X. Chen, B. Francia, M. Li, B. McKinnon, and A. Seker, "Shared information and program plagiarism detection," *IEEE Trans. Inf. Theory*, vol. 50, no. 7, pp. 1545–1551, 2004.

[73] W. Lee and D. Xiang, "Information-theoretic measures for anomaly detection," in *2001 Symposium on Security and Privacy, Oakland, California, USA May 14-16, 2001*, 2001, pp. 130–143.

[74] T. M. Mitchell, *Machine learning, International Edition*, ser. McGraw-Hill Series in Computer Science. McGraw-Hill, 1997.

[75] F. Escolano, P. Suau, and B. Bonev, *Information Theory in Computer Vision and Pattern Recognition*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[76] P. Savarese, S. S. Y. Kim, M. Maire, G. Shakhnarovich, and D. McAllester, "Information-theoretic segmentation by inpainting error maximization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2021, pp. 4029–4039.

[77] J. Stone, *Principles of Neural Information Theory: Computational Neuroscience and Metabolic Efficiency*, 06 2018.

[78] A. Borst and F. E. Theunissen, "Information theory and neural coding," *Nature Neuroscience*, vol. 2, pp. 947–957, 1999.

[79] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, vol. 1, 2005, pp. 886–893 vol. 1.

[80] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Journal of the American Podiatry Association*, vol. 60, Jan. 2009, p. 6.

[81] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu, "Automated curriculum learning for neural networks," *arXiv e-prints*, 2017.

[82] G. Hacohen and D. Weinshall, "On the power of curriculum learning in training deep networks," in *International Conference on Machine Learning*, 2019.

[83] D. Weinshall, G. Cohen, and D. Amir, "Curriculum learning by transfer learning: Theory and experiments with deep networks," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 80, July 2018, pp. 5235–5243.

[84] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016, pp. 761–769.

[85] M. Kumar, B. Packer, and D. Koller, "Self-paced learning for latent variable models," in *Advances in Neural Information Processing Systems*, vol. 23. Curran Associates, Inc., Dec. 2010, pp. 1189–1197.

[86] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[87] L. Jiang, D. Meng, Q. Zhao, S. Shan, and A. G. Hauptmann, "Self-paced curriculum learning," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, ser. AAAI'15, 2015, p. 26942700.

[88] T. Matiisen, A. Oliver, T. Cohen, and J. Schulman, "Teacherstudent curriculum learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3732–3740, 2020.

[89] T. Wang, J. Zhu, A. Torralba, and A. A. Efros, "Dataset distillation," *arXiv e-prints*, 2018.

[90] T. Nguyen, R. Novak, L. Xiao, and J. Lee, "Dataset distillation with infinitely wide convolutional networks," in *Advances in Neural Information Processing Systems 34*, December 2021, pp. 5186–5198.

[91] B. Zhao and H. Bilen, "Dataset condensation with differentiable siamese augmentation," in *Proceedings of the 38th International Conference on Machine Learning*, M. Meila and T. Zhang, Eds., vol. 139, 18–24 Jul 2021, pp. 12 674–12 685.

[92] H. Xiao, Z. Wang, Z. Zhu, J. Zhou, and J. Lu, "Shapley-nas: Discovering operation contribution for neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2022, pp. 11 892–11 901.

[93] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[94] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, Dec. 2017, pp. 5998–6008.

[95] P. Ren, Y. Xiao, X. Chang, P. Huang, Z. Li, X. Chen, and X. Wang, "A comprehensive survey of neural architecture search: Challenges and solutions," *ACM Comput. Surv.*, vol. 54, no. 4, pp. 76:1–76:34, 2022.

[96] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, pp. 55:1–55:21, 2019.

[97] M. Wistuba, A. Rawat, and T. Pedapati, "A Survey on Neural Architecture Search," *arXiv e-prints*, May 2019.

[98] G. Bender, H. Liu, B. Chen, G. Chu, S. Cheng, P. Kindermans, and Q. V. Le, "Can weight sharing outperform random architecture search? an investigation with tunas," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2020, pp. 14 311–14 320.

[99] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *5th International Conference on Learning Representations*, Apr. 2017.

[100] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *5th International Conference on Learning Representations*, April 2017.

[101] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, February 2018, pp. 2787–2794.

[102] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2019, pp. 2820–2828.

[103] Y. Gao, H. Yang, P. Zhang, C. Zhou, and Y. Hu, "Graph neural architecture search," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, July 2020, pp. 1403–1409.

[104] P. J. Angeline, G. M. Saunders, and J. B. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE Trans. Neural Networks*, vol. 5, no. 1, pp. 54–65, Nov 1994.

[105] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," in *6th International Conference on Learning Representations*, May 2018.

[106] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning*. Omnipress, 2010, p. 807814.

[107] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution," in *7th International Conference on Learning Representations*. OpenReview.net, May 2019.

[108] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive DARTS: bridging the optimization gap for NAS in the wild," *Int. J. Comput. Vis.*, vol. 129, no. 3, pp. 638–655, 2021.

[109] X. Zheng, R. Ji, L. Tang, B. Zhang, J. Liu, and Q. Tian, "Multinomial distribution learning for effective neural architecture search," *IEEE/CVF International Conference on Computer Vision*, pp. 1304–1313, 2019.

[110] J. Mellor, J. Turner, A. J. Storkey, and E. J. Crowley, "Neural architecture search without training," in *Proceedings of the 38th International Conference on Machine Learning,*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139, July 2021, pp. 7588–7598.

[111] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 80, Jul. 2018, pp. 4095–4104.

[112] S. Xie, H. Zheng, C. Liu, and L. Lin, "SNAS: stochastic neural architecture search," in *7th International Conference on Learning Representations*, May 2019.

[113] Y. Guo, Y. Chen, Y. Zheng, P. Zhao, J. Chen, J. Huang, and M. Tan, "Breaking the curse of space explosion: Towards efficient nas with curriculum search," in *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[114] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "NAS-bench-101: Towards reproducible neural architecture search," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. Long Beach, California, USA: PMLR, Jun. 2019, pp. 7105–7114.

[115] X. Dong, L. Liu, K. Musial, and B. Gabrys, "NATS-Bench: Benchmarking nas algorithms for architecture topology and size," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021, doi:10.1109/TPAMI.2021.3054824.

[116] X. Dong and Y. Yang, "NAS-Bench-201: Extending the scope of reproducible neural architecture search," in *International Conference on Learning Representations*, 2020.

[117] M. Lindauer and F. Hutter, "Best practices for scientific research on neural architecture search," *arXiv e-prints*, 2019.

[118] I. Radosavovic, R. Kosaraju, R. Girshick, K. He, and P. Dollar, "Designing network design spaces," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2020, pp. 10 425–10 433.

[119] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search," in *Automated Machine Learning - Methods, Systems, Challenges*, ser. The Springer Series on Challenges in Machine Learning. Springer, 2019, pp. 63–77.

[120] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, "Understanding and simplifying one-shot architecture search," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, Jul. 2018, pp. 550–559.

[121] R. Pasunuru and M. Bansal, "Continual and multi-task architecture search," in *Proceedings of the 57th Conference of the Association for Computational Linguistics*, July 2019, pp. 1911–1922.

[122] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware," *arXiv e-prints*, Dec. 2018.

[123] Q. Yao, J. Xu, W.-W. Tu, and Z. Zhu, "Efficient neural architecture search via proximal iterations," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, pp. 6664–6671, Apr. 2020.

[124] N. Parikh and S. P. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 127–239, 2014.

[125] J. Yu, P. Jin, H. Liu, G. Bender, P. Kindermans, M. Tan, T. S. Huang, X. Song, R. Pang, and Q. Le, "Bignas: Scaling up neural architecture search with big single-stage models," in *Proceedings of the European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 12352. Springer, Aug. 2020, pp. 702–717.

[126] D. Wang, M. Li, C. Gong, and V. Chandra, "Attentivenas: Improving neural architecture search via attentive sampling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2021, pp. 6418–6427.

[127] M. L. Artur Jordao, Fernando Yamada and W. R. Schwartz, "Stage-wise neural architecture search," in *International Conference on Pattern Recognition*, 2020.

[128] X. Wan, B. Ru, P. M. Esperança, and F. M. Carlucci, "Approximate neural architecture search via operation distribution learning," in *Winter Conference on Applications of Computer Vision*, January 2022, pp. 3545–3554.

[129] B. Chen, P. Li, C. Li, B. Li, L. Bai, C. Lin, M. Sun, J. Yan, and W. Ouyang, "Glit: Neural architecture search for global and local image transformer," in *International Conference on Computer Vision*, October 2021, pp. 12–21.

[130] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, "Single path one-shot neural architecture search with uniform sampling," in *Proceedings of the European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 12361. Springer, Aug. 2020, pp. 544–560.

[131] M. Chen, J. Fu, and H. Ling, "One-shot neural ensemble architecture search by diversity-guided search space shrinking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2021, pp. 16 530–16 539.

[132] X. Xia, X. Xiao, X. Wang, and M. Zheng, "Progressive automatic design of search space for one-shot neural architecture search," in *Winter Conference on Applications of Computer Vision*. IEEE, January 2022, pp. 3525–3534.

[133] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002.

[134] H. Shi, R. Pi, H. Xu, Z. Li, J. T. Kwok, and T. Zhang, "Bridging the gap between sample-based and one-shot neural architecture search with BONAS," in *Advances in Neural Information Processing Systems*, Dec. 2020.

[135] K. Ahmed and L. Torresani, "Maskconnect: Connectivity learning by gradient descent," in *Proceedings of the European Conference on Computer Vision*, vol. 11209. Springer, Sep. 2018, pp. 362–378.

[136] Y. Yang, H. Li, S. You, F. Wang, C. Qian, and Z. Lin, "Ista-nas: Efficient and consistent neural architecture search by sparse coding," *Advances in Neural Information Processing Systems*, vol. 33, Dec. 2020.

[137] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Advances in Neural Information Processing Systems*, vol. 28. Curran Associates, Inc., 2015, pp. 2377–2385.

[138] Y. Xu, L. Xie, X. Zhang, X. Chen, G. Qi, Q. Tian, and H. Xiong, "PC-DARTS: partial channel connections for memory-efficient architecture search," in *8th International Conference on Learning Representations*, April 2020.

[139] Y. Jiang, C. Hu, T. Xiao, C. Zhang, and J. Zhu, "Improved differentiable architecture search for language modeling and named entity recognition," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, Nov. 2019, pp. 3585–3590.

[140] S. You, T. Huang, M. Yang, F. Wang, C. Qian, and C. Zhang, "Greedynas: Towards fast one-shot NAS with greedy supernet," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2020, pp. 1996–2005.

[141] M. Cho, M. Soltani, and C. Hegde, "One-Shot Neural Architecture Search via Compressive Sensing," *arXiv e-prints*, Jun. 2019.

[142] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani, "A sparse-group lasso," *Journal of Computational and Graphical Statistics*, vol. 22, no. 2, pp. 231–245, 2013.

[143] A. A. M. Javier Garcia Lpez and F. M. Noguer, "E-dnas: Differentiable neural architecture search for embedded systems," in *International Conference on Pattern Recognition*, 2020.

[144] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[145] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on Pure and Applied Mathematics*, vol. 57, no. 11, pp. 1413–1457, 2004.

[146] K. Yu, C. Sciuto, M. Jaggi, C. Musat, and M. Salzmann, "Evaluating the search phase of neural architecture search," in *8th International Conference on Learning Representations*, April 2020.

[147] X. Chu, B. Zhang, and R. Xu, "Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search," in *International Conference on Computer Vision*, october 2021, pp. 12 219–12 228.

[148] Y. Gu, L. Wang, Y. Liu, Y. Yang, Y. Wu, S. Lu, and M. Cheng, "DOTS: decoupling operation and topology in differentiable architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2021, pp. 12 311–12 320.

[149] J. Peng, J. Zhang, C. Li, G. Wang, X. Liang, and L. Lin, "Pi-nas: Improving neural architecture search by reducing supernet training consistency shift," in *International Conference on Computer Vision*, October 2021, pp. 12 334–12 344.

[150] Y. Yang, S. You, H. Li, F. Wang, C. Qian, and Z. Lin, "Towards improving the consistency, efficiency, and flexibility of differentiable neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2021, pp. 6667–6676.

[151] K. Yu, R. Ranftl, and M. Salzmann, "Landmark regularization: Ranking guided super-net training in neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2021, pp. 13 723–13 732.

[152] L. S. Shapley and A. E. Roth, *The Shapley value : essays in honor of Lloyd S. Shapley / edited by Alvin E. Roth*. Cambridge University Press Cambridge [Cambridgeshire] ; New York, 1988.

[153] L. S. Shapley, *17. A Value for n-Person Games*. Princeton University Press, 2016, pp. 307–318.

[154] X. Dong and Y. Yang, "Network pruning via transformable architecture search," in *Advances in Neural Information Processing Systems*, Dec. 2019, pp. 759–770.

[155] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *The Conference on Neural Information Processing Systems Workshop*, 2014.

[156] C. Li, J. Peng, L. Yuan, G. Wang, X. Liang, L. Lin, and X. Chang, "Block-wisely supervised neural architecture search with knowledge distillation," pp. 1986–1995, 2020.

[157] J. Pan, C. Sun, Y. Zhou, Y. Zhang, and C. Li, "Distribution consistent neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2022, pp. 10 884–10 893.

[158] C. Li, T. Tang, G. Wang, J. Peng, B. Wang, X. Liang, and X. Chang, "Bossnas: Exploring hybrid cnn-transformers with block-wisely self-supervised neural architecture search," in *International Conference on Computer Vision*. IEEE, 2021, pp. 12 261–12 271.

[159] S. Saxena and J. Verbeek, "Convolutional neural fabrics," in *Advances in Neural Information Processing Systems*, Dec. 2016, pp. 4053–4061.

[160] X. Dong and Y. Yang, "One-shot neural architecture search via self-evaluated template network," in *International Conference on Computer Vision*, Oct. 2019, pp. 3680–3689.

[161] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "SMASH: one-shot model architecture search through hypernetworks," in *6th International Conference on Learning Representations*, May 2018.

[162] A. Zela, T. Elsken, T. Saikia, Y. Marrakchi, T. Brox, and F. Hutter, "Understanding and robustifying differentiable architecture search," in *8th International Conference on Learning Representations*. OpenReview.net, april 2020.

[163] H. Liang, S. Zhang, J. Sun, X. He, W. Huang, K. Zhuang, and Z. Li, "DARTS+: Improved Differentiable Architecture Search with Early Stopping," *arXiv e-prints*, Sep. 2019.

[164] X. Dong and Y. Yang, "Searching for a robust neural architecture in four gpu hours," pp. 1761–1770, 2019.

[165] M. Zhang, H. Li, S. Pan, X. Chang, and S. W. Su, "Overcoming multi-model forgetting in one-shot NAS with diversity maximization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2020, pp. 7806–7815.

[166] S. Yan, Y. Zheng, W. Ao, X. Zeng, and M. Zhang, "Does unsupervised architecture representation learning help neural architecture search?" *Advances in Neural Information Processing Systems*, vol. 33, Dec. 2020.

[167] T. N. Kipf and M. Welling, "Variational Graph Auto-Encoders," *arXiv e-prints*, Nov. 2016.

[168] T. Yang, Y. Liao, and V. Sze, "Netadaptv2: Efficient neural architecture search with fast super-network training and architecture optimization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2021, pp. 2402–2411.

[169] D. Bruggemann, M. Kanakis, S. Georgoulis, and L. Van Gool, "Automated search for resource-efficient branched multi-task networks," in *BMVC*, 2020.

[170] D. Ha, A. M. Dai, and Q. V. Le, "Hypernetworks," in *5th International Conference on Learning Representations*, Apr. 2017.

[171] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, june 2019, pp. 10 734–10 742.

[172] C. Simon, P. Koniusz, L. Petersson, Y. Han, and M. Harandi, "Towards a robust differentiable architecture search under label noise," in *Winter Conference on Applications of Computer Vision*. IEEE, January 2022, pp. 3584–3594.

[173] Z. Yu and C.-S. Bouganis, "Svd-nas: Coupling low-rank approximation and neural architecture search," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, January 2023, pp. 1503–1512.

[174] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, "Neural architecture optimization," in *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc., Dec. 2018, pp. 7816–7827.

[175] W. Wen, H. Liu, Y. Chen, H. H. Li, G. Bender, and P. Kindermans, "Neural predictor for neural architecture search," in *Proceedings of the European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 12374, August 2020, pp. 660–676.

[176] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, January 2019, pp. 4780–4789.

[177] B. Baker, O. Gupta, R. Raskar, and N. Naik, "Accelerating neural architecture search using performance prediction," in *6th International Conference on Learning Representations, Workshop Track Proceedings*. OpenReview.net, May 2018.

[178] L. Li, K. G. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *J. Mach. Learn. Res.*, vol. 18, pp. 185:1–185:52, 2017.

[179] R. Luo, X. Tan, R. Wang, T. Qin, E. Chen, and T.-Y. Liu, "Neural Architecture Search with GBDT," *arXiv e-prints*, Jul. 2020.

[180] Y. Xu, Y. Wang, K. Han, Y. Tang, S. Jui, C. Xu, and C. Xu, "Renas: Relativistic evaluation of neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2021, pp. 4411–4420.

[181] A. Gretton, O. Bousquet, A. J. Smola, and B. Schölkopf, "Measuring statistical dependence with hilbert-schmidt norms," in *Algorithmic Learning Theory, 16th International Conference*, ser. Lecture Notes in Computer Science, vol. 3734. Springer, October 2005, pp. 63–77.

[182] J. Mok, B. Na, J.-H. Kim, D. Han, and S. Yoon, "Demystifying the neural tangent kernel from a practical perspective: Can it be trusted for neural architecture search without training?" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2022, pp. 11 861–11 870.

[183] N. Cavagnero, L. Robbiano, B. Caputo, and G. Averta, "Freerea: Training-free evolution-based architecture search," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, January 2023, pp. 1493–1502.

[184] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *The Thirty-Third Conference on Artificial Intelligence*, January 2019, pp. 4780–4789.

[185] H. Tanaka, D. Kunin, D. L. K. Yamins, and S. Ganguli, "Pruning neural networks without any data by iteratively conserving synaptic flow," in *Advances in Neural Information Processing Systems 33*, december 2020.

[186] K. Kandasamy, W. Neiswanger, J. Schneider, B. Póczos, and E. P. Xing, "Neural architecture search with bayesian optimisation and optimal transport," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018*, December 2018, pp. 2020–2029.

[187] C. Villani, *Optimal Transport: Old and New*, ser. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 2008.

[188] H. Jin, Q. Song, and X. Hu, "Auto-keras: An efficient neural architecture search system," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, August 2019, pp. 1946–1956.

[189] H. Zhou, M. Yang, J. Wang, and W. Pan, "Bayesnas: A bayesian approach for neural architecture search," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 97. PMLR, June 2019, pp. 7603–7613.

[190] R. M. Neal, *Bayesian Learning for Neural Networks*. Berlin, Heidelberg: Springer-Verlag, 1996.

[191] C. White, W. Neiswanger, and Y. Savani, "BANANAS: bayesian optimization with neural architectures for neural architecture search," in *Thirty-Fifth AAAI Conference on Artificial Intelligence*, February 2021, pp. 10 293–10 301.

[192] Y. Chen, T. Yang, X. Zhang, G. MENG, X. Xiao, and J. Sun, "Detnas: Backbone search for object detection," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., Dec. 2019.

[193] T.-Y. Lin, P. Dollr, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017, pp. 936–944.

[194] J. Peng, M. Sun, Z.-X. ZHANG, T. Tan, and J. Yan, "Efficient neural architecture transformation search in channel-level for object detection," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., Dec. 2019.

[195] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "Nas-fpn: Learning scalable feature pyramid architecture for object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7029–7038.

[196] H. Xu, L. Yao, Z. Li, X. Liang, and W. Zhang, "Auto-fpn: Automatic network architecture adaptation for object detection beyond classification," *IEEE/CVF International Conference on Computer Vision*, pp. 6648–6657, 2019.

[197] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8759–8768.

[198] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu, and A. Berg, "Ssd: Single shot multibox detector," in *Proceedings of the European Conference on Computer Vision*, 2016.

[199] N. Wang, Y. Gao, H. Chen, P. Wang, Z. Tian, C. Shen, and Y. Zhang, "Nas-fcos: Fast neural architecture search for object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2020.

[200] Z. Tian, C. Shen, H. Chen, and T. He, "FCOS: Fully convolutional one-stage object detection," in *Proc. Int. Conf. Computer Vision*, 2019.

[201] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollr, "Focal loss for dense object detection," in *IEEE International Conference on Computer Vision*, 2017, pp. 2999–3007.

[202] Y. Wu and K. He, "Group normalization," in *Proceedings of the European Conference on Computer Vision*, Sep. 2018.

[203] T. Liang, Y. Wang, Z. Tang, G. Hu, and H. Ling, "OPANAS: one-shot path aggregation network architecture search for object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2021, pp. 10 195–10 203.

[204] L. Chen, M. D. Collins, Y. Zhu, G. Papandreou, B. Zoph, F. Schroff, H. Adam, and J. Shlens, "Searching for efficient multi-scale architectures for dense image prediction," in *Advances in Neural Information Processing Systems*, Dec. 2018, pp. 8713–8724.

[205] X. Zhang, H. Xu, H. Mo, J. Tan, C. Yang, L. Wang, and W. Ren, "DCNAS: densely connected neural architecture search for semantic image segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2021, pp. 13 956–13 967.

[206] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1915–1929, 2013.

[207] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *International Conference on Computer Vision*. IEEE Computer Society, Dec. 2015, pp. 2650–2658.

[208] P. H. O. Pinheiro and R. Collobert, "Recurrent convolutional neural networks for scene labeling," in *International Conference on Machine Learning*, ser. JMLR Workshop and Conference Proceedings, vol. 32. JMLR.org, Jun. 2014, pp. 82–90.

[209] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention*, Oct. 2015, pp. 234–241.

[210] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, 2017.

[211] G. Lin, A. Milan, C. Shen, and I. D. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jul. 2017, pp. 5168–5177.

[212] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, 2018.

[213] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian, "A real-time algorithm for signal analysis with the help of the wavelet transform," in *Wavelets*, J.-M. Combes, A. Grossmann, and P. Tchamitchian, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990, pp. 286–297.

[214] A. Giusti, D. C. Ciresan, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Fast image scanning with deep max-pooling convolutional neural networks," in *IEEE International Conference on Image Processing*, Sep. 2013, pp. 4034–4038.

[215] S. Saxena and J. Verbeek, "Convolutional neural fabrics," in *Advances in Neural Information Processing Systems*, D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, Eds., Dec. 2016, pp. 4053–4061.

[216] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, June 2019, pp. 6105–6114.

[217] R. S. Molday and O. L. Moritz, "Photoreceptors at a glance." *Journal of cell science*, vol. 128, pp. 4039–45, 2015.

[218] N. Ahmed, T. Natarajan, and K. Rao, "Discrete cosine transform," *IEEE Transactions on Computers*, vol. C-23, no. 1, pp. 90–93, 1974.

[219] ITU-T, "Series H: Audiovisual and multimedia systems infrastructure of audiovisual services - coding of moving video," 1994.

[220] ——, "Video coding for low bit rate communication," 1996.

[221] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "A survey on adversarial attacks and defences," *CAAI Trans. Intell. Technol.*, vol. 6, no. 1, pp. 25–45, 2021.

[222] C. Solomon and T. Breckon, *Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab*. Wiley-Blackwell, 2010.

[223] R. C. González and R. E. Woods, *Digital image processing, 4th Edition.* Pearson Education, 2018.

[224] S. Akcay, A. Atapour-Abarghouei, and T. P. Breckon, "Ganomaly: Semi-supervised anomaly detection via adversarial training," in *Asian Conference on Computer Vision.* Springer, 2018, pp. 622–637.

[225] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial Autoencoders," *arXiv e-prints*, Nov. 2015.

[226] A. Atapour-Abarghouei and T. Breckon, "Real-time monocular depth estimation using synthetic data with domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 1–8.

[227] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, Jan. 2015.

[228] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2016, pp. 3234–3243.

[229] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2016, pp. 3213–3223.

[230] R. Szeliski, *Computer Vision - Algorithms and Applications, Second Edition*, ser. Texts in Computer Science. Springer, 2022.

[231] "Object recognition datasets and challenges: A review," *Neurocomputing*, vol. 495, pp. 129–152, 2022.

[232] A. Khanum, C.-Y. Lee, and C.-S. Yang, "Involvement of Deep Learning for Vision Sensor-Based Autonomous Driving Control: A Review," *IEEE Sensors Journal*, vol. 23, no. 14, pp. 15 321–15 341, Jul. 2023.

[233] J. Janai, F. Guney, A. Behl, and A. Geiger, "Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art," *Foundations and Trends in Computer Graphics and Vision*, vol. 12, 04 2017.

[234] M. P. Shah, "Semantic segmentation architectures implemented in pytorch." https://github.com/meetshah1995/pytorch-semseg, 2017.

[235] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He, "Detectron," https://github.com/facebookresearch/detectron, 2018.

[236] H. Pan, "Pytorch realtime multi-person pose estimation," https://github.com/last-one/Pytorch_Realtime_Multi-Person_Pose_Estimation, 2018.

[237] J. Y. Huang, "two-stream-action-recognition," https://github.com/jeffreyyihuang/two-stream-action-recognition/, 2019.

[238] S. Desai and H. G. Ramaswamy, "Ablation-cam: Visual explanations for deep convolutional network via gradient-free localization," in *Winter Conference on Applications of Computer Vision, WACV 2020*, March 2020, pp. 972–980.

[239] H. Nyquist, "Certain topics in telegraph transmission theory," *Transactions of the American Institute of Electrical Engineers*, vol. 47, no. 2, pp. 617–644, 1928.

[240] C. Shannon, "Communication in the presence of noise," *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, 1949.

[241] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2019, pp. 4690–4699.

[242] M. Wang and W. Deng, "Mitigating bias in face recognition using skewness-aware reinforcement learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2020, pp. 9319–9328.

[243] M. Wang, W. Deng, J. Hu, X. Tao, and Y. Huang, "Racial faces in the wild: Reducing racial bias by information maximization adaptation network," in *International Conference on Computer Vision*, October 2019, pp. 692–702.

[244] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, "Deep image prior," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 9446–9454.

[245] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is "nearest neighbor" meaningful?" in *Database Theory*, 1999, pp. 217–235.

[246] A. Razavi, A. van den Oord, and O. Vinyals, "Generating diverse high-fidelity images with vq-vae-2," in *Advances in Neural Information Processing Systems*, Dec. 2019, pp. 14 866–14 876.

[247] S. Akçay, A. Atapour-Abarghouei, and T. P. Breckon, "Skip-ganomaly: Skip connected and adversarially trained encoder-decoder anomaly detection," in *International Joint Conference on Neural Networks*. IEEE, 2019, pp. 1–8.

[248] T. Tang, W. Kuo, J. Lan, C. Ding, H. Hsu, and H. Young, "Anomaly detection neural network with dual auto-encoders GAN and its industrial inspection applications," *Sensors*, vol. 20, no. 12, p. 3336, 2020.

[249] L. van der Maaten and G. Hinton, "Viualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, Nov. 2008.

[250] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Jun. 2015, pp. 815–823.

[251] R. Fisher, T. Breckon, K. Dawson-Howe, A. Fitzgibbon, C. Robertson, E. Trucco, and C. Williams, *Dictionary of Computer Vision and Image Processing, Second Edition*. Wiley, 2014.

[252] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 7482–7491.

[253] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," in *Advances in Neural Information Processing Systems*, Dec. 2017, pp. 6306–6315.

[254] J. W. Barker and T. P. Breckon, "PANDA: perceptually aware neural detection of anomalies," in *International Joint Conference on Neural Networks*. IEEE, Jul. 2021, pp. 1–8.

[255] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2022, pp. 10 684–10 695.

[256] Y. Xian, S. Sharma, B. Schiele, and Z. Akata, "F-VAEGAN-D2: A feature generating framework for any-shot learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2019, pp. 10 275–10 284.

[257] S. Gur, S. Benaim, and L. Wolf, "Hierarchical patch VAE-GAN: generating diverse videos from a single sample," in *Advances in Neural Information Processing Systems*, Dec. 2020.

[258] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[259] M. Kaya and H. S. Bilge, "Deep metric learning: A survey," *Symmetry*, vol. 11, no. 9, p. 1066, 2019.

[260] Y. Sun, C. Cheng, Y. Zhang, C. Zhang, L. Zheng, Z. Wang, and Y. Wei, "Circle loss: A unified perspective of pair similarity optimization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2020, pp. 6397–6406.

[261] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations*, Y. Bengio and Y. LeCun, Eds., May 2015.

[262] I. Loshchilov and F. Hutter, "SGDR: stochastic gradient descent with warm restarts," in *5th International Conference on Learning Representations*, April 2017.

[263] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[264] J. Cha, "pt.darts," https://github.com/khanrc/pt.darts, 2019.

[265] chenxin061, "pdarts," https://github.com/chenxin061/pdarts, 2020.

[266] X. Dong, "Autodl-projects," https://github.com/D-X-Y/AutoDL-Projects, 2022.

[267] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[268] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," *arXiv e-prints*, Aug. 2017.

[269] J. Fonseca and K. Taghva, *The State of Reproducible Research in Computer Science*, 01 2020, pp. 519–524.

[270] D. Towers, M. Forshaw, A. Atapour-Abarghouei, and A. S. McGough, "Long-term Reproducibility for Neural Architecture Search," *arXiv e-prints*, Jul. 2022.

[271] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *3rd International Conference on Learning Representations*, Y. Bengio and Y. LeCun, Eds., May 2015.

[272] D. Manandhar, M. Bastan, and K. Yap, "Tiered deep similarity search for fashion," in *European Conference on Computer Vision Workshops*, ser. Lecture Notes in Computer Science, L. Leal-Taixé and S. Roth, Eds., vol. 11131. Springer, September 2018, pp. 21–29.

[273] C. Wang, A. Bochkovskiy, and H. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,*, June 2023, pp. 7464–7475.

[274] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2014, pp. 1386–1393.

[275] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," in *International Conference on Computer Vision*. IEEE, October 2019, pp. 6568–6577.

[276] Y. Sun and M. Zhang, "Compositional metric learning for multi-label classification," *Frontiers Comput. Sci.*, vol. 15, no. 5, p. 155320, 2021.

[277] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical Text-Conditional Image Generation with CLIP Latents," *arXiv e-prints*, Apr. 2022.

[278] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *International Conference on Computer Vision*, October 2017, pp. 618–626.

[279] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," in *International Conference on Computer Vision*. IEEE Computer Society, October 2017, pp. 2813–2821.

[280] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[281] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Computer Vision - ECCV 2016 - 14th European Conference,*, ser. Lecture Notes in Computer Science, vol. 9906, 2016, pp. 694–711.

[282] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 586–595.