Understanding Domain Adaptation: Applications in Reinforcement Learning, and Extrapolation

Thomas Patrick Carr

Doctor of Philosophy, Engineering & Physical Sciences

Aston University

December 2020

Abstract

Deep Learning has made impressive progress in a number of data processing domains. A large part of this progress comes from building large and complex models which are costly to train and, when data is limited, prone to over-fitting. We explore solutions to this problem through the domain adaptation paradigm.

Domain adaptation assumes that although data may be visually dissimilar, drawn from differing distributions such as photographs versus paintings, they still contain the same content and so share a representation space. We propose a model for domain adaptation building on the recent concept of generative adversarial networks. We show how this model can be used for domain adaptation with applications to reinforcement learning. We further investigate the utility of the model for explicit extrapolation problems.

Having proposed a model, we develop a further understanding of the conditions that lead to better shared embedding spaces. With this understanding, we propose a penalty term for the adversarial domain adaptation problem which we demonstrate achieves state of the art performance on a number of benchmark domain adaptation datasets.

We also consider the problem of adaptation from the perspective of extrapolation along the sampling space. Domain adaptation research does not explicitly consider this case and we propose two new datasets to examine the concept.

Further to this, we take a broader view of the potential applications of various transfer learning techniques and apply one-shot learning to a recently proposed extrapolation task. Through our extrapolation experiments we demonstrate the need for new datasets and testing protocols in order to appropriately verify model generalisation performance, which may otherwise be difficult to judge.

Additional Keywords

(Domain Adaptation , Deep Learning , Transfer Learning , Generalisation)

Acknowledgements

This dissertation is the culmination of many years of hard work. It has been a wonderful journey with many new friends who enriched the experience. A particular shout out to Thomas Griffiths for all the coffee chats over the years and to Deepika Garg for the years of friendship. I am also grateful to the HungryGuys, let the meals together continue for years to come.

I would like to thank my supervisors Maria Chli and George Vogiatzis for their support and guidance through the entire process. Finally, I want to thank my parents for their continued love and support.

Don't stop here; this is just the beginning

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the quest for ever more capable and, hopefully, intelligent machines much has been accomplished. From rule-based systems and logical theorem provers to deep learning the field has undergone many shifts [1]. The advent of Deep Learning was made possible through improved hardware capabilities and with them the ability to process large volumes of data. Since then significant progress has been made in diverse fields.

Deep learning is a powerful optimisation paradigm that is applied to many different types of data e.g. Image, Video, Language [2] and time-series [3], graph [4] data. The successes of machine learning and deep learning models are embedded into many of the technologies we use everyday, from video and product recommendations on streaming and retail sites, to automatic content filtering on social media. The technology is being built into our systems and procedures and has the potential to improve even complex tasks, from the development of self driving cars [5] to medical diagnostics [6] and drug discovery [7].

The success of the various methods often relies on the availability of large labeled datasets which are difficult to obtain for many problems and for many reasons; from the cost of employing experts to appropriately label the data to meeting ethical guidelines for potentially sensitive datasets such as medical data. As datasets grow in size, capturing more complexities of the world and requiring answers to more complex questions, models also grow in size. The time and cost premium of collecting data and training models makes the single-use nature of machine learning and deep learning models questionable: are they worth the cost and is it possible to use a model for more than one problem?

In this thesis we will seek to address the second question: Is it possible to use a neural network model for more than one problem? Direct transfer and pre-training approaches to employing machine learning models certainly show that it is possible. There are, however, many reasons direct transfer might fail. One such reason is applying a model to data sampled from a distribution that is distinct from the training distribution. This issue is the core problem addressed in this thesis. In the remainder of this chapter we provide a larger description of the problem we are addressing and outline what the objectives and challenges are before summarising our contributions.

1.1 The Problem

Although machine learning models, especially deep learning models, are capable of learning solutions to complex problems, the data and time required to train those models is often restrictive; some models require weeks of training on large servers, for example AlphaGo [8] where they used 50 GPUs and 3 weeks training time. The problems associated with requiring a large number of data samples are not limited to gathering observations but in many cases also our ability to appropriately label those images. Each trained model comes with a large development and training cost and therefore if we can use those models in other contexts we could reduce that incurred cost.

Developing methods that generalise learning is part of the solution to re-usable models and future development. Directly applying the models on new data or to a new task is not guaranteed to, and frequently does not, generalise. Trained models are often re-used, as they demonstrate limited generalisation and larger models can be difficult to train tabula-rasa. Indeed it has become standard practice in many areas of machine learning to use a model that has been pre-trained on very large competitive datasets. Transferring learning from one task to another is possible. How best to achieve that is a question that needs to be addressed.

The problem so is this: Given a trained model, how can we apply it to data sampled from out-of-distribution. This is quite a broad problem description and many different approaches could be taken to solve various aspects of the incurred problem. We will therefore restrict the problem further in order to focus our research questions. How can a model be adapted to out of domain samples representing the same set of classes? This question poses a methodological question, however, it also raises questions about what it means for a model to generalise and how we can measure degrees of out-of-distribution performance.

1.2 Objectives

This dissertation is focused on developing an understanding of and solutions to the problem of out-of-distribution generalisation. Out-of-distribution generalisation is an important problem for the machine learning community. It involves methods for leveraging unlabeled data and using existing models on new tasks. In this work we

develop insights into the problem, developing methods, loss functions and training methodologies. This work is carried out with a focus on the domain adaptation problem.

Domain adaptation as a method for adapting models to out of distribution data, is part of the wider context of solving generalisation problems within machine learning. We also explore this broader context with work developing an understanding of how domain adaptation can be used in contexts where extrapolation is made explicit. We broaden this discussion on generalisation and demonstrate that one-shot learning methods can also be utilised in such a context.

1.3 Challenges

Neural Networks can be brittle when it comes to unexpected changes in their input space. This recently came to more public attention, as highlighted by [9], with the recent Covid-19 pandemic affecting people's purchasing habits and the news having a gloomy outlook which has led to AI systems not performing as well as expected and requiring human intervention to keep systems on track. Overcoming input space differences poses a challenge for all neural networks when it comes to out-of-distribution generalisation.

Models can fail to generalise for numerous reasons. Convolutional Neural Networks are highly constrained by their learned feature maps which may not be developed to match previously unseen data. Indeed, generalising to unseen out-of-distribution data has long been a challenge for machine learning systems. Another failure case [10] demonstrated in recent research shows that models sharing the same architecture, trained on the same data and scoring equally on the same test set which was sampled from the same distribution as the training data, can exhibit significantly varying performance on out-of-distribution data.

Adversarial Domain Adaptation methods focus on aligning the embedding space of the source and target data. This puts the focus on aligning neural network filters to accommodate visual changes in the data. It is then usually assumed the learned classifier can be used directly with these newly aligned features.

Measuring adaptation is also a complex question. Models can be compared on specific transfer tasks and relative performance across tasks can indicate which tasks are easier or more difficult to adapt to. It is, however, unclear how to appropriately

measure distance between datasets in the context of adaptation. How far out of distribution can models be adapted and how does that effect their generalisation capabilities.

We summarise the challenges in the following list:

Understanding the shared embedding space and how to construct it.

Understanding the benefits and limits of Domain Adaptation.

Understanding extrapolation as part of the generalization problem and the role of domain adaptation.

1.4 Contributions

This thesis makes contributions to each of the challenges identified in section 1.3.

Constructing an Embedding One of the core challenges of domain adaptation is developing methods that appropriately align source and target domain embeddings. To address this challenge we present a new architecture in chapter 3 utilising modern machine learning techniques combined with traditional regularisation approaches to domain adaptation. We then investigate characteristics of the embedding itself. In chapter 4 we examine the impact of embedding dimensionality and in chapters 5 and 6 we introduce and explore the utility of a newly proposed penalty[11] and propose a number of new variants providing insight into the impact of entropy on the success of adaptation.

Benefits and Limits In our examination of applying domain adaptation to reinforcement learning in chapter 3 we demonstrate that domain adaptation can be used to learn a better initial representation. In chapter 7 we discuss the issue of extrapolation with a framing of the problem as out-of-distribution generalisation and the limits of current work in examining well defined distribution differences. In chapter 8 we address these issues introducing two datasets and demonstrating the effect of adapting from various sub-domains of the distribution space.

Extrapolation In chapter 7 we introduce the problem of extrapolation and provide a framing of extrapolation as out-of distribution generalization in chapter 8. This

work demonstrates that the source domain and target domain can have significant impact on performance and that both model interpolation and extrapolation can be improved through domain adaptation. Further to this we introduce and examine one-shot learning as a method that has an affinity for extrapolation in chapter 9. This chapter also corroborates the results in chapter 8, demonstrating that the source task used for training can have significant impact on the degree of extrapolation achieved by the model

The work presented in this dissertation covers a broad range of issues. We provide insight into the mechanics of domain adaptation and use that insight to develop methods to improve adaptation. The issues of generalization and out-of-distribution data are also discussed. These are critical when evaluating whether a domain adaptation or transfer learning method will provide benefit. We summarise our contributions in the following list:

1. We propose a new Domain Adaptation Architecture
2. We propose and justify a new penalty term for adversarial domain adaptation methods
3. We demonstrate the applicability of Domain Adaptation to Extrapolation
4. We demonstrate the utility of one-shot learning for extrapolating analogical reasoning

Chapter 2

Background

In this chapter we will introduce the subjects that are important for understanding the rest of this paper. These are broken up into 3 major sections. We first introduce Machine Learning with an emphasis on Deep learning as well as introducing specific architectures, we then introduce Reinforcement Learning and finally we introduce the Transfer Learning and the specific problem of Domain Adaptation.

2.1 Machine Learning and Transfer Learning

Machine Learning is a diverse research field with many methods for solving many types of problems. Transfer learning methods are additions to the machine learning paradigm which extend the applicability of models to new problems. Combining various machine learning and transfer learning approaches provides significant advantages in many cases, for instance when only unlabeled data is available for the desired task it may be possible to use a related but distinct labeled dataset to improve a models performance. Many large models are also time consuming to train tabula-rasa and so pre-trained versions are often used as initialisations for new problems.

Machine learning models fall into one of two categories, discriminative or generative. Discriminative models learn the conditional distribution $p(y|x)$ for input data x and target data y. Generative models by contrast learn the joint-distribution $p(x, y)$ and as such are capable of generating new data-samples. Both types of model play an important role in the training process of many transfer learning algorithms.

2.1.1 Deep Learning an Important Architectures

Deep Learning is a well established machine learning field. Convolutional neural networks have been a core part of the advancement of deep learning applications to problems such as image recognition [12]. However, even in cases where image data might not seem like a natural input to a model, CNN's have demonstrated their utility; these areas include audio processing models such as WaveNet [13] and genetic data as in [14]. These demonstrations add to our confidence in pursuing the application of CNN's to new problems and as a likely component in any machine learning system.

The deep learning framework and learning algorithm provides a lot of flexibility in how layers and networks of layers are connected and updated to solve various tasks.

In the deep learning community many network designs have risen to prominence. Low-level architecture specifications provide common starting points for applications on specific types of data and problem. Higher level designs, specifying interactions between network blocks, have also come to prominence. In the following we describe some of these higher-level architectures which are used throughout the community and which are important in understanding our model in the chapter 3.

Generative Adversarial Networks

An important deep learning architecture is the Generative Adversarial Network (GAN). As the name implies this is used as a generative modelling tool and the architecture uses 2 networks trained in competition with each other, hence adversarial. This model style has garnered immense popularity with a great number of papers proposing variants and applications. This has been driven both by the perceived performance strength of these models and by the well known difficulty of training them to live up to that performance.

For instance consider the task of generating new faces, as done here `https://thispersondoesnotexist.com/` , as an example of the task as performed by a GAN based network. Given a dataset of face pictures we would like to learn a generative model of the space of faces. To that end we need a face generator, which needs to be conditioned on some input distribution; for our purposes we use a standard Gaussian distribution. So we define a neural network to generate images given a sample from our Gaussian prior. Now we can generate something from some noise, but we have not yet trained it. What we have are samples of real faces and so we can ask does the generated image look like it came from the true distribution of faces represented by our dataset. To answer this question a second network, which we call the Discriminator, can be trained on samples from the real dataset and the generative model with the goal of classifying them as real or generated. Given a model that can separate our generated samples from the real samples we can derive a goal for the generator, which is that it should generate images that will be misclassified by the discriminator. As we train these models in opposition to each-other we gradually find a model which can generate faces that appear to be real. We now formalise this description of a GAN.

A GAN consists of 2 networks, a generator G and a discriminator D. The generators' task is to generate samples that are similar to samples from the given target

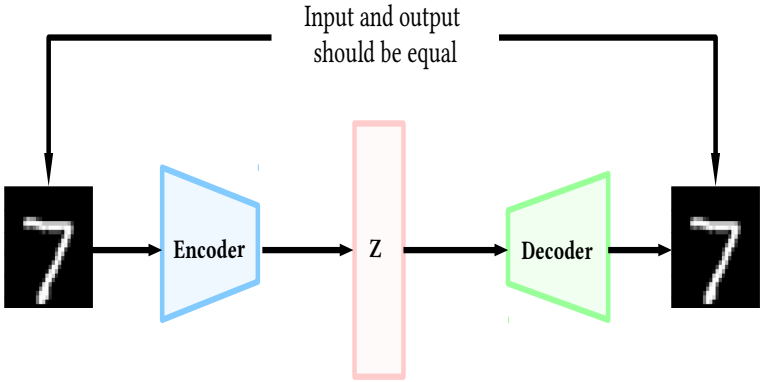


Figure 2.1: An AutoEncoder

dataset. The discriminators' task is to classify which samples are forgeries created by the generator and which are samples from the true dataset. The generator represents a model $P_\theta(x|z)$ where $z \sim N(0, \mathbb{I})$, which we define as a neural network parameterized by weights θ that takes z as input giving a generator model $G_{\theta_g}(z)$. The discriminator then takes data samples as input and tries to discriminate between them giving $D_{\theta_d}(x)$. Both networks can then be optimised given the minimax objective:

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[log(D(x))] + E_{z\sim_{p_z(z)}}[log(1 - D(G(z)))] \tag{2.1}$$

This formulation was proposed in [15] and since then many papers have worked on developing improvements and solving issues with the GAN framework. It is also worth noting links to earlier works such as [16], which are highlighted in this more recent paper [17]. However the problem is described and formulated it is true to say that GANs have been hugely popular and influential and have been used to great effect for generative modelling.

AutoEncoders

The AutoEncoder is a tool for unsupervised learning. The goal of the AE model is to reconstruct its input in the output layer. Ostensibly this could be done trivially and uselessly with an identity mapping and so AEs are usually trained to reconstruct the mapping through a bottleneck layer. The core aim of an AE model is to learn a compressed representation of the input data. Making this representation useful and informative for other tasks is a goal of the encoder part of the network and the

decoder part of the network can be seen as a learned generative model of the data.

The standard Autoencoder is defined in two parts, a generator network $G_{\theta_g}(x)$ and a decoder network $D_{\theta_d}(z)$ where z is the output of network G

This simple construction has myriad applications. In many scenarios AEs can be used as a pre-training step to better initialise the weights of the network for the desired task, rather than using the standard random initialisation techniques; an early type of direct transfer. AEs are also used for tasks such as image de-noising and can be used as generative models for dataset augmentation.

The standard autoencoder however does not necessarily learn a smooth continuous embedding space and the generative aspect of the model can therefore be less than useful. To fix this issue we can impose constraints on the learned representation, for instance we could train the network to represent data as if it was sampled from a Gaussian Distribution. This is the approach used in the Variational autoencoder and the adversarial autoencoder, which we now describe.

Variational Autoencoder.

The Variational autoencoder recasts our encoder as learning the data distribution, predicting the mean and variance of the output distribution and the mean and variance of the encoder distribution. We can then sample from the Gaussian defined by these parameters. However, this introduces a problem, since we cannot differentiate through stochastic processes we are not be able to train our autoencoder models like we normally would using backpropagation.

The solution is to use the 'reparameterization trick' where the stochastic element is pushed into an extra variable sampled from the unit Gaussian and taking advantage of the fact that any Gaussian distribution can be achieved by multiplying by the standard deviation and adding the mean of the desired distribution to the unit Gaussian.

Adversarial Autoencoder.

As an alternative to the VAE we can combine the idea of an autoencoder with that of a GAN in order to formulate the Adversarial AutoEncoder [18]. This architecture has the same goal as the VAE, we want to impose a prior on the distribution of the z encoding vector of the autoencoder. In the AAE setup this is done by presenting

z samples generated by the encoder and z samples, sampled directly from the prior, to a discriminator network.

In this formulation we have 3 models, usually instantiated as neural networks, an encoder $E_{\theta_e}(x)$ which encodes an image from the dataset, a decoder/generator network $G_{\theta_g}(E(x))$ which reconstructs the image x from the encoding of x produced by E and a discriminator $D_{\theta_d}(z)$

The networks are then trained jointly to optimise the GAN loss 2.1 and AE loss. this can be done jointly or as sequential optimisation steps.

2.2 Reinforcement Learning

The reinforcement learning paradigm describes the methods by which agents learn through interaction with their environments to achieve a goal. Such an agent may be a disembodied algorithmic trader or a more concrete physical robot. In both cases training the agent to utilise experience from previous tasks could provide numerous benefits. Here we briefly introduce reinforcement learning and in chapter 3 we evaluate the application of our newly proposed domain adaptation architecture.

The Reinforcement Learning problem is to solve a sequential interaction task driven by a reward signal to indicate success or failure. The problem is usually formalised as a Markov Decision Process (MDP). The MDP, which represents the interaction problem to be solved, is defined as a tuple.

$$< S, A, T, R, \gamma > \tag{2.2}$$

In equation 2.2, S is the set of states, A is the set of actions the agent can execute, T is the transition function, a distribution over the state space given a state and an action; R is the reward function, which returns a real value for every state transition and γ is the discount factor.

Solving the MDP amounts to optimising the strategy an agent uses to interact with the world, selecting actions given states, in order to maximize the expected discounted sum of future rewards. For a full exposition refer to [19].

2.2.1 Deep Reinforcement Learning

Reinforcement learning has benefited from the introduction of deep learning methodologies. Deep convolutional networks have made the application of reinforcement learning to many complex visual domains significantly easier.

They allow the model designers to take a step back from state design making it much easier to handle complex visual domains and allowing learning to be end to end. However, the introduction of function approximation methods also introduces new complexities.

The problems we initially addressed in a few papers [20, 21] focused on the Q-learning approach through the introduction of a few key algorithmic designs. The first is the introduction of replay memory, first explored in [22]. The replay memory acts as a store of previous experience which is sampled from every time we perform an update of the Q function. The second fix is to introduce a target Q-function which is used as a prediction target and updated less frequently as a copy of the current approximate Q-function.

Many methods have since been introduced to further improve various aspects of the algorithm. Further policy based approaches have also been extended with deep learning based methods. However, Deep Learning based methods in particular suffer from slow training times due to a number of issues inherent to the problem specification. Datasets are gathered during training and so interaction time with the environment slows down the learning processes. A lack of sample diversity can also limit the speed of learning. This makes research into methods such as domain adaptation for reinforcement learning an important avenue to improving the general utility of trained models.

The tabula-rasa nature of training Deep Reinforcement learning algorithms makes the additional problem of training a deep convolutional network major restriction. Reinforcement learning can be sped up through transfer learning techniques to achieve a variety of transfer goals. Transfer learning can improve overall solution speed through time to threshold. Alternatively transfer learning may be used to improve asymptotic performance or to provide an initial performance boost, called a jumpstart [23].

2.3 Transfer Learning

Having introduced the important concepts for deep learning along with various important architectural setups and the reinforcement learning paradigm, we now turn our attention to the problem of knowledge transfer. In this section we introduce and taxonomise the various forms transfer learning can take before taking a deeper dive into domain adaptation which is the focus of the methods and investigations of this dissertation.

The deep learning approach to function approximation offers many benefits however it also suffers from a number of practical drawbacks. We have learned that deeper models [24, 25, 26], with specific extensions can be very powerful and perform very well on a number of complex tasks. However, because the models are so large with many millions of parameters, training them requires a lot data. They also require large volumes of computing resources and time. These drawbacks can make training a network from random weights a highly restrictive task; this is especially true in cases where data is limited or computing resources are limited.

Further, trained models are often limited in their use even on very similar seeming tasks. Indeed there is a strong case that models are limited to data that comes from the same data distribution as was used during training [27]. Models can generalise by predicting unseen data points sampled from within the training data distribution but if the data is sampled from a different distribution success is far from guaranteed. Such considerations motivate transfer learning and domain adaptation as they can be used to overcome some of these distribution changes.

The training time and data requirements are partially overcome by using existing models and it is now ubiquitous to use pre-trained models as a starting point for solving a wide variety of tasks; with the major frameworks providing a variety of pre-trained models for easy download and use. Such an approach is known as direct weight/parameter transfer and many papers have contributed to the development of the idea [28, 29]. Auto-encoders have been used to prime neural network parameters in cases where data is scarce, and layer-wise pre-training has been used to build up model representations iteratively. Some of the biggest advancements have come in tandem with the Imagenet challenge [30]. Large models developed to solve Imagenet are trained on a wide variety of images and so learn very general features which are useful for many downstream tasks; this includes models such as Alexnet[12] ,

and Resnet[24] as well as others. Such transfer is also evident in problems such as natural language processing(NLP) and natural language understanding(NLU) where a number of large pre-trained language models [31, 32] are used as the basis for solving downstream tasks. The ubiquity of this approach is such that the term transfer learning is often used to mean direct weight transfer of the sort described, often including a fine-tuning step. In this thesis we use the term transfer learning as a more general overarching term and use the term direct transfer for the type of transfer learning thus far described.

Going beyond direct weight transfer there are many problems, such as differences in source and target sampling domains, the focus of this thesis, for example photographs v.s. paintings are not solved by such an approach. In terms of content when identifying content in photographs or paintings it seems clear that a shared representation space could exist. However, visual differences often lead to poor shared embeddings. This is one of the core problems addressed throughout this thesis.

2.3.1 A Taxonomy of Transfer Learning Paradigms

Transfer Learning serves as an attempt to improve the prospects of neural network training in the face of these limitations. In the following we explain the various sub-fields of transfer and the problems they seek to overcome. There is potential for overlap between methods and indeed there can be room for integrating various aspects of each subdivision.

The wide variety of transfer learning paradigms demonstrate how the framing influences the questions we have outlined in chapter 1. The importance of the learned input representation and what and how it is learned varies across techniques. The degree to which a model will remain use-able in the source task also varies, indicating not all models become more general with transfer. Such considerations are useful when considering transfer learning techniques to separate the similarities from the dissimilarities.

We can consider two main sets of consideration when designing models for transfer learning. We can consider the problem they are being used to solve, is there labeled data for the target domain, is the task the same or different. This problem splits models into the various paradigms, such as multi-task learning and zero-shot learning. The other question is what method is used for sharing or transferring

knowledge for example we could share model parameters or priors, or alternatively instance samples can be shared by mapping them to the task.

Sequential Learning

Sequential transfer is likely the most common form of transfer learning. In this paradigm a model is trained to solve one task and then trained to solve a second task and then potentially more after that. The most straight forward version of this is taking a pre-trained model and applying it to the new dataset. This is often the approach taken when applying a model to a specific problem case where limited training data is available so using other data sources can improve learning outcomes on the target task.

Multi-Task learning

Multi-task learning [33, 34] is a common paradigm for transfer in reinforcement learning. In this paradigm we have a known set of tasks and train a model to solve all of these tasks at the same time, this might mean in parallel or sequentially but with the sequence possibly repeating tasks previously solved. One of the major problems with this approach is the problem of catastrophic forgetting, where the learner "forgets" what it has already learned in the process of learning to solve the new task; this is less of an issue with sequential transfer as it is not intended that the model may have to be used on the source task again.

Few-Shot learning

Few-shot learning [35] describes a family of techniques for achieving transfer with few, sometimes only one, labeled samples. In these approaches a representation is learned such that it captures the factors of variance between classes and comparison with a known sample allows new data to be appropriately classified. MAML [36], model agnostic meta-learning, is a powerful variant of this type of approach.

Concept Drift

Concept drift occurs as the intended classification changes [37]. This can occur with data that has a temporal aspect, for instance movie sentiment reviews might change over time as subsequent audiences opinions change. Indeed the relevance of this problem can be seen with the failure of the market to respond quickly enough

to the changes in demand for products such as toilet paper during the COVID-19 pandemic panic [38].

Domain Adaptation

Domain adaptation usually focuses on the cases where the training input and target input distributions are different but the target task remains the same [39]. There is a presumed mapping from a set of concepts or features to the desired function output and the aim of models in this context is: given that concept/feature space learn or adapt a mapping from the target domain to that space as represented by the source domain model.

Zero-Shot Learning

Zero-shot learning refers to techniques used to adapt models to classify entities whose instances may not have been included in the training data. It is in a sense an extreme case of domain adaptation where certain classes were never seen at training time, thus their instances are out of domain samples. This problem is tackled with a number of competing approaches for which a short summary can be found in [40, 41]. A key aspect of the solution is to map categories to vectors, using a word embedding model such as word2vec [42, 43] which allows for performing classification of new samples in label-space using approaches such as nearest-neighbours.

2.3.2 Deep Transfer Learning

Directly utilising the weights from an existing model appears to be useful but we have not yet addressed the question of when to apply it, is it always useful and can it be employed in all situations? There are a number of variables to consider when designing an approach to transfer learning. Early work transferring neural networks took a very direct approach, asking the question: What happens if we directly copy the weights of our network to solve a new task by cutting the network at various layers [29]?

This question is natural as we are aiming to train general functions which should work on unseen data. For instance consider the problem of digit recognition. If we have trained a model to classify images of handwritten digits, our model has presumably learned what each digit looks like. If we then show the model images of

digits on houses we might expect the model to classify those numbers as well. This is the essence of direct transfer. What we find however is that our models are very brittle and subtle changes in the input can have drastic effects on the classifier.

Directly using the model on a new dataset may be intuitive but may not be the best approach. To take a more fine-grained view of a neural network model we can say that it is constructed by various layers, convolutional,recurrent,fully-connected and groups of layers can be viewed as 'modules' that do different jobs. In the classic sense then a neural network is constructed from two modules, a feature extractor and a classifier. Treating a neural network as a tool for learning useful representations is now common practice and many models exist to learn this representation, which is then employed to solve other tasks. This is common in NLP tasks where training a language model on a large corpus of text is really restrictive and so many models start with word embeddings, see Appendix , from pre-trained models such as BeRT [31]

Exploring the transferability of layerwise weights was the focus of [29]. An important finding was the specialisation of weights the deeper into the network they were. Early convolution filters were more transferable than later weights. They also found that breaking connections between layers can lead to un-recoverable connections due to co-adaptations.

2.4 Understanding Domain Adaptation

The Domain Adaptation problem poses an interesting problem for machine learning models as it highlights where the abstraction that these models learn fails, especially in comparison to human abilities.

To illustrate the problem consider a machine learning model trained to recognise various animals from photographs and apply that model to paintings and drawings, or artistic renderings, here our models tend to fail badly. This appears like a major problem as this task seems simple to a human. The problem goes deeper than considering transfer to abstract renderings of animals; if we train a model on digits from the MNIST digits dataset we don't adequately transfer to SVHN. These models can be really accurate on MNIST, however, if we employ the same model directly on the SVHN dataset we don't achieve anywhere near the performance of training a model from scratch on that problem.

Domain adaptation tends to focus on the problem of transferring learning across differing representations of data for the same task. As seen in the examples above, this is a generic transfer skill focussing on how data is abstracted; however, the problem can occur in very practical contexts such as when developing robots. Robots can be difficult to train and neural nets require a lot of data so physically resetting a single robot multiple times is very time consuming and restrictive and the alternative of setting up a robot farm is extremely costly. The usual solution is to use a simulator and train the robot in a simulation, which is much faster to do and not very expensive. However, the learned policies often do not perform as well when used on the real target robot due to differences between the physical and simulated sensors, among other issues. This constitutes a very practical problem for a which a solution would likely be useful.

This problem is therefore a natural problem to focus on as our ability to train individual task focused models matures. We now introduce some of the methods that have been employed to address the problem. A large volume of this work focuses on the standard supervised learning classification case, moving into unsupervised learning territory. The application of these methods to Reinforcement Learning is significantly limited and this Dissertation seeks to rectify that in later chapter 3

2.4.1 Classic Domain Adaptation

A long time goal of artificial intelligence is to have methods that can generalise, easily, to data not seen during training. This goal encompasses a lot of problems from variations in input to non-overlapping output domains. The domain adaptation case usually focuses on changes to the input domain, such as moving from paintings to photographs. A key idea behind research in this domain is that of a shared subspace. We assume that the data from each domain can be mapped to a shared embedding space which captures the elements of the data which are essential for the given task.

With this view of a shared subspace a traditional approach to domain adaptation is to treat the problem as a regularisation problem.

2.4.2 Deep Image To Image Domain adaptation

With the advent of deep learning and more recently with the success of deep generative models, one way to re-use a model on different image data is to physically transform the image from one domain to another; for example consider the task of turning a picture of a face into an emoji version as in [44]. If we then had an algorithm trained on emoji face data, we could apply it to the newly generated face to emoji translation.

This idea extends work on concepts such as style transfer, an example of which is StyleGAN [45], and so appears to be a natural approach to use for domain adaptation as utilised in [46]. An important aspect for this to work is that the visual contents of the image be similar. This restriction may or may not be important depending on the application domain, however, it is worth noting as it limits the potential distance between domains when performing transfer.

2.4.3 Adversarial Domain Adaptation

A newly developed style of Domain Adaptation, which this thesis builds upon, is adversarial domain adaptation. This is a paradigm centered on learning a shared representation space where the originating domain of a given embedding sample is obfuscated. This obfuscations is achieved through training a domain classifier to separate domains given an embedding. Simultaneously an encoder network is trained against the domain-classifier to make the domain embeddings inseparable to the domain-classifier. Training these two models forms the basis of the learning problem

The domain classifier,domain encoder learning problem relies on optimizing the two models in opposition to each-other. There are two main methods for approaching this task; you can use a GAN based approach and treat the domain classifier as the discriminator and the encoder as the generator or alternatively you can introduce a gradient reversal layer between the domain encoder and domain discriminator, allowing for a more standard optimization approach.

Alternatively, instead of employing a domain classifier, a divergence based method can be employed. These methods include Maximum Mean Discrepancy (MMD) [47] and correlation alignment based approaches.

The MMD approach maps samples through a reproducing kernel hilbert space

(RKHS) and then compares the means of the samples, if they are different they come from different distributions. The learning problem is designed to bring the samples together and a number of variations exist [48, 49, 50]. Correlation alignment [51] works with a similar principle except the alignment is computed through second order statistics.

2.4.4 Domain Adaptation vs Few-shot Learning and Metric Learning

The remainder of this document is focused on approaches that use domain adaptation, however, why domain adaptation is applied instead of one of the other approaches might not be clear as the methods solve highly related and similar tasks. In this section we briefly distinguish the methods from each-other.

Metric learning and few-shot learning are both focused on learning embeddings of data that make classifying unseen data from unseen classes possible. This already introduces a difference with (standard) domain adaptation, where the model classifies data from already seen classes that may not exactly resemble the training data. This distinction can be relaxed to some degree when considering recent extensions such as partial domain adaptation.

Deep metric learning is focused on learning an embedding function that accurately clusters classes far apart from each-other by utilising some measure of distance to bring related classes close together and unrelated classes further apart. While this then can be applied to unseen data, the training data is all labelled. This again distinguishes metric learning from domain adaptation where we assume the target domain is unlabelled.

The distinction is in many ways subtle and can depend on the specific formulation of the problem. Indeed domain adaptation has been applied to the problem of metric learning [52] and metric learning can be considered a form of domain adaptation.

2.4.5 Recent Extensions to Domain Adaptation

The standard assumption in domain adaptation is that the source and target domains entirely overlap in the label space. In reality this assumption may be too restrictive and work relaxing this constraint has recently been proposed. The variants of this can broken down as follows.

Closed Set Domain Adaptation This is the standard domain adaptation paradigm where the source and target label space are shared and the focus is on reducing the domain gap between source and target.

Partial Domain Adaptation. Partial Domain Adaptation [53, 54, 55] Covers the case where the target label set is a subset of the source label space. This is a practical consideration as many large powerful pre-trained models are trained on large datasets such as ImageNet which have a large output space which may lead to class imbalances when performing standard domain adaptation. The approach in [53] is to build multiple per-class domain discriminators combined with instance and class based weighting. This enables them perform distribution matching on a per-class basis.

Open Set Domain Adaptation Covers the case where the not all elements of the source and target label space over-lap, where not every target class is contained in the source and including the case where not every source class is included in the target. Open Set Domain Adaptation was proposed by [56] where they treat the classes unknown to both domains as a single unknown class. The case where the source is entirely contained in the larger target label space by [57]. In their approach the source classifier is extended to include an unknown class target.

Chapter 3

Transfer In Reinforcement Learning: An Adversarial Architecture

3.1 Introduction

In this chapter we present our approach to transferring knowledge for the reinforcement learning problem. The move to deep reinforcement learning introduces a function approximation method for encoding state representations. State representations are the bedrock of classical reinforcement learning, given a 'good' state representation the problem becomes much easier to solve. Each state is its own unique representation to be mapped to an action. Deep Reinforcement Learning introduces the problem of mapping observations to states, or embeddings, which are used to predict the value or action for a given state. A learned function approximation method then transforms these approximate state representations into actions. In traditional reinforcement learning creating the appropriate state representation can be quite difficult. Deep reinforcement learning demonstrates this difficulty, in part, through its requirement for very large numbers of samples from many attempts to learn the problem. This difficulty is exacerbated by the fact the model is solving two problems simultaneously. The first problem of how to map observations to the state, and the second problem of mapping states to actions/values.

An important concept in transfer and domain adaptation is the idea of problems sharing a representation space. Through separating the deep reinforcement learning problem into the two tasks of mapping observations to states and mapping states to actions we can also separate the learning processes. If the mapping of states to actions is already known an adaptation algorithm can focus on aligning the mapping from observations to states.

Deep Reinforcement Learning (DRL) is a powerful paradigm for learning solutions to complex control tasks from raw sensor data. The effectiveness of DRL has been demonstrated through solving complex vision based tasks such as Atari games [21], the complex game of GO [8] and continuous control tasks [58, 59]. Many of these successes rely on simulation because they require numerous interactions with the world to learn a policy.

Since gathering data through interaction with the environment is expensive, especially for physical robots, and acquiring labels even more so, it would be desirable to leverage learning already carried out on a related task. These considerations lead us to consider a domain adaptation approach to the learning problem. The domain adaptation problem considers the case where we have labeled data from a source

domain, which is assumed to be different from, but related to the target domain. By exploiting labeled source data and unlabeled target domain data we aim to reduce the number of samples needed to learn a new policy for the target task. The application of domain adaptation to robotics can be used to solve the move between simulated and real environments and thus can reduce the training time needed to tune tasks in the more expensive real world.

In this chapter we propose a method that aligns the distribution of hidden state representations of a DRL solution to a given source task with that of a new target task. Our view is that for sufficiently similar tasks the state space should share the same features and so we wish to impose these features on the encoding space of an AutoEncoder. We can impose this regularization on the encoding space of an AutoEncoder via a Discriminator network, which learns to separate state encodings from the source and target domains.

The rest of this chapter presents a brief background before outlining our method and architecture. We then present results which demonstrate our method reduces the number of interactions needed to learn a policy. We also show that domain adaptation can work even in cases where the source and target share different action and reward spaces.

3.2 The problems of Deep Reinforcement Learning

Deep Reinforcement Learning, as noted previously, has a lot of potential and has been used to solve many difficult tasks, however, it is also slow to learn and requires a large number of interactions with the environment. In classical reinforcement learning, as in classical machine learning, a lot of effort is put into designing the input features for the algorithm. Having the appropriate features makes learning the solution easier. When it comes to deep learning, a large amount of feature engineering is removed with the promise that a deep network will learn the appropriate function to transform the input features. This promise, however, comes with the expectation of a lot of data samples and the reward signal of RL can be noisy making it difficult to learn.

When considering the problems of Transfer Learning [60, 23] and Domain Adaptation [61] for the reinforcement learning problem we also come up against the decision on whether to employ value-based or policy-based RL [19]? Value-based

methods predict real values for each of the actions and can be seen as a regression style task, policy-based methods are more commonly classification style tasks and Domain Adaptation is more naturally applied in that context.

A second consideration is that even for visually similar games the policy might be very different as it depends on the goal. This introduces a level of complexity when applying these methods to RL on the Atari. When using the OpenAI gym platform [62] to play the Atari, action spaces are masked down to necessary actions for the game so the learned policies do not necessarily overlap. Not only do the prediction spaces not necessarily overlap but even when they do the correct predictions for the same state may vary wildly depending on the goal. We handle this problem by using ADDA to better initialise the embedding and show that this improved embedding improves learning over the baseline.

3.3 Adversarial Autoencoder Domain Adaptation

A common way to interpret domain adaptation is to assume we have a model that correctly classifies the data from a given data distribution. Given such a model we now want to classify some new data, however, the data distribution has been shifted, maybe the colors have been altered. Domain adaptation seeks to learn a mapping back to the source domain from the target.

Generalising from that idea, it may often be the case that some or all of the target classes do not overlap, however, there is still shared information in the domain. Aligning the hidden representation allows for a greater diversity of adaptation as the existing classifier can still be used but alternatively it could be modified or extended. For the reinforcement learning problem it is unclear how these actions could be used to overlap, when visually they represent different concepts, but conceptually they represent the same action.

Our approach is to draw on classic approaches and adapt the embeddings using regularization techniques. If the source embeddings represent samples from the source data embedding distribution, then we want to regularise our target data embedding distribution towards that. Various autoencoder models have been used to impose a , usually Gaussian, distribution on the embedding space of the encoder. We therefore re-purpose The adversarial autoencoder for domain adaptation.

We take inspiration from [63] for reinforcement learning of invariant feature spaces

for robotics. Two autoencoders are set up one for the source and one for the target. The two differently jointed robots then attempt to solve the same task, since they have different sensors the robots "see" the world differently but it is the same world. Samples are aligned through time and the KL-divergence of the autonencoder bottleneck representations is minimised alongside the reconstruction loss. This allows the authors to translate between source and target observations to determine high-level actions.

The time alignment here is a sticking point, not all problems can be aligned in this manner. What we would like to do is impose the KL-Divergence between samples without knowing the alignment apriori. The standard approach for a single autoencoder is to fit the embedding to Gaussian distribution using a variational autoencoder. A recent alternative to this approach is impose the gaussian distribution using a GAN style model, where a discriminator has to predict whether the sample was produced by the encoder or was sampled directly from a Gaussian.

3.4 Background

Here we present the concepts and review recent work in Reinforcement Learning, Domain Adaptation and Adversarial AutoEncoders - which are the main building blocks of our approach. We also introduce Progressive Neural Networks, a leading competitor, for transfer in reinforcement learning on the Atari.

3.4.1 Reinforcement Learning (RL)

RL is a machine learning paradigm centred on learning how to act in an environment. The problem is modelled as solving a Markov Decision Process (MDP), defined by the tuple: (S, A, T, R, γ) where S is a set of states, A is a set of actions, T is the transition function $T : S \times A \to \mathbb{S}$, R is the reward function $R : S \times A \times S \to \mathbb{R}$ with discount factor γ with $0 \leq \gamma \leq 1$. The aim is to learn a policy $\pi : S \to A$ that maximizes the discounted expected reward where the discounted expected reward is :

$$\mathbb{E}_{\pi}[\Sigma_{t=0}^{T} \gamma^{t} R(s_t, a_t)]. \tag{3.1}$$

In our work we make use of the A2C [64, 65] algorithm. A2C is a variant of the A3C [66] algorithm that performs synchronous updates. A2C is an actor-critic

algorithm, so a state value function and action distribution are learned jointly; with the value function critiquing the policy. Actor-Critic algorithms are versatile algorithms, their relationship to generative adversarial networks has been explored by [67] and recent work has applied them to the problem of distributed multi-task learning [68].

3.4.2 Domain Adaptation

Domain Adaptation is a transfer learning approach that seeks to align knowledge gained on a supervised source task with an unlabelled or sparsely labelled target dataset from a different domain. These datasets usually share prediction labels so it is only the representation of the information that changes.

Many methods seek to align the representation vector of the source and target data. This can be done in a supervised or unsupervised manner, depending on labelling assumptions, however, target data is usually limited if it does exist at all. Aligning representations can be done in many ways; perhaps the simplest is to impose a constraint between the predicted representation of source and target data.

This alignment is often pursued through regularization of the embedding space as in [63, 69]. These approaches show much promise, however, they require some alignment of samples from source to target tasks. There are various approaches to this, from using supervised data [70], to assumed correspondences [63], to unsupervised approaches [71, 69]. Adversarial Discriminative Domain Adaptation[69] (ADDA) is a simple and powerful application of a GAN based framework for domain adaptation and a number of works build on this approach [72, 46]. It is also closely related to the Adversarial Autoencoder [18]

3.4.3 Adversarial AutoEncoders

Adversarial Autoencoders (AAE) consist of three networks, the encoder and decoder f and g, respectively, and a discriminator d. Networks f and g are standard components of an autoencoder but the discriminator d seeks to align in an adversarial sense, the encoding distribution with a user-supplied, regulariser distribution. For a dataset that comes from a distribution P_x and a regularizer distribution P_{reg}, the AAE is trained using the following two objectives: (a) the Autoencoder reconstruction loss denoted by:

$$L_{AE}(f,g) = \mathop{\mathbb{E}}_{x\sim P_x(x)} (x - g(f(x)))^2, \tag{3.2}$$

and (b) the GAN loss denoted by

$$\begin{aligned} L_{GAN}(f,d) =& \mathbb{E}_{z\sim P_{reg}(z)}[\log d(z)] + \\ & \mathbb{E}_{x\sim P_x(x)}[1 - \log(d(f(x)))] \end{aligned} \tag{3.3}$$

These two objectives are combined into:

$$L_{AAE}(f,g,d) = L_{AE}(f,g) + \alpha L_{GAN}(f,d) \tag{3.4}$$

where α is a weight parameter. This loss is minimized with respect to the auto-encoder networks and maximized with respect to the discriminator, giving rise to the following MinMax objective [18]:

$$\min_{f,g} \max_{d} L_{AAE}(f,g,d) \tag{3.5}$$

This model treats the encoder of the AE (f) as the generator of the GAN framework; its encodings are trained to mimic the imposed prior distribution and to encode the essential domain information. This is done by passing the encoding through both the GAN discriminator (d) and the AE decoder (g). The network is trained by the two loss functions of the AE and GAN, by interleaving steps of both optimizations. In this way the GAN can be seen as regularizing the encoding of the AE.

3.5 Architecture and Method

In order to achieve knowledge transfer across RL tasks we take the view that related tasks can be described using a shared state representation. Learning the state representation for a single task is achieved as a consequence of learning the policy and as such is slow to achieve. We propose that the learned state representation is also a good initialisation for related tasks. To use this state space as an initialisation we need to learn a mapping from our new observation space to this existing state space.

When considering what is learned by a neural network for a reinforcement learn-

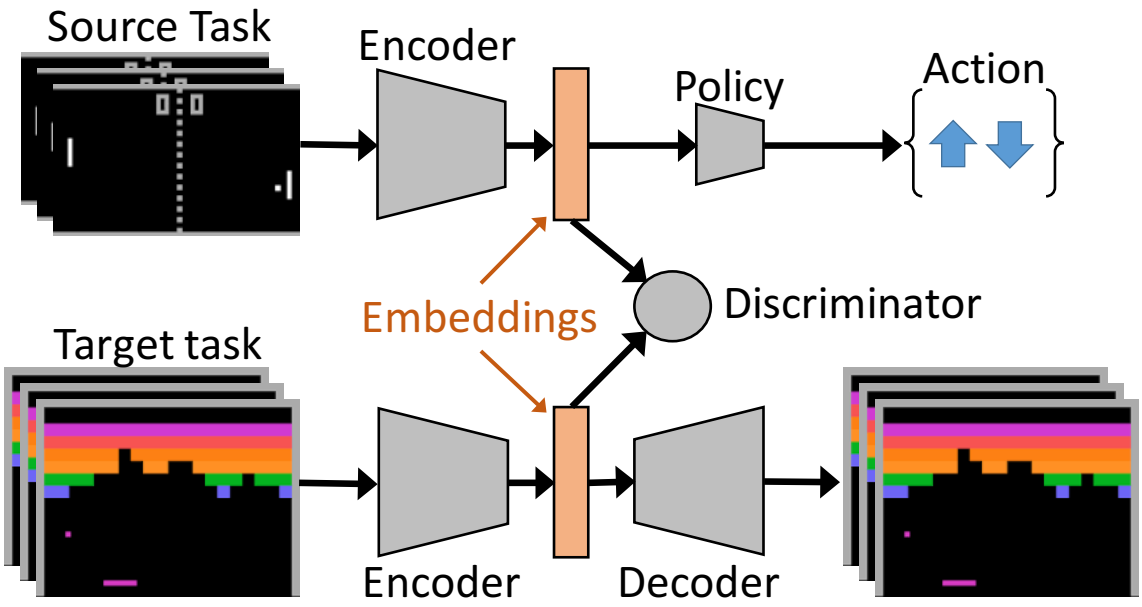


Figure 3.1: Our agent's architecture for domain adaptation, combining adversarial discriminative domain adaptation and adversarial autoencoder architectures. A fully trained source task is depicted on top, while on the bottom is the target task autoencoder.

ing task we can take the view that the network learns some state-representation given an observation of the environment and learns a mapping from that state to a combination of the policy and value function. This view of deep reinforcement learning is not uncommon and can be extended to domain adaptation and multi-task learning. We extend the state-space learning to the idea that there exists a shared subspace which can accurately describe each domain with a set of shared features. Consider games such as Pong and Breakout, both games that involve using a paddle with one degree of freedom to hit a ball; hand engineered features such as centre of the paddle, centre of the ball and ball velocity come to mind as likely shared feature candidates.

So in order to achieve knowledge transfer across RL tasks we will take the view that tasks can be described by a shared subspace and if that subspace is already known learning a new policy will require less new experience.

In this section we propose an architecture for domain adaptation in reinforcement learning. The proposed approach takes inspiration from two existing methods. The first method was proposed by [73]. In this approach two auto-encoders are trained and the KL-Divergence between the embedding spaces of the two encoders is minimised; through this a shared embedding space is learned. The authors focus on differing robots solving the same task and pair input samples through time. Pairing

samples is important when minimising the KL-divergence between samples in order to align the embedding space of the two observation distributions.

The second method we take inspiration from is the Adversarial AutoEncoder [18],described in chapter 2 and briefly recapped here. The AAE is an alternative to the variational autoencoder, imposing a penalty on the embedding space of the autoencoder to create a more-contiguous embedding space. This turns the decoder into a generative model which can be easily sampled from. The AAE as a method applies the penalty on the embedding space by enforcing the KL-divergence between the distribution of autoencoder embeddings and a given desired distribution. This architecture is useful because we can replace the standard Gaussian distribution with any other distribution and sampling method.

One of the issues we have with the approach described in [73] is that we do not know which observations should be mapped to the same regions in the embedding space. This is what they used time based alignment for, however, time does not necessarily apply in every case. We need an alternative. In order to achieve this alternative we will combine the AAE with the autoencoder approach applied in [73].

Given a trained source encoder network b, which maps source observations to state embeddings, we can initialise f with the same weights and freeze the source network weights, as done in ADDA. We can now add a discriminator network, that will be trained to separate samples coming from f from samples produced by b. Encoder f only operates on observations from the target task and b from the source. Into this framework we add the decoder network g which is trained to recreate the target input data from its encoding. This sets up the AAE style architecture and we can use the L_{AAE} we defined in equation 3.4. The addition of the decoder network to the ADDA setup acts as a constraint on the encoding space to force it to retain all relevant information in the image.

The input to the encoder network are samples from the target domain. This is initialised randomly in our experiments, however in many domain adaptation works it is initialised with the source task weights. In many cases this makes sense when there is significant overlap between domains as in transfer between SVHN and MNIST.

We also note that AEs have been used successfully in a number of RL algorithms as part of multi-step training pipelines and end-to-end paradigms. The most obvious

use of the AE is to learn a low-dimensional state representation [74, 75, 76, 77]. Many RL algorithms are designed for low-dimensional state spaces and AEs provide a powerful unsupervised method to learn such low-dimensional representations, rather than building hand-crafted features. This also serves to motivate the use of the AAE.

Viewing the architecture (Figure 3.1) we can say that we initialise our state representation using an AE as a pre-training step. The GAN is used to regularise the embedding distribution. This is the view taken in [18], where the regularising distribution is typically a Gaussian. We utilise the flexibility of the GAN to achieve transfer through the use of non-standard distributions.

Algorithm 1

Trained Source Policy $\pi_{\theta_s}(a|s)$

Initialise discriminator d
Initialise decoder g
Initialise encoder f with pre-trained weights of π_{θ_s} up to the final hidden layer.
for $i = 1 \rightarrow numEpochs$ **do**
  for $j = 1 \rightarrow numBatches$ **do**
    sample target frames: $X \sim \pi_{random}(a_t|s_t)$
    sample source encodings: $Z \sim \pi_{\theta_s}(a_s|s_s)$
    update d by maximizing eq. 3.4
    update f and g by minimizing eq. 3.4
  end for
end for
return f

Algorithm 1 sets out our method in detail. We break with the training procedure for the AAE, by training the networks f and g, with a combined loss of $L_{GAN}+\alpha L_{AE}$ weighting the autoencoder loss with weight α. The discriminator d is trained in the standard fashion. This is more inline with other domain adaptation approaches such as [46] and domain transfer approaches such as [44]

Once this alignment is learned the weights of f can be used to initialise the weights of a new RL agent for the target task with a new layer added for the policy and value functions in our case. This follows other domain adaptation approaches except that our task has changed and so new task specific layers must be learned and the state encoder is allowed to fine-tune.

3.6 Experiments

Our experiments investigate the applicability of Adversarial Domain Adaptation to the reinforcement learning problem. We use the Arcade Learning Environment

(ALE) [78] to provide a set of arcade games which are commonly used as benchmarks for Deep RL. We interface with ALE through the OpenAi Gym platform [62]. Our experiments focus on transfer between pairs of games. We begin by selecting pairs of games which we deem to be related in order to verify the effectiveness of our approach; we leave automatically identifying appropriate source tasks for future work. In this work we focus on three games: Pong, Breakout and Tennis.

Pong is the virtualisation of Ping-Pong and is played with the ball moving sideways across the screen and the player's paddle moving along the vertical axis. Breakout is a brick-breaking game where the player bounces a ball up to break the bricks while moving a paddle side-to-side at the bottom of the screen. Breakout and Pong use a very similar mechanic; however, the image domain, action space, transition function and reward structure are different.

Tennis is a more difficult task than Pong or Breakout and many deep RL algorithms have reported mediocre scores [66]; although human performance reported in [21] tends to be worse. This game also focuses on returning a ball. The action space for Tennis is much larger; in full it is an 18-dimensional action space, as the agent can move forward and backward to cover the whole court. Tennis also switches the agent's sides on the court, multiple times per set, so the control of the agent becomes more complicated as the algorithm must identify which of the two agents it is playing as. This leads to an exploration problem where the agent tends not to score well for a significant number of games.

We start with transfer from Pong (source) to Breakout (target). We describe the experiment procedure in terms of these games, however, we follow the same procedure for the other experiments, averaging the results over four trials.

1. Train agent to play Pong in the standard RL fashion; we use A2C.
2. Use the trained policy to sample data from the source task,Pong, to create a source dataset with 100k frames.
3. Run a random policy through our target game, Breakout, and capture 10000 frames. No learning happens at this stage.
4. Using these two data sets train the AAE.
5. Transfer the weights of the encoder f of the AAE, to initialise the weights of a new policy network for the target task. The policy and value function layer

Task \Method	ADDA	DTN	AAEDDA(ours)
SVHN→MNIST	76.0[69]	84.4[44]	82.8

Table 3.1: A comparison of accuracy achieved by various domain adaptation approaches with our own on the SVHN to MNIST adaptation task.

weights are randomly initialised, as we only learn a state encoder, f, for the embedding

6. Train the policy on the target task.

The model architecture is kept constant across all experiments. We use the standard architecture from [66]. We use 4 seeds and the same seeds are used for each variation of the adaptation and direct transfer and standard training version of the model. The number of adaptation steps is also difficult to ascertain in the context of RL, as the action layer still needs to be learned after adaptation and so standard approaches of checking accuracy are more expensive to use. Instead we train 3 varied values of adaptation and show their results.

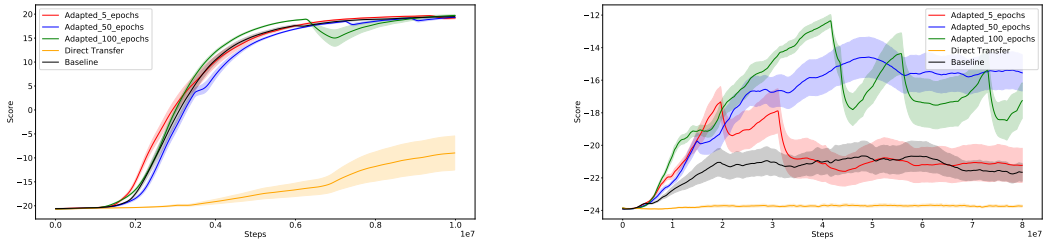
3.7 Results

Our first experiments offer a comparison of our implementation on the task of adaptation between MNIST and SVHN against that of ADDA [69] and DTN [44].

Our algorithm compares favourably against other domain adaptation methods see Table 3.1. We achieve similar accuracy to the DTN. Our model achieves this using a very weak SVHN classifier which was only 88.1% accurate on average. A new model was trained for each run of the experiment. Our method compares favourably to ADDA with the added benefit of a source-to-target translation, via the source encoder and target decoder. This could be further exploited during training in a manner similar to that used in the DTN, which with a consistency loss would also move representations to the correct number to number mappings.

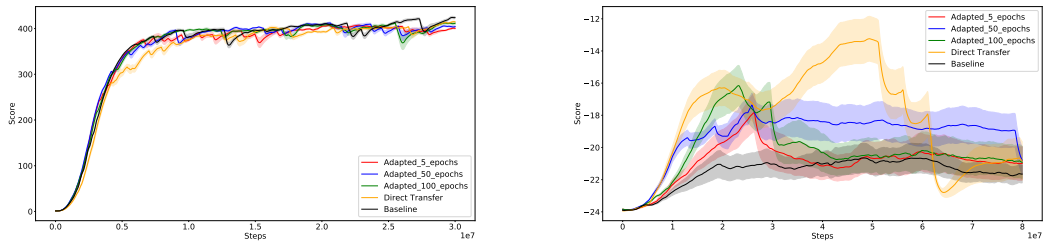
3.7.1 Reinforcement Learning Transfer

Our experiments compare against a direct transfer baseline where the source network, excluding the policy and value function layer, is directly transferred. This network with new policy and value function heads is allowed to train on the target task.



(a) (b)

Figure 3.2: Learning Curves for Pong and Tennis, showing the average score per game against number of steps. We show adaptation models with varying adaptation training time and direct transfer alongside a baseline learning the task as standard. The adapted models are off-set by the 10000 frames from the target task used for adaptation. Target Task: Pong/Tennis, Source Task: Breakout .

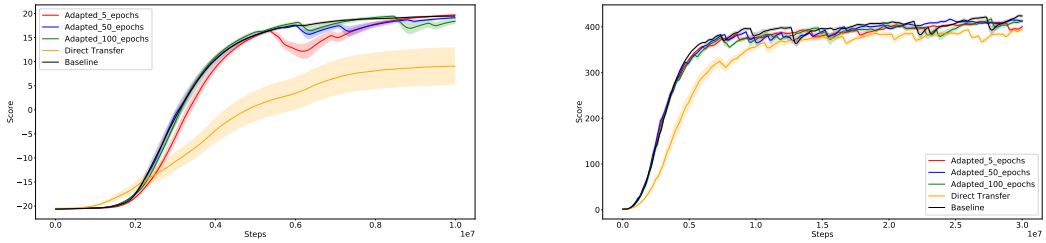


(a) (b)

Figure 3.3: Learning Curves for Breakout and Tennis, showing the average score per game against number of steps. We show adaptation models with varying adaptation training time and direct transfer alongside a baseline learning the task as standard. The adapted models are off-set by the 10000 frames from the target task used for adaptation. Target Task: Breakout/Tennis, Source Task: Pong .

For each game, we train 4 baseline models. For each adaptation task we use the corresponding baseline model to sample 10000 frames from the source game, and a randomly initialised network to sample from the target game. After adaptation the adapted model is trained on the target task with a new policy and value function layer. For each stage the same seed is used and we use the same seeds across each of the transfers. We also run the tasks with varying numbers of adaptation steps and find similar behaviour across adaptations. In all the following graphs the adapted learning curves are off-set by the 10000 samples used for adaptation.

Transfer for Reinforcement Learning on the Atari produces a mix of results. What can be said is adaptation can produce a positive impact and is generally not harmful but may not produce significant benefit. Adaptation is also relatively easy to apply in comparison to training the full RL model and our example uses only 10000 samples which is a tiny fraction of the total required to train an RL model on the Atari.



(a) adapted to Pong

(b) Adapted to Breakout

Figure 3.4: Learning Curves for Pong and Breakout, showing the average score per game against number of steps. We show adaptation models with varying adaptation training time and direct transfer alongside a baseline learning the task as standard. The adapted models are off-set by the 10000 frames from the target task used for adaptation. Target Task: Pong/Breakout, Source Task: Tennis

Breakout presents as the strongest overall game to transfer from, providing a small jumpstart when used to adapt to Pong and producing significant performance improvements when used to adapt to Tennis. Perhaps interestingly, the least successful direct transfer also arises when using Breakout as the source task.

Direct transfer of Tennis to Pong or Breakout hurts overall learning speed and performance as seen in figure 3.4. The effect of adaptation is to largely mitigate the negative effect of direct transfer, however as can be seen adaptation from Tennis does not lead to improved learning. This may in part be related to the weakness of the Tennis model itself, a much harder game, it is difficult to learn tabula-rasa.

Further our exploration of the number of epochs used for adaptation demonstrates the largely stable nature of the adaptation setting. However, it also demonstrates the difficulty of RL and unsupervised domain adaptation more generally. Setting the correct cut-off point for adaptation remains a challenge.

Our results show that adaptation can provide benefit to the reinforcement learning task and is seldom harmful to performance. This is distinct from direct transfer which can produce significantly negative effects. We thus conclude in the reinforcement learning case that our approach is more stable than direct transfer with potentially strong benefits. However those benefits are task dependant.

3.8 Related Work

Our method is closely related to a number of AE transfer learning methods that seek to align the encoding distribution of two data-domains. This is usually done with reference to the Kullback-Leibler (KL)-divergence between the two embedding

domains which can be implicitly or explicitly minimized. In [70] an AE with encoding and decoding weights shared between source and target domains is trained and the KL-divergence between their embeddings is explicitly minimised in the cost function. Their cost function also directly incorporates source label information as a regularization term by using the embedding layer as input for a softmax prediction layer.

The work in [73] imposes a Kullback-Liebler divergence constraint on the encoding layer of two AEs trained in parallel. This approach estimates an alignment through time, as two differently actuated robots solve the same task. This approach is successful but requires estimating some form of pairwise alignment between samples. Our method differs from those methods by aligning encoding distributions in an unsupervised manner via Discriminator network feedback as in the AAE method. This method utilises the Generative Adversarial Network learning algorithm to incorporate source domain information into our target domain embedding. Our approach extends the use of AE-based transfer learning in reinforcement learning by utilising methods developed for adversarial adaptation.

The adversarial approach used is related to a variety of Adversarial Regularization methods[79, 80]. The most similar of which is the AAE of [18] where the discriminator is used as a regularisation term on the AE embedding. We adapt this by introducing a regularisation space that transfers information from a source task. The AAE itself is closely related to a number of approaches such as variational autoencoders (VAE) [81].

A similar approach is taken in [82] where target images are mapped to the embedding output of a given layer in the source network via an adversarial module. Recent work [80] has also utilised a similar architecture and approach in a supervised context and demonstrated some success, however, in their work the source and target tasks share an encoder. Our approach by contrast demonstrates successful transfer across a significantly wider domain shift. [69] provides a general framework that describes the various parts of Adversarial Domain Adaptation approaches that can be changed, and what has so far been attempted. In [44] instead of aligning the embeddings, as in ADDA, the authors train a shared generator, aligning the output image space, using a discriminator. The method trains an autoencoder to process input samples from the source or target domain and reconstruct or generate the im-

age as if it came from the target domain. An extra consistency loss is included such that the encoder of the autoencoder encodes the source image and the reconstructed source image to the same embedding, this ensures consistency. This method can be used to generate noisy samples for the target domain with labels, so a model can now be trained on source domain data through this method.

A recent approach [83] used a GAN to transfer between two problems in the RL setting, using the GAN loss as an extra reward variable within their RL training. The approach explores how this reward augmentation can be used to achieve learning when the environment reward is missing or uninformative. The domain, however, is much more aligned as they focus on transfer for robotic agents moving from simulation environments to real environments.

3.8.1 Transfer and the Atari

Transfer learning has been attempted in various forms on the Atari. An early approach following the DQN [21] was the actor-mimic [84] where a multi-task atari agent was trained trying to match expert actions and representations. This network can then be used to intialise the weights of an agent for some target task. This pre-training mechanism initialises the filters of the target agent allowing it to start with a useful state-space.

The PNN [85] approach while useful appears weaker in the single step transfer case and understanding which source tasks to use for transfer remains an open question. Recent work (progress and compress) by [86] has looked at limiting the growth of progressive networks, [85], by consolidating the learnt target network into the source network during compress phases and their approach shows interesting results.

Recent work [79] has attempted an alignment similar to ours with mixed results. In their approach the authors only seek to train the state encoder to fool a discriminator without the inclusion of the reconstruction objective. This worked well for variations of the same game, which they introduced, however, transferring across games proved more difficult. Some games such as Pong to Breakout did transfer favourably, which our own results corroborate.

3.9 Related Methods

As discussed our proposed approach has been developed as repurposing of the AAE, using the the discriminator to impose an arbitrary distribution on the embedding space of the target encoder network. Our approach however is part of a family of methods that was developed concurrent to our own work.

Adversarial Discriminative Domain Adaptation is a framework for describing models that perform domain adaptation using some form of adversarial signal. This can be a GAN, as in the case of ADDA, or alternatively a gradient reversal layer has been proposed [87].

Other similar work develops the idea of a shared space through the use of autoencoders, rather than through the inclusion of a domain discriminator. The deep reconstruction classification network [88] is one-such approach. In the robotics space a variational autoencoder approach has been employed by [89], where a fixed decoder is shared between source and target, and a pair of autoencoder embeddings are learned in parallel and aligned using KL-divergence in [63].

These models form a family of approaches built around the idea of learning a shared, invariant, embedding which can be used in conjunction with an existing source policy. Our work is one such member of the family incorporating an penalty in the form of a reconstruction loss on the adversarially aligned embedding space and demonstrates that the policy does not need to remain constant for the adaptation effects to be beneficial.

3.10 Discussion

We have presented an adversarial method for knowledge transfer in reinforcement learning. We have demonstrated how this approach can be used for domain adaptation to improve performance on the difficult task of learning to play Atari games. We demonstrate how we can also change the action space and the reward function and derive benefit from the transfer approach.

Our work demonstrates how effective the Adversarial AutoEncoder can be when applied to the unsupervised transfer learning problem. We demonstrate positive transfer in problems where intuitively transfer should be possible, even under significant domain shift. We also observe the difficulty of applying direct transfer.

Adversarial Adaptation methods offer a powerful framework for domain adaptation. We have demonstrated how a simple application of the technique can help learning in the reinforcement learning case, even when the final layer needs to be relearned as the action space and task have changed. This shows that the technique is not limited to simpler or more closely-related domains as previously explored such as MNIST to USPS, or SVHN to MNIST. Future work will extend the architecture and apply the method to the sim-to-sim and sim-to-real transfer problem. Such an application could aid in the use of DRL for the robotics case as it helps to reduce the volume of experience required in the target, real, environment. Robotics applications can further benefit from maintaining the same action space which allows for a fuller application of domain adaptation as the policy layer can be kept. An obvious drawback of our current approach is the random sampling of data followed by alignment, followed by reinforcement learning. An extension we envisage is integrating the adaptation with the reinforcement learning through iterative sample target then adapt training cycles.

The Adversarial Domain adaptation approach offers a rich space of algorithms for learning a shared representation of the world. In this paper, we have demonstrated its versatility across very distinct image domains, with far greater semantic differences than are demonstrated in other similar works. Our approach demonstrates transfer across different worlds with different targets through representation alignment. This demonstrates the power of adversarial domain adaptation methods and provides for a variety of future research directions.

Chapter 4

Understanding the Effect of Embedding Dimensionality

In chapter 3, we introduced Adversarial Autoencoder Disciminative Domain Adaptation (AAEDDA). The method presented learns an embedding representation that is shared between the source and target task. The use of the autoencoder is designed to shape the learned embedding. We now probe other methods of controlling the embedding space. In this chapter we examine the effect of embedding dimensionality and in chapters 5 and 6 we explore the use of singular value decomposition for interpreting and controlling the learned embedding.

4.1 Introduction

In this chapter we address the question of *transferability* and *discriminability* of the learned model in relation to the dimensionality of the embedding vector. These characterise a models ability to classify samples in the source and target domains. Domain adaptation utilising GAN-based methods has had some success with numerous improvements and application works demonstrating the utility of the method. With many potential applications for applying domain adaptation methods it is perhaps reasonable to examine the source models used and whether or not small changes to source training can improve the generalisation of the model to new domains.

GAN models are often used in domain adaptation and there are still many questions surrounding the development and use of GANs themselves, not considering their use as tools in more complex systems. As was noted recently in [90] the size of the noise dimension passed to a GAN has received very limited scrutiny and can affect the generative ability of GAN models. This seems natural; the capacity of the noise dimension can be too low or too high and so a sweet spot exists, until significantly larger noise dimensions are used. For domain adaptation the size of the embedding dimension may have an impact on the transferability of a model.

A critical aspect of Adversarial Domain Adaptation is the trade-off between the transferability and discriminability of the embedding [61, 11]. Models that are very capable of discriminating in the source task are susceptible to learning domain-specific features and over-fitting to characteristics of the training domain instead of the underlying characteristics of the problem, which are assumed to be more general and transferable. More general features may be more prominent when learning, however other data specific details may improve classification accuracy as well. Through

restricting the dimensionality of the data embedding a more general representation can be learned.

From another point of view it could be that it is more difficult to align larger dimensional embeddings as many small differences add up to bigger changes. Thus, we propose that the optimal embedding is large enough to capture a useful representation for the source task, but that past a threshold size it becomes harder to align a target embedding with, as it contains spurious information not present in the target data set.

We can see in works such as Wide Residual Networks [91], that the number of feature maps per layer, and thus number of features generated per-layer, impacts the learning process. [91] show classification improvement with increasing width. This runs somewhat counter to the belief that depth provides a regularising effect and wider networks lead to over fitting. For domain adaptation this potential effect is perhaps more problematic as we wish to generalise more broadly across domains than traditional classification tasks.

Given this landscape, we investigate the effect of varying the width of the final hidden representation of our model. If there is a size based effect, is it dependant on the model or on the learning problem and what guidelines can be derived for developing models for domain adaptation in the future.

In the rest of this chapter we highlight the ADDA model for domain adaptation and apply it across a range of models with varying hidden dimension sizes. From this we show a thresholding effect on the size of the embedding space that can improve transfer to the target task.

4.2 Background

Domain adaptation focuses on the case where we have labeled data from a source domain and unlabeled data from a semantically-related target domain. The aim of domain adaptation then is to take a model trained to solve the source task and apply it to solve the same task in the target domain. We can solve this task through aligning the classifier embeddings of the target with the embeddings of the source, either during source training or as an adaptation step after.

Adversarial Discriminative Domain Adaptation is a framework for adapting the weights of a learned network to better classify data from an unlabeled target domain.

This is achieved through the use of an adversarial training structure by utilising a generative adversarial network (GAN) training framework.

The basic GAN setup is to use two networks, a generator network and a discriminator network. The generator network creates samples that should match the source domain distribution and the Discriminator network separates real samples from generated samples. These networks are trained in opposition to each-other and a generative model of the source domain is learned.

In the case of ADDA the source domain is the embedding or label space of the trained source network and the generator is the target network which is trained to match the source distribution of a pre-trained source classifier. The discriminator acts as a domain discriminator and is trained to distinguish between source and target encodings. Many alternatives and variants have been proposed for this problem, some with more overhead than others.

4.3 Domain Adaptation and Embedding Dimensionality

In adversarial domain adaptation, the classifier is usually assumed and what needs to change are the low level filters which do not fully capture the new input distribution. Through adapting the neural network filters and better aligning the target embedding with the source embedding to create a shared embedding space. In this chapter we investigate the effect of controlling the dimensionality of the shared embedding space and whether narrower embeddings allow for better adaptation.

We use the Adversarial Discriminative Domain Adaptation (ADDA) model [92] and explore two alignment options across a variety of feature embedding sizes. We show the effect of the hidden dimensionality on adaptation and from this we provide guidelines for building models that will be used in adversarial domain adaptation contexts.

4.4 Experiments

Our experiments train models with varying hidden embedding dimension size and show the change in transferability and discriminability as the dimensionality of the embeddings increases. We investigate the effect of the embedding size on the direct transferability of the model and how well that model can be adapted by the ADDA

method.

4.4.1 $SVHN \rightarrow MNIST$

The number recognition tasks of SVHN and MNIST provide a useful domain for experiments where we would expect a model to generalise. For this task we use the network architecture used in [44]. This aids in allowing us to compare our results against other published results; with the caveat that we adjust the dimensionality of the final hidden layer. This model was also used in [46].

4.4.2 OFFICE-31

The office-31 dataset contains images of office furniture sourced from 3 different settings. This allows us to create 6 transfer tasks. We use a pre-trained AlexNet model and train a new embedding layer of various sizes alongside the new classifier layer. We extend the model with a new classifier layer and our variable embedding dimension replaces the normal output layer, which is traditionally changed for new domains. We also note that the pytorch implementation of AlexNet appears not to achieve the direct adaptation achieved by implementations used in other papers; we believe this is a known issue relating to differences between the pytorch and caffe implementations.

Office-31 provides 3 domains of images to transfer between with 31 classes. The images come from Amazon product photos, Webcam photos or DSLR camera photos. This creates 6 potential transfer tasks. Our aim is not to compete with state of the art approaches to domain adaptation, but to highlight how the embedding dimensionality can impact performance [1].

4.5 Results

4.5.1 $SVHN \rightarrow MNIST$

We show, similar to [11], that models that place importance across many of the top features, as indicated by the singular value decomposition, achieve higher accuracy than models that place more importance on a single value, see figures 4.1 and 4.2.

[1] To that end we note that the adaptation scores reported here for AlexNet are less than those reported elsewhere. To the best of our knowledge this is due to differences in implementation between the Caffe and Pytorch framework's provided models.

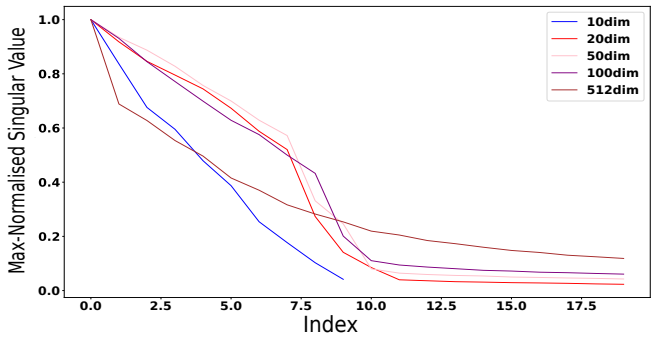


Figure 4.1: A comparison of the max-normalised top singular values of the 10, 20, 50, 100, 512 dimensional embedding models. We can observe that the 20,50 and 100 dimensional models weight more features as being of high importance.

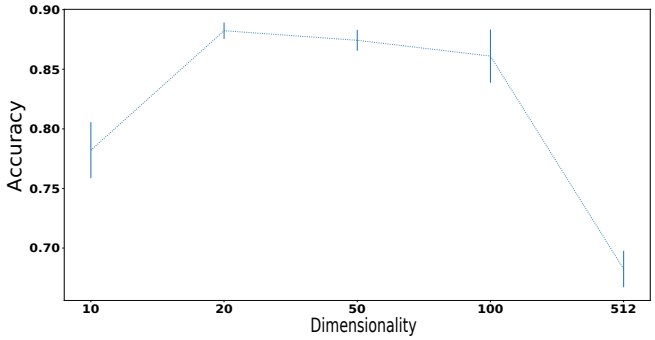


Figure 4.2: A comparison of the accuracies of the 10, 20, 50, 100, 512 dimensional embedding models. We can observe that relatively small embeddings perform best.

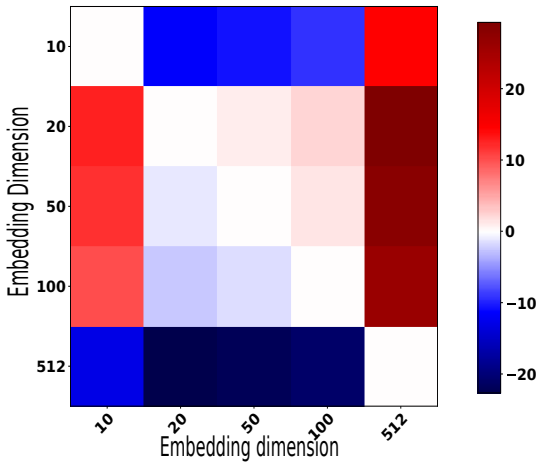


Figure 4.3: A heat map comparing the difference in accuracy between the various embedding size models on the SVHN to MNIST task for the DTN based network architecture. We can see the central region produces the most improvement with larger or smaller embeddings being worse.

This therefore serves to corroborate the claims made by [11] as well as demonstrate that simple architectural changes can achieve similar effects to state of the art penalties.

Clearly when aligning the feature embedding space dimensionality matters and our results show positive adaptation results for middling values, neither too big nor small, as seen in the heatmap presented in figure 4.3. We now show how the feature space dimensionality affects adaptation of the classifier embedding. We show that the dimensionality of the embedding impacts the performance of the model post adaptation and that lower dimensional embeddings spaces appear preferable. Further we also show the direct transfer performance of the models.

4.5.2 OFFICE-31

Here we show the results of transfer from the Amazon image dataset to the DSLR and Webcam image datasets in figure 4.4. Observing the results for direct transfer from Amazon data it can be seen that the models peak and then remain relatively stable with a small decline. This is suggestive of the model potentially overfitting on the source task as it can capture more degrees of variation that may be task specific. We also observe that accuracy on both source tasks peaks with only 150 dimensions, significantly fewer than the 4096 standard for AlexNet.

Adaptation follows a similar pattern. We observe an early performance peak after which accuracy begins to decline. We can also examine the comparison of accuracy between embedding dimensions after adaptation which we show in figure 4.5. Observing the relative improvement based on embedding dimension we observe that neither small nor large embeddings produce the best results. It is not good to be too small, but too large is also possible.

These observations make sense when considering the problem being solved. A general representation needs to capture the problem space but should not focus on domain specific features. Further, feature selection has in some cases been shown to improve model performance [93] and reducing the feature vector size can be viewed as a form of feature regularization.

We conclude that the embedding dimensionality does indeed have an effect on the transferability of models and that smaller embeddings tend to lead to better adaptation accuracy in the target task.

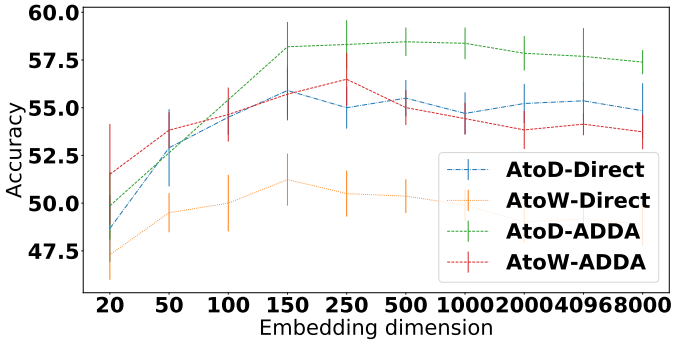
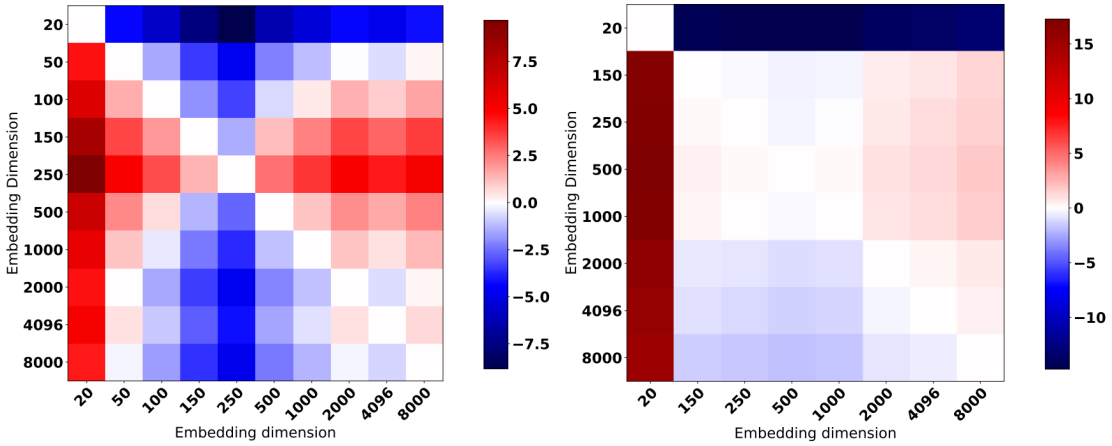


Figure 4.4: Direct transfer and ADDA Adaptation for the Amazon to Webcam, AtoW, and DSLR, AtoD, tasks are shown. We compare the effects of varying embedding dimensionality from [20, 50, 100, 150, 250, 500, 1000, 2000, 4096, 8000]. ADDA adaptation provides improvement over direct transfer and increases the degree to which embedding dimensionalty affects performance.



(a) Amazon to DSLR-Direct

(b) Amazon to DSLR-ADDA

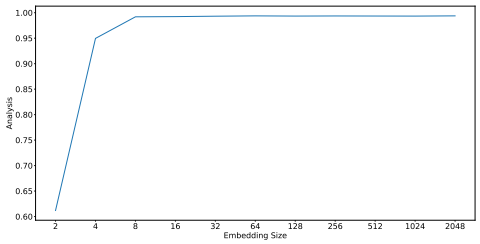
Figure 4.5: HeatMaps showing relative improvement or dis-improvement of a given embedding size against each other embedding. Read row-wise we can observe the relative merits of that embedding to each other. We show direct transfer from Amazon to DSLR, and Adaptation from Amazon to DSLR. Here we can see 20 dimensions is too small and is always worse. We also observe that embedding size has a similar impact in each case, highlighting how relatively small embeddings produce the best results.

4.5.3 Examining Source Embeddings

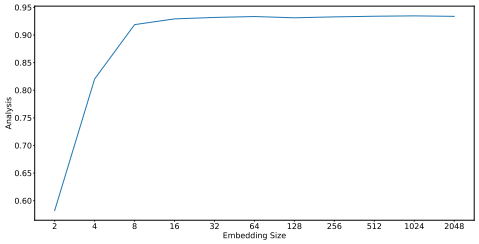
To further our discussion we also include an examination of the minimum embedding dimensionality required on the source tasks.

Examining MNIST and SVHN as seen in figure 4.6 the results are unsurprising. A small embedding of size 2 struggles to capture the full solution however an embedding not significantly larger of 8 or 16 points captures the solution, with larger embeddings being unnecessary.

Examining the same phenomenon in Office-31 we split the datasets into train-test sets, as they don't normally come with separate train and test sets. We use

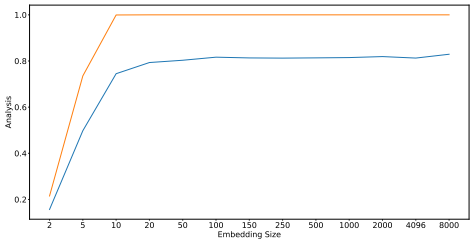


(a) MNIST

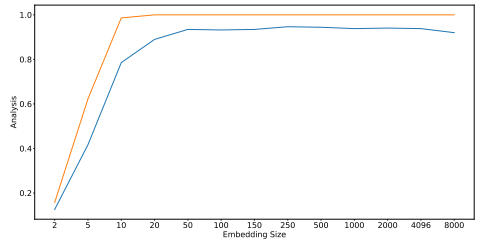


(b) SVHN

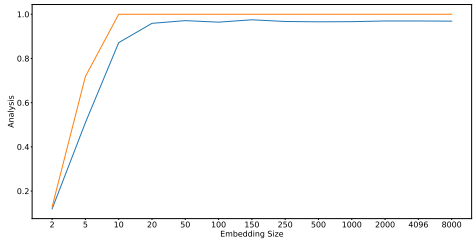
Figure 4.6: an examination of the embedding dimensionality required to solve mnist and svhn



(a) Amazon



(b) DSLR



(c) Webcam

Figure 4.7: an examination of the embedding dimensionality required to solve the office-31 tasks, The training accuracy is recorded in orange and the testing accuracy in Blue. At or above 20-dimensional embeddings the benefits of more dimensions appear to level off.

Pytorch pre-trained Alex-net and replace the final emebedding and classifier layers as appropriate for the desired dimensionality. Finally we train the new classifier on our training split. We do this for 5 seeds.

We observe similar behaviour to MNIST and SVHN as expected. Above a small minimum number of dimensions adding more dimensions does not significantly improve classification accuracy. This coincides with our investigation of adaptation. There does exist a minimum size but above that more dimensions do no necessarily add value and may hinder adaptation.

4.6 Discussion

The importance of controlling the feature space of a model is well known. Feature engineering can provide boosts in accuracy and avoid spurious relationships. Neural network models are well known to overfit to the problem they are trained on, using spurious features of the dataset that are not intrinsic to the concept being predicted. Our contention is that this can hinder domain adaptation approaches and that smaller feature vectors are forced to hold more transferable information which avoids some degree of overfitting.

Our results indicate the importance of the concept of neural networks as feature extractors and show that designing source models, with the intent of using them for transfer, can be easily done to achieve a performance improvement. Direct transfer is impacted by the dimensionality of the learned feature space. Small feature spaces do not fully capture the shared space and are less transferable. Large feature spaces capture too much of the source task and do not generalise well. The feature spaces that achieve the best direct transfer are significantly smaller than the default embedding size for the chosen models, which serves to verify our claims.

Further investigations of the effect of embedding size domain adaptation validate the idea of using smaller embeddings which benefit from being better aligned before adaptation. We conclude that controlling the size of the embedding leads to improved performance. While our work proposes directly restricting the number of features used there remains scope for a learned dynamic approach to reducing the number of features used during alignment.

Our contribution is among the first to explore the impact of embedding dimensionality specifically for domain adaptation. We demonstrate the benefits of using

smaller dimensions for adaptation. In particular, the width of the final linear embedding before the classifier. This provides a focus on the dimensionality required to represent a task.

Related work subsequent [94] ours examined complexity, within the DANN adaptation framework. They demonstrate the impact of network depth and embedding width on the solution. However their aligned embedding is not the final embedding before the classifier and it is difficult to directly compare results. Their results demonstrate a larger impact can be derived from network depth than width, however network width does produce an impact as well. Width is also easier to change, with deeper networks requiring more compute.

Our work also serves to further demonstrate the utility of the our proposed adversarial autoencoder domain adaptation architecture. We explored the role of embedding dimensionality in this context which also differs from the DANN model explored by [94]

4.7 Related Work

Domain adaptation offers a tool for making models that are costly to train, through the need for specialist annotated data in domains where such data is hard to come by, more applicable as it can help overcome differences in data-collection introduced through different machines or collection processes. A number of papers have shown how ADDA can be applied to fields such as healthcare.

An alternative to embedding alignment is to perform image to image translation. Given an image translate it to look like a sample from the source domain and then apply the trained source domain classifier to classify the image. ADDA can also be in an image translation domain, as shown in [46], where after domain translation is done, the translated images are further aligned using ADDA.

ADDA offers versatility in application. Aligning network emebeddings from various levels of the network, [72, 95], offers a more fine-grained approach to domain adaptation where the part of the network that is shared can vary in degree, aligning only the initial filters or the whole classifier network [87, 92]

The batch spectral penalty offers insight into an important aspect of domain adaptation, highlighting the trade-off between domain alignment and classifyability. In the paper a penalty on the singular value of the batched embeddings is intro-

duced to force the target network to generate samples that spread the importance of features across the embedding counteracting the tendency of the model to learn an emebedding which focuses importance on one or two key features. The penalty seeks to maintain the discriminability of the samples while aligning them.

4.8 Conclusions

There are a number of levers to control ADDA, and similar approaches, through the addition of various auxiliary tasks and losses. These all serve to control what is aligned and what is not. Domain adaptation is concerned with detecting the common elements of two distributions. Smaller distributions force the model the concentrate on distinguishing factors.

We have shown that the dimensionality of the embedding space used for domain adaptation has an impact on the achieved performance. Embeddings need to be large enough to capture the problem however too large and they start to decline in performance after adaptation. Our observations here suggest that feature compression methods may be useful in achieving better adaptation. In future work a compressor layer could be learned which would allow for better alignment without the need to train a large model from scratch with the smaller embedding space. This would make the application much more versatile.

Chapter 5

The effect of Autoencoders and the Batch Spectral Penalty on transferability and discriminability

Auto encoders are a powerful unsupervised mechanism for representation learning and data compression. Many variants have been produced and we have explained some of the basics in chapter 2. In this chapter we investigate how an autoencoder can be used to produce transferable and adaptable representations. We also look at the autoencoder in conjunction with the batch spectral penalty, which had a positive impact when applied to existing adaptation architectures.

5.1 Introduction

Classifiers are only as good as the data they have to classify. Neural networks transform complex raw data into compressed separable representations from which various tasks, such as classification, are achieved. Arriving at a useful representation is achieved through architectural design of the networks internal mechanisms, through the inclusion of additional optimisation targets which guide the network towards 'good' representations or even through the use of additional pre-training tasks such as auto-encoders or the use of pre-trained networks such as VGG. Indeed the widespread use of pre-trained networks suggests that having a good initial embedding is an important aspect of neural network training.

When considering the importance of individual features a natural approach is to look at the singular value decomposition. This was done in recent work when investigating the trade-off between transferability and discriminability when performing domain adaptation. In [11] they hypothesise that transferable representations will have fewer important singular values, placing more weight on one or two shared key features, in comparison to discriminative representations and that by raising the importance of more features transferred representations will also be more discriminable.

While classifiers may not gain much benefit from such a penalty there are many unsupervised learning models where we wish to learn and impose a structure on the data embedding. Understanding the utility and applications of the newly proposed penalty term is the goal of this paper.

We investigate the utility of the Batch Spectral Penalty (BSP)[11] when used during standard model training of creating more discriminable representations. We propose a variant of the penalty, which we call the Batch Spectral Ratio Penalty (BSRP). We evaluate the learned embeddings within domain as well as evaluating

the model performance when applied to out of domain data. The remainder of this chapter introduces the BSP and our proposed variants. We then describe our experiments and discuss the results.

5.2 The Batch Spectral Penalty: Method

The batch spectral penalty was proposed in [11] where the authors note a key factor of adversarial adaptation methods is that they align best, to fool the discriminator, when one or a few elements of the embedding space are important as judged by the singular value decomposition. This however can come with a negative consequence as discriminative representations rely on a broader array of parameters to separate small details. This creates a tension between learning representations that will improve classifications and representations which will easily transfer.

In this chapter we look at how the singular values are influenced by different source learning methods and whether the BSP can be used to improve the direct transferability of the source networks. We propose a variant of the BSP and compare the effects of the Autoencoder element of our AAEDDA approach on the singular values and how the BSP performs in conjunction with that.

5.2.1 The Batch Spectral Penalty

The Batch Spectral Penalty (BSP) was recently proposed in [11] in the context of domain adaptation models. The authors note that the discriminative power of an embedding is often traded off against the transferability of those embeddings. When using an adversarial domain adaptation model, including a penalty to keep the discriminative power of the adapted embedding intact is their solution. To design a penalty that maintains the discriminability of the learned embedding we can look to the singular values of the feature space. Intuitively a representation that has a sharp drop off in the importance of features, as evidenced by the singular values, relies heavily on one or two key features and so we may be less able to discriminate between similar data points.

With this in mind we can think of a penalty that stops the top features from becoming too important by penalizing their values. Recall that for a given feature

matrix $F = [f_1 ... f_b]$, its singular value decomposition can be written:

$$F = U\Sigma V \tag{5.1}$$

where b is the batch size for each mini-batch in our training procedure. The singular values are then represented by Σ.

SVD is commonly used in traditional machine learning. The largest singular values represent the most significant concepts captured by the singular vectors. Applying a penalty to the singular values forces the algorithm to spread focus, from one or few concepts to relying on more input features. This approach reduces the chance the learning algorithm will focus on a single feature to make classifications, such as using colour as a proxy for actual recognition in image processing tasks.

Through restricting the use of dominant concepts a more general representation is possible since the algorithm will have to focus on identifying less common features that can be used to distinguish classes. This insight originates in the observation that adversarial domain adaptation tends to hyper-focus the learned representations along one alignable concept, to the detriment of classification performance. In this chapter we examine the application of the penalty in the standard classification task and demonstrate that generalisation can be improved through the inclusion of the penalty.

Understanding the embedding as a learned feature space we can apply SVD in a similar manner to the embeddings to understand the most important features. Since these features are learned we want to limit the importance of features unless they really are important. By penalising the singular values for being too large, or too small, the information can be spread across more of the embedding space. colorblack

We can now write a loss to suppress the dimensions with the top singular values.

$$L_{bsp}(F) = \Sigma_{i=1}^{k}(\sigma_i^2) \tag{5.2}$$

where σ_i represents the i^{th} largest singular value. This gives us the source-only version of the BSP penalty proposed by [11]. This penalty however can pull every value down in an unrestrained fashion. Indeed we observed the singular values after training with this penalty could be almost all pulled to close to 0, which tended to lead to worse performance this motivates our proposal of two alternatives, the Batch

Spectral Sum Penalty (BSSP) and the Batch Spectral Ratio Penalty (BSRP).

The sum penalty takes the focus off the scale of the Singular singular values, and reduces the pull effect to make more important features closer in value to less important ones.

$$L_{bsp}(F) = \Sigma_{i=1}^{k}(\sigma_i) \tag{5.3}$$

The BSRP also takes focus away from the scale of the values and also adds the possibility of raising the importance of the weakest feature leading to more relevant features overall. This effect removes the strong minimizing effect of the original BSP.

$$L_{bsp}(F) = \Sigma_{i=1}^{k}(\frac{\sigma_1}{\sigma_k}) \tag{5.4}$$

5.3 Experiments

In this chapter we investigate the utility of the BSP when applied to a single model that may be re-purposed for another task after training. Does the BSP generally help us to learn embeddings that are more discriminable even when we train on a non-discriminative task such as using an autoencoder?

Autoencoders are well known unsupervised pre-training models that can be used to learn representations as useful if not better than using some off the shelf pre-trained network such as ResNet50 with much fewer embedding features [96]. Since an autoencoder is not trained as a discriminative model we can use such a model to evaluate whether or not the BSP creates a more discriminable embedding space.

We train an autoencoder model on a variety of base datasets; we use MNIST, SVHN and FashionMNIST. These autoencoder models are trained with either one of the three BSP penalties or without any BSP penalty, in addition to a standard reconstruction loss. This creates a set of trained embedding models, the encoder part of the autoencoder, which we can freeze and use to encode data.

When applying the BSP penalty we have a choice over the number of elements to sum over, following [11] we use the top 1 value only for the BSP and BSSP. In the case of the BSRP we also sum only the ratio of the first to last singular value. The number of singular values computed by the SVD is limited by the size of the batch. Here we train with batches of 100 data points.

Returning to the trained models, we can then use these encoders to evaluate the discriminability of the learned representation. To do this we train a logistic regression model using 10-fold cross-validation on the representation of the test-set learned by the autoencoder; this is done to avoid potential leakage from the training set data which we used to learn the representations and follows the evaluation method used in [96].

Another method to evaluate the embeddings is to evaluate how well the embeddings cluster. To do this we perform K-Means clustering using cluster purity as our evaluation metric. Each of the datasets has 10 classes and so we set k to 10. Cluster purity is defined as:

$$Purity(\Omega, C) = \frac{1}{N}\Sigma_k \max_j |\omega_k \cap c_j| \tag{5.5}$$

where Ω is the set of clusters and C is the set of classes. K is the number of clusters and N is the total number of datapoints.

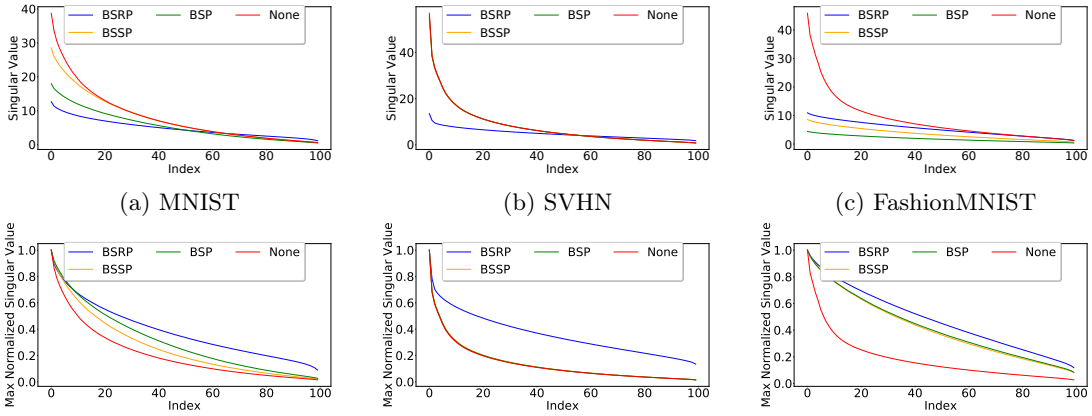
To train better models is also to train models that are robust to changes to the input domain. We investigate this by performing a direct transfer domain adaptation experiment. We use the pre-trained autoencoder network bodies and apply them to an alternate dataset. We can then train a new logistic regression classifier on this new data, again using 10-fold cross-validation, on the more limited size test set, as done in the previous experiments.

We also investigate the application of the BSP and BSRP to models trained as classifiers. Such models are already trained to be discriminative and so we can ask whether or not the unsupervised BSP penalties are a help or hindrance in such a situation. To that end we proceed in a similar manner to the autoencoders but training a deep convolutional classifier instead.

5.4 Results and Discussion

If controlling the importance of individual features via the singular values of the network is expected to improve the discriminability of the model's learned representation then we should see that a linear model, trained on top of the embedding space, would produce higher accuracy than a linear model trained on an embedding space without a BSP style loss augmentation. We demonstrate that this is the case

for task transfer in the source domain and we subsequently show that the embedding learned a version of the BSP that is also better when the input domain data is changed.



(a) MNIST (b) SVHN (c) FashionMNIST

(d) MNIST,max-norm (e) SVHN, max-norm (f) FashionMNIST,max-norm

Figure 5.1: Here we present an analysis of the embedding space learned by an autoencoder using each of the penalties compared with none. The left column is run on MNIST, the middle column is run on SVHN and the right column is run on FashionMNIST.The top row shows the singular values of the autoencoders embedding space on the test set, the bottom row shows the max normalised singular values. For the associated accuracy values see table 5.1

5.4.1 Evaluating improved discriminability in the source domain

We observe that for the autoencoder models trained with our BSRP penalty the learned image embedding is much more versatile and more easily able to discriminate between samples with a very simple model. This serves to confirm the utility of the idea of controlling the singular values as part of the loss.

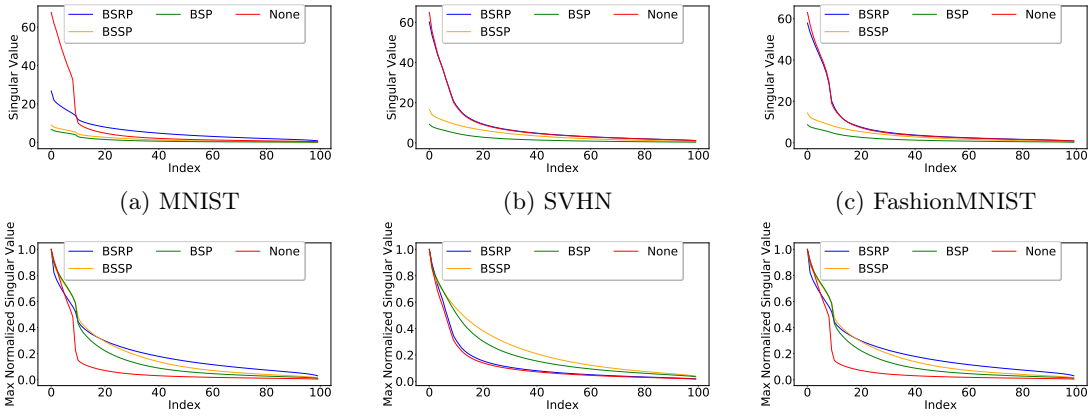
We can also observe in figure 5.1 that the max normalised singular values of the standard BSP penalty and BSPSum penalty are close to that of not applying the penalty at all. This is, in part, due to choosing a low weight for the penalty in this case as higher weights tended to produce worse classification results in the hyper-parameter tuning phase. Small weights for the BSP, in the autoencoder case, were chosen because when inspecting the un-normalised singular values for many larger weight values the singular values were pulled very close to zero and that appears to be detrimental even if the max-normalised values appear to produce a gradual decrease. It is also the case that the penalty doesn't optimise a specific classification or reconstruction goal and so should not dominate the optimization process.

We also observe the effect the penalties have on training a dicriminative convolutional classifer. Here differences are less pronounced and the original BSP penalty

	BSRP	BSSP	BSP	No Penalty
MNIST AE	$\mathbf{0.945 \pm 0.002}$	0.931 ± 0.001	0.929 ± 0.002	0.931 ± 0.001
FashionMNIST AE	$\mathbf{0.850 \pm 0.000}$	0.847 ± 0.001	0.846 ± 0.001	0.846 ± 0.001
SVHN AE	$\mathbf{0.743 \pm 0.005}$	0.562 ± 0.009	0.562 ± 0.007	0.566 ± 0.005
MNIST CLF	0.981 ± 0.020	0.994 ± 0.000	$\mathbf{0.994 \pm 0.000}$	0.992 ± 0.000
FashionMNIST CLF	0.912 ± 0.002	0.919 ± 0.002	$\mathbf{0.922 \pm 0.001}$	0.913 ± 0.002
SVHN CLF	0.925 ± 0.002	0.933 ± 0.001	$\mathbf{0.933 \pm 0.001}$	0.926 ± 0.001

Table 5.1: Here we show the average accuracy achieved by 10-fold cross-validation training of a logistic regression classifier trained on the representations learned by an autoencoder or convolutional classifier. Experiments using our proposed penalties are shaded in grey.

appears to provide a small advantage. Looking at the real values of the singular values the other penalties appear not to pull the top singular values down significantly. This is potentially due to the fewer epochs needed to train the classifiers on these problems. What is evident is that controlling the singular values does create embeddings that are more discriminable. We can see from the max-normalised graphs, in figure: 5.2, that regularising the importance of the learned features is indeed beneficial, however, it is also critical that the top values not be significantly larger than the rest of the singular values; for example see the MNIST classifier in figure 5.2, which puts a lot of weight on the first features.



(a) MNIST (b) SVHN (c) FashionMNIST

(d) MNIST,max-norm (e) SVHN, max-norm (f) FashionMNIST,max-norm

Figure 5.2: Here we present an analysis of the embedding space learned by a classifier using each of the penalties compared with none. The left column is run on MNIST, the middle column is run on SVHN and the right column is run on FashionMNIST.The top row shows the singular values of the classifiers embedding space on the test set, the bottom row shows the max normalised singular values. For the associated accuracy values see table 5.1

5.4.2 Evaluating Discriminability Across Domains

We can also analyse the usefulness of the proposed penalties in their effect on the networks ability to generalize representations across domains. In this case we take

	BSRP	BSSP	BSP	No Penalty
MNIST Model:				
FashionMNIST AE	$\mathbf{0.843 \pm 0.002}$	0.837 ± 0.001	0.833 ± 0.001	0.837 ± 0.001
SVHN AE	$\mathbf{0.576 \pm 0.016}$	0.494 ± 0.010	0.462 ± 0.008	0.490 ± 0.009
FashionMNIST CLF	0.906 ± 0.002	0.919 ± 0.002	$\mathbf{0.922 \pm 0.001}$	0.913 ± 0.002
SVHN CLF	0.922 ± 0.002	0.933 ± 0.001	$\mathbf{0.933 \pm 0.001}$	0.926 ± 0.001
FashionMNIST Model:				
MNIST AE	$\mathbf{0.934 \pm 0.001}$	0.924 ± 0.001	0.915 ± 0.001	0.929 ± 0.001
SVHN AE	$\mathbf{0.560 \pm 0.003}$	0.483 ± 0.004	0.412 ± 0.002	0.555 ± 0.006
MNIST CLF	0.992 ± 0.000	0.994 ± 0.000	$\mathbf{0.994 \pm 0.000}$	0.992 ± 0.000
SVHN CLF	0.925 ± 0.002	0.933 ± 0.001	$\mathbf{0.933 \pm 0.001}$	0.926 ± 0.001
SVHN Model:				
MNIST AE	$\mathbf{0.958 \pm 0.000}$	0.940 ± 0.001	0.940 ± 0.001	0.940 ± 0.001
FashionMNIST AE	$\mathbf{0.852 \pm 0.002}$	0.847 ± 0.001	0.846 ± 0.003	0.846 ± 0.002
MNIST CLF	0.992 ± 0.000	0.994 ± 0.000	$\mathbf{0.994 \pm 0.000}$	0.992 ± 0.000
FashionMNIST CLF	0.912 ± 0.002	0.919 ± 0.002	$\mathbf{0.922 \pm 0.001}$	0.913 ± 0.002

Table 5.2: Here we show the average accuracy achieved by 10-fold cross-validation training of a logistic regression classifier trained on the transferred representations acquired by applying autoencoder or convolutional classifier to the new dataset. Experiments using our proposed penalties are shaded in grey.

a model trained on one task, such as MNIST, and apply the model to another task, such as SVHN, to generate embeddings. If we have learned a representation that generalizes well across domains we expect, in this case, that a classifier trained on those embeddings would perform better as the embeddings maintain their data separability across domains.

To investigate this we can take the models trained in the previous section and apply them to a new dataset. In the same way as before we then train a logistic regression classifier on a 10-fold cross validation of the new test dataset. Our experiments again show that by raising the relative importance of the features in the embedding we achieve improved discriminability when using the penalty.

In the case of domain transfer as with the in domain experiments we can see that the generative autoencoder models perform best when trained with our proposed BSRP penalty, see table 5.2, however the discriminative classifier models perform best with the original BSP penalty. The BSRP penalty also generates the most useful representations, when using an autoencoder. This behaviour for both the in-domain and transfer case is perhaps reasonably explained by the nature of the two distinct training paradigms. Classifier models naturally learn discriminative features and the BSP helps simply by preventing one or a few features from dominating. In contrast the generative model does not learn discriminative features and so the BSRP which can pull low importance features up allows a more discriminative representation to be learned.

	No Penalty	BSP	BSSP	BSRP
MNIST AE	0.566 ± 0.020	0.569 ± 0.007	$\mathbf{0.585 \pm 0.008}$	0.528 ± 0.037
FashionMNIST AE	0.598 ± 0.018	0.629 ± 0.025	$\mathbf{0.633 \pm 0.031}$	0.617 ± 0.049
SVHN AE	0.157 ± 0.004	0.158 ± 0.003	0.156 ± 0.005	$\mathbf{0.207 \pm 0.008}$
MNIST CLF	0.99236 ± 0.000	$\mathbf{0.99376 \pm 0.000}$	0.99326 ± 0.000	0.90328 ± 0.178
FashionMNIST CLF	0.90770 ± 0.003	$\mathbf{0.91874 \pm 0.001}$	0.90370 ± 0.017	0.90880 ± 0.002
SVHN CLF	$\mathbf{0.91117 \pm 0.002}$	0.90491 ± 0.026	0.88311 ± 0.040	0.90848 ± 0.001

Table 5.3: Here we show the average purity of clusters generated by KMeans clustering on the learned embeddings of each of our models. Experiments using our proposed penalties are shaded in grey.

	No Penalty	BSP	BSSP	BSRP
MNIST AE	0.566 ± 0.020	0.569 ± 0.007	$\mathbf{0.585 \pm 0.008}$	0.528 ± 0.037
FashionMNIST AE	$\mathbf{0.586 \pm 0.017}$	0.577 ± 0.011	0.574 ± 0.009	0.5732 ± 0.007
SVHN AE	0.176 ± 0.007	0.173 ± 0.007	$\mathbf{0.177 \pm 0.009}$	0.163 ± 0.004
MNIST CLF	0.99236 ± 0.000	$\mathbf{0.99376 \pm 0.000}$	0.99326 ± 0.000	0.90328 ± 0.178
FashionMNIST CLF	0.90770 ± 0.003	$\mathbf{0.91874 \pm 0.001}$	0.90370 ± 0.017	0.91306 ± 0.002
SVHN CLF	$\mathbf{0.91117 \pm 0.002}$	0.90491 ± 0.026	0.88311 ± 0.040	0.90993 ± 0.002

Table 5.4: The Purity of the MNIST model clusters transferred to each of the other datasets as labeled. Experiments using our proposed penalties are shaded in grey.

5.4.3 Embedding

Learning a better more discriminable representation of the data is a difficult to define metric. We have already shown that the BSRP provides benefit over not applying the penalty and is more consistent than the alternatives when training an autoencoder. The case for classifiers indicates that the penalty is indeed useful however the results are small. We now look at how well the KMeans clustering algorithm performs on our learned embedding space. We can do this by measuring the purity of the clusters.

In the case of vanilla autoencoders this cluster purity is fairly weak, however, some interesting results do arise. In table 5.3 we show the average purity of clusters generated by KMeans clustering of the embedding space created by the autoencoder model trained with one of the Penalty variations and no penalty. It is clear that applying one of the BSP creates representations that are better able to be clustered with one of the proposed BSPSum and BSRP always performing best. If instead we look at the cluster purity generated by the classifiers we see that, as expected, it is much closer to 1 for all of the models. We also observe here that the original BSP performs best overall.

The picture that emerges for within domain training is that batch spectral penalization is a tool that is useful in creating representations that generalise better.

The effect is present when training discriminative or generative models, however, it is more pronounced in the generative case.

The transfer case appears more complicated from the cluster purity perspective. The transferred network can benefit from having been trained with a version of the BSP, as seen in transfer from MNIST in table 5.4 and transfer from FashionMNIST in table 5.5. This, however, appears to contrast with our previous results with the linear classifier.

	No Penalty	BSP	BSSP	BSRP
MNIST AE	$\mathbf{0.595 \pm 0.002}$	0.562 ± 0.032	0.570 ± 0.023	0.536 ± 0.007
FashionMNIST AE	0.598 ± 0.018	0.629 ± 0.025	$\mathbf{0.633 \pm 0.031}$	0.617 ± 0.049
SVHN AE	$\mathbf{0.188 \pm 0.005}$	0.161 ± 0.001	0.175 ± 0.003	0.172 ± 0.003
MNIST CLF	0.99236 ± 0.000	$\mathbf{0.99376 \pm 0.000}$	0.99326 ± 0.000	0.99198 ± 0.000
FashionMNIST CLF	0.90770 ± 0.003	$\mathbf{0.91874 \pm 0.001}$	0.90370 ± 0.017	0.90880 ± 0.002
SVHN CLF	$\mathbf{0.91117 \pm 0.002}$	0.90491 ± 0.026	0.88311 ± 0.040	0.90848 ± 0.001

Table 5.5: The Purity of the FashionMNIST model clusters transferred to each of the other datasets as labeled. Experiments using our proposed penalties are shaded in grey.

	No Penalty	BSP	BSSP	BSRP
MNIST AE	0.609 ± 0.014	0.601 ± 0.018	$\mathbf{0.625 \pm 0.009}$	0.584 ± 0.020
FashionMNIST AE	0.607 ± 0.013	$\mathbf{0.613 \pm 0.004}$	0.599 ± 0.017	0.583 ± 0.009
SVHN AE	0.157 ± 0.004	0.158 ± 0.003	0.156 ± 0.005	$\mathbf{0.207 \pm 0.008}$
MNIST CLF	0.99236 ± 0.000	$\mathbf{0.99376 \pm 0.000}$	0.99326 ± 0.000	0.99198 ± 0.000
FashionMNIST CLF	0.90770 ± 0.003	$\mathbf{0.91874 \pm 0.001}$	0.90370 ± 0.017	0.90880 ± 0.002
SVHN CLF	$\mathbf{0.91117 \pm 0.002}$	0.90491 ± 0.026	0.88311 ± 0.040	0.90848 ± 0.001

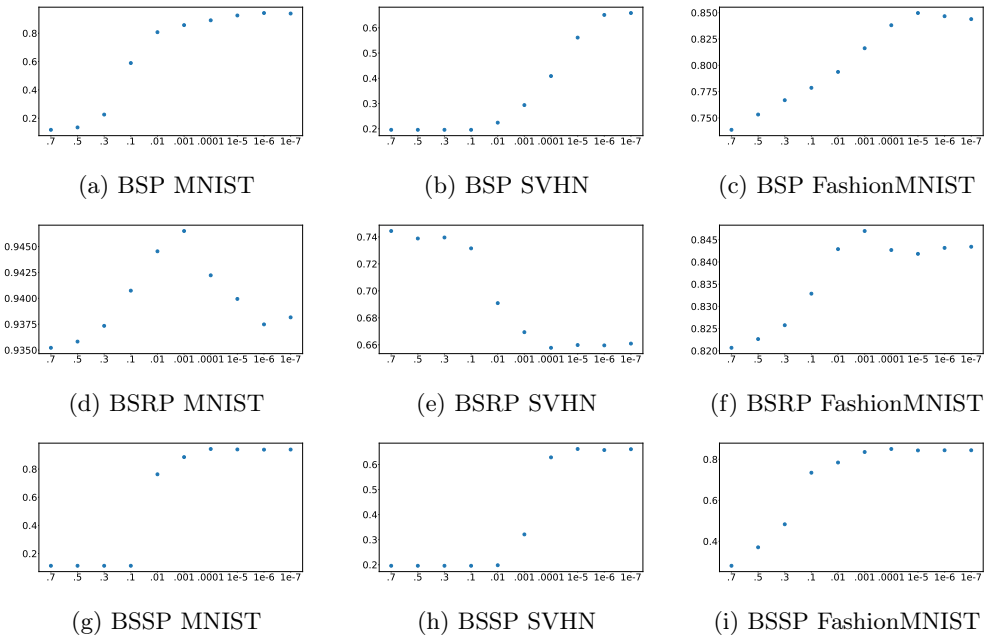
Table 5.6: The Purity of the SVHN model clusters transferred to each of the other datasets as labeled. Experiments using our proposed penalties are shaded in grey.

5.4.4 On the Impact of Batch and penalty weight

Further to our investigations we explore the role of the penalty weight and batch size. we show training across multiple weights for the BSP on both autoencoders and classifiers. We can see that generally a small penalty produces the best response. Since the penalty limits the importance of each learned concept, too strong a penalty will learn an uninformative representation. A small penalty, by contrast allows for important concepts to be influential, but limits how important, allowing the penalty secondary features to also influence the classification.

The benefit of a small penalty is more clear in figure 5.4 which shows the effect of the penalty in relation to classifiers. We note that differences at the lower level of application are relatively small.

Investigating the effect of batch size produces mixed results. For autoencoders, as

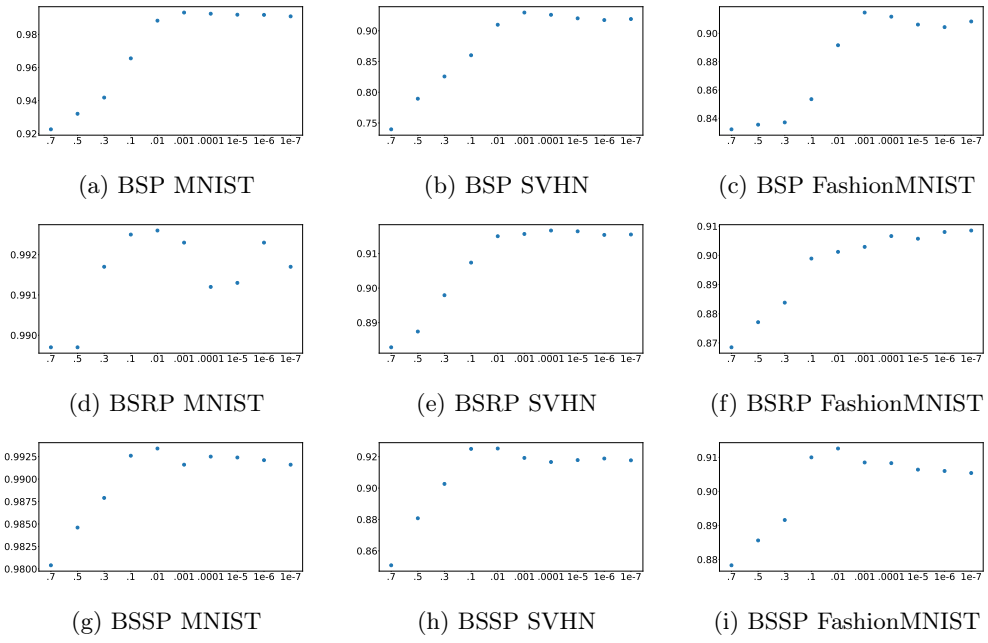


(a) BSP MNIST (b) BSP SVHN (c) BSP FashionMNIST

(d) BSRP MNIST (e) BSRP SVHN (f) BSRP FashionMNIST

(g) BSSP MNIST (h) BSSP SVHN (i) BSSP FashionMNIST

Figure 5.3: We examine the effect of training with varying penalty weights across the source tasks for autoencoders. We show results for each of the BSP,BSRP and BSSP across MNIST, SVHN and FashionMNIST. We note that the ratio penalty demonstrates better performance with a higher penalty value, however a penalty around 0.0001 is usually a good choice

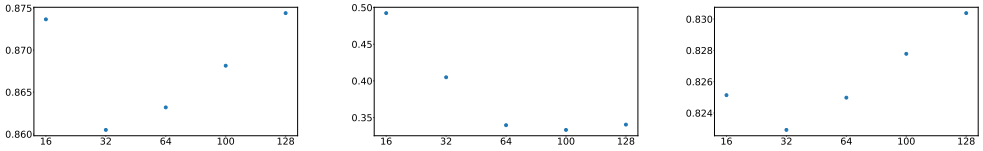


(a) BSP MNIST (b) BSP SVHN (c) BSP FashionMNIST

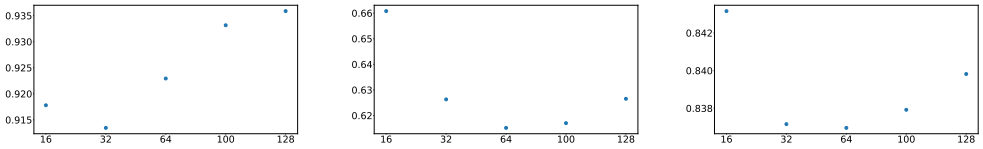
(d) BSRP MNIST (e) BSRP SVHN (f) BSRP FashionMNIST

(g) BSSP MNIST (h) BSSP SVHN (i) BSSP FashionMNIST

Figure 5.4: We examine the effect of training with varying penalty weights across the source tasks for classifiers. We show results for each of the BSP,BSRP and BSSP across MNIST, SVHN and FashionMNIST. We note that a penalty around 0.0001 is usually a good choice

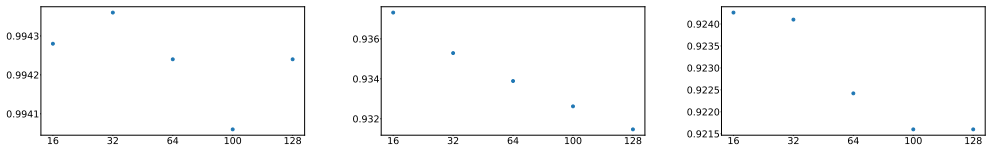


(a) BSP MNIST (b) BSP SVHN (c) BSP FashionMNIST

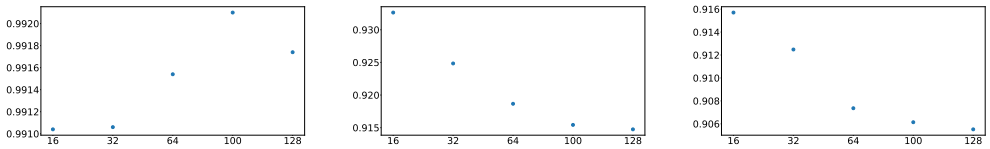


(d) BSRP MNIST (e) BSRP SVHN (f) BSRP FashionMNIST

Figure 5.5: We examine the effect of training with varying batch sizes across the source tasksusing AutoEncoders. We show results for each of the BSP,BSRP and BSSP across MNIST, SVHN and FashionMNIST.



(a) BSP MNIST (b) BSP SVHN (c) BSP FashionMNIST



(d) BSRP MNIST (e) BSRP SVHN (f) BSRP FashionMNIST

Figure 5.6: We examine the effect of training with varying batch sizes across the source tasks using classifiers. We show results for each of the BSP,BSRP and BSSP across MNIST, SVHN and FashionMNIST.

in figure 5.6, larger batches appear to be beneficial, except in the case of SVHN. The variations in results are largely small, however it appears that both large and small batches can produces improvements. When examining the classifiers performance in figure 5.6, the results are more consistent and appear to demonstrate a preference for small batches. It must again be noted however that the real difference in accuracy is very small; often by tenths of a percent.

5.5 Conclusion

The Batch Spectral Penalty has been shown to improve the adaptation of neural network models in the domain adaptation context and provides a lens through which to view the question: 'What makes an embedding transferable?'. This question is

complicated in the domain adaptation case as fooling the discriminator through being more transferable was shown to be at odds with the task of discriminating in the target task.

In this chapter we explored the application of the BSP and BSRP analysing how the penalty improved the discriminative properties of a learned representation through a series of experiments with autoencoders. We show that, for generative models, the BSRP provides the best boost to discriminative performance, for within domain task transfer and that the benefits persist if the domain changes. We also show that the standard BSP most improves the discriminability of the learned representation. This corroborates the research of [11] in the single task case where we may wish to perform some type of pre-training

From our work and previous work [11] the BSRP has merit when applied to models that are not designed solely as discriminators. For small batches it is easy to compute and include in a model without adding a large time cost to training and creates more separable representations. Future work would explore the application of the penalty to fine-tuning larger pre-trained models and would further investigate the less favourable effects of applying the BSRP penalty to discriminative models.

Chapter 6

Singular Value Entropy: Improving the Batch Spectral Penalty

6.1 Introduction

Deep Learning models have been shown to learn extremely versatile representations that can be used for many tasks, however, they are not always optimal. Large powerful models such as AlexNet[12], ResNet50 [24] and DenseNet[97] are commonly used as network initialisations and image encoders for many subsequent tasks. This direct transfer is not perfect however, when faced with out-of-distribution data these models fail to generalise. Overcoming this difficulty is the aim of domain adaptation.

Domain Adaptation is a field focused on overcoming the discrepancy of applying a model trained on one data distribution to, usually unlabeled, samples from a new target data distribution. It offers potential for overcoming the sim-to-real disparity when moving from simulated training data to applications on real data. It can also help to overcome disparities in image capturing techniques such as when different cameras are used; this is potentially useful in the medical domain where different hospitals may have unlabeled data generated by a machine produced by a different manufacturer. A prominent set of domain adaptation models focuses on aligning the embedding of data from the same class structure but sampled from different data distributions.

Adversarial learning approaches, where a shared network embedding space is learned using a domain discriminator which separates source and target samples, have performed well. It has recently been shown, however, that the domain discriminator unduly restricts the learned embedding, with the embedding relying heavily on one or two important features, as evidenced by the singular value decomposition of the embedding space. A penalty, Batch Spectral Penalization (BSP) [11], was proposed to alleviate this issue, with significant improvements observed.

In this chapter we offer an alternative interpretation of the effect of the BSP. We combine insights from recent work in deep metric learning [98]; which showed a negative correlation between the entropy of singular value decomposition of the embedding space of metric learning models and their accuracy. This leads to overly compressed representations which do not generalise well. A similar problem exists for adversarial learning methods such GANs where a lack of diversity and mode collapse are significant problems. We link this observation with the effect of the BSP and propose an improved version of the penalty, taking our insight as to the effect of the entropy of the singular value distribution into account.

Our contributions are to provide a new method of interpreting domain adaptation models, offering new insight into the effect of the recent BSP, and to introduce a new penalty term utilising our new interpretation and incorporating the proposals of [11]. We show the effect that the entropy of the singular values of the model features has when learning a general representation, by proposing a penalty term to directly regularise the entropy of the singular value distribution. We provide a new perspective on the effect of the penalty term proposed in [11] highlighting how the penalty term also affects the entropy of the singular value distribution. We link our work with recent deep metric learning research [98] showing a similar negative correlation between entropy and accuracy. Finally, we propose a penalty combining our proposed entropy penalty with the penalty of [11], dubbed Batch Spectral Entropy Penalization (BSEP), and demonstrate improved results across a number of domain adaptation benchmarks.

The contributions of this chapter are summarised as follows:

A new understanding of the role of entropy in domain adaptation

An alternative interpretation of the Batch Spectral Penalty

The Batch Spectral Entropy Penalty (BSEP) is proposed as a new domain adaptation penalty

6.2 Background

Domain adaptation is a set of approaches with the goal of reducing domain shift between a source and target domain, often with the target domain samples being unlabeled. A variety of approaches exist to tackle this domain shift such as learning feature representations that are invariant over the domains or weighting samples according to their relatedness to the target domain. A problem with many applications from helping to bridge the sim-2-real gap to bridging between different imaging machines and manufacturers in a medical setting where labelled training data is of limited availability.

Adversarial Discriminative Domain Adaptation [92], focuses on models that learn feature representations that obfuscate the sampling domain through confusing a domain discriminator trained to classify the source domain of the embedding. These approaches have a number of instantiations and have been very successful.

Recent work by [11], however, showed how learning to fool the discriminator creates representations that are over-reliant on a single feature increasing the transferability of the representation at the cost of the discriminability. The Batch Spectral Penalty (BSP) [11] is introduced in the context of Adversarial domain adaptation [87, 92, 72]. The BSP paper focuses on the joint learning and adapting approach of [99] however they also show strong adaptation results with the separated train source then adapt approach of ADDA [92]. The penalty thus appears promising for domain adaptation and another paper [100] focusing on the use of the penalty as a regulariser when fine-tuning large pre-trained networks also shows promise.

Work in deep metric learning [98] has recently shown a similar phenomenon to that elucidated by the BSP[11] and BSRP. They show how the less uniform the singular value distribution is the better the generalization capability of the model. They [98] then go on to introduce a label switching method in order to induce more variation in the per class embeddings. Considering this observation and recalling the use of the BSP is to reduce the reliance of the representation on a single feature, especially in the context of domain adaptation, serves to further verify the utility of a BSP-based penalty when learning representations which may rely on singularly discriminant features. This observation is also corroborated by the recent use of the BSP in the context of few-shot learning [101].

The take-away from these works is that controlling the trade-off between transferability, e.g. all these objects are cats, and discriminability, e.g. this cat is brown and this cat is ginger, allows for better overall generalization, through building representations that maintain dominant features, representing inter-class variance, but not at the expense of maintaining features representing intra-class variance.

6.3 Method

The success of the low singular value entropy models in [98] is achieved indirectly through stochasticly inverting sample labels. In this work we propose a direct penalization of the batch-level singular value entropy.

Singular Value Decomposition is a versatile method for interpreting and understanding data. It is used in principle component analysis and for low-rank approximations generally, which can be used to visualise the data space. Recent work in deep metric learning has shown how the entropy of the singular values of the

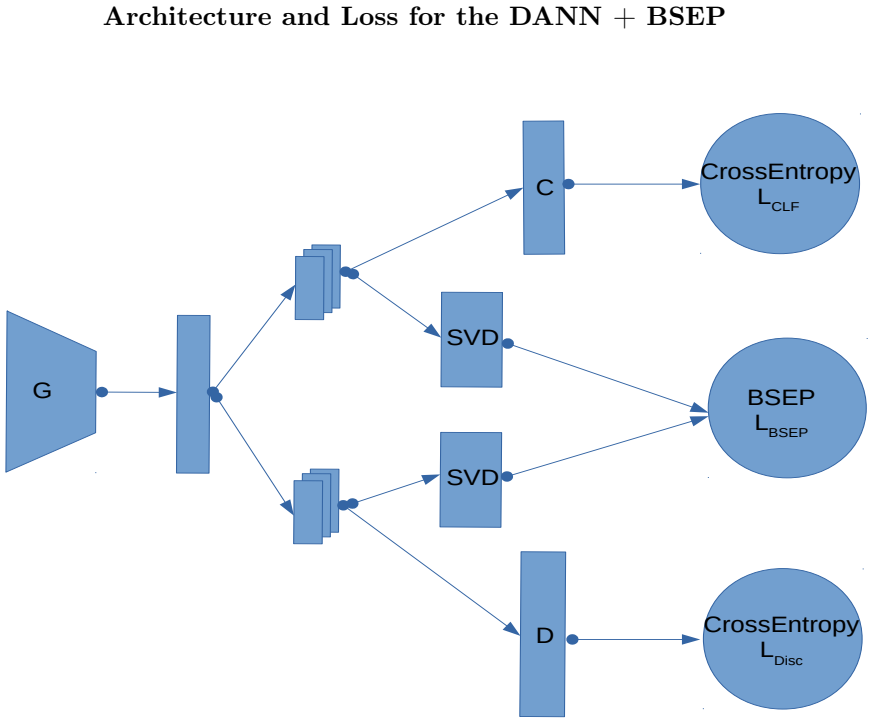


Figure 6.1: Architecture diagram of the Adversarial domain adaptation architecture including the BSEP penalty term

learned feature space is inversely correlated with the accuracy of metric learning models. This is increased entropy is then induced through a label-flipping training mechanism, however as demonstrated in [11] a penalty can be directly applied to the singular values. We can therefore directly induce the structure we desire on the singular value space.

Recall that for a given feature matrix $F = [f_1 ... f_b]$, where $f_k \in \mathbb{R}^N$ and b is the batch size for each mini-batch in the training procedure. F's singular value decomposition can be written:

$$F = U\Sigma V \tag{6.1}$$

with the singular values s represented by the diagonal of Σ.

Taking these singular values we can achieve a probability distribution by applying a softmax function. In line with the findings of [98] the aim is to minimise the entropy of this distribution. We can instantiate a new penalty using the KL-divergence between this new distribution and a uniform distribution U, which we will maximise during training.

$$L_{entropy} = D_{KL}\left(SoftMax(\Sigma), U\right) \tag{6.2}$$

We employ two versions of this term, given in equation 6.2. The first uses every singular value when calculating the entropy, the second penalty drops the first singular value and calculates the entropy of the remaining distribution. This is done because as noted in [11, 98] the top singular value tends to be significantly larger than the others and can distort the distribution. Our experiments demonstrate that both entropy penalties improve performance on the domain adaptation task but they also show that excluding the top singular value from the penalty has a small but consistent advantage.

6.3.1 The Batch Spectral Penalty

The Batch Spectral Penalty (BSP) recently proposed in [11] and described in chapter 5, as a penalty on the SVD of an adapted embedding, is briefly recapped here.

To design a penalty that maintains the discriminabilty of the learned embedding we can look to the singular values of the feature space. Intuitively a representation that has a sharp drop off in the importance of features, as evidenced by the singular values, relies heavily on one or two key features and so we may be less able to discriminate between similar data points. With this in mind we can think of a penalty that stops the top features from becoming too important by penalizing overly important features.

This can be done through penalising the top singular values

$$L_{bsp}(F) = \Sigma_{i=1}^{k}(\sigma_i^2) \tag{6.3}$$

where σ_i represents the i^{th} largest singular value. This gives us the source-only version of the BSP penalty proposed by [11]; it is normally applied to both the source and target. This penalty however can pull every value down in an unrestrained fashion.

6.3.2 Combining the BSP and Entropy: The BSEP

Relating the BSP approach to recent work in deep metric learning [98], which argues that a uniform distribution of singular values is inversely correlated with generali-

sation, we note that the BSP in essence decreases the entropy of the distribution as the less important features now become less uniform in reference to the top singular value.

However the singular value curves can still appear relatively uniform and so we propose a combination of the entropy and BSP penalties, which we dub the Batch Spectral Entropy Penalty (BSEP).

Our approach combines the two penalties in a stochastic manner. applying the Entropy penalty with probability β and the BSP with probability $1-\beta$ each of the penalties is also individually weighted.

$$BSEP(x) = \begin{cases} L_{Entropy}(x) & \text{with probability } \beta \\ BSP(x) & \text{with probability } 1-\beta \end{cases} \tag{6.4}$$

Where β in our experiments is 0.5. This penalty has the effect of controlling for the runaway importance of the top singular values as well as penalizing the lower end of the distribution for being too uniformly distributed.

Computational Complexity The proposed penalties do not significantly extend the computation needed to apply the original BSP and thus do incur extra computational overhead. The overhead of full SVD would be expensive, $\mathcal{O}(\min(m^2n, mn^2))$, however because we use the embedding dimension and generally have a small batch size, b, the overhead is generally light, $\mathcal{O}(b^2d)$, with only a small number of samples per batch being used to compute the penalty.

6.3.3 Domain Adaptation

Adversarial domain adaptation sets up training as a 2-player mini-max game fashioned in the style of Generative adversarial networks. The encoder model learns to encode target images such that a domain discriminator model is unable to correctly identify the true source domain of the embedding.

Thus we have a classification loss:

$$L_{clf}((x,y) \in D_{source}) = -\Sigma_i C(G(x_i)) \log(y_i) \tag{6.5}$$

$$L_{domain}(x \in D) = -\Sigma_i D(G(x_i)) \log(domain) \tag{6.6}$$

The training procedure is set up to optimize the mini-max expressions:

$$\min_{G,C} L_{clf} + \lambda L_{domain} + \rho BSEP \tag{6.7}$$

$$\max_{D} L_{domain} \tag{6.8}$$

We employ the adversarial domain adaptation method of [99] a relatively early and vanilla approach to the deep adversarial discriminative domain adaptation approach. The method trains a shared encoder network on source and target data. The model is jointly trained to accurately classify source data and the domain discriminator is trained to identify the originating domain of each embedding, a gradient reversal layer is used to flip the signal from discriminator to generator in order to induce the mini-max training.

6.3.4 Domain Adaptation Theory

[61] developed the domain adaptation theory and proposed a bound on the error,see equation 6.9, of domain adaptation utilising three components; the expected error on the source domain, the distance between source and target as a measure of domain discrepancy and the ideal joint hypothesis . λ represents the error of the ideal joint hypothesis, $h^* = \min_h \mathcal{E}_{\mathcal{S}}(h) + \mathcal{E}_{\mathcal{T}}(h)$, given in equation 6.10

Most recent domain adaptation papers focus on the domain alignment aspect of the theory, if the domains are better aligned they are more transferable. However the discriminability of the features in the shared space is also an important consideration.

The BSP penalty is motivated by the lack of focus on the discriminability of target embeddings, which are hindered through the adaptation process. The domain discriminator dominates the representation learning phase creating a transferable representation, which minimizes part of the transfer equation described by [61] and given in equations 6.9 and 6.10. The BSP is used to increase the discriminability of the target features.

$$\mathcal{E}_{\mathcal{T}}(h) \leq \mathcal{E}_{\mathcal{S}}(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}((S),(T)) + \lambda \tag{6.9}$$

$$\lambda = \mathcal{E}_{\mathcal{S}}\left(h^*\right) + \mathcal{E}_{\mathcal{T}}\left(h^*\right) \tag{6.10}$$

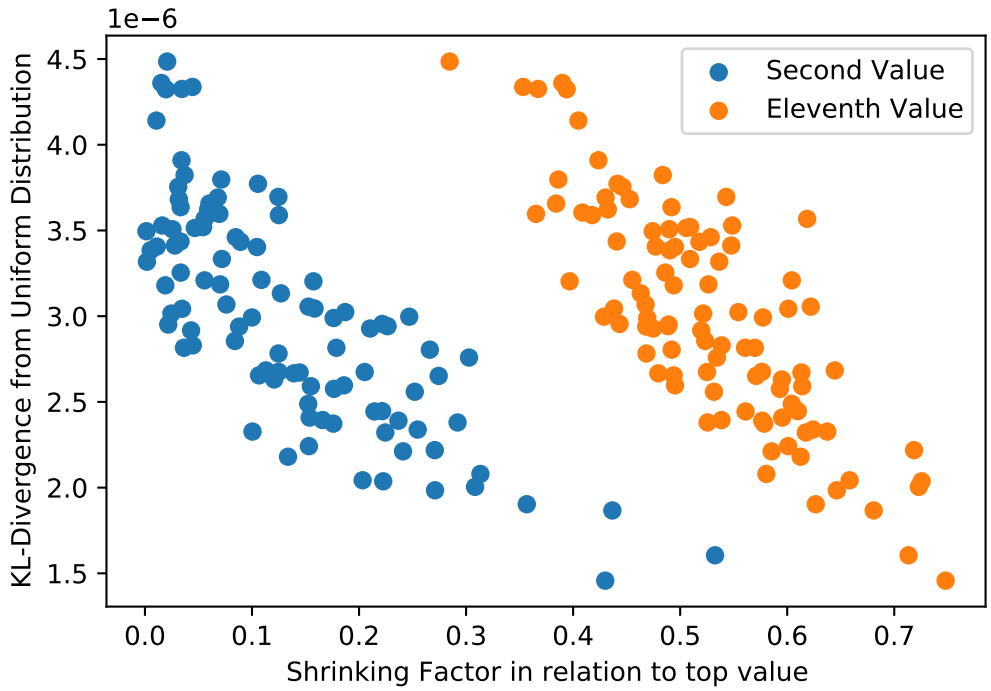


Figure 6.2: Here we show the negative correlation between KL-divergence from the uniform distribution and the size of the second and eleventh values in the given distribution relative to the maximum value. This serves to demonstrate that decreasing the top value, or raising the lower values can decrease the entropy of the distribution. Note this is for max-normalised distributions.

Our entropy penalty is used to demonstrate the correlation between lower entropy and higher accuracy. We show how discriminability of the learned target representation is improved when we apply our entropy penalty, see figure 6.6. The discriminability of the target representation is, however, just one aspect of the domain adaptation theory proposed in [61] and so we do not see the improvements in target embedding discriminabilty directly correlate with the adaptation accuracy, this is likely due to the continued dominance of the top singular values, and why we propose our combination BSEP penalty.

6.3.5 On Entropy

It may not be obvious that reducing the importance of the top singular value and thus making the the values appear more entropic does not in fact make the distribution more uniform. Indeed this effect is tied to normalizing the values and only applies to the max normalized values and not to the un-normalised distribution. This itself may prove interesting explore further as it suggests it is the relative importance of features that is indicative of performance even when the true values are

not uniform.

To demonstrate the relationship between entropy and the relative importance of features we draw samples from a variety of exponential distributions, these curves approximate the behaviour of adversarial adaptation models, with large top values declining quickly. We max-normalise the samples and compute the entropy. We then plot this entropy against the relative difference between the top value and either the second or eleventh largest value. We find a negative correlation, pearson coefficient -0.83 and -0.85 respectively, between these values and the KL-divergence from uniform; indicating that less entropic distributions have more features that are relatively more important, see figure6.2.

This observation aids in motivating our work, decreasing the entropy of the singular value distribution through raising the relative importance of features. We can also observe that this is the effect of the BSP [11], decreasing the relative distance between the top singular value and the remainder of the distribution.

6.4 Results and Discussion

Method	$A \rightarrow W$	$D \rightarrow W$	$W \rightarrow D$	$A \rightarrow D$	$D \rightarrow A$	$W \rightarrow A$	Average
ResNet50[24]	68.4 ± 0.2	96.7 ± 0.1	99.3 ± 0.1	68.9 ± 0.2	62.5 ± 0.3	60.7 ± 0.3	76.1
DAN[48]	80.5 ± 0.4	97.1 ± 0.2	99.6 ± 0.1	78.6 ± 0.2	63.6 ± 0.3	62.8 ± 0.2	80.4
DANN[99]	82.0 ± 0.4	96.9 ± 0.2	99.1 ± 0.1	79.7 ± 0.4	68.2 ± 0.4	67.4 ± 0.5	82.2
CDAN [102]	93.1 ± 0.2	98.2 ± 0.2	$\mathbf{100.0 \pm 0.0}$	89.8 ± 0.3	70.1 ± 0.4	68.0 ± 0.4	86.6
CTSN [103]	90.6 ± 0.3	$\mathbf{98.6 \pm 0.5}$	99.9 ± 0.1	89.3 ± 0.3	73.7 ± 0.4	74.1 ± 0.3	87.7
DANN+BSP[11]	93.2 ± 0.2	98.0 ± 0.2	$\mathbf{100.0 \pm 0.0}$	90.0 ± 0.4	71.9 ± 0.3	$\mathbf{73.0 \pm 0.3}$	87.7
CDAN +BSP[11]	93.3 ± 0.2	98.2 ± 0.2	100.00 ± 0.0	93.0 ± 0.2	73.6 ± 0.3	72.6 ± 0.3	88.5
DANN Entropy	92.8 ± 0.1	98.3 ± 0.1	$\mathbf{100.0 \pm 0.0}$	93.2 ± 0.1	69.9 ± 0.6	68.0 ± 0.6	87.0
DANN EntropyFull	91.7 ± 0.3	98.2 ± 0.1	$\mathbf{100.0 \pm 0.0}$	92.44 ± 0.2	71.85 ± 0.2	68.02 ± 0.4	87.0
DANN + BSEP (Proposed)	93.1 ± 0.1	98.2 ± 0.2	99.9 ± 0.1	92.4 ± 1.6	$\mathbf{74.3 \pm 0.2}$	69.7 ± 0.5	87.9
CDAN + BSEP(Proposed)	$\mathbf{93.6 \pm 0.6}$	98.2 ± 0.3	99.9 ± 0.1	$\mathbf{94.4 \pm 0.2}$	73.6 ± 0.5	72.4 ± 0.5	**88.7**

Table 6.1: Evaluating domain adaptation on the office-31 datasets

We show in this section how penalising the entropy of the singular values of each batch strongly resembles the effect of applying the batch spectral penalty with similar performance. We go on to show how combining the penalties improves performance, sometimes significantly.

Our experiments are conducted on standard domain adaptation dataset of office-31 and office-home. These datasets provide a challenge even for state of the art domain adaptation algorithms. Our approach uses ResNet-50[24] as a backbone architecture and we apply our penalty to the DANN[99] adversarial learning training paradigm and to the more recent Conditional Domain Adaptation Network (CDAN)

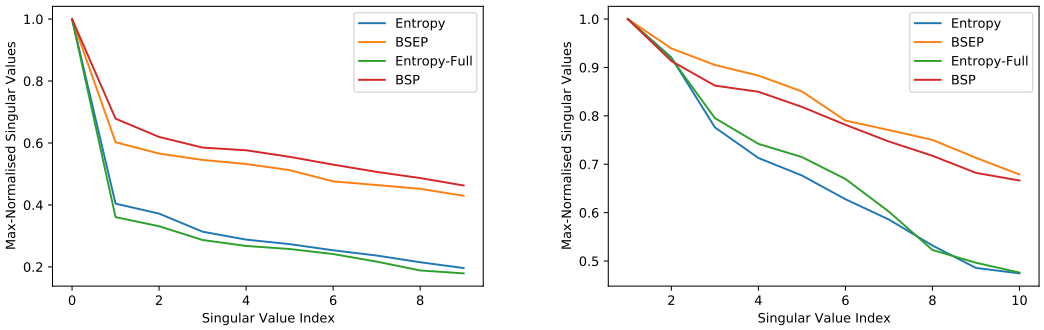


Figure 6.3: The singular value curves for adaptation in the office31 dataset from amazon to dslr images. We can see the outsize weight of the top singular value influences the relative importance of features even after applying the BSEP.

[102].

6.4.1 The effect of singular value entropy in adversarial domain adaptation

Our experiments with two variants of regulariser for the entropy of the singular value distribution indicate a significant improvement over the results of the baseline model. Indeed the benefits are comparable to applying the BSP. In some instances achieving top performance for any variant of the penalty applied to the DANN, and achieving a significant improvement in the average case. This serves to corroborate insights of [98] who negatively correlated the entropy of the singular value spectra with accuracy in the deep metric learning context.

By examining the singular values of our trained models we can see in figure 6.3 that the singular value spectra are still dominated by very important first features, which skew the entropy distribution. This was observed by [98] and their negative correlation involved removing the top singular value from calculations.

We can filter out the top singular value and calculate the entropy of the remainder of the singular value distribution and we find that the BSP reduces the entropy of the singular value distribution significantly more than directly penalizing the entropy. Even so when we look at table 6.1 we see the entropy penalty significantly improves over not including it and in many cases is comparable to the BSP and contemporary methods such as CDAN.

Our experiments and investigations thus confirm that the entropy of the singular value spectra is indicative of performance potential in domain adaptation and that

the entropy of the spectra is skewed by large values of the top singular value. These observations thus serve to motivate a combination of our proposed entropy penalty with that of the BSP.

6.4.2 The Batch Spectral Entropy Penalty

We introduced our batch spectral entropy penalty, BSEP, as a stochastic combination of the BSP and our proposed entropy penalization. When we examine the singular value curves we see a decreased emphasis on the top singular value, however it still dominates the distribution. When we max normalise and plot the singular values after removing the top value we see that the top 10 features of our model are given more weight than those of the model using the BSP alone.

Filtering out the top singular value we see the entropy of our proposed BSEP is the lowest of all methods, marginally lower than the BSP and significantly lower than the entropy only penalties. This serves to highlight the significance of the observation in [11] that the top singular value dominates the embedding and the utility of the BSP. Examining the results of adaptation we can see that the DANN model trained with the BSEP improves performance on average over using only the BSP or only the entropy penalty.

Our extended penalty serves to reinforce the idea that the entropy of the singular value distribution is an indicator of good performance and also provides state of the art results on difficult transfer tasks. In figure 6.4 we highlight the inverse correlation between entropy and accuracy. We can see that our proposed BSEP penalty reduces the entropy of the distribution. We can also see the the inverse correlation from the regression line which indicates falling performance with increasing entropy.

We note also that the representation compression effect observed in metric learning models [98] is also relevant to Adversarial Domain Adaptation models. GANs are known to suffer from a lack of diversity due to issues such as mode collapse. It is likely such a problem also affects domain adaptation and we show that controlling the entropy of the singular values improves performance.

Our proposed penalty and experiments also highlight the reasoning for the sensitivity analysis in [11] where they observe the best performance applying the BSP penalty to the top 2 or, as in their experiments, top 1. Minimising more of the singular values serves to make the distribution more uniform. Decreasing the entropy

Method	$A \to C$	$A \to P$	$A \to R$	$C \to A$	$C \to P$	$C \to R$	$P \to A$	$P \to C$	$P \to R$	$R \to A$	$R \to C$	$R \to P$	Average
ResNet50[24]	34.9	50.0	58.0	37.4	41.9	46.2	38.5	31.2	60.4	53.9	41.2	59.9	46.1
DAN[48]	43.6	57.0	67.9	45.8	56.5	60.4	44.0	43.6	67.7	63.1	51.5	74.3	56.3
DANN[99]	45.6	59.3	70.1	47.0	58.5	60.9	46.1	43.7	68.5	63.2	51.8	76.8	57.6
CDAN[102]	49.0	69.3	74.5	54.4	66.0	68.4	55.6	48.3	75.9	68.4	55.4	80.5	63.8
DANN(BSP)[11]	51.4	68.3	75.9	56.0	67.8	68.8	57.0	49.6	75.8	70.4	57.1	80.6	64.9
CDAN(BSP)[11]	52.0	68.6	76.1	58.0	70.3	70.2.8	58.6	50.2	77.6	72.2	59.3	81.9	66.3
DANN Entropy	49.7	67.1	74.3	53.8	63.7	64.6	54.6	49.7	75.7	69.4	55.1	81.3	63.3
DANN EntropyFull	49.7	67.1	74.1	53.8	63.7	63.8	54.3	48.7	75.4	69.6	54.5	80.9	63.0
DANN + BSEP(Proposed)	51.8	68.8	74.8	56.7	64.6	66.1	57.9	51.5	76.4	70.9	56.7	81.0	64.8
CDAN+BSEP(Proposed)	**52.51**	68.11	**76.23**	**60.25**	**70.63**	70.17	**59.51**	**52.74**	**78.46**	**72.45**	58.18	**82.69**	**66.83**

Table 6.2: Evaluating domain adaptation on the office-home datasets

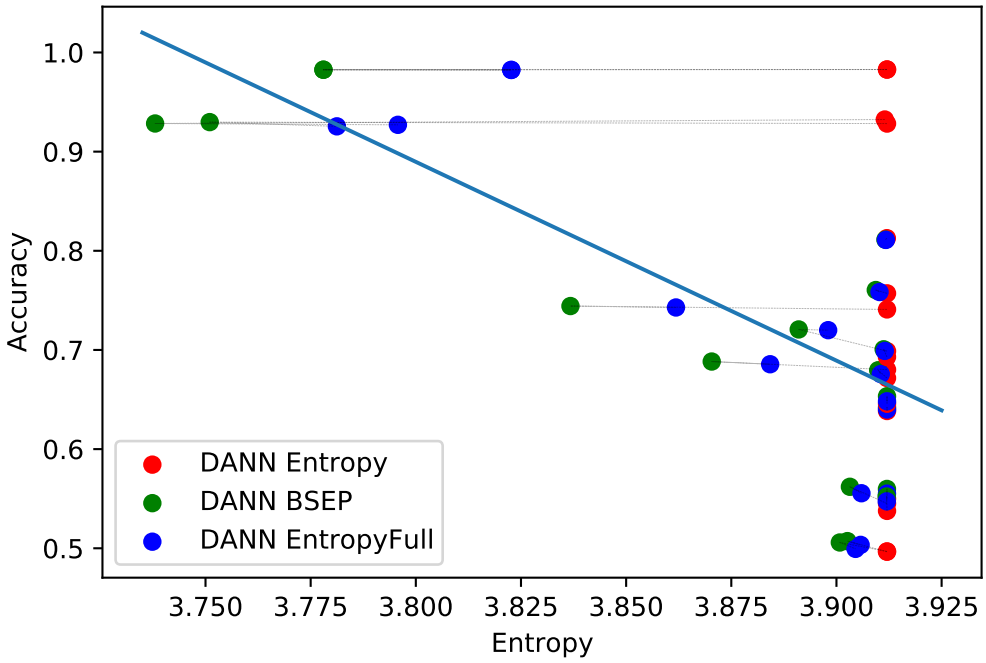


Figure 6.4: Plot of Accuracy against entropy for various penalty methods using DANN across a variety of transfer tasks. Models applied to the same task are linked together by grey lines. A regression line highlights the negative correlation between accuracy and entropy.

of the remainder of the distribution serves to alleviate this issue.

6.4.3 Ablation: Balancing application of Singular Value Entropy and BSP penalties

Here we provide an ablation study examining the effect of applying varying ratios of entropy minimisation and top singular value depression. We can see in figure 6.5 that combining both penalties in equal propoportion provides the best results. Controlling the entropy of the distribution alone is not enough to overcome the outsized influence of the top singular value controlled by the BSP. Similarly we can see the BSP alone can be improved upon by incorporating entropy penalization. This serves as further evidence that the relative importance of features is a strong

strong indicator of successful domain adaptation.

The benefit added through combining the BSP and entropy penalties is small, however it does not hurt and is not computationally expensive. Further our work elucidates a key insight to develop an intuition for the effect of the BSP. Through understanding that the relative importance of discerned concepts, as discovered by SVD, is what impacts adaptation performance. Our entropy penalty demonstrates this through comparing the $DANN$ and $DANNEntropy$ models in 6.2.

The functional difference between the entropy penalty and BSP is through the balancing mechanism. Both methods limit the impact of concepts unless they are relevant to either the adaptation or classification task. The entropy penalty, however, has more freedom to impact that importance and can also contribute to raising the importance of concepts. This potentially contributes to the small benefit observed when combining the BSP and Entropy penalties. Since less important concepts which would otherwise not have a strong pull now may be increased in importance allowing small features to make an impact and improving discriminability.

However these results also demonstrate the strength of the BSP penalty and the observation of the overbearing influence of the top singular value. The same observation can be made in table 6.1. Further on the size of the effect it is apparent that stronger models are more difficult to improve upon; the benefit of the BSP to the DANN model is more than twice as much as the benefit to the CDAN model.

6.4.4 Ablation: Improved discriminability of the learned target embedding

We provide an evaluation of the improved discriminability of the learned embedding space. We train a logistic regression model using 5-stratified train-test splits of our training data. The encoder of the adapted model is used to generate data embeddings which we use to train our logistic-regression model.

With this experiment we highlight the improved discriminability of the target representation after adaptation using our entropy and BSEP penalties. This further demonstrates how the entropy of the singular value distribution is negatively correlated with model performance.

The results of our experiments serve to demonstrate the impact of the relative importance of features learned. We show how the BSP can be interpreted as a

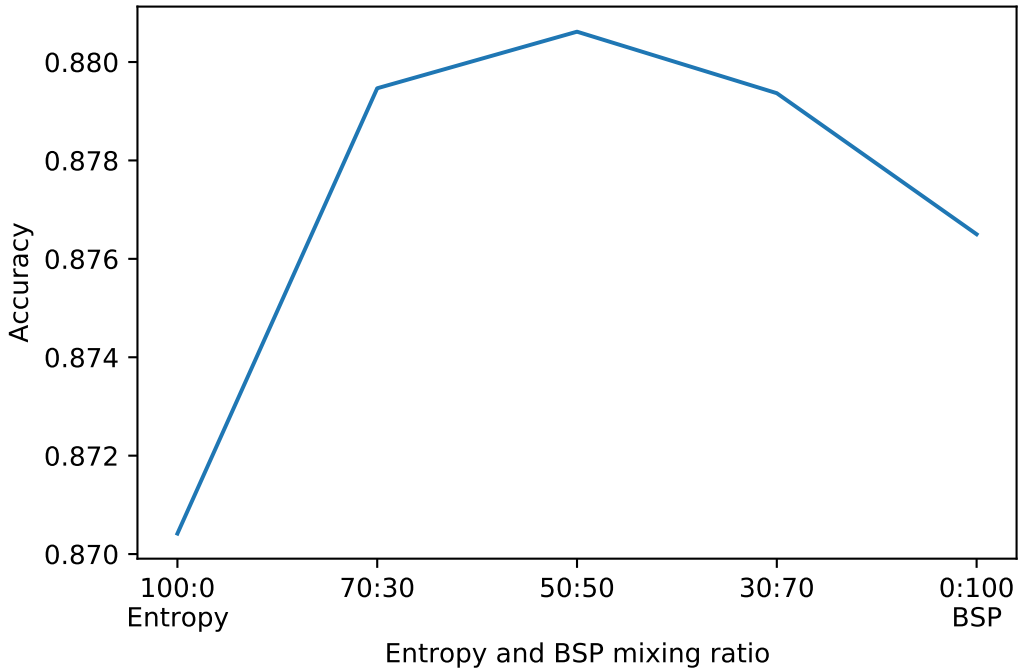


Figure 6.5: We show here the effect of varying the ratio of inclusion of the entropy penalty and the BSP. We apply various ratios across the set of Office31 adaptation tasks and average the results. What we observe is an equal mix is the most effective with performance falling off on either side using entropy only or BSP only.

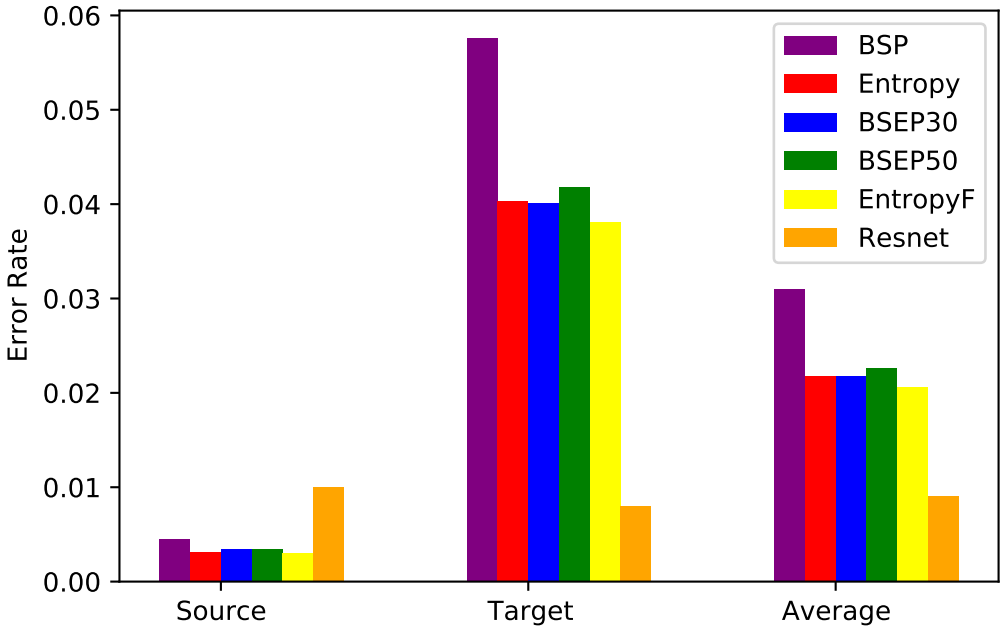


Figure 6.6: Evaluating the ideal joint hypothesis of the embedding space for various DANN models compared with ResNet50. A fixed encoder network is used and source and target labels are used to train a logistic regression classifier. We report the error rate, averaged across the office-31 transfer task models.

method of increasing the relative importance of features. We further show how controlling the relative importance of features through minimising the entropy of the singular value distribution leads to significant performance benefits over baseline methods and is competitive with more advanced methods. Finally we show that the entropy penalty is still dominated by the top singular value and that by applying the BSP in tandem with the entropy penalty, dubbed BSEP, we can achieve state of the art performance on domain adaptation tasks.

Our Penalty also benefits from its simplicity and lack of substantial overhead in terms of extra computation time needed. Integrating the penalty amounts to a partial SVD call and the application of a call to softmax. This serves to make our penalty easy to integrate into any adversarial adaptation context with very little additional cost for significant improvements over baseline algorithms that do not include the penalty.

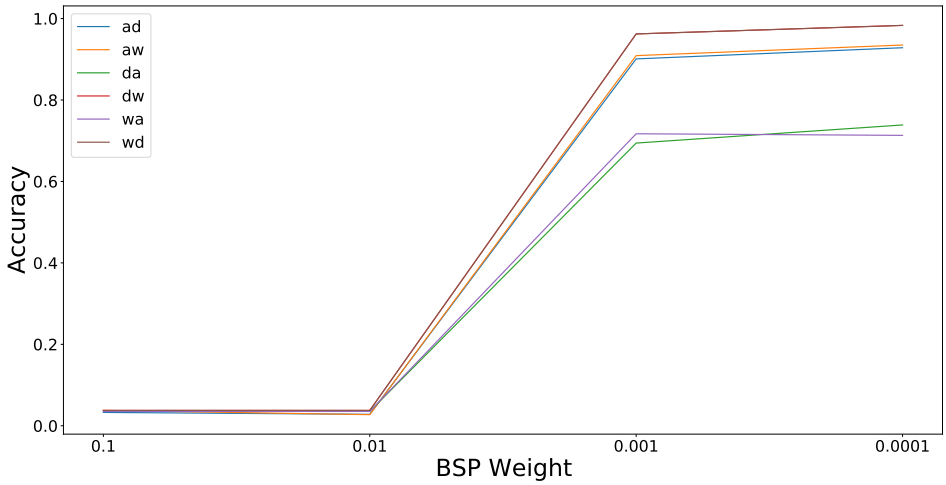


Figure 6.7: An analysis of the effect of varying the weight of the BSP penalty for each of the Office-31 tasks.

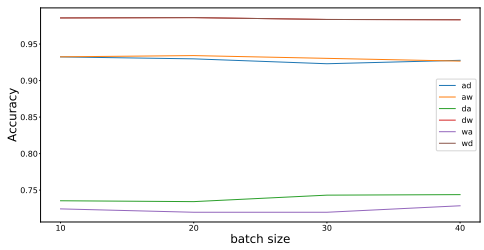
6.4.5 Ablation: Penalty weights and Batches

We further investigate the effect the penalty weight has on the adaptation. When mixing the penalty with the main task we can see the relative importance does have an impact. Too strong a weight inhibits learning, This is a fairly natural observation, A large penalty is too restrictive and makes it such that a model does not discover useful discriminative features. Similar to the previous chapter too large a weight has a negative effect on learning. A weight of 0.0001 produces strong results.

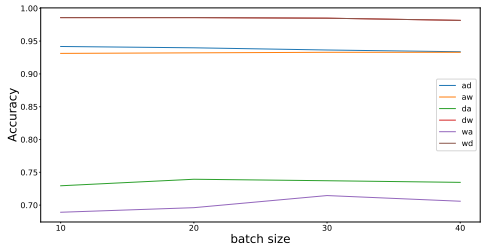
Since the penalty works on batch-wise SVD of the data we also examine the the impact of the batch size on the results. Interestingly the batch size does not appear to have a large impact on the results. We show that batch size has a very limited impact across the range range of penalty options. In figure 6.8 we can observe a slight benefit to utilising larger batches when the model is less strong. Strong models however appear to benefit from smaller batches. This is likely related to fact the original BSP has the most impact when penalising only the top singular value. It is likely that the batch size is also less important as training works through fine-tuning a pre-trained ResNet-50 model and the data-sets are relatively small.

6.5 Related Work

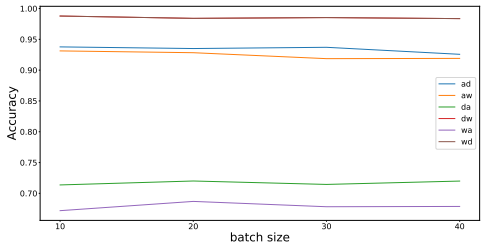
Deep neural networks learn representations that are transferable [29], however domain shift between the training and target data remains a problem. Many problems,



(a) BSP



(b) BSEP



(c) Entropy

Figure 6.8: We show the impact of batch size on solving the Office-31 task with either the BSP, BSEP or Entropy penalties. The impact of batch size on the results appears to be largely negligible.

such as the sim2real transfer problem for robotics applications to general classifiers trained on photographs and being applied to paintings, would benefit from methods that improve over direct transfer.

The aim and challenge of domain adaptation is to reduce domain shift to transfer learning from one domain to another. A number of approaches exist, mostly focused on adapting an internal network embedding to learn a shared subspace. This is a longstanding problem. More recent work has developed solutions based on deep neural networks and extended the application to the problem of adapting to domains that only partially overlap.

There are a variety of approaches to achieving adaptation. One set of approaches implements an adaptation layer between one or more of the deeper networks layers minimizing a divergence measure. Measures such as maximum mean discrepancy (MMD) and contrastive domain discrepancy are common. The DAN model developed by [48] uses MMD. A related set use adaptive normalization layers to overcome the domain discrepancy.

Further to these another set of approaches integrates a domain discriminator into the architecture [92, 99]. In this formulation a feature encoder network is trained against a domain discriminator which is trying to classify source and target samples, this can be done with GAN [15] style training or using a gradient reversal layer[87].

Such approaches are related to image to image translation schemes which often include a similar domain discriminator mechanism, though this mechanism may be applied to the image space rather than the embedding space. Through translating target images to source images a trained source classifier can then be applied to the transferred images. Of course combinations of such approaches are possible such as Cycada , which learns domain to domain translation utilising a cyclegan [45] architecture and also applies ADDA[92] to the embedding space for further alignment.

Some models, such as the RTN [49] an MMD based approach utilising residual modules for adaptation, have experimented with entropy using it to penalize the predicted target class distribution. Other work [104] has extended the DANN framework by forcing the discriminator to also act as class classifier. They present results using the AlexNet [12] architecture .

A recent method [103] using the MMD approach to learn shared features, utilises classifier entropy to distinguish easy samples from tough samples and applies an extra learning step to the tough samples. Their approach is compared in table 6.1 and is comparable to DANN+BSP.

These approaches make use of the classifier entropy distribution and while offering interesting alternatives they do not explore the impact their approaches have on the singular values of the shared representation or its impact on accuracy.

Our work is distinct from these approaches, we seek to add an improved understanding of the mechanisms underlying performance of domain adaptation algorithms. Our work complements the work of [11] where they proposed the BSP and we incorporate insights from the deep metric learning literature corroborating the insight into the impact of embedding entropy on accuracy. We further propose a new penalty for adversarial domain adaptation methods that achieves performance competitive with state of the art approaches on a variety of benchmark datasets.

6.6 Conclusion

In this chapter we have provided a new perspective to understand the success of the BSP by utilising the observations of [98]. The entropy of the learned distribution is indicative of good performance and weighting more features forces the network to learn a better representation.

We demonstrate the validity of this observation through our two penalties on the singular values, showing how maximising the KL-divergence from the uniform distributions leads to significant improvement of baseline method performance. While some recent works have also produced strong results on a number of benchmarks our simple penalty remains competitive and in some cases better than these more complex models.

Further to this, we take these observations and combine entropy penalisation with the BSP to propose our own Batch Spectral Entropy Penalty which in many cases outperforms either penalization strategy on its own. This fresh perspective we hope will guide future development of domain adaptation research, as the BSP perspective already provided state of the art results for domain adaptation tasks. Future work in this direction could more closely examine the link between the lack of diversity in GAN based generation and the compressed representations observed here. Applying some of the many GAN extensions may also provide a path to increasing embedding diversity and decreasing singular value entropy.

Our work serves to demonstrate an alternate pathway to apply the BSP and highlights what the BSP means for the SVD of the embedding space. The entropy penalty allows for the importance of less influential concepts to be grown and removes the totally downward pressure of the BSP. This is what contributes to the small improvements we observe through combining the BSP and entropy penalties into the BSEP. Where adversarial adaptation seeks to emphasise a single feature, better classification can be achieved through a broader focus on smaller differences. This mechanism is highlighted by our work and our proposed penalty benefits from this.

Combining and synthesising insight from various transfer learning domains and offshoots is likely to provide further benefit to the community through both extended understanding and improved modelling. Our paper has shown an alternative perspective for understanding the BSP with positive improvement coming from the synthesis of insights from deep metric learning and domain adaptation. Future work would also seek to investigate the effect of the various alternative domain adaptation models and their individual extensions on the nature of the singular value distribution. Methods such as pseudo labelling are likely to increase the influence of more features, such that inputs can be classified. Applying the BSEP to other adversarial

learning problems and domains may also provide benefit if the nature of adversarial alignment is such that it causes a similar effect on the relative importance of features.

Chapter 7

The Extrapolation Problem

7.1 What is Extrapolation?

Extrapolation can and often is defined as predicting data that lies outside the convex hull of the training data. If data falls within the convex hull of our model, then the model should be able to interpolate, if the data falls outside the convex hull the model is said to extrapolate.

We take our definition of extrapolation from [105]. Definition: interpolation occurs for a sample x when it belongs to the convex hull of a set of samples : $X \triangleq \{x_1, x_2, ..., x_N\}$, otherwise, extrapolation occurs.

This definition captures the concept of interpolation and extrapolation, however as noted in [105] it also may be lacking as most samples will be guaranteed to fall into the extrapolation region given a sufficiently large representation.

Indeed the convex hull definition also fails to capture what is often meant by extrapolation, favouring a view of interpolation that is closely related to linear interpolation between data. However non-linear interpolation of data that lies on a curved manifold also makes sense and is not captured by the given definition. This leads to the need for new ways to view and measure extrapolation, interpolation and generalisation. In this chapter we do not seek to formally define extrapolation, but more aim to elucidate the types of problems that may be considered extrapolation and to add to the discussion of this complex topic.

To that end, while considering the above definition, we will also take a more philosophical approach throughout this chapter. We argue for the need to consider extrapolation when discussing machine learning and to develop better methods to test and evaluate it, as well as arguing for the need to improve our language around the concept.

In the remainder of this chapter we introduce the concept of extrapolation and discuss some of the difficulties in evaluating success on the task. In chapters 8 and 9, we examine the applicability of two different types of models to the problem of extrapolation.

Our work in chapter 8 seeks to enter this space from a dataset perspective. We design a data-set which requires extrapolation in image space to solve. For our purposes extrapolation involves a progressive change in the data characteristics, where the task remains the same and the data is generated from a sequential area of image space. This follows the concept of a convex hull in the image space and

is nicely illustrated in Domain adaptation and other learning methods such as one-shot learning seek to learn models that will generalise outside of the larger training domain used to train the model.

7.2 What are the problems?

The problem of extrapolation is central to science and at the heart of machine learning. Extrapolation is part of the generalization problem. Generalization can be separated into two parts, interpolation where inputs are unseen but lie within the bounds of the observed parameter space and extrapolation where the data is again unseen but sampled from outside the bounds of the observed parameter space.

Generalization is often discussed in the context of a trained model generalizing to the test set. This context is important to evaluate to ensure the model has learned a viable function, however, since the training and testing data are sampled from the same distribution it may not always be clear that the learned model is not unduly biased. One such example of this hidden bias is examined in [106] where various groups of people significantly less likely to be accurately classified due to bias in the training data, and the bias is difficult to detect due to bias in the test data which hides the scale and without detailed testing goes unnoticed.

Recent work has also highlighted the problem of underspecification [10]. This problem arises when large models, such as deep neural networks, are able to learn multiple possible solutions, which perform well on the existing test data but which use very different solutions which do not generalise equally. As large models are now common, controlling for underspecification is important as judging models based on their given test accuracy alone does not provide ample evidence of they generalisability.

Extrapolation is at the core of the problem of discovering the underlying function that explains a certain aspect of reality. These functions can be incomplete, consider Newton's laws of motion, but still general. Machine Learning algorithms are striving towards these functions, and attempting to distill them from complex data-distributions and user defined tasks. Understanding the mechanisms of this problem and solutions to it involves development over a number of dimensions which we describe below.

7.3 What directions of work are required

The extrapolation problem may appear insurmountable, how can we verify performance on data we do not know about? Our models aren't built with the knowledge of the existence of unknown unknowns. Humans on the other hand are capable of discovering formulae which work, even for data we have not seen. These equations allow us to make predictions about the real world.

An interesting aside here is that the general formulae which describe processes in the natural world, which are approximations, are often relatively simple, compared to the complex functions learned by neural networks, and this makes them explainable. Through learning the true general function we may discover neural networks become more explainable. Indeed we could also take inspiration from the physical sciences and heavily enforce sparsity of the function.

In the following we lay out some of the developments, we believe, will help us achieve models that can extrapolate. This involves infrastructure, architecture and structure of the learning algorithm.

7.3.1 Datasets

Datasets are an integral part of driving machine learning development and success stories. The ImageNet challenge for example has driven progress in the computer vision domain, with many of the well known base architectures proven on the problem. Further to this models, trained on ImageNet are used as base initialisations for direct transfer approaches, especially in data limited domains.

Existing datasets also contain and can introduce bias. This problem is pressing because it is often not obvious that the dataset is biased or that the learned models also contain this bias. Fully documenting the dataset is likely to be beneficial in the future [107]. Furthermore explicitly modelling test problems as extrapolation can uncover whether the model has truly learned the general function.

Datasets designed with extrapolation in mind can be combined into a standard train and test set, however, the extra freedom to train on specific segments of the data-distribution can be used to form stress tests for the model, to verify the function learned is the desired function. Further these tests can reveal data that is easier or more difficult to extrapolate from, perhaps certain dimensions are easier to extrapolate to than others. Understanding such questions will provide insight for

model development in the future.

In chapter 8 we introduce two shape counting datasets that are used to demonstrate the impact the choice of training region has on extrapolation. We further demonstrate the varied impact training region has on extrapolation in chapter 9. Indeed our experiments demonstrate the difficulties associated with underspecification [10] and the need to introduce experiments to detect and avoid it.

7.3.2 Modelling approaches

How a problem is modelled can influence whether or not a given solution will work. Understanding the various factors that influence extrapolation and how various modelling approaches overcome or are hindered by these factors is important for developing robust solutions.

In chapter 9 we demonstrate a one-shot learning approach and training setup that extrapolates with state of the art performance. Our approach is compared against a new normalization method [108] and demonstrates that algorithm design can have as much influence as architectural solutions in this space.

Transfer Learning

Transfer learning is a natural candidate for models that generalise to out-of-distribution data. This is, after all, the stated goal of the various sub-divisions. However the definition of out-of-distribution is often limited to data from a different dataset and otherwise unexamined. The extent to which this is extrapolation in the domain adaptation case is not obvious. Many domains are pushed right up against eachother. Developing a more formal collection of datasets of increasing distance from the source would prove useful in evaluating models that perform extremely well in a number of cases.

In the case of domain adaptation we can ask questions about the potential of the adapted model. If there is a gap between the source and target data distributions does the adapted model also demonstrate improved performance over that, now, interpolation region? Similarly is the model better able to extrapolate to data beyond the target dataset? These are questions we examine in chapter 8

Other transfer learning methods are also worth investigating in this context. Meta-Learning models have potential to generalise and some rudimentary experi-

ments have demonstrated this [109]. Our work in chapter 9 demonstrates the potential of one-shot learning methods in this context.

Explicit Functions

Other work[110, 111] has proposed shallow architectures for learning equations, given data from a subset of the functions domain. These approaches show a strong potential for extrapolation, however, they are currently limited to more controlled input settings. Examining the extent to which the proposed techniques could be adapted to visual learners may prove to be a viable path forward.

The authors[110] also argue for sparsity of the function as this maps well to the type of functions humans use to describe systems, which are typically compact and allow for explanation. In order to achieve sparsity the authors argue for a training scheme that uses regularisation terms at different points during training, starting without any and then adding regularisation.

7.3.3 Normalisation

How the data is used and processed can also influence the learning that occurs. A long standing method of improving generalisation from training set to test set is to employ regularisation. Regularisation methods restrict what can be learned by introducing constraints on the learning model. One example from recent work [108] introduced temporal context normalisation, a temporal version of batch normalisation. They use the approach to solve an analogical reasoning problem and demonstrate significantly improved performance over other existing normalisation techniques.

7.3.4 Curriculum learning

The smart utilisation of data can also be employed to improve knowledge and skill acquisition. This is the promise of curriculum learning [112, 113], where a schedule is used to expose the learner to samples that are specific to the current model's ability.

Related to this is data augmentation, a common approach in machine learning used to improve generalisation and avoid over-fitting on smaller datasets [114]. Augmenting the training, and perhaps testing data in order to induce an extrapola-

tion challenge, may be a viable path towards evaluating extrapolation performance when samples from across the full data-distribution are unavailable. Targeted use of augmentations may produce models that learn a function that generalises out of distribution more reliably. Such an approach could be combined with ideas from Curriculum learning.

7.3.5 Is it all interpolation?

Other recent work has suggested that learning and predictive power comes from an extreme interpolation approach, dubbed Direct Fit[27]. Direct Fit is a fourth case of the usual description of model fits to data, namely Under Fit, Exact Fit and Over Fit. If interpolation is the answer, we can still ask questions about our various transfer approaches. If a model is adapted to a new dataset does the interim range between the source and target become an interpolative domain? If so how separate can the source and target be before the performance benefit on the interpolation region is lost? Such questions are also worth investigating but are largely missing from the discussion in a lot of sub-fields.

7.4 Extrapolation for Transfer Learning and Beyond

The problem of extrapolation is deeply tied to the generalisation problem in machine learning systems. To ensure algorithms are capable of working in the wild, even when faced with out-of-distribution data, it is important to consider solutions to these questions in the lab. Developing an evaluation infrastructure, testing existing transfer and adaptation methods and developing new methods are all important pathways to progress on this complex topic.

A secondary issue, though important in its own right, is the issue algorithms introducing bias when applied in real-world systems. This problem has been demonstrated in relation to word embeddings [115] and image processing [106]. In the image processing case it was shown that algorithms significantly misclassified the gender of people as they were sampled with darker and darker skin. This issue is multi-faceted and in part is caused by the data-gathering process. The difficulty though is that it is an issue that goes un-noticed when examining the accuracy on the whole dataset. This is an issue that could be identified through the adoption of a rigorous testing infrastructure. Stress tests, that evaluate specific sampling condi-

tions and indeed sample from data outside the sampling range of the training data, will aid in the development of algorithms and in ensuring better more thorough evaluations which will limit the potential negative effects of misclassification.

Evaluating these problems as a form of extrapolation to under-sampled data spaces provides a way to both think about these problems and to develop solutions. Further developing datasets with extrapolation in mind provides a pathway to identify similarly under-represented and likely misclassified data-points. A recent suggestion, which would likely be beneficial alongside the proposed direction, is the adoption of 'Datasheets for Datasets' as proposed in [107].

Learning and evaluating models that can extrapolate will require development across a range of issues, as we have discussed, however it is likely to also bring significant improvement in model performance. In the next two chapters we will examine the applicability of existing transfer learning approaches to the problem of extrapolation.

Chapter 8

Domain Adaptation as Extrapolation

8.1 Introduction

Neural networks and machine learning are successful because of their ability to generalise. This is usually in-domain generalisation to unseen test data, however, generalisation to out-of-distribution data has also been made possible through various transfer learning techniques. One such technique is domain adaptation which is usually applied to sufficiently similar datasets. Sufficiently similar is left largely to the user to interpret with obvious dataset examples being used, such as MNIST and SVHN. This leaves us with the problem of having an experimental feeling for when domain adaptation will apply and what problems it solves.

In order to address this question it is necessary to more explicitly evaluate the generalization achieved. In order to evaluate generalisation it is standard practice measure performance on an unseen test set, but this data is usually sampled from the same domain with the same, or similar, underlying data-distribution. In many cases this may be sufficient for many applications, however it is not without issues [106] and does not necessarily illuminate all the important characteristics of a model [10]. In the case of unsupervised domain adaptation the separation between training and test data is often less defined, with the test set often being the same as the target with models evaluated not on a held out test set but directly on the dataset that the model was adapted to.

A further underlying question is whether or not neural networks learn the underlying function, which would allow the model to more fully generalise, extrapolating to new data as well as interpolating unseen data-points from within the data distribution. Indeed experiments have demonstrated a failure of the model to extrapolate, even when that model can interpolate extremely well to data that was never trained on [27]. In the context of this question about a network's ability to extrapolate, domain adaptation as method for out-of-distribution generalisation is a clear candidate for evaluation.

Understanding the potential of domain adaptation explicitly in the context of extrapolation and interpolation has so far been left unexamined. A lack of datasets makes such investigations difficult and existing datasets are not well suited to such investigations. Further to this questions as to how general is the model after adaptation, what happens in the data-distribution space between the source and target tasks, are also unexplored due to the same issues of not having appropriate datasets.

In order to investigate these questions a simple set of datasets are introduced which we call the Ellipse dataset and the Polygons dataset. The task in both cases is to predict the number of shapes in a given frame. The ellipses in the frames can have varying eccentricities and are placed randomly at random rotations around the image. The polygons vary by number of edges. Through controlling these variables we can create a range of sequentially distant tasks to extrapolate to. We then demonstrate the network's inability to extrapolate beyond the training data and show that domain adaptation can significantly improve performance on extrapolated data. We also examine the secondary benefits of adaptation through its effect on interpolation and further extrapolation.

8.2 Background

In machine learning problems there is usually an underlying assumption that, in order for the model to work at test time, the training data is sampled from the same underlying distribution as the target data. If this is the case then a trained model will make accurate predictions. If test data differs from training data learned models can and do fail. This problem of applying a trained model to data sampled from a different domain to the source domain used for training gives rise to the problem of domain adaptation.

Domain adaptation is a family of methods used for transfer learning in cases where the data the model will be applied to differs from the data it was trained with. This can be useful in situations where the target data represents similar information to the source data, but where labeled data in the target domain is unavailable. Utilising a pre-trained model as a basis for classifying target data accuracy can be improved on the target task by regularising the learned embeddings.

ADDA, a method and adversarial learning framework for unsupervised domain adaptation has achieved success in this problem space on a number of research problems, such as transfer from SVHN to MNIST, and more practical problems such as medical image analysis. The standard ADDA setup is to pose an adversarial learning problem where a generator G is trained to on target data to generate samples that appear to have originated from the source domain and a domain discriminator D is trained to separate samples from G from samples generated by the source model S. This learns a shared embedding space and then the source classifier can be applied

to samples from G.

We also investigate the effect of two variants of ADDA[69] by extending it with either an autoencoder loss as in [72] or with the batch spectral penalty as in [11]. The AAEDDA [72] method adds a reconstruction penalty to the learned embedding of the generator. The model should not discard information about the target domain in order to match the source domain. This is easily added to the ADDA framework through the inclusion of a decoder model Dec to reconstruct the input from the source and target models. The decoder is trained with a reconstruction penalty from source and generator model embeddings of source and target data respectively. The loss for the generator then has the reconstruction penalty added to its loss.

The other method we apply is to utilise the batch spectral penalty(BSP)[11]. Recall that, for a given feature matrix F, the singular value decomposition is given by $SVD(F) = U\Sigma V$ with Σ containing the singular values. The BSP method applies a penalty to the singular values of the embedding to restrict the importance of the top N features, using the reasoning that while transferability is improved by focussing on the few common features discriminability is improved when more features are considered. Applying the penalty is not expensive and it was shown to be effective across a range of adaptation models. The penalty is given by the equation 8.1

$$L_{BSP}(F) = \Sigma_{n=1}^{N} \sigma_{f,n}^2 \tag{8.1}$$

8.3 The Ellipse datasets

Neural network models are not known to be able to extrapolate beyond the range of their training data, even in cases where an explicit fully specified underlying function exists and on training data appears to be fully learned [27]. If data is outside the training range of the network then we suggest the problem can be viewed as a new domain. In that context we will explore the applicability of the domain adaptation to the problem of extrapolation in neural networks.

In order to investigate the applicability of domain adaptation to the problem of extrapolation we create a set of datasets containing images of black ellipses on white background. Our target function involves recognising and counting ellipses. This provides a task that can be used to investigate the extrapolative powers of a neural

network.

An ellipse generalises a circle and as such the equation of an ellipse centred at the origin is given by equation 8.2, which when a and b are equal gives the equation of the circle. The eccentricity of an ellipse, how squashed one side is can then be defined as shown in equation 8.3 .

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \tag{8.2}$$

$$e = \sqrt{1 - (\frac{b}{a})^2} \tag{8.3}$$

By varying the eccentricity we can create a variety of ellipses that go from low, fully circular, to high, very elliptical, eccentricity. This creates a diverse dataset from which we can create extrapolation tasks. To make this more concrete we show in figure 8.1 an example of a single ellipse as we vary the eccentricity from high to low. A more detailed example is provided in appendix B.

We allow for rotations in our generated dataset and therefore we need only control either a or b in equation 8.3 to generate wide or tall ellipses. Recognizing ovals is not a difficult task, and counting them is not difficult either, though issues of occlusion can occur as ovals partially overlap or partially fall outside of the visible area. Since it is not a difficult task we can use it to evaluate how well our network extrapolates.

8.3.1 Extrapolation when counting

(a) .95 (b) .70 (c) .45 (d) .20 (e) 0

Figure 8.1: A sample of the change in ellipse shape by eccentricity from 0.95 to 0.0.

The ellipse counting task is set up as a classification task; given a frame with between 0 and 5 ellipse, predict the corresponding number of ellipses.

We create 3 training datasets, with non-overlapping eccentricity boundary controls:

low $[0.01, 0.2]$

mid $[0.4, 0.6]$

high $[0.8, 1.0]$

We also create datasets consisting of samples from across the entire range in steps of 0.05. This creates a set of datasets that allow for evaluation of interpolation between adaptation regions and extrapolation to and beyond the target adaptation region.

The task is to train a model on samples from one of the training eccentricity ranges and then to adapt the model to one of the remaining two ranges.

8.4 The Polygon dataset

We introduce a second shape counting dataset. This dataset is constructed with polygons of varying numbers of sides. This gives rise to a dataset transitioning from triangles to almost circular objects.

(a) sides:3 (b) sides:8 (c) sides:13 (d) sides:18 (e) sides:21

Figure 8.2: An example of the change in polygon edges from 3 to 21

Similarly to the ellipse dataset we break the polygon dataset into 3 datasets that can be used for training and adaptation. There are also gaps in the space between the datasets, this allows for evaluation of the interpolation that occurs between source and target dataset after adaptation. The polygons are varied by number of sides from $[3, 20]$ and we create three training sets dubbed low, mid and high, after the subset of the range of sides used to generate the datasets. An sample of the polygon transition range is provided in figure 8.2 and the full range is presented in appendix B.

Polygon training datasets are composed as in the following list:

low: 3,4,5 sides

mid: 9,10,11 sides

high: 15,16,17 sides

8.5 Method

We provide experiments to verify the need to apply domain adaptation on the proposed transfer tasks and datasets. We then further probe domain adaptation methods as methods of extrapolation. Through the design of our datasets and experiments this also allows for the investigation of the interpolation performance between the source and target adaptation tasks.

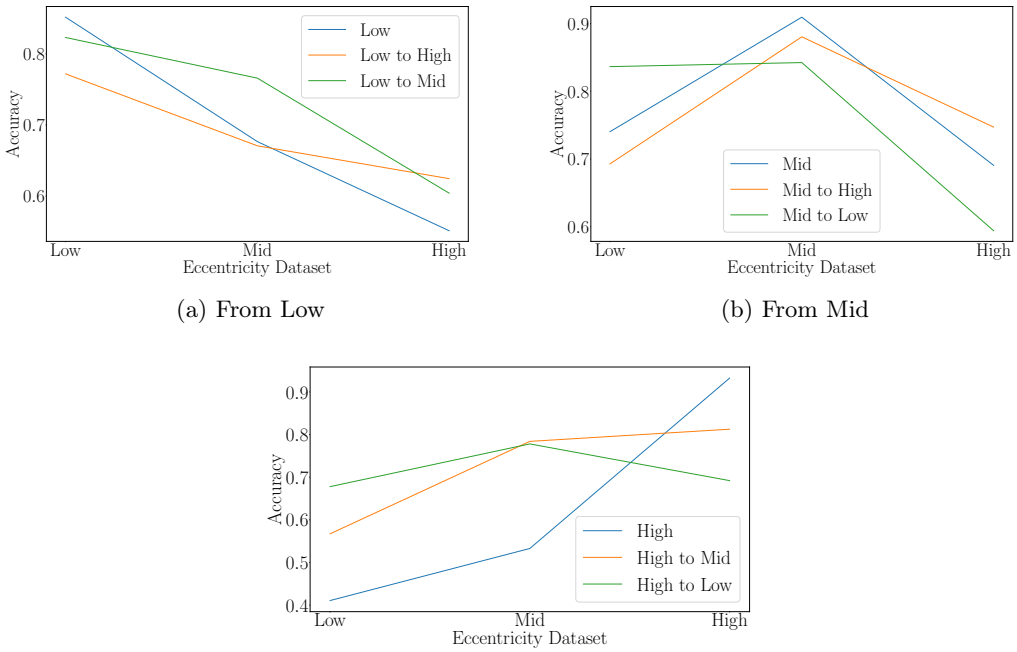
The suitability of our proposed datasets is verified through the application of direct transfer. These experiments indicate that the solution on the source datasets can be learned but that generalisation is more difficult. In order to investigate the potential for domain adaptation to improve the generalisation of a model we compare the direct transfer approach with ADDA.

We investigate 4 variants of ADDA-based domain adaptation. We investigate standard ADDA with and without the BSP penalty of [11]. We also investigate the AAEDDA approach, with and without the BSP penalty. In this case we amend the training procedure of the AAEDDA and the Decoder network is used to reconstruct both source and target embeddings, this is distinct from applying the encoder-decoder architecture solely on the target data. The base model is based on Pytorch's pre-trained Alexnet, however, we edit the embedding layer down to 1000 dimensions. We experiment with both pre-trained and randomly initialised instances, but find performance to be substantially similar.

We train on a dataset generated from one of these range settings and as expected we see performance decline when testing on the others. The effect can also be observed by testing only on single eccentricity samples from those ranges with a steady decline in performance as the samples come from further out of range. This can be observed for models trained on each set; however, the model trained in the high range does generalise better to the mid range range.

8.6 Results

Our results demonstrate improvements on the adaptation target combined with reduced performance on the source. This behaviour is expected, however, our extrapolation datasets also reveal varying patterns and behaviours depending on the chosen source. These results highlight the need to consider the impact and design of the training set when considering a domain adaptation approach.



(a) From Low

(b) From Mid

(c) From High

Figure 8.3: We show the adaptation, using ADDA, from each of the ellipse training sets, low,mid and high, to each of the adaptation sets. We observe the largely expected patterns, adaptation generally helps and works in the direction of adaptation. Adaptation usually hurts performance on the source task, though much less so when adapting from high.

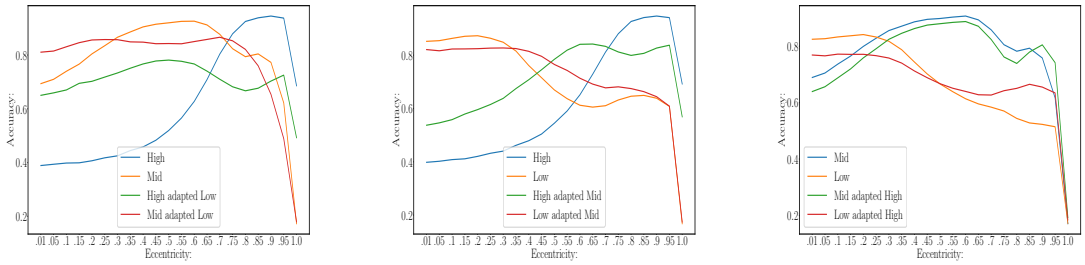
8.6.1 Does AlexNet Extrapolate and Does ADDA Help

We provide a high-level examination of adaptation across our tasks in figure 8.3, where we apply ADDA to each of the ellipse datasets. We observe some interesting effects with these experiments. We show the expected decline of model accuracy when evaluating the base AlexNet model on out-or-distribution ellipse eccentricities. We also can observe that adaptation is generally a good idea; however, adaptation does not achieve a perfect score demonstrating that this task is complex enough to offer interesting observations.

One such observation is that adaptation direction appears to be important.

Adapting from high-to-low provides a significant improvement in accuracy. This is not the case adapting from low-to-high, where the benefit is much less significant. This is particularly true in the interpolation region where mid-range performance is significantly improved for the high-to-low case. Adapting from mid range eccentricities improves accuracy in the direction of adaptation, however, it reduces accuracy in the other direction. In the case of mid adapting high the model scores very poorly on low.

The second observation worth noting is the benefit across the whole eccentricity range when adapting from high-to-low. We never see nor adapt to data in the mid eccentricity range, however, the adapted function provides similar improvement to adapting directly to the mid range values. This suggests links to the ideas in [27] where by fitting a large enough data space our network will be able to interpolate. The adaptation in this case is promising because not only do we see adaptation extrapolating, but also interpolating to unseen data in the new larger data domain. We also see indications that neural networks learn extrapolative functions by observing the increase in performance on out of range data when adapting the model towards it but not fully, for instance adapting the model low-to-mid also improves performance on high eccentricity ellipses too.



(a) To Low (b) To Mid (c) To High

Figure 8.4: We show the adaptation, using ADDA, from each of the ellipse training sets, low,mid and high, to each of the adaptation sets across the range of eccentricities. We observe the largely expected patterns, adaptation generally helps and works in the direction of adaptation. Adaptation usually hurts performance on the source task, though much less so when adapting from high.

A more fine-grained analysis can also be undertaken by evaluating our model on datasets across the range eccentricities. Shown in figure 8.4 are comparisons between adapting the various source eccentricity ranges to the specified target range across this range. We note the obvious that the model improves on the eccentricity range it is adapted to. This demonstrates that domain adaptation can adapt to an

extrapolated data space; an expected result as ADDA is a method for out-of-domain adaptation. The observation that is more informative is the comparison by distance. The model trained on mid-range ellipses and adapted to low adapts significantly higher than the model adapted from a classifier trained on the high range dataset. This pattern is repeated when adapting to the High-range ellipse dataset, where the mid-range trained models perform better. This is also corroborated by observing the same pattern when comparing the base un-adapted models.

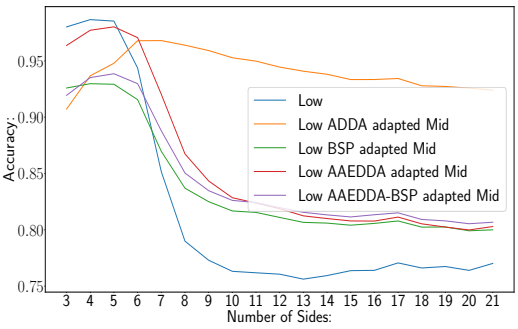
Examining the extrapolation potential of domain adaptation we look at adaptation to the mid-range values. Adapting from high-to-mid range improves performance over the mid-range data; however, that benefit is also maintained when evaluating on further out of range on the low-eccentricity data. The other adaptation direction from low-range does not appear to have the same benefit. This suggests that some adaptation tasks are better for extrapolation than others as well as demonstrating why this task requires further research developing datasets.

8.6.2 Do ADDA variants offer improved extrapolation

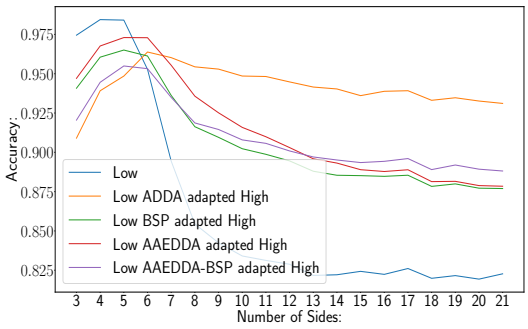
We have shown how ADDA can be applied in our experimental context to extrapolate the learned filters and functions to new unseen data and how the performance on the data in-between the 2 datasets is also improved, demonstrating interpolation. We now investigate the effect of extending ADDA to investigate whether the techniques used to improve adaptation are also useful for this explicit extrapolation task.

We adapt three variants for this investigation. The first is to use our AAEDDA [72] proposed in chapter 3. This method applies a regularising penalty, in the form of an autoencoder loss to the encoder. A change here is that the decoder is trained on both source and target embeddings. The second extension we investigate is the Batch Spectral Penalty [11], BSP, which has been shown to be a low cost high benefit penalty. The final extension combines the BSP penalty with the AAEDDA.

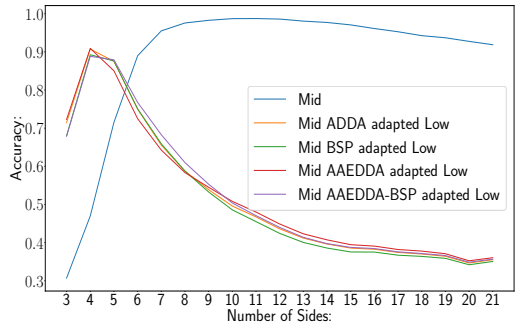
We adapt each of these models on the polygon task which reveals that most of the adaptation methods perform comparably with the exception of adapting from low-sided polygons. Here the standard ADDA approach adapts and extrapolates best. We do, however, observe the AAEDDA model retaining the most of its source performance, indicating a potential benefit for utilising the auto-encoder based method



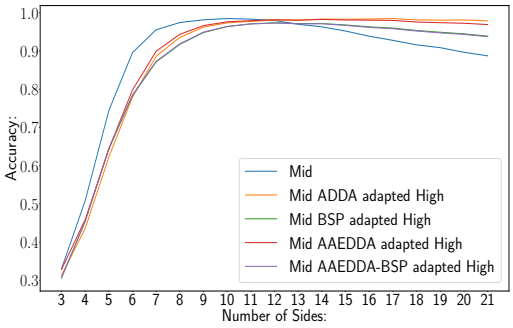
(a) Low to Mid adaptation



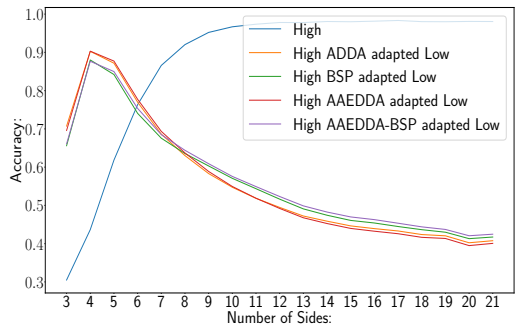
(b) Low to High adaptation



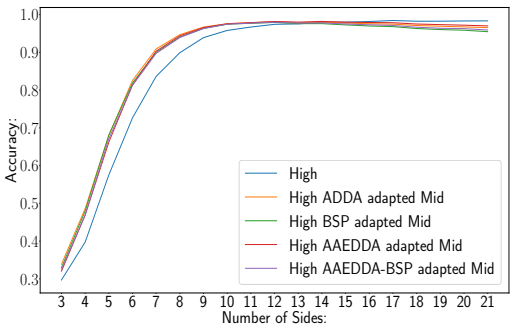
(c) Mid to Low adaptation



(d) Mid to High adaptation



(e) High to Low adaptation



(f) High to Mid adaptation

Figure 8.5: We show the performance of each of the models evaluated across the range of polygons before and after adaptation. Adapting to the low range dataset is a detrimental policy with regard to the rest of the data-distribution but does significantly improve performance on the new task. Other adaptations provide more significant benefit. Adapting from Low to High improves performance in the interpolation region and adapting from Low to Mid appears to have extrapolative benefits.

if source performance is also a concern after adaptation.

Considering the questions of extrapolation and interpolation the results on this set of tasks is mixed. Both tasks involving adaptation to mid-sided polygons exhibit improved performance relative to the un-adapted model on the subsequent extrapolation region. Adapting to the low-sided polygon task is detrimental to performance of the model across all other values. By comparison adapting from low-sided polygons is significantly less detrimental. The benefit of adaptation to interpolation is evident in adaptation from low-to-high sided polygons with models performing better on the mid-sided polygons they were not explicitly adapted to than when they we explicitly adapted. These observations suggest the importance of considering the direction of adaptation and of developing further datasets to probe various modelling approaches to overcome such difficulties and exploit such benefits.

We also examine the effect of the same models on the ellipse tasks. As expected a similar picture emerges. The Mid-Eccentricity base model performs well when generalising since it is closer to both extrapolation sets. The decline in performance is more pronounced for models trained on either edge of the distribution. The AAEDDA method is competitive exhibiting an ability to maintain source accuracy after adaptation, though for the more difficult task, based on the performance of the base model, the adaptation performance is worse. Indeed the more difficult task appears to be dominated by ADDA. Potentially significant divergence from the source distribution is more difficult for specialised methods to overcome.

The results presented demonstrate that the direction of adaptation is worth considering when applying domain adaptation. Generalisation is possible in the direction of extrapolation, as evidenced by High-to-Mid and, Low-to-Mid adaptation in the polygon dataset shown in figure 8.5. It is also useful to consider the benefits of interpolation after adaptation. If the data to be tested on lies in the interpolation range between the source and target datasets adaptation also benefits model performance in the interpolation range. This suggests that distant target datasets may provide the best overall generalisation potential.

Considering the problem of developing methods that can be used for generalisation we examined the effect of domain adaptation on constructed shapes datasets to discern the potential for ADDA style models to generalise. A secondary observation is the disparity in base model performance when evaluated on out-of-domain

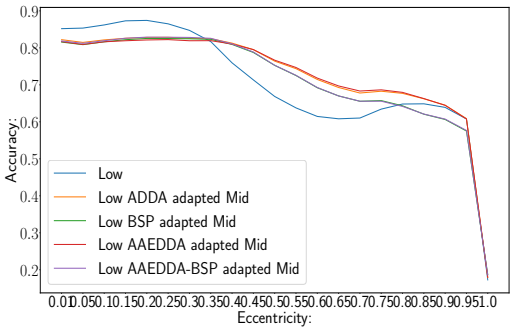
data, indicating that not all directions of transfer are equal. What we observe is that low-sided shapes, and high eccentricity ellipses are less aligned with the other respective datasets. This highlights the difficulty in evaluating the performance of domain adaptation models and the need for further development and documentation of datasets for transfer learning and domain adaptation.

The success of standard ADDA on the hardest adaptation tasks, as indicated by the direct transfer evaluation, further underscores the need to develop datasets that explicitly consider the distribution the data is sampled from. It may be that more aligned datasets are better candidates for newer adaptation techniques but that they fail for more distant distributions. Developing this understanding will require more datasets to be identified or created that directly address and allow for the investigation of this question. Further the difficulties experienced by both datasets in adapting from low-sided or high-eccentricity indicate that the problem of measuring relatedness of data is difficult and our simple datasets while informative perhaps do not capture the full nature of the problem. This further highlights the need for further work and the development of new datasets to evaluate this.

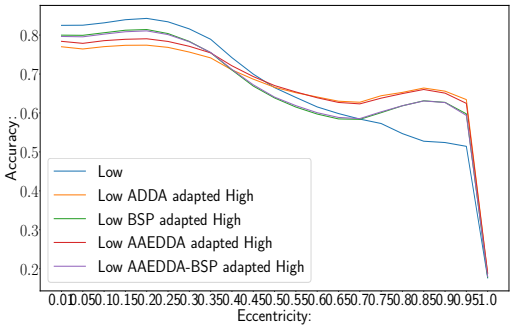
8.7 Related Work

Generalization is the bedrock assumption of machine learning. That after training our model can be applied to unseen data and be expected to work. Under certain assumptions this is the case and machine learning has experienced numerous successes because of this. However machine learning models are brittle and sensitive changes in the input. This is the case in reinforcement learning when moving a model from simulated data to real-world data. It is also the case when applying models to different styles of photograph. Such issues are problematic but difficult to probe as it is difficult to measure the distance between simulated data and real-world data.

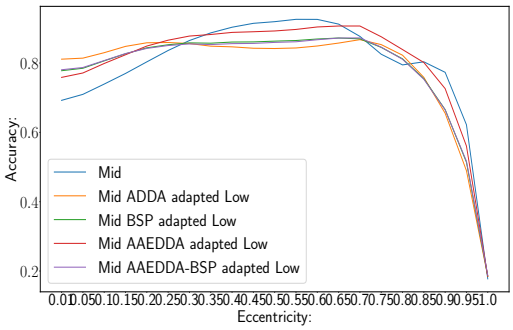
Recently proposed work with the aim of examining and improving extrapolation proposed datasets in the context of visual analogies [116]. This is a problem of sequence recognition and completion, with some shared factor affecting the sequence, for a given source sequence and an unseen target sequence. This is a very human problem and a worthwhile target for examining extrapolation. The extrapolation problem posed by the visual analogy task appears to be quite a difficult . This line of work proposes a learning problem of classifying *AistoBasCistoD* relationships



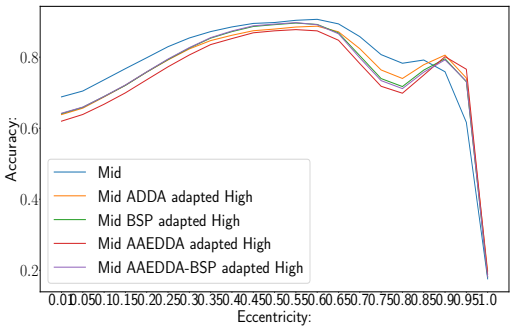
(a) Low to Mid adaptation



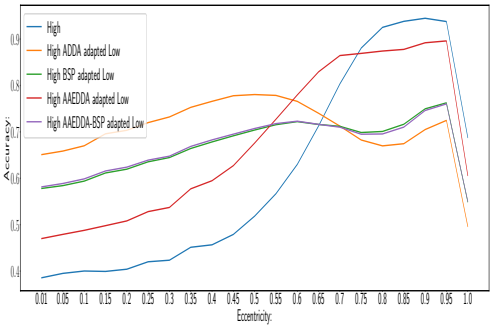
(b) Low to High adaptation



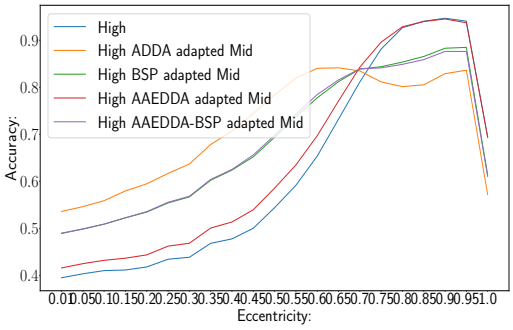
(c) Mid to Low adaptation



(d) Mid to High adaptation



(e) High to Low adaptation



(f) High to Mid adaptation

Figure 8.6: We show the performance of each of the models evaluated across the range of ellipses before and after adaptation.

for a given A, B, C and a target value D. The work in this area shows both the difficulty but also ability of neural network models to extrapolate various relations to different shapes or scales.

However, problems which on the surface appear easier can also cause issues with extrapolation for neural network models. For instance the sin prediction problem demonstrated in [27]. Learning to predict the $\sin(x)$ for a given range of x values is shown not to have predictive power outside of the range used for training. Our view, and what our work shows, is that our neural networks can and do learn approximations to the underlying function, and though these approximations are biased to the training data they can be adapted to work in a more general setting. Our proposed dataset focuses on learning the underlying function which, in our case, describes the change in shape of an ellipse in relation to its eccentricity and applying that to identify shapes and count them. The space of understanding extrapolation and the potential tasks

As a form of out-of-distribution generalization domain adaptation methods appear to be useful for this task, however, explicitly evaluating extrapolation and interpolation between source and target data is a problem that has not been addressed by the community. ADDA and related variants are used for domain adaptation and have been shown to be effective in many contexts [99, 69, 117]. The utility of domain adaptation models is in their applicability to unlabeled datasets, allowing researchers and practitioners to apply pre-trained models more successfully to new data, gathered for instance from new machines or with different imaging effects.

Our work offers insight into the large potential utility of ADDA style methods but also helps to illuminate where these models fail. The direction of adaptation usually goes from larger more complex to smaller less complex data; though this is not usually discussed. Common adaptation directions are SVHN to MNIST, Office-31, and we show in a controlled manner that the direction of adaptation has a large impact on the success or failure of the model.

8.8 Conclusions

In this chapter we introduced the Polygon and Ellipse extrapolation datasets to test domain adaptation models for their effect on generalisation. The problem appears to be a relatively easy task and training a model to high accuracy across the full range

of eccentricities is not difficult. However, by restricting the range of eccentricities seen during training we can create an adaptation problem which is non-trivial. The application and success of domain adaptation models has much potential, however, evaluating the models on larger and more complex tasks fails to demonstrate the limits of this approach. We have taken a first step here to developing tasks that can be used to probe these methods.

This development serves to highlight the need for new datasets to be developed. A better understanding of the training data used will elucidate where models fail and what development needs to take place. This observation is supported by other recent work, demonstrating hidden failure cases caused by biases learned during training [106]. The under-specification problem [10] also illustrates that the need for newer testing protocols which evaluate the learned function is indeed the desired function.

Further to better understanding the potential and limitations of Adversarial Domain Adaptation we show that ADDA can be used to better learn the underlying function. This shows great potential as the network is not limited to data that exists in the subspace of the training data but data that is not captured by the source and target data, save for sharing an underlying function, can also be correctly classified.

We have shown that ADDA style methods can improve a networks extrapolation potential, and that adaptation can also provide interpolation benefits between the source and target data-distributions. We also show that adaptation direction is an important consideration. This highlights issues with applying domain adaptation, especially for cases where it might be used, when limited labeled data is available and justifies our calls for further work to establish which methods are best suited to the full generalisation task.

Future work would seek to further investigate extrapolative potential and limitations of other domain adaptation models. The ellipse dataset is one of a number of potential shape based problems and developing new datasets is an important direction of work to fully explore this problem space. Applying ADDA in the context of the recently proposed Visual Analogy Extrapolation Challenge datasets could lead to more insights in relation to use cases and limitations.

Chapter 9

One-shot Extrapolation

9.1 Introduction

Machine learning algorithms are often bound by their training data relying on the assumption that the training and test data are i.i.d and are unable to generalise to data outside of the training range. An explicit example of this is the failure of a simple neural network to predict a sin wave outside of the domain of the training data; however, the model is able to interpolate very well, even when trained on, for example, only even numbers [27].

This brittle generality is limiting when it comes to applying AI systems in the real world for tasks such as level 5 autonomous driving. An AI algorithm needs to be able to function in entirely new circumstances. In [118] the concept of generality is used to define the abilities of an AI system. To extrapolate or to learn the general rule which can be applied in context is a stepping stone on the way to generality, and adapting to unknown domains and tasks. One-shot learning and meta-learning have the potential to bridge this gap.

One-shot learning is a powerful paradigm for learning in domains where data is scarce. In many ways, one-shot learning resembles extrapolation, for example classifying letters from previously unseen languages given a single example [119]. This is, however, difficult to directly measure as extrapolation. When considering the concept of generalisation with respect to humans much work has been conducted to measure and understand our capabilities. One approach to the problem is to consider the problem of analogical reasoning, reasoning by analogy. The problem is to recognise a pattern and generally select an object that completes the pattern. Ravens Progressive Matrices are one such form of analogical reasoning tasks. These have recently been used to explore machine reasoning. A recent dataset proposed in [108] allows for a more fine-grained analysis of extrapolation, breaking the analogy dataset into regions that are progressively more distant from the starting region.

In this chapter we observe the similarities between the visual analogy problem setup and the framing of one-shot learning algorithms with support and query sets, and use that to demonstrate that one-shot learning can be used to learn representations which extrapolate well and competes with other approaches to solving the problem of extrapolating reasoning about visual analogies.

We pose these questions since extrapolation constitutes a long standing problem and question for the machine learning community. Our statistical tools appear to

work extremely well when data falls into the interpolation range of our training set. However, seemingly simple functions, which we appear to have learned well within the range of training data, catastrophically fail when applied to data outside their normal range, for instance the problem of predicting the value of the *sin* function given an input [27].

Analogical reasoning has been suggested as being a crucial aspect of human intelligence and yet machines have found it difficult. In this chapter we explore the application of one-shot learning to the problem of analogical reasoning. Analogical reasoning problems usually present a set of options to complete an analogy and by re-framing the question in a very human way we can show one-shot learning is a powerful tool for analogical reasoning.

In the remainder of this chapter we introduce the analogical reasoning problem and the two datasets used. We then describe one-shot learning in the context of proto-net and frame the analogy problem as a one-shot learning problem. We then show the success of the approach to analogical reasoning and extrapolation outside of the training data. Further to this we examine the effect of extrapolating from different starting points. Our experiments then lead us to propose a training mechanism to improve generalisation performance across the range of tasks.

9.2 Problem Description

Understanding where and how a model fails is an important task. In machine learning shifts in the input data-distribution can cause a model to fail. Domain shift is an important problem that affects a model's ability to generalise. Many existing datasets however make it difficult to judge the distance from the training set at which the model fails to extrapolate. To provide a solution to that the Visual Analogy Extrapolation Challenge, VAEC, dataset was proposed in [108].

A visual analogy consists of 4 images in the form $A : B :: C : D$; read A is to B as C is to D. That is, the relationship between A and B is the same as the relationship between C and D. A visual analogy problem sampled from the dataset consists of a context relation $A : B :: C :?$ and a set of potential options to complete the analogy, the aim of the network is to accurately identify the correct sample to finish the analogy.

The VAEC dataset provides a true examplar in the form $A : B$ it provides a

sample C which is the start of the second relation C :? and it provides a number of additional inputs as foils F_i each of A, B, C can also be included as foils to create $A : B :: C : F_i$ relationships. This gives us 7 possible combinations of $C : F_i$ one of which is correct.

The VAEC dataset is comprised of two extrapolation regimes. Each regime provides a training set and 5 test sets, each further from the the training set. We will refer to these datasets as Region $1-6$ and Scale $1-6$, respectively. The analogies are constructed from green squares on grey backgrounds varying in brightness, size and X and Y co-ordinates. Each of the ranges is broken into 42 discrete levels.

9.2.1 Translation Regime

The translation regime tests for invariance to translation along the dimensions of variation, size, brightness and location. The space is divided into $7 \times 7 \times 7 \times 7$ regions along the diagonal, giving 6 translation extrapolation tasks. Small dim objects are found in Region 1 and large bright objects are found in Region 6. Individual analogy problems vary along a single dimension. This setup allows measuring gradual performance change over increasingly distant test regions.

9.2.2 Scale Regime

The Scale Extrapolation Regime also comes with six datasets, referred to as Scale 1 through Scale 6. Scale 1 is sampled identically to Region 1 in the Translation Extrapolation Regime. Objects are sampled along each dimension within the first $1-7$ levels. Scale $2-6$ are constructed by multiplying sampled values by a scalar ranging from $2-6$. Each of the Scale and Translation regime datasets consists of 19040 analogy problems and testing datasets contain data outside of the range used during training.

9.3 One-shot Learning

One shot learning offers a variety of learning architectures that learn models that can accurately classify a previously unseen sample given a small set of possible answers; an overview of the field its issues and advancements can be found in [120, 35]. This is useful in cases where some examples may be extremely rare and so we do not have sufficient data at training time to train a traditional deep learning model. It also

enables a very human like learning ability, most people do not need many examples of an object before they can recognise other versions of the object.

In this work we apply Prototypical networks [121] for the one-shot learning problem. The model learns a representation that minimises a distance function between a Query sample and the true answer in the given support set. Each input is processed by a neural network to produce an embedding and then the distance between the query and each labelled support is calculated. The network then computes a loss so as to minimise the distance between the query and the correct support sample.

In essence Proto-Nets learn a distance metric that allows the use of a nearest neighbours technique to then accurately classify data. Other options for this type of learning exist. Deep Metric Learning, is a field designed around learning this metric function with various loss and algorithmic designs. We employ the triplet loss as a related but alternative approach to Proto-Nets which will serve to verify the applicability of the general approach to extrapolation.

We choose Proto-Nets and Triplet loss methods as two instantiations of the one-shot learning approach. Both methods learn a relatively simple metric between embeddings and provide strong baselines to evaluate the idea that the structure of the learning problem is a useful consideration when designing models that explicitly extrapolate.

9.3.1 Prototypical Networks

The prototypical network is designed to produce a classification function by learning an embedding function for a given exemplar. The learned embedding represents a class prototype; for each possible class such a prototype is generated and the input to be classified is compared to each prototype.

To formalise the explanation, we are given a support set S with n pairs of the form (x_n, y_n) where $x_i \in R^D$ the D-dimensional embedding and $y_i \in 1...K$ the target class with S_k. The labeled set of samples of class k is denoted S_k

A prototype then can be computed as a transformation of support points $F_\psi : R^D \rightarrow R^M$. The prototype $c_k \in R^M$ is then computed as the mean of the transformed support points belonging to its class. In the one-shot case this mean is computed from a single point per-class, and therefore is simply the embedding of the data-point.

$$c_k = \frac{1}{|S_k|}\Sigma_{(x_i,y_i)\in S_k} f_\psi(x_i) \tag{9.1}$$

For a given distance function $d : R^M X R^M \rightarrow (0, +\inf]$, a distribution across prototypes can be computed for a given query sample x using a softmax over the distances to each of the embedded prototypes.

$$p_\psi(y = k|x) = \frac{\exp(-d(f_\psi(x), c_k))}{\Sigma_{k'} \exp(-d(f_\psi(x), c_{k'}))} \tag{9.2}$$

This distribution can then be used for learning through minimizing the negative log likelihood $J(\psi) = -\log p_{\psi(y=k|x)}$ of the true class.

9.3.2 Triplet Loss

The triplet loss approach to metric learning requires a triplet $< A, P, N >$ with the goal of learning a representation which under the loss reduces the distance between the Anchor Example A and the positive example P while simultaneously increasing the the distance between A and negative Example N.

The triplet network calculates an output given by

$$Triplet(x, x^+, x^-) = \begin{bmatrix} \|Net(x) - Net(x^-)\|_2 \\ \|Net(x) - Net(x^+)\|_2 \end{bmatrix} \tag{9.3}$$

which can be used to form the triplet loss, similar to the protonet loss, by computing a softmax over the distances and minimizing the negative log-likelihood, creating a binary classification problem; though it is noted that an MSE loss produced better results. The model can then be used to generate embeddings for multi-class classification using SVM or KNN approaches.

The triplet loss learns a metric function between samples where closeness is determined by class membership. Another consideration is how to sample positive and negative examples from the dataset. There are a variety of approaches and the method chosen has been shown to impact the effectiveness of the metric chosen, for a recent detailed summary and discussion see [98].

9.4 Method

We review the structure of the presented problem and solution in the VAEC and reformulate the problem as a one-shot learning problem. The authors in [108] follow the approach of [116] and train an encoder neural network followed by a recurrent layer to build an analogy representation over A, B, C and the target option D. This composes the whole relation and problem structure as one problem combining over four inputs. These inputs together generate a score, one such score is generated for each foil F_i example and the highest scoring predictions is used as the label. We break from this formulation by instead looking for the relation between pairs. We are provided with A, B and C, F_i as inputs. The goal then is to match pairs from a given set of options, which we will show can be well fit into the one-shot learning paradigm.

In the standard application of one-shot learning with proto-nets, the samples in the support set are labeled examples and we appropriately label the query sample by linking it to the support sample. For our application we restructure this usage to form what we call a ReverseProtoNet (RPN). Our support set is composed of the unlabeled foil example pairs and our query set is the given relation $A : B$. This setup fits the problem as we know the true relation $C : D$ is included among the foils.

There are many few-shot learning architectures in the literature and we choose protonet due to the simplicity of the model and learning structure. This allows us to clearly evaluate the one-shot learning method without adding more involved and potentially more powerful methods. The method is also closely related to the approach of [108] where a score is produced for each 4 tuple $< A, B, C, F_i >$ and a softmax taken across the produced scores. The scores we learn, however, are given by distance and represent a relation distance between the anchor pair relation and the subsequent relation.

We also apply the triplet loss formulation to the problem. Here we follow the same general structure as with the RPN example however we do not compare distances between all possible pairs. In the triplet loss set-up we learn the distance in relation to a given negative example. The structure of the problem is such that there exists only one anchor and one positive example per problem. While relation information is available as part of the dataset we do not use it to construct other examples for

	Region 1(Training)	Region 2	Region 3	Region 4	Region 5	Region 6
TCN.	99.1 ± 0.6	$\mathbf{77.0 \pm 5.8}$	$\mathbf{73.2 \pm 6.4}$	$\mathbf{72.5 \pm 5.1}$	$\mathbf{71.7 \pm 5.5}$	61.6 ± 4.9
Sub-Batch Norm.	83.4 ± 3.3	56.1 ± 1.5	51.7 ± 1.7	50.5 ± 2.2	47.0 ± 2.7	46.3 ± 1.5
No Norm.	94.8 ± 3.3	23.9 ± 2.6	23.4 ± 2.4	19.1 ± 1.9	17.8 ± 1.7	16.7 ± 1.4
Triplet (one-shot).	99.2 ± 1.1	63.3 ± 19.8	56.3 ± 15.0	41.4 ± 16.7	29.3 ± 16.4	24.5 ± 16.1
RPN (one-shot).	99.7 ± 0.6	73.1 ± 19.3	70.5 ± 18.6	62.7 ± 18.0	60.5 ± 17.2	$\mathbf{63.1 \pm 16.4}$
RPN (one-shot) (Median).	99.9	83.1	75.5	64.9	61.9	62.4
RPN-3-ensemble (one-shot) mode.	99.99	79.4 ± 14.0	77.7 ± 14.0	69.8 ± 15.0	65.7 ± 15.0	64.7 ± 15.0
RPN-3-ensemble (one-shot) dist.	99.99	$\mathbf{85.7 \pm 12.0}$	$\mathbf{86.9 \pm 13.0}$	$\mathbf{80.9 \pm 15.0}$	$\mathbf{76.8 \pm 16.0}$	$\mathbf{75.7 \pm 15.0}$
RPN-3-ensemble (one-shot) norm.	99.99	81.8 ± 13.0	82.6 ± 13.0	75.6 ± 14.0	71.2 ± 15.0	69.9 ± 15.0

Table 9.1: Extrapolation on the Translation Regime. We report results for TCN, Sub-Batch Norm, the best competitor approach to TCN in [108], and the baseline approach, No Norm, as reported in [108]. We can observe that the one-shot learning approaches are both improvements over competitor methods and the proto-net based approach matches extrapolation performance at the extreme edge. We also see that the small ensemble model significantly improves performance overall.

learning; this maintains our ability to directly compare with other recent work.

9.4.1 Model Structure

Our model is implemented using pytorch and follows the model of [116, 108]. We employ a feature encoder model, which consists of 4 convolution layers and 2 linear layers; each layer is ReLu activated. This is followed by a single RNN layer which combines the query tuple , and the support tuples. This forms the basis of the model structure. We then proceed to compute distances between the query example embedding and each embedding of the support set using the infinity norm distance.

We train 100 models on each of the VAEC training sets, and evaluate each of the models across the appropriate training and test sets.

9.5 Results

Our method utilises the same network architecture as proposed in [108], excluding their proposed temporal context normalisation or other normalisation layers. We show compelling performance through utilising an one-shot learning paradigm. We also show that using different loss methods does influence the results.

What is interesting is that we maintain the simple network architecture proposed and we do not include any special architectural tricks to improve extrapolation; extrapolation is achieved purely through the choice of training function and problem structure. Our experiments also highlight simple methods that boost performance through the use of ensembles. We also show an alternative training condition that can be used to boost model performance and explore how the direction of extrapolation and choice of training set can influence overall performance.

9.5.1 VAEC: Translation

In the Translation task condition our algorithmic approach improves over the many alternative architecture variants proposed in [108]. Our RPN model is competitive with the TCN approach in Region 6, the furthest and most difficult extrapolation task. The Triplet loss based approach struggles more; and while improving performance on near range tasks compared to Sub-Batch Norm, the best performing of the alternatives examined in [108], Triplet loss based models significantly degrade with further extrapolation.

A negative observation in our results is the wide margin in the standard deviation of the mean, which is significantly larger than the 6% reported. However this is largely attributable to a few significantly low outlier models which can be seen if we examine the median. This is further investigated through the generation of barplots, figure 9.1, of the accuracy achieved by the models on Region 2, binned in 5% intervals. We do this to demonstrate the higher median achieved by our network and to indicate the wide standard deviation is caused by a small number of significantly bad models. We attempted to address this through the introduction of a simple ensemble based approach. We take a random N samples of our existing models and evaluate them on the extrapolation tasks. We evaluate three methods of combining the ensemble predictions, listed here.

Majority Vote (Mode)

Distance Sum

Normalized Distance Sum

This significantly improves the average performance though it does not significantly improve the large standard deviation. Majority vote is the simplest application, simply taking the mode value of the model classifications. The Distance Sum approach sums the distance predictions from each model. We also evaluated a Normalized Distance Sum, max normalizing distances per-model, however, this approach was similar but slightly worse than the Distance Sum approach and we have not reported the values here.

When testing we set $N = 3$ for an ensemble of 3 classifiers. What we find is that the weighted distance metric provides the strongest performance. The variance of the accuracy is reduced, however it is still relatively large. This is skewed by a small

	Scale 1 (Training)	Scale 2	Scale 3	Scale 4	Scale 5	Scale 6
TCN.	98.8 ± 0.7	$\mathbf{77.8 \pm 1.8}$	$\mathbf{61.2 \pm 3.8}$	$\mathbf{54.4 \pm 3.3}$	$\mathbf{51.2 \pm 2.5}$	$\mathbf{48.7 \pm 2.2}$
Layer-Norm.(Reccurent)	100.0 ± 0.0	44.1 ± 5.0	28.1 ± 2.4	23.4 ± 1.6	20.2 ± 1.2	18.3 ± 0.8
No Norm.	94.4 ± 3.5	20.9 ± 1.0	17.6 ± 0.7	17.6 ± 0.6	16.7 ± 0.7	16.7 ± 0.6
RPN (one-shot).	99.66 ± 0.7	68.6 ± 8.9	51.2 ± 7.4	43.0 ± 6.3	38.0 ± 5.7	34.6 ± 6.1
Triplet (one-shot).	99.1 ± 1.1	56.6 ± 11.8	40.6 ± 6.8	32.1 ± 3.5	25.7 ± 3.3	21.4 ± 3.2
Triplet-Ensemble (one-shot) mode.	100.0 ± 0.1	58.1 ± 8.8	42.8 ± 4.7	33.8 ± 2.4	25.7 ± 2.3	20.6 ± 2.3
Triplet-Ensemble (one-shot) dist.	100.0 ± 0.0	61.3 ± 8.6	42.4 ± 4.4	32.4 ± 2.5	24.0 ± 2.8	18.8 ± 2.7
RPN-Ensemble (one-shot) mode.	100.0 ± 0.1	70.7 ± 6.7	52.8 ± 5.1	44.5 ± 3.8	40.0 ± 3.7	37.3 ± 4.1
RPN-Ensemble (one-shot) dist.	100.0 ± 0.0	75.0 ± 7.6	56.3 ± 6.2	47.7 ± 5.2	43.1 ± 4.8	40.4 ± 5.0

Table 9.2: Extrapolation on scale-based analogies. We show results from [108] for their proposed approach, TCN, their next top performing approach, Layer-Norm, and the baseline No Norm. The scale analogy problem is more difficult and the TCN approach performs best. We note however that our proposed approach, structuring the problem as a one-shot learning problem, performs significantly better than the next best approach from [108]

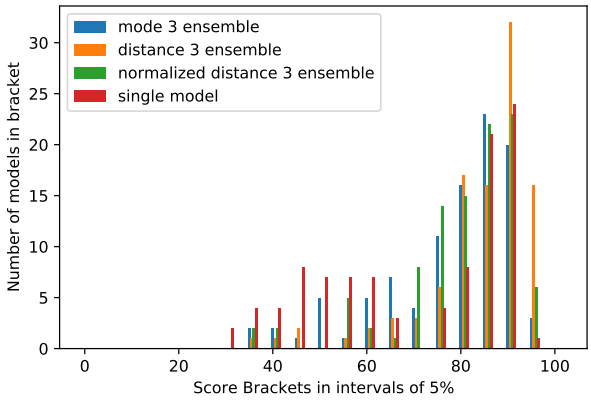


Figure 9.1: A plot showing the binned accuracy counts of each of our RPN model evaluations on test1. We see a large peak at the high accuracy end, the long tail of the distribution helps to explain the lower final mean value. Incorporating an ensemble based evaluation approach we can see an increase in the high scoring models

number of particularly low values with the median being significantly higher than the mean. We note that the variance of the TCN approach [108] is also large, so perhaps it is an artifact of the extrapolation task where not all models extrapolate, indeed such a problem has been noted in [10] where underspecification leads to wide differences between otherwise similar models.

9.5.2 VAEC:scale

Extrapolation to the scale-based task appears to be more difficult for the TCN [108] approach and this also appears to be the case for the one-shot models. The performance of the one-shot models remains promising, however, both models out perform the next best competitor, recurrent layer-norm in this case.

As noted previously our models do not include any normalisation approach and so we also compare to the basic model used in [108]. We can compare the difference

Train/Test Dataset	Region 1	Region 2	Region 3	Region 4	Region 5	Region 6	Average
Triplet (Region 1)	99.2 ± 1.1	63.3 ± 19.8	56.3 ± 15.0	41.4 ± 16.7	29.3 ± 16.4	24.5 ± 16.1	52.3
Triplet (Region 4)	38.1 ± 11.5	56.4 ± 17.4	79.2 ± 17.8	99.9 ± 0.4	78.0 ± 22.5	70.5 ± 26.3	70.4
Triplet (Region 6)	36.2 ± 10.1	61.1 ± 14.2	73.2 ± 12.9	82.0 ± 8.8	94.1 ± 6.3	$99.8 \pm 1.$	73.7
RPN (Region 1)	99.7 ± 0.6	73.1 ± 19.3	70.5 ± 18.6	62.7 ± 18.0	60.5 ± 17.2	63.1 ± 16.4	71.60
RPN (Region 4)	56.9 ± 11.1	78.1 ± 12.6	91.0 ± 10.5	100.0 ± 0.0	79.5 ± 20.0	76.0 ± 20.7	80.25
RPN (Region 6)	49.4 ± 7.8	75.0 ± 13.1	83.8 ± 11.8	89.47 ± 8.6	98.3 ± 3.1	100.0 ± 0.0	**82.50**

Table 9.3: Extrapolation From Varying Training sets on the Translation regime. We train both Triplet and RPN models on Region 1, 4 or 6 respectively. We evaluate performance on all other regions and observe that training on Region 6 leads to the best Extrapolation performance by a significant margin, 20% mean improvement for the triplet model, and that it also controls the the variance significantly more than when models are trained on the other sets.

between modelling approaches to achieve extrapolation. We clearly demonstrate the one-shot learning approach as being significantly more aligned with the task of extrapolation. The standard deviation is again relatively large compared to the TCN method, a problem that is likely in part related to the underspecification problem.

The ensemble approach to improving performance is also less effective on this task. Indeed for the triplet loss based methods extrapolation appears to be less successful in the distant regions. The triplet-loss still improves over the second best performing model used in [108], this may suggest that the modelling approach we use is a significant contributor to the performance.

9.5.3 Forward, Backward and In-between: Extrapolation Direction influences performance in VAEC

A question left uninvestigated in [108] is how the direction of extrapolation influences model performance. Is the given training set the best set to use, extrapolating forward, or should we use the final test set as the training set instead; perhaps the middle set would be best as it has 2 equally nearby sets and so should generalise better as it has twice the number of neighbours. We investigate this problem and find that some directions are indeed easier to extrapolate from than others.

We train 100 networks for both the RPN and Triplet loss methods on each of the translation and scale regions 1, 4 and 6. We observe in all the models that performance, as expected, declines the further from the training set it is. A natural assumption then might be that training a model on the middle test set would then lead to improved results since it has more close neighbours and fewer neighbours that are further away. Indeed we can see that this is the case and training on the middle set improves performance over training on the suggested training set.

We might expect that training on the furthest away test set, region 6 or scale 6,

Train/Test Dataset	Scale 1	Scale 2	Scale 3	Scale 4	Scale 5	Scale 6	Average
Triplet (Scale 1)	99.1 ± 1.1	56.6 ± 11.8	40.6 ± 6.8	32.1 ± 3.5	25.7 ± 3.3	21.4 ± 3.2	45.9
Triplet (Scale 4)	40.0 ± 3.2	62.5 ± 5.2	55.7 ± 8.0	99.4 ± 0.8	62.6 ± 8.5	57.6 ± 7.8	62.6
Triplet (Scale 6)	42.8 ± 3.7	70.7 ± 5.8	61.6 ± 7.5	91.4 ± 3.1	74.8 ± 8.6	98.9 ± 1.1	73.4
RPN (Scale 1)	99.7 ± 0.7	68.6 ± 8.9	51.2 ± 7.4	43.0 ± 6.3	38.0 ± 5.7	34.6 ± 6.1	55.8
RPN (Scale 4)	60.4 ± 8.0	87.01 ± 8.8	83.3 ± 10.3	99.9 ± 0.3	84.0 ± 7.2	74.3 ± 6.2	81.5
RPN (Scale 6)	63.3 ± 6.0	91.3 ± 4.2	80.6 ± 9.4	97.6 ± 1.6	89.9 ± 7.8	99.6 ± 1.0	**87.1**

Table 9.4: Extrapolation From Varying Training sets on the scale regime. We train the Triplet and RPN models on each of Scale 1,4 or 6 and evaluate on each other scale region. Training on scale 6 indicate the best performance for both models and a significant improvement of 30% mean accuracy over training on scale1. The scale 6 model also does not show a deterioration in performance further from the training set except for a significant dip for scale 1.

respectively, would lead to behaviour similar to training on the region 1 or scale 1 and performance would be worse than training in the centre of the dataset distribution; however, this is not the case. We show that the last test set provides the best extrapolation.

What we observe in table 9.3 is that the dataset used for training significantly impacts the overall results. The observation that training on Region 6 provides the best overall extrapolation is also unexpected. Indeed the strong performance of the RPN Region 6 model on all datasets except Region 1, combined with the signifcant dip in performance of both Region 1 models evaluated on Region 2 appears to indicate that Region 1 is less aligned with the dataset as a whole.

A similar effect is observed in the scale-based extrapolation regime in table 9.3. We again see that training on the Scale 6 task, the furthest scale task from the default training task, produces considerably better results on all the test sets. We also observe a comparatively significant dip in performance when evaluating on Scale 1. It is unclear why models trained on either Region 1 or Scale 1 appear to be less capable of generalising or why performance so significantly dips in comparison to the other evaluation tasks.

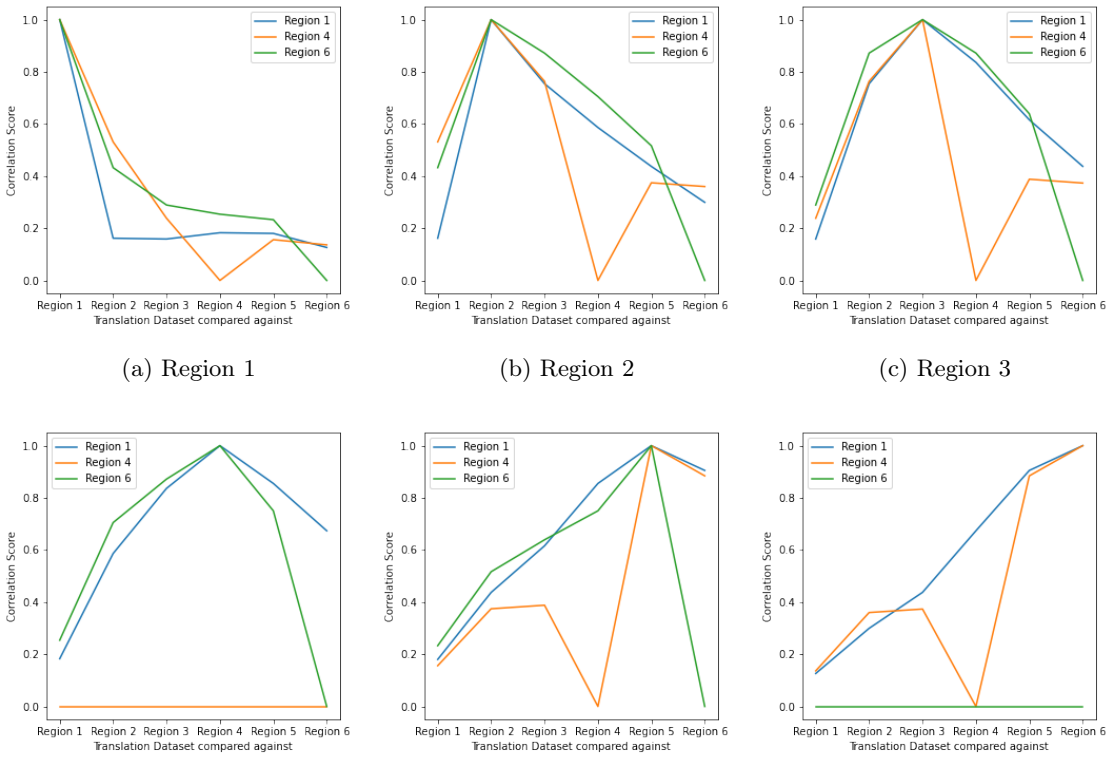
Our work shows that extrapolation is dependant on the region of space used for training and that large differences can emerge. Further work is necessary to understand the specifics of the VAEC dataset that impact the change in performance and whether or not larger values and scales more generally are better for extrapolation.

9.5.4 Taking advantage of correlation

The VAEC task is designed as an extrapolation challenge. The datasets all represent samples from the same underlying data distribution, sampled with disjoint sampling intervals. The aim of learning a function on any of the datasets is to learn one such

that it also works on the other datasets. This implies that one function should work on all the datasets and that performance on each dataset should correlate with each-other, and we would expect a positive correlation. Indeed we verify this in this chapter; further results can be viewed in Appendix C.

We explore our results by examining the correlation between performance on each pair of datasets. We observe in figure 9.2 very little correlation between performance on the training set and performance on any other set. Further investigation of the extrapolation effect provides evidence that supports the need for evaluating models on the extrapolation domains. We observe that the performance on the dataset used for training is not predictive or correlated with performance on the extrapolation tasks, this holds when we vary the training data from the default dataset. We do observe in figure 9.2 correlation between performance of a model on pairs of nearby test sets, not used during training. This highlights the fact that both tasks share an underlying solution function and that nearby datasets are more related than those further apart.

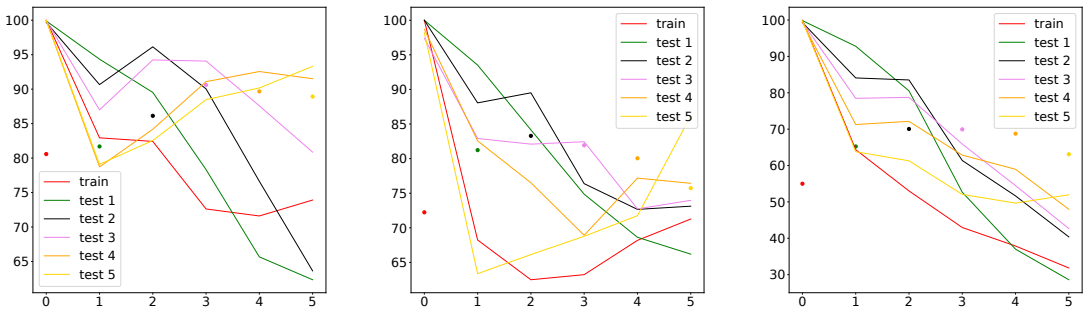


(a) Region 1 (b) Region 2 (c) Region 3

(d) Region 4 (e) Region 5 (f) Region 6

Figure 9.2: The correlation between performance on each dataset compared to each other dataset from a model trained on region 1, region 4 and region 6

An important detail highlighted by our experiments, comparing extrapolation



(a) ProtoNet (b) ProtoNet + TCN (c) Triplet

Figure 9.3: Here we show the effect of filtering models based on performance on a given test set. We show the performance across the range of tasks as well as the means indicated as the corresponding scatter plot. We see that in every case filtering on an extrapolation case offers improvement over selecting models based on training performance. We also observe that middle distance test sets offer the best overall improvement in performance.

from varying starting points, is the obvious disconnection of the default training set from the other test sets. We have observed significantly better performance when using an alternative training set as well as observing a significant dip in performance when evaluating on the default training set. These observations require further investigation in order to determine their origin, however, they do further motivate our call for adaptation problems that are more formally defined, such that we can measure the adaptation distance bridged by our model.

The observations we have made motivate a method to improve predictions. We can select a dataset from the distribution and use it as a test set. We can also choose a model selection method; for demonstration purposes we select the top N performing models on the selected test dataset and use only those models for prediction on the extrapolation problems; an alternative accuracy thresholding method is examined in the Appendix C.

The second variable in the filtering method is the extrapolation dataset used for thresholding. We show in our experiments, figure 9.3, that the dataset used to measure extrapolation improves the average result across all tasks, for each of our models. We also observe that the benefit is most pronounced on the task used for filtering as well as nearby datasets; as the datasets are more removed from the filter task the accuracy increase is reduced or even negative. This also leads to the observation that filtering in the middle of the range, in this case on Region 2 or Region 3, provides the strongest increase across all models and tasks.

9.6 Related Work

The work of [116] tackles the visual analogy problem using a very simple network combining all embeddings in the sequence with a recurrent network. They then introduce a training mechanism to improve performance. In [108] the same network structure and basic training style is used. They introduce a time based normalization module across the inputs to the recurrent network and demonstrate that other normalization techniques do not offer significant benefits to performance. Indeed, many normalization techniques appeared to offer little to no benefit.

Ravens Progressive matrices offer a related task that has drawn attention over the years. As a task used to measure human cognitive abilities it would seem an appropriate target task for AI systems. A dataset based on RPMs was recently proposed in [122] where a large number of matrices were algorithmically generated. The task requires the algorithm to identify a relationship among the given tiles and infer what the missing tile should be. This was tackled with a relation-net based model named Wren [122], a similar approach to our proposed method, though not focused on measuring extrapolation.

Towards building general AI the ARC task was proposed in [118]. The task is designed as a goal to further the development of more general AI. It is unclear where extrapolation falls within the definitions of generality provided, it is a known unknown to the developer, but since it does not fall into the interpolation range it would appear to be an unknown unknown to the algorithm. The large decline in performance observed in the baseline model appears to be indicative of this.

Considering one-shot and zero-shot learning algorithms as a class they are predisposed to this type of extrapolation, the problem formulation often involving samples not seen during training. The difficulty being that the datasets make it difficult to distinguish between interpolation and extrapolation. The analogy dataset of [116] provides a distinct train/test environment for extrapolation making it difficult to determine levels of extrapolation. The VAEC dataset provides problem space designed to test extrapolation and to provide a measure of distance from the training data, which allows for more fine-grained training and evaluation than the dataset proposed in [116].

Another consideration when examining the problem of extrapolation is the relation to the recent work on over-specification as a problem for machine learning

systems. This work shows how the same architecture trained on the same data and performing similarly on the given test set can generalise very differently to each-other on real world data. This is because the model is over-specified and many possible functions that can solve the given training problem exist. Without accounting for this, models may or may not learn the correct function and so may fail to generalise. This problem may be related to the large variances we observe when testing our model, or indeed the large variances shown by the TCN network. The general function can be learned from the data, to a greater or lesser degree, however other solutions, less general solutions are possible. Developing mechanisms to overcome this would likely prove useful.

9.7 Conclusion

Extrapolation is a complex problem for neural networks to solve and it is influenced by many factors from problem structure and modelling choices to the choice of training data. Our work has demonstrated how the training method can impact on the problem without external interventions in architecture and further shows how dataset choice significantly effects the dynamics of extrapolation.

We are encouraged by the effect this simple and easy to implement training mechanism has on the extrapolative power of the model. Our work demonstrates that the function to be solved greatly impacts the potential for extrapolation and that architectural changes are not the only mechanism by which we can make improvements.

The approach we present in this paper is well suited to the problem domain; however, it highlights that even simple neural networks can reasonably extrapolate beyond their training regions. As detailed previously this work focused on Proto-Net due to the simplicity of the learning structure which fits with keeping the network small and simple. This demonstrates that the formulation of the learning problem has significant impact on whether or not a model is capable of extrapolating. Further the specific segment of the data-distribution used for training was also demonstrated to be important in terms of improving model results. Our work thus provides useful insights which can be used to develop a deeper understanding of the extrapolation and how to build learning algorithms to solve it. We also provide an understanding of how filtering on a dataset known to be in the extrapolation range also allows for

the selection of significantly better models during training.

Extending this work, it would be interesting to adapt alternative one-shot learning mechanisms and alternative distance functions in order to observe the effect of the chosen function on extrapolative potential. It would also be prudent to examine the effectiveness of various existing pre-trained architectures in this setting. Analogical reasoning is used as an indicator of human fluid intelligence. A model trained tabula-rasa therefore may be limited in terms of visual features, which were not acquirable on the source task; this is particularly true of the relatively small sized VAEC dataset.

Further to this, a question we have not investigated and leave for future work is the corollary question that arises from our filtering approach. As the performance is more significantly maintained, where in the function does the error arise from? Have we learned a sufficiently general prediction layer, such that the difference in performance can be accounted for by un-tuned filters? Such a question may be probed via a variety of methods. Comparing learned filters, through the use of visualisation methods, from models trained on either end of the distribution, may provide insight. Further to that applying a domain adaptation approach, to align the image encodings may overcome the filter differences. Alternatively, it may be the case that our learned function is still not general enough and the prediction layer could be further refined. These questions are left for future work.

These analogical reasoning datasets are an interesting starting point for developing more generally intelligent machine learning models. In conjunction with other work such as RPM and ARC [118], new and interesting insights and models could be developed.

Chapter 10

Conclusion

The proliferation of machine learning applications makes understanding and improving the generalisation potential of neural networks an important challenge. Fundamental to applying models in risk averse contexts, such as healthcare, is understanding and verifying that models indeed can be made to generalise and to do so reliably. The assumption that they do, coupled with the difficulties of training large models from scratch, has led to a proliferation of algorithms trained from an initially transferred base model.

When labeled data for the designated task is scarce, developing methods which can utilise transferred knowledge is integral to the viability of large neural network models. This is an important motivation for research into domain adaptation and related transfer learning paradigms. Methods developed for domain adaptation significantly improve performance in these contexts. In our work we have contributed both methodology and understanding to the process.

Fundamental questions in relation to model evaluation and generalisation have also become apparent to the community. The problem of underspecification[10] illustrates the wide range of generalisation that can be expected from large models even when standard model evaluation would rate them as equivalent. Further work exploring how the problem affects the choice of pre-trained models from various libraries, especially in the context of transfer and adaptation, may provide further insight into which methodologies are more robust adaptation methods.

Our own work, alongside this, highlights the need for better developed evaluation frameworks. Evaluating the true generalisation of a learned model. This may require new datasets but also better documentation of dataset creation building on the proposals of Data-Sheets for datasets [107]. This couples with developing a better understanding of generalisation through the lens of extrapolation. Applying a model to related out-of-distribution data is the core task of domain adaptation. It is often difficult, however, to determine how related two datasets are during domain adaptation. Defining datasets such that we can evaluate such questions amongst others would undoubtedly create new methods and insights.

10.1 Our Contributions

Our work has developed a number of ideas based around the concept of the embedding and what it represents. We build on adaptation ideas of utilising a shared

representation space, which can then be passed to a shared classifier. Understanding and constructing that representation is thus an important scientific goal.

To achieve this goal we proposed a method of domain adaptation which we applied as a network initialisation model for transfer in reinforcement learning. We later examined the model in relation to its ability to extrapolate. Our AAEDDA method integrates the adversarial auto-encoder method into the adversarial domain adaptation framework. We perform adversarial alignment, utilising a domain discriminator, and impose a regularisation penalty on the learned representation in the form of an auto-encoder.

Having developed a model we also proposed and examined a number of variants to the BSP[11]. Our BSEP produces improved adaptation performance and our investigation exposed the importance of the entropy singular values of the embedding space. This elucidated the issue of compressed representations arising from adversarial training and further explains the success of the original BSP.

Finally the problem of extrapolation and its links to domain adaptation are examined. We first discuss the problem highlighting why it is important. We then examine various research directions that will lead to progress on the problem. We highlight the need for tests that more explicitly define the difference between data distributions. Such considerations are also important in light of the recent description of the underspecification [10] failure case.

Having described the problem of Extrapolation we propose a shape-counting dataset and illustrate how domain adpatation models can be applied to improve performance on an extrapolation task. We follow this work with an investigation of one-shot learning applied to the recently proposed VAEC [108] task. We show the potential for one-shot learning models in the extrapolation domain but also highlight issues around dataset design where some domains and directions appear more closely related than others.

Our work provides novel insights into the domain adaptation process. We discuss current issues and call for further development of a testing infrastructure designed to highlight where exactly adaptation models succeed and fail. We demonstrate that domain adaptation can be applied to the extrapolation problem. We further show that one-shot learning approaches can also be utilised and that direction of adaptation can significantly influence performance.

10.2 Future Directions

There is much work yet to be done to extend our understanding of domain adaptation. Domain adaptation is a rich field with many potential questions and applications. The influence of entropy on the embedding space and success of our batch spectral entropy penalty is promising. Examining the impact of adversarial learning on the embedding of models more generally may provide interesting insights. Furthermore, in the context of domain adaptation understanding the extent to which entropy of the embedding space is influenced by various methods may provide an explanation of part of the success of those methods.

The problems relating to understanding out-of-domain generalisation remain. What constitutes out-of-domain, and are all supposedly related out-of-domain tasks of equal difficulty? The datasets we have proposed show how varying directions of extrapolation can be easier or harder than others; however, larger and more complex datasets are needed to further investigate.

Appendix A

Few-shot and Zero-shot learning

A recent direction for general neural networks capable of learning transferable models and representations is that of few-shot learning and the related zero-shot learning paradigms. These are important approaches but are part of a distinct research direction separate from the main dissertation. A discussion of the approaches is included here for completeness and useful as an extended background for chapter 9.

A.1 Few-shot Learning

Few-shot learning represents a paradigm for neural network model learning where the goal is to learn a model such that given a small number of examples, as few as one in some cases, of the target data point correctly classify the unseen samples. The aim of this problem formulation is to produce models that can work well in data-poor environments such as when data might be expensive to acquire. A benefit of this approach is that once trained, a model can be used to classify a sample from a previously unseen class given just a single example of the class.

The solutions to this problem are related to domain adaptation in that there is a focus on representation learning; however, labeled data is available and the generating source remains relatively stable such that the model is not necessarily compensating for variance in the input data distribution.

The datasets commonly used to develop and test few-shot models are datasets such as omniglot and mini-imagenet may illustrate the problem that is being solved. Omniglot [119] is an MNIST-like dataset containing examples of characters from 50 languages. The goal ,usually, is to train an algorithm to correctly identify letters from an unseen test language given an exemplar. Mini-imagenet is a dataset composed of 100 classes each with 600 images, sampled from the larger imagenet dataset. Mini-imagenet comtains images that are larger and contain more complex content than the omniglot dataset. The data-set is usually split along class lines, with heldout classes being used to evaluate the few-shot learning approach.

A range of methods have been employed such as matching networks [123] and proto-nets [121].

The prototypical networks (proto-net) model offers an easy to understand algorithm that we use here to exemplify the desired goal. Many models perform a nearest neighbours approach to classification. In few shot learning the neighbours to compare with are known as the support set, which class exemplars and are la-

belled datapoints. A query example then is the unlabeled sample to be classified. In the proto-net case an embedding model is learned such that the euclidean-distance between the exemplar data-points and the query data-point is minimised. This is done through softmaxing the distance between the query data point and each of the exemplar data-points. This then allows the model to be trained end-to-end via gradient descent. When testing the model is able to classify previously unseen classes given a single labelled exemplar. Extensions to this idea can focus on replacing the euclidean distance with another metric or some learned function. This is the case in Relation-Net [124] where a Relation score, between 0 and 1 is learned for paired Support and query embeddings.

These approaches often fall under the umbrella of meta-learning or more colloquially learning-to-learn. To make the idea really concrete what we do with proto-net is to learn a model that learns a distance function, given a small sample of labeled data-points. The specific distance function used is fixed in the case of proto-net but the model can be extended.

An alternative approach to the problem is that proposed in MAML [36], a popular and powerful example of this discipline. Model Agnostic Meta Learning (MAML), is an approach that focuses on learning a model parameterisation that is easy to adapt via a gradient step, for a future target task.

A.2 Zero-shot Learning

Zero-shot learning seeks to solve a seemingly impossible problem, given an input from a never before seen class and correctly classify it. Humans have a capability for this type of task, we can identify examples of previously unseen classes with some degree of accuracy if not ease. While human learning and the knowledge we draw from is still much richer than that of machine learning algorithms, zero-shot learning offers some insight into how this type of learning can be achieved.

Zero-shot learning approaches often focus on classifying objects based on their attributes. A classifier can be built for objects based on their attributes and classifiers can be trained to detect various attributes in images. This setup allows for the classification of previously unseen data as demonstrated in [125]. This setup and paradigm allows for many extensions from designing methods to map from images to attributes to learning attribute embeddings [40].

Language models are also an important part of the zero-shot learning space. Word embeddings models such as word2vec[42] or glove [126] are useful as a method of representing attributes in vector fashion. Language models are also useful as methods for discerning attributes from text descriptions allowing for more accurate mapping from attributes to class.

A.3 Conclusions

Zero-shot and few-shot learning provide an interesting approach to the problem of solving new tasks. We can easily add a new class into the prediction problem given just a small number of samples or a suitable class-label embedding.

However these models do not necessarily integrate well with other network adaptation methods and it's not clear they solve the the domain-discrepancy issue that domain adaptation focuses on. Appropriately combining domain adaptation with meta-learning approaches could prove to be a fruitful avenue of research for future work.

Appendix B

Datasets

In this chapter we briefly introduce the datasets used throughout this document. The datasets are briefly discussed where they are used in the relevant chapters but are summarised here for ease of reference and completeness.

B.1 Atari

The Arcade Learning Environment (ALE)[78] provides a set of games of varying difficulty that has become a standard evaluation environment for many new RL algorithms. ALE provides a number of benefits for the RL community, providing a benchmark suite of games for algorithm development which are also difficult for humans. Success on many games helped kickstart development in deep-RL with works such as [20, 21] developing Deep Q-Learning and [66] A3C, which had a lot of early success on the ALE environment.

A recent evaluation of progress and various uasages and extensions to the ALE are discussed in [127].

In this work we access the Atari problem set through the OpenAI Gym [62] which provides a framework and wrapper for many common reinforcement learning tasks. OpenAI Gym reduces the determinism of the ALE through the introduction of frame-skipping where an agents actions are repeated for up to 4 steps at a time.

B.2 Digits datasets

A variety of digit recognition datasets exist in the literature. The task is to classify the images based on the digit in the frame. In the context of domain adaptation the task covers varying resolutions, written vs printed style and colour changes. In the following sections we briefly describe the make-up of the various digits datasets used.

B.2.1 USPS

USPS[128] is a small dataset consisting of 7,291 train and 2,007 test images. The images are 16*16 grayscale pixels containing a number in the range $[0-9]$. The task is to classify the digits as one of the 10 possible numbers.

B.2.2 MNIST

MNIST[129] is a subset of and was constructed from the NIST datasets. It consists of a training set of $60,000$ images and test set of $10,000$ images. The images consist of $28x28$ gray-scale images of hand written digits sampled from a mix of US high-school students and US Census workers.

The task is to classify the handwritten digits from the range $[0-9]$ and many algorithms have been applied to the task. It is now largely considered solved, however it is still useful for verifying learning algorithms and in the case of transfer and domain adaptation still provides some useful tasks.

B.2.3 SVHN

SVHN (Street View House Numbers) is a dataset of photographs of house numbers compiled from the Google StreetView application.The numbers are composed of multiple digits, forming the full house number, which adds a layer of difficulty to the task as digit classification as the numbers must first be segmented and then classified. The images are also in full colour rather than the black and white of MNIST and USPS.

The dataset was introduced in [130] where the problem is to correctly predict each number in the sequence of the image.

In the context of domain adaptation the dataset is used only for digit recognition and the sequential identification of numbers is left as a separate problem. An already segmented version of the dataset exists with each of the numbers in a sequence centered, though sometimes with other numbers overlapping. The dataset is used as a number prediction task that is more complex than MNIST and USPS, containing numbers of various colours on various colour backgrounds.

B.3 Office-31

Office-31 was introduced in [131]. It consists of images gathered from three settings, the web (www.Amazon.com), photographs taken with a DSLR camera and photos taken using a webcam. The images cover 31 categories of various office equipment such as keyboards, computer monitors, knapsacks and office chairs. There is also a large size imbalance between the three domains, with the most data being gathered

in the web setting.

B.4 Office-Home

The Office-Home dataset [132] contains images sampled from the web from 4 different categories, Clipart, Artistic images, Product images and Real World images. The images are sampled from 65 classes from office and home environments. The dataset represents a challenging domain adaptation task. The dataset consists of approximately $15,500$ images total.

B.5 Extrapolation

B.5.1 Visual Analogy Extrapolation Challenge

The Visual Analogy Extrapolation Challenge (VAEC)[108] dataset was recently proposed. It consists 2 extrapolation tasks across 6 domains of distance from the source.

Each image in the dataset consists of a green square on a grey background (R= 0.5,G= 0.5,B= 0.5) with 4 degrees of variation tiled over a linear range of 42 discrete levels, with the 4 degree of variation being

brightness spanning the range $brightness \in [0.41.0]$,

size spanning the range $width \in [385]$,

x-location spanning the range $center \in [4384]$

y-location spanning the range $center \in [4384]$

The dataset is built on the problem of solving relationships between images. The problem consists of 3 context images as part of the relationship $A : B :: C :?$ where a number of foil objects are provided and the objective is to select the correct foil object D which correctly completes the relation $A : B :: C : D$ The images in the foil take the same values as those in the context along the irrelevant dimensions of the problem but take different values along the relevant dimension.

The task is made difficult by the nature of the dataset. Each dataset is sampled from a particular generalization regime. Two generalization regimes are created. The first regimes tests for invariance to translation of the data along each of the 4 dimensions. The dimensions are broken up into regions along the diagonal with 7

values for each dimension, from the possible 42, leading to 6 regions. Each analogy dataset contains $19,040$ analogy problems.

The second extrapolation regime is test for invariances to scale. There are 6 scale datasets. Scale1 is sampled from the range$[1-7]$ and the remaining datasets are constructed from samples from scale1 multiplied by 2 to 6 to produce each of the remaining datasets.

B.5.2 Shapes Extrapolation

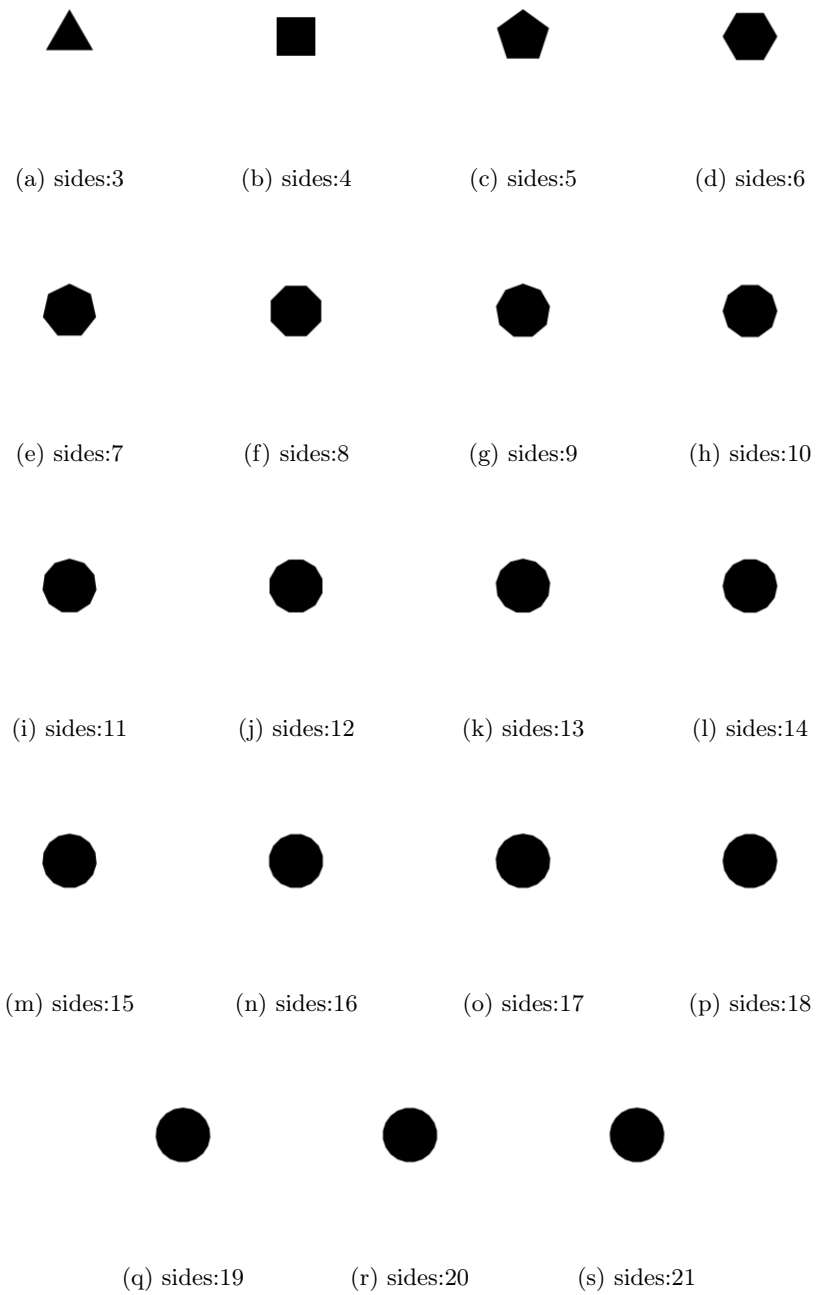
In chapter 8 we introduced two shape counting datasets for evaluating extrapolation and interpolation. The datasets are designed to vary a specific property of the shape which transitions along a range. By training on a subset of the the range and evaluating on the remainder of the range we can evaluate whether the function learned a general recognizer for a given shape or not. Further, such a dataset also allows for the evaluation of transfer learning and domain adaptation algorithms across a more defined gap than is normally used.

Our two datasets consist of a polygon dataset, where the number of edges a shape has is varied from 3 to 21; transitioning from triangles to circles. The ellipse dataset consists of ellipses where the eccentricity is varied between 0 and 1 producing a range of ellipses.

As we demonstrate in chapter 8, the datasets are easily solvable given the full range of data. Given only a subset, however, demonstrates the expected gradual decline in accuracy when applied on out-of-distribution data. We also demonstrate the potential positives and negatives of applying domain adaptation and further support our call for investigation of explicit extrapolation and interpolation behaviour in the context of transfer learning and domain adaptation.

Polygon Dataset

In this section we present the full range of potential polygon dataset.



(a) sides:3 (b) sides:4 (c) sides:5 (d) sides:6

(e) sides:7 (f) sides:8 (g) sides:9 (h) sides:10

(i) sides:11 (j) sides:12 (k) sides:13 (l) sides:14

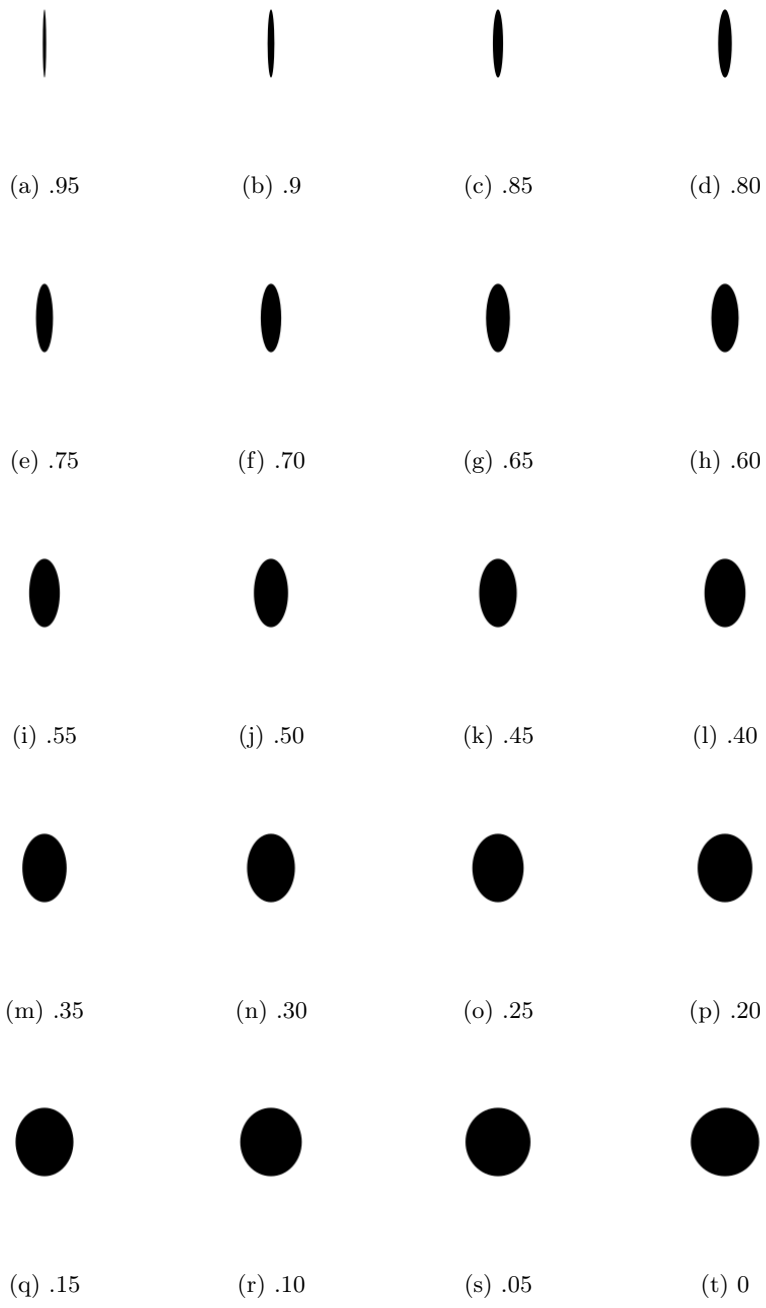
(m) sides:15 (n) sides:16 (o) sides:17 (p) sides:18

(q) sides:19 (r) sides:20 (s) sides:21

Figure B.1: An example of the change in polygon edges from 3 to 21

Ellipses Dataset

In this section we show the full range of the Ellipses dataset.



(a) .95 (b) .9 (c) .85 (d) .80

(e) .75 (f) .70 (g) .65 (h) .60

(i) .55 (j) .50 (k) .45 (l) .40

(m) .35 (n) .30 (o) .25 (p) .20

(q) .15 (r) .10 (s) .05 (t) 0

Figure B.2: An example of the change in ellipse shape by eccentricity from 0.95 to 0.0

Appendix C

One-Shot Extrapolation: Examining Dataset Correlations

In chapter 9 we proposed a one-shot learning approach is capable of solving an extrapolative analogical reasoning task. We demonstrated that selecting a test set and choosing the models that evaluate best on that set we can improve performance across the range of task. Such an approach is informative, however, it introduces an element of luck and with the large standard deviation from the mean observed an alternative may be better for practical scenarios.

A useful fact is that for the true function we will be able to compute the correct result, even on unseen data from an extrapolated region of sample space. When training a machine learning model with current methods it is not expected that the model will extrapolate. However some models may be more aligned to the true function than others and if that is the case we can expect those models to perform better across all the extrapolation tasks.

We verify this assumption by investigating the correlation in performance of models between each pair of test sets. This verification process also highlights the relationships between different segments of the VAEC dataset. This sheds light on some of the performance differences we observe depending on direction of extrapolation. With these experiments we also verify the need for out of distribution test sets and evaluation infrastructure as proposed in chapter 7

C.1 Accuracy Filters to improve overall performance

As we showed in the chapter 9 filtering by the top 10 performing models on a given test set the overall model accuracy can be greatly increased across the extrapolation task. Here we more fully explore this method. In practice we may not want to train many models to identify the top performing models. Introducing an accuracy threshold can be similarly effective. We show how filtering across test-sets and accuracy thresholds can improve performance significantly

When training models on the training set, we observe generally strong improvement. For mid range data extrapolation performance in the forward direction appears to not contribute or even negatively contribute to performance in the backwards direction. A generally more positive effect appears to be observed if the model generalises backwards. Filtering in the reverse direction from a model trained on region 6 appears to have a significantly smaller impact; this may in part be related to the overall stronger performance of models trained on region 6.

Another interesting observation is the significant difference in performance, for models trained on either region 4 or region 6, evaluated on the region 1 dataset compared to the other datasets. This might suggest there is something fundamentally more difficult or distant about the region 1 set. Further evidence for this is the comparison between models in sub-figure a, where a significant gap exists between training and test accuracy. Our proposed filtering method generally improves all other tests for models trained on range 1, however, without it such models appear to be significantly less aligned with the range of test tasks than their counterparts. This same observation can be observed in the other graphs where the training set used does not degrade performance on nearby datasets nearly as much as in the case of training on the region 1 set.

Other useful observations are the that interpolation as well as extrapolation are generally positively effected by thresholding on just a single test set. Interpolation is observed by applying filter 6 to model 1. The entire range of test sets in between also see their accuracy increase. This suggests that filtering selects for more general models overall and that it is difficult to find a model that performs well on both edges of a distribution but not on the data in between. The further the filter task is from the training task the better the interpolation performance with the caveat that the datasets remain aligned. Region 1 indicates this through the significantly reduced response we observe on the other regions when filtering on region 1.

Extrapolation is also observed, when applying a filter on any test region, the model generally demonstrates a positive response beyond and outside of the range of the test region. Too strong a filter can be detrimental for test regions further out of range but is less harmful for nearby regions.

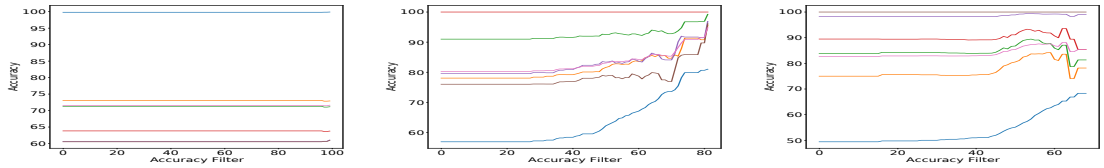
C.2 conclusion

Extrapolation is a complex problem, the method and dataset used heavily influences the expected extrapolation performance. Correlations between extrapolation domains exist however not all directions are equally correlated and some models and directions are much closer than others. This observation helps to solidify our call for further development of datasets specifically designed to evaluate extrapolation. Methods for the classification of transfer learning tasks and datasets would also be useful, providing a way to measure the degree of extrapolation or separation

that exists between datasets. Developing extrapolation conditions and a method for classifying the distance between domains will provide better ways to compare and evaluate algorithms as well as verify the learned function and the models suitability for deployment.

Despite the internal issues of varying correlations we have identified, we show that using any extrapolation dataset as a model filter increases performance across the range of tasks. The accuracy filtering approach we present here is a more practical method than selecting the top N models as accuracy thresholds appear to significantly boost performance across tasks. Such an approach may also be useful in instigating a new training regime with an early stopping condition or altering gradient steps.

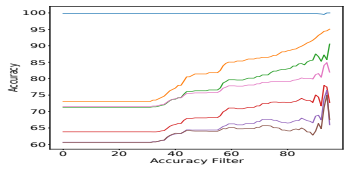
Our experiments and examinations here serve to verify the need for test data that takes account of an extrapolation context. Furthermore, when considering the transfer learning problem and domain adaptation problem, the need to validate the difficulty of the adaptation and the impact that may have in relation to verifying model performances and differences between models becomes apparent. The experiments presented here will, we hope, draw attention to the problem and attract further work developing new datasets which account for the problem of generalization.



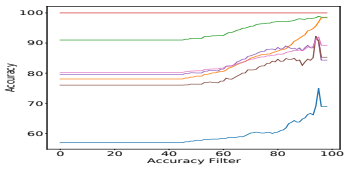
(a) Filter 1 Model 1

(b) Filter 1 Model 4

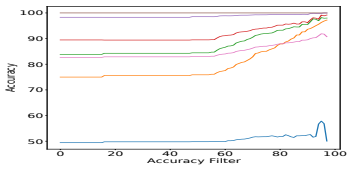
(c) Filter 1 Model 6



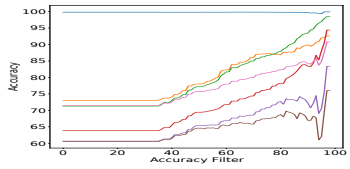
(d) Filter 2 Model 1



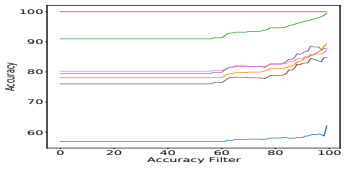
(e) Filter 2 Model 4



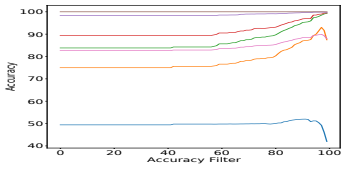
(f) Filter 2 Model 6



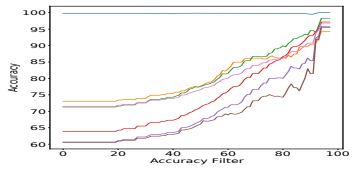
(g) Filter 3 Model 1



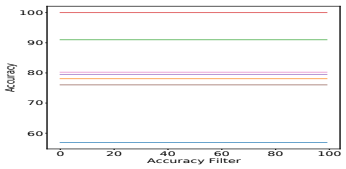
(h) Filter 3 Model 4



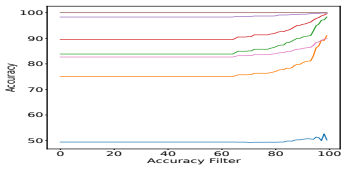
(i) Filter 3 Model 6



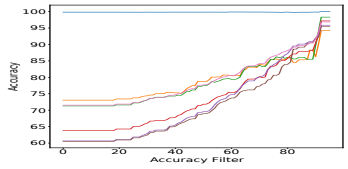
(j) Filter 4 Model 1



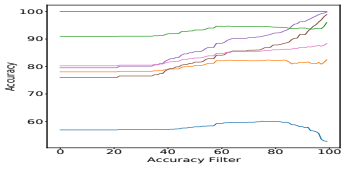
(k) Filter 4 Model 4



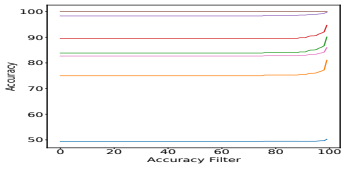
(l) Filter 4 Model 6



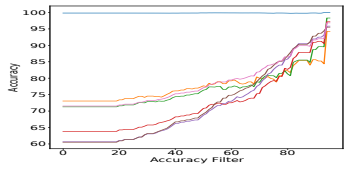
(m) Filter 5 Model 1



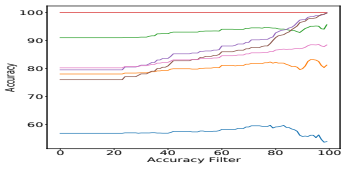
(n) Filter 5 Model 4



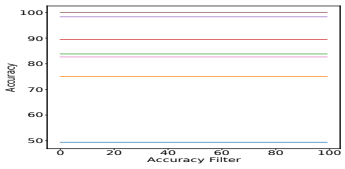
(o) Filter 5 Model 6



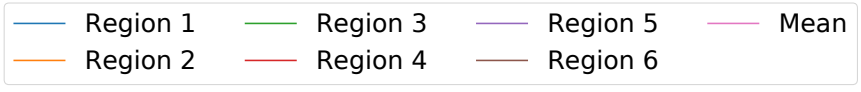
(p) Filter 6 Model 1



(q) Filter 6 Model 4



(r) Filter 6 Model 6



(s) Legend

Figure C.1: Here we elucidate the effectiveness of utilising an accuracy filter applied to the named Region, 1..6, across a range of accuracy thresholds. Each column represents models trained using one of the training datasets used, Region $1, 4, 6$ respectively. Each row indicates the dataset used to filter models based on accuracy achieved on that dataset. Accuracy is plot on the Y-axis and the filter accuracy threshold is plot on the X-axis. We can observe in figures a,k and r, that filtering with the training data is ineffective at improving overall accuracy. Filtering on datasets not used during training is generally helpful producing large performance boost in extrapolation, both in the direction of the filter and away from it as well as improving performance in the interpolation region.

Bibliography

[1] Bruce G Buchanan. A (very) brief history of artificial intelligence. Ai Magazine, 26(4):53–53, 2005.

[2] Daniel W Otter, Julian R Medina, and Jugal K Kalita. A survey of the usages of deep learning for natural language processing. IEEE Transactions on Neural Networks and Learning Systems, 2020.

[3] Bryan Lim and Stefan Zohren. Time series forecasting with deep learning: A survey. arXiv preprint arXiv:2004.13408, 2020.

[4] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, pages 1–21, 2020.

[5] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. Journal of Field Robotics, 37(3):362–386, 2020.

[6] Abdul Mueed Hafiz and Ghulam Mohiuddin Bhat. A survey of deep learning techniques for medical diagnosis. In Information and Communication Technology for Sustainable Development, pages 161–170. Springer, 2020.

[7] Hongming Chen, Ola Engkvist, Yinhai Wang, Marcus Olivecrona, and Thomas Blaschke. The rise of deep learning in drug discovery. Drug discovery today, 23(6):1241–1250, 2018.

[8] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. nature, 529(7587):484–489, 2016.

[9] Will Douglas Heaven. Our weird behavior during the pandemic is messing with ai models, 2020.

[10] Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. Underspecification presents challenges for credibility in modern machine learning. arXiv preprint arXiv:2011.03395, 2020.

[11] Xinyang Chen, Sinan Wang, Mingsheng Long, and Jianmin Wang. Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation. In International Conference on Machine Learning, pages 1081–1090, 2019.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[13] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499, 2016.

[14] Lex Flagel, Yaniv Brandvain, and Daniel R Schrider. The unreasonable effectiveness of convolutional neural networks in population genetic inference. Molecular biology and evolution, 36(2):220–238, 2019.

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680, 2014.

[16] Jiirgen Schmidhuber. Making the world differentiable: On using self-supervised fully recurrent n eu al networks for dynamic reinforcement learning and planning in non-stationary environm nts. 1990.

[17] Jürgen Schmidhuber. Generative adversarial networks are special cases of artificial curiosity (1990) and also closely related to predictability minimization (1991). Neural Networks, 2020.

[18] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. arXiv preprint arXiv:1511.05644, 2015.

[19] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. MIT press, 2018.

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. Nature, 518(7540):529, 2015.

[22] Long-Ji Lin. Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993.

[23] Alessandro Lazaric. Transfer in reinforcement learning: a framework and a survey. In Reinforcement Learning, pages 143–173. Springer, 2012.

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

[25] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1–9, 2015.

[26] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2818–2826, 2016.

[27] Uri Hasson, Samuel A Nastase, and Ariel Goldstein. Direct fit to nature: An evolutionary perspective on biological and artificial neural networks. Neuron, 105(3):416–434, 2020.

[28] Lorien Y Pratt, Jack Mostow, Candace A Kamm, and Ace A Kamm. Direct transfer of learned information among neural networks. In Aaai, volume 91, pages 584–589, 1991.

[29] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In Advances in neural information processing systems, pages 3320–3328, 2014.

[30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In CVPR09, 2009.

[31] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

[32] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. OpenAI blog, 1(8):9, 2019.

[33] Yu Zhang and Qiang Yang. A survey on multi-task learning. arXiv preprint arXiv:1707.08114, 2017.

[34] Michael Crawshaw. Multi-task learning with deep neural networks: A survey. arXiv preprint arXiv:2009.09796, 2020.

[35] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. arXiv preprint arXiv:1904.04232, 2019.

[36] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. arXiv preprint arXiv:1703.03400, 2017.

[37] Indrė Žliobaitė, Mykola Pechenizkiy, and Joao Gama. An overview of concept drift applications. In Big data analysis: new algorithms for a new society, pages 91–114. Springer, 2016.

[38] Patrick Robotham. Why your models might not work after covid-19. `https://medium.com/eliiza-ai/why-your-models-might-not-work-after-covid-19-a00509e4920b`. Accessed: 2020-08-04.

[39] Wouter Marco Kouw and Marco Loog. A review of domain adaptation without target labels. IEEE transactions on pattern analysis and machine intelligence, 2019.

[40] Yongqin Xian, Bernt Schiele, and Zeynep Akata. Zero-shot learning-the good, the bad and the ugly. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4582–4591, 2017.

[41] Wei Wang, Vincent W Zheng, Han Yu, and Chunyan Miao. A survey of zero-shot learning: Settings, methods, and applications. ACM Transactions on Intelligent Systems and Technology (TIST), 10(2):1–37, 2019.

[42] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013.

[43] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems, pages 3111–3119, 2013.

[44] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. arXiv preprint arXiv:1611.02200, 2016.

[45] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proceedings of the IEEE international conference on computer vision, pages 2223–2232, 2017.

[46] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In International conference on machine learning, pages 1989–1998. PMLR, 2018.

[47] Karsten M Borgwardt, Arthur Gretton, Malte J Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J Smola. Integrating structured biological data by kernel maximum mean discrepancy. Bioinformatics, 22(14):e49–e57, 2006.

[48] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In International conference on machine learning, pages 97–105. PMLR, 2015.

[49] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. In Advances in neural information processing systems, pages 136–144, 2016.

[50] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In International conference on machine learning, pages 2208–2217. PMLR, 2017.

[51] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. arXiv preprint arXiv:1511.05547, 2015.

[52] Kihyuk Sohn, Wenling Shang, Xiang Yu, and Manmohan Chandraker. Unsupervised domain adaptation for distance metric learning. In International Conference on Learning Representations, 2018.

[53] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Michael I Jordan. Partial transfer learning with selective adversarial networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2724–2732, 2018.

[54] Zhangjie Cao, Lijia Ma, Mingsheng Long, and Jianmin Wang. Partial adversarial domain adaptation. In Proceedings of the European Conference on Computer Vision (ECCV), pages 135–150, 2018.

[55] Jing Zhang, Zewei Ding, Wanqing Li, and Philip Ogunbona. Importance weighted adversarial nets for partial domain adaptation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 8156–8164, 2018.

[56] Pau Panareda Busto and Juergen Gall. Open set domain adaptation. In Proceedings of the IEEE International Conference on Computer Vision, pages 754–763, 2017.

[57] Kuniaki Saito, Shohei Yamamoto, Yoshitaka Ushiku, and Tatsuya Harada. Open set domain adaptation by backpropagation. In Proceedings of the European Conference on Computer Vision (ECCV), pages 153–168, 2018.

[58] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971, 2015.

[59] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In International conference on machine learning, pages 1889–1897, 2015.

[60] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. Journal of Machine Learning Research, 10(7), 2009.

[61] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. Machine learning, 79(1-2):151–175, 2010.

[62] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[63] Abhishek Gupta, Coline Devin, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. arXiv preprint arXiv:1703.02949, 2017.

[64] a2c announcement. `https://blog.openai.com/baselines-acktr-a2c/`, 2018. accessed 09/03/2018.

[65] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. arXiv preprint arXiv:1611.05763, 2016.

[66] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In International Conference on Machine Learning, pages 1928–1937, 2016.

[67] David Pfau and Oriol Vinyals. Connecting generative adversarial networks and actor-critic methods. arXiv preprint arXiv:1610.01945, 2016.

[68] Sergio Valcarcel Macua, Aleksi Tukiainen, Daniel Garcıa-Ocana Hernandez, David Baldazo, Enrique Munoz de Cote, and Zazo Santiago. Diff-dac: Distributed actor-critic for multitask deep reinforcement learning. arXiv preprint arXiv:1710.10363, 2017.

[69] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In Computer Vision and Pattern Recognition (CVPR), volume 1, page 4, 2017.

[70] Fuzhen Zhuang, Xiaohu Cheng, Ping Luo, Sinno Jialin Pan, and Qing He. Supervised representation learning: Transfer learning with deep autoencoders. In IJCAI, pages 4119–4125, 2015.

[71] Haitham Bou Ammar, Eric Eaton, Paul Ruvolo, and Matthew E Taylor. Unsupervised cross-domain transfer in policy gradient reinforcement learning via manifold alignment. In Proc. of AAAI, 2015.

[72] Thomas Carr, Maria Chli, and George Vogiatzis. Domain adaptation for reinforcement learning on the atari. In Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, pages 1859–1861. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

[73] Abhishek Gupta, Coline Devin, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. 03 2017.

[74] Sascha Lange and Martin Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In Neural Networks (IJCNN), The 2010 International Joint Conference on, pages 1–8. IEEE, 2010.

[75] Herke van Hoof, Nutan Chen, Maximilian Karl, Patrick van der Smagt, and Jan Peters. Stable reinforcement learning with autoencoders for tactile and visual data. In Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on, pages 3928–3934. IEEE, 2016.

[76] Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Deep spatial autoencoders for visuomotor learning. In Robotics and Automation (ICRA), 2016 IEEE International Conference on, pages 512–519. IEEE, 2016.

[77] Gabriel Barth-Maron. Learning deep state representations with convolutional autoencoders. PhD thesis, Master's thesis, Brown University, 2015.

[78] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research, 47:253–279, jun 2013.

[79] Shu-Hsuan Hsu, I Shen, Bing-Yu Chen, et al. Transferring deep reinforcement learning with adversarial objective and augmentation. arXiv preprint arXiv:1809.00770, 2018.

[80] Han Zhao, Junjie Hu, Zhenyao Zhu, Adam Coates, and Geoffrey J Gordon. Deep generative and discriminative domain adaptation. In AAMAS, pages 2315–2317, 2019.

[81] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

[82] Qi Dou, Cheng Ouyang, Cheng Chen, Hao Chen, and Pheng-Ann Heng. Unsupervised cross-modality domain adaptation of convnets for biomedical image segmentations with adversarial loss. arXiv preprint arXiv:1804.10916, 2018.

[83] Markus Wulfmeier, Ingmar Posner, and Pieter Abbeel. Mutual alignment transfer learning. arXiv preprint arXiv:1707.07907, 2017.

[84] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. arXiv preprint arXiv:1511.06342, 2015.

[85] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. arXiv preprint arXiv:1606.04671, 2016.

[86] Jonathan Schwarz, Jelena Luketina, Wojciech M Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. arXiv preprint arXiv:1805.06370, 2018.

[87] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. arXiv preprint arXiv:1409.7495, 2014.

[88] Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, David Balduzzi, and Wen Li. Deep reconstruction-classification networks for unsupervised domain adaptation. In European Conference on Computer Vision, pages 597–613. Springer, 2016.

[89] Tadanobu Inoue, Subhajit Chaudhury, Giovanni De Magistris, and Sakyasingha Dasgupta. Transfer learning from synthetic to real images using variational autoencoders for robotic applications. arXiv preprint arXiv:1709.06762, 2017.

[90] Manisha Padala, Debojit Das, and Sujit Gujar. Effect of input noise dimension in gans. arXiv preprint arXiv:2004.06882, 2020.

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

[92] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 7167–7176, 2017.

[93] Martijn J Post, Peter Van Der Putten, and Jan N Van Rijn. Does feature selection improve classification? a large scale experiment in openml. In International Symposium on Intelligent Data Analysis, pages 158–170. Springer, 2016.

[94] Ching-Yao Chuang, Antonio Torralba, and Stefanie Jegelka. The role of embedding complexity in domain-invariant representations. arXiv preprint arXiv:1910.05804, 2019.

[95] Kaichao You, Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Universal domain adaptation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2720–2729, 2019.

[96] Gabriel Cavallari, Leonardo Ribeiro, and Moacir Ponti. Unsupervised representation learning using convolutional and stacked auto-encoders: a domain and cross-domain feature space analysis. In 2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), pages 440–446. IEEE, 2018.

[97] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4700–4708, 2017.

[98] Karsten Roth, Timo Milbich, Samarth Sinha, Prateek Gupta, Bjoern Ommer, and Joseph Paul Cohen. Revisiting training strategies and generalization performance in deep metric learning. arXiv preprint arXiv:2002.08473, 2020.

[99] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. The Journal of Machine Learning Research, 17(1):2096–2030, 2016.

[100] Xinyang Chen, Sinan Wang, Bo Fu, Mingsheng Long, and Jianmin Wang. Catastrophic forgetting meets negative transfer: Batch spectral shrinkage for safe transfer learning. In Advances in Neural Information Processing Systems, pages 1906–1916, 2019.

[101] Bingyu Liu, Zhen Zhao, Zhenpeng Li, Jianan Jiang, Yuhong Guo, Haifeng Shen, and Jieping Ye. Feature transformation ensemble model with batch spectral regularization for cross-domain few-shot classification. arXiv preprint arXiv:2005.08463, 2020.

[102] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In Advances in Neural Information Processing Systems, pages 1640–1650, 2018.

[103] Lin Zuo, Mengmeng Jing, Jingjing Li, Lei Zhu, Ke Lu, and Yang Yang. Challenging tough samples in unsupervised domain adaptation. Pattern Recognition, page 107540, 2020.

[104] Vinod Kumar Kurmi and Vinay P Namboodiri. Looking back at labels: A class based domain adaptation technique. In 2019 International Joint Conference on Neural Networks (IJCNN), pages 1–8. IEEE, 2019.

[105] Randall Balestriero, Jerome Pesenti, and Yann LeCun. Learning in high dimension always amounts to extrapolation. arXiv preprint arXiv:2110.09485, 2021.

[106] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In Conference on fairness, accountability and transparency, pages 77–91, 2018.

[107] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. Datasheets for datasets. arXiv preprint arXiv:1803.09010, 2018.

[108] Taylor W Webb, Zachary Dulberg, Steven M Frankland, Alexander A Petrov, Randall C O'Reilly, and Jonathan D Cohen. Learning representations that support extrapolation. arXiv preprint arXiv:2007.05059, 2020.

[109] Chelsea Finn and Sergey Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. arXiv preprint arXiv:1710.11622, 2017.

[110] Georg Martius and Christoph H Lampert. Extrapolation and learning equations. arXiv preprint arXiv:1610.02995, 2016.

[111] Subham S Sahoo, Christoph H Lampert, and Georg Martius. Learning equations for extrapolation and control. arXiv preprint arXiv:1806.07259, 2018.

[112] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, page 41–48, New York, NY, USA, 2009. Association for Computing Machinery.

[113] Sanmit Narvekar. Curriculum learning in reinforcement learning. In IJCAI, pages 5195–5196, 2017.

[114] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. Journal of Big Data, 6(1):60, 2019.

[115] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In Advances in neural information processing systems, pages 4349–4357, 2016.

[116] Felix Hill, Adam Santoro, David GT Barrett, Ari S Morcos, and Timothy Lillicrap. Learning to make analogies by contrasting abstract relational structure. arXiv preprint arXiv:1902.00120, 2019.

[117] Cheng Ouyang, Konstantinos Kamnitsas, Carlo Biffi, Jinming Duan, and Daniel Rueckert. Data efficient unsupervised domain adaptation for cross-modality image segmentation. In International Conference on Medical Image Computing and Computer-Assisted Intervention, pages 669–677. Springer, 2019.

[118] François Chollet. On the measure of intelligence. arXiv preprint arXiv:1911.01547, 2019.

[119] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. Science, 350(6266):1332–1338, 2015.

[120] Shruti Jadon. An overview of deep learning architectures in few-shot learning domain. arXiv preprint arXiv:2008.06365, 2020.

[121] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In Advances in neural information processing systems, pages 4077–4087, 2017.

[122] Adam Santoro, Felix Hill, David Barrett, Ari Morcos, and Timothy Lillicrap. Measuring abstract reasoning in neural networks. In International Conference on Machine Learning, pages 4477–4486, 2018.

[123] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In Advances in neural information processing systems, pages 3630–3638, 2016.

[124] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1199–1208, 2018.

[125] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. IEEE transactions on pattern analysis and machine intelligence, 36(3):453–465, 2013.

[126] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pages 1532–1543, 2014.

[127] Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew J. Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. Journal of Artificial Intelligence Research, 61:523–562, 2018.

[128] J. J. Hull. A database for handwritten text recognition research. IEEE Transactions on Pattern Analysis and Machine Intelligence, 16(5):550–554, 1994.

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

[130] Ian J Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. arXiv preprint arXiv:1312.6082, 2013.

[131] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, Computer Vision – ECCV 2010, pages 213–226, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[132] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In (IEEE) Conference on Computer Vision and Pattern Recognition (CVPR), 2017.