

An Evaluation of Optimisation Algorithms for Image Augmentation Strategy Discovery

Daniel Harborne, Author

Prof. David Marshall, Supervisor

Dr. Kirill Sidorov, Moderator

School of Computer Science and Informatics



May 2019

Abstract

In this project, an evaluation of optimisation algorithms has been conducted in the context of image augmentation policy discovery for the purpose of training Deep Learning image classification models. Existing literature has established that image augmentation during the training of Deep Learning models can provide significant increases to performance. However, little work has been conducted to establish a method for automating the discovery of a optimal policy with the majority of prior work resorting to manual trial and error. This work builds upon a recently published effort to formalise and tackle the problem of automatic augmentation policy discovery and offers a number of engineering and scientific contributions. As an engineering contribution, a Python tool was created to facilitate job submission and management on the Welsh Super Computing resources. This tool has now been adopted by researchers at one of the University's research institutes. Scientifically, this work led to a research paper submission containing two main contributions. Firstly, an update to the formalisation of the augmentation policy search space was provided with the aim of providing further insight for approaching this problem domain. Secondly, the paper provides an evaluation of optimisation algorithms for the problem of automated augmentation policy discover and in doing so, has discovered a policy which provides improvement over the state-of-the-art.

1 Introduction

The findings from this study formed the basis of a paper submitted to the British Machine Vision Conference(BMVC) 2019 ¹. The submitted paper is included as section 3 and its inclusion provides an introduction and background that targets the work in the paper specifically, along with a presentation of its results and scientific contributions. Prior to the paper, in Section 2, a broader background is provided outlining some of the fundamental concepts that underpin this work. Following the paper, Section 4 of this report covers the results and a discussion relating to the Particle Swarm Optimisation algorithm(PSO), which was removed from the final draft of the paper (the reasoning behind this removal are also included within the section). Section 5 relates to the main engineering contribution — the ARCCA Python tool. In section 6, a reflection on the learning achieved throughout the work is provided. Finally the report concludes in Section 7.

2 Background

2.1 Machine Learning

Machine Learning (ML) is a sub-field of Artificial Intelligence (A.I.) in which algorithms are designed to discover the solution to a problem rather than being explicitly programmed with a set of rules to solve it. ML approaches can be particularly useful when an engineer or subject matter expert would find it difficult to explicitly list a complete set of rules to provide the answers or actions required. This difficulty often comes from the large number of rules required by an AI to accurately approach some problems or purely from the fact a solution to a given problem hasn't been discovered (and therefore can't be explicitly programmed within an AI).

There are three main categories of Machine Learning: supervised learning, unsupervised learning and reinforcement learning.

¹<https://bmv2019.org/>

In supervised learning, a Machine Learning algorithm is trained by being shown example input data for a problem, along with the correct answers for those examples. The result of this training process is a model which, given input data without the answers, can predict the answer accurately. For example, if an A.I. is required for predicting the sale price of houses, a Machine Learning approach may train a model by providing it with input examples that contain characteristics of previously sold houses (e.g. number of rooms) along with the sale prices. Then when a new house come on the market, its characteristics can be given as input to the model and a prediction of the sales price can be provided. In supervised learning, the dataset used to build the predictive model, is often split in to training data and test data. During training, the model receives both the input features of the example and the answer for it. The test data is not given to the model while it is learning the task, instead it is used to evaluate how well the final model performs. The model makes a prediction across all the examples in the test data and the percentage of correct predictions forms the model's accuracy score.

In unsupervised learning, models are trained from the input data alone, without being provided the correct output values. Approaches of this category often are used when the aim is to gain insights from data such as grouping entities in the data by similarity or detecting anomalies in the data. For example, when analyzing documents, it may be useful to have them grouped together by similar topic. This can be achieved using an unsupervised learning clustering algorithm.

Reinforcement learning (RL) (Kaelbling et al. 1996) approaches involve the interplay between two entities, an actor and an environment. The environment provides a current state to the actor, the actor then takes an action within the environment and receives the updated state and a value that represents the quality of the action performed, known as the reward. The actor, which is governed by a ML algorithm, is attempting to maximize the reward received. Over time, through many steps of receiving a state, making an action and receiving a reward, the RL algorithm learns which actions, given a current state have a strong likelihood of granting a high reward. This technique is often used for action-based tasks that are too complex to explicitly program an A.I. to perform the task explicitly. For example, an A.I. designed to play the video game Pac-Man may treat the player's location, the maze and the location of the enemies as the environment and receive this state via the current image being output to the screen by the game. In turn, it would make actions (moving the player) and the current score would be returned as the reward value.

This work specifically looks at improving the performance of A.I. approaching a supervised learning problem, Image Classification as outlined in Section 2.2).

2.2 Image Classification

The task of image classification is a supervised learning problem in which the input data is imagery and the desired output is a label signifying which of a set of class labels the image belongs. For example, given medical MRI images, an A.I. may be built to detect whether the scan shows signs of cancer ("positive" class) or not ("negative" class). Classification problems with only two possible classes (like this example) are often referred to as binary classification problems, though classification problems can, and often will, have more than one class.

Like with many research topics, image classification has an array of popular benchmarking datasets. These are shown in Table 1 which includes examples of the data, the current best accuracy and the paper attributed to have achieved the current state-of-the-art performance (according to the website: *papers with*

*code*²). All of the listed state-of-the-art performances were achieved through algorithms from a sub-field of Machine Learning, Artificial Neural Networks (ANNs) (Haykin 1994) and further within that, a sub-field, Deep Learning (DL) (LeCun et al. 2015). Deep Learning based models achieving the state-of-the-art performance is a trend seen across many different research areas not just image classification. However, in order to achieve this high performance on supervised learning tasks, ANNs and in particular Deep Neural Networks (DNNs) often require large volumes of labeled training examples which proves a barrier to adoption in real world scenarios.



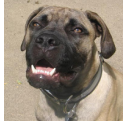

Benchmark Dataset	Example Image	Top Accuracy	State of the Art
MNIST (LeCun et al. 1998)		0.9982	(Kowsari et al. 2018)
CIFAR-10 (Krizhevsky & Hinton 2009)		0.99	(Huang et al. 2018)
ImageNet (Deng et al. 2009)		0.843	(Huang et al. 2018)
SVNH (Netzer et al. 2011)		0.986	(Zhang et al. 2019)

Table 1: Details of benchmarking datasets used within image classification research.

2.3 The Requirement for Labelled Data

As mentioned in Section 2.2, the current state-of-the-art approach to many problems is achieved through techniques from the field of Deep Learning. This includes problems such as diagnosis from medical imagery, self driving cars, speech recognition and many other key technological challenges seen today. In order to achieve this performance Deep Neural Networks require large volumes of labelled examples. This is because of the architecture of DNNs and the algorithms used to train them. Neural Networks, and in turn, Deep Neural Networks are built from a series of connected layers of neurons. The connections between the neurons in one layer to the next are weighted. Learning the correct weights (also refereed to as learnable parameters) is what allows the networks to perform a task well. The process for learning these parameters, back propagation (LeCun et al. 1990), along with the large quantity of parameters to update in modern state-of-the-art networks is the cause for the large volume of data. Although the exact requirement for labelled examples is an ongoing research topic, it has been proposed that it may take a number of examples up to ten times the amount of parameters in a networks to train it correctly (Abu-Mostafa et al. 2012). The state-of-the-art model (Huang et al. 2018) for the image classification benchmark Imagenet has nearly six hundred million parameters. Through the development of various accompanying strategies it has been shown that in practise, models can be trained with less than this upper bound of

²<https://paperswithcode.com/task/image-classification>

required examples. With that said, the requirement has not been reduced to a point where there aren't still significant challenges. Firstly, collecting example data of rare edge cases is difficult due to their infrequency among the data collected. This means that once all examples are labelled there can still be a shortage of specific types of input features the model would need to learn in order to perform a task well. For example, in the field of self driving cars, there is a huge number of life threatening scenarios which can occur during a journey, most of them happen very rarely. A lack of training examples for these scenarios will could cause A.I. based driving solutions to act unpredictable if these scenarios are encountered whilst being used. Secondly, there are times when the data could be difficult to collect in some other way outside of rarity. For example, collecting x-ray data exposes patients to harmful radiation and so the collection of data has to be limited due to safety concerns. Finally, assuming that the domain does not have rare edge cases or other data collection limitations, the process of actually labelling large volumes of examples is time consuming. As mentioned in the paper submission included in Section 3, the estimated cost of labelling 100 000 images from a leading provider of labelling tools and crowd-sourced human labelling (Amazon (Services 2019)) is over ten thousand dollars.

From these three factors, it is clear that methods to reduce the volume of labelled examples required will increase general model performance, the safety of using models in many use cases and reduce the cost to produce these state-of-the-art solutions.

2.4 Image Augmentation

This work focuses on image augmentation as a technique to reduce the volume of labelled images required for image classification problems. Image augmentation works by applying transformations to already labelled data to create new labelled examples without requiring further collection or labelling effort. The transformations are limited such that they don't cause large amount of deformation to the semantics contained within the original image but they change the image enough so that the new image provides a unique example of the class that the original image belongs to. The transformations include spatial operations, such as rotate, flip and sheer as well as colour-space transformations such as adjusting the contrast and brightness of the image. Examples of image transformations can be seen in Table 2. A review of the image augmentation literature is provided within section 2 of the paper submission included in Section 3 of this report but there is a key point to note for this section, as it forms the motivation behind this work. In the literature, there are many examples of how image augmentation has been applied to successfully improve model performance, often as part of a solution that becomes the (at the time) state-of-the-art. However, across these solutions, the selection of image augmentations to apply (the augmentation strategy) that provided the authors with the best performance for their specific model on their specific task is different. Commonly, these augmentation strategies are manually hand-crafted through trial and error. Automating this process of strategy discovery would be more efficient and likely lead to more optimal strategies. This was the aim of the paper by Google Brain (Cubuk et al. 2018), which posed AutoAugment as a Reinforcement Learning technique to select image augmentation technique to apply to a dataset during the training of an image classification model. The work in this project builds upon the ideas in that paper, expanding upon them and using them within alternative optimisation strategies to explore whether these alternatives can provide a better automated search of the space of augmentation policies.

Augmentation	Example Image
Original Image	
Horizontal Flip	
Vertical Flip	
Rotate	
Increase Contrast	
Increase Brightness	
Shear X Axis	
Shear Y Axis	
Translate X Axis	
Translate Y Axis	

Table 2: Examples of image augmentation techniques applied to an image from the benchmarking dataset, CIFAR-10.

3 BMVC Paper Submission

On the 29th August, the following paper (contained within the proceeding twelve pages) was submitted to the British Machine Vision Conference(BMVC) 2019. The announcement of acceptance for the papers submitted to BMVC is scheduled for 24th June 2019.

An Evaluation of Optimisation Algorithms for Image Augmentation Strategy Discovery

Daniel Harborne
harborned@cardiff.ac.uk
Kirill Sidorov

School of Computer Science &
Informatics
Cardiff University
Cardiff, UK

David Marshall

Abstract

When using Deep Learning for image classification, training robust and accurate models often requires a vast amount of labelled training data. The process of conducting this image labelling can be time consuming and therefore costly. One method for increasing the volume of labelled data produced from human labelling effort is image augmentation. This approach applies spatial and colour transformations to already labelled images in order to create new, pre-labelled images and thus increase the quantity of training examples available. Although this method has proven to be an essential part of training state-of-the-art models, the precise strategies used are often derived through trial and error and little work has been done to offer an informed approach to discovering these strategies automatically. Recently published work offers one solution (AutoAugment) but the team behind the work concluded that it may be far from the optimal approach. This work explores alternative discovery approaches for image augmentation strategy, discussing their characteristics and their effectiveness. This work provides a number of key contributions. Firstly, it provides an updated definition of the image augmentation search space. Secondly it provides a mapping of the augmentation strategy discovery problem to popular optimisation approaches and then offers an assessment of the effectiveness of these approaches. Finally, it uses these optimisation approaches to discover an augmentation policy that offers an improvement over the baseline that's nearly twice as large as the improvement provided by the policy discovered using AutoAugment.

1 Introduction

In the field of image classification, approaches using Deep Learning have become the standard for achieving state of the art performance [6, 23]. These algorithms “learn” how to perform their tasks rather than being explicitly programmed. In order to do this effectively, they often require large volumes of task-relevant examples in the form of labeled training data [20]. This poses two main problems. Firstly, performing the manual labelling of training images takes the commitment of a large number of human hours. These human hours equate to a large cost. Following the pricing of one of the leading providers of image labelling tools and crowd-sourced labels (Amazon [24]), to label 100000 images (a reasonable amount for contemporary image classification models) costs over ten thousand dollars.

Secondly, in many domains, there are rare edge cases for which examples are hard to collect. This ultimately means that even when all the collected images are labelled, these edge cases still prove to cause problems for the Deep Learning models performing the classification due to their low representation in the training data.

One approach to mitigate this issue is to use image augmentation [20, 30, 32] which produces “new” training images using image operations such as transformation, contrast change and mirroring. If this process is applied to already labelled images, the new images can be automatically labeled with the same label as the originals. Though it is widely recognized as being beneficial, there is a lack of studies that have explored developing best practises for image augmentation. Many state-of-the-art developments to Deep Learning techniques have been achieved with augmentation as part of their training pipeline but the strategies adopted are developed manually through trial and error. Although improvements to Deep Learning model architecture and techniques have been explored regularly within the literature [9, 10, 12, 27, 31], the gains seen by this pursuit are shrinking [4] and so pursuit of successful augmentation strategies appears to be a promising path for improvements. Furthermore, as stated by Sutskever et al. [13], the Deep Learning training process would benefit if the selection of effective augmentation strategies could be automated.

This is the motivation behind the recent paper from Google Brain [4], which explicitly outlines a possible search space of image augmentation techniques. In addition, the paper details an algorithm for automatically trialing samples of augmentations from this space on a subset of a dataset in order to find effective strategies to utilize in the final training on the full dataset. Although, the paper itself states that the search algorithm they propose (AutoAugment) may likely be improved upon, it already helped train models to beyond state-of-the-art performance on many well established image classification challenge datasets such as CIFAR-10 [17] and ImageNet [5].

The contribution of this work is an exploration of alternatives methods for optimising the image augmentation strategy. Provided are results for approaches based on Genetic Algorithms [25] and Tabu Search [8]. In the tests conducted, both methods discovered policies that offered improvement over the policy proposed by Cubuk et al. [4]. This work discusses the implications of this benchmarking — analyzing their characteristics with the goal of informing the direction of future work in this field.

2 Related Work

As mentioned in Section 1, image augmentation has been accepted as one technique for improving model performance and generalisability. The idea can be traced back to Baird et al. [2], in which the modification of existing image samples was used to improve performance on the task of character recognition. This section summarizes some of the key existing work in the image augmentation space and closes with a look at the work of Cubuk et al. [4] — the primary inspiration and foundation of this work. Today, plenty of the state-of-the-art models for the popular benchmarking image classification datasets use Deep Learning. For example, Real et al.’s augmentation strategy [23] was used to achieve the best performance on CIFAR-10. The strategy in that work was manually crafted through trial and error, however attempts have been made to automate this process. These approaches are mainly techniques which attempt to create synthetic examples based on the original training data but also include some techniques which attempt to automatically compose the best image augmentation strategy from a set of defined augmentation techniques.

Chawla et al. [3] used interpolation between two data samples to create new images. Similarly, Lemley et al. [18] propose Smart Augmentation — which uses a model pipeline to create training data formed by merging multiple labelled examples from the same class into a new labelled image. The idea of learning a feature space and applying interpolations were combined by Devries et al. [7] and augmentations were applied both to the vector representations of features before decoding and to the images themselves. Tran et al. [29] generated new synthetic examples after learning a distribution using a bayesian network. Taylor et al. [28] tested a range of augmentation techniques as well as generating new data via principal component analysis (PCA) [15].

A portion of the literature [1, 11, 19, 26, 35] uses Generative Adversarial Networks (GAN) [21] to generate synthetic images. However, Ratner et al. [22] explore using GANs to develop an augmentation strategy.

Finally, Cubuk et al. [4] offered a Reinforcement Learning (RL) [16] approach to building augmentation strategies (AutoAugment). Using this method, the authors achieved state-of-the-art performance on benchmarking datasets such as CIFAR-10, ImageNet and SVHN. A more detailed discussion of this technique, specifically the aspects that have carried over to the exploration carried out in this work, will be provided in Section 3.

3 Image Augmentation Strategies

In this work, many characteristics of image augmentation strategies outlined for the AutoAugment approach in Cubuk et al. [4] are utilised within alternative optimisation strategies. This section contains a summary of these adopted characteristics along with a justification of why they have been used. In Section 4, more specifics are detailed for each of the optimisation algorithms.

3.1 Image Augmentations

Taylor et al. [28] outline that image augmentations fall in to two categories. Geometric (spatial transformations) and photometric (color space transformations). Cubuk et al.[4], provide a list of augmentations which is composed using the augmentations available in the python library PIL¹ as well as two additional augmentations, Cutout [6] and SamplePairing [14]. The resulting list (shown in Table 7 of Cubuk et al.[4]) whilst comprehensive, may not be exhaustive. However, as the objective of this work (and of Cubuk et al.[4]) is to develop a way of automatically composing the optimal augmentation strategy from a list of possible augmentations, the list does not need to be exhaustive at this stage. If additional augmentation techniques are proposed in future literature, they can be added to the search space for these optimisation algorithms.

To apply an augmentation, a number of additional values are needed. An augmentation technique accompanied by these values will be referred to as a technique configuration. Concretely, a technique configuration consists of the technique itself, the probability of applying the technique to an image and the magnitude at which to apply it. The range these values can take, the representation of the configuration and the method of applying these configurations is determined by the approach used to form the augmentation strategy. In Section 3.2, a discussion of these choices is given.

¹<https://pillow.readthedocs.io/en/5.1.x/>

3.2 Image Augmentation Policy Structures

When developing an augmentation policy, there are a number of structures the policy could take by arranging augmentation technique configurations. For example, a simple approach would be to form a sequential list of configurations, one for each technique. By only requiring one set of values per technique (a magnitude and a probability) this greatly reduces the search space (a total of two values multiplied by the number of unique augmentation techniques considered). However, this poses two main issues. Firstly, the policy is limited to only one configuration and therefore one magnitude and probability per augmentation technique. In turn, the policy loses the ability to have a technique applied with small magnitude often (high probability) but also with high magnitude occasionally (lower probability value). Secondly, a policy of this flat structure is unable to group combinations of augmentation techniques together with respect to any synergies of apply certain technique sequentially. At the opposite end of the policy structure space, a policy could consist of any number of combinations of technique configurations, allowing for the policy to establish chains of synergistic configurations and in some way (e.g. randomly) select between these set combinations. This also allows for multiple instances of the same technique to be included (with different magnitudes). The issue with this approach is that, without a cap, it allows for an infinite search space, with techniques being chained indefinitely within a combination and an infinite number of combinations being possible.

In Cubuk et al. [4], the authors defined a structure that falls between these two extremes. This work also makes use of this structure and as such, this section will summarise the structure and also provide further justification for its use.

The full augmentation policy is made of a number of sub-policies, each of which consist of an amount of augmentation technique configuration. These configurations consist of an augmentation technique, the probability of applying the technique and the magnitude at which to apply the technique.

The range of probability values and the range of magnitude values can be discretized in order to reduce the search space to a finite set of values. In Cubuk et al. [4] and in this work, the probability values increment in steps of 0.1 from 0.0 to 1.0 inclusive (11 values) and the magnitudes increment in steps of 1 from 1 to 10 inclusive (10 values).

3.3 Applying an Augmentation Policy during Model Training

To incorporate an augmentation policy with the defined structure within training, the following approach is taken:

1. Randomly select a sub-policy from the augmentation policy (uniformly)
2. For each technique configuration in the sub-policy:
 - Determine if the technique will be applied — A randomly generated number between 0.0 and 1.0 should fall below the probability of the technique configuration).
 - If the technique is to be applied, apply it with the given magnitude.

In addition, in Cubuk et al. [4] and in this work, the previously establish augmentation strategy for the dataset is applied (e.g. for CIFAR-10, the strategy outlined by Real et al. [23]).

This structure and application process allows for multiple instances of the same technique with different magnitudes to be included, it allows pairs of augmentations to be formed that

can be applied in the most effective order and, by limiting the size and number of sub-policies, bounds the search space of possible policies. This structure does still often lead to a large search space depending on size and number of sub-policies. The search space is discussed further in Section 3.4.

During policy discovery, in Cubuk et al. [4] AutoAugment is used to search the space of policies that consist of 5 sub-policies, each with 2 technique configurations. Once the discovery process is complete, the authors concatenated the best 5 policies together to get a final policy of 25 sub-policies. The final policy used to achieve state of the art performance on CIFAR-10 is shown in Table 8 of Cubuk et al. [4].

3.4 The Search Space of Adopted Augmentation Policy Structure

In this section, a calculation is provided of the search space of a policy structure with 25 sub-policies each with two technique configurations. This expands on the calculations in Cubuk et al. [4] and includes some observations which reduce the calculated policy search space by three orders of magnitude.

As three augmentation techniques (AutoContrast, Invert and Equalize) don't use the magnitude value, the range of magnitude values for technique configurations including these techniques does not need to be searched during policy optimisation. Thus the updated calculation for the search space of technique configurations is as follows:

$$\begin{aligned} |T^m| &= \text{The number of techniques which utilise a magnitude value} \\ |T^w| &= \text{The number of techniques which do not utilise a magnitude value} \\ |P| &= \text{The number of distinct discrete probability values} \\ |M| &= \text{The number of distinct discrete magnitude values} \\ |S^C| &= \text{The size of the search space for technique configurations} \end{aligned}$$

$$\begin{aligned} |S^C| &= |T^m| \times |P| \times |M| + |T^w| \times |P| \\ |S^C| &= 13 \times 11 \times 10 + 3 \times 11 = 1463 \end{aligned}$$

The space of sub-policies consists of permutations of these technique configurations and therefore the space of sub-policies is defined as:

$$\begin{aligned} |S^{SP}| &= \text{The size of the search space for subpolicies} \\ |S^{SP}| &= |S^C|^2 \\ |S^{SP}| &= 1463^2 = 2137444 \end{aligned}$$

Updating the calculations from Cubuk et al. [4], it can be observed that choosing a number of sub-policies to form a policy is a combination and not a permutation calculation (as the order of the sub-policies within the policy doesn't matter). Therefore the search space for policies can be defined as:

$$\begin{aligned} |A| &= \text{The number of subpolicies within a full augmentation policy} \\ |S^A| &= \text{The size of the search space for policies} \end{aligned}$$

$$\begin{aligned} |S^{SP}| &= \binom{|A| + (|S^{SP}| - 1)}{|A|} \\ |S^{SP}| &= \binom{5 + 2137443}{5} \\ |S^{SP}| &\approx 3.7178727 \times 10^{29} \end{aligned}$$

4 Exploration of Optimisation Algorithms

In this section, the optimisation algorithms explored will be detailed including pseudo code and motivations behind the choice of the algorithms. The section begins with a summary of the evaluation process used for policies during the running of an algorithms and the evaluation process of the best found policy.

4.1 Evaluating Policies

4.1.1 Dataset Choice

During both policy discovery and final evaluation, the policies are evaluated based on the accuracy achieved by a model trained and tested on CIFAR-10. This dataset is complex enough that image augmentation is beneficial but the image size is small enough that the training process is relatively short. This makes it a good candidate for performing a comparison of the optimisation techniques. In Cubuk et al. [4], it was shown that a policy discovery strategy that works for one dataset can be applied to further datasets. It was also shown that policies learned on one dataset can transfer to another with noticeable improvements to results. As such, a comparison of optimisation techniques across further datasets was not required at this stage and it is suggested as future work.

4.1.2 Evaluation During Discovery vs Final Evaluation

In order to evaluate policies during discovery, models are trained that incorporate the policy in their training pipeline. As in Cubuk et al. [4], a less resource-demanding training pipeline is used during the policy discovery process. Unlike in Cubuk et al. [4], in this work the objective is to critically assess the characteristics of various optimization algorithms rather than search for a new state-of-the-art model. As such, in the interest of computational resources, this work has continued to use the Wide-ResNet model [34] in the final evaluation over Pyramid [9] with ShakeDrop [33] in Cubuk et al. [4]. Table 1 shows the differences between the evaluation pipelines. In the discovery phase, validation data is used to avoid over-fitting the policy to the test data. Expanding on Cubuk et al.'s approach, in all policy evaluations, three child models are trained and their accuracy is averaged. This aims to minimise the impact to evaluation caused by the variance caused by the random application of the augmentation techniques.

4.2 Genetic Algorithm Optimisation

Genetic Algorithm [25] approaches are inspired by the evolution process seen in nature. There consists a population of candidates that are evaluated and based on this evaluation

Phase	Model	Epochs	Training Images
Discovery	Wide-ResNet	120	4000
Final evaluation in Cubuk et al.	Pyramid with ShakeDrop	200	50000
Final evaluation in this work	Wide-ResNet	200	50000

Table 1: Parameters from the evaluation pipelines used during policy discovery and final evaluation.

(known as fitness) they are evolved such that the population in the next generation contains the strongest candidates (or mutations of those candidates).

Table 2 provides the mapping between aspects of a genetic algorithm and features of the image augmentation problem used in this work.

Algorithm Aspect	Mapping to Problem
Chromosome:	Policy
Fitness:	Child model accuracy
Mutation:	Technique — Change to a different technique Probability — Increase/decrease by 0.1 (Adhering to valid range) Magnitude — Increase/decrease by 0.1 (Adhering to valid range)
Crossover:	Choose split point between sub-policies, crossover at point.

Table 2: Mappings between aspects of a Genetic Algorithm implementation and image augmentation policies.

4.2.1 Discussion of Algorithm Properties

Due to the nature of evaluating a member of the population in this problem space (having to train a child model for every chromosome) and even with parallel training of child models, the size of the population has to be kept small in order to move through generations at a computationally reasonable cost. Further to this, the nature of the mutation process means that when a technique mutates to another technique within a configuration, the probability and magnitude remain as they were. This means that the chromosomes which provide an example of a technique may be evaluated with unsuitable parameters. With that being said, the concept of cross-over within genetic algorithms applies well to the search from a strong policy. Crossing over two policies that contain a few strong sub-policies to create a new policy which contains all of these sub policies is a good way to get jump up the highest performance of the next generation.

4.3 Tabu Search Optimisation

In Tabu Search [8], an initial candidate is selected then a number of neighbourhood moves are made. To make these moves, the entire neighbourhood of the current candidate is evaluated and the best possible move that is not contained within the Tabu list. The Tabu list is populated with moves that were recently already explored. The number of steps in to the past that are considered to be recent is a hyper-parameter and for this work, 10 was used.

4.3.1 Definition of Neighbour for Augmentation Policies

For the augmentation policy structure used, this work considers a neighbouring policy to follow the following rules:

1. All technique configurations except one are the same as in the current policy.
2. The differing configuration is changed only in one of the following ways:
 - The augmentation technique is changed.
 - The probability is 0.1 higher or lower than the current value (within the bounds defined).
 - The magnitude is 0.1 higher or lower than the current value (within the bounds defined).

4.3.2 Discussion of Algorithm Properties

The key relevant characteristic of Tabu search is that it includes an exhaustive search of the current policies neighbourhood. This, coupled with the large computational cost of evaluating candidates, makes starting from a randomly initialised policy not viable. As such, Tabu search does not offer a useful tool by itself for policy discovery. However, it does pose a very useful tool for policy refinement, in the case where the current proposed policy is not a local optimum, Tabu search can be used to make incremental improvements.

5 Results and Evaluation

The results seen in Table 3 show that both Genetic Algorithms and Tabu Search were able to find a policy that lead to a final model accuracy higher than that of the AutoAugment discovered policy. For Tabu Search an increase of 0.9% over the baseline (of no policy) is observed. This is close to doubling the gain seen from the AutoAugment discovered policy (a gain of 0.5%).

The conclusion made from this is that both a Genetic Algorithm approach and Tabu Search show promise for augmentation policy discovery. Further to this, it can be observed that Genetic Algorithms was able to discover an improved policy with less computational resource than Tabu Search but the policy discovered from a few iterations of Tabu Search (whilst being computational intensive) is better.

Discovery Technique	Accuracy with WRN
No Policy (baseline)	0.965
AutoAugment	0.970
Genetic Algorithm	0.973
Tabu Search	0.974

Table 3: Accuracy results on CIFAR-10 from Wide-ResNet models with training pipelines that include applying the best augmentation policy found during policy discovery using optimisation algorithm .

6 Future Work

There are a number of directions for future work based on this work.

The results generated were made using the Wide-ResNet model. Whilst this allows for comparison of the techniques in a computationally efficient way, this minimises the chance of training a final model that reaches the current state of the art. Though this was not the objective of this work, future work should explore this.

In this work (and in Cubuk et al. [4]), the policy structure was defined and then the search space of this structure was explored. Instead, an expansion on the genetic algorithm approach could start with a base structure and evolve the policy's structure as well as the policy's contents.

It may be possible to evaluate the individual transform techniques and this evaluation combined with some analysis may allow for the combining or the removal of some techniques. This could lead to a considerable reduction in the size of the search space.

Finally, this work explored one modality of data (imagery) and one dataset (CIFAR-10) future work should explore other examples of both categories.

7 Conclusion

This work has provided a critical assessment of optimization algorithms applied to the problem of image augmentation policy discovery for Deep Learning image classification which has provided a number of contributions. The definition of the search space for this problem has been refined which will help inform the community looking at tackling it. This search space has been mapped to two optimisation algorithm approaches, Genetic Algorithms and Tabu Search. From the results of applying these two algorithms to discover augmentation policies, it can be concluded that Genetic Algorithms shows a promising approach for policy discovery with Tabu Search providing stronger policy refinement but at a much higher computational cost to move through the search space. Ultimately, an augmentation policy has been discovered that, during testing, offered an improvement over the baseline of 0.9%. This is 80% larger than the improvement offered by the policy discovered using AutoAugment.

8 Acknowledgements

We acknowledge the support of the Supercomputing Wales project, which is part-funded by the European Regional Development Fund (ERDF) via Welsh Government.

References

- [1] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv: 1711.04340*, nov 2017.
- [2] H.S. Baird. Document image defect models and their uses. pages 62–67. IEEE Comput. Soc. Press, 1993. ISBN 0-8186-4960-7. doi: 10.1109/{ICDAR}.1993.395781. URL <http://ieeexplore.ieee.org/document/395781/>.
- [3] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, jun 2002. ISSN 1076-9757. doi: 10.1613/jair.953. URL <https://jair.org/index.php/jair/article/view/10302>.
- [4] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.
- [5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. Ieee, 2009.
- [6] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.
- [7] Terrance DeVries and Graham W. Taylor. Dataset augmentation in feature space. *arXiv preprint arXiv: 1702.05538*, feb 2017.
- [8] Fred Glover and Manuel Laguna. Tabu search. In Ding-Zhu Du and Panos M. Pardalos, editors, *Handbook of combinatorial optimization*, pages 2093–2229. Springer US, Boston, MA, 1999. ISBN 978-1-4613-0303-9. doi: 10.1007/978-1-4613-0303-9_33. URL http://link.springer.com/10.1007/978-1-4613-0303-9_33.
- [9] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. In *Computer Vision and Pattern Recognition (CVPR)*, pages 6307–6315. IEEE, 2017.
- [10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [11] Hu Hu, Tian Tan, and Yanmin Qian. Generative adversarial networks based data augmentation for noise robust speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICA SSP)*, pages 5044–5048. IEEE, apr 2018. ISBN 978-1-5386-4658-8. doi: 10.1109/{ICA\SSP}.2018.8462624. URL <https://ieeexplore.ieee.org/document/8462624/>.
- [12] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Computer Vision and Pattern Recognition (CVPR)*, pages 7132–7141, 2018.
- [13] Tim Salimans Ilya Sutskever, John Schulman and Durk Kingma. Requests for research 2.0, 2018. URL <https://blog.openai.com/requests-for-research-2>.

- [14] Hiroshi Inoue. Data augmentation by pairing samples for images classification. *arXiv preprint arXiv: 1801.02929*, jan 2018.
- [15] Ian Jolliffe. Principal component analysis. In Miodrag Lovric, editor, *International encyclopedia of statistical science*, pages 1094–1096. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-04897-5. doi: 10.1007/978-3-642-04898-2_455. URL http://link.springer.com/10.1007/978-3-642-04898-2_455.
- [16] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, may 1996. ISSN 1076-9757. doi: 10.1613/jair.301. URL <https://jair.org/index.php/jair/article/view/10166>.
- [17] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.
- [18] Joseph Lemley, Shabab Bazrafkan, and Peter Corcoran. Smart augmentation learning an optimal data augmentation strategy. *IEEE access : practical innovations, open solutions*, 5:5858–5869, 2017. ISSN 2169-3536. doi: 10.1109/{ACCE\SS}.2017.2696121. URL <http://ieeexplore.ieee.org/document/7906545/>.
- [19] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv: 1712.04621*, dec 2017.
- [20] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.
- [21] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv: 1511.06434*, nov 2015.
- [22] Alexander J Ratner, Henry R Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher RÃl. Learning to compose domain-specific transformations for data augmentation. *Advances in neural information processing systems*, 30:3239–3249, dec 2017. URL <https://www.ncbi.nlm.nih.gov/pubmed/29375240>.
- [23] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. *arXiv preprint arXiv:1802.01548*, 2018.
- [24] Amazon Web Services. Amazon sagemaker ground truth pricing, 2019. URL <https://aws.amazon.com/sagemaker/groundtruth/pricing/>.
- [25] Dan Simon. *Evolutionary Optimization Algorithms*. Wiley, may 2013. ISBN 9780470937419.
- [26] Leon Sixt, Benjamin Wild, and Tim Landgraf. Rendergan: generating realistic labeled data. *Frontiers in Robotics and AI*, 5, jun 2018. ISSN 2296-9144. doi: 10.3389/frobt.2018.00066. URL <https://www.frontiersin.org/article/10.3389/frobt.2018.00066/full>.

- [27] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.
- [28] Luke Taylor and Geoff Nitschke. Improving deep learning using generic data augmentation. *arXiv preprint arXiv: 1708.06020*, aug 2017.
- [29] Toan Tran, Trung Pham, Gustavo Carneiro, Lyle Palmer, and Ian Reid. A bayesian data augmentation approach for learning deep models. *arXiv preprint arXiv: 1710.10564*, oct 2017.
- [30] Cristina Nader Vasconcelos and Bárbara Nader Vasconcelos. Increasing deep learning melanoma classification by classical and expert knowledge based image transforms. *CoRR, abs/1702.07025*, 1, 2017.
- [31] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, pages 5987–5995. IEEE, 2017.
- [32] Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. Improved relation classification by deep recurrent neural networks with data augmentation. *arXiv preprint arXiv:1601.03651*, 2016.
- [33] Yoshihiro Yamada, Masakazu Iwamura, Takuya Akiba, and Koichi Kise. ShakeDrop regularization for deep residual learning. *arXiv preprint arXiv: 1802.02375*, feb 2018.
- [34] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *British Machine Vision Conference*, 2016.
- [35] Xinyue Zhu, Yifan Liu, Jiahong Li, Tao Wan, and Zengchang Qin. Emotion classification with data augmentation using generative adversarial networks. In Dinh Phung, Vincent S. Tseng, Geoffrey I. Webb, Bao Ho, Mohadeseh Ganji, and Lida Rashidi, editors, *Advances in knowledge discovery and data mining*, volume 10939 of *Lecture notes in computer science*, pages 349–360. Springer International Publishing, Cham, 2018. ISBN 978-3-319-93039-8. doi: 10.1007/978-3-319-93040-4_28. URL http://link.springer.com/10.1007/978-3-319-93040-4_28.

4 Additional Algorithm: Particle Swarm Optimisation

In Particle Swarm Optimisation (PSO) (Parsopoulos & Vrahatis 2010), vectors of values from 0.0 to 1.0 are used to represent the parameters that are to be optimised. This creates a multi-dimensional vector space that the algorithm searches within to find an optimal selection of values. To do this, a population of vectors (particles) is created and distributed across the space. At each time step, the algorithm aims to move the swarm of particles closer to an optimum (hopefully, but not guaranteed to be a global optimum). To do this, the algorithm tracks each particle's velocity (inertia) and all-time best position. It also tracks the global all-time best discovered position among all particles. When a particle's position in the vector space is updated, it makes a move based on a combination of these three values. The weightings given to these three values are tune-able hyper-parameters. For this work the implementation of PSO provided in the library PySwam³ was used along with its recommended values for the weighting hyper-parameters.

Mapping the Augmentation Policy Search Space to a PSO Vector Space

The mapping of the search space of policies to a vector space proved challenging. As currently defined in the policy structure, magnitude and probability are fairly trivial to represent as values between 0.0 and 1.0 within a particle vector however, representing the choice of augmentation technique within a technique configuration is not straightforward. Unfortunately, some of this work's conclusions about the nature of augmentation policy structure (as described in section 3.2 of the submission to BMVC) were formed after the experimentation with PSO was completed. Specifically, the disadvantages of a flat policy structure were not factored in when mapping to the PSO vector space. As such, the mapping shown in Table 3 has these disadvantages which may explain the results generated for the best policy (shown in Table 5 found with PSO being close to the baseline of "no policy"). An alternative mapping is shown in Table 4. This mapping allows for the representation of the policy structure used within the other algorithms. Further work should aim to generate more results using this mapping. Due to the PSO approach used not accurately capturing its potential, the technique was removed from the discussion of the paper submitted to BMVC.

Discussion of Algorithm Properties

As the vector space for this search problem is large and the particles begin distributed across it, the time to converge on an optimal solution is likely to be long. This may be a second factor in the generated results for PSO being close to the no-policy baseline. A notable characteristic of PSO when compared to genetic algorithms is that local moves made by each particle are informed (rather than the random mutations seen in genetic algorithms). Because of this, given enough iteration steps, PSO may have better potential to converge on an optimal solution but further work to formalise this and measure it would need to be conducted.

³<https://pythonhosted.org/pyswarm/>

Characteristic	Mapping to Problem
Total Vector Length:	3 X Number of Augmentation Techniques (a float for each technique, it's probability and it's magnitude)
Mapping of Technique:	Each technique is assigned an index between 0 and the number of techniques - 1 . The order each technique is applied is determined by ordering the float values at each index.
Mapping of Probability:	Each probability for applying a technique is assigned an index between the number of techniques and (2 x the number of techniques - 1). When mapping back from the vector to a policy, each value is rounded to one decimal place.
Mapping of Magnitude:	Each magnitude to apply a technique at is assigned an index between the 2 x the number of techniques and (3 x the number of techniques - 1). When mapping back from the vector to a policy, each value is rounded to one decimal place.

Table 3: Mapping used within this work between values of a PSO vector and augmentation policies.

Characteristic	Mapping to Problem
Total Vector Length:	3 X the number of technique configurations in sub-policy X the number of sub-policies within policy.
Mapping of Policy Structure:	Each element (technique, probability magnitude) within each technique configuration is given an index within the particle vector.
Mapping of Technique:	Mapping the value within a particle vector at an index that is representing the augmentation technique of a configuration can be done by dividing the range of numbers in to a number of buckets equal to the number of augmentation techniques.
Mapping of Probability:	When mapping back from the vector to a policy, each value representing a probability is rounded to one decimal place.
Mapping of Magnitude:	When mapping back from the vector to a policy, each value representing a magnitude is rounded to one decimal place.

Table 4: Improved mapping between values of a PSO vector and image augmentation policies.

Discovery Technique	Accuracy with WRN
No Policy (baseline)	0.965
Particle Swarm Optimisation	0.967
AutoAugment	0.970
Genetic Algorithm	0.973
Tabu Search	0.974

Table 5: Updated results table to include omitted PSO result.

5 Implementation Details

In this section, further details of the implementation used for the experiments and result generations are provided. There are two code repositories for the work, both available on GitHub. The first ⁴ is a tool built to make use of the resources of the Advanced Research Computing at Cardiff department (ARCCA) ⁵ with python. The second ⁶ is the code base used for running the experiments.

5.1 ARCCA Python Tool & Parallel Job Framework

As outlined in Section 3 and in Section 4, all three algorithms explored in this work were based on evaluating candidate policies by including the policy in the training pipeline of an image classification model. The evaluation is based on the final validation accuracy of these trained models and within each time step of all three algorithms, the training and evaluation of multiple models could be conducted independently and in parallel. This parallelism was obtained by submitting jobs to ARCCA which was facilitated by a Python tool, written as part of this project. This tool has been documented (available on the GitHub link and as an appendix item A) and is now in use within other research projects conducted by myself and colleagues at Cardiff's Crime and Security Research Institute ⁷. The ARCCA tool offers ways to submit new jobs and check job statuses. It also allows file-based functionality such as sending, receiving and deleting files to, from and on the ARCCA server. Full details of the functionality is detailed within the documentation on the repository. In addition to this tool (which offers the basis for interaction with the ARCCA server), a management class was developed for this project that uses the ARCCA tool to start and manage parallel jobs. In general, the precise management of parallel jobs will be specific to the type of algorithm needing to be parallelised. In this work, as all three algorithms parallelise the same type of job (training models using augmentation policies), they all use the algorithmic structure shown in figure 1. The processing that takes place within lines 4-7 is always the same for each algorithm and it is this functionality captured within the second management class. By moving the code for the job management in to a separate class, the work behind evaluating each policy is decoupled from the optimisation algorithms themselves. This allows for better debugging, testing readability and also allows for the flexibility to swap in an alternative policy evaluation approach. Each policy could be evaluated sequentially locally, or a new tool could be developed to post jobs to other high-performance computing providers (E.g. AWS ⁸ or Google Cloud ⁹).

In ongoing work at Cardiff's Crime and Security Research Institute, researchers have adapted the parallel jobs management class in order to coordinate distributed training of action recognition models for video data using the resources of ARCCA.

⁴https://github.com/HarborneD/ARCCA_Python_Tool

⁵<https://www.cardiff.ac.uk/advanced-research-computing>

⁶https://github.com/HarborneD/genetic_augment

⁷crimeandsecurity.org

⁸<https://aws.amazon.com/>

⁹<https://cloud.google.com/>

Figure 1 Evaluation of Policies using ARCCA

```
1: Create initial batch of candidate augmentation policies (Algorithm Specific)
2:
3: for each iteration of the optimisation algorithm do:
4:   Send policies to ARCCA sever
5:   Start a model training job on ARCCA for each candidate policy
6:   Wait for all jobs to finish
7:   Collect evaluation results from server
8:
9:   Update best discovered policy tracking
10:
11:   Use evaluations to create next batch of candidates (Algorithm Specific)
12:
13: return best discovered policy
```

6 Reflection on Learning

During this work, there have been a number of decisions and experience that serve as cause for reflection. The main lesson comes from the volume of insights gained through the process of writing the work into a paper submission. I have written research papers in the past, but often the work has been discussed in detail with collaborators as the work progresses. As this project was more independent, there was less opportunity for in-depth discussion and as such, I had not placed some of the ideas and approaches under the same level of ongoing scrutiny. The paper writing process served as a replacement for this scrutiny, but it came at the end of the work. As such, although this process did reveal a similar level of insights as I have come to expect through collaborative scrutiny, these realisations came at a time where it was difficult to consider them as more than opportunities for further work. From prior projects, I had learned to document the work as it progresses but the key lesson is that this documentation needs to be more substantial in order to pick up possible avenues missed whilst actively moving the work forward. This does have the trade off requiring more time to be spent on documenting the work, however it provides the added benefit of having content that is usable in the final publication ready ahead of the final writing process. As such, going forward, when working on projects independently, I will set aside time periodically to formalise the current work being carried out as an additional level of scrutiny.

The other key learning experience came from the implications of conducting research in which the generating of results was time and resource intensive. Early on, I identified that the level of resources required could be restrictive and as such, I gave priority early in the project to test the extent of this. Doing this was an important step as from this testing, it was clear that training the models locally (even on high performance hardware) would be too timely to generate results within the allocated time for the project. By identifying this early, plans could be quickly put in place to make use of the resources available through ARCCA which helped minimise any delay caused by organising access and training. This forward planning worked well and as such, in the future I will aim to identify similar risks that need assessing early in a project's timeline. Using the super computing resources also came with further lessons. Firstly, the current provisions and documentation for using python on ARCCA are limited and some time had to be dedicated to figuring the best approach to run a python job. After succeeding at this, I was approached by other students using ARCCA for their projects and from this, I was given the

opportunity to refine my ability to teach others how to make use of resources and how to impart my own lessons learned. Secondly, building the ARCCA python tool for this project gave the opportunity to develop something that could be used by the wider community. Once it appeared the tool would be useful for myself and other researchers, I made an effort to write and document the tool in a way that would allow it to be used for more than just my immediate use case. This was good practise for writing tools that could be used by others and will help me to write reusable code in the future.

7 Conclusion

To conclude, I believe this project met the majority of its aims and objectives. It moved from a project that aimed to replicate existing research (as outlined in the original project plan) to a achieving an improvement on that research, summarised within a research submission. In this submission, it explored multiple optimisation algorithms and in doing so, discovered a state-of-the-art augmentation policy. The characteristics of using these algorithms within the problem space have been captured and whilst there is still a lot of further work, this project has made reasonable contributions to the field. The work carried out has also provided a tool that is being used within other research projects and so this work has made impacting engineering contributions. There are some lessons to be learned with regard to deeply scrutinising the work as it progresses, this will help to ensure that the methods being used within experiments are appropriate and that important theory has not been overlooked.

Additional References

Abu-Mostafa, Y. S., Magdon-Ismail, M. & Lin, H.-T. (2012), *Learning From Data*, AMLBook.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V. & Le, Q. V. (2018), ‘Autoaugment: Learning augmentation policies from data’, *arXiv preprint arXiv:1805.09501* .

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. & Fei-Fei, L. (2009), Imagenet: A large-scale hierarchical image database, in ‘Computer Vision and Pattern Recognition (CVPR)’, Ieee, pp. 248–255.

Haykin, S. (1994), *Neural networks*, lovelammas.co.uk.

URL: <http://www.lovelammas.co.uk/19550271/by-www-lovelammas-co-uk/neural-networks-and-learning-machines.pdf>

Huang, Y., Cheng, Y., Chen, D., Lee, H., Ngiam, J., Le, Q. V. & Chen, Z. (2018), ‘GPipe: Efficient training of giant neural networks using pipeline parallelism’, *arXiv* .

URL: <https://arxiv.org/abs/1811.06965>

Kaelbling, L. P., Littman, M. L. & Moore, A. W. (1996), ‘Reinforcement learning: A survey’, *Journal of Artificial Intelligence Research* 4, 237–285.

URL: <https://jair.org/index.php/jair/article/view/10166>

Kowsari, K., Heidarysafa, M., Brown, D. E., Meimandi, K. J. & Barnes, L. E. (2018), RMDL: random multimodel deep learning for classification, in ‘Proceedings of the 2nd International Conference on

Information System and Data Mining - ICISDM '18', ACM Press, New York, New York, USA, pp. 19–28.

URL: <http://dl.acm.org/citation.cfm?doid=3206098.3206111>

Krizhevsky, A. & Hinton, G. (2009), Learning multiple layers of features from tiny images, Technical report, Citeseer.

LeCun, Y., Bengio, Y. & Hinton, G. (2015), 'Deep learning.', *Nature* 521(7553), 436–444.

URL: <http://www.nature.com/doi/10.1038/nature14539>

LeCun, Y., Boser, B. & Denker, J. (1990), 'Handwritten digit recognition with a back-propagation network', *Advances in neural ...*

URL: <http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network.pdf>

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998), 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE* 86(11), 2278–2324.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B. & Ng, A. Y. (2011), Reading digits in natural images with unsupervised feature learning, in 'NIPS workshop on deep learning and unsupervised feature learning', Vol. 2011, p. 5.

Parsopoulos, K. E. & Vrahatis, M. N. (2010), 'Particle swarm optimization and intelligence: Advances and applications', *Information Science Publishing (IGI Global), Hershey, PA, U.S.A.*

URL: <http://rgdoi.net/10.13140/2.1.3681.1206>

Services, A. W. (2019), 'Amazon sagemaker ground truth pricing'.

URL: <https://aws.amazon.com/sagemaker/groundtruth/pricing/>

Zhang, H., Dauphin, Y. N. & Ma, T. (2019), 'Fixup initialization: Residual learning without normalization', *arXiv*.

URL: <https://arxiv.org/abs/1901.09321v2>

Appendices

A Python ARCCA Tool Documentation

The following pages contain the documentation from the README of the python ARCCA tool repository ¹⁰. It is included to provide an indication of what is possible with the tool currently and the ease of it's use.

¹⁰https://github.com/HarborneD/ARCCA_Python_Tool

ARCCA_Python_Tool

A python tool for interacting with the Super Computing Wakes ARCCA super computing resources.

This tool provides a python class to programitcally submit and check jobs on the ARCCA job server from within python scripts.

The tool also allows for SFTP actions to made such as sending, fetching and deleting files to/from/on the SCW server.

Account Registration

To make use of the tool (and the SCW resources) be sure to first set up an account and join, or set up a project(Adhering to the rules and pre-requests to do so).

This can be done [here](#).

Credentials File

Once registered you must create a credentials file in the root of the ARCCA_Python_Tool directory. If one does not exist and you attempt to instantiate the class, a template credentials file will be created and the script will terminate.

If you want to manually create the credentials file, it should look like this:

```
{
  "username": "USERNAME",
  "pw": "PASSWORD"
}
```

This file is ignored by the repository but of course, be careful with the privacy/accessability of this file.

Please Note: Due to security concerns and lack of support in the python SSH module being used (paramiko),the tool's login to SCW does not perform the check against man in the middle attacks using ECDSA host keys. Instead, the tool uses your system's approved host keys. This requires you to manual log on to the SCW login server once via SSH and add the key to your approved list.

Instantiating Tool

To use the tool simply instantiate an instance of it, passing the address of your login server. The current host addresses are summarised below (if you encounter issues, check SCW for the most up to date addresses).

Institution	Server	Address
Cardiff or Bangor	Hawk	hawklogin.cf.ac.uk
Swansea or Aberystwyth	Sunbird	sunbird.swansea.ac.uk

Instantiation:

```
host = "hawklogin.cf.ac.uk"
arcca_tool = ArccaTool(host)
```

Job Management

Job management is approached similarly to if you were remotely accessing the job server. As such, it is reccomended that you are familiar with how to submit jobs using remote access. Details of this can be found [here](#).

This also means that slurm job script files (and any other required files, such as data) should be already on your file system (or should be transferred using this tool ahead of starting the job). This tool does not aim to create these for you - simply to enable run them and manage them programitcally.

Submitting a Job

```
#The project account to run the job under
account = "scw1234"

#The absolute path to the directory of your file sapce to run the job from
run_from_path = "/home/c.c1234567/my_folder"

#The realtive path to the job script from that directory
script_name = my_slurm_script.sh

#A string repesenting the command line arguments to be passed to the job script.
#This string will simply be concatenated as if they were typed after the job script name.
arguments = "MY_FIRST_ARG MY_SECOND_ARG OPTIONAL_ARG=10"

arcca_tool.StartBatchJob(account,run_from_path,script_name, args=arguments)
```

Checking Job Status

```
#get status of all jobs on or after this date
start_time="2019-04-01"

status_list = arcca_tool.CheckJobsStatuses(start_time="2019-02-01")
```

status_list is a list containing dictionaries with the following format:

```
status = {
  'account': 'scw1427'
  'cpu_alloc': '1'
  'exit_code': '11:0'
  'job_id': '7896543'
  'job_name': 'my_job'
  'partition': 'gpu'
  'state': 'FAILED'
}
```

Checking the Predicted Start Time of a Job

```
arcca_tool.CheckStartTime(job_id)
```

Cancelling a Job

```
arcca_tool.CancelJob(job_id)
```

File Management

Send File to Server

```
source_path = "my_local_file.txt" #path to local file
destination_path = "explicit/path/to/destination #absolute path to file on SCW filesystem

arcca_tool.SendFileToServer(source_path,destination_path)
```

Fetch File from Server

```
source_path = "explicit/path/to/destination #absolute path to file on SCW filesystem  
destination_path = "my_local_file.txt" #path to local file  
  
arcca_tool.FetchFileFromServer(source_path, destination_path)
```

List Remote Directory Contents

```
arcca_tool.ListRemoteDir(path)
```

Check Remote Path Exists

```
arcca_tool.CheckPathExists(path)
```

Check if Remote Path is a Directory

```
arcca_tool.CheckRemotePathIsDirectory(path)
```

Create Remote Directory

```
arcca_tool.CreateFolder(path)
```

Move Remote Files/Directories from a Remote Location to Another Remote Location

```
arcca_tool.MoveRemoteFile(remote_source, remote_destination)  
  
arcca_tool.MoveRemoteDirectory(remote_source, remote_destination)
```

Delete Files/Folders/Either

```
arcca_tool.RemoveRemoteDirectory(path) #only will delete path if it is a directory.  
  
arcca_tool.DeleteRemoteFile(path) #only will delete path if it is a file.  
  
arcca_tool.RemoveRemoteItem(path) #will check if path is a directory and then use appropriate removal function
```