Using evolutionary computing to find the best performing data augmentation for a given network

TLDLIR001

Liron Toledo
University of Cape Town
Rondebosch 7700
Cape Town, South Africa
liron.toledo@outlook.com

1 May 2018

ABSTRACT

Deep learning (DL) has rapidly become one of the most researched topics within the field of A.I. The increase in the study and application of DL has led to the huge increase in the demand for quality labeled data. However, large amounts of labeled data are not always readily available for certain problems. As such, regularisation techniques such as Data Augmentation (DA) have been developed and used extensively in order to mitigate the problem of overfitting on too little data and to better generalise deep neural networks. However, the number of different ways DAs can be combined and applied is vast and considering the fact that specific DAs work better on certain datasets than others has led to a number of attempts to automate the DA process. Neuroevolution (NE) has been found to be good at optimisation tasks and as such could prove to be a novel and effective tool in finding some optimal DA that for a given dataset and network architecture best minimises a network's error and maximises its accuracy.

KEYWORDS

Machine Learning; Supervised Learning; Deep Learning; Artificial Neural Network; ANNs; Convolutional Neural Networks; CNNS; Data Augmentation; DA; Evolutionary Computing; Evolutionary Algorithms; EA; NEAT; DeepNeat; CoDeepNeat

1 INTRODUCTION

Machine learning and supervised learning is increasingly being used in order to solve modern problems [23, 24] but as the use of machine learning continues to grow [24], there exists an increasing demand for useful labeled training data. Unfortunately, finding large amounts of labeled training data for certain tasks such as image classification can be extremely difficult [21, 29]. Not having enough quality labeled data to train on results in overfitting. This causes a neural network to become highly biased to the data it is trained on, and as a consequence, the trained network will not be able to generalize to other samples. Therefore, regularisation techniques were created to successfully train neural networks when training data is limited. One such regularization technique is called data augmentation (DA).

DA is a technique that artificially inflates the training data-set by performing transformations that maintain their relevant labels to add more uniform examples. DA can be considered as a type of regularisation that reduces the chance of overfitting by extracting more general information from the training data set and then passing that information to the network. The use of DA is especially important when their exists little or limited training data or when training data is difficult to find. It is extremely popular and widely used in deep learning and in particular image classification [1, 6, 12].

In summary, DA provides massive utility in regards to avoiding overfitting and better generalising the network. However, it has been found that certain data augmentations result in inflated data sets with better accuracy levels then others when used on the same network [1, 6, 12, 29]. As such we can infer that certain augmentations work better than others for specific networks. Normally one would have to manually test the different augmentations to see which results in better accuracy [1, 12] but this can be a long and arduous process, additionally, one might not even find the augmentation that would result in the best accuracy. Knowing this, how can one find a data augmentation to best increase the accuracy of the network without using a laborious trial and error strategy?

There have been attempts in the past to try find an optimal DA strategy with varying degrees of success [6, 17, 33]. However, based on the widespread success of evolutionary computing in similar optimisation problems [7, 8, 18, 20]. A potential novel solution not yet attempted is to use evolutionary computing, more specifically neuroevolution to find an optimal DA. Evolutionary computing refers to the group of algorithms used for optimisation purposes that are inspired and behave similarly to biological evolution. Evolutionary algorithms (EA) will replicate biological behaviour such as, reproduction, mutation, selection and recombination with the goal of selecting some mutation that returns the best results [31].

Through the use of neuroevolution (which uses EAs), one might be able to find a data augmentation or a combination of two or more augmentations that result in an inflated data that best generalises a neural network.

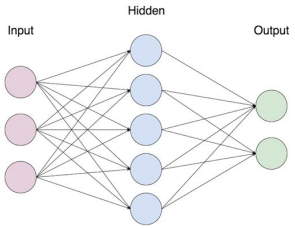

Figure 1: Graph representation of a simple ANN

This literature review will specifically evaluate and overview literature relating to machine learning, neuroevolution and data augmentation with the goal of using EAs to produce some optimal DA for a given neural network.

2 MACHINE LEARNING

Machine learning is a subcategory of artificial intelligence that deals with the creation of systems that automatically learn and improve from experience. It is one of the most researched and widely used applications of artificial intelligence in modern society [23, 24]. Given that they have access to the relevant training data, their ability to complete tasks without being explicitly programmed has proved invaluable in optimising certain problems such as image classification or integral in the use of fields such as robotics [23].

Machine learning is predominantly broken up into four separate sub-categories; supervised, unsupervised, semi-supervised and reinforcement learning [4]. Each category provides a method for how a system might learn how to solve a problem and are usually used for different sets of problems. The category most relevant to this paper is supervised learning.

2.1 Supervised Learning

Supervised learning involves a system deducing a function from labeled training data. In supervised learning, each training example is a pair consisting of an input value and a desired output value. The training data is then analysed and used to deduce a function which can be used to map examples outside of the training data. Ideally, the function found by analysing the training data should generalize such that the correct class labels for all unseen instances can be determined [25].

2.2 Artificial Neural Network (ANNs)

Supervised learning is regularly used in conjunction with artificial neural networks (ANNs) [4]. ANNs are systems loosely modelled after the human brain and nervous system. They are incredibly useful tools for finding patterns that would be impractical for a programmer to extract and teach the machine to recognize.

Simple ANNs (or perceptrons) as seen in figure 1 consist of an input layer, a hidden layer and an output layer that are linked by various weighted connections. When using supervised learning, after the input example is forward passed, the hidden layer extracts a set of high level features, eventually the network will create complex feature detectors that for example, in the case of image classification, can be used to figure out which image elements are commonly found together. Labels can then be given to the network's output. If the network's output does not match the input example's desired output, backpropagation can be used to correct any mistakes the network made during its forward pass and in doing so re-evaluate the values of the network's weights. After some time, the ANN will be able to perform its own classification tasks on unseen instances without any human intervention.

2.3 Deep Learning

Nevertheless, simple single layer ANNs are often not powerful enough to help solve more complex problems [26]. This is where the importance of deep learning can be highlighted. Deep learning refers to the creation of neural networks with multiple hidden layers. Larger networks learn categories incrementally through their multiple hidden layers. Using facial recognition as an example, Low-level categories like the light or dark areas are defined first before higher level categories like the lines and shape of the face are defined and so on and so forth until a nested hierarchy of categories are defined, with each category defined in relation to simpler ones.

In traditional machine learning, a domain expert needs to identify the majority of applied features in order to reduce the complexity of training data and make patterns more clear to learning algorithms. This can often be an extremely time consuming activity. The manner in which deep learning techniques incrementally try to learn high-level features from data eliminates the need for domain experts and manual feature engineering all together. This allows deep learning techniques to achieve far greater power and flexibility than traditional machine learning techniques [26].

However, large networks with many layers and connections are extremely computationally expensive to train and certain models can sometimes take weeks to fully train even with the use of powerful hardware [26, 22]. Additionally, in order to achieve satisfactory results, deep learning techniques require much larger amounts of training data (thousands of examples are often not enough) [22]. Training on too little data is likely to produce a model that is unlikely to outperform other approaches. In today's society, data may be abundant for certain problems but not always for others [21, 29].

Despite these disadvantages, deep learning has been used in many fields to much success [4]. One such relevant field is that of computer vision. In particular, there is one deep learning algorithm that has played an integral part in the advancements in computer vision over the years. This algorithm is known as the convolutional neural network (CNN).

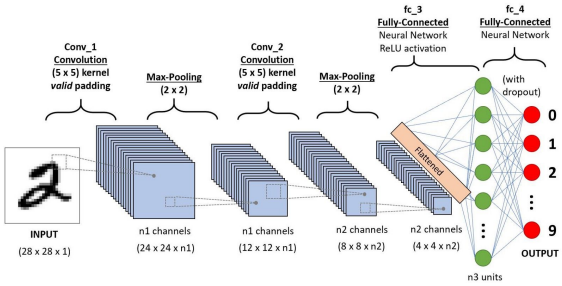

Figure 2: Representation of CNN

2.4 Convolutional Neural Network (CNNs)

A CNN is a deep learning algorithm which takes in an input image and then finds and allocates importance (weights and biases) to a multitude of different aspects in the image which can all be differentiated from one another. CNNs have found widespread success due to their pre-processing requirements being much less demanding than other deep learning classification algorithms and their consistent ability to perform complex classification tasks successfully [14].

As seen in figure 2, CNNs consist of a convolutional layer and a fully connected layer. The convolutional layer is able to capture the temporal and spatial dependencies in an image through the implementation of filters. These filters reduce the image (which is simply a matrix of pixels) into a form which is easier to process but does so in such a way that it still remembers the features which are vital for getting a satisfactory prediction. Once the input image has been converted into a suitable flattened form that still preserves its high-level features it is forward passed into the fully-connected layer which is a neural network that classifies the image.

CNNs can be extremely powerful but as mentioned earlier, deep learning algorithms such as CNNs require massive amounts of training data to perform well [26, 22]. The overfitting that arises from training on too little data can significantly affect the ability of CNNs to accurately classify unseen instances but why exactly does overfitting happen?

2.5 Overfitting & Underfitting

Nate Silver [28] defines the "signal" as the true underlying pattern you wish to determine from a data set (trends and patterns). Additionally, he defined "noise" as the non-important or random information in a data set (outliers). Noise can interfere with finding trends and patterns in data but a well trained neural network can isolate a dataset's signal from its noise. If the network's architecture is too complex or has been trained on too little data, it will end up memorising the noise rather than finding the signal and consequently will make predictions based on that noise. The model is known to be overfit if it performs excellently on the training data but poorly on unseen instances.

There also exists the opposite problem of underfitting [28], which occurs when a network's architecture is too simple or has not learned the features of a dataset correctly. Underfitting results in an ANN that won't be able to model the training data or generalize to unseen data. Underfitting can be avoided by building a architecture that better extrapolates the important features in a data set.

3 DEEP LEARNING WITH SMALL DATA

One way to avoid overfitting is to simply train on more data [27], however finding large amounts of data to adequately train on is not always possible. This is especially the case for deep learning algorithms which require huge amounts of training data to perform well. However, if one only has access to small amounts of training data there exists a group of methods that can be used to assist in avoiding overfitting. These methods are collectively known as regularisation.

3.1 Regularisation

Regularisation, simply put, attempts to restrict the network at training time. Regularisation methods attempt to limit the complexity of a network either in the network's weights or in the data that it uses. In doing so it addresses the two major causes for overfitting, a lack of quality labeled training data or a network architecture that is too complex [27]. The end goal being to eliminate overfitting and illicit faster generalisation.

3.2 Regularisation Techniques

There are many regularisation techniques that have been used in the past to much success [34, 35]. Some examples include Transfer Learning, Batch Normalisation, Active Learning, Dropout and Data Augmentation

The Dropout method randomly decides to not update specific neurons during backpropagation. By doing this, it reduces interdependence among the network's neurons. In other words, by randomly making some node not perform its job the other nodes must learn to perform the job in absent node's stead.

Batch Normalisation, in short, attempts to balance a network. This is achieved by normalising a network's hidden node values. In doing so, it decreases the amount the hidden values can change. This minimises overfitting because it effectively adds noise to every hidden layer output.

Transfer Learning is a technique that involves using the knowledge of a pre-trained model and applying it to another problem. This is achieved by using an ANN that already works well on some similar problem and retraining it on the data that is relevant to the desired task.

Active Learning is a method that creates more data for training. Along with a small amount of existing labeled data, Active Learning uses an ANN to decide which other data instances need to be labeled. After training the desired

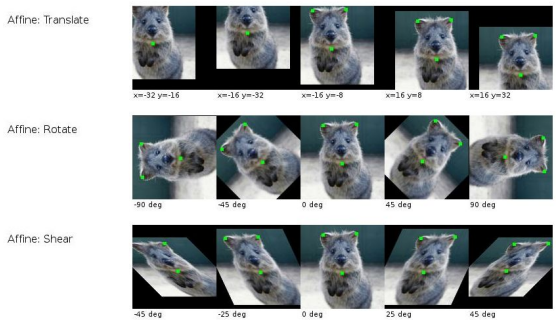

Figure 3: Examples of generic DA transformations

model on the existing labeled data, it is used to find the data instances that it is most unconfident in regarding its label. Quantifying this unconfidence can be hard but the easiest method to do so would be to take the distance the data instance is from the decision boundary. The ANN can

then flag all data instances that are too close to the decision boundary. All the flagged data instance can then be reviewed by a human, labeled accordingly and used for retraining.

Active learning may be useful in its ability to reduce the amount of data needed to be labeled but it also has some problems that can't be ignored. If the initial data set is not a good representation of the whole dataset it will cause issues and it can be very time consuming due to its need for humans to label data. However, there exists another regularisation technique that, similarly to Active Learning, creates more data for training but does so without any of its drawbacks. This method is called Data Augmentation.

4 DATA AUGMENTATION

Data Augmentation was previously defined as a regularisation technique to inflate a training data set by performing label preserving transformations to create more uniform examples. DA has proven to be excellent in reducing the overfitting in ANNs caused by training on too little data [36, 1, 6].

It has already been established that deep learning algorithms such as CNNs require huge amounts of data to perform well. Unfortunately, it is not always possible to find large amounts of quality labeled data for a given problem. As a consequence, it can be difficult to implement deep learning algorithms for certain tasks. However, it is possible to use DA to inflate small datasets such that we can get satisfactory performance out of ANNs and CNNs [1, 6, 12].

DA is especially popular for use in image classification. It has proven outstanding in improving the accuracy of large CNNs [6, 12] and as a result is used frequently in computer vision tasks. However, it is has also found success in other areas of study such as sound classification and voice recognition [11, 16].

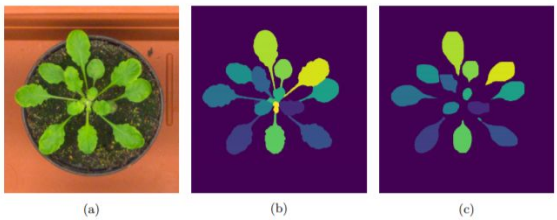

Figure 4: Use of synthetic models in [21] for leaf segmentation and counting

4.1 Types of Data Augmentation

One such DA scheme that is often used for image classification to great effect is the generic DA scheme [1, 12] as seen in figure 3. These are computationally inexpensive methods that consist of transformations that alter the geometry of an image (Geometric transformations) or the light or colour of an image (Photometric transformations). Through geometric and photometric transformations (e.g. rotation, cropping, greyscaling, colour jittering) one can create numerous new label preserving images to use in training a network. The use of Generic DA is ubiquitous in image classification due to its history of significantly reducing overfitting in CNNs [1, 12]

In recent studies, the Generative Adversarial Network (GAN) has been used extensively and successfully for data synthesis [32]. GANs can generate new samples after being trained on samples from some dataset. In [32] we see GANs being used to generate realistic images of flowers and birds based on text descriptions. GANs are often used in another increasingly popular DA scheme called complex DA [1]

Complex DA inflates the data set by using domain specific synthesization to produce more training examples. Complex DA methods have the ability to generate much more valuable training data when compared to the generic augmentation methods [1, 21]. For example, in [21] researchers attempting to use machine learning for leaf segmentation and counting tasks used generative semi-synthetic models as seen in figure 4 and applied various kinds of transformations to these models to inflate their training data set. Their DA method achieved state of the art results when applied to the standard benchmark in their field. Despite the powerful performance and utility of complex DAs, they are far more computationally expensive and time-consuming to implement when compared to generic DAs.

4.2 Data Augmentation Methods

Is has been mentioned but is important to note the significance that all augmentations are label preserving. An augmented image of a jet should still bear a resemblance to a jet if it is going to be useful to an ANN that classifies images of jets. Typically, if a human can recognise that whatever object depicted in the augmented image is the same object depicted in the original image, it can be

Figure 5: Images generated when using style transfer as a DA [9, 10]

considered label preserving [27]. Making augmented data label preserving allows for it to be easily and quickly added to the original training data set. Additionally, because all transformations are label preserving, it allows for multiple different DA to be implemented on a single instance of data to create compound augmented data. For example, two generic augmentations such as rotation and colour jittering could both be applied to the same image of a jet to create a new compound augmented image. It is unknown whether compounded transformations are more or less effective than individual ones when inflating a data set and as such definitely warrants further research.

Ensuring that transformations are label preserving is vital but when exactly do we perform these transformations? One can either augment data prior to training the ANN or during the training process. The experiments performed in [1] highlight the advantages and disadvantages of both approaches. Augmenting prior to training saves time

because during this period, no new images will have to be generated but this results in more space being taken up during an ANN's training. Augmenting during training doesn't use much space due to the fact that augmented data can be discarded once it has been used by the ANN, However, it does add additional overhead to training time, which in the case of deep learning algorithms can already take extremely long.

To further drive home the importance of label preserving transformations. Researches in [15] experimented with DA in data space and DA in feature space. DA in data space refers to any and all label preserving transformation. DA in feature space refers to transformations that cannot easily be validated as label preserving. As such DAs in feature space may have features handcrafted for them to extract important information on which the classifier performs learning. DA in data space out performed DA in feature space across all datasets when used with a variety of classifiers one of which was a CNN.

4.3 Creating Optimal Data Augmentations

How and when to apply DA has a significant effect on the quality of ANNs trained on inflated data sets but as seen in [1, 29] the type of transformations used can produce inflated data sets which produce ANNs with different levels of accuracies. If certain augmentations produce inflated data sets which produce ANNs with better accuracy than others, Is it possible to find a single DA that will increase accuracy across all datasets and architectures?

In the last few years, researchers have attempted to create new DAs that outperform the traditional ones regularly used but also increase the performance of ANNs no matter the dataset or architecture being used [5, 13, 9, 10]

One such attempt can be seen in [5] where there was an attempt to create a DA that outperforms the commonly used generic DAs through the development of a new DA technique called SampleParing. SampleParing synthesises a new image by taking two random labeled images in the dataset and simply averages the intensity of each pixel to create a new image. Experimentation on multiple different data sets concluded that using SampleParing improved the accuracy of their classifier (architecture remained largely the same throughout experimentation) significantly when compared to other generic augmentations.

Another example is shown in [13] which attempted to improve CNN accuracy by creating a new DA approach called "random erasure" that randomly occludes different parts of an image during each training iteration. Experimentation found "random erasure" to be effective in improving accuracy on a limited amount of different datasets and CNN architectures without being too computationally expensive.

A final example is showcased in [9, 10] where a technique known as style transfer was used in order to create a DA technique that would better improve the performance of CNNs and other computer vision tasks. Style transfer involves engineering digital images to take on the visual style of other images as seen in figure 5. By using style transfer, the texture and colour of an image can be changed while still preserving important features such as its shape. In both sources, experiments demonstrated that in comparison to traditional DA techniques, the use of style transfer resulted in remarkable improvements in the accuracy of many different computer vision tasks across a variety of different datasets and architectures but was very computationally expensive to implement and as such, a large amount of training time was required.

4.4 Automatic Augmentation

However, more recently, instead of trying to manually create DAs to improve performance, researchers have begun to develop strategies that attempt to automatically find some optimal DA that will result in the best possible performance increase for a specific network architecture and data set and do so without any human involvement [6, 17, 33]. In [6] researchers attempted to create a method named neural augmentation that involves using a neural network to learn augmentations that best improve a classifier's accuracy. At training time, the network stochastically selects two images from the training data set and outputs a single label preserving image as seen in figure 6. This new image is then fed into the classifying network in addition to the original training data. The training loss is then back-propagated across both the augmenting network and the classifying

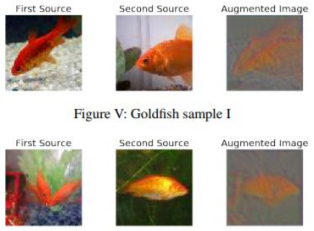

Figure V: Goldfish sample I

Figure 6: DA images generated from automatic DA system in [6]

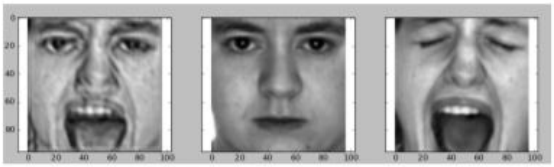

Figure 7: DA image generated from Smart Augmentation [17] (left image is generated from the two images on the right)

network and in doing so trains the model to identify the optimal augmentation for a given dataset. Experiments on classifying animals found that neural augmentations performed only slightly better than traditional augmentations and were on average three times more computationally expensive.

However this is contrasted in [17] where a almost identical method called Smart Augmentation is used. Much like in [6] Smart Augmentation utilises a deep neural network that learns how to generate augmented data as seen in figure 7 during the training stage of some target network and attempts to do so in a way that minimises the error of that target network. Experiments that were conducted on increasingly more difficult datasets concluded that Smart Augmentation increased the accuracy of the target network significantly across all data sets and achieved similar performance levels to traditional augmentation on smaller network sizes but struggled on larger ones.

Finally, a different approach can be seen in [33] a technique called AutoAugment was created to improve the accuracy of image classifiers. AutoAugment uses reinforcement learning to automatically search for the optimal augmentation or combination of augmentations to use from a search space of 2.9 x 10^{32} generic transformation possibilities. This process is visualised in figure 8. Additionally, AutoAugment predicts the per-image probability and magnitude of the transformation used and as such can prevent the an image from being engineered in the same way. It was found during experimentation that AutoAugment chose different augmentation policies for different datasets, this corroborates results found in papers such as [1, 29, 6] that suggest that certain augmentations work better than other for specific problems. AutoAugment performed

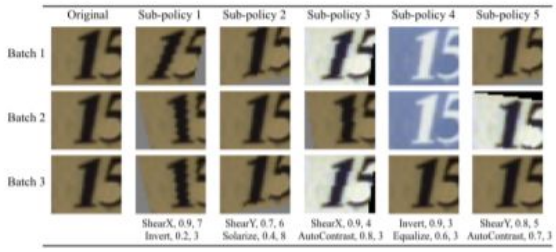

Figure 8: augmentation policies generated by AutoAugment [33]

exceptionally well on multiple data sets (achieved state of the art accuracies for imageNet data set) used and proved to be computationally efficient when compared to traditional augmentations. It is also worth noting that augmentation policies found by AutoAugment are transferable, meaning they can be reused in other computer vision tasks.

As seen in the previous methods, there have been a few successful attempts to create an automatic augmentation method. However, all of these methods use either deep neural networks or reinforcement learning to do so. There has yet to be a automatic augmentation method developed that utilises evolutionary algorithms and more specifically neuroevolution.

5 NEUROEVOLUTION

Evolutionary algorithms (EA) have previously been declared as a group of algorithms that replicate biological behaviour for optimisation purposes. EAs use darwinian evolution to find the fittests member in a population. The population includes all individuals created from some initial state. An individual's ability to complete a task is determined by some pre-defined fitness function. Individuals with good fitness (as well as some randomly chosen) are mutated or crossovered (fused with other individuals) to create a new generation. The process is then repeated until the fittest individual is found.

Neuro-evolution (NE) is a subset of A.I that uses evolutionary algorithms to evolve ANNs by manipulating their hyper-parameters, topology or rules [31]. Initially, NE was used only to evolve hyper-parameters [19, 31]. Later iterations, namely the topology and weight evolving neural networks (TWEANNs) NE algorithms search for optimal topologies and weights for a network [19, 31]. Additionally, NE algorithms have been developed that only tweak the hyper-parameters and topology of a network while weight updates are handled by backpropagation [2, 3]. However, because this method has to train multiple networks with backpropagation each generation it can takes significantly longer than older NE methods. Training networks concurrently or on different machines can alleviate some of the extra training time.

5.1 Neuro-Evolution of Augmenting Topologies (NEAT)

NEAT is a member of the previously mentioned TWEANN group of NE algorithms [19, 31, 20]. Meaning that it attempts to find a network's optimal weights and its best topology. NEAT achieves this by evolving the network's nodes (or neurons) in addition to the links between these nodes. Furthermore, it attempts to optimise the weights of the network's connections on each iteration of the topology.

In traditional EAs there exists a common problem called competing conventions [19]. This occurs when two mutations that encode homogenous behaviour but with very different genomes crossover. In other words, when two networks that have a similar fitness but radically different topologies crossover. The children (mixed topologies) of these crossovers often perform worse than either of their parents because they are unlikely to take the best aspects of two networks that are so vastly different.

NEAT solves this problem through the use of speciation [19]. Speciation involves clustering the population by some similarity metric. NEAT achieves this through the use of the historical marking of genes (or individuals) in the population.

Historical markers provide knowledge about the ancestral history of a genes and as such can be used to group genes that share a common ancestry. This makes historical marking a perfect tool to group genes into different species. As a consequence of speciation by historical marking, the problem of competing conventions is much less severe as only genes with a common parents can be crossovered.

Additionally, speciation allows for the protection of innovation [19]. Topological innovation is the act of adding a new structure to a network as seen in figure 9. Initially, topological innovation in a network can cause it to become less optimized and as such make it receive worse fitness scores but in time it could may innovate itself and prove to be a fit gene. Because speciation causes genes to only compete within their own species. They are protected from other genes in the population that may be more optimised (have higher fitness) but have worse architectures. As such they are given the ability to grow and evolve when they otherwise would have been eliminated.

A summary of how the NEAT algorithm works is as follows. NEAT begins with evaluating a small number of initial ANNs. The initial population is kept small to avoid NEAT wasting time on evaluating and mutating inefficient topologies. In doing so, NEAT avoids creating ineffective species. NEAT then proceeds to evaluate every ANNs and assign each a fitness score. Those with the highest fitness scores (Top performers) may be crossed over but only if their historical markings match up while others may be randomly mutated. Mutations involve either changing the topology (adding nodes and connections) or the weights of the ANN. All new children created will be grouped into their respective species as the next generation. The process

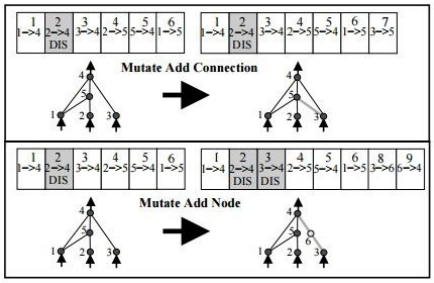

Figure 9: visual representation of node and connection mutations in NEAT

will then repeat itself until an ANN is found with the best fitness.

5.2 DeepNeat

DeepNeat, as one might assume, is an extension of the NEAT algorithm but is used in the creation of deep neural networks [2]. DeepNeat still utilises NEAT's method of speciation by historical marking to avoid negative crossovers but, it has two major differences from its predecessor. The first being that DeepNeat is not a TWEANN algorithm. Unlike NEAT, DeepNeat is only concerned with the hyper-parameters and topology of genes and does not attempt to optimise weights. Weights are instead updated during backpropagation. The second difference being that unlike in NEAT where every node in a gene (graph) represents a single neuron, in deepNeat, every node now represents a different layer in the ANN. The connections between nodes in DeepNeat are not weighted as they only indicate how layers pass their outputs to each other. Nodes ordinarily represent convolutional, fully-connected or recurrent layers.

Because DeepNeat is not a TWEANN algorithm, nodes only contain information on the hyper-parameters of that layer and not the weights, which are handled in backpropagation. DeepNeat represents genes as abstracted deep neural networks and as such needs to be able to construct genes into actual deep neural networks during evaluation. This is done by using the hyper-parameters contained within each node. These hyper-parameters represent the important information needed to construct the entire deep ANN that genes represent. This information includes things such as, activation functions, number of nodes in the layer, type of layer, anything else of relevance. This allows for graphs to be easily constructed during evaluation.

The deep ANNs assembled from their abstract representation are trained with backpropagation on the relevant training data. The fitness function is then used to judge the accuracy of the trained ANN. Similarly to NEAT, mutations and crossovers will then be performed to generate the next generation where the process will begin again.

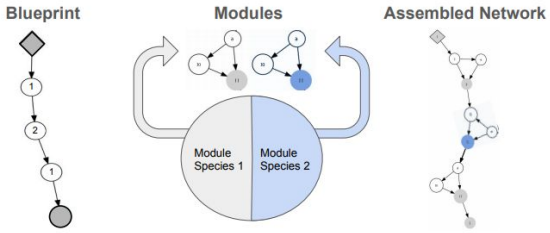

Figure 10: Visual representation of modules and blueprints used in CoDeepNeat [3]

5.3 CoDeepNeat

CoDeepNeat extends DeepNeat with one major adjustment to how the algorithm operates [3]. This adjustment is the use of a common EA technique named cooperative evolution [3]. Cooperative evolution works by combining similar individuals together to find more complex ones. In the context of NEAT, this means combining smaller and more simple ANNs into bigger and more complex ones.

Essential to the operation of CoDeepNeat is the system of modules and blueprints [3] as seen in figure 10. Modules represent abstracted deep neural networks in the same way that individuals are represented in DeepNeat. Meaning that every node in the graph represents a layer in the ANN. A big difference from DeepNeat is the use of unique feature called Blueprints. Blueprints take the form of graphs where every node points to a specific species of modules. Unlike in DeepNeat, CoDeepNeat constructs large and complex ANNs by amalgamating simple modules instead of focusing on manipulating the modules themselves.

A Blueprint is used to construct a graph of connected modules by transforming every node in its graph into a stochastically selected module from the species that the node is pointing to. However, once a node has randomly chosen a module from a species, if another node points to the same species, it will choose the same module previously chosen from that species. Meaning that no blueprint can have more than one unique module from the same species.

The use of Cooperative evolution is evident in how blueprints are used as the graph created by the blueprint is essentially a larger more complex graph assembled by fusing together many different smaller networks. The graph created can then be parsed into a practical neural network and trained for a limited amount of time on some training data set. After training, the accuracy of the network determines the fitness of all the modules used in the network as well as the blueprint. The fitness value is then used as the basis for decision making regarding the mutation and crossover on all the modules as well as the blueprint. CoDeepNeat will eventually find some top performing network which will then be taken and trained such that it converges correctly on unseen data sets.

CoDeepNeat is a primary example of automatic machine learning (AutoML). AutoML systems attempt to optimise ANNs without requiring any expert knowledge from humans. Currently however, AutoML methods take significant amount of time to compute. Unfortunately CoDeepNeat as well as DeepNeat are not exceptions. CoDeepNeat's biggest drawback is its long training time. Because the whole population of deep ANNs have to be trained with backpropagation every generation such they can converge satisfactorily. In [3] it took researchers over 9000 CPU hours in order to get their best results using CoDeepNeat. This can be a massive barrier to researchers who do not have the necessary resources to train the ANNs concurrently (or in parallel). However, computing resources are becoming increasingly faster, cheaper and easier to acquire, meaning that before long AutoML systems will take much less time to compute. A such the potential applications of their use should certainly be researched further.

6 CONCLUSION

There exists many methods of creating deep learning models with minimal data. Through the use of regularisation techniques such as DA, researches have been able to successfully generalise networks and solve problems which in the past would have been impossible. DA in particular has been used successfully in a number of projects ranging from image classification [1, 6, 17, 33] to voice recognition and sound classification [11, 16].

However, the extensive use of DA brings forth a new problem. It was found that specific augmentations worked better with certain datasets than others [1, 6, 12, 29]. That being the case, how does one choose an augmentation that results in the best accuracy? Traditionally, a number of DAs would be manually experimented with to see which DA results in a dataset that minimises error and maximises accuracy [1, 6, 12] but this approach proved to be very strenuous and time consuming.

As a consequence, there has been an effort to find DAs that perform well no matter the dataset or architecture used [5, 9, 10, 13]. Certain methods such as style transfer [9, 10] showed extremely promising results but as seen in [29, 1, 12] certain datasets and architectures react in different ways to different DA methods and as such it is impossible to guarantee that a single DA can return the best results all the time.

As a result, in recent studies, there has been a limited amount of successful attempts to generate some system that can automatically determine and create the best possible augmentation for a given dataset and architecture [6, 17, 33]. However, all attempts thus far have utilised either ANNs or reinforcement learning to find an optimal DA.

Given the progress and success found in using NE and autoML for optimisation problems [19,2,3], the use of NE and AutoML could prove to be a potential novel approach to the creation of an automatic DA system. Based on the best in class performance of NE algorithms like CoDeepNeat, it is not hard to imagine designing a system that uses NE to find some top performing DA that best improves accuracy for a specific dataset and architecture much like how DeepNeat or CoDeepNeat find top performing ANNs.

REFERENCES

[1] Taylor, L. and Nitschke, G., 2018, November. Improving Deep Learning with Generic Data Augmentation. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 1542-1547). IEEE.

[2] Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., Raju, B., Shahrzad, H., Navruzyan, A., Duffy, N. and Hodjat, B., 2019. Evolving deep neural networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing* (pp. 293-312). Academic Press.

[3] Liang, J., Meyerson, E., Hodjat, B., Fink, D., Mutch, K. and Miikkulainen, R., 2019. Evolutionary Neural AutoML for Deep Learning. *arXiv preprint arXiv:1902.06827*.

[4] Jones, D. and Nitschke, G., 2019. Evolutionary Deep Learning to Identify Galaxies in the Zone of Avoidance. *arXiv preprint arXiv:1903.07461*.

[5] Inoue, H., 2018. Data augmentation by pairing samples for images classification. *arXiv preprint arXiv:1801.02929*.

[6] Wang, J. and Perez, L., 2017. The effectiveness of data augmentation in image classification using deep learning. *Convolutional Neural Networks Vis. Recognit*.

[7] Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y.L., Tan, J., Le, Q.V. and Kurakin, A., 2017, August. Large-scale evolution of image classifiers. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (pp. 2902-2911). JMLR. org.

[8] Real, E., Aggarwal, A., Huang, Y. and Le, Q.V., 2018. Regularized evolution for image classifier architecture search. arXiv preprint arXiv:1802.01548.

[9] Zheng, Xu & Chalasani, Tejo & Ghosal, Koustav & Lutz, Sebastian & Smolic, Aljoscha., 2019. STaDA: Style Transfer as Data Augmentation. 107-114. 10.5220/0007353401070114.

[10] Jackson, P.T., Atapour-Abarghouei, A., Bonner, S., Breckon, T. and Obara, B., 2018. Style Augmentation: Data Augmentation via Style Randomization. arXiv preprint arXiv:1809.05375.

[11] Salamon, J. and Bello, J.P., 2017. Deep convolutional neural networks and data augmentation for environmental sound classification. IEEE Signal Processing Letters, 24(3), pp.279-283.

[12] Ding, J., Chen, B., Liu, H. and Huang, M., 2016. Convolutional neural network with data augmentation for SAR target recognition. IEEE Geoscience and remote sensing letters, 13(3), pp.364-368.

[13] Zhong, Z., Zheng, L., Kang, G., Li, S. and Yang, Y., 2017. Random erasing data augmentation. arXiv preprint arXiv:1708.04896.

[14] Sharif Razavian, A., Azizpour, H., Sullivan, J. and Carlsson, S., 2014. CNN features off-the-shelf: an astounding baseline for recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops (pp. 806-813).

[15] Wong, S.C., Gatt, A., Stamatescu, V. and McDonnell, M.D., 2016, November. Understanding data augmentation for classification: when to warp?. In 2016 international conference on digital image computing: techniques and applications (DICTA) (pp. 1-6). IEEE.

[16] Cui, X., Goel, V. and Kingsbury, B., 2015. Data augmentation for deep neural network acoustic modeling. IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP), 23(9), pp.1469-1477.

[17] Lemley, J., Bazrafkan, S. and Corcoran, P., 2017. Smart augmentation learning an optimal data augmentation strategy. IEEE Access, 5, pp.5858-5869.

[18] Suganuma, M., Shirakawa, S. and Nagao, T., 2017, July. A genetic programming approach to designing convolutional neural network architectures. In Proceedings of the Genetic and Evolutionary Computation Conference (pp. 497-504). ACM.

[19] Stanley, K.O. and Miikkulainen, R., 2002. Evolving neural networks through augmenting topologies. Evolutionary computation, 10(2), pp.99-127.

[20] Zhang, H., Kiranyaz, S. and Gabbouj, M., 2018. Finding Better Topologies for Deep Convolutional Neural Networks by Evolution. arXiv preprint arXiv:1809.03242.

[21] Kuznichov, D., Zvirin, A., Honen, Y. and Kimmel, R., 2019. Data Augmentation for Leaf Segmentation and Counting Tasks in Rosette Plants. arXiv preprint arXiv:1903.08583.

[22] Brownlee, J. (2019). What is Deep Learning?. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/what-is-deep-learning/ [Accessed 1 May 2019].

[23] Canuma, P. (2019). What problems can we solve using AI?. [online] Towards Data Science. Available at: https://towardsdatascience.com/what-problems-can-we-solve-using-ai-ec7131f8159b [Accessed 1 May 2019].

[24] Young, S. (2019). 10 trends of Artificial Intelligence (AI) in 2019. [online] Becoming Human: Artificial Intelligence Magazine. Available at: https://becominghuman.ai/10-trends-of-artificial-intelligence-ai-in-2019-65d8a373b6e6 [Accessed 1 May 2019].

[25] McNulty, E., Zhang, J., Poughia, E., Desens, C., Zhang, J., Poughia, E., Desens, C., Zhang, J., Poughia, E. and Desens, C. (2019). What's The Difference Between Supervised and Unsupervised Learning? - Dataconomy.

[online] Dataconomy. Available at: https://dataconomy.com/2015/01/whats-the-difference-between-supervised-and-unsupervised-learning/ [Accessed 1 May 2019].

[26] Mahapatra, S. (2019). Why Deep Learning over Traditional Machine Learning?. [online] Towards Data Science. Available at: https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063 [Accessed 1 May 2019].

[27] EliteDataScience. (2019). Overfitting in Machine Learning: What It Is and How to Prevent It. [online] Available at: https://elitedatascience.com/overfitting-in-machine-learning#signal-vs-noise [Accessed 1 May 2019].

[28] SILVER, N. (2012). The signal and the noise: why so many predictions fail--but some don't. New York, Penguin Press.

[29] Hussain, Z., Gimenez, F., Yi, D. and Rubin, D., 2017. Differential data augmentation techniques for medical imaging classification tasks. In AMIA Annual Symposium Proceedings (Vol. 2017, p. 979). American Medical Informatics Association.

[30] Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y.L., Tan, J., Le, Q.V. and Kurakin, A., 2017, August. Large-scale evolution of image classifiers. In Proceedings of the 34th International Conference on Machine Learning-Volume 70 (pp. 2902-2911). JMLR. org.

[31] Floreano, D., Dürr, P. and Mattiussi, C., 2008. Neuroevolution: from architectures to learning. Evolutionary Intelligence, 1(1), pp.47-62.

[32] Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X. and Metaxas, D.N., 2017. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision (pp. 5907-5915).

[33] Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V. and Le, Q.V., 2018. Autoaugment: Learning augmentation policies from data. arXiv preprint arXiv:1805.09501.

[34] Ioffe, S. and Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.

[35] Jain, S. (2019). An Overview of Regularization Techniques in Deep Learning (with Python code). [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/ [Accessed 1 May 2019].