Deep learning with small data

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

Sasha Abramowitz
University of Cape Town
Cape Town, South Africa
reallysasha@gmail.com

Shane Acton
University of Cape Town
Cape Town, South Africa
shaneacton745.sa@gmail.com

1 PROBLEM DESCRIPTION

Training deep Convolutional Artificial Neural Network (CNN) classifiers can yield accurate models provided you have enough data to train towards generalisation [19]. Many problem domains however, suffer from a lack of quality training data. It is thus useful to explore methods that allow deep-CNN's to train better with smaller datasets. For our purposes we will define two terms. First we will define data efficiency (DE), which is the property of an ANN that describes how well the ANN can perform with the amount of training data it is provided. This is to say that DE is proportional to a trained networks accuracy and inversely proportional to the amount of data it was trained on. Secondly we will define small datasets as datasets which contain fewer than 1000 training examples per class. For a small data benchmark we will be using the LUNA16 grand-challenge [22] which is a 2 class, 888 total examples training set with an accompanying testing set.

AutoML is the process of automating the design of machine learning solutions. We will be employing autoML solutions to deep-CNN's. Specifically we will be using Neuro-Evolution (NE), which is the application of evolutionary algorithms (EA) in the design of ANN's. We will be using a variation of the popular CoDeepNEAT (CDN) [1, 2] genetic algorithm which we will design and implement to try and offer an AutoML solution to deep-CNN classification, with a focus on DE. AutoML has shown to be able to create state-of-the-art ANN's [1], with the notable requirement that it needs enormous computing resources. We will thus be exploring methods of incorporating Lamarckian weight inheritance into CDN to speed up the training time of evolved topologies. Lamarckian weight inheritance is the idea of using lamarckian inheritance [17] to pass on parent ANN's weight values to children (see Section 4.2). If done correctly this enables child ANN's to be evaluated with fewer training epochs. This will be our first major adaptation to CDN, and it was inspired by the work on the LEMONADE [4] NE algorithm.

Co-evolution in Deep Neuroevolution of Augmenting Topologies (CoDeepNEAT) [1, 2] is a genetic algorithm which supports Pareto-Non-Dominated Multi-Objective Optimisation (MOO). The co-evolution refers to co-evolving populations of 'blueprints' and 'modules', where modules are segments of Deep Neural Networks (DNN), and blueprints are graphs indicating how to connect modules into complete DNN's.

A successful approaches to promoting DE during training is to use a process known as Data Augmentation (DA). DA is the process of using label preserving transformations to your data, to artificially inflate your training set size, to allow for better generalisation [19, 5, 7]. DA is considered a regularisation scheme and is known to be able to reduce overfitting [19]. For a transformation to be 'label preserving', the resulting transformation of the data should retain its original class. More formally, the transformation can be represented by the mapping:

$$\theta : (X, Y) \rightarrow (X', Y)$$

Where θ is the transformation, the input (X) is of class Y and the output (X' which is distinct from X) remains in class Y.

We propose a novel method of including DA schema in the evolvable hyperparameter set of the CDN algorithm to promote DE. Hyperparameters are all parameters of an ANN other than its weights. This will be our second major adaptation to CDN. Since the number of independent DA transformations which can be performed on images is large [5, 7] and the combination space of these methods is enormous, we believe that using a black-box optimiser such as CDN will allow maximum utility of DA, and aid in allowing networks to train on smaller datasets. Previous work on automatically generating DA schema have focused on training ANNs and Reinforcement Learning agents to create optimal schema, and have shown to be successful [7, 12, 16]. We will be investigating two methods of evolving DA schemes. The first is including DA scheme hyperparameters at the blueprint level of CDN. The second is co-evolving a third population, alongside modules and blueprints, of DA schemes, and linking this population to blueprints in much the same way that CDN links modules to blueprints [2].

Multi-Objective Neuro Evolution (MO-NE) is the class of Neuroevolution algorithms which allow for optimisation of multiple objective metric simultaneously. Many successful approaches (including CDN [1, 2]) use a concept called Pareto based non-dominated sorting, which is used to rank objective metric tuples into fronts where each successive front dominates (outperforms in all metrics) the next [23]. The reason for this type of sorting is that it does not implicitly preference any one objective over another, and does not assume static tradeoffs between the objectives either. Pareto-based MOO concludes with a set of distinct solutions, each offering a unique tradeoff between objectives. CDN uses an unnamed implementation of

Pareto-based sorting. Upon analysis, we hypothesise that the method used biases solutions towards the 'primary objective' and does not satisfactorily explore the tradeoff space between the objectives. Thus our third proposed improvement to CDN will be to use NSGA-II [18] a non-dominated sorting algorithm in place of the one used in CDN.

The secondary objective used by Liang, J et al. [1, 2] for CDN was to minimise network complexity. They found that biasing less complex networks allowed for greater training generalisation '*This finding suggests that minimizing network complexity produces a regularization effect that also improves the generalization of the network.*' [1]. This is inline with the common intuition that larger networks have a greater ability to overfit. Since minimising network complexity indirectly improves DE we hypothesis that switching out network complexity for a more direct and explicit DE metric as a second objective may provide even greater generalisation capabilities.

2 PROBLEM STATEMENT

Design methods to aid the NE algorithm CDN in training on smaller datasets, as well as speed up the convergence time to lessen the computational burden of autoML. We will also be investigating an improvement to the MOO aspects of CDN by using an algorithm known as NSGA-II.

To achieve this we propose three modifications to CDN. First, adding another evolutionary process which selects the best combination of DAs to train with, we believe this will promote DE. Second, adding a form of lamarckian weight inheritance which we believe will decrease computation time. Third, replace CDN's non-dominant sorting with NSGA-II as a pareto front exploration method to address the shortcomings of the MOO used in CDN mentioned in Section 1.

The three distinct goals of the project are:

1. Determine if one can adapt CDN to create an automated system that finds optimal DA schemes for a given data set and network architecture.
2. Compare the accuracy, training time and solution exploration of original CDN (as designed in [1, 2]) versus CDN with our proposed modifications (see section 3.1). Experiments would consist of testing our modifications both independently and simultaneously.
3. Establish if building a metric for the DE of a network could aid the MO-NE algorithm CDN in creating networks which are maximally efficient with limited training data.

These goals will contribute to image classification by attempting to improve the existing state-of-the-art CDN method [1]. Goal one suggests a potentially new way to automatically augment data, this is commonly used as a regularisation technique and if proven successful could have implications outside the realm of autoML. Goal two offers approaches to improve the training time and predictive accuracy of CDN and thus allow it to be more efficient.

Extensions	Accuracy	Training time	Exploration
DACDN	++	++	=
LCDN	-	--	=
NCDN	-	=	+

Table 1: Expected improvements gained from each modification. ++ implies greater increase than +.

Goal three tries to leverage tailored MOO metrics so that the most data efficient networks are generated (see section 5.2). This may allow CDN and other MO-NE solutions to work with fewer data points. Generating ANN's which are data efficient using AutoML may allow the generation of ideal pre-trained networks to be used later for transfer learning (TL) [20]. TL is the process of using an ANN pre-trained to perform one task and retraining it on data relevant to another task. Pre-trained networks made to be maximally data efficient, may allow faster retraining in TL.

3 PROCEDURES AND METHODS

Three separate experiments will be conducted. These experiments will attempt to improve and measure CDN in respect to DE, accuracy and training time.

3.1 CoDeepNEAT extensions

3.1.1 Lamarckian evolution

Our first proposed extension to CDN will be referred to as *LCDN*. Lamarckian weight inheritance will be added into CDN immediately after the reproduction phase. Therefore as soon as a child blueprint has selected the modules that it will use, the most architecturally similar parent will pass its weights to the new network. The parents topology is unlikely to exactly match the child's, therefore carefully designed methods will have to be used in order for the child's inherited weights to best approximate the function the parent ANN represents.

3.1.2 Data augmentation

Our second proposed extension to CDN will be referred to as *DACDN*. This will allow blueprints to *choose* which DAs will be performed on their training data. Since we cannot know what combination of DAs the EA will choose we cannot create and store the augmented images beforehand. Thus the DA will take place immediately before training. However the selection of DA schemes will either be mutated along with blueprints or as its own co-evolutionary pool, we will run tests to determine which implementation performs better. Certain DAs (such as complex DAs) will likely have a measurable impact on training time which will need to be mitigated. A possible mitigation strategy would be to incorporate an additional objective that penalises computationally expensive DAs.

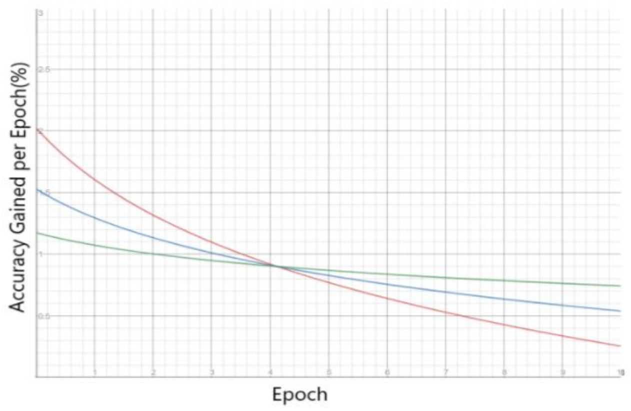


Figure 1: Hypothetical results for three different networks being trained on the same dataset

3.1.3 NSGA-II

Our third proposed extension to CDN will be referred to as NCDN. NSGA-II will replace the pareto front optimisation algorithm used by CDN at the selection phase. We hypothesise that this will allow CDN to better explore the pareto front and produce more diverse results. CDN puts more emphasis on the primary objective (accuracy) [1, 2] and therefore is unlikely to uniformly explore the pareto front. This improved pareto front diversity will allow the end users more choice as to which objective is more valuable.

3.2 Optimizing Data Augmentation Schemes

DACDN, our proposed automatic DA system will be benchmarked against hand-crafted DA schemes as well as an existing state-of-the-art automatic DA system, AutoAugment [16]. Evaluation will primarily be made based on best case accuracy gains by comparing performance with and without DA in each case, using the same dataset as AutoAugment [16] (ImageNet). If time allows, DACDN can additionally be compared to the automatic DA systems found in [12, 7]. This would require either the implementation of these systems on the same data sets used in our initial comparison or the evaluation of DACDN on the datasets used in [12, 7].

3.3 Benchmarking Modified CoDeepNEAT

This experiment will compare our modified CDN against the our implementation of the original. To achieve this, three metrics will be evaluated namely, accuracy, training time and pareto front exploration. We intend to build CDN in a way that allows extensions to be added in any combination of ways. Thus, once we have obtained results comparing the performance of our modifications to the original, we will combine the best performing modifications (which may be all of the above: *LN-DACDN*) and compare the newly modified version to the original. This will allow us to find the best modification or combination of modifications and to ascertain if any of these modifications improve the original CDN.

3.4 Using Multi-objectivity To Improve Data Efficiency

We will be comparing Network Size (Complexity) and DE as a second objective in MO-NE, for promoting generalisation capabilities in generated networks. This requires an objective metric for DE.

Ideally to test two networks and decide which was more data efficient one would train each on various sized subsets of the full training set and rank based on testing accuracy when using the smaller subsets, while controlling for overall performance at each subset size. This is however very computationally wasteful. Our proposed compromise is to train initial epochs with smaller subsets of the training set, and incrementally increase the subset size to 100% of training set as the epoch number approaches the last epoch. We hypothesise that networks which gain more testing accuracy earlier on (figure 1 red), are likely to be more data efficient. Where as networks which struggle in earlier epochs on smaller subsets and then start performing well later on the larger subsets (figure 1 green) are likely less data efficient. We then propose a DE metric which reflects the distribution of the gained accuracy, where distributions skewed to the earlier epochs are scored a higher DE and distributions skewed to the later epochs are scored lower DE.

4 RELATED WORK

4.1 Deep Learning

Deep learning refers to the creation of ANNs with multiple hidden layers. A CNN is a deep learning architecture used for computer vision tasks. In a CNN, images are initially passed through convolutions, so that they are downsized, then passed through fully connected layers. CNNs are currently state-of-the-art in the domain of image recognition [19].

4.2 Evolutionary Algorithms

EAs are a type of optimization method that use the principles of darwinian evolution to evolve a population. In darwinian evolution only the fittests survive, therefore in EAs there needs to be a way to find the fittest members (individuals) from the population. This is done using a predefined fitness function which is a measure of how good an individual is at a specific task. Individuals with good fitness are selected for crossover (analogous to sexual reproduction) and mutated to create new generations. The process is then repeated for a specified number of generations. Finally, the fittest individual is chosen from the final generation. EAs lay the groundwork for all NE methods.

4.3 Neuroevolution

4.3.1 CoDeepNEAT

The performance of an ANN is highly dependant on its topology. NEAT [3] is a traditional NE method. It attempts to find both a network's optimal topology and weights using evolutionary methods. However the main difficulty when using EAs to search for better ANN topologies is that during crossover of two good solutions a damaged offspring could be produced. NEAT solves this by introducing historical markings. These markings allow individuals to keep track of their parents. As such using this system NEAT will only crossover networks with overlapping parents which minimizes the cases where crossover damages offspring [3] and maximizes its efficiency.

CDN is a recently developed co-evolutionary, multi-objective NE algorithm which has achieved state-of-the-art results [1]. As the name implies CDN, is an extension of NEAT, however it is significantly different. First, it is not a topology and weight evolving neural network, instead CDN updates its weights via backpropagation. Second, individuals are co-evolved using a system of blueprints and modules. However, CDN still uses NEATs main contribution, historical markings.

The blueprint and module system is the most significant contribution of CDN [1, 2]. Blueprints are graphs where each node points to a specific species. modules represent an ANN as a graph, where each node in the module is a layer in the ANN. Each module belongs to a species.

CDN works as follows; select individuals for mutation and crossover using non-dominated sorting of the pareto front. The selected individuals are assembled into networks from blueprints by transforming each node in the blueprint into a randomly selected module from the species it was pointing to and then transforming each module into the ANN it represents. Finally all created individuals are trained and evaluated. The evaluation scores for each objective are returned to the blueprints as the fitness for that objective. For any module its fitness is the average fitness of all blueprints that used it.

4.3.2 LEMONADE

LEMONADE is similar to CDN, in that it achieves the same goal, but uses intelligent methods to greatly increase the efficiency. LEMONADE stands for **L**amarckian **E**volutionary algorithm for **M**ulti-**O**bjective **N**eural **A**rchitecture **DE**sign. This encapsulates the idea of the algorithm quite nicely, as it uses Lamarckian inheritance (which is a means of passing an individual's skills to its children [17]) to greatly decrease training time when compared to other NE methods [4]. This inheritance takes the weights from a similar parent network and uses them as the initial weights for the current network.

4.4 Data Augmentation

DA is a regularisation technique that artificially inflates the training data-set by performing label-preserving transformations to add more uniform examples. It 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 that better generalise and increase the accuracy of networks when used on the same data set [5, 7, 10, 14]. As such, we can infer that certain augmentations work better than others for different network architectures and datasets

4.4.1 Generic Data Augmentations

Generic augmentations consist of computationally inexpensive transformations that alter the geometry of an image (Geometric transformations) or the light or colour of an image (Photometric transformations).

Some examples of generic DA schemes include:

1. Rotating / Cropping / Colour jittering
2. SamplePairing [6]
3. Random Erasure [11]

4.4.2 Complex Data Augmentation

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 but are far more computationally expensive.

Some examples of complex DA schemes include:

1. Generative synthetic models [13]
2. Image synthisation through GANs [15]
3. Style transfer as DA [8, 9].

4.4.3 Automatic Data Augmentation

Automatic DA refers to systems that use strategies 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

Notable examples of automatic DA systems:

1. Neural augmentation [7]
2. Smart Augmentation [12]
3. AutoAugment [16]

5 ANTICIPATED OUTCOMES

5.1 System

Our system will consist of CDN at the core, with 3 swappable modules. Namely Lamarckian weight evolution, NSGA-II and evolvable DA schemes.

5.1.1 LCDN

CDN making use of Lamarckian weight inheritance to speed up training time of networks

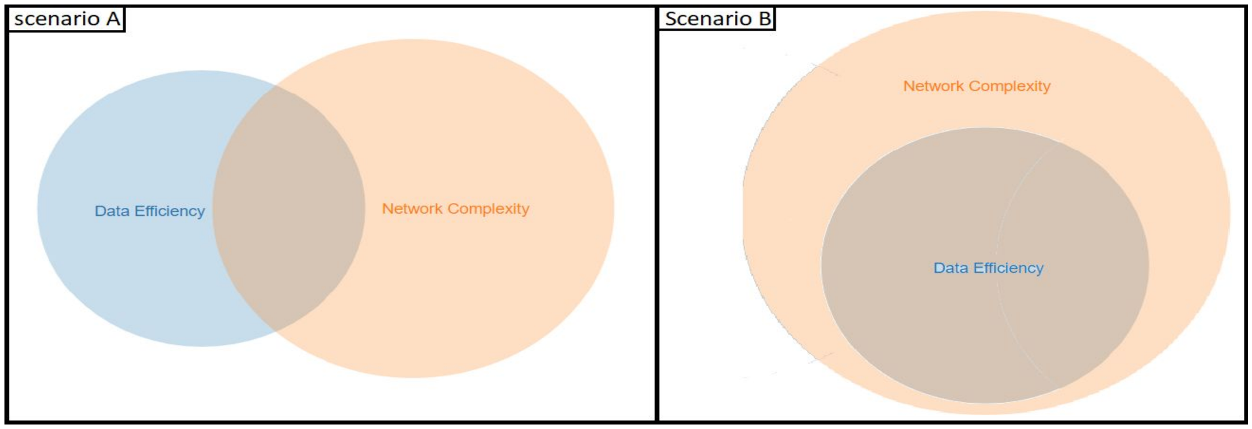


Figure 2: Possible relationships between the variations of network complexity and Data Efficiency

5.1.2 NCDN

CDN using NSGA-II non-dominated sorting in place of the one used in CDN

5.1.3 DACDN

CDN which evolves DAs through mutation and crossover. DAs are chosen from a large pool of both generic and complex DAs.

5.1.4 LNCDN, N-DACDN, L-DACDN

Each of the above mentioned versions of CDN have two of the three possible extensions activated.

5.1.5 LN-DACDN

CDN which incorporates all three described modules.

5.1.6 Testing suite

A subsystem which orchestrates the execution of experiments as well as data recording, preparations, and presentation for the results of the experiments.

5.2 Contributions

5.2.1 Optimising Data Augmentation schemes.

It is our belief that DACDN will return DA schemes that outperform hand-crafted DAs in terms of accuracy, but will require long training times to do so. Additionally, we hypothesise that DACDN will be less accurate in comparison to AutoAugment [16] but will be easier to implement. The potential impact of creating such a system is the availability of an Auto ML DA system that returns close to state-of-the-art accuracy results but can be applied considerably more easily and requires no direct human involvement.

5.2.2 Benchmarking modified CoDeepNEAT

We hypothesise that the experimental results when comparing the modified versions of CDN against the original will be in line with table 1. However when testing all three modifications in conjunction we believe LN-DACDN to have favourable training time versus performance trade-off. Thus we believe that LN-DACDN will have the best improvement spread over accuracy, training time and pareto front exploration.

5.2.3 Using multi-objectivity to improve data efficiency

The findings from paper [2] show that there is a relationship between network complexity and DE such that Decreasing complexity increases DE. We hypothesise that ANN complexity is not the only factor which influences DE, as is the case in figure 2 scenario A. The alternative being scenario B where all DE variation is explainable by network complexity. Thus we believe optimising networks for an explicit DE metric will result in faster generalising networks than when optimising for low Complexity.

5.3 Success Factors

We will judge the success of the project based on the following criteria:

1. Successfully implementing CDN, LCDN, NCDN, DACDN - making full use of code modularity to exploit similarities and to allow easy implementation of LN-DACDN.
2. Finishing our computationally expensive experiments in time to draw conclusions.
3. Offering some measurable improvements to the CDN algorithm. Including adding in DA evolution.
4. Clarifying the relationship between DE and network complexity.

6 ETHICAL, PROFESSIONAL AND LEGAL ISSUE

6.0.1 Ethical consideration

For the purposes of our research, no experimentation involving human beings will be implemented. As such, no prior consent or ethical considerations are necessary.

6.0.2 Legal issues and intellectual property rights

All research and experimentation will be conducted in compliance with the third-party use specifications of the software libraries. All code written will be released under the Creative Commons licence.

7 PROJECT PLAN

7.1 Risks

Risks and risk mitigation strategies are defined in the risk matrix seen in Appendix A

7.2 Timeline

The project timeline is represented by a Gantt Chart seen in Appendix B

7.3 Resources

In order to competently conduct our research and experimentation there is a number of software and hardware resources required. These resources are listed as follows:

7.3.1 Software resources:

1. Pytorch

7.3.2 Hardware resources:

1. Personal Computers
2. UCT cluster
3. Google Collab [21]
4. Amazon Web services

7.3.3 Data resources:

1. Training data sets:
    a. ImageNet
    b. CIFAAR
    c. MNST
    d. LUNA16

7.4 Deliverables

See deliverables table in appendix C.

7.5 Milestones

The milestones listed below constitute the major steps that are required for the successful completion of the project. They are in line with what is seen in the Gantt chart in appendix B

1. **Completion of project proposal and presentation:** Ensure the project proposal has been written to the best of our ability and recieve any constructive feedback from our supervisor.
2. **Implementation of CDN architecture:** Complete the initial CDN architecture and ensure that it is working satisfactorily
3. **Completing all extensions to CDN:**
    3.1. Successful addition of a secondary evolutionary process which selects the best combination of DAs from a given pool.
    3.2. Successful addition of lamarckian weight inheritance.
    3.3. Successful replacement of CDN's non-dominant sorting with NSGA-II as a pareto front exploration method.
4. **All Testing Completed:** Ensure that our architecture and algorithms are producing effective solutions. Additionally, it allows for the possible modification of algorithms to better optimise or increase their efficiency.
5. **Complete all experimentation:** The Training period is anticipated to take a considerable amount of time. As such, it is imperative that be begin the training process as early as possible.
6. **Findings evaluated and conclusions completed:** Determine if our hypotheses were correct or incorrect. Viewing our experiment's results and forming conclusions based on them.
7. **Report and code submission:** Ensure the final report and code is written to the best of our ability before finally submitting it.

7.6 Work Allocation

The implementation of CDN and all its extensions will be collaborated on as a team. Experimentation for each group member's hypothesis will be done individually. The final report will be worked on together and include each individual group member's findings.

REFERENCES

[1] 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.

[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] Stanley, K.O. and Miikkulainen, R., 2002. Evolving neural networks through augmenting topologies. Evolutionary computation, 10(2), pp.99-127, 1-12 22-24

[4] Elsken, T., Metzen, J.H. and Hutter, F., 2018. Efficient Multi-objective Neural Architecture Search via Lamarckian Evolution. arXiv preprint arXiv:1804.09081, 1-9

[5] 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.

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

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

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

[9] 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

[10] 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.

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

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

[13] 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.

[14] 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

[15] 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)

[16] 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.

[17] Grefenstette, J.J., 1995. *Lamarckian learning in multi-agent environments*. NAVY CENTER FOR APPLIED RESEARCH IN ARTIFICIAL INTELLIGENCE WASHINGTON DC.

[18] Deb, K., Agrawal, S., Pratap, A. and Meyarivan, T., 2000, September. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In International conference on parallel problem solving from nature (pp. 849-858). Springer, Berlin, Heidelberg.

[19] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105), 1-3

[20] Pan, S.J. and Yang, Q., 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, *22*(10), pp.1345-1359.

[21] Bonner, A. (2019). Getting Started With Google Colab. [online] Towards Data Science. Available at: https://towardsdatascience.com/getting-started-with-google-colab-f2fff97f594c [Accessed 22 May 2019].

[22] Luna16.grand-challenge.org. (2019). LUNA16 - Home. [online] Available at: https://luna16.grand-challenge.org/ [Accessed 22 May 2019].

[23] Abbass, H.A., Sarker, R. and Newton, C., 2001. PDE: a Pareto-frontier differential evolution approach for multi-objective optimization problems. In Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546) (Vol. 2, pp. 971-978). IEEE.

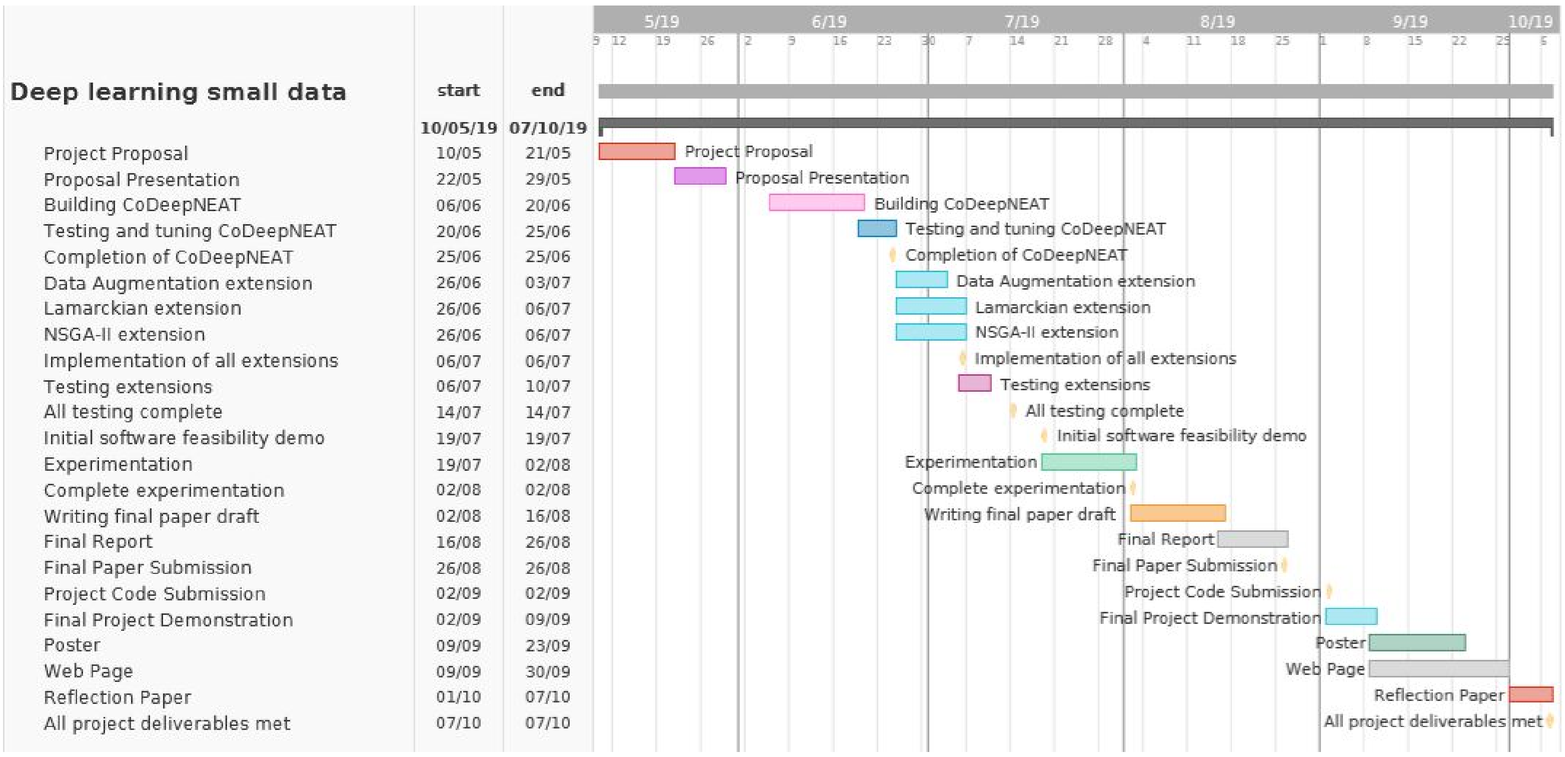
[24] http://www.insticc.org/node/TechnicalProgram/icpram/presentationDetails/74563 - DE

APPENDIX

A) Risk Matrix

Risk	Probability	Impact	Mitigation	Management
Difficulty implementing CoDeepNEAT	Medium	Critical	Begin development early. Stick to the time allocated in the created Gantt chart to help with this.	Consult with the project supervisor. Possibly reduce the scope of the project
Difficulty extending CoDeepNEAT	Medium	Medium	Start development on this as early as possible. Build CoDeepNEAT as modularly and extendably as possible.	Reduce scope by removing one of the extensions from the project
Incorrect training setup	High	Medium	Have all team members check that the setup is correct. Run shorter versions of the experiment before running the longer version.	Create logging tools that can be used live during training, so that experiments can be checked while they are running.
Long training time	High	Critical	Use smaller training sets wherever possible.	Leave ample time for experimentation. Use the Gantt chart to help with time management.
Inability to obtain sufficient hardware	Low	Critical	Ask UCT for time on their cluster. Make use of free cloud options e.g Google collab.	Run experiments on our personal computers with reduced number of generations.
Scope creep	High	Very low	Be weary of adding extra functionality and experiments. Adhere to the Gantt chart as closely as possible.	Remove the least impactful and interesting experiments. Feature freeze may be necessary.

B) Gantt Chart



Deep learning small data	start	end
	10/05/19	07/10/19
Project Proposal	10/05	21/05
Proposal Presentation	22/05	29/05
Building CoDeepNEAT	06/06	20/06
Testing and tuning CoDeepNEAT	20/06	25/06
Completion of CoDeepNEAT	25/06	25/06
Data Augmentation extension	26/06	03/07
Lamarckian extension	26/06	06/07
NSGA-II extension	26/06	06/07
Implementation of all extensions	06/07	06/07
Testing extensions	06/07	10/07
All testing complete	14/07	14/07
Initial software feasibility demo	19/07	19/07
Experimentation	19/07	02/08
Complete experimentation	02/08	02/08
Writing final paper draft	02/08	16/08
Final Report	16/08	26/08
Final Paper Submission	26/08	26/08
Project Code Submission	02/09	02/09
Final Project Demonstration	02/09	09/09
Poster	09/09	23/09
Web Page	09/09	30/09
Reflection Paper	01/10	07/10
All project deliverables met	07/10	07/10

C) Deliverables Table

ID	**Deliverable**	**Due Date**
8	Project Proposal	21/05/2019
9	Project Proposal Presentation	29/05/2019
14	Initial Software feasibility Demonstration	19/07/2019
23	Final Complete Draft of paper	16/08/2019
24	Project Paper Final Submissions	26/08/2019
25	Project Code Final Submission	02/09/2019
26	Final Project Demonstration	16/09/2019
27	Poster Due	23/09/2019
28	Web Page	30/09/2019
29	Reflection Paper	07/10/2019