# Plan

Term Paper Plan

Josias Moukpe
Computer Engineering and Sciences
Florida Institute of Technology
Melbourne Florida USA
jmoukpe2016@my.fit.edu

## 1 Introduction

Today, Artificial Neural Networks (ANN) have proven themselves to be very powerful Machine Learning (ML) model, capable of learning and approximating any arbitrary function, given enough learnable parameters.

However, they aren't currently without flaws. Anyone who has developed an ANN to fit a particular application dataset would have undoubtedly have to toil with the frustration of searching and tuning the numerous hyperparameters of the neural network model. This is because ANN require the adequate combination of hyperparameters values to perform its best, and those hyperparameters are not learned by the model, leaving this arduous task of hyperparameter optimization to the developer. To make things worse, the search space for those hyperparameters is often infinite. The generate list of potential hyperparameters to tune include but is not limited to:

1. Numerical hyperparameters
   - Number of hidden layers
   - Number of hidden units within each hidden layer
   - Learning rate
   - Number of training epochs
   - Momentum rate
   - Batch size (if input batches are being used)
   - Dropout rate
   - Weight Initialization tensor
   - Weight decay rate
   - Number of folds (if K-fold cross validation is applied)

2. Non-numerical hyperparameters
   - Activation function (possibly at each unit)
   - Optimization algorithm
   - Loss function
   - Regularization techniques
   - Network topology (given cell type modules such as fully connected, convolutions, pooling, multi-head attention, etc…)

Furthermore, some hyperparameters can even be adaptively adjusted for performance gains (e.g., adaptive learning rate to reduce training time).

Unlike other machine learning approaches such as Decision Trees, ANN have to many hyperparameters to tune, resulting in a lot of development time spend fine-tuning them and retraining each time, before optimally training for the best model for the chosen application. Deep Learning (DL), the branch of machine learning currently dealing with ANN, currently is more art than science and that unfortunately slows the progress that can be made in not only DL, but also ML, and Computer Science as a whole. It's imperative to solve the problem of hyperparameter initialization and tuning by providing an automated or learned way to reliably set all the hyperparameters to the optimal value for any given task. This problem is also called Auto-ML or meta-machine learning [6].

## 2 Related Work

Our problem being hyperparameter search, we are aiming at providing a method that perform better while being more reliable and efficient at hyperparameter initialization and tuning than the simple manual search. Other attempts at hyperparameter search or optimization are random search, grid search, automated hyperparameter tuning using Bayesian optimization or Genetic Algorithm, and Artificial Neural Network tuning using deep reinforcement learning [3]. Before exploring our approach, we first look at some state-of-the-art approaches.

Manual search is the most basic and the time-consuming approach, taking anywhere from hours to months. After manual search, the next technique that comes to mind is grid search, where every hyperparameter and its values are arranged on a grid and exhaustively searched to find the best combination of hyperparameter values. Unfortunately, Grid search is only applicable at low dimension (2-4 hyperparameters to search) and is impractical at much higher dimension where we usually need hyperparameter tuning for practical reasons. Random search improves on that but at the cost of guaranteed optimality. Random search is application at larger search spaces and provides better results in less iteration compared to Grid search. Random search can also be run in parallel. Word done by Bergstra et al.

demonstrate the superiority of Random Search when compared to Grid search and manual search [4]. However, like Grid Search, Random search doesn't leverage the information gain from previous iteration, each new guess being independent from the previous one. Work done by Zoph et al. explore the application of using ANN or gradient based methods to search optimal architectures (number of hidden layers, hidden units hyperparameters, and the type of layer units). This work leverages the information obtained at every guess and does better at providing the optimal model than just a random search [9]. However, since this work exploit a neural network to optimize the hyperparameters of another neural network, there's no significant reduction in hyperparameters to optimize. Finally, significant strides have been made in applying evolution to hyperparameter optimization. Real et al. successfully applied genetic based search to hyperparameter optimization with the introduction of aging evolution in the search [5]. Their work proved to systematically find more optimal model and find them quicker than Random search or ANN based search.

Another more recent work by Li, Ang, et al. build on it and provide a general framework to population-based training (PBT). Unlike other approaches so far which first optimize the parameters then train the models, population-based training jointly optimized hyperparameters and learnable parameters together [7, 8]. In addition to the optimal models, this approach also provides hyperparameters schedules that more dynamically apply hyperparameters such as learning rate to optimally train models (not just a single hyperparameter value). Evolution based methods have the added advantage of being able to be massively parallelized. However, PBT hasn't been developed to support deep architecture searches, and doesn't cover design choices pertaining to the neural network topology. Our approach then aims to augment PBT with architecture search to provide the optimal AutoML solution.

## 3 Approach

Our approach is building from the works by Real, Esteban, et al., Jaderberg, Max, et al., and Li, Ang, et al. to devise a population-based method that, not just optimized usual numeric hyperparameters such as learning rate, dropout rate, momentum, etc, but also searches for the optimal topology or architecture of the neural network given an initial set of architecture cells or module (such as convolution, pooling, attention, fully connected, etc) to compose from.

Our algorithm will work by providing an efficient way to encode and include the neural network topology in the population-based search, eventually coming up with the optimal model with the optimal topology and hyperparameter schedule in a single joint efficient search.

## 4 Evaluation

To solve AutoML, we are proposing an augmentation of a general framework for population-based training that also search for the optimal network architecture. To evaluate our approach against current methods, we will be attempting to find optimal hyperparameters for optimal ANN models on the following datasets:

- Iris Dataset
- Tennis Dataset
- Identity Dataset

The ANN models would be feedforward neural networks. We aim to demonstrate the superiority of our approach at providing the optimal models with the optimal set of hyperparameters in a quick, reliable, and computationally inexpensive fashion. The performance metrics we will use to evaluate our method include:

- Top Test Accuracy (or the accuracy of the best model produced by the approach, higher being better).
- Experiment time (or the amount of time it took for the approach to run overall, lower being better).
- Model conciseness (or the overall size of the model expressed in term of the number of parameter weights and biases in the model, lower being better).

With these metrics, we aim to demonstrate that our method performs better consistently across the board when compared to other similar approaches such as grid search, random search, and ANN search.

## REFERENCES

[1] Koutsoukas, Alexios, et al. "Deep-Learning: Investigating Deep Neural Networks Hyper-Parameters and Comparison of Performance to Shallow Methods for Modeling Bioactivity Data." Journal of Cheminformatics, vol. 9, no. 1, 2017,
[2] Mitchell, Tom Michael. Machine Learning. McGraw-Hill, 1997.
[3] Ippolito, Pier Paolo. "Hyperparameters Optimization." Medium, Towards Data Science, 26 Sept. 2019,
[4] Bergstra, James, and Yoshua Bengio. "Random Search for Hyper-Parameter Optimization." Journal of Machine Learning Research 13, (2012) 281-305, 2 Dec. 2012.
[5] Real, Esteban, et al. "Regularized Evolution for Image Classifier Architecture Search." Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 4780–4789.,
[6] Gozzoli, Alessio. "Practical Guide to Hyperparameters Optimization for Deep Learning Models." FloydHub Blog, FloydHub Blog, 1 July 2020,
[7] Jaderberg, Max, et al. "Population based training of neural networks." arXiv preprint arXiv:1711.09846 (2017).
[8] Li, Ang, et al. "A Generalized Framework for Population Based Training." Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019,
[9] Zoph, B., and Le, Q. V. 2016. Neural architecture search with reinforcement learning. In ICLR.