ASHESI

ASHESI UNIVERSITY

SCALABLE AUTOMATED MACHINE LEARNING

UNDERGRADUATE THESIS

B.Sc. Computer Science

Gbetondji Jean-Sebastien Dovonon

2020

ASHESI UNIVERSITY

SCALABLE AUTOMATED MACHINE LEARNING

UNDERGRADUATE THESIS

Undergraduate Thesis submitted to the Department of Computer Science, Ashesi University in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Science.

Gbetondji Jean-Sebastien Dovonon

2020

DECLARATION

I hereby declare that this Undergraduate Thesis is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

....

Candidate's Name:

....

Date:

....

I hereby declare that preparation and presentation of this Undergraduate Thesis were supervised in accordance with the guidelines on supervision of Undergraduate Thesis laid down by Ashesi University.

Supervisor's Signature:

....

Supervisor's Name:

....

Date:

....

Acknowledgements

I am grateful to God for seeing me through this, and to my supervisor for the meaningful discussions and the much needed guidance.

Abstract

Automated machine learning holds a lot of promise to revolutionize and democratize the field of artificial intelligence. Neural architecture search is one of the main components of AutoML and is usually very computationally expensive. Autokeras is a framework that proposes a Bayesian optimization approach to neural architecture search in order to make it more efficient [8]. AutoKeras suffers from two major limitations: (i) the lack of support for parallel Bayesian optimization, which limits applicability in distributed settings and (ii) a slow start issue which limits the performance when time is limited. Solving these two problems would make Autokeras more flexible, and allow it to scale to the available resources of the user. We address these two problems. First we design and implement two algorithms for parallel bayesian optimization. Then we incorporate a greedy algorithm to tackle the slow start problem. To evaluate the performance of those algorithms, we first evaluate the Autokeras Bayesian searcher and compare the results to the algorithms we have implemented. On a Tesla T4 GPU, running for 12 hours, the Bayesian searcher got to 80.9% for. Our first parallel algorithm, GP-UCB-PE got 81.85% on 4 GPUs for 12 hours. Our second parallel algorithm, GP-BUCB got 81.89% on GPUs for 12 hours. By incorporating the greedy approach, we achieved 86.78% after running for 3 hours.

Table of Contents

List of Figures

Nomenclature

AutoML	Automated Machine Learning
GP-BUCB	Gaussian Process Batch Upper-Confidence Bound
GP-UCB	Gaussian Process Upper Confidence Bound
GP-UCB-PE	Gaussian Process Upper-Confidence Bound Pure Exploration
HPO	Hyperparameter Optimization
NAS	Neural Architecture Search
UCB	Upper-Confidence Bound

Chapter 1: Introduction

Machine learning and data science provide researchers with tools to analyze data, interpret it, and use it to make predictions or replicate human intelligence [10, 3, 11, 9]. Because of the amount of data that is generated by pretty much any computer-based system nowadays, it is quite common to see machine learning being used by users who do not necessarily have a broad machine learning expertise. These users range from researchers from other fields to software engineers trying to include a new cool predictive feature in their new mobile application. In those cases, the user is likely not to have the required experience to build machine learning models. Automating machine learning, to some extent is highly desirable in such contexts. A framework to automate machine learning would, however, need to account for the available resources, resources which can vary a lot depending on the user. Time and GPU are the resources used here to train machine learning models. The amount of resources is not fixed and is not the same for every use. In this diversity of working environments, we are trying to come up with a framework to automate machine learning in a way that performance scales with the available resources. We will focus on deep learning for image classification.

1.1 Motivation

Automated machine learning (AutoML) consists of a set of techniques and approaches to automating parts of the machine learning process. The machine learning process (defining and training architectures that perform) is tedious and is more of an art than a science (involves a lot of intuition, guesses, trial and error), AutoML seeks to automate this process and make it data-driven [7].

The usual AutoML framework has two main concerns when it comes to deep learning: neural architecture search (NAS) and hyperparameter optimization (HPO).

HPO is not specific to deep learning but to any machine learning method that requires hyperparameters to be set. hyperparameters are not learned during training but rather define the training procedure, therefore setting the right hyperparameters has a considerable influence on the performance of the model.

Most machine learning frameworks include default hyperparameter settings, however, in general, hyperparameters tuned for the specific context outperform the default configuration, sometimes by a large margin. HPO can help discover configurations that are optimal for the given context.

NAS is the process of searching for deep learning architectures for a specific task. The search is done over a search space, then evaluated using the relevant metric, the next architecture to evaluate is the chosen following the search policy [5].

For both HPO and NAS, AutoML has been able to outperform experts, given enough resources. The main limitation of AutoML is that it requires huge computation power and time, mostly because of the cost of NAS. Recent methods have made it possible to perform NAS in resource-constrained environments. Those approaches do not always perform as well as the state of the art but can get close [8].

When it comes to training environments, there are several possible configurations, and resource-constrained environments include a vast range of possible constraints, both in terms of computation power (CPU and GPU) and time. It is beneficial to have a framework that can make full use of the available resources available no matter where the configuration falls within the possible range of possible training environments.

1.2 Objective

We are trying to make AutoML scale better with the available resources. The goal is to have a platform that can perform in low resource environments but can improve performances when additional resources are available. In the end, the users should be able to specify the resources available. The resources to be specified would be:

1. GPUs specs
2. Time

We will focus on the NAS aspect of AutoML for image classification.

Chapter 2: Related Work

Neural architecture search (NAS) is one of the more resource-heavy aspect of the AutoML process. It consists of searching for an architecture that maximizes a given objective (typically a desired test accuracy).
Neural architecture search typically has three main components [5]:

- the search space that characterizes the possible networks that can be searched for and evaluated
- the optimization strategy that defines how to search the search space for the best network
- the evaluation strategy that estimates the performance of a network (typically training the network from scratch)

2.1 State of the art neural architecture search

The best NAS methods usually work based on reinforcement learning, evolutionary techniques, random search [6].
Several methods achieve competitive performance and can often challenge experts in a competitive setting. However, their cost, (in GPU days) is usually huge. NASnet , which uses reinforcement learning, takes 2000 GPU days to run CIFAR 10[15].
Several techniques exist to reduce the cost of AutoML [5]:

- low fidelity estimates that work by evaluating architectures with a lower fidelity (a subset of the data, few epochs)[12].
- learning curve extrapolation which can help estimate the learning outcome after just a few epochs [1].
- weight inheritance methods base weights on previously trained architectures. Applied in systems like AutoKeras [8].
- training a one shot model.

2.2 AutoKeras

AutoKeras is a system that uses network morphing operations guided by Bayesian optimization in order to do NAS efficiently [8]. It uses a tree-like search space with a base network. The base network is trained first and serves as root to the tree. The network morphism operations that can be applied to that network constitutes branches that link to other network nodes in the tree-like search space.

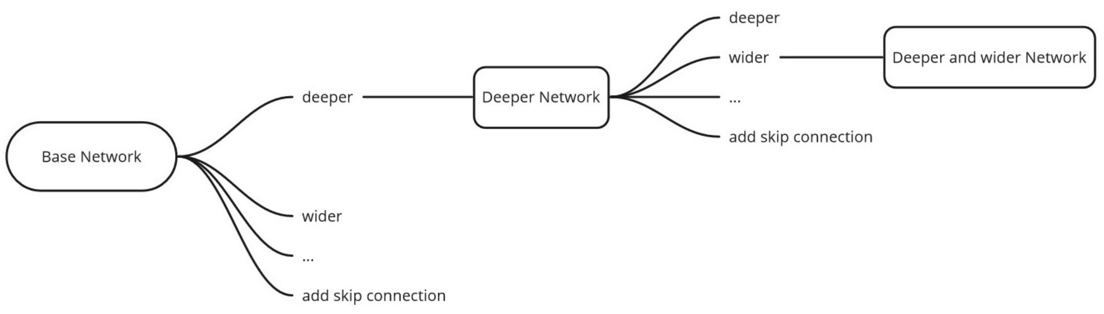

Figure 2.1: Autokeras tree like search space

The optimization strategy used by Autokeras uses Bayesian optimization with the upper confidence bound (UCB) acquisition function. It uses the A* search algorithm to explore the tree-like search space and find a network with a high acquisition value. This Bayesian optimization procedure is based on a Gaussian process and the UCB acquisition function, we will refer to it as GP-UCB (Gaussian Process Upper Confidence Bound).

Chapter 3: Methodology

Our main goal is to develop an AutoML method that scales in terms of time and computation power. We do this by proposing two additions to the Autokeras algorithm:

- parallel evaluation: making it possible to search for multiple networks and evaluate them simultaneously on several GPUs
- faster search using a greedy approach

3.1 Parallel Evaluation

Our proposed method is based on the efficient Bayesian optimization method proposed in AutoKeras [8]. They adapted BO to work on a tree-based search space where every network is a node, and the edges are morphism operations that connect a parent network and a child network. Their acquisition function is the upper confidence bound (UCB) and is optimized using the $A*$ search algorithm with simulated annealing. The search finds the node with that maximizes the acquisition function. Simulated annealing is used to avoid always choosing the best architecture. The reason for avoiding that is one weakness inherent to network morphism: it can only increase the size of a network [14].

3.1.1 UCB for GPU parallelism

In order to achieve performance scaling with GPU number, it is possible to train as many networks as there are GPUs, all in parallel. Doing so is non-trivial because Bayesian Optimization is a sequential algorithm. So at every step t, after evaluating a candidate network f_t, generating f_{t+1} requires the feedback y_t.

First, systematically using the same algorithm to optimize the same acquisition function will result in a higher chance of generating the same networks to evaluate. One solution is to have a joint acquisition function for a defined number of evaluations. We can then plan

how many concurrent evaluations should be done and choose a number q of points to evaluate using the q expected improvement in a way that we maximize the information gain from the q evaluations [13]. This is also non-trivial and would add the constraint of knowing the number q of parallel evaluations before optimizing the acquisition function.
The following is the UCB acquisition function that is used in AutoKeras:

$$\alpha(f) = \mu(y_f) + \beta\dot{\sigma}(y_f) \tag{3.1}$$

Work has been done on UCB specifically to make the process parallel[2, 4].

3.1.2 GP-UCB-PE

The first method is the Gaussian process with upper confidence bound and pure exploration (GP-UCB-PE) [2]. This method chooses the first element by optimizing UCB then chooses the remaining q-1 points using pure exploration. An advantage of using UCB is that it provides us with a clear balance between exploration and exploitation, regulated by the balancing factor β. We can then focus on exploration.
The next step consists in using a greedy strategy to choose the next q-1. The greedy step here is a pure exploration step because we maximize the variance instead of the UCB acquisition function. At each step of the greedy optimization process, we use the updated variance $\sigma(f)^k$ updated with the previously generated networks. This is possible because $\sigma(f)^k$ does not depend on the feedback y_f obtained after evaluating f. We can then use the same search algorithm proposed in AutoKeras to optimize the variance [8].
Using pure exploration is one exploration strategy, and it can be used an approximation to maximizing the information gain from choosing the next q-1 points, as shown in [2]. Other exploration strategies could also be used, like for random selection, or choosing the q top selections using UCB.

3.1.3 GP-BUCB

Another algorithm can be used to tackle the issue of parallel Bayesian optimization, the Gaussian process batch upper confidence bound (GP-BUCB) [4].
GP-BUCB works by sequentially choosing q architectures using UCB. After using UCB to select a point to evaluate, the mean predicted value is used to update the GP, then the next point to evaluate is selected.

Even though it does not pose a problem for the first point, there is a problem of delayed feedback for the next q-1 points to select. To solve that, it *hallucinates* the feedback by using the mean predicted value as feedback, then proceeds.
In the case of AutoKeras we would use the same search algorithm to optimize the acquisition function on the hallucinated GP, then use the batch of delayed feedback to update the original GP.

3.2 Greedy Search

One of the issues faced when using network morphism is that the child network is always at least as big as the father network. To avoid networks growing too big, Autokeras searches over the entire tree search space (including the root) and not only the leaves. This procedure makes it possible to produce a new network to evaluate that is not always as big as the most recently evaluated networks. It can become an inconvenience when the time available is limited, or the hardware resources cannot train networks fast enough. Neural networks' performances are usually related to their size, with bigger networks performing better than smaller ones. With the usual Autokeras optimization strategy the networks are shallow during the first iterations causing a slow start effect.
In order to avoid that, we propose to use a greedy searcher that restricts the search space to the children of the best leaf. This restricts the search space to a minimal number of networks. The network with the best acquisition value is then evaluated.

Chapter 4: Experiments and Results

In this chapter, we show how our approaches — GP-BUCB, GP-UCB-PE, Greedy approach — provide significant improvement over the baseline search algorithm provided with AutoKeras.

4.1 Implementation

We provide three python classes for neural architecture search. The first class implements the GP-UCB-PE algorithm, and the second the GP-BUCB algorithm. The last class implements a Bayesian searcher that uses a search space restricted by incorporating our greedy approach. For that searcher, we reduced the maximum number of epochs to 30 to avoid over-fitting. We measure the improvements provided against the original Bayesian searcher provided by Autokeras. In order to evaluate the algorithms, we use the CIFAR 10 dataset.

4.2 Experimental setup and baseline

All experiments are ran using PyTorch and use a Tesla T4 GPU. For the methods that use parallel evaluation, we use four Tesla T4 GPUs. We use a batch size of 64. The models are trained on 80% of the training set and validated on the remaining 20%. The performances are reported using the test set.

As a baseline, we run the Autokeras Bayesian searcher for 12 hours. The best accuracy reached is 80.9%. During this run, 24 models were evaluated.

The GP-BUCB algorithm reached 81.89%, and evaluated 93 models in 23 search iterations. The GP-UCB-PE reached 81.85%, and evaluated 129 models in 33 search iterations. The GP-UCB-PE has a better performance early on, but the GP-BUCB has a smoother learning curve. Both algorithms improve the baseline accuracy by 1 percent for a similar training time.

The greedy approach achieved 86.78% with 38 models evaluated in 3 hours.

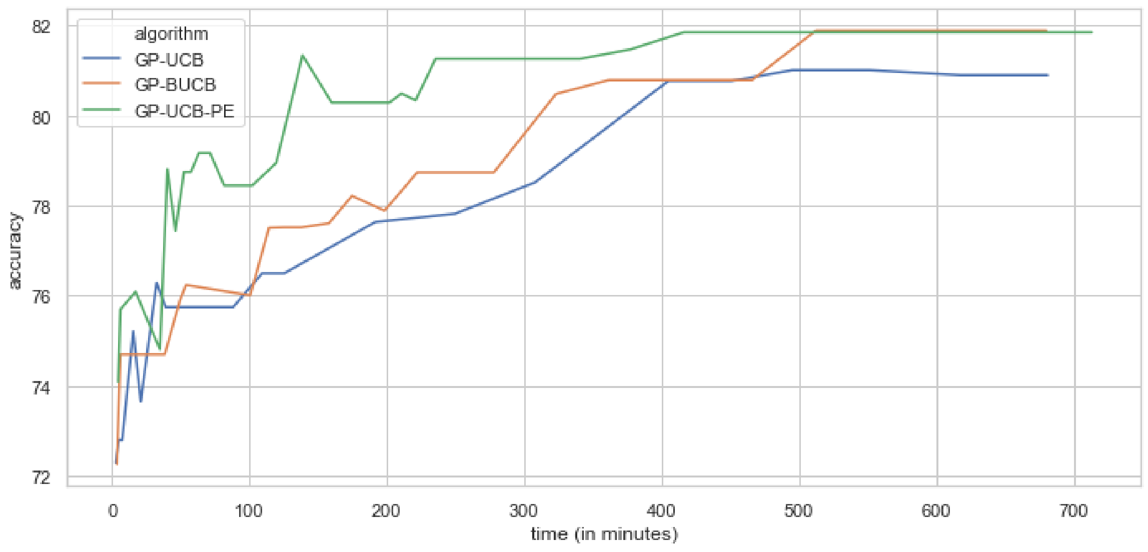

Figure 4.2: Our two parallel methods achieve 1 percent better test accuracy compared to the Autokeras baseline.

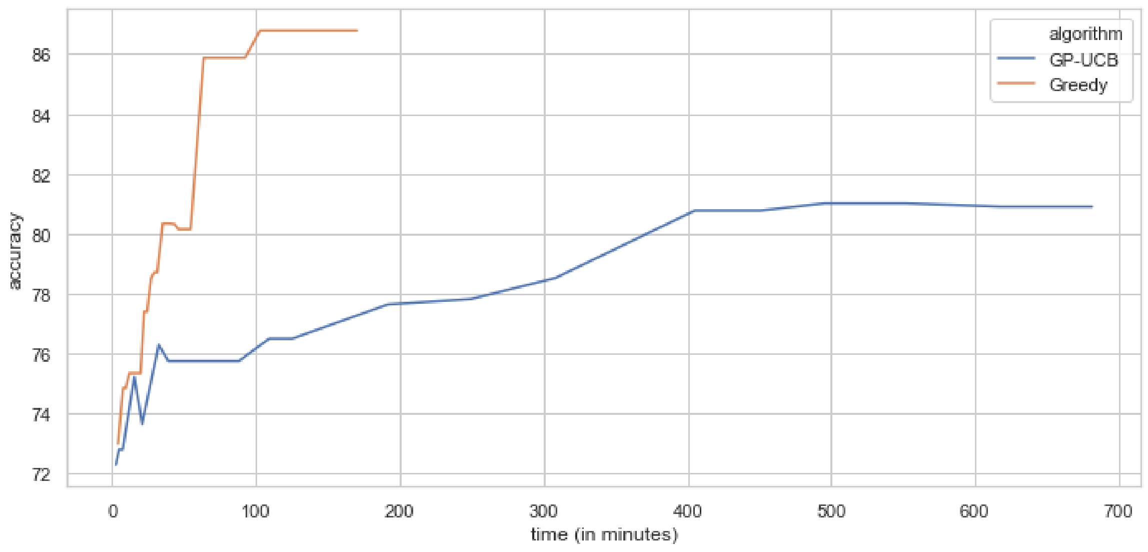

Figure 4.3: Our greedy approach achieves 6 percent better test accuracy compared to the Autokeras baseline with a 4 times shorter training time.

Chapter 5: Conclusion and Future Work

5.1 Summary

The goal of this research was to develop AutoML techniques that are more flexible in the sense that they scale with the available resources (GPU and time). To achieve that, we based our work on Autokeras, a tool for resource-efficient AutoML. We designed and implemented three approaches: GP-BUCB, GP-UCB-PE, and a greedy approach that achieved a 1 percent, 1 percent and 6 percent improvement in test accuracy on the CIFAR 10 dataset. Our approaches make it possible to have performance scale better with the number of GPUs and the time available, making it possible for users to make a full use of their training resources.

5.2 Future Work

We plan to increase the number of searcher classes available, using other Bayesian optimization algorithms, and eventually release the code as a python package. The approaches we designed have not yet been evaluated on other datasets like Fashion MNIST or CIFAR 100. These tests would help verify the robustness of the suggested methods. A theoretical analysis of the methods suggested also needs to be done in order to assess their time and space complexity. Possible improvements include adding more options for building blocks, distance functions and search spaces. The type of problems that can be handled is also an area to work on in order to make it possible for the system to work for text data, audio data or even image generation.

References

[1] BAKER, B., GUPTA, O., RASKAR, R., AND NAIK, N. Accelerating Neural Architecture Search using Performance Prediction. *arXiv e-prints* (May 2017), arXiv:1705.10823.

[2] CONTAL, E., BUFFONI, D., ROBICQUET, A., AND VAYATIS, N. Parallel Gaussian Process Optimization with Upper Confidence Bound and Pure Exploration. *arXiv e-prints* (Apr 2013), arXiv:1304.5350.

[3] DANNENHAUER, D., FLOYD, M. W., DECKER, J., AND AHA, D. W. Dungeon Crawl Stone Soup as an Evaluation Domain for Artificial Intelligence. *arXiv e-prints* (Feb. 2019), arXiv:1902.01769.

[4] DESAUTELS, T., KRAUSE, A., AND BURDICK, J. W. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *Journal of Machine Learning Research 15* (2014), 4053–4103.

[5] ELSKEN, T., HENDRIK METZEN, J., AND HUTTER, F. Neural Architecture Search: A Survey. *arXiv e-prints* (Aug 2018), arXiv:1808.05377.

[6] HE, X., ZHAO, K., AND CHU, X. AutoML: A Survey of the State-of-the-Art. *arXiv e-prints* (Aug 2019), arXiv:1908.00709.

[7] HUTTER, F., KOTTHOF, L., AND VANSCHOREN, J. *Automated Machine Learning: Methods, Systems, Challenges*. Springer, 2019.

[8] JIN, H., SONG, Q., AND HU, X. Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2019), ACM, pp. 1946–1956.

[9] LEE, J., YOON, W., KIM, S., KIM, D., KIM, S., SO, C. H., AND KANG, J. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *arXiv e-prints* (Jan. 2019), arXiv:1901.08746.

[10] NAKANO, R. Neural Painters: A learned differentiable constraint for generating brush-stroke paintings. *arXiv e-prints* (Apr. 2019), arXiv:1904.08410.

[11] RADFORD, A., WU, J., CHILD, R., LUAN, D., AMODEI, D., AND SUTSKEVER, I. Language models are unsupervised multitask learners.

[12] REAL, E., AGGARWAL, A., HUANG, Y., AND LE, Q. V. Regularized Evolution for Image Classifier Architecture Search. *arXiv e-prints* (Feb 2018), arXiv:1802.01548.

[13] WANG, J., CLARK, S. C., LIU, E., AND FRAZIER, P. I. Parallel Bayesian Global Optimization of Expensive Functions. *arXiv e-prints* (Feb 2016), arXiv:1602.05149.

[14] WEI, T., WANG, C., RUI, Y., AND CHEN, C. W. Network Morphism. *arXiv e-prints* (Mar 2016), arXiv:1603.01670.

[15] ZOPH, B., VASUDEVAN, V., SHLENS, J., AND LE, Q. V. Learning Transferable Architectures for Scalable Image Recognition. *arXiv e-prints* (Jul 2017), arXiv:1707.07012.