AUTO-ML BASED PIPELINE FOR DEEP LEARNING TASKS

Astha Gupta, Gabe Cemaj

(ag7982, gc2728)

Abstract

Over the last decade, deep learning has gained traction in the field of Computer Science. Neural networks-based models are being leveraged to perform many tasks that once required human intervention. Thus, it is no surprise that the number of new architectures, the optimisers and hyperparameters being explored is ever-increasing. In such a scenario, it can often become increasingly difficult to find the right combination for the task at hand. The problem of this domain is no longer the unavailability of robust models, but rather the use of the best of the available ones. One solution to the problem of this time-consuming task has been answered by the discovery of AutoML. Through this project, we present our own implementation of an end-to-end AutoML pipeline. Using the underlying concepts of HML Opt and Differentiable Architecture Search (DARTS), we build a system that takes in raw data as input and uses a probabilistic machine learning algorithm to determine the best deep learning model parameters for that task.

Keywords: AutoML, Neural Architecture Search , Deep Learning

Introduction

In recent years, the idea of AutoML has become more prevalent and gained significant popularity. Described as 'AI that creates AI', AutoML systems can be used to build entire pipelines for many deep learning tasks. The concept of automating the process of picking and designing a ML pipeline is very attractive as this is often a very time consuming step. With the growth of ML and DL more and more decisions must be made to pick the correct combination of parameters to build a ML pipeline.

The space from which to pick extends well beyond traditional hyperparameters. Not only must we now pick an architecture, depth, width, activation function, and optimizations, but also, there is a large array of options to pick from with regards to the preprocessing and normalization steps. The performance of any model is greatly impacted by the values chosen for these hyper-parameters. Finding the perfect fit is a time-consuming and a resource-intensive task, and conventionally requires a trial-and-error approach. This large search space would benefit from an automated way of exploring it to find a combination that results in the best performance.

Related Works

Two main studies formed the basis of our implementation. For the generation of the pre-processing steps and hyperparameter values, we used a Contextual Formal Grammar (CFG) of nodes consisting of the possible candidates. A probabilistic

algorithm was then used to select the resulting steps. This is based on [1], which is an AutoML framework based on probabilistic grammatical evolution. Researchers can define a problem-specific grammar that encodes their domain knowledge about a problem. This way, researchers define which pipelines are interesting to explore, while HML-Opt evaluates and searches for the best pipeline. The implementation of this algorithm is available as an open-source repository called AutoGOAL [2]. The advantage of this method is that it gives us the best possible pre-processing steps- something which can be often ignored in comparison to the neural network search.

For the neural network search, we replied on Differentiable Architecture Search (or DARTS) [3]. DARTS assumes the space of candidate architectures is continuous, not discrete, and this allows it to use gradient-based approaches, which are vastly more efficient than the inefficient black-box search used by most neural architecture search algorithms. The main benefit of choosing DARTS over other options was its efficiency. To learn a network for Cifar-10, DARTS takes just 4 GPU days, compared to 1800 GPU days for NASNet and 3150 GPU days for AmoebaNet (all learned to the same accuracy). In our implementation, we make use of Progressive DARTS (or PDARTS [4]), which is a more enhanced version of DARTS, and deals with the issue of high memory consumption in DARTS, by employing search space optimisation.

Objective

This project aims to string together many of the aspects of AutoML to build a grand unified solution to generate end to end ML pipelines. The idea is to combine neural architecture search, along with hyperparameter tuning and grander exploration of preprocessing combinations to automatically generate a full end to end pipeline for performing deep learning tasks.

Challenges

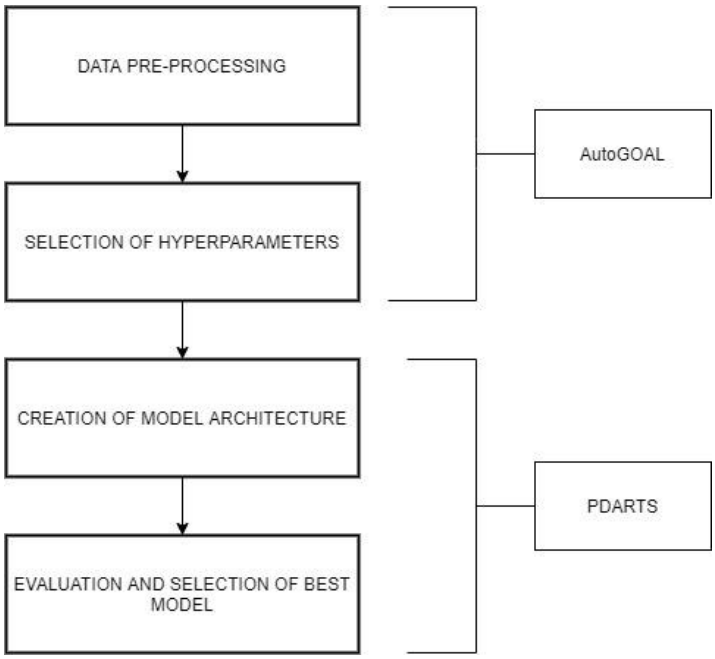
There are two categories of challenges that need to be addressed.

The first set of challenges were caused due to the computation and time constraints. In an ideal world, with unlimited GPU and memory resources, we could focu on just finding the model with the best performance in terms of accuracy. There would be no limit to the number of candidate architectures, hyperparameters or pre-processing steps we could work with. The search space would be enormous and we would have the time to find the best possible architecture. However, in a real world scenario, this is simply not feasible. Thus, there needs to be a limit placed on the number of candidates being chosen to be part of the search space. The resulting system should not only give us a result with good accuracy, but it should also be able to generate that result in a reasonable amount of time.

The second category of challenges are those that are inherent in most deep learning tasks. The black-box nature of the Neural Architecture Search algorithm was one such problem. A small change in the initial parameters could result in a big change in the results. Another issue was that of human bias. In selecting the candidates, it becomes highly likely for researchers to add ones that they are familiar with, instead of choosing newer unexplored possibilities.

Architecture Diagram

The following architecture diagram is the basis of our work. The first two steps of any autoML task are data pre-processing and the selection of hyperparameters. This was done using HML-Opt (or AutoGOAL). The last two steps, namely creation of the neural network and the selection of the best possible model based on some evaluation, was performed using the PDARTS algorithm.



Implementation

The main hurdle of our implementation was combining the two sub-architectures into a single one.

The hyperparameters and pre-processing steps form the non-differentiable features. Usually, a grid-based search is used in such a scenario. The search space is explored using a CFG with associated probabilities. After each experiment, the probability of picking an option is updated based on a pre-defined metric or score. We then employ this CFG hyperparameter strategy to pick any hyperparameter that cannot be smartly picked. This includes hyperparameters for the PDARTS system as well as the data cleaning and augmentation ones.

PDARTS is utilised to search for the Neural Architecture,which is differentiable by nature. The first step involves a trial with different candidate preprocessing pipelines. All these runs are then compared and the one with the highest fitness score is chosen. The combination of these two processes, results in a fully automatable system that generates end-2-end ML pipelines. The fitness score ideally captures two parameters- accuracy and speed. We first estimate an ideal time and an ideal accuracy. The following formula is used to calculate the score of a model.

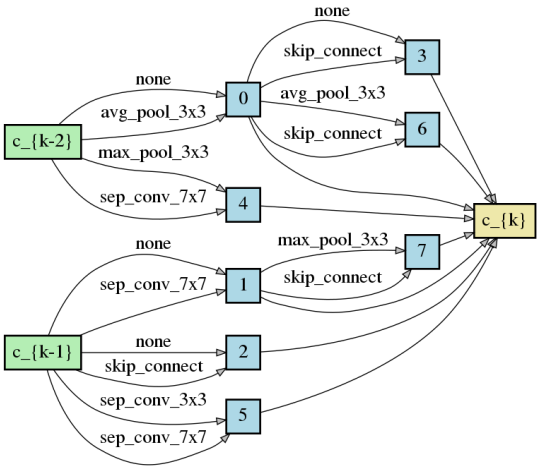
```
fitness(actual_acc, actual_time) = (actual_acc - est_acc) + (est_time - actual_time) * alpha
```

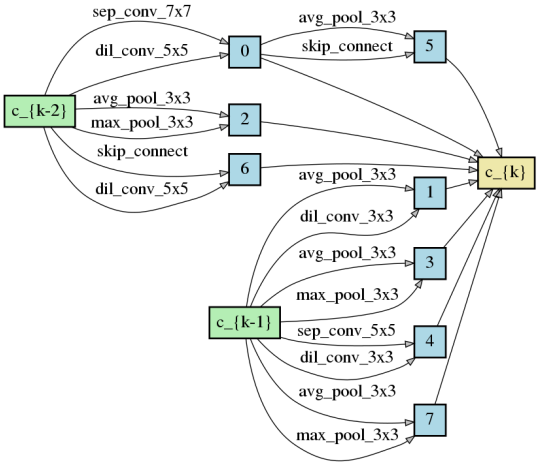
In applications where speed is important, the chosen value of alpha should be higher. In applications where accuracy is important the value of alpha should be smaller.

As for the implementation details, PyTorch was mainly used for the code, using Python 3.9. A V100 GPU was used for the system run. The implementation for AutoGOAL and PDARTS was leveraged via their open-source repositories.

Results and Conclusion

We ran our pipeline for the MNIST dataset and the following results were obtained. The pre-processing steps chosen for the dataset were Gaussian Blur (kernel size 41), jittering, random horizontal flips, cutout, cropping and normalisation. The generated neural network architecture consisted of four stacks of the following Normal and Reduce cells, respectively.





Our model was able to achieve a high accuracy (98 percent), in just over 90 minutes). This is comparable to the performances of other autoML based systems. In the future, we would experiment with larger datasets to test out our system's performance in comparison to various state-of-the-art systems.

.

References

1. Suilan Estevez-Velarde et. al, “General-purpose hierarchical optimisation of machine learning pipelines with grammatical evolution”,Information Sciences,Volume 543,2021
2. GitHub for AutoGOAL: https://autogoal.github.io/#installation
3. Computational Linguistics. Hanxiao Liu and Karen Simonyan and Yiming Yang, “DARTS: Differentiable Architecture Search”, arXiv preprint arXiv:1806.09055, 2018
4. Xin Chen et. al, “Progressive Differentiable Architecture Search: Bridging the Depth Gap between Search and Evaluation”, arXiv:1904.12760v1