BENCHMARKING AUTOKERAS PERFORMANCE

Jan Kruszewski **Damian Skowroński** **Tomasz Krupiński**

June 8, 2022

ABSTRACT

The purpose of automatic machine learning (AutoML) is to facilitate the creation of a good machine learning model. Over the last few years many AutoML systems have been developed. In order to determine whether or not the systems are effective, benchamarks were created. In this article we take a closer look and present the results of a benchmark performed on the AutoML system called AutoKeras.

Keywords AutoKeras · AutoML · OpenML

1 Introduction

Machine learning has developed significantly in recent years. This has resulted in a demand for specialists in this field. Hence the development of a derivative of machine learning - automatic machine learning (AutoML). Use of AutoML allows a data scientist to reduce the number of repetitive, manual tasks such as data cleaning or feature engineering and concentrate on analysis, explanation, and validation of results or deployment of the process. It is a solution to the problem of time-consuming model building and the shortage of machine learning specialists. AutoML is also a helpful tool for not experienced data scientist as it allows to learn advanced machine learning topics without significant programming skills.

In this article we focus on binary classification of tabular data using an AutoML system called AutoKeras[1]. The article[1] written by the authors of AutoKeras helped us understand how the system works. We referred to it mainly in the second section. To explore the effectiveness of AutoKeras we used the binary classification datasets, which were previously used in an open source benchmark[2]. The article[2] describing the benchmark was also helpful in our evaluation. Analysis of the results and comparison to other AutoML libraries helped us determine whether the use of AutoKeras brings satisfactory effects.

2 Descripion of the framework

AutoKeras is a Python library, developed by a group of people associated with Texas AM University. It is based on Keras, an open source deep learning framework. The main idea of the developers was to create a system that would give a chance of using machine learning to people with little experience in this field. The package has an extensive and clear documentation.

[1] https://autokeras.com/

[2] https://new.openml.org/search?type=benchmark&sort=tasks_included&study_type=task&id=218

2.1 Division of AutoKearas into modules

The AutoKeras framework is divided into 4 main modules.

- The Searcher module, is responsible for searching the neural architecture. It uses Bayesian optimisation and a Gaussian process. The search algorithms use the CPU.
- Neural networks are trained in the Trainer module.
- Graph is a module that processes computational graphs that Searcher controls to morph the network.
- The last module is Storage. It is responsible for saving the final models. Due to their size they are not stored in RAM.

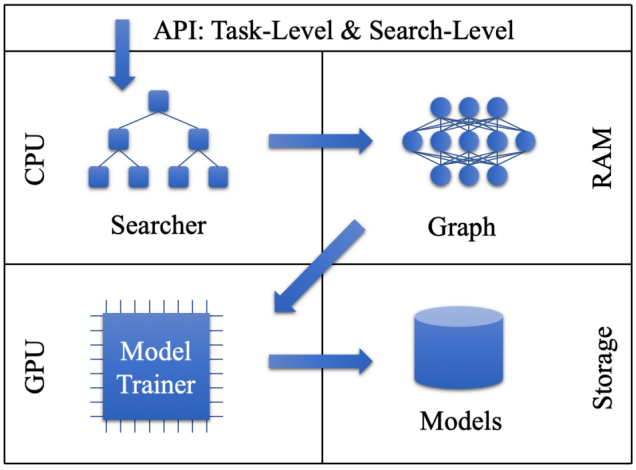

Figure 1: AutoKeras modules - source [3]

2.2 What distinguishes AutoKeras?

As stated before, the authors wanted to make AutoKeras easily available. Because of that it can be used locally, there is no need to use cloud services. The authors ensure data security and privacy. It is also worth mentioning that AutoKeras focuses on deep learning problems as opposed to frameworks like SMAC [3], TPOT [4], Auto-WEKA [5] or Auto-Sklearn [6] which focus on shallow models.

2.3 API

AutoKeras works only on Unix The inteface of AutoKeras has been created in a similar way to Sklearn [7]. It can be built in 3 lines of code using the constructor, `fit` and `predict` methods.It is worth noting that it has two levels *task-level* for users with less knowledge of the system and *search-level* for advanced users who want to control the preprocessing and neural network architecture themselves. In general, AutoKeras offers us the following models: `ImageClassifier`, `ImageRegressor`, `TextClassifier`, `TextRegressor`, `StructuredDataClassifier`, `StructuredDataRegressor` and `AutoModel`. The framework has also many user-friendly features, but we have not managed to find an implementation of cross-validation, so we believe it is not supported.

2.4 Preprocessing

Package does the necessary preprocessing for us before training. Hence, our function does not have any additional code snippets devoted to this activity. AutoKeras performs for us the basic vectorisation, data cleaning and normalisation, which is an important step for neural networks.

[3] https://www.automl.org/automated-algorithm-design/algorithm-configuration/smac/
[4] http://epistasislab.github.io/tpot/
[5] https://www.automl.org/automl/autoweka/
[6] https://automl.github.io/auto-sklearn/master/
[7] https://scikit-learn.org/stable/

2.5 Use of resources

AutoKeras is designed to take full advantage of the CPU, GPU and RAM, where only the currently important information is stored. The rest of the data is stored in other places such as the hard drive.

3 Results of the evaluation

In this section we focus on evaluation of AutoKeras performance.

3.1 Example model

The example model, whose diagram is shown in the Figure 2, was selected after twenty searches for the best model. The model starts with basic layers such as the input layer, the encoding layer and the normalisation layer. Next are mainly dense layers, with a linear function and others with a ReLU function. As it was a classification task, it ends with a classifier layer. Thus, it is not a very complex architecture. Nevertheless, this architecture fulfils its purpose

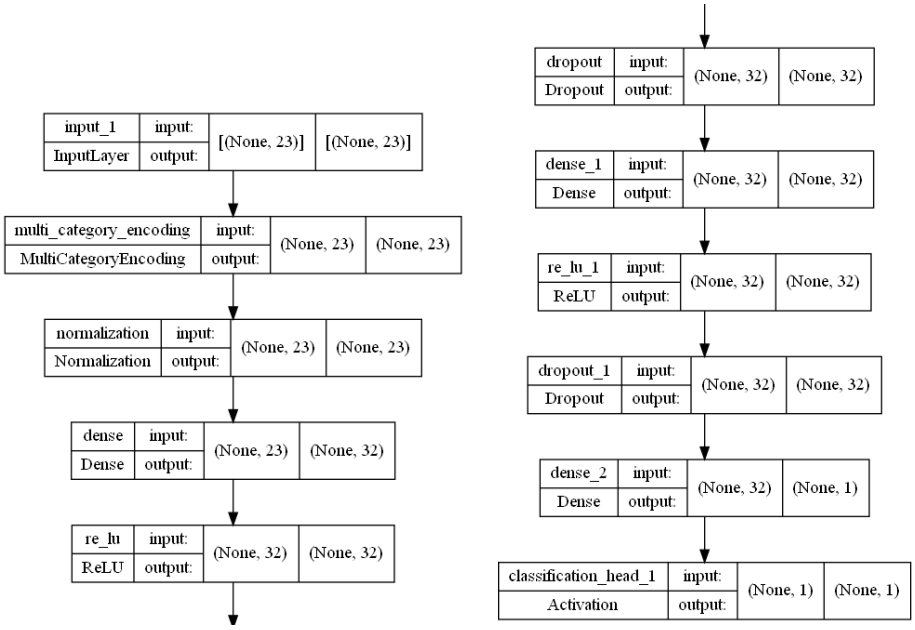

Figure 2: Example of AutoKeras neural network

3.2 Our benchmark

Similarly to the aforementioned Open Source AutoML Benchmark we tried to test AutoKeras on 22 different binary classification datasets, We used it successfully on 17 of them, and encountered problems with 5 of them, namely: *KDDCup09_appetency, guillermo, riccardo, albert,* and *christine*. The issue was either with the kernel constantly dying due to what we believe was not enough RAM, or with the system running indefinitely with no response. For training models we used a 5-fold cross-validation with the default metric - the accuracy score. We measured the elapsed time for the full process of training a model on each dataset, as well as the ROC AUC score of each fold. On each dataset we used 4 different settings of AutoKeras parameters:

1. default constructor, default fit
2. `max_trials = 10`, default fit
3. `max_trials = 10`, `epochs = 100` in fit
4. `max_trials = 20`, default fit

Unfortunately our machine had no GPU so all computation was done using a CPU. For this reason the elapsed time measured is a way of comparing the different settings rather than the actual time a user would have to wait. Nevertheless, what is important, the ROC AUC score should not be affected by it.

3.3 Results of benchmark

The results are shown below.

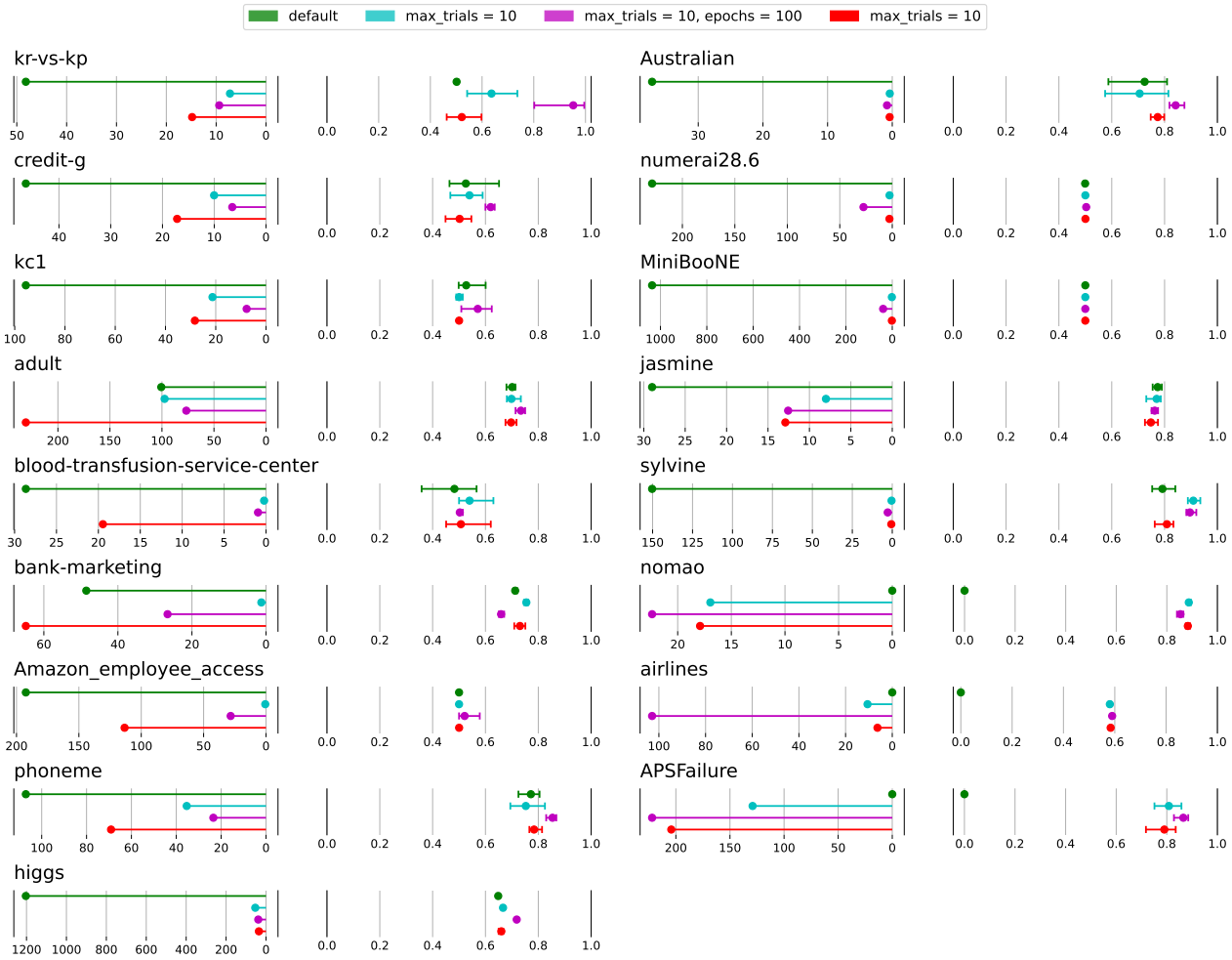

Figure 3: Results of different settings for AutoKeras. For each set there are two plots. The left ones show times taken for execution in minutes, and the right ones show average (point), minimal and maximal (whiskers) ROC AUC.

We can draw several conclusions from Figure 3:

- The default (green) setting was by far the most time consuming almost every time. It was in fact so time consuming, that we couldn't get the results from the last three datasets after over a day of execution.
- Even though the default setting usually took much longer than the others, it was rarely better. On the contrary, every time it was among the worst. We believe that it may be due to overfitting.
- The best scores were the most often from the setting with set max_trials and epochs (magenta). It it usually quite fast as well.
- Bigger number in max_trials (red) rarely gives better outcome than smaller number (cyan).
- Usually the gap between minimal and maximal scores on a given dataset and setting is not big.

3.4 Average performance

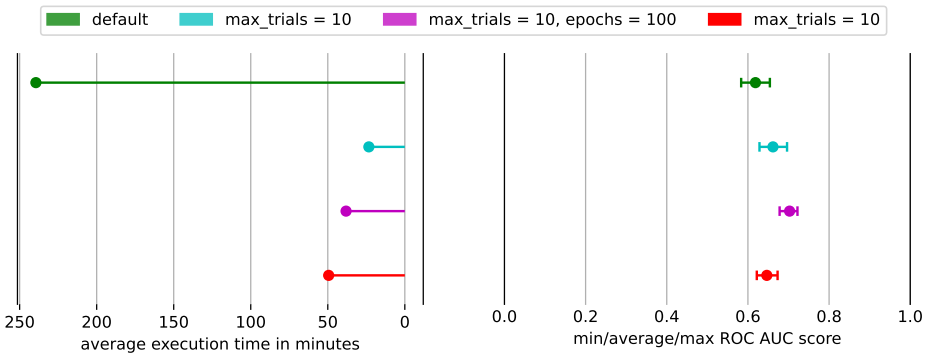

Figure 4: Average performances for each setting over the datasets.

It seems that in our benchmark the setting with `max_trials = 10` and `epochs = 100` came out on top. Not only was it usually the best when it came to ROC AUC, but it also was one of the fastest to execute.

4 Group competition

There were five groups in our case study and each one tried different AutoML framework. Together we made a test on one particular tabular binary classification set in order to compare the quality of the pipelines made to create predictions.

4.1 Our contribution

We used AutoKeras in three different settings:

1. default contractor, default fit - a baseline
2. `max_trials = 10`, `epochs = 100` - in previous section we concluded that it works better than default
3. Customised Model using AutoModel API implemented in AutoKeras - as a quick demo

Figure 5: Graphical representation of our Customised Model

The Customised Model is very simple. We specified:

- `categorical_encoding = True` and `normalize = True` in StructuredDataBlock
- `metrics = 'AUC'` in ClassificationHead, because AUC score was the used for comparison

4.2 Results of the competition

Team	AUC	framework
KTR	0.7912	FLAML
Gakubu	0.7908	AutoGluon
Tojada	0.7864	AutoKeras (setting 2)
Gakubu	0.7852	Own implementation
WTF	0.7835	AutoPytorch
Moja grupa	0.7815	AutoPytorch
Moja grupa	0.7808	Own implementation
Tojada	0.7795	Customised Model (setting 3)
Moja grupa	0.7789	Autosklearn
WTF	0.7768	Own implementation
KTR	0.7617	Own implementation
Tojada	0.7469	AutoKeras (setting 1)

Table 1: Results of each group

As you can see not every implementation created neural networks. One could therefore intuitively conclude that neural networks would have the best results, obviously compensating for the training time compared to, for example, regular algorithms. However, it turned out that AutoKeras with default setting achieved the worst results of all the other frameworks.

The other two settings, however, turned out better, placing in top 3 (setting 2) and somewhere in the middle (our Customised Model). Although the Customised Model wasn't the best out of our three settings, it still was a small success, as it took several times less time to execute, than even the fast setting 2.

5 Conclusion

We presented an accessible description of an autoML framework called AutoKeras along with a comparison to other autoML frameworks, both in terms of main features and benchmarks. In peer-to-peer competition, at first AutoKeras placed last. Later however, we managed to increase our score to third place with the usage of more complex parameters.

6 References

1. Haifeng Jin, Qingquan Song, Xia Hu. Auto-Keras: An Efficient Neural Architecture Search System *arXiv:1806.10282* URL: https://arxiv.org/abs/1806.10282
2. Pieter Gijsbers, Erin LeDell, Janek Thomas, Sébastien Poirier, Bernd Bischl, Joaquin Vanschoren. An Open Source AutoML Benchmark URL: *arXiv:1907.00909* https://arxiv.org/abs/1907.00909