ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1.
Available from http://www.i6doc.com/en/.

Neural Architecture Search for Sentence Classification with BERT

Philip Kenneweg[1], Sarah Schröder[1] and Barbara Hammer[1] *

1- Bielefeld University - Faculty of Technology
Inspiration 1, 33615 Bielefeld - Germany

Abstract. Pre training of language models on large text corpora is common practice in Natural Language Processing. Following, fine tuning of these models is performed to achieve the best results on a variety of tasks. In this paper we question the common practice of only adding a single output layer as a classification head on top of the network. We perform an AutoML search to find architectures that outperform the current single layer at only a small compute cost. We validate our classification architecture on a variety of NLP benchmarks from the GLUE dataset. The source code is open-source and free (MIT licensed) software, available at https://github.com/TheMody/NASforSentenceEmbeddingHeads.

1 Introduction

In the last couple of years, the transformer architecture pioneered by Vaswani et al. [1] has enabled large pre-trained neural networks to efficiently tackle previously difficult Natural Language Processing (NLP) tasks with relatively few training examples. Pre-trained transformer based language models, which produce a vectorized representation of a given text input of arbitrary lengt, constitute a common tool which is deployed to facilitate fast transfer of knowledge. These language models produce contextualized word embeddings that are easy to process for a multitude of different tasks by shallow neural networks or other machine learning tools - common technologies which are used on top of such embeddings are methods such as SVM, clustering algorithms and linear regression [2, 3, 4]. Most language models are pre-trained on a large corpus of text data (for example Wikipedia, Reddit, etc.) using a variety of different unsupervised pre-training objectives (Masked Language Modeling, Next Sentence Prediction, etc.) [5, 6]. Further, besides an often shallow layer on top of the embedding, many common architectures use a fine-tuning step on a specific task to achieve good performance [7].

In this paper we take a closer look at NLP architectures built on top of transformer models using subsequent fine tuning steps for its optimization. Based on current best practices, we aim for an automation of these optimization steps w.r.t architectural choices and training procedure. We achieve this by two contributions: first, we define a landscape of promising options for a classification head beyond the commonly used single layer neural network with a softmax activation. Second, we introduce an AutoML pipeline [8], which enables us to automatically search these options to find the best possible classification architecture.

At present, the most common approach to fine tuning a language model is to process the outputs of the transformer with a single layer neural network [1, 5]. In the last

*We gratefully acknowledge funding by the BMWi (01MK20007E), from the MKW NRW in the project Bias aus KI-Modellen and by the BMBF (01IS19080 A).

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1.
Available from http://www.i6doc.com/en/.

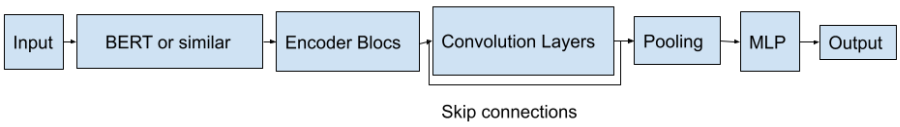

Fig. 1: Rough outline of all possible options for the search space of the proposed AutoML pipeline. (Not all options are used all the time)

years, more sophisticated fine tuning approaches have evolved to improve this baseline approach. Some approaches focus on reducing the number of parameters that have to be fine tuned for each individual task. This can be done by adding specialized classification layers to the original network while keeping the original network weights frozen [9].

Other approaches take different avenues to address the challenges which occur while fine tuning . One challenge is that the language model tends to forget the knowledge acquired by its extensive pre-training during fine tuning, or it over-fits on new data due to its inherent complexity and an often comparatively small size of the fine tuning data set. The approach dubbed SMART [7] addresses these shortcomings by introducing a smoothness inducing regularization technique and an optimization method, which prevents aggressive updating of the network weights.

In this paper we particularly introduce AutoML technologies, which enables us to investigate a variety of additional layers on top of the language model, while also fine tuning the weights of the underlying language model. To the best of our knowledge, this constitutes one of the first approaches in which the effect of additional classification architecture on the fine tuning process is extensively investigated.

2 Neural Architecture Search Space

The difficulty in finding a better classification head is caused by the fact that the transformer architecture is very powerful – hence it is able to adapt to most classification tasks well but running the risk of forgetting. Hence a new and possibly complex classification head has to be able to add novel capacity to this capability, while not causing problems with vanishing gradients or other confounding factors for the base transformer. With this in mind we determine a space of architectural choices, which constitute the possible search options in the neural architecture search which we are going to propose:

- pooling type (max, mean, [CLS])
- freeze base architecture (True,False)
- fully connected network (MLP); subchoices are:
  number of layers (1-5); number of neurons per layer (5-200))
- convolutional layer (True,False); subchoices are:
  number of heads (5-200); kernel size (3-11); number of layers (1-5); skip connections (True,False)

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

- encoder blocs (True,False); subchoices are:
  number of heads (1-16); number of layers (1-5)

A rough outline of the proposed Search Space for our neural architecture search can be seen in Figure 1. The pooling type is specific to BERT and similar bidirectional language models upon which we focus in this work. During training, BERT is conditioned to produce a good embedding for classification in place of the [CLS] token output. Other options common in literature to feed the output into the fully connected network are mean and max pooling over all output tokens. In between the dense layers of the MLP and the convolutional layers the activation function used is ReLU. 5 layers and 200 neurons for any type of layer was deemed the maximum to keep the parameter count of the new classification head to a reasonable amount (below 1% of the original BERT architecture). The convolutional layers were padded with zeroes to keep the input and output dimensions the same.

The search space spans $7.5e7$ different possible combination options if one assumes 10 different search options for the number of neurons per layer and number of heads during convolution. The search strategy used for our AutoML is Bayesian Optimization [10] with Hyperband Scheduling [11], as a particularly promising technology in the domain of architecture optimization. We refer to the output of this procedure, i.e., the best configuration found by this approach in a specific traning task, as $BERTtuned$.

3 Experimental Approach

In this section we detail our experimental design to investigate the effects of different architectures for Natural language classification tasks. We utilize AutoGluon [12] and the BertHugginface library [13] for implementation and the pre-trained Bert model ('bert-base-uncased') for all experiments.

3.1 Datasets

The Glue dataset by Wang et al. [14] is a collection of various popular datasets in NLP, and it is widely used to evaluate common natural language processing capabilites. All datasets used are the version provided by tensorflow-datasets 4.0.1. We also evaluate all approaches on the $small$ datasets. These are the same datasets as described below, only with the training data set size reduced to 500 randomly drawn samples. This scaling enables us to judge the capability of the architectural choices to adapt to new tasks based on a small number of training data only. More specifically, we use the Corpus of Linguistic Acceptability *cola*, the Stanford Sentiment Treebank *sst2*, the Microsoft Research Paraphrase Corpus *mrpc*, Recognizing Textual Entailment *rte*, and the Multi-Genre Natural Language Inference Corpus *mnli*.

3.2 Baseline and Implementation Details

As a Baseline comparison we evaluate BERT with a single dense layer used for classification on top and a learning rate of 2e-5, henceforth referenced to as $BERTbase$. The pooling operation used is [CLS]. Mean and max pooling were also tried but produced

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

Table 1: The final classification architectures found for the GLUE datasets.

method	sst2	cola	mrpc	mnli	rte	qqp
pooling	mean	[CLS]	[CLS]	max	[CLS]	max
number linear layers	5	1	4	3	1	1
hidden dim linear layers	50	-	74	117	-	-
number conv layers	0	0	4	3	0	2
number heads conv	-	-	107	9	-	159
kernel size	-	-	7	11	-	7
skip connection	-	-	True	True	-	True
number attention layers	1	0	4	0	0	0
number attention heads	4	-	16	-	-	-

inferior results. These values are taken from the original paper [5] and present good values for a variety of classification tasks.

All models are trained using a cosine decay of their learning rate with warm starting. All models are trained for 5 epochs on the glue datasets and for 10 epochs on the $small$ glue datasets. Batch size used for training was 32 and the Adam optimizer with betas (0.9,0.999) and epsilon 1e-08 was used.

4 Results

For our experiments, the 6 glue tasks cola, sst2, mnli, mrpc, rte and qqp are considered, following common evaluation approaches from literature. As can be seen in Table 2 $BERTtuned$ outperforms $BERTbase$ by a solid margin. On the cola as well as on the rte task the architecture search seems to have yielded little improvement, but on the mrpc task the accuracy was improved more significantly by 4%. On average the accuracy was improved by 0.9%. Larger accuracy improvement seems to correlate to more extensive changes to the classification head architecture as displayed in Table 1. The architectures found were using multiple convolutional layers and multiple encoder blocks, but never more than 2 dense layers in the MLP. All optimal architectures are adding skip connections if a convolutional layer is added. For all results (thus not in displayed in Table 1) the base layer was not frozen (freeze Base = False); this indicates, that the underlying transformer architecture is also adjusted towards the new tasks and not simply replaced by a new classification head.

In Tables 4 and 3 we see according results of the architecture search for the $small$

Table 2: Classification accuracies, on the sst2, cola, mrpc, rte, qqp and mnli datasets. Improvements are marked in **bold**.

method	sst2	cola	mrpc	mnli	rte	qqp	average
$BERTbase$	0.925	0.831	0.821	0.829	0.700	0.899	0.833
$BERTtuned$	**0.930**	0.831	**0.860**	**0.835**	0.700	**0.900**	**0.842**

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

Table 3: The final classification architectures found for the GLUE $small$ datasets.

method	sst2	cola	mrpc	mnli	rte	qqp
pooling	max	[CLS]	mean	max	mean	[CLS]
number linear layers	1	1	2	2	1	2
hidden dim linear layers	-	-	60	172	-	122
number conv layers	0	0	2	2	5	1
number heads conv	-	-	90	75	14	43
kernel size	-	-	7	11	5	11
skip connection	-	-	True	False	True	False
number attention layers	1	0	0	0	0	0
number attention heads	8	-	-	-	-	-

datasets. Here we see even more significant improvements than on the full datasets. This can be explained by the overall lower performance of the models and hence larger margin for improvement. The classification architectures found outperform the baseline in all cases. In this case the architectures found generally had less parameters to train than in the case of the full datasets. In all cases either only a convolutional layer, or an encoder block was added on top of the network, never both. On average the classification performance was improved by 3% over $BERTbase$. In comparison to the full datasets we see that even with fewer training examples it is still possible to upgrade the performance of $BERTbase$; thereby, the additional architecture has fewer additional parameters to avoid overfitting on the training data.

5 Conclusions

In this paper we presented a classification architecture for the BERT sentence embedder which improves the classification performance on a variety of datasets. In comparison to other approaches we do not modify the underlying transformer architecture or provide additional regularization, but rather append more complex network heads to the BERT network. This is agnostic towards the underlying architecture as well as task and so is more flexible in improving classification performance among a variety of architectures and tasks than many other approaches.

The exact contribution of the classification architectures found in comparison to a single layer is of interest and merits further research. Possible further research also includes other options for the search space of the NAS, as well as different transformer

Table 4: Classification accuracies, for the $small$ datasets. Improvements are marked in **bold**.

method	sst2	cola	mrpc	mnli	rte	qqp	average
$BERTbase$	0.835	0.761	0.733	0.463	0.566	0.736	0.682
$BERTtuned$	**0.869**	**0.763**	**0.752**	**0.529**	**0.606**	**0.751**	**0.712**

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1.
Available from http://www.i6doc.com/en/.

based language models.
The source code is available at
https://github.com/TheMody/NASforSentenceEmbeddingHeads,
It is open-source and free (MIT licensed) software.

Acknowledgements

We gratefully acknowledge funding by the BMWi (01MK20007E), from the MKW NRW in the project Bias aus KI-Modellen and by the BMBF (01IS19080 A).

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[2] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. *CoRR*, abs/1902.00751, 2019.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

[4] A. Radford, Jeffrey Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[6] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.

[7] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. SMART: robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *CoRR*, abs/1911.03437, 2019.

[8] Xin He, Kaiyong Zhao, and Xiaowen Chu. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, 2021.

[9] Asa Cooper Stickland and Iain Murray. BERT and pals: Projected attention layers for efficient adaptation in multi-task learning. *CoRR*, abs/1902.02671, 2019.

[10] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms, 2012.

[11] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

[12] Aaron Klein, Louis Tiao, Thibaut Lienart, Cedric Archambeau, and Matthias Seeger. Model-based asynchronous hyperparameter and neural architecture search. *arXiv preprint arXiv:2003.10865*, 2020.

[13] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, RÃ©mi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

[14] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. 2019. In the Proceedings of ICLR.