COS 484: Project Proposal

Transformer Design Space Exploration

Bhishma Dedhia, Xiaorun Wu, Shikhar Tuli
{bdedhia, xiaorunw, stuli} @princeton.edu

April 12, 2021

1 Introduction

Recent advancements in neural architecture search (NAS) provide various techniques to explore and optimize different models in the deep learning domain, from image recognition to speech recognition and machine translation [1]. For instance, in the computer-vision domain, there is a plethora of CNN architectures that exploit different design decisions (and some having even new basic operations [2]) to give different performances on different tasks [3, 4].

Similarly, in the Transformer domain, many architectures have been proposed. Among these, the popular ones include BERT, GPT, XLM, XLNet and many more. Just like in CNNs, many variations to these basic architectures have also been proposed to improve performance or efficiency: for example, BART, BORT, ConvBERT, DeBERTa, RoBERTa, etc. A unified approach is required to characterize and test 'why' each architecture is 'best' in its own way. For this, we need to modularize these popular Transformers and benchmark the design decisions for each such network on popular NLP tasks.

2 Project Objective

In this project, our aim is three-fold: first, we aim to explore the Transformer design space to show how certain design-decisions lead to better performance on different tasks; second, we would like to represent all the literature on different Transformer architectures using a unified embedding space. Furthermore, we intend to use this embedding space to train a surrogate model (using active learning), which predicts the performance of different architectures for every NLP benchmarking task.

Specifically, to achieve the desired objective, we plan to explore the following:

1. Taking inspiration from a recent work by Khetan and Karnin [5], we start with a basic exploration of various design choices in BERT. For instance, Table 1 shows the design choices in the vanilla BERT architecture. However, rather than keeping a fixed objective, i.e. pruning to optimize efficiency with the same accuracy, we test the question – 'which direction should we change the hyper-parameters in the design space to improve performance on different tasks?'. For this, various NAS algorithms could be used. As a proof-of-concept standard Bayesian Optimization techniques could be used. After this exercise, the task would be to argue 'why' these architectural decision were helpful for this specific tasks – highlighting the nuances of that task and the advantages of certain architectures over others for that task. Different datasets we could look at include - SQuAD, MNLI, MRPC, SST-2, etc. Note that each new architecture that would be

tested would be fine-tuned on the 'nearest' pre-trained Transformer. We define the computation of the nearest Transformer next.

BERT-base		
number of enoder layers	l	12
hidden size	h	768
number of self-attention heads	a	12
feed-forward dimension	f	$4h$
key-query dimension for attention	k	h/a
value dimension for attention	v	h/a

Table 1: Architecture design decisions for vanilla BERT. Source: [5]

A sample image of BERT architecture is shown below:

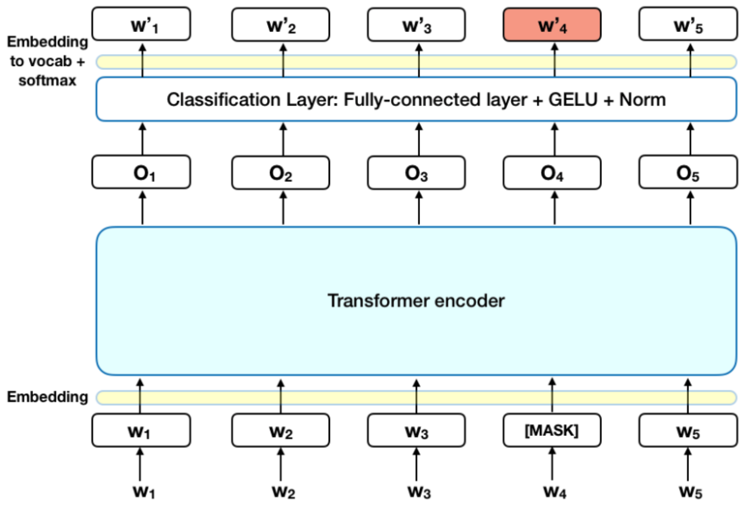
w'1
w'2
w'3
w'4
w'5
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
w1
w2
w3
[MASK]
w5
w1
w2
w3
w4
w5

Figure 1: BERT architecture

2. To visualize and characterize the design space efficiently, we would convert every possible Transformer architecture possible (within the confines of the hyper-parameters) to a computational graph. Using Weisfeiler-Lehman (WL) Kernel (with additional domain-knowledge) a similarity metric between two graphs can be computed. Once a similarity metric is available, Multi-Dimensional Scaling (MDS) or other techniques could be used to obtain a 'Transformer2vec' embedding. On this, co-sine similarity could be used to get the nearest Transformer architecture in the design space.

3. Once we have an embedding that represents the entire design space of Transformers, we can train a surrogate model that predicts the accuracy for every task given the architecture embedding. This could be done using standard active learning approaches. For instance, Bayesian Optimization or Mixture-Density Networks could be used to optimize uncertainty of the model over the design space [6, 7]. This simplified network could speed up architecture search in downstream tasks, which could be optimized in the second step with actual training around the optimum [8].

3 Related Work

There has been a lot of work in architecture search recently. However, most works take into account either CNNs or simple-RNNs or LSTM [1, 9, 10]. A recent work explores the design space around BERT [5], however, it only concerns pruning BERT more efficiently and does not target optimizing accuracy over different tasks. Further, they only target to get a computationally lighter model with a fixed set of trainable parameters, rather than searching for the highest accuracy model and inspecting its architecture. For the Transformer2vec embedding, we took inspiration from a recent work by Hsin-Pai Cheng, et al. [11], where they show how structural information of a computational graph helps in AutoML tasks. Still, this work only focuses on CNNs and a follow-up study on Transformers is one of the objectives of this project.

4 Code & Model References

Although the work by Khetan and Karnin [5] would have served as a good starting point, unfortunately their code-source is not available. However, a 'modular' BERT could easily be implemented by taking inspiration from this work and by building upon the this comprehensive repository: `https://github.com/huggingface/Transformers`. The surrogate model could be generated using the Surrogate Modeling Toolbox: `https://github.com/SMTorg/smt`. Relevant graph operations already implemented for CNNs could be used in this project as well: `https://github.com/google-research/nasbench`.

5 Computational Requirements

For fine-tuning every new Transformer architecture in the design space, a GPU could be used from the Princeton Tiger cluster. Multiple architecture points in the design space could be trained in parallel on the cluster. Since each fine-tune operation would take the weights from the nearest trained Transformer model, it should not take more than a few hours to train each model. This training time would decrease as more points are trained in the design space.

6 Timeline

- Mar 27 – Reading relevant papers, make sure we understand the BERT structure, relevant implementations
- Apr 1 – Prepare a modular simulator that can train a model with the given hyper-parameters of the architecture design space.
- Apr 12 – Formulate graph representation of each model and train a Transformer2vec embedding.
- Apr 18 – Use an embedding to train a surrogate model that can be used for fast and efficient architecture search. This will require parallel training of different models in the design space.
- Apr 25 – Analyse the results and prepare the project report.
- May 4 – Proof-read, and conclude the project

References

[1] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. 2017.

[2] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[4] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[5] Ashish Khetan and Zohar Karnin. schuBERT: Optimizing elements of BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2807–2818, Online, July 2020. Association for Computational Linguistics.

[6] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[7] Burr Settles. Active learning literature survey. 2009.

[8] Xiaoliang Dai, Peizhao Zhang, Bichen Wu, Hongxu Yin, Fei Sun, Yanghan Wang, Marat Dukhan, Yunqing Hu, Yiming Wu, Yangqing Jia, Peter Vajda, Matt Uyttendaele, and Niraj K. Jha. Chamnet: Towards efficient network design through platform-aware model adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[9] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. NAS-bench-101: Towards reproducible neural architecture search. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7105–7114. PMLR, 09–15 Jun 2019.

[10] Hanna Mazzawi, X. Gonzalvo, A. Kraun, P. Sridhar, Niranjan Subrahmanya, I. Lopez-Moreno, H. Park, and Patrick Violette. Improving keyword spotting and language identification via neural architecture search at scale. In *INTERSPEECH*, 2019.

[11] Hsin-Pai Cheng, Tunhou Zhang, Yixing Zhang, Shiyu Li, Feng Liang, Feng Yan, Meng Li, Vikas Chandra, Hai Li, and Yiran Chen. NASGEM: Neural Architecture Search via Graph Embedding Method, 2020.