“IMPACT OF FINE-TUNING ON MODEL TRAINING: A COMPREHENSIVE STUDY USING THE SQuAD BENCHMARK”

Suresh Kumar Choudhary, Swiss School of Business and Management Geneva, Switzerland, sureshdewenda@gmail.com

Bojan Kostandinovic, Swiss School of Business and Management Geneva, Switzerland, bojan@ssbm.ch

“Abstract”

Purpose *- This study investigates the effect of fine-tuning on model training, and the impact of model training on the environment in terms of carbon footprint emissions. As language modeling evolves, we are obtaining complex and large models with a large number of trained parameters. However, only large enterprises are able to train these large models using large datasets. It is not possible for small and mid-size enterprises due to their computation resources capability. This study proposes a fine-tuning model training pipeline for such enterprises to get the same model capability as large enterprises with less number of model parameters training using SQuAD benchmark. The pre-trained BERT base model is incorporated in combination with fine-tuning methods in this study.*

Findings *- The results explain us that fine-tuned models need less computation time and less number of trainable parameters without losing too much model capabilities compared to the fully trained model, which indirectly helps us to protect the environment as it emits less carbon footprints.*

Keywords: *LLMs; PEFT; Fine Tuning, Carbon footprints; BERT; SQuAD*

1 Introduction

It is essential to look into alternative approaches of fine-tuning(FT) instead of full model training with the language model evolution. Pre-trained models learn the representation of languages and the inner semantics as well, which is helping day to day life of common people by solving impossible tasks in possible ways and establishing an interest for researchers to deep dive into this field. These models can be used by FT on small datasets for diverse tasks. SQuAD benchmark by Rajpurkar (2016) has helped a lot of research and researchers to get insights to solve diverse tasks in language modeling. We are using it to assess the effectiveness of FT methods.
Here, we look into the different FT strategies and evaluate the effect of it on model training. We focus on model computation time and number of trainable parameters required for model training. This study aims to build effective strategies for optimizing language models for specific tasks, paving the way for more efficient and effective implementations without losing the model capability in real-world applications.

2 Background

2.1 Benchmark dataset

In this study, we selected a widely adapted benchmark dataset named as Stanford Question Answering Dataset (SQuAD) (Rajpurkar, 2016). It contains around 100000 question answer pairs. It is acquired using more than 500 Wikipedia articles where questions were framed by crowd-workers. These articles were from different domains. It contains the domain category, id, questions framed by crowd-workers, wikipedia article text paragraphs, and answers. It is divided in train and test dataset with 87599 and 10570 samples respectively. The dataset information can be found in Table I.

Dataset	Features	Number of Rows
train	[’id’, ’title’, ’context’, ’question’, ’answers’	87599
test	[’id’, ’title’, ’context’, ’question’, ’answers’	10570

Table I: Dataset information

2.2 Large language models

Attention based transformer model by (Vaswani *et al.*, 2017) revolutionized the research in the large language models (LLMs). Invention of LLMs like BERT (Devlin *et al.*, 2018), GPT (Radford *et al.*, 2018), XLNET (Z. Yang *et al.*, 2019) etc. attracted the attention of people. BERT used a bidirectional transformers training approach which improved the quality and efficiency to understand tokens and semantics of the token of a language. GPT was fine-tuned on large text data to generate coherent and contextually relevant text. XLNet was proposed to overcome the limitations of the BERT model. It used the permutation-based training objectives and looked at every possible permutation of the input pattern. Sun *et al.*, (2019) proposed ERNIE which refines the embedding quality by incorporating knowledge from structured knowledge bases and semantic matching. ERNIE uses lexical and structural information better than the conventional pre-training methods.
RoBERTa by Y. Liu *et al.*, (2019) builds upon BERT’s success by tuning the hyperparameters and training objectives. This results in a robust model which outperforms BERT in various natural language understanding tasks. Trained on large datasets with dynamic masking and longer sequences, RoBERTa achieves cutting-edge findings in analysis of emotions, identification of specific entities and responding to inquiries and shows significant progress on various tasks.
ALBERT by Lan *et al.*, (2019) solves the LLMs problem by using parameter sharing and cross layer parameter sharing. Fewer parameters and same performance ALBERT is more efficient, it can scale to larger batch size and sequence length, faster training and inference time and same efficiency across various tasks.
Raffel *et al.*, (2020) introduced T5, a unified transformer architecture that every NLP task is framed as a problem involving the conversion of text to text. By unifying activities such as interpretation, condensing, and responding questions into a single text-to-text framework, T5 gets cutting edge results across many tasks. Zaheer *et al.*, (2020) introduced Big Bird, a transformer model designed to handle longer sequences better. Big Bird combines sparse attention and local- global attention patterns to scale to sequences of tens of thousands while performing well on many sequence modeling tasks.
Recently there have been more and more advanced and resource hungry LLMs that require a lot of memory (RAM). For example, the earlier models like BERT, RoBERTa, XLNet, T5 large and GPT-3 have parameter count ranging from millions to billions. Deploying and FT, these models require a lot of memory resources, hence the need for solutions that reduce computational cost and are production ready applications.

2.3 Fine-tuning approaches

Fine-tuning (FT) of a full model involves adjusting all layers of a model on the target task, which is effective when the target dataset is more or less similar to the trained model (Devlin *et al.*, 2018). Alternatively, feature extraction is applied only on the final layers of the model, keeping earlier layers fixed to reduce computational costs (Yosinski *et al.*, 2014). **Domain Adaptation** reduces the difference between source and target domains by employing adversarial methods, allowing mod- els to perform well in different domains despite distributional differences (Zhang *et al.*, 2019). It incorporates a layer with optical flow reversal to guarantee that feature representations remain consistent across different domains, thereby improving domain adaptation. Intermediate task FT involves adapting a model on a related but different task before FT on the final target task, so more effective adaptation (Howard and Ruder, 2018). In **Parameter-Efficient FT (PEFT)**, Adapter layers add small modules in each model layer and fine-tuned, while keeping the original weights frozen to make it more efficient (Poth *et al.*, 2023). LoRA (Low-Rank Adaptation) incorporates low-rank decompositions into the model’s weight matrices, reducing the number of parameters updated during FT (Hu *et al.*, 2021). PEFT tackles the computational cost of full FT as model grows in size. PEFT methods update only the model parameters to optimize efficiency. Houlsby *et al.*, (2019) comes up with new ways to tackle the computational cost of FT PLMs. The paper proposes Top-K Tuning, a method that updates only the top k layers of the PLM while keeping the lower pre-trained layers

intact, efficiency without performance compromise. Also, the authors explore adapter-based approaches, adding small adapter modules for FT instead of modifying the original model parameters. Through these strategies, the paper offers valuable insights into achieving parameter-efficient in tasks, opening doors for increased scalable and efficient deployment of models in practical scenarios. **Knowledge Distillation** approach involves FT in a ”student” model by leveraging the output probabilities (soft targets) from a ”teacher” model, transferring knowledge effectively (Hinton *et al.*, 2015). L2 regularization imposes an extra concept to the loss function in order to avoid fitting problems by constraining deviations from pre-trained weights (H. Wang *et al.*, 2020). Elastic Weight Consolidation (EWC) adds a quadratic penalty to preserve important weights from previous tasks, mitigating catastrophic forgetting (Kirkpatrick *et al.*, 2017). The research paper by Lee *et al.*, (2019) proposes Mixout, a novel regularization technique for FT models. It addresses overfitting by randomly mixing a fraction of weights with their pre-trained values during training. This encourages robust representations, improving generalization on text classification tasks. Mixout is easily integrated, outperforming dropout in generalization performance. The paper provides theoretical justifications and practical implementation guidelines, offering a promising approach for regularization in fine-tuned models, enhancing generalization and robustness. Task-specific data augmentation involves generating augmented training data to enhance model robustness (Shorten and Khoshgoftaar, 2019). Curriculum learning starts with simpler examples and gradually increases difficulty to aid in progressive learning (Bengio *et al.*, 2009). In **Multi-Task Learning**, joint FT involves training a model on multiple tasks simultaneously, sharing representations across tasks to improve generalization (Peters *et al.*, 2019). FT of sequential task leverages of prior task knowledge to enhance performance on subsequent ones (Rusu *et al.*, 2016). **Meta-Learning** equips a model to rapidly adjust to new tasks with minimal extra training, improving its adaptability (Finn *et al.*, 2017). Contrastive learning FT employs contrastive loss to improve the model’s capacity to differentiate between categories or tasks (Chen *et al.*, 2020). Finally, pruning and quantization refine models by removing less significant weights or quantizing them to enhance efficiency (Han *et al.*, 2015).
Recent language models have made progress in two areas. Firstly, there’s been an adoption of few-shot learning methods, so models can be fine-tuned with minimal labeled data and be more versatile across tasks. For example, Brown *et al.*, (2020) tested few-shot learning of LLMs.
Secondly, efforts are being made to develop efficient FT methods to reduce computational cost and memory requirement of large models. Hu *et al.*, (2021) proposed a method called LoRA for effective FT with low-dimensional adaptation. In this paper, we are focusing on the second area.

3 Method

In this study, we created a pipeline to perform experiments to see the impact of FT on SQuAD dataset, which has been depicted in Figure 1. It includes data normalization, pre-tokenization, tokenization, pre-trained model selection, FT method selection, model training, postprocessing of model output and finally model evaluation on test dataset.

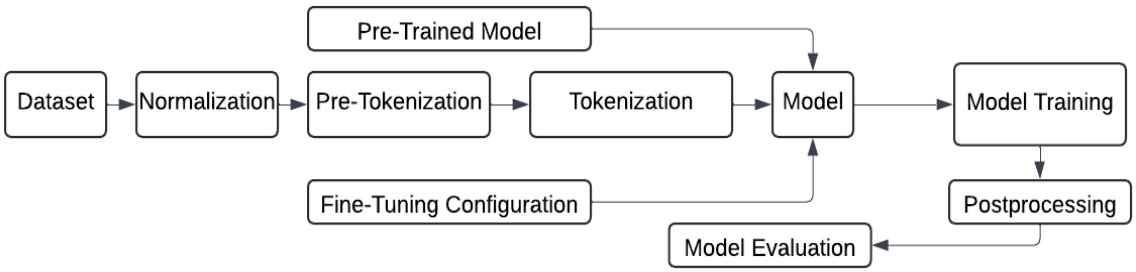

Figure 1: Study Methodology Pipeline

3.1 Text normalization

In this step, the dataset text is converted into a standard text. It involves removing extra spaces, special characters, ascents from the text and converts the text into lower case to make text uniform. There are many ways to normalize the text, we are discussing only which has been used in this study. There are many other data normalization standards as well, one can follow that as well based on the requirements by looking into pros and cons. This step helps the model to learn the text in a uniform

way without worrying about the cases and variations in the text. We have followed the same implementation for data normalization like in BERT (Devlin *et al.*, 2018).

3.2 Pre-tokenization

This step includes splitting the text into chunks or words using spaces or special symbols. It prepares the data for the tokenization step. Special tokens [CLS] and [SEP] are added at the beginning of sentence and at the end of the sentence respectively.

3.3 Tokenization

This step takes input from the previous step (pre-tokenization) and converts words or chunks into tokens according to vocabulary. It uses a word-piece algorithm (Devlin *et al.*, 2018) to convert into tokens. These tokens are mapped with sequence numbers in vocabulary to convert it into numeric form which is suitable for any model to train the model.

3.4 Model selection

There are many pretrained base models available which have been trained on large text with millions to billions of parameters. We selected the popular BERT model (Devlin *et al.*, 2018) for this experiment which suits our computation resources. The statistics of this model can be found in Table II.

Statistic	Value
Number of Parameters	≈ 110 million
Number of Layers	12
Hidden Size	768
Number of Attention Heads	12
Intermediate Layer Size	3072
Maximum Sequence Length	512
Vocabulary Size	30,522
Token Embedding Dimension	768
Attention Head Size	64

Table II: Statistics of BERT-base-uncased Model

3.5 Fine-tuning approaches

There are many PEFT methods, however, we selected the more recent and popular methods for this study like full parameters FT (FT BERT), LoRA (Hu *et al.*, 2021), LoHA (Hyeon-Woo *et al.*, 2021), p-tuning (X. Liu *et al.*, 2023), prefix tuning (Li and Liang, 2021), and IA3 (H. Liu *et al.*, 2022). LoRA uses the low rank matrix for trainable parameters and updates it instead of updating the big matrix. LoHA is a variation of LoRA with a new concept of matrix product instead of multiplication. P-tuning and prefix-tuning are soft prompt-based FT approaches. IA3 focuses more on the attention part of the model, it was introduced to improve the LoRA.

3.6 Model training, postprocessing and evaluation

The selected pretrained model configured with the selected FT method is trained on a training dataset. After training, we process the output to make it usable for model evaluation. In the model evaluation step, we use the evaluation metric to get the model results on the test dataset.

4 Implementation

To implement the above-mentioned pipeline in Section 3, we used python and pytorch packages. To track the experiment, we used MLflow (MLflow, 2024). The implemented code has been uploaded on GitHub[1].

5 Empirical Results

As part of experimental setup configuration, we used the maximum sequence length 384 (question and text), sliding token window length (doc stride = 128), learning rate 2e-5, batch size 16, train epochs 2, weight decay 0.01, and maximum answer length 30 as a parameter to the model. As part of FT method configuration, we used LoRA(r = 8,alpha = 32,module dropout = 0.1), LoHA(r = 8,alpha = 32,module dropout = 0.1), IA3(default), p-tuning(num virtual tokens = 20, encoder hidden size = 128), prefix-tuning(num_virtual_tokens = 20). The loss, Exact Match (EM), and F1-score used to evaluate the model (Rajpurkar, 2016). We calculated the number of trainable parameters required and training time for this study as well.
The table III shows the model performance calculated by us on a CPU machine (16GB RAM, 6 cores per task) based on the used FT approach and BERT base model. It lists training and testing losses, Exact Match (EM), F1-score, number of trainable parameters, total parameters, the percentage of trainable parameters, and the training time. It shows FT BERT and LoRA+BERT have the lowest training loss (1.16, 2.76) and test loss (1.02, 2.12) respectively compared to the others, while P-tuning+BERT shows the highest test loss (4.56) on unseen data. FT BERT and LoRA+BERT also perform best in terms of EM (79.20%, 46.23%) and F1-score (86.97%, 56.73%). P-tuning+BERT shows a relatively low F1-score (10.56%) and Exact Match (6.49%) compared to the other models. FT BERT, LoRA+BERT and LoHA+BERT have significantly higher numbers of trainable parameters (110 million, 296,450 and 591,362, respectively) compared to the other models. This gives them greater flexibility in adapting to the task. IA3+BERT has the smallest number of trainable parameters (66,050), which could explain its performance as the model has fewer parameters to adjust or train during training. FT BERT and LoHA+BERT have the highest percentage of trainable parameters (100%, 0.5401% respectively), meaning more of its parameters are being updated during training. IA3+BERT has the lowest percentage of trainable parameters (0.0606%), which suggests that only a small portion of the model is being fine-tuned, potentially limiting its ability to perform well on the task. FT BERT and LoHA+BERT require the longest training time (11504.30, 1170.57 minutes respectively), which is expected due to its larger number of trainable parameters. P-tuning+BERT has the shortest training time (911.17 minutes), which might be due to its fewer trainable parameters (230,914).

Model	Train Loss	Test Loss	EM (%)	F1 (%)	Trainable Params	Total Params	Trainable (%)	Train Time (min)
FT BERT	1.16	1.02	79.20	86.97	110 Million	110 Million	100	11504.30
LoRA+BERT	2.76	2.12	46.23	56.73	296 450	109 189 636	0.2715	933.31
IA3+BERT	4.42	4.11	5.62	12.56	66 050	108 959 236	0.0606	997.51
LoHA+BERT	4.18	3.77	8.33	15.35	591 362	109 484 548	0.5401	1170.57
Prefix-tuning+ BERT	4.56	4.24	5.45	11.82	370 178	109 263 364	0.3388	964.67
P-tuning+BERT	4.59	4.56	6.49	10.56	230 914	109 124 100	0.2116	911.17

Table III: Model Results and Parameters Summary

6 Discussion

In this study, our aim was to propose a FT model pipeline and to see the effect of FT on computation time (train time) and memory resources (in this case the no. of trainable params) with limited computation resources as small and mid-size enterprises don't have computation resources compared to large enterprises. So, they can fine tune the existing model as per their requirements. We can clearly infer from the Table III that FT reduces a significant train time compared to full model parameters training with significantly less required trainable parameters. So, it emits a significantly less Carbon footprint (Strubell *et al.*, 2020) compared to full training of a model. As per Rajpurkar (2016), the human evaluation result on the test set was 77.0% (EM), and 86.8% (F1). The baseline result was 40.4%(EM) and 51.0% (F1) with a logistic regression model. In our case, LoRA+BERT is slightly better and FT BERT significantly better compared to baseline model, however it seems

there is a chance of under-fitting in LoRA+BERT. It needs more parameters to get trained. So, we need to correctly configure the parameters of the LoRA+BERT model because Hu *et al.*, (2021) has shown comparable results with a fully trained model. Hyeon-Woo *et al.*, (2021), H. Liu *et al.*, (2022), Hu *et al.*, (2021), X. Liu *et al.*, (2023), and Li and Liang (2021) fine-tuned the model with significantly less number of parameters and have obtained the same capability like full model training with correctly configured parameters on other datasets. In our case, we need to find the right hyper-parameters to get full capability like the full trained model. Due to our limited computation resources, we didn't get a chance to search the hyper-parameters to configure in the model. The T5 (11 billion parameters) (Raffel *et al.*, 2020) has provided the gold standard for this dataset with 90.06 (EM) and 95.64 (F1). However, it is a single alone model with a significant large number of parameters and not fine-tuned. According to Zafrir *et al.*, (2021), the fully trained BERT model (110 million parameters) got results with 80.80 (EM) and 88.50 (F1) on this dataset, which is approximately equal to our FT BERT model.

P-tuning+BERT and prefix-tuning+BERT models use the soft prompt approach, and we didn't have any prompts in this dataset so results on these models are not surprising, however still there is a scope to significantly improve it with right parameters configuration.

Future research could look into the other PEFT methods on this dataset to get better performance. One can explore and investigate the impact of carbon footprints on the environment due to these models.

7 Conclusion

We have proposed a FT model pipeline with clear steps highlighting the FT using the SQuAD benchmark. This pipeline can be helpful for small and mid-enterprises which don't have enough computation resources. The results show that a fine-tuned model requires less training time and less number of parameters. So, it is more environmentally friendly in terms of carbon footprint emissions. While a fully trained model shows best performance, it will still be crucial to understand and optimize PEFT methods to build efficient and effective models for diverse applications in future.

References

Bengio, Y., Louradour, J., Collobert, R. and Weston, J. (2009) 'Curriculum learning'. *Proceedings of the 26th annual international conference on machine learning*, 41–48.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A. et al. (2020) 'Language models are few-shot learners'. *Advances in neural information processing systems*, 33, 1877–1901.

Chen, T., Kornblith, S., Norouzi, M. and Hinton, G. (2020) 'A simple framework for contrastive learning of visual representations'. *International conference on machine learning*, PMLR, 1597–1607.

Choudhary, S.K. (2024) [online] GitHub repository: https://github.com/sureshkuc/Freie-Universitat-Berlin/tree/main/LLMs.

Devlin, J., Chang, M., Lee, K. and Toutanova, K. (2018) 'BERT: Pre-training of deep bidirectional transformers for language understanding'. *arXiv preprint arXiv:1810.04805*.

Finn, C., Abbeel, P. and Levine, S. (2017) 'Model-agnostic meta-learning for fast adaptation of deep networks'. *International conference on machine learning*, PMLR, 1126–1135.

Han, S., Mao, H. and Dally, W. J. (2015) 'Deep compression: Compressing deep neural networks with pruning, trained quantization, and Huffman coding'. *arXiv preprint arXiv:1510.00149*.

Hinton, G., Vinyals, O. and Dean, J. (2015) 'Distilling the knowledge in a neural network'. *arXiv preprint arXiv:1503.02531*.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B. et al. (2019) 'Parameter-efficient transfer learning for NLP'. *International conference on machine learning*, PMLR, 2790–2799.

Howard, J. and Ruder, S. (2018) 'Universal language model fine-tuning for text classification'. *arXiv preprint arXiv:1801.06146*.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z. et al. (2021) 'LoRA: Low-rank adaptation of large language models'. *arXiv preprint arXiv:2106.09685*.

Hyeon-Woo, N., Ye-Bin, M. and Oh, T. H. (2021) 'FedPara: Low-rank Hadamard product for communication-efficient federated learning'. *arXiv preprint arXiv:2108.06098*.
Kirkpatrick, J., Pascanu, R., Rabinowitz, N. et al. (2017) 'Overcoming catastrophic forgetting in neural networks'. *Proceedings of the national academy of sciences*, 114(13), 3521–3526.
Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P. and Soricut, R. (2019) 'ALBERT: A lite BERT for self-supervised learning of language representations'. *arXiv preprint arXiv:1909.11942*.
Lee, C., Cho, K. and Kang, W. (2019) 'Mixout: Effective regularization to fine-tune large-scale pre-trained language models'. *arXiv preprint arXiv:1909.11299*.
Li, X. L. and Liang, P. (2021) 'Prefix-tuning: Optimizing continuous prompts for generation'. *arXiv preprint arXiv:2101.00190*.
Liu, H., Tam, D., Muqeet, A., Agarwal, O. et al. (2022) 'Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning'. *Advances in neural information processing systems*, 35, 1950–1965.
Liu, X., Zheng, Y., Du, Z. et al. (2023) 'GPT understands, too'. *AI Open*.
Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O. et al. (2019) 'RoBERTa: A robustly optimized BERT pretraining approach'. *arXiv preprint arXiv:1907.11692*.
MLflow (2024) 'MLflow'. Available at: https://mlflow.org/docs/latest/index.html.
Peters, M. E., Ruder, S. and Smith, N. A. (2019) 'To tune or not to tune? Adapting pretrained representations to diverse tasks'. *arXiv preprint arXiv:1903.05987*.
Poth, C., Winther, M., Thomas, P. et al. (2023) 'Adapters: A unified library for parameter-efficient and modular transfer learning'. *arXiv preprint arXiv:2311.11077*.
Radford, A., Narasimhan, K., Salimans, T. and Sutskever, I. (2018) 'Improving language understanding by generative pre-training'.
Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y. et al. (2020) 'Exploring the limits of transfer learning with a unified text-to-text transformer'. *Journal of machine learning research*, 21(140), 1–67.
Rajpurkar, P., Zhang, J., Lopyrev, K. and Liang, P. (2016) 'SQuAD: 100,000+ questions for machine comprehension of text'. *arXiv preprint arXiv:1606.05250*.
Rusu, A. A., Rabinowitz, N. C., Desjardins, G. et al. (2016) 'Progressive neural networks'. *arXiv preprint arXiv:1606.04671*.
Shorten, C. and Khoshgoftaar, T. M. (2019) 'A survey on image data augmentation for deep learning'. *Journal of big data*, 6(1), 1–48.
Strubell, E., Ganesh, A. and McCallum, A. (2020) 'Energy and policy considerations for modern deep learning research'. *Proceedings of the AAAI conference on artificial intelligence*, 34(09), 13693–13696.
Sun, Y., Wang, S., Li, Y., Feng, S. et al. (2019) 'ERNIE: Enhanced representation through knowledge integration'. *arXiv preprint arXiv:1904.09223*.
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J. et al. (2017) 'Attention is all you need'. *Advances in neural information processing systems*, 30.
Wang, H., Zhang, Z., Li, Z., Zhang, Z. et al. (2020) 'Neural pruning via growing regularization'. *arXiv preprint arXiv:2012.09243*.
Yang, Z., Dai, Z., Yang, Y. et al. (2019) 'XLNet: Generalized autoregressive pretraining for language understanding'. *Advances in neural information processing systems*, 32.
Yosinski, J., Clune, J., Bengio, Y. and Lipson, H. (2014) 'How transferable are features in deep neural networks?'. *Advances in neural information processing systems*, 27.
Zafrir, O., Boudoukh, G., Izsak, P. et al. (2021) 'Prune once for all: Sparse pre-trained language models'. *arXiv preprint arXiv:2111.05754*.
Zaheer, M., Guruganesh, G., Dubey, A. et al. (2020) 'Big bird: Transformers for longer sequences'. *Advances in neural information processing systems*, 33, 17283–17297.
Zhang, Y., Li, T., Ding, Z. and Sun, X. (2019) 'Domain-symmetric networks for adversarial domain adaptation'. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5031–5040.