HybridBERT - Making BERT Pretraining More Efficient Through Hybrid Mixture of Attention Mechanisms

Gokul Srinivasagan
Saarland University
Saarland Informatics Campus
Saarbrücken, Germany
gokulsrinivasagan@gmail.com

Simon Ostermann
German Research Center for
Artificial Intelligence (DFKI)
Saarbrücken, Germany
simon.ostermann@dfki.de

Abstract

Pretrained transformer-based language models have produced state-of-the-art performance in most natural language understanding tasks. These models undergo two stages of training: pretraining on a huge corpus of data and fine-tuning on a specific downstream task. The pretraining phase is extremely compute-intensive and requires several high-performance computing devices like GPUs and several days or even months of training, but it is crucial for the model to capture global knowledge and also has a significant impact on the fine-tuning task. This is a major roadblock for researchers without access to sophisticated computing resources. To overcome this challenge, we propose two novel hybrid architectures called HybridBERT (HBERT), which combine self-attention and additive attention mechanisms together with sub-layer normalization. We introduce a computing budget to the pretraining phase, limiting the training time and usage to a single GPU. We show that HBERT attains twice the pretraining accuracy of a vanilla-BERT baseline. We also evaluate our proposed models on two downstream tasks, where we outperform BERT-base while accelerating inference. Moreover, we study the effect of weight initialization with a limited pretraining budget. The code and models are publicly available at: www.github.com/gokulsg/HBERT/.

1 Introduction

The last few years have witnessed ground-breaking research on pretrained transformer-based language models. These large language models usually follow a two-stage training process: the initial pretraining stage for learning global knowledge using large text collections and the later fine-tuning stage for adapting the learned knowledge to a specific task. Pretraining is the most crucial and most computationally expensive phase and often requires modern computing devices like GPUs or TPUs and several weeks or even months. Pretraining is thus a major roadblock for researchers who do not have access to sophisticated computing resources. For example, the BERT model (Devlin et al., 2018) was pretrained on 16 TPUs for 4 days. Such modern computing devices cannot be easily accessed by individual researchers, which limits the freedom of researchers to explore other architectures for a given task and is a hurdle to the development of highly optimized neural architectures.

Following the scaling laws (Kaplan et al., 2020), researchers tried to improve the performance by increasing the model size, data volume and training time. This resulted in extremely huge models with several billion parameters. Even fine-tuning such enormous language models with a limited compute power is extremely challenging and it is one of the main reasons that motivated researchers to explore alternative approaches to make the best use of the existing pretrained models rather than training from scratch. Though approaches like prompt tuning (Lester et al., 2021) or adapters (Houlsby et al., 2019) provide competitive results, they restrict any architectural modifications to the underlying pretrained model. Making the pretraining process more computationally inexpensive would motivate researchers to explore other architectures.

In our work, we try to address this problem by imposing restrictions on computational devices and training time during the pretraining stage of a BERT model. We also introduce two new models which we call *HybridBERT* (HBERT) and use - for the first time, to the best of our knowledge - a hybrid mixture of self-attention and additive attention together with sub-layer normalization. We show that on a limited time budget of 1 or 2 days, our usage of additive attention and sub-layer normalization increases both the pretraining performance and downstream performance of a reference vanilla BERT model on two tasks, namely intent classification and emotion recognition.

With our work, we aim to provide a means to researchers without access to large computing devices to pretrain their own high-performance language models. Making pretraining more efficient and not dependent on sophisticated computing resources also helps to save cost and lower the emission of CO_2, thus proving to be more environmentally friendly.

2 Related work

The BERT model (Devlin et al., 2018) was pretrained on 16 TPUs for 4 days. The equivalent time on 8 Nvidia V100 GPUs will be 11 days. There has been recent work on introducing restrictions on the BERT pretraining: Izsak et al. (2021) tried to train BERT-large for a single day using 8 V100 GPUs. Later work (Geiping and Goldstein, 2022) reduced the GPU usage from 8 to 1, still training the model for a single day. Inspired by these works, we restrict the pretraining to a single GPU, utilizing one of the most commonly used GPUs for pretraining.

After the success of transformer-based models on natural language understanding tasks, the field of efficient attention mechanisms came into the spotlight. Several works (Child et al., 2019; Beltagy et al., 2020) tried to reduce the quadratic complexity of the self-attention mechanism. Few approaches (Kitaev et al., 2020) used hashing techniques to accelerate the self-attention computation. Approximating the self-attention mechanisms by low-rank matrices (Wang et al., 2020; Xiong et al., 2021) is also an active research direction. With a linear time complexity, *additive attention* (Wu et al., 2021) proves to be an efficient alternative to self-attention, which is why we employ it in this work.

Layer normalization is a lightweight component in the BERT architecture that can influence the learning capabilities of the model. While the conventional BERT-based models use post-layer normalization, the decoder-based model (Radford et al., 2019) and vision transformers (Dosovitskiy et al., 2020) show improvements using pre-layer normalization. Earlier work (Geiping and Goldstein, 2022) suggested pre-layer normalization to be more beneficial during computing resource crunch. Recent work (Wang et al., 2022) tries to generalize layer normalization across different models by introducing sub-layer normalization. Sub-layer normalization has been shown to improve the performance of models on various tasks in the text, speech, and vision domains, which is why we decided to employ it for our work.

3 Method

3.1 Additive attention

The architecture of additive attention mechanism is as shown in the figure 1. Unlike the pairwise interaction of tokens in self-attention, additive attention uses a global context vector for transforming the representations of tokens as follows:

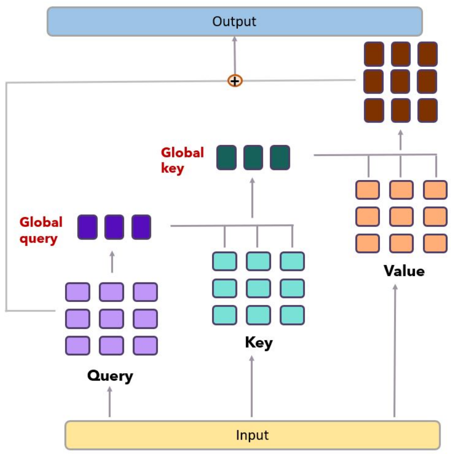


Figure 1: Additive attention mechanism

$$\alpha_i = \frac{\exp\left(\mathbf{w}_q^T \mathbf{q}_i/\sqrt{d}\right)}{\sum_{j=1}^{N}\exp\left(\mathbf{w}_q^T \mathbf{q}_j/\sqrt{d}\right)}$$

$$\mathbf{Q}_{\mathbf{global}} = \sum_{i=1}^{N} \alpha_i \mathbf{q}_i$$

where α_i is the attention weight of the query vector i, w is a learnable parameter and $\sqrt{d}$ is a scaling factor. Then element-wise product ($*$) is computed between the global query (Q_{global}) and key vector (k_i) which is represented as:

$$\mathbf{p_i} = \mathbf{Q}_{\mathbf{global}} * \mathbf{k}_i$$

Similar to global query computation, the global key vector is computed as follows:

$$\beta_i = \frac{\exp\left(\mathbf{w}_k^T \mathbf{p}_i/\sqrt{d}\right)}{\sum_{j=1}^{N}\exp\left(\mathbf{w}_k^T \mathbf{p}_j/\sqrt{d}\right)}$$

$$\mathbf{K}_{\mathbf{global}} = \sum_{i=1}^{N} \beta_i \mathbf{p}_i$$

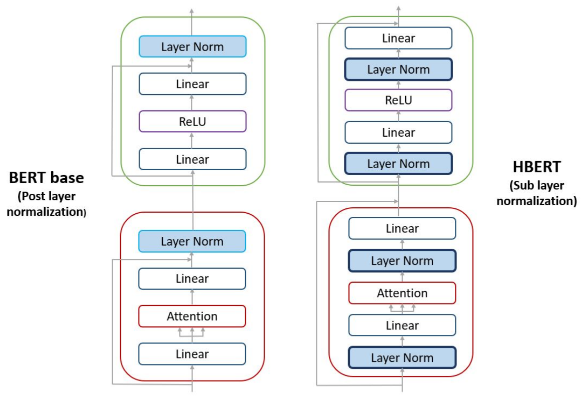


Figure 2: Architectural modifications on HBERT highlighting sub-LayerNorm compared to vanilla BERT.

where β_i is the attention weight of the key vector i, w is a learnable parameter, $\sqrt{d}$ is a scaling factor and K_{global} is the global key vector. Finally, a linear transformation is performed to capture global context-aware attention values which are then added to the attention query vectors.

3.2 HybridBERT (HBERT)

Although additive attention (Wu et al., 2021) can reduce the complexity of the model, it can hurt the performance. To mitigate the drop in performance, we propose a hybrid architecture that combines self-attention (Vaswani et al., 2017) and additive attention (Wu et al., 2021) in a single network with sub-layer normalization. The overall architecture of our proposed hybrid model is similar to the BERT-base (Devlin et al., 2018) architecture. Figure 4 (in the appendix) compares the overall architecture of our proposed hybrid models with BERT-base model. Both our hybrid models have 12 layers and a hidden dimension of 768. In *HybridBERTv1* (HBERTv1), additive attention is used in the odd-numbered layers (1, 3, 5, 7, 9, and 11), and self-attention is used in the even-numbered layers (0, 2, 4, 6, 8, and 10) thus giving equal importance to the additive and self-attention mechanisms. In *HybridBERTv2* (HBERTv2), self-attention is used in the early and later layers (0, 1, 10, and 11) and additive attention in the remaining intermediate blocks. The intuition behind this split is that the earliest and latest layers in BERT encode crucial information (Lin et al., 2019; Kovaleva et al., 2019), indicating that changes in the middle layers might be less critical. In HBERTv2, additive attention dominates in the model architecture with a ratio of 2:1.

We also use sub-layer normalization (Wang et al., 2022) in our HBERT models, where layer normalization is applied four times instead of twice in each encoder layer, as depicted in figure 2. Following the architectural modifications suggested in (Geiping and Goldstein, 2022), we remove bias from the feed-forward network (FFN) layers. Since the model is only pretrained for a limited time, the chances of over-fitting are extremely low. Consequently, we reduce the dropout value to a very small number. Following the optimization technique to accelerate inference (Sun et al., 2020), we use the RELU activation function instead of GELU.

4 Experiments

4.1 Dataset

We pretrain our hybrid models on a Wikipedia dump and the BookCorpus (Zhu et al., 2015), following Devlin et al. (2018), and reserved 5% of the data as the test set. We evaluate all models on two downstream classification tasks. *Massive* (FitzGerald et al., 2022) is a parallel dataset with more than 1 million utterances in 51 languages annotated for Natural Language Understanding tasks. We use only the English subset for our experiments, which has annotations for 60 intents. Models are fine-tuned on 11,514 training samples and validated on 2,033 samples. The *Emotion dataset* (Saravia et al., 2018) consists of annotations of 6 emotion classes: sadness, joy, love, anger, fear, and surprise. It has 16,000 training samples and 2,000 testing samples. Accuracy is used as an evaluation metric for all experiments.

4.2 Implementation details

We train our models on a single Nvidia RTX A6000 GPU. We use the same word-piece tokenizer and follow the same Masked Language Modeling (MLM) training procedure as vanilla BERT. Adam is used for optimization with a learning rate of 1e-4. We use sinusoidal positional embeddings and limit the maximum length of our input to 512. We reduce the dropout value from 0.1 to 0.005 and use a batch size of 48 for pretraining. We set the vocabulary size to 30,522. *Deepspeed zero stage-2* (Rasley et al., 2020) is used for memory off-loading on a single GPU. For fine-tuning, the models are trained for 10 epochs on the Emotion dataset and 15 epochs on the Massive dataset. We use a batch size of 64 for all fine-tuning experiments. We use a BERT-base model and a BERT-base model with additive attention on all layers (henceforth *AddBERT*)

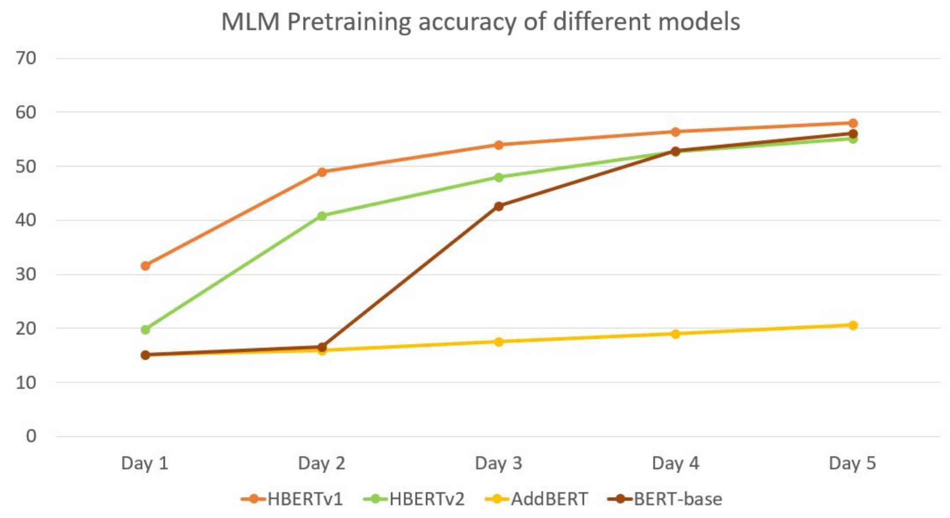


Figure 3: Comparison of masked language modeling accuracy (MLM) after each day of pretraining for different models. X-axis: number of days of pretraining on a single GPU. Y-axis: MLM accuracy.

as baselines.

4.3 Results and Discussion

Figure 3 plots the pretraining accuracy of our HBERT models and both baselines over 5 days. From this chart, we can see that HBERTv1 reaches a pretraining accuracy of 31.59% after just one day of training, compared to 15% for vanilla BERT and AddBERT, which is more than double the accuracy of the baselines. HBERTv2, with a higher fraction of additive attention layers, seems to learn slower and lags behind v1.

Even increasing the training time to two days shows very little improvement in the performance of the baseline models. Our proposed model is more effective for pretraining under a period of 2 days. After 2 days, the BERT model shows a sharp rise in accuracy, and at the end of 5 days, the BERT-base model can provide competing results with our hybrid models. AddBERT shows a minimal increase in accuracy after each training day, showing that the use of additive attention alone seems to be problematic for performance. At the end of day 5, the performance gap between AddBERT and BERT-base is more than 35%.

Table 1 shows the performance of all models on the fine-tuning tasks. Especially after pretraining for one day only, the hybrid architectures outperform vanilla BERT by 1% (Massive) and 2.5% (Emotion). After 2 days of pretraining, vanilla BERT performs almost on par with the hybrid models after fine-tuning. Interestingly, AddBERT performs almost at par with the other models, despite its much worse performance during pretraining. This indicates that a minimal amount of pretraining seems to help the models during the fine-tuning stage, even if pretraining accuracy is low.

Table 2 compares the number of parameters and the inference time of all models. For computing the inference, the model is loaded into a CPU that has 4 cores, and an input sentence with 211 word tokens was used. Our hybrid models have slightly more parameters when compared with the BERT-base model, due to the addition of 2 normalization layers. Their inference time is slightly faster due to the presence of additive attention layers which has been shown to accelerate model inference.

Consequently, AddBERT, employing only additive attention, shows the lowest inference time of 606 ms, which is around 15% lower than the BERT-base model.

4.4 Additional Experiments: Weight Initialization

We additionally experiment with initializing HBERT from a pretrained BERT model, for the self-attention parts of HBERT that are common with the vanilla implementation (i.e. excluding additive attention and normalization layers). 50% of the attention layers in HBERTv1 and 66.67% of the attention layers in HBERTv2 are still randomly initialized. While starting from an already pretrained model may defy the purpose of accelerating pretraining with additive inference, we still believe it is interesting to look at this configuration: If a researcher wants to modify architectural compo-

	Massive		Emotion	
	1 day	**2 day**	**1 day**	**2 day**
BERT-base (full)	**88.59%**		**93.85%**	
BERT-base	84.59%	**86.23%**	86.11%	89.03%
AddBERT	82.49%	83.70%	80.53%	86.70%
HBERTv1	**85.51%**	85.74%	87.80%	88.15%
HBERTv2	85.15%	85.83%	**88.65%**	**89.55%**

Table 1: Performance of all models pretrained for 1 and 2 days on the downstream tasks. The top line presents the performance of a fully pretrained BERT model.

Model	**Parameters**	**Inference time**
HBERTv1	119.8 M	742 ms
HBERTv2	118.7 M	701 ms
BERT-base	**109.5 M**	713 ms
AddBERT	116.4 M	**606 ms**

Table 2: Comparison of model size and inference time of BERT-base with HBERT. Parameters in millions and inference time in milliseconds.

nents of the existing model, they can exploit the weights from unmodified layers in order to improve performance rather than pretraining from scratch.

We find that continual pretraining boosts both pretraining and downstream performance. After just one day of pretraining, HBERT reaches a pretraining accuracy of 40.7% (v1) and 48.95% (v2), outperforming all other models, including vanilla BERT. The trend continues after 2 days of training, where 52.81% (v1) and 51.70% (v2) are reached. For the downstream tasks, we see a similar trend, as can be seen in table 3. Both our hybrid models outperform all other models.

	Massive		Emotion	
	1 day	**2 day**	**1 day**	**2 day**
BERT-base	84.59	86.23	86.11	89.03
HBERTv1$_{init}$	**87.46**	**87.36**	89.75	90.85
HBERTv2$_{init}$	87.01	86.87	**93.05**	**93.10**

Table 3: Performance of HBERT with weight initialization from a pretrained BERT model.

5 Conclusion

In this work, we altered the BERT architecture by combining self-attention and additive attention and employing sub-layer normalization. Our experiments show that on a limited compute budget, our architecture outperforms vanilla BERT both during pretraining and fine-tuning. In the future, we would like to study the effect of knowledge distillation from larger teacher models during fine-tuning. Also, we want to study the effect of hybrid architecture on decoder-based models. Compressing our hybrid models using other model compression approaches is another research direction.

Limitations

All our experiments focus on Natural Language Understanding (NLU) tasks. The effectiveness of our models on generative tasks is a big question. For those models, the pretraining procedure is even more expensive and it is essential to produce coherent, error-free text. So, pretraining for a day or two might have some negative impacts on the model performance. At the moment, our model size is still very huge to be deployed on low-resourced devices. In this work, we did not thoroughly explore model compression. There is a greater scope to use model compression approaches in our hybrid models and we will incorporate them in future work.

Ethics Statement

With this work, we try to improve the access possibilities of people with inferior computing hardware to pretrain large language models. As such, we work towards making the training of large language models more inclusive, allowing researchers with a smaller budget to pretrain their own models. Also, model pretraining is very energy-hungry and hence produces lots of CO_2. We hope to contribute towards making pretraining more green and more environmentally friendly by showing that a limited pretraining budget is often sufficient to arrive at high-performing models.

References

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv* *preprint arXiv:2004.05150*.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, et al. 2022. Massive: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages. *arXiv preprint arXiv:2204.08582*.

Jonas Geiping and Tom Goldstein. 2022. Cramming: Training a language model on a single gpu in one day. *arXiv preprint arXiv:2212.14034*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Peter Izsak, Moshe Berchansky, and Omer Levy. 2021. How to train bert with an academic budget. *arXiv preprint arXiv:2104.07705*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of bert. *arXiv preprint arXiv:1908.08593*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059. Association for Computational Linguistics.

Yongjie Lin, Yi Chern Tan, and Robert Frank. 2019. Open sesame: getting inside bert's linguistic knowledge. *arXiv preprint arXiv:1906.01698*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. 2018. Carer: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 3687–3697.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Hongyu Wang, Shuming Ma, Shaohan Huang, Li Dong, Wenhui Wang, Zhiliang Peng, Yu Wu, Payal Bajaj, Saksham Singhal, Alon Benhaim, et al. 2022. Foundation transformers. *arXiv preprint arXiv:2210.06423*.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

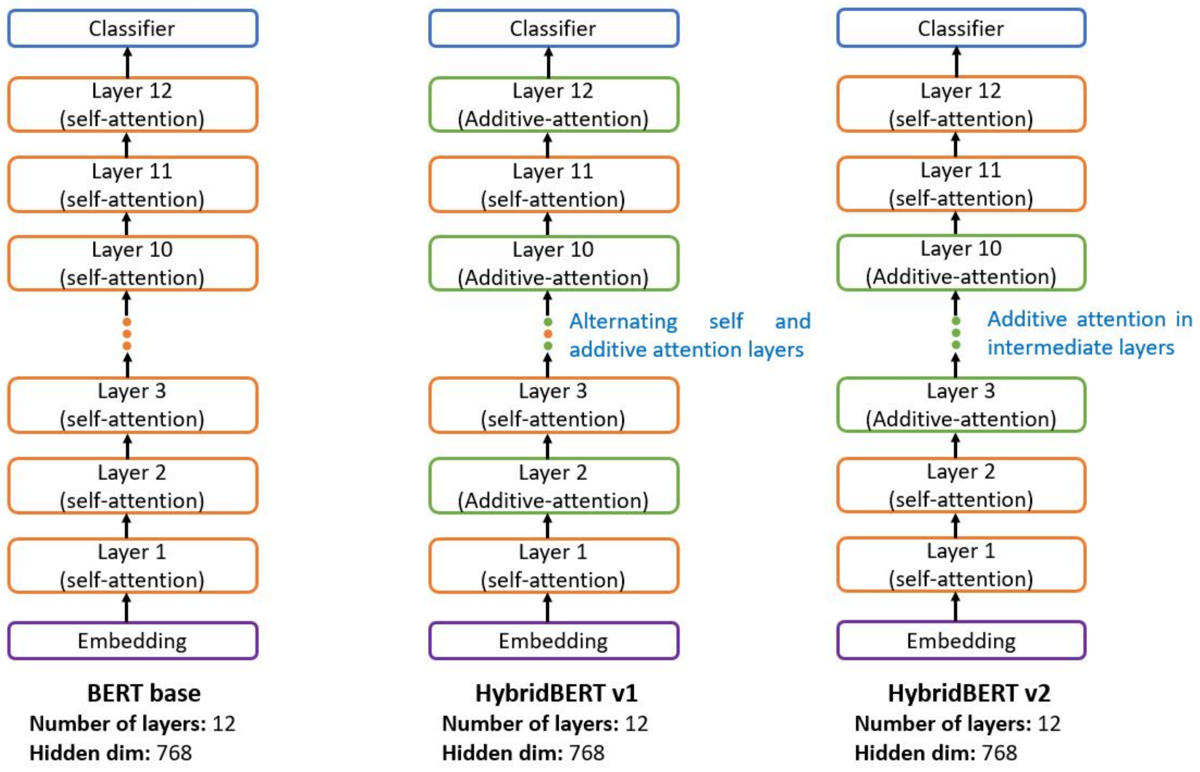
Chuhan Wu, Fangzhao Wu, Tao Qi, Yongfeng Huang, and Xing Xie. 2021. Fastformer: Additive attention can be all you need. *arXiv preprint arXiv:2108.09084*.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. 2021. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14138–14148.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

A Appendix

A.1 HBERT architecture



Classifier
Layer 12 (self-attention)
Layer 11 (self-attention)
Layer 10 (self-attention)
Layer 3 (self-attention)
Layer 2 (self-attention)
Layer 1 (self-attention)
Embedding
BERT base
Number of layers: 12
Hidden dim: 768
Classifier
Layer 12 (Additive-attention)
Layer 11 (self-attention)
Layer 10 (Additive-attention)
Alternating self and additive attention layers
Layer 3 (self-attention)
Layer 2 (Additive-attention)
Layer 1 (self-attention)
Embedding
HybridBERT v1
Number of layers: 12
Hidden dim: 768
Classifier
Layer 12 (self-attention)
Layer 11 (self-attention)
Layer 10 (Additive-attention)
Additive attention in intermediate layers
Layer 3 (Additive-attention)
Layer 2 (self-attention)
Layer 1 (self-attention)
Embedding
HybridBERT v2
Number of layers: 12
Hidden dim: 768

Figure 4: The overall architecture of our HybridBERT model. HybridBERTv1 uses additive attention in the odd-numbered layers and self-attention in even-numbered layers. HybridBERTv2 uses self-attention in the early and later layers and additive attention in all the intermediate layers.