Project Title: Scalable and Sub-Quadratic Sequence Transformation

Group Number: 3

Group Members

Surname, First Name	**Student ID**	**Department**
Liang, Kai	k4liang	Electrical and Computer Engineering
Ngan, Alex	a4ngan	Data Science
Sharma, Aman	a74sharm	Data Science
Zheng, Shu	s232zhen	Electrical and Computer Engineering

Main Contributions

Our main contributions are:

1. Evaluate the efficacy of mixing efficient attention mechanisms (block sparse attention of BigBird and regular linear attention) to handle long-range dependencies

2. Explore the effectiveness of mixing various attention mechanisms

Tools and Technologies Used

The only programming language used was Python. We used a variety of tools to implement our models, including PyTorch, sklean, and keras. Furthermore, we used additional models available through the HuggingFace platform. To run out code, we ran models on both local machines using the NVIDIA CUDA toolkit and Google Colab. To visualize our evaluation results, we used TensorBoard.

Project Implementation Details

Check the boxes that best describe your project:

- ☑ We used an existing implementation of the algorithm(s).
- ☑ We implemented the algorithm(s) ourselves.
- ☑ We used an existing dataset.
- ☐ We created a new dataset.

Abstract

This project explores a hybrid transformer architecture that combines linear attention and sparse attention mechanisms to address the challenge of efficiently modeling long-range dependencies of long-sequence data. Traditional transformer models rely on quadratic self-attention mechanisms, which become computationally prohibitive for long sequences. Our hybrid model alternates between linear attention and Bigbird sparse attention layers, aiming to balance efficiency and performance.

Through experiments on the Long Range Arena (LRA) benchmark, we evaluate the hybrid model on two tasks: ListOps and text classification. Results indicate that the hybrid model achieves a trade-off between training efficiency and performance. These results suggest that combining different attention mechanisms can effectively integrate their strengths, offering a practical solution for long-sequence modeling. However, challenges such as training instability and sensitivity to hyperparameters were observed, likely due to the model's smaller parameter size and lack of large-scale pre-training.

1 Introduction

Transformers have revolutionized natural language processing and other sequence modeling tasks, largely due to their ability to model long-range dependencies. However, the standard self-attention mechanism used in transformers scales quadratically with sequence length, making it computationally expensive for long sequences. This limitation has motivated the development of scalable attention mechanisms that aim to reduce this computational burden while preserving performance.

Linear attention mechanisms, such as those proposed in (Katharopoulos et al., 2020), reduce the quadratic complexity to linear by approximating attention with kernel-based methods. While they offer substantial speedups, these models often underperform on tasks requiring precise modeling of long-range dependencies. On the other hand, sparse attention mechanisms like BigBird (Zaheer et al., 2020) introduce sparsity patterns to reduce complexity while maintaining strong performance, especially on long-range tasks. However, they tend to be slower than linear attention models.

This project explores a middle ground between these two paradigms. We conjecture that it is possible to develop an attention mechanism that achieves better performance (especially for long-range dependencies) than existing linear attention models while maintaining a computational profile that is faster or more efficient than sparse transformers like BigBird. Our objective is to test this hypothesis by implementing and evaluating a hybrid attention mechanism, assessing its scalability, and investigating its viability for deeper architectures and potential pretraining.

The significance of this work lies in its potential to push the boundaries of scalable transformer architectures, enabling the application of deep learning to even longer sequences and more resource-constrained environments without sacrificing performance.

2 Related Work

The inefficiency of standard self-attention in transformers—scaling quadratically with input length—has led to a surge of research into more scalable attention mechanisms. Among these, linear attention and sparse attention have emerged as the two dominant paradigms, each offering a different tradeoff between computational efficiency and model performance.

Linear Attention: Linear attention mechanisms (Katharopoulos et al., 2020; Choromanski et al., 2021) achieve sub-quadratic complexity by approximating the softmax operation with kernel functions or projections that allow reordering the computation. Notable examples include Performer (Choromanski et al., 2021), which introduces kernelized attention with randomized projections, and Linear Transformers (Katharopoulos et al., 2020), which treat attention as a linear mapping with feature map transformations. While these approaches significantly reduce memory usage and runtime, they tend to struggle on tasks that require modeling precise or complex long-range dependencies—especially without large-scale pretraining. As a result, their adoption in high-performance settings remains limited.

Sparse Attention: Sparse attention approaches (Child et al., 2019; Zaheer et al., 2020) tackle the quadratic bottleneck by explicitly zeroing out portions of the attention matrix, retaining only a sparse pattern of connections. BigBird (Zaheer et al., 2020) is a key example that uses a combination of global, local, and random attention to maintain full information flow with provable expressivity. Although sparse transformers can scale to long sequences and often outperform linear attention models, they typically require careful design of sparsity patterns and may be harder to implement and optimize efficiently on modern hardware compared to fully dense models. Their runtime, while sub-quadratic, is still generally slower than linear attention variants in practice.

Hybrid and Modular Attention: Recognizing the limitations of purely sparse or purely linear approaches, recent works have explored hybrid attention mechanisms. The Jamba architecture (Dao et al., 2024), for example, combines state-space models like Mamba (Gu et al., 2024) with transformers to benefit from both efficient sequential processing and global receptive fields. Other modular designs, like LongNet (Ding et al., 2023) and RetNet (Sun et al., 2023), incorporate recurrence or hierarchical decomposition to scale transformers efficiently to millions of tokens. While these methods show promising results, they are often tightly coupled to specific architectural assumptions or require additional components like recurrence or convolution, which may complicate integration with standard transformer pipelines.

Gaps and Project Motivation: Despite these advances, a key research gap remains: no widely adopted attention mechanism achieves both high performance on long-range tasks and efficiency comparable to linear attention. Linear models are fast but weak in accuracy; sparse models are better-performing but computationally heavier. Existing hybrid models either rely on complex architectures or lack generalizability.

Our project aims to bridge this gap by exploring a novel combination of lightweight sparsity and linear approximation, seeking an attention mechanism that:

1. Outperforms current linear attention baselines on long-range benchmarks,
2. Retains sub-quadratic computational complexity, and
3. Is simple enough to be scaled up to deeper models and potentially pretrained.

By focusing on this middle ground, we aim to contribute a practical and performant alternative for scalable sequence modeling.

3 Methodology

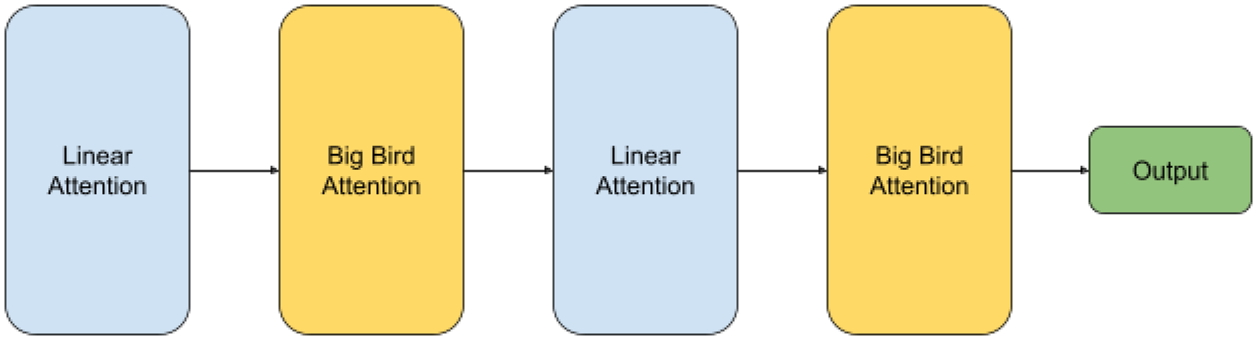

Figure 1: Diagram of the architecture of our proposed hybrid transformer architecture

The central goal of this project is to design a novel hybrid transformer architecture that alternates between Linear Attention and BigBird layers to address the challenge of modeling long-range dependencies in long-sequence data. Traditional self-attention mechanisms in transformers, such as those used in the original Transformer architecture, have a quadratic time and memory complexity of $O(n^2)$, where n is the sequence length. This makes them computationally expensive and often infeasible for tasks involving very long documents or sequences. To mitigate this, several efficient transformer variants have been proposed. Among them, Linear Attention (Katharopoulos et al., 2020) stands out for achieving linear time complexity $O(n)$, by replacing the softmax operation in the attention mechanism with kernel-based feature maps. This approximation allows for a factorized attention computation where the query-key interaction is computed without explicitly forming the full attention matrix. While this significantly improves efficiency and enables scaling to longer sequences, a key drawback of Linear Attention is its reduced ability to model rich token-to-token interactions—especially across distant positions—due to its approximative nature.

To overcome these limitations while maintaining computational efficiency, we propose a hybrid transformer model that incorporates the BigBird attention (Zaheer et al., 2021) mechanism in alternating layers with Linear Attention. Figure 1 illustrates the architecture. BigBird employs a sparse attention strategy that reduces the complexity from O(n^2) to O(√n) in its standard form, or even O(n) under certain configurations, by combining three attention types: global attention (where selected tokens like [CLS] can attend to all other tokens), random attention (to maintain connectivity and theoretical expressiveness), and sliding window attention (which preserves local token interactions). BigBird is theoretically proven to be a universal approximator of sequence functions, like the full attention model, while achieving near-linear runtime and memory performance.

Our hybrid architecture, which alternates between Linear Attention and BigBird layers, inherits the linear complexity of Linear Attention and the sparse, sub-quadratic complexity of BigBird, resulting in an overall sub-quadratic complexity profile for the full model. The alternating structure allows the model to benefit from the efficiency of linear layers and the structural richness and long-range expressiveness of sparse attention, without resorting to dense full-attention computations. This design enables the model to process long sequences more efficiently than standard transformers while maintaining superior performance on tasks that require capturing both local and global dependencies over extended contexts.

To rigorously evaluate our hybrid architecture, we utilize tasks from the Long Range Arena (LRA) benchmark (Tay et al., 2020), which is specifically designed to test transformer models on tasks requiring reasoning over long sequences. From the LRA benchmark, we focus on two representative tasks: ListOps and IMDB classification. The ListOps task is a synthetic benchmark involving nested mathematical operations expressed in prefix notation (e.g., (MAX 2 9 (MIN 4 7))), where the model must correctly parse and evaluate the expression. This task tests the model's ability to handle hierarchical structure and track dependencies across tokens that are not linearly related, which is particularly challenging for models with weak long-range capabilities. The dataset was generated by Tay et al. (2020), after adapting the dataset by (Nangia & Bowman, 2018). The IMDB sentiment classification task involves binary classification of movie reviews (Maas et al., 2011), where the input consists of long-form natural language text. Sentiment cues in this dataset are often distributed across the review, requiring the model to maintain and integrate information from both the beginning and the end of the sequence to make an accurate prediction (Tay et al., 2020).

Both tasks serve as complementary tests: ListOps emphasizes structural reasoning, while IMDB emphasizes semantic integration over long spans. Our implementation leverages PyTorch and the HuggingFace Transformers library (Wolf et al., 2020) for efficient prototyping, with custom modules designed to alternate the attention layers in the transformer encoder stack. The training pipeline includes standard tokenization and preprocessing steps, along with optimized batching strategies to handle long sequences. For optimization, we use the Adam optimizer (Kingma & Ba, 2017) with learning rate warm-up and cosine decay, and performance is measured using accuracy and loss metrics across multiple random seeds to ensure stability. Through this hybrid approach, we hypothesize and aim to empirically demonstrate that the combination of Linear Attention and BigBird can outperform either approach individually, especially in scenarios requiring efficient yet expressive modeling of long-range dependencies.

4 Experiments and Results

We replace the attention layers in the BERT-style model (Devlin et al., 2018) with our hybrid attention implementation. Due to the lack of computation resources, only four encoder layers were implemented in our Hybrid-BERT-Mini model. The first and third layers contained the block sparse attention seen in BigBird models (Zaheer et al., 2021), while the second and fourth layers contained regular linear attention (Katharopoulos et al., 2020). We also evaluated a BERT-mini model but with its standard attention layers replaced by linear attention layers. We refer to this model as Linear-BERT-Mini. Additionally, we evaluated the performance of an adapted BERT-mini model, where the adaptation enables a BERT model to read token sequence lengths greater than 512 (by default, this is the fixed token sequence length) (Devlin et al., 2018). Finally, an augmented BigBird model, called BigBird-Mini, with four block sparse attention layers was evaluated too. Across all models, a 2-layer feedforward network head was present after all the encoder layers to transform the model's general-purpose representations into predictions tailored to downstream tasks. The general hyperparameters that were constant across all the experiments can be found in Table 1.

Table 1: This table contains the general Hyperparameters that remained constant across all models during experimentation. Note: GELU (Gaussian Error Linear Unit) was proposed by Hendrycks and Gimpel (2016).

Hyperparameter	Linear-BERT-Mini	BERT-Mini	BigBird-Mini	Hybrid-BERT-Mini
Number of Encoder Blocks	4	4	4	4
Hidden Dimensionality of Encoder Blocks	256	256	256	256
Number of Attention Heads	4	4	4	4
Hidden Dimensionality of FFN Layer	1024	1024	1024	1024
Activation Function in Encoder Layer	GELU	GELU	GELU	GELU
Dropout Probability in all FC Layers	10%	10%	10%	10%
Dropout Probability in Attention Layers	10%	10%	10%	10%
Block Size in Block Sparse Attention Layer	-	-	64	64
Number of Random Blocks in Block Sparse Attention Layer	-	-	4	4
Feature Map	elu + 1	-	-	elu + 1

We adapted code from Google's LRA implementation (Tay et al., 2020) and a PyTorch translation (Dar, 2024) of these benchmarks to run our LRA experiments. Furthermore, we used HuggingFace's BERT and BigBird models (Wolf et al., 2020) to test their performance against these benchmarks.

Experiment on ListOps Task The first experiment conducted was against the ListOps task of LRA benchmark. The training dataset contains 96000 samples, and the test dataset contains test samples. (Tay et al., 2020). We did not pre-train the model with a large corpus and fine-tune it against the ListOps dataset due to a lack of computational resources. First, we passed the dataset through a custom tokenizer and applied the necessary padding to standardize token sequence length. We trained for 20 epochs (80000 total steps, 4000 steps per epoch, batch size of 24), with early stopping (patience of 3 epochs, accuracy threshold of 0.001). We used Pytorch's cross-entropy loss function to train our models. We used the Adam optimizer (Kingma & Ba, 2017) with its standard settings ($\beta_1 = 0.9$, $\beta_2 = 0.999$), a learning rate of 0.001, and a weight decay of 1×10^{-5}. We deployed a learning scheduler with a warmup of 400 steps. We used accuracy to measure a model's performance against the test dataset. This experiment was conducted on an NVIDIA GeForce RTX 2080 Ti. Table 2 contains the results from the tested models.

Table 2: This table generates the experimental results on ListOps benchmark. It states the number of epochs required for training, the training duration, the training loss, and the final accuracy against the test dataset. The models were trained and evaluated on an NVIDIA GeForce RTX 2080 Ti.

Model	Number of Epochs Before Stop	Training Duration	Training Loss	Test Accuracy
Linear-BERT-Mini (4M)	4	28min	2.287	0.196
BERT-Mini (4M)	4	1h 41min	2.281	0.148
BigBird-Mini (4M)	3	1h 3min	2.255	0.1715
Hybrid-BERT-Mini (4M)	4	1h	2.282	0.1944

Table 2 contains our experimental results against the ListOps benchmark. After a few epochs, most models had comparable training loss, except for BigBird-Mini, which had a marginally better loss. When comparing the training duration, one can observe that the linear-attention model trained much faster compared to its counterparts. Our hybrid-attention model trained comparably as fast as BigBird, while the adapted BERT model trained the slowest. Yet, our model performed near the best alongside Linear-BERT-Mini. This highlights the promise of our attention mixture. Although there were a few epochs, our model's performance was near the top while maintaining a respectable training duration.

Experiment on Text Task The second experiment was against the text task of the LRA benchmark. The dataset consists of 25000 training samples and 25000 test samples (Tay et al., 2020). Because of the lack of computational resources, we could only train our models against the training dataset. We trained for 10 epochs (1954 total steps, about 196 steps per epoch, batch size of 128). Early stopping was deployed with a patience of 3 epochs and an accuracy threshold of 0.01. Pytorch's cross-entropy loss trained our models. The standard Adam optimizer (Kingma & Ba, 2017) with its standard settings ($\beta_1 = 0.9$, $\beta_2 = 0.999$), a learning rate of 0.005, and a weight decay of 1×10^{-5}. A learning scheduler was deployed with a warmup of 20 steps. We used accuracy to measure a model's performance against the test dataset. This experiment was conducted on an NVIDIA A100 Tensor Core GPU.

Table 3: This table contains the results against LRA's text benchmark. It states the required amount of training epochs, the training duration, the training loss, and the accuracy against the test set. The models were trained and evaluated on an NVIDIA A100.

Model	Number of Epochs Before Stop	Training Duration	Training Loss	Test Accuracy

Linear-BERT-Mini (6M)	4	2h 14min	0.6930	0.50321
BERT-Mini (6M)	4	2h 19min	0.6928	0.50401
BigBird-Mini (6M)	4	2h 42min	0.6930	0.51987
Hybrid-BERT-Mini (6M)	4	2h 24min	0.6928	0.51106

From Table 3, all models achieved very similar training losses of around 0.6930. However, the training duration varied. One can obtain that our hybrid attention implementation trained marginally slower than Linear-BERT-Mini and BERT-Mini, but faster than BigBird-Mini. Due to the addition of block sparse attention layers, it was expected that our model would perform slower than its linear-only counterpart, but still much faster than the BigBird counterpart due to the presence of linear attention layers. However, our implementation trained slower than BERT-Mini, which contradicts the theory that the quadratic complexity of BERT attention would blow up the runtime. This may have occurred due to the memory access overhead present in sparse attention. However, concerning test accuracy, our hybrid model performed better than Linear-BERT-Mini and BERT-Mini, while performing marginally worse than its BigBird counterpart. Overall, these results affirm that our hybrid attention mechanism provides a nice balance of effectiveness between linear attention and block sparse attention of BigBird. Our implementation trained a bit slower than its linear counterpart but outperformed in test accuracy. Compared to BigBird, our model trained much faster while achieving marginal accuracy degradation.

5 Discussion and Conclusion

Through comparative experiments conducted on the LRA benchmark, we observed that the linear attention mechanism achieved the best performance and the fastest training speed in the ListOps task. Our hybrid model exhibited marginally lower training efficiency and accuracy compared to the linear model. Similarly, in the text classification task, the hybrid model demonstrated performance comparable to BigBird while with accelerated training speeds. These results indicate that the architectures integrating diverse attention encoder mechanisms can effectively combine the strengths of both approaches, thereby achieving relatively stable performance across different tasks and improved training efficiency.

Limitations and Future Work During the training process, we had several challenges, including minimal reductions and fluctuation in training loss, rapid fitting, and convergence results of models that were sensitive to the learning rate. These issues are likely attributed to the relatively small model size and lack of large-scale pre-training. Furthermore, due to computational constraints, we could not pre-train our model against a large corpus, and then pre-train against our various benchmarks. Finally, we could not scale our models with a greater number of parameters and test these models against a greater plethora of benchmarks for a more conclusive outcome.

Yet, our results are very promising. Looking into the future, one can scale our implementation into larger transformer models. For instance, one can explore the efficacy of our linear and block sparse attention mixture in base-size or large-size transformer models, especially with proper pre-training on a large corpus. Furthermore, the exploration of mixing various efficient attention mechanisms offers a new direction to obtain high-performing transformer models. Our model mixed both linear attention and block-sparse attention of BigBird. However, other efficient attention mechanisms may be more effective. Furthermore, one could also introduce gating or routing mechanisms to dictate the best attention mechanisms for each attention layer. For example, a transformer model can learn that for language modeling tasks, every other attention layer should be a block-sparse attention layer, while for next-sentence prediction, Mamba may be more suitable. These avenues open new exciting opportunities to explore the power of efficient attention mechanisms and improve their adoption in state-of-the-art transformer models.

References

1. Katharopoulos, A., Vyas, A., Pappas, N., & Fleuret, F. (2020). Transformers are RNNs: Fast autoregressive transformers with linear attention. *arXiv preprint arXiv:2006.16236*.
2. Zaheer, M., Guruganesh, G., Dubey, A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., & Ahmed, A. (2021). Big Bird: Transformers for longer sequences. *arXiv preprint arXiv:2007.14062*.
3. Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., ... & Weller, A. (2021). Rethinking attention with performers. *International Conference on Learning Representations (ICLR)*.
4. Child, R., Gray, S., Radford, A., & Sutskever, I. (2019). Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.
5. Dao, T., Fu, Y., Li, C., Zhang, H., Rudra, A., Recht, B., ... & Ma, T. (2024). Jamba: Hybrid state-space model and transformer architecture. *arXiv preprint arXiv:2402.09234*.
6. Gu, A., Dao, T., Ermon, S., Ré, C., & Rudra, A. (2024). Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.
7. Ding, M., Dai, Z., Yang, Z., Li, S., Zhou, D., Liu, Y., ... & Tang, J. (2023). LongNet: Scaling Transformers to 1B tokens. *arXiv preprint arXiv:2307.02486*.
8. Sun, Z., Wang, Y., Liu, S., Dong, L., & He, D. (2023). Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*.
9. Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., & Metzler, D. (2020). Long Range Arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*.
10. Nangia, N., & Bowman, S. R. (2018). ListOps: A diagnostic dataset for latent tree learning. *arXiv preprint arXiv:1804.06028*.
11. Maas, A. L., Daly R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C.. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. http://www.aclweb.org/anthology/P11-1015.
12. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., & Rush, A. M. (2020). *Transformers: State-of-the-art natural language processing*. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (pp. 38–45). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-demos.6.
13. Kingma, D. P., & Ba, J. (2017). Adam: A method for stochastic optimization. *arXiv preprint arxiv:1412.6980*.

14. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
15. Hendrycks, D., & Gimpel, K. (2016). Bridging nonlinearities and stochastic regularizers with Gaussian Error Linear Units. *arXiv preprint arxiv:1606.08415.*
16. Dar, G. (2024). lra-benchmarks [Source code]. GitHub. https://github.com/guy-dar/lra-benchmarks.

Appendix

Ablation Study To help us better understand and analyze the importance of different parts of the model, we experimentally swapped the order of encoders in the model.

Table 4: This table generates the experimental results on ListOps benchmark. It states the number of epochs required for training, the training duration, the training loss, and the final accuracy against the test dataset.

Model	**Number of Epochs Before Stop**	**Training Duration**	**Training Loss**	**Test Accuracy**
Linear-BERT-Mini (4M)	4	28min	2.287	0.196
BERT-Mini (4M)	4	1h 41min	2.281	0.148
BigBird-Mini (4M)	3	1h 3min	2.255	0.1715
Hybrid-BERT-Mini (4M)	4	1h	2.282	0.1944
Hybrid-BERT-Mini-Swapped (4M)	7	2h 2min	2.295	0.1944

We can find that the modified model (with swapped BigBird and linear attention layers) requires extended training duration to achieve convergence, yet the final accuracy is the same as the original architecture. This empirical result substantiates the soundness of our hybrid architecture.