Hybrid Model Architecture for Language Tasks

Sadman Sakib **Riyad Hassen** **Fahad Touseef**

1 Introduction

Transformer models have been the state-of-the-art in natural language processing (NLP) tasks for the past few years. However, they have limitations in handling long-range dependencies due to the quadratic increase in computation with sequence length. State-space models (SSM) have been proposed as an alternative to transformer models, which can store the history of a sequence in its states and outperform transformer models in tasks involving long-range dependencies (Gu et al., 2021). Recently proposed hybrid architectures have combined the strengths models from different architecture families and enjoyed best-of-all-worlds performance. Hybrid model design is challenging as it requires expert-driven experimentation and training from scratch.

As part of the course project, we have reimplemented the Manticore architecture (Roberts et al., 2024) that automates the search for hybrid model architectures and can even combine pretrained models to create pretrained hybrids. Manticore splits component models into a sequence of blocks and incorporates projectors and combiners between blocks to merge models in an architecture-independent way. Our experiments show that Manticore hybrid outperforms some exisiting manually-designed hybrids and improves upon the component transformer and SSM models.

2 Literature Survey

State-space models (SSM) contain blocks with hidden states that opearate similar to recurrence-based models. The recurrence operation makes it faster compared to transformer-based models for which number of computations increase quadratically with sequence length. By carefully choosing the parameters, SSM can effectively store the history of a sequence in its states and has to shown outperform transformer-based models in tasks involving long-range dependencies (Gu et al., 2021). Mamba model makes the parameters of SSM input-dependent to selectively store information from the input and employs a hardware-aware parallel-scan algorithm to efficiently compute recurrent operations (Gu and Dao, 2024). While Mamba matches or outperforms transformer-based models in zero-shot setting, attention mechanism in transformers provides superior performance in few-shot language tasks compared to Mamba.

A recent study has evaluated the performance of transformer and Mamba models on various in context-learning tasks (Park et al., 2024). Their study shows that Mamba outperforms previous SSM models (e.g. S4) and can perform on-par with transformers on simple ICL tasks. Mamba even performs better than transformer-based models on some ICL tasks. However, Mamba models struggle on certain ICL tasks such as retrieval. They developed a hybrid architecture by replacing MLP blocks in a transformer with Mamba blocks and placing a Mamba block at the beginning. MambaFormer was found to consistently perform well on the studied ICL tasks in some of which transformers and Mamba models individually struggled.

There has been a growing interest in hybrid models that combine different computational blocks such as linear, convolution, recurrence and attention. Mechanistic Architecture Design (MAD) (Poli et al., 2024) framework suggests that some small-scale synthetic tasks can prove sufficient to estimate the performance of such hybrid models when they scale up. The token-manipulation based synthetic tasks include selective copying, compression, memorization and different versions of recall (in-context, fuzzy in-context, and noisy in-context). The study forms different hybrid architectures by placing computational primitives in sequential, striped and sparse parallel manner.

Merging pretrained models has recently gained attention as a way to combine capabilities of dif-

ferent models. Merging model from same architecture has been studied to make models skilled across diverse tasks. Model Breadcrumbs (Davari and Belilovsky, 2024) takes the weight difference before and after fine-tuning on a downstream task, and merges them with proper sparsification to simultaneously improve performance across multiple tasks. This has also motivated merging models with different architectures with large-scale evolutionary search (Akiba et al., 2024). However, this method is computationally expensive and requires many training runs. In comparison, Manticore does not require training a large number of models.

3 Proposed Method

We will implement the Manticore framework using pretrained models and different training mechanisms as described in the Manticore paper (Roberts et al., 2024).

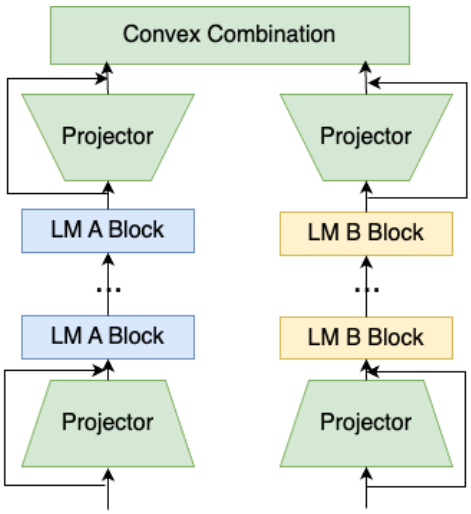

Figure 1: A Manticore block

3.1 Structure

Manticore Framework consists of three main parts: individual LMs to mix, projectors to translate feature representations between participating LMs and convex combination mixture weights that specify how projected features are combined.

3.1.1 Component Model

Any modern LM, including transformer-based LMs, recurrent models, and state-space models, can be used as a component model in Manticore according to (Roberts et al., 2024).

3.1.2 Projector

Suppose, the two pretrained component models are M and M'. The goal of the projectors is to enable the blocks of M and M' to speak a common language, such that the features are compatible and can be reused in the resulting hybrid model. The projectors are simple linear transformations with a gated residual connection.

If Manticore hybrid is made from K different pretrained models, denoted $M_{(1)}, M_{(2)}, ..., M_{(K)}$, with model dimensions $d_{(M_1)}, d_{(M_2)}, ..., d_{(M_K)}$, then define $d_{max} = max_{k=1}^{K} d_{(M_k)}$. The input and output projectors are then,

$ProjIn_{(k)}(x, \alpha) = (1 - \alpha) \cdot Linear_{d_{max} \to d_{M_{(k)}}}(x) + \alpha \cdot Trunc(x, d_{M_{(k)}})$

$ProjOut_{(k)}(x, \alpha) = (1 - \alpha) \cdot Linear_{d_{M_{(k)}} \to d_{max}}(x) + \alpha \cdot Pad(x, d_{max})$

where $Trunc(x, d)$ and $Pad(x, d)$ truncate and zero-pad input to dimension d. $Linear_{d \to d'}$ is a learnable linear transformation. Gating weights are parameterized as $\alpha \in [0, 1]$.

3.1.3 Mixture Weights

Next, the activations of different component models have to be combined. This is performed by taking a convex combination of the projector's outputs $h_k(x)$,

$Mix_\alpha = \sum_{k=1}^{K} \alpha_k h_k(x)$

The convex combination weights are reused as gating weights in the projectors. This provides the convenient property that when $\alpha = 1$, for one component, and 0 for all other components, the projector outputs are ignored and only that component's output is taken as the hybrid block's output. The α parameters are computed with a softmax on the trainable parameters. That is, $\alpha_k = \frac{exp(a_k)}{\sum_{j \in [K]} exp(a_j}$ for all $k \in [K]$.

3.2 Training Mechanisms

Manticore paper describes three training methods:

Training Hybrids from Scratch. This is performed by treating mixture weights as learnable parameters along with the rest of the architecture parameters.

Fine-tuning Pretrained Hybrids. First, the LM heads and embeddings of pretrained models are replaced with initialized LM heads and embedding layers. Then, the projectors are pretrained on a small amount of general language data while keeping the original component model weights frozen. To fine-tune on downstream task data, first mixture

weights are searched by training all parameters, then the mixture weights are fixed, and the component models and projectors are fine-tuned.

4 Dataset

4.1 ELI5

The ELI5 dataset is a large-scale corpus of questions and answers for long-form question answering. The ELI5 dataset is made up of 270,000 threads from the Reddit forum "Explain Like I'm Five" (ELI5). The forum is an online community where users answer questions in a way that's understandable for five-year-olds. The dataset includes a variety of questions that require multi-sentence answers. We used this dataset to test the hybrid model on casual inference.

4.2 IMDB

IMDB dataset has 50K movie reviews for natural language processing or text analytics. This is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets. We used this dataset to perform classification analysis.

4.3 MAD

The MAD framework (Poli et al., 2024) consists of tasks that test the basic capabilities of different language model architectures in a small scale. The tasks are-

- **Selective Copying**: The model has to selectively memorize and ignore information from the input.
- **In-context Recall**: The input comprises a sequence of key-value pairs, and the model is tasked to predict all values for the keys present in the sequence.
- **Noisy In-context Recall**: The noisy in-context recall task is a variation of in-context recall, in which noise tokens are randomly inserted into the input sequence.
- **Fuzzy In-context Recall**: The fuzzy in-context recall tasks adapt the in-context recall task by representing keys and values by a variable number of adjacent tokens.
- **Memorization**: The memorization task uses a fixed key-value dictionary, representing the facts to be learned.

4.4 SQUAD

Stanford Question Answering Dataset (SQUAD) contains 150,000 data points, where each contains a context, a question from the context and its answer. SQUAD v2 additionally contains 50,000 data points with adversarial questions that cannot be answered from the context.

5 Implementation

Since the paper did not provide a code base, we implemented a modular framework which can be used to combine models to create hybrids. It consists of classes such as Combiners, Splitters and HybridModel. The Github repository can be accessed at `https://github.com/sadmankiba/hybrid-model`. The instructor has been added as a collaborator to the repository.

In our experiments, we used a decoder-only transformer - *'EleutherAI/gpt-neo-125M'* - and a mamba model - *'state-spaces/mamba-130m-hf'* - from HuggingFace to build our hybrid model. We ran experiments similar to the ones described in the Manticore paper(Roberts et al., 2024). Our hybrid model trained on ELI5 dataset can be found at `https://huggingface.co/fahad-touseef/manticore-hybrid-gptneo-mamba`.

5.1 Fine-tuning Pretrained Hybrids

The main study compares performance of pretrained Pythia-410M, Mamba-370M, and Manticore hybrid built from these models on natural language fine-tuning datasets such as Imdb, Alpaca (Taori et al., 2023) and fine-web dataset. In our experiments, we used pre-trained GPT-Neo-125M and Mamba-130M models. We used these smaller models because we were short of GPU resource. The main study used >= 16GB GPU memory whereas we're limited by 12GB Nividia RTX 2080 GPUs. Following the Manticore paper, we first trained the projectors on ELI5 dataset keeping the language model weights frozen.

6 Experiments and Results

We followed the settings presented in the paper (Roberts et al., 2024). Following that we trained hybrids from scratch and fine-tune them using pretrained models.

For training configuration we used AdamW(Loshchilov and Hutter, 2017) optimizer with linear learning rate scheduler, setting

initial learning rate of $5e-5$ and weight decay 0.01 with batch size 16.

6.1 Training on IMDB dataset from scratch

We trained small randomly-initialized models from scratch on the IMDB dataset. These consisted of a transformer (4 layers), a mamba (4 layers) and a hybrid between these (1 hybrid block only). The results are shown in Table 1.

Model	Transformer	Mamba	Hybrid
Accuracy	58.7	89.5	89.7

Table 1: Training on Imdb dataset from scratch with 4 blocks of transformer, mamba and hybrid model

The transformer did not perform well, but the mamba did. The hybrid was able to leverage this and also perform well. This training was done for 3 epochs. It seems that the transformer required more training.

6.2 Training on ELI5

Next, we trained a hybrid model on the ELI5 dataset using pretrained backbones i.e. the 125M GPT-Neo and 130M Mamba. The backbones were frozen and only the projectors, embeddings and language modeling head were learned. We trained for 3 epochs and got a test loss of 0.927. This model was pushed to Hugging Face Hub. We use this to get initialized projectors for section experiment 6.3. Table 2 presents mixture weight result after training with ELI5.

α value	Combiner	Splitter
Block1	0.55	0.07
Block2	0.015	0.73

Table 2: Mixture weight α for 2 block hybrid architecture after 3 epoch

In our implementation, we pass the transformer block output in the first parameters of the combiner and splitters, and mamba block output in the second parameters. The mixture weights above indicate that at combination, the first block of the hybrid architecture considers both model output equally with slight preference given towards the Transformer model. In the combination of the second block, the hybrid block output was almost entirely coming from Mamba block output. The splitter weights in these two blocks are scaling the input towards the preferred distribution of the component blocks.

6.3 Fine-tuning on IMDB dataset

For fine-tuning pretrained hybrids we replace the LMheads and embedding with randomly initalized LMhead and embedding layer.Then we pre-train projector on smaller dataset based of 6.2 while replacing the backbones frozen. We then loaded the above model from the Hugging Face Hub and fine-tuned it for 3 epochs on the IMDB dataset. We also separately finetuned the pretrained 125M GPT-Neo and 130M Mamba models on the IMDB dataset for 3 epochs for comparison. The results are shown in Table 3.

Model	Transformer	Mamba	Hybrid
Accuracy	84.1	80.2	61.1

Table 3: Finetuning on Imdb dataset with pretrained transformer(GPTNeo 125), mamba(Mamba 130M) and Hybrid model

Both the transformer and the mamba performed well, but the hybrid did not. We suspect that the projectors need to be trained on more data (e.g. fine-web, as in the paper) or for longer epochs till it finds a better mixture weight and embedding layer for the hybrid to perform well on downstream tasks.

Table 4 presents the mixture weight after training the data with imdb dataset.

α value	Combiner	Splitter
Block1	0.49	0.07
Block2	0.017	0.72

Table 4: Mixture weight α for 2 block hybrid architecture after 3 epoch

The mixture weights stayed mostly stable and are updated slightly during training on the IMDB dataset. Similar to pre-training, the first block considers both model output equally and the second block gives far more weight to the Mamba block output. Scaling performed by the splitters are also similar to pre-training.

6.4 Training on MAD Tasks

The Manticore study compares the Manticore hybrid with other hybrid models (e.g. MambaFormer)

Table 5: Configuration for MAD tests

MAD parameter	value	Training parameter	value
Number of layer	2	Epochs	20
Vocab size	16	Batch size	32
Sequence length	32	Learning rate	5e-4
Train examples	4096		
Test examples	256		
Hidden size	64		
Number of transformer heads	4		
Noise fraction	0.2		

and with component models on synthetic tasks from MAD framework.

To test on MAD framework, we also implemented the Mambaformer architecture from same component models and compared the performance of hybrid models and component unit models. Table 5 shows the training configuration for the MAD tests we conducted.

Table 6 shows the exact-match accuracy of the hybrid and component models. We made two variations of the Manticore model containing 1 and 2 hybrid blocks. The results confirm that the Manticore hybrid model performs at least as well as its component models. In the fuzzy in-context recall task, both variations of the Manticore hybrid model outperform the component models and Mambaformer, thanks to their flexibility in selecting convex combination parameters.

6.5 Fine-tuning on SQUAD Dataset

We experimented with fine-tuning hybrid model and pretrained component models on SQUAD dataset. We considered the answering task as a text generation problem. We concatenated the context and question field and the model was tasked to generate the answer. We measured three metrics-exact token match, BLEU and ROUGE-L. The performance of the models were not as expected. We hyphothesize the following methods could improve the performance. Due to time constraint, we were not able to retrain the models with these techniques.

1. Fine-tuning only a handful of layers near the end or a new LM head instead of full-parameter training
2. During training, calculating loss only on answer tokens, and not on context and question tokens.

7 Conclusion

The main goal of this project was to experiment with the proposed framework from the paper Manticore hybrid architecture (Roberts et al., 2024). These hybrid blocks are selected from LMs without exploring the entire search space such as NAS (Thomas Elsken, 2019). Results on MAD tasks and IMDB classification dataset establish that hybrid models trained from scratch can learn to utlize the best-performing component model and combine the strengths to provide better performance. Pretraining hybrid model on ELI5 dataset showed that the combination and split parameters are capable of utlizing both or a single component model.

From the results we got on pretraining and training the hybrid from scratch we have found same conclusive result presented on the main paper. Hybrid trained from scratch (table 1) outperforms models on fine-tuned tasks (table 3).

Our experiments showed that performance of pretrained hybrids are heavily dependent on pretraining of the projectors and embedding layers. If the community agrees to using a universal tokenizer and embeddings, learning new hybrids using the Manticore framework wouldn't require learning a new common embedding matrix greatly reducing the pretraining cost for Manticore hybrid models.

In conclusion, architecting hybrid models to leverage each component and get the best out of them has the potential to develop LLMS. As noted in the main paper this framework has a limitation in that NAS was not always able to recover the best architecture in the search space. Some of the results are promising however there needs to be in-depth research and exploration to find the best technique for making hybrid models.

8 Contributions

All members contributed roughly equal time and effort to the project. Below we are describing each member's contribution in detail.

Sadman worked on implementation of the Hybrid Model. He modularized the implementation of Combiner, Splitter and HybridModel classes. He also trained the models on MAD tasks and SQUAD dataset

Table 6: Best Test Accuracy on MAD Tasks

Task	Transformer	Mamba	Manticore 1 block	Manticore 2 block	MambaFormer
Selective Copying	0.91	1.0	0.99	0.99	0.98
In-context Recall	0.35	1.0	1.0	1.0	0.99
Noisy In-context Recall	0.42	1.0	1.0	0.99	1.0
Fuzzy In-context Recall	0.35	0.36	0.42	0.43	0.15
Memorization	1.0	1.0	1.0	1.0	1.0

Fahad implemented causal inference with the Hybrid Model. He worked training initialized models on IMDB. He also trained the projectors in the hybrid model on the ELI5 dataset and pushed the hybrid model to Hugging Face Hub for downstream usage. Fahad also fine-tuned the models on IMDB classification with pre-trained backbones. He modularized the Trainer class for general usage.

Riyad collaborated on implementation of the Hybrid Model. He ran experiments on training initialized models on IMDB classfication. He also worked on training the projectors in the hybrid model on the ELI5 dataset. Moreover, he fine-tuned the models on IMDB dataset with pre-trained backbones in differnt configurations. He also collaborated with Fahad to modularize the Trainer class.

References

Takuya Akiba, Makoto Shing, Yujin Tang, Qi Sun, and David Ha. 2024. Evolutionary optimization of model merging recipes.

MohammadReza Davari and Eugene Belilovsky. 2024. Model breadcrumbs: Scaling multi-task model merging with sparse masks.

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces.

Albert Gu, Karan Goel, and Christopher Ré. 2021. Efficiently modeling long sequences with structured state spaces. *CoRR*, abs/2111.00396.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Jongho Park, Jaeseung Park, Zheyang Xiong, Nayoung Lee, Jaewoong Cho, Samet Oymak, Kangwook Lee, and Dimitris Papailiopoulos. 2024. Can mamba learn how to learn? a comparative study on in-context learning tasks.

Michael Poli, Armin W Thomas, Eric Nguyen, Pragaash Ponnusamy, Björn Deiseroth, Kristian Kersting, Taiji Suzuki, Brian Hie, Stefano Ermon, Christopher Ré, Ce Zhang, and Stefano Massaroli. 2024. Mechanistic design and scaling of hybrid architectures.

Nicholas Roberts, Samuel Guo, Zhiqi Gao, Satya Sai Srinath Namburi GNVV, Sonia Cromp, Chengjun Wu, Chengyu Duan, and Frederic Sala. 2024. Pretrained hybrids with mad skills.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Frank Hutter Thomas Elsken, Jan Hendrik Metzen. 2019. Neural architecture search: A survey. *arXiv preprint arXiv:1808.05377*.