BRAINFORMERS: TRADING SIMPLICITY FOR EFFICIENCY

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformers are central to recent successes in natural language processing and computer vision. Transformers have a mostly uniform backbone where layers alternate between feed-forward and self-attention in order to build a deep network. Here we investigate this design choice and find that more complex blocks that have different permutations of feed-forward layers, self-attention layers, and gated layers (for routing through sparse structures) can be more efficient. Using this insight, we develop a complex block, named Brainformer, that consists of a diverse sets of layers such as sparsely gated feed-forward layers with different gating mechanisms, dense feed-forward layers, attention layers, and various forms of layer normalization and activation functions. Brainformer consistently outperforms the state-of-the-art dense and sparse Transformers, in terms of both quality and efficiency. A Brainformer model with 8 billion activated parameters per token demonstrates 2$\times$ faster training convergence and 5$\times$ faster step time compared to its GLaM counterpart. In downstream task evaluation, Brainformer also demonstrates a 3% higher SuperGLUE score with fine-tuning compared to GLaM with a similar number of activated parameters. Finally, Brainformer largely outperforms a Primer dense model with similar computation per token on oneshot evaluation for three important generative tasks.

1 INTRODUCTION

In recent years, large neural networks derived from from the Transformer architecture (Vaswani et al., 2017) have demonstrated superior results on language understanding and generative tasks. Many improvements on Transformer variants have come from scaling the size of models (Raffel et al., 2020; Brown et al., 2020a; Shoeybi et al., 2019; Chowdhery et al., 2022), scaling the training tokens (Hoffmann et al., 2022; Shoeybi et al., 2019), better training data quality (Du et al., 2022), and sparsely activated model architectures (Du et al., 2022; Lepikhin et al., 2021; Roller et al., 2021; Lewis et al., 2021).

Among the efficient transformer language model techniques (Wang et al., 2020; Choromanski et al., 2020; Tay et al., 2021; Hua et al., 2022), there is a focus on improving attention-layer efficiency using low-rank approaches or approximations. However, recent work has also identified that dense feed-forward layers constitute most of the computational cost for common sequence lengths ($\leq$2048), particularly when the model is large (Du et al., 2022; Zhou et al., 2022b). To further improve compute efficiency such as total FLOPs used during training to reach convergence, sparsely gated Mixture-of-Experts (Lepikhin et al., 2021; Fedus et al., 2021; Du et al., 2022; Zhou et al., 2022b; Roller et al., 2021; Lewis et al., 2021; Jaszczur et al., 2021) have become prevalent, giving the model a larger overall capacity to improve quality while holding computational cost fixed. Sparsely activated models not only reduce the computational cost, but also have better specialization by training different experts on different data distributions through the use of learned routing functions without reducing the effective training time for each expert. The MoE architectures in this line of work are based on uniform transformer blocks or manually interleaving dense and sparse layers (Du et al., 2022).

Various forms of neural architecture search (So et al., 2021; Su et al., 2021; Zhao et al., 2021; Zhou et al., 2022a) have been devised targeting different working scenarios. However, few of the existing methods have been extended to the large model regime, particularly models with billions of parameters and sparsely activated layers. We find that in a traditional, non-sparse architecture search,

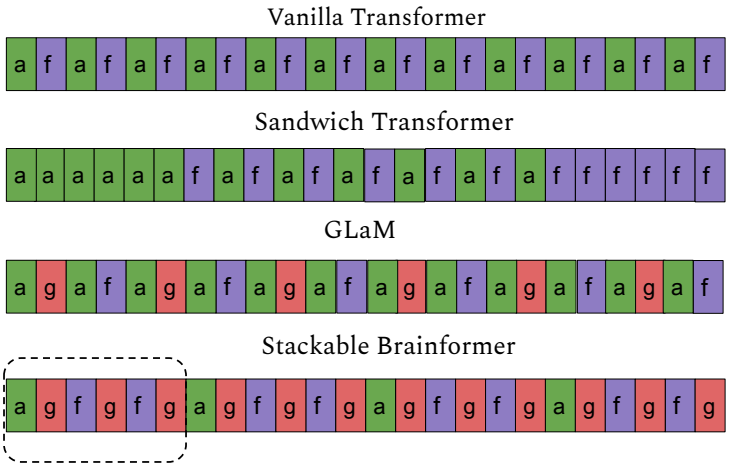


Figure 1: High-level Comparison with Related Work. 'a': attention, 'f': feed-forward, 'g': sparsely gated feed-forward. GLaM interleaves dense transformer blocks with sparse transformer blocks. Brainformer reduces the frequency of attention and changes layer widths together with layer types.

the best architecture looks very similar to the vanilla transformer model (e.g. as demonstrated in the Primer work (So et al., 2021)). However, if we expand the search space to include model families with sparsely-activated layers, the optimal model architecture can look very different from the vanilla transformer.

Resonating with the layer-wise architecture scaling in EfficientNet (Tan & Le, 2019) and layer reordering in the sandwich transformer (Press et al., 2019), we think a non-uniform architecture can be more efficient than a uniform architecture such as the vanilla transformer architecture. In our proposed non-uniform architecture, there is no strict layer interleaving as in the vanilla transformer in fig. 1. Instead, we trade off architecture regularity by allowing the search space to compose different sub-layers in different orders, discovering architectures that are more efficient by arbitrarily choosing the order of sub-layers from the search space. In addition to the attention layer and dense feed-forward layer in a transformer model, we introduce a sparsely gated feed-forward layer (MoE layer) with different types of gating mechanisms, various layer normalizations, and activations in the search space. We only treat the MoE layer as a general method to sparsify the model. However, in practice, any conditional computation method can be blended in. We apply a simple evolutionary search to discover many attributes, such as the best way to interleave layers and layer capacities, when to fuse layers, and when to specialize layers with MoE modules. For ease of scaling, we propose a block-wise sub-layer grouping, such that stacking a variable number of blocks produces models of different scales, as illustrated in Stackable Brainformer in fig. 1. As our results in Section 5 show, this approach has proven effective in our evaluation.

2 Related Work

Large Language Models: Language models have demonstrated strong performance for many natural language processing tasks (Mikolov et al., 2010; Sutskever et al., 2011; Dai & Le, 2015). Scaling up model capacity and number of training tokens has shown huge success in enhancing the performance of computer vision architectures (He et al., 2016a;b; Ghiasi et al., 2019; Dai et al., 2021) as well as neural language models (Radford et al., 2018; Brown et al., 2020b; Kaplan et al., 2020; Raffel et al., 2020; Shoeybi et al., 2019; Hoffmann et al., 2022).

Sparsely Activated Models: Conditional computation effectively increases the capacity of a deep neural network without increasing the total amount of computation, by activating certain parameters and computation on demand, based off the input token or sequence (Cho & Bengio, 2014; Puigcerver et al., 2020; Lin et al., 2019). The gating decisions may be binary or sparse and continuous, stochastic or deterministic. In a multi-device setting, sparsely-gated MoE (Shazeer et al., 2017) demonstrates

massive improvements in model capacity, training time, or model quality with gating. Various MoE architectures including Switch Transformer (Fedus et al., 2021) and GLaM (Du et al., 2022) have been proposed. They adopt a token-based gating where an auxiliary loss is imposed to counter load imbalance issues. Recently, more advanced gating functions are devised to ameliorate load imbalance, improve speed, and downstream generalization (Roller et al., 2021; Dua et al., 2021; Zuo et al., 2021; Gross et al., 2017; Zhou et al., 2022b; Jaszczur et al., 2021).

Non-uniform Architectures: EfficientNet represents one of the very early non-uniform architectures that leverages layer heterogeneity to achieve SoTA. Instead of searching for a new operator or a new block of operators, EfficientNet focuses on optimizing the layer compound coefficients to scale the model effectively. This heterogeneity leads to a model more than $8\times$ smaller and more than $6\times$ faster on inference (Tan & Le, 2019). Sandwich Transformer, on the other hand, promotes a non-interleaved, non-uniform architecture for language modeling tasks. However, the sandwich reordering pattern does not guarantee performance gains across every task. In this work, we take inspiration from the earlier work but further improve scaling and generalization.

3 METHOD

3.1 DERIVING OUR MODEL COMPONENTS

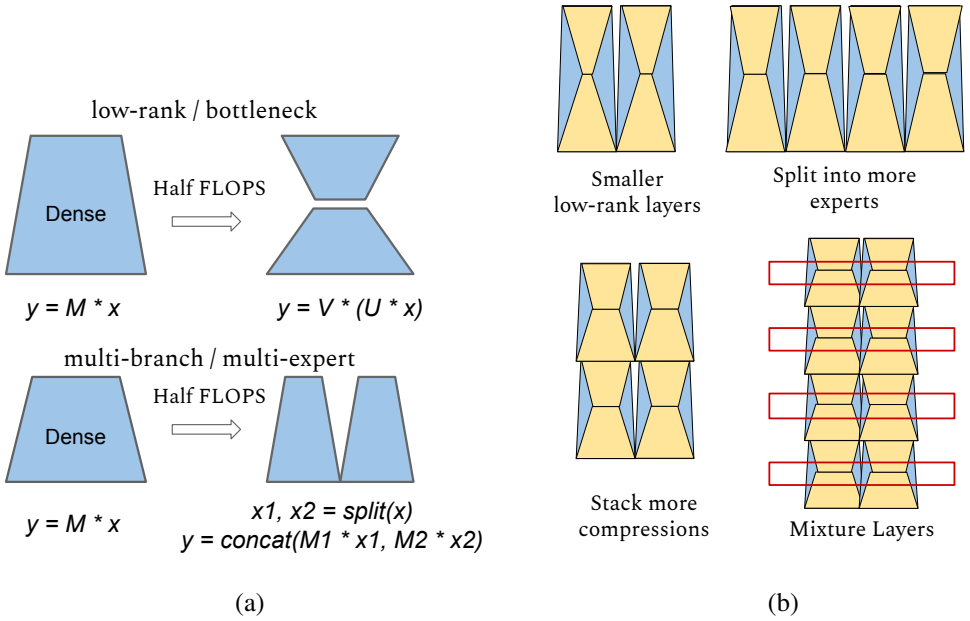


Figure 2: (a) Two methods of matrix factorization. (b) Evolving matrix factorization into transformer-styled model architecture.

From a computational efficiency perspective, there are various forms of computation factorization that can lead to lower computation cost or faster computation without hurting model accuracy. As indicated in fig. 2 (a), low-rank and multi-expert layers are two major methods for factorizing a matrix multiplication. When devising an efficient neural network, as indicated in fig. 2 (b), low-rank and multi-expert can be combined and stacked to achieve more interesting model architectures that are computationally efficient. Finally, by also allowing a temporal mixture layer to be used as one of the building blocks (e.g. attention (Vaswani et al., 2017), gMLP (Liu et al., 2021) or MLP mixer (Tolstikhin et al., 2021)) which captures the causal relations between tokens, the network becomes closer to a transformer variant.

However, constructing an efficient network does not require conforming to the uniformity of the model architecture as illustrated in the last figure of fig. 2 (b). We hypothesize that by carefully selecting layer types and layer interleaving, as well as hidden dimensions and expansion factors in the feed-forward layers, we could construct more efficient model architectures that converge faster

Table 1: Search Space Table: F_{attn} is a self-attention layer, F_{moe} is a sparsely gated FFN layer, and F_{ffn} is a regular dense FFN layer. The baseline is a 100M 12-layer dense transformer model with $H_{\text{model\_dim}}$ = 768.

Search Item	Search Space
Layer Type ($\mathcal{F}_i$)	$\mathcal{F}_{\text{attn}}$, $\mathcal{F}_{\text{moe}}$, $\mathcal{F}_{\text{ffn}}$
Model Dim. (d)	512, 768, 1024
MoE Hidden Dim. (d_{moe})	1536, 2048, 3072, 4096
FFN Hidden Dim. (d_{ffn})	1536, 2048, 3072, 4096
Attention Heads. (h)	12, 16, 20
Gating Func. (g)	*Top-2*, *Expert Choice*
Capacity Factor (c)	1, 2, 3, 4
Activation Func. (a)	*Gated ReLU*, *ReLU*, *GeLU*, *Gated GeLU*

to a higher quality. This leads our exploration towards a more training-efficient architecture by adopting low-rank and multi-expert compression methods with some additional considerations: 1) Optimal operator mixing – many prior works including gMLP (Liu et al., 2021) have discovered that with a modest number of attention heads, a MLP can match a transformer architecture on both vision and language tasks at much lower computational cost. Proper interleaving of dense and sparse layers enables not only network specialization via MoE but also fusion with dense layers. 2) Changing the expansion factor in sparse feed-forward layers, as the original expansion factor (where the feed-forward layer hidden dimension is projected $4\times$ from its model dimension) is optimized around dense model families. 3) Introducing heterogeneity in the gating function so different optimal architectures can be devised according to the gating mechanism.

3.2 Layer-wise Search Space

We largely take inspiration from the layer-wise compound scaling in EfficientNet (Tan & Le, 2019). We construct a layer operator search space where the restriction of interleaving self-attention and feed-forward layers is removed. Instead, we create a generic layer as a function $Y_i = \mathcal{F}_i(X_i), \mathcal{F}_i \in \{\mathcal{F}_{\text{attn}}, \mathcal{F}_{\text{moe}}, \mathcal{F}_{\text{ffn}}\}$ where $\mathcal{F}_i$ is an operator selected from the operation set consisting of self attention, sparsely gated feed-forward (MoE), and dense feed-forward sub-layers. Input X_i has a tensor shape of $\{B, L, H\}$ and $H \in \{\frac{3}{4}, 1, \frac{3}{2}\} \times H_{\text{model\_dim}}$ where B is the batch size, L is the sequence length, and H is a tunable model dimension. The intuition behind tuning model dimension is to enable more flexible network topologies with various factorization methods as described in section 3.1.

Unlike a traditional simple, uniform transformer block, a Brainformer block is a complex block $\mathcal{N}$ that can be represented by a list of composed layers: $\mathcal{N} = \mathcal{F}_k \odot ... \odot \mathcal{F}_2 \odot \mathcal{F}_1(X_1) = \bigodot_{j=1...k} \mathcal{F}_j(X_1)$. We can stack an arbitrary number of Brainformer blocks to create a model at a predefined scale.

The search objective is to find an optimal layer architecture $\mathcal{F}_i$, and model scaling multipliers for multiple model inner dimensions that minimizes the perplexity. Table 5.3 summarizes the search space in a Brainformer architecture. The introduction of mixture-of-experts effectively creates a wider layer, therefore tuning the hidden dimensions and the expansion factor in the feed-forward layers becomes essential.

3.3 Fair Comparisons Across Model Architectures

Prior NLP model scaling studies (Raffel et al., 2020; Radford et al., 2018; Brown et al., 2020b; Rae et al., 2021) typically explore quality scaling with fixed model capacity and training steps/tokens. For example, a scaling plot typically fixes training steps/tokens while varying the model parameters. However, when training a model, users typically have a fixed budget and can trade-off training time, compute resources, and quality to stay within that budget. If what we care about is computational cost and training convergence time, then comparing model qualities while fixing total parameters is not fair, particularly when comparing across model architectures and model families. For example, it may discriminate against models with more total parameters that consume fewer computational FLOPs, such as sparsely activated models. The GLaM paper (Du et al., 2022) addresses this by conducting a scaling study on activated memory (which approximates the computational cost), rather than the total parameter size, on a fixed number of training tokens. However, comparing models

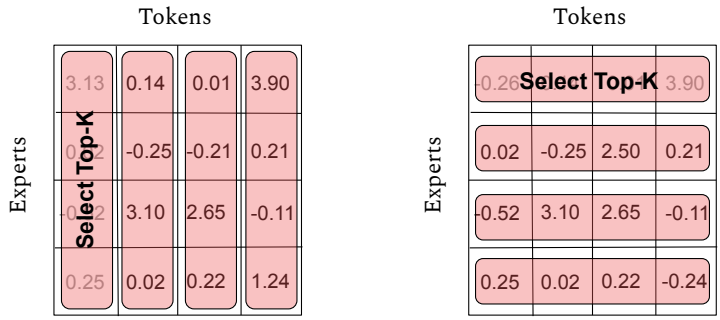


Figure 3: Token-based routing vs. Expert-based routing.

with a fixed amount of training tokens may still also not be fair as some smaller models can benefit more from additional training data and outperform a bigger model with the same total training cost (e.g. GPU hours, TPU hours, etc.). The Chinchilla paper (Hoffmann et al., 2022) is the first to suggest compute-efficient scaling, which varies both model capacity and training tokens at a fixed computational cost. Resonating with compute-efficient model scaling, we further take model architectural change into consideration during the search for efficient model architectures with better training convergence and inference time. More particularly, we compare across models with a fixed training cost and model inference time, which allows the search algorithm to trade off between model capacity and training tokens.

3.4 Training Time Constrained Search

Unlike many existing NAS methods, where total training steps is fixed during the proxy training of the sampled models, we fix the wall clock time for each trial which encourages identifying models with faster training convergence. The objective is to find model architectures that yield higher training accuracy. More particularly, the controller minimizes the pre-training validation cross-entropy loss while making sure model step time is smaller than the baseline model. In the search, we create a search space around a 100M dense vanilla transformer model, as illustrated in Table 5.3. Instead of searching for models with similar parameter count, identifying models with similar inference time or step time can be more relevant as a smaller parameter count does not always indicate faster speed. Each trial is trained with a fixed wall clock time so that faster models can be compensated with more training steps. **We empirically find that fixing training steps favors bigger models, however fixing training wall clock time favors models with faster convergence**.

$$\min_{\mathcal{F}_{1:k},d,d_{\text{moe}},d_{ffn},h,g,c,a} \mathcal{L}(\mathcal{N}(\mathcal{F}_{1:k},d,d_{\text{moe}},d_{ffn},h,g,c,a)) \tag{1}$$

$$\mathcal{F}_i = \begin{cases} \mathcal{F}_i^{d,h,a}, & \text{if} \quad \mathcal{F}_i = \mathcal{F}_{attn} \\ \mathcal{F}_i^{d,d_{ffn},a}, & \text{else if} \quad \mathcal{F}_i = \mathcal{F}_{ffn} \\ \mathcal{F}_i^{d,d_{\text{moe}},g,c,a}, & \text{otherwise} \quad \mathcal{F}_i = \mathcal{F}_{\text{moe}} \end{cases} \tag{2}$$

$$s.t. \quad \mathcal{N}(\mathcal{F}_{1:k},d,d_{\text{moe}},d_{ff},h,g,c,a) = \bigodot_{i=1\ldots k} \mathcal{F}_i(X_1) \tag{3}$$

$$\text{Step\_Time}(\mathcal{N}) \leq \text{baseline\_step\_time} \tag{4}$$

4 Token-based Routing Versus Expert-based Routing

While there are various routing methods in existing MoE literature, we primarily focus on two classes of routing: token-based routing and expert-based routing, to illustrate the idea that routing strategy can change the optimal model architecture when sparsely activated layers are introduced.

As an example, in Figure 3, the rows and columns contain un-normalized scores computed for four tokens and four experts. Each value is produced by the dot product of the token embedding

and the expert embedding. Once the token-to-expert affinity scores are generated, there are a few ways to decide which experts each token should be routed to. In token-based routing, the model routes to the top-k experts for each token, while in an expert-based routing, the experts choose top-k tokens. More particularly, we follow the top-2 gating approach used in GShard (Lepikhin et al., 2021) and GLaM (Du et al., 2022) as top-2 has demonstrated stronger empirical performance than top-1 gating. For the expert-based gating, we follow the Expert Choice gating (Zhou et al., 2022b) where perfect load balance is achieved with heterogeneous parameter allocation. Expert-based routing has demonstrated superior performance on language understanding tasks yet still provides competitive results in generative tasks.

There are various ways of generating the token-to-expert affinity scores. One possible way is to create a trainable gating matrix W_g that projects the input feature space to a token-to-expert score. The score should be normalized either along the token dimension or the expert dimension. To avoid causal leakage in decoding mode, we suggest normalizing along the expert dimension for both token-based routing and expert-based routing.

Token-based Gating: Normally, a noisy top-k function is applied in token-based gating. Before taking the softmax function, we add tunable Gaussian noise and then keep only the top k values in the gating matrix, setting the rest to $-\infty$. The noise term helps with load balancing and generalization.

$$\begin{aligned} H_i &= (X \cdot W_g)_i + \text{StandardNormal}() \times \text{Softplus}((X \cdot W_{\text{noise}})_i) \\ G &= \text{Softmax}(\text{KeepTopK}(H_i, k)) \end{aligned} \tag{5}$$

Expert-based Gating: $TopK()$ selects the k largest entries along the token dimension, as illustrated in eq. (6). I is the selected indices and the output computation of the layer is the linearly weighted combination of each expert's computation on the token by the gate value.

$$\begin{aligned} S &= \text{Softmax}(X \cdot W_g), \quad S \in \mathbb{R}^{n \times e} \\ G, I &= \text{TopK}(S^\top, k) \end{aligned} \tag{6}$$

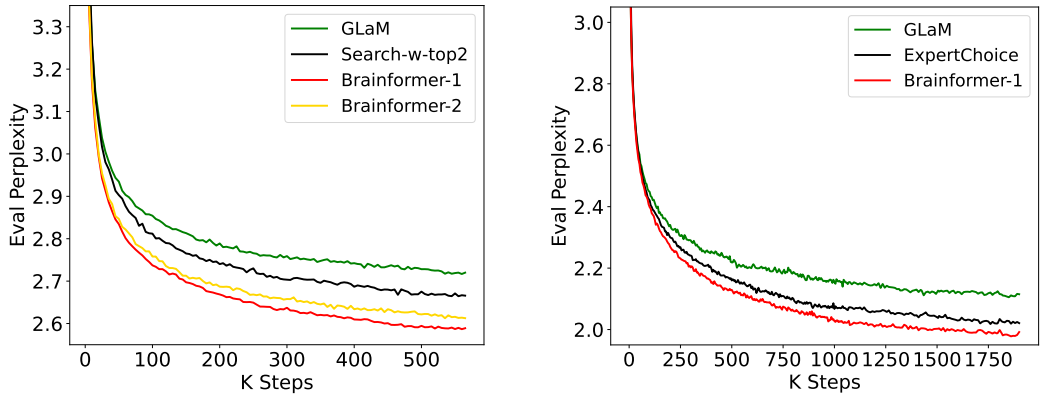
5 Evaluation

Setup: Table 2 summarizes the hyperparameter settings of different baseline MoE models. In the baseline MoE GLaM (Du et al., 2022) model, we interleave transformer blocks with regular dense FFNs and transformer blocks with sparsely gated FFNs (MoE layer). As a reference point, we also include the respective dense model configurations with comparable numbers of activated parameters per-token during inference in the table. With a similar number of activated parameters as a 0.1B dense model, 0.1B/32E represents the sparse model with every other transformer layer replaced by a 32-expert MoE layer. While n_{params} is the total number of trainable parameters, $n_{\text{act-params}}$ represents the number of activated parameters per token. $n_{\text{act-params}}$ roughly approximates the computational expensive of a model. L is the total number of Transformer layers, M is the model dimension, H is the hidden dimension after the projection in each transformer layer, n_{heads} is the number of attention heads, and d_{head} is the hidden dimension of each attention head. We train and evaluate our Brainformer models and baseline models on 64 Cloud TPU-V4 chips, except for models at the 8B-scale which take 512 Cloud TPU-V4 chips to train.

Table 2: Sizes and architectures of baseline dense models and MoE (GLaM) models. Models are grouped by the number of activated parameters per token.

Model	Type	n_{params}	$n_{\text{act-params}}$	L	M	H	n_{heads}	d_{head}	E
0.1B	Dense	130M	130M	12	768	3,072	12	64	–
0.1B/32E	MoE	1.9B	145M						32
1.7B	Dense	1.7B	1.700B	24	2,048	8,192	16	128	–
1.7B/64E	MoE	27B	1.879B						64
8B	Dense	8.7B	8.7B	32	4,096	16,384	32	128	-
8B/64E	MoE	143B	9.8B						64

Dataset: We use the high-quality dataset from GLaM of 1.6 trillion tokens that are representative of a wide range of natural language use cases. This dataset consists of a high-quality filtered subset of webpages that are combined with smaller corpora of books, Wikipedia pages, conversations, forums,



(a) (b)

Figure 4: (a) Pre-training perplexity comparison for 100M32E (100M parameters per expert, 32 experts). Search-w-top2 is the model found by using neural architecture search but with fixed top-2 token-based gating. (b) Training perplexity comparison for 8B64E (8B parameters per experts, 64 experts). Expert Choice is the GLaM architecture with expert-based gating function.
and news to create the final dataset. A more detailed description of the dataset including the data and mixture weights can be found in the GLaM paper (Du et al., 2022).

Model Training: We train a few decoder-only models using the searched best Brainformer blocks and related baselines. Brainformer-1 and Brainformer-2 are two selected best models. With limited computational resources, we only scale Brainformer-1 to 1B and 8B scales. Our model training follows the setup of GLaM where a maximum sequence length of 1024 tokens is used. We use an Adafactor optimizer (Shazeer & Stern, 2018) with first-moment decay $\beta_1 = 0$ and second-moment decay $\beta_2 = 0.99$. The learning rate is kept constant for the first 10K training steps, then is decayed with an inverse square root schedule. We use the SentencePiece subword tokenizer with a vocabulary of size of 256K. The 100M-scale models and 1B-scale models are trained with 64 TPU V4 chips, while the largest model (8B/64E) evaluated is trained on 512 TPU V4 chips. We use a dropout rate of 0 during training as the number of tokens in the training data corpus is much greater than the total number of tokens used during training. We don't use any dropout during training because the training corpus is large enough that each sample is only encountered once.

Model Evaluation: We mainly focus on two types of downstream evaluation: 1) Fine-tuning performance on 11 selected classification tasks from the GLUE and SuperGLUE benchmarks (Wang et al., 2018; 2019). 2) We evaluate oneshot performance with three language generation tasks focused on question answering.

5.1 Training Convergence

In this section, we evaluate Brainformer top models with related baselines including 1) Top-2 gating based model architecture search (Search-w-Top2) and 2) GLaM (Du et al., 2022), a manually crafted architecture with fixed top-2 gating. Providing the flexibility of tuning the gating function and network architecture significantly improves pre-training efficiency. As shown in table 4, our searched best Brainformer models outperform the baselines in terms of computational cost (activated parameters), training step time (steps/sec), and training perplexity (PPLX) for fixed training steps. When scaled to 8B64E, Brainformer is more than 5x faster in step time and 2x faster in training convergence to the same perplexity using the same hardware configuration (512 Cloud TPU-V4 chips). With a fixed 600B training tokens, Brainformer is much more accurate than the baselines at 8B scale.

5.2 Finetuning Results

We pretrain the models for a total **fixed wall clock time** as the baseline GLaM model. We then finetune the models with eleven selected GLUE and SuperGLUE classification tasks. At two different scales, 100M64E and 1B64E, Brainformers outperform the baseline GLaM model by a significant margin of 2-4% average score. The fine-tuning results indicates that Brainformer not only excels at training convergence but also generalizes well to downstream tasks.

Table 3: Training efficiency comparison. Brainformer models have better training convergence and faster step times, compared to GLaM, fixed gating search, and expert-based gating but with fixed architecture. Brainformer-1 and Brainformer-2 are two selected best models. With limited computational resources, we only scale Brainformer-1 to 1B and 8B scales.

Model	Total Params	Activated Params	Train Steps	Steps/Sec	PPLX
100M32E					
GLaM	1B	145M	0.5M	1.92	2.73 +/- 0.002
Search-w-Top2	1.87B	210M	0.5M	2.03	2.67 +/- 0.005
Brainformer-1	3.19B	156M	0.5M	2.03	**2.57** +/- 0.003
Brainformer-2	3.33B	266M	0.5M	**2.16**	2.59 +/- 0.005
1B64E					
GLaM	27B	1.88B	1.0M	1.23	2.25 +/- 0.004
Search-w-Top2	27B	3.05B	1.0M	1.27	**2.21** +/- 0.003
Brainformer-1	30B	1.38B	1.0M	**2.00**	2.25 +/- 0.002
Brainformer-2	52B	**1.31B**	1.0M	1.76	2.23 +/- 0.001
8B64E					
GLaM	143B	9.8B	1.5M	0.39	2.12 +/- 0.002
Expert-based Gating	143B	9.8B	1.5M	0.50	2.03 +/- 0.005
Brainformer-1	158B	**7.4B**	1.5M	**1.96**	**1.99** +/- 0.002

Table 4: Finetuning Results: Brainformers at 100M and 1B significantly outperform GLaM counterparts, yielding over 3% gains in overall scores.

Size	Model	BoolQ	CB	CoLA	MNLI	MRPC	QNLI
100M64E	GLaM	0.791	0.859	0.818	0.849	0.833	0.901
	Brainformer-1	0.812	0.922	0.828	0.855	0.870	0.907
1B64E	GLaM	0.829	0.938	0.831	0.860	0.857	0.919
	Brainformer-1	0.859	0.938	0.863	0.896	0.875	0.938
Size	**Model**	**QQP**	**RTE**	**SST2**	**WiC**	**WNLI**	**AVG**
100M64E	GLaM	0.907	0.808	0.952	0.687	0.609	0.819
	Brainformer-1	0.812	0.840	0.952	0.702	0.635	**0.840**
1B64E	GLaM	0.911	0.816	0.945	0.711	0.547	0.833
	Brainformer-1	0.917	0.899	0.972	0.720	0.719	**0.873**

Table 5: Oneshot evaluation on three important generative tasks. All models are trained with 200B training tokens. Brainformer-CF1 and Brainformer-CF3 are both trained with capacity factor of one, but during decoding, Brainformer-CF3 uses a larger capacity factor of three.

Model	Nqs	Triviaqa	Webqa	Steps/Sec
GLaM 1B64E	**9.14**	41.8	10.8	0.55
Primer 1B (So et al., 2021)	4.82	24.7	6.50	1.50
Brainformer-CF1 1B64E	7.06	40.7	11.2	1.37
Brainformer-CF3 1B64E	8.23	**43.4**	**12.0**	1.37

5.3 Fewshot Results

Aligned with prior work in fewshot in-context learning, we compare Brainformer oneshot performance on three selected generative tasks: Natural Questions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and Web Questions (Berant et al., 2013), with a sparse model GLaM and a dense model Primer (So et al., 2021) of similar activated memory size. Brainformer outperforms Primer by a large margin on all the tasks except SQuAD-V2. GLaM yields competitive scores while being 2x slower than Brainformer.

6 DISCUSSION

6.1 VISUALIZING A BRAINFORMER BLOCK

In this section, fig. 5 provides a visualization of a Brainformer architecture block. Unlike a conventional transformer block, where there is only an attention layer and a dense feed-forward (FFN) layer, a Brainformer block contains 8 sub-layers. The Brianformer block is repeated 3 times, 6 times, and 8 times respectively in the 100M, 1B, and 8B scale. Brainformer block is repeated 3 times. Sandwiching dense layers and sparse layers creates specialization in the MoE layers while enabling information fusion in the dense FFNs. In a conventional transformer model, a dense FFN layer has an optimized expansion ratio of 4, which results in a hidden dimension 4x wider than the model dimension. In the optimized Brainformer block, the search algorithm picks a slightly larger model dimension of 1024 (as compared to 768) and a smaller expansion factor in the dense FFNs and MoE layers (as compared to 3072). This is a reasonable optimization, as MoE layers effectively widen the network with more experts. Keeping the expansion ratio the same as a dense model, MoE models can be hard to optimize at a large model capacity (e.g 64B). In the MoE layers, the search algorithm picks the expert choice gating function (Zhou et al., 2022b) with a capacity factor of one. This indicates a very sparse network in which each token can be routed to a single expert on average.

6.2 CAN WE SIMPLIFY?

We did an ablation study on block simplification. A very natural question to ask is whether we can simplify the architecture block. In exploring the answer to this question we were able to extrapolate some patterns. We find that the ratio of different layer types is critical to model quality: replacing a layer with a different layer results in degraded quality. However, the network is relatively insensitive to layer order, such that swapping any two layers would not affect performance much. For example, to create a simplified pattern, we can interleave the dense FFNs and MoE layers or simply creating contiguous layers of the same type.

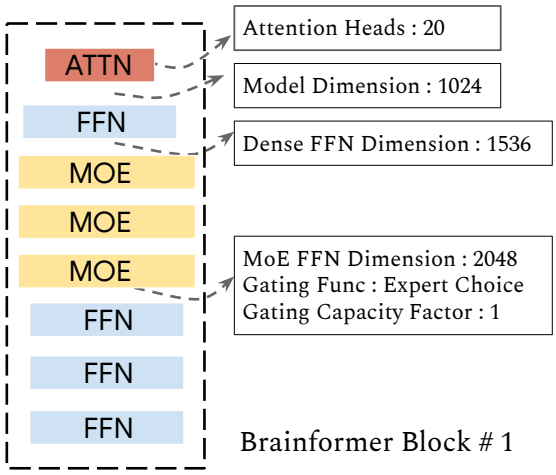


Figure 5: Brainformer Block

7 CONCLUSION

Using an evolutionary search algorithm, we have developed and evaluated a complex architecture block, named Brainformer, that consists of a diverse sequence of layers, including a sparsely gated feed-forward layer. Along with the new block, we also propose evaluating using a fixed training time search, which enables fair comparisons across model families. Brainformer demonstrates up to 2× faster training convergence and 5× faster step time compared to its GLaM counterpart. In downstream task evaluation, Brainformer also demonstrates a 3% higher SuperGLUE score with fine-tuning compared to GLaM, and greatly outperforms Primer on oneshot evaluation for three out of three generative tasks. Our work shows that searching for better non-uniform Transformer model blocks can result in significant gains over classic uniform stacking.

REFERENCES

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1533–1544, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1160.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020a. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020b.

Kyunghyun Cho and Yoshua Bengio. Exponentially increasing the capacity-to-computation ratio for conditional computation in deep learning. *arXiv preprint arXiv:1406.7362*, 2014.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/7137debd45ae4d0ab9aa953017286b20-Paper.pdf.

Zihang Dai, Hanxiao Liu, Quoc V. Le, and Mingxing Tan. CoAtNet: Marrying convolution and attention for all data sizes. In *Advances in Neural Information Processing Systems*, 2021.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pp. 5547–5569. PMLR, 2022.

Dheeru Dua, Shruti Bhosale, Vedanuj Goswami, James Cross, Mike Lewis, and Angela Fan. Tricks for training sparse translation models. *arXiv preprint arXiv:2110.08246*, 2021.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2021.

Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 7036–7045, 2019.

Sam Gross, Marc'Aurelio Ranzato, and Arthur Szlam. Hard mixtures of experts for large scale weakly supervised vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6865–6873, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016b.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc Le. Transformer quality in linear time. In *International Conference on Machine Learning*, pp. 9099–9117. PMLR, 2022.

Sebastian Jaszczur, Aakanksha Chowdhery, Afroz Mohiuddin, Lukasz Kaiser, Wojciech Gajewski, Henryk Michalewski, and Jonni Kanerva. Sparse is enough in scaling transformers. *Advances in Neural Information Processing Systems*, 34:9895–9907, 2021.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada, July 2017. Association for Computational Linguistics.

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

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 2019.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*, 2021.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*, pp. 6265–6274. PMLR, 2021.

Min Lin, Jie Fu, and Yoshua Bengio. Conditional computation for continual learning. *arXiv preprint arXiv:1906.06635*, 2019.

Hanxiao Liu, Zihang Dai, David So, and Quoc V Le. Pay attention to mlps. *Advances in Neural Information Processing Systems*, 34:9204–9215, 2021.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, pp. 1045–1048. Makuhari, 2010.

Ofir Press, Noah A Smith, and Omer Levy. Improving transformer models by reordering their sublayers. *arXiv preprint arXiv:1911.03864*, 2019.

Joan Puigcerver, Carlos Riquelme, Basil Mustafa, Cedric Renggli, André Susano Pinto, Sylvain Gelly, Daniel Keysers, and Neil Houlsby. Scalable transfer learning with expert models. *arXiv preprint arXiv:2009.13239*, 2020.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

Stephen Roller, Sainbayar Sukhbaatar, Jason Weston, et al. Hash layers for large sparse models. *Advances in Neural Information Processing Systems*, 34:17555–17566, 2021.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pp. 4596–4604. PMLR, 2018.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

David So, Wojciech Mańke, Hanxiao Liu, Zihang Dai, Noam Shazeer, and Quoc V Le. Searching for efficient transformers for language modeling. *Advances in Neural Information Processing Systems*, 34:6010–6022, 2021.

Xiu Su, Shan You, Jiyang Xie, Mingkai Zheng, Fei Wang, Chen Qian, Changshui Zhang, Xiaogang Wang, and Chang Xu. Vision transformer architecture search. *arXiv e-prints*, pp. arXiv–2106, 2021.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *ICML*, 2011.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pp. 6105–6114. PMLR, 2019.

Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention for transformer models. In *International conference on machine learning*, pp. 10183–10192. PMLR, 2021.

Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems*, 34: 24261–24272, 2021.

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019.

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

Yuekai Zhao, Li Dong, Yelong Shen, Zhihua Zhang, Furu Wei, and Weizhu Chen. Memory-efficient differentiable transformer architecture search. *arXiv preprint arXiv:2105.14669*, 2021.

Qinqin Zhou, Kekai Sheng, Xiawu Zheng, Ke Li, Xing Sun, Yonghong Tian, Jie Chen, and Rongrong Ji. Training-free transformer architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10894–10903, 2022a.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew Dai, Zhifeng Chen, Quoc Le, and James Laudon. Mixture-of-experts with expert choice routing, 2022b. URL `https://arxiv.org/abs/2202.09368`.

Simiao Zuo, Xiaodong Liu, Jian Jiao, Young Jin Kim, Hany Hassan, Ruofei Zhang, Tuo Zhao, and Jianfeng Gao. Taming sparsely activated transformer with stochastic experts. *arXiv preprint arXiv:2110.04260*, 2021.