DARTFormer: Finding The Best Type Of Attention

Jason Ross Brown
University of Cambridge
jrb239@cam.ac.uk

Yiren Zhao
Imperial College London and University of Cambridge
a.zhao@imperial.ac.uk

Ilia Shumailov
University of Oxford
ilia.shumailov@chch.ox.ac.uk

Robert D Mullins
University of Cambridge
robert.mullins@cl.cam.ac.uk

Abstract

Given the wide and ever growing range of different efficient Transformer attention mechanisms, it is important to identify which attention is most effective when given a task. In this work, we are also interested in combining different attention types to build heterogeneous Transformers. Focussing on NLP-based tasks, we first propose a DARTS-like Neural Architecture Search (NAS) method to find the best attention for a given task. In this setup, all heads use the same attention (homogeneous models). Our results suggest that NAS is highly effective at this, and it identifies the best attention mechanisms for IMDb byte level text classification and Listops. We then extend our framework to search for and build Transformers with multiple different attention types, and call them heterogeneous Transformers. We show that whilst these heterogeneous Transformers are better than the average homogeneous models, *they cannot outperform the best*. We explore the reasons why heterogeneous attention makes sense, and why it ultimately fails.

1 Introduction

Since the first proposal of the Transformer architecture by Vaswani et al. (2017), many alternatives have been proposed for the attention mechanism (Zaheer et al., 2020; Katharopoulos et al., 2020; Wang et al., 2020; Liu et al., 2018b; Beltagy et al., 2020; Choromanski et al., 2020; Kitaev et al., 2020; Child et al., 2019; Tay et al., 2021) due to the original dot-product attention mechanism having quadratic complexities in time and space with respect to the length of the input sequence. Recent work (Tay et al., 2020b) showed that these alternative architectures often perform well at different tasks when there is no pretraining, thus there is no clear attention that is best at every type of task. Therefore we ask: *Can we efficiently learn the best attention for a given long range task?*

It is thought that each attention head in the Transformer can learn a different relationship, much like how in Convolutional Neural Networks (CNNs) each kernel learns a different feature. Tay et al. (2020b) hypothesizes that each attention mechanism represents a different functional bias for which attention relationships should be learned, and that the utility of this bias is dependent on the task and its processing requirements. Thus if we had many different attention types in a Transformer, they could each more easily learn different types of relationship, and thus make the Transformer more effective overall. *So is the optimal attention for a task a mixture of different attentions?*

In this paper we apply Neural Architecture Search (NAS) techniques to the Transformer attention search space and propose DARTFormer, a DARTS-like method (Liu et al., 2018a) for finding the best attention, a high level illustration of the method is presented in Figure 1. To do this we use multiple candidate attention types in parallel and sum their outputs in the attention mechanism of the

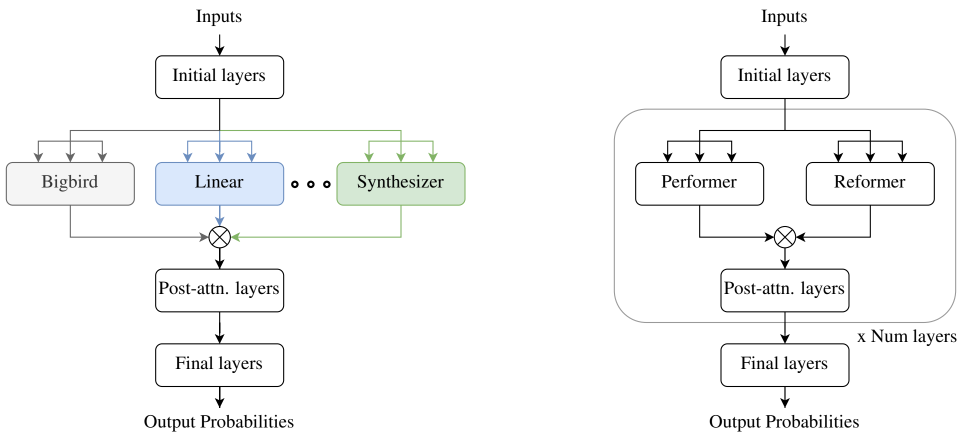


Figure 1: Our supernetwork architecture (left) that we search over, and an example final derived architecture (right) that is heterogeneous across heads and homogeneous across layers. The supernetwork is a single layer Transformer with several different multi-head attention blocks whose outputs are averaged.

Transformer. Following Wang et al. (2021) we use masked validation accuracy drop as our metric for determining the performance of each attention type.

For clarity we refer to computation of the QKV linear projections, multi-head attention, and then concatenation and linear projection back down to the sequence features as the attention block. In a standard Transformer there is only a single attention block that has multiple heads. When training mixed attention models (either supernetworks or a final heterogeneous model) we use multiple attention blocks, each containing only a single attention type.

Following Tay et al. (2020b), we use a representative mixture of different attention mechanisms as part of our search space in order to cover the main methods of achieving efficient attention. The specific attentions we use are: Bigbird (Zaheer et al., 2020), Linear Transformer (Katharopoulos et al., 2020), Linformer (Wang et al., 2020), Local attention (Liu et al., 2018b), Longformer (Beltagy et al., 2020), Performer (Choromanski et al., 2020), Reformer (Kitaev et al., 2020), Sparse Transformer (Child et al., 2019), and Synthesizer (Tay et al., 2021).

We use our setup to investigate two key paradigms. The first is learning the best attention for a new task with a single layer Transformer, this means only one full-scale Transformer needs to be trained after a good attention mechanism is found. The second paradigm, illustrated in Figure 1, is using a single layer Transformer to find the best head-wise heterogeneous attention mixture for that task, and then using that mixture in each layer of a full Transformer model, making it layer-wise homogeneous. We test these paradigms on three different tasks. They are taken from Tay et al. (2020b) and were specifically designed to test the capabilities of efficient long range Transformers.

In this paper we make the following contributions:

- We propose a DARTS-like framework to efficiently find the best attention for a task.
- We extend this framework to building and searching for optimal heterogeneous attention Transformer models.
- We empirically show that heterogeneous Transformers *cannot* outperform the best homogeneous Transformer for our selected long range NLP tasks.

2 Related Work

2.1 Transformer Attention

Since Vaswani et al. (2017), a large variety of replacement attention mechanisms were proposed. These use different approaches, such as, low rank approximations and kernel based methods

(Wang et al., 2020; Choromanski et al., 2020; Katharopoulos et al., 2020; Tay et al., 2021), fixed/factorised/random patterns (Tay et al., 2021; Child et al., 2019; Zaheer et al., 2020; Beltagy et al., 2020; Tay et al., 2020a), learnable patterns (Kitaev et al., 2020; Tay et al., 2020a), recurrence (Dai et al., 2019) and more (Lee et al., 2019). Tay et al. (2020c) gives a detailed survey of different attention mechanisms.

Tay et al. (2020b) compares the performance, speed and memory usage of many of these different attention mechanisms on a variety of tasks, including NLP and image processing. Their main finding is that the performance of each attention mechanism is highly dependent on the nature of the task being learned when pretrained embeddings or weights aren't used. This motivates the initial part of our research.

2.2 NAS on Transformers

Recent work of Liu et al. (2022) applies RankNAS (Hu et al., 2021) to cosFormer (Qin et al., 2022) and standard Transformers (Vaswani et al., 2017). They perform a search of the hyperparameters on a cosFormer network and compare these to the same method applied to the standard Transformer. Tsai et al. (2020) searches the hyperparameter space of a BERT (Devlin et al., 2018) model architecture heterogeneously to find an optimal efficient network. These methods search hyperparameters, whereas we fix hyperparameters and search over the space of attention mechanisms.

3 Method

3.1 DARTS Style NAS

We train a supernetwork that contains attention heads of each attention type. This is analogous to the edges in a DARTS cell containing all possible operations for that edge. We also use 'fixed α' with masked validation accuracy as our metric for the strength of each edge as detailed in Wang et al. (2021). Standard DARTS assigns a weight for each edge in the computation graph and softmaxes this on the output of each cell (see Figure 2, left, in Appendix C), since we are using a 'fixed α' approach we do not need to do this and simply average our edges without using learnable weights. This allows us to train the supernetwork until convergence and then select the best attention (or prune out the worst). Note that this removes the need to carry out the bi-level optimisation that is required in the original DARTS paradigm that uses validation performance to train the edge weights. Our approach is illustrated in Figure 1. The DARTS supernetwork cell and the standard Transformer architecture are given in Appendix C in Figure 2, to show how our architecture relates to them.

3.2 Architecture and Search Space

Our Transformer encoder supernetwork consists of an embedding layer, an attention block supernetwork, a feed-forward network (FFN), and a linear classifier layer. In normal Transformer attention, each attention block consists of: QKV linear projections from features to number of heads $\times$ head dimension, scaled dot product attention on each head, concatenation of heads, and a final dense projection back to the feature dimension (Vaswani et al., 2017). In theory, we want to search over the space of alternatives for the scaled dot product attention operation independently for each head. This would replace scaled dot product attention in a normal Transformer with an average of candidate attention mechanisms in a DARTS-like paradigm. The averages from each head would then be concatenated and linearly projected back to the feature dimension as normal.

However, some attention mechanisms, such as Reformer (Kitaev et al., 2020), modify the linear projections or concatenation operations. Because of this we instead search over candidate multi-head attention blocks where each one implements the projections, attention, and concatenation operations. This is illustrated in Figure 4 in Appendix C . When the candidate blocks are single head and we have H blocks per candidate attention mechanism, after the architecture search is complete, the computation graph is equivalent to if we searched over the attention mechanisms themselves with H heads. With the search at the block level with single heads, each attention mechanism can learn its own linear projections, whereas in a search over just the attention mechanisms these would be shared within each head. The disadvantage of this is increased memory and computation during the search, since now heads do not share the linear projection.

3.3 Experiments

3.3.1 Finding Optimal Homogeneous Attention

For the first experiment we train a single layer network with a block for each attention mechanism. These blocks are each initialized with the desired number of final heads for that task. After training we perform a single masked validation accuracy trial and pick the highest scoring mechanism as a good candidate mechanism for that task when used in a full Transformer model homogeneously. This paradigm is summarized in an algorithmic form in Algorithm 1 in Appendix D.

3.3.2 Finding Optimal Heterogeneous Attention

In this paradigm we try to learn an optimal single layer with a fixed number of heads and stack it, analogous to searching for an optimal cell in DARTS.

We begin with H attention blocks of each attention type, each one a single head, where H is the desired number of final attention heads in our heterogeneous Transformer. We train the supernetwork to rough convergence and then alternate between removing the worst attention block and fine-tune training the remaining model. We do this until we have only H single-head attention blocks left, each potentially a different attention mechanism.

Once we have the optimal layer architecture for the task, we reinitialize the Transformer network with multiple layers where each one contains the mix of attentions found to be optimal. Where possible we group attentions of the same type into the same attention block with multiple heads. We then train this full Transformer model on the task until convergence. This paradigm is summarized in Algorithm 2 in Appendix D. We will refer to this method as 'NAS Prune'.

We also test a simpler method where we take the top 4 scoring attentions from the single layer to find the best homogeneous attention, and then train a full Transformer network with an even mix of these attentions at each layer. This allows us to evaluate the general capabilities of heterogeneous attention. We will refer to this method as 'NAS One-shot'.

We also experiment with a head-wise homogeneous, layer-wise heterogeneous attention Transformer. This is detailed in Appendix B.

3.4 Attention Evaluation

In order to evaluate the strength of each block we used a masked validation accuracy drop metric (Wang et al., 2021). First we get a baseline accuracy by running our model on the validation set. Then, for each block, we mask it out and record the new validation accuracy. This is illustrated in Figure 3 in Appendix C. The best block is the one which drops the accuracy the most, and the worst either drops it the least or increases it the most.

4 Evaluation

For each experiment we run the search on 3 different tasks taken from Long Range Arena (LRA) (Tay et al., 2020b). These are: byte level binary text classification on IMDb dataset with 1k sequence length, Listops - 10-way classification with 2k sequence length, and byte level document matching which uses 4k sequence length. Hyperparameters typically match those used in LRA, details are given in the Appendix A. One notable difference is that we train each homogeneous model 3 times and take an average. The Listops and document matching tasks are also trained for longer to allow the models to better converge.

4.1 Finding Optimal Homogeneous Attention

Table 1 shows us that our selection process is effective at identifying the best attention mechanism for the text classification and Listops tasks. For both it also picks out the third best performing model with either its own second or third best. For the document matching task, however, the single layer fails to find one of the top 3 attention mechanisms within top 3 picks. One notable difference between its' document matching scores and the scores for the other two are that they are much smaller. This could indicate that when the score is low magnitude, the model is not confident in its selection. This can then be used to identify if it made a poor choice in advance of training the full homogeneous model.

Table 1: Test accuracy for the full model based on results of Tay et al. (2020b), and masked validation accuracy impact (more positive is better) for our trained single layer multiple attention block network (**first**, <u>second</u>, *third*).

Attention type	Text classification		Listops		Document Matching	
	Acc	Score	Acc	Score	Acc	Score
Bigbird	*62.7*	*0.30*	36.7	0.25	63.9	*0.011*
Linear Transformer	<u>64.4</u>	0.17	<u>37.1</u>	-0.40	64.2	-0.006
Linformer	58.6	-0.28	29.9	0.00	*64.5*	0.006
Local	56.3	0.14	36.8	0.30	58.0	0.000
Longformer	61.8	-0.20	36.8	*0.40*	61.2	**0.039**
Performer	**64.5**	**0.51**	36.2	0.30	<u>65.0</u>	0.000
Reformer	55.7	<u>0.44</u>	**37.8**	**11.85**	58.3	<u>0.028</u>
Sparse Transformer	61.3	0.02	*37.0*	<u>0.50</u>	63.2	0.000
Synthesizer	61.4	-0.03	36.5	0.30	**71.1**	-0.006

4.2 Finding Optimal Heterogeneous Attention

Table 2 shows the performance for the best homogeneous model, the fully trained model with an architecture chosen by the pruning NAS method (NAS Prune), and the fully trained model with an architecture chosen by taking the best four attentions from the single layer selection to find the best attention (NAS One-shot).

Table 2: The selected attention mixes with accuracy of full model vs the accuracy of the best homogeneous models for each task

Task	H	Best Homogeneous		NAS Prune		NAS One-shot	
		Attention	Acc	Attentions	Acc	Attentions	Acc
Text Classification	8	Performer	64.5	Linear Performer x2 Reformer x2 Sparse x2 Synthesizer	63.9	Bigbird x2 Linear x2 Performer x2 Reformer x2	64.4
Listops	8	Reformer	37.8	Linear Longformer Performer Reformer x3 Sparse x2	36.3	Local x2 Longformer x2 Reformer x2 Sparse x2	37.1
Document Matching	4	Synthesizer	71.1	BigBird Linear Sparse Synthesizer	67.0	Bigbird Linformer Longformer Reformer	64.7

This shows us that heterogeneous Transformer networks fail to beat the best homogeneous model for any task. We also experimented with choosing a mixture of the best performing homogeneous attentions, and having more attention heads of the better attention mechanisms for that task, but neither of these two variations gave consistent improvement on the NAS One-shot method. This also shows us that the expensive pruning method poses no consistent advantages over the far cheaper NAS One-shot when it correctly identifies good attentions.

5 Theories For Heterogeneous Attention Sub-optimality

Whilst searching for the optimal homogeneous attention works well and validates our NAS method, our found heterogeneous attention Transformer models cannot beat the best homogeneous models.

This conflicts with the idea that the different mechanisms provide different biases which help the Transformer learn many different and useful relationships. Here we provide some theories to explain what we have observed.

Different attention mechanisms do not bias the relationships learned. It might be that the relationships learned are not affected by the attention type, and the performance differences noted in Tay et al. (2020b) are due to some other effect. Thus the amount of functional bias by attention mechanisms is small.

Each tasks is best solved with attention that has learned specific types of relationships. Whilst learning lots of relationships might be useful, perhaps each tasks requires a narrow subset of relationships to be learned, and thus a homogeneous attention well suited to this narrow subset is optimal.

Bad attention degrades performance. It might be that the different attention mechanisms do learn different relationships, and this generally helps, but a small number of poorly suited attention types degrade model performance. This could be tested by more comprehensive studies on the effects of combining different attention types.

The training of other aspects of the Transformer is dependent on the attention mechanism. It might be that when trained in homogeneous models, the attention mechanism influences how other parts of the model learn, such as the feed-forward network. Thus if there were multiple attention mechanisms, these parts might have conflicting pressures on their learning which cause them to not be well suited to any of the attention mechanisms.

6 Conclusion

For a given task it is often not clear which attention mechanism will perform best. Using a DARTS-like method you can train a single layer mixed attention Transformer and use masked validation accuracy drop to determine the best attention to use in a full Transformer model. If the masked validation drop scores are very low, the best attention may not have been correctly identified.

Heterogeneous attention Transformer networks typically perform better than the average homogeneous one. Reasonable combinations of attention can be found via a DARTS-like method. However, heterogeneous Transformers cannot beat a homogeneous model that is using the best attention mechanism for that given task.

References

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

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

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.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

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

Chi Hu, Chenglong Wang, Xiangnan Ma, Xia Meng, Yinqiao Li, Tong Xiao, Jingbo Zhu, and Changliang Li. Ranknas: Efficient neural architecture search by pairwise ranking. *arXiv preprint arXiv:2109.07383*, 2021.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pp. 5156–5165. PMLR, 2020.

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

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pp. 3744–3753. PMLR, 2019.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018a.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*, 2018b.

Zexiang Liu, Dong Li, Kaiyue Lu, Zhen Qin, Weixuan Sun, Jiacheng Xu, and Yiran Zhong. Neural architecture search on efficient transformers and beyond. *arXiv preprint arXiv:2207.13955*, 2022.

Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. cosformer: Rethinking softmax in attention. *arXiv preprint arXiv:2202.08791*, 2022.

Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse sinkhorn attention. In *International Conference on Machine Learning*, pp. 9438–9447. PMLR, 2020a.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020b.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Computing Surveys (CSUR)*, 2020c.

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.

Henry Tsai, Jayden Ooi, Chun-Sung Ferng, Hyung Won Chung, and Jason Riesa. Finding fast transformers: One-shot neural architecture search by component composition. *arXiv preprint arXiv:2008.06808*, 2020.

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.

Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable nas. *arXiv preprint arXiv:2108.04392*, 2021.

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

Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. *arXiv preprint arXiv:1907.05737*, 2019.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297, 2020.

A Experimental Details

In all tasks we tested on, we used an Adam optimizer with linear warm-up and square root decay. We used a base learning rate of 0.05, $\beta_1 = 0.9$, and $\beta_2 = 0.98$. We used a batch size of 32 for every task. All attention mechanisms used the [CLS] token for classification. Task specific hyperparameters are given in table 3. In all Transformer models the feed-forward network (FFN) has a single hidden dimension. Before testing we take the model which had the best validation accuracy during training, the long train times are to ensure convergence.

Table 3: Task specific hyperparameters, see introduction for definitions.

Task	**Training Steps**	**Warmup**	**Seq. Length**	E	A	M
IMDb Token Level	$30k$	$8k$	$1k$	512	64	2048
IMDb Byte Level	$30k$	$8k$	$1k$	512	64	2048
Listops	$10k$	$1k$	$2k$	512	64	2048
Document Matching	$10k$	$8k$	$4k$	128	32	512

B Layer-wise Heterogeneous Attention

Here we try another method of heterogeneous attention. Instead of using layers with an identical mixture of attentions at each layer, we make each layer homogeneous in its attention but use different attentions for different layers.

Given we want each layer to have H attention heads of a single type, we initialize an attention block with H heads for each candidate attention mechanism in each layer. We train the supernetwork to rough convergence and then alternate between removing an attention block in a layer and fine-tune training the remaining model. We go through each layer starting with the first, removing the worst attention block before moving on to the next layer. Once we remove a block from the final layer we begin again at the first layer. We keep passing through the network, removing and fine-tuning, until only one attention block remains in each layer.

We then take the attention blocks learned for each layer and train a new Transformer from scratch with those attentions at each layer. This paradigm is summarized in algorithm form in the appendix as Algorithm 3.

Due to the high memory requirements of this task, we use block sampling when training. This means that for each training batch we only use a randomly selected subset of the candidate blocks in each layer. This reduces memory requirements as there are less intermediate values and computations, but can affect the training dynamics. Using this method within a DARTS like paradigm is shown to work well by Xu et al. (2019).

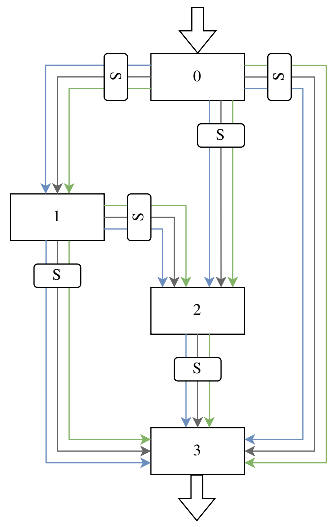
We test this method on the text classification task which has 1k input sequence length to further limit computation requirements. As the Transformers used in LRA for this task had 6 layers, we searched for a 6 layer model with layer-wise attention heterogeneity. The resulting architecture was:

Performer $\rightarrow$ Bigbird $\rightarrow$ Performer $\rightarrow$ Sparse $\rightarrow$ Performer $\rightarrow$ Sparse

When trained with the same hyperparameters as the LRA models, it achieved a test accuracy of 64.6%. This puts this model around average for this task, unable to beat several attentions such as the Linear Transformer which had a test accuracy of 65.9%, or the Performer which achieved 65.4%.

C Supplementary Diagrams

Figures 2 to 4 help illustrate how our approach relates to DARTS and the original Transformer, attention heads being pruned during the search process in finding optimal heterogeneous attention, and an overview of the differences between the theoretical and actual supernetwork.



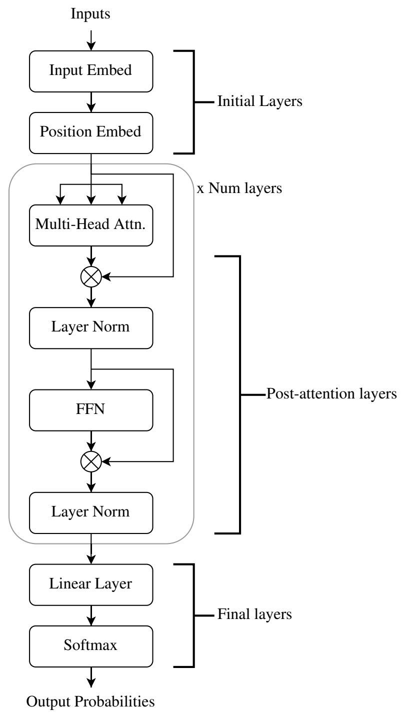


Figure 2: DARTS Cell (left) and typical Transformer architecture (right). In DARTS inputs to all blocks are summed and 'S' denotes a softmax layer. Also shown is how the layers in the Transformer relate to the diagram in figure 1.

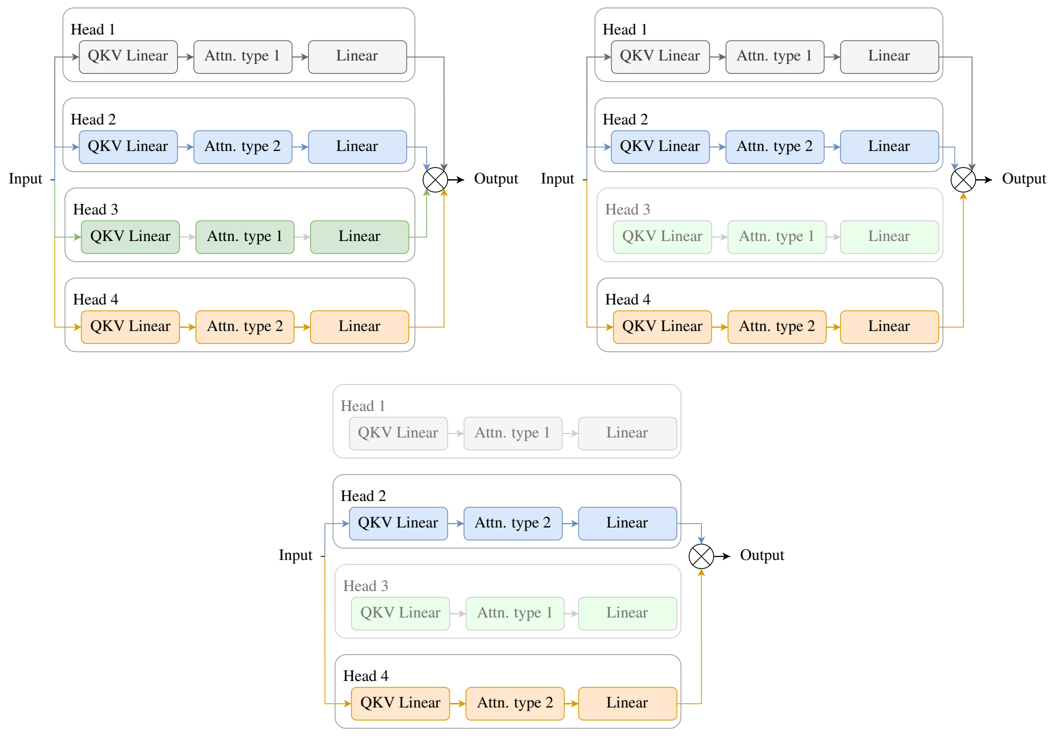
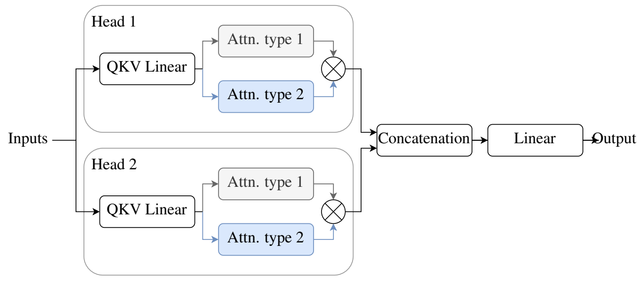
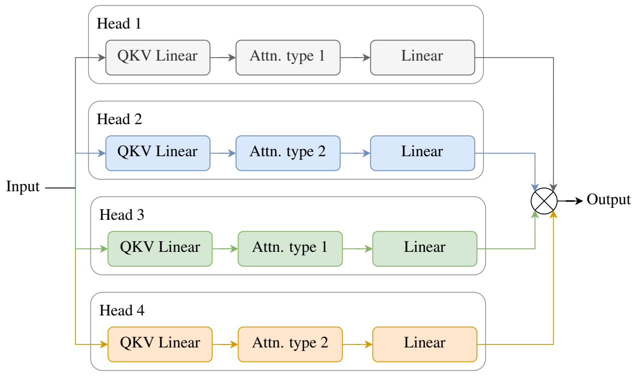


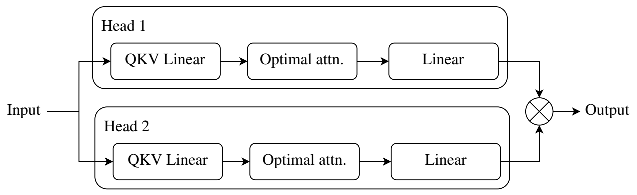
Figure 3: From left to right and top to bottom this figure shows how we remove from the model the worst attention block until we are left with just the best ones.



(a) Theoretical multi head attention supernetwork



(b) Actual multi head attention supernetwork



(c) Final multi attention layer after architecture search

Figure 4: Diagrams of the multi head attention layer with the theoretical attention search space and the actual one used. Also shown is the resulting multi head attention architecture which would be equivalent in both cases.

D Algorithms

Algorithms 1 and 2 correspond to the paradigms outlined in section 3.3.

Let $\mathbb{B}_i$ denote the set of attention blocks in layer i, with $\mathbb{L}$ denoting the set of layers. We also use a to represent accuracy and s to represent a score with $\mathbb{S}_i$ representing the set of scores for the attention blocks in layer i. For single layer networks, the layer subscript is omitted.

Algorithm 1 Finding optimal homogeneous attention

```
 1: Initialise a single layer supernetwork with a multi-head attention block for each attention type
 2: Train supernetwork until convergence
 3: $a_{base} \leftarrow$ Validation accuracy of supernetwork
 4: for $b_i \in \mathbb{B}$ do
 5:     Mask out $b_i$
 6:     $a_i \leftarrow$ Validation accuracy
 7:     $s_i \leftarrow a_{base} - a_i$
 8: end for
 9: Selected attention mechanism $\leftarrow \max(\mathbb{S})$
10: Initialise full Transformer using the selected attention mechanism
11: Train until convergence
```

Algorithm 2 Finding optimal heterogeneous attention

```
 1: Initialise a single layer super network with $H$ single-head attention blocks for each attention
    type, where $H$ is the desired number of heads per layer in the final network.
 2: Train supernetwork until rough convergence
 3: repeat
 4:     $a_{base} \leftarrow$ Validation accuracy of supernetwork
 5:     for $b_i \in \mathbb{B}$ do
 6:         Mask out $b_i$
 7:         $a_i \leftarrow$ Validation accuracy
 8:         $s_i \leftarrow a_{base} - a_i$
 9:     end for
10:     if $|\mathbb{B}| > H$ then
11:         From $\mathbb{B}$ remove block corresponding to $\min(\mathbb{S})$
12:         Train remaining supernetwork a small amount
13:     end if
14: until $|\mathbb{B}| = H$
15: $\mathbb{B}_{optimal} \leftarrow \mathbb{B}$
16: Initialise a new network with multiple layers, where each layer has blocks $\mathbb{B}_{optimal}$
17: Train until convergence
```

Algorithm 3 Head-wise homogeneous, layer-wise heterogeneous mixed attention search

```
1: Initialise a multi layer supernetwork with multi-head attention block for each attention type in
   each layer
2: Train supernetwork until rough convergence
3: repeat
4:     for $i \in \mathbb{L}$ do
5:         $a_{base} \leftarrow$ Validation accuracy of supernetwork
6:         for $b_{ij} \in \mathbb{B}_i$ do
7:             Mask out $b_{ij}$
8:             $a_{ij} \leftarrow$ Validation accuracy
9:             $s_{ij} \leftarrow a_{base} - a_{ij}$
10:        end for
11:        if $|\mathbb{B}_i| > 1$ then
12:            From $\mathbb{B}_i$ remove block corresponding to $\min(\mathbb{S}_i)$
13:            Train remaining supernetwork a small amount
14:        end if
15:    end for
16: until $|\mathbb{B}_i| = 1, \forall i \in \mathbb{L}$
17: Initialise a full Transformer model where the attention mechanism used in layer $i$ corresponds to
    the type of the last block left in $\mathbb{B}_i$
18: Train until convergence
```