Pay Attention to MLPs

Hanxiao Liu, Zihang Dai, David R. So, Quoc V. Le
Google Research, Brain Team
{hanxiaol,zihangd,davidso,qvl}@google.com

Abstract

Transformers [1] have become one of the most important architectural innovations in deep learning and have enabled many breakthroughs over the past few years. Here we propose a simple attention-free network architecture, gMLP, based solely on MLPs with gating, and show that it can perform as well as Transformers in key language and vision applications. Our comparisons show that self-attention is not critical for Vision Transformers, as gMLP can achieve the same accuracy. For BERT, our model achieves parity with Transformers on pretraining perplexity and is better on some downstream tasks. On finetuning tasks where gMLP performs worse, making the gMLP model substantially larger can close the gap with Transformers. In general, our experiments show that gMLP can scale as well as Transformers over increased data and compute.

1 Introduction

Transformers [1] have enabled many breakthroughs in natural language processing (e.g., [2, 3, 4, 5, 6]) and have been shown to work well for computer vision (e.g., [7, 8, 9, 10]). Thanks to this success, Transformers have largely replaced LSTM-RNN [11] as the default architecture in NLP, and have become an appealing alternative to ConvNets [12, 13, 14, 15, 16] in computer vision.

The Transformer architecture combines two important concepts: (1) a recurrent-free architecture which computes the representations for each individual token in parallel, and (2) multi-head self-attention blocks which aggregate spatial information across tokens. On one hand, the attention mechanism [17] introduces the inductive bias that the model can be dynamically parameterized based on the input representations. On the other hand, it is known that MLPs with static parameterization can represent arbitrary functions [18]. It therefore remains an open question whether the inductive bias in self-attention is essential to the remarkable effectiveness of Transformers.

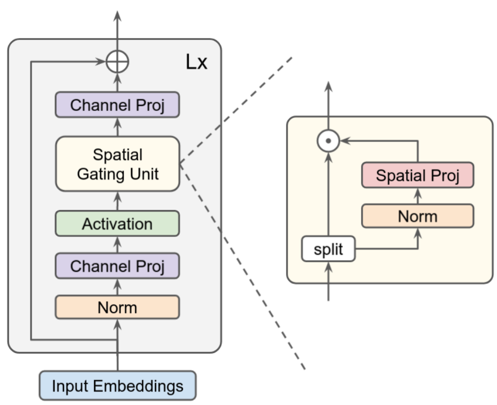
Here we study the necessity of self-attention modules in key language and vision applications of Transformers and propose an attention-free, MLP-based alternative to Transformers, which consists of channel projections, spatial projections and gating (Figure 1). We experiment with several design choices for the MLP-like architecture and find spatial projections work well when they are linear and paired with multiplicative gating. We name the model *gMLP* because it is built out of basic MLP layers with gating.

We apply gMLP to image classification and obtain strong results on ImageNet. gMLP achieves comparable performance with DeiT [8], namely Vision Transformer (ViT) [7] enhanced with improved regularization, in a similar training setup. With 66% less parameters, a gMLP model is 3% more accurate than MLP-Mixer [19]. Together with Tolstikhin et al. [19], Melas-Kyriazi [20] and Touvron et al. [21], our results question the necessity of self-attention layers in Vision Transformers.

We apply gMLP to masked language modeling (MLM) in the BERT [2] setup, one of the most well-established applications of Transformers, and find that it is as good as Transformers at minimizing perplexity during pretraining. Our experiments indicate that perplexity is only correlated with model capacity and is insensitive to the presence of attention. As capacity increases, we observe that both pretraining and finetuning metrics for gMLPs improve as quickly as for Transformers. This is remarkable because it indicates gMLPs scale just as well as Transformers despite the absence of self-attention, and any performance gap can always be offset by training a larger model with increased data and compute. With a standard 256-batch size $\times$ 1M-step training setup as in original BERT, our MLP-like model achieves 86.4% accuracy on MNLI and 89.5% F1 on SQuAD v1.1. Note, these are comparable to the results reported in Devlin et al. [2] obtained using Transformers.

For BERT's finetuning, Transformers can be more practically advantageous over gMLPs on tasks that require cross-sentence alignment (e.g., by 1.8% on MNLI), even with similar capacity and pretraining perplexity. This problem can be addressed by making gMLPs substantially larger — $3\times$ as large as Transformers. A more practical solution is to blend in only a tiny bit of attention—a single-head attention with size up to 128 is sufficient to make gMLPs outperform Transformers on all NLP tasks we evaluated with even better parameter efficiency. The improvement is sometimes very significant (e.g., +4.4% on SQuAD v2.0 over $\text{BERT}_{\text{large}}$ in our experiments).

The effectiveness of gMLPs, the lack of benefits of self-attention in vision and the case-dependent benefits of attention in NLP, call into question the necessity of attention across various domains. Overall, our results suggest that self-attention is not a necessary ingredient for scaling up machine learning models. With increased data and compute, models with simple spatial interaction mechanisms such as gMLP can be as powerful as Transformers and the capacity allocated to self-attention can be either removed or substantially reduced.



Pseudo-code for the gMLP block

```
def gmlp_block(x, d_model, d_ffn):
  shortcut = x
  x = norm(x, axis="channel")
  x = proj(x, d_ffn, axis="channel")
  x = gelu(x)
  x = spatial_gating_unit(x)
  x = proj(x, d_model, axis="channel")
  return x + shortcut

def spatial_gating_unit(x):
  u, v = split(x, axis="channel")
  v = norm(v, axis="channel")
  n = get_dim(v, axis="spatial")
  v = proj(v, n, axis="spatial", init_bias=1)
  return u * v
```

Figure 1: Overview of the gMLP architecture with Spatial Gating Unit (SGU). The model consists of a stack of L blocks with identical structure and size. In each block, "$\odot$" refers to element-wise multiplication and all projection operations are linear. The input/output format follows BERT (for NLP pretraining & finetuning) and ViT (for vision). Unlike Transformers, gMLPs do not require positional encodings, nor is it necessary to mask out the paddings during NLP finetuning.

2 Model

Our model, gMLP, consists of a stack of L blocks with identical size and structure. Let $X \in \mathbb{R}^{n\times d}$ be the token representations with sequence length n and dimension d. Each block is defined as:

$$Z = \sigma(XU) \tag{1}$$

$$\tilde{Z} = s(Z) \tag{2}$$

$$Y = \tilde{Z}V \tag{3}$$

where σ is an activation function such as GeLU [22]. U and V define linear projections along the channel dimension — the same as those in the FFNs of Transformers (e.g., their shapes are 768×3072 and 3072×768 for $\text{BERT}_{\text{base}}$). Shortcuts, normalizations and biases are omitted for brevity.

A key ingredient in the aforementioned formulation is $s(\cdot)$, a layer which captures spatial interactions (see below). When s is an identity mapping, the above transformation degenerates to a regular FFN, where individual tokens are processed independently without any cross-token communication. One of our major focuses is therefore to design a good s capable of capturing complex spatial interactions across tokens. The overall block layout is inspired by inverted bottlenecks [23] which define $s(\cdot)$ as a spatial depthwise convolution. Note, unlike Transformers, our model *does not require position embeddings* because such information will be captured in $s(\cdot)$.

Our model uses exactly the same input and output format as BERT (for NLP) and ViT (for vision). For example, when finetuning on language tasks, we concatenate together multiple segments followed by paddings, and the predictions are deduced from the last-layer representation of a reserved `<cls>` symbol. Although many of these protocols were introduced for Transformers and hence can be suboptimal for gMLPs, strictly following them helps avoid confounding factors in our experiments and makes our layers more compatible with existing Transformer implementations.

2.1 Spatial Gating Unit

To enable cross-token interactions, it is necessary for the layer $s(\cdot)$ to contain a contraction operation over the spatial dimension. The simplistic option would be a linear projection:

$$f_{W,b}(Z) = WZ + b \tag{4}$$

where $W \in \mathbb{R}^{n\times n}$ is a matrix for which the size is the same as the sequence length, n, and b refers to a bias term which can either be a matrix or a scalar. For example, if the input sequence has 128 tokens, the shape for the spatial projection matrix W will be 128×128. In this work, we define the spatial interaction unit as the multiplication of its input and the spatially transformed input:

$$s(Z) = Z \odot f_{W,b}(Z) \tag{5}$$

where $\odot$ denotes element-wise multiplication. For training stability, we find it critical to initialize W as near-zero values and b as ones, meaning that $s(\cdot)$ defined in Equation (5) is approximately an identity mapping at the beginning of training. This initialization ensures each gMLP block behaves like a regular FFN at the early stage of training, where each token is processed independently, and only gradually injects spatial information across tokens.

The multiplicative gating can be viewed as a mechanism to "modulate" individual token representations using the spatial signal. In other words, the magnitude for each element in Z can be rapidly tuned according to the gating function $f_{W,b}(\cdot)$.

We further find it effective to split Z into two independent parts (Z_1, Z_2) along the channel dimension for the gating function and for the multiplicative bypass, as is typically done in GLUs:

$$s(Z) = Z_1 \odot f_{W,b}(Z_2) \tag{6}$$

We also normalize the input to $f_{W,b}$ which empirically improves stability of large NLP models. This gives us the unit illustrated in Figure 1, which we refer to as the *Spatial Gating Unit* (SGU) in the rest of the paper. In Table 3, we provide ablation studies to compare SGU with several other variants of $s(\cdot)$, showing that it works better and narrows the performance gap with self-attention.

Remark. The overall formulation of SGU is closely related to Gated Linear Units (GLUs) [24, 25, 26]. A key distinction is that our gating is computed based on the spatial (cross-token) dimension rather than the channel (per-token) dimension. It also resembles Squeeze-and-Excite blocks [27] in terms of element-wise multiplicative interaction, but instead of doing pooling, SGU allows learnable spatial transformations. The spatial projection in SGU could learn to express superficial depthwise convolutions—unlike typical depthwise convolutions with channel-specific filters, SGU learns only a single transformation shared across channels. Finally, we note SGUs offer an alternative way to capture high-order relationships other than self-attention. Specifically, the output for Equation (5) contains up to 2nd-order interactions (e.g., z_iz_j) and the output for self-attention (assuming no nonlinearity) contains up to 3rd-order interactions (e.g., $z_iz_jz_k$). In terms of computation cost, SGU has $n^2e/2$ multiply-adds which is comparable to the $2n^2d$ of dot-product attention.[1] Both are linear over the input channel size and quadratic over the sequence length n.

3 Image Classification

Here we examine gMLP in the vision domain by applying it to the image classification task on ImageNet without extra data. We compare our attention-free models with recent attentive models

[1]The input channel size e for SGU is typically larger than the input channel size d for self-attention, because the former is applied in the middle of the block after a channel expansion.

based on vanilla Transformers, including Vision Transformer (ViT) [7], DeiT [8] (ViT with improved regularization), and several other representative convolutional networks.

Table 1 summarizes the configurations of our gMLP image classification models. The input and output protocols follow ViT/B16 where the raw image is converted into 16×16 patches at the stem. The depth and width are chosen so that the models are comparable with ViT/DeiT in capacity. Like Transformers, we find gMLPs tend to drastically overfit the training data. We therefore apply a similar regularization recipe as the one used in DeiT.[2] To avoid extensive tuning, we adjust only the strengths of stochastic depth [28] as we move from smaller to larger models in Table 1. All the other hyperparameters remain shared across our three models. See Appendix A.1 for details.

	#L	d_{model}	d_{ffn}	Params (M)	FLOPs (B)	Survival Prob
gMLP-Ti	30	128	768	5.9	2.7	0.99
gMLP-S	30	256	1536	19.5	8.9	0.95
gMLP-B	30	512	3072	73.4	31.6	0.90

Table 1: Architecture specifications of gMLP models for vision. The survival probability of stochastic depth is the only hyperparameter change as we move from smaller to larger models.

Our ImageNet results are summarized in Table 1 and Figure 2. It is interesting to see that gMLPs are comparable with DeiT [8], namely ViT [7] trained using improved regularization. The results suggest that attention-free models can be as data-efficient as Transformers for image classification. In fact, when the models are properly regularized, their accuracies seem better correlated with capacity instead of the presence of attention. Moreover, the accuracy-parameter/FLOPs tradeoff of gMLPs surpasses all concurrently proposed MLP-like architectures [19, 20, 21], which we attribute to the effectiveness of our Spatial Gating Unit (see Table 3 in the next section for an ablation). We also note while gMLPs are competitive with vanilla Transformers, their performance is behind the best existing ConvNet models (e.g., [29, 30]) or hybrid attentive models (e.g., [31, 10, 32, 33]).

Table 2: ImageNet-1K results without extra data.

Model	ImageNet Top-1 (%)	Input Resolution	Params (M)	MAdds (B)
		ConvNets		
ResNet-152 [15]	78.3	224	60	11.3
RegNetY-8GF [34]	81.7	224	39	8.0
EfficientNet-B0 [16]	77.1	224	5	0.39
EfficientNet-B3 [16]	81.6	300	12	1.8
EfficientNet-B7 [16]	84.3	600	66	37.0
NFNet-F0 [29]	83.6	192	72	12.4
		Transformers		
ViT-B/16 [7]	77.9	384	86	55.4
ViT-L/16 [7]	76.5	384	307	190.7
DeiT-Ti [8] (ViT+reg)	72.2	224	5	1.3
DeiT-S [8] (ViT+reg)	79.8	224	22	4.6
DeiT-B [8] (ViT+reg)	81.8	224	86	17.5
		MLP-like†		
Mixer-B/16 [19]	76.4	224	59	12.7
Mixer-L/16 [19]	71.8	224	207	44.8
ResMLP-12 [21]	76.6	224	15	3.0
ResMLP-24 [21]	79.4	224	30	6.0
ResMLP-36 [21]	79.7	224	45	8.9
gMLP-Ti (ours)	72.0	224	6	1.4
gMLP-S (ours)	79.4	224	20	4.5
gMLP-B (ours)	81.6	224	73	15.8

† Tokenization & embedding process at the stem can be viewed as a convolution.

[2] Unlike DeiT, we do not use repeated augmentation or random erasing.

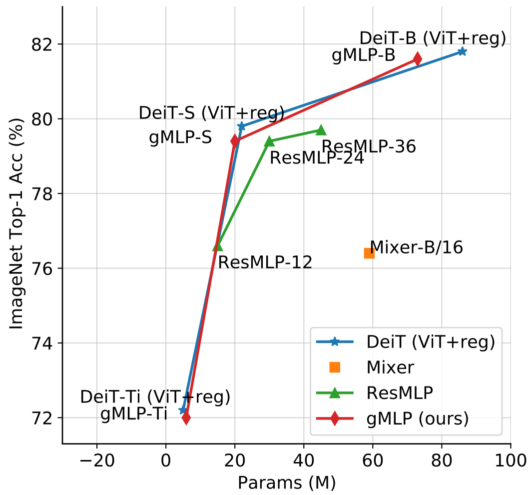


Figure 2: ImageNet accuracy vs model capacity.

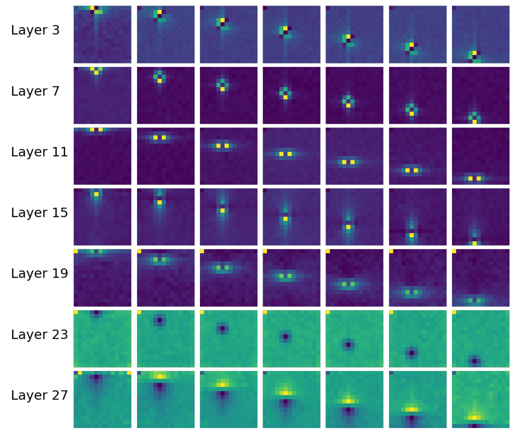


Figure 3: Spatial projection weights in gMLP-B. Each row shows the filters (reshaped into 2D) for a selected set of tokens in the same layer.

Figure 3 visualizes the spatial projection matrices in gMLP-B. Remarkably, the spatial weights after learning exhibit both locality and spatial invariance. In other words, each spatial projection matrix effectively learns to perform convolution with a data-driven, irregular (non-square) kernel shape.

4 Masked Language Modeling with BERT

Here we conduct empirical studies over the masked language modeling (MLM) task. The input/output format for both pretraining and finetuning follows BERT [2]. Different from Transformer-based models, we do not use positional encodings. We also find it unnecessary to mask out `<pad>` tokens in gMLP blocks during finetuning as the model can quickly learn to ignore them. For ablations and case studies, all models are trained with batch size 2048, max length 128 for 125K steps over the RealNews-like subset of C4 [5]. For main results, models are trained with batch size 256, max length 512 for 1M steps over the full English C4 dataset. See Appendix A.2 for details.

For MLM tasks, shift invariance is a desired property because any offset of the input sequence should not affect the slot filling outcome. This property implies a Toeplitz spatial weight matrix W. We adopt this constraint in our MLM experiments because it reduces model parameters and empirically has negligible impact on quality[3] or efficiency. In this case, $f_{W,b}(\cdot)$ is analogous to a wide depthwise convolution whose receptive field covers the entire sequence. However, unlike depthwise convolutions which learn channel-specific filters, we learn only a single W shared across channels.

4.1 Ablation: The Importance of Gating in gMLP for BERT's Pretraining

In Table 3 below, we establish baselines for our ablation studies. These include:

1. BERT with a Transformer architecture and learnable absolute position embeddings.
2. BERT with a Transformer architecture and T5-style learnable relative position biases [5]. The biases are both layer- and head-specific as we find this yields the best results.
3. Same as above, but we remove all content-dependent terms inside the softmax and only retain the relative positional biases. This baseline is interesting because it is a straightforward attention-free variant of Transformers, which can also be viewed as a Synthesizer [35].

We compare these baselines against several versions of gMLPs with similar sizes in Table 3. Note that Multiplicative, Split (last row) is the Spatial Gating Unit we describe in the method section and use in the rest of the paper. First, SGU outperforms other variants in perplexity. Secondly and remarkably, gMLP with SGU also achieves perplexity comparable to Transformer. Note the difference between

[3] This is because gMLP will learn shift invariance even without this constraint (Figure 9 in Appendix C).

Model	Perplexity	Params (M)
$\text{BERT}_{\text{base}}$	4.37	110
$\text{BERT}_{\text{base}}$ + rel pos	4.26	110
$\text{BERT}_{\text{base}}$ + rel pos - attn	5.64	96
Linear gMLP, $s(Z) = f(Z)$	5.14	92
Additive gMLP, $s(Z) = Z + f(Z)$	4.97	92
Multiplicative gMLP, $s(Z) = Z \odot f(Z)$	4.53	92
Multiplicative, Split gMLP, $s(Z) = Z_1 \odot f(Z_2)$, $Z = Z_1 \| Z_2$	4.35	102

Table 3: MLM validation perplexities of Transformer baselines and four versions of gMLPs. f refers to the spatial linear projection in Equation (4) with input normalization. Each gMLP model contains L=36 layers with d_{model}=512 and d_{ffn} = 3072. No positional encodings are used for gMLPs.

the strongest baseline (perplexity=4.26) and ours (perplexity=4.35) is insignificant relative to the perplexity change when the models are scaled (see Table 4 in the next section). Spatial projection weights learned by gMLPs are visualized in Figure 4.

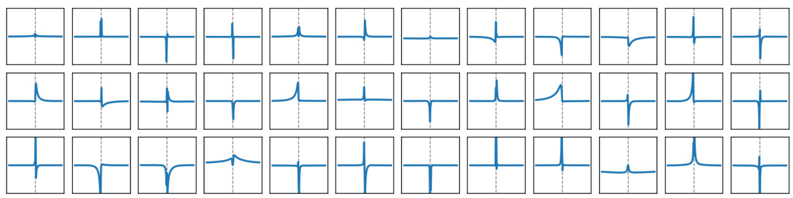


Figure 4: Visualization of the spatial filters in gMLP learned on the MLM task. For each layer in the model we plot the row in W associated with the token in the middle of the sequence. The x-axis of each subplot has a length of 128 which equal the number of tokens in the sequence. The learned filters appear to be smooth and have several types: forward-looking (e.g., 1st in 2nd row), backward-looking (e.g., 5th in 2nd row) and bi-directional (e.g., 2nd last in the last row).

4.2 Case Study: The Behavior of gMLP as Model Size Increases

In Table 4, we investigate the scaling properties of Transformers and gMLPs in BERT as their model capacity grows. Specifically, we scale the depth of these models by a factor of $\{0.5, 1, 2, 4\}\times$ and report the their pretraining MLM perplexities on the validation set as well as finetuning results on the dev sets of two tasks in GLUE [36]. Note each individual Transformer layer is effectively two consecutive blocks: one for self-attention and one for FFN. In the table below we use the notation of 12 + 12 to refer to 12 of attention blocks plus 12 of FFN blocks in the Transformer baselines.

Model	#L	Params (M)	Perplexity	SST-2	MNLI-m
Transformer	6+6	67	**4.91**	90.4	81.5
gMLP	18	59	5.25	91.2	77.7
Transformer	12+12	110	**4.26**	91.3	83.3
gMLP	36	102	4.35	92.3	80.9
Transformer	24+24	195	3.83	92.1	85.2
gMLP	72	187	**3.79**	93.5	82.8
Transformer	48+48	365	3.47	92.8	86.3
gMLP	144	357	**3.43**	95.1	84.6

Table 4: Pretraining and dev-set finetuning results over increased model capacity. We use the relative positional encoding scheme for Transformers which performs the best in Table 3.

The results above show that a deep enough gMLP is able to match and even outperform the perplexity of Transformers with comparable capacity.[4] In addition, the perplexity-parameter relationships for

[4]We also experimented with deeper-and-thinner Transformers (with capacity fixed) but found increasing depth further does not improve perplexity. See Appendix B for more details.

both architecture families approximately follow a power law (left of Figure 5). This implies the empirical scaling laws originally observed for Transformer-based language models [37] might be broadly applicable across different model families.

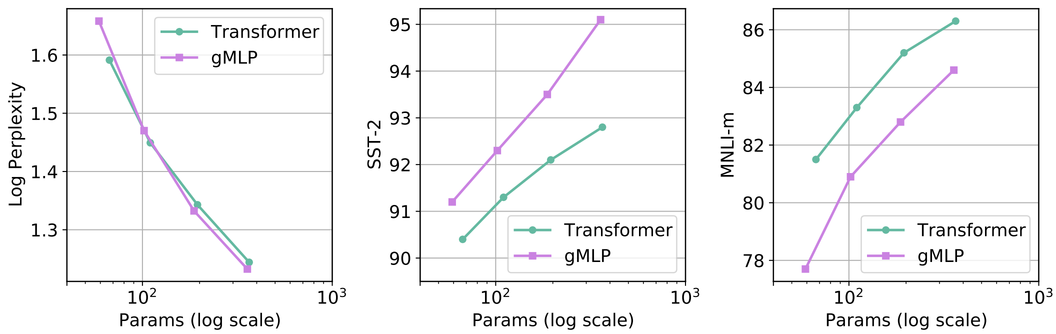


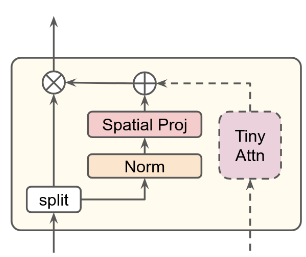
Figure 5: Scaling properties with respect to perplexity and finetuning accuracies. The figures show that for pretraining, gMLPs are equally good at optimizing perplexity as Transformers. For finetuning, the two model families exhibit comparable scalability despite task-specific offsets.

Table 4 also leads to an interesting observation that the pretraining perplexities across different model families are not equal in terms of finetuning. While gMLPs outperform Transformers on SST-2, they are worse on MNLI. The results imply that the finetuning performance for NLP tasks is a function of not only the perplexity but also the inductive bias in the architecture. Figure 5 shows that despite the architecture-specific discrepancies between pretraining and finetuning, gMLPs and Transformers exhibit comparable scalability (slope) on both finetuning tasks. This means one can always offset the gap by enlarging the model capacity. In other words, the results indicate that model scalability with respect to downstream metrics can be independent from the presence of self-attention.

4.3 Ablation: The Usefulness of Tiny Attention in BERT's Finetuning

So far we have found that self-attention is not a required component to achieve strong MLM perplexity or scalability. At the meantime, we also identified NLP finetuning tasks where gMLPs transfer less well than Transformers (Table 4). The fact that our attention-free model is advantageous on SST-2 but worse on MNLI is particularly informative—the former is a single-sentence task whereas the latter involves sentence pairs (premise and hypothesis) [38]. We suspect the role of self-attention during finetuning is related to cross-sentence alignment.

To isolate the effect of attention, we experiment with a hybrid model where a tiny self-attention block is attached to the gating function of gMLP (Figure 6). Since gMLP itself is already capable in capturing spatial relationships, we hypothesize that this extra attention module does not have to be heavy, and that its presence is more relevant than its capacity. A typical tiny attention module in our experiments has only a single head with size 64, significantly smaller than a typical multi-head attention in Transformers with 12 heads and a total size of 768. In the following, we refer to the hybrid model, namely gMLP with a tiny attention, as *aMLP* ("a" for attention).



Pseudo-code for the tiny attention module

```
def tiny_attn(x, d_ffn, d_attn=64):
  qkv = proj(x, 3 * d_attn, axis="channel")
  q, k, v = split(qkv, 3, axis="channel")
  w = einsum("bnd,bmd->bnm", q, k)
  a = softmax(w * rsqrt(d_attn))
  x = einsum("bnm,bmd->bnd", a, v)
  return proj(x, d_ffn, axis="channel")
```

Figure 6: Hybrid spatial gating unit with a tiny self-attention module. We use the input of the gMLP block (after the first normalization) as the input to the tiny attention module.

In Figure 7, we investigate the transferability of MLM models via the calibration plots between their pretraining perplexities and finetuning metrics. Models evaluated include $\text{BERT}_{\text{base}}$, gMLP and its hybrid version aMLP with a 64-d single head-attention (Figure 6). The data points were collected by varying the model depth by $\{0.5, 1, 2\}\times$ or data by $\{1, 2, 4, 8\}\times$. It can be seen that gMLPs transfer better to SST-2 than Transformers regardless of the presence of attention, While gMLP performs worse on MNLI, attaching a tiny bit of attention is sufficient to close the gap. In Appendix D we visualize the tiny attention modules in aMLP over MNLI examples, showing that they are primarily responsible for the alignment between sentence pairs.

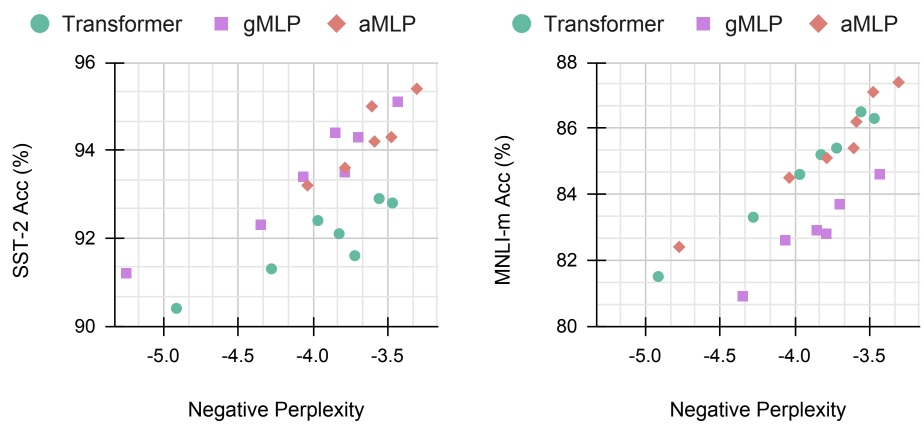


Figure 7: Transferability from MLM pretraining perpexity to finetuning accuracies on GLUE. aMLP refers to gMLP enhanced with a 64-d single-head self-attention, as illustrated in Figure 6. In contrast, each self-attention module in the BERT baseline contains 12 heads with a total size of 768.

In Figure 8 we put together the scaling properties of the three models, showing that aMLP (gMLP + tiny attention) consistently outperforms Transformer on both finetuning tasks.

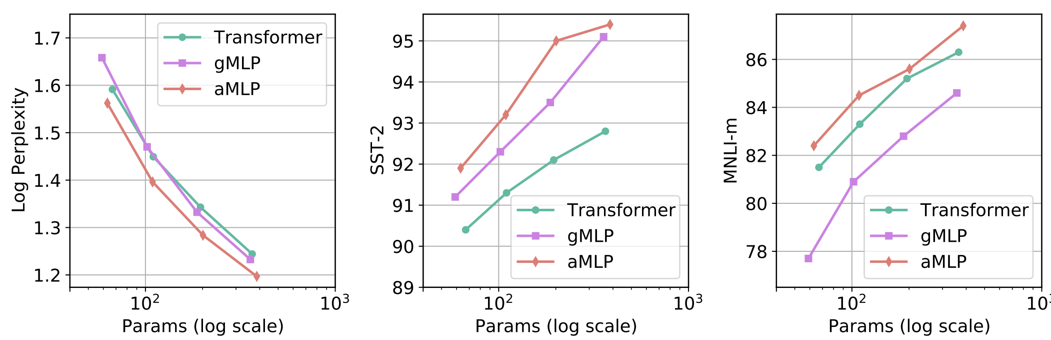


Figure 8: Comparing the scaling properties of Transformers, gMLPs and aMLPs (with 64-d, single head attention). Results were obtained using the same setup in Section 4.2.

4.4 Main Results for MLM in the BERT Setup

Below we present pretraining and finetuning results in the full BERT setup. Different from ablation and case studies, here we use the full English C4 dataset and adopt a common MLM setup with batch size 256, max length 512 and 1M training steps. For fair comparison, we adjust the depth and width of gMLPs to ensure comparable model capacity with the Transformer baselines. The model specifications are given in Table 5 and details of our hyperparameters are provided in Appendix A.2. For finetuning, we report the dev-set performance for SST-2 and MNLI in GLUE [36] and each result entry was obtained by taking the median of five independent runs. In addition, we report finetuning results on SQuAD [39, 40] to test the models' ability in reasoning over a longer context.

Results are presented in Table 6. Consistent with our findings earlier in Section 4.1 and Section 4.2, gMLPs are competitive with Transformers in terms of perplexity, especially in the larger scale setup. There are several observations related to the finetuning results:

First, on finetuning tasks where gMLPs underperform Transformers, the performance gap tends to narrow as the model capacity increases. For example, while gMLP performs worse by 8.5% on

	Params (M)	FLOPs (B)	#L	d_{model}	d_{ffn}
$\text{BERT}_{\text{base}}$	110	100.8	12+12	768	3072
$\text{gMLP}_{\text{base}}$	130	158.0	48	512	3072
$\text{aMLP}_{\text{base}}$	109	128.9	36	512	3072
$\text{BERT}_{\text{large}}$	336	341.2	24+24	1024	4096
$\text{gMLP}_{\text{large}}$	365	430.1	96	768	3072
$\text{aMLP}_{\text{large}}$	316	370.3	72	768	3072

Table 5: Model specifications in the full BERT setup.

	Perplexity	SST-2	MNLI (m/mm)	SQuAD v1.1	SQuAD v2.0	Attn Size	Params (M)
$\text{BERT}_{\text{base}}$ [2]	–	92.7	84.4/-	88.5	76.3	768 (64×12)	110
$\text{BERT}_{\text{base}}$ (ours)	4.17	93.8	85.6/85.7	90.2	78.6	768 (64×12)	110
$\text{gMLP}_{\text{base}}$	4.28	94.2	83.7/84.1	86.7	70.1	–	130
$\text{aMLP}_{\text{base}}$	3.95	93.4	85.9/85.8	90.7	80.9	64	109
$\text{BERT}_{\text{large}}$ [2]	–	93.7	86.6/-	90.9	81.8	1024 (64×16)	336
$\text{BERT}_{\text{large}}$ (ours)	3.35	94.3	87.0/87.4	92.0	81.0	1024 (64×16)	336
$\text{gMLP}_{\text{large}}$	3.32	94.8	86.2/86.5	89.5	78.3	–	365
$\text{aMLP}_{\text{large}}$	3.19	94.8	88.4/88.4	92.2	85.4	128	316

Table 6: Pretraining perplexities and dev-set results for finetuning. "ours" indicates models trained using our setup. We report accuracies for SST-2 and MNLI, and F1 scores for SQuAD v1.1/2.0.

SQuAD-v2.0 in the base scale, the performance gap relative to the baseline decreases to 2.7% at the larger scale. Notably, our $\text{gMLP}_{\text{large}}$ achieves 89.5% F1 on SQuAD-v1.1 without any attention or dynamic parameterization mechanism [26], which is well above the 88.5% reported for $\text{BERT}_{\text{base}}$ in Devlin et al. [2] and is only 1.4% away from the original result for $\text{BERT}_{\text{large}}$. While this is clearly not a fair comparison due to different training settings, it can be viewed as as an existence proof that attention-free, MLP-like models can be competitive on challenging downstream NLP tasks.

Furthermore, we show that blending in a tiny single-head attention of size either 64 or 128 is sufficient to make gMLPs outperform Transformers of similar capacity, sometimes by a significant margin. For example, our hybrid model $\text{aMLP}_{\text{large}}$ achieves 4.4% higher F1 than Transformers on the more difficult SQuAD-v2.0 task. The results suggest that the capacity in the multi-head self-attention of Transformers can be largely redundant, and that the majority of its functionalities can be captured by the spatial gating unit in gMLPs. The results also imply that the inductive biases in the spatial gating unit of gMLPs and the tiny attention are complementary to each other. While the benefits of architectural inductive bias may vanish over increased compute, tiny attention does improve the practical value of gMLPs in the regime that we investigate in this work.

5 Conclusion

Since the seminal work of Vaswani et al. [1], Transformers have been widely adopted across NLP and computer vision. This adoption has enabled many impressive results especially in NLP. To date, it is still unclear what empowers such success: is it the feedforward nature of Transformers or is it the multi-head self-attention layers in Transformers?

Our work studies this question in depth and suggests that we generally don't need much attention. We show that gMLPs, a simple variant of MLPs with gating, can be competitive with Transformers in terms of BERT's pretraining perplexity and ViT's accuracy. gMLPs are also comparable with Transformers in terms of the scalability over increased data and compute. As for BERT finetuning, we find gMLPs can achieve appealing results on challenging tasks such as SQuAD without attention, and can significantly outperform Transformers in certain cases. We also find the inductive bias in

Transformer's multi-head self-attention useful on downstream tasks that require cross-sentence alignment. However in those cases, making gMLP substantially larger closes the gap with Transformers. More practically, blending a tiny bit of single-head attention into gMLP allows for an even better architecture without the need for increasing model size.

Acknowledgements

We thank Gabriel Bender, Neil Houlsby, Thang Luong, Niki Parmar, Hieu Pham, Noam Shazeer, Ilya Sutskever, Jakob Uszkoreit and Ashish Vaswani for their feedback to the paper.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2018.

[3] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, 2019.

[4] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[5] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 2020.

[6] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[8] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.

[9] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

[10] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

[12] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1989.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 2012.

[14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[16] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019.

[17] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[18] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[19] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021.

[20] Luke Melas-Kyriazi. Do you even need attention? a stack of feed-forward layers does surprisingly well on imagenet. *arXiv preprint arXiv:2105.02723*, 2021.

[21] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, and Hervé Jégou. Resmlp: Feedforward networks for image classification with data-efficient training. *arXiv preprint arXiv:2105.03404*, 2021.

[22] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[23] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

[24] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *ICML*, 2017.

[25] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[26] Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. In *ICLR*, 2019.

[27] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.

[28] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *ECCV*, 2016.

[29] Andrew Brock, Soham De, Samuel L Smith, and Karen Simonyan. High-performance large-scale image recognition without normalization. *arXiv preprint arXiv:2102.06171*, 2021.

[30] Mingxing Tan and Quoc V Le. Efficientnetv2: Smaller models and faster training. In *ICML*, 2021.

[31] Irwan Bello. Lambdanetworks: Modeling long-range interactions without attention. In *ICLR*, 2021.

[32] Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. Scaling local self-attention for parameter efficient visual backbones. In *CVPR*, 2021.

[33] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. *arXiv preprint arXiv:2101.11605*, 2021.

[34] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *CVPR*, 2020.

[35] Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention in transformer models. *arXiv preprint arXiv:2005.00743*, 2020.

[36] 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. In *ICLR*, 2019.

[37] 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.

[38] Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*, 2017.

[39] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*, 2016.

[40] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. In *ACL*, 2018.

A Hyperparameters

A.1 Image Classification

	gMLP-Ti	gMLP-S	gMLP-B
Stochastic depth survival prob	0.99	0.95	0.90
Data augmentation	AutoAugment		
Repeated Augmentation	off		
Input resolution	224		
Epochs	300		
Batch size	4096		
Warmup steps	10K		
Hidden dropout	0		
GeLU dropout	0		
Attention dropout (if applicable)	0		
Classification dropout	0		
Random erasing prob	0		
EMA decay	0		
Cutmix α	1.0		
Mixup α	0.8		
Cutmix-Mixup switch prob	0.5		
Label smoothing	0.1		
Peak learning rate	1e-3		
Learning rate decay	cosine		
Optimizer	AdamW		
Adam ϵ	1e-6		
Adam (β_1, β_2)	(0.9, 0.999)		
Weight decay	0.05		
Gradient clipping	1.0		

Table 7: Hyperparameters for Image classification on ImageNet-1K

A.2 Masked Language Modeling

	Ablation Studies	Full Results (Table 5)
Data	C4/RealNews	C4/English
Max sequence length	128	512
Batch size	2048	256
Peak learning rate	7e-4	1e-4
Number of steps	125K	1M
Warmup steps	10K	
Hidden dropout	0	
GeLU dropout	0	
Attention dropout (if applicable)	0	
Learning rate decay	Linear	
Optimizer	AdamW	
Adam ϵ	1e-6	
Adam (β_1, β_2)	(0.9, 0.999)	
Weight decay	0.01	
Gradient clipping	0	

Table 8: Hyperparameters for MLM pretraining on C4.

	SST-2	MNLI	SQuAD v1.1/v2.0
Max sequence length	128		512
Batch size	{16, 32}		32
Peak learning rate	{1e-5, 2e-5, 3e-5}		5e-5
Number of steps/epochs	5 epochs		8K
Warmup steps/portion	10%		1K
Hidden dropout	0.1		
GeLU dropout	0		
Attention dropout (if applicable)	0.1		
Learning rate decay	Linear		
Optimizer	AdamW		
Adam ϵ	1e-6		
Adam (β_1, β_2)	(0.9, 0.999)		
Weight decay	0.01		
Gradient clipping	0		

Table 9: Hyperparameters for MLM finetuning on GLUE and SQuAD.

B Deep-and-Thin Transformers

Perplexity	#L	d_{model}	#heads	Params (M)
4.83	12 + 12	768	12	110
5.08	24 + 24	512	8	92
4.99	48 + 48	384	12	98
5.30	96 + 96	256	8	84

Table 10: MLM results with increasingly deeper & thinner Transformers. As the depth increases, we adjust the model width accordingly to maintain comparable capacity. We observe that the perplexity is insensitive to the model depth at a fixed capacity, and worsens beyond 48 layers. Note these results were obtained using a similar yet different training setup from the rest of the paper.

C Shift Invariance in MLM

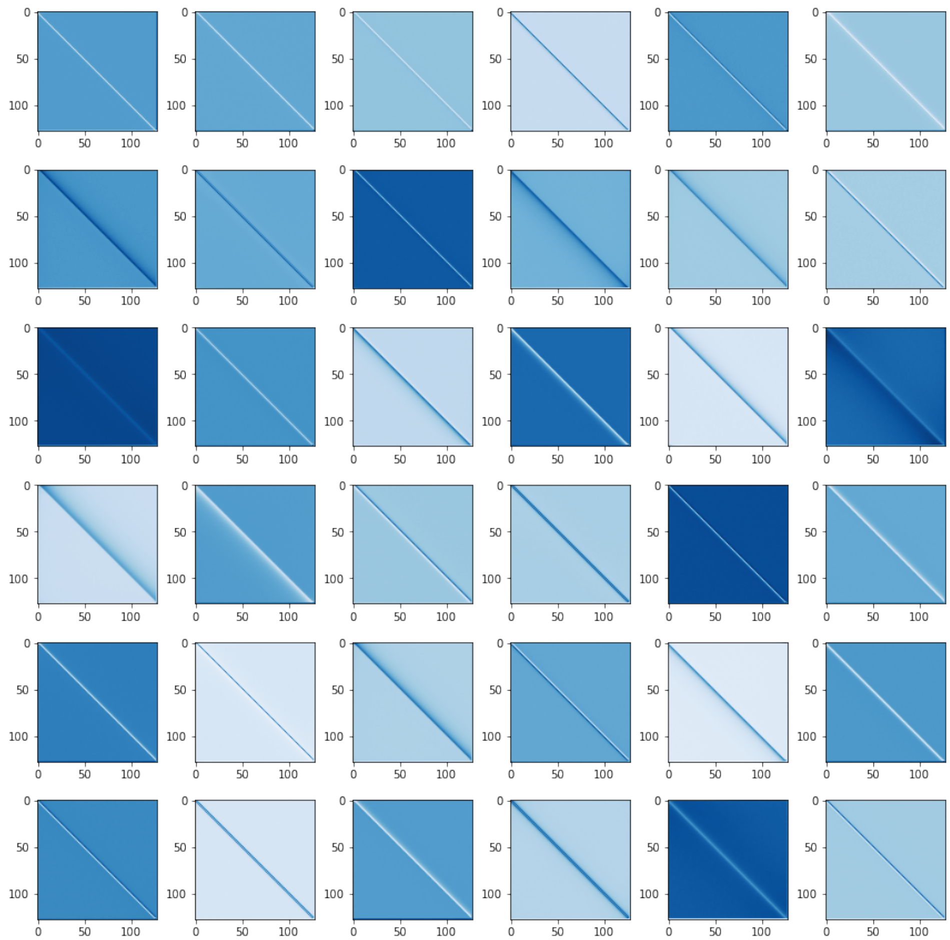


Figure 9: Spatial projection matrices learned on the MLM pretraining task without the shift invariance prior (that each individual W being a Toeplitz matrix). The plots show that gMLP learns Toeplitz-like matrices (hence the notion of shift invariance) regardless.

Creating a Toeplitz Matrix (used in MLM experiments)

```
def create_toeplitz_matrix(n):
  w = tf.get_variable(
    "weight",
    shape=[2 * n - 1],
    initializer=WEIGHT_INITIALIZER)
  r = w.shape[0].value // 2
  t = tf.pad(w, [[0, n]])
  t = tf.tile(t, [n])
  t = t[:-n]
  t = tf.reshape(t, [n, n + w.shape[0] - 1])
  return t[:, r:-r]
```

D Visualizing Tiny Attention

Here we visualize the attention maps of the tiny attention modules in aMLP, after finetuning on MNLI-m. Each element in the heatmap below denotes the maximum attention weight of the corresponding token pair ever received during the first half of the network.

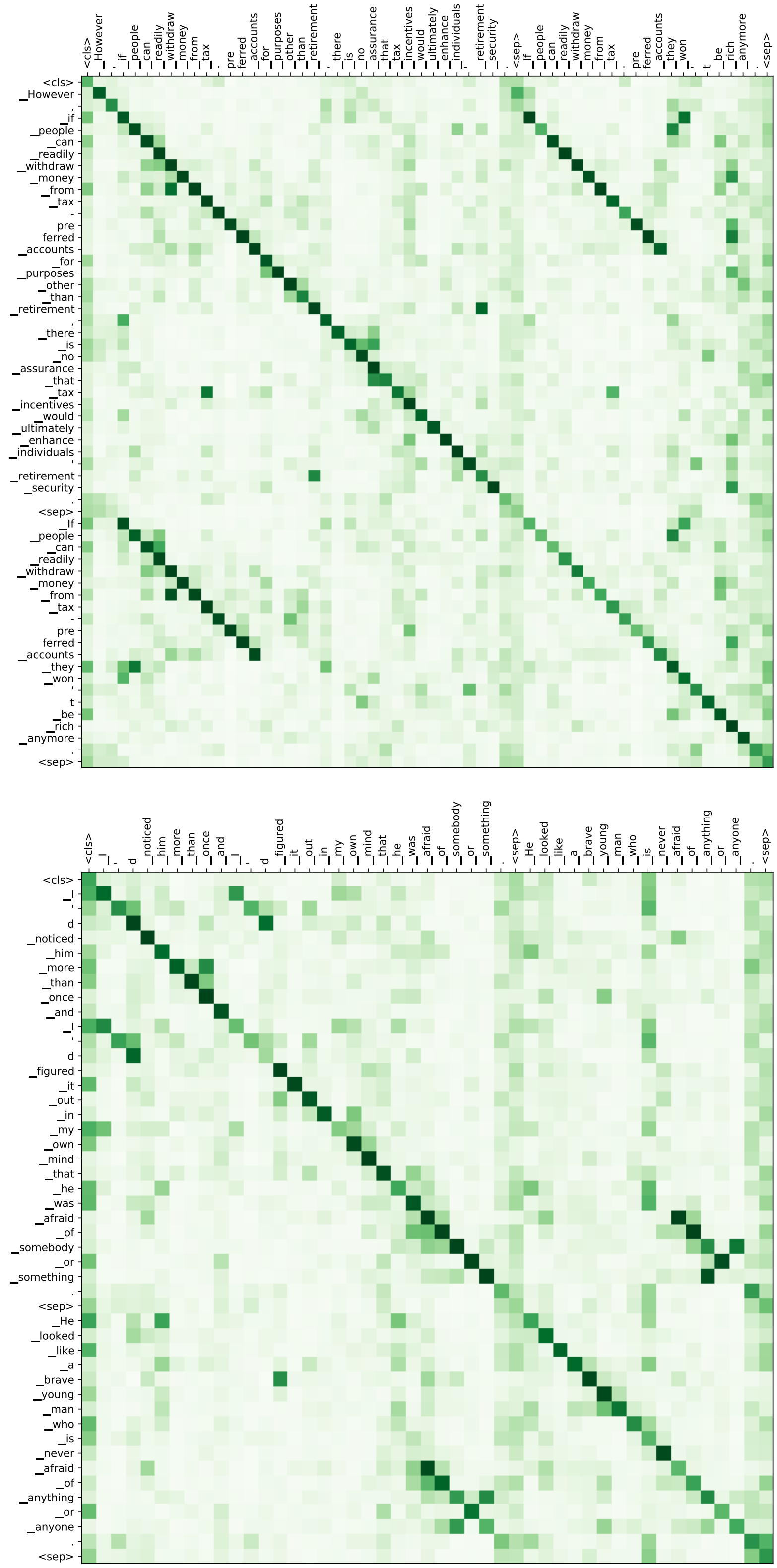


Figure 10: Attention maps in aMLP over selected examples in MNLI-m.