REMIXERS: A Mixer-Transformer Architecture with Compositional Operators for Natural Language Understanding

Anonymous ACL submission

Abstract

Recent work such as MLP-Mixers (Tolstikhin et al., 2021) have demonstrated the promise of All-MLP architectures. While All-MLP architectures have demonstrated reasonable performance in computer vision and garnered recent interest, we argue that making them effective in NLP applications is still an uphill battle. Hence, there may be no solid reason to drop the self-attention modules altogether. In this paper, we propose a new Mixer-Transformer architecture, showing that Transformers and Mixer models can be quite complementary indeed. Fundamentally, we show that Mixer models are capable of acting as persistent global memory (in a similar vein to standard MLPs) while being imbued with global receptive fields at the same time. Hence, interleaving sample-dependent and input-local self-attention with persistent Mixer modules can be an effective strategy. Additionally, we propose compositional remixing, a new way of baking compositional operators (multiplicative and subtractive composition) within the mixing process to improve the expressiveness of the model. This allows us to effectively model relationships between unmixed and mixed representations - an inductive bias that we postulate is powerful for NLU applications. Via extensive experiments on 14 challenging NLU datasets (e.g., SuperGLUE, entailment and compositional generalization), we show that the proposed architecture consistently outperforms a strong T5 baseline (Raffel et al., 2019). We believe this work paves the way for more effective synergies between the two families of models.

1 Introduction

While Transformers (Vaswani et al., 2017) remain as the dominant choice for sequence processing, there has been recent surging interest in All-MLP architectures (Liu et al., 2021; Tolstikhin et al., 2021; Lee-Thorp et al., 2021; Touvron et al., 2021). The key idea in these approaches is to imbue the MLP layers with global receptive fields and is often referred to as *token mixing* - a simple but relatively powerful paradigm. Intuitively, the canonical self-attention module can also be subsumed under the family of mixers - although the crucial difference here is that the mixing is input-local and the mixing process is guided by the pairwise dot product of tokens instead.

While MLP-Mixers have had moderate success in computer vision tasks, its competitiveness in the domain of language, to this date, is at best just speculative. In recent work, Mixers have only been applied in limited setups (BERT style, encoder only) (Liu et al., 2021) and it is still uncertain if they would work in autoregressive setups (GPT-like (Brown et al., 2020)) or encoder-decoder setups (Raffel et al., 2019). Mixer architectures also lack the pseudo cross-attention inductive bias in the encoder, which is crucial for modeling relationship between sentence pairs. This can be mitigated by conveniently adding a tiny bit of self-attention (Liu et al., 2021), but clearly breaks the paradigm and promise of All-MLP architectures. Our early experiments show that MLP-Mixer architectures only marginally outperform simple neural bag-of-words models (CBoW) on SuperGLUE (Wang et al., 2019a).

The benefits of adopting All-MLP paradigms in language is also unclear. In our early experiments, we find that All-MLP architectures are only very marginally faster than Transformers and consume an approximately similar parameter footprint. The token mixing operation is also a function of the sequence length L and is therefore bound to similar quadratic-bottleneck efficiency issues faced in Transformer models (Tay et al., 2020b). On top of all that, we find that MLP-Mixers take a significant hit in quality when compared to vanilla Transformer models.

Fundamentally, the role of interleaving self-attention and MLPs in Transformers can be interpreted as locally-conditioned[1] (*sample-dependent*) mixing and then followed by refining these representations point-wise. For this reason, there is also evidence that the mixing should happen before refinement (Press et al., 2019) and that *persistent* (globally-shared) memory, i.e., shared MLPs for all data points is important (Sukhbaatar et al., 2019). To this end, Mixers behave at the intersection of self-attention and MLPs layers, i.e., they are persistent (globally-shared) and yet they allow a full receptive field. Hence, they can be powerful if used correctly.

In this paper, we propose that there is no solid reason to drop the self-attention module altogether. Hence, we investigate leveraging lightweight token mixing operations to improve Transformers. To this end, we propose REMIXER, a new architecture for language understanding that marries the advantages of Transformers and Mixers. In REMIXER, the self-attention acts as a locally conditioned Mixer and the Remixing block *remixes* this in a globally-shared and persistent fashion. The outcome is a Transformer-like architecture with interleaved global and local mixing operations at every single layer while maintaining a balance of persistent and non-persistent memory. This is also in similar spirit to the neural global workspace model (Goyal et al., 2021) in which the remixing operation can be interpreted as trying to achieve coherence amongst specialists.

Given the role of the remixing operation, we further increase the expressiveness of the REMIXER architecture by introducing compositional operators (e.g., multiplicative, subtractive) to model the relationships between mixed and unmixed representations - an inductive bias that is lacking in standard Transformers and has been shown to be beneficial for NLU (Chen et al., 2016; Wang and Jiang, 2016). We refer to this as *'compositional remixing'*. As we later show in our experiments, we believe that this inductive bias improves the ability of the model to compositionally reason and therefore can be beneficial for NLU and/or language inference tasks, along with improving its (compositional) generalization capability.

We conduct an extensive set of experiments across 8 SuperGLUE (Wang et al., 2019a) tasks, five entailment tasks (e.g., MultiNLI (Williams et al., 2017), Adversarial NLI (Nie et al., 2019), Conjugate NLI (Saha et al., 2020), Abductive NLI (Bhagavatula et al., 2019) and QNLI (Rajpurkar et al., 2016)) and a challenging compositional generalization challenge (Kim and Linzen, 2020). Our experimental results show that Remixers not only substantially outperform a strong T5 baseline but also achieves state-of-the-art on the compositional generalization challenge.

[1]Here, locally conditioned refers to the fact that they depend on the current data point. We distinguish from local windows with respect to the sequence length.

2 Remixer Model

This section introduces the Remixer model. Figure 1 illustrates the proposed model architecture. The overall backbone of the model remains similar to a standard Transformer. Instead of position-wise MLPs, we use the proposed Remixer blocks instead. We keep the self-attention modules unchanged in the REMIXER model.

2.1 Remixer Block

In the first step, we apply a gated linear unit with GeLU activations (Hendrycks and Gimpel, 2016). Given $X_\ell \in \mathbb{R}^{L \times d_{model}}$, the input to this layer ℓ for input length L, this is written as follows:

$$X'_\ell = \sigma_g(X_\ell \mathbf{W}_{\mathbf{1},\ell}) \odot X_\ell \mathbf{W}_{\mathbf{2},\ell}$$

where $\boldsymbol{W}_{1,\ell}, \boldsymbol{W}_{2,\ell} \in \mathbb{R}^{d_{model} \times d_{model}}$ are learnable parameters. The GLU unit here is analogous to the first MLP layer[2] in the Remixer model. Note that this is GLU-based MLP projection is also used in the *T5.1.1* baselines (Shazeer, 2020; Raffel et al., 2019). The core novelty of our approach lies in the following steps.

2.1.1 Remixing of Representations

The next step takes X' and remixes the representations via a form of global persistent memory. In order to do so, we then apply a multiplication of X' with $\sigma(\mathbf{H})$.

$$X_{S,\ell} = \sigma_s(\mathbf{H}_\ell) X'_\ell$$

where $\mathbf{H}_\ell \in \mathbb{R}^{L \times L}$ is a learnable parameter and is globally and persistently shared across all input samples. σ_s is an activation function. It is clear that a multiplication of $\mathbf{H}$ will allow the input sequence to have a global receptive field since this equation is partially reminiscent of the self-attention operation, albeit H_ℓ is learned and shared across all examples

[2]The standard parameter costs of the MLP in vanilla Transformers is $2 \times D_{model} D_{FFN}$. Here we balance parameter cost by reducing the size of W_1 and W_2 by $\frac{1}{3}$. This is the same strategy adopted in T5.1.1 variants.

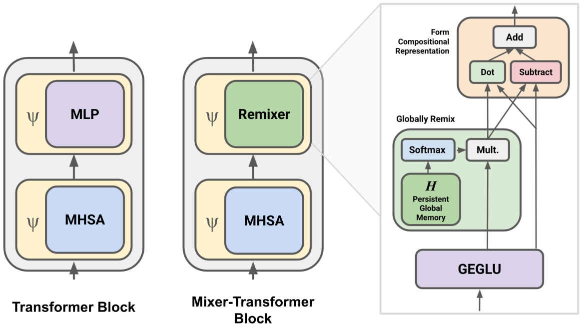

Figure 1: Illustration of a Remixer block in comparison to a standard Transformer block. We propose a Remixer block that learns to remix using a persistent global memory. We then use compositional remixing to learn expressive representations.

instead of being learned via input-dependent dot-product attention. Here, the remix operation uses $\sigma_s = \text{Softmax}$ as it's activation function, which simulates a normalized form of mixing and allows us to keep the transform bounded. Notably, this remixing operation, being persistent and globally shared across all examples and can be interpreted as a form of persistent memory (Sukhbaatar et al., 2019; Geva et al., 2020).

Global Workspace Perspective In (Goyal et al., 2021) the authors proposed the notion of a global workspace where specialists (*positions* in this context) coordinate with one another. In contrast to pairwise relationships in dot product attention which may not achieve global coordination, an interpretation here is that $\boldsymbol{H}$ acts as a global workspace since it is persistent. Specialists (tokens) can write and read from $\boldsymbol{H}$ in order to coordinate and influence other tokens. Hence, Mixers are a form of global workspace.

2.1.2 Compositional Relationships between Mixed and Unmixed Representations

Intuitively, X_s contains the sequences of X that have been re-aligned (or 'mixed') by $\mathbf{H}$. At this point, we apply compositional operators to capture fine-grained information between the unmixed[3] and mixed sequences. This can be written as:

$$X_{C,\ell} = \alpha(X_{S,\ell} \odot X'_\ell) + (1-\alpha)(X'_\ell - X_{S,\ell})$$

where X_C is the construction of taking $X_S \odot X'$ and adding it with $X' - X_s$. In the token mixing operation, very vector in position i in $X_{S,\ell}$ would correspond to $\sum_{j=0}^{\ell} h_{ij} x'_\ell$, a sum of all vectors in X' weighted by matrix $\boldsymbol{H}$. The intuition is here is that $\boldsymbol{H}$ would align globally relevant tokens to X' and the composition operator would model the similarity (or difference) between these unmixed and mixed representations. An alternative interpretation is to allow global information to influence each position in X'. The term α refers to a vector or scalar value $\in [0,1]$ to control the weight between multiplicative and subtractive composition. α may be parameterized (via gating or conditioning on X') or may be set as a hyperparameter.

Multiplicative Composition Multiplicative relationships form the bedrock of modern gating mechanisms (Dauphin et al., 2017; Cho et al., 2014) and are extremely powerful in the field of deep learning. The first term in constructing $X_{C,\ell}$ corresponds to a Hadamard product between pre-mixed and post-mixed representations and is in similar fashion to gating. This can be either be interpreted as modeling the multiplicative relationship (similarity) between unmixed and mixed representations and/or influencing/conditioning the original unmixed sequence with sequence-wise information. This is in similar spirit to how (Liu et al., 2021) motivates the spatial gating unit in the gMLP model.

Subtractive Composition In standard Transformers, there is no subtractive (e.g., $a - b$) interactions between aligned or mixed sequences, an inductive bias which may be important for NLI/NLU models (Chen et al., 2016) since the subtraction operator is known to be able to model negation (Zhu et al., 2014). Notably, the negation operation is also asymmetrical, which makes it uniquely distinct in Transformer models. This is unlike regular dot products, which are fully symmetrical $f(a,b) = f(b,a)$. It is worth to note that asymmetrical $f(a,b) \neq f(b,a)$ operations further helps to model a sense of direction since there is a clear direction of unmixed and mixed relationships.

Output Finally, the output of the Remixer block is computed as:

$$Y_\ell = X_{C,\ell} \boldsymbol{W}_{3,\ell} + X'_\ell$$

where $\boldsymbol{W}_{3,\ell} \in \mathbb{R}^{d_{model} \times d_{model}}$ are trainable parameters. In short, this equation describes a linear transform across $X_{C,\ell}$ followed by a residual connection with X'_ℓ.

[3] For clarity, we refer to each input that arrives at this layer as unmixed (before mixing) even if they have been mixed in subsequent layers.

Remixer Stack The entire Remixer architecture is stacked blocks of Self-Attention followed by Remixer blocks that replace the original MLP layers.

$$X'_\ell = \psi(\text{MHSA}_\ell(X_\ell))$$
$$Y_\ell = \psi(\text{RemixerBlock}_\ell(X'_\ell))$$

where $\psi(.)$ are submodule wrapper operations (i.e., layer norm + residual connections) and MHSA is a standard multi-headed self-attention block (Vaswani et al., 2017).

Parameter Complexity The Remixer block takes up slightly more parameters compared to standard Transformer blocks. Concretely, there is an addition of a L^2 parameters to each layer. We explore options to compensate for this parameter increase. In particular, we found that sacrificing some decoder layers to balance the increase cost of **H** to be useful in practice. In experiments, we refer to this as the *scaled* base model that matches the parameters of the T5 base model. Given ℓ_E and ℓ_D layers in the standard T5 model where $\ell_E = \ell D$, we adopt $\ell'_E = \ell_E + \frac{\ell_D}{2}$ and $\ell'_D = \frac{\ell_D}{4}$. This effectively drops a quarter of the decoder layers to compensate for the increase in parameters due to $\boldsymbol{H}$. See compute metrics in experimental setup for more details.

3 Experiments

This section describes our experiments. To ascertain the effectiveness of Remixers, we conduct experiments on 8 NLU tasks in the SuperGLUE suite, 5 entailment tasks and a challenging compositional generalization task.

3.1 Experimental Setup

This section describes our experimental setup. Most of our experiments follow the seq2seq paradigm (Sutskever et al., 2014) and uses the T5 architecture (Raffel et al., 2019). This is largely because the seq2seq paradigm is fundamentally superior given its ability to subsume encoder-only tasks and decoder-heavy tasks (generation, translation) within the same model architecture.

3.1.1 Pre-training Setup

We follow the setup of (Raffel et al., 2019) and pretrain all our models from scratch for $524K$ steps with the Cleaned Colossal CommonlyCrawl Corpus (C4;Raffel et al. (2019)) using a batch size of 128 and an input sequence length of 512. We use the span corruption objective with a span size of 3 and 15% corruption rate. The pretraining task optimizes the seq2seq loss and is trained with teacher forcing. We pretrain our models on 16 TPU-V3 chips.

3.1.2 Baselines and Implementation Details

Baselines For all experiments, we compare our model with a very competitive state-of-the-art T5 model (Raffel et al., 2019). We use the *T5.1.1* version which no longer shares input and output embeddings, and uses GeLU activations with gated linear units (Dauphin et al., 2017; Shazeer, 2020). We also compare with a MLP-Mixer model adapted for language tasks. Since there is no prior work that adapts MLP-Mixer for encoder-decoder setups, we compare with two variants - using the MLP-Mixer encoder only and/or adapt the MLP-Mixer model to a seq2seq setup. In the decoder, we simply adapt the token mixing to a fixed window size w. All models that we evaluate have been pretrained in the same setup as the REMIXER model. Whenever applicable, we also directly compare with a BERT (Devlin et al., 2018) baseline from prior work. The compute metrics (FLOPS, speed and parameter count) of the baselines are reported below in Table 1. The FLOPs is the number of floating point operations for a single forward pass of the model. We denote the scaled version of REMIXER$_{Base}$ as REMIXER$_{SBase}$.

Implementation Details All models use the same $32K$ sentencepiece (Kudo and Richardson, 2018) vocabulary. We use the default sentencepiece from (Raffel et al., 2019). Our code is implemented in Mesh Tensorflow[4] (Shazeer et al., 2018) and train all models with the Adafactor optimizer. We apply a dropout of 0.1 during finetuning on all MLP layers. We also experimenting with applying dropout on $\boldsymbol{H}$ amongst $\{0.0, 0.1, 0.2\}$ and find that dropping out values from $\boldsymbol{H}$ on some downstream tasks. Models are trained with *bfloat16* precision.

3.2 Natural Language Understanding

We conduct experiments on the SuperGLUE benchmark (Wang et al., 2019a) where we finetune our model on all SuperGLUE tasks in a co-training setup. SuperGLUE comprises of 8 tasks including BoolQ (Clark et al., 2019), CommitmentBank (De Marneff et al., 2019), CoPA (Roemmele et al.,

[4] https://github.com/tensorflow/mesh

Table 1: Compute Metrics for different models in our experiments.

Model	Params	FLOPS	Steps/s
T5.1.1$_{Base}$	248M	3.4×10^{13}	9
Mixer$_{Base}$	212M	1.2×10^{13}	11
Remixer$_{SBase}$	224M	1.3×10^{13}	8
Remixer$_{Base}$	324M	2.1×10^{13}	6

2011), MultiRC (Khashabi et al., 2018), ReCoRD (Zhang et al., 2018), RTE (Dagan et al., 2005), WiC (Pilehvar and os'e Camacho-Collados, 2018) and WSC (Levesque et al., 2012). This is similar to (Narang et al., 2021; Raffel et al., 2019). Likewise, we do the same for all T5 baselines that we run.

Hyperparameters and Setup We finetune our models for $200K$ steps with a batch size of 128 and a constant learning rate of 10^{-3} using the Adafactor optimizer. We use a dropout of 0.1. Similar to (Raffel et al., 2019), we also compare both T5 and Remixer in the setup where we co-trained on a downstream mixture of GLUE, SuperGLUE and SQuAD tasks along with the C4 span corruption task. We pretrain and co-train for 1M steps in this setup. We label this co-train variant as *MT* in our experiments which stands for multi-task pretraining.

3.2.1 Results on SuperGLUE

The results of Remixer on SuperGLUE are generally very positive. Without multi-task pretraining, the Remixer$_{Base}$ outperforms the T5.1.1$_{Base}$ by $+1.5\%$ absolute points on the SuperGLUE average. It also outperforms T5 on 7 out of 8 SuperGLUE tasks. With multi-task pretraining (denoted MT), Remixer$_{Base,MT}$ outperforms T5.1.1$_{Base,MT}$ by $+3.1\%$ absolute percentage points. Similarly, it also outperforms T5 on 7 out of 8 tasks considered. It is also noteworthy that performance gains on certain tasks such as WSC are almost an increment of $+6\%$ and $+4\%$ for CB task. Finally, we note that the performance of Mixers[5] on this task is only slightly better than the CBoW model.

3.3 Entailment Tasks

Entailment, or natural language inference, is a core NLU task that aims to predict of two sentences entail or contradict each other. We use five well-established entailment tasks, namely MultiNLI (Williams et al., 2017), Adversarial NLI (Nie et al., 2019) and Conjugate NLI (Saha et al., 2020), Abductive NLI (Bhagavatula et al., 2019) and Question Answering NLI (QNLI) (Rajpurkar et al., 2016; Wang et al., 2019b). For each dataset we finetune all models for $100K$ steps with a learning rate of 10^{-3} using 16 TPU-v3 chips.

3.3.1 Experimental results on Entailment

Table 3 reports results on entailment. On all five datasets, we observe that Remixer (both sizes) outperforms the T5.1.1 model. Notably, the Remixer$_{SBase}$ model ($\approx$220M) parameters outperforms a BERT large model ($335M$ parameters). The Remixer$_{Base}$ model substantially outperforms T5. This shows that Remixer is a powerful inductive bias for entailment tasks. We note that Mixers generally are incapable of performing this task to a reasonable level because they lack the pseudo cross-attention inductive bias in the encoder. Hence, the tokens across premise and hypothesis sentences are often *blindly* mixed.

3.4 Compositional Generalization Challenge (Semantic Parsing)

We conduct experiments on compositional generalization challenge (Kim and Linzen, 2020). Compositional generalization (or systematic generalization (Bahdanau et al., 2018)) is the task of generalizing to unseen combinations of seen objects in training. The challenge is framed as a semantic parsing task in which the task is to generate a semantic representation given natural language. We refer interested readers to (Kim and Linzen, 2020) for examples and details. Here, all models evaluated are sequence-to-sequence models. We finetune our pre-trained models on this task for $50K$ steps with a constant learning rate of 10^{-3} and batch size of 128. Models are evaluated on exact match (EM).

3.4.1 Experimental Results on Compositional Generalization

Table 4 report results on the compositional generalization challenge. We show that the proposed Remixer achieves state-of-the-art performance on this dataset. Remixers outperform T5$_{Base}$ by $+2.3\%$ relative percentage points and even outperforms T5$_{Large}$ which has more than double the parameters of Remixer. The Mixer$_{Enc}$ model does decently but is outperformed by the $T5_{Base}$ model. We failed to produce decent results with

[5] We verified that our Mixers are correctly implemented, as they achieve reasonable negative log perplexity during pretraining.

Table 2: Results on SuperGLUE dev set for base models. BERT results reported from SuperGLUE paper. Remixer outperforms state-of-the-art T5 model consistently across all setups. On average, there is a $+2.0\%$ to $+4.1\%$ relative performance gain across apples to apples comparisons/setups.

Model	BQ	CB	CP	MultiRC	ReC	RTE	WiC	WSC	Avg
CBoW	62.4	71.4/49.6	63	20.3/ 0.3	14.4/13.8	54.2	55.3	61.5	47.7
BERT$_{Large}$	77.7	94.6/93.7	69	70.5/24.7	70.6/69.8	75.8	74.9	68.3	72.2
BERT+$_{Large}$	80.1	96.4/95.0	78	70.5/24.7	70.6/69.8	82.3	74.9	68.3	74.6
Mixer$_{Enc}$	67.9	65.7/66.1	59	56.6/9.7	53.8/52.4	54.5	56.4	64.4	56.8
Mixer$_{EncDec}$	62.2	79.9/80.4	56	53.3/0.3	52.7/48.7	49.1	50.0	64.4	54.9
T5$_{Base}$	77.8	92.4/92.9	75	72.2/30.4	73.7/72.8	75.8	69.7	82.7	74.8
T5.1.1$_{Base}$	79.3	92.4/92.9	72	74.2/32.8	74.9/73.9	79.8	70.2	81.7	75.4
T5.1.1$_{Base,MT}$	**82.8**	89.2/92.9	65	**78.6/44.2**	77.9/77.1	84.1	68.3	79.8	76.2
Remixer$_{SBase}$	80.2	**98.7/98.2**	65	76.0/35.9	75.6/74.8	81.6	69.1	82.7	76.0
Remixer$_{Base}$	80.5	96.4/98.1	68	74.4/32.7	77.8/77.0	81.2	**72.3**	84.6	76.9
Remixer$_{Base,MT}$	81.4	94.3/96.4	**77**	77.5/42.6	**78.1/77.2**	**85.2**	69.4	**88.5**	**79.3**
Rel. Gain$_{Base}$	+1.5%	+4.3/5.6%	-5.9%	±0%	+3.9/4.2%	+1.8%	+3%	+3.5%	+2.0%
Rel. Gain$_{MT}$	-1.2%	+5.7/3.7%	+19%	-1.4/3.8%	±0%	+1.3%	+1.6%	+11%	+4.1%

Table 3: Experimental results on entailment (natural language inference). For ConjNLI and ANLI, we do not train on MNLI/SNLI. We observe a $+0.9\%$ to $+2.7\%$ improvement across NLI tasks.

Model	MNLI	AdvNLI	ConjNLI	AbNLI	QNLI
BERT$_{Base}$	84.6 / 84.8	-	58.1	-	88.4
BERT$_{Large}$	86.6 / -	57.2 / 49.0 / 43.5	-	-	92.3
T5.1.1$_{Base}$	86.1 / 86.0	59.5 / 48.3 / 48.0	67.4	67.8	91.6
Mixer$_{Base}$	59.2 / 58.2	45.9 / 43.5 / 43.6	62.6	51.1	59.3
Remixer$_{SBase}$	86.6 / 86.9	60.3 / 48.4 / **48.8**	67.4	66.8	92.3
Remixer$_{Base}$	**87.4 / 87.2**	**60.7 / 49.5** / 48.2	**68.5**	**69.6**	**92.4**
Relative Gain	+1.5%/1.4%	+2.0%/+2.5%/+0.4%	+1.6%	+2.7%	+0.9%

Table 4: Results on Compositional Generalization Challenge Benchmark. Remixer base outperforms both T5 base and T5 large on generalization performance.

Model	Params	Gen. EM
Results from (Kim and Linzen, 2020)		
LSTM	11M	32.0
BiLSTM	10M	16.0
Transformer	9.5M	35.0
Mixer$_{Enc,Base}$	212M	76.5
Mixer$_{EncDec,Base}$	212M	N/A
T5.1.1$_{Base}$	248M	77.4
T5.1.1$_{Large}$	738M	77.8
Remixer$_{Base}$	302M	**79.2** (+2.3%)

the Mixer$_{EncDec}$ model.

4 Analysis

In this section, we provide some analysis such as ablations and visualisations. We also discuss some limitations with our understanding of the model.

4.1 Ablation

Table 5 reports the results of ablation studies in which we demonstrate the effect of some of our design choices. In ablation (1), we skip the computation of compositional remixing equation X_C. We show that performance when doing that is lowered. We also tried another ablation (2), that learns a gating vector $\sigma(G(x))$ to parameterize α. Intuitively, this gate controls whether the model decides to use the compositional remixing module. Our findings show that (1) compositional remixing is helpful and (2) a simple formulation works best and gating worsen performance.

Table 5: Ablation experiments on SuperGLUE dataset.

Ablation	Avg
Remixer$_{Base}$	76.9
(1) - w/o comp. remixing	74.8
(2) w gating between X_c and X_s	75.6

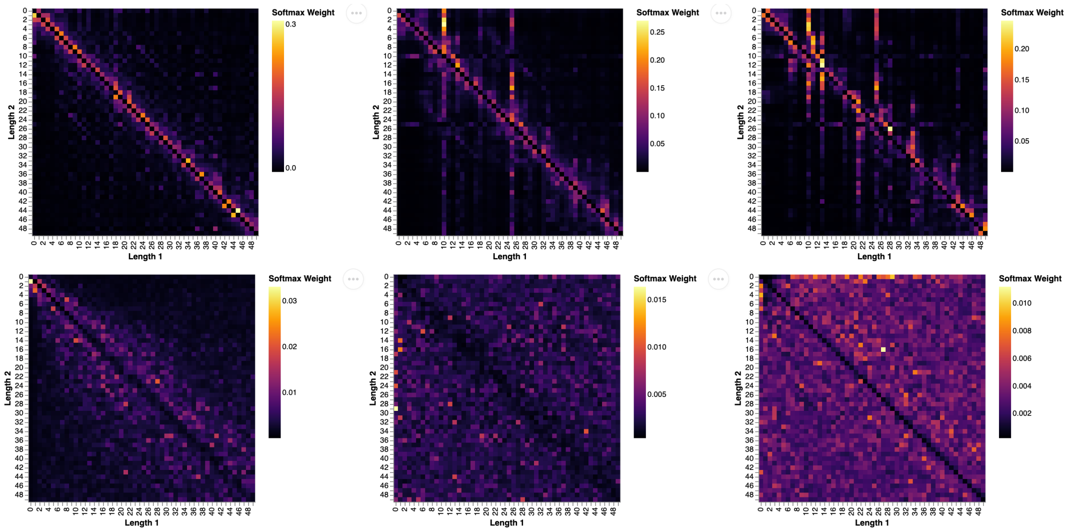

Figure 2: Visualization of $\sigma_s(\mathbf{Q}_\ell \mathbf{K}_\ell^T)$, top row, and $\sigma_s(\mathbf{H}_\ell)$, bottom row, at $\ell = 1, 9, 18$ of a REMIXER model finetuned on MNLI. To simplify the visualization, we show the mean of all self-attention heads.

4.2 What does H learn?

To investigate what $\boldsymbol{H}$, the global persistent memory used to remix representations, actually learns in practice, we provide a visualization of $\boldsymbol{H}$ trained on the MNLI task in Figure 2. We compare this global memory to the input-conditioned $\boldsymbol{QK}^T$ found in the multi-headed self-attention module. We see that self-attention begins local in the first layer, and grows to be more distant and specific in the last layer. Likewise, REMIXER memory begins local and grows to global in the length dimension (however consistently avoiding self-relationships). We also observe that distribution of $\boldsymbol{H}$ does not vary greatly between pretraining and finetuning, or between different finetuning tasks. However, we do observe some differences with the degree of remixing that the model decides to do between tasks. E.g., for a model finetuned on MNLI, there are three layers where $\boldsymbol{H}$ is near-zero, versus only one layer for the same model finetuned on COGS (Kim and Linzen, 2020).

4.3 Limitations

The proposed architecture takes a step forward towards integrating persistent memory modules that have global receptive fields (e.g., Mixers) with Transformer architectures. There are still many open questions regarding the adoption of $\boldsymbol{H}$ as a global remixing matrix. There are natural questions of whether we are able to scale to a large number of tasks or languages by keeping a single $\boldsymbol{H}$ across all tasks/languages. While we observe that (1) $\boldsymbol{H}$ remains static regardless of the input and that it is correct and (2) it differs distinctly from self-attention, not much can be interpreted from $\boldsymbol{H}$. We are also puzzled by why $\boldsymbol{H}$ remains similar irregardless of the nature of the task. We believe that further and deeper understanding and investigation of this type of architecture is warranted.

5 Related Work

This section describes the background and related work for this paper. We briefly describe attention and Transformers, followed by works that question the need for self-attention in Transformers. We then move on to discuss recent trends in All-MLP architectures. Finally, we touch on some works that explain the importance of MLPs in Transformers.

5.1 Attention is All you Need

Transformer (Vaswani et al., 2017) architectures are the dominant choice for sequence processing. A myriad of variants have been proposed over the years (So et al., 2019; Dehghani et al., 2018; Fedus et al., 2021; Lan et al., 2019). We refer interested readers to a comprehensive survey and empirical evaluation of many of these models at (Narang et al., 2021). A key defining characteristic of Transformers is the self-attention mechanism that learns locally conditioned alignment weights via dot product attention. Owing to the quadratic complexity nature of self-attention, many variants have been proposed to tackle this problem (Choromanski et al., 2020; Wang et al., 2020). See (Tay et al., 2020b) for a detailed review of these architectures.

5.2 Do we need attention?

The true utility of self-attention has been questioned numerous times across the literature. (Raganato et al., 2020) proposed fixed encoder attention and shows that one can attain reasonable or better performance on machine translation. (Tay et al., 2020a) proposed the notion of random synthetic attention matrices and show competitive performance on machine translation. (You et al., 2020) proposed random Gaussian attention which also sets attention matrices to be random.

5.3 You don't need Attention.

A recent trend shows that one may not need attention after all! The key idea behind MLP-Mixers (Tolstikhin et al., 2021) is to imbue the MLP layers with a token mixing operation. In practice, this is simply done by transposing the length (L) dimension and d_{model} dimension before applying the MLPs. Essentially, the model is a learned projection across the length dimension. By applying a linear projection across L, dimensions across each token in the sequence are effectively *'mixed'* and therefore are sequence-aware / obtain a global receptive field. There have been other types of mixers that have been proposed, including FNet (Lee-Thorp et al., 2021) which performs fourier transform based mixing and/or gMLP (Liu et al., 2021) which proposes a spatial gating mechanism for mixing. In parallel, (Wu et al., 2019) proposed lightweight and dynamic convolutions that outperform Transformers on a range of sequence generation tasks and (Tay et al., 2021) demonstrated pretrained convolutions may be competitive to pretrained Transformers.

5.4 The role of MLPs in Transformers

At least two thirds of a Transformer's parameters are in the MLPs. This can be significantly more in sparse models (Fedus et al., 2021). We look at works that study the influence and importance of MLPs in Transformers. (Sukhbaatar et al., 2019) proposed the notion of persistent memory vectors and argues that MLPs in Transformers act as a form of persistent memory that is globally shared. They then go on to propose All-Attention networks that fold the MLP layers into the self-attention module. (Geva et al., 2020) showed that MLPs in Transformers are key-value memories and react to different types of knowledge. (Goyal et al., 2021) proposed a neural shared workspace and suggests that alignment learned via pairwise interactions cannot achieve global coordination.

5.5 Natural Language Inference and Understanding

The task of NLI (natural language inference) (MacCartney and Manning, 2008) is to determine if two sentences entail or contradict each other. Before the advent of large pretrained Transformer models (Devlin et al., 2018; Raffel et al., 2019), researchers and practitioners have spent tremendous effort designing custom inductive biases (Chen et al., 2016; Wang and Jiang, 2016; Tay et al., 2017) for a myriad of natural language understanding tasks. Today the domain of NLU can be broadly used to refer to question answering reading comprehension or language inference tasks. Models that performed well on these problems also relied quite a lot on the inductive bias of composition operators between aligned sequences, e.g., the ESIM model (Chen et al., 2016) explicitly models contradiction and agreement using $[a, a', a \odot a', a - a']$ where a' is the newly re-aligned sequence. The Compare-Aggregate model (Wang and Jiang, 2016) similarly adopts this formulation. We note that this inductive bias is specifically missing in modern Transformer architectures.

6 Conclusion

In this paper, we first showed that MLP-Mixers perform poorly on language tasks, achieving only roughly similar performance to the neural bag-of-words baseline in SuperGLUE. We highlight the limitations of the Mixer model and show that there might be a tremendous amount of effort required to make Mixers work in an All-MLP style for language (i.e., such as adding tiny attention (Liu et al., 2021)). To this end, we postulate that Mixers are best employed as a form of persistent global memory that has a full receptive field. To this end, we propose Remixers, a Mixer-Transformer architecture that marries the benefit of self-attention and Mixers. We conduct extensive experiments over 8 SuperGLUE tasks, 5 natural language inference tasks and a challenging compositional generalization tasks. Our experimental results show that Remixers consistently outperform strong T5 baseline models.

References

Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville. 2018. Systematic generalization: what is required and can it be learned? *arXiv preprint arXiv:1811.12889*.

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Scott Wen-tau Yih, and Yejin Choi. 2019. Abductive commonsense reasoning. *arXiv preprint arXiv:1908.05739*.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2016. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

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

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *NAACL*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks.

Marie-Catherine De Marneff, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. *proceedings of Sinn und Bedeutung 23*.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2018. Universal transformers. *arXiv preprint arXiv:1807.03819*.

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

William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2020. Transformer feed-forward layers are key-value memories.

Anirudh Goyal, Aniket Didolkar, Alex Lamb, Kartikeya Badola, Nan Rosemary Ke, Nasim Rahaman, Jonathan Binas, Charles Blundell, Michael Mozer, and Yoshua Bengio. 2021. Coordination among neural modules through a shared global workspace. *arXiv preprint arXiv:2103.01197*.

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

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface:a challenge set for reading comprehension over multiple sentences. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*.

Najoung Kim and Tal Linzen. 2020. Cogs: A compositional generalization challenge based on semantic interpretation. *arXiv preprint arXiv:2010.05465*.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. 2021. Fnet: Mixing tokens with fourier transforms. *arXiv preprint arXiv:2105.03824*.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Hanxiao Liu, Zihang Dai, David R So, and Quoc V Le. 2021. Pay attention to mlps. *arXiv preprint arXiv:2105.08050*.

Bill MacCartney and Christopher D Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 521–528.

Sharan Narang, Hyung Won Chung, Yi Tay, William Fedus, Thibault Fevry, Michael Matena, Karishma Malkan, Noah Fiedel, Noam Shazeer, Zhenzhong

Lan, et al. 2021. Do transformer modifications transfer across implementations and applications? *arXiv preprint arXiv:2102.11972*.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2019. Adversarial nli: A new benchmark for natural language understanding. *arXiv preprint arXiv:1910.14599*.

Mohammad Taher Pilehvar and os'e Camacho-Collados. 2018. Wic: 10, 000 example pairs for evaluating context-sensitive representations. *CoRR*, abs/1808.09121.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Alessandro Raganato, Yves Scherrer, and Jörg Tiedemann. 2020. Fixed encoder self-attention patterns in transformer-based machine translation. *arXiv preprint arXiv:2002.10260*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI Spring Symposium Series*.

Swarnadeep Saha, Yixin Nie, and Mohit Bansal. 2020. Conjnli: Natural language inference over conjunctive sentences. *arXiv preprint arXiv:2010.10418*.

Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.

Noam Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, HyoukJoong Lee, Mingsheng Hong, Cliff Young, et al. 2018. Mesh-tensorflow: Deep learning for supercomputers. In *Advances in Neural Information Processing Systems*, pages 10414–10423.

David R So, Chen Liang, and Quoc V Le. 2019. The evolved transformer. *arXiv preprint arXiv:1901.11117*.

Sainbayar Sukhbaatar, Edouard Grave, Guillaume Lample, Herve Jegou, and Armand Joulin. 2019. Augmenting self-attention with persistent memory. *arXiv preprint arXiv:1907.01470*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*.

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

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020b. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*.

Yi Tay, Mostafa Dehghani, Jai Gupta, Dara Bahri, Vamsi Aribandi, Zhen Qin, and Donald Metzler. 2021. Are pre-trained convolutions better than pre-trained transformers? *arXiv preprint arXiv:2105.03322*.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2017. Compare, compress and propagate: Enhancing neural architectures with alignment factorization for natural language inference. *arXiv preprint arXiv:1801.00102*.

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

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint arXiv:1905.00537*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019b. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In the Proceedings of ICLR.

Shuohang Wang and Jing Jiang. 2016. A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747*.

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

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*.

Weiqiu You, Simeng Sun, and Mohit Iyyer. 2020. Hard-coded gaussian attention for neural machine translation. *arXiv preprint arXiv:2005.00742*.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. Record: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint arXiv:1810.12885*.

Xiaodan Zhu, Hongyu Guo, Saif Mohammad, and Svetlana Kiritchenko. 2014. An empirical study on the effect of negation words on sentiment. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 304–313, Baltimore, Maryland. Association for Computational Linguistics.