A Mixture of $h - 1$ Heads is Better than h Heads

Hao Peng[♠] **Roy Schwartz**[◇♠] **Dianqi Li**[♣] **Noah A. Smith**[◇♠]
[◇]Allen Institute for Artificial Intelligence
[♠]Paul G. Allen School of Computer Science & Engineering, University of Washington
[♣]Department of Electrical & Computer Engineering, University of Washington
{hapeng,roysch,nasmith}@cs.washington.edu, dianqili@uw.edu

Abstract

Multi-head attentive neural architectures have achieved state-of-the-art results on a variety of natural language processing tasks. Evidence has shown that they are overparameterized; attention heads can be pruned without significant performance loss. In this work, we instead "reallocate" them—the model learns to activate different heads on different inputs. Drawing connections between multi-head attention and mixture of experts, we propose the **m**ixture of **a**ttentive **e**xperts model (MAE). MAE is trained using a block coordinate descent algorithm that alternates between updating (1) the responsibilities of the experts and (2) their parameters. Experiments on machine translation and language modeling show that MAE outperforms strong baselines on both tasks. Particularly, on the WMT14 English to German translation dataset, MAE improves over "transformer-base" by 0.8 BLEU, with a comparable number of parameters. Our analysis shows that our model learns to specialize different experts to different inputs.[1]

1 Introduction

The transformer architecture and its variants achieve state-of-the-art performance across a variety of NLP tasks, including machine translation (Vaswani et al., 2017; Ott et al., 2018), language modeling (Radford et al., 2018; Baevski and Auli, 2019), semantic role labeling (Strubell et al., 2018), and more (Devlin et al., 2019; Liu et al., 2019b; Yang et al., 2019b). Under the hood, multi-head attention provides the driving force: multiple separately parameterized attention functions act in parallel to contextualize the input representations; their outputs are then gathered by an affine transformation, and fed to onward computation.

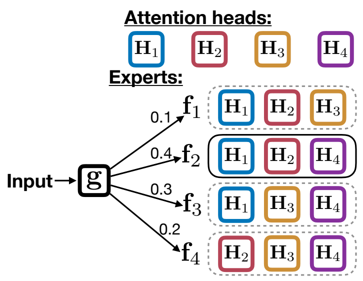
Attention heads:
H_1 H_2 H_3 H_4
Experts:
Input
g
0.1 f_1 H_1 H_2 H_3
0.4 f_2 H_1 H_2 H_4
0.3 f_3 H_1 H_3 H_4
0.2 f_4 H_2 H_3 H_4

Figure 1: Illustration of MAE: a mixture of attentive experts. Each $\mathbf{H}_i$ box is an attention head in a given layer; there are h of them in total. Experts are groups of $h-1$ attention heads. MAE learns an input-dependent distribution of the experts ($\mathbf{g}$). At each training step, a single expert is selected and updated (solid line); during the evaluation, experts' outputs are linearly combined with weights produced by $\mathbf{g}$.

Recent efforts by Voita et al. (2019) and Michel et al. (2019) suggest that typical transformer networks are overparameterized, in the sense that at test time, many of the heads, or even a full layer (Fan et al., 2020), can be removed without significant loss in performance.[2] In response to this observation, they propose to prune the unimportant attention heads in the model after it is trained, aiming for faster inference.

In this paper, we ask whether, instead of reducing the model capacity, we can use it more effectively. We propose **m**ixture of **a**ttentive **e**xperts (MAE). MAE retains all attention heads, and learns to activate different heads on different inputs (see illustration in Figure 1). We start by showing that multi-head attention can be seen as an uniform, input-agnostic mixture of experts (Jacobs et al., 1991), by grouping a subset of atten-

[1]Our implementation is publicly available at https://github.com/Noahs-ARK/MAE.

[2]We do not argue that overparameterization is bad for training. In fact, it may be necessary for successful optimization and good generalization (Neyshabur et al., 2014; Zhang et al., 2016; Soudry and Carmon, 2016, *inter alia*). Rather, we try to explore more efficient ways to use the modeling capacity, than, e.g., removing part of the model.

tion heads as an expert (§2.2). We then introduce MAE, which instead of uniformly weighting the experts, complements the experts with a learned, input-dependent function that assigns their responsibilities (§2.3). To train MAE, we propose a two-step algorithm based on block coordinate descent (§3), which alternates between updating the experts' responsibilities and their parameters.

We evaluate MAE on machine translation and language modeling (§4). Our approach outperforms strong baselines on both; on the WMT14 English to German MT dataset, MAE outperforms transformer-base (Vaswani et al., 2017) by 0.8 BLEU with a negligible increase in the number parameters. Our analysis shows that MAE learns to encourage different experts to specialize on different inputs (§5).

2 MAE: Mixture of Attentive Experts

This section describes MAE in detail. It is inspired by a mixture-of-experts view of multi-head attention, which we present in §2.2. Specifically, we show that multi-head attention can be viewed as a mixture of uniformly weighted experts, each consisting of a subset of attention heads. Based on this observation, we propose MAE, which learns to weight the experts (§2.3) depending on the input. We begin by laying out notation and necessary background in §2.1.

2.1 Background: Mixture of Experts

Mixture of experts is a well-established technique for ensemble learning (Jacobs et al., 1991). It jointly trains a set of **expert models** $\{\mathbf{f}_i\}_{i=1}^{k}$ that are intended to specialize across different input cases. The outputs produced by the experts are aggregated by a linear combination, with a "gating function" $\mathbf{g} = [g_1, \dots, g_k]$ determining the importance of each expert in the final decision:

$$\mathrm{MoE}(\mathbf{x}) = \sum_{i=1}^{k} g_i(\mathbf{x}) \cdot \mathbf{f}_i(\mathbf{x}). \tag{1}$$

The gating function can be parameterized by, e.g., a neural network. We will also refer to $\mathbf{g}$ as the **responsibilities** or **weights** of the experts.

2.2 Multi-Head Attention: a Mixture-of-Experts Perspective

Multi-head attention is the key building block for the state-of-the-art transformer architectures (Vaswani et al., 2017). At its core are multiple separately parameterized attention heads. An attention head takes as input a n-by-d matrix $\mathbf{X}$, with each row being the vector representation of an input element. It contextualizes the input using a dot-product attention mechanism:

$$\widetilde{\mathbf{H}}_i = \mathrm{softmax}\left(\mathbf{X}\mathbf{Q}_i\mathbf{K}_i^\top\mathbf{X}^\top\right)\mathbf{X}\mathbf{V}_i, \tag{2}$$

where $\mathbf{Q}_i$, $\mathbf{K}_i$, and $\mathbf{V}_i$ are learned matrices,[3] and the $\mathrm{softmax}$ normalizes row-wise. The outputs of attention heads are then concatenated and fed through a learned affine transformation:

$$\mathbf{Z} \triangleq \mathrm{MultiHead}\left(\mathbf{X}\right) = \left[\widetilde{\mathbf{H}}_1; \dots; \widetilde{\mathbf{H}}_h\right]\mathbf{W} \tag{3}$$

where $\mathbf{W}$ is a learned matrix, and h denotes the number of attention heads.

We now present a different computation equivalent to Eq. 3, aiming for a smoother transition into following sections. Let $\mathbf{H}_i = \widetilde{\mathbf{H}}_i\mathbf{W}_i$, where $\mathbf{W}_i$ is a block submatrix of $\mathbf{W}$, i.e., $\mathbf{W} = [\mathbf{W}_1^\top; \mathbf{W}_2^\top, \dots; \mathbf{W}_h^\top]^\top$. Then

$$\mathbf{Z} = \left[\widetilde{\mathbf{H}}_1; \dots; \widetilde{\mathbf{H}}_h\right]\mathbf{W} = \sum_{i=1}^{h}\mathbf{H}_i. \tag{4}$$

Eq. 4 provides a different view of the output computation of the multi-head attention: each attention head first projects the contextualized representation with a learned matrix (i.e., $\mathbf{H}_i = \widetilde{\mathbf{H}}_i\mathbf{W}_i$), then their outputs are gathered with a sum (Eq. 4). We now show that this can be seen as a uniformly weighted mixture of experts.

A mixture-of-experts perspective. Let us take a closer look at Eq. 4 and rewrite it:

$$\begin{aligned}
\mathbf{Z} &= \frac{1}{h-1}\sum_{i=1}^{h}\left(-1+h\right)\mathbf{H}_i \\
&= \frac{1}{h-1}\left(-\sum_{i=1}^{h}\mathbf{H}_i + \sum_{i=1}^{h}\sum_{j=1}^{h}\mathbf{H}_j\right) \\
&= \sum_{i=1}^{h}\underbrace{\frac{1}{h}}_{\text{gate } g_i}\underbrace{\frac{h}{h-1}\left(-\mathbf{H}_i + \sum_{j=1}^{h}\mathbf{H}_j\right)}_{\text{expert } \mathbf{f}_i\left(\mathbf{X};\boldsymbol{\theta}_i\right)}.
\end{aligned} \tag{5}$$

Eq. 5 interprets multi-head attention as a mixture of $\binom{h}{h-1} = h$ experts. It first constructs a set of h experts $\{\mathbf{f}_i(\cdot;\boldsymbol{\theta}_i)\}$, with $\boldsymbol{\theta}_i$ denoting $\mathbf{f}_i$'s param-

[3]Some authors explicitly distinguish queries, keys, and values (Vaswani et al., 2017). These inputs can sometimes differ, e.g., in encoder-decoder attention. We suppress such differences for clarity.

eters. $\mathbf{f}_i(\cdot; \boldsymbol{\theta}_i)$ is a parameterized function of the input, which calculates a sum of the outputs by all but the ith attention head. This is achieved by subtracting $\mathbf{H}_i$ from $\sum_{j=1}^{h} \mathbf{H}_j$, then scaling up the results by $h/(h-1)$. The experts share part of the parameters: any two share $h-2$ attention heads. A uniform responsibility of $1/h$ is used.

Discussion. Viewing multi-head attention through this MoE lens suggests some interesting consequences. One can replace the input-agnostic responsibility in Eq. 5 with a function over the input. Indeed, we have good reasons for doing so. Voita et al. (2019) and Michel et al. (2019) show that for transformer networks, a handful of important attention heads are sufficient to achieve good test-time performance. They propose to prune the rest using an input-agnostic procedure. Instead of doing so, here we see a potential alternative: keep all the heads, but only activate those that are important to the input. This motivates MAE, which we now introduce.

2.3 MAE: Learning to Weight Experts

MAE is inspired by the connections between MoE and multi-head attention we draw in §2.2. On top of multi-head attention, MAE learns an input-dependent parameterized gating function $\mathbf{g}(\cdot; \boldsymbol{\phi})$ to complement the experts. More formally, the uniform responsibility $1/h$ in Eq. 5 is replaced by $\mathbf{g}(\cdot; \boldsymbol{\phi})$: given input $\mathbf{X}$, MAE outputs

$$\sum_{i=1}^{h} g_i(\mathbf{X}; \boldsymbol{\phi}) \cdot \mathbf{f}_i(\mathbf{X}; \boldsymbol{\theta}_i). \tag{6}$$

Experts $\mathbf{f}_i$ are the same as those in Eq. 5.

$\mathbf{g}(\cdot; \boldsymbol{\phi})$ is parameterized with a multi-layer perceptron (MLP) followed by a $\mathrm{softmax}$. It first averages $\mathbf{X}$ along the row (i.e., the sequence direction), and then feeds the results through a two-layer $\tanh$-MLP. $\mathbf{g}(\cdot; \boldsymbol{\phi})$ outputs a normalized h-dimensional vector using a $\mathrm{softmax}$, indicating the responsibilities of the experts. It can be seen as a learned probability distribution over the experts.

MAE can learn to assign more responsibility to the experts that are more important to the given input, allowing them to contribute more. MAE is applicable wherever multi-head attention is used. For example, in a machine translation experiment (§4.2), we replace with MAE all the multi-head attention in a transformer network, including the self-attention in all encoder and decoder layers, as well as those attending over the encoded source from the decoder. Each of them is separately treated as a mixture of experts, and has its own gating function. The additional parameter overhead is small: gating functions account for only 3–5% parameters of the full model (Appendix A).

3 Training MAE with Block Coordinate Descent

It is straightforward to jointly train the experts and the gating functions in an MAE model using backpropagation. However, in line with previous observations (Shen et al., 2019), we empirically observe that this is prone to degenerate solutions where the gating functions tend to learn to similarly weight the experts (see §5.1).[4]

As a remedy, we propose a block coordinate descent (BCD) training. At a high level, training is decomposed into two interleaving steps: A **G step** updates the gating function $\mathbf{g}(\cdot; \boldsymbol{\phi})$, fixing the experts; an **F step** fixes the gating function and updates *one* randomly selected expert $\mathbf{f}_i(\cdot; \boldsymbol{\theta}_i)$.[5] The computations for G and F steps differ:

- In a G step, MAE outputs a linear combination of the experts' outputs, and only updates the gating function's parameters (Algorithm 1). *No* expert is updated.
- An F step computes the experts' responsibilities $\mathbf{g}(\mathbf{X})$, according to which an expert i is then sampled (Algorithm 2). MAE computes the output with $\mathbf{f}_i$, which is then updated, without updating the gating function or other experts.[6]

A non-differentiable sampling from $\mathbf{g}$ is involved in F steps. It does not create difficulties for the

[4]Besides the undesired degeneracy, we also find that the model suffers worse overfitting when $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are jointly updated (Appendix B). One possible reason is that, compared to the standard multi-head attention, the learned gates give the model additional capacity to compensate for the experts' errors with others' outputs at *training* time, hurting generalization (Jacobs et al., 1991). Another common degeneracy of MoEs is the "rich get richer" where one of the experts is always picked and others ignored. As observed by Voita et al. (2019), this can happen when the experts are trained to be sparsely weighted. When tuning the hyperparameters, we observe the "rich get richer" degeneracy if the learning rate is set too large.

[5]For clarity, our discussion focuses on $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$. The rest of the model, e.g., the word embeddings in a transformer network, are updated along with $\boldsymbol{\theta}$. Training aims to minimize loss $\mathcal{L}$ over $\{\boldsymbol{\theta}, \boldsymbol{\phi}\}$.

[6]In mini-batch training, which we use in the experiments, different experts can be sampled for different instances in a mini-batch. This is because $\mathbf{g}$ depends on the inputs. This means that multiple experts will be updated in an F step, but each due to a subset of the examples in the mini-batch.

Algorithm 1 A G step update for MAE, with step size η.

```
1: procedure MAEG($\mathbf{X}$)
2:     $\mathbf{Z} \leftarrow \sum_{i=1}^{h} g_i(\mathbf{X};\boldsymbol{\phi}) \cdot \mathbf{f}_i(\mathbf{X};\boldsymbol{\theta}_i)$
3:     Forwardprop with $\mathbf{Z}$ and calculate $\mathcal{L}$.
4:     Calculate $\nabla_{\boldsymbol{\phi}}\mathcal{L}$ with backprop.
5:     $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \eta \cdot \nabla_{\boldsymbol{\phi}}\mathcal{L}$.
6: end procedure
```

Algorithm 2 An F step update for MAE, with step size η.

```
1: procedure MAEF($\mathbf{X}$)
2:     Draw $i \sim \text{Cat}(\mathbf{g}(\mathbf{X};\boldsymbol{\phi}))$
3:     $\mathbf{Z} \leftarrow \mathbf{f}_i(\mathbf{X};\boldsymbol{\theta}_i)$
4:     Forwardprop with $\mathbf{Z}$ and calculate $\mathcal{L}$.
5:     Calculate $\nabla_{\boldsymbol{\theta}_i}\mathcal{L}$ with backprop.
6:     $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i - \eta \cdot \nabla_{\boldsymbol{\theta}_i}\mathcal{L}$.
7: end procedure
```

backpropagation, since an F step never calculates the gradients w.r.t. $\boldsymbol{\phi}$. At test time, the computation is the same as that in a G step, i.e., MAE outputs a linear combination of the experts, weighted by $\mathbf{g}$.

Training time overhead. A straightforward training procedure is to, for each training instance, first take a G step, and then an F step. This doubles the forward propagation computation overhead. In practice, it is not necessary to take G steps as frequently as F steps, since they only update a small portion of the model. In the experiments, we take G steps one fifth as frequently as F steps: we make G updates every 5 epochs while always take F steps. In preliminary experiments, we find this reduces training time overhead without significant impact on the performance.[7]

Algorithm 3 summarizes the block coordinate descent training in a given epoch.

Connections to dropout. In the above block coordinate descent training algorithm, an F step samples an expert to update, and ignores the rest in both forward and backward computation. It is reminiscent of dropout (Srivastava et al., 2014). Specifically, selecting expert $\mathbf{f}_i$ is equivalent to dropping head i.[9] In other words, the F steps (Algorithm 2) can be seen as a structured dropout applied to the attention heads, but with learned input-dependent drop probabilities. When $\mathbf{g}$ is a constant vector with elements $1/h$, it recovers the head dropout, which is also explored by concurrent work (Fan et al., 2020).

[7]In this way, training time for MAE is roughly 1.2 times longer than that of the transformer network it builds on.

[8]Although we assume supervised learning, we suppress the gold outputs for notational clarity. We slightly overload the notation and denote by $\mathbf{X}_i$ the training instance, although they cab also be the outputs of intermediate layers.

Algorithm 3 Block coordinate descent (BCD) training for MAE, at epoch e. $\mathcal{D}$ denotes the training data.[8]

```
1: procedure BCD($\mathcal{D} = \{\mathbf{X}_i\}_i$, $e$)
2:     for $\mathbf{X}_i \in \mathcal{D}$ do
3:         ▷ Take G steps every 5 epochs.
4:         if $e \bmod 5 = 0$ then
5:             MAEG($\mathbf{X}_i$)
6:         end if
7:         ▷ Always do F step updates.
8:         MAEF($\mathbf{X}_i$)
9:     end for
10: end procedure
```

So far, we view MAE as a mixture of h experts, each consisting of $h-1$ attention heads. One can, of course, generalize this to other settings, e.g., mixing $\binom{h}{h-2}$ experts, each containing $h-2$ heads. From the dropout view, this translates to dropping more attention heads: dropping t heads out of h is equivalent to applying a dropout with drop probability t/h, in the sense that their expected numbers of dropped units are the same.

Despite the similarity between MAE and dropout, a key difference exists between the two: with the latter, the constant dropout probability is set *a priori*, while MAE uses a gating function $\mathbf{g}(\cdot;\boldsymbol{\phi})$ to calculate a learned, input-dependent dropout probability.

4 Experiments

We empirically evaluate MAE on machine translation (§4.2) and language modeling (§4.3) benchmarks. We first introduce the compared models (§4.1).

4.1 Compared Models

MAE is evaluated under two settings:

- MAE-7 mixes 8 experts each with 7 attention heads.

[9]Recall from Eq. 5 that $\mathbf{f}_i$ includes all but head i.

- MAE-6 is similar to MAE-7, but mixes $\binom{8}{2} = 28$ experts each with 6 attention heads.[10]

We compare MAE to the following baselines.

- BASE is a sequence-to-sequence model based on the transformer architecture.
- NOBCD is the same model as MAE, but does not use block coordinate descent training. Instead, it jointly updates *all* experts and the gating function at training time, as discussed at the start of §3.
- UNI-MAE-7 is similar to MAE but does not have parameterized gating functions. It builds on BASE, and mixes 8 experts, each with 7 attention heads. Constant uniform responsibilities are assigned to the experts. At each training step, it updates *one* uniformly sampled expert; at test time, the outputs of all experts are averaged according to Eq. 5.
- UNI-MAE-6 mixes 28 6-attention-head experts, and is otherwise the same as UNI-MAE-7.

We refer the readers to Appendix A for implementation details.

4.2 Machine Translation

Datasets. We experiment with two machine translation datasets:

- WMT14 EN-DE (Bojar et al., 2014).[11] Following previous practice (Vaswani et al., 2017) we train on WMT14, and designate newstest2013 and newstest2014 as development and test data respectively. Our preprocessing follows that of Vaswani et al. (2017) and Ott et al. (2018). A shared source-target vocabulary is used, with 32k byte pair encoding types (BPE; Sennrich et al., 2016).
- IWSLT14 DE-EN (Cettolo et al., 2014).[12] It is based on TED talks, and is much smaller compared to WMT14. We use the preprocessing from Edunov et al. (2018). Following previous practice, we use separate vocabularies for the source and target, with around 9K and 7K BPE types respectively.

Table 1 summarizes some statistics of the datasets.

[10]Preliminary results show that mixing experts with fewer heads leads to underwhelming performance. We conjecture this is due to too strong a regularization effect (§3).

[11]`https://drive.google.com/a/haopeng.name/uc?export=download&id=0B_bZck-ksdkpM25jRUN2X2UxMm8`

[12]`http://workshop2014.iwslt.org/`.

Data	Train	Dev.	Test	Vocab.
WMT14	4.5M	3K	3K	32K
IWSLT14	160K	7K	7K	9K/7K

Table 1: Some statistics for WMT14 and IWSLT14 datasets. We use separate source and target vocabularies in IWSLT14 experiments.

Evaluation. The models are evaluated using BLEU (Papineni et al., 2002). A beam search with beam size 5 is used. In the WMT14 experiments, we follow Vaswani et al. (2017), and apply a compound split postprocessing.[13]

Results. Table 2 summarizes WMT14 EN-DE translation test performance. The base and large sized transformer models are due to Vaswani et al. (2017). To control for compounding factors, we additionally compare to our implementation of the base sized model (BASE). It achieves slightly better performance than Vaswani et al. (2017), with a 0.3 BLEU edge. MAE-7 improves over the base transformer by 0.8 BLEU, obtaining similar performance to the large-size transformer of Vaswani et al. (2017) using less than a third as many parameters. Since we do not see similar improvement by UNI-MAE-7, we attribute this gain to input-dependent expert weighting. Having a smaller number of heads for each expert, MAE-6 slightly underperforms MAE-7, and so does UNI-MAE-6 in comparison to UNI-MAE-7. Finally, NOBCD gets worse performance than the transformer baseline, demonstrating the importance of the block coordinate decent training.

We observe similar trends on the IWSLT14 DE-EN dataset, summarized in Table 3. The BASE model here is similar to the base-sized transformer in the WMT14 experiment, but with a smaller hidden dimension. MAE-7 outperforms BASE by 0.9 BLEU. Interestingly, UNI-MAE-7 improves over BASE by 0.3 BLEU, possibly because the regularization effect of random expert selection training helps more on this smaller dataset.[14]

4.3 Token-level Language Modeling

Dataset. We experiment with the WikiText-103 dataset (Merity et al., 2016). It contains articles

[13]`https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/utils/get_ende_bleu.sh`

[14]Selecting an expert can be seen dropping one attention head in training (§3).

Model	BLEU	# Params.
Base Transformer	27.3	65M
Large Transformer	28.4	213M
BASE	27.6	61M
‡NOBCD	27.5	63M
†UNI-MAE-7	27.7	61M
†UNI-MAE-6	27.6	61M
†‡MAE-7	**28.4**	63M
†‡MAE-6	28.1	63M

Table 2: WMT14 EN-DE translation test performance on newstest2014. † randomly select an expert to update for each training instance, and ‡ learns a gating function to weight the experts. Transformer performance in the first two rows are due to Vaswani et al. (2017).

Model	BLEU	# Params.
BASE	34.6	39M
‡NOBCD	34.8	41M
†UNI-MAE-7	34.9	39M
†UNI-MAE-6	35.0	39M
†‡MAE-7	**35.5**	41M
†‡MAE-6	35.4	41M

Table 3: IWSLT14 GE-DE test set performance. See Table 2 caption for indications of the superscripts.

from English Wikipedia, with a 268K-sized vocabulary. The training/development/test data respectively have 103M/218K/246K tokens.

Setting. Here the BASE model is the strong language model by Baevski and Auli (2019). It is based on a 16-layer transformer network; each multi-head attention layer has 8 heads. It uses different embedding dimensions for the tokens, based on their frequencies. We closely follow Baevski and Auli (2019) in terms of hyperparameters and training procedures. The readers are referred to their paper and Appendix A for further architecture and hyperparameter details.

Notes on context size. Baevski and Auli (2019) study the effect of context window, i.e., the number of history tokens the model attends over. They find that using larger context sizes lead to better performance (Baevski and Auli, 2019, Table 5). Their best setting uses a 3,072 training context size, and 2,048 at test time (i.e., the model has access 2,048 tokens before predicting any token at test time). However, we are not able to train MAE,

Model	Perplexity	# Params.
⋆BASE (B&A, 2019)	18.70	247M
BASE (B&A, 2019)	19.03	247M
‡NOBCD	19.12	249M
†UNI-MAE-7	19.26	247M
†‡MAE-7	**18.71**	249M

Table 4: Language modeling performance on WikiText-103 test set (lower is better). ⋆Trains/evaluates with 3,072/2,048 context sizes and therefore not directly comparable to other models which use 512/480 sized ones. See Table 2 caption for the indications of other superscripts. Bold font indicates the best performance using smaller context sizes. The first two rows are due to Table 5 of Baevski and Auli (2019).

nor replicate their results, under this setting—our GPUs have far less memory, and it is impossible to even load a 3,072-token context chunk.[15] Therefore we train and evaluate MAE and UNI-MAE-7 with smaller 512/480 context sizes, also explored by Baevski and Auli (2019), which allows for a head-to-head comparison.

Results. Table 4 shows the perplexity on WikiText-103 test data. When trained under the same setting, MAE outperforms Baevski and Auli (2019) by more than 0.3 perplexity. Interestingly, despite the much smaller context at both training and test time, MAE matches the best setting by Baevski and Auli (2019). UNI-MAE-7 and NOBCD *underperform* the baseline (higher perplexity).

5 Analysis

This section first empirically confirms that MAE learns to activate different experts on different inputs in §5.1. We then run a synthetic experiment to explore MAE's potential in transfer learning (§5.2).

5.1 Does MAE Learn to Specialize the Experts?

One of the appealing properties of MoE models is that they could learn to activate different experts, depending on what "expertise" is needed for the

[15]Baevski and Auli (2019) use NVIDIA Tesla V100 GPUs with 32GB memory, while we only have access to GeForce RTX 2080 Ti, with 11GB memory.

Model	**BLEU**	**Diff.**
UNI-MAE-7	26.6	-
One random expert	$25.8_{\pm0.2}$	$\downarrow 0.8_{\pm0.2}$
NOBCD	26.7	-
Most specialized expert	26.0	$\downarrow$ 0.7
MAE-7	27.1	-
Most specialized expert	26.8	$\downarrow$ 0.3

Table 5: Performance decrease for different models on WMT14 development set when only one expert is used for each multi-head attention layer (5.1).

Expert 1	Expert 2	Expert 3	Expert 4
neumann	bell	candidacy	veil
debuted	zero	rose	monument
rental	computing	submission	fox
worthy	decentralized	palm	unnerved
landloards	reuters	roles	remainder
Expert 5	**Expert 6**	**Expert 7**	**Expert 8**
spoil	menses	romans	odds
anybody	technological	sticker	heat
endorsed	inevitably	outdated	marvel
reserve	bet	analyst	ornate
pending	punk	venues	anticipating

Table 6: Indicative tokens for each expert (§5.1). Tokens attributed to Expert 2 are mostly computer science terminology; trends for other experts are less clear.

input. Does MAE learn to do so? We empirically study this question, and present evidence indicating that it does, at least in part. We consider the encoders of the UNI-MAE-7, NOBCD, and the MAE-7 models trained on WMT14.[16]

We first study whether BCD training helps drifting MAE away from uniformly weighting the experts agnostic to the inputs. We treat the gating values as probabilities, and calculate their entropies: $\mathcal{H}(\mathbf{g}) = -\sum_{i=1}^{h} g_i \cdot \log g_i$, which are then averaged across different layers. The average entropy on the development set for MAE-7 is 1.91, lower than the 2.02 by the NOBCD model trained without BCD. In comparison, UNI-MAE-7 uniformly weights the experts and has the entropy of 2.08. This indicates that gating weights of MAE trained with BCD are more "focused" on one or a subset of experts than trained without.

Second, we study whether MAE learns to specialize different experts for different inputs. To do so we attribute the development instances to the experts that maximize the gating weights. For the first encoder layer of MAE-7, the percentages of instances attributed to each of the 8 experts are relatively balanced: 13%, 14%, 9%, 16%, 10%, 15%, 10%, 12%.[17] This suggests that all experts are assigned a substantial part of the input, and it is *not* the case that BCD leads to a "rich get richer" outcome.

We then continue and explore whether MAE performs reasonably well when using only the most "specialized" experts. For each development instance, we select those experts maximizing the gating weights and ignore the rest, instead of linearly combining them as in Eq. 6. We see from Table 5 a 0.3 BLEU decrease under this setting. In comparison, NOBCD has a larger performance decrease of 0.7 BLEU. NOBCD's performance drop is similar to that of UNI-MAE-7, for which we randomly select an expert at each layer and average the performance over 5 runs. These results support the proposition that MAE specializes better when trained with BCD.

Finally, we search for the tokens that are more likely to activate each expert. We compute the pointwise mutual information (PMI; Church and Hanks, 1990) between tokens and experts:

$$\mathrm{PMI}(\mathrm{token}_i, \mathrm{expert}_j) = \log \frac{p(\mathrm{token}_i, \mathrm{expert}_j)}{p(\mathrm{token}_i)p(\mathrm{expert}_j)}.$$

Table 6 lists the most indicative tokens of each expert, for the first layer. While some of the terms for some experts seem loosely related (e.g., *bell, reuters*, and *computing* for expert 2, it is hard to find clear patterns in most of them.

5.2 MAE's Potential in Transfer Learning: A Case Study

We now turn to evaluate another property of MAE: its potential for data-efficient transfer learning, by only updating the gating functions, freezing the experts. We consider the pretrain-then-finetune setting. Due to computation limits, we are unable to explore MAE for pre-training contextual representations (Peters et al., 2018; Devlin et al., 2019). Rather, we focus on the following small-scale machine translation experiments.

Setting. We explore finetuning on IWSLT14 EN-DE data, a MAE model pretrained on the

[16]The same experiments can be done with the decoders, where the inputs to gating functions are German sentences. The authors lack German expertise, and interpretation of a following analysis would not have been possible for us.

[17]We observe similar trends in other layers. See Appendix C for more details.

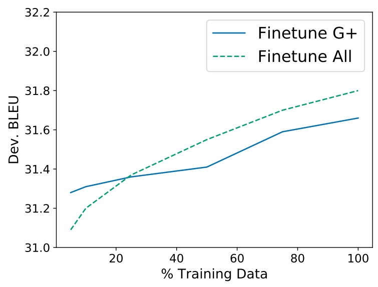

Figure 2: IWSLT14 development performance of FTG+ and FTALL using different amount of training data (§5.2). When trained on less than 20% subset of the original training data, FTG+ outperforms FTALL.

Method	BLEU	# Params.	# Steps.
SCRATCH	28.8	41M	52K
NOFT	29.3	0	0
FTG	30.1	2M	1K
FTG+	31.6	7M	1K
FTALL	**31.8**	63M	12K

Table 7: IWSLT14 development set performance of different finetuning methods (§5.2). The last two columns indicate the number of parameters to update, and the number of gradient steps needed to achieve the best development performance.

much larger WMT14 dataset.[18] We compare three finetuning methods:

- FTG finetunes the gating functions' parameters (i.e., ϕ), keeping the rest frozen.
- FTG+ updates the parameter matrix $\mathbf{W}$ in Eq. 4 in addition to ϕ. The rest of the model parameters are fixed.
- FTALL updates all parameters.

As a baseline, NOFT is the out-of-box pretrained model without any finetuning. SCRATCH trains a MAE model from scratch.

Table 7 summarizes the IWSLT14 EN-DE development set performance. Surprisingly, NOFT already outperforms SCRATCH without any finetuning. We attribute this improvement to the larger pretraining (WMT14) data. Only updating the gating functions, FTG improves over NOFT by 0.8 BLEU. Yet there is still a significant gap of 1.8 BLEU between FTG and FTALL. Interestingly, FTG+ almost matches the performance of FTALL, but only updates 1/9 as many parameters. Both FTG and FTG+ reach the best performance after around 1K gradient updates, i.e., one epoch, significantly less than FTALL or SCRATCH.

We further compare FTG+ and FTALL where less downstream training data is available. To simulate this, we randomly sample [5%, 10%, 25%, 50%, 75%] subsets of IWSLT14 training data, on which the pretrained model is finetuned. Figure 2 plots their performance. We see a clear trend: as less training data is available, the gap between FTG+ and FTALL decreases; when less than 20% of the training data is available, FTG+ outperforms FTALL. These results suggest that finetuning MAE with FTG+ can be viable in low-resource transfer learning.

[18]Here we reverse the translation direction of IWSLT14: §4.2 experimented with DE-EN, here we use EN-DE.

6 Related Work

Multi-head attention. An increasing amount of effort has been devoted into developing better attention mechanisms (Malaviya et al., 2018; Deng et al., 2018; Sukhbaatar et al., 2019; Correia et al., 2019; Maruf et al., 2019, *inter alia*), and improving transformer architectures (Shaw et al., 2018; Dehghani et al., 2019; Hao et al., 2019; Correia et al., 2019; Yang et al., 2019a, *inter alia*). Closely related, Iida et al. (2019) applies another attention mechanism over the attention heads, allowing a learned reweighting of them. Our work focuses on the connection between multi-head attention and MoE, and the BCD training it suggests and benefits from. Concurrent to our work, (Fan et al., 2020) study structurally pruning transformer layers for more efficient inference.

Another line of work aims to better understand the working of transformer models (Clark et al., 2019; Liu et al., 2019a; Tenney et al., 2019, *inter alia*).

Mixture of experts. One of the most successful applications of MoE is ensemble learning (Caruana et al., 2004; Liu et al., 2018; Dutt et al., 2017, *inter alia*). Recent efforts also explore MoE in sequence learning (Shazeer et al., 2017), and to promote diversity in text generation (He et al., 2018; Shen et al., 2019; Cho et al., 2019, *inter alia*).

7 Conclusion

We presented MAE. It is inspired by a mixture-of-experts perspective of multi-head attention. With a learned gating function, MAE activates different experts on different inputs. MAE is trained using a block coordinate descent algorithm, which alternates between updating the responsibilities of the experts and their parameters. Our experiments show that MAE outperforms the transformer baselines on machine translation and language modeling benchmarks. The analysis shows that MAE learns to activate different experts. The code is publicly available at https://github.com/Noahs-ARK/MAE.

Acknowledgments

We thank the anonymous reviewers, Yoav Artzi, Mandar Joshi, Jungo Kasai, Lingpeng Kong, Kenton Lee, Kelvin Luu, Will Merrill, Phoebe Mulcaire, Mark Neumann, Nikos Pappas, Ofir Press, Lianhui Qin, Swabha Swayamdipta, Vivek Srikumar, Sam Thomson, and Dani Yogatama for their helpful feedback. This work was supported in part by NSF grant 1562364, a Google Fellowship, and NVIDIA Corporation through the donation of a Tesla GPU.

References

Alexei Baevski and Michael Auli. 2019. Adaptive input representations for neural language modeling. In *Proc. of ICLR*.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proc. of WMT*.

Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. 2004. Ensemble selection from libraries of models. In *Proc. of ICML*.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th IWSLT evaluation campaign. In *Proc. of IWSLT*.

Jaemin Cho, Minjoon Seo, and Hannaneh Hajishirzi. 2019. Mixture content selection for diverse sequence generation. In *Proc. of EMNLP*.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proc. of BlackBoxNLP*.

Gonçalo M. Correia, Vlad Niculae, and André F.T. Martins. 2019. Adaptively sparse transformers. In *Proc. of EMNLP*.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2019. Universal transformers.

Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, and Alexander Rush. 2018. Latent alignment and variational attention. In *Proc. of NeurIPS*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL*.

Anuvabh Dutt, Denis Pellerin, and Georges Quénot. 2017. Coupled ensembles of neural networks. arXiv:1709.06053.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Classical structured prediction losses for sequence to sequence learning. In *Proc. of NAACL*.

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *Proc. of ICLR*.

Jie Hao, Xing Wang, Baosong Yang, Longyue Wang, Jinfeng Zhang, and Zhaopeng Tu. 2019. Modeling recurrence for transformer. In *Proc. of NAACL*.

Xuanli He, Gholamreza Haffari, and Mohammad Norouzi. 2018. Sequence to sequence mixture model for diverse machine translation. In *Proc. of CoNLL*.

Shohei Iida, Ryuichiro Kimura, Hongyi Cui, Po-Hsuan Hung, Takehito Utsuro, and Masaaki Nagata. 2019. Attention over heads: A multi-hop attention for neural machine translation. In *Proc. of ACL: Student Research Workshop*.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. of ICML*.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. In *Proc. of NAACL*.

Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. 2018. Towards robust neural networks via random self-ensemble. In *Proc. of ECCV*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. RoBERTa: A robustly optimized BERT pretraining approach. arXiv:1907.11692.

Chaitanya Malaviya, Pedro Ferreira, and André FT Martins. 2018. Sparse and constrained attention for neural machine translation. In *Proc. of ACL*.

Sameen Maruf, André FT Martins, and Gholamreza Haffari. 2019. Selective attention for context-aware neural machine translation. In *Proc. of NAACL*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. arXiv:1609.07843.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Proc. of NeurIPS*.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. 2014. In search of the real inductive bias: On the role of implicit regularization in deep learning. In *Proc. of ICLR: Worshop Tack*.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proc. of WMT*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018. Language models are unsupervised multitask learners.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proc. of ACL*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proc. of NAACL*.

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

Tianxiao Shen, Myle Ott, Michael Auli, and Marc'Aurelio Ranzato. 2019. Mixture models for diverse machine translation: Tricks of the trade. In *Proc. of ICML*.

Daniel Soudry and Yair Carmon. 2016. No bad local minima: Data independent training error guarantees for multilayer neural networks. arXiv:1605.08361.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proc. of EMNLP*.

Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. 2019. Adaptive attention span in transformers. In *Proc. of ACL*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proc. of ACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. of NeurIPS*.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proc. of ACL*.

Baosong Yang, Longyue Wang, Derek F. Wong, Lidia S. Chao, and Zhaopeng Tu. 2019a. Convolutional self-attention networks. In *Proc. of NAACL*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019b. XLNet: Generalized autoregressive pretraining for language understanding. arXiv:1906.08237.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2016. Understanding deep learning requires rethinking generalization. In *Proc. of ICLR*.

Appendices

A Architectures and Implementations

Our model is implemented using the PyTorch toolkit and the fairseq codebase.[19]

Machine translation with WMT'14 Our BASE model in this experiment is the transformer-base by Vaswani et al. (2017). Its encoder and decoder are both of 6 transformer layers. Each multi-head attention layer is of hidden size 512, and uses 8 attention heads; the hidden dimensions for the feed forward networks are 2,048. We follow issue #346 of the fairseq's GitHub repository to replicate the results by Vaswani et al. (2017).[20] When training MAE, we mostly use the same hyperparameters, with the only exception being that we warmup the learning rate for 8,000 updates, instead of 4,000.[21]

At evaluation time, we apply early stopping based on development set loss, and then average the most recent 5 checkpoints of the model, following Vaswani et al. (2017).

Machine translation with IWSLT'14. The BASE model in this experiment is due to the fairseq codebase.[22] It mostly follows the transformer-base architecture, but uses a larger dropout rate (0.3 vs. 0.1), a smaller feed forward network hidden size (1,024 vs. 2,048), and a larger weight decay (10^{-4} vs. 0). We use 8,000 warmup updates.

Language modeling with WikiText-103. For the BASE model, we follow the model by Baevski and Auli (2019). The learning rate is warmed up for $240,000$ steps.

For all three experiments, the gating functions in our MAE model and the NOBCD baseline are implemented as tanh-MLPs. They have 256 hidden dimensions. We apply a batch normalization (Ioffe and Szegedy, 2015) to the input to the MLPs. We can see that the gating functions only have a small amount of parameters, accounting for less than 5% parameters of the full MAE model. A dropout of 0.1 is applied to the output of the first layer. No weight decay is used. ϕ are updated using SGD with a fixed learning rate 1, separate from the one for the rest part of the models. This aims to avoid using momentum-based optimizing algorithms (e.g., Adam) for the gating functions, which we empirically find helps alleviate the "rich gets richer" degeneracy.[23]

In the language modeling experiment, most recent 100 input vectors are averaged and then fed into the gating functions; while we average all the input vectors in the machine translation as the inputs to $\mathbf{g}(\cdot;\phi)$.

B Learning Curve Comparison for MAE and NOBCD

In §3 (footnote 4) we discuss an overfitting issue by jointly updating the experts and the gating function. This section empirically studies it. We compare the learning curves of BASE, NOBCD, and MAE-7 trained on the IWSLT14 dataset, plotted in Figure 3. The models are described in §4.1. We tune dropout and ℓ_2 regularization based on development performance. Other hyperparameters are the same for the compared models.

The training loss for NOBCD decreases much faster than BASE; however, on the development set, it never outperforms BASE, and the development loss starts increasing after epoch 40. MAE-7 finds a nice middle ground in terms of training loss. It outperforms both BASE and NOBCD on the validation set. This provides further evidence for the importance of BCD training.

C Addtional Results for §5.1

§5.1 describes a experiment with the MAE-7 model where we attribute the development instances of WMT14 to the experts maximizing the gating weights. Table 8 presents more results. The number of instances each expert receives is relatively balanced, and the trend is consistent across different layers.

[19]https://pytorch.org/; https://github.com/pytorch/fairseq

[20]https://github.com/pytorch/fairseq/issues/346

[21]Due to the randomness in random expert selection, we find that warming up learning rate more slowly helps stabilize early training.

[22]https://github.com/pytorch/fairseq/tree/master/examples/translation

[23]It is not entirely clear to us why using momentum-based optimization algorithms to learn the gating functions leads to degenerate solutions more often. One possible reason is that the accumulated momentum steers the gating functions to keeping selecting the experts they pick at the early stage of training.

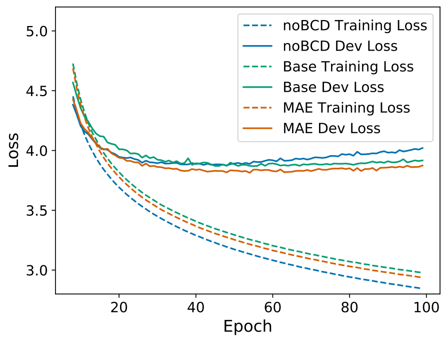

Figure 3: Learning curves of BASE, NOBCD, and MAE-7 (§B), trained on the IWSLT14 EN-DE using the same setup. NOBCD quickly fits the training data, but it does not outperform BASE on validation set. Trained with BCD, MAE finds a nice middle ground. For better readability, x-axis starts at epoch 8.

Layer	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**	**E7**	**E8**
1	13.1	13.9	8.9	16.1	10.3	15.3	10.1	11.6
2	13.8	14.5	10.7	10.8	15.4	7.9	16.0	10.9
3	14.0	14.4	12.4	10.6	14.3	9.8	15.4	9.0
4	14.5	13.7	10.4	8.3	15.1	11.8	11.2	15.1
5	11.9	13.8	13.7	15.7	10.1	16.4	6.9	11.5
6	12.9	10.0	12.4	14.6	9.5	15.2	15.7	9.8

Table 8: The percentage of WMT14 development instances attributed to each of the experts in MAE-7's encoder layers (§5.1).