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 mixture of attentive experts 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 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.

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. 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 mixture of attentive experts (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 attention heads into experts.

1 Our implementation is publicly available at https://github.com/Noahs-ARK/MAE.
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 \( \{f_i\}_{i=1}^k \) 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” \( g = [g_1, \ldots, g_k] \) determining the importance of each expert in the final decision:

\[
\text{MoE}(x) = \sum_{i=1}^k g_i(x) \cdot f_i(x). \tag{1}
\]

The gating function can be parameterized by, e.g., a neural network. We will also refer to \( g \) 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 \( X \), with each row being the vector representation of an input element. It contextualizes the input using a dot-product attention mechanism:

\[
\tilde{H}_i = \text{softmax} \left( QXK_i^\top X^\top \right) XV_i, \tag{2}
\]

where \( Q, K, \) and \( V \) are learned matrices, and the softmax normalizes row-wise. The outputs of attention heads are then concatenated and fed through a learned affine transformation:

\[
Z = \text{MultiHead}(X) = \left[ \tilde{H}_1; \ldots; \tilde{H}_h \right] W \tag{3}
\]

We now present a different computation equivalent to Eq. 3, aiming for a smoother transition into following sections. Let \( \tilde{H}_i = \tilde{H}_i W_i \), where \( W_i \) is a block submatrix of \( W \), i.e., \( W = [W_1 \otimes W_2 \otimes \cdots \otimes W_h]^{\top} \). Then

\[
Z = \left[ \tilde{H}_1; \ldots; \tilde{H}_h \right] W = \sum_{i=1}^h H_i. \tag{4}
\]

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., \( H_i = \tilde{H}_i W_i \)), 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:

\[
Z = \frac{1}{h-1} \sum_{i=1}^h (-1 + h) H_i
\]

\[
= \frac{1}{h-1} \left( -\sum_{i=1}^h H_i + \sum_{i=1}^h \sum_{j=1}^h H_j \right) \tag{5}
\]

\[
= \sum_{i=1}^h \frac{1}{h} \frac{h}{h-1} \left( -H_i + \sum_{j=1}^h H_j \right).
\]

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

\footnote{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. \( f_i(\cdot; \theta_i) \) is a parameterized function of the input, which calculates a sum of the outputs by all but the \( i \)th attention head. This is achieved by subtracting \( H_i \) from \( \sum_{j=1}^{h} H_j \), 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 \( g(\cdot; \phi) \) to complement the experts. More formally, the uniform responsibility \( 1/h \) in Eq. 5 is replaced by \( g(\cdot; \phi) \): given input \( X \), MAE outputs

\[
\sum_{i=1}^{h} g_i(X; \phi) \cdot f_i(X; \theta_i) \tag{6}
\]

Experts \( f_i \) are the same as those in Eq. 5.

\( g(\cdot; \phi) \) is parameterized with a multi-layer perceptron (MLP) followed by a softmax. It first averages \( X \) along the row (i.e., the sequence direction), and then feeds the results through a two-layer tanh-MLP. \( g(\cdot; \phi) \) outputs a normalized \( h \)-dimensional vector using a 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).

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 \( g(\cdot; \phi) \), fixing the experts; an \( F \) step fixes the gating function and updates one randomly selected expert \( f_i(\cdot; \theta_i) \). 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 \( g(X) \), according to which an expert \( i \) is then sampled (Algorithm 2). MAE computes the output with \( f_i \), which is then updated, without updating the gating function or other experts.

A non-differentiable sampling from \( g \) is involved in \( F \) steps. It does not create difficulties for the

---

4Besides the undesired degeneracy, we also find that the model suffers worse overfitting when \( \theta \) and \( \phi \) 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.

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

6In mini-batch training, which we use in the experiments, different experts can be sampled for different instances in a mini-batch. This is because \( g \) 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.
Specifically, selecting expert reminiscent of dropout (Srivastava et al., 2014).

Connections to dropout. From the dropout view, this translates to dropping mixing two or more 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 \textit{a priori}, while MAE uses a gating function \( g(\cdot; \phi) \) to calculate a learned, input-dependent dropout probability.

Algorithm 1 A G step update for MAE, with step size \( \eta \).

```plaintext
1: procedure MAEG(X)
2: \[ Z \leftarrow \sum_{i=1}^{h} g_{i}(X; \phi) \cdot f_{i}(X; \theta_{i}) \]
3: Forwardprop with Z and calculate \( \mathcal{L} \).
4: Calculate \( \nabla_{\phi} \mathcal{L} \) with backprop.
5: \[ \phi \leftarrow \phi - \eta \cdot \nabla_{\phi} \mathcal{L}. \]
6: end procedure
```

Algorithm 2 An F step update for MAE, with step size \( \eta \).

```plaintext
1: procedure MAEF(X)
2: Draw \( i \sim \text{Cat}(g(X; \phi)) \)
3: \[ Z \leftarrow f_{i}(X; \theta_{i}) \]
4: Forwardprop with Z and calculate \( \mathcal{L} \).
5: Calculate \( \nabla_{\theta_{i}} \mathcal{L} \) with backprop.
6: \[ \theta_{i} \leftarrow \theta_{i} - \eta \cdot \nabla_{\theta_{i}} \mathcal{L}. \]
7: end procedure
```

dropping head \( i \).\(^7\) 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 \( g \) is a constant vector with elements \( 1/h \), it recovers the head dropout, which is also explored by concurrent work (Fan et al., 2020).

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.

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

```plaintext
1: procedure BCD(\( \mathcal{D} = \{X_{i}\}_{i}, e \))
2: for \( X_{i} \in \mathcal{D} \) do
3: \udit{Take G steps every 5 epochs.}
4: if \( e \mod 5 = 0 \) then
5: \[ \text{MAEG}(X_{i}) \]
6: end if
7: \udit{Always do F step updates.}
8: \[ \text{MAEF}(X_{i}) \]
9: end for
10: end procedure
```

\(^7\)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 \( X_{i} \) the training instance, although they can also be the outputs of intermediate layers.

\(^9\)Recall from Eq. 5 that \( f_{i} \) includes all but head \( i \).
**MAE-6** is similar to **MAE-7**, but mixes \((8) = 28\) 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.

**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 descent 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

---

\(^{10}\)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_b2ck-kasdpkpM25jUXUN2X2UwM08

\(^{12}\)http://workshop2014.iwslt.org/

\(^{13}\)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).
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).

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
<th># Params.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Transformer</td>
<td>27.3</td>
<td>65M</td>
</tr>
<tr>
<td>Large Transformer</td>
<td>28.4</td>
<td>213M</td>
</tr>
<tr>
<td>Base</td>
<td>27.6</td>
<td>61M</td>
</tr>
<tr>
<td>†NOBCD</td>
<td>27.5</td>
<td>63M</td>
</tr>
<tr>
<td>†UNI-MAE-7</td>
<td>27.7</td>
<td>61M</td>
</tr>
<tr>
<td>†UNI-MAE-6</td>
<td>27.6</td>
<td>61M</td>
</tr>
<tr>
<td>‡MAE-7</td>
<td>28.4</td>
<td>63M</td>
</tr>
<tr>
<td>‡MAE-6</td>
<td>28.1</td>
<td>63M</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
<th># Params.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>34.6</td>
<td>39M</td>
</tr>
<tr>
<td>†NOBCD</td>
<td>34.8</td>
<td>41M</td>
</tr>
<tr>
<td>†UNI-MAE-7</td>
<td>34.9</td>
<td>39M</td>
</tr>
<tr>
<td>†UNI-MAE-6</td>
<td>35.0</td>
<td>39M</td>
</tr>
<tr>
<td>‡MAE-7</td>
<td>35.5</td>
<td>41M</td>
</tr>
<tr>
<td>‡MAE-6</td>
<td>35.4</td>
<td>41M</td>
</tr>
</tbody>
</table>

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).

<table>
<thead>
<tr>
<th>Model</th>
<th>Perplexity</th>
<th># Params.</th>
</tr>
</thead>
<tbody>
<tr>
<td>*BASE (B&amp;A, 2019)</td>
<td>18.70</td>
<td>247M</td>
</tr>
<tr>
<td>BASE (B&amp;A, 2019)</td>
<td>19.03</td>
<td>247M</td>
</tr>
<tr>
<td>†NOBCD</td>
<td>19.12</td>
<td>249M</td>
</tr>
<tr>
<td>†UNI-MAE-7</td>
<td>19.26</td>
<td>247M</td>
</tr>
<tr>
<td>†‡MAE-7</td>
<td><strong>18.71</strong></td>
<td><strong>249M</strong></td>
</tr>
</tbody>
</table>

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 task. 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 to 2,048 tokens before predicting any token at test time). However, we are not able to train MAE, nor replicate their results, under this setting—our GPUs have less memory, and it is impossible to even load a 3,072-token context chunk. 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).

Notes on context size. 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.
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: $H(g) = -\sum_{i=1}^{h} g_i \cdot \log g_i$, 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:

$$PMI(token_i, expert_j) = \log \frac{p(token_i, expert_j)}{p(token_i)p(expert_j)}.$$  

Table 6 lists the most indicative tokens of each expert (§5.1). Tokens attributed to Expert 2 are mostly computer science terminology; trends for other experts are less clear.

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

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
<th>Diff.</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNI-MAE-7</td>
<td>26.6</td>
<td>-</td>
</tr>
<tr>
<td>One random expert</td>
<td>25.8±0.2</td>
<td>↓ 0.8±0.2</td>
</tr>
<tr>
<td>NOBCD</td>
<td>26.7</td>
<td>-</td>
</tr>
<tr>
<td>Most specialized expert</td>
<td>26.0</td>
<td>↓ 0.7</td>
</tr>
<tr>
<td>MAE-7</td>
<td>27.1</td>
<td>-</td>
</tr>
<tr>
<td>Most specialized expert</td>
<td>26.8</td>
<td>↓ 0.3</td>
</tr>
</tbody>
</table>

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).

<table>
<thead>
<tr>
<th>Expert 1</th>
<th>Expert 2</th>
<th>Expert 3</th>
<th>Expert 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>neumann</td>
<td>bell</td>
<td>candidacy</td>
<td>veil</td>
</tr>
<tr>
<td>depted</td>
<td>zero</td>
<td>submission</td>
<td>monument</td>
</tr>
<tr>
<td>rental</td>
<td>computing</td>
<td>decentralized</td>
<td>fox</td>
</tr>
<tr>
<td>worthy</td>
<td>decentralized</td>
<td>landloands</td>
<td>unerved</td>
</tr>
<tr>
<td>landloands</td>
<td>computational</td>
<td>roles</td>
<td>remainder</td>
</tr>
<tr>
<td>experts</td>
<td>inevitably</td>
<td>held</td>
<td>unabed</td>
</tr>
<tr>
<td>reserve</td>
<td>bet</td>
<td>analyst</td>
<td>ornate</td>
</tr>
<tr>
<td>pending</td>
<td>punk</td>
<td>venues</td>
<td>anticipating</td>
</tr>
</tbody>
</table>

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.

---

16The 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.

17We observe similar trends in other layers. See Appendix C for more details.
much larger WMT14 dataset. We compare three finetuning methods:

- **FtG** finetunes the gating functions’ parameters (i.e., $\phi$), keeping the rest frozen.
- **FtG+** updates the parameter matrix $W$ in Eq. 4 in addition to $\phi$. 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.

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.

## 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 baseline models 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 McCalla, 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


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


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. $\phi$ 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 $g(\cdot; \phi)$.

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 $\ell_2$ 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 Additional 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.

\(^{23}\)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.

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