

Supplement to “Tight Clusters Make Specialized Experts”

Table of Contents

A	Technical Proofs	14
A.1	Proof of Theorem 1	14
A.2	Proof of Proposition 1	16
A.2.1	Proof of Lemma 1	16
A.2.2	Proof of Lemma 2	16
A.3	Proof of Proposition 2	17
B	Implementation Procedure and Computational Efficiency	19
C	Experimental Details and Additional Experiments	19
C.1	Language Modeling	19
C.1.1	Datasets	19
C.1.2	Model, Optimizer, & Train Specification	19
C.2	Image Classification	20
C.2.1	Datasets and Attacks	20
C.2.2	Model, Optimizer, & Train Specification	21
C.3	Adversarial Attack At Higher Perturbation Budget	21
C.4	Cluster Visualization	21
C.5	Ablation Studies	22
C.5.1	Measures of Dispersion	22
C.5.2	Layer Placement	22
C.5.3	Random Ablation	23
C.6	Cluster Weight Mixing	24
C.7	Adaptive Clustering Integration into Soft Mixture of Experts	24
C.8	Image Classification in Swin Transformer Base Configuration	25
C.9	Router Stability	25
C.10	Dynamic Routing	25
D	Broader Impact	26

A TECHNICAL PROOFS

A.1 PROOF OF THEOREM 1

To begin with, we present the following lemma to show the existence of constants α_k for $k \in [E]$ that satisfy Eqn. 7:

Lemma 3. *For any $\lambda > 0$, Eqn. 7 has exactly d real solutions with respect to α_k .*

Proof of Lemma 3. Without loss of generality, assume that $s_{1k} \geq s_{2k} \geq \dots \geq s_{dk}$. Denote

$$\varphi(\alpha) := \sum_{q \in [d]} \frac{1}{s_{qk} + \alpha} - \frac{d}{\lambda}. \quad (12)$$

Then, the existence of solutions to Eqn. 7 is equivalent to the condition $\varphi(\alpha_l) = 0$. Note that $\varphi(\alpha)$ is a strictly decreasing function in its connected continuity domains since

$$\varphi'(\alpha) = - \sum_{q \in [d]} \frac{1}{(s_{qk} + \alpha)^2} < 0 \quad (13)$$

for all $\alpha \in \mathbb{R} \setminus \{-s_{1k}, \dots, -s_{dk}\}$. Further, we observe that

$$\lim_{\alpha \rightarrow -s_{qk}^-} \varphi(\alpha) = -\infty, \quad \lim_{\alpha \rightarrow -s_{qk}^+} \varphi(\alpha) = +\infty \quad (14)$$

for all $q \in [d]$, and

$$\lim_{\alpha \rightarrow \pm\infty} \varphi(\alpha) = -\frac{d}{\lambda} < 0. \quad (15)$$

Now consider the domain of continuity of $\varphi(\alpha)$, namely $(-\infty, -s_{1k}) \cup (-s_{1k}, -s_{2k}) \cup \dots \cup (-s_{dk}, \infty)$. Due to the monotonicity and limits 14 & 15, there exists a unique solution in each of the intervals except for $(-\infty, -s_{1k})$ where the function is always strictly negative, thus, yielding d roots in total. \square

Now we follow up with the main proof of this section.

Proof of Theorem 1. First, let $\mathcal{I}_k := \{i : r(i) = k\}$ for convenience. Now let us restate the clustering optimization problem (4) here once again:

$$\begin{aligned} \min_{\mathbf{w}_k} Q(c, \{\mathbf{w}_k\}_{k \in [E]}) &= \sum_{k \in [E]} \frac{1}{N_k^2} \sum_{i, j \in \mathcal{I}_k} \sum_{q \in [d]} \left(w_{qk} \rho_{ijq} + \frac{\lambda}{d} \log \frac{1}{dw_{qk}} \right), \\ \text{such that } \sum_{q \in [d]} w_{qk} &= 1, \quad \forall k \in [E], \end{aligned} \quad (16)$$

where we have immediately used the fact that

$$D_{\text{KL}}(\mathbf{u} \parallel \mathbf{w}_k) = \sum_{q \in [d]} \frac{1}{d} \log \frac{1/d}{w_{qk}}. \quad (17)$$

Also, note that

$$\begin{aligned} \sum_{q \in [d]} \left(w_{qk} \rho_{ijq} + \lambda \frac{1}{d} \log \frac{1}{dw_{qk}} \right) &= \sum_{q \in [d]} \left(w_{qk} \rho_{ijq} - \lambda \frac{1}{d} \log(dw_{qk}) \right) \\ &= \sum_{q \in [d]} \left(w_{qk} \rho_{ijq} - \frac{\lambda}{d} \log w_{qk} \right) - \lambda \log d. \end{aligned} \quad (18)$$

We can ignore the term $\lambda \log d$ since it does not depend on the optimization variable. Method of Lagrange multipliers turns this constrained optimization problem into the following unconstrained counterpart:

$$\min_{\mathbf{w}_k, \boldsymbol{\alpha}} \mathcal{L}(c, \{\mathbf{w}_k\}_{k \in [E]}, \boldsymbol{\alpha}) = \sum_{k \in [E]} \frac{1}{N_k^2} \sum_{i, j \in \mathcal{I}_k} \sum_{q \in [d]} \left(w_{qk} \rho_{ijq} - \frac{\lambda}{d} \log w_{qk} \right) + \sum_{k \in [E]} \alpha_k \left(\sum_{q \in [d]} w_{qk} - 1 \right),$$

where $\boldsymbol{\alpha} = [\alpha_1 \dots \alpha_L]^\top$ is the vector of Lagrange multipliers. Note that the last optimization problem can be separated into the following L independent optimization subproblems:

$$\min_{\mathbf{w}_k, \boldsymbol{\alpha}} \mathcal{L}_k(c, \mathbf{w}_k, \boldsymbol{\alpha}) = \frac{1}{N_k^2} \sum_{i, j \in \mathcal{I}_k} \sum_{q \in [d]} \left(w_{qk} \rho_{ijq} - \frac{\lambda}{d} \log w_{qk} \right) + \alpha_k \left(\sum_{q \in [d]} w_{qk} - 1 \right),$$

for $k \in [E]$. Since the objective function is a positive combination of convex functions, the optimization problem is also convex. By setting the derivatives of \mathcal{L}_k with respect to both optimization variables to 0, we obtain the following system of equations:

$$\begin{cases} \frac{\partial \mathcal{L}_k}{\partial w_{qk}} = s_{qk} - \frac{\lambda}{d} \frac{1}{w_{qk}} + \alpha_k = 0, \\ \frac{\partial \mathcal{L}_k}{\partial \alpha_k} = \sum_{q \in [d]} w_{qk} - 1 = 0 \end{cases}$$

for all $k \in [E]$, where s_{qk} is the data dispersion measure defined in the theorem statement. The first equation yields

$$w_{qk} = \frac{\lambda}{d} \frac{1}{s_{qk} + \alpha_k}, \quad (19)$$

where α_k is found from $\sum_{q \in [d]} w_{qk} = 1$ which in fact gives

$$\sum_{q \in [d]} \frac{1}{s_{qk} + \alpha_k} = \frac{d}{\lambda} \quad (20)$$

for all $k \in [E]$ as desired. \square

A.2 PROOF OF PROPOSITION 1

Since Proposition 1 is a composition of Lemma 1 and Lemma 2, we proceed by providing their proofs.

A.2.1 PROOF OF LEMMA 1

Proof of Lemma 1. Notice that we can expand inequality (1) as

$$\sum_{i \in [d]} m_i \delta \mu_i^2 \geq \sum_{i \in [d]} \delta \mu_i^2,$$

where we let $\delta \mu := \mu_b - \mu_a$. Since M_a entries are mean-scaled, we can rewrite them as

$$m_i = \frac{dm'_i}{\sum_{j \in [d]} m'_j} \quad (21)$$

for some initial dispersion estimates $\{m'_j\}_{j \in [d]}$. Without loss of generality, assume that $[d']$ is the set of dimension indices for which the dispersions are relatively much smaller than those in the rest of the dimensions in the sense that $m'_i \gg m'_j$ for any $i \in [d']$ and $j \in [d] \setminus [d']$. Then, there exists a positive $\alpha \ll 1/2$ such that $\sum_{i \in [d']} m_i > d - \alpha$ and $\sum_{i \in [d] \setminus [d']} m_i < \alpha$. By the assumption that clusters are best-separated along the features for which they cluster tightly, this means that the weight matrix M_a maximizes the contribution of largest d' terms in $\sum_{i \in [d]} m_i \delta \mu_i^2$ corresponding to individual feature-wise distances in dimensions where the feature dispersions are the smallest instead of giving uniform weights to all dimensions, which leads to inequality (1). \square

A.2.2 PROOF OF LEMMA 2

Proof of Lemma 2. Since we use the \mathcal{L}_2 distance between the token \mathbf{h} and μ_c as a similarity metric, we assign cluster g_{k^*} to the token \mathbf{h}' iff $\|\mathbf{h}' - \mu_{k^*}\| \leq \|\mathbf{h}' - \mu_k\|$. Assume that the token \mathbf{h}' is a noisy observation of an underlying true token \mathbf{h} which actually originates from cluster g_{k^*} . Then, the token \mathbf{h}' can be decomposed as $\mathbf{h}' = \mathbf{h} + \epsilon$ for a random noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \Sigma_\epsilon)$. Now define the decision variable $\mathcal{D}(\mathbf{h}') := \|\mathbf{h}' - \mu_{k^*}\|^2 - \|\mathbf{h}' - \mu_k\|^2$ which turns the clustering condition to $\mathcal{D}(\mathbf{h}') \leq 0$ for the cluster g_{k^*} . Let us analyze the decision variable \mathcal{D} as a random variable where randomness may come from the underlying sampling strategy and noise. Note that

$$\begin{aligned} \mathcal{D}(\mathbf{h}') &= \|\mathbf{h} + \epsilon - \mu_{k^*}\|^2 - \|\mathbf{h} + \epsilon - \mu_k\|^2 \\ &= \|\mathbf{h} - \mu_{k^*}\|^2 - \|\mathbf{h} - \mu_k\|^2 + 2(\mu_k - \mu_{k^*})^\top \epsilon \\ &= \mathcal{D}(\mathbf{h}) + 2\delta \mu^\top \epsilon, \end{aligned} \quad (22)$$

where $\delta\boldsymbol{\mu} := \boldsymbol{\mu}_k - \boldsymbol{\mu}_{k^*}$. Due to the assumption that \mathbf{h} is drawn from the distribution g_{k^*} , it can be rewritten as $\mathbf{h} = \boldsymbol{\mu}_{k^*} + \boldsymbol{\nu}$ with $\boldsymbol{\nu} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{k^*})$. Then for the first term in Eqn. 22, we have

$$\begin{aligned}\mathcal{D}(\mathbf{h}) &= \|\mathbf{h} - \boldsymbol{\mu}_{k^*}\|^2 - \|\mathbf{h} - \boldsymbol{\mu}_k\|^2 \\ &= \delta\boldsymbol{\mu}^\top (2\mathbf{h} - \boldsymbol{\mu}_{k^*} - \boldsymbol{\mu}_k) \\ &= \delta\boldsymbol{\mu}^\top (2\boldsymbol{\nu} - \delta\boldsymbol{\mu}) \\ &= 2\delta\boldsymbol{\mu}^\top \boldsymbol{\nu} - \|\delta\boldsymbol{\mu}\|^2.\end{aligned}\quad (23)$$

Substituting this back into Eqn. 22, we get

$$\mathcal{D}(\mathbf{h}') = 2\delta\boldsymbol{\mu}^\top (\boldsymbol{\nu} + \boldsymbol{\epsilon}) - \|\delta\boldsymbol{\mu}\|^2. \quad (24)$$

This shows that $\mathcal{D}(\mathbf{h}') \sim \mathcal{N}(-\|\delta\boldsymbol{\mu}\|^2, 4\delta\boldsymbol{\mu}^\top (\boldsymbol{\Sigma}_{k^*} + \boldsymbol{\Sigma}_\epsilon) \delta\boldsymbol{\mu})$. Since $\mathcal{D}(\mathbf{h}')$ follows a normal distribution with the derived parameters, the probability that \mathbf{h}' is assigned to cluster g_{k^*} is given by

$$\Pr(\text{correct cluster}) = \Pr(\mathcal{D}(\mathbf{h}) \leq 0) = \Phi\left(\frac{\|\delta\boldsymbol{\mu}\|^2}{2\sqrt{\delta\boldsymbol{\mu}^\top (\boldsymbol{\Sigma}_{k^*} + \boldsymbol{\Sigma}_\epsilon) \delta\boldsymbol{\mu}}}\right), \quad (25)$$

where Φ denotes the CDF of normal distribution as usual. Since Φ is an increasing function, the probability that the noisy token \mathbf{h} is assigned to the correct cluster is proportional to the distance between the cluster centroids and inverse proportional to the covariance matrices of the cluster and the additive noise. On the other hand, for the incorrect clustering probability, we have

$$\Pr(\text{incorrect cluster}) = 1 - \Phi\left(\frac{\|\delta\boldsymbol{\mu}\|^2}{2\sqrt{\delta\boldsymbol{\mu}^\top (\boldsymbol{\Sigma}_{k^*} + \boldsymbol{\Sigma}_\epsilon) \delta\boldsymbol{\mu}}}\right) \quad (26)$$

as claimed. \square

A.3 PROOF OF PROPOSITION 2

Proof of Proposition 2. Let the router be given by g and let the softmax function be given by $g_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^d$, parameterized by expert embeddings $\{e_i\}_{i \in [E]}$. The network loss depends on expert embeddings only through the router function g . We shall explore the exclusive contribution of each expert embedding in minimizing $\mathcal{L}^{\text{ACMoE}}$. In order to do this, we look at the network loss as a scalar function of i^{th} expert embedding vector while treating all other network parameters as fixed. Then, we can write $\mathcal{L}^{\text{ACMoE}} : \mathbb{R}^d \rightarrow \mathbb{R}$ such that $\mathcal{L}^{\text{ACMoE}} = \mathcal{L}^{\text{ACMoE}}(g_\theta(e_i))$. For simplicity, we shall omit the subscript θ . The gradient that comes from back-propagation is then given by

$$\nabla_{e_i} \mathcal{L}^{\text{ACMoE}} = (\nabla_g \mathcal{L}^{\text{ACMoE}})^\top \nabla_{e_i} g, \quad (27)$$

where $\nabla_{e_i} g \in \mathbb{R}^{d \times d}$ denotes the Jacobian matrix of g since for $g_k := (g_\theta(e_i))_k$, we can write

$$\frac{\partial}{\partial e_{is}} \mathcal{L}^{\text{ACMoE}}(g_1, \dots, g_d) = \sum_k \frac{\partial \mathcal{L}^{\text{ACMoE}}}{\partial g_k} \frac{\partial g_k}{\partial e_{is}}. \quad (28)$$

Note that for $g_k = \text{softmax}(\mathbf{h}^\top \mathbf{M} e_k)$, we have

$$\frac{\partial g_k}{\partial e_{is}} = m_s h_s g_k (\delta_{ki} - g_i) = m_s h_s b_{ki}. \quad (29)$$

Then, the element of the Hessian matrix of the network loss at index $(s, t) \in [d] \times [d]$ can be written as

$$\begin{aligned}\mathbf{H}_{st}^{(i)}(\mathcal{L}^{\text{ACMoE}}) &= \frac{\partial^2 \mathcal{L}^{\text{ACMoE}}}{\partial e_{is} \partial e_{it}} = \frac{\partial}{\partial e_{it}} \sum_k \frac{\partial \mathcal{L}^{\text{ACMoE}}}{\partial g_k} \frac{\partial g_k}{\partial e_{is}} \\ &= \sum_k \left(\sum_j \frac{\partial^2 \mathcal{L}^{\text{ACMoE}}}{\partial g_k \partial g_j} \frac{\partial g_j}{\partial e_{it}} \right) \frac{\partial g_k}{\partial e_{is}} + \frac{\partial \mathcal{L}^{\text{ACMoE}}}{\partial g_k} \frac{\partial^2 g_k}{\partial e_{is} \partial e_{it}} \\ &= m_s h_s m_t h_t \left[\sum_k \left(\sum_j \frac{\partial^2 \mathcal{L}^{\text{ACMoE}}}{\partial g_k \partial g_j} b_{ji} \right) b_{ki} + \frac{\partial \mathcal{L}^{\text{ACMoE}}}{\partial g_k} b'_{ki} \right] \\ &= m_s h_s m_t h_t B_i,\end{aligned}\quad (30)$$

where B_i is some constant that depends only on index i . Due to Eqn. 30, the Hessian takes the following matrix form

$$\mathbf{H}^{(i)} = B_i(\mathbf{M}\mathbf{h})(\mathbf{M}\mathbf{h})^\top. \quad (31)$$

Taking expectation from both sides, we obtain

$$\mathbb{E}_{\mathbf{h} \sim (\mu, \Sigma)} [\mathbf{H}^{(i)}] = B_i \mathbb{E}_{\mathbf{h} \sim (\mu, \Sigma)} [\mathbf{M}(\mathbf{h}\mathbf{h}^\top)\mathbf{M}] = B_i \mathbf{M}(\Sigma)\mathbf{M}, \quad (32)$$

where we assume \mathbf{h} is centered. Now recall that $\mathbf{M} = \text{diag}(m_1, \dots, m_d)$ where for each i , $m_i \sim 1/\sqrt{\Sigma_{ii}}$ holds. Assume that the covariance matrix Σ is symmetric positive definite. Then, it is diagonalizable as $\Sigma = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ with $\mathbf{\Lambda} = \text{diag}(\lambda_1, \dots, \lambda_d)$, a diagonal matrix with eigenvalues of Σ . With the transformation \mathbf{M} , we get

$$\mathbf{M}\Sigma\mathbf{M} = \mathbf{M}\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top\mathbf{M} = \mathbf{U}\mathbf{M}\mathbf{\Lambda}\mathbf{M}\mathbf{U}^\top \quad (33)$$

$$= \mathbf{U} \begin{bmatrix} m_1^2 \lambda_1 & & \\ & \ddots & \\ & & m_d^2 \lambda_d \end{bmatrix} \mathbf{U}^\top. \quad (34)$$

Since the eigenvalues capture the variances along the principal components of the covariance matrix, m_i^2 , as a reciprocal of a measure of dimension-wise dispersion, is reasonably correlated with $1/\lambda_i$, as demonstrated by Lemma 4, implying $\lambda_j \leq \lambda_i \implies m_j \geq m_i$ with high probability. Therefore, we obtain that

$$\kappa(\mathbf{M}\Sigma\mathbf{M}) = \frac{\lambda_{\max}(\mathbf{M}\Sigma\mathbf{M})}{\lambda_{\min}(\mathbf{M}\Sigma\mathbf{M})} \approx \frac{m_{\min}^2 \lambda_{\max}(\Sigma)}{m_{\max}^2 \lambda_{\min}(\Sigma)} \leq \kappa(\Sigma), \quad (35)$$

which implies the claim. \square

Lemma 4 (Correlation between dimension-wise variances and covariance eigenvalues). *Let $\{\mathbf{b}_i\}_{i \in [d]}$ be the set of normalized basis vectors of \mathbb{R}^d . Consider a symmetric positive definite covariance matrix Σ and its unit eigenvectors $\{\mathbf{v}_i\}_{i \in [d]}$. Assume that the eigenvector \mathbf{v}_i is a reasonably small perturbation of the basis vector \mathbf{b}_i such that $\mathbf{v}_i^\top \mathbf{b}_i \geq 1 - \epsilon$ for all $i \in [d]$ and a small constant $\epsilon > 0$. Then, for all $i \in [d]$, we have*

$$|\lambda_i - \Sigma_{ii}| \leq \epsilon \cdot \max_{j \neq i} |\lambda_i - \lambda_j|, \quad (36)$$

where $\{\lambda_i\}_{i \in [d]}$ is the set of ordered eigenvalues of Σ corresponding to eigenvectors $\{\mathbf{v}_i\}_{i \in [d]}$.

Proof of Lemma 4. Note that each diagonal element of the SPD covariance matrix Σ can be written as

$$\Sigma_{ii} = \mathbf{b}_i^\top \Sigma \mathbf{b}_i = \mathbf{b}_i^\top \left(\sum_{j \in [d]} \lambda_j \mathbf{v}_j \mathbf{v}_j^\top \right) \mathbf{b}_i = \sum_{j \in [d]} \lambda_j (\mathbf{v}_j^\top \mathbf{b}_i)^2. \quad (37)$$

Then, the difference on the left hand side of Eqn. 36 can be bounded as

$$\begin{aligned} |\lambda_i - \Sigma_{ii}| &= \left| \lambda_i - \sum_{j \in [d]} \lambda_j (\mathbf{v}_j^\top \mathbf{b}_i)^2 \right| = \left| \lambda_i (1 - (\mathbf{v}_i^\top \mathbf{b}_i)^2) - \sum_{j \neq i} \lambda_j (\mathbf{v}_j^\top \mathbf{b}_i)^2 \right| \\ &= \left| \lambda_i \sum_{j \neq i} (\mathbf{v}_j^\top \mathbf{b}_i)^2 - \sum_{j \neq i} \lambda_j (\mathbf{v}_j^\top \mathbf{b}_i)^2 \right| \end{aligned} \quad (38)$$

$$\begin{aligned} &= \left| \sum_{j \neq i} (\lambda_i - \lambda_j) (\mathbf{v}_j^\top \mathbf{b}_i)^2 \right| \\ &\leq \max_{j \neq i} |\lambda_i - \lambda_j| \sum_{j \neq i} (\mathbf{v}_j^\top \mathbf{b}_i)^2 \\ &= \max_{j \neq i} |\lambda_i - \lambda_j| (1 - (\mathbf{v}_i^\top \mathbf{b}_i)^2) \\ &\leq \epsilon \max_{j \neq i} |\lambda_i - \lambda_j|, \end{aligned} \quad (39)$$

where we used the fact that

$$\sum_{j \in [d]} (\mathbf{v}_j^\top \mathbf{b}_i)^2 = \left(\sum_{j=1}^n (\mathbf{v}_j^\top \mathbf{b}_i) \mathbf{v}_j \right)^\top \left(\sum_{k=1}^n (\mathbf{v}_k^\top \mathbf{b}_i) \mathbf{v}_k \right) = \mathbf{b}_i^\top \mathbf{b}_i = 1$$

to obtain Eqn. 38 and Eqn. 39 since the eigenvectors of Σ are orthonormal. \square

B IMPLEMENTATION PROCEDURE AND COMPUTATIONAL EFFICIENCY

Training and Inference. Given the AC routing scheme requires requires the expert assignment per token from the previous layer, we can only implement AC routing from the second layer on. We incorporate AC routing into both training and inference stages. This is because, firstly, AC routing is designed to offer improvements to both clean and contaminated data, and so even in the presence of completely clean train and test data, it is advantageous to incorporate the AC method into both stages. Secondly, it is commonplace to encounter data contamination only at the test stage and indeed highly possible to encounter it in train as well. Therefore, in the interest of robustness as well, AC routing is incorporated into both stages.

Computational Efficiency. Computing the required $\{w_k\}_{k \in [E]}$ for number of experts E requires no learnable parameters and is obtained simply by computing the mean absolute deviation for each set of tokens assigned to the k^{th} expert. This can be computed using just two computations of the mean – once for the mean per cluster and once again for the mean of the absolute deviations per cluster – done in parallel over all clusters using `torch.index_reduce()` and is of the order $\mathcal{O}(2nd) = \mathcal{O}(n)$ for n tokens. Hence the upper-bound time complexity of the MoE layer is unaffected. We provide in Table 7 additional efficiency analysis in terms of throughput, max GPU memory allocated, and parameters which shows no significant efficiency loss compared to baseline MoE architectures.

C EXPERIMENTAL DETAILS AND ADDITIONAL EXPERIMENTS

C.1 LANGUAGE MODELING

C.1.1 DATASETS

WikiText-103. The WikiText-103³ dataset contains around 268K words and its training set consists of about 28K articles with 103M tokens. This corresponds to text blocks of about 3600 words. The validation set and test sets consist of 60 articles with 218K and 246K tokens respectively.

EnWik-8. The EnWik-8 dataset is a byte-level dataset of 100 million bytes derived from Wikipedia that, in addition to English text, also includes markup, special characters, and text in other languages. EnWik-8 contains 90M characters for training, 5M for validation, and 5M for testing.

Stanford Sentiment Treebank-2. The Stanford Sentiment Treebank-2 (SST2) (Socher et al., 2013) is a 2 class corpus with fully labeled parse trees for analysis of the compositional effects of sentiment in language. The dataset consists of 11,855 single sentences extracted from movie reviews. It was parsed with the Stanford parser and includes 215,154 unique phrases from the parse trees, each annotated by 3 human judges.

Stanford Sentiment Treebank-5. Stanford Sentiment Treebank-5 (SST5) (Socher et al., 2013) is a 5 class dataset used for sentiment analysis. It consists of 11,855 single sentences extracted from movie reviews. It includes 215,154 unique phrases from parse trees, each annotated by 3 human judges. Phrases are classified as negative, somewhat negative, neutral, somewhat positive, or positive.

Banking-77. Banking-77 (B77) (Casanueva et al., 2020) is a highly fine-grained 77 class classification dataset comprising 13083 customer service queries labelled with 77 intents.

C.1.2 MODEL, OPTIMIZER, & TRAIN SPECIFICATION

Models. We use as backbones the Switch Transformer (Fedus et al., 2022) and Generalist Language Model (Du et al., 2022). Table 8 contains the specification over self-attention (SA) layers, feed-forward network (FFN) layers, Mixture-of-Experts (MoE) layers, attention span (Att. Span),

³www.salesforce.com/products/einstein/ai-research/the-wikitext-dependency-language-modeling-dataset/

embedding size and parameter count for both backbones at small and medium configurations for each pretraining task. All backbones use 16 experts with top-2 expert routing.

Table 8: Language Modeling Backbone Specifications

Model	SA Layers	FFN Layers	MoE Layers	Att. Span	Embed Size	Params
<i>WikiText-103 Pretrain</i>						
Switch-small	3	-	3	256	128	70M
Switch-medium	6	-	6	1024	352	216M
GLaM-small	6	3	3	2048	144	79M
GLaM-medium	12	6	6	2048	352	220M
<i>EnWik-8 Pretrain</i>						
Switch	8	-	8	2048	352	36M

Optimizer. All experiments use Adam with a base learning rate of 0.0007. Small configurations use 3000 iterations of learning rate warmup while medium configurations use 4000 iterations.

Pretrain Specification. For WikiText-103 pretraining, small Switch backbones are trained for 40 epochs with a batch size of 96 and medium Switch backbones are trained for 80 epochs with a batch size of 48. Small GLaM backbones are trained for 60 epochs with a batch size of 48 and medium GLaM backbones are trained for 120 epochs with a batch size of 48. We use 0.01 auxiliary load balancing loss.

For EnWik-8 pretraining, both Switch and GLaM backbones are trained for 80 epochs with batch size 48. We use 0.01 auxiliary load balancing loss.

Finetune Specification. For SST2 and SST5 finetuning, we finetune for 5 epochs using Adam and a base learning rate of 0.001 without warmup and a batch size of 16. For B77 we finetune for 50 epochs using Adam and a base learning rate of 0.00001 without warmup and a batch size of 16.

Compute Resources. All models are trained, evaluated, and finetuned on four NVIDIA A100 SXM4 40GB GPUs.

C.2 IMAGE CLASSIFICATION

C.2.1 DATASETS AND ATTACKS

ImageNet-1K. We use the full ImageNet dataset that contains 1.28M training images and 50K validation images. The model learns to predict the class of the input image among 1000 categories. We report the top-1 and top-5 accuracy on all experiments.

ImageNet-A/O/R. ImageNet-A (Hendrycks et al., 2021b) contains real-world adversarially filtered images that fool current ImageNet classifiers. A 200-class subset of the original ImageNet-1K’s 1000 classes is selected so that errors among these 200 classes would be considered egregious, which cover most broad categories spanned by ImageNet-1K.

ImageNet-O (Hendrycks et al., 2021b) contains adversarially filtered examples for ImageNet out-of-distribution detectors. The dataset contains samples from ImageNet-22K but not from ImageNet1K, where samples that are wrongly classified as an ImageNet-1K class with high confidence by a ResNet-50 are selected.

Imagenet-R (Hendrycks et al., 2021a) contains various artistic renditions of object classes from the original ImageNet dataset, which is discouraged by the original ImageNet. ImageNet-R contains 30,000 image renditions for 200 ImageNet classes, where a subset of the ImageNet-1K classes is chosen.

Adversarial Attacks. We use produce corrupted ImageNet samples using white box attacks fast gradient sign method (FGSM) (Goodfellow et al., 2014) and projected gradient descent (PGD) (Madry et al., 2017), and black box simultaneous perturbation stochastic approximation (SPSA) (Uesato et al., 2018). FGSM and PGD use a perturbation budget of $1/255$ while SPSA uses a perturbation budget 1. All attacks perturb under l_∞ norm. PGD and uses 20 steps with step size of 0.15 and SPSA uses 20 iterations.

C.2.2 MODEL, OPTIMIZER, & TRAIN SPECIFICATION

Models. Our results are based off of the Swin Transformer (Liu et al., 2021) architecture. This backbone uses 4 base layers of depth 2, 2, 18, and 2. The first two base layers each contain 2 self-attention layers and 2 feed-forward layers. The third base layer contains 18 self-attention layers with alternating feed-forward and MoE layers. The final base layer contains 2 self-attention layers with one feed-forward and one MoE layer. The embedding dimension is 96 and the heads per base layer are 3, 6, 12, and 24. We use 16 total experts and present results for both top-1 and top-2 expert routing. The total parameter count is 280M.

Optimizer. We use AdamW with a base learning rate of $1.25e-4$, minimum learning rate of $1.25e-7$, 0.1 weight decay and cosine scheduling.

Train Specification. We train for 60 epochs with a batch size of 128 and 0.1 auxiliary balancing loss.

Compute Resources. All models are trained and evaluated on four NVIDIA A100 SXM4 40GB GPUs.

C.3 ADVERSARIAL ATTACK AT HIGHER PERTURBATION BUDGET

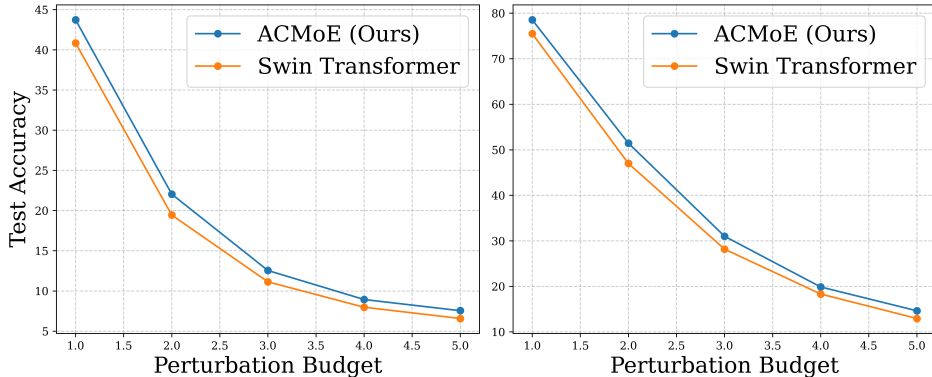


Figure 3: ACMoE and Swin Transformer under PGD attack at increasing perturbation budgets. ACMoE widens its performance gain over Swin at increasingly severe attacks in both top-1 test accuracy (left) and top-5 test accuracy (right), starting at approximately 7% improvement at $1/255$ and ending at just over 10% at $5/255$.

Figure 3 shows that for PGD perturbation budgets $1/255$ through to $5/255$, ACMoE widens its already substantive robust performance gain over Swin, with top-1 and top-5 test accuracy improvements increasing from 7% to approximately 10%.

C.4 CLUSTER VISUALIZATION

We pass random ImageNet batches through Swin and ACMoE and plot the representations along with their assigned experts, using t-sne to represent the high dimensional data in 2 dimensions. The result is shown in Fig. 4, where we see Swin learns overlapping and indistinguishable expert clusters. ACMoE, on the other hand, performs better in learning the clusters, producing much clearer and better-distinguished clusters.

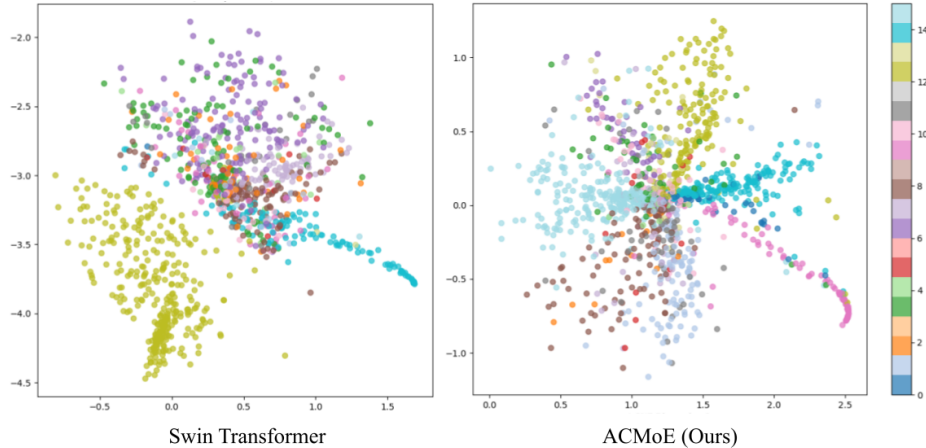


Figure 4: Cluster Visualization on ImageNet. Each token is represented as a point and colored by its assigned expert. **Left:** Swin identifies one cluster clearly (yellow/gold) but otherwise fails to distinguish remaining clusters **Right:** ACMoE learns better-defined expert clusters.

Table 9: Ablation on Measure of Spread in Switch Transformer (Fedus et al., 2022)

Measure of Spread	Test PPL (\downarrow)
Variance	34.87
MAD	34.42

Table 10: Ablation on Layer Placement in Switch Transformer (Fedus et al., 2022)

Layer Placement	Test PPL (\downarrow)
Back Half	34.95
Alternating	34.80
Skip 1	34.42
Full	34.88

C.5 ABLATION STUDIES

C.5.1 MEASURES OF DISPERSION

We present in Tables 9 and 11 results for Switch-ACMoE and Swin-ACMoE when changing the measure of dispersion used in the AC routing transformation (Definition 1) from mean absolute deviation (MAD) to variance. We see mean absolute deviation outperforms variance as a measure of spread. This is an intuitive finding given that squared distances, as used in variance computations, are highly sensitive to outliers. Using mean absolute deviation as an alternative measure of spread reduces this issue and produces a more robust estimate of dispersion. We note that MAD is not the only robust measure of spread. We conjecture that taking interquartile range as an additionally robust measure of spread may produce good results in both clean and contaminated data. We, however, leave this interesting direction to future research as interquartile range poses implementation challenges as it requires designing concurrent linear scans over the expert clusters. MAD, by contrast, requires just two computations of the mean which is easily parallelizable using `torch.index_reduce()`.

C.5.2 LAYER PLACEMENT

We consider the effect of layer placement in the Switch-medium configuration and in the Swin Transformer (see Sections C.1.2 and C.2.2 for the full model specifications). In particular, Switch is a 6 layer model and Swin is a 24 layer model. With regard to Swin, we focus on the deepest block of depth 18 to implement our ACMoE layers. This is due to the change in embedding size between base layers, meaning we are restricted to this base layer of depth 18. Note further that Swin only uses MoE layers in an alternating pattern with feed-forward networks between each MoE layer. For example, for Switch, a full ACMoE specification would mean placing ACMoE on layers 2,3,4,5,6. For Swin, a full specification means placing ACMoE on layers 4,6,8,10,12,14,16,18. To examine the effect of layer placement we consider the following models:

Table 11: Ablation on Measure of Spread in Swin Transformer

Measure of Spread	Test Acc.	
	Top 1	Top 5
<i>Swin-Top1 (Liu et al., 2021)</i>		
Variance	75.06	92.49
MAD	75.39	92.56
<i>Swin-Top2 (Liu et al., 2021)</i>		
Variance	76.11	93.08
MAD	76.31	93.14

Table 12: Ablation on Layer Placement in Swin Transformer

Layer Placement	Test Acc.	
	Top 1	Top 5
<i>Swin-Top1 (Liu et al., 2021)</i>		
Back Half	75.16	92.46
Skip 2	75.34	92.42
Skip 1	75.35	92.45
Full	75.39	92.56
<i>Swin-Top2 (Liu et al., 2021)</i>		
Back Half	76.16	93.02
Skip 2	76.10	92.93
Skip 1	76.29	92.98
Full	76.31	93.14

- *Alternating*: For Switch this means we place ACMoE on layers 2,4,6. For Swin this means we place ACMoE on layers 4,8,12,16.
- *Back Half*: For Switch this means we place ACMoE on just the last 3 layers of the network. For Swin this means we place ACMoE on just the last 5 layers of the network.
- *Skip 2*: For Swin this means we place ACMoE on layers 8,10,12,14,16,18.
- *Skip 1*: For Switch this means we place ACMoE on layers 3,4,5,6. For Swin this means we place ACMoE on layers 6,8,10,12,14,16,18.
- *Full*: We place ACMoE on every possible layer.

We present in Table 10 results for Switch and Swin ACMoE models when changing the positions of the ACMoE layers throughout the network. The results agree with our expectation that, generally speaking, more ACMoE layers improve performance, but in some circumstances a threshold is met at the point where ACMoE layers are used too early in the network such that the model has not been able to learn reasonably good approximations of the cluster membership of the tokens yet.

We find that in the Switch backbone, performance improves the more ACMoE layers we add, which agrees with our expectation that more ACMoE layers improve performance. However, we find that top performance is attained when allowing two standard MoE layers to go before the first ACMoE, as opposed to the minimum of 1 standard MoE layer. We conjecture this is because we need to give the model a few layers before the first ACMoE in order to learn decent representations such that we have good enough estimated cluster assignments for use in the ACMoE layer. Encouragingly, we find just one additional standard MoE layer is sufficient for the benefits of ACMoE to be obtained.

We find in Table 12 that with Swin, best performance is obtained using ACMoE on every possible layer, again agreeing with our expectation that more ACMoE layers improve performance. With Swin, however, we do not face any drop in performance from placing ACMoE too early in the network, and indeed we see *Full* attaining top performance. We conjecture that Swin does not encounter this issue since Swin uses four layers of feed forward networks before the first MoE layer, and so by the first MoE layer the representations are of reasonably good quality to produce good estimates of the cluster membership.

C.5.3 RANDOM ABLATION

We show the efficacy of the adaptive clustering transformation M (Definition 1) in our AC router at capturing meaningful feature-wise information by ablating it against an alternate $d \times d$ diagonal matrix made up of normal random variables with mean 1 and standard deviation 0.5 (where we clip any negative values to prevent negative weights). We present in Tables 13 and 14 results for language modeling (using Switch) and image classification (using Swin), which show fairly substantial drops in performance in both backbones. This offers evidence to the claim that our AC routing transformation is meaningfully weighting features to improve routing, and that performance gains

of our proposed method do not flow from a kind of implicit regularization of introducing noise into the router.

Table 13: Random Ablation in Switch (Fedus et al., 2022)

Model	Test PPL (\downarrow)
<i>Switch-Random</i> (Fedus et al., 2022)	38.17
Switch-ACMoE	34.42

Table 14: Random Ablation in Swin (Liu et al., 2021)

Model	Top 1 Acc.	Top 5 Acc.
<i>Swin-Random</i>	74.22	91.87
Swin-ACMoE	76.31	93.14

C.6 CLUSTER WEIGHT MIXING

The AC routing scheme estimates the cluster membership of each token based on its highest affinity cluster assigned in the previous layer. We could also further leverage the top-k structure of the MoE models by mixing the cluster-wise feature weights with weights corresponding to the affinities in the top-k routing. For example, if \mathbf{h} has affinity scores α and $1 - \alpha$ to clusters k and k' respectively, then we could also obtain the required AC routing transformation for \mathbf{h} as $\mathbf{M}_{k^*} = \alpha \mathbf{M}_k + (1 - \alpha) \mathbf{M}_{k'}$. This approach therefore factors in the confidence with which we believe \mathbf{h} belongs to cluster k or k' , and can be used for integrating ACMoE into higher expert granularity backbones (i.e higher top-k settings). Tables 15 and 16 show results for computing \mathbf{M}_{k^*} by mixing the top-affinity cluster weights (Mix 2) in Switch and GLaM with top-2 routing, versus our presented results which compute \mathbf{M}_{k^*} just based off of the highest affinity cluster (Mix 1). We see that GLaM-ACMoE benefits substantially from cluster weight mixing whereas Switch-ACMoE prefers just using its top affinity cluster weights. For consistency across models, we present in our main body the Mix 1 results, as GLaM-ACMoE already performs extremely strongly using Mix 1 and so we prefer to opt for the added performance gain in the Switch backbone.

Table 15: Results on Cluster Weight Mixing in Switch (Fedus et al., 2022)

Clusters Mixed	Test PPL (\downarrow)
Mix 2	34.66
Mix 1	34.42

Table 16: Results on Cluster Weight Mixing in GLaM (Du et al., 2022)

Clusters Mixed	Test PPL (\downarrow)
Mix 2	35.29
Mix 1	36.26

C.7 ADAPTIVE CLUSTERING INTEGRATION INTO SOFT MIXTURE OF EXPERTS

We present here results for integrating ACMoE into SoftMoE (Puigcerver et al., 2023). To use ACMoE in the SoftMoe setting, which can be understood as a top-E routing setting where all experts are active for every token, we compute \mathbf{M}_{k^*} using cluster weight mixing (Section C.6) over the top-8 highest affinity clusters. We present the performance of Soft-ACMoE on clean data, adversarially attacked data, and ImageNet-A/O/R in the following Tables 17 and 18.

Table 17: Test Accuracy on ImageNet corrupted PGD, FGSM, and SPSA using SoftMoE (Puigcerver et al., 2023) backbone

Model	Clean Data		PGD		FGSM		SPSA	
	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
<i>SoftMoE</i> (Puigcerver et al., 2023)	72.86	90.92	45.29	78.91	56.95	85.60	66.59	88.70
Soft-ACMoE (Ours)	73.21	91.23	48.25	80.49	59.01	86.69	70.63	93.22

We see in Tables 17 and 18 the efficacy of ACMoE in the SoftMoE backbone, offering evidence of the adaptability of our framework into further MoE setups. In particular, the SoftMoE framework models a setting in which expert clusters are highly overlapping, as each token is soft assigned to all experts. Therefore, the performance gains shown in clean and contaminated data of Soft-ACMoE demonstrates that our AC router is well-suited to modeling such a clustering structure.

Table 18: Test Accuracy on Image Classification in Imagenet-A/O/R using SoftMoE (Puigcerver et al., 2023) backbone

Model	Im-A	Im-R	Im-O
	Top-1 Acc. (↑)	Top-1 Acc. (↑)	AUPR (↑)
<i>SoftMoE</i> (Puigcerver et al., 2023)	6.69	31.63	17.97
Soft-ACMoE (Ours)	6.93	32.18	18.35

C.8 IMAGE CLASSIFICATION IN SWIN TRANSFORMER BASE CONFIGURATION

We further evaluate the performance ACMoE when scaling up model size in Table 19. We integrate ACMoE into the Base configuration of Swin (0.5B parameters) and evaluate on clean ImageNet-1K as well as under adversarial attacks.

Table 19: Test Accuracy on ImageNet corrupted PGD, FGSM, and SPSA using Swin Base (Liu et al., 2021) backbone

Model	Clean Data		PGD		FGSM		SPSA	
	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
<i>Swin-Base</i> (Liu et al., 2021)	79.06	94.37	44.61	79.20	59.91	87.72	68.94	89.00
Swin-ACMoE-Base (Ours)	79.25	94.42	46.28	80.24	61.78	87.55	70.18	89.33

C.9 ROUTER STABILITY

We present in Fig. 5 the routing stability of ACMoE, SMoE, XMoE, and StableMoE in the Switch backbone evaluated on WikiText-103. Routing instability computes over adjacent layers the proportion of tokens that are assigned to different experts across the two layers. Specifically, for n tokens $[h_1, \dots, h_n]$, we compute at layer ℓ the matrix $S^\ell \in \mathbb{R}^{n \times n}$ such that $S_{ij}^\ell = 1$ if the i^{th} and j^{th} tokens are assigned to the same expert in layer ℓ and is 0 otherwise. The router instability at layer ℓ can then be calculated as $r^\ell = \text{mean}(|S^{\ell-1} - S^\ell|)$. This metric therefore captures the degree to which tokens that are assigned to the same experts remain together through the model. A high r^ℓ indicates the router doesn't maintain consistent expert assignments, as tokens that it considers semantically similar at one layer it considers different at the next.

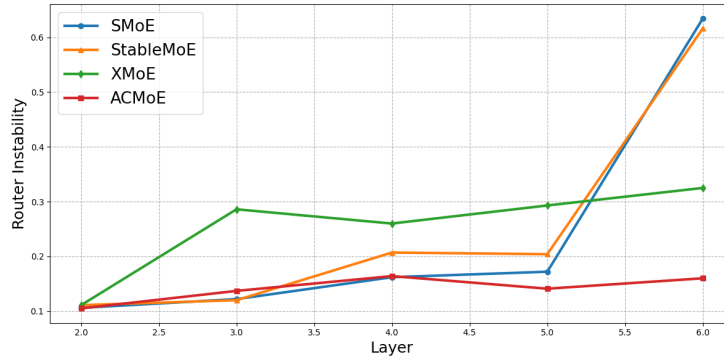


Figure 5: Router Instability of ACMoE, SMoE, XMoE, and StableMoE. ACMoE maintains consistent routing, while baseline routers more frequently change the expert assignments of tokens.

In Fig. 5, we see that baseline routers reach high levels of instability, where in the case of SMoE and StableMoE, at the last layer over 60% of tokens are assigned to a different expert. ACMoE, by contrast, maintains a more consistent, stable assignment through the model, with no more than 20% of tokens changing expert assignment across any layer.

C.10 DYNAMIC ROUTING

We further test the compatibility of our Adaptive Clustering routing scheme in dynamic top-p routing. In this setting, rather than routing each token to its top-k highest affinity experts in each MoE

layer, we route each token to all experts that have affinity over a certain threshold p . This setting permits activating more or less experts for different tokens at different layers throughout the model, therefore dynamically assigning experts to tokens. We integrate our AC routing directly into this setting using the same setup as in Section 3, where the AC routing transformation is computed based on the estimated cluster membership of each token using the top affinity assignment of the previous layer. We present the results for Switch transformer on WikiText-103 language modeling in the following Table 20.

Table 20: Results on Top- p Dynamic Routing in Switch Backbone (Fedus et al., 2022)

Model	Test PPL (\downarrow)
<i>Fixed top-k routing (Shazeer et al., 2017)</i>	
<i>Switch-medium (Fedus et al., 2022)</i>	35.48
<i>ACMoE-medium (Ours)</i>	34.42
<i>Dynamic top-p routing (Guo et al., 2024)</i>	
<i>Switch-Fixed p</i>	35.20
<i>Switch-ACMoE-Fixed p (Ours)</i>	34.14
<i>Switch-Learnable p</i>	34.29
<i>Switch-ACMoE-Learnable p (Ours)</i>	33.49

For fixed p , we set $p = 0.05$. For learnable p , we initialize the parameter to 0.05. We select this initialization as it reproduces approximately similar performance in the Switch backbone under default top-2 routing, thereby aiding direct comparison between fixed top-k and dynamic top- p routing. We see in the dynamic routing setting, ACMoE maintains the same consistent improvement over the Switch baseline of roughly 1 full PPL. These results suggest ACMoE is well-suited to the dynamic routing setting.

D BROADER IMPACT

Our research offers benefits to Mixture-of-Expert (MoE) architectures in both clean and contaminated settings. In particular, our work offers socially beneficial outcomes with regard to defense against adversarial attack, which we hope can be used to protect important AI systems from malicious actors. Furthermore, as large language models, many of which are built on MoE backbones, continue to profligate and be used in important societal settings, we hope our improved robustness to data contamination can aid this promising technology to continue to grow and improve in realistic settings of noisy training and evaluation data. Our research also shows substantially faster convergence than comparative baselines. We believe this faster convergence can deliver significant social benefit in terms of reducing the energy requirements of large model training, thereby helping to ease the growing environmental burden of AI training runs. We recognize there will always be risk of misuse with AI systems, however we hope that our work can be used to enhance and protect socially beneficial AI while also decreasing the environmental impact of this technology. We furthermore hope that our research can spur others on to continue building on robust and efficient AI for social good.