Supplement to "MomentumSMoE: Integrating Momentum into Sparse Mixture of Experts"

Table of Contents

A **Proof of Lemma 1**   17

B **Interpretation of Robust Momentum**   19

C **Empirical Evidence**   19

D **Experiment Details**   20

   D.1 WikiText-103 Language Modeling   21

   D.2 ImageNet-1K Object Recognition   22

   D.3 Pseudocode   23

E **Additional Experimental Results**   24

   E.1 Hyperparameters   24

   E.2 WikiText-103 Language Modeling   25

   E.3 ImageNet-C Full Results   25

   E.4 Further Extensions Beyond Adam and Robust Momentum   25

   E.5 Comparison of Computational Efficiency   29

F **Broader Impacts**   29

A Proof of Lemma 1

We can find the eigenvalues of matrix A explicitly as

$$\lambda_{\{\begin{smallmatrix}1\\2\end{smallmatrix}\}} = \frac{(1 + \mu - \gamma\sigma(n)) \pm \sqrt{(1 + \mu - \gamma\sigma(n))^2 - 4\mu}}{2}$$

If $\gamma\sigma(n) \leq 0$,

$$
\begin{aligned}
\lambda_1 &= \frac{(1 + \mu - \gamma\sigma(n)) + \sqrt{(1 + \mu - \gamma\sigma(n))^2 - 4\mu}}{2} \\
&\geq \frac{(1 + \mu) + \sqrt{(1 + \mu)^2 - 4\mu}}{2} \\
&= \frac{(1 + \mu) + |1 - \mu|}{2} \\
&= \max\{1, \mu\} \\
&\geq 1
\end{aligned}
$$

Then, we must have $\mu\sigma(n) > 0$. Letting the expression in the square root be Δ, expanding it,

$$
\begin{aligned}
\Delta &:= (1 + \mu - \gamma\sigma(n))^2 - 4\mu \\
&= (1 - \gamma\sigma(n))^2 + 2(1 - \gamma\sigma(n))\mu + \mu^2 - 4\mu \\
&= \mu^2 - 2(1 - \gamma\sigma(n))\mu + (1 - \gamma\sigma(n))^2
\end{aligned}
$$

and finding the roots of the equation yields, $\mu = (1 \pm \sqrt{\gamma\sigma(n)})^2$. Then, we consider two cases.

17

Case 1: $\Delta \geq 0$ when $\mu \geq (1 + \sqrt{\gamma\sigma(n)})^2$ or $\mu \leq (1 - \sqrt{\gamma\sigma(n)})^2$

1a: If $\mu \geq (1+\sqrt{\gamma\sigma(n)})^2$, then $1+\mu-\gamma\sigma(n) \geq 1+(1+\sqrt{\gamma\sigma(n)})^2-\gamma\sigma(n) \geq 2+2\sqrt{\gamma\sigma(n)} \geq 0$. Then, $\lambda_{1,2} > 0$ and $\lambda_1 \geq \lambda_2$. For the system to converge, we need

$$\lambda_1 = \frac{(1+\mu-\gamma\sigma(n)) + \sqrt{\Delta}}{2} < 1 \iff \sqrt{\Delta} < 1 - \mu + \gamma\sigma(n) \tag{19}$$

However, by assumption, we have a contradiction

$$0 \leq \sqrt{\Delta} < 1 - \mu + \gamma\sigma(n) \leq 1 - (1+\sqrt{\gamma\sigma(n)})^2 + \gamma\sigma(n) = -2\sqrt{\gamma\sigma(n)} < 0.$$

Hence, we cannot have $\mu \geq (1 + \sqrt{\gamma\sigma(n)})^2$.

1b: If $\mu \leq (1 - \sqrt{\gamma\sigma(n)})^2$, we further divide into two more subcases,

    1bi: If $1 + \mu - \gamma\sigma(n) \geq 0$, again we have, $\lambda_{1,2} > 0$ and $\lambda_1 \geq \lambda_2$. By Eqn. 19, we require $0 \leq \sqrt{\Delta} < 1-\mu+\gamma\sigma(n)$. As $1-\mu+\gamma\sigma(n) \geq 1-(1-\sqrt{\gamma\sigma(n)})^2+\gamma\sigma(n) = 2\sqrt{\gamma\sigma(n)} > 0$, it is enough to check that $\Delta < (1 - \mu + \gamma\sigma(n))^2$ can be satisfied.

$$
\begin{aligned}
\Delta < (1 - \mu + \gamma\sigma(n))^2 &\iff (1+\mu-\gamma\sigma(n))^2 - 4\mu < (1-\mu+\gamma\sigma(n))^2 \\
&\iff (1+\mu-\gamma\sigma(n))^2 - (1-\mu+\gamma\sigma(n))^2 < 4\mu \\
&\iff (1+\mu-\gamma\sigma(n)+1-\mu+\gamma\sigma(n)) \\
&\qquad\qquad (1+\mu-\gamma\sigma(n)-1+\mu-\gamma\sigma(n)) < 4\mu \\
&\iff 4(\mu-\gamma\sigma(n)) < 4\mu \\
&\iff 0 < \gamma\sigma(n)
\end{aligned}
$$

Therefore, the conditions for convergence are $i)$ $\mu \leq (1 - \sqrt{\gamma\sigma(n)})^2$, $ii)$ $1+\mu-\gamma\sigma(n) \geq 0$ and $iii)$ $\gamma\sigma(n) > 0$. Then by $ii)$, $\mu \geq \gamma\sigma(n) - 1 \geq -1$. Suppose for a contradiction that $\mu \geq 1$, then by $i)$, $1 \leq \mu \leq (1 - \sqrt{\gamma\sigma(n)})^2$ which implies that $\gamma\sigma(n) \geq 2\sqrt{\gamma\sigma(n)}$ and hence, $\gamma\sigma(n) \geq 4$. Combining this with condition $i)$, we have $\sqrt{\mu} \leq \sqrt{\gamma\sigma(n)} - 1$ which leads to the following contradiction with $ii)$

$$
\begin{aligned}
\mu \leq (1 - \sqrt{\gamma\sigma(n)})^2 &\implies \mu \leq 1 - 2\sqrt{\gamma\sigma(n)} + \gamma\sigma(n) \\
&\implies \mu + 2(\sqrt{\gamma\sigma(n)} - 1) \leq \gamma\sigma(n) - 1 \\
&\implies \mu + 2\sqrt{\mu} \leq \gamma\sigma(n) - 1 \\
&\implies 1 + \mu - \gamma\sigma(n) + 2\sqrt{\mu} \leq 0.
\end{aligned}
$$

Then, we must have a last condition for convergence, $iv)$ $|\mu| < 1$.

    1bii: If $1 + \mu - \gamma\sigma(n) \leq 0$, then $|\lambda_2| \geq |\lambda_1|$ and

$$\lambda_2 = \frac{(1+\mu-\gamma\sigma(n)) - \sqrt{\Delta}}{2} < 0$$

For the system to be stable, we need $\lambda_2 > -1 \iff 3 + \mu - \gamma\sigma(n) > \sqrt{\Delta} \geq 0$. Hence, we require $i)$ $\mu \leq (1 - \sqrt{\gamma\sigma(n)})^2$, $ii)$ $1+\mu-\gamma\sigma(n) \leq 0$, $iii)$ $3+\mu-\gamma\sigma(n) > 0$ and $iv)$

$$
\begin{aligned}
(3+\mu-\gamma\sigma(n))^2 > \Delta &\iff (3+\mu-\gamma\sigma(n))^2 > (1+\mu-\gamma\sigma(n))^2 - 4\mu \\
&\iff 4\mu > (1+\mu-\gamma\sigma(n))^2 - (3+\mu-\gamma\sigma(n))^2 \\
&\iff 4\mu > (1+\mu-\gamma\sigma(n)+3 \qquad\qquad +\mu-\gamma\sigma(n)) \\
&\qquad\qquad (1+\mu-\gamma\sigma(n)-3-\mu+\gamma\sigma(n)) \\
&\iff 4\mu > -4(2+\mu-\gamma\sigma(n)) \\
&\iff 2+2\mu-\gamma\sigma(n) > 0
\end{aligned}
$$

Combining $ii)$ and $iv)$, we have, $2+2\mu-\mu-1 > 0 \implies \mu > -1$. Similarly, we assume for a contradiction that $\mu \geq 1$. Then, $\gamma\sigma(n) \geq 1+\mu \geq 2$ and by condition $i)$, $\sqrt{\mu} \leq \sqrt{\gamma\sigma(n)} - 1$

18

from which follows the contradiction

$$1 \leq \sqrt{\gamma\sigma(n)} - \sqrt{\mu} \implies \sqrt{\gamma\sigma(n)} + \sqrt{\mu} \leq (\sqrt{\gamma\sigma(n)} - \sqrt{\mu})(\sqrt{\gamma\sigma(n)} + \sqrt{\mu}) = \gamma\sigma(n) - \mu$$
$$\implies \sqrt{\gamma\sigma(n)} + \sqrt{\mu} \leq \gamma\sigma(n) - \mu < 3 \quad \text{(by } iii \text{))}$$
$$\implies \sqrt{\mu} + \sqrt{\gamma\sigma(n)} - 1 < 2$$
$$\implies 2\sqrt{\mu} < 2 \quad \text{(by } i \text{) again)}.$$

Therefore, we need $v)$ $|\mu| < 1$ for convergence as well.

To summarize, for **Case 1**, in order to have a stable system, we require:

a) $\mu \leq (1 - \sqrt{\gamma\sigma(n)})^2$

b) If $1 + \mu - \gamma\sigma(n) \geq 0$,

     $i)$ $\gamma\sigma(n) > 0$
     $ii)$ $|\mu| < 1$

c) If $1 + \mu - \gamma\sigma(n) \leq 0$,

$$\begin{array}{ll} i) & 2 + 2\mu - \gamma\sigma(n) > 0 \\ ii) & |\mu| < 1 \end{array} \implies \left\{ \begin{array}{ll} i) & 3 + \mu - \gamma\sigma(n) > 0 \\ ii) & 2 + 2\mu - \gamma\sigma(n) > 0 \\ iii) & |\mu| < 1 \end{array} \right.$$

Case 2: $\Delta < 0$ when $(1 + \sqrt{\gamma\sigma(n)})^2 > \mu > (1 - \sqrt{\gamma\sigma(n)})^2$

Then, the eigenvalues are complex and given by

$$\lambda_{\{^1_2\}} = \frac{(1 + \mu - \gamma\sigma(n)) \pm i\sqrt{4\mu - (1 + \mu - \gamma\sigma(n))^2}}{2}.$$

For convergence, we require $|\lambda_{1,2}| = \sqrt{\mu} < 1$ or equivalently, $\mu < 1$. Hence, the necessary condition becomes $1 > \mu > (1 - \sqrt{\gamma\sigma(n)})^2$ and to avoid a contradiction, we also require $1 > (1 - \sqrt{\gamma\sigma(n)})^2$. This is satisfied when $\gamma\sigma(n) < 2 + 2\mu$ and $|\mu| < 1$.

Therefore, from all the considered cases, to have a stable system, we require $0 < \gamma\sigma(n) < 2 + 2\mu$ and $|\mu| < 1$, concluding the proof.

B   Interpretation of Robust Momentum

To provide intuition behind robust momentum, we follow [10] and view the update rule in Eqn. 16 as a Lur'e feedback control system. For simplicity, we consider the case where, $x_t \in \mathbb{R}$ and write the update in matrix form

$$\begin{pmatrix} x_t \\ x_{t+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -\mu & 1+\mu \end{pmatrix} \begin{pmatrix} x_{t-1} \\ x_t \end{pmatrix} + \begin{pmatrix} 0 \\ -\gamma \end{pmatrix} f(y_t); \quad y_t = \begin{pmatrix} -\alpha & 1+\alpha \end{pmatrix} \begin{pmatrix} x_{t-1} \\ x_t \end{pmatrix} \quad (20)$$

In the frequency domain, in order for the system 20 to be stable, we require that for all $|z| = 1$,

$$\text{Re}((1 - pz^{-1})((k-1)\tilde{G}(pz) - 1)) < 0,$$

where $\tilde{G}(pz)$ is a transformed transfer function, and p, a scaling factor. The imaginary axis, where $\text{Re} = 0$, is the stability boundary, and the specific construction of the parameters γ, μ and α pushes this boundary into the negative real axis to $-v = (1 + p)(1 - k + 2kp - kp^2)/2p$, hence achieving robust stability through the design of γ, μ and α.

C   Empirical Evidence

In this section, we provide the full plots presented in Section 2.3 and 6. We visualize the average norm of the SMoE and MoE outputs respectively, for 80 epochs of training in all 6 layers in Fig. 6 and 7. Notably, in all plots, less a minor exception in layer 3 of SMoE, there is a decrease in the norm of each SMoE and MoE layer output throughout training. This is consistent with our expectation that the gate learns the optimal α^* in a multi-objective optimization problem (Eqn. 8 and 9).

19
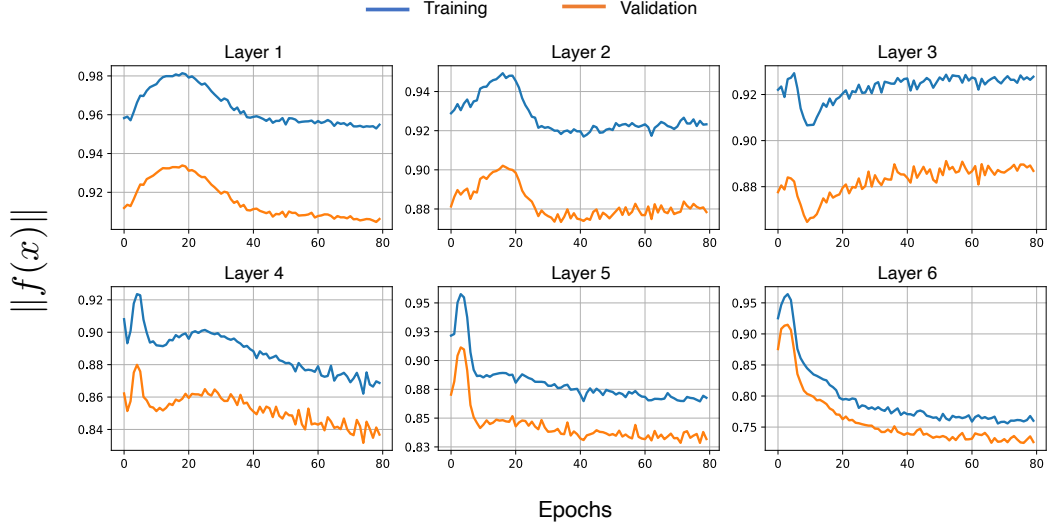
Figure 6: Average norm of the SMoE outputs at all layers during 80 epochs of training for the baseline SMoE.
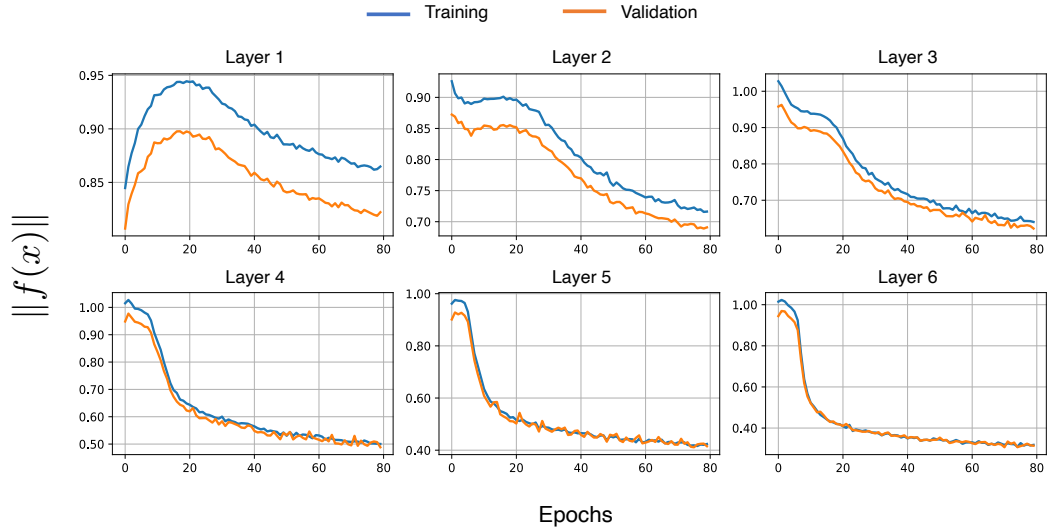
Figure 7: Average norm of the MoE outputs at all layers during 80 epochs of training for the baseline MoE.

In Fig. 8, 9 and 10, we present the proportion of time each expert is chosen in decreasing magnitude of the norm of the expert outputs for SMoE, MomentumSMoE and AdamSMoE. As discussed in Section 6, in accordance with our connection between GD and SMoE, a more even distribution of expert selection based on the size of the norm of the expert outputs should yield improved performance of the model. We demonstrate in Section 5.1 that both MomentumSMoE and AdamSMoE significantly exceed the baseline performance and correspondingly, flattens the normed-based load distribution among experts as observed in Fig. 8, 9 and 10. These strong empirical evidences serve to reinforce our optimization framework for SMoE.

D   Experiment Details

For clarity, we summarize the new models developed in this paper for each task in Table 4 and address the lacking implementations here. First, as the introduction of AdamSMoE into V-MoE leads to unstable training, we defer this challenge to future work. Second, our primary goal when studying the Momentum-Soft MoE model is to showcase the universality of our MomentumSMoE method

20
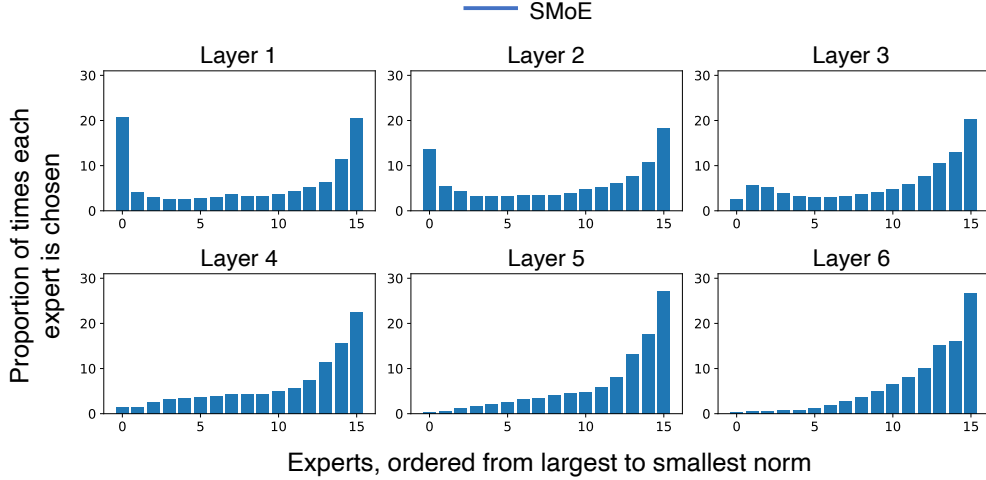
Figure 8: Proportion of each expert chosen, ordered from the largest norm of each expert output to the smallest norm, in all layers of baseline SMoE.
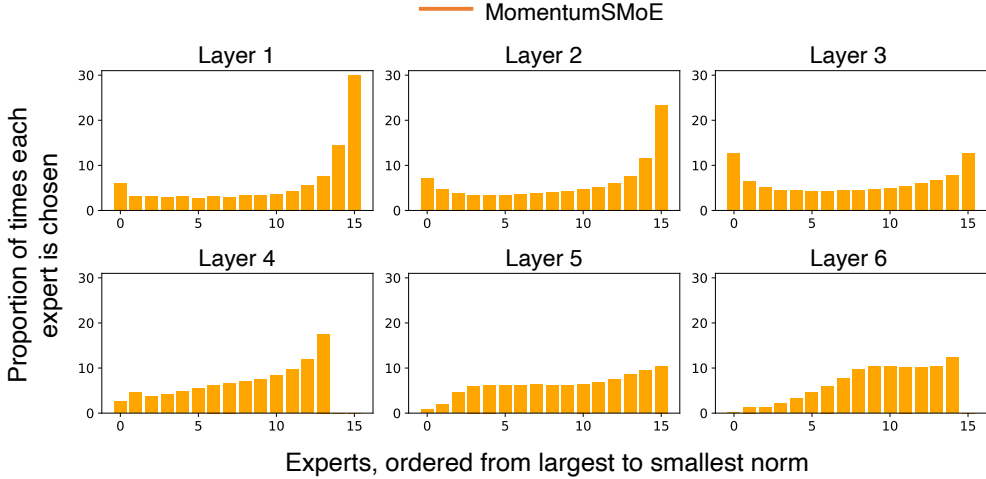
Figure 9: Proportion of each expert chosen, ordered from the largest norm of each expert output to the smallest norm, in all layers of MomentumSMoE.

Table 4: A summary of the new models developed for WikiText-103 language modeling and ImageNet-1K object recognition tasks.

Task	Model/Method	MomentumSMoE	AdamSMoE	Robust MomentumSMoE
WikiText-103	SMoE GLaM	✓	✓	✓
ImageNet-1K ImageNet-R ImageNet-A ImageNet-C	V-MoE	✓	✗	✓
ImageNet-1K	Soft MoE	✓	✗	✗

across both SMoE and MoE, hence we did not integrate AdamW and Robust Momentum into Soft MoE. We leave these implementations for future work.

D.1 WikiText-103 Language Modeling

Dataset: The WikiText-103 dataset [43] is derived from Wikipedia articles and is designed to capture long-range contextual dependencies. The training set contains about 28,000 articles, with a

21
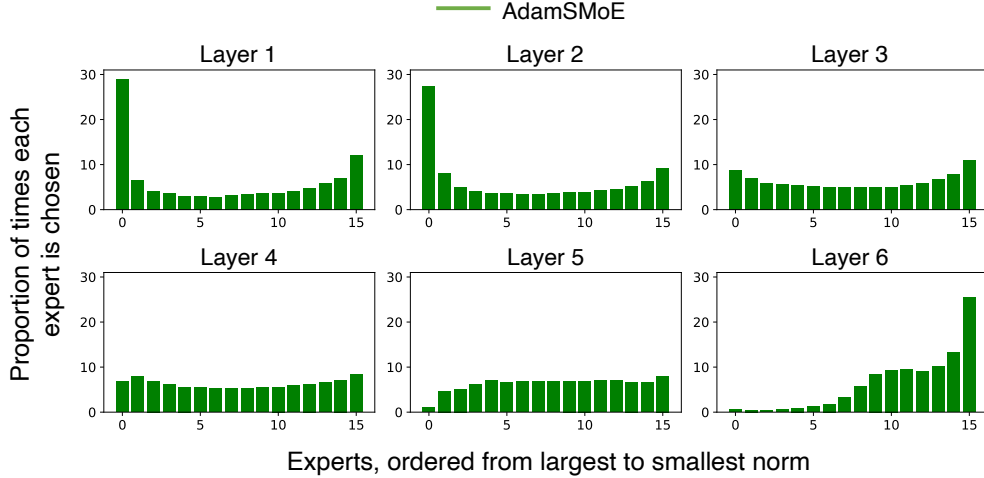
Figure 10: Proportion of each expert chosen, ordered from the largest norm of each expert output to the smallest norm, in all layers of AdamSMoE.

total of 103 million words. Each article is divided into text blocks with approximately 3,600 words. The validation and test sets have 218,000 and 246,000 words, respectively, with both sets comprising 60 articles and totaling about 268,000 words. On the attacked dataset, we corrupt the both validation and test data to demonstrate the robustness of MomentumSMoE and AdamSMoE using TextAttack's word swap attack [46]. This adversarial attack randomly replaces words in the dataset with a generic "AAA" for evaluation making it difficult for the model to predict the next word in the sequence correctly.

Model and baselines: We use the Switch Transformer [18], referred to as SMoE in our tables and figures, and GLaM [16] baselines, which replaces each multilayer perceptron (MLP) layer and every other MLP layer in a vanilla language modeling transformer with a SMoE layer, respectively. For MomentumSMoE, we replace each MLP layer with a MomentumSMoE layer and initialise each momentum vector, p_0 at 0. We do the same for MomentumGLaM with every other MLP layer.

For consistency, we define the number of layers in each model as the number of SMoE layers. The default model used in each experiment is medium sized with 6 layers, but we include a comparison between a smaller model with 3 layers as well as a larger one with 12 layers in Appendix E.2. Each model has 16 experts in every SMoE, MomentumSMoE and AdamSMoE layer and selects 2 experts ($K = 2$) per input. All models use the same sparse router function consisting of a linear network receiving the input data followed by the TopK, then the Softmax function. The small models train for 60 epochs, the medium and large SMoE models train for 80 epochs and the GLaM models train for 120 epochs without any additional load balancing loss. Our implementation is based on the code base developed by [54], publicly available at https://github.com/ofirpress/sandwich_transformer and https://github.com/giangdip2410/CompeteSMoE/tree/main.

AdamSMoE/AdamGLaM: It is observed that Adam has certain divergent behavior during large-scale training leading to unstable loss curves [8, 44]. In line with this observation, during the initial implementations of AdamSMoE, we experience similar instability. A widely used solution to this, proposed by [32], is to switch from Adam to gradient descent at a suitable point during training. We follow suit and use AdamSMoE in the first layer of the model and MomentumSMoE for the subsequent layers. We find that implementing AdamSMoE in the first layer is enough to significantly improve the model's overall performance and accelerate its convergence.

D.2 ImageNet-1K Object Recognition

Datasets: We use the ImageNet-1K dataset that contains 1.28M training images and 50K validation images. There are 1000 classes of images and the model learns an object recognition task. For robustness to common corruptions, we use ImageNet-C (IN-C) [25] which consists of 15 different types of corruptions applied to the ImageNet-1K validation set with 5 levels of severity. We provide a breakdown of our results on each corruption type and the mean Corruption Error (mCE) across

22

```
Hyperparameters: mu

def MomentumSMoE(x, momentum):
    momentum = - SMoE(x) + mu * momentum
    x = x + gamma * momentum
    return x
```

Figure 11: Pseudocode for MomentumSMoE implementation in python with 1 hyperparameter μ.

```
Hyperparameters: mu, beta, eps = 1e-8

def AdamSMoE(x, gradient, squared_gradient):
    gradient = - (1 - mu) * SMoE(x) + mu * momentum
    squared_gradient = beta * squared_gradient + (1 - beta) * SMoE(x) ** 2
    x = x + gamma / (torch.sqrt(squared_gradient) + eps) * gradient - k * x
    return x
```

Figure 12: Pseudocode for AdamSMoE implementation in python with 2 hyperparameters μ and β.

escalating levels of severity for two corruption types in Appendix E.3. To test robustness to input data distribution shifts, we use ImageNet-A (IN-A) [26]. IN-A contains a 200 class subset of ImageNet-1K classes with adversarially filtered images. Finally, we test our model on ImageNet-R (IN-R) [24] which contains various artistic renditions of images. This evaluates the model's generalization ability to abstract visual renditions.

Metrics: On ImageNet-1K, ImageNet-A and ImageNet-R, we report the top-1 accuracies for all experiments. On ImageNet-C, the standard metric for evaluation is the mCE. To calculate this, we average the top-1 error rate for each corruption type across the 5 levels of severity and divide them by AlexNet's average errors, then take the final average across all corruption types. The direction of increasing or decreasing values of these metrics signifying greater robustness will be indicated in the table with an arrow.

Model and baselines: We use a small Vision Mixture of Experts (V-MoE) [62] model as the SMoE baseline for ImageNet-1K object recognition task as well as the standard robustness benchmarks. This variant of V-MoE consists of 8 Vision Transformer (ViT) blocks [15] with every odd block's MLP being replaced by a SMoE layer. In turn, in MomentumV-MoE, we replace every other MLP layer with a MomentumSMoE layer and similarly for Robust MomentumV-MoE. For all vision SMoE models, we select 2 experts ($K = 2$) at every SMoE layer for each patch. We follow the training configurations and setting as in [62] and their code base is available here https://github.com/google-research/vmoe/.

For our MoE baseline on clean ImageNet-1K data, we use Soft Mixture of Experts (Soft MoE) [56]. A Soft MoE model is designed to side step the challenging discrete optimization problem of assigning each token to an expert, as in SMoE, through a soft token assignment. Instead of each token being routed to one expert, in a Soft MoE layer, each expert is assigned a certain number of slots and each slot is allocated a weighted average of all tokens. These weights are dependant on both the tokens and the experts. In this case, Soft MoE is not considered as a SMoE, but instead a MoE. We use the smallest model, Soft MoE-tiny with 12 layers. The first 6 layers consists of standard ViT blocks and the last 6 layers replace the MLP in those blocks with a Soft MoE layer. In Momentum-Soft MoE, we implement the momentum parameter into each Soft-MoE layer as in a SMoE layer.

As there were no training details provided for Soft MoE on ImageNet-1K, we follow the training procedure in [69] for 120 epochs, using their published code base https://github.com/facebookresearch/deit. We train Momentum-Soft MoE and the baseline model using the PyTorch implementation of Soft MoE at https://github.com/bwconrad/soft-moe.

D.3 Pseudocode

In this section, we provide the pseudocode as written in python for MomentumSMoE, AdamSMoE, and Robust MomentumSMoE for clarification on our implementation. These are found in Figures 11, 12 and 13 respectively.

23

```
Hyperparameters: p, L, m

def RobustMomentumSMoE(x, momentum):
    k = L / m
    gamma = k * ((1 - p) ** 2) * (1 + p) / L
    mu = k * p ** 3 / (k - 1)
    alpha = p ** 3 / ((k - 1) * ((1 - p) ** 2) * (1 + p))
    y = x + alpha * gamma * momentum
    momentum = - SMoE(y) + mu * momentum
    x = x + gamma * momentum
    return x
```

Figure 13: Pseudocode for Robust MomentumSMoE implementation in python with 3 hyperparameters p, L and m.

Table 5: Perplexity (PPL) results on clean and attacked WikiText-103 validation and test data for standard MomentumSMoE with tuned hyperparameters μ and γ and MomentumSMoE trained with different learning settings for μ and γ that do not require tuning: *i)* Both μ and γ are scalar learnable parameters in Pytorch. *ii)* Only γ is learned with fixed $\mu = 0.7$.

Model	μ	γ	WikiText-103	Valid PPL ↓	Test PPL ↓
SMoE (baseline)	-	-	Clean	33.76	35.55
			Attacked	42.24	44.19
MomentumSMoE	0.7	1.0	Clean	32.29	**33.46**
			Attacked	40.94	42.33
MomentumSMoE *(i)*	Learnable	Learnable	Clean	**32.28**	33.87
			Attacked	40.41	42.32
MomentumSMoE *(ii)*	0.7	Learnable	Clean	**32.28**	33.69
			Attacked	**39.96**	**41.84**

E   Additional Experimental Results

E.1   Hyperparameters

In this section, we discuss two additional methods to avoid tuning MomentumSMoE hyperparameters for ease of implementation and efficiency.

Time-varying momentum: Another form of the classical momentum proposed by [48] replaces the constant momentum parameter with a time-varying one $t - 1/t + 2$, which removes the need to choose an appropriate momentum hyperparameter. We adopt this modification into MomentumSMoE to replace μ while keeping γ fixed at 1.0, and the MomentumSMoE time-varying (TV) layer is as follows

$$\boldsymbol{p}_t = -f(\boldsymbol{x}_t) + \frac{t-1}{t+2}\boldsymbol{p}_{t-1}; \quad \boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \gamma\boldsymbol{p}_t$$

Zero-order hold μ and γ: Recall Eqn. 10, the proposed accelerated MomentumSMoE layer

$$\boldsymbol{p}_t = -f(\boldsymbol{x}_t) + \mu\boldsymbol{p}_{t-1}; \quad \boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \gamma\boldsymbol{p}_t.$$

When we replace MLP layers with MomentumSMoE layers, as we do in our language model MomentumSMoE, each expert function is a linear network. Explicitly expressing the MomentumSMoE layer results in the following expression

$$\boldsymbol{p}_t = (-\sum_{i=1}^{E} g(\boldsymbol{x}_t)_i \boldsymbol{W}_i^\top)\boldsymbol{x}_t + \mu\boldsymbol{p}_{t-1} = \bar{\boldsymbol{B}}\boldsymbol{x}_t + \mu\boldsymbol{p}_{t-1}; \quad \boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \gamma\boldsymbol{p}_t$$

and can be interpreted as a state space representation with constant state and output matrices, μ and γ. From this perspective, we experiment with learning a discretized μ_t and γ_t by applying the Zero-order hold (ZOH) such that they become adaptive parameters as is a common practise in optimization algorithms. μ and γ are then parameterized as follows

$$\mu_t = e^{\Delta\mu}; \quad \gamma_t = (\Delta\mu)^{-1}(e^{\Delta\mu} - 1)\Delta\gamma$$

24

Table 6: Perplexity (PPL) results on clean and attacked WikiText-103 validation and test data for standard MomentumSMoE with tuned hyperparameters μ and γ and MomentumSMoE trained with different learning settings for μ and γ that do not require tuning: *i)* Both μ and γ are scalar learnable parameters in Pytorch. *ii)* Only γ is learned with fixed $\mu = 0.7$. *iii)* Only γ is learned as a linear network then composed with a sigmoid function with fixed μ. Included are the time-varying MomentumSMoE (TV) and Zero-order hold MomentumSMoE (ZOH).

Model	μ	γ	WikiText-103	Valid PPL $\downarrow$	Test PPL $\downarrow$
SMoE (baseline)	-	-	Clean	33.76	35.55
			Attacked	42.24	44.19
MomentumSMoE	0.7	1.0	Clean	32.29	**33.46**
			Attacked	40.94	42.33
MomentumSMoE *(i)*	Learnable	Learnable	Clean	32.28	33.87
			Attacked	40.41	42.32
MomentumSMoE *(ii)*	0.7	Learnable	Clean	32.28	33.69
			Attacked	**39.96**	**41.84**
MomentumSMoE *(iii)*	0.7	Linear network	Clean	32.30	33.96
			Attacked	40.35	42.39
MomentumSMoE (TV)	$t - 1/t + 2$	1.0	Clean	33.02	35.08
			Attacked	41.44	43.97
MomentumSMoE (ZOH)	$e^{\Delta\mu}$	$(e^{\Delta\mu} - 1)\Delta\gamma/\Delta\mu$	Clean	**32.21**	34.00
			Attacked	40.74	42.70

where Δ is the Softplus function of a learned scalar parameter, μ a learned scalar parameter and γ a linear network with scalar outputs.

Results: We report the PPL for MomentumSMoE (TV) and MomentumSMoE (ZOH) on clean and word swap attacked WikiText-103 validation and test data in Table 6, an extended version of Table 5. We also include an additional setting of learning γ as the sigmoid of a linear network. In this setting, every input has a different learning rate. While these models do improve over the baseline, there is no advantage over the standard MomentumSMoE and the other learnable μ and γ settings. Hence, we do not recommend implementations in this section over those and keep the results here for reference.

E.2 WikiText-103 Language Modeling

We present the results of all three sizes of SMoE and MomentumSMoE models in Table 7 and observe that with increasing model depth, the effectiveness of the momentum parameter in improving model performance increases as well. This aligns with the analogy of each layer being a GD step and with a higher number of iterations, the momentum term becomes more effective. While beneficial for large models, as implementing MomentumSMoE is computationally cost efficient, and hence, minimally affected by model depth, we note that there is an adverse effect in small models such as SMoE-small with only 3 layers. We hypothesize that the primary reason for this are the insufficient layers for the momentum term to make a positive impact and is a limitation of our method.

E.3 ImageNet-C Full Results

We provide the full results of the top-1 accuracy and mCE of all 15 corruption types in ImageNet-C for V-MoE, MomentumV-MoE, Robust MomentumV-MoE and Sharpness Aware MomentumV-MoE (SAM-V-MoE), developed in Appendix E.4, in Table 8. Included in Figure 14 is a plot of the mCE with escalating severity of impulse noise and gaussian noise corruption, which illustrates the advantages of our methods with increasing data corruption. We observe that across all 15 corruption types, except for motion blur, Robust MomentumV-MoE outperforms the baseline V-MoE, with as high as a 6.5% increase in top-1 accuracy and 8 mCE decrease on fog corruption.

E.4 Further Extensions Beyond Adam and Robust Momentum

The advantage of an optimization perspective for SMoEs are the countably many descent algorithms available that can be used to improve the model. In this section, we elaborate on six more extensions

25

Table 7: Perplexity (PPL) of baseline SMoE-small/medium/large, MomentumSMoE-small/medium/large, AdamSMoE-medium, baseline GLaM-medium, MomentumGLaM-medium and AdamGLaM-medium on clean and attacked WikiText-103 validation and test data.

Model/Metric	Clean WikiText-103		Attacked WikiText-103	
	Valid PPL ↓	Test PPL ↓	Valid PPL ↓	Test PPL ↓
SMoE-small (baseline)	**84.26**	**84.81**	**98.60**	**99.29**
MomentumSMoE-small	85.71	86.65	100.26	101.18
SMoE-medium (baseline)	33.76	35.55	42.24	44.19
MomentumSMoE-medium	32.29	33.46	40.94	42.33
AdamSMoE-medium	**31.59**	**33.25**	**39.27**	**41.11**
GLaM-medium (baseline)	36.37	37.71	45.83	47.61
MomentumGLaM-medium	33.87	35.29	42.15	43.64
AdamGLaM-medium	**32.99**	**34.32**	**41.09**	**42.81**
SMoE-large (baseline)	29.31	30.33	36.77	37.83
MomentumSMoE-large	**27.58**	**29.03**	**35.21**	**36.78**

Table 8: Top-1 accuracy (%) and mean corruption error (mCE) of V-MoE, MomentumV-MoE, Robust MomentumV-MoE and SAM-V-MoE on each corruption type in ImageNet-C.

Corruption Type	Model/Metric	*V-MoE (Baseline)*	MomentumV-MoE	Robust MomentumV-MoE	SAM-V-MoE
Brightness	Top-1 ↑	67.25	67.77	**67.79**	67.34
	mCE ↓	58.01	57.08	**57.06**	57.85
Contrast	Top-1 ↑	36.94	40.94	**42.71**	41.64
	mCE ↓	73.91	69.22	**67.15**	68.41
Defocus Blur	Top-1 ↑	39.89	41.01	**41.02**	40.26
	mCE ↓	73.31	71.95	**71.94**	72.86
Elastic Transform	Top-1 ↑	53.14	53.08	**53.19**	52.89
	mCE ↓	72.53	72.63	**72.45**	72.92
Fog	Top-1 ↑	38.23	42.00	**44.85**	44.03
	mCE ↓	75.39	70.79	**67.31**	68.31
Frost	Top-1 ↑	50.12	51.51	**51.83**	51.45
	mCE ↓	60.35	58.66	**58.27**	58.73
Gaussian Noise	Top-1 ↑	49.38	50.92	**52.08**	50.92
	mCE ↓	57.11	55.37	**54.06**	55.36
Glass Blur	Top-1 ↑	36.65	36.74	**37.30**	36.43
	mCE ↓	76.67	76.56	**75.88**	76.93
Impulse Noise	Top-1 ↑	47.72	49.30	**50.59**	49.15
	mCE ↓	56.66	54.95	**53.56**	55.11
JPEG Compression	Top-1 ↑	60.21	60.44	**60.53**	60.23
	mCE ↓	65.61	65.23	**65.08**	65.58
Motion Blur	Top-1 ↑	**43.59**	43.10	43.35	42.08
	mCE ↓	**71.77**	72.40	72.08	73.69
Pixelate	Top-1 ↑	61.66	62.61	**63.14**	62.30
	mCE ↓	53.41	52.09	**51.35**	52.52
Shot Noise	Top-1 ↑	47.96	49.03	**50.51**	49.66
	mCE ↓	58.18	56.98	**55.33**	56.28
Snow	Top-1 ↑	36.91	37.31	**37.32**	37.12
	mCE ↓	72.78	72.32	**72.31**	72.54
Zoom Blur	Top-1 ↑	35.03	35.88	**36.14**	35.20
	mCE ↓	81.38	80.31	**79.99**	81.17

to MomentumSMoE, namely, Nesterov accelerated gradient [48], time-varying momentum with scheduled restart [64, 47], RMSprop [27], sharpness aware minimization [64] (SAM), negative momentum [20], and complex momentum [41]. We report their results on clean and word swap attacked WikiText-103 language modeling task in Table 9. The nature of SAM is to find local minima where the geometry of the loss landscape is flat to improve the generalization ability of the model. This aligns with the objective of improving the robustness of models to distribution shifts. Hence, we implement SAM in V-MoE as well and provide a comparison with V-MoE, MomentumV-MoE

26
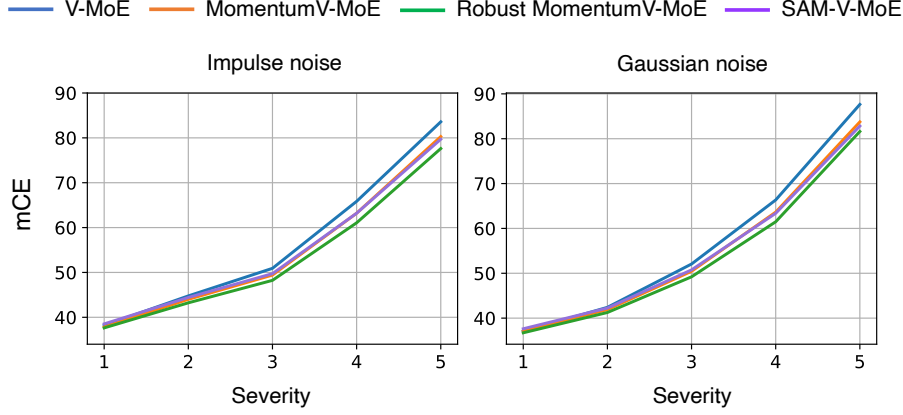
Figure 14: mCE (lower is better) of baseline V-MoE, MomentumV-MoE, Robust MomentumV-MoE and SAM-V-MoE on increasing severities of impulse noise and gaussian noise corruption. As the severity increases, the effect of momentum, robust momentum and SAM becomes increasingly apparent.

Table 9: Perplexity (PPL) results on clean and word swap attacked WikiText-103 validation and test data for baseline SMoE, NAG-SMoE, SR-SMoE, rms-SMoE, SAM-SMoE, Robust MomentumSMoE, Negative-MomentumSMoE, and Complex-MomentumSMoE.

Model/Metric	Clean WikiText-103		Attacked WikiText-103	
	Valid PPL ↓	Test PPL ↓	Valid PPL ↓	Test PPL ↓
SMoE (baseline)	33.76	35.55	42.24	44.19
NAG-SMoE	33.83	35.46	41.94	43.97
SR-SMoE	32.96	35.01	41.21	43.72
rms-SMoE	32.43	34.25	40.58	42.60
SAM-SMoE	33.39	35.05	41.47	43.44
Robust MomentumSMoE	33.22	34.45	41.49	42.78
Negative-MomentumSMoE	33.48	35.09	41.68	43.62
Complex-MomentumSMoE	**32.08**	**33.34**	**40.24**	**41.66**

and Robust MomentumV-MoE in Table 10. In all models, we replace all SMoE layers with their corresponding newly derived layer unless stated otherwise.

Nesterov accelerated gradient (NAG): Nesterov accelerated gradient (NAG) [48] takes the momentum method a step further by looking ahead to where the momentum term will move the parameters and providing a correction. The foresight of the update prevents the parameters from moving in an undesirable direction too quickly, preventing large oscillations especially when the learning rate is high. We implement NAG in our SMoE gradient framework as

$$\boldsymbol{p}_t = -\gamma f(\boldsymbol{x}_t - \mu \boldsymbol{p}_{t-1}) + \mu \boldsymbol{p}_{t-1}; \quad \boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{p}_t$$

where $f(\boldsymbol{x}_t)$ is the SMoE output and $\mu \in (0,1)$ and $\gamma > 0$ are two hyperparameters corresponding to the momentum coefficient and step size respectively. We refer to this implementation as a NAG-SMoE.

Time-varying momentum with scheduled restart: An enhancement of the time-varying momentum covered in Section E.1 is to include a scheduled restart for the momentum parameter, $t-1/t+2$. Such a modification can help to recover an optimal convergence rate as accelerated methods do not necessarily maintain a monotonic decrease in objective value [64, 47]. As our model has 6 layers, we choose to restart after layer 3 and the RS-SMoE update is

$$\boldsymbol{p}_t = -f(\boldsymbol{x}_t) + \frac{t \mod 3}{t \mod 3 + 3}\boldsymbol{p}_{t-1}; \quad \boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \gamma \boldsymbol{p}_t$$

where $f(\boldsymbol{x}_t)$ is the SMoE output and $\gamma > 0$ is the step size.

RMSprop: RMSprop is an adaptive step size algorithm that scales the gradient by an exponentially decaying average of the squared gradients [27]. The algorithm replaces a global learning rate, which

27

Table 10: Top-1 accuracy (%) and mean corruption error (mCE) of V-MoE, MomentumV-MoE, Robust MomentumV-MoE and SAM-V-MoE on validation and test ImageNet-1K data and popular standard robustness benchmarks for image classification.

Model	Valid IN-1K Top-1 ↑	Test IN-1K Top-1 ↑	IN-R Top-1 ↑	IN-A Top-1 ↑	IN-C Top-1 ↑	IN-C mCE ↓
V-MoE (baseline)	76.49	73.16	36.10	5.25	46.98	67.14
MomentumV-MoE	**76.92**	**73.26**	37.45	**6.48**	48.11	65.77
Robust MomentumV-MoE	76.66	73.20	**37.57**	6.37	**48.82**	**64.92**
SAM-V-MoE	76.26	72.84	36.64	6.27	48.05	65.88

is difficult to tune due to the widely differing magnitudes of the gradients of each parameter, with one that adapts to the individual parameter gradients. In addition, by keeping a running average of the squared gradients, we avoid extreme fluctuations in the step size due to stochastic batch updates. Incorporating rmsprop into the SMoE optimization perspective, we have the following update in our rms-SMoE layer,

$$\boldsymbol{p}_t = \mu \boldsymbol{p}_{t-1} + (1-\mu)f(\boldsymbol{x}_t)^2; \quad \boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \frac{\gamma}{\sqrt{\boldsymbol{p}_t + \epsilon}}f(\boldsymbol{x}_t)$$

where $f(\boldsymbol{x}_t)$ is the SMoE output, $\mu \in (0,1)$ is the moving average parameter and, $\gamma > 0$ the step size. Similar to AdamSMoE, we experience some instability when using rms-SMoE to replace all SMoE layers. Hence, we follow suit, and only replace the first layer with rms-SMoE and replace subsequent SMoE layers with MomentumSMoE.

Sharpness-Aware Minimization: In a standard machine learning approach, training models with the usual optimization algorithms minimize the training empirical loss, which is typically non-convex. This results in multiple local minima that may have similar loss values but lead to vastly different generalization abilities. As such, models that perform well on training data, may still have inferior validation performance. From studies relating the geometry of the loss landscape to generalization, specifically, and intuitively, a flatter minima should yield greater generalization ability [17, 29, 31]. Further, this would increase the model's robustness to distribution shifts in input data or labels.

The sharpness-aware minimization (SAM) algorithm [64] aims to take advantage of this relationship and seek not just any local minimum during training, but a minima whose neighbourhood also has uniformly low loss values, in other words, lower sharpness, to improve robustness. Implementing the algorithm with the l_2-norm in the SMoE optimization framework yields the SAM-SMoE layer

$$\hat{\epsilon}(\boldsymbol{x}_t) = \rho f(\boldsymbol{x}_t)/\|f(\boldsymbol{x}_t)\|_2; \quad \boldsymbol{p}_t = f(\boldsymbol{x}_t + \hat{\epsilon}(\boldsymbol{x}_t)); \quad \boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \gamma \boldsymbol{p}_t$$

where $f(\boldsymbol{x}_t)$ is the SMoE output, $\rho > 0$ is a hyperparameter controlling the size of the neighbourhood and, $\gamma > 0$ the step size.

Complex and Negative Momentum: Classic momentum works well in a gradient-based optimization when the Jacobian of our update step, as a fixed-point operator, has real eigenvalues. However, this might not always be the case. Negative momentum, which simply chooses negative momentum parameters, is preferred by operators with complex eigenvalues [20]. In situations where the spectrum is purely imaginary or has mixtures of complex and imaginary eigenvalues, complex momentum is more robust than both classic and negative momentum [41]. This is due to its oscillations at fixed frequencies between adding and subtracting the momentum term. We oscillate the sign of the momentum term by choosing a complex momentum parameter and then updating our weights by only the real part of the momentum term. This translates to the following update in the Complex-MomentumSMoE layer:

$$\boldsymbol{p}_t = -f(\boldsymbol{x}_t) + \mu \boldsymbol{p}_{t-1}; \quad \boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \gamma \Re(\boldsymbol{p}_t), \tag{21}$$

where $\mu \in \mathbb{C}$ and $\gamma > 0$ are hyperparameters corresponding to the momentum coefficient and step size, respectively and $\Re$ extracts the real component of the momentum term.

Results: On the language modeling task, though all models, except NAG-SMoE on clean WikiText-103 validation data, do outperform the baseline across all metrics, most have only marginal gains that fall short of the performance gap achieved with MomentumSMoE and AdamSMoE. This is

28

Table 11: Run time per sample, memory and number of parameters of MomentumSMoE and AdamSMoE as compared to the baseline SMoE during training and test time.

Model	Sec/Sample (Training)	Sec/Sample (Test)	Memory (Training)	Memory (Test)	Parameters
SMoE (baseline)	0.0315	0.0303	22168MB	17618MB	216M
MomentumSMoE	0.0317	0.0304	22168MB	17618MB	216M
AdamSMoE	0.0321	0.0307	22168MB	17618MB	216M

Table 12: Total computation time for SMoE, MomentumSMoE and AdamSMoE to reach 38 PPL on WikiText-103 validation data.

Model	Time (minutes)
SMoE (baseline)	85.56
MomentumSMoE	**81.77**
AdamSMoE	84.23

with the exception of Complex-MomentumSMoE, which has an even greater improvement over the baseline than MomentumSMoE. Complex-MomentumSMoE's enhanced results further verify the power and promise of our momentum-based framework in designing SMoE. On the computer vision task, we find that SAM-V-MoE does perform relatively well on corruption benchmarks, exceeding the baseline by more than 1% on ImageNet-A and ImageNet-C.

E.5 Comparison of Computational Efficiency

A common consideration when introducing modified layers into deep models is the potential increase in computational overhead. We aim to alleviate that concern by providing the run time per sample, memory and number of parameters of MomentumSMoE and AdamSMoE as compared to the baseline SMoE during both training and test time in Table 11. We also provide the total computation time required for all models to reach the same PPL level on WikiText-103 validation data in Table 12. We observe that MomentumSMoE and AdamSMoE are comparable to the baseline SMoE across all metrics at both training and test time with negligible computational cost.

F Broader Impacts

Our research enhances both clean data handling and robust performance, particularly in socially impactful domains. Notably, we demonstrate improved results in object recognition, benefiting self-driving cars, and language modeling, enhancing AI chatbot assistants. We show significant advancements in resisting data perturbation, aiming to protect critical AI systems from malicious actors. Furthermore, we achieve competitive performance in language modeling with contaminated data, reflecting real-world scenarios where data is often imperfect. While the potential for AI misuse exists, our work provides substantial improvements in fundamental architectures and theory, which we hope will lead to further socially beneficial outcomes.

29