A APPENDIX

A.1 NATURAL LANGUAGE UNDERSTANDING

A.1.1 DATA

GLUE is a collection of nine NLU tasks. The benchmark includes question answering (Rajpurkar et al., 2016), linguistic acceptability (CoLA, Warstadt et al. 2019), sentiment analysis (SST, Socher et al. 2013), text similarity (STS-B, Cer et al. 2017), paraphrase detection (MRPC, Dolan & Brockett 2005), and natural language inference (RTE & MNLI, Dagan et al. 2006; Bar-Haim et al. 2006; Giampiccolo et al. 2007; Bentivogli et al. 2009; Williams et al. 2018) tasks. Details of the GLUE benchmark, including tasks, statistics, and evaluation metrics, are summarized in Table 13.

All the texts were tokenized using wordpieces, and were chopped to spans no longer than 512 tokens.

A.1.2 TRAINING DETAILS

To fine-tune BERT-base and RoBERTa-large models on individual tasks, we append a task-specific fully-connected classification layer to them as in Devlin et al. (2018).

Table 7 present the hyper-parameter configurations. We tune this set of hyper-parameters on a single seed, and report the averaged results obtained with the same configuration over all seeds. For SAGE experiments, We slightly tune β_0 within a range of 0.1 on different seeds. We apply a linear weight decay rate of 0.01 and a gradient norm clipping threshold of 1 for all experiments. All experiments are conducted on Nvidia V100 GPUs.

Hyper-param	**Experiment**	**RTE**	**MRPC**	**CoLA**	**SST-2**	**STS-B**	**QNLI**	**QQP**	**MNLI**
Learning Rate	$\text{BERT}_{\text{BASE}}$, Adam	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5	2e-5	2e-5
	$\text{BERT}_{\text{BASE}}$, Adam-SAGE	1e-4	8e-5	8e-5	3e-5	1e-4	8e-5	4e-5	5e-5
	$\text{BERT}_{\text{BASE}}$, Adamax	1e-4	1e-4	1e-4	5e-5	1e-4	1e-4	1e-4	8e-5
	$\text{BERT}_{\text{BASE}}$, Adamax-SAGE	3e-4	3e-4	2e-4	2e-4	5e-4	5e-4	3e-4	2e-4
	$\text{RoBERTa}_{\text{LARGE}}$, Adamax	5e-5	5e-5	3e-5	1e-5	5e-5	1e-5	1e-4	1e-5
	$\text{RoBERTa}_{\text{LARGE}}$, Adamax-SAGE	6e-5	2e-4	8e-5	2e-5	8e-5	3e-5	2e-4	8e-5
β_0	$\text{BERT}_{\text{BASE}}$, Adam-SAGE	0.60	0.80	0.70	0.80	0.60	0.70	0.75	0.70
	$\text{BERT}_{\text{BASE}}$, Adamax-SAGE	0.65	0.80	0.75	0.70	0.75	0.70	0.75	0.85
	$\text{RoBERTa}_{\text{LARGE}}$, Adamax-SAGE	0.75	0.65	0.70	0.75	0.80	0.80	0.65	0.60
Batch Size	$\text{BERT}_{\text{BASE}}$	16	8	32	32	32	32	32	32
	$\text{RoBERTa}_{\text{LARGE}}$	16	8	32	32	32	32	32	32
Epoch	$\text{BERT}_{\text{BASE}}$	6	6	6	6	6	3	6	3
	$\text{RoBERTa}_{\text{LARGE}}$	15	6	6	6	10	10	15	3
Dropout	$\text{BERT}_{\text{BASE}}$	0.1	0.1	0.1	0.1	0.1	0.1	0.0	0.3
	$\text{RoBERTa}_{\text{LARGE}}$	0.1	0.1	0.1	0.1	0.1	0.1	0.0	0.3
Warmup	$\text{BERT}_{\text{BASE}}$	0.1	0.1	0.1	0.1	0.1	0.1	0.0	0.1
	$\text{RoBERTa}_{\text{LARGE}}$	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

Table 7: Hyper-parameter configurations for GLUE experiments. "Epoch" refers to the total training epochs; we adopt early-stopping strategy in practice. "Dropout" refers to classification layer dropout ratio. "Warmup" refers to the ratio of learning rate linear warmup iterations to total training iterations.

A.1.3 EVALUATION RESULTS

Statistics of the dev set results. Table 8 shows the standard deviation of the dev set results.

Average score computation formula. For dev set results, we first obtain a score for each task by averaging the scores of all metrics (e.g., Acc and F1) and test sets (e.g., MNLI-m and MNLI-mm) within this task, then compute a task-average score. For test set results, we directly averages scores of all reported metrics following Devlin et al. (2018).

Model	Optimizer	RTE	MRPC	CoLA	SST-2	STS-B	QNLI	QQP	MNLI
$\text{BERT}_{\text{BASE}}$	Adam-SAGE	0.35	0.32	0.85	0.25	0.12	0.06	0.05	0.06
	Adamax-SAGE	0.56	0.69	0.12	0.23	0.03	0.06	0.08	0.10
$\text{RoBERTa}_{\text{LARGE}}$	Adamax-SAGE	0.51	0.78	0.50	0.19	0.08	0.00	0.05	0.05

Table 8: Standard deviation of the dev set results.

A.2 Neural Machine Translation

A.2.1 Data

Table 9 shows the number of sentence pairs in each dataset. We use the standard newstest-2013 and newstest-2014 as dev and test set for WMT'16 En-De. We follow Ott et al. (2019) to split the dev/test sets for IWSLT'14 De-En.

All datasets are encoded using byte-pair encoding (BPE, Sennrich et al. (2016)). We preprocess IWSLT'14 De-En data following *fairseq*[8] and adopt the preprocessed WMT'16 En-De from Google[9].

Data	Train	Dev	Test
IWSLT'14 De-En	160K	7283	6750
WMT'16 En-De	4.5M	1061	1019

Table 9: The number of parallel sentences in NMT datasets.

A.2.2 Training Details

We adopt the Transformer-base model for both datasets. For IWSLT'14 De-En, we share the decoder and encoder output embeddings. For WMT'16 En-De, we share all the embeddings.

Table 10 presents the hyper-parameter configurations for the best models. We apply a linear weight decay rate of 1×10^{-4} and a label smoothing ratio of 0.1 for all experiments. All experiments are conducted on Nvidia V100 GPUs.

For IWSLT'14 De-En, we report the BLEU score of the best checkpoint using a beam size of 5 and length penalty of 1. For WMT'16 En-De, we report the average of the last 10 checkpoints with a beam size of 4 and length penalty of 0.6.

Hyper-param	Experiment	IWSLT'14 De-En	WMT'16 En-De
Learning Rate	Adam	5e-4	7e-4
	Adam-SAGE	1e-3	2e-3
β_0	Adam-SAGE	0.8	0.4
Batch size	Both	4096	32768
Epoch	Both	60	40
Dropout	Both	0.3	0.1
Warmup	Both	8000	4000

Table 10: Hyper-parameter configurations for NMT experiments. "Warmup" refers to the learning rate linear warmup iterations.

[8] https://github.com/pytorch/fairseq/blob/master/examples/translation

[9] https://pytorchnlp.readthedocs.io/en/latest/_modules/torchnlp/datasets/wmt.html

A.3 IMAGE CLASSIFICATION

A.3.1 DATA

For CIFAR100, we apply random cropping and random horizontal flipping to the training data.

A.3.2 TRAINING DETAILS

Table 11 present the hyper-parameter configurations for the best models. All experiments are conducted on Nvidia V100 GPUs.

Hyper-param	**Experiment**	**CIFAR100**	**ImageNet**
Learning Rate	ViT-B/32, SGD-SAGE	0.02	0.05
	ViT-L/32, SGD-SAGE	0.02	0.08
β_0	ViT-B/32, SGD-SAGE	0.95	0.95
	ViT-L/32, SGD-SAGE	0.85	0.95
Training Steps	All	10000	20000
Dropout	All	0.0	0.0

Table 11: Hyper-parameter configurations for ViT experiments on CIFAR100 and ImageNet.

A.4 SUPPLEMENTS FOR METHOD AND ANALYSIS

A.4.1 ADAM-SAGE ALGORITHM

Algorithm 2 Adam-SAGE ($\odot$ denotes Hadamard product and $\oslash$ denotes Hadamard division)

Input: Model parameters $\mathbf{\Theta} \in \mathbb{R}^J$; Data $\mathcal{D}$; Learning rate schedule $\eta(\cdot)$; Total training iteration T; Moving average coefficient $\beta_0, \beta_1, \beta_2$.

Initialize $\widehat{I}^{(0)}, m^{(0)}, v^{(0)} = \mathbf{0} \in \mathbb{R}^J$.
for $t = 1, ..., T$ **do**
  Sample a minibatch $b^{(t)}$ from $\mathcal{D}$.
  Compute gradient $g^{(t)} = \nabla_{\mathbf{\Theta}^{(t)}} L(b^{(t)}, \mathbf{\Theta}^{(t)})$.
  Compute sensitivity $I^{(t)} = |\mathbf{\Theta}^{(t)} \odot g^{(t)}|$.
  $m^{(t)} = \beta_1 m^{(t-1)} + (1-\beta_1) g^{(t)}$
  $v^{(t)} = \beta_2 v^{(t-1)} + (1-\beta_2)(g^{(t)})^2$
  $\widehat{I}^{(t)} = \beta_0 \widehat{I}^{(t-1)} + (1-\beta_0) I^{(t)}$.
  $\widehat{m}^{(t)} = m^{(t)} / (1-\beta_1)$
  $\widehat{v}^{(t)} = v^{(t)} / (1-\beta_2)$
  $\widehat{I}^{(t)} = \widehat{I}^{(t)} / (1-\beta_0)$
  $U^{(t)} = |I^{(t)} - \widehat{I}^{(t)}|$.
  Update $\mathbf{\Theta}^{(t+1)} = \mathbf{\Theta}^{(t)} - \eta^{(t)}((U^{(t)} + \epsilon) \odot \widehat{m}^{(t)}) \oslash ((\widehat{I}^{(t)} + \epsilon) \odot (\sqrt{\widehat{v}^{(t)}} + \epsilon)) \odot g^{(t)}$.
end for

A.4.2 IMPLEMENTATION DETAILS FOR SECTION 5.1

Figure 2 experiments: Due to the extremely large model size, we only sample 110K parameters per layer (in total 12×110K parameters) to calculate the distribution. We select the hyper-parameters that yield the best generalization performance on the BERT-base model, and we evaluate the sensitivity of each parameter using the entire training set.

Figure 4 experiments: Following previous experiment's practice, we randomly sample 110K parameters per layer (in total 12×110K parameters), and for visualization purposes, we plot 60 randomly selected iterations. We adopt the learning rate corresponding to the best training performance for both SAGE and the baselines.

A.4.3 Implementation Details for Section 5.2

Plotting the parameter sensitivity distribution throughout training can be computational expensive. The distribution varies significantly throughout training and often fails to provide a meaningful visualisation. As a result, we compute the structured sensitivity score instead of the parameter sensitivity score. Specifically, we compute a single sensitivity score for each Transformer weight block Θ at iteration t using the structured counterpart of the parameter sensitivity metric widely adopted in the existing structured pruning literature (Michel et al., 2019; Liang et al., 2021). Following common structured pruning practice, we split Transformer models into 12 feed-forward weight modules and 12 multi-head attention weight modules, and plot the average and variance of the sensitivity of these modules' sensitivity scores throughout the training.

We present the results obtained with the hyper-parameters that yield the best generalization performance on the BERT-base model for both Adamax (Baseline) and Adamax-SAGE (SAGE).

A.4.4 Ablation Study

To further interpret the role of the parameter sensitivity I and the local temporal variation U, we conduct an ablation study on these two factors. Specifically, we check five variants of Eq. (4):

$$\begin{aligned}
&\text{Variant 1.} \quad \eta_j^{(t)} = \eta^{(t)}(\widehat{I}_j^{(t)} + \epsilon)(U_j^{(t)} + \epsilon) \\
&\text{Variant 2.} \quad \eta_j^{(t)} = \eta^{(t)}(\widehat{I}_j^{(t)} + \epsilon)/(U_j^{(t)} + \epsilon) \\
&\text{Variant 3.} \quad \eta_j^{(t)} = \eta^{(t)}(\widehat{I}_j^{(t)} + \epsilon) \\
&\text{Variant 4.} \quad \eta_j^{(t)} = \eta^{(t)}/(\widehat{I}_j^{(t)} + \epsilon) \\
&\text{Variant 5.} \quad \eta_j^{(t)} = \eta^{(t)}(U_j^{(t)} + \epsilon)
\end{aligned}$$

For Variants 1,2 and 3, we aim to check the performance of giving a high/low-sensitive parameter a high/low, instead of low/high learning rate. Specifically, we place $(\widehat{I}_j^{(t)} + \epsilon)$ in the numerator, so that the learning rates increase for the high sensitive parameters and decrease for low sensitive parameters.

For Variants 4 and 5, we aim to check the performance of eliminating the influence of one of these factors. Specifically, we fix the local temporal variation term to 1 in Variant 4 and fix the sensitivity term to 1 in Variant 5.

A.4.5 Hyper-parameter Study

We investigate the influence of hyper-parameters learning rate and β_0 on the performance of SAGE (Figure 8). As can be seen, SAGE requires a larger learning rate than the baselines to offset the small scale of the modulation term (the optimal baseline learning rate lies in $5 \times 10^{-5} \sim 1 \times 10^{-4}$ for MNLI, $5 \times 10^{-4} \sim 7 \times 10^{-4}$ for IWSLT 14 De-En and $0.1 \sim 0.2$ for CIFAR10). Furthermore, switching to a larger learning rate requires a lower β_0 to maintain the same level of performance.

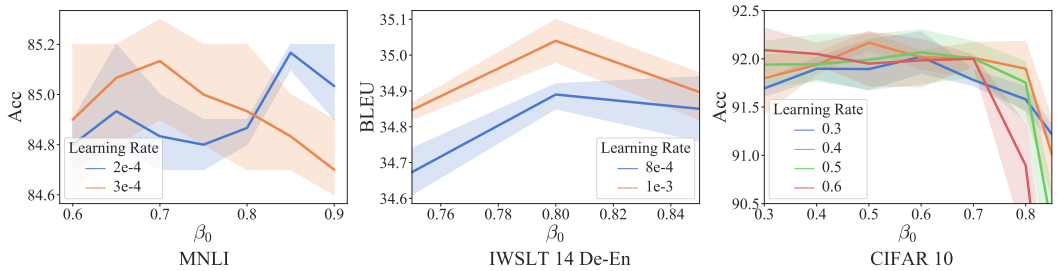

Figure 8: Parameter study on learning rate and β_0.

All five variants show no clear gain upon the baseline on both RTE and SST-2 datasets after careful hyper-parameter tuning. Specifically, we observe that the Variants 1 and 3 converge very fast at the early stage of training, and then quickly start overfitting. In Variants 2 and 4, the training collapses due to gradient explosion or vanishing.

Variant Name	**Learning Rate Modulating Term**	**RTE**	**SST-2**
Adam	1	63.5	92.9
Adam-SAGE	$(U_j^{(t)} + \epsilon)/(\widehat{I}_j^{(t)} + \epsilon)$	73.3	93.5
Variant 1.	$(\widehat{I}_j^{(t)} + \epsilon)(U_j^{(t)} + \epsilon)$	63.5	91.2
Variant 2.	$(\widehat{I}_j^{(t)} + \epsilon)/(U_j^{(t)} + \epsilon)$	Unconverged	Unconverged
Variant 3.	$\widehat{I}_j^{(t)} + \epsilon$	63.8	91.1
Variant 4.	$1/(\widehat{I}_j^{(t)} + \epsilon)$	Unconverged	Unconverged
Variant 5.	$U_j^{(t)} + \epsilon$	63.8	91.1

Table 12: Ablation study on parameter sensitivity and local temporal variations.

Corpus	Task	#Train	#Dev	#Test	#Label	Metrics
Single-Sentence Classification (GLUE)						
CoLA	Acceptability	8.5k	1k	1k	2	Matthews corr
SST	Sentiment	67k	872	1.8k	2	Accuracy
Pairwise Text Classification (GLUE)						
MNLI	NLI	393k	20k	20k	3	Accuracy
RTE	NLI	2.5k	276	3k	2	Accuracy
QQP	Paraphrase	364k	40k	391k	2	Accuracy/F1
MRPC	Paraphrase	3.7k	408	1.7k	2	Accuracy/F1
QNLI	QA/NLI	108k	5.7k	5.7k	2	Accuracy
Text Similarity (GLUE)						
STS-B	Similarity	7k	1.5k	1.4k	1	Pearson/Spearman corr

Table 13: Summary of the GLUE benchmark.