Efficiently Harnessing Parameter Importance for Better Training

Tianjian Li, Haoran Xu, Philipp Koehn, and Kenton Murray
Center of Language and Speech Processing
Johns Hopkins University
Baltimore, MD, 21218
{tli104,hxu64,phi,kenton}@jhu.edu

Abstract

Modern Neural Networks are often over-parameterized while a model's performance could be achieved with fewer parameters. Model pruning methods take advantage of over-parameterization by pruning redundant parameters that reduce computational overhead while maintaining comparable performance. Such methods locate the redundant parameters by estimating an importance score for each parameter and then prune the parameters with the lowest scores. Conversely, other lines of work [19, 40, 10] make use of parameter importance estimation by encouraging all parameters to contribute equally, which results in improved performance across different tasks. In our work, we break down the design of state-of-the-art methods that harness parameter importance during training and present a unified framework that establishes connections between them. Moreover, we identify the serious computational overhead induced by previous importance-guided optimization methods and propose a novel and lightweight approach that harnesses parameter importance during training that requires neither additional forward passes nor storing a full copy of the model throughout training. Comprehensive experiments on Machine Translation and Language Model Fine-tuning showcase the efficiency of our method. We are able to observe a 2.4 $\times$ boost in training speed and a 60.8% reduction in memory while achieving comparable performance to strong baselines.

1 Introduction

Model pruning removes redundant parameters to reduce the cost in both training [6, 22, 7, 42, 33, 37] and inference [39, 15, 8] while achieving comparable performance. The negligible degradation in performance after model pruning suggests that there exist many redundant parameters that do not contribute. Pruning algorithms first estimate an "importance score" for each individual [6, 7] or group [25, 39] of parameters, and remove parameters with low importance.

However, recent works [19, 40] hypothesize that redundant parameters are actually a by-product of our current optimization methods, and thus proposed optimizers [19] or auxiliary loss functions [40] to balance the contribution of parameters in neural networks. The effectiveness of these methods when applied to a wide range of tasks and model architectures indicates that such redundant parameters are insufficiently trained, and through better optimization techniques, we can improve the generalizability of models by explicitly using information about parameter contribution during training.

Findings on harnessing parameter importance during training can be useful not only in model pruning but also in model training echoing the findings in continual learning literature [14, 10] where regularizing the updates of important parameters can overcome catastrophic forgetting [9]. We show that such methods can be extended beyond continual learning and improve the generalizability of

Preprint. Under review.

Method	No Additional Forward Passes	No Additional Memory
Intra-Distillation [40]	✘	✘
EWC [14]	✘	✘
LFR-CM [10]	✘	✘
SAGE [19]	✔	✘
Weighted Freezing (Ours)	✔	✔

Table 1: An overview of the computational overhead induced by different methods that balance parameter contribution. Our proposed method requires neither multiple forward passes through the model nor additional memory to store the importance of each parameter.

models even trained on a single task. In our work, we present a unified view of methods that make use of parameter importance during training. Using this novel formulation, we are able to cultivate a deeper understanding of these methods and shed light on designing future optimization methods that make use of parameter importance.

Moreover, previous studies all introduce serious computational overhead either by estimating parameter importance after every gradient update, storing importance scores from previous iterations for a more accurate estimate of importance [14, 10, 19] or passing the input to the model multiple times [40] to train on a different subset of parameters by utilizing the randomness in dropout [32]. However, the computational overhead introduced by such methods can severely hinder training efficiency when the model and data size is scaled up. We provide a comparison between the computational overhead induced by previous state-of-the-art methods In Table 1.

In our work, we first break down the design of previous methods and draw the connection between them and Elastic Weight Consolidation (EWC) [14] to show that they either implicitly or explicitly encourage parameters to contribute equally (§3.1). We thus present a unified view of importance-guided optimization methods (§3.2). We then aim to reduce the computational overhead of such methods by proposing our new method that balances parameter contribution with minimal computational overhead (§4). To sum up, our contribution is two-fold:

1. We systematically study the state-of-the-art methods that encourage individual parameters to contribute equally and present a unified view of such methods, and demonstrate that they all induce serious computational overhead, which hinders their application on larger models;
2. We propose a novel method that balances parameter contribution with minimal computational overhead. Comprehensive experiments show the effectiveness and efficiency of our method across mono and multilingual language model fine-tuning and machine translation.

2 Preliminaries

In model pruning literature, the importance of a single parameter is often measured by the increase in the loss when the parameter is removed from the model, or by the magnitude of the parameter itself. In continual learning literature, the importance of a parameter is measured by the smoothness of the loss function around the parameter, often approximated by the empirical Fisher [24]. We provide an overview of parameter importance estimation metrics at §2.1 and provide the background on two existing methods that balance parameter contribution: §2.2 SAGE [19], §2.3 Intra-Distillation [40], as well as two methods that harness parameter importance in continual learning: §2.4 EWC [14] and §2.5 LFC-CM [10]. We then establish a connection between them in §3.

2.1 Importance Estimation of Parameters

Loss-Preserving. The canonical way to estimate the importance of parameters θ_j is measured by the difference in loss before and after θ_j is removed from the model by zeroing it out. We use $\theta_{j=0}$ to denote the set of parameters when $\theta_j = 0$. We use $\mathcal{I}(\theta_j)$ to denote the importance scores of one or a set of parameters θ_j and $\mathcal{L}(\theta)$ to denote the loss function, the loss-preserving metric for parameter importance estimation is given by:

$$\mathcal{I}(\theta_j) = |\mathcal{L}(\theta) - \mathcal{L}(\theta_{j=0})| \tag{1}$$

Approximating $\mathcal{L}(\theta_j)$ using a first-order Taylor expansion yields:

$$\mathcal{I}(\theta_j) = |\mathcal{L}(\theta) - \mathcal{L}(\theta_{j=0})| \approx \left| \cancel{\mathcal{L}(\theta)} - \left(\cancel{\mathcal{L}(\theta)} - \frac{\partial \mathcal{L}(\theta)}{\partial \theta_j}(\theta_j - 0) \right) \right| = \left| \frac{\partial \mathcal{L}(\theta)}{\partial \theta_j}(\theta_j - 0) \right|$$

This importance metric was originally proposed to prune convolutional networks [26, 27, 34], which is then extended to Transformer pruning [19, 40].

Magnitude. Another line of work [6, 7, 30] uses the magnitude of parameters as the importance to prune models:

$$\mathcal{I}(\theta_j) = |\theta_j|$$

Further research [23] shows that the magnitude-based metric also measures how well the parameter preserves loss, establishing an equivalency between the loss-preserving importance score and the magnitude-based metric importance score so that in model pruning, their interchangeable use is justified.

Fisher Information. The loss-preserving and magnitude-based metric is proposed to measure the importance of parameters in **models after training** for pruning. Although such methods of estimation are also well justified after minimal training [23], a more natural way to estimate parameter importance is by the local curvature of the loss function around a parameter (The Hessian of the loss function), which measures how well each parameter preserves the gradient dynamics **during training**. In practice, we approximate the Hessian with the empirical Fisher Information Matrix [5, 24].

$$I(\theta_j) = \nabla^2 \mathcal{L} \approx F(\theta_j) \approx \left(\frac{\partial \mathcal{L}(\theta)}{\partial \theta_j} \right) \left(\frac{\partial \mathcal{L}(\theta)}{\partial \theta_j} \right)^\top$$

2.2 Importance Weighted Optimization Methods

In this section, we introduce methods recently proposed to balance parameter contribution: SAGE [19] and Intra-Distillation [40]. We also revisit two methods in continual learning literature that utilizes parameter importance during training: Elastic Weight Consolidation [14] and LFR-CM [10].

SAGE [19] SAGE balances parameter contribution by using a larger learning rate on less important parameters. Formally, given a importance score $\mathcal{I}_t(\theta_j)$ of a set of parameters θ_j at update iteration t, the learning rate at time t is given by

$$\eta_t(\theta_j) = \eta_t \frac{|\mathcal{I}_t(\theta_j) - \hat{\mathcal{I}}_t(\theta_j)| + \epsilon}{\mathcal{I}_t(\theta_j) + \epsilon} \tag{2}$$

Where ϵ is a smoothing hyper-parameter and $\hat{\mathcal{I}}_t(\theta_j)$ is an exponential moving average to overcome the variance in estimating parameter importance from individual batches:

$$\hat{\mathcal{I}}_t(\theta_j) = \beta \hat{\mathcal{I}}_{t-1}(\theta_j) + (1 - \beta)\mathcal{I}_t(\theta_j)$$

SAGE introduces computational overhead in calculating the parameter importance in **every** mini-batch, which does not scale well when the number of parameters is large. In practice, they only adapt the learning rate of attention projection weights and feed-forward weights, which is the same set of parameters that dropout is applied to.

Intra-Distillation [40] Intra-distillation utilizes the randomness in dropout to train on a subset of parameters to balance contribution. Formally, we first pass the same input x to obtain k different

probability distributions $p_1, ...p_k$, the Intra-Distillation loss is the sum of the symmetric divergence between each distribution to the average of the distributions:

$$\bar{p} = \sum_{i=1}^{k} p_i$$
$$\mathcal{L}_{ID} = \sum_{i=1}^{k} \text{KL}(p_i \| \bar{p}) + \text{KL}(\bar{p} \| p_i) \tag{3}$$

Then Intra-Distillation loss is then added to the standard training objective:

$$\mathcal{L} = \mathcal{L}(\theta) + \alpha \mathcal{L}_{ID}$$

Where α is a scaling factor. Intra-distillation induces significant computational overhead. We need to compute k forward passes and store all of the output logits from the forward passes to compute the symmetric KL divergence between them.

Elastic Weight Consolidation [14] Elastic Weight Consolidation was proposed to solve the problem of catastrophic forgetting in continual learning by regularizing the updates of more important parameters. Suppose B is the task the model is currently trained on and A is the task that we don't want the model to forget, the loss objective for EWC is:

$$\mathcal{L} = \mathcal{L}_B(\theta_j) + \frac{\lambda}{2} F_j (\theta_j - \theta_{j,A})^2$$

Where $\theta_{j,A}$ is the parameter θ_j trained to convergence on task A, and F_j is the Fisher information importance of parameter θ_j.

LFR-CM [10] LFR-CM freezes the parameters with high Fisher information and only trains on the parameter with low Fisher information. The only difference is that EWC makes the update of parameters with high Fisher information smaller while LFR-CM freezes them. The loss function of LFR-CM is:

$$\mathcal{L} = \mathcal{L}_B(\theta_j) * \mathbb{1}(F_j \leq \tau)$$

Where τ is a pre-defined threshold for filtering out the important parameters that are frozen during training on task B. LFR-CM can be viewed as "Hard-EWC" in that instead of assigning a continuous importance score to each parameter, it assigns a binary label of important/non-important and freezes the parameters labeled as important instead of regularizing them from updating more aggressively.

3 A Unified View of Importance Weighted Optimization

In this section, we first empirically show that EWC can also improve performance on a single task by balancing parameter contribution (§3.1). We then theoretically draw connections between methods that adapt the learning rate according to parameter importance (SAGE) to methods that add a regularization term weighted by parameter importance (EWC) (§3.2). We analyze the induced computational overhead for each method and propose an efficient way to balance parameter contribution with minimal extra compute (§3.3). At last, we propose a unified framework that includes state-of-the-art importance-weighted optimization methods as instantiations (§3.4).

3.1 EWC Improves Model Generalizability on a Single Task

EWC regularizes important parameters from updating too aggressively from a consolidated weight learned from the previous task in continual learning. However, we can view each mini-batch of data as a separate task, we can apply EWC on a single task, and the loss becomes:

$$\mathcal{L} = \mathcal{L}_B(\theta_j^{t-1}) + \frac{\lambda}{2} F_j (\theta_j^{t-1} - \theta_j^{t-2})^2$$

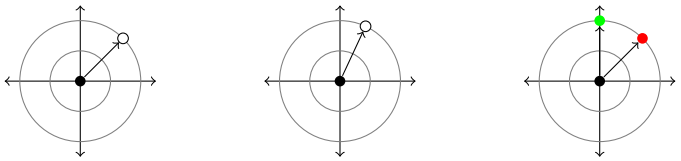

Figure 1: Illustration of the effect of SAGE and Weighted Freezing on gradient updates when the x direction has a higher importance score than the y direction. Standard updates treat both directions as equal (left) while SAGE updates more aggressively on the direction with a low importance score (middle). Weighted Freezing (right) assigns a higher probability for the parameter to update to the green dot and a lower probability for the parameter to update to the red dot.

We train a standard Transformer [36] model and apply our modified version of EWC after the warm-up steps on the IWSLT' 14 De-En Machine Translation dataset and compare it with a baseline of standard training. The results in Table 2 show that EWC can improve model performance even applied to a single task because it balances parameter contribution.

	Standard Training	EWC
IWSLT' 14 De-En	32.75	33.20

Table 2: BLEU scores on the IWSLT' 14 German-English Machine Translation task. Using EWC on a single task can also improve performance over standard training.

3.2 Connecting SAGE with EWC

Equation 2 describes the update rule for SAGE that scales down the learning rate for parameters with a high importance score, calibrated by an exponential moving average of previous iterations of parameter importance estimation. Here we simplify the parameter update rule for SAGE as:

$$\theta_j^t \leftarrow \theta_j^{t-1} - \eta \frac{C}{I_t} \nabla \mathcal{L}(\theta_j^{t-1}) \tag{4}$$

Where C is a constant that calibrates the uncertainty of the parameter importance calculated through a mini-batch of data.

We derive the update rule for EWC (λ is the regularization loss scaling constant):

$$\theta_j^t \leftarrow \theta_j^{t-1} - \eta \nabla \mathcal{L}(\theta_j^{t-1}) - \eta\lambda(\theta_j^{t-1} - \theta_{j,A}) \tag{5}$$

Rearranging 5 gives us:

$$\theta_j^t \leftarrow \theta_j^{t-1}(1 - \eta\lambda F_j) - \eta \nabla \mathcal{L}(\theta_j^{t-1}) + \eta\lambda F_i \theta_{j,A} \tag{6}$$

Both of these methods can be written in the following functional form:

$$\theta_j^t \leftarrow f(I_j)\theta_j^{t-1} + g(I_j)\eta \nabla \mathcal{L}(\theta_j) + h(I_j)$$

Where $f(I_j), g(I_j)$ and $h(I_j)$ are functions of the parameter importance score.

3.3 Weighted Freezing

Although previous methods have achieved success in balancing parameter contribution to boost model performance, they either require additional memory equivalent to the memory required to store a full copy of the model parameters (SAGE, EWC) or, additionally, multiple forward passes (Intra-Distillation).

We propose to freeze the important parameters and only train on the unimportant ones to encourage the equal contribution of parameters. While SAGE uses a moving average to calibrate the uncertainty, we calibrate the uncertainty by generating a Bernoulli mask on the gradients, such that important parameters have a higher chance of being frozen. Although we expect the mini-batch estimation of parameter importance to have high variance, it is still a good estimate of what are the most important parameters. The update rule for Weighted Freezing is

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla \mathcal{L}(\theta^{t-1}) * \textit{Bernoulli}(\tilde{I}_t) \tag{7}$$

Where $\tilde{I}_t = \frac{I_t - \max I_t}{\max I_t - \min I_t}$ is the normalized importance score at iteration t. We define our Weighted Freezing method in Algorithm 1.

Algorithm 1 Weighted Freezing ($\odot$ denotes Hadarmard Product)

Require: Function for estimating parameter importance I, Total training iterations T, Data $\mathcal{D} = (x, y)$, Loss function $\mathcal{L}$, Model parameters θ, learning rate schedule $\eta()$.
for $t = 1$ to T **do**
  Sample a mini-batch of data x_t, y_t
  Compute gradients $\nabla \mathcal{L}(\theta^t; x_t, y_t)$
  $I_t = I(\theta^t, \nabla \mathcal{L}(\theta^t; x_t, y_t))$ ▷ Importance score estimation
  $I_t = \frac{I_t - \min(I_t)}{\max(I_t) - \min(I_t)}$ ▷ Normalize scores
  $M = \textit{Bernoulli}(\tilde{I}_t)$ ▷ Generate masks
  $\nabla \mathcal{L}(\theta^t; x_t, y_t)) = \nabla \mathcal{L}(\theta^t; x_t, y_t)) \odot M$
  $\theta^{t+1} = \theta^t - \eta_t \nabla \mathcal{L}(\theta^t; x_t, y_t))$
end for

3.4 Unified View of Importance Weighted Optimization

Unifying equation 4 and 6, we cast existing methods that harness parameter importance during training as a modification term to the parameter itself $f(I)$, plus a modification term to the gradient $g(I)$, plus a constant $h(I)$ where each term is a function of the parameter importance.

Method	Importance Metric I	$f(I)$	$g(I)$	$h(I)$
SAGE [19]	Loss-Preserving	1	C/I	0
EWC [14]	Empirical Fisher	$1 - \eta\lambda I$	1	$\eta\lambda I\theta_A$
LFR-CM [10]	Empirical Fisher	1	$\mathbb{1}(I \leq \tau)$	0
Weighted Freezing (Ours)	Magnitude	1	$\textit{Bernoulli}(\tilde{I})$	0

Table 3: An overview of importance weighted optimization methods and the weights that depend on parameter importance metric I. Here τ in LFR-CM is a threshold to filter out important parameters to be frozen during training.

The main difference between the methods is in their way of calculating parameter importance: SAGE uses the loss-preserving metric while EWC and its variants use an approximated version of the empirical fisher. Empirically, we found that using the Magnitude metric Weighted Freezing performs the best (see table 6), while also being the most computationally efficient. Therefore Weighted Freezing uses the magnitude-based metric to estimate the importance of the parameters.

4 Experiments

We evaluate our method on three semantically diverse tasks in natural language processing to verify that our method improves the generalizability of models: zero-shot cross-lingual transfer fine-tuning on multilingual models, machine translation, and English fine-tuning on natural language understanding.

4.1 Datasets

We directly verify that balancing parameter contribution improves the generalizability of models using cross-lingual transfer tasks (§4.2). In this case, a multilingual model is fine-tuned on data for a task only in English but is tested in other languages. This requires strong generalizability of our model since there is a domain (language) mismatch between the training data and the test data. We experiment with a state-of-the-art multilingual encoder XLM-RoBERTa-Large [3]. We use the official Huggingface implementation [38], and choose XNLI [4] and PAWS-X [41], two textual entailment classification tasks, as well as two multilingual question answering datasets: MLQA [17] and TyDiQA [2] from the XTREME benchmark [12].

To verify that our method generalizes beyond text classification and to compare with previous methods, we experiment on the IWSLT' 14 Machine Translation dataset (§4.3). We use the official *fairseq* [28] codebase implementation and select the 'transformer_iwslt_de_en' as our model. We report the BLEU [29] scores for each translation direction.

4.2 Experiment Setups

For a fair comparison, we implement a modified version of SAGE that does not store a moving average of parameter importance, which roughly consumes the same amount of memory as storing an additional copy of the full model without the embedding and LayerNorm parameters. We denote our resource-constrained version of SAGE as "SAGE*". Since Intra-Distillation passes the same input through the model multiple times, we constrain the number of updates for Intra-Distillation to be one-third of other methods and set the number of forward passes to be 3 for a fair comparison. We denote this resource-constrained version of Intra-Distillation as "Intra-Distillation (1/3)".

We note that even under our resource-constrained setting, Intra-Distillation (1/3) and SAGE* still require more computational resources than our method since Intra-Distillation needs to store the output logits for every forward pass and SAGE requires you to calculate the Hadarmard product between the gradient and parameters for each weight matrix. We report the experiment results on the full version of SAGE and Intra-Distillation as "SAGE (Full)" and "Intra-Distillation (Full)".

4.3 Main Results

Cross-lingual Transfer. The results on 4 cross-lingual transfer tasks that require strong generalizability are in Table 4. Our weighted freezing method outperforms state-of-the-art methods under a constrained resource setting in all the tasks that we tested (first 4 rows), resulting in a 1.8 increase in accuracy in XNLI and a 0.2 increase in accuracy in PAWS-X over the best performance of state-of-the-art methods. On two question-answering tasks, weighted freezing also outperforms existing methods when using the same level of resources, and is only behind 0.6 F1 and 1.3 Exact Match against the best-performing method in each task.

	XNLI Acc.	PAWS-X Acc.	MLQA F1/EM	TyDiQA-GoldP F1/EM
With the same amount of compute				
Standard Training	76.4	87.6	71.2/54.8	66.9/46.2
SAGE*	76.2	87.0	69.5/51.0	67.0/45.9
Intra-Distillation (1/3)	77.1	88.0	72.1/54.2	**67.2**/46.5
Weighted Freezing (Ours)	**79.1**	**88.2**	**72.5/55.1**	67.1/**46.9**
With 2 × additional memory				
SAGE (Full)	78.4	87.9	73.1/**56.3**	67.4/47.3
Intra-Distillation (Full)	78.1	**88.6**	**73.5**/55.8	67.1/**48.4**

Table 4: Experiment Results on Cross-lingual Transfer on the development set of two semantic entailment classification tasks (XNLI, PAWS-X) and two question answering tasks (MLQA, TyDiQA-GoldP). Our method outperforms the state-the-of-art methods under a resource-constrained setting and achieves comparable performance when using much fewer resources. Note that Intra-Distillation (Full) also uses 3 × more compute.

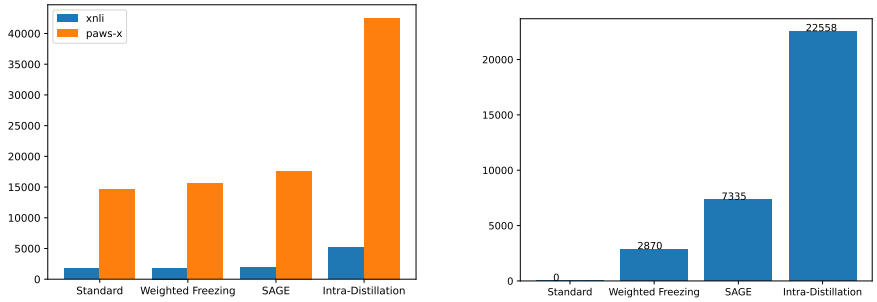

Figure 2: Averaged seconds per epoch (left) and additional memory used (right) for fine-tuning on PAWS-X and XNLI for different parameter importance weighted optimization methods, our method is 3× as faster as Intra-Distillation [40] and uses less than 50% of the additional memory compared to SAGE [19] while achieving comparable performance.

We included the full version of SAGE and Intra-Distillation in the bottom two rows of Table 4. Our method can perform comparably to the full version of SAGE and Intra-Distillation albeit using much fewer resources.

	De	Es	He	Ar
With the same amount of compute				
Standard Training	32.75	38.74	34.30	35.13
SAGE*	32.08	38.63	33.92	35.28
Intra-Distillation (1/3)	33.10	37.76	34.44	**36.21**
Weighted Freezing (Ours)	**33.30**	**39.15**	**36.02**	36.15
With 2 × additional memory				
SAGE (Full)	33.24	39.52	34.30	36.68
Intra-Distillation (Full)	**33.74**	**39.87**	35.71	**36.44**

Table 5: Results on the IWSLT' 14 X-En Machine Translation dataset. Weighted Freezing outperforms other methods with the same amount of compute, and is only behind 0.26 BLEU points when compared with methods that consume more than 2× of additional memory.

Machine Translation. The machine translation results from 4 directions on the IWSLT' 14 dataset is reported in Table 5. Our method outperforms strong baselines in 3 out of four translation directions under a resource-constrained setting and is only behind 0.26 BLEU points compared to the best-performing model.

4.4 A Detailed Study on Efficiency

We report the averaged clock time in seconds for training on 1 epoch and the memory consumption in MiB on previous methods and Weighted Freezing in Figure 2. All experiments are conducted on a single NVIDIA RTX A6000 (48GB). Intra-Distillation induces serious overhead by requiring multiple forward passes through the model (in this case 3), resulting in more time and memory to store the logits of each forward pass. Although SAGE uses a comparable amount of time, it requires more than 2× the additional memory over weighted freezing. This means that applying SAGE would result in an out-of-memory issue on a more common, commodity-level 24GB GPU, while Weighted Freezing does not. This is a necessary overhead for SAGE since it needs to additionally store the exponential moving average of importance score for every parameter.

4.5 Ablation of Importance Metrics

We ablate the effect of different parameter importance metrics on our method and SAGE. We empirically found that using the magnitude-based metric for Weighted Freezing performs the best and requires much fewer floating point operations than calculating other metrics.

	IWSLT' 14 De-En BLEU	PAWS-X Acc.
Standard Training	32.75	87.6
WF Magnitude	**33.30**	**88.2**
WF Loss-preserving	32.18	88.1
WF Fisher	32.74	87.2
SAGE Magnitude	31.27	87.9
SAGE Loss-preserving	33.24	87.9
SAGE Fisher	32.15	87.4

Table 6: Ablation studies on different importance metrics for Weighted Freezing (WF) and SAGE.

We did not observe an agreement on which metric performs the best for SAGE on the two tasks we tested. We also found that using the empirical fisher information performs the worst for both our method and SAGE. We hypothesize that it is because there is a higher variance in estimating the empirical fisher using mini-batches. Therefore, we recommend using the magnitude-based metric to estimate parameter importance during training since it is the easiest to compute and performs comparably to the loss-preserving metric.

5 Related Works

Parameter Importance Estimation. Numerous works have proposed ways to quantify the importance of individual parameters inside a neural network [16, 27, 14, 22, 23, 37]. The most prominent way is to utilize a second-order Taylor expansion [16, 11] to measure the increase in loss when the parameter is zeroed out. Such a method is then applied to convolutional neural network [27] pruning, as well as Transformers pruning [43, 18]. Another line of work uses the magnitude of the parameter as its importance, under the assumption that in converged neural networks, the gradient should be negligible. Magnitude-based importance is also applied to model pruning of convolutional neural nets [6] and large transformer models [20, 30, 1]. [23] shows equivalency between magnitude and loss-preserving metric in neural network pruning. Another line of work uses the curvature around parameters, approximated by the trace of the empirical Fisher [24] information matrix to quantify the importance of individual parameters [14, 35, 10, 21].

Importance Guided Optimization. Parameter importance is useful beyond model pruning: by forcing parameters to contribute equally, one can learn models that have better generalization performance. [19] uses a larger learning rate on less important parameters, and [40] shows that self-distillation has an implicit effect of balancing parameter contribution. [14, 35] uses the empirical fisher to guide model updates in continual learning. [10] also uses the empirical fisher to find parameters that are in a smooth region and only update them in continual learning of neural machine translation. Studies have found that the trace of the Fisher information matrix is correlated with the learning dynamics as well as the generalizability of models [21]

6 Conclusion

We propose a unified view of methods that harnesses parameter contribution throughout training and identify the serious computational overhead induced by existing methods which hinders their usage when the model size is scaled up. We propose Weighted Freezing which balances parameter contribution with minimal additional resources. We demonstrate that Weighted Freezing improves the performance of Transformer based models on various NLP tasks.

We acknowledge that we only conducted experiments on Transformer [36] based models, though there exist massive redundant neurons in other network architectures [6, 7, 31] which we expect to behave similarly with our method.

References

[1] Tianlong Chen et al. "The Lottery Ticket Hypothesis for Pre-trained BERT Networks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 15834–15846. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/b6af2c9703f203a2794be03d443af2e3-Paper.pdf.

[2] Jonathan H. Clark et al. "TyDi QA: A Benchmark for Information-Seeking Question Answering in Typologically Diverse Languages". In: *Transactions of the Association for Computational Linguistics* 8 (2020), pp. 454–470. DOI: 10.1162/tacl_a_00317. URL: https://aclanthology.org/2020.tacl-1.30.

[3] Alexis Conneau et al. "Unsupervised Cross-lingual Representation Learning at Scale". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 8440–8451. DOI: 10.18653/v1/2020.acl-main.747. URL: https://aclanthology.org/2020.acl-main.747.

[4] Alexis Conneau et al. "XNLI: Evaluating Cross-lingual Sentence Representations". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 2475–2485. DOI: 10.18653/v1/D18-1269. URL: https://aclanthology.org/D18-1269.

[5] R. A. Fisher. "Theory of Statistical Estimation". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 22.5 (1925), pp. 700–725. DOI: 10.1017/S0305004100009580.

[6] Jonathan Frankle and Michael Carbin. "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=rJl-b3RcF7.

[7] Jonathan Frankle et al. "Linear Mode Connectivity and the Lottery Ticket Hypothesis". In: *Proceedings of the 37th International Conference on Machine Learning*. ICML'20. JMLR.org, 2020.

[8] Elias Frantar and Dan Alistarh. "Optimal Brain Compression: A Framework for Accurate Post-Training Quantization and Pruning". In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh et al. 2022. URL: https://openreview.net/forum?id=ksVGCOlOEba.

[9] Robert M. French. "Catastrophic forgetting in connectionist networks". In: *Trends in Cognitive Sciences* 3.4 (1999), pp. 128–135. ISSN: 1364-6613. DOI: https://doi.org/10.1016/S1364-6613(99)01294-2. URL: https://www.sciencedirect.com/science/article/pii/S1364661399012942.

[10] Shuhao Gu, Bojie Hu, and Yang Feng. "Continual Learning of Neural Machine Translation within Low Forgetting Risk Regions". In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 1707–1718. URL: https://aclanthology.org/2022.emnlp-main.111.

[11] B. Hassibi, D.G. Stork, and G.J. Wolff. "Optimal Brain Surgeon and general network pruning". In: *IEEE International Conference on Neural Networks*. 1993, 293–299 vol.1. DOI: 10.1109/ICNN.1993.298572.

[12] Junjie Hu et al. "XTREME: A Massively Multilingual Multi-task Benchmark for Evaluating Cross-lingual Generalisation". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 4411–4421. URL: https://proceedings.mlr.press/v119/hu20b.html.

[13] Jared Kaplan et al. *Scaling Laws for Neural Language Models*. 2020. arXiv: 2001.08361 [cs.LG].

[14] James Kirkpatrick et al. "Overcoming catastrophic forgetting in neural networks". In: *Proceedings of the National Academy of Sciences* 114.13 (2017), pp. 3521–3526. DOI: 10.1073/pnas.1611835114. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.1611835114. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1611835114.

[15] Woosuk Kwon et al. "A Fast Post-Training Pruning Framework for Transformers". In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh et al. 2022. URL: https://openreview.net/forum?id=0GRBKLBjJE.

[16] Yann LeCun, John Denker, and Sara Solla. "Optimal Brain Damage". In: *Advances in Neural Information Processing Systems*. Ed. by D. Touretzky. Vol. 2. Morgan-Kaufmann, 1989. URL: https://proceedings.neurips.cc/paper_files/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf.

[17] Patrick Lewis et al. "MLQA: Evaluating Cross-lingual Extractive Question Answering". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 7315–7330. DOI: 10.18653/v1/2020.acl-main.653. URL: https://aclanthology.org/2020.acl-main.653.

[18] Chen Liang et al. "HomoDistil: Homotopic Task-Agnostic Distillation of Pre-trained Transformers". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=D7srTrGhAs.

[19] Chen Liang et al. "No Parameters Left Behind: Sensitivity Guided Adaptive Learning Rate for Training Large Transformer Models". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=cuvga_CiVND.

[20] Chen Liang et al. "Super Tickets in Pre-Trained Language Models: From Model Compression to Improving Generalization". In: *arXiv preprint arXiv:2105.12002* (2021).

[21] Chen Cecilia Liu et al. *Improving Generalization of Adapter-Based Cross-lingual Transfer with Scheduled Unfreezing*. 2023. arXiv: 2301.05487 [cs.CL].

[22] Zhuang Liu et al. "Rethinking the Value of Network Pruning". In: *ICLR*. 2019.

[23] Ekdeep Singh Lubana and Robert Dick. "A Gradient Flow Framework For Analyzing Network Pruning". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=rumv7QmLUue.

[24] Alexander Ly et al. "A Tutorial on Fisher information". In: *Journal of Mathematical Psychology* 80 (2017), pp. 40–55. ISSN: 0022-2496. DOI: https://doi.org/10.1016/j.jmp.2017.05.006. URL: https://www.sciencedirect.com/science/article/pii/S0022249617301396.

[25] Paul Michel, Omer Levy, and Graham Neubig. "Are Sixteen Heads Really Better than One?" In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/2c601ad9d2ff9bc8b282670cdd54f69f-Paper.pdf.

[26] Pavlo Molchanov et al. "Importance Estimation for Neural Network Pruning". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 11256–11264. DOI: 10.1109/CVPR.2019.01152.

[27] Pavlo Molchanov et al. "Pruning Convolutional Neural Networks for Resource Efficient Inference". In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/forum?id=SJGCiw5gl.

[28] Myle Ott et al. "fairseq: A Fast, Extensible Toolkit for Sequence Modeling". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 48–53. DOI: 10.18653/v1/N19-4009. URL: https://aclanthology.org/N19-4009.

[29] Kishore Papineni et al. "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: https://aclanthology.org/P02-1040.

[30] Sai Prasanna, Anna Rogers, and Anna Rumshisky. "When BERT Plays the Lottery, All Tickets Are Winning". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 3208–3229. DOI: 10.18653/v1/2020.emnlp-main.259. URL: https://aclanthology.org/2020.emnlp-main.259.

[31] Ghada Sokar et al. *The Dormant Neuron Phenomenon in Deep Reinforcement Learning*. 2023. arXiv: 2302.12902 [cs.LG].

[32] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[33] Jingtong Su et al. “Sanity-Checking Pruning Methods: Random Tickets can Win the Jackpot”. In: *arXiv preprint arXiv:2009.11094* (2020).

[34] L. Theis et al. “Faster gaze prediction with dense networks and Fisher pruning”. arXiv:1801.05787. 2018. URL: https://arxiv.org/abs/1801.05787.

[35] Brian Thompson et al. “Overcoming Catastrophic Forgetting During Domain Adaptation of Neural Machine Translation”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 2062–2068. DOI: 10.18653/v1/N19-1209. URL: https://aclanthology.org/N19-1209.

[36] Ashish Vaswani et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[37] Chaoqi Wang, Guodong Zhang, and Roger Grosse. “Picking Winning Tickets Before Training by Preserving Gradient Flow”. In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=SkgsACVKPH.

[38] Thomas Wolf et al. “Transformers: State-of-the-Art Natural Language Processing”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[39] Mengzhou Xia, Zexuan Zhong, and Danqi Chen. “Structured Pruning Learns Compact and Accurate Models”. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 1513–1528. DOI: 10.18653/v1/2022.acl-long.107. URL: https://aclanthology.org/2022.acl-long.107.

[40] Haoran Xu, Philipp Koehn, and Kenton Murray. “The Importance of Being Parameters: An Intra-Distillation Method for Serious Gains”. In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 170–183. URL: https://aclanthology.org/2022.emnlp-main.13.

[41] Yinfei Yang et al. “PAWS-X: A Cross-lingual Adversarial Dataset for Paraphrase Identification”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3687–3692. DOI: 10.18653/v1/D19-1382. URL: https://aclanthology.org/D19-1382.

[42] Haoran You et al. “Drawing Early-Bird Tickets: Toward More Efficient Training of Deep Networks”. In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=BJxsrgStvr.

[43] Qingru Zhang et al. “PLATON: Pruning Large Transformer Models with Upper Confidence Bound of Weight Importance”. In: *arXiv preprint arXiv:2206.12562* (2022).

A Text Classification

	RTE Acc.	CoLA Mcc.	MRPC F1
Standard Training	67.9	61.3	89.1
SAGE-	68.2	58.3	88.4
Intra-Distillation (K=3)	68.1	61.0	89.3
Weighted Freezing (Ours)	69.0	62.3	**90.2**
SAGE (Full)	69.3	60.3	89.0
Intra-Distillation (Full)	**71.4**	**62.5**	89.9

Table 7: Experiment Results of Weighted Freezing on 3 text classification tasks from the GLUE benchmark. The first 4 rows correspond to the resource-constrained setting. Our method outperforms state-of-the-art methods with the same amount of resources and performs comparably to the full version of these methods with much fewer resources.

B Hyper-Parameters: XLM-RoBERTa

	Learning Rate	Batch Size	Max Length	Epochs	Optimizer
XNLI	5e-5, 1e-5, 1e-6	16	256	3	AdamW
PAWS-X	5e-5, 1e-5, 1e-6	16	256	5	AdamW
TyDiQA, MLQA	7e-6, 3e-6	16	512	5	AdamW

Table 8: Hyper-parameters for XLM-R experiments

C Hyper-Parameters: Machine Translation

Warm Up Updates	4000
Batch Tokens	4096
Max Updates	20,000
Adam β	(0.9, 0.99)
Max Learning Rate	5e-4, 1e-3
Weight Decay	0.0001

Table 9: Hyper-parameters for IWSLT' 14 Machine Translation

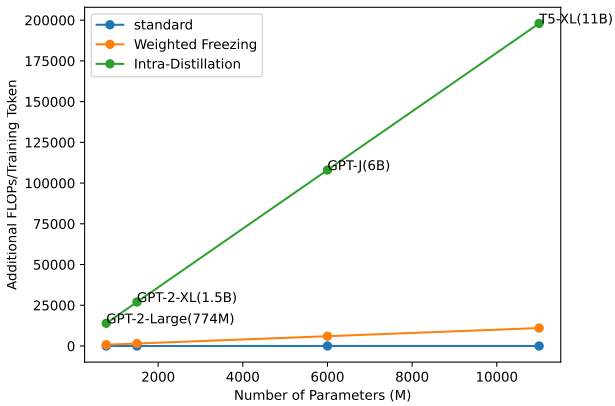

Figure 3: A comparison between the number of additional FLOPs required for standard training Intra-Distillation, and Weighted Freezing (Ours), calculated according to [13]. Our method scales well as the model size becomes larger.

D Scaling