Early Weight Averaging meets High Learning Rates for LLM Pre-training

Sunny Sanyal[*] UT Austin
Atula Neerkaje[*] UT Austin
Jean Kaddour UCL
Abhishek Kumar Google DeepMind
Sujay Sanghavi UT Austin

Abstract

Training Large Language Models (LLMs) incurs significant cost; hence, any strategy that accelerates model convergence is helpful. In this paper, we investigate the ability of a simple idea – checkpoint averaging along the trajectory of a training run – to improve both convergence and generalization quite early during training. Here we show that models trained with high learning rates observe higher gains due to checkpoint averaging. Furthermore, these gains are amplified when checkpoints are sampled with considerable spacing in training steps. Our training recipe outperforms conventional training and popular checkpoint averaging baselines such as exponential moving average (EMA) and stochastic moving average (SWA). We evaluate our training recipe by pre-training LLMs, where high learning rates are inherently preferred due to extremely large batch sizes. Specifically, we pre-trained nanoGPT-2 models of varying sizes—small (125M), medium (335M), and large (770M)—on the OpenWebText dataset, comprised of 9B tokens. Additionally, we present results for publicly available Pythia LLMs, ranging from 1B to 12B, which were trained on the PILE-deduped dataset containing 207B tokens. Code is available at `https://github.com/sanyalsunny111/Early_Weight_Avg`.

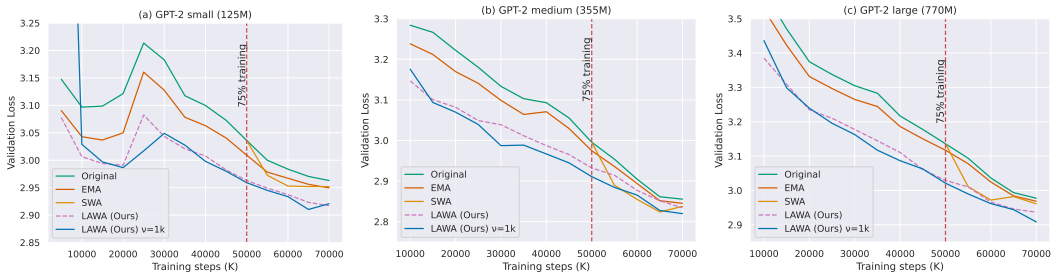
1 Introduction

Large Language Models (LLMs) have made a significant leap from billion to trillion scale, both in terms of parameters [5, 31] and pre-training data size [11] [37, 38]. This surge in both data and model size has rendered LLM pre-training increasingly slow and resource-intensive. For instance, a Llama 2 70B model trained with 2T tokens took 1720K GPU hours to train. To accelerate the training process, it is a popular practice in LLM pre-training [3, 37] to utilize exceptionally large batch sizes, thereby ensuring maximal GPU utilization. The usage of large batch sizes requires scaling the learning rates proportional to it's batchsize [9], [18] for SGD or proportional to the square root of it's batch size for adaptive gradient methods [24]. Overall, high learning rates are preferred when utilizing large batch sizes.

In this paper, our goal is to improve the test generalization (log perplexity) of LLM pre-training while reducing the number of training steps, all without increasing the compute budget. To achieve this, we first demonstrate that: (a) models trained with higher learning rates exhibit greater improvements when averaged along the training trajectory, and (b) averaging distant model weights from a single training trajectory further amplifies these gains. We integrate these two insights to adapt LAWA

[*]These authors contributed equally to this work. Correspondence to sanyal.sunny@utexas.edu

Workshop on Advancing Neural Network Training at 37th Conference on Neural Information Processing Systems (WANT@NeurIPS 2023).



(a) (b) (c)

Figure 1: Across all model sizes, LAWA achieves faster convergence and generalizes better in comparison to original pretraining run and other baseline averaging schemes. Validation loss on OpenWebText with 70K training steps; (a) GPT2-small (125M) with Original is 2.963, EMA is 2.949, SWA is 2.952 and LAWA (ours-best) is 2.917, (b) GPT2-medium (355M) with Original is 2.855, EMA is 2.845, SWA is 2.837 and LAWA (ours-best) is 2.819, and (c) GPT2-large (770M) with Original is 2.977, EMA is 2.968, SWA is 2.961 and LAWA (ours-best) is 2.908.

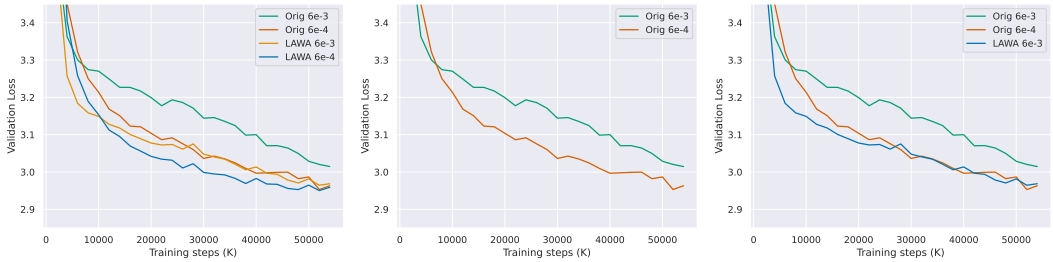
(LAtest Weight Averaging) [14]—a technique that performs checkpoint averaging throughout a training trajectory using a sliding window—for pre-training LLMs.

We evaluate our methodology by pre-training nanoGPT-2 models of various scales, specifically 125M (small), 355M (medium), and 770M (large), using the OpenWebText dataset, which comprises 9B tokens. The experiments with nanoGPT-2 are conducted in a controlled environment to gain a deeper understanding of our training recipe. Furthermore, we extend our evaluation to publicly available Pythia LLMs [3], which include model sizes of 1B, 2.8B, 6.9B, and 12B, trained using 207B tokens. Our experiments with Pythia LLMs aim to demonstrate the impact of our work on real-world LLMs.

Main Contributions. In summary, our findings are as follows,

1. We empirically show that models trained with high learning rate (LR) show pronounced gains over original training on performing checkpoint averaging very early on during training (Figure 2). This gain further amplifies when we sample distant checkpoints in the training run (Figure 1). We provide a intuitive explanation of this phenomenon in Section 2.
2. We observe that the training trajectory of LAWA closely mimics that of a model being trained with a low LR. The primary advantage of LAWA is that it allows LLMs to be trained with high LR without compromising generalization (Section 4).
3. We show that LAWA improves test generalization with fewer training steps compared to original training starting very early on during training; for both nanoGPT-2 and Pythia LLMs (Figure 1 and Figures 5-8). LAWA also improves zero-shot performance for nanoGPT-2 and Pythia LLMs, as shown in Table 2, 3.
4. We compare our recipe with conventional training and popular baselines such as Exponential Moving Average - EMA [35] and Stochastic Weight Averaging - SWA [12]. These baselines were not originally proposed or evaluated for LLM pre-training, but we adapt them to set meaningful baselines. Our training recipe outperforms conventional pre-training, EMA, and SWA.
5. Additionally, we perform a preliminary investigation of early weight averaging for a diffusion model for image generation (specifically, a 422M sized UNet model trained with the standard DDPM objective [10]). We observe thematically similar improvements (evaluated by the FID metric) as shown in Figure 9.

Paper outline. The structure of the remainder of this paper is as follows: Initially, we introduce the problem with a simple example and explain the intuition behind our approach in Section 2. This is followed by a detailed description of our experimental setup in Section 3, and the presentation of our main findings in Section 4. Subsequently, we compare our work with previous studies in Section 5. The paper concludes with a summary of our key findings and suggests possible avenues for future research.



(a) (b) (c)

Figure 2: We compare two independently trained nanoGPT-2 (125M) models with LR $= [6 \times 10^{-3}, 6 \times 10^{-4}]$ on OpenWebText data. (a) Pre-training curve with and without LAWA. LLMs trained with higher LR observes higher gain due to LAWA. (b) The model trained with a high LR generalizes poorly compared to its counterpart trained with low/tuned LR. (c) The generalization gap caused by the high LR is effectively mitigated by LAWA.

2 Intuition and Method

2.1 Toy Setting

We explain the setting using a simple toy problem of minimizing a two-dimensional loss function, represented as $L(w_1, w_2)$, where w_1 and w_2 are the parameters of the model. In this scenario, there exists an optimal batch size, $\mathcal{B}_0$, and an optimal learning rate, η_0, that minimizes the loss function. Assuming, for reasons outlined in Section 1, we are compelled to use a batch size, $\mathcal{B}$, and a learning rate, η, such that both are significantly larger than their optimal counterparts, i.e., $\mathcal{B} \gg \mathcal{B}_0$ and $\eta \gg \eta_0$ respectively. Suppose also that the loss function $L(w_1, w_2)$ exhibits a much higher curvature along w_1, as compared to w_2. It is widely known that the updates from the AdamW optimizer are mostly uniform across all weight dimensions [23]. When $\eta \gg \eta_0$ the weight updates of w_2 will be accelerated, however this will cause oscillations along w_1 which in a long run adversely affects the convergence of w_1. A naive approach to mitigate this problem is to use a smaller LR or decay LR to 0, which might hinder progress in flatter regions. It is conceivable that a LLM might exhibit extremely heterogeneous curvatures exacerbating this issue during pre-training.

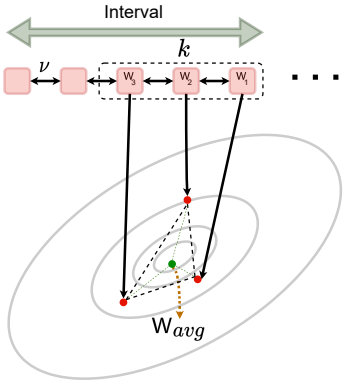


Figure 3: LAWA illustration: Given weights $\mathsf{W}_1, \mathsf{W}_2, ...\mathsf{W}_k$ from a high LR trajectory separated by k-stepsize (ν, Algorithm 1), LAWA computes W_{avg} at a given step.[2]

2.2 Intuition of Our Approach

Optimization Viewpoint. We propose performing checkpoint averaging of model weights relatively early during training with high learning rates (η). The rationale behind this step stems from the fact that checkpoint averaging serves as a surrogate to LR decay, as demonstrated by Sandler et al. [33]. However, this surrogate LR decay is decoupled from the weight update during optimization, as checkpoint averaging is conducted in a post-hoc manner. Employing this simple technique, we mitigate the oscillations in w_1 while swiftly traversing through w_2, achieving enhanced generalization in fewer training steps as illustrated in Figure 3.

[2]Note that W refers to the set of weights $\{w_1, w_2, ...\}$.

Diversity Viewpoint. The practice of averaging the weights of model checkpoints is broadly recognized as being functionally analogous to ensembling [12, 40]. In model ensembling literature it is well established that diverse models improve the performance of the ensemble [19]. Therefore, it is fair to assume that this principle also applies to model averaging as well. In our context, we define the diversity of a model at two distinct training steps, $\frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\left[y_i^{\mathsf{W}_1} \neq y_i^{\mathsf{W}_2}\right]$ which calculates the number of disagreements between the two checkpoints. This equation computes the number of samples from the same held-out set where the checkpoint W_1 disagrees with checkpoint W_2. A recent study by Athiwaratkun et al. has demonstrated that a higher LR can result in the generation of diverse model checkpoints. We observed (Figure 1) that this phenomenon can be further amplified by sampling far apart checkpoints in terms of training step. We combine both these insights to induce diversity in our checkpoints.

2.3 LAWA: LAtest Weight Averaging

We explain the Latest Weight Averaging (LAWA) algorithm below along with a python-style pseudo code (Algorithm 1). As shown in Figure 3, LAWA maintains a first in first out (FIFO) queue of periodically sampled checkpoints with a large number of intervening steps (ν) in between two succesive samples. We adapt LAWA for our setting with minor modifications. Specifically, we introduce k_stepsize (ν), and decoupled interval and k to effectively sample distant checkpoints in the training run. The original LAWA algorithm [14] assumes interval = k.

LAWA runs a moving window at a predetermined interval to collect k latest checkpoints on sequence of saved checkpoints $\boldsymbol{\theta}_t$. The LAWA derived checkpoints are computed as $\boldsymbol{\theta}_t^{\text{LAWA}} := \frac{1}{\mathsf{k}}\sum_{s=t-\mathsf{k}}^{t} \boldsymbol{\theta}_s$ where $\boldsymbol{\theta}_t$ the original checkpoints are sampled several training steps apart in the training process.

3 Experimental Setup

NanoGPT-2 Experiments. We conduct all our experiments utilizing autoregressive decoder-style Large Language Models (LLMs), specifically nanoGPT-2 and Pythia LLMs. We utilize three distinct sizes of nanoGPT-2 models: small (125M), medium (355M), and large (770M). We train nanoGPT-2 models from scratch using the OpenWebText dataset, which includes 9 billion training tokens and 4.4 million validation tokens. Throughout the experiments, we maintain a consistent sequence length of 1024 and a fixed batch size of 131K tokens per batch, the latter being the maximum batch size accommodated by our GPUs. The configurations for the model and pre-training were adapted from Sophia's [23] AdamW baseline, with adjustments made to the learning rate and batch size to align with our specific needs. Notably, we trained all the models with learning rates that were ten times higher and batch sizes that were twice as large compared to the configurations in [23], where the learning rate was tuned through a grid search. We compare LAWA with the original pre-training recipe, EMA [35], and SWA [12], which we adapt for LLMs. For EMA, we set the decay to 0.9 as per [14] and update the EMA model at every step, which is a standard practice. For SWA, we adhere to the original pre-training procedure until 75% completion, after which SWA training is initiated with a new SWA scheduler (cosine annealing). We compute the SWA uniform average every 10 steps.

Algorithm 1 LAWA: Pytorch-style pseudocode

```
def LAWA(ckpts, interval, k, k_stepsize):
    # ckpts: list of checkpoints
    # k: number of checkpoints to average
    # k_stepsize: distance in training steps
        between checkpoints
    averaged_ckpts = []
    for i in range(0, len(ckpts), interval):
        start_idx = i - k
        end_idx = i - 1
        # select last k checkpoints
        # with k_stepsize between each
            checkpoint.
        selected_ckpts = ckpts[end_idx-k+1:
            end_idx+1:k_stepsize]
        avg = average(selected_ckpts)
        averaged_ckpts.append(avg)
    return averaged_ckpts
```

Model Size	Layers	Hidden Size	Heads	Learning Rate	Equivalent Models
125M	12	12	768	6.0×10^{-3}	nanoGPT-2 (small)
335M	24	1024	16	3.0×10^{-3}	nanoGPT-2 (medium)
770 M	36	1280	20	2.0×10^{-3}	nanoGPT-2 (large)
1.0 B	16	2048	8	3.0×10^{-4}	—
2.8 B	32	2560	32	1.6×10^{-4}	GPT-Neo 2.7B, OPT-2.7B
6.9 B	32	4096	32	1.2×10^{-4}	OPT-6.7B
12 B	36	5120	40	1.2×10^{-4}	—

Table 1: Overview of models and their architecture from the nanoGPT-2 suite and Pythia suite [3] used in our experiments. The model nomenclature for Pythia LLMs is pythia-xx with model size. Models marked as "equivalent" have the same architecture and number of non-embedding parameters.

Pythia Experiments. The **Pythia LLMs** are publicly available in the Pythia suite [3]. We report results on Pythia-1B, Pythia-2.8B, Pythia-6.9B, and Pythia-12B; Table.1 summarizes the details of these models. For our experiments, we use intermediate model checkpoints; saved after every 1000 update steps. The models are trained by Biderman et al. [3] on the PILE dataset [8], a publicly available, curated collection of English text corpus of size 800GB. The original PILE dataset is curated using 5 different genres of data namely, academia, internet, prose, dialogue and miscellaneous. The PILE dataset contains 300 billion tokens prior to deduplication, and this number reduces to 207 billion tokens after the deduplication process. Our experiments use Pythia models trained with PILE-deduped dataset, as such models tend to memorize less [21]. The batch size for all the Pythia models was set at 2.09 million tokens and the learning rate was scaled following [44].

Evaluation We evaluate the language modelling performance of nanoGPT-2 models pretrained for 70K steps using log perplexity (perplexity and loss used interchangeably) on the held-out/val set. For nanoGPT-2 we use the moving window interval $=$ 1K, k = 5 and we sample checkpoints interval $=$ 200,1K apart for LAWA. Next we analyze the original training trajectories of Pythia LLMs and demonstrate the improvements achieved in test generalization using LAWA. We also present zero-shot evaluation results on Lambada OpenAI [27], SciQ [39], AI2 Reasoning Challenge-easy (ARC-e) [6], and Wikitext [25]. We evaluate 4 Pythia LLMs using the intermediate model checkpoints on a subset of the test and validation dataset following the methodology prescribed by [42]. For the purpose of evaluation we use the open source library `lm-evaluation harness`[3]. We select representative subsets from the diverse genres encompassed by the full PILE validation and test dataset. This subset comprises PILE-philosophy papers, PILE-bookcorpus2, and PILE-YouTube subtitles datasets.

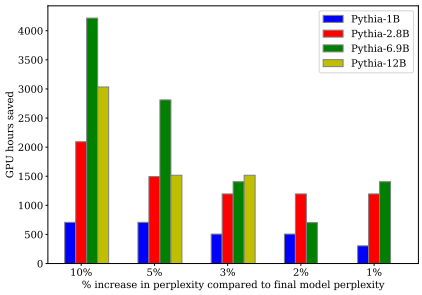


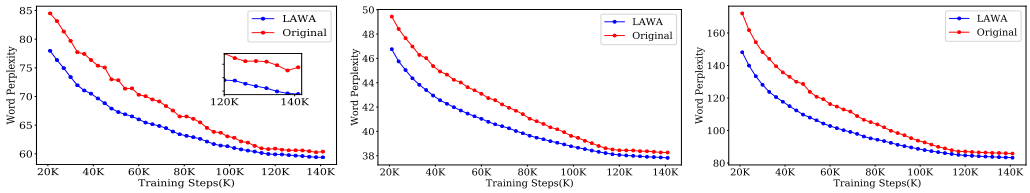
Figure 4: LAWA saves significant amount of GPU hours compared to original training. We compare the savings in GPU hours as a function of increase in final perplexity, i.e. perplexity achieved at 141K training step by the original checkpoint. This plot is created using a held out set from the training subset PILE-philosophy papers.

We conduct zero-shot evaluation on the Pythia LLMs. For zero-shot evaluation we provide a natural language description of the downstream task, along with a textual example. The models then generate responses that are either open-ended or discriminatively select a proposed answer. This evaluation setup serves as a robust academic benchmark, as it assesses Pythia models of various scales on reasonably large PILE subsets and downstream datasets, both in terms of test performance and zero-shot. For both the test generalization and zero-shot experiments, we evaluate model checkpoints starting 21K steps to 141K steps (recall that subsequent checkpoints are 1K steps apart). Moreover, we choose to slide the averaging window at 3K steps (i.e. interval $=$ 3K) and average last k intermediate

[3]https://github.com/EleutherAI/lm-evaluation-harness

Models	Steps	Lambada openai		ARC-easy		SciQ		BoolQ		Average	
		LAWA	Original	LAWA	Original	LAWA	Original	LAWA	Original	LAWA	Original
nanoGPT-2(125M)	50 K	33.77	27.65	44.65	44.07	74.10	71.60	52.08	48.26	51.15	47.89
	70 K	35.57	33.57	44.65	44.36	75.8	74.6	54.31	54.43	52.5	51.74
nanoGPT-2(335M)	50 K	35.88	30.95	44.74	44.28	75.8	72.9	50.76	50.49	49.72	51.72
	70 K	36.95	33.59	45.96	45.33	75.2	74.2	52.94	51.62	52.76	51.18
nanoGPT-2(770M)	50 K	31.28	28.08	42.21	40.99	65.2	65.1	58.17	51.1	49.21	46.31
	70 K	33.51	29.92	43.6	42.72	68.2	67.7	56.39	52.6	50.42	48.23

Table 2: The zero-shot performance of nanoGPT-2 LLMs on academic question answering and knowledge assessment downstream tasks is improved by LAWA. The checkpoints derived using our recipe require fewer steps to reach higher zero-shot performance than the checkpoints derived using original training.



(a) Philpapers (b) Bookcorpus2 (c) Youtube Subtitles

Figure 5: **LAWA speeds up convergence for Pythia-1B on subset of tasks from the original pretraining dataset i.e. PILE**. We present the original and the LAWA training trajectories for 3 different tasks from PILE namely philpapers, bookcorpus2 and youtube subtitles.

checkpoints as discussed in LAWA algorithm (Algorithm 1). Our selection of LAWA parameters such as k = 5 and start step = 21K are based on the experiments discussed in Section B.1.

4 Results

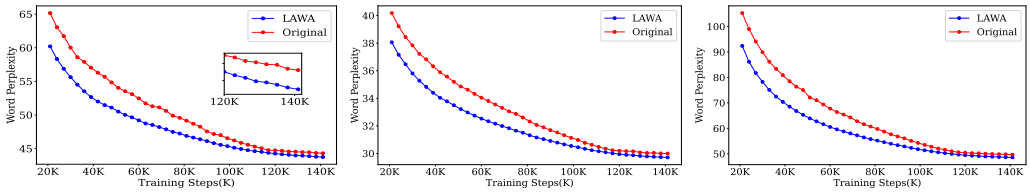
4.1 Exploring LLM pre-training with nanoGPT-2 at small scale

LLMs trained with higher LRs observe higher gains with LAWA. We ran controlled experiments to better understand the correlation between LR and gains due to checkpoint averaging. Initially we train nanoGPT-2 small with two different LRs $(6 \times 10^{-3}, 6 \times 10^{-4})$, keeping batch size and all relevant hyperparameters the same. 6×10^{-4} is the assumed optimal LR computed using a grid search reported in [23]. Subsequently, we pre-trained the same model using an LR of 6×10^{-3}, which is tenfold higher in magnitude than the former. As shown in Figure 2(a), models trained with higher LR observe higher gains compared to its counterpart trained with lower LR due to post hoc checkpoint averaging in LAWA. From Figure 2(b) we observe that the model trained with a higher LR converges faster but compromises on generalization, a phenomenon also observed by [16]. The gap in generalization is effectively mitigated by checkpoint averaging through LAWA, as shown in Figure 2(c). Interestingly, we note that the training trajectory of LAWA approximates that of a model trained with a lower LR. This is an important insight: checkpoint averaging acts as a surrogate for LR decay, thereby enabling the model to be trained with a higher LR. In practical LLM pre-training scenarios, where conducting a grid search is challenging due to the model's size, adopting our proposed training recipe could be advantageous. One might select a higher LR (that doesn't cause divergence) and train an LLM faster without compromising much generalization compared to conventional pre-training strategy.

LAWA improves test generalization in fewer training steps compared to original pre-training and relevant baselines. LAWA clearly outperforms the original pre-training run starting very early on during training, as shown in Figure 1. Since we employed a reasonably high LRs for all the nanoGPT-2 LLMs, we observe higher gains in the early-mid stages of pre-training, and the gains start diminishing towards the final stages due to the LR scheduler that continuously decays the weight throughout the training cycle. Additionally LAWA also outperforms important baselines such as EMA and SWA. LAWA clearly has an edge over EMA throughout all training phases. Our experiments

Models	Steps	Lambada openai		SciQ		WikiText(↓)		ARC-easy	
		LAWA	Original	LAWA	Original	LAWA	Original	LAWA	Original
Pythia-1B	48 K	50.32	46.85	84.6	84.3	18.34	19.33	54.50	54.25
	60 K	50.77	47.00	84.6	84.4	17.91	18.82	55.18	54.50
	105 K	57.84	56.39	86.1	86.3	16.83	17.12	56.77	56.27
	141 K	**58.99**	58.68	86.7	87.6	**16.50**	16.71	**58.33**	58.16
Pythia-2.8B	48 K	63.5	61.9	86.5	85.6	14.60	15.37	61.4	60.6
	60 K	64.3	63.8	86.8	86.3	14.17	14.87	62.7	62.1
	105 K	64.77	63.14	87.7	87.4	12.91	13.08	63.67	63.04
	141 K	**65.47**	65.26	**88.8**	88.6	**12.59**	12.70	**64.68**	64.56
Pythia-6.9B	48 K	65.8	62.3	88.7	88.0	13.55	14.25	63.7	62.9
	60 K	67.1	64.6	88.6	89.0	13.04	13.61	64.1	62.9
	105 K	**68.05**	67.78	**91.1**	91.2	11.92	12.07	**67.88**	67.08
	141 K	**69.08**	68.85	**92.0**	91.7	**11.61**	11.70	**68.13**	67.80
Pythia-12B	48 K	66.8	65.4	89.7	88.8	13.09	13.35	66.4	65.1
	60 K	67.8	66.2	90.3	90.5	12.54	12.76	67.3	60.1
	105 K	**71.06**	70.65	91.6	91.9	11.17	11.33	69.78	69.31
	141 K	**71.56**	71.00	**92.8**	92.3	**10.84**	10.91	70.58	70.74

Table 3: LAWA improves zero shot performance of Pythia LLMs on academic question answering and knowledge assessment downstream tasks starting very early on in the training. The checkpoints derived using LAWA requires less steps to reach higher zero-shot performance than the checkpoints derived using original training. We indicate the scores in bold font when the performance of LAWA surpasses the final score (at 141K steps) obtained using the original training or achieves comparable performance significantly earlier, specifically at 105K steps.



(a) Philpapers (b) BookCorpus2 (c) Youtube Subtitles

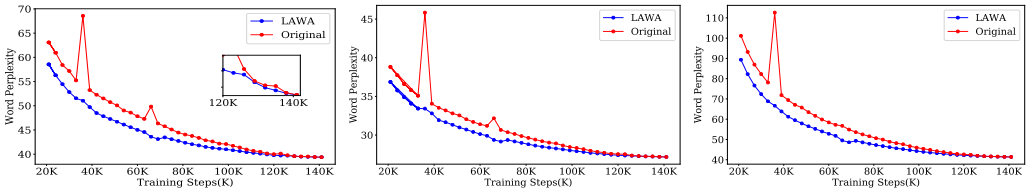
Figure 6: **LAWA speeds up convergence for Pythia-2.8B on subset of tasks from the original pretraining dataset i.e. PILE.**. We present the original and the LAWA training trajectories for 3 different tasks from PILE namely philpapers, bookcorpus2 and youtube subtitles.

also reveal that applying SWA during the early stages of training leads to divergence (Figure 11). Consequently, LAWA outshines SWA in both performance and ease of implementation.

The gains due to LAWA amplifies with far checkpoint averaging. As shown in Figure 1, LAWA with higher k_stepsize (ν) performs better particularly for larger models. Intuitively, we believe that the diversity between nearby checkpoints might be very low given that larger models learn faster [22]. Hence, one needs to sample more distant checkpoints for larger models. This observation is also consistent with billion-parameter Pythia LLMs, as shown in Figure 10.

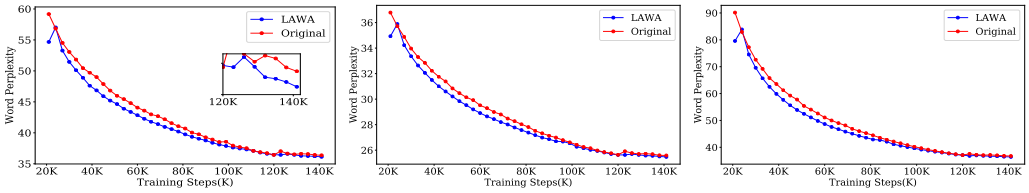
4.2 Scaling to Billion parameter Pythia LLMs

Figures 5 and 6 illustrate that the checkpoints derived using LAWA demonstrate better test generalization than the checkpoints saved during original training for the Pythia-1B and Pythia-2.8B models i.e. (moderate size LLMs). In Figures 7 and 8, we observe significant improvements in test generalization during early-mid training regime and minor improvements towards the end for both Pythia-6.9B and Pythia-12B models. All the LAWA LLMs achieve lower perplexity with lesser training steps compared to the original training trajectory, thereby saving significant amount of GPU hours (Figure 4), subsequent training costs and ingested training data. The savings are computed based on Table 4



(a) Philpapers (b) BookCorpus2 (c) Youtube Subtitles

Figure 7: **LAWA speeds up convergence for Pythia-6.9B on subset of tasks from the original pretraining dataset i.e. PILE**. We present the original and the LAWA training trajectories for 3 different tasks from PILE namely philpapers, bookcorpus2 and youtube subtitles.



(a) Philpapers (b) BookCorpus2 (c) Youtube Subtitles

Figure 8: **LAWA speeds up convergence for Pythia-12B on subset of tasks from the original pretraining dataset i.e. PILE**. We present the original and the LAWA training trajectories for 3 different tasks from PILE namely philpapers, bookcorpus2 and youtube subtitles.

in the appendix. Additionally, LAWA proves beneficial in situations where training similar models from scratch is necessary but can only be conducted over a limited number of training steps due to strict compute budgets.

To analyze the phenomenon wherein moderate size LLMs exibit higher gains compared to their larger counterparts in test performance across both early-mid and final training trajectories, we delve into the Pythia suite's training methodologies. The authors of Pythia suite [3] report that they have used an exceptionally large batch size (2M tokens) and learning rate for Pythia-1B and Pythia-2.8B models to expedite the convergence. For the larger Pythia models, such as Pythia-6.9B and Pythia-12B, learning rates are reduced to 1.2×10^{-4} while maintaining the batch sizes to 2M tokens, in line with prior work. Overall we observe thematically similar trends with the nanoGPT-2 experiments.

Recent work [15, 44, 37, 7, 41] report **loss spikes**– brief degradations in the performance along a training trajectory when scaling up the model size, batch size, and learning rate. In our evaluations, we observe two perplexity spikes (Figure 7). Interestingly, we find that LAWA mitigates the spikes during evaluation quite effectively. This can be intuitively explained by the **smoothing effect** – average out the outliers to fit the trend – resulting from averaging checkpoint weights over a range of steps that are far apart. Note that since we have sampled the checkpoints at an interval of 3K for LAWA, we may have inadvertently overlooked some checkpoints exhibiting loss spikes in models other than Pythia-6.9B. 7

4.3 Improved Zero-shot performance

LAWA enhances the zero-shot performance of both nanoGPT-2 and Pythia LLMs (Table 2). In the nanoGPT-2 models, we assess zero-shot performance at 50K and 70K steps, revealing that the checkpoints generated using LAWA at 50K consistently outperform original checkpoints—derived from conventional at 50K training steps (always) and also at 70K training (in the majority of instances).

For Pythia LLMs LAWA improves the zero-shot performance in several ways. First, we observe that zero-shot performance of early-mid checkpoints (48K, 60K) achieves higher performance almost consistently, regardless of the scale as shown in Table 3. For instance, the LAWA Pythia-1B checkpoint evaluated at 24K steps on the Lambada OpenAI task achieves higher accuracy than the original checkpoint evaluated at the 48K step. We note that the checkpoints derived using LAWA also exhibit improvements in the later stages of training (105K,141K) on the majority of tasks, highlighted by the bolded numbers in Table 3. Moreover, we consistently witness gains until 105K steps across all

models, which constitutes approximately 75% of the total training steps. Therefore our recipe proves to be beneficial in a compute optimal LLM training scenario where early stopping is employed at 75% of total training. Additionally, we find that our LAWA derived checkpoints of Pythia-6.9B reach the final accuracy/perplexity on multiple downstream tasks considerably earlier, specifically at the 105K step mark. Intuitively, we know that higher zero-shot performance on various different downstream tasks is associated with low perplexity in language modelling during training, a correlation that is mathematically substantiated [34]. Therefore, all the observations we made in Section 4.1 naturally apply to zero-shot performance as well.

4.4 Diffusion models

We also experiment with image diffusion models to gauge the effectiveness of LAWA on generative models beyond language. The underlying model is a 422M parameter UNet [32, 10] trained with ϵ-prediction objective and standard cosine schedule [10] on ImageNet 128x128 dataset. The model was trained with the Adam [17] optimizer using a learning rate of 1e-4. Figure 9 shows the FID on the evaluation set for the baseline checkpoints and LAWA averaging over the baseline checkpoints. Note that the baseline checkpoints themselves are obtained using the Exponential Moving Average (EMA) with decay rate of 0.9999 over the training trajectory, following the standard practice in training diffusion models. It is noteworthy that LAWA checkpoint averaging improves the FID over the already EMA'ed checkpoints. We defer a more thorough empirical investigation of LAWA for the family of diffusion models to future work.

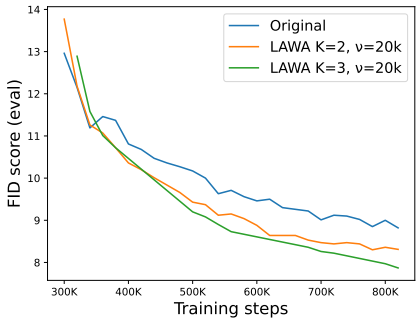


Figure 9: LAWA speeds up the convergence for Image diffusion model, measured in terms of FID on the evaluation set for ImageNet-128x128.

5 Related work

Weight Averaging (WA) has been studied and employed since the 1960s, predominantly in simple linear [20] and convex settings [28, 26]. Recent WA approaches in deep learning can be broadly classified into two categories; First, approaches that simultaneously trains multiple models with different initialization and hyper-parameters [40, 30, 13, 29] to later average them for better generalization. Second, approaches that focus on improving generalization of the final model or models close to convergence [36, 12, 2, 43, 4]. Stochastic WA (SWA) [12] employs a similar technique of averaging checkpoints along training trajectories but only works in the later stages of training (i.e. post 75% of the training run) with a new LR scheduler. This unsual halting and restarting the training with SWA with a new schedular limits its integration. We also show that SWA when applied early on during training diverges (Section B.2). Our recipe focuses on getting early gains through early averaging and can be generally applied to a wide range of training regimes. We expand the key differences of our work with SWA in Appendix A.

6 Conclusion and Future Work

In this paper we investigated a LLM pre-training setting where the LR is significantly higher than what is conventionally used. This scenario is particularly practical as LLMs are often trained using numerous GPUs in parallel, necessitating higher batch sizes for optimal GPU utilization. Here we introduce early weight averaging throughout the training trajectory utilizing LAWA. Our findings indicate that this strategy enables LLMs to generalize more effectively in fewer steps compared to the original pre-training scheme, and key baselines as demonstrated using nanoGPT-2 and Pythia models. Looking forward we see several extensions of our work in the realm of federated fine-tuning and continual training of intermediate checkpoints.

References

[1] Ben Athiwaratkun, Marc Finzi, Pavel Izmailov, and A. Wilson. There are many consistent explanations of unlabeled data: Why you should average. *International Conference on Learning Representations*, 2018.

[2] Ben Athiwaratkun, Marc Finzi, Pavel Izmailov, and Andrew Gordon Wilson. Improving consistency-based semi-supervised learning with weight averaging. *arXiv preprint arXiv:1806.05594*, 2(9):11, 2018.

[3] Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling. *arXiv preprint arXiv: Arxiv-2304.01373*, 2023.

[4] Junbum Cha, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park. Swad: Domain generalization by seeking flat minima. *Advances in Neural Information Processing Systems*, 34:22405–22418, 2021.

[5] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *GOOGLE*, 2022.

[6] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457, 2018.

[7] Nolan Dey, Gurpreet Gosal, Zhiming, Chen, Hemant Khachane, William Marshall, Ribhu Pathria, Marvin Tom, and Joel Hestness. Cerebras-gpt: Open compute-optimal language models trained on the cerebras wafer-scale cluster. *arXiv preprint arXiv: Arxiv-2304.03208*, 2023.

[8] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv: Arxiv-2101.00027*, 2020.

[9] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv: Arxiv-1706.02677*, 2017.

[10] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[11] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[12] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In Amir Globerson and Ricardo Silva (eds.), *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pp. 876–885. AUAI Press, 2018. URL `http://auai.org/uai2018/proceedings/papers/313.pdf`.

[13] Alexia Jolicoeur-Martineau, Emy Gervais, Kilian Fatras, Yan Zhang, and Simon Lacoste-Julien. Population parameter averaging (papa). *arXiv preprint arXiv: Arxiv-2304.03094*, 2023.

[14] Jean Kaddour. Stop wasting my time! saving days of imagenet and bert training with latest weight averaging. *arXiv preprint arXiv:2209.14981*, 2022.

[15] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[16] Simran Kaur, Jérémy E. Cohen, and Zachary Chase Lipton. On the maximum hessian eigenvalue and generalization. *ICBINB*, 2022. doi: 10.48550/arXiv.2206.10654.

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[18] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv: 1404.5997*, 2014.

[19] Balaji Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *NIPS*, 2016.

[20] Chandrashekar Lakshminarayanan and Csaba Szepesvari. Linear stochastic approximation: How far does constant step-size and iterate averaging go? In *International Conference on Artificial Intelligence and Statistics*, pp. 1347–1355. PMLR, 2018.

[21] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. *arXiv preprint arXiv: Arxiv-2107.06499*, 2021.

[22] Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, K. Keutzer, D. Klein, and Joseph Gonzalez. Train large, then compress: Rethinking model size for efficient training and inference of transformers. *International Conference On Machine Learning*, 2020.

[23] Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. *arXiv preprint arXiv: 2305.14342*, 2023.

[24] Sadhika Malladi, Kaifeng Lyu, A. Panigrahi, and Sanjeev Arora. On the sdes and scaling rules for adaptive gradient algorithms. *Neural Information Processing Systems*, 2022. doi: 10.48550/arXiv.2205.10287.

[25] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016. URL `https://arxiv.org/abs/1609.07843`.

[26] Gergely Neu and Lorenzo Rosasco. Iterate averaging as regularization for stochastic gradient descent. In *Conference On Learning Theory*, pp. 3222–3242. PMLR, 2018.

[27] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Q. N. Pham, R. Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and R. Fernández. The lambada dataset: Word prediction requiring a broad discourse context. *Annual Meeting Of The Association For Computational Linguistics*, 2016. doi: 10.18653/v1/P16-1144.

[28] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

[29] Alexandre Rame, Kartik Ahuja, Jianyu Zhang, Matthieu Cord, Leon Bottou, and David Lopez-Paz. Model ratatouille: Recycling diverse models for out-of-distribution generalization.

[30] Alexandre Rame, Matthieu Kirchmeyer, Thibaud Rahier, Alain Rakotomamonjy, patrick gallinari, and Matthieu Cord. Diverse weight averaging for out-of-distribution generalization. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=tq_J_MqB3UB`.

[31] Xiaozhe Ren, Pingyi Zhou, Xinfan Meng, Xinjing Huang, Yadao Wang, Weichao Wang, Pengfei Li, Xiaoda Zhang, A. V. Podolskiy, G. Arshinov, A. Bout, Irina Piontkovskaya, Jiansheng Wei, Xin Jiang, Teng Su, Qun Liu, and Jun Yao. Pangu-Σ: Towards trillion parameter language model with sparse heterogeneous computing. *ARXIV.ORG*, 2023. doi: 10.48550/arXiv.2303.10845.

[32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241. Springer, 2015.

[33] Mark Sandler, Andrey Zhmoginov, Max Vladymyrov, and Nolan Miller. Training trajectories, mini-batch losses and the curious role of the learning rate. *arXiv preprint arXiv:2301.02312*, 2023.

[34] Nikunj Saunshi, Sadhika Malladi, and Sanjeev Arora. A mathematical exploration of why language models help solve downstream tasks. *International Conference On Learning Representations*, 2020.

[35] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *Computer Vision And Pattern Recognition*, 2015. doi: 10.1109/CVPR.2016.308.

[36] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30, 2017.

[37] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *ARXIV*, 2023.

[38] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv: 2307.09288*, 2023.

[39] Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. *ArXiv*, abs/1707.06209, 2017.

[40] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, pp. 23965–23998. PMLR, 2022.

[41] Mitchell Wortsman, Tim Dettmers, Luke Zettlemoyer, Ari Morcos, Ali Farhadi, and Ludwig Schmidt. Stable and low-precision training for large-scale vision-language models. *arXiv preprint arXiv: Arxiv-2304.13013*, 2023.

[42] M. Xia, Mikel Artetxe, Chunting Zhou, Xi Victoria Lin, Ramakanth Pasunuru, Danqi Chen, Luke Zettlemoyer, and V. Stoyanov. Training trajectories of language models across scales. *ARXIV.ORG*, 2022. doi: 10.48550/arXiv.2212.09803.

[43] Guandao Yang, Tianyi Zhang, Polina Kirichenko, Junwen Bai, Andrew Gordon Wilson, and Chris De Sa. Swalp: Stochastic weight averaging in low precision training. In *International Conference on Machine Learning*, pp. 7015–7024. PMLR, 2019.

[44] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models. *ARXIV.ORG*, 2022.

Supplementary Materials: Appendix

Contents

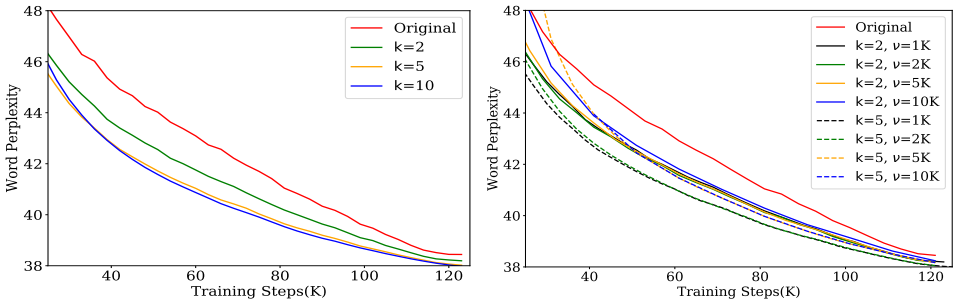
- A: Additional Related Work.
- B: Supplementary Experiments and Results.
- C: Compute Details.

A Additional Related Work

This section serves as a supplement to the main related work detailed in Section 5. Here we further elaborate on a prior work-SWA, and highlight the key differences of LAWA in the backdrop of training LLMs. SWA [12] is a checkpoint averaging scheme similar to LAWA but is extensively used in vision settings. In this paper we adapt SWA (also EMA) for LLM pre-training setting and presented it as a key baseline. The major differences of our recipe and SWA are listed below.

- In the SWA approach the checkpoints are averaged post 75% of regular training, i.e. close to convergence. This phase of training is termed as SWA training. Moreover the gain in generalization through SWA method is achieved at the end of training. In contrast, LAWA averages checkpoints very early on during training (post 10-15% of training) and achieves gains in test and zero-shot generalization with fewer training steps (less than 100% TB). SWA when applied early on during training performs poorly due to divergence of the training curve as shown in Figure 11.
- SWA modifies the learning rate scheduler mid-training and requires different learning rate schedulers for different architectures. Additionally, for models employing batch normalization, it necessitates a full pass over the entire dataset to update batch normalization statistics. LAWA is free from these constraints and is simpler to implement for large scale training.

B Supplementary Experiments and Results



(a) (b)

Figure 10: Ablations studying test performance as a function of (a) number of checkpoints to be averaged $\mathsf{k} = \{2, 5, 10\}$ at $\nu = 1\text{K}$, (b) distance of checkpoints to be averaged $\nu = \{1\text{K}, 2\text{K}, 5\text{K}, 10\text{K}\}$ at $\mathsf{k} = \{2, 5\}$. The checkpoints used for ablations is Pythia-1B.

B.1 Ablations

We perform ablations to better understand the interplay between the performance of number of checkpoints to be averaged $\boldsymbol{\theta}_t^{\text{LAWA}}$ with varying k and distance between averaged checkpoints ν in training. We study the Pythia-1B model with held-out subset of PILE-bookcorpus2. Additionally, we also provide training trajectory of Pythia-1B model on a subset of PILE datasets much earlier than 21K steps (refer Section B).

(a) (b) (c)

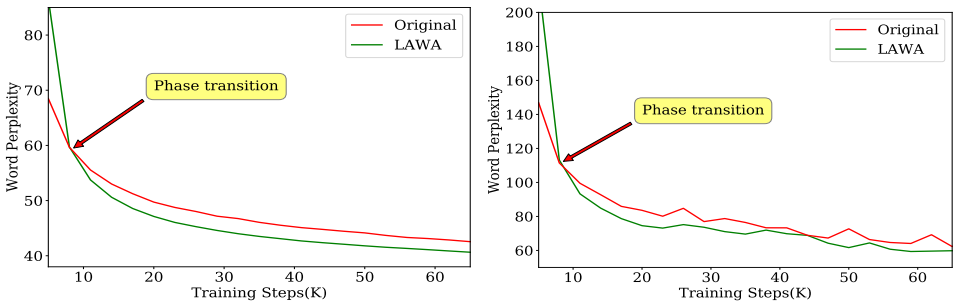
Figure 11: Across all model sizes (125M, 355M, 770M), we observe that the early version of stochastic weight averaging diverges, in contrast to the originally proposed late version.

Test Performance with varying k and fixed $\nu = 1\text{K}$. We investigate the impact of varying k on the model's test performance, while keeping ν constant at 1K. Our aim is to determine the optimal number of checkpoints to include in the average when selecting the latest k checkpoints following the LAWA approach. As outlined in Section 3, the Pythia checkpoints are saved at a frequency of 1K, so we have maintained ν at 1K for this analysis. From Figure 10(a), it is apparent that a smaller k could be detrimental, but performance remains fairly stable for reasonably large k values. Consequently, for our LLM experiments, we opted for $\mathsf{k} = 5$ since $\mathsf{k} = 10$ tends to occupy a substantial amount of disk space, especially for larger models such as Pythia-12B.

Test Performance with varying ν for $\mathsf{k} = \{2, 5\}$. Memory requirements remain a key bottleneck in saving model checkpoints throughout the training trajectory particularly for extremely large billion parameter models. Therefore, it is very interesting to know how far away checkpoints in a training trajectory can be averaged? We investigate this question with far checkpoint averaging at $\nu = \{1\text{K}, 2\text{K}, 5\text{K}, 10\text{K}\}$ training steps apart for $\mathsf{k} = \{2, 5\}$. As shown in Figure 10(b), we observe for both $\mathsf{k} = 2$ and $\mathsf{k} = 5$, we find that averaging more recent checkpoints (keeping ν small) works better than averaging stale weights (higher ν). For instance LAWA using $\mathsf{k} = 5$ and $\nu = \{1\text{K}, 2\text{K}\}$ consistently performed better than against LAWA with other parameters. Overall, we empirically find that a moderate number of checkpoints ($\mathsf{k} = 5$) saved in smaller frequencies ($\nu = 1\text{K}$) works best.

B.2 Early-SWA Experiments

Stochastic Weight Averaging [12] has previously shown gains for smaller models, particularly in late stages of training (typically $> 75\%$). We experimented with initializing SWA in early stages of training. As shown in Figure 11, we observe that it diverges quickly - showing that SWA does not provide any gains earlier in training.



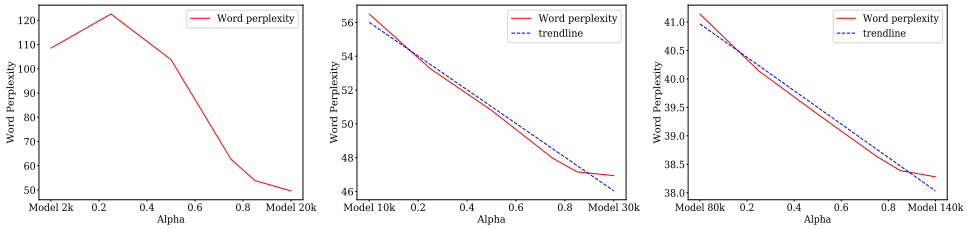
(a) BookCorpus2 (b) Enron Emails

Figure 12: **Early weight averaging doesn't work at the very beginning of the training but works reasonably early during the training process.** Here we compare original and LAWA early training trajectories for Pythia-1B model on 2 different tasks namely bookcorpus2 and enron emails using held out set.

Algorithm 2 Pytorch-style pseudocode of EMA/SWA

```
def EMA_SWA(ckpt, alpha, step_size, init_point):
    """
    ckpt: list of data points (could be checkpoints or any data series)
    alpha: smoothing factor, 0 < alpha <= 1. If alpha < 0, enables SWA.
    step_size: How often to calculate average. Typically set to 1 for EMA.
    init_point: Step after which to start averaging. Typically 0 for EMA.
    """
    # Initialize the series with the first data point
    series = [ckpt[0]]
    n_models = 1
    for i in range(1, len(ckpt)):
        # Calculate EMA/SWA
        if i%step_size==0 and i>init_point:
            if alpha < 0:
                factor = n_models/(n_models + 1)
                value = (1 - factor) * series[i-1] + factor * ckpt[i]
                n_models += 1
            else:
                value = (1 - alpha) * series[i-1] + alpha * ckpt[i]
            series.append(value)

    return series
```



(a) (b) (c)

Figure 13: **The model checkpoints attains linear mode connectivity quite early but not at the very beginning of the training process.** We plot word perplexity as a function of the model derived from the convex combination of 2 different checkpoints i.e. $\boldsymbol{\theta}_{\text{LMC}}$ at $\alpha = \{0, 0.2, 0.4, 0.6, 0.8, 1\}$. In (a) we see an error barrier that means model checkpoint at 2K and 20K are not linear mode connected, whereas both (b) and (c) shows the checkpoints under consideration are linear model connected.

B.3 Phase Transition and Linear Mode Connectivity

Averaging very initially i.e. before 8K steps during training may not always yield beneficial results (Figure 12). However, the technique does start showing efficacy fairly early in the training process. We highlight this phenomenon by presenting experimental results with the Pythia-1B model using a held-out set of PILE-bookcorpus2 and PILE-enron emails. We observe that LAWA trajectory undergoes a phase transition at the 8K training step. Beyond this transition, significant improvements in test performance can be seen. Such a phase transition may not occur for all Pythia LLMs. Following this phenomenon we presented our results starting 21K steps in Figures 5-8. We further examine this phenomenon through the lens of linear mode connectivity.

To better comprehend the linear model connectivity of checkpoints, we perform a convex combination of model checkpoints at different training stages. For instance, a model checkpoint at 2K and 20K can be combined in this manner: $\alpha \times \boldsymbol{\theta}_{2k} + (1-\alpha) \times \boldsymbol{\theta}_{20k}$. In Fig. 10, we plot word perplexity as a function of α using PILE-bookcorpus2. Here we observe that initially the model checkpoints are not linear mode connected. However, based on the evaluated checkpoints shown in Figure 13, we posit that the model checkpoint attains linear mode connectivity (LMC) quite early and maintains this property until the end of training.

C Amount of Compute

We compute the savings in GPU hours based on the Table. 6 of Pythia suite [3] as shown below.

Model Size	GPU Count	Total GPU hours required
1.0 B	64	4,830
2.8 B	64	14,240
6.9 B	128	33,500
12 B	256	72,300
Total		136,070

Table 4: Table from [3]. Model sizes in the Pythia suite, number of GPUs used during training, and the total number of GPU hours, calculated as (iteration time (s) $\times$ number of iterations $\times$ number of GPUs $\div$ 3600 s/hour). All GPUs are A100s with 40GB of memory.