SPDF: Sparse Pre-training and Dense Fine-tuning for Large Language Models

Abstract

The pre-training and fine-tuning paradigm has contributed to a number of breakthroughs in Natural Language Processing (NLP). Instead of directly training on a downstream task, language models are first pre-trained on large datasets with cross-domain knowledge (e.g., Pile, MassiveText, etc.) and then fine-tuned on task-specific data (e.g., natural language generation, text summarization, etc.). Scaling the model and dataset size has helped improve the performance of LLMs, but unfortunately, this also lead to highly prohibitive computational costs. Pre-training LLMs often require orders of magnitude more FLOPs than fine-tuning and the model capacity often remains the same between the two phases. To achieve training efficiency w.r.t training FLOPs, we propose to decouple the model capacity between the two phases and introduce Sparse Pre-training and Dense Fine-tuning (SPDF). In this work, we show the benefits of using unstructured weight sparsity to train only a subset of weights during pre-training (Sparse Pre-training) and then recover the representational capacity by allowing the zeroed weights to learn (Dense Fine-tuning). We demonstrate that we can induce up to 75% sparsity into a 1.3B parameter GPT-3 XL model resulting in a 2.5x reduction in pre-training FLOPs, without a significant loss in accuracy on the downstream tasks relative to the dense baseline. By rigorously evaluating multiple downstream tasks, we also establish a relationship between sparsity, task complexity and dataset size. Our work presents a promising direction to train large GPT models at a fraction of the training FLOPs using weight sparsity, while retaining the benefits of pre-trained textual representations for downstream tasks.

1 INTRODUCTION

Large language models (LLMs) have contributed to significant advances in natural language understanding (NLU) and natural language generation (NLG) due to the introduction of pre-training methods [Devlin et al., 2019, Radford and Narasimhan, 2018] on massive unannotated datasets (e.g., Pile [Gao et al., 2020], MassiveText [Rae et al., 2021], etc.). While scaling the model and dataset size has improved the quality of LLMs [Wei et al., 2022], it has also substantially increased the computational cost of pre-training. For instance, GPT-3 175B [Brown et al., 2020] is estimated to cost millions of dollars to train [Li, 2022]. Various techniques have been proposed to reduce the computational cost of training LLMs, including sparse attention [Dao et al., 2022b, Jaszczur et al., 2021], improved optimization techniques [Tang et al., 2021] and sequence-level curriculum learning [Li et al., 2022]. While these methods can help reduce computation time, weight sparsity is one promising technique orthogonal to the above methods. Here, a subset of model parameters are set to zero, reducing the FLOPs required during training.

Despite recent advances in sparse training [Hoefler et al., 2022], it has yet to be widely adopted by practitioners. First, it is difficult and expensive to find the optimal sparsity pattern [Frankle and Carbin, 2018, Ma et al., 2022] that can maintain the same level of accuracy as dense models. Second, unstructured sparsity can be difficult to accelerate on hardware architectures optimized for dense computation [Hooker, 2020]. In this work, we show how we can leverage weight sparsity to reduce training FLOPs, and then recover the lost representational capacity by shifting to dense weight matrices when fine-tuning on downstream tasks. In addition, while specialized software kernels have been developed to achieve inference acceleration with unstructured sparsity [Gale et al., 2020, NeuralMagic, 2021, Elsen et al., 2019, Ashby et al., 2019, Wang, 2021], recent work has shown that we can realize the gains of unstructured weight sparsity on specialized hardware (e.g.,

Submitted to the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023). **To be used for reviewing only**.

Cerebras CS-2 [Lie, 2022, 2021]) when training LLMs. For example, Lie [2021] shows the measured speedup for a matrix multiplication kernel w.r.t to the sparsity level on a single GPT-3 layer (see Appendix C for more details). Therefore, as unstructured sparse training techniques continue to become co-designed with the hardware, we can expect the FLOP reduction to translate into performance and wall-clock speedups.

Prior work on sparsifying LLMs focus on reducing training [Chen et al., 2022a, Dao et al., 2022a] or inference FLOPs [Chen et al., 2020], while matching standard dense training. Chen et al. [2022a] and Dao et al. [2022a] replace dense matrices with butterfly-based structured sparse weight matrices to reduce a model's size and accelerate pre-training on block-oriented hardware (e.g., GPUs [Krashinsky et al., 2020], TPUs [He et al., 2020]). Training with structured sparsity requires maintaining a regular sparse structure, which can reduce expressivity at higher sparsity levels. This is a well-known constraint observed when imposing structured sparsity in dense weight matrices [Zhou et al., 2021, Jiang et al., 2022]. The recent innovations in hardware architectures aim to facilitate the widespread use and adoption of unstructured weight sparsity, enabling the ability to achieve higher compression ratios while attaining practical speedups w.r.t wall-clock time. Our work focuses on pre-training with unstructured weight sparsity to reduce the FLOPs for training language models.

In the recent NLP literature, it is common to first pre-train, then fine-tune a language model. Fine-tuning pre-trained LLMs on downstream tasks leads to significantly better accuracy than the zero or few-shot settings [Alt et al., 2019, Ouyang et al., 2022]. The pre-training phase takes significantly longer compared to fine-tuning on a much smaller dataset to learn the domain-specific task. In the standard setup, the model size and capacity is generally kept the same between the two phases. We propose to break this assumption and show the benefits of modifying the model capacity between pre-training and fine-tuning with weight sparsity. First, we pre-train a sparse GPT model to reduce computational training FLOPs. Then, during the fine-tuning phase, we densify the GPT model, allowing the zeroed weights to learn and increase the modelling capacity to more accurately learn the downstream task.

While previous work has explored sparse-to-dense training to mitigate the difficulties of sparse-to-sparse training [Dao et al., 2022a] and improve the accuracy of dense models [Han et al., 2017], we perform fully sparse pre-training and only transition to dense weight matrices during fine-tuning. We refer to this framework as Sparse Pre-training and Dense Fine-tuning (SPDF) and demonstrate the ability of the sparse pre-trained model to transfer effectively to different downstream tasks (e.g., natural language generation and text summarization). The main contributions of our work are:

1. We propose Sparse Pre-training and Dense Fine-tuning (SPDF) as a new framework to reduce the FLOPs required during the pre-training phase, while maintaining accuracy on downstream tasks.
2. We demonstrate that we can train GPT-3 XL, at 75% sparsity, reducing the overall training FLOPS by 2.5x, while retaining the benefits of pre-trained textual representations in LLMs across a majority of tasks and evaluation metrics.
3. We establish a correlation between the optimal sparsity level during pre-training and the fine-tuning dataset size and task difficulty.

2 METHODOLOGY

This section presents our method to reduce pre-training FLOPs using unstructured weight sparsity. We first explain our intuition and hypotheses, followed by our methodology for the SPDF framework.

2.1 INTUITION AND HYPOTHESES

Prior works have shown that overparameterization of neural networks improves optimization and generalizability [Soltanolkotabi et al., 2019, Neyshabur et al., 2019, Allen-Zhu et al., 2019], but leads to an increase in compute cost [Brown et al., 2020]. Recent work on the Lottery Ticket Hypothesis Frankle and Carbin [2018] demonstrates that overparameterized dense networks contain sparse subnetworks which can be trained to the same accuracy as their dense counterparts, as long as one initializes the training with a good sparsity mask ("lottery ticket"). However, the process of searching for highly quality sparse subnetworks is computationally expensive [Frankle and Carbin, 2018, Ma et al., 2022]. Existing sparse training methods [Evci et al., 2020, Mocanu et al., 2018, Jayakumar et al., 2020] aim to discover the winning lottery ticket (i.e., optimal sparsity mask) in a single training run, but often fall short of the dense model's accuracy. In our framework, we mitigate the loss in representational power due to difficulties in sparse optimization [Evci et al., 2019], by transitioning to fully dense weight matrices during the fine-tuning phase. Even though we perform dense fine-tuning, the computational costs associated with fine-tuning are significantly lower than the cost of pre-training LLMs. Therefore, our method targets the phase which dominates the training FLOPs (i.e., pre-training). Based on recent theoretical findings and empirical studies on overparameterization and sparse neural networks, we lay out a set of hypotheses which we aim to study in our work through extensive experimental evaluation:

Hypothesis 1: High degrees of weight sparsity can be used during the pre-training phase of LLMs while preserving the downstream accuracy with dense fine-tuning.

Inducing sparsity during pre-training may cause a loss in representational power due to difficulties in sparse optimization and inability to discover optimal sparsity masks [Evci et al., 2019]. To mitigate these challenges, we aim to increase the representational power by allowing the zeroed weights to grow during fine-tuning (i.e., dense fine-tuning). Additionally, note the full capacity of the pre-trained model is often not required to generalize on the simpler downstream task, when using sparsity during pre-training [Ding et al., 2022]. Aghajanyan et al. [2021] investigate this phenomenon from a different angle and show pre-trained language models can learn a large set of NLP tasks with only a few parameters. This indicates that the full parameterization of the model is not needed to generalize well across downstream fine-tuning tasks. Hence, we can exploit weight sparsity during pre-training while retaining important textual representations despite the model's lower representational capacity.

Hypothesis 2: The performance of the sparse pre-trained model is correlated with the dataset size and degree of difficulty in the downstream task.

Liu et al. [2023] evaluate sparse networks on a diverse set of tasks with varying degrees of difficulty and show a strong correlation between a model's ability to be sparsified and the task difficulty. Hence, we hypothesize that models trained on complex tasks with high sparsity levels can suffer more from sparse training and experience a greater drop in performance compared to simpler tasks. We also note that small fine-tuning datasets may trigger over-fitting [Li and Zhang, 2021]. Therefore, we hypothesize that larger datasets can allow the sparse model to improve its generalization error on the task, and recover from training with high sparsity.

Hypothesis 3: As we increase the size of the language model, larger models become more amenable to higher levels of sparsity during pre-training.

Existing work [Liu et al., 2022] has shown that the quality of a network trained with random static sparsity (even at high sparsity levels) improves quickly to match its dense counterpart as the network grows wider and deeper. Also, larger models tend to have a smaller intrinsic dimension [Aghajanyan et al., 2021], which suggests that all parameters are not required to represent the average NLP task. Therefore, we expect the gap in downstream performance between the sparse pre-trained model and its dense counterpart to grow smaller as the size of the model increases.

2.2 SPARSE PRE-TRAINING AND DENSE FINE-TUNING

Our training procedure consists of two phases. The first phase involves pre-training a sparse language model on a large corpus of text, in an unsupervised manner. Here, we induce unstructured weight sparsity into the neural network to reduce the pre-training FLOPs. This is followed by a dense fine-tuning stage, where we expand the representational capacity of the model by allowing zeroed weights to learn, and adapt to a discriminative task with labeled data.

Unsupervised Dense Pre-training While our proposed framework is agnostic to the training objective, we focus on autoregressive language modeling as our motivating use case. In an autoregressive language model, the sequence generation process is modeled as a Markov chain, where the token to be predicted depends on all the previous tokens [Bengio et al., 2003]. Hence, the standard approach is to learn the probability distribution over sequences of tokens from an unsupervised pre-training corpus. Given an unsupervised pre-training corpus of tokens $\mathcal{U} = \{u_1, u_2, \ldots, u_{|\mathcal{U}|}\}$, where $|\mathcal{U}|$ is the total number of tokens. We aim to maximize the likelihood using the language modeling objective formulated as follows,

$$\mathcal{L}(\mathcal{U}) = \sum_{i=1}^{|\mathcal{U}|} \log(p(u_i|u_{i-k}, \ldots, u_{i-1}, \theta)),$$

where k is the size of the context window, and the conditional probability p is modeled using a neural network with parameters $\theta \in \mathbb{R}^N$. The parameters of the lth layer $\in L$ total layers are denoted as θ_l, along with the total number of parameters represented as N_l. We note that the network parameters θ are considered to be dense.

Unsupervised Sparse Pre-training To induce sparsity into the lth layer, we drop $s_l \in (0, 1)$ of its connections, where s_l to refer to the sparsity of layer l. This results in a total of $(1 - s_l)N_l$ parameters. Finally, the overall sparsity of a sparse subnetwork is defined as the ratio of zeroes to the total number of parameters in the original dense network, i.e., $S = \frac{\sum_l^L s_l N_l}{N}$. In our sparse training setup, we apply a binary sparsity mask $m \in \{0, 1\}^{|\theta|}$ on the initial parameters θ^0, such that its initialization is $m \odot \theta^0$. Here, the values 0 and 1 in the mask denote inactive (i.e., zero) and active (i.e., non-zero) weights, respectively. As a result, the sparse language model minimizes the following objective instead,

$$\mathcal{L}(\mathcal{U}) = \sum_{i=1}^{|\mathcal{U}|} \log(p(u_i|u_{i-k}, \ldots, u_{i-1}, m \odot \theta)). \quad (1)$$

In our work, we focus solely on static sparsity (i.e., m remains fixed throughout training) and the weights are randomly pruned at initialization. Specifically, we remove weights in each layer $l \in L$ randomly to the target sparsity s_l. Although several works have explored generating different layer-wise sparsity ratios at initialization (e.g., Erdös-Rényi-Kernel Evci et al. [2020], Ideal Gas Quota [Chen et al., 2022b], SNIP [Lee et al., 2019], GraSP [Wang et al., 2020], SynFlow [Tanaka et al., 2020], etc.), we focus on the

Figure 1: Sparse Pre-training and Dense Fine-tuning (SPDF) framework. In this framework, we sparsify a dense network and perform sparse pre-training followed by dense fine-tuning (green connections indicate newly activated weights). We use SPDF to pre-train large GPT models at a fraction of the training FLOPs using weight sparsity, and still retain the benefits on downstream tasks with dense fine-tuning.

simplest setup, which is uniform sparsity [Gale et al., 2019]. In uniform sparsity, each sparsified layer is pruned to the same target sparsity level.

For the language model, we use GPT [Radford et al., 2019, Brown et al., 2020] in our experiments, which is a variant of the Transformer [Vaswani et al., 2017]. We train the network with objective shown in Eq. 1 and AdamW [Loshchilov and Hutter, 2017] optimizer on an unsupervised pre-training dataset for a total of j iterations, arriving at parameters θ^j. Then, we adapt (i.e., fine-tune) the final pre-trained autoregressive language model $p_{m \odot \theta}$ to the supervised target task.

Dense Fine-tuning Following Hu et al. [2022] and Li and Liang [2021a], each downstream fine-tuning task is represented by a training dataset consisting of context-target pairs defined as: $\mathcal{Z} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_{|x|}, y_{|y|})\}$, where both x and y are sequences of tokens. For example, in structured data-to-text (e.g., E2E [Novikova et al., 2017]), x corresponds to a linearized data table and y a textual description; in text summarization (e.g., Curation Corpus [Curation, 2020]), x is the content of an article and y is its summary.

We initialize the start of dense fine-tuning to the final pre-trained parameters θ^j and during fine-tuning are updated to $\theta^j + \Delta\theta$. For each downstream task, we learn a different set of parameters with the task-specific parameter increment $\Delta\theta$ whose dimension $|\Delta\theta|$ equals $|\theta|$. Other works have explored more parameter efficient approaches to reduce the size of the task-specific parameters for the purpose of deploying fine-tuned models [Ben Zaken et al., 2022, Houlsby et al., 2019b, Hu et al., 2022]. However, in our approach, we focus on reducing the pre-training FLOPs with unstructured weight sparsity and perform dense fine-tuning to mitigate the challenges of sparse optimization by increasing representational power of the network. In the dense fine-tuning phase, we essentially remove the sparsity mask m to allow the inactive weights to grow. More specifically, we increase the representational capacity in θ^j by reviving all $\sum_l^L s_l \cdot N_l$ inactive weights, where all newly activated weights are initialized to 0. We evaluated other initializations like scaled normal distribution, but this did not lead to better results. Finally, the network is updated in a dense manner with the objective shown below,

$$\mathcal{L}(\mathcal{Z}) = \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|\boldsymbol{y}|} \log(p(y_t|(x_1, \ldots, x_{t-1}), \theta^j + \Delta\theta)).$$

The generic Sparse Pre-training and Dense Fine-Tuning (SPDF) framework, illustrated in Figure 1, consists of the following three steps:

1. Sparsify a given dense network to some target sparsity level, s_l, at each sparsifiable layer.
2. *Pre-train* the sparse model following the same training schedule as the original dense model.
3. *Fine-tune* the pre-trained sparse network on a given downstream task in a dense manner by allowing the zeroed weights to learn.

3 EXPERIMENTAL SETUP AND RESULTS

First, we provide details on our pre-training settings for GPT-2 Small (125M) and GPT-3 XL (1.3B), as well as our setups for the downstream fine-tuning tasks. Then, we compare sparse pre-training and sparse fine-tuning against sparse pre-training and dense fine-tuning to highlight the benefits of fine-tuning in a dense manner. Next, we validate our hypotheses (refer to Section 2.1) by evaluating SPDF across several tasks in natural language generation and text summarization. Following this, we compare the parameter subspaces between the pre-trained and fine-tuned models. Last, we present the advantages in training efficiency w.r.t total training FLOPs when using SPDF versus standard dense pre-training and dense fine-tuning.

Flop Optimal Pre-training via Chinchilla Scaling Law It was previously conventional in the literature to train all large language models (e.g., GPT-3 [Brown et al., 2020], Gopher [Rae et al., 2021], Jurassic [Lieber et al., 2021], etc.) on approximately 300B tokens of data. More recently, Chinchilla [Hoffmann et al., 2022] shows how parameters and data should be scaled equally as compute budget increases, which leads to significant gains in FLOP efficiency. In our pre-training setup, we follow Chinchilla's scaling law which suggests that we need approximately 20 tokens per parameter. Thus, for GPT-2 Small, a model with 125M parameters needs to be pre-trained on 2.5B tokens. Then, for GPT-3 XL, a model which has 1.3B parameters, needs to be pre-trained on 26B tokens. Unless stated otherwise, we pre-train our sparse GPT models from scratch on the Pile dataset [Gao et al., 2020] across sparsity levels $S \in \{50\%, 75\%\}$.

Fine-tuning on Downstream Tasks We studied dense fine-tuning on several downstream tasks in natural language generation and text summarization. We follow Hu et al.

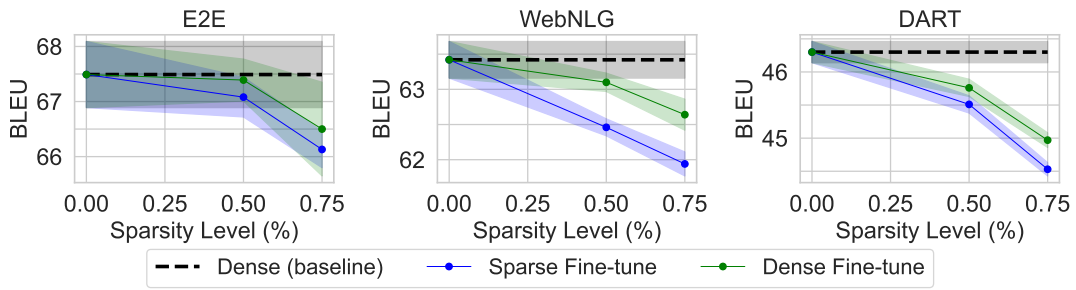

Figure 2: Comparison of sparse-to-dense vs sparse-to-sparse pre-training and fine-tuning with GPT-2 Small on E2E, WebNLG and DART. Across tasks dense fine-tuning noticeably outperforms sparse fine-tuning, especially at 75% sparsity.

[2022] in using the three standard natural language generation benchmark datasets (i.e., E2E [Novikova et al., 2017], WebNLG [Gardent et al., 2017] and DART [Nan et al., 2021]). In addition, we fine-tune on Curation Corpus [Curation, 2020] according to the details described in [Rae et al., 2021]. We fine-tune all parameters of the pre-trained GPT models and evaluate the final fine-tuning performance using the official evaluation scripts. More details on the hyperparameters can be found in Appendix A.

3.1 DETAILS ON THE FINE-TUNING DATASETS

Our work uses four fine-tuning datasets to investigate the efficacy of our SPDF framework. These datasets were chosen for studying the effect of sparse pre-training on different sizes and types of data, along with the varying degree of difficulty in the tasks.

End-2-End (E2E) NLG challenge dataset contains approximately 45k training, 4.6k validation, and 4.6k test examples with 8 distinct fields from the restaurant domain. The goal of the task is to generate natural language descriptions in the restaurant domain from meaning representations. We use the official evaluation script, which reports BLEU [Papineni et al., 2002], NIST [Belz and Reiter, 2006], METEOR [Lavie and Agarwal, 2007], ROUGE-L [Lin, 2004], and CIDEr [Vedantam et al., 2015].

WebNLG dataset consists of 18k training, 2.2k validation, and 2.4k test examples, where the input is a sequence of (subject, property, object) triples. In the training and validation splits, the input describes entities from 9 distinct DBpedia categories. The test set contains 15 different domains where 10 are present only in the training data. Here, the test data is split into two parts, where categories seen in the train set are in the first half, while the second half consists of 5 unseen categories. We use the official evaluation script, which reports BLEU, METEOR and TER [Snover et al., 2006]. The WebNLG dataset is the smallest of the three NLG tasks we evaluate on.

DART is an open domain DAta-Record-to-Text (i.e., table-to-text) dataset, with a similar input format to WebNLG. It consists of 62.6k training, 6.9k validation, and 12.5k test examples from several sources: WikiSQL [Zhong et al., 2017], WikiTableQuestions [Pasupat and Liang, 2015], Cleaned E2E[1], and WebNLG 2017[2] and applies some manual or automated conversion. We use the official evaluation script and report BLEU, METEOR and TER. The DART dataset is considered to be the most challenging NLG task out of the three we evaluate.

Curation Corpus is a recently introduced dataset comprised of 40,000 bespoke text summaries of finance articles for the task of text summarization. We follow the instructions in the Curation Corpus GitHub repository[3] to download approximately 40k article summary pairs. After filtering examples where either the article or the summary are empty, we are left with 39,911 examples. Following Marfurt and Henderson [2021], we split them into train/validation/test sets as 80/10/10 to arrive at split sizes of 31,929/3,991/3,991.

3.2 SPARSE FINE-TUNING VS DENSE FINE-TUNING

In this section, we first empirically establish the need for dense fine-tuning to help mitigate the difficulties of sparse-to-sparse training (i.e., sparse pre-training followed by sparse fine-tuning). In Figure 2, we compare dense fine-tuning against sparse fine-tuning on GPT-2 Small and show that across all three NLG tasks (i.e., E2E, WebNLG and DART), dense fine-tuning helps reduce the drop in BLEU score relative to the respective dense baselines. For example, the 75% sparse GPT-2 Small model on WebNLG observes a delta of -1.48 and -0.78 in the BLEU scores, when sparse fine-tuning and dense fine-tuning, respectively. This suggests that fully sparse end-to-end pre-training and fine-tuning can prevent the model from generalizing well on downstream tasks. However, we can mitigate the difficulties of poor generalizability due to sparse-only training by transitioning from sparse to dense matrices during the fine-

[1] https://github.com/tuetschek/e2e-cleaning

[2] https://gitlab.com/shimorina/webnlg-dataset/-/tree/master/webnlg_challenge_2017

[3] https://github.com/CurationCorp/curation-corpus

Table 1: Downstream accuracy of GPT-2 Small and GPT-3 XL across various tasks (i.e., E2E, WebNLG, DART and Curation Corpus) at sparsity levels 50% and 75% during pre-training. In the metric column, the direction of the arrow indicates better result (e.g., up indicates higher is better).

Model	Pre-Train Sparsity	E2E	WebNLG	DART	Curation Corpus
		BLEU↑			PPL↓
GPT-2 Small	0%	$67.49_{\pm 0.60}$	$63.42_{\pm 0.26}$	$46.30_{\pm 0.16}$	$13.38_{\pm 0.02}$
	50%	$67.39_{\pm 0.38}$	$63.10_{\pm 0.13}$	$45.74_{\pm 0.10}$	$15.09_{\pm 0.04}$
	75%	$66.50_{\pm 0.85}$	$62.64_{\pm 0.22}$	$44.97_{\pm 0.11}$	$17.14_{\pm 0.01}$
GPT-3 XL	0%	$68.10_{\pm 0.60}$	$63.62_{\pm 0.23}$	$47.71_{\pm 0.11}$	$8.28_{\pm 0.01}$
	50%	$67.98_{\pm 0.63}$	$63.47_{\pm 0.21}$	$47.10_{\pm 0.13}$	$9.21_{\pm 0.02}$
	75%	$67.66_{\pm 0.59}$	$63.06_{\pm 0.11}$	$46.96_{\pm 0.08}$	$11.03_{\pm 0.02}$

tuning phase. Although dense fine-tuning consumes more FLOPs compared to sparse fine-tuning, the overall fine-tuning FLOPs relative to pre-training, still remains insignificant (discussed further in Section 3.5).

3.3 SPDF ON NATURAL LANGUAGE GENERATION AND TEXT SUMMARIZATION

We perform an extended study on SPDF to further investigate its effectiveness on a diverse set of fine-tuning tasks, when using sparse pre-trained GPT-2 Small and GPT-3 XL models. In this section, we focus on natural language generation (i.e., E2E, WebNLG, and DART) and text summarization (i.e., Curation Corpus) tasks and refer to Table 1 for all the discussion points. We note that in Appendix B, we provide evaluation scores on all the metrics used to officially evaluate E2E, WebNLG and DART, respectively.

First, we validate Hypothesis 1 that high degrees of weight sparsity can be induced during pre-training. Our results indicate that in most settings, we can pre-train these GPT models with up to 75% sparsity without significant degradation across all NLG tasks. On the 75% sparse GPT-3 XL model, we observe deltas of -0.44, -0.56, and -0.75 in the BLEU scores for E2E, WebNLG and DART, respectively. In addition, the 50% sparse GPT-2 Small model observes deltas of -0.10, -0.32, and -0.56 in the BLEU scores for E2E, WebNLG and DART, respectively. Overall, our findings show that these GPT models can be pre-trained with 50%-75% sparsity without losing significant accuracy on these downstream tasks.

Second, we validate Hypothesis 2 that the performance of the sparse pre-trained model is correlated with the difficulty of the fine-tuning task. E2E, WebNLG and DART are NLG tasks which focus on mapping structured data content to a text describing this content. The Curation Corpus task focuses on summarizing the text description. While both tasks involve generating semantically coherent natural language, the summarization tasks are more difficult, since it require understanding of long sequences and compressing the sequence without loss of information. On the E2E, WebNLG and DART tasks, GPT-3 XL can be pre-trained up to 75% sparsity without a significant drop in BLEU score, as discussed previously. In contrast, on Curation Corpus, GPT-3 XL pre-trained at 75% sparsity loses 2.75 perplexity. In general, all data-to-text NLG tasks obtain a lower degradation compared to the more difficult Curation Corpus summarization task at higher levels of sparsity.

Finally, we validate Hypothesis 3 that as the size of the model increases, it becomes more amenable to higher sparsity levels. We analyze the relative drop in performance between the dense baseline and its sparse variants for GPT-2 Small and GPT-3 XL. This trend is clearly evident on the more difficult Curation Corpus task at 75% sparsity, where relative to the dense baseline, the larger GPT-3 XL model has a perplexity delta of +2.75 compared to a worse +3.76 delta observed in the smaller GPT-2 Small model. Similarly, on the DART task, the most challenging NLG task out of the three we evaluated, the delta in the BLEU score is -1.33 and -0.75 for GPT-2 Small and GPT-3 XL, respectively. These observations indicate that as the size of the language model increases, it suffers less on downstream task performance when training with high sparsity.

3.4 PRE-TRAINING VS FINE-TUNING PARAMETER SUBSPACES

In this section we analyze the parameter subspaces of the pre-trained model and its fine-tuned parameters across all layers to further understand (a) the behaviour of dense and spare pre-trained representations when fine-tuned, and (b) the effect of scaling the model size on parameter subspaces between the two phases. Inspired by Radiya-Dixit and Wang [2020], we measure the angular distance (i.e., cosine distance) between the pre-trained model parameters and its fine-tuned parameters on a given downstream task. Specifically, in all layers of the language model, we inspect the four weight matrices in the self-attention module; W_Q (query), W_K (key), W_V (value) and W_D (attention output projection) and the two in the MLP module; W_I (intermediate) and W_O (MLP output projection). In this analysis we focus on DART, the most difficult NLG task, and report the cosine distances for all modules in each layer of the dense and 75% sparse pre-trained GPT-2 Small and GPT-3 XL.

First, we aim to understand the behaviour of the parameter subspaces of the dense and sparse pre-trained models when fine-tuned. In GPT-2 Small (see Figure 3) and GPT-3 XL (see Figure 4), we observe that the dense pre-trained parameters and its fine-tuned parameters have very small cosine distances in almost all modules across each layer, whereas the 75% sparse model has larger cosine distances in certain modules (e.g., W_D and W_O) across all layers. Here, the dense model's fine-tuned parameters require less change in the parameter subspace relative to the pre-trained

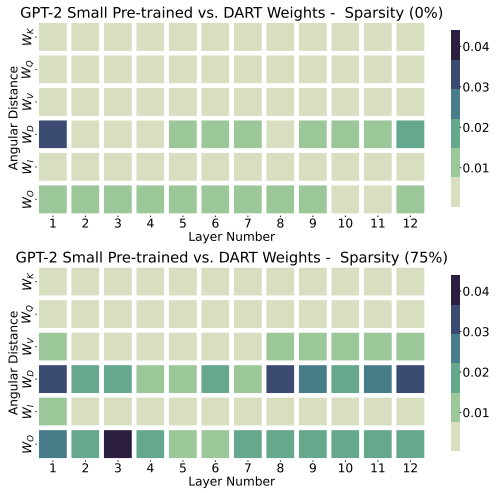

Figure 3: The angular distances in parameter subspaces between dense (top) and 75% sparse (bottom) pre-trained and fine-tuned DART weights for GPT-2 Small.

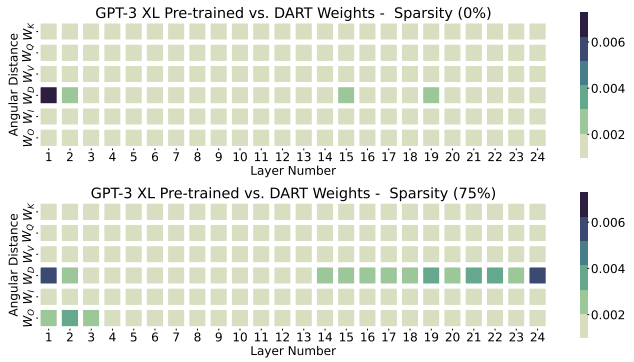

Figure 4: The angular distances in parameter subspaces between dense (top) and 75% sparse (bottom) pre-trained and fine-tuned DART weights for GPT-3 XL.

parameters, while the sparse model requires more movement in certain modules to learn the downstream task. This indicates that pre-trained models which learn high quality textual representations need less movement in the parameter subpsace to adapt to the downstream task. Although the sparse model has less representational capacity in its pre-trained parameters, it is capable of adapting certain modules through dense fine-tuning to learn the downstream task and stay competitive with the dense model's performance.

Next, we study the effect of model size and the parameter subspaces of the pre-trained and fine-tuned parameters. Evidently, in Figure 4, we observe that the dense pre-trained GPT-3 XL model has very small cosine distances across all modules in almost each layer, in comparison to GPT-2 Small. This suggests that as we increase the modeling capacity of the language model, only a few model parameter updates traverse a very short distance in the parameter space. This results in the pre-trained and fine-tuned weights being highly close across all modules in almost each layer. The larger language model is more capable of learning high quality representations, thus requires less movement in the fine-tuning parameter subspace. Even at 75% sparsity, the GPT-3 XL model requires significantly less change to the pre-trained parameters compared to GPT-2 Small in order to perform competitively well with the dense model. Given that many layers experience a very small change in the parameter subspace, we leave the investigation of freezing these modules during the fine-tuning phase for future work.

3.5 SPDF TRAINING EFFICIENCY

We compare the standard dense pre-training followed by dense fine-tuning framework to SPDF and highlight the potential FLOP reduction we can achieve. In Table 2, we report the total FLOPs (i.e., both the forward and backward propagations) needed for pre-training and dense fine-tuning GPT-2 Small and GPT-3 XL models on each of the tasks we evaluated. We note that in the GPT-2 Small model, the percentage of attention and vocab embeddings FLOPs account for approximately 13.3% and 27% of the total FLOPs, respectively. Therefore, at 75%, we achieve approximately 1.65x FLOP reduction over the dense baseline. However, in the larger GPT-3 XL, the percentage of attention and vocab embeddings FLOPs account for 13.3% and 6.8%, respectively. As a result, at the GPT-3 XL scale, SPDF provides almost 2.5x FLOP reduction over the dense baseline when pre-training with 75% sparsity. The trend of FLOP reduction relative to the dense baseline continues to increase with larger models, so the potential gains from sparse pre-training improves as model size grows. We also emphasize that the total fine-tuning FLOPs is a small fraction of the total pre-training FLOPs. In Appendix A.4, we provide details on how the total pre-training and fine-tuning FLOPs for GPT-2 Small and GPT-3 XL were calculated.

4 RELATED WORK

Zero-Shot vs. Fine-tuning Recent works have shown that large language models can achieve reasonable performance without any parameter updates [Brown et al., 2020, Chowdhery et al., 2022, Rae et al., 2021, Smith et al., 2022], often referred to as the zero-shot or few-shot setting. When no parameters are fine-tuned, framing a target task in terms of the pre-training objective enables zero-shot or few-shot learning to use a task-specific prompt and a few examples of a task [Brown et al., 2020]. However, while such few-shot learning is simple using such large models, there are alternative methods to obtain similar task accuracy using smaller models [Schick and Schütze, 2021]. In recent work, Cohen et al. [2022] demonstrate that while scaling the size of LaMDA can improve quality, combining scaling with fine-tuning can improve the model across all metrics including quality, safety and groundness. Solaiman and Den-

Table 2: Total FLOPs along with the associated theoretical speedup w.r.t the dense baseline (in brackets) for each of the evaluated fine-tuning tasks on GPT-2 Small and GPT-3 XL. The reported training FLOPs includes both pre-training and dense fine-tuning FLOPs. GPT-3 XL 75% SPDF provides $\approx$ 2.5x FLOP reduction over end-to-end dense training.

Model	Pre-Train Sparsity	Pre-training + Fine-tuning FLOPs ($\times 10^{18}$)			
		E2E	WebNLG	DART	Curation Corpus
GPT-2 Small	0%	2.48 (1.00x)	2.48 (1.00x)	2.45 (1.00x)	2.44 (1.00x)
	50%	1.84 (1.34x)	1.82 (1.35x)	1.84 (1.34x)	1.81 (1.35x)
	75%	1.52 (1.64x)	1.49 (1.65x)	1.52 (1.64x)	1.48 (1.65x)
GPT-3 XL	0%	236.62 (1.00x)	236.62 (1.00x)	236.33 (1.00x)	236.32 (1.00x)
	50%	142.40 (1.66x)	142.10 (1.66x)	142.01 (1.66x)	142.40 (1.66x)
	75%	95.29 (2.48x)	94.98 (2.49x)	95.29 (2.48x)	94.90 (2.49x)

nison [2021] show that fine-tuning also helps update language model behaviour to mitigate harmful outputs, which is highly critical for real-world deployment of LLMs (e.g., ChatGPT [OpenAI, 2022], Bard [Pichai, 2023], etc.). To achieve the best performance in practice, fine-tuning will continue to be the modus operandi when using pre-trained LLMs. Hence, our work focuses on pre-training and fine-tuning language models across a diverse set of tasks, including natural language generation and text summarization.

Efficient Fine-tuning While most large-scale models such as GPT [Brown et al., 2020, Smith et al., 2022] or T5 [Raffel et al., 2022] are trained dense, there are works [Houlsby et al., 2019a, Li and Liang, 2021b, Zaken et al., 2021, Hu et al., 2022] that explore using limited capacity (tuning a few layers or subset of parameters) in the pre-trained models to fine-tune on downstream tasks. These works are indicative that the total modeling capacity is unnecessary for fine-tuning on downstream tasks. Our work draws some inspiration from these works for exploiting the limited capacity of models for final tasks. However, we choose to reduce FLOPs for pre-training (significantly more training FLOPs than fine-tuning) and then add all the modeling capacity back during fine-tuning. This allows us to train large models efficiently and yet retain accuracies comparable to dense baselines. Although we do not explore efficient fine-tuning in our study, we leave the exploration of using alternative sparsity schedules [Zhu and Gupta, 2018, Liu et al., 2021], adapting a subset of parameters during fine-tuning [Ding et al., 2022] and imposing low-rank structures [Hu et al., 2022] for future work.

Weight Sparsification Techniques Many unstructured weight sparsification techniques have been proposed in the literature for training neural networks [Hoefler et al., 2022], which can be categorized as static sparsity and dynamic sparsity. Static sparsity methods have a fixed sparsity structure (i.e., sparsity mask) determined at initialization [Lee et al., 2019, Wang et al., 2020]. In contrast, dynamic sparse training (DST) methods iteratively prune (drop) and add (regrow) weights during training [Mocanu et al., 2018, Evci et al., 2020, Jayakumar et al., 2020, Huang et al., 2022] to find the best possible sparse subnetwork while retaining accuracy comparable to dense baselines. Although, dynamic sparse training methods can help achieve Pareto improvements in terms of number of training FLOPs to accuracy, we leave this for future work. Inspired by [Li et al., 2022], which shows that scaling the size of CNNs closes the gap between a randomly pruned sparse network and its dense counterpart, we focus our study on language models with static sparsity. While Dao et al. [2022a] demonstrate the benefits of sparse-to-dense training, they mainly apply it during pre-training and instead, focus their studies on dense-to-sparse fine-tuning similar to other efficient fine-tuning efforts. In our work, we show that sparse pre-training followed by dense fine-tuning on downstream tasks can be on par with the accuracy of a dense pre-trained model on many tasks, while significantly lowering overall training FLOPS.

5 CONCLUSION AND FUTURE WORK

In this work, we introduced Sparse Pre-training and Dense Fine-tuning (SPDF) to reduce the computational FLOPs of training GPT models using weight sparsity. To the best of our knowledge, this is the first time a large GPT model has been pre-trained with high sparsity (50%-75%) without significant loss in downstream task metrics. In our work, we only use simple static sparsity, which is arguably the most naïve way to induce sparsity in neural networks. As for future work, there are several natural directions for improving our results on even larger models, including dynamic sparsity methods, better optimization techniques for sparse training, and architectures amenable to sparse training. Moreover, to limit the computational cost of our study, we trained our GPT models following the Chinchilla scaling law. Although the Chinchilla pre-training schedule has been shown to be FLOP-optimal for dense models, we plan to investigate how well it transfers to sparse models. Our future work will also investigate sparse scaling outside the Chinchilla dense scaling laws. Regardless, we see the tremendous promise of unstructured weight sparsity to accelerate the training of LLMs, enabled by the recent advances in deep learning hardware accelerators.

References

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *ACL*, 2021.

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *ICML*, 2019.

Christoph Alt, Marc Hübner, and Leonhard Hennig. Fine-tuning pre-trained transformer language models to distantly supervised relation extraction. In *ACL*, 2019.

Mike Ashby, Christiaan Baaij, Peter Baldwin, Martijn Bastiaan, Oliver Bunting, Aiken Cairncross, Christopher Chalmers, Liz Corrigan, Sam Davis, Nathan van Doorn, Jon Fowler, Graham Hazel, Basile Henry, David Page, Jonny Shipton, and Shaun. Steenkamp. Exploiting unstructured sparsity on next-generation datacenter hardware. 2019.

Anja Belz and Ehud Reiter. Comparing automatic and human evaluation of NLG systems. In *ACL*, 2006.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *ACL*, 2022.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *JMLR*, 2003.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020.

Beidi Chen, Tri Dao, Kaizhao Liang, Jiaming Yang, Zhao Song, Atri Rudra, and Christopher Re. Pixelated butterfly: Simple and efficient sparse training for neural network models. In *ICLR*, 2022a.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The lottery ticket hypothesis for pre-trained bert networks. In *NeurIPS*, 2020.

Tianlong Chen, Zhenyu Zhang, pengjun wang, Santosh Balachandra, Haoyu Ma, Zehao Wang, and Zhangyang Wang. Sparsity winning twice: Better robust generalization from more efficient training. In *ICLR*, 2022b.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv*, 2022.

Aaron Daniel Cohen, Adam Roberts, Alejandra Molina, Alena Butryna, Alicia Jin, Apoorv Kulshreshtha, Ben Hutchinson, Ben Zevenbergen, Blaise Hilary Aguera-Arcas, Chung ching Chang, Claire Cui, Cosmo Du, Daniel De Freitas Adiwardana, Dehao Chen, Dmitry (Dima) Lepikhin, Ed H. Chi, Erin Hoffman-John, et al. Lamda: Language models for dialog applications. In *arXiv*. 2022.

Curation. Curation corpus base, 2020.

Tri Dao, Beidi Chen, Nimit S Sohoni, Arjun Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Ré. Monarch: Expressive structured matrices for efficient and accurate training. In *ICML*, 2022a.

Tri Dao, Daniel Y Fu, Stefano Ermon, Atri Rudra, and Christopher Re. Flashattention: Fast and memory-efficient exact attention with IO-awareness. In *NeurIPS*, 2022b.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv*, 2019.

Ning Ding, Yujia Qin, Guang Yang, Fu Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Haitao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juan Li, and Maosong Sun. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv*, 2022.

Erich Elsen, Marat Dukhan, Trevor Gale, and Karen Simonyan. Fast sparse convnets. *arXiv*, 2019.

Utku Evci, Fabian Pedregosa, Aidan N. Gomez, and Erich Elsen. The difficulty of training sparse neural networks. *arXiv*, 2019.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *ICML*, 2020.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*, 2018.

Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv*, 2019.

Trevor Gale, Matei Zaharia, Cliff Young, and Erich Elsen. Sparse gpu kernels for deep learning. In *SC*, 2020.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv*, 2020.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. The WebNLG challenge: Generating text from RDF data. In *INLG*, 2017.

Song Han, Jeff Pool, Sharan Narang, Huizi Mao, Enhao Gong, Shijian Tang, Erich Elsen, Peter Vajda, Manohar Paluri, John Tran, Bryan Catanzaro, and William J. Dally. DSD: Dense-sparse-dense training for deep neural networks. In *ICLR*, 2017.

Xin He, Subhankar Pal, Aporva Amarnath, Siying Feng, Dong-Hyeon Park, Austin Rovinski, Haojie Ye, Yuhan Chen, Ronald Dreslinski, and Trevor Mudge. Sparse-tpu: Adapting systolic arrays for sparse matrices. In *ACM International Conference on Supercomputing*, 2020.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. In *JMLR*, 2022.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack William Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In *NeurIPS*, 2022.

Sara Hooker. The hardware lottery. *arXiv*, 2020.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *ICML*, 2019a.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *ICML*, 2019b.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

Shaoyi Huang, Bowen Lei, Dongkuan Xu, Hongwu Peng, Yue Sun, Mimi Xie, and Caiwen Ding. Dynamic sparse training via more exploration. *arXiv*, 2022.

Sebastian Jaszczur, Aakanksha Chowdhery, Afroz Mohiuddin, Lukasz Kaiser, Wojciech Gajewski, Henryk Michalewski, and Jonni Kanerva. Sparse is enough in scaling transformers. In *NeurIPS*, 2021.

Siddhant Jayakumar, Razvan Pascanu, Jack Rae, Simon Osindero, and Erich Elsen. Top-kast: Top-k always sparse training. In *NeurIPS*, 2020.

Peng Jiang, Lihan Hu, and Shihui Song. Exposing and exploiting fine-grained block structures for fast and accurate sparse training. In *NeurIPS*, 2022.

Ronny Krashinsky, Olivier Giroux, Stephen Jones, Nick Stam, and Sridhar Ramaswamy. Nvidia ampere architecture in-depth, May 2020. URL `https://developer.nvidia.com/blog/nvidia-ampere-architecture-in-depth/`.

Alon Lavie and Abhaya Agarwal. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Second Workshop on Statistical Machine Translation*. ACL, 2007.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. SNIP: Single-Shot Network Pruning based on Connection Sensitivity. In *ICLR*, 2019.

Chuan Li. Openai's gpt-3 language model: A technical overview. *GPU Cloud, Workstations, Servers, Laptops for Deep Learning*, Aug 2022. URL `https://lambdalabs.com/blog/demystifying-gpt-3`.

Conglong Li, Minjia Zhang, and Yuxiong He. The stability-efficiency dilemma: Investigating sequence length warmup for training GPT models. In *NeurIPS*, 2022.

Dongyue Li and Hongyang Zhang. Improved regularization and robustness for fine-tuning in neural networks. In *NeurIPS*, 2021.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *ACL*, 2021a.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv*, 2021b.

Sean Lie. Thinking outside the die: Architecting the ml accelerator of the future, Nov 2021. URL `https://www.microarch.org/micro54/media/lie-keynote.pdf`.

Sean Lie. Cerebras architecture deep dive: First look inside the hw/sw co-design for deep learning. *Cerebras Systems Blog*, Dec 2022. URL `https://www.cerebras.net/blog/cerebras-architecture-deep-dive-first-look-inside-the-hw/sw-co-design-for-deep-learning`.

Opher Lieber, Or Sharir, Barak Lenz, and Yoav Shoham. Jurassic-1: Technical details and evaluation. *AI21 Labs*, 2021.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*. ACL, 2004.

Shiwei Liu, Tianlong Chen, Xiaohan Chen, Zahra Atashgahi, Lu Yin, Huanyu Kou, Li Shen, Mykola Pechenizkiy, Zhangyang Wang, and Decebal Constantin Mocanu. Sparse training via boosting pruning plasticity with neuroregeneration. In *NeurIPS*, 2021.

Shiwei Liu, Tianlong Chen, Xiaohan Chen, Li Shen, Decebal Constantin Mocanu, Zhangyang Wang, and Mykola Pechenizkiy. The unreasonable effectiveness of random pruning: Return of the most naive baseline for sparse training. In *ICLR*, 2022.

Shiwei Liu, Tianlong Chen, Zhenyu Zhang, Xuxi Chen, Tianjin Huang, Ajay Kumar Jaiswal, and Zhangyang Wang. Sparsity may cry: Let us fail (current) sparse neural networks together! In *ICLR*, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2017.

Xiaolong Ma, Minghai Qin, Fei Sun, Zejiang Hou, Kun Yuan, Yi Xu, Yanzhi Wang, Yen-Kuang Chen, Rong Jin, and Yuan Xie. Effective model sparsification by scheduled grow-and-prune methods. In *ICLR*, 2022.

Andreas Marfurt and James Henderson. Sentence-level planning for especially abstractive summarization. In *Third Workshop on New Frontiers in Summarization*, 2021.

Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H. Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, 2018.

Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zaidi, Mutethia Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, Yi Chern Tan, Xi Victoria Lin, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. DART: Open-domain structured data record to text generation. In *ACL*, 2021.

NeuralMagic. Deepsparse, 2021. URL `https://github.com/neuralmagic/deepsparse`.

Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. The role of overparametrization in generalization of neural networks. In *ICLR*, 2019.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. The E2E dataset: New challenges for end-to-end generation. In *SIGdial Meeting on Discourse and Dialogue*, 2017.

OpenAI. Chatgpt: Optimizing language models for dialogue, Nov 2022. URL `https://openai.com/blog/chatgpt/`.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *ACL*, 2002.

Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. In *ACL and IJCNN*, 2015.

Sundar Pichai. An important next step on our ai journey, Feb 2023. URL `https://blog.google/technology/ai/bard-google-ai-search-updates/`.

Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. *OpenAI Blog*, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.

Evani Radiya-Dixit and Xin Wang. How fine can fine-tuning be? learning efficient language models. In *AISTATS*, 2020.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv*, 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. In *JMLR*, 2022.

Timo Schick and Hinrich Schütze. It's not just size that matters: Small language models are also few-shot learners. In *ACL*, 2021.

Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv*, 2022.

Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *Association for Machine Translation*, 2006.

Irene Solaiman and Christy Dennison. Process for adapting language models to society (palms) with values-targeted datasets. In *NeurIPS*, 2021.

Mahdi Soltanolkotabi, Adel Javanmard, and Jason D. Lee. Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *IEEE Transactions on Information Theory*, 2019.

Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In *NeurIPS*, 2020.

Hanlin Tang, Shaoduo Gan, Ammar Ahmad Awan, Samyam Rajbhandari, Conglong Li, Xiangru Lian, Ji Liu, Ce Zhang, and Yuxiong He. 1-bit adam: Communication efficient large-scale training with adam's convergence speed. *arXiv*, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *NeurIPS*, 2017.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *CVPR*, 2015.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *ICLR*, 2020.

Ziheng Wang. Sparsednn: Fast sparse deep learning inference on cpus. *arXiv*, 2021.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *TMLR*, 2022. Survey Certification.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv*, 2021.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv*, 2017.

Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. Learning n:m fine-grained structured sparse neural networks from scratch. In *ICLR*, 2021.

Michael Zhu and Suyog Gupta. To prune, or not to prune: Exploring the efficacy of pruning for model compression. In *ICLR*, 2018.