SPARSE MOE WITH RANDOM ROUTING AS THE NEW DROPOUT: TRAINING BIGGER AND SELF-SCALABLE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Exploiting scale to revamp information absorption has recently become central to the success of deep learning and transformers have become *de facto* choice achieving numerous breakthrough performances on many real-world applications. Despite their enormous success, gigantic transformers suffer not only from exorbitant computational and memory footprints during training but also from severe collapse as evidenced by a high degree of parameter redundancy. Recently proposed Sparsely-activated Mixture-of-Experts (SMoEs) models have shown promise to mitigate the issue of training efficiency, yet they have some critical limitations. In particular, SMoEs models are prone to *redundant experts* due to representational collapse and *poor scalability during inference and downstream fine-tuning* primarily due to overfitting of the learned routing policy to the number of activated experts during training. As recent research efforts are predominantly focused on improving routing policies to encourage expert specializations, our work focuses on *exploring the overlooked scalability bottleneck of SMoEs*, and leveraging it as an auxiliary tool to effectively benefit scaling large-scale dense transformers. To this end, we propose a new plug-and-play training framework, **SMoE-Dropout** to enable scaling transformers to better accuracy in the full capacity setting without collapse. Specifically, SMoE-Dropout consists of a *randomly initialized and fixed* router network to activate experts and gradually increase their number as training progresses over time. SMoE-Dropout naturally provides a **"self-slimmable"** property offering consistent boosted performance for transformers with an increase in activated experts during inference and downstream fine-tuning, subjected to resource availability. Our extensive experiments across diverse transformer architectures on a variety of tasks validate superior performance and substantial computation savings, compared to densely trained baselines with equivalent parameter counts. More precisely, our trained BERT outperforms their densely trained counterpart with consistent improvements of $\{1.03\%, 0.78\%, 1.09\%\}$ on challenging reasoning tasks {`ASDiv-A`, `MAWPS`, `SVAMP`}, respectively. Codes and models will be publicly released.

1 INTRODUCTION

Scaling neural networks, historically with the blessing of modern hardware, have dramatically improved the state-of-the-art on a wide array of real-world machine learning applications and leaderboards, conforming to the empirical scaling laws (Kaplan et al., 2020), where the final model quality has been found to have a power-law relationship with the amount of data, model size, and compute time. Transformers (Vaswani et al., 2017), swiftly after their introduction have become *de facto* choice for many natural language processing (NLP) (Yang et al., 2019c; Liu et al., 2019b; Talmor et al., 2018; Jaiswal et al., 2021; Yang et al., 2019b; Wang et al., 2018; Ding et al., 2019; Chowdhery et al., 2022; Wei et al., 2022) and computer vision (Dosovitskiy et al., 2020; Han et al., 2020; Touvron et al., 2021; Mao et al., 2022; Zheng et al., 2021; Parmar et al., 2018) applications and now their parameter counts are typically measured in billions rather than millions. Unfortunately, this *exploitation of parameters actuates a roughly quadratic blow-up in training costs*, as both the model size and the number of training examples increase especially for dense advanced transformer-based models (*e.g.,* BERT (Devlin et al., 2018) and GPT (Brown et al., 2020)) and need thousands of GPU days for training. Additionally, these gigantic transformers suffer from **representation collapse** is-

sue during vanilla training, which is affirmed by a high degree of parameter redundancy (Guo et al., 2019; Ganesh et al., 2020; McCarley et al., 2019) and observed ineffective usage of the transformer expressiveness (Michel et al., 2019; Chen et al., 2022a).

Sparse Mixture-of-Experts (SMoEs), which perform *input-dependent conditional computing* provide an effective way to scale model capacity given a fixed computational cost, and facilitate training transformers with significantly high parameter counts at moderately increased cost, compared to their dense counterpart, thereby *improving the training efficiency*. For instance, with similar training FLOPS, Switch-Large (Fedus et al., 2021) (a kind of SMoE) is $35\times$ larger than a T5-Large dense model (Raffel et al., 2020). Despite their benefits in mitigating computational and energy footprints, SMoEs tend to have many critical limitations. **Firstly**, the current learning-based routing mechanisms in SMoEs tend to push hidden representations clustering around expert centroids (Chi et al., 2022), implying a trend toward **representation collapse**, which in turn lead to redundant experts, inferior expert specialization, thereby substandard performance (Mittal et al., 2022; Chen et al., 2022b). **Secondly**, SMoEs suffer from **poor scalability during inference and downstream fine-tuning** prominently due to *overfitting of the learned routing policy* to the number of activated experts during training. Adopting naive changes to mitigate the issue of the immutability of sparsely activated subnetworks, often lead to performance degradation. As recent research efforts for SMoEs are predominantly focused on improving routing policies to encourage expert specializations, we explore the overlooked scalability bottleneck of SMoEs to benefit transformers and ask: *Does there exist a principled, and neatly pluggable approach to modify SMoE training in pursuit to benefit scalability at inference and downstream fine-tuning of large-scale transformers, by dynamically adapting the number of activated experts subject to resource availability?*

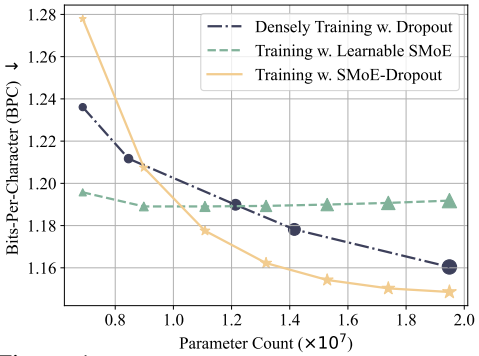

Figure 1: Bits-Per-Character (↓) on `enwik8`'s test-set with a 4-layer Transformer-XL. SMoE-Dropout demonstrates a "self-slimmable" property where inference performance is smoothly boosted along with the increase of activated parameters. Learnable SMoEs tend to overfit certain levels of network capacity. Note that only gray curve is produced by (5) different dense models.

To this end, this paper proposes a novel plug-and-play training framework, **SMoE-Dropout** to enable scaling transformers to better accuracy in the full capacity setting without collapse. More specifically, SMoE-Dropout consists of a randomly initialized and fixed router network to activate experts and gradually increase their number as training progresses over time. Our simple, yet highly effective strategy has a multi-fold win-win for trained transformers, specifically: ❶ obtaining a **"self-slimmable"** property during inference and downstream fine-tuning subject to resource availability, which delivers a once-for-all in-situ trade-off between efficiency and performance; ❷ mitigating *representational collapse* and effectively utilizing the full model capacity, where activating more experts produces superior performance (Figure 1 (blue)); ❸ eliminating the overhead of learning routing policies for SMoE. Note that SMoE-Dropout can be swiftly adapted for training any deep learning network (e.g. CNNs), given some splitting techniques (Zhang et al., 2021), but this work primarily focus on transformers considering their exploding computational footprints. Our innovative contributions can be summarized as:

- ⋆ We propose a new plug-and-play training framework, **SMoE-Dropout**, to enable scaling transformers in the full capacity setting without collapse. SMoE-Dropout facilitates the randomly and sparsely activated structure of network modules, playing an *implicit regularization role similar to dropout*. Our new framework leads to an enhanced generalization and reduced training costs (*e.g.*, up tp 37% running time savings) in comparison with the vanilla training of large dense transformers having equivalent parameter counts.
- ⋆ SMoE-Dropout naturally provides a **"self-slimmable"** property offering consistent boosted performance for transformers with an increase in activated experts during inference and downstream fine-tuning (Figure 1 (blue)), subjected to resource availability using a randomly initialized and fixed router network. It enjoys a once-for-all in-situ trade-off between efficiency and performance when considering deployment.
- ⋆ Our extensive experiments across representative architectures on a variety of tasks validate the effectiveness of our proposed SMoE-Dropout. Specifically, during pre-training,

our approach has $\{1.37,\ 4.10\}$, $\{2.53,\ 12.44\}$ and $\{154.12,\ 188.00\}$ ($\times 10^{-2}$) lower BPC than {vanilla dense training (with the same parameter counts), learned SMoE} for Transformer-XL, BERT, and RoBERTa, respectively; after transferring, SMoE-Dropout obtains $\{0.07\%, 1.03\%, 0.78\%, 1.09\%\}$ performance improvements for BERT and $\{-, 5.88\%, 0.07\%, 5.04\%\}$ for RoBERTa, on {`CSQA`, `ASDiv-A`, `MAWPS`, `SVAMP`} reasoning tasks compared to its dense training counterpart.

2 Related Works

Mixture of Experts (MoE). MoE is a special kind of neural network, where its parameters are partitioned into a series of sub-modules (or called experts), and a conditional computation is performed in an input-dependent fashion (Jacobs et al., 1991; Jordan & Jacobs, 1994; Chen et al., 1999; Yuksel et al., 2012). The traditional dense MoEs require intensive computational cost since they adopt all experts for each input (Eigen et al., 2013). Fortunately, recent investigations (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2021) have proved the effectiveness of MoEs with sparsely activated experts (*i.e.*, SMoE) at both training and inference stages, which greatly trim down the cost and scale language models to enormous sizes like trillions of parameters (Fedus et al., 2021). This efficient fashion of SMoEs gains increasing popularity in various NLP (Shazeer et al., 2017; Lepikhin et al., 2020; Zhou et al., 2022; Zhang et al., 2021; Zuo et al., 2022; Jiang et al., 2021) and vision (Riquelme et al., 2021; Eigen et al., 2013; Ahmed et al., 2016; Gross et al., 2017; Wang et al., 2020; Yang et al., 2019a; Abbas & Andreopoulos, 2020; Pavlitskaya et al., 2020) tasks.

However, its sparse-gated manner incurs several downsides of: (1) *Unstable training*. Zoph et al. (2022) points out that many methods like gradient clipping can stabilize SMoEs' training, but usually at the expense of worse quality. The router z-loss (Zoph et al., 2022) is an ideal solution for the double-win of performance and stability. (2) *Poor specialization*. One of the intriguing goals of SMoE is to divide-and-conquer the learning task by solving each piece of the task with adaptively selected experts (Aoki et al., 2021; Hazimeh et al., 2021; Ma et al., 2018; Mittal et al., 2022). To encourage the specialization and reduce the redundancy across experts (Chen et al., 2022b), Dai et al. (2022) pre-defines the expert assignment for different input categories, and Hazimeh et al. (2021) advocates multiple diverse router policies. (3) *Representation collapse and load imbalance among experts*. As the major issue behind learning-based SMoEs, various approaches are proposed to mitigate its negative effects, which date back to 2017. Shazeer et al. (2017) injects Gaussian noises into gating networks to promote the routing balance. Later on, for the same purpose, Lepikhin et al. (2020); Fedus et al. (2021) apply an auxiliary loss of load balancing regularizers; Lewis et al. (2021) performs the routing by dealing with a linear assignment problem; Clark et al. (2022) utilizes reinforcement learners; and Zhou et al. (2022) routes top-k inputs for each expert instead of selecting experts for each input sample. Beyond learned routing policies, Roller et al. (2021) and Zuo et al. (2022) design deterministic hashing and stochastic assignments, respectively, which discard the need for router networks.

Zuo et al. (2022), one close literature, also recommends the advantage of a stochastic assignment. They randomly activate experts for each input during training and inference, which leads to inconsistent inference results. To alleviate the prediction randomness, Zuo et al. (2022) uses a consistent regularized loss to penalize the discrepancy between different experts. However, such regularization is prone to enforce redundancy to SMoEs and sacrifice the network capacity. In our proposal, the fixed router with random weights generates deterministic inferences. Meanwhile, the presented "self-slimmable" attribute suggests sufficiently exploited expressiveness of the full model.

Dropout and Other Training Techniques for Transformers in NLP. Dropout (Srivastava et al., 2014) is invented to prevent overfitting in the training phase of over-parameterized networks by randomly omitting neurons and their associated connections. Followup studies develop plenty of dropout variants (Zhang & He, 2020; Wan et al., 2013; Ba & Frey, 2013; Kingma et al., 2015; Gal et al., 2017; Wu & Gu, 2015; Tompson et al., 2015; DeVries & Taylor, 2017; Park & Kwak, 2016; Semeniuta et al., 2016). In parallel, McAllester (2013); Mou et al. (2018); Mianjy & Arora (2020); Zhang & Xu (2022); Neklyudov et al. (2017); Gal & Ghahramani (2016) devote themselves in deriving the theoretical foundation for dropout and explaining its implicit regularization impacts.

Other notorious bottlenecks of transformer training mainly lie in overfitting and instability, resulting from poor optimization (Zhang et al., 2019; Liu et al., 2019a; 2020a), insufficient or heterogeneous downstream data (Variš & Bojar, 2021; Zhang & Vaidya, 2021), *etc.*. Accordingly, numerous

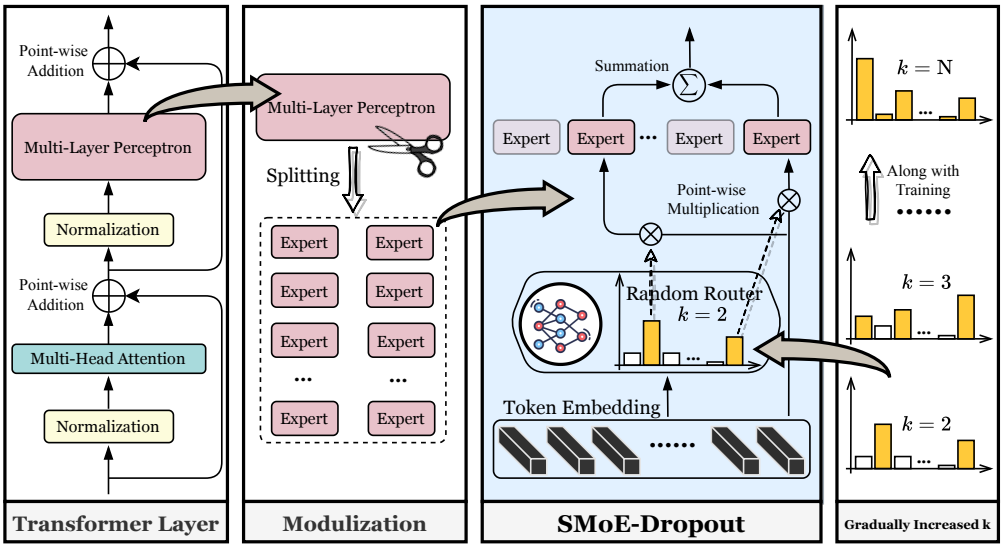

Figure 2: Overview of our proposed SMoE-Dropout. *Left* describes the standard transformer layer, consisting of multi-head attention and multi-layer perceptron (MLP) components. *Middle Left* shows the process of modulization. It splits the original MLP evenly and constructs a series of experts which are smaller MLPs with a reduced hidden dimension. *Middle Right* presents the overall procedure of SMoE-Dropout. The random router selects the top-k experts given a token embedding and then reweights the features from activated experts. In the end, a summation is conducted to aggregate all features. *Right* displays the gradually increased number of chosen experts, along with the training procedure.

remedies are developed to address the issues. For example, data augmentations (Sun et al., 2020), improved initialization (Liu et al., 2020b;a; Xu et al., 2020; Zhu et al., 2021), upgraded normalization (Wang et al., 2022; Yang et al., 2022), enhanced optimizers (Cohen et al., 2022), weight decay (Loshchilov & Hutter, 2017), and early stopping.

3 Methodology

3.1 Preliminary

Sparse Mixture of Experts (SMoEs). SMoE models take advantage of conditional computation activating different subsets of a network for different inputs. A building block of SMoEs is the expert layer including a multi-head attention block and multiple experts in parallel. In this work, we consider SMoE for Transformers, where SMoE layers are incorporated into contiguous Transformer blocks. SMoE expert can be normally constructed by either splitting the vanilla MLP of transformers into smaller pieces (Zhang et al., 2021) or replicating the MLP (Fedus et al., 2021). Most existing SMoE works deal with the MLP component in transformers because MLPs constitute roughly $2/3$ of total model parameters counts storing significant amounts of learned knowledge as memory networks (Geva et al., 2020; Dai et al., 2021).

Let $\{\mathcal{E}_i\}_{i=1}^{\mathrm{N}}$ denote the experts, where i is the index of expert and N is the total number of experts. A gating network or router $\mathcal{R}$ is inserted to choose the top-k experts with the largest scores $\mathcal{R}(\boldsymbol{x})_i$, and $\boldsymbol{x}$ represents the input embedding. Usually, $k \ll \mathrm{N}$, which implies a sparsely activated setting. Specifically, the resultant output of the expert layer can be depicted as follows:

$$\boldsymbol{y} = \sum_{j=1}^{k} \mathcal{R}(\boldsymbol{x})_j \cdot \mathcal{E}_j(\boldsymbol{x}); \mathcal{R}(\boldsymbol{x}) = \texttt{TopK}(\texttt{softmax}(\mathcal{G}(\boldsymbol{x})), k); \texttt{TopK}(\boldsymbol{v}, k) = \begin{cases} \boldsymbol{v} & \text{if } \boldsymbol{v} \text{ is the top } k \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $\mathcal{G}$ is the critical part of a router $\mathcal{R}$. For a learnable routing, $\mathcal{G}$ is a neural network that can be one or a few layers MLP (Shazeer et al., 2017; Fedus et al., 2021). $\mathcal{E}_j(\boldsymbol{x})$ stands for features from the expert $\mathcal{E}_j$. It will be further summed with a scaling coefficient $\mathcal{R}(\boldsymbol{x})_j$ to form the final output $\boldsymbol{y}$. The $\texttt{TopK}$ function maintains the largest k values and sets the reset elements to zero. In practice, a load or important balancing loss (Shazeer et al., 2017) is employed to avoid the representation collapse issue, *i.e.*, always picking the same experts for different inputs and ignoring others.

Dropout and its variants. Dropout is a classic training technique for mitigating potential overfitting. The vanilla dropout is typically applied to fully connected layers with a dropping probability p. For each training iteration, neurons will be disabled with the probability p. In other words, the

omission of neurons follows a $\texttt{Bernoulli}(p)$ distribution. As for the inference phase, there is no dropout and all neurons are activated. To compensate for the additional information during training, the output logits are reweighted by $1 - p$. In this paper, we selected two representatives among diverse proposed dropout variants, concrete dropout (Gal et al., 2017) and dropblock (Ghiasi et al., 2018) as our comparison baselines.

$\triangleright$ *Concrete Dropout.* It replaces the discrete $\texttt{Bernoulli}(p)$ distribution of dropout with a continuous relaxation, *i.e.*, $\texttt{Concrete}$ distribution, and allows an automatic tuning of the dropping probability p. For example, considering the one-dimensional case, as shown in Gal et al. (2017), a $\texttt{Concrete}$ random variable $\boldsymbol{z}$ is described as $\boldsymbol{z} = \texttt{sigmoid}\big(\frac{1}{t} \times \big(\log(p) - \log(1-p) + \log(u) - \log(1-u)\big)\big)$, where $u \sim \texttt{Unif}(0, 1)$ is a uniform random variable and t denotes a temperature hyperparameter. Note that parameter p is optimized in a data-driven way.

$\triangleright$ *DropBlock.* Instead of performing Bernoulli dropping per feature map, Ghiasi et al. (2018) applies it in areas within feature maps. They claim that DropBlock improves the generalization and limits overfitting by hiding certain areas of features or input samples.

3.2 A New Training Pipeline: SMoE-Dropout

Modulization. Our first step in our SMoE-Dropout, turns a large densely connected MLP into multiple smaller MLPs with the same size, as demonstrated in Figure 2. Without loss of generality, in Figure 2, we use a single-layer MLP f with a dimension d for illustrations. After the modulization, it is divided into a set of MLPs $\{\mathcal{E}_1, \mathcal{E}_2 \cdots, \mathcal{E}_{\mathrm{N}}\}$, where they have the same hidden dimension $\frac{d}{\mathrm{N}}$.

Random Routing Policy. Few prior works have investigated some form of random routing policies, such as Roller et al. (2021) utilizes a hash table to enforce a pre-defined deterministic random mapping from inputs to experts and Zuo et al. (2022) adopts a fully random assignment in each training iteration. Although they have shown some benefits from random policies, both methods suffer from inconsistent inference predictions, and can not outperform the densely trained models with the same parameter counts. Our proposed framework, SMoE-Dropout considers a *randomly initialized and fixed router network* to guide the token assignment. Different from previous works, our proposal's assignment is (1) implicitly optimized during training, since feature embeddings are kept updated for the same input sample; (2) deterministic during inference thanks to the fixed weights in $\mathcal{R}$. Extensive results in Section 4 verify the superiority of our proposal, compared to existing random policies and the dense baseline with the same model parameters.

Additionally, another crucial design in SMoE-Dropout's routing is the *progressively enlarged number of activated experts (k).* Riquelme et al. (2021); Jiang et al. (2021) reveals that altering k in the inference phase incurs significant performance degradation if the SMoE is learned with a fixed k. For example, (Riquelme et al., 2021)'s SMoE trained with $k = 1$ has $20\% \sim 30\%$ accuracy drops on ImageNet, when activating $k \geq 7$ experts during the evaluation. This drawback largely limits the practical usage of SMoEs. Because in real-world applications, different scenarios have distinctive resource budgets, requiring flexible and effective network capacity at the inference stage. To tackle this limitation, we adopt a gradually enriched active network capacity along with the training, which is implemented through linearly increasing the number of selected experts k. Such training approach coincided with the principle of curriculum learning brings the attractive **"self-slimmable"** ability offering *consistent boosted performance* for transformers with an increase in activated experts during inference and downstream fine-tuning, as shown in Figure 1.

SMoE-Dropout. Our effective proposal is formed by three simple and highly effective steps, as described in Figure 2. First, it breaks the MLP into a series of MLPs with a reduced size for modulization (*Middle Left* of Figure 2). Then, a random policy parameterized by fixed weights is introduced to route token embeddings to k experts with the largest response (*Middle Right* of Figure 2). Finally, it progressively actives more experts, preventing the overfitting to the amounts of used network capacity during training. (*Right* of Figure 2).

4 Experiment

4.1 Implementation Details

Network Architectures and Comparison Baselines. In our experiments, we have adopted three representative transformer-based networks, including BERT (Devlin et al., 2018), Transformer-XL (Dai et al., 2019), and RoBERTa (Liu et al., 2019b). Specifically, we use double-size $\text{BERT}_{\text{base}}$ / $\text{RoBERTa}_{\text{base}}$ that have 12 transformer layers, 768-dimensional encoder layers, 6144-/3072-

dimensional feed-forward networks (MLPs), and 12 attention heads. For both Transformer-XL, we choose a reduced size due to limited resources, which has 4 layers, 256-dimensional encoder layers, 8192-dimensional feed-forward networks, and 8 attention heads with a head size of 64.

For sufficient comparisons with our proposal, Training w. SMoE-Dropout, we consider five baselines: (i) Densely Training w. Dropout, where the vanilla dropout is applied to feed-forward networks (MLPs); (ii) Densely Training w. Concrete Dropout (Gal et al., 2017); (iii) Densely Training w. DropBlock (Ghiasi et al., 2018). Note that both Concrete Dropout and DropBlock are inserted in feed-forward networks, replacing the vanilla dropout; (iv) Training w. Learnable SMoE (Fedus et al., 2021); (v) Training w. THOR (Zuo et al., 2022), where THOR is another random SMoE that randomly activates a pair of experts for each input sample and adopts an auxiliary consistency regularization based on Kullback-Leibler (KL) divergence. To compute the regularization term, two forward processes are needed in each training iteration.

Pre-Training. $\triangleright$ *Datasets.* Transformer-XL is pre-trained on `enwik8` (Mahoney, 2011) dataset, while we use `BooksCorpus` (Zhu et al., 2015) for BERT and RoBERTa. $\triangleright$ *Training Configurations.* For Transformer-XL, we follow the official training setups, using Adam optimizer and the learning rate starts from 2.5×10^{-4} and decreases according to a cosine annealing scheduler. We use a batch size of 22 and optimize the network for 4×10^5 iterations. As for BERT pre-training, we adopt an AdamW optimizer with an initial learning rate of 5×10^{-5} that linearly decays to 0. The batch size and total training steps are 64 and 1×10^5, respectively. RoBERTa's pre-training configurations strictly follow the default from HuggingFace[1], but with reduced training steps of 1×10^5. Moreover, we conduct a grid search and set the coefficient of THOR's regularization term as 2. Similarly, the temperature in Concrete dropout is $t = 0.1$. $\triangleright$ *Evaluation Metrics.* Since both performance and efficiency are essential, we assess the pre-training performance via Bits-Per-Character (BPC) on the hold-out validation set, where a smaller BPC value indicates a better pre-training; and we report training time per iteration & the number of floating point operations (FLOPs) of single-sample inference, for evaluating the efficiency. $\{1$ RTX A6000, batch size $22\}$ and $\{8$ V100, batch size $64\}$ are adopted for time measurements of Transformer-XL and BERT/RoBERTa, respectively.

Downstream Fine-Tuning. $\triangleright$ *Datasets.* Five benchmarks across three downstream tasks are examined in this paper, including text classification (`SST-2` (Socher et al., 2013)), arithmetic reasoning (`ASDiv-A` (Miao et al., 2020), `MAWPS` (Koncel-Kedziorski et al., 2016), `SVAMP` Patel et al. (2021)), and commonsense reasoning (`CSQA` (Talmor et al., 2018)). $\triangleright$ *Training Configurations.* We perform dense fine-tuning for all approaches. Given a downstream parameter budget, SMoE-based methods will select the most voted experts based on their routing policies. Detailed training setups are listed as follows. We fine-tune the pre-trained Transformer-XL with a smaller learning rate of 1×10^{-4} and a batch size of 64 on `SST-2` benchmark. And for BERT and RoBERTa, we fine-tune the models on the aforementioned four reasoning datasets. The learning rate is fixed at 2×10^{-5} and the batch size is 64. In each downstream task, the fine-tuning continues for 3 epochs, while other configurations are kept the same as the ones in pre-training. $\triangleright$ *Evaluation Metrics.* At the evaluation phase, accuracy (%) and the problem solving rate (%) (Wei et al., 2022) are reported on the test set of `SST-2` and other reasoning tasks, respectively.

4.2 Superior Performance of SMoE-Dropout

We adopt classical transformer-based models, *i.e.*, {Transformer-XL, BERT, RoBERTa}, and train them on {`enwik8`, `BookCorpus`, `BookCorpus`} in a dense or SMoE-based manner. Evaluation results are summarized in Table 1, where all models are compared under the same number of parameter counts. The following observations can be drawn: ❶ Our SMoE-Dropout demonstrates superior performance compared to all other training algorithms. Specifically, SMoE-Dropout with all experts selected obtains $1.37 \sim 18.49$, $0.56 \sim 12.44$, and $152.82 \sim 188.00$ ($\times 10^{-2}$) lower BPC for Transformer-XL, BERT, and RoBERTa, respectively. It validates the effectiveness of our proposals. ❷ Appropriate random routing policies show consistent performance benefits across all three network backbones. Meanwhile, our randomly weighted router surpasses the completely random allocation in THOR, which is within expectation since our assignment is implicitly "optimized" given evolved feature embeddings. ❸ In terms of training efficiency, SMoE-Dropout has up to 21%, 37%, and 25% training time savings compared to the dense training of three backbones. If only activating half experts ($k = \frac{N}{2}$), ours enjoy extra $23\% \sim 34\%$ inference FLOPs reduction with a comparable BPC. Although learnable SMoE reaches the best efficiency, it ends up with inferior performance.

[1] `https://huggingface.co/docs/transformers/model_doc/roberta.`

Table 1: Testing performance of {Transformer-XL, BERT, RoBERTa} network backbones on {`enwik8`, `BookCorpus`, `BookCorpus`} datasets, respectively. **All models are compared under the same number of parameter counts**. Training time (s) and inference FLOPs ($\times 10^{10}$) are reported. For THOR (Zuo et al., 2022), SMoE, and SMoE-Dropout, evaluations are performed with half ($k = \frac{\mathrm{N}}{2}$) or all ($k = \mathrm{N}$) experts activated.

Methods	Transformer-XL			BERT			RoBERTa		
	BPC (↓)	Time	Infer. FLOPs	BPC (↓)	Time	Infer. FLOPs	BPC (↓)	Time	Infer. FLOPs
Dense w. Dropout	1.1623	5.1298	7.7579	7.6546	0.2088	135.72	8.0903	0.1898	101.75
Dense w. Concrete Dropout	1.3335	6.3519	7.7579	7.6419	0.3031	135.72	8.0820	0.2410	101.75
Dense w. DropBlock	1.2468	5.3902	7.7579	7.6349	0.2119	135.72	8.0773	0.1934	101.75
THOR ($k = \mathrm{N}$)	1.3110	4.8830	7.7620	7.6434	0.1439	135.73	8.0778	0.1607	101.76
SMoE ($k = \mathrm{N}$)	1.1896	4.7982	7.7620	7.7537	0.1387	135.73	8.4291	0.1538	101.76
SMoE-Dropout ($k = \frac{\mathrm{N}}{2}$)	1.1776	5.0220	5.6145	7.6372	0.1905	89.330	6.7693	0.1799	78.558
SMoE-Dropout ($k = \mathrm{N}$)	1.1486	5.0220	7.7620	7.6293	0.1905	135.73	6.5491	0.1799	101.76

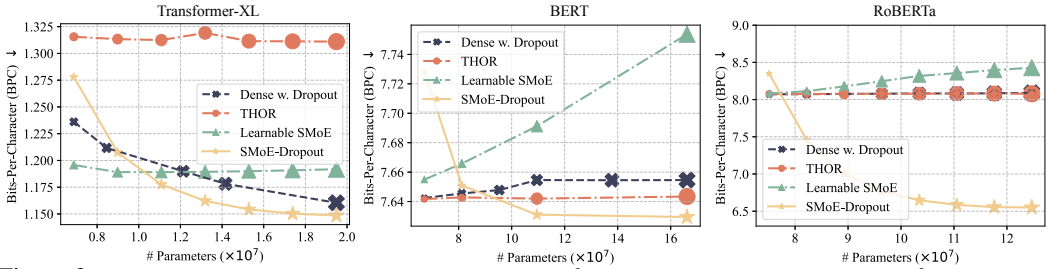

Figure 3: Testing performance over # parameter counts of {Transformer-XL, BERT, RoBERTa} networks on {`enwik8`, `BookCorpus`, `BookCorpus`} datasets, respectively. A smaller BPC suggests a better model.

Besides, we report another group of experiments varying the expert numbers (parameter counts) during evaluation. As shown in Figure 3, for SMoE-based approaches, we directly change the number of activated experts at the inference stage, which is an *in-situ* fashion from the single trained transformer. While for dense training baselines, each dot in their curve requires a separately trained model since it does not allow modifications of network capacity without further fine-tuning. From the results, we see: ❶ The performance of SMoE-Dropout is stably improved along with more parameters used, and it outperforms the others after 1.0, 10, and 8 ($\times 10^7$) parameter counts for three backbones. Such "slimmable" property enables scaling transformers to the full capacity without collapse, bringing a *once-for-all* trade-off respected to inference resource availability. ❷ In contrast, learnable SMoE's and THOR's BPC are quickly saturated and deteriorated when adopting more experts, which implies the existence of expert redundancy (or representation collapse). The potential reasons for their substandard results are (i) the overfitting to fixed # experts utilized during training for learnable SMoE, (ii), and the consistency regularization between experts' predictions for THOR.

4.3 Transfer Study of SMoE-Dropout: Self-Slimmable

We further investigate SMoE-Dropout and its intriguing "self-slimmable" property in a transfer learning scenario. Pre-trained models from Section 4.2 are *densely* fine-tuned on various downstream tasks, including text classification {`SST-2`} and challenging arithmetic & commonsense reasoning {`CSQA`, `ASDiv-A`, `MAWPS`, `SVAMP`}. The performance[2] is collected in Table 2. We find: equipped with SMoE-Dropout, Transformer-XL achieves $0.47\% \sim 2.43\%$ accuracy improvements on `SST-2`, BERT / RoBERTa obtain $\{0.07\% \sim 9.72\%, 0.42\% \sim 3.78\%, 0.26\% \sim 1.30\%, 1.09\% \sim 4.90\%\}$ and $\{-, 2.10\% \sim 5.88\%, 0.07\% \sim 0.27\%, 5.04\% \sim 5.93\%\}$ performance boosts on {`CSQA`, `ASDiv-A`, `MAWPS`, `SVAMP`} respectively, suggesting an enhanced transferability.

Similarly, we alter the model capacity during downstream fine-tuning. Starting from one pre-training, the SMoE-based method first computes the selected times of each expert based on one feedforward pass with downstream data, then chooses the top activated experts to meet certain parameter budgets, and performs the subsequent dense fine-tuning. As displayed in Figure 4, our SMoE-Dropout has a continually increased accuracy or problem-solving rate when involving more parameters, and clearly surpasses the rest of approaches at parameter counts beyond 0.8, 8, and 10.5 ($\times 10^7$) for Transformer-XL, BERT, and RoBERTa respectively. It shows a flexible capacity adjustment, *i.e.*, "self-slimmable", according to the downstream resource constraint.

[2]Due to limited computation resources, {our, official} pre-trained BERT/RoBERTa models are produced with $\{10^5, 10^6\}$ training iterations, {128, 256} batch size, {MLM, MLM and NSP} tasks, on {`BookCorpus` (800M words), `BookCorpus` (800M words) and `English Wikipedia` (2,500M words)} dataset, respectively. The huge gap of pre-training outlays justifies the difference between our and official performance.

Table 2: Transfer performance {Accuracy (% ↑), Problem Solving Rate (% ↑)} of {Transformer-XL, BERT, RoBERTa} networks on {SST-2, CSQA, ASDiv-A, MAWPS, SVAMP} datasets. **All models are compared under the same number of parameter counts**. The same densely fine-tuning is adopted for all approaches, while THOR, SMoE, and SMoE-Dropout are tuned with half ($k = \frac{N}{2}$) or all ($k = N$) experts activated.

Methods	Transformer-XL	BERT				RoBERTa		
	SST-2	CSQA	ASDiv-A	MAWPS	SVAMP	ASDiv-A	MAWPS	SVAMP
Dense w. Dropout	81.94	30.44	55.27	80.47	34.24	49.58	78.06	28.90
THOR ($k = N$)	81.13	20.79	52.52	79.95	30.43	53.36	77.86	28.44
SMoE ($k = N$)	79.98	29.27	55.88	80.73	33.15	52.10	77.86	27.98
SMoE-Dropout ($k = \frac{N}{2}$)	81.60	30.32	54.97	80.99	33.65	52.94	76.30	31.19
SMoE-Dropout ($k = N$)	82.41	30.51	56.30	81.25	35.33	55.46	78.13	33.94

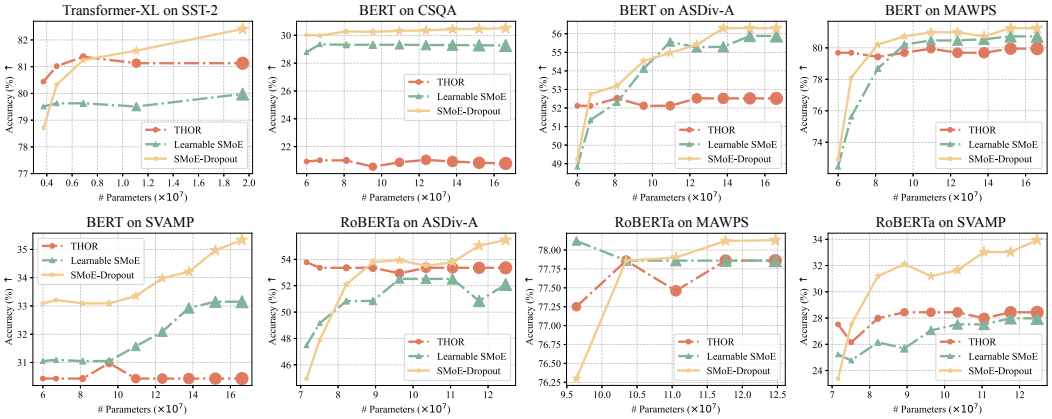

Figure 4: Transfer performance over # parameter counts of {Transformer-XL, BERT, RoBERTa} networks on downstream {SST-2, CSQA, ASDiv-A, MAWPS, SVAMP} datasets, respectively. Only the fine-tuning of Dense w. Dropout needs multiple pre-trained models with different amounts of network capacity.

4.4 Extra Investigation and Ablation Study

***Q1:* When does SMoE-Dropout outperform other baselines? *A1:* Sufficient Model Capacity.**

To answer **Q1** and understand SMoE-Dropout's superiority in diverse scenarios, we investigate our proposal with different model capacities by varying model depth (*e.g.*, layers) & width (*e.g.*, experts).

Model Depth - Different Number of Layers. We conduct experiments on enwik8 dataset with Transformer-XL that has $2, 4, 8, 12$ layers and each layer is turned into the SMoE layer through a modularization. The comparison results of Densely Training w. Dropout and Training w. Learnable SMoE are reported in Figure 5 (a). We find that densely trained transformer performs the best when the network capacity is small like 2 layers, while with sufficiently large model capacity (≥ 4 layers), SMoE-Dropout demonstrates a consistent advantage compared to the others. Meantime, along with the increase of layers, the performance gap of SMoEs between the learned policy and our random policy keeps enlarging, signifying SMoE-Dropout's better scalability.

Model Width - Different Number of Experts. Similarly, we study the influence of model capacity by examining Transformer-XL with different widths of $2, 4, 8, 16$ experts. Results are summarized in Figure 5 (b). Consistent observations can be drawn that: (i) Densely Training w. Dropout outperforms SMoE-based training under small network widths such as ≤ 8 experts; (ii) SMoE-Dropout presents enhanced performance when applied to large models with 16 experts; (iii) Learnable routing policies are effective with a small number of experts like ≤ 8 experts, while it gets worse results than our random routing with a sufficient number of experts, *e.g.*, 16 experts.

***Q2:* What is a better SMoE-Dropout design? *A2:* Random Weight Router; Later-layer SMoE.**

To answer **Q2**, we focus on the main constituents of SMoE-Dropout: *Modularization*, *Random Routing Policies*, and *Gradually Increased* k. Comprehensive ablations are depicted as below.

Ablation on Diverse Random Routing Policies. An appropriate design of random routing policies determines the achievable performance of SMoE-Dropout. We compare our random initialized and fixed router to SMoE with fully random assignments (Zuo et al., 2022) and random hash SMoE with a pre-defined deterministic random assignment (Roller et al., 2021). Transformer-XL results on

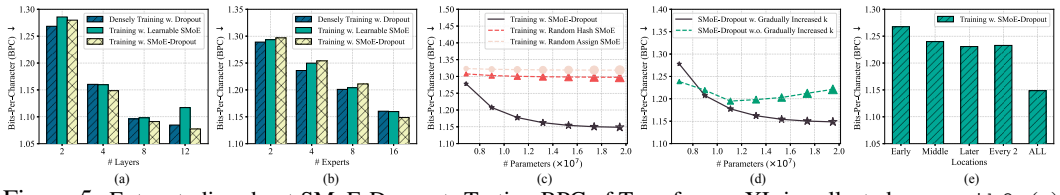

Figure 5: Extra studies about SMoE-Dropout. Testing BPC of Transformer-XL is collected on `enwik8`. (a) and (b) investigate diverse training mechanisms under different model depths and widths, respectively. (c) is the ablation of random routing policies. (d) examines the effects of gradually increased k. (e) studies the appropriate locations to insert SMoE expert layers.

`enwik8` are collected in Fig. 5 (c), where our proposed random routing obtains substantially lower BPC of $2.96 \sim 170.11$ $(\times 10^{-2})$ than the other two under different amounts of model parameters.

Ablation on w./w.o. Gradually Increased k. Figure 5 (d) investigates SMoE-Dropout variants with and without gradually increased k. We see that disabling the progressive manner of enlarging the number of activated experts causes unsatisfied performance. Also, as a result, the "self-slimmable" property completely disappears, *e.g.*, adopting all model parameters leads to worse BPC.

Ablation on Different Positions for Modularization. It remains mysterious where is the best position to insert SMoE layers. To address this question, we perform modularization to different transformer layers and record their performance in Figure 5 (e). Specifically, given a 4-layer Transformer-XL, we compare four options: (i) *Early*, the first two layers are SMoE layers; (ii) *Middle*, the 2nd and 3rd layers are SMoE layers; (iii) *Later*, the last two layers are SMoE layers; (iv) *Every-2*, there is one SMoE layer every two transformer layers, *i.e.*, the 2nd and 4th layers. From the results, introducing SMoEs to later layers is in general more beneficial than modulizing earlier transformer layers. One behind possible reason is that shallow layers might capture common features that need to be shared across input samples. More dissections are left for future works.

***Q3:* Extra benefits from SMoE-Dropout? *A3:* Improved Distillation and Less Overfitting.**

Distilling into Single Expert on Downstream. Besides all the benefits in pre-training inference and downstream transfer, we explore additional advantages of SMoE-Dropout under the distillation scheme that is usually preferred in resource-limited applications. As shown in Table 3, we distill all pre-trained Transformer-XLs into the same smaller variant with a single expert on the `SST-2` downstream task. Our algorithm produces the most distillable models among all four methods by a clear accuracy margin of $0.76\% \sim 1.89\%$.

Table 3: Distillation results of Transformer-XL on `SST-2`.

Method	Accuracy (↑)
Dense w. Dropout	81.25
THOR	80.76
SMoE	80.12
SMoE-Dropout	82.01

Overfitting. We examine SMoE-Dropout, SMoE, and densely trained transformers regarding how likely they overfit the training data distribution with the increase of model parameters. As shown in Figure 5 (a) and (b), experiments are conducted on `enwik8` with Transformer-XL, and three approaches are compared under the same parameter counts. We observe both SMoE-Dropout and Densely Training w. Dropout do not show any sign of overfitting. That is, the performance is consistently improved as we increase the layers from 2 to 12 or experts from 2 to 16. In contrast, Training w. Learnable SMoE incurs BPC deterioration owing to overfitting when we expend the transformer to 12 layers, similar to the findings in Zoph et al. (2022). We credit the reduced overfitting to the implicit regularization effect of SMoE-Dropout and Dropout.

5 Conclusion

In this paper, we present a novel plug-and-play strategy for training over-parameterized transformer models, which leverages SMoEs to empower scaling transformers to better performance in full-capacity settings without collapse. Unlike existing SMoEs, we design a randomly initialized and fixed router to assign experts and gradually increase their number along with the training. As a result, our proposal grants an appealing "self-slimmable" property to large transformers during inference and downstream fine-tuning, based on resource availability. It implies alleviated representation collapse and delivers a once-for-all in-situ trade-off between efficiency and performance. Extensive empirical studies across various combinations of network backbone and dataset, consistently demonstrate the significantly improved performance and training time savings from our algorithm. Future work includes the extension of other network architectures and tasks like vision recognition.

REFERENCES

Alhabib Abbas and Yiannis Andreopoulos. Biased mixtures of experts: Enabling computer vision inference under data transfer limitations. *IEEE Transactions on Image Processing*, 29:7656–7667, 2020.

Karim Ahmed, Mohammad Haris Baig, and Lorenzo Torresani. Network of experts for large-scale image categorization. In *European Conference on Computer Vision*, pp. 516–532. Springer, 2016.

Raquel Aoki, Frederick Tung, and Gabriel L Oliveira. Heterogeneous multi-task learning with expert diversity. *arXiv preprint arXiv:2106.10595*, 2021.

Jimmy Ba and Brendan Frey. Adaptive dropout for training deep neural networks. *Advances in neural information processing systems*, 26, 2013.

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. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Ke Chen, Lei Xu, and Huisheng Chi. Improved learning algorithms for mixture of experts in multi-class classification. *Neural networks*, 12(9):1229–1252, 1999.

Tianlong Chen, Zhenyu Zhang, Yu Cheng, Ahmed Awadallah, and Zhangyang Wang. The principle of diversity: Training stronger vision transformers calls for reducing all levels of redundancy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12020–12030, 2022a.

Tianyu Chen, Shaohan Huang, Yuan Xie, Binxing Jiao, Daxin Jiang, Haoyi Zhou, Jianxin Li, and Furu Wei. Task-specific expert pruning for sparse mixture-of-experts. *arXiv preprint arXiv:2206.00277*, 2022b.

Zewen Chi, Li Dong, Shaohan Huang, Damai Dai, Shuming Ma, Barun Patra, Saksham Singhal, Payal Bajaj, Xia Song, and Furu Wei. On the representation collapse of sparse mixture of experts. *arXiv preprint arXiv:2204.09179*, 2022.

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 preprint arXiv:2204.02311*, 2022.

Aidan Clark, Diego de las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, et al. Unified scaling laws for routed language models. *arXiv preprint arXiv:2202.01169*, 2022.

Jeremy M Cohen, Behrooz Ghorbani, Shankar Krishnan, Naman Agarwal, Sourabh Medapati, Michal Badura, Daniel Suo, David Cardoze, Zachary Nado, George E Dahl, et al. Adaptive gradient methods at the edge of stability. *arXiv preprint arXiv:2207.14484*, 2022.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, and Furu Wei. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*, 2021.

Yong Dai, Duyu Tang, Liangxin Liu, Minghuan Tan, Cong Zhou, Jingquan Wang, Zhangyin Feng, Fan Zhang, Xueyu Hu, and Shuming Shi. One model, multiple modalities: A sparsely activated approach for text, sound, image, video and code. *arXiv preprint arXiv:2205.06126*, 2022.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. Cognitive graph for multi-hop reading comprehension at scale. *arXiv preprint arXiv:1905.05460*, 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts. *arXiv preprint arXiv:1312.4314*, 2013.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*, 2021.

Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. *Advances in neural information processing systems*, 29, 2016.

Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. *Advances in neural information processing systems*, 30, 2017.

Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Deming Chen, Marianne Winslett, Hassan Sajjad, and Preslav Nakov. Compressing large-scale transformer-based models: A case study on bert. *arXiv preprint arXiv:2002.11985*, 2020.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2020.

Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. *Advances in neural information processing systems*, 31, 2018.

Sam Gross, Marc'Aurelio Ranzato, and Arthur Szlam. Hard mixtures of experts for large scale weakly supervised vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6865–6873, 2017.

Fu-Ming Guo, Sijia Liu, Finlay S Mungall, Xue Lin, and Yanzhi Wang. Reweighted proximal pruning for large-scale language representation. *arXiv preprint arXiv:1909.12486*, 2019.

Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, Zhaohui Yang, Yiman Zhang, and Dacheng Tao. A survey on visual transformer. *ArXiv*, abs/2012.12556, 2020.

Hussein Hazimeh, Zhe Zhao, Aakanksha Chowdhery, Maheswaran Sathiamoorthy, Yihua Chen, Rahul Mazumder, Lichan Hong, and Ed Chi. Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Ajay Jaiswal, Liyan Tang, Meheli Ghosh, Justin Rousseau, Yifan Peng, and Ying Ding. Radbert-cl: Factually-aware contrastive learning for radiology report classification. *Proceedings of machine learning research*, 158:196–208, 2021.

Hao Jiang, Ke Zhan, Jianwei Qu, Yongkang Wu, Zhaoye Fei, Xinyu Zhang, Lei Chen, Zhicheng Dou, Xipeng Qiu, Zikai Guo, et al. Towards more effective and economic sparsely-activated model. *arXiv preprint arXiv:2110.07431*, 2021.

Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

Jared Kaplan, Sam McCandlish, T. J. Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeff Wu, and Dario Amodei. Scaling laws for neural language models. *ArXiv*, abs/2001.08361, 2020.

Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. *Advances in neural information processing systems*, 28, 2015.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. Mawps: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1152–1157, 2016.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*, pp. 6265–6274. PMLR, 2021.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019a.

Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. Understanding the difficulty of training transformers. *arXiv preprint arXiv:2004.08249*, 2020a.

Xiaodong Liu, Kevin Duh, Liyuan Liu, and Jianfeng Gao. Very deep transformers for neural machine translation. *arXiv preprint arXiv:2008.07772*, 2020b.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1930–1939, 2018.

Matt Mahoney. Large text compression benchmark, 2011.

Zhiyuan Mao, Ajay Jaiswal, Zhangyang Wang, and Stanley H. Chan. Single frame atmospheric turbulence mitigation: A benchmark study and a new physics-inspired transformer model. *ArXiv*, abs/2207.10040, 2022.

David McAllester. A pac-bayesian tutorial with a dropout bound. *arXiv preprint arXiv:1307.2118*, 2013.

J. S. McCarley, Rishav Chakravarti, and Avirup Sil. Structured pruning of a bert-based question answering model. *arXiv preprint arXiv:1910.06360*, 2019.

Poorya Mianjy and Raman Arora. On convergence and generalization of dropout training. *Advances in Neural Information Processing Systems*, 33:21151–21161, 2020.

Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 975–984, 2020.

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32, 2019.

Sarthak Mittal, Yoshua Bengio, and Guillaume Lajoie. Is a modular architecture enough? *arXiv preprint arXiv:2206.02713*, 2022.

Wenlong Mou, Yuchen Zhou, Jun Gao, and Liwei Wang. Dropout training, data-dependent regularization, and generalization bounds. In *International conference on machine learning*, pp. 3645–3653. PMLR, 2018.

Kirill Neklyudov, Dmitry Molchanov, Arsenii Ashukha, and Dmitry P Vetrov. Structured bayesian pruning via log-normal multiplicative noise. *Advances in Neural Information Processing Systems*, 30, 2017.

Sungheon Park and Nojun Kwak. Analysis on the dropout effect in convolutional neural networks. In *Asian conference on computer vision*, pp. 189–204. Springer, 2016.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam M. Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *ICML*, 2018.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*, 2021.

Svetlana Pavlitskaya, Christian Hubschneider, Michael Weber, Ruby Moritz, Fabian Huger, Peter Schlicht, and Marius Zollner. Using mixture of expert models to gain insights into semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 342–343, 2020.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*, 2022.

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. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34, 2021.

Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, and Jason E Weston. Hash layers for large sparse models. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=lMgDDWb1ULW.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. Recurrent dropout without memory loss. *arXiv preprint arXiv:1603.05118*, 2016.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Lichao Sun, Congying Xia, Wenpeng Yin, Tingting Liang, Philip S Yu, and Lifang He. Mixup-transformer: dynamic data augmentation for nlp tasks. *arXiv preprint arXiv:2010.02394*, 2020.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*, 2018.

Yixuan Tang, Hwee Tou Ng, and Anthony KH Tung. Do multi-hop question answering systems know how to answer the single-hop sub-questions? *arXiv preprint arXiv:2002.09919*, 2020.

Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 648–656, 2015.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herv'e J'egou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021.

Dušan Variš and Ondřej Bojar. Sequence length is a domain: Length-based overfitting in transformer models. *arXiv preprint arXiv:2109.07276*, 2021.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pp. 1058–1066. PMLR, 2013.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1,000 layers. *arXiv preprint arXiv:2203.00555*, 2022.

Xin Wang, Fisher Yu, Lisa Dunlap, Yi-An Ma, Ruth Wang, Azalia Mirhoseini, Trevor Darrell, and Joseph E Gonzalez. Deep mixture of experts via shallow embedding. In *Uncertainty in artificial intelligence*, pp. 552–562. PMLR, 2020.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

Haibing Wu and Xiaodong Gu. Towards dropout training for convolutional neural networks. *Neural Networks*, 71:1–10, 2015.

Peng Xu, Dhruv Kumar, Wei Yang, Wenjie Zi, Keyi Tang, Chenyang Huang, Jackie Chi Kit Cheung, Simon JD Prince, and Yanshuai Cao. Optimizing deeper transformers on small datasets. *arXiv preprint arXiv:2012.15355*, 2020.

Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. *Advances in Neural Information Processing Systems*, 32, 2019a.

Qiming Yang, Kai Zhang, Chaoxiang Lan, Zhi Yang, Zheyang Li, Wenming Tan, Jun Xiao, and Shiliang Pu. Unified normalization for accelerating and stabilizing transformers. *arXiv preprint arXiv:2208.01313*, 2022.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. End-to-end open-domain question answering with bertserini. *arXiv preprint arXiv:1902.01718*, 2019b.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019c.

Seniha Esen Yuksel, Joseph N. Wilson, and Paul D. Gader. Twenty years of mixture of experts. *IEEE Transactions on Neural Networks and Learning Systems*, 23(8):1177–1193, 2012. doi: 10.1109/TNNLS.2012.2200299.

Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank J Reddi, Sanjiv Kumar, and Suvrit Sra. Why adam beats sgd for attention models. 2019.

Minjia Zhang and Yuxiong He. Accelerating training of transformer-based language models with progressive layer dropping. *Advances in Neural Information Processing Systems*, 33:14011–14023, 2020.

Wancong Zhang and Ieshan Vaidya. Mixup training leads to reduced overfitting and improved calibration for the transformer architecture. *arXiv preprint arXiv:2102.11402*, 2021.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Moefication: Conditional computation of transformer models for efficient inference. *arXiv preprint arXiv:2110.01786*, 2021.

Zhongwang Zhang and Zhi-Qin John Xu. Implicit regularization of dropout. *arXiv preprint arXiv:2207.05952*, 2022.

Minghang Zheng, Peng Gao, Renrui Zhang, Xiaogang Wang, Hongsheng Li, and Hao Dong. End-to-end object detection with adaptive clustering transformer. *ArXiv*, abs/2011.09315, 2021.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew Dai, Zhifeng Chen, Quoc Le, and James Laudon. Mixture-of-experts with expert choice routing. *arXiv preprint arXiv:2202.09368*, 2022.

Chen Zhu, Renkun Ni, Zheng Xu, Kezhi Kong, W Ronny Huang, and Tom Goldstein. Gradinit: Learning to initialize neural networks for stable and efficient training. *Advances in Neural Information Processing Systems*, 34:16410–16422, 2021.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pp. 19–27, 2015.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. Designing effective sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

Simiao Zuo, Xiaodong Liu, Jian Jiao, Young Jin Kim, Hany Hassan, Ruofei Zhang, Jianfeng Gao, and Tuo Zhao. Taming sparsely activated transformer with stochastic experts. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=B72HXs80q4.

A1 More Implementation Details

Algorithm 1: Concrete Dropout in a PyTorch-like style

```
class ConcreteDropout(nn.Module):
   def __init__(self, weight_regularizer=1e-6,
             dropout_regularizer=1e-5, init_min=0.1, init_max=0.1):
      super(ConcreteDropout, self).__init__()

      self.weight_regularizer = weight_regularizer
      self.dropout_regularizer = dropout_regularizer
      init_min = np.log(init_min) - np.log(1. - init_min)
      init_max = np.log(init_max) - np.log(1. - init_max)
      self.p_logit = nn.Parameter(torch.empty(1).uniform_(init_min,
          init_max))

   def forward(self, input, next_layer):
      p = torch.sigmoid(self.p_logit)
      # Apply Concrete Dropout
      output = self._concrete_dropout(input, p)
      # Feed forward through the next layer
      output = next_layer(output)
      # Calculate the weight regularizer
      sum_of_square = 0
      for param in next_layer.parameters():
         sum_of_square += torch.sum(torch.pow(param, 2))
      weights_regularizer = self.weight_regularizer * sum_of_square / (1
          - p)
      # Calculate the dropout regularizer
      dropout_regularizer = p * torch.log(p)
      dropout_regularizer += (1. - p) * torch.log(1. - p)
      input_dimensionality = input[0].numel()
      dropout_regularizer *= self.dropout_regularizer *
          input_dimensionality
      regularization = weights_regularizer + dropout_regularizer

      return output, regularization

   def _concrete_dropout(self, input, p):
      eps = 1e-7; temp = 0.1
      # Calculate the dropout probability matrix
      unif_noise = torch.rand_like(input)
      drop_prob = (torch.log(p + eps)
            - torch.log(1 - p + eps)
            + torch.log(unif_noise + eps)
            - torch.log(1 - unif_noise + eps))
      drop_prob = torch.sigmoid(drop_prob / temp)
      random_tensor = 1 - drop_prob
      retain_prob = 1 - p
      # Apply Concrete Dropout
      output = torch.mul(input, random_tensor)
      output /= retain_prob

      return output
```

Algorithm 2: DropBlock in a PyTorch-like style

```
def drop_block(input, drop_prob, block_size):
   # Calculate the mask with zero value for block-wise elements
   mask = torch.rand_like(x).lt(drop_prob).float()
   mask = F.max_pool1d(mask, block_size, 1, block_size // 2)
   mask = 1 - mask
   output = input * mask # Apply dropout
   return output
```

Details of Concrete Dropout and DropBlock. In our experiments, we use Concrete-Dropout or DropBlock to replace the original dropout layer in MLP blocks. And for Concrete Dropout, we adopt the official implementation `https://github.com/yaringal/ConcreteDropout/blob/master/concrete-dropout-pytorch.ipynb`. For DropBlock, we follow the method in Ghiasi et al. (2018). And the PyTorch-style pseudo codes for both methods are presented in Algorithm 1 and 2.

A2 More Experiment Results

A2.1 Stability Analysis

To evaluate the stability of the improvement obtained by our SMoE-Dropout, we carry out further experiments of Transformer-XL on `SST-2`. The results are reported in Table A4, from which we can observe that our SMoE-Dropout achieves a statistically significant improvement of $0.93\% \sim 1.17\%$ accuracy gains compared with other SMoE-variants and the dense network, where there is no overlap between the error bars (one standard deviation).

Table A4: Test accuracy of Transformer-XL on `SST-2`. Both the average and standard deviation of accuracy are reported across 3 independent runs.

Method	Accuracy (↑)
Dense w. Dropout	81.39 ± 0.31
THOR ($k = \mathrm{N}$)	81.15 ± 0.55
SMoE ($k = \mathrm{N}$)	81.20 ± 0.50
SMoE-Dropout ($k = \frac{\mathrm{N}}{2}$)	82.03 ± 0.26
SMoE-Dropout ($k = \mathrm{N}$)	82.32 ± 0.14

A2.2 Comparison with Learnable SMoEs w. Gradually Increased k

Table A5 demonstrates that both random routing policy and progressively increasing the number of activated experts are beneficial for alleviating representation collapse and providing self-slimmable property, yet not as good as combining both. To be specific, when applying the strategy of progressively enlarging the number of activated experts, the learnable SMoEs suffer less representation collapse and achieve better performance, *i.e.*, 0.31% higher accuracy. Meanwhile, We find that learnable SMoE with curriculum learning has the self-slimmable property only when activating experts from $k = 1$ to $k = 8$. However, the performance starts to degrade if using more experts like $k = 16$. As for our SMoE-Dropout with a random routing, it enjoys a better "self-slimmable" property from $k = 1$ to $k = 16$ (full model capacity), with up to 0.87% higher accuracy on `SST-2` across all scenarios, compared to its learnble varints.

Table A5: Testing accuracy (%) over # activated experts of Transformer-XL on `SST-2`.

# Activated Experts	1	2	4	8	16
SMoE w.o. Gradually Increased k	80.06	80.79	80.58	80.99	81.20
SMoE w. Gradually Increased k	79.40	81.02	81.13	81.71	81.51
SMoE-Dropout	79.02	81.25	82.00	82.03	82.32

A2.3 Transfer Study on Multi-Step Reasoning Tasks

We conduct a further transfer study of the pre-trained BERT networks on a multi-hop question-answering dataset, `HotpotQA` Yang et al. (2018). And We use exact match (EM) accuracy to assess networks' performance. Following the same metric in Press et al. (2022), we calculate the compositionality gap, i.e., the gap of EM accuracy between multi-hop question answering and its all single-hop sub-questions Tang et al. (2020), of each network. As shown in Table A6, our SMoE-Dropout is beneficial for reducing the compositionality gap, which achieves the best performance with up to 0.30% higher EM score and 0.30% narrower compositionality gap, compared with the learnble SMoE and its dense counterpart.

Table A6: The EM score and compositionality gap of the pre-trained BERT networks on `HotpotQA`.

Metrics	Dense w. Dropout	SMoE	SMoE-Dropout
EM Accuracy (%)	15.10	14.90	15.20
Compositionality Gap (%)	14.90	15.00	14.70