Acceleration of Large Transformer Model Training by Sensitivity-Based Layer Dropping

Yujie Zeng[1], Wenlong He[1], Ihor Vasyltsov[2], Jiali Pang[1], Lin Chen[1]
[1]Samsung R&D Institute China Xian
[2] Samsung Advanced Institute of Technology
{yujie.zeng, wenlong.he, ihor.vasiltsov, jiali.pang, lin81.chen}@samsung.com

Abstract

Transformer models are widely used in AI applications such as Natural Language Processing (NLP), Computer Vision (CV), etc. However, enormous computation workload becomes an obstacle to train large transformer models efficiently. Recently, some methods focus on reducing the computation workload during the training by skipping some layers. However, these methods use simple probability distribution and coarse-grained probability calculation, which significantly affect the model accuracy. To address the issue, in this paper we propose a novel method to accelerate training—Sensitivity-Based Layer Dropping (SBLD). SBLD uses layer-wise sensitivity data to switch on/off transformer layers in proper order to keep high accuracy. Besides, we adjust the probability of skipping transformer layers with a scheduler to accelerate training speed and get faster convergence. Our results show that SBLD solves the accuracy drop issue compared with prior layer dropping methods. Our SBLD method can decrease end-to-end training time by 19.67% during training of GPT-3 Medium model, the same time increasing the accuracy by 1.65% w.r.t. baseline. Furthermore, for SwinV2-L model the obtained Top-1 and Top-5 accuracies are also higher vs. the baseline. Thus, the proposed method is efficient and practical to improve the large transformer model training.

Introduction

Natural Language Processing models such as transformer are widely used in AI application nowadays. Texts, images and audios all can be used to train DL models for tasks such as question answering, sentiment analysis, information extraction, image captioning and so on. According to existing research results (Arici et al. 2021, Brown et al. 2020, Liu et al. 2021, Dehghani et al. 2019, Shoeybi et al. 2019), the NLP models with transformer can solve the text processing related tasks adequately. However, due to the huge parameter numbers in transformer models, enormous computation is involved in training. Transformer-based models cost too much time and resources in unsupervised pre-training phase. Therefore, it is urgently required to find the methods allowing accelerate training, while maintain the original transformer accuracy.

One of the effective speed-up idea is to reduce computation in training. Progressive Layer Dropping (PLD) (Zhang and He 2020) is one of the layer dropping methods that can reduce computation efficiently. PLD method determines to drop some layers with a probability decided by the depth position of the layers. However, such an approach cannot describe these probabilities for different transformer structures accurately enough. Also, PLD proposes a progressive schedule to change the probability of layer dropping. PLD method increases the layers' dropping rate for deeper layers and longer training iterations. However, when the model scale is too large (e.g., GPT-3 (Brown et al. 2020) model), PLD's schedule is coarse-grained and causes serious accuracy loss because many important layers are dropped. Therefore, it is necessary to design an acceleration method that can better keep the transformer structures to maintain the accuracy even when the model scale is large.

In this paper, we propose a new layer dropping method to better tradeoff between high accuracy and low training cost. Our proposal is based on several strategies combined:

- We apply a layer-wise sensitivity method, which allows allocating the drop probability to each layer, in order to control switchable transformer layers during the training.
- To further increase the model accuracy, we add Block/Allow Lists on top of layer-wise sensitivity, which avoid dropping the most sensitive layers.
- We develop a variance-based scheduler to adjust the probability of dropping transformer layers to accelerate the convergence speed while maintaining the original transformer structure.

Experiment results showed that SBLD can increase throughput by about 20.0% in the pre-training stage, compared with baselines on GPT-3 model with the nearly the same accuracy. Besides, SBLD improves accuracy of GPT-3 Medium model on LAMBADA (Paperno et al. 2016)

benchmark from 48.09% to 50.75%. Moreover, we evaluate the effect of SBLD on Swin Transformer (Liu et al. 2022) and find that both Top-1 and Top-5 accuracies have increased also, even if we drop some of the transformer layers.

Related Works

There are several methods based on dropping layers of the model in order to reduce DL training time, such as Autofreeze (Liu et al. 2021), Token Dropping (Hou et al. 2022), FCA (Zhao et al. 2022), and PLD (Zhang and He 2020). AutoFreeze is a system that uses an adaptive approach to choose training partial layers to accelerate model fine-tuning while preserving accuracy. Token Dropping method accelerate the pre-training of transformer models by dropping unimportant tokens identified by Masked Language Modeling (MLM) loss starting from an intermediate layer in the model to make the model focus on important tokens. FCA is a fine- and coarse-granularity hybrid self-attention method that reduces the computation cost through progressively shortening the computational sequence length in self-attention. PLD method is based on the Switchable-Transformer (ST) block structure that can be easily switched on/off with predefined probabilities. Therefore, it stabilizes transformer network training with some layers dropped in stochastic manner. PLD assumes that layer dropping probability should increase along the temporal dimension, so it exponentially decays the keep probability along the time dimension. These existing methods all showed effectiveness for transformer training. However, they all somehow limited for specific scenarios. AutoFreeze is dedicated for fine-tune, Token Dropping is dedicated for Language models, and FCA is only for classification task. PLD causes performance degradation due to the coarse-grained layer importance allocation.

Comparing with existing methods, our proposed SBLD method can be used for pre-training and fine-tuning of all transformer-based models such as language transformers, vision transformers and other transformers. In addition, SBLD method can accelerate distributed model training by layer dropping with no obvious accuracy drop or even improve the model training accuracy.

Methodology

This section describes the details and workflow of our SBLD method. SBLD method works as an independent component in distributed DL model training process. It is completely compatible with the original transformer model. Before applying SBLD, we convert the transformer layers in the model to be switchable transformer layers similarly to PLD. Then every switchable transformer layer should get a probability to be dropped or kept. *Drop probability* is the probability that we drop a layer. *Keep probability* is the probability we keep a layer. *Drop probability* = 1 - *Keep probability.* Therefore, we need to allocate the optimal keep probability value for each layer in each iteration.

SBLD method includes four steps as shown in Figure 1:

- Firstly, we collect base layer-wise sensitivity S_{base} with a few training iterations by dropping each layers of the whole model.
- Secondly, we define Block/Allow Lists according to the base layer-wise sensitivity data.
- Thirdly, we define a variance-based scheduler to calculate the optimal keep probability (θ_t) of the whole model for current iteration.
- Fourthly, we combine the keep probability of current iteration and base layer-wise sensitivity to update the optimal keep probability list P_t.

Layer-Wise Sensitivity

It is known that every layer in the model has different impact on the overall model convergence (Dong et al. 2019). Different with the linear decay strategy used in PLD method (Zhang and He 2020), layer-wise sensitivity can describe the importance of each layer more accurately. Therefore, we use layer-wise sensitivity to define the importance of each layer in the model. The higher sensitivity value a layer has, the more model accuracy is affected if we drop this layer. Therefore, we propose to drop with lower probability the layers which have higher sensitivity and drop with higher probability those which have lower sensitivity.

Layer-wise sensitivity is a commonly used method in DL area. There are many existing methods to analyze the layer-wise sensitivity of the model. For instance, second-order information (Wang et al. 2019) is used to solve the exponentially large search space in Mixed-Precision quantization. HAWQ (Yao et al. 2021) allows for the automatic selection of the relative quantization precision of each layer, based on the layer's Hessian spectrum. Here we use a pre-analyze method to collect layer-wise sensitivity data.

We first train the original model with few iterations (e.g., 500) to get loss change trend of our baseline model. Then we set *Drop probabilit*y = 0.9 for one specific layer and *Drop probabilit*y = 0.0 for other layers in the model. Here we use *Drop probabilit*y = 0.9 since higher drop probability allows to get more accurate and clear sensitivity value. We next train model with another round with same iteration numbers as baseline. Finally, we can get a loss gap value between baseline and layer dropped case for the specific layer. Figure 2 briefly shows an example to get loss gap values of different layers.

With all loss gap values of all the layers collected, we normalize the loss gap values to range [0.00, … , 1.00] and get the base sensitivity list S_{base}= [α_1, α_2, α_3… α_n], where α_1, α_2, α_3… α_n are the sensitivity values of corresponding layer. Note, that only few training iterations account for a small fraction of the overall training process. Generally, less than

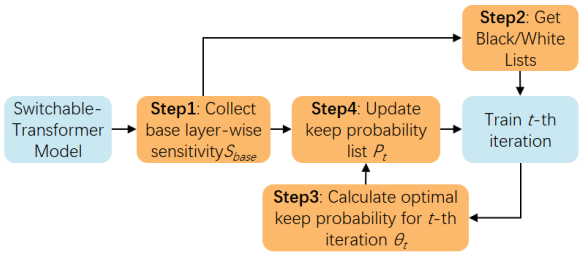

Figure 1: General flow of the proposed SBLD method

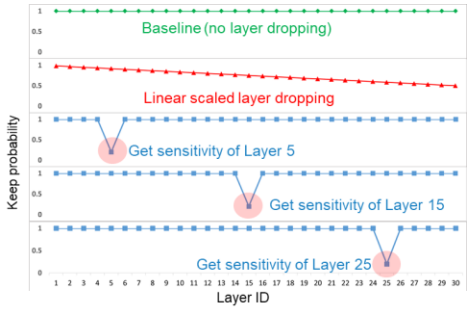

Figure 2: Sensitivity measurement of each layer.

5% of the total training iterations is needed to prepare the sensitivity list S_{base}. In addition, we only measure the sensitivity data once and reuse it repeatedly in the later training trials. Therefore, it would not involve too much computation overhead to the training process. E.g., during training of GPT-3 Medium (0.35B) model, we reuse sensitivity data from GPT-3 (1.7B) model, since both models have the same number of layers.

Block/Allow Lists

During the measurement of sensitivity, we find that dropping some of the layers can get lower loss than baseline. This observation means that dropping some specific layers can improve model accuracy. These layers should be dropped more often than others. On the contrary, there are also some other layers that result in serious increase of the loss value. These layers should be dropped more carefully. Therefore, we use Block/Allow Lists (B/A lists) to further improve model accuracy.

We sort the base sensitivity list with sensitivity value. Typically, we recommend to put 10~20% of the layers into B/A lists to ensure well trade-off between accuracy and acceleration. For the layers in Block list, keep probability is always 1 for all the iterations. That is, we never drop the layers in Block list. For the layers in Allow list, keep probability is always 0.4 (the minimal value of keep probability) for all the iterations. That is, we drop layers in Allow list with a higher probability.

Figure 3 shows an example of unnormalized layer-wise sensitivities, from which the layers to B/A lists can be selected. There are total 30 transformer layers in the model. As we can see that orange bars shows dropping layer result

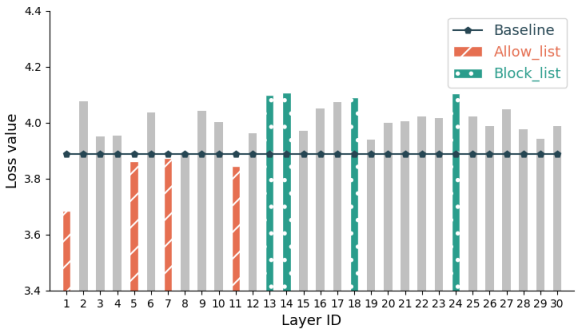

Figure 3: An example of layer-wise sensitivity list with 30 transformer layers

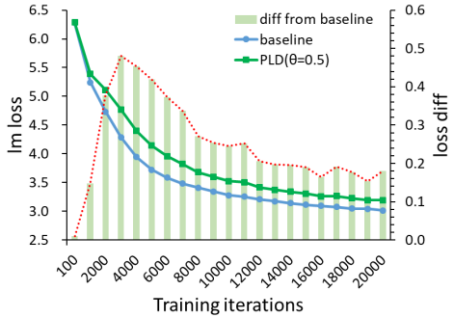

Figure 4: Loss for training iteration in PLD and all-case transformer. Light green is the distribution of training error

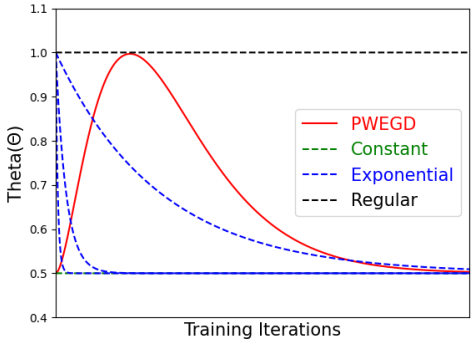

Figure 5: The distribution fitting with common functions.

in lower loss value while green bars result in higher loss. Therefore, we put layer 1, 5, 7, and 11 into Allow list and put layer 13, 14, 18, and 24 into Block list.

Variance-Based Scheduler

Except the static layer-wise sensitivity from the model structure's view, dropping the same layer in different stages of training also has a different impact on the model convergence as Zhang and He (2020) proposed. Therefore, we should consider the training iteration as another factor when searching the optimal keep probability. In order to reveal the impact of dropping layer as the training iteration proceeds, we analyzed the training loss change of baseline

and layer dropping case. We find that the value variance between baseline and layer dropping increases fast and then drop from a highest point as shown in Figure 4.

The error between dropping layer and baseline is crucial for model accuracy. Therefore, we use Poly-Weighted Exponential Gamma Distribution (PWEGD) (Iqbal T. and Iqbal M. Z. 2020) to fit the change curve of variance between baseline and layer dropping over training time. PWEGD is a long-tail distribution combined with exponential distribution. In data fitting application, PWEGD has displayed a strong compatibility and better fitting effect by fusing multiple distributions characteristics. Theoretical analysis showed that with increased weighted-polynomial function, the estimates and the statistical properties such as the expectation, variance, standard error and cumulative distribution function (CDF) are significantly better than the distributions of the existing root distribution or Bernoulli distributions (Iqbal T. and Iqbal M. Z. 2020). Therefore, the PWEGD is worthwhile to obtain a more proper layers' dropping rate. The layers' drop probability schedule obtained by PWEGD can performs better than the exponential schedule in maintaining the structure integrity of the transformer model.

We use the PWEGD scheduler to find optimal keep probability of the whole model for each training step. Firstly, we calculate the influence factor of training process with Eq. (1):

$$sch(t)=\frac{2^{(a+c)}}{\Gamma(a+c)b^{(a+c)}}(t-\varepsilon)^{(a+c-1)}\,e^{\left(-2\frac{t-\varepsilon}{b}\right)}\,, \qquad (1)$$

where a is the shape parameter, b is the scale parameter, c is the polynomial of the weighting function, $\boldsymbol{\varepsilon}$ is the threshold parameter, Γ is the gamma function used to model positively scaled data, and e is the exponential function.

Secondly, amplify the influence factor with a limit value (θ) for keep probability with Eq. (2):

$$\theta_t=\frac{\eta}{\theta^2}sch(t)+\theta\,, \qquad (2)$$

where η controls the change amplitude of scheduler's result value. Combining Eq. (1) and Eq. (2), the final formula to calculate the keep probability θ_t of the whole model for iteration t is:

$$\theta_t=\frac{2^{(a+c)}}{\Gamma(a+c)b^{(a+c)}}(t-\varepsilon)^{(a+c-1)}\,e^{\left(-2*\frac{t-\varepsilon}{b}\right)}\frac{\eta}{\theta^2}+\theta\,, \qquad (3)$$

The changing curve of θ_t is shown in Figure 5. It shows a similar changing trend as the error between dropping layer and baseline. Therefore, the scheduler chooses the optimal keep probability of the whole model at the each iteration of training.

Update Keep Probability

This section describes how to combine the base layer-wise sensitivity S_{base} and the optimal keep probability θ_t of the whole model for *t-th* iteration to get the keep probability list P_t. We combine S_{base} and θ_t according to the pre-defined weights to calculate the optimal keep probability $p(t,l)$ for *l-th* layer. And then append $p(t,l)$ into keep probability list P_t. Before each training step starts, we update the keep probability list P_t of current training iteration t with Algorithm 1.Then the updated keep probability list P_t of current iteration is applied on switchable transformer layers to switch on/off transformer layers.

Overall Workflow

Algorithm 2 shows the whole workflow of SBLD. We initiate transformer layers in original model into switchable transformer layers. Then we measure layer-wise sensitivity to get the base sensitivity list S_{base} and define Block/Allow lists according to S_{base}. User can set a limit value θ to define how many layers are expected to drop. Before each training iteration starts, we calculate an optimal keep probability θ_t for current training iteration. Then update keep probability list P_t for every layer of the model with S_{base} and θ_t. At last, every transformer layer l in the model made a decision randomly from a Bernoulli distribution of probability $P_t[l]$.

The recommended value range of limit value θ is 0.4~1.0. Limit value θ less than 0.4 may make the model become too difficult to train and cause accuracy drop. Figure 6 shows another example of limit value θ of different layers changing with training iterations (the higher the maximum value, the higher the sensitivity of one layer). There are total five transformer layers in the model. As shown in the figure, layer 3 is the most sensitive layer and layer 4 is the least sensitive layer. Keep probability for different iterations of all the layers follow the same change trend with variance-based scheduler.

Experiments

In this section, we demonstrate that proposed SBLD method outperforms existing PLD and no layer dropping methods as baseline (B/L) on various Transformer models. In order to evaluate the performance improvement of SBLD method, we firstly conduct a set of experiments on the pre-training of GPT-3 models. Next, we compared the performance of our pre-trained GPT-3 model on downstream tasks LAMBADA (Paperno et al. 2016). Finally, we apply SBLD method in pre-training and fine-tuning of Swin Transformer V2 Large model. Like PLD, SBLD method can be adjusted with parameter θ to prefer higher accuracy or higher throughput. Larger θ results in higher accuracy. We conducted experiments with different θ values.

Evaluation Environment

All the experiments are performed on DIT GPU-based Supercomputer. Each server has 8 NVIDIA A100-80GB

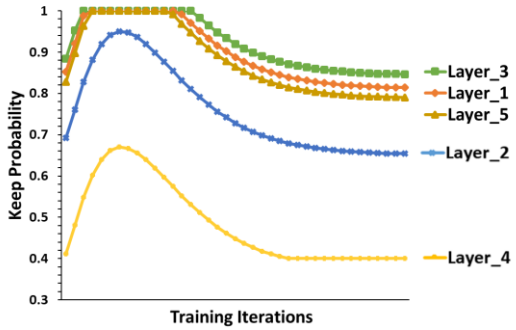

Figure 6: Changing trend of keep probabilities for different layers with training iterations.

Algorithm 1: Update keep probability list

Input: current training step t, base sensitivity list S_{base}, current θ_t
Parameter: $\alpha \leftarrow$a weight for θ_t, $\beta \leftarrow$a weight for base sensitivity data, $\theta_{min} \leftarrow$the minimal value of θ, $\theta_{max} \leftarrow$the maximal value of θ
Output: the calibrated keep probability list P_t

P_t=[]
for each transformer layers in the model:
    l = the layer ID of transformer layer
    $p(t, l)$ = **clamp**($\alpha\theta_t + \beta S_{base}[l]$, θ_{min}, θ_{max})
    Pt.**append**($p(t, l)$)
end
return P_t

GPUs, 128 CPU cores (AMD EPYC 7543 32-Core Processor) and 1.0 TB of RAM. The GPU-to-GPU inter connection is supported by 100G Infiniband and the network bandwidth connecting machines is 40Gbps. The machines run Red Hat 8.4 operating system and the software environment includes Megatron v3.0, Python 3.8.7, CUDA 11.4, PyTorch-1.12.0 and NCCL 2.8.3.

Evaluation on GPT-3 Pre-train

We perform pre-training experiments with two GPT-3 models with different sizes: 1.7 billion parameters and 3.6 billion parameters on 32 and 64 NVIDIA A100-80GB GPUs with BookCorpus dataset (Zhu et al. 2019) for 10,000 steps. Table 1 and Figure 7 shows the result of GPT-3 (1.7B) model. Comparing with baseline, SBLD method improves throughput by 8.36%, 20.74%, 34.69% respectively. Table 2 and Figure 2 shows the result on GPT-3 (3.6B) model. Comparing with baseline, SBLD method improves throughput by 7.91%, 19.28%, 31.18% respectively. For both models, SBLD method achieves lower validation loss than original PLD method. Especially when θ=0.9 in Figure 8, SBLD method showed no accuracy drop, even achieving lower validation loss than baseline in GPT-3 (3.6B) model.

Algorithm 2: SBLD Algorithm

Input: Limit value θ for keep probability
Parameter: shape parameter a, scale parameter b, polynomial of the weighting function c, threshold parameter ε, amplify factor η, total training steps T, total transformer layer numbers L

Init Transformer
S_{base} = **Init Sensitivity**
Define *Block list* and *Allow list*
for t **ranges from** 1 **to** T, **do**:
    $\theta_t = \frac{2^{(a+c)}}{\Gamma(a+c)b^{(a+c)}}(t-\varepsilon)^{(a+c-1)}e^{\left(-2*\frac{t-\varepsilon}{b}\right)}\frac{\eta}{\theta^2}+\theta$
    P_t = ***update_keep_probability_list*** (t, θ_t, S_{base})
    for l **ranges from** 1 **to** L **do:**
        if l in *Block list* **then** p =1
        else if l in *Allow list* **then** p = 0.4
        else = [l]
        action ~ Bernolli(p)
        if action == 0 **then**
            $x_{l+1} = x_l$ //*drop l-th layer*
        else
            $x_{l+1} = layer(x_l)$ //*keep l-th layer*
        l++
    end
    $Y' \leftarrow$output_layer(x_L)
    loss $\leftarrow$loss_fn(Y', Y)
    backward(loss)
end

Evaluation on LAMBADA

The downstream task LAMBADA (Paperno et al. 2016) is a benchmark dataset to evaluate the model by asking the model to predict the last word of sentences after reading a paragraph of context. We perform experiments with LAMBADA with GPT-3 Medium (0.35B) on 64 GPUs.
We train GPT-3 Medium model with PLD and SBLD method with different θ value (0.5/0.7/0.9) for the same iterations. Table 3 shows the accuracy and end-2-end training time of different θ value of PLD and SBLD. Comparing the same iteration for end-to-end training, SBLD method gets a higher accuracy (50.75% vs. 48.09%) with 8.06% less time.
Table 4 shows the accuracy trend of each training iterations. For most of the cases after 300,000 steps (except 900,000) SBLD method (θ=0.9) continuously outperforms all other methods in terms of accuracy. Figure 9 shows the training convergence comparison of baseline and SBLD method (θ=0.9). As we can see, SBLD achieves a better convergence trend from 400K iterations comparing with baseline. Moreover, comparing the same accuracy, SBLD method can even get 35% speedup as shown in Figure 9

GPT-3 model 1.7B		Throughput (TFLOP/s)	Validation loss
B/L		127.29	3.438
θ=0.9	PLD	132.63 **(+4.19%)**	3.497 **(+0.059)**
	SBLD	137.93 **(+8.36%)**	3.473 **(+0.035)**
θ=0.7	PLD	145.56 **(+14.35%)**	3.620 **(+0.182)**
	SBLD	153.69 **(+20.74%)**	3.533 **(+0.095)**
θ=0.5	PLD	161.63 **(+26.97%)**	3.714 **(+0.276)**
	SBLD	171.45 **(+34.69%)**	3.629 **(+0.191)**

Table 1: Throughput and Accuracy comparison of B/L, PLD (Zhang and He 2020) and SBLD on GPT-3 (1.7B).

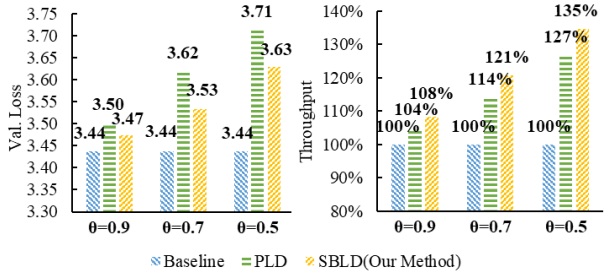

Figure 7: Val. Loss and Throughput comparison of B/L, PLD (Zhang and He 2020) and SBLD on GPT-3 (1.7B)

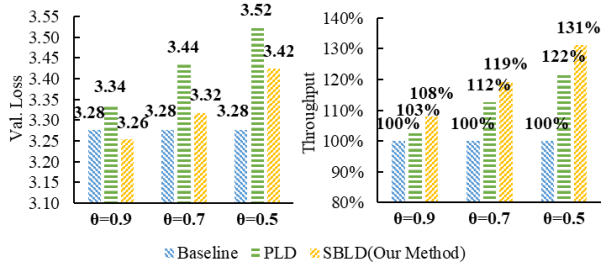

Figure 8: Val. Loss and Throughput comparison of B/L, PLD (Zhang and He 2020) and SBLD on GPT-3 (3.6B)

Evaluation on Swin Transformer

Swin Transformers (Liu et al. 2022) are transformer-based vision model such that the basic component is also transformer block. However, Swin Transformer blocks are designed with four different stages. In addition, in Stage3, there are 18 blocks for SwinV2-L model. Moreover, these18 blocks are designed with "W-MSA" and "SW-MSA" pairs. "W-MSA" and "SW-MSA" work together to ensure that Swin Transformer model can get both the local information and neighbor information. SBLD method drop the "W-MSA" or "SW-MSA" block separately in Stage3.

GPT-3 model 3.6B		Throughput (TFLOP/s)	Validation loss
B/L		128.63	3.277
θ=0.9	PLD	133.01 **(+3.40%)**	3.338 **(+0.061)**
	SBLD	138.81 **(+7.91%)**	3.255 **(-0.022)**
θ=0.7	PLD	144.69 **(+12.49%)**	3.435 **(+0.158)**
	SBLD	153.43 **(+19.28%)**	3.318 **(+0.041)**
θ=0.5	PLD	153.13 **(+21.87%)**	3.523 **(+0.246)**
	SBLD	168.74 **(+31.18%)**	3.424 **(+0.147)**

Table 2: Throughput and Accuracy comparison of B/L, PLD (Zhang and He 2020) and SBLD on GPT-3 (3.6B).

GPT-3 model 0.35B		Accuracy (%)	e2e Time (hours)
B/L		48.09	99.34
θ=0.9	PLD	49.41 (+1.32)	97.12 (-2.23%)
	SBLD	50.75 **(+2.66)**	91.33 **(-8.06%)**
θ=0.7	PLD	49.35 (+1.26)	89.59 (-9.81%)
	SBLD	49.35 **(+1.26)**	85.40 **(-14.03%)**
θ=0.5	PLD	47.49 (-0.60)	81.74 (-17.71%)
	SBLD	49.74 **(+1.65)**	79.79 **(-19.67%)**

Table 3: E2E training time and Accuracy comparison of B/L, PLD (Zhang and He 2020) and SBLD on GPT-3 (0.35B).

GPT-3 0.35B	B/L	PLD (Acc. %)			SBLD (Acc. %)		
Steps	-	θ=0.5	θ=0.7	θ=0.9	θ=0.5	θ=0.7	θ=0.9
100K	32.43	29.19	30.14	31.96	30.62	29.92	30.43
300K	40.77	37.36	39.45	38.87	37.36	38.85	39.55
500K	44.62	41.41	43.86	44.03	42.19	42.29	44.79
700K	46.59	42.15	45.00	46.42	43.7	44.69	47.22
900K	47.41	44.27	46.79	46.91	44.95	46.15	46.85
1.1M	47.58	45.41	48.13	47.35	47.29	48.75	49.29
1.3M	**48.09**	47.49	**49.35**	**49.41**	**49.74**	**49.35**	**50.75**

Table 4: Accuracy comparison of B/L, PLD (Zhang and He 2020) and SBLD with GPT-3 (0.35B) on LAMBADA.

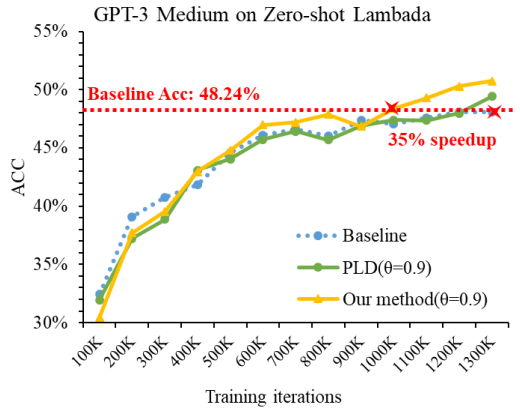

Figure 9: Training convergence comparison of baseline and SBLD method on LAMBADA (θ=0.9).

SwinV2-L	Top-1 Acc	Training time (h)
Baseline (30 Eps)	87.60%	5.973
SBLD (20 Eps)	87.62%	3.86 (-35.37%)
SBLD (30 Eps)	87.74%	5.967 (-0.11%)

Table 5: Final Accuracy comparison of baseline and SBLD on SwinV2-L model.

We conduct experiments for pre-training on ImageNet-21k (image size: 192x192) and fine-tuning on ImageNet-1k (image size: 384x384) with SwinV2-L model on 128 and 64 GPUs respectively.

With the same training iterations, SBLD method achieves better accuracy (87.74% vs. 87.60%) than baseline as shown in Table 5. Moreover, SBLD method shows a faster convergence trend than baseline as shown in Figure 10, what proves the outperforming characteristics of the proposed method. SBLD method can reduce the fine-tune time of SwinV2-L model by 35% (3.86h with 20 epochs vs 5.97h with 30 epochs) with less Top-1 accuracy improvement (87.62% vs. 87.60%).

Ablation Study

We evaluated the effectiveness of important components of SBLD method with GPT-3 model (3.6B) training for 20,000 steps. The results are shown in Table 6.

Variance-based scheduler. To show the effectiveness of variance based scheduler, we compared the performance of two different schedulers: Exponential scheduler and PWEGD scheduler. The result from Table 6 shows that PWEGD scheduler provide 3.8% extra throughput improvement and 0.02 extra loss reduction.

Block/Allow Lists. We test how Block/Allow List feature contribute to SBLD method by compare the performance of the same scheduler disabling and enabling Block/Allow List feature. The result shows Block/Allow List feature can provide 2.54% throughput improvement and 0.01 loss reduction.

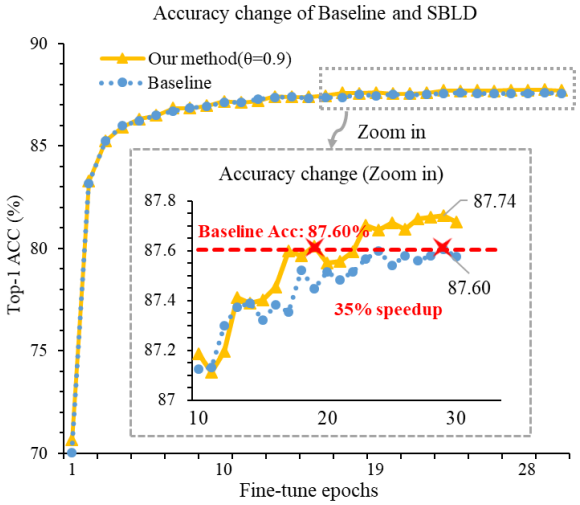

Figure 10: Fine-tune convergence comparison of baseline and SBLD on SwinV2-L (θ=0.9).

GPT-3 model 3.6B	Throughput (TFLOP/s)	Validation loss
Baseline	128.63	3.012
SBLD (Exp.)	138.81 (+10.13%)	3.111 (+0.099)
SBLD (PWEGD)	146.56 (+13.94%)	3.089 (+0.077)
SBLD (PWEGD + Block/Allow List)	**149.83** (+16.48%)	**3.080** (+0.068)

Table 6: Accuracy comparison of baseline and SBLD with different components on GPT-3 (3.6B)-L model.

Conclusion

Facing the challenge of training large transformer models efficiently, we propose SBLD method to accelerate transformer model training. We design pre-profiled layer-wise sensitivity to accurately describe the static importance of each transformer layer in the model. Besides, in order to reduce the negative impact of layer dropping on overall training convergence, we design a variance-based scheduler to find optimal keep probability for different training stage. Our experiment results show that the proposed method can improve both training throughput and model accuracy.

There are several new directions to explore and further improve our method in the future. First, we intend to employ a dynamic sensitivity measurement method to extend SBLD method to other models easily. Second, we plan to verify the efficiency of SBLD method on CNN and DNN-based networks.

References

Arici T.; Seyfioglu M. S.; Neiman T.; Xu Y.; Train S.; Chilimbi T.; Zeng B.; and Tutar I. 2021. MLIM: Vision-and-Language Model Pre-training with Masked Language and Image Modeling. arXiv:2109.12178.

Brown T. B.; Mann B.; Ryder N.; Subbiah M.; Kaplan J.; Dhariwal P.; Neelakantan A.; Shyam P.; Sastry G.; Askell A.; et al. 2020. Language models are few-shot learners. arXiv preprint arXiv:2005.14165.

Dehghani M.; Gouws S.; Vinyals O.; Uszkoreit J.; and Kaiser Ł. 2019. Universal Transformers. arXiv:1807.03819.

Dong Z.; Yao Z.; Gholami A.; Mahoney M.; Keutzer K. HAWQ: Hessian AWare Quantization of Neural Networks with Mixed-Precision. arXiv:1905.03696

Fan A.; Grave E.; and Joulin A. 2020. Reducing transformer depth on demand with structured dropout. In 8th International Conference on Learning Representations 2020. Addis Ababa, Ethiopia. April 26-30.

Hou L.; Pang R. Y.; Zhou T.; Wu Y.; Song X.; and Zhou D. 2022. Token Dropping for Efficient BERT Pretraining. arXiv:2203.13240.

Huang G.; Sun Y.; Liu Z.; Sedra D.; and Weinberger K. 2016. Deep networks with stochastic depth. arXiv:1603.09382

Iqbal T. and Iqbal M. Z. 2020. On the Mixture Of Weighted Exponential and Weighted Gamma Distribution. *Int. J. Anal. Appl.*. 18 (3): 396-408.

Li C.; Zhuang B.; Wang G.; Liang X.; Chang X.; and Yang Y. 2022. Automated Progressive Learning for Efficient Training of Vision Transformers. arXiv:2203.14509.

Liu Y.; Agarwal S.; and Venkataraman S. 2021. AutoFreeze: Automatically Freezing Model Blocks to Accelerate Fine-tuning. arXiv:2102.01386.

Liu Z.; Hu H.; Lin Y.; Yao Z.; Xie Z.; Wei Y.; Ning J.; Cao Y.; Zhang Z.; Dong L.; Wei F.; and Guo B. 2022. Swin Transformer V2: Scaling Up Capacity and Resolution. arXiv:2111.09883.

Liu Z.; Lin Y.; Cao Y.; Hu H.; Wei Y.; Zhang Z.; Lin S.; and Guo B. 2021. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. arXiv:2103.14030.

Nkemnole1 E. B. and Ikegwu E. M. 2020. Poly-Weighted Exponentiated Gamma Distribution with Application. *Journal of Statistical Theory and Applications*. 19(3): 446-459. doi.org/10.2991/jsta.d.201016.002.

Paperno D.; Kruszewski G.; Lazaridou A.; Pham Q. N.; Bernardi R.; Pezzelle S.; Baroni M.; Boleda G.; and Fernández R. 2016. The lambada dataset: Word prediction requiring a broad discourse context. arXiv:1606.0603.

Shoeybi M.; Patwary M.; Puri R.; LeGresley P.; Casper J.; and Catanzaro B. 2019. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. arXiv:1909.08053.

Wang J. and Zhang T. 2019. Utilizing Second Order Information in Mini-batch Stochastic Variance Reduced Proximal Iterations. *Journal of Machine Learning Research*. 20(42): 1−56.

Yao Z.; Dong Z.; Zheng Z.; Gholami A.; Yu J.; Tan E.; Wang L.; Huang Q.; Wang Y.; Mahoney M. W.; and Keutzer K. 2021. HAWQV3: Dyadic Neural Network Quantization. arXiv:2011.10680.

Zhang M. and He Y. 2020. Accelerating Training of Transformer-Based Language Models with Progressive Layer Dropping. arXiv:2010.13369.

Zhao J.; Wang Y.; Bao J.; Wu Y.; and He X. 2022. Fine- and Coarse-Granularity Hybrid Self-Attention for Efficient BERT. arXiv:2203.09055.

Zhu Y.; Kiros R.; Zemel R.; Salakhutdinov R.; Urtasun R.; Torralba A.; and Fidler S. 2019. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. arXiv:1506.06724.