BEYOND UNIFORM SCALING: EXPLORING DEPTH HETEROGENEITY IN NEURAL ARCHITECTURES

Akash Guna R.T. [*1], **Arnav Chavan**[*1,2], **Deepak Gupta**[2]
[1]Nyun AI [2]Transmute AI Lab (Texmin Hub), IIT (ISM) Dhanbad, India
arnav.chavan@nyunai.com, guptadeepak2806@gmail.com

ABSTRACT

Conventional scaling of neural networks typically involves designing a base network and growing different dimensions like width, depth, etc. of the same by some predefined scaling factors. We introduce an automated scaling approach leveraging second-order loss landscape information. Our method is flexible towards skip connections a mainstay in modern vision transformers. Our training-aware method jointly scales and trains transformers without additional training iterations. Motivated by the hypothesis that not all neurons need uniform depth complexity, our approach embraces depth heterogeneity. Extensive evaluations on DeiT-S with ImageNet100 show a 2.5% accuracy gain and 10% parameter efficiency improvement over conventional scaling. Scaled networks demonstrate superior performance upon training small scale datasets from scratch. We introduce the first intact scaling mechanism for vision transformers, a step towards efficient model scaling.

1 INTRODUCTION

Efforts to improve the performance of deep learning models are crucially dependent on scaling network architectures. Scaling typically involves repeating the same core structure as the base model in different dimensions. Prominent model families such as ResNet, BERT, GPT-3, and ViT (He et al., 2016; Devlin et al., 2019; Brown et al., 2020; Kolesnikov et al., 2021) have consistently expanded networks by adjusting depth, width, and layer dimensions, informed by empirical observations rather than a rigorous scientific foundation. A recent study by Wu et al. (2019) introduced an innovative width expansion approach, leveraging a computationally efficient Hessian approximation to identify neurons linked with saddle points for growth. However, this method, limited to width expansion that disrupts skip connections in linear layers, restricting its applicability to state-of-the-art transformers.

Existing attempts at scaling depth, as in Net2Net(Chen et al., 2015), uniformly increase depth throughout the network. We contend that such uniform scaling may be sub-optimal, as different network regions may require distinct scaling proportions. Notably, current approaches lack provisions for non-uniform depth scaling of neural networks.

This paper breaks away from uniform network scaling and explores depth heterogeneity for non-uniform scaling of neural architectures. Our approach embraces varied depth allocations across neurons within the same layer, allowing for adaptive and efficient utilization of network resources. Through experimental comparisons, we demonstrate that our method outperforms conventional scaling methods, achieving an accuracy gain exceeding 2.5% while utilizing 10% fewer parameters.

2 METHOD

The central idea underlying our scaling approach is to design a training-aware scaling strategy at the individual neuron level; hence, it is crucial to carefully select neurons and employ a viable scaling technique. **The core of our method lies in escaping saddle points of the loss landscape through localized scaling of individual neurons rather than the whole layer.** Plateaus with small curvature surround saddle points leading to slow convergence (Dauphin et al., 2014), hence eliminating saddle points via localized scaling enables faster convergence. Our approach selects neurons based on a pre-defined criterion for localized scaling. During the localized scaling, new neurons are added

[*]Equal contribution.

to a pseudo-layer and the output is then projected back to the selected neurons via individual skip connections. More details on the generic idea are presented in Appendix A.

Choosing the right neurons is crucial for the efficient scaling of the network architectures. We achieve this through the identification of neurons with the smallest negative eigenvalues derived from a Hessian approximation introduced by Wu et al. (2019). Existence of both positive and negative eigenvalues have been proved to contribute to the saddle points of the loss landscape (Alain et al., 2019). This selection criterion tends to accelerates the shifting of negative eigenvalues towards zero to escape saddle points, a phenomenon that naturally occurs during neural network training (Sagun et al., 2017).

Finally, ensuring function preservation during scaling is critical (Chen et al., 2015). Adding neurons without modifying existing neuron weights and biases ensures that even intermediate functions are preserved. To achieve such function preservation, we incorporate a strategy of adding two neurons with the same magnitude but opposite polarities for each selected neuron. This preserves the overall function without altering the weights of the scaled neurons while handling complex network topologies, like skip connections successfully. The mathematical formulation is detailed in Appendix B. **In summary, our neuron scaling method involves the selection of neurons with minimal negative eigenvalues, accommodating skip connections, and ensuring function preservation by introducing paired neurons with opposite polarities.** This innovative approach contributes to the scalability of transformers at a neuron level, paving the way for more effective and robust training procedures.

3 Experiments

For the experiments presented in this paper, we use ImageNet100 dataset, obtained through random sampling of 100 classes from the Imagenet1K (Deng et al., 2009). Next, we build a base model which will eventually be used for scaling using our approach. For the base model, we shrink DeiT-S (Touvron et al., 2021) by reducing the intermediate hidden dimension of MLP and MHSA modules by 50%. We train the base model for warmup period of 50 epochs.We scale neurons that fit our selection criteria every 30 epochs. Table 1 shows that our method was able to perform better than DeiT-S with 28 % less parameters and was able to gain over a 2.5 % accuracy gain with 10 % less parameters than DeiT-S. Our scaled networks were resilient towards overfitting. In Table 2, we show that the our network scaled on ImageNet100 works significantly better at training CIFAR-100(Krizhevsky et al., 2009) from scratch than DeiT-S. Our intuition is that DeiT-S has overfitted CIFAR100. This behaviour of Deit-S would likely translate to training small scale datasets from scratch.

Scaling	Base Param. (M)	Final Param. (M)	Base FLOPs (G)	Final FLOPs (G)	Top-1	Top-5
Homogeneous	21.7	21.7	4.6	4.6	77.80	93.16
Heterogeneous	11.0	15.6	2.3	3.1	79.16	94.00
Heterogeneous	11.0	19.4	2.3	3.9	80.36	94.58

Table 1: Performance of the proposed scaling method to scale DeiT-S on ImageNet100 dataset.

Model	Param. (M)	FLOPs (G)	Top-1	Top-5
Deit-S (Homogeneous)	21.7M	4.6	58.9	78.9
Deit-S (Heterogeneous)	19.4M	3.9	78.1	95.0

Table 2: Performance on our scaled DeiT-S on CIFAR100 upon training from scratch.

4 Conclusion

In this paper, we presented a novel neural architecture scaling approach to scale modern neural architectures efficiently through localized and non-uniform scaling of neurons. Through multiple experiments, we have demonstrated the efficacy of our approach, and we believe this work will pave way for future research towards building large-scale efficient neural architectures through proposed scaling of smaller networks.

URM Statement

The authors acknowledge that the first author of this work meets the URM criteria of ICLR 2024 Tiny Papers Track.

References

Guillaume Alain, Nicolas Le Roux, and Pierre-Antoine Manzagol. Negative eigenvalues of the hessian in deep neural networks. *arXiv preprint arXiv:1902.02366*, 2019.

Tom Brown et al. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020.

Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Advances in neural information processing systems*, 27, 2014.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Alexander Kolesnikov, Alexey Dosovitskiy, Dirk Weissenborn, Georg Heigold, Jakob Uszkoreit, Lucas Beyer, Matthias Minderer, Mostafa Dehghani, Neil Houlsby, Sylvain Gelly, Thomas Unterthiner, and Xiaohua Zhai. An image is worth 16x16 words: Transformers for image recognition at scale. 2021.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Levent Sagun, Leon Bottou, and Yann LeCun. Eigenvalues of the hessian in deep learning: Singularity and beyond, 2017.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10347–10357. PMLR, 18–24 Jul 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Lemeng Wu, Dilin Wang, and Qiang Liu. Splitting steepest descent for growing neural architectures. *Advances in neural information processing systems*, 32, 2019.

A Workflow Description: Exploring Depth Heterogeneity

Our aim is to scale neural networks through depth heterogeneity. To achieve the scaling, we start from transformers with reduced width at intermediate layers. We reduce width of only intermediate layers to leave skip connections unaffected. We scale neurons by adding new neurons as skip connections to neurons selected to scale. Figure 1 displays the basic workflow of our scaling technique.

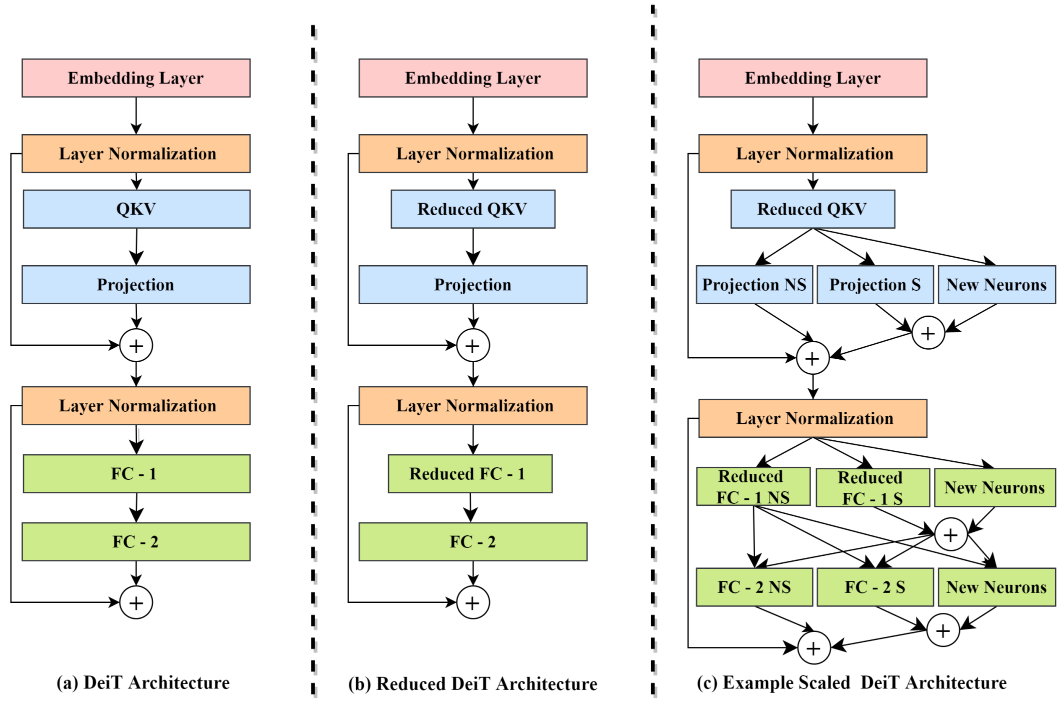

Figure 1: The Basic Workflow of the Proposed Scaling Technique. (a) shows the regular DeiT architecture. (b) shows a reduced DeiT architecture where parameters are reduced from intermediate layers to form bottlenecks. (c) shows an example scaled DeiT architecture grown from a reduced DeiT architecture.We scale only selected neurons (S) for scaling through skip connections and do not scale other neurons (NS) present in the layer. Our scaling technique is applicable to QKV, Projection and Fully Connected layers of DeiT.

B Mathematical Description

B.1 Using Hessian to Identify the Right Neurons

Neural networks aim to escape saddle points to reach local minima (Dauphin et al., 2014; Sagun et al., 2017). Based on this motivation, we use the identification of saddle points as a criterion for scaling the network. We select neurons using Hessian to identify the right neurons to scale. The second-order derivative of the activation value represented by a neuron with respect to the loss is a Hessian matrix. It monitors the direction of the gradients, the first-order derivative. Therefore, we could identify saddle points upon analyzing the magnitude of the Hessian's eigenvalues. A saddle point exists when a Hessian possesses a mix of positive and negative eigenvalues. The magnitude of the negative eigenvalues provides information on the steepness of the saddle point.

Despite the Hessian's ability to effectively identify saddle points, they are costly to compute. Wu et al. (2019) utilized a compute-efficient Hessian approximation called the splitting matrix to scale neurons by splitting existing neurons. Let the loss computed over the activation value σ of a neuron, L(σ), be $\phi(\sigma(X))$ where ϕ is the loss with respect to the σ for the given input X. The Hessian for a

σ is given by

$$Hessian(\sigma) = \phi^{''}(\sigma(X))\sigma^{'}(X)^T\sigma^{'}(X) + \phi^{'}(\sigma(X))\sigma^{''}(X) \tag{1}$$

where $(.)^{'}$ represents first order derivative and $(.)^{''}$ represents second order derivative. Computing $\phi^{''}$ for a σ_i requires computing all the preceding gradients $\{\sigma^{'}_n, \sigma^{'}_{n-1}, ..., \sigma^{'}_{i-1}\}$ which constitutes the bulk of computation required for Hessian calculation. Omitting $\phi^{''}(\sigma(X))\sigma^{'}(X)^T\sigma^{'}(X)$ from computation was still able to produce a good enough approximation (Wu et al., 2019) and is known as splitting matrix. The splitting matrix is formally defined as

$$SplittingMatrix(\sigma) = \phi^{'}(\sigma(X))\sigma^{''}(X) \tag{2}$$

We use the eigenvalues from the splitting matrix to identify neurons with saddle points that are suitable to scale.

B.2 Selecting Neurons to Scale

Our base network is a DeiT-S with reduced parameters, which we scale to 20 million parameters during training. We train that network for an *Intial Warmup* period of 50 epochs. Starting from the 50th epoch, we scale the network every 30 epochs. We denote this interval as the *Scaling Interval*. Neurons with the smallest negative eigenvalues are selected for scaling during each splitting interval.

We select the neurons until the added neurons exhausts a *Parameter Budget*. Parameter Budget is defined as the expected count of newly added neurons at each scaling interval. Since, we are reducing the parameters from both MHSA and MLP blocks we scale neurons present in both these blocks. The resulting scaled network add parameters to both MLP and MHSA blocks . We use a *Layer Threshold* to set a minimum bar for the number of eligible neurons a layer must posses to be scaled. This ensures that sparse neuron scaling is prevented. We set the layer threshold to be 60 neurons for scaling a Deit-S transformer.

B.3 Scaling neurons

We scale transformers at the neuron level. Due to extensive usage of skip connections between transformer blocks, existing neuron level scalers like Chen et al. (2015); Wu et al. (2019) are not directly applicable. We overcome this issue by adding neurons in a manner that mimics skip connections. We add outputs of added neurons with their existing counterparts without altering the existing output dimensional space of the transformer blocks. Our architecture initializes added neurons such that existing neuron outputs and weights remain unchanged, ensuring function preservation (Chen et al., 2015).

Since the widely adopted transformer architecture (Vaswani et al., 2017) is constructed using linear layers, we design our scaling technique around linear layers. Let the output of a layer that consists neurons selected to be scaled be $O_L(I) = W_LI + B_L$ for a given input I where W_L and B_L are the weights and bias of the layer respectively. Our goal is to expand a subset of neurons N_S from a layer consisting N_L neurons without disturbing the weightsW_S and bias B_S. To achieve this, we perform scaling by adding two neurons for each selected neuron with equal weights W_A and biases B_A but opposite polarities. W_A and B_A are W_S and B_S scaled by a *scaling factor* of 0.2. This initialization helps in preserving W_S and B_S while ensuring the flow of gradients. The equation illustrating how our scaling mechanism by adding a skip connection between new neuron outputs and existing outputs of selected neurons is given by:

$$O^{'}_S = O_S + \underbrace{GeLU(O_{A+} + O_{A-}) + O_{A+} + O_{A-}}_{NeuronScaling} \tag{3}$$

where O_S and $O^{'}_S$ are outputs of a neuron from N_S before and after scaling. Further, GeLU(.) adds non-linearity to the outputs of newly added neurons $(O_{A+}, O_{A\_})$. When initialized, the addition between O_{A+} and O_{A-} becomes 0 leaving W_S and B_S unmodified. We have skip connections from O_{A+} and $O_{A\_}$ to ensure proper gradient flow during backpropagation when newly intialized.

B.4 Proof of function preservation upon Initialization

Here, we present the proof that our scaling mechanism preserves functions during initialization from a layer perspective. Let us denote newly added positive and negative neurons collectively as linear layers (L_{A+} and L_{A-}). L_{A+} and L_{A-} differ by polarity of their weights and biases. The outputs of L_{A+} and L_{A-} is denoted as O_{A+} and O_{A-} respectively and are defined as

$$O_{A\pm} = (\pm W_S)I + (\pm B_S) \tag{4}$$

Let S denote the sum of O_{A+} and O_{A-}

$$S = O_{A+} + O_{A-} \tag{5}$$
$$S = W_S I + B_S - W_S I - B_S \tag{6}$$
$$S = W_S(I - I) + B_S(Id - Id) \tag{7}$$
$$S = W_S(Z^I) + B_S(Z^{Id}) \tag{8}$$
$$S = Z^S \tag{9}$$

where Z^α is a null matrix of α dimensions and Id is the identity matrix. Therefore, S becomes equal to a null matrix of the same dimensionality. When we substitute S in Equation 3 we get

$$O_S^{'} = O_S + GeLU(S) + S \tag{10}$$
$$O_S^{'} = O_S + GeLU(Z^S) + Z^S \tag{11}$$

Since $GeLU(Z^S) = Z^S$, $O_S^{'} = O_S$. Therefore, we have shown that functions are preserved upon initialization.

C Experiments: Configuration Details

C.1 Hyperparameters

For all experiments using ImageNet-100 we utilize a batch size of 1024 equally distributed among 4 NVIDIA L4 GPUs. We train all networks for 300 epochs and adopt the same training hyper-parameters as in Touvron et al. (2021) unless specified explicitly. For the CIFAR-100 experiments we set batch size to be 512 (/4 GPUs), gradient clipping to 1.0, and weight decay to 0.0001. .

C.2 Identifying the ideal base model

Since we scale DeiT transformers with reduced parameters, it is important to ideally reduce parameters to maximize effectiveness of scaling. We reduce parameters from the base network by reducing the width of intermediate layers, forming bottlenecks. We reduce parameters from QKV layers (ATTN) and the first fully-connected layer of the MLP block (FC). Other layers have skip connections therefore, width cannot be modified. We performed a grid search to identify the best ratio at which parameters could be reduced from ATTN and FC layers. We tried reducing the width of ATTN and FC layers at equal proportions and reducing the width of only ATTN or FC Layers. Table 3 shows the result of the grid search. Reducing the width by half at both ATTN and FC layers produced the best results.

Model	Base Param. (M)	Grown Param. (M)	Base FLOPs (G)	Grown FLOPs (G)	FC ↓	ATTN ↓	Top-1	Top-5
Deit-S	11.0	19.4	2.3	3.9	/2	/2	80.36	94.58
Deit-S	11.4	19.7	2.4	4.1	/4	/1	78.64	93.88
Deit-S	17.0	19.4	3.4	3.8	/1	/4	79.44	93.36
Deit-S	21.7	21.7	4.6	4.6	/1	/1	77.80	93.16

Table 3: Effect of base network reduction on final performance on the ImageNet100 dataset.

C.3 Identifying the appropriate scaling interval

To explore the ideal splitting interval, we searched across a range of candidate intervals {10,20,30,50} to scale a reduced Deit-S transformer. We scale the DeiT-S transformer by reducing intermediate layers in MLP and MLHA blocks by 50 % in a 1:1 ratio, resulting in a base network with 11 million parameters. We then grew all the models to 20 million parameters with an $\pm.5$ million allowed fluctuation in the final parameter count. Our experimental investigation revealed that scaling DeiT-S at a regular interval of 30 epochs yielded optimal performance while possessing the least amount of parameters. 4 shows the results of the experimentation.

Grown Param. (M)	Grown FLOPs (G)	Splitting Interval	Top-1	Top-5
20.4	4.2	10	79.78	94.36
20.0	4.1	20	79.44	94.04
19.4	3.9	**30**	**80.36**	**94.58**
19.9	4.1	50	79.94	93.92

Table 4: Results on ablating scaling interval for a DeiT-S transformer trained on ImageNet100 dataset. Scaling Interval is the interval between two scalings.

D Analysis of Eigenvalues to study Saddle Points

Examining the presence of saddle points at the start and end of training helps assess the effectiveness of our scaling method. We plot negative eigenvalues to assess the presence of saddle points and their steepness. Negative eigenvalue plots for a QKV layer from the MHSA block and the first FC layer(FC1) from the MLP block are shown in Figure 2. The plots show the presence of saddle points at both the start and end of the training, but the magnitude of negative eigenvalues is very close to 0 towards the end of the training, denoting that the remaining saddle points are shallow. These plots show that the scaled transformer successfully escaped steep saddle points.

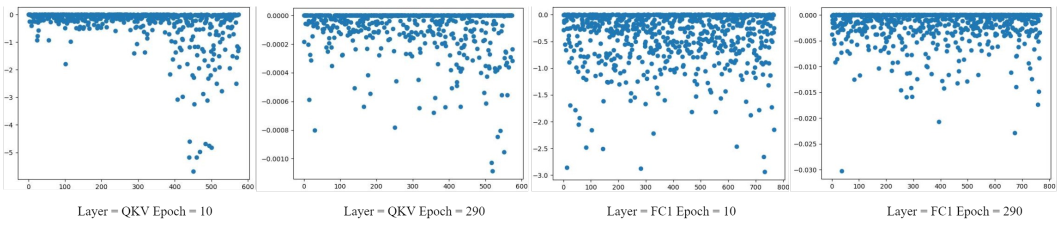

Figure 2: Plots showing negative eigenvalues of neurons in QKV and FC1 layers of the first transformer block. Each neuron in the X-axis has its magnitude shown in the Y-axis. (Zoom to view X and Y axes).