When To Grow? A Fitting Risk-Aware Policy for Layer Growing in Deep Neural Networks

Haihang Wu*, Wei Wang, Tamasha Malepathirana, Damith Senanayake, Denny Oetomo, Saman Halgamuge

Department of Mechanical Engineering, The University of Melbourne
haihangw@student.unimelb.edu.au

Abstract

Neural growth is the process of growing a small neural network to a large network and has been utilized to accelerate the training of deep neural networks. One crucial aspect of neural growth is determining the optimal growth timing. However, few studies investigate this systematically. Our study reveals that neural growth inherently exhibits a regularization effect, whose intensity is influenced by the chosen policy for growth timing. While this regularization effect may mitigate the overfitting risk of the model, it may lead to a notable accuracy drop when the model underfits. Yet, current approaches have not addressed this issue due to their lack of consideration of the regularization effect from neural growth. Motivated by these findings, we propose an under/over fitting risk-aware growth timing policy, which automatically adjusts the growth timing informed by the level of potential under/overfitting risks to address both risks. Comprehensive experiments conducted using CIFAR-10/100 and ImageNet datasets show that the proposed policy achieves accuracy improvements of up to 1.3% in models prone to underfitting while achieving similar accuracies in models suffering from overfitting compared to the existing methods.

Introduction

Deep neural networks (DNNs) have significantly advanced the state of the art in various computer vision tasks, including image classification (He et al. 2016; Dosovitskiy et al. 2021), object detection (Fang et al. 2021), and image segmentation (Kirillov et al. 2023). For instance, with a model size 10 times larger than ResNet-50 (He et al. 2016), the pioneering convolutional neural network ConvNext (Liu et al. 2022) achieves a substantial improvement in classification accuracy on the ImageNet dataset (Deng et al. 2009) compared to ResNet. However, this improvement comes at the expense of approximately 50 times more computational burden (Liu et al. 2022). Consequently, the development of efficient training methods for DNN becomes imperative.

Existing works demonstrate that neural growth can significantly reduce training time with minimal impact on accuracy (Li et al. 2022; Chang et al. 2018). For instance, studies (Chang et al. 2018; Dong et al. 2020) show neural growth can save around 40% training time for deep ResNet on CIFAR datasets with minor accuracy loss (Krizhevsky 2009), and a similar finding has also been observed in vision transformer growth (Li et al. 2022). Although the study of neural growth encompasses various dimensions (Maile et al. 2022; Maile, Luga, and Wilson 2022), such as initialization of new neurons/layers, when to grow (growth timing), and where to grow, these works predominantly focus on the initialization of new neurons or layers.

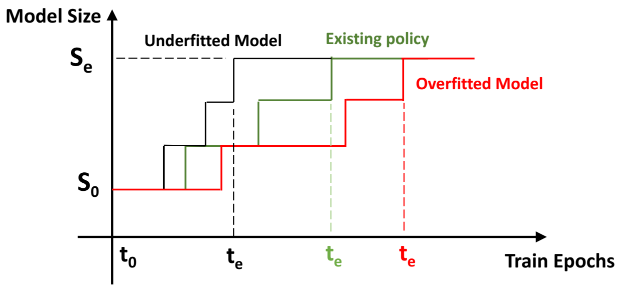

Figure 1: When to grow policy. To expedite the training process of the target large model with size S_e, neural growth grows the initial small model from size S_0 at time t_0 into S_e at time t_e. For the prevailing conventional strategies (illustrated by the green curve), t_e is not influenced by overfitting or underfitting risks directly, and these strategies may fail to address under/over fitting risks effectively. By contrast, our fitting risk-aware policy accelerates growth (black curve) by reducing t_e to mitigate underfitting risks when the target model exhibits underfitting tendencies. Conversely, it decelerates growth (red curve) by extending the t_e for models displaying signs of overfitting, addressing the overfitting risk.

Few studies (Wen et al. 2020; Dong et al. 2020) investigate the "when to grow" policy for neural growth. Two major policies are periodic growth which adds layers or neurons at regular intervals, and convergent growth which adds layers only when the current model has converged. In this study, we investigate the impact of neural growth on the model accuracy and then develop a "when to grow" policy that considers this impact.

Our investigation reveals that neural growth inherently

*Corresponding author.

provides a regularization effect on the final model, whose strength is controlled by the growth timing policy. Figure 2 (b) shows that neural growth reduces the average training epochs when compared with the vanilla method without neural growth. Reduced exposure to training data prevents the model from memorizing irrelevant information (noise) in the training dataset, potentially applying regularization to the model. Moreover, the regularization strength is controlled by the average training epochs or the growth timing policy. Although this regularization effect mitigates the overfitting risk by reducing learned noise, the model might not capture sufficient signals from the training dataset, particularly when it underfits the data due to this regularization effect. Nevertheless, existing "when to grow" policies disregard this matter due to their lack of consideration for the neural growth-induced regularization effect. Table 6 shows that existing growth timing policies (periodic growth and convergent growth) achieve slightly better or similar accuracy with shorter training time on the overfitting CIFAR10/100 dataset compared to the vanilla method without neural growth. However, they exhibit a significant accuracy decline on the underfitting ImageNet dataset compared to the vanilla method.

We introduce a novel **F**itting **R**isk-**A**ware **G**row policy (FRAGrow) to tackle this concern. This policy evaluates underfitting and overfitting risks at each growth phase and dynamically adjusts the growth timing to mitigate these risks. As Figure 1 shows, our policy accelerates growth to introduce milder regularization on the final model when dealing with underfitting models. Conversely, when overfitting is detected, the policy slows down the growth rate, thereby applying stronger regularization to the final model. By doing so, our "when to grow" policy effectively addresses the risks of overfitting and underfitting. Experimental results demonstrate that our approach achieves superior accuracy compared to existing policies, albeit with a trade-off in training time.

In summary, our contributions are:

- We identify that neural growth inherently exhibits a regularization effect, the intensity of which is governed by the growth timing.
- Based on this observation, we propose a fitting risk-aware policy that dynamically adjusts the growth timing by evaluating fitting risks with the proposed overfitting risk level.
- Compared to existing approaches, FRAGrow avoids significant accuracy drops in both overfit and underfit cases with a reasonable trade-off in training time.

Related Work

Training Acceleration via Neural Growth

Most research works that investigate efficient training via neural growth focus on how to initialize new neurons/layers. (Aosong and Panda 2020; Li et al. 2022; Dong et al. 2020). An early study on new neuron initialization employs the random initialization (Istrate et al. 2018). To accelerate the training of the large neural net, new neurons are initialized via function-preserving transformation to preserve the function and knowledge of the smaller net (Chen, Goodfellow, and Shlens 2016). This function preservation initialization approach is then extended to initialize the new layers or blocks for neural growth in depth (Wei et al. 2016; Wei, Wang, and Chen 2019, 2021). To further speed up training, the new weights are initialized to not only preserve the small net's knowledge but also maximize the gradient of new weights (Evci et al. 2022). However, research works also report that function initialization may not outperform the random initialization in terms of the final accuracy (Li et al. 2022; Wen et al. 2020). Another line of works initializes the new neurons by copying the weights from their neighboring neurons (Chang et al. 2018; Dong et al. 2020) or the historical ensemble of these neighboring weights (Li et al. 2022). This simple method proves effective performance in ResNet (He et al. 2016) and vision transformers (Li et al. 2022), and is therefore utilized in our research.

Studies (Li et al. 2022; Chang et al. 2018) also investigate "where to grow" policies. Previous works choose to either interpolate new layers in between the old ones (Chang et al. 2018; Dong et al. 2020) or stack the new layer after the old ones (Gong et al. 2019). A recent approach "elastic supernet" (Li et al. 2022) is built upon the interpolation method, and grows one layer per time by optionally activating new layers and selecting the layer with the highest accuracy. Nevertheless, these methods either lead to a doubling of the layer count (Chang et al. 2018; Dong et al. 2020) or need additional computation cost to search the growth location (Li et al. 2022) each time the network undergoes expansion. Another line of works simply trains layers sequentially by appending a new layer right to the old layers (Hinton and Osindero 2006; Nøkland and Eidnes 2019; Belilovsky, Eickenberg, and Oyallon 2019) for higher training efficiency, and this method is employed in our study as the "where to grow" policy.

Despite these, few studies (Wen et al. 2020) investigate the "when to grow" policy. Two main policies used in the literature are the periodic growth (Li et al. 2022; Chang et al. 2018) and the convergence policy (Istrate et al. 2018; Belilovsky, Eickenberg, and Oyallon 2019). Periodic growth employs a fixed growth speed, and convergent growth's speed is mainly determined by the model's convergence speed. Without explicitly considering the risks of overfitting and underfitting, these policies might accelerate the training process but could result in a significant reduction in accuracy.

Neural Architecture Search by Neural Growth

Neural growth is also used for neural architecture search (Liu, Wu, and Wang 2019; Wu et al. 2020b,a; Zhu et al. 2020; Wen et al. 2020). Firefly (Wu et al. 2020a) and NeSt (Dai, Yin, and Jha 2019) grow neurons based on the largest initial gradient. Besides growth location studies, another study (Wen et al. 2020) also explores growth timing and learning rate scheduling in neural architecture search, and reveals that neural growth shows a preference for fast periodic growth and a constant large learning rate.

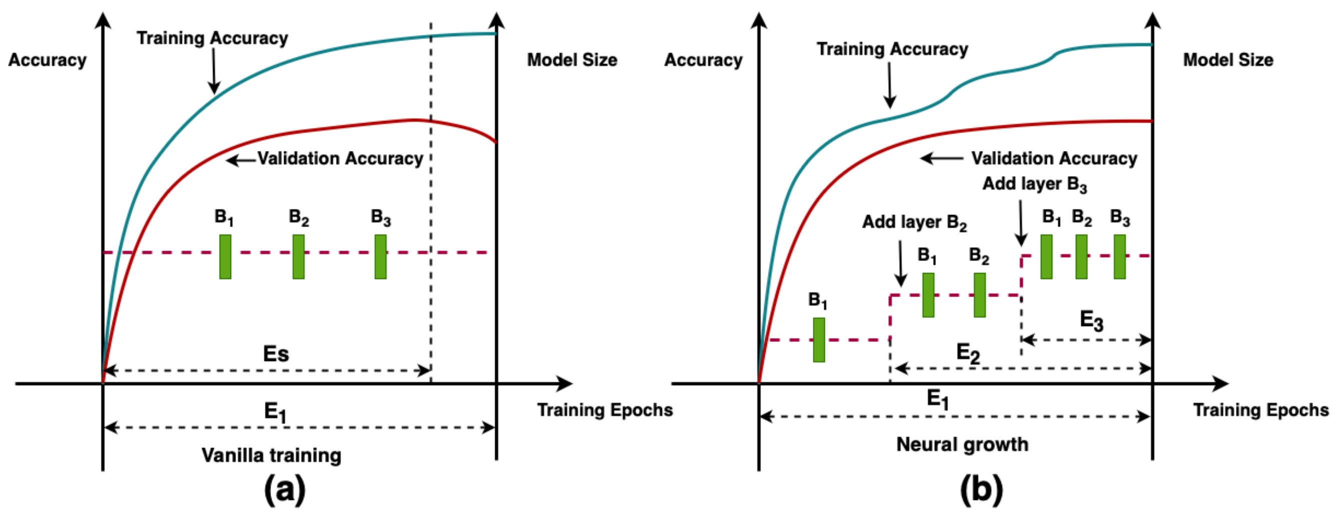

Figure 2: Comparison of vanilla training and neural growth. In standard training (vanilla), all layers undergo training for E_1 epochs. However, in the context of neural growth, only layer B_1 is trained for E_1 epochs, while the majority of layers (B_2 and B_3) are subjected to reduced training epochs (E_2 and E_3 respectively).

Method

The Regularization Effect of Neural Growth

Neural growth exhibits a regularization effect. As Figure 2 (a) shows, the vanilla method trains the final large model with layers B_1, B_2, and B_3 directly for E_1 epochs. By contrast, neural growth, demonstrated by Figure 2 (b), starts with layer B_1 and sequentially inserts and trains layers B_2 and B_3 for E_2 and E_3 number of epochs respectively. Due to reduced training epochs (E_2 and E_3) for the majority of the layers (B_2 and B_3), their final weight values may be closer to their initial weight values compared to vanilla training, resulting in the regularization effect on the final model. As the regularization effect may increase the training error (Goodfellow, Yoshua, and Aaron 2016), one may expect the model trained with neural growth to have higher training error than the model trained by vanilla method without neural growth, and this phenomenon is observed in Table 1.

The neural growth-induced regularization strength is determined by the average training epochs $\bar{E}$ in Equation 1 where n is the number of added blocks. A smaller $\bar{E}$ may have a stronger regularization effect as shown in Table 1 where the slow growth approach, with larger growth interval and smaller $\bar{E}$, yields higher training errors due to its relatively stronger regularization effect compared to the fast growth approach. This finding is consistently reaffirmed by Figure 3 where a higher growth speed (larger $\bar{E}$) consistently leads to decreased training errors, further highlighting the influence of $\bar{E}$.

$$\bar{E} = \frac{1}{n}\sum_{i=1}^{n} E_i \tag{1}$$

The impact of the regularization effect from neural growth on the model's generalizability can vary depending on the type of fitting risk present. If the final model overfits the training dataset, the regularization effect from neural growth may improve the model's generalizability. This can be confirmed by the CIFAR100 results in Table 1 where VGG and ResNet exhibit overfitting on the CIFAR100 dataset, with a low training error (approximately 1%) but a significantly higher test error (over 25%). In this context, the regularization effect from neural growth, as shown by slow growth in ResNet and fast growth in VGG, slightly enhances test accuracy. However, when the final target model underfits the training dataset, the regularization effect from neural growth may harm the model's learning capability. For example, Table 1 shows both ResNet and VGG models tend to underfit the ImageNet dataset evidenced by the high training error. In this case, the test errors of neural growth approaches (fast and slow growth) are higher than the vanilla method without neural growth. In summary, an appropriate "when to grow"

Model	Method	CIFAR100		ImageNet	
		Test	Train	Test	Train
ResNet	Slow growth	**28.91**	6.24	24.73	21.68
	Fast growth	29.33	1.82	24.29	18.48
	Vanilla	29.56	**1.12**	**24.14**	**17.82**
VGG	Slow growth	27.34	7.40	25.70	29.47
	Fast growth	**26.86**	1.16	24.44	25.97
	Vanilla	26.96	**0.32**	**24.15**	**25.05**

Table 1: The regularization effect in neural growth. Compared to slow growth, fast growth employs a smaller growth interval for the neural network, leading to a more rapid increase in model size and larger average training epochs $\bar{E}$ for fast growth. By contrast, the vanilla method trains the large final model directly without neural growth. We report the test error (%) and the training error (%).

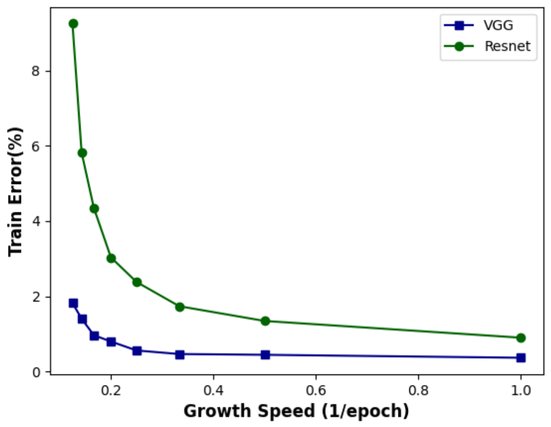

Figure 3: Regularization effect of neural growth on CIFAR100 dataset. In this experiment, the network is grown periodically, and new network layers are added after each growth interval. The growth speed is measured by the inverse of the growth interval. Increasing growth speed leads to larger average training epochs $\bar{E}$ and a weaker regularization effect, resulting in smaller training errors.

Model	CIFAR100		TinyImageNet		ImageNet	
	Train	ORL	Train	ORL	Train	ORL
ResNet	1.12	31.35	0.01	45.69	17.82	10.76
VGG	0.32	36.61	0.04	37.86	25.05	4.30

Table 2: The overfitting risk level (ORL) as the predictor for over/underfitting. We report the training error (%) and validation error (%).

policy should be tailored to the fitting risk type.

When to Grow Policy

A "when to grow" policy determines the growth timing $t_1, ..., t_n$ for n added blocks, which in turn influences the average training epochs $\bar{E}$ and controls the regularization strength. Therefore, it has the potential to mitigate over/under fitting risk of the model by adjusting the growth timing and regularization strength of the model. However, it is necessary to predict the fitting risk type (overfit or underfit) to decide the appropriate growth timing.

To predict overfitting and underfitting, the overfitting risk level (ORL) in equation 2 is used, which involves comparing the train accuracy and validation accuracy (Bejani and Ghatee 2020). A large ORL suggests that the model may learn excessively irrelevant information from the training dataset, indicating potential overfitting of the model to the dataset. This is evidenced by the CIFAR100 and TinyImageNet (Wu, Zhang, and Xu 2017) (Table 2) where the models (ResNet and VGG) overfit these two datasets, with ORL values exceeding 25%. In contrast, lower ORL values signify good generalization and underfitting risk, demonstrated by ImageNet results in Table 2, with ResNet and VGG exhibiting around 20% training error and ORL below 12%. Thus, ORL during growth is a reliable predictor for identifying the overfitting and underfitting tendencies of the final model.

Algorithm 1: Deep Network Growth Algorithm

```
1:  Input: A seed shallow network $f_s$; A target deep neural
    network $f_d$; The number of training epochs $E_T$
2:  Initialization: The current growing network $f_c = f_s$;
3:  for $i = 1$ to $E_T$ do
4:    Train the model
5:    if $|f_c| < |f_d|$ then
6:      if When to grow criterion (Eq. 3) is met then
7:        Add a new layer at the growth location determined by the "where to grow" policy
8:        Initialize the new layer by the Initialization policy
9:      end if
10:   else
11:     finetune the target model $f_d$
12:   end if
13: end for
14: Output: A trained target deep neural network $f_d$
```

$$ORL = \text{train accuracy} - \text{validation accuracy} \tag{2}$$

We propose a simple "when to grow" strategy to mitigate under/over fitting risks based on ORL. It appends network layers after the dynamic growth interval I specified by Equation 3 where I_{max} is the maximum allowed growth interval, and α is a hyperparameter. To allow sufficient time for model finetuning, I_{max} is constrained by Equation 4, where E_T is the total train epochs, and E_F^{min} is the minimum finetuning epochs, and n is the number of added blocks. When the final model is likely to underfit the dataset via the indication of a small ORL, our "when to grow" policy Equation 3 reduces the growth interval for faster growth, and exerts weaker regularization on the final model. Conversely, if the overfitting risk of the final model on the dataset is detected, the Equation 3 increases the growth interval, resulting in slower growth and stronger regularization on the final model. Despite its simplicity with only one hyperparameter α, we will show its effectiveness in the experiments. A potential tradeoff is that our policy may require extended training time compared to periodic growth with a growth interval of I_{max}.

$$I = \frac{I_{max}}{1 + e^{\alpha - ORL}} \tag{3}$$

$$I_{max} = \frac{E_T - E_F^{min}}{n} \tag{4}$$

Overhead. The proposed "when to grow" policy incurs its main overhead from computing the validation accuracy of ORL in Equation 2. However, since the validation accuracy is computed only at the end of every epoch and the validation set size represents only 1% of the training dataset in our

Model	Learning rate	CIFAR100
ResNet	Constant	**28.91**
	Cosine Annealing	29.32
VGG	Constant	**27.34**
	Cosine Annealing	28.06

Table 3: Effect of learning rate schedule during neural growth. As with slow growth in Table 1, we grow the model periodically at random growth locations. Test error (%) is reported.

experiments, the overhead is negligible when compared to the overall training time.

Experiments

Setup

Datasets. Three datasets, CIFAR10, CIFAR100 (Krizhevsky 2009) and ImageNet (Deng et al. 2009), are utilized to evaluate the algorithms in this study. The ImageNet dataset encompasses 1,000 classes and comprises a total of 1.28 million training images which are divided into a training set with 1.27 million images and a validation set with 0.01 million images in our study. The original 50,000 validation images serve as the test set. The images are resized with the smaller edge adjusted to 256 pixels, and subsequently, a crop of size 224x224 is extracted from the resized image. In contrast, CIFAR10 and CIFAR100 are smaller-scale image datasets with 10 and 100 classes respectively, each consisting of 49,500 training images, 500 validation images, and 10,000 test images. The images in CIFAR datasets have a resolution of 32x32 pixels.

Seed Shallow Networks and Target Deep Models. Three representative convolution neural network architectures single-branch network-VGGNet (Simonyan and Zisserman 2014), and multiple-branch networks ResNet (He et al. 2016) and MobileNetV2 (Sandler et al. 2018) are used to evaluate the algorithms. These architectures are composed of multiple stages, wherein each stage stacks several blocks with identical output feature maps. For VGG, the notation VGG-1-1-1-1-1 is used, indicating a VGG model with one VGG block in each of the five stages. Similarly, for ResNet, the notation ResNet-2-2-2-2 is utilized, signifying a ResNet architecture with two residual blocks in each of the four stages. In this study, we investigate the growth of the seed shallow networks (VGG-1-1-1-1-2, ResNet-2-2-2-2 and MobileNetV2-1-1-1-1-1 architectures) into the target deep models (VGG-2-2-4-4-4, ResNet-8-8-8-8 and MobileNetV2-2-3-4-3-3 architectures). The study utilizes VGG-1-1-1-1-2 to avoid a slow convergence issue when the second block of the final stage is inserted.

Contenders. In this study, we utilize three baseline methods: periodic growth, convergent growth and lipgrow (Dong et al. 2020). As with the established convention of dividing the overall training procedure into multiple equidistant phases (Li et al. 2022; Tan and Le 2021), the periodic growth approach in this study involves inserting new layers at a fixed interval of I_{max} in Equation 4. The convergent growth method adds blocks only when the accuracy shows stagnation. Lipgrow (Dong et al. 2020) grows the model when the Lipschitz constant of the model exceeds a threshold value.

	CIFAR10/100	ImageNet
Initial learning rate	0.5/0.1/0.1	0.1
Batch size	128	256
Total train epochs E_T	180	120
Weight decay (10^{-4})	1/5/1	1/1/0.4
Optimizer	SGD	
Momentum	0.9	
Learning rate scheduling	cosine decay	
Weight initialization	He (He et al. 2015)	
Min finetuning epochs E_F^{min}	30 (Dong et al. 2020)	

Table 4: Training Hyperparameters: Triplet values correspond to ResNet, VGG, and MobileNetV2 (left to right), while the single value remains consistent across all architectures.

Implementation Details. For each growth, we append a new block to old blocks in a stage until this stage reaches its target capacity. We then proceed to the later stages until the model attains the target size. We initialize the new block by duplicating the weight values from its precedent block (Dong et al. 2020; Chang et al. 2018). If the precedent block is a downsampling block, the new block will be randomly initialized (Chang et al. 2018; He et al. 2015). We investigate two learning rate scheduling during growth phase: a constant large learning rate (Wen et al. 2020) and cosine annealing with restart (Loshchilov and Hutter 2017). Table 3 demonstrates that a constant large learning rate outperforms cosine annealing with restart, and thus constant large learning rate is employed (Wen et al. 2020) in this study. After the model is grown into the final model, it is finetuned by the cosine decayed learning rate without restart. We employed the standard data augmentation (horizontal flip and random cropping). The training hyperparameters are summarized in Table 4. The experiments are repeated 3 times.

Main Results

Table 5 compares our neural growth method with vanilla methods that train small or large models directly, demonstrating comparable accuracy with the large model baseline while requiring shorter training time. For the CIFAR10/100 dataset, our approach yields approximately 0.3% improvement in test error for ResNet and VGG, while reducing training time by about 20% for all models. This accuracy improvement may be attributed to the regularization effect of neural growth on the final target models, VGG-2-2-4-4-4, and ResNet-8-8-8-8, which tend to overfit the CIFAR10/100 dataset. Conversely, on the ImageNet dataset, we observe a slight accuracy drop of around 0.32%, yet the method saves approximately 10% of the training time compared to the vanilla approach that directly trains the large target model. Due to MobileNet's 7% higher training error and increased underfitting on ImagenNet than VGG and ResNet, our policy grows MobileNet faster, resulting in diminished train-

Model	Method	CIFAR10		CIFAR100		ImageNet	
		Test	Time	Test	Time	Test	Time
ResNet	Small	8.37	**39.23**	32.97	**36.95**	29.06	**58.79**
	Large	6.66	100.00	29.56	100.00	**24.14**	100.00
	Proposed	**6.32**	82.45	**29.14**	79.93	24.32	90.00
VGG	Small	8.27	**48.21**	31.12	**51.71**	31.01	**54.35**
	Large	6.22	100.00	26.96	100.00	**24.15**	100.00
	Proposed	**6.20**	81.65	**26.57**	84.00	24.39	88.76
MbNet	Small	7.32	**38.12**	27.49	**40.07**	36.97	**52.70**
	Large	**5.22**	100.00	23.95	100.00	**29.71**	100.00
	Proposed	5.60	79.42	**23.94**	73.82	30.25	93.75

Table 5: Main Results. The terms "small" and "large" refer to training a seed shallow network and a target deep model from scratch, respectively, without neural growth. MbNet means MobileNetV2. Test error (%) and normalized training time (%) are reported.

ing time savings on MobileNet. Nevertheless, Table 6 reveals that our accuracy drop is notably smaller than the accuracy reductions observed with the periodic growth (approximately 1.1%) and convergent growth (approximately 1.7%) strategies. Finally, it is noteworthy that the time-saving on the CIFAR10/100 dataset surpasses that on the ImageNet dataset due to our method's automatic detection of under-/overfitting risks, resulting in slower model growth for the overfitted CIFAR10/100 dataset than for the underfitted ImageNet dataset, and leading to more substantial time savings in the former.

Table 6 and Table 7 show the proposed method achieves superior accuracy compared to existing approaches in the case of underfitting while exhibiting similar accuracies in overfitting scenarios. On the overfitting CIFAR10/100 dataset, all methods except lipgrow (Dong et al. 2020) achieve comparable accuracy with the target large model, indicating their appropriate growth speed and regularization strengths through neural growth. On the underfitting ImageNet dataset, our method outperforms the strongest baseline method, periodic growth, by 0.47% and 1.31% for ResNet and VGG architectures, respectively. This improvement results from our method's ability to detect underfitting risks and increase growth speed accordingly, reducing the regularization impact of neural growth. In contrast, while periodic growth and convergent growth exhibit suitable growth speeds for the overfitting CIFAR10/100 dataset, their growth speeds prove too slow for underfitting ImageNet dataset, imposing excessive regularization on the model and leading to lower accuracies. The fast growth speed is particularly necessary for the VGG model on the ImageNet dataset due to the more pronounced underfitting risk compared to ResNet. Nevertheless, it should be noted that our method requires longer training time than periodic growth, as the growth interval in our method never exceeds that of periodic growth, resulting in faster growth speed and less training time saved compared to periodic growth.

Model	Method	CIFAR10		CIFAR100		ImageNet	
		Test	Time	Test	Time	Test	Time
ResNet	Periodic	6.58	102.33	29.29	98.62	24.79	93.39
	Conv	6.35	104.39	29.25	107.42	25.30	**86.59**
	Lipgrow	7.18	**67.22**	29.23	**96.09**	25.10	88.86
	Proposed	**6.32**	100.00	**29.14**	100 .00	**24.32**	100.00
	Vanilla	6.66	121.29	29.56	125.11	24.14	111.11
VGG	Periodic	6.40	92.23	26.73	93.02	25.70	92.40
	Conv	6.33	99.72	26.83	92.65	26.42	**77.61**
	Lipgrow	7.05	**75.92**	29.82	**83.26**	27.03	103.91
	Proposed	**6.20**	100.00	**26.57**	100.00	**24.39**	100.00
	Vanilla	6.22	122.48	26.96	119.05	24.15	112.66

Table 6: Comparison of when to grow policies on image classification tasks. Except for the vanilla method that trains the target large model without neural growth, other methods employ neural growth with different when-to-grow policies. Lipgrow (Dong et al. 2020) doubles blocks for each growth. Conv means convergent growth policy. Test error (%) and normalized training time (%) are reported.

Model	Method	CIFAR10		CIFAR100	
		Test	Time	Test	Time
MobileNetV2	Periodic	5.66	**95.11**	24.35	**99.67**
	Convergent	**5.50**	105.86	24.11	106.87
	Lipgrow	5.64	117.30	24.32	116.65
	Proposed	5.60	100.00	**23.94**	100.00
	Vanilla	5.22	132.92	23.95	123.83

Table 7: Comparison of when to grow policies on MobileNetV2 (Sandler et al. 2018). Except for the vanilla method that trains the target large model without neural growth, other methods employ neural growth with different when-to-grow policies. Lipgrow (Dong et al. 2020) doubles blocks for each growth. Test error (%) and normalized training time (%) are reported.

Ablation Study

We study two aspects of our method, aiming to answer the following questions: How does the choice of hyperparameters impact the performance of FRAGrow? How robust is our growth timing policy to growth order and initialization of new layers of neural growth?

Effect of Hyperparameter α. We investigate the impact of α in equation 3 that determines the growth interval dynamically. Results in Table 8 show that reducing α from 6 to 2 led to fluctuating test errors or a drop in accuracy (around 0.6% for ResNet on ImageNet). The accuracy decline is attributed to the slower growth speed with smaller α, resulting in a stronger regularization effect and reduced accuracy on underfitting ImageNet datasets. However, the slower growth with a smaller α saves more training time, as confirmed by VGG on CIFAR100. Therefore, an optimal α value should balance training efficiency and regularization strength, with

Model	α	CIFAR100		ImageNet	
		Test	Time	Test	Time
ResNet	2	**29.11**	99.04	24.86	**98.23**
	4 (Ours)	29.14	100.00	24.32	100.00
	6	29.20	**97.70**	**24.27**	100.90
VGG	2	26.75	**95.11**	**24.22**	101.20
	4 (Ours)	**26.57**	100.00	24.39	**100.00**
	6	26.89	110.58	24.32	99.17

Table 8: Effect of alpha. All neural growth policies are consistent with those described in the method section. Test error (%) and normalized training time (%) are reported.

Model	Method	CIFAR100		ImageNet	
		Test	Time	Test	Time
ResNet	Periodic	28.41	101.30	24.62	**98.20**
	Convergent	29.15	107.91	25.07	99.13
	Proposed	**28.39**	**100.00**	**24.52**	100.00
VGG	Periodic	27.00	**85.74**	26.19	94.85
	Convergent	27.00	95.10	26.32	**94.78**
	Proposed	**26.63**	100.00	**24.18**	100.00

Table 9: Effect of where to grow policies. The "when to grow," "how much to grow," and initialization policies remain consistent with those outlined in the method section. Test error (%) and normalized training time (%) are reported.

our experiments indicating that an α value of 4 performed well for the considered datasets.

Effect of Growth Order. To assess the influence of the "where to grow" policy, we replace the layerwise growth with the circulation growth (Wen et al. 2020) that iterates over the stages and append a new block to a stage whenever it is visited. The experimental results are presented in Table 9. The proposed approach exhibits robustness to the growth order and achieves comparable or higher accuracy compared to baseline methods (periodic growth and convergent growth). Particularly noteworthy is the experiment involving the VGG model on the underfitting ImageNet dataset, where the proposed method shows an accuracy improvement of approximately 2% over its contenders. This outcome reinforces the significance of increasing growth speed for underfitting models.

Effect of Initialization. To evaluate the impact of new layers initialization, we employ the moment growth (Li et al. 2022), involving copying the weights of new layers from the historical ensemble of its preceding layer. The results are presented in Table 10. Our method demonstrates comparable accuracy to baseline techniques on the overfitting CIFAR100 dataset while exhibiting approximately 1% higher average accuracy on the underfitting ImageNet dataset. This underscores the resilience of our method towards various initialization strategies. It should be noted that our proposed approach exhibits a marginal improvement in time efficiency compared to the baseline methods on the ImageNet dataset. This disparity arises due to the intrinsic requirement of the moment growth technique to calculate moving averages of historical weight values throughout each training step during model expansion, leading to additional computational overhead. As the proposed method grows the model fast on the ImageNet dataset, the associated extra computational burden becomes comparatively diminished, yielding a slightly reduced training time compared to the baseline methods.

Model	Method	CIFAR100		ImageNet	
		Test	Time	Test	Time
ResNet	Periodic	28.94	**97.47**	24.72	105.01
	Convergent	29.82	105.80	25.28	101.12
	Proposed	**28.80**	100.00	**24.65**	**100.00**
VGG	Periodic	26.85	93.43	26.15	106.74
	Convergent	26.86	**89.27**	26.52	105.01
	Proposed	**26.67**	100.00	**24.24**	**100.00**

Table 10: Effect of initialization. The "when to grow," "where to grow," and "how much to grow" policies remain in line with the method section's description. Test error (%) and normalized training time (%) are reported.

Conclusion

In this paper, we have investigated the growth timing policy to address under/overfitting risks. Specifically, we have discovered that neural growth induces a regularization effect on the target large model, with the regularization strength being governed by the "when to grow" policy. To address these risks effectively, we introduced the concept of overfitting risk level, allowing us to predict and manage the under/overfitting risks for the target large model by adjusting the growth speed accordingly. Through experimental evaluations on image recognition tasks involving both single-branch networks (VGG) and multiple-branch networks (ResNet), we demonstrated the efficacy of our proposed FRAGrow method. As part of future work, we intend to extend our method to more vision tasks, e.g., dense prediction tasks, and improve the training efficiency of these tasks.

Acknowledgements

This study was supported by the Melbourne Graduate Research Scholarship for the first author and the Australian Research Council grant DP220101035 for the first, third, fourth, and last authors. The research utilized the LIEF HPC-GPGPU Facility at the University of Melbourne, established with support from LIEF Grant LE170100200.

References

Aosong, F.; and Panda, P. 2020. Energy-efficient and Robust Cumulative Training with Net2Net Transformation. In *International Joint Conference on Neural Networks (IJCNN)*. ISBN 9781728169262.

Bejani, M. M.; and Ghatee, M. 2020. Convolutional Neural Network With Adaptive Regularization to Classify Driving Styles on Smartphones. *IEEE Transactions on Intelligent Transportation Systems*, 21(2): 543–552.

Belilovsky, E.; Eickenberg, M.; and Oyallon, E. 2019. Greedy Layerwise Learning Can Scale to ImageNet. In *ICML*.

Chang, B.; Meng, L.; Haber, E.; Tung, F.; and Begert, D. 2018. Multi-level Residual Networks from Dynamical Systems View. In *International Conference on Representation Learning*.

Chen, T.; Goodfellow, I.; and Shlens, J. 2016. Net2Net: Accelerating Learning via Knowledge Transfer. In *4th International Conference on Learning Representations*, 1–12.

Dai, X.; Yin, H.; and Jha, N. K. 2019. NeST: A Neural Network Synthesis Tool Based on a Grow-and-Prune Paradigm. *IEEE Transactions on Computers*, 68(10): 1487–1497.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *IEEE conference on computer vision and pattern recognition*, 248–255.

Dong, C.; Liu, L.; Li, Z.; and Shang, J. 2020. Towards Adaptive Residual Network Training: A Neural-ODE Perspective. In *International Conference on Machine Learning*, 2616–2626.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*.

Evci, U.; van Merriënboer, B.; Unterthiner, T.; Vladymyrov, M.; and Pedregosa, F. 2022. GradMax: Growing Neural Networks using Gradient Information. In *International Conference on Representation Learning*.

Fang, Y.; Liao, B.; Wang, X.; Fang, J.; Qi, J.; Wu, R.; Niu, J.; and Liu, W. 2021. You Only Look at One Sequence: Rethinking Transformer in Vision through Object Detection. In *NeurIPS*.

Gong, L.; He, D.; Li, Z.; Qin, T.; Wang, L.; and Liu, T.-Y. 2019. Efficient Training of BERT by Progressively Stacking. In *ICML*.

Goodfellow, I.; Yoshua, B.; and Aaron, C. 2016. *Deep learning*. MIT press.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2015 Inter, 1026–1034. ISBN 9781467383912.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-Decem, 770–778. ISBN 9781467388504.

Hinton, G. E.; and Osindero, S. 2006. A Fast Learning Algorithm for Deep Belief Nets. *Neural computation*, 18(7): 1527–1554.

Istrate, R.; Malossi, A. C. I.; Bekas, C.; and Nikolopoulos, D. 2018. Incremental Training of Deep Convolutional Neural Networks. In *CEUR Workshop Proceedings*, 1–7.

Kirillov, A.; Mintun, E.; Ravi, N.; Mao, H.; Rolland, C.; Gustafson, L.; Xiao, T.; Whitehead, S.; Berg, A. C.; Lo, W.-Y.; Dollár, P.; and Girshick, R. 2023. Segment Anything. In *ICCV*.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images. Technical report.

Li, C.; Zhuang, B.; Wang, G.; Liang, X.; Chang, X.; and Yang, Y. 2022. Automated Progressive Learning for Efficient Training of Vision Transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12486–12496.

Liu, Q.; Wu, L.; and Wang, D. 2019. Splitting steepest descent for growing neural architectures. In *Advances in Neural Information Processing Systems*.

Liu, Z.; Mao, H.; Wu, C.-Y.; Feichtenhofer, C.; Darrell, T.; and Xie, S. 2022. A ConvNet for the 2020s. In *CVPR*.

Loshchilov, I.; and Hutter, F. 2017. SGDR: Stochastic gradient descent with warm restarts. *ICLR*, 1–16.

Maile, K.; Luga, H.; and Wilson, D. G. 2022. Structural Learning in Artificial Neural Networks: A Neural Operator Perspective. *Transaction on machine learning research*.

Maile, K.; Rachelson, E.; Luga, H.; and Wilson, D. G. 2022. When, where, and how to add new neurons to ANNs. In *First Conference on Automated Machine Learning*.

Nøkland, A.; and Eidnes, L. H. 2019. Training Neural Networks with Local Error Signals. In *ICML*.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *CVPR*.

Simonyan, K.; and Zisserman, A. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*.

Tan, M.; and Le, Q. V. 2021. EfficientNetV2: Smaller Models and Faster Training. In *ICML*.

Wei, T.; Wang, C.; and Chen, C. W. 2019. Stable Network Morphism. In *International Joint Conference on Neural Networks (IJCNN)*, 14–19. ISBN 9781728120096.

Wei, T.; Wang, C.; and Chen, C. W. 2021. Modularized Morphing of Deep Convolutional Neural Networks: A Graph Approach. *IEEE Transactions on Computers*, 70(2): 305–315.

Wei, T.; Wang, C.; Rui, Y.; and Chen, C. W. 2016. Network Morphism. In *International conference on machine learning*, 564–572.

Wen, W.; Yan, F.; Chen, Y.; and Li, H. 2020. AutoGrow: Automatic Layer Growing in Deep Convolutional Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 833–841.

Wu, J.; Zhang, Q.; and Xu, G. 2017. Tiny ImageNet Challenge. Technical report.

Wu, L.; Liu, B.; Stone, P.; and Liu, Q. 2020a. Firefly Neural Architecture Descent: a General Approach for Growing Neural Networks. In *Advances in Neural Information Processing Systems*, 22373–22383.

Wu, L.; Ye, M.; Lei, Q.; Lee, J. D.; and Liu, Q. 2020b. Steepest Descent Neural Architecture Optimization: Escaping Local Optimum with Signed Neural Splitting.

Zhu, H.; An, Z.; Yang, C.; Hu, X.; Xu, K.; and Xu, Y. 2020. Efficient Search for the Number of Channels for Convolutional Neural Networks. In *International Joint Conference on Neural Networks (IJCNN)*. ISBN 9781728169262.