Deep Networks with Stochastic Depth

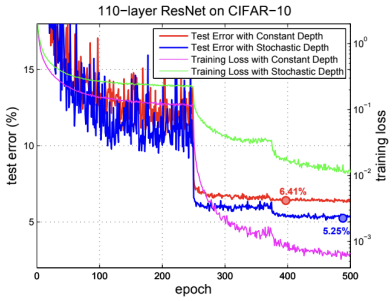
Group: Jose Vizueth, Prakriti Tandon, Rachael Close, Tanay Punjabi, Tasmin Sangha
Github: https://github.com/rachaelclose/CS4782-5782FinalDeliverable

Introduction

Training deep neural networks allows models to capture increasingly complex features, which improves the model's ability to generalize from raw input to high-level concepts. However, these models often face challenges such as vanishing gradients, diminished feature reuse, and extensive training times. Addressing these challenges is crucial for improving the generalization capabilities of deep models. The paper "Deep Networks with Stochastic Depth" by Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, Kilian Weinberger introduces a novel training technique where ResNet layers are randomly dropped out during training according to a survival probability schedule. The main contribution of this paper include: proposing this novel stochastic depth training approach for deep neural networks, demonstrating significant improvements in test error across different datasets such as CIFAR-10, CIFAR-1000, SVHN, and ImageNet, and establishing stochastic depth as an effective method to mitigate the vanishing gradient problem.

Chosen Result

We aim to reproduce the performance shown in Figure 3, where stochastic depth lowers the test error compared to the traditional constant-depth ResNets. The figure illustrates the improvements in test error and training efficiency on the CIFAR-10 dataset using stochastic depth in residual networks (ResNets). We choose to reproduce figure 3 because it demonstrates the technique's effectiveness reducing test error and improving training efficiency.



Methodology

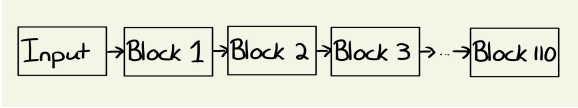
Dataset

The dataset used in our experiments is CIFAR-10, which contains 60,000 color images of size 32×32 pixels, evenly distributed across 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each class contains 6,000 images. We split the original 50,000 training images into 45,000 for training and 5,000 for validation, while keeping the 10,000 test images separate. To improve generalization and prevent overfitting, we applied standard data augmentation techniques: random horizontal flipping and random translation by up to 4 pixels.

Training

We trained two versions of a ResNet-110 model: one using a constant depth architecture and the other using stochastic depth. Both models were trained by using SGD for 500 epochs, with a batch size of 128, momentum of 0.9, and weight decay of 1e-4. The initial learning rate was set to 0.1 and reduced by a factor of 10 at epochs 250 and 375, following standard training schedules used in prior ResNet implementations. Both models used the same architecture and optimization settings to ensure a fair comparison.

Constant Depth



In the constant depth baseline, all 110 residual blocks are active throughout training and testing. Each residual block consists of two convolutional layers, batch normalization, and a ReLU activation, with a skip connection that adds the input directly to the output. This configuration is consistent with the original ResNet architecture, designed to alleviate vanishing gradients while maintaining full model depth. Unlike the stochastic version, no layers are ever skipped, so the full network is used at every forward and backward pass, making the training more stable but potentially slower and more prone to overfitting compared to dynamic-depth approaches.

Stochastic Depth

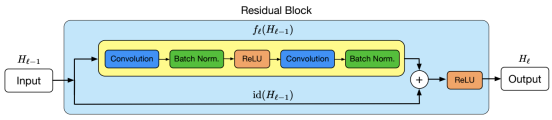
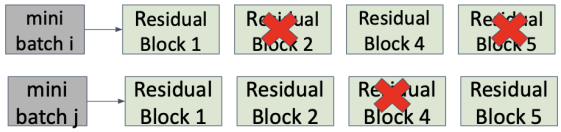


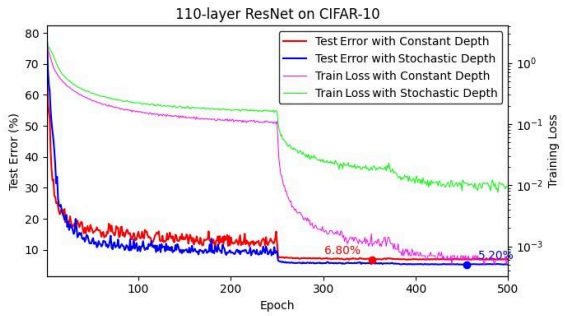
Fig. 1. A close look at the ℓ^{th} ResBlock in a ResNet.

The base architecture for ResNetDrop (model for stochastic depth) was the same as that for ResNet. When training, for each mini-batch, during the forward pass of the residual block b, we sampled a Bernoulli variable that was 1 with survival probability p_b. If it is 1, the block was not dropped and $H_b = ReLU(f(H_{b-1}) + H_{b-1})$. Otherwise, $H_b = H_{b-1}$. Overall, as calculated in the paper, in expectation, 40 out of the 54 residual blocks are active during training.



During inference, no layers were dropped. Instead, we scale each residual block's output by p_b because in expectation, each block is active only with a p_b probability. So, $H_b = ReLU(\boldsymbol{p_b} f(H_{b-1}) + H_{b-1})$. We determined the survival probability using a linear decay rule, such that input has a p_0=1 and the last layer L has p_L= 0.5 . Then, $p_b = 1 - \frac{b}{L}(1 - p_L)$ where $1 \leq b \leq L$. We gave earlier blocks a higher chance of survival because earlier layers capture more low-level and basic details of the image whereas later layers capture finer details.

Results & Analysis



The picture above describes our re-implementation results. We highlighted the test error from the epoch with the lowest validation error and obtained an error of **6.80**% for constant depth and an error of **5.20**% for stochastic depth, which is very comparable to the paper's error of **6.41**% and **5.25**%, respectively. As such, we got results that were extremely similar to the paper's. One potential source of discrepancy is that we implemented our model using Python, while the paper's authors used Lua. However, given our results are extremely similar, then it is likely not a big concern. One of the greatest challenges came from training our models, since even using the smallest dataset with only 110 layers and 500 epochs, it took us over 10 hours to train the two models. Since stochastic depth benefits are best seen with deeper models, then making smaller models still did not allow for in-depth experimentation. Due to our results, it is safe to assume that stochastic depth is a consistently good regularizer as they coincide closely to the paper's. Furthermore, it demonstrates how good stochastic depth is at reducing test error, as seen in the graph. As such, stochastic depth has a lot of potential in becoming another great tool to use alongside various machine learning models and become an essential part in the deep learning "toolbox".

Reflections

Re-implementing stochastic depth deepened our understanding of how dynamic layer skipping can improve both training efficiency and generalization in deep networks. We found that carefully matching the original paper's setup—particularly the training schedule, survival probability decay, and validation timing—was essential to reproducing results accurately. Visualizing training curves gave us valuable insight into how the models differed not just in final accuracy, but also in convergence behavior throughout training. Our results showed that stochastic depth consistently outperformed constant-depth ResNets in both test error and training time. By randomly dropping residual blocks during training, the method effectively mitigates vanishing gradients and enables faster, more stable learning. These findings closely aligned with those in Figure 3 of the original paper, validating its core claims. Moving forward, we're interested in applying stochastic depth to deeper networks and larger datasets like ImageNet, experimenting with alternative or learned survival schedules, and exploring how this technique generalizes to

other architectures such as DenseNets and vision transformers.

References

1) Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K. Q.: Deep networks with stochastic depth (2016)
2) CIFAR-10 and CIFAR-100 Datasets, www.cs.toronto.edu/~kriz/cifar.html.