CMSC 35401 Topics in Machine Learning
Final project report: Incremental learning

Bohdan Kivva[1]

[1]Department of Mathematics, The University of Chicago

bkivva@uchicago.edu

Abstract

In this project we explore effect of incremental networks learning. We will compare test accuracy for models trained from sketch and for models where some of the layers are initialized by the weights of smaller network trained on the same dataset. We show that incremental learning technique applied to the standard ResNet architecture is very likely to improve test accuracy, while requires 10-15% less training time on CIFAR-10 and CIFAR-100 datasets.

1. Introduction

Deep Neural networks are widely used as a main tool for various machine learning tasks. Despite the enormous interest to the subject and amount of the research already done we are still very far from understanding optimal strategies of training deep neural networks. Many practical evidences show that larger (i.e deeper or wider) neural networks tends to provide better classifiers for various tasks. At the same time, with large depth of the network various problems arise, such as problem of vanishing gradients. This resulted in development of various CNN structures (ResNet is one of examples), as well as wide usage of transfer learning approach.

One of the ideas to resolve some of the training difficulties (e.g vanishing gradients) for huge NN is to train network in stages. Namely, first train k-layer CNN for specific task, and then replace classification layer with several additional training layers, or enlarge width of some layers, etc. In this way, on every stage, large portion of the network will be initialized with non-random values, with weights that compute relevant features for the task under consideration.

In this project we explore incremental learning approach for the basic ResNet architectures on CIFAR-10 and CIFAR-100 datasets. We also consider iterated incremental learning approach and compare results with networks trained from sketch and enlarged just once. We show that network obtained by growing smaller network provides better classification performance on the test set, than network of the same depth trained from sketch, while requires less training time. At the same time we get that iteration of this process does not give any advantage, at least without non-trivial hyperparameter tuning.

2. History and prior work

At this moment idea of incremental learning did not find large support among researchers as most of the state-of-the-art results are obtained by direct training of deep CNNs. However, several recent papers argue that probably we just have not explored all the power of this method.

In [1] authors show that on SUN-397 dataset incremental learning techniques combined with transfer learning provide significantly better results than classical fine-tuning. They explored both depth and width augmentation of AlexNet and VGG-16 as well as various combinations of both approaches. See Figures 1 and 2 below. One of the crucial observations they made is that we should carefully adjust learning rate for pre-trained weights, namely it should be considerably smaller than learning rate used for newly added nodes. As shown in [2] if naively the same learning rate is used, incremental learning technique will fail.

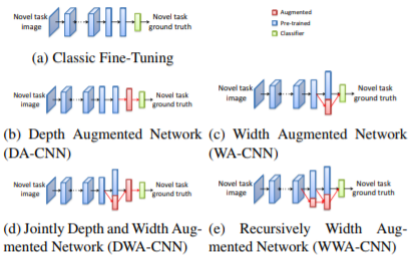

Figure 1: *Various incremental techniques explored in [1].*

Network	Type	Method	Acc (%)			
			New	FC_7–New	FC_6–New	All
AlexNet	Baselines	Finetuning-CNN	53.63	54.75	54.29	55.93
		[1, 15]	48.4	—	51.6	52.2
	Single (Ours)	DA-CNN	54.24	56.48	57.42	58.54
		WA-CNN	**56.81**	56.99	57.84	58.95
	Combined (Ours)	DWA-CNN	56.07	56.41	56.97	57.75
		WWA-CNN	56.65	**57.10**	**58.16**	**59.05**
VGG16	Baselines	Finetuning-CNN	60.77	59.09	50.54	62.80
	Single (Ours)	DA-CNN	61.21	62.85	63.07	65.55
		WA-CNN	**63.61**	**64.00**	**64.15**	**66.54**

Figure 2: *Classification accuracy of various developmental networks on SUN-397 dataset.*

In [3] authors explore different technique of enlarging neural network. On every stage currently trained network is duplicated and training dataset is splitted into 2 parts. Idea is that we want each network to be an expert on its own scope of the dataset. This approach was used to give state-of-the-art performance on Shanghaitech dataset.

Many other interesting incremental learning techniques are proposed in the literature [2, 4, 5].

Our technique is slightly different from the mentioned above.

3. Questions we study and motivation. Experiment design

Residual neural networks are good candidates for exploring incremental learning technique. The reason lies in the theoretical motivation for their structure. Namely, skip connections allows us to treat every next layer as an additive increment to the result computed by the network prior this layer (see Figure 3). Moreover, initializing all weights of the layer by 0 is equivalent to precomposing function realized by the network below this layer with identity function.

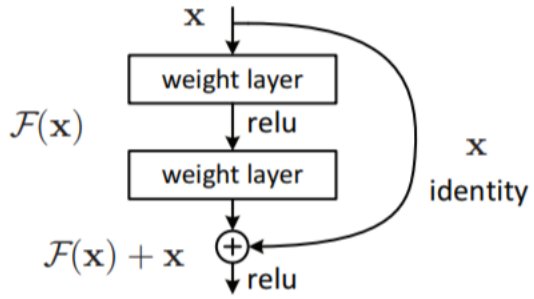

Figure 3: *Residual layer*

Therefore, mathematically, including additional residual layers (initialized by zeros) into well-trained model does not decrease accuracy of the model, and provides model additional capacity for further training.

In this project we investigate the effect of retraining well-trained residual networks after growing the network with additional residual blocks. Instead of initializing weights of additional layers by zeros we will initialize them by random noise of small amplitude. This corresponds to a "small" random perturbation of pre-trained model before training enlarged model.

We focus on studying effect of growing networks for two architectures proposed in [8]. See section 4.1 for more details. Both architectures consists of three stages, where each stage have the same number n of residual layers (but with distinct input dimension and number of channels).

We will grow network by increasing the number of residual layers on each stage from n to $m > n$. Then, first n layers of each stage of enlarged network will be initialized with the weights of pre-trained initial network. All other weights will be initialized with "small" random noise.

In our experiments we will train ResNets with parameters n and $m > n$ for 100 epochs, after that ResNet with parameter n trained for 50 epochs will be enlarged to ResNet with parameter m and will be trained for another 50 epochs. We will compare classification accuracy and training time for all three models. We will run experiments on CIFAR-10 and CIFAR-100 datasets. See Section 5 for more details and results.

4. Implementation and dataset details

4.1. Implementation

We will use two initial basic implementations of ResNets form the seminal paper [8]. First implementation as a residual layer uses double 3×3 Conv2D-BN-ReLU layers. Second implementation uses so called "bottle-neck" layer as a residual layer, namely combination of (1×1)-(3×3)-(1×1) BN-ReLU-Conv2D layers. We will refer to them as version 1 and version 2. See Figure 4.

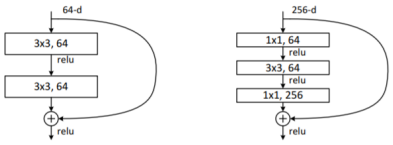

Figure 4: *Standard residual layer and "bottleneck" residual layer*

ResNets in both versions consist of three stages, each stage consists of n residual layers of corresponding version. As all our experiments will be on 32×32 pixel images, layers of the first stage have input dimension 32×32, layers of second stage - 16×16 and layers of third stage - 8×8. In the first version the number of filters on the first stage is 16 and is doubled for every next stage. In the second version the number of filters on the first stage is 64 and is doubled for every next stage.

We use categorical crossentropy as a loss function, ReLU activations and Batch Normalization as regularization.

For set initial learning rate for training to be 0.001. All the models trained from sketch are trained for 100 epochs with learning rate drop after 17 (by 0.5), 35 (by 0.1), 60 (by 0.01), 75 (by 0.003) and 87 (by 0.0001) epochs. All models trained in a "growing" regime are trained for 50 epochs with learning rate drop after 15 (by 0.5), 25 (by 0.1), 38 (by 0.003) and 43 (by 0.001) epochs.

4.2. Dataset

In this project we work with CIFAR-10 and CIFAR-100 datasets. Both datasets consists of 32×32 color images. CIFAR-10 contain 50000 training and 10000 test images evenly distributed among 10 classes. CIFAR-100 has the same amount of training and test images (as CIFAR-10) evenly distributed among 100 classes.

5. Results

5.1. ResNet: standard version. CIFAR-100

We train three standart ResNets of depth 32, 44 and 80 (n = 5, 7, 13 residual layers per stage) for 100 epoch. Also we train two enlarged networks: we enlarge ResNet32 to ResNet44, and ResNet44 to ResNet 80.

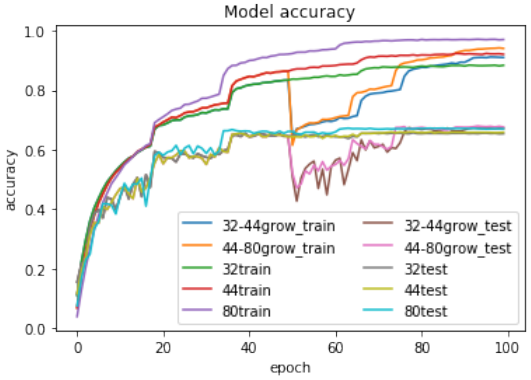

Figure 5: *Training and validation accuracy of ResNetv1 of depth 32, 44, 80 and their "grown" versions*

To enlarge ResNet32 to ResNet44 on each of three stages of network architecture we initialize first 5 residual layers with the weights of trained ResNet32. Other two layers of each stage are initialized with "small" random noise. We choose random initialization instead of zero initialization as a possible tool to reduce overfitting effect. Initial ResNet32 is trained for 50 epochs, then enlarged model is trained for another 50 epochs.

In the table below we summarize test accuracy of each model, training time and number of training parameters.

Model	Test accuracy	Time/epoch	Tr. param.
ResNet32v1	65.68%	68sec	$0.47 \cdot 10^6$
ResNet44v1	66.05%	85sec	$0.67 \cdot 10^6$
ResNet32-44v1	**67.41%**	77sec	$0.67 \cdot 10^6$
ResNet80v1	67.07%	139sec	$1.26 \cdot 10^6$
ResNet44-80v1	**67.97%**	112sec	$1.26 \cdot 10^6$

Table 1: *Test accuracy, average training time per one epoch and number of training parameters for ResNet with standard residual layer on CIFAR-100 dataset after 100 epoch of training*

We see improvement in the test accuracy in both cases of growing network by significant margin (+1.36% for ResNet44 and +0.9% for ResNet80), while entire training takes $\sim$ 10% less time.

Below (Figure 6) we provide the graph for the validation accuracy in the "final" stage of training. We see that networks grown from the versions of smaller depth significantly outperform networks of the same depth trained from sketch.

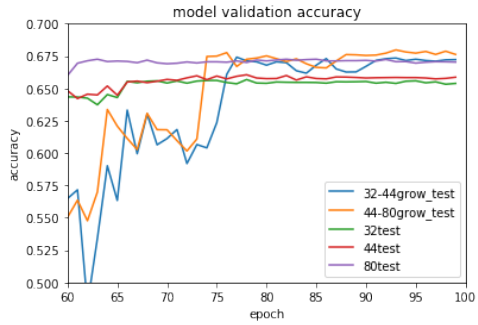

Figure 6: *Validation accuracy of ResNetv1 of depth 32, 44, 80 and their "grown" versions on CIFAR-100*

5.2. ResNet: standard version. CIFAR-10

We perform similar experiment on CIFAR-10 dataset. As deep models achieve almost indistinguishable test accuracy on CIFAR-10 we report experiment result only for growing ResNet32 to ResNet44. For the same reason, we will not run this experiment on bottleneck version. Graphs for CIFAR-10 look similar with CIFAR-100 graphs, so we just report accuracy and training time in the table below (Table 2).

We see slight improvement in test accuracy, while $\sim$ 10% improvement in the training time as in CIFAR-100 case.

5.3. ResNet: bottleneck version

In this part of the experiment we explore growing of ResNets with bottleneck residual layer (see Section 4.1) for details. We

Model	Test accuracy	Time/epoch	Tr. param.
ResNet32v1	91.23%	73sec	$0.47 \cdot 10^6$
ResNet44v1	91.61%	91sec	$0.67 \cdot 10^6$
ResNet32-44v1	**92.06%**	82sec	$0.67 \cdot 10^6$

Table 2: *Test accuracy, average training time per one epoch and number of training parameters for ResNet with standard residual layer on CIFAR-10 dataset after 100 epoch of training*

train three ResNets of depth 47, 65 and 92 (n = 5, 7, 13 residual layers per stage) for 100 epochs. We use the same technique to enlarge ResNet as in the case of standard residual layer structure (see Section 5.1). Namely we will use weights of pre-trained network of smaller size to initialize weights for first n layers of each stage. Rest of the layers are initialized with random noise.

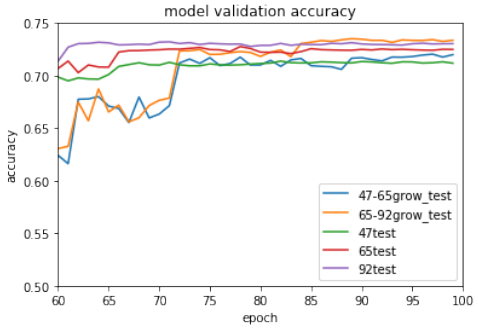

Figure 7: *Validation accuracy of ResNetv2 of depth 47, 65, 92 and their "grown" versions*

In the table below we summarize our results for this part of the experiment.

Model	Test accuracy	Time/epoch	Tr. param.
ResNet47v2	71.38%	113sec	$1.4 \cdot 10^6$
ResNet65v2	**72.76%**	151sec	$1.97 \cdot 10^6$
ResNet47-65v2	72.20%	132sec	$1.97 \cdot 10^6$
ResNet92v2	73.21%	205sec	$2.78 \cdot 10^6$
ResNet65-92v2	**73.51%**	178sec	$2.78 \cdot 10^6$

Table 3: *Test accuracy, average training time per one epoch and number of training parameters for ResNet with bottleneck residual layer on CIFAR-100 dataset after 100 epoch of training*

We see that growing of ResNet from depth 47 to depth 65 failed to give better test performance. At the same time, growing of ResNet from depth 65 to depth 92 yields 0.3% improvement for the test accuracy, while required $\sim$ 14% less training time.

5.4. Growing ResNet twice

In this part of experiment we will explore the effect of regrowing the network. We first enlarge ResNet32 to ResNet44, and then enlarge obtained model to Resnet80 for standard residual layer. And we will enlarge Resnet47 to ResNet65, and then to ResNet92 for bottleneck layer design. We train first model for 50 epochs, we train first enlarged model for 25 epochs and train final enlarged model for extra 50 epochs.

We summarize the results in the table below.

Model	Test accuracy	Epochs
ResNet65v2	72.76%	100
ResNet92v2	73.21%	100
ResNet65-92v2	**73.51**%	50+50
ResNet47-65-92v2	72.36%	50+25+50
ResNet44v1	66.05%	100
ResNet80v1	67.07%	100
ResNet44-80v1	**67.97**%	50+50
ResNet32-44-80v1	67.85%	50+25+50

Table 4: *Iterated incremental learning. Test accuracy, and number of epochs for corresponding model on CIFAR-100 dataset*

As we see, in both cases growing ResNet twice gives worse result than direct growing of the network, moreover it requires larger number of epochs.

6. Conclusions and future work

We saw that for ResNet with basic residual layer design in all of our experiments ResNet trained using incremental learning technique gives better test accuracy and requires ($\sim 10\%$) less training time than ResNet of the same depth trained from sketch. For the ResNet with bottleneck residual layer the best test accuracy we achieved is given by the model trained using incremental learning. At the same time for this ResNet design in one of the cases (growing from depth 47 to 65) direct training gave better result than corresponding model trained via incremental learning.

As a possible future work we consider trying similar incremental learning technique for more advanced ResNet architectures, for instance, Wide ResNets [9]. We think that even though even improved results on considered architectures are far from state of the art they give a hope that incremental learning may be useful as a tool for more advanced architectures.

7. Acknowledgments

Author is grateful to Professor Michael Maire for providing the project idea and giving valuable suggestions. Author also would like to thank Google Colab for providing free computational time on its GPU.

8. References

[1] Wang, Y. X., Ramanan, D., Hebert, M. (2017). Growing a brain: Fine-tuning by increasing model capacity. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2471-2480).

[2] Li, Z., Hoiem, D. (2018). Learning without forgetting. IEEE transactions on pattern analysis and machine intelligence, 40(12), 2935-2947.

[3] Aljundi, R., Chakravarty, P., Tuytelaars, T. (2017). Expert gate: Lifelong learning with a network of experts. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3366-3375).

[4] Shmelkov, K., Schmid, C., Alahari, K. (2017). Incremental learning of object detectors without catastrophic forgetting. *In Proceedings of the IEEE International Conference on Computer Vision* (pp. 3400-3409).

[5] Rebuffi, S. A., Kolesnikov, A., Sperl, G., Lampert, C. H. (2017). icarl: Incremental classifier and representation learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2001-2010).

[6] Huang, Y., Cheng, Y., Chen, D., Lee, H., Ngiam, J., Le, Q. V., Chen, Z. (2018). Gpipe: Efficient training of giant neural networks using pipeline parallelism. *arXiv preprint arXiv:1811.06965*.

[7] Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., Le, Q. V. (2018). Autoaugment: Learning augmentation policies from data. arXiv preprint arXiv:1805.09501.

[8] He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[9] Zagoruyko, Sergey, and Nikos Komodakis. "Wide residual networks." arXiv preprint arXiv:1605.07146 (2016).