CSC413 Project Proposal

Feiyang Fan
feiyang.fan@mail.utoronto.ca

Junyuan Li
junyuan.li@mail.utoronto.ca

Abstract

Convolutional neural network is the dominant architecture in image classification. This paper explores two convolutional neural network architectures: ResNet and DenseNet, and attempts to reproduce some experimental results from two existing papers about image classification on CIFAR-10+(Augmented). Then we extend these methods on the datasets Fashion MNIST and SLI10 respectively, as well as quantitatively analyse their hyperparameter sensitivities and compare their performance after limited epoches of training.

1 Introduction

Starting from LeNet-5 [5] in 1998, deep convolutional neural networks have led to a series of breakthroughs for image classification. For us to have a more in-depth understanding of the convolutional neural network, it is important to re-implement some existing methods and re-evaluate the implementations against some standard benchmarks.

Therefore, the main focus of this paper is to reproduce the experimental results from two papers: ResNet [3] and DenseNet [10], both on CIFAR-10+(Augmented) dataset [2]. Then we perform sensitivity analysis on hyperparameters and find the best hyperparameters of each model on the same dataset: CIFAR-10+ [2], to compare their performance. After that, the two algorithms are applied to new datasets to further analyze their performance, and the strength and weakness of these two models are discussed using the experimental data.

2 Related Works

ResNet. Residual techniques in convolutional neural networks which was first introduced by ResNet [3] in 2015 has been part of the foundations of emerging deep learning models for image classification such as the IRRCNN [6]. An excellent analogy of the problem in a CNN that we may solve this way is the "traffic problem"(i.e. Vanishing Gradient) in the Highway Networks [9]. For these reasons, ResNet [3] has been seriously considered by the medical imaging field recently. Techniques that mix the residual connections with autoencoder and deconvolutional network have been proposed on low-dose CT [1]. Dropout layers [7] has also been put to good use alongside with residual connections in detecting diabetic retinopathy [8].

DenseNet. DenseNet [10] is an architecture that is clearly inspired by ResNet [3] since it adds many more skip connections to a block of layers and call it the "dense block" on the basis of the skip connection patterns of the ResNet [3] architecture. Similar models such as ReNeXt [11] and Deep Network With Stochastic Depth [12] both heavily rely on the original ResNet [3], but with their own ideas for improving performance: DenseNet [10] has also been put to practical use in Camera Model Identification and Post-processing Detection [13] which means it has been proven to be useful, which is why we will try to gain a deeper understanding by reproducing it.

3 Method / Algorithm

3.1 ResNet

The ResNet model we are trying to reproduce was introduced by Microsoft Research in 2015 [3]. The idea of this model is to adopt residual learning to every few stacked layers in the network. Each few stacked layers is considered as a building block. The formal definition of a building block is:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}$$

Where $\mathcal{F}$ is the residual function to be learned, which often has two or more layers. The reason we want $\mathcal{F}$ to have more than a single layer is that the build block $\mathbf{y}$ will be a linear layer $W_1\mathbf{x}+\mathbf{x}$, which has no advantages over normal CNN models [3]. The $\mathcal{F}+\mathbf{x}$ is performed by a shortcut connection and element-wise addition.
When the dimensions of $\mathbf{x}$ and $\mathcal{F}$ mismatch, i.e., the input/output channels are changed, a linear projection W_s is used in the shortcut connection to match the dimensions as follows:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + W_s\mathbf{x}$$

The final residual network we are going to reproduce contains 20 layers. Most of the convolutional layers have 3x3 filters. Downsampling is performed using convolutional layers that have a stride of 2. It ends with a global average pooling layer and a 1000-way fully connected layer with softmax.

3.2 DenseNet

DenseNet [10] has an architecture that consists of a 7x7 convolution layer followed by 3x3 max pooling layer, both of stride 2. Then the following is a series of dense blocks with transition layers between every two dense blocks, and ends with a global average pooling layer and finally a 1000 dimension fully connected layer with softmax. Each dense block contains some convolution layers and the l^{th} layer in each dense block has incoming connection from all previous layers in the current dense block:

$$\mathbf{x}_l = H_l([\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{l-1}])$$

where $\mathbf{x}_l$ is output of the l^{th} layer and $[\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{l-1}]$ represents concatenation of the outputs produced from layers 0 to $l-1$ (layer 0 is input of the DB) and H_l is the l^{th} ConvLayer in the dense block. Each transition layer consists of a 1x1 convolution followed by a 2x2 average pooling layer with stride 2.

4 Experiments and Discussions

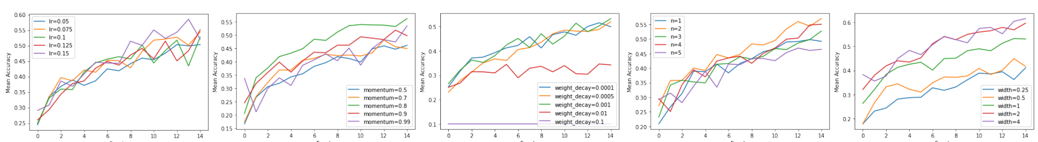
4.1 ResNet

4.1.1 Reproducing ResNet

We attempted to reproduce ResNet20 trained on augmented CIFAR10 described in the original ResNet paper. In the model, weight decay is 10^{-4}, momentum is 0.9 and we follow the same weight initialization[14] and batch normalization[15] techniques. Learning rate is initialized to 0.01 for warmup, then switched to 0.1 after 400 batch iterations, switched back to 0.01 and scaled further down to 0.001 after 32k and 48k batch iterations respectively and trained with in total 64k batch iterations with a 45k/5k/10k training, validation, testing split on the 60k CIFAR10 dataset. Data augmentation as in [16] is practiced and the optimizer is SGD. In particular, for the sake of exploration we adopted option (B) for matching dimensions in some skip connections, namely that non-trainable 1x1 convolutions are used with stride 2.
The final testing accuracy after 64k iterations is 0.8683 which is reasonably close to the author's figure of 0.9125 with option (A). However, this shows that option (A) still performs significantly better than option (B). This may be because zero padding preserves original information whereas convolution transforms it.

4.1.2 Hyperparameter Analysis

Due to training time and usage limit, we analyse ResNet hyperparameters sensitivities trained by 15 epochs of CIFAR 10 data with batch size 128. We also use a 50k/10k training test split to increase our trainable images and use the test images each epoch for visualizing the trend of accuracy. Learning rates are 0.05, 0.075, 0.1, 0.125 and 0.15. From the figure we see that as learning rate increase, initial accuracy increase with increasing fluctuations. This may be because of irregular loss surface yet using the initialization technique as in [14] we are already in a good spot for SGD. Momentum values are 0.5, 0.7, 0.8, 0.9 and 0.99. The figure clearly shows that momentum of 0.8 performs better than other values in most of the epochs. This is reasonable as the figure of the learning rate suggests that the initialization prepares us for SGD but the loss surface is still rough. This means that the right amount of momentum works best for a correct amount of inertia is important. Weight decay values are set to 0.0001, 0.0005, 0.001, 0.01, 0.1. From the figure, clearly weight decay at the scale of 10^{-3} or lower helps us reduce overfitting but at the scale of 10^{-2} it hinders learning for making feature distances too small. At scale of 10^{-1} this is so severe it stops training altogether. The value of n in ResNet model for CIFAR10 is essentially the depth of the model, is set to 1, 2, 3, 4 and 5. The corresponding number of layers of the deep models are 8, 14, 20, 26, 32. Suprisingly the model with n=2 outperforms other models in most of the epochs. This may be because for n=1 the model is not deep enough for the variability of this large dataset, but for larger n the model needs more data to utilize its large amount of training parameters. The value of width in ResNet model is the width of the feature maps with values 0.25, 0.5, 1, 2 and 4. Corresponding feature map sizes are (4, 8, 16), (8, 16, 32), ..., (64, 128, 256). Unsurprisingly the larger the width of the model, the better results it get.



(a) Learning rate (b) Momentum (c) Weight Decay (d) n value (e) Width

Figure 1: ResNet Hypermeter Results

4.2 DenseNet

4.2.1 Reproducing DenseNet

The DenseNet121 network is trained using stochastic gradient descent on augmented CIFAR-10[2] with a 45k/5k/10k training, validation, testing split as described in the original DenseNet[10] paper. The batch size is set to 64. Due to usage limit, the number of epochs trained is 100 epochs, rather than 300 epochs. However, the final test accuracy of our model comes close to the test result present in the original paper. The initial learning rate is set to 0.1, and is divided by 10 at 50% and 75% of the total number of training epochs. We used a weight decay of 10^{-4} and a momentum of 0.9 without dampening. We follow the same weight initialization[14] and batch normalization[15] techniques. The final test accuracy after 100 epochs on augmented CIFAR-10[2] is 0.865, with an error rate of 13.5%. The test error rate of DenseNet121 (k=12) on augmented CIFAR-10 in original DenseNet paper is 5.24% after 300 epochs[10], and around 11% after 100 epochs. We can see that our model did reproduce a close result to the result in the original paper.

4.2.2 Hyperparameter Analysis

We analyzed five hyperparameters: learning rate, momentum, weight decay, growth rate, and block configs. For each hyperparameter analysis, we used the same dataset augmentation[2], weight initialization[14], and batch normalization[15] techniques.
First, we trained DenseNet121 with 5 different learning rates, 0.1, 0.05, 0.075, 0.125, 0.15 for 5

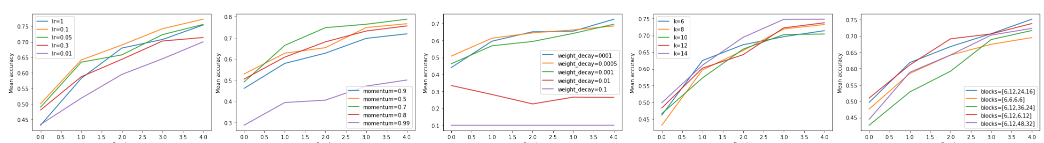
epochs. All other hyperparameters stay the same as the original paper. we can see that as the learning rate increases or decreases from 0.1 used in the original paper, the test accuracy decreases. Figure (a) in figure 2 below shows the accuracies of different learning rates on CIFAR-10+[2].

Then we used five different momentums: 0.9, 0.5, 0.7, 0.8, 0.99. We observed that with higher momentum, the accuracy is significantly lower than smaller momentum. The best momentum we found that improved the performance is 0.7. The final test accuracy is 0.78, which is 0.05 higher than 0.73 using momentum 0.9. Figure (b) in figure 2 below shows the accuracies of different momentums.

Weight decay values are 0.0001, 0.0005, 0.001, 0.01, 0.1. From figure (c) in figure 2, we can see that large weight decay yields extremely poor accuracy (0.1), while smaller weight decay values produces roughly the same accuracy (0.7).

Next, we analyzed the effect of growth rate k. Growth rate k means each composite function of a DenseNet produces k feature-maps. k determines the width of the model. We trained the model with five different growth rate, 6, 8, 10, 12, 14 for 5 epochs. We can see from figure (d) in figure 2, that as k decreases from 12, the growth rate used in the original paper, the accuracy decreases. As k increases from 12, the accuracy increases.

Last but not the least, we trained the DenseNet121 model on five different block configs. Block config represents how many layers in each pooling block, i.e., it determines the depth of the model. From figure (e) in figure 2, we can see that, generally, as we have more layers in each pooling block, the accuracy increases. The fewer layers in each pooling block, the less accurate the model is. Number of bottleneck blocks in each dense layer:



(a) Learning rate (b) Momentum (c) Weight Decay (d) Growth rate (e) Block Config

Figure 2: DenseNet Hypermeter Results

4.3 Train on Different Datasets

For ResNet, it is trained on Fashion-MNIST with the same hyperparameters, the final test accuracy is 0.93 for 180 epochs, which is better than the final test accuracy on CIFAR-10+. For DenseNet121, it is trained on SLI10[17]. All hyperparameters stay the same as the ones used on CIFAR-10+. The final test accuracy is 0.604 for 5 epochs, which is slightly better than DenseNet121 (0.412) on CIFAR-10+ with the same hyperparameters for 5 epochs.

4.4 Discussions

The final test accuracy of DenseNet is better than ResNet for the same number of epochs. DenseNet trains significantly slower than ResNet, we believe this is due to the number of layers in DenseNet is much larger than the number of layers in ResNet. The hyperparameters used in the original papars are always the best combinations of values, since the original authors put a lot of efforts into finding the best hyperparameters.

5 Conclusion

We reproduced ResNet[3] and DenseNet[10] on CIFAR-10+, and got close final test results as the original paper. We also performed sensitive hyperparameter analysis on these two networks. Lastly, we trained them on Fashion-MNIST and SLI10 respectively to see their performance, and found that DenseNet is slightly better than ResNet. Also for small epochs it is obvious that Densenet outperforms ResNet significantly.

References

[1] Chen, Hu & Zhang, Yi & Kalra, Mannudeep & Lin, Feng & Chen, Yang & Liao, Peixi & Zhou, Jiliu & Wang, Ge. (2017). Low-Dose CT With a Residual Encoder-Decoder Convolutional Neural Network. IEEE Transactions on Medical Imaging. 36. 2524-2535. 10.1109/TMI.2017.2715284.

[2] Krizhevsky, A. & Hinton, G.. (2009). Learning multiple layers of features from tiny images. Computer Science Department, University of Toronto, Tech. Rep. 1.

[3] He, Kaiming & Zhang, Xiangyu & Ren, Shaoqing & Sun, Jian. (2015). Deep Residual Learning for Image Recognition.

[4] Vinyals, Oriol & Blundell, Charles & Lillicrap, Timothy & Kavukcuoglu, Koray & Wierstra, Daan. (2016). Matching Networks for One Shot Learning.

[5] Lecun, Yann & Bottou, Leon & Bengio, Y. & Haffner, Patrick. (1998). Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE. 86. 2278 - 2324. 10.1109/5.726791.

[6] Alom, Md. Zahangir & Hasan, Mahmudul & Yakopcic, Chris & Taha, Tarek & Asari, Vijayan. (2020). Improved Inception-Residual Convolutional Neural Network for Object Recognition. Neural Computing and Applications. 32. 10.1007/s00521-018-3627-6.

[7] Srivastava, Nitish & Hinton, Geoffrey & Krizhevsky, Alex & Sutskever, Ilya & Salakhutdinov, Ruslan. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 15. 1929-1958.

[8] Fanany, Mohamad Ivan & Rufaida, Syahidah. (2017). Residual Convolutional Neural Network for Diabetic Retinopathy. 10.1109/ICACSIS.2017.8355060.

[9] Khan, Asifullah & Chouhan, Naveed. (2016). Highway Networks. 10.13140/RG.2.2.23573.32489.

[10] Huang, Gao & Liu, Zhuang & van der Maaten, Laurens & Weinberger, Kilian. (2017). Densely Connected Convolutional Networks. 10.1109/CVPR.2017.243.

[11] Xie, Saining & Girshick, Ross & Dollar, Piotr & Tu, Z. & He, Kaiming. (2017). Aggregated Residual Transformations for Deep Neural Networks. 5987-5995. 10.1109/CVPR.2017.634.

[12] Huang, Gao & Sun, Yu & Liu, Zhuang & Sedra, Daniel & Weinberger, Kilian. (2016). Deep Networks with Stochastic Depth. 9908. 646-661. 10.1007/978-3-319-46493-0_39.

[13] Rafi, Abdul Muntakim & Kamal, Uday & Hoque, Rakibul & Abrar, Mohammed Abid & Das, Sowmitra & Ere, Robert & Hasan, Md Kamrul. (2019). Application of DenseNet in Camera Model Identification and Post-processing Detection.

[14] He, Kaiming & Zhang, Xiangyu & Ren, Shaoqing & Sun, Jian. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. IEEE International Conference on Computer Vision (ICCV 2015). 1502. 10.1109/ICCV.2015.123.

[15] Ioffe, Sergey & Szegedy, Christian. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.

[16] Lee, Chen-Yu Xie, Saining Gallagher, Patrick Zhang, Zhengyou Tu, Z.. (2015). Deeply-Supervised Nets.

[17] Adam Coates, Honglak Lee, Andrew Y. Ng (2011). An Analysis of Single Layer Networks in Unsupervised Feature Learning AISTATS.