Tiny ImageNet Classification with CNN

Xu Zheng

February 25, 2018

1 Introduction

Deep Convolutional Neural Network has been playing an important role for image classification tasks during past few years. In this report, 3 dierent start-of-art Deep Convolutional Neural Network models have been evaluated as the classification algorithms for Tiny ImageNet dataset.

Tiny ImageNet dataset has 200 classes. Each class has 500 training images, 50 validation images, and 50 test images, so it has 120000 images in total. All images are of size 64*64*3[1].

To achieve a high accuracy and reduce overfitting, many techniques of CNN model are applied, including data augmentation, dropout, and residual layers. Based on the observation of the 3 state-of-the-art models I created a model named alchNet and achieved a top-1 validation accuracy rate of 55.7%.

2 Background

2.1 CNN Models

2.1.1 AlexNet

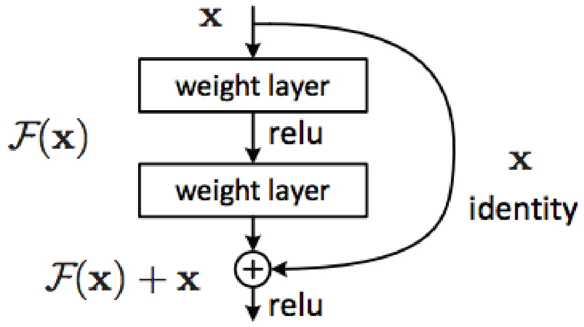
In this report, 3 famous state-of-art CNN models have been evaluated. They are alexNet, vggNet and resNet. AlexNet [2] was the winner of ILSVRC 2012 and the first model to win the challenge. It made CNN models popular for image classification tasks. In this work, an 8 layers CNN model is introduced, including 5 conv layers and 3 fully-connected layers. Data augmentation techniques such as image translations, horizontal reflections, and patch extractions were used to avoid overfitting. ReLU and dropout are also used in this paper.

2.1.2 VggNet

VggNet is another popular model, which is a simple but deep model [3]. This model strictly used 3x3 filters with stride and pad of 1, along with 2x2 max-pooling layers with stride 2. It Secured the first and the second places in the localization and classification tasks respectively in ILSVRC-14 and reached 92.7% top-5 test accuracy in ImageNet. Vgg used scale jittering as one data augmentation technique during training. Even it can achieve high accuracy, it took a very long time to train: Trained on 4 Nvidia Titan Black GPUs for two to three weeks.

2.1.3 ResNet

ResNet [4] is a 152 layer network architecture that won ILSVRC 2015 with an incredible error rate of 3.6%. The idea behind a residual block is that you have your input x, after conv layer, relu layer, and normalization layer series, you will get feature maps: F(x). That result is then added to the original input x: H(x) = F(x) + x, and then continue the training. One residual block looks like:



x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

The residual networks attempt to avoid feature diminishing problem in a very deep network.

2.1.4 AlchNet

Based on the observation of the results from above 3 models, I created my baseline model named alchNet, which is inherited from the resNet. I used 3*3 small filters for all layers. Normalization and dropout are applied to avoid overfitting. Compared with resNet, I increased one more fully connected layers. The final model has 19 layers with 8 residual blocks.

2.2 Data Augmentation

Data augmentation is of great importance for a successful training since deep neural network model needs a large amount of labeled data. We only have 10000 training images in tiny imageNet, data augmentation becomes even more important in this project, which will be proved later. When processing training, I flip each image left to right and up to down. Besides, I also take the random crop of for each image with size 56*56, then add padding to make sure the cropped image is of size 64*64. In this way, we need not get the crop from the validation set.

3 Implementation

3.1 AlexNet

Because our input images are rather small(64*64), which is much smaller than the images in original alexNet(277*277), so I adjusted the original network by making the stride and filter size on the first layer smaller. And the size of the fully-connected layers is also reduced from 4096 to 1024. But the whole architecture is kept the same, we also have 5 conv layers with 3 fully connected layers.

3.2 VGGnet

A shallow vggNet is implemented in this project. I strictly followed the principle from the original work: using 3x3 filters with stride and pad of 1. The architecture is very simple. The whole architecture has 11 layers, so it will be called vgg11 in this report.

[conv(64)*2 + conv(128)*2 + conv(256)*2 + conv(512)*2 + FC*3]

3.3 AlchNet

Based on resNet, I created alchNet, which means residual block has been applied in this model. Dropout is applied on the first fully connected layer. To understand how depth will impact the 1training accuracy, two different alchNet model have been created: alch11 and alch19. The architecture of both alchNet look like:

- alch11: [conv + resLayer(64) + resLayer(128) + resLayer(256) + resLayer(512) + FC*2]

- alch19: [conv + [resLayer(64)]*2 + [resLayer(128)]*2 + [resLayer(256)]*2 + [resLayer(512)]*2 + FC*2]

Since it is quite similar with resNet, the results of resNet has been omitted in this report.

4 Results

4.1 Experiments

- In this report, as I noticed that a model with higher top-1 accuracy always has a higher top-5 accuracy, so I will use the top-1 accuracy for all the models as the main evaluation metrics.
- All the models are implemented in tflearn. These models are trained on google cloud platform with one GPU Tesla K80, 4 CPUs and 16g memory.
- For all the models we use the same pre-processing strategies:
  - Scale each sample by the over all standard deviation
  - Zero center every sample with overall mean
- The data augmentation strategy is also the same(if have):
  - Flip each image left to right.
  - Flip each image up to down.
  - Random crop with size(56*56) and padding the same to keep the network input is (64*64).
- Initial learning rate is 0.001 and batch size is 256.

4.2 Results

In below chart, -a means that model was trained without data augmentation, and -d means that model was trained without dropout. for alchNet19+c, I try to change the order of relu and batch normalization layer after every conv layer of alchNet. All the results will be analyzed in next section.

Model	Epochs	Training Time	Top-1 Val	Top-5 Val
alexNet-a	60	6.5h	27.6	-
alexNet	60	7h	30.8	48.6
vgg11-a	65	4h	26.7	-
vgg11	65	4h	29.9	47.1
alchNet11	65	8.5h	45.8	-
alchNet11-d	65	8.5h	45.3	-
alchNet19+c	72	14h	47.6	-
alchNet19	72	13.5h	55.7	-

5 Analysis

5.1 Data augmentation

4 models have trained to help gain a better understanding of the effect of data augmentation

- alexNet with data augmentation
- alexNet without data augmentation
- vgg11 with data augmentation

- vgg11 without data augmentation

+a means training with augmentation and -a means training without augmentation

Model	Top-1 Val
alexNet-a	27.6
alexNet+a	30.8
vgg11-a	26.7
vgg11+a	29.9

As shown in above chart, data augmentation increased the top-1 accuracy by around 3% for both models within 60 epochs of training. Since we know data augmentation would be helpful to gain a high accuracy, we applied this technique for every alchNet training. Since alchNet is a more deep model, I believe data augmentation is more important for it to get a high accuracy.

5.2 Depth

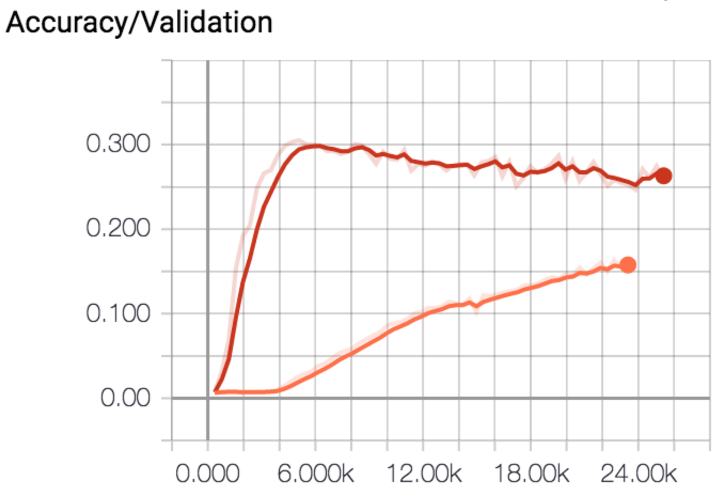
Depth shows an great importance of the models during the training process. While alchNet11 achieved 45.3% top-1 accuracy, alchNet19 achieved 55.7% accuracy. I think it is because that if we keep the filter size the same, by increasing the depth, we can increase the number of filters, which results that we have more weights. If we can avoid the overfitting using some techniques, the model with more layers will have more capability to do the classification.

5.3 Dropout

Dropout has been proven an effective method to increase regularization and prevent overfitting in other papers. However, I tested dropout with alchNet11. Finally, they achieved a very similar accuracy after 72 epochs of training, 45.7 with dropout and 45.2 without dropout respectively. So for my alchNet, dropout has a very limited positive impact. For my final model(alchNet19), dropout is still used before the first fully connected layer.

5.4 Initial learning rate

Another interesting finding is that appropriate initial learning rate is quite important. If the initial learning rate is too low, it will take lots of time to train because steps towards the minimum of the loss function are tiny, even it can give a more reliable result. But if the learning rate is too high, then training may not converge and overshoot the optimal point. Below train process shows the two different situations, The red curve is above the orange one:



The orange curve is the learning curve for vgg11 with initial learning rate 0.0005, while the red curve is the learning curve for vgg11 with initial learning rate 0.001.

During my testing, 0.001 is proved to be a good initial learning rate. Besides, the learning rate for the optimizer should be decreased gradually to make sure the model get the optimal result.

Because if we always keep a high learning rate, we will overfit the model as the model get close to a good local minimum. By delaying the learning rate, each mini-batch makes a smaller change and will be more reliable to get closer to the global optimal rather than the local ones.

5.5 Other Findings

Training time is also a very important factor we need to consider. As shown in above chart, we can see resNet with 72 epochs takes more than 13 hours to train. Actually, the validation accuracy seems not going down after 72 epochs. But with limited time and resources(only one GPU machine), I have to try other models and stop further training.

From the chart, we can also see that changing the order of relu and batch normalization hurt the performance of alchNet19. So in my final model, I applied batch_normalization before the relu layer.

6 Summary

In this project, I explored the classification performance of several models. Based on the observation of the results, I also created myself models to achieve 55.7% top1 accuracy on Tiny ImageNet. Due to limited resources, there are still many aspects can be improved.

For alchNet19, it achieved over 55% validation accuracy after 13 hours training, while the training accuracy is also around 55%. It may achieve a higher validation accuracy with more training time since the learning curvey shows no decreasing trend. In future work, I can spend more time to evaluate this model with more powerful GPU and more training time.

The deepest model I tried in this project has 19 layers. I observed that deep layers model has higher accuracy from the results of alchNet11 and alchNet19, hence I can try a deeper model in the future to see if I can get a higher accuracy.

Another idea is to try different combinations of data augmentation strategies. Data augmentation has shown its importance from our observation, trying different combinations of data augmentation techniques would be an optimizing method to get a high accuracy. For example, I can explore how flip and crop impact the model performance respectively.

A combination of several trained models has been proved to be a good method to increase the testing accuracy [3], even for tiny imagenet[5]. In the future work, I can train the same model many times and ensemble the models together to see if it I can get a higher accuracy.

References

[1] Leon Yao and John Miller. Tiny imagenet classification with convolutional neural networks. *CS 231N*, 2015.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[3] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[4] 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*, pages 770–778, 2016.

[5] Zach Barnes, Frank Cipollone, and Tyler Romero. Techniques for image classification on tiny-imagenet.