Cogs 181 Final Project

Yicheng Bao A13301594

Abstract

In this class, we have covered a couple of neural networks like AlexNet, VGGNet and GoogleNet to do some difficult tasks such as image classification or Natural Language Processing. We know that different neural networks have their own advantages and disadvantages. For example, CNNs fit more for dealing with image problems, while RNNs fit more for language problems. According to this situation, in my research paper, I will use different neural networks for Image Classification task to see which one performs the best on Tiny ImageNet Dataset. Firstly, I will introduce some experiment results of other researchers and explain what my dataset is. Secondly, I am going to explain what my models are and how I use these models to compare the performance. Eventually, we can see the results of hyperparameters tuning. The importance of this paper is to provide readers some insights of model selection as well as hyper-parameters tuning.

Introduction

The ImageNet dataset is a really popular dataset for deep learning researchers to do benchmark on image classification. The Tiny ImageNet is a smaller version of ImageNet that is more convenient and efficient for researchers to evaluate the performance of different models and hyperparameters. Inside the Tiny ImageNet Dataset, there are 200 image classes and each of them contains 500 samples. The training set contains 100,000 images and validation set contains 10,000 images. Each of the image samples has size 64x64x3.

Here are some examples of images in the Tiny ImageNet dataset:

Figure 1.1-image samples

Since the task is to do the image classification, I decided to use three models including AlexNet, VGG-16 and a simple two-layer neural network. Furthermore, I adjusted many hyperparameters such as the Dropout rate, different optimizers, different pooling functions and different activation functions. After finishing the training process, I validated these three neural networks on my validation dataset and found out that VGGNet has the highest accuracy, AlexNet has the second highest accuracy and simple two-layer neural network has the lowest accuracy.

Before we get into the details of these three neural networks, other researchers like Jiayu Wu, Qixiang Zhang, and Guoxi Xu had got some results on the Tiny ImageNet dataset. They train the

dataset on VGGNet, ResNet, and Inception-ResNet. Their results are included in the following table [1]:

Model	Top-1 Val	Top-5 Val	Top-1 Test	# Params
Inception-Resnet	37.56%	18.96 %	43.10%	8.3M
ResNet	43.50%	20.30%	46.90%	11.28M
VGG-19	45.31%	23.75%	50.22%	40.2M
VGG-16	47.22%	25.20%	51.93%	36.7M

Table 1. Summary of Model Error Rates

As we can see in their resulting table, the VGG-16 has around 48% of accuracy on image classification. My experiment will show that if my results on VGG-16 are similar to those and provide a more detailed hyperparameters tuning and a more comprehensive analysis on AlexNet and simple two-layer neural network.

In the following section, I will explain these three models in detail to let readers have a thorough understanding of their structures.

Method

Models

For each model, there are some parameters such as the dropout rate, kernel sizes, pooling functions and activation functions, etc. Therefore, inside each model description, I will provide the parameters of each model and compare them in the next section.

Simple Two-layer Neural Network:

The simple two-layer neural network contains two convolution layers, two pooling layers and a fully-connected layer with dropout operation. Each convolution layer has an activation function, whose default setting is ReLu function.

In order to visualize the structure, I made a figure with different colors:

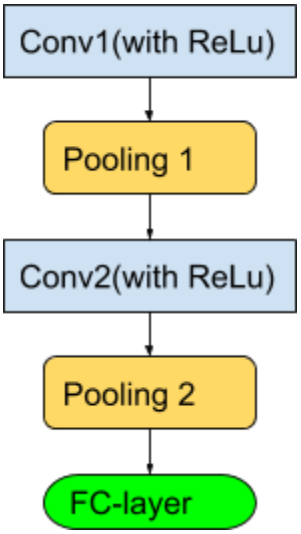


Figure 2.1-Structure

Here is a figure that indicates the number of kernels, kernel sizes and strides of each convolutional layer.

	# of kernel	Kernel size	Stride
conv1	32	(2,2)	2
conv2	64	(5,5)	2

Figure 2.2-Convolution Parameters

The parameters of pooling layers are included in the following table:

	Kernel size	stride
pool1	(2,2)	2
pool2	(2,2)	2

Figure 2.3-Pooling Parameters

Hyper-parameters that can be changed in this model are pooling functions (Average pooling and Max pooling), activation functions (ReLu and Sigmoid) ,

Dropout rate and optimizers (Adam and SGD).

AlexNet:

AlexNet is the name of a convolutional neural network which has had a large impact on the field of machine learning, specifically in the application of deep learning to machine vision. It famously won the 2012 ImageNet LSVRC-2012 competition by a large margin (15.3% VS 26.2% (second place) error rates) [2]. The network had a very similar architecture as LeNet by Yann LeCun et al but was deeper, with more filters per layer, and with stacked convolutional layers. It consisted of 11×11, 5×5,3×3, convolutions, max pooling, dropout, data augmentation, ReLU activations, SGD with momentum. It attached ReLU activations after every convolutional and fully-connected layer [2].

The following figure can better visualize the structure of AlexNet.

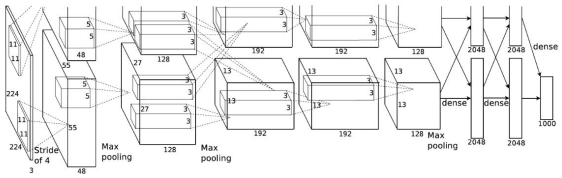


Figure 3.1 - AlexNet architecture

To have a clearer summary, AlexNet contains 5 convolutional layers and 3 fully connected layers. Relu is applied after very convolutional and fully connected layer. Dropout is applied before the first and the second fully connected year.

When it comes to the complexity of the model, the network has 62.3 million parameters and needs 1.1 billion computation units in a forward pass. This means that the model requires plenty of parameters to extract the features of images. Then, I will introduce a better model : VGG-16 which solves the disadvantage of AlexNet to some extent.

VGG-16: VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another [3].

A brief visualization of VGG-16 can be found in the paper:

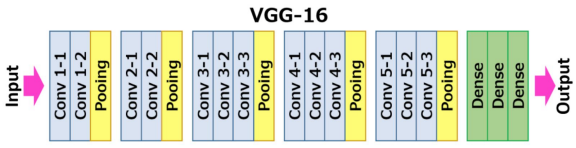


Figure 3.2 - VGG-16 architecture [3]

Performance Metrics

Since all of my three models are used to do the image classification, I will just compare the top-1 accuracy of these models.

Experiment

Data Augmentation

The reason I want to do the data augmentation is that I think it is a good way to improve the quality of training dataset, which will finally increase the accuracy of classification. For each 64x64x3 image, I randomly crop it to multiple 56x56x3 images for training.

Problem Description

The task is doing the image classification which contains 200 classes in total. The goal is to make the validation accuracy as high as possible using different models

Hyper-parameters

For the simple two-layer neural network, I run it with different hyper-parameters : dropout rate(0.0 or 0.5), batch-size(32 or 16), and two different optimizers (Adam and SGD).

For the AlexNet, I have the same hyper-parameters as two-layer neural network : dropout rate(0.0 or 0.5), batch-size(32 or 16), and two different optimizers (Adam and SGD).

For the VGG-16, I tried another optimizer : Momentum & SGD optimizers. I would like to see if the Momentum optimizer will have better results.

Training Process and Results

Since doing GridSearch for all of these hyperparameters will be too time-consuming, I will set the default hyper-parameters as {dropout:0.5,batch-size:32,optimizer:Adam}

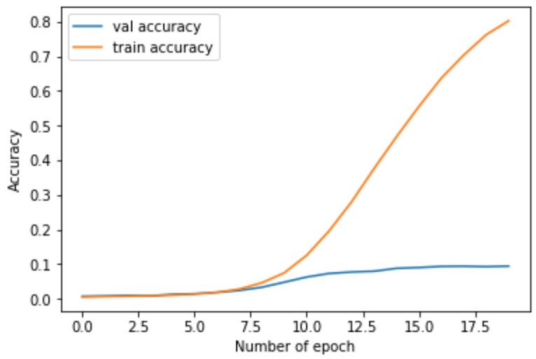
Simple Two-Layer Neural Net:

For the simple two-layer neural network, I run it with different hyper-parameters : dropout rate(0.0 or 0.5), batch-size(32 or 16), and two different optimizers (Adam and SGD). I train around 20 iterations to reach the peak validation accuracy. The overall validation accuracies are included in the following table :

default	No dropout	batch=16	SGD
9.7%	5.9%	7.8%	8.9%

After tuning the hyper-parameters, the best result on validation dataset is 9.7%. This neural network serves as the baseline of the other two models for comparison. Here is the table that records all accuracy.

The graph of training accuracy and validation accuracy with the default setting {dropout:0.5,batchsize:32,optimizer:Adam} and the loss graph are the following :



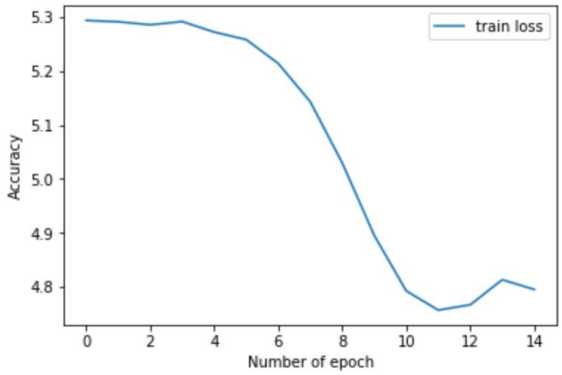


Figure 4.1- Acc and loss plot

Then, I changed the dropout rate to 0.0 see if there is a significant variation of validation accuracy. ({dropout:0.0,batchsize:32,optimizer:Adam)

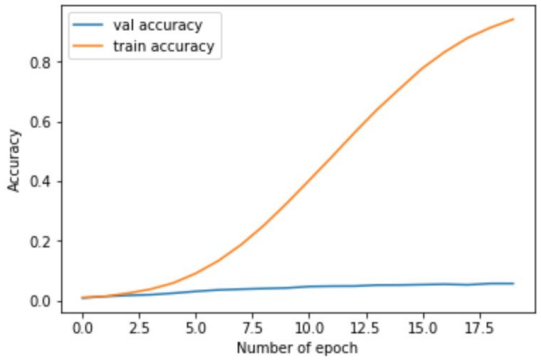


Figure 4.2 - Accuracy plot

I also changed the batchsize and type of optimizers and the results are in the following graph:

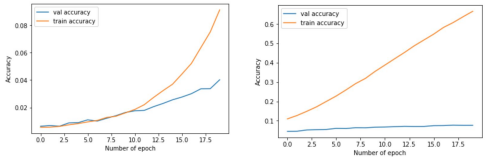


Figure 4.3 - batch size = 16

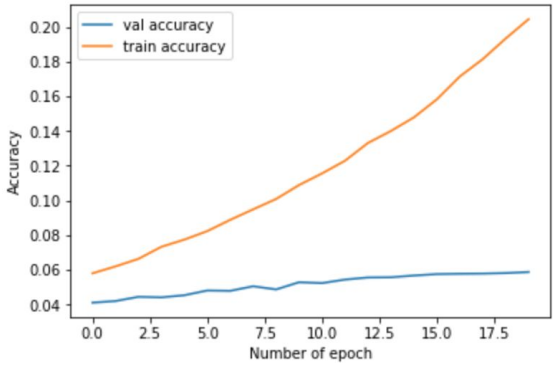
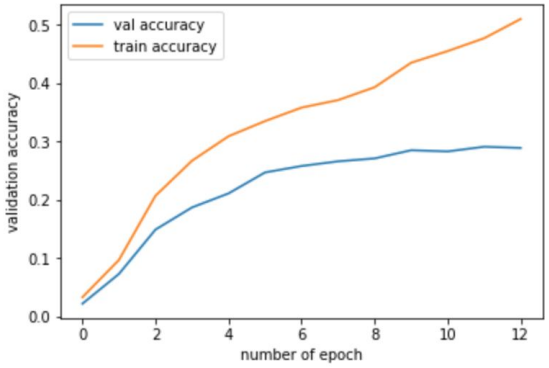


Figure 4.4 - SGD Optimizer

These figures above are some visualization of the performance of different hyper-parameters.

AlexNet:

For the AlexNet, the accuracy and loss are included in the following graph with different hyper-parameters. Here is the accuracy and loss graph for the default setting {dropout:0.5,batch-size:32,optimizer:Adam} :

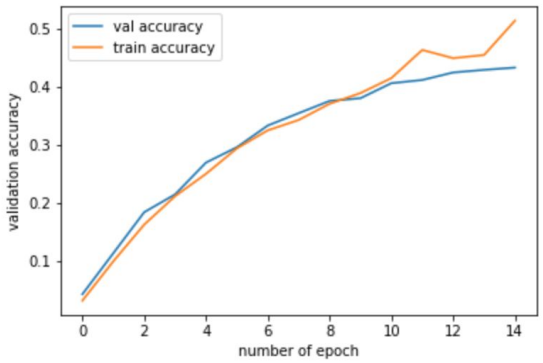


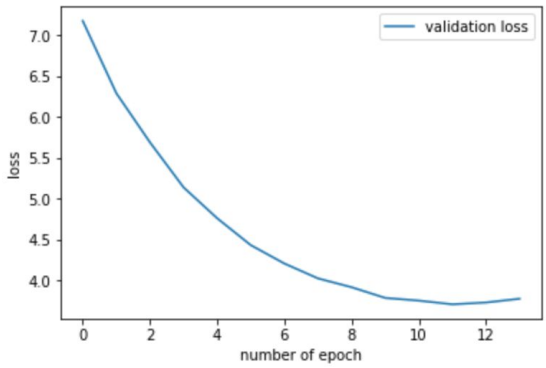
I train the network for 20 iterations and get 29.1% of accuracy on the validation set, which is a good improvement on the baseline model. The overall validation accuracies are the following:

Acc	batch16	No dropout	SGD
train	47.5%	83%	21.1%
val	28.7%	26.4%	20.7%

VGG-16:

For the VGG-16, the accuracy and loss are included in the following plot with dropout equals 0.0 or 0.5 and Momentum or SGD optimizer. Here is the training curve of default hyper-parameters setting.





I train the VGG-16 for only 14 iterations due to extreme long training process and get 43.75% of accuracy, which shows that VGG is potentially the best model among these three.

The results of other variation of hyperparameters are included in the following table:

Acc	default	No dropout	SGD
train	51.5%	80.28%	47.5%
val	43.75%	42.24%	38.7%

Conclusion and Future Improvement

Model	Top-1 acc	Loss
2-layer	9.7%	4.7
AlexNet	29.1%	4.28
VGG-16	43.75%	3.77

Fig 4.1-Best Accuracy and Loss

As we can see Fig 4.1, generally, VGG-16 is definitely the best model among these three as we can see it has the highest validation accuracy. The reason why VGG-16 has the lowest loss is that each layer of VGG-16 has l2-regularizer which can effectively prevent the overfitting. Adam outperforms SGD optimizer : Adam converges faster and reduce loss better. Batch size does affect the final performance

: small batch size sometimes needs more iterations.

For the simple two-layer neural network, we can see that there is a huge overfitting no matter how I use the dropout or not. This is possibly because the neural network does not have a strong ability of generalization. However, when we compare the results of different hyper-parameters, we know that dropout=0.5 can effectively prevent the overfitting to some extent. When we have dropout=0.5, the training accuracy is 80% and the validation accuracy is 10%. If we don't have any dropout, the training accuracy is over 85% and the validation accuracy is only 6%. Furthermore, the batch-size also has much impact on the final results. We can see that if we set a really small batch-size like 16, the loss is unstable : moving up and down for a small step. Finally, the optimizer is really important for the training process. Adam optimizer is a good choice to start and SGD can be regarded as an alternation.

According to the result, AlexNet and VGG-16 are really good image classifiers that can extract useful features from images, and they are much better than the baseline model in terms of its performance. One factor that may cause the AlexNet to behavior much worse than the VGG-16 is that AlexNet usually does not deal with 64x64 picture which is too small for AlexNet. VGG-16 has better ability of generalization as it has l2-norm for each layer, but in this research project, I do not have enough time to add l2-norm for other models.

Finally, there still exist some potential ways to improve the results further: try a learning rate with some decay, try batch-norm, and add regularizers to each layer of neural networks. Currently, for all of these three models, I use a consistent learning rate, so when the loss sometimes may decrease and then increase. If there is a learning rate decay, it may have higher accuracy. Moreover, batch normalization is a good technique that can prevent overfitting. In my baseline model, there is a significant overfitting caused by insufficient ability of generalization. Eventually, in these three models, only VGG-16 use l2-norm in each layer, which produces the best results. Therefore, these three ways are potential methods that can improve the performance significantly. I hope readers can try these methods to see if it really works!

Bonus

For this project, I select the Tiny ImageNet Dataset which contained 200 classes. I did the data augmentation which crop the image randomly to make the training set more comprehensive. Second, I tuned many hyper-parameters : different architectures, with or without dropout, batch-size, and types of optimizers (Adam, SGD, Momentum).

Reference

[1] Jiayu Wu, Qixiang Zhang, and Guoxi Xu, 'Tiny ImageNet Challenge', Stanford University, CA, USA

[2]https://neurohive.io/en/popular-networks/alexnet-imagenet-classification-with-deep-convolutional-neural-networks/
[3]https://neurohive.io/en/popular-networks/vgg16/