Comparing Different Neural Network Models and Optimizing Parameters

Farros Mufid
University of Sydney
Camperdown NSW 2006
fmuf5270@uni.sydney.edu.au

Yuhong Xia
University of Sydney
Camperdown NSW 2006
yxia2172@uni.sydney.edu.au

Abstract

This project is built to make classification for 18 classes of fruits. Different parameters are optimized then different models are compared to achieve higher accuracy. It was found that more layers require more computational resources such as advanced GPUs. Therefore, transfer learning method is being used in this paper where only few or last layer of the neural network are trained while preserving parameters that is already trained on Image Net.

1. Introduction

This project is built to make classification for 18 classes of fruits. Distinct kinds of models are used to build the neural networks. The main direction is ResNet and AlexNet. In addition, vgg is also used to make the comparison with ResNet and AlexNet. Parameters are optimized so that the network achieve high accuracy with less loss without overfitting.

2. Previous Work

2.1. AlexNet

This is the first proposed convolutional neural network. The network was made up of 5 conv layers, max-pooling layers, dropout layers, and 3 fully connected layers. The network they designed was used for classification with 1000 possible categories.

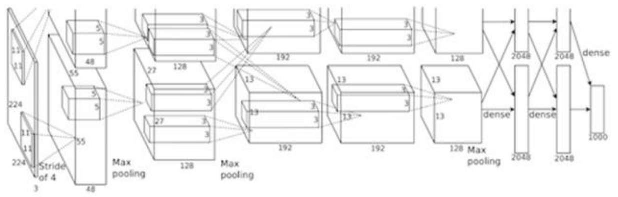
This network is trained on ImageNet data, which contained over 15 million annotated images from a total of over 22,000 categories.

ReLU is used for the nonlinearity functions (Found to decrease training time as ReLUs are several times faster than the conventional tanh function).

Data augmentation techniques that are used are image translations, horizontal reflections, and patch extractions.

AlexNet implemented dropout layers to combat the problem of overfitting to the training data.

Lastly this model was trained using batch stochastic gradient descent, with specific values for momentum and weight decay. The model was trained on two GTX 580 GPUs for five to six days. This model achieved 15.4% error rate.[1]

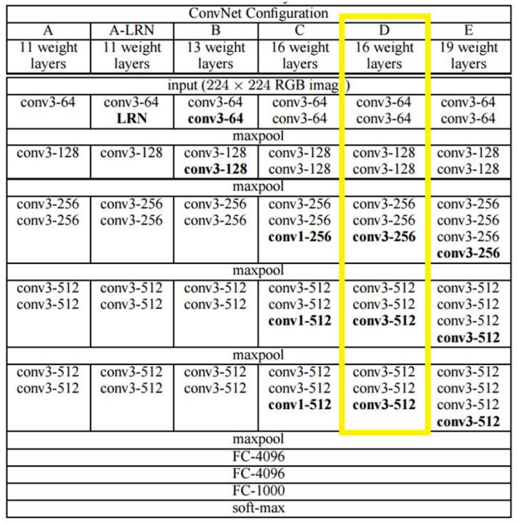


AlexNet architecture (May look weird because there are two different "streams". This is because the training process was so computationally expensive that they had to split the training onto 2 GPUs)

Figure 1: AlexNet Architecture

2.2. Vgg Net

This network promotes simplicity and depths. Created in 2014, this model achieved 7.3% error rate. There are 19 later of CNN that strictly used 3x3 filters with stride and padding of 1, along with 2x2 max pooling layers with stride of 2.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB imag)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

The 6 different architecures of VGG Net. Configuration D produced the best results

Figure 2: Vgg Net Architecture

This network used scale jittering as one of data augmentation technique during training and ReLu layers after each conv layer and trained with batch gradient descent. This network is trained on 4 Nvidia Titan Black

GPUs for two to three weeks.[1]

2.3. ResNet

ResNet is a new 152-layer network architecture that set new records. Aside from the new record in terms of number of layers, ResNet won ILSVRC 2015 with an error rate of 3.6%.

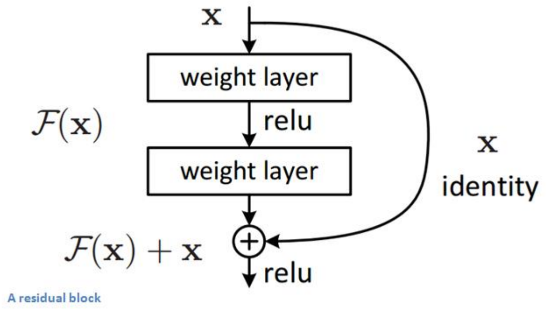

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu
A residual block

Figure 3: A residual block

The idea behind this residual network is that there is an input x that go thorugh conv-relu-conv series. Instead of just computing transformation from x to F(x), the input term x is added to F(x). The module is inputting a delta to get a slightly altered representation.[1]

3. Our Network

3.1. Data Augmentation

3.1.1 Train

During the training set the image are resized into 224 by 224 pixels then doing random horizontal flip. Random horizontal flip is useful for the neural network since convolution processes are done backwards for similar images. Image are then normalized with a mean of [0.485, 0.456, 0.406] and standard deviation of [0.229, 0.224, 0.225]. Those values are used because ImageNet pictures are used for the model.

3.1.2 Test

For the test set, the transformation method that are being used are resizing the image to 256, then doing a center crop by 224. This is done to verify that the neural network can detect pictures of partial images by cropping few part of the original picture.

3.2. Batch Size

This experiment uses 2417 training images and 355 validation images. The batch size for this experiment is chosen to be four. Therefore, there will be 605 iterations in the training set and 89 iterations in the validation set. High batch size will result higher use of memory while lower batch size will result less accuracy.

3.3. Loss Function

3.3.1 Cross Entropy Loss

The loss function that is being used are cross entropy loss. Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label.[2] Cross entropy loss are used so that predicting a probability of 0.011 when the actual observation label is 1 would be a bad and result in a high loss value.

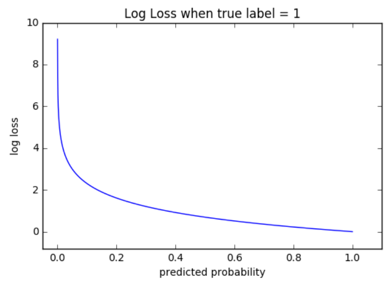


Fig 4: Cross Entropy Loss

3.4. Optimizer

3.4.1 Stochastic Gradient Descent

Optimizer that are being used are stochastic gradient descent with a learning rate of 0.001 and a momentum of 0.9. Using a bigger value for the learning rate may result the neural network to diverge. Using smaller value for the learning rate may result gradient descent to be slow.[3] 0.9 values are the best because with lower values, when doing exponentially weighed averages, sequence can be fluctuating a lot while using higher values one can get a smoother curve, but a little bit shifted to the right. 0.9 gives balance between two extremes.[4]

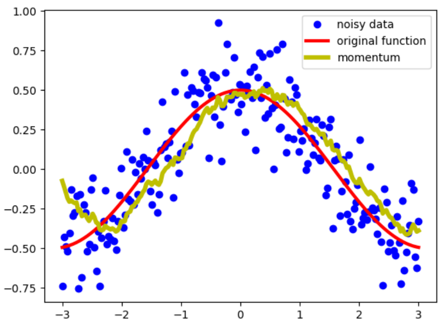


Fig 5: Data from exponentially weighed averages

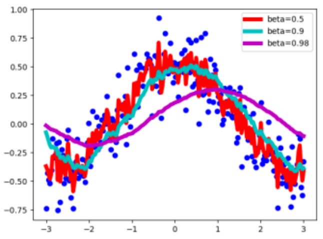


Fig 6: Exponentially weighed averages with different values of beta

3.4.2 Learning Rate Scheduler

Below figure depicts that a high learning rate will lead to a random vector around the local minima. While using slow learning rate will result the vector in getting stuck into false minima. Therefore, knowing this, a learning rate scheduler is important. One should use a higher value of learning rate first then reduce the number of the learning rate to achieve true local minima. The number highly depends on the model. Our algorithm multiples learning rate by 0.1 for every 7 epochs.[5]

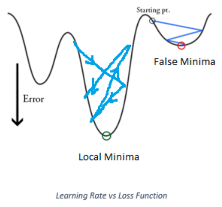


Fig 7: Learning rate vs Loss Function

3.5. Network Used

After using and comparing different models, it is concluded that the best network that achieve highest accuracy are using ResNet152 which was 86%. This is done by using predefined neural network from ImageNet, freezing all the layers then only train the last layer.

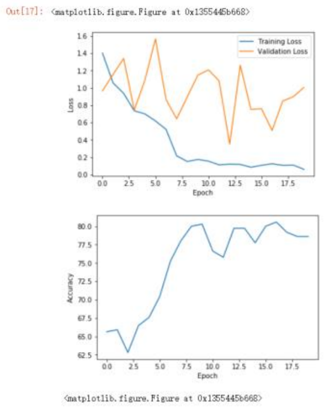
4. Improvements

4.1. Improvements by Changing Different Networks

After choosing different parameters for the neural network, different neural network models were used are compared. 20 epochs are used when experimenting with different models.

4.1.1 AlexNet

After implementing this model, it is found that the accuracy of using Alexnet model is 79.155%. Below is the graph for training loss, validation loss and accuracy for every epoch. The model overfits after training 8 epochs.



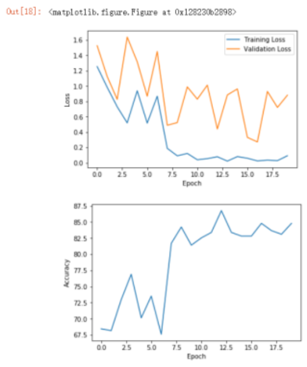
```
Out[17]: <matplotlib.figure.Figure at 0x1355445b668>
```

```
<matplotlib.figure.Figure at 0x1355445b668>
```

Fig 8: Loss and Accuracy Plot for AlexNet Model.

4.1.2 Vgg Net

Using Vgg Net, overfitting point is around 8 epochs. The accuracy is tested to be 83.099% with its maximum point of 86%. Only the last layer of the Vgg Net is trained.



```
Out[18]: <matplotlib.figure.Figure at 0x128230b2898>
```

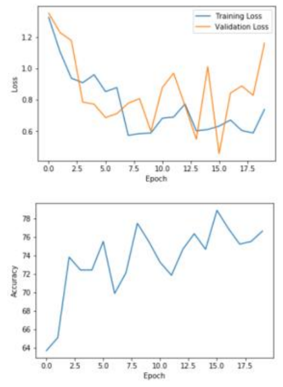
Fig 9: Loss and Accuracy Plot for Training Last Layer of Vgg Net.

4.1.3 ResNet

4.1.3.1 ResNet18

First, only the last layer of the ResNet18 is trained. The existing weights is used from image net. It's overfitted above 16 epochs. Therefore, the maximum accuracy is 78.873%. The following graph shows the result of this process.

```
Out[13]: <matplotlib.figure.Figure at 0x1ddd5c81b70>
```



```
<matplotlib.figure.Figure at 0x1ddd5c81b70>
```

Fig 10: Loss and Accuracy Plot for Only Training the Last layer for ResNet18.

Training all layers of ResNet18 increases the accuracy by 2%, which is 80.45%.

```
Out[14]: <matplotlib.figure.Figure at 0x27081fcf4a8>
```

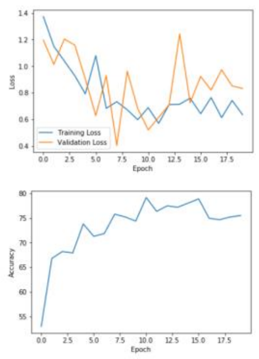


Fig 11: Loss and Accuracy Plot for Training All Layers of ResNet18.

4.1.3.2 ResNet50

With the same process using ResNet50, accuracy was tested to be 80.845%, The following graph shows the trends of the loss. In the graph, it is found that the loss is decreasing, and the accuracy is increasing, which means the result is getting better than the previous model.

```
Out[19]: <matplotlib.figure.Figure at 0x1f3ad9ea400>
```

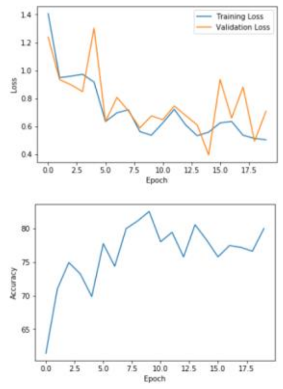


Fig 12: Loss and Accuracy Plot for Training the Last Layer of ResNet50

After freezing 6 layers and training 4 layers out of the 10 layers maximum accuracy is achieved to be 82.563% which is 2% increase from the previous experiment.

```
Out[13]: <matplotlib.figure.Figure at 0x228639342b0>
```

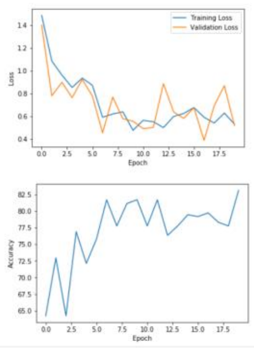
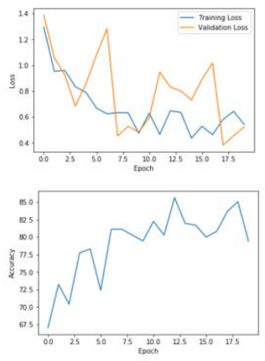


Fig 13: Loss and Accuracy Plot for Training the Last 4 layers out of 10 for ResNet50

4.1.3.3 ResNet152

Lastly, the theoretical best model ResNet152 are used. The following graph indicates the result of using Resnet152.

```
Out[27]: <matplotlib.figure.Figure at 0x28354ba3f98>
```



```
<matplotlib.figure.Figure at 0x28354ba3f98>
```

Fig 14: Loss and Accuracy Plot for Training the Last Layer for

Resnet152

The final accuracy is tested to be 83.944% which is the most accurate. Furthermore, the highest accuracy in this process was 86% which is the same as Vgg model. However when using the Vgg model, it was found that the network overfits after 8 epochs.

5. Conclusion

By research and comparison, it is found that training more layers of the ResNet increases its accuracy and better plot where the training loss and validation loss decreases exponentially. It is proven theoretically that the model ResNet152 has the highest accuracy which is 86% in this experiment. Higher accuracy can be used by training more layers of ResNet152.

6. Future Work

In the future, more layers of ResNet152 will be trained. The process will be the same with what was being done with ResNet18 and ResNet50 where training more layers yields higher accuracy.

References

[1] https://adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html

[2] https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html

[3] https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10

[4] https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d

[5] https://towardsdatascience.com/learning-rate-scheduler-d8a55747dd90