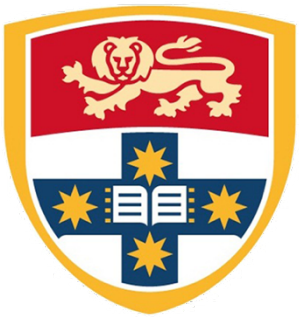
University of Sydney

ELEC5307 Deep Learning – Project #2 Report: Challenge in Image Classification

Xu Yang (480192581), Yeming Chen (480517728), Hangyu Chen(490075948)



THE UNIVERSITY OF SYDNEY

1 Introduction

In this project, the dataset contains 21 kinds of fruit, and we will build a pipeline for fruit classification, then we will use the neural networks' knowledge we learned to select a suitable neuron network with fine-tuned parameters to improve and reach satisfactory accuracy. After this project, we will understand the structure and training process of different neural networks.

2 Previous Work

There are two main methods to accomplish fruit classification: one is to use the fruits' biological characteristics (using the gas sensor to detect the odor of the fruit and do the classification), the other one is to use the images-based classification [1]. The low cost and excellent performance make image-based classification mainstream in the fruit classification [1]. Before the invention of the convolutional neural network, the maximum accuracy for fruit classification was about 90% [2]. In this project, we mainly focused on convolutional neural networks. We will compare the different CNNs (e.g., AlexNet, GoogLeNet etc.). After that, we will modify the structure of these CNNs to improve the overall performance.

3 Methodology

3.1 Dataset, Augmentation, and Training Process

In this project, our dataset contains a total of 2227 images of 21 kinds of fruits. And we set 80% of the dataset to our training dataset, 20% of the total dataset to the validation dataset.

Because some very deep networks are used for this classification task, one important issue is that we must use data augmentation techniques during training for this relatively small dataset to prevent overfitting and improve our model generalization. A few transformation methods are tested when using GoogleNet as a baseline model and compared based on the final accuracy on the validation set. Based upon the comparison results, RamdmCrop, ColorJitter, RandomHorizontalFlip, Normalisation and a few Resize functions are chosen to be used for training networks. After resizing the data to 256 by 256 in shape, RandomCrop crops the data randomly into 224 by 224 in shape. Then ColorJitter with brightness = 0.3 randomly changes the brightness of a picture in a range from -30% to 30%, and RandomHorizontalFlip flip the picture randomly with a probability of 0.5. A ToTensor function is used to convert the intensity values of an image from 0-255 to 0-1, and Nomalize further converts this range into -1-1, which is more suitable for machine learning and weight updating.

The essential parameters of training process are batch size, learning rate, and epoch. These parameter values are chosen by control variable experiments by using GoogleNet as training baseline. After a few sets of experiments, we recorded the best results and determine the parameter values that will be used throughout this project. The batch size we selected in this project is 10, with dynamic learning rate and epoch used and tuned by us during training process by regularly printing and observing the training loss and validation accuracy during each training subprocess of 10 epochs. Besides, we use SGD to build our optimizer and also implement exponential learning rate scheduler with a decay parameter of 0.99 to help tune the learning rate during each subprocess.

3.2 Baseline selection

We test classical networks including AlexNet, GoogLeNet, ResNet (pretrained), and DenseNet to find our baseline to do further modifications based on their final accuracy on the validation set. The initial learning rate is 0.001 which is decreased when the training loss does not change too much during a subprocess (10 epoch). The smallest learning rate we use is 0.00001 for each network because the accuracy will not increase by using any smaller learning rates.

3.3 Network Modification

According to the experimental results, GoogLeNet reaches the highest validation accuracy with very fast convergence speed and small parameter size. Therefore, this network architecture is thought to be suitable for this task and is chosen to be our baseline structure.

3.3.1 Inception

The novel method used in GoogLeNet is the introduction of inception block. Instead of using just one type of inception blocks of the original version, we choose 3 types of inception blocks from the Inception v4 network to use in the modified structure, which contains more sequential CNN layers in some paths with smaller kernels to reduce the parameter size. The structure of the network can then be divided into 4 parts: basic filtering region, Inception A region, Inception B region, Inception C region, output region. In addition, because the inception blocks do not alter the size of the images pass through it, we also add a size reduction block between each 2 inception regions.

3.3.2 Auxiliary Classifier

Another feature of GoogLeNet is the implementation of auxiliary classifier which has the potential to solve the gradient vanishing or exploding problems. Although in the later version like Inception v3 and Inception v4 such classifier was thought as useless in the early stage of the network and thus was used only near the end of the network, experiments with Inception v3 yields worse validation accuracy compared to the original GoogLeNet on our dataset. As a result, we choose to maintain the 2 auxiliary classifiers in our network backbone and add the classifier outputs (with a weight of 0.3) to the final output of the network.

3.3.3 Shortcut Connection

As suggested by networks like ResNet and DenseNet, the use of shortcut connection, namely, the residual learning can reduce the negative impact of deepening the network structure by skipping some blocks and increase the training efficiency. To make use of this advantage to our network, we add such shortcut connection in each inception block except the size reduction blocks. Within such shortcut connection, we add nothing or a 1 by 1 CNN filter to change the input image channel size, depending on whether the current block operation change the image channel size. After each connection end we add the output to get F(x) + x or F(x) + Wx, where W is the channel resize operation, and pass the result to an activation function, namely, ReLu layer.

3.3.4 Final Network Architecture

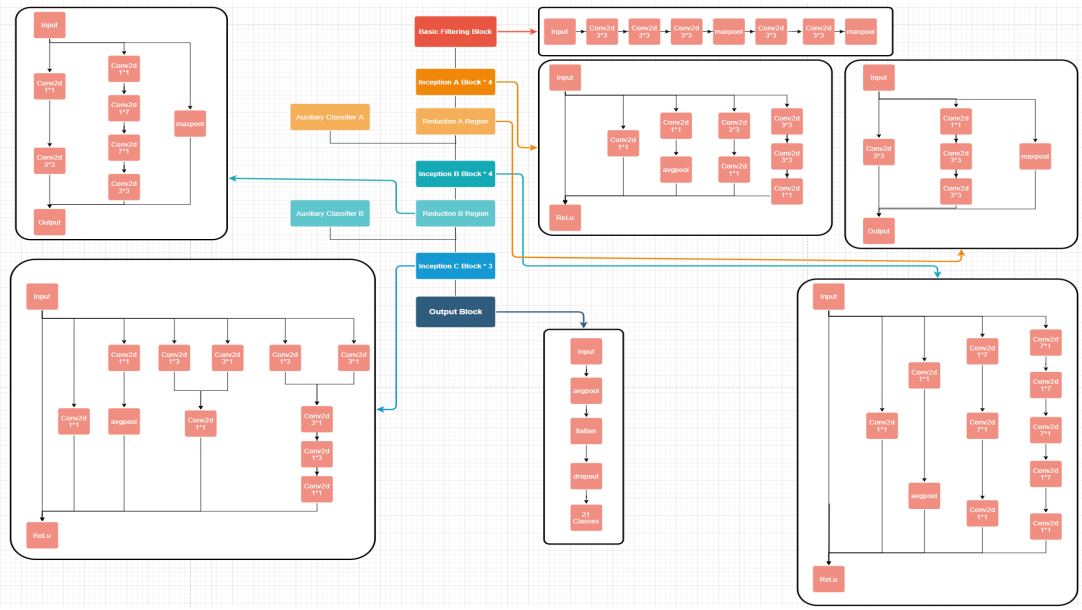


Figure 1-Network Architecture

4 Experiments

4.1 Baseline Choosing Process

4.1.1 GoogLeNet

Before we use GoogLeNet, we tried the AlexNet first. The AlexNet has a simpler structure compared with other networks. So it is proper for us to begin our project from AlexNet. The shallow layer structure reduced the difficulty of modification. However, the result we ran from the AlexNet is not acceptable. The accuracy after the first epoch is only 13%. Through the previous experience, our group decided to use a deeper network. GoogLeNet contains 22 layers, and apart of these layers, there are 9 inception modules. This structure provided us with a deeper network and reduced the parameters. And GoogLeNet has various convolutions filters, which can make it more efficient in image classification. After the five epochs, the accuracy we got from GoogLeNet reached 80%. And compared to the ResNet and DenseNet, the GoogLeNet has a more straightforward structure, which means we can make more changes to improve the performance. So, our group decided to use GoogLeNet as our baseline. But before that, we still experimented with the other networks to check whether GoogLeNet was the best option for us.

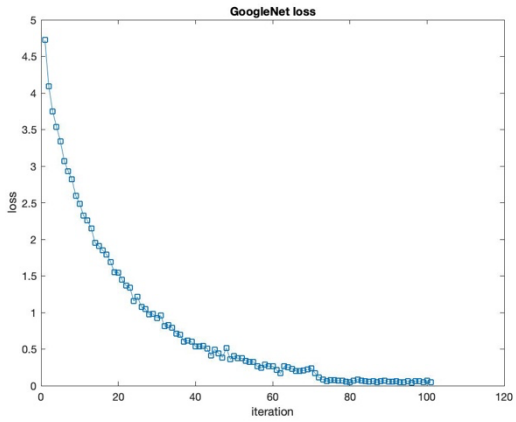


Figure 2.1 – GoogLeNet Training Loss

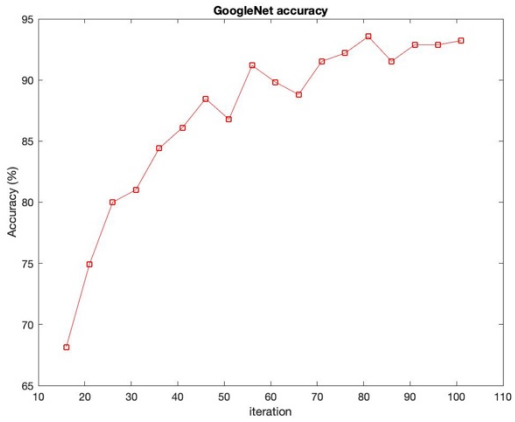


Figure 2.2 GoogLeNet Validation Accuracy

4.1.2 Pretrained ResNet

Although Hastad (1987) claimed that deep circuits are more efficient in representing certain Boolean functions than shallow circuits [3], which some people used to believe deeper neural networks can represent functions that shallow networks cannot. However, as we increased the depth of our baseline, we started to think about the drawback of deeper networks. A report said the deeper network is harder to train, and in some situations, the deeper network may have higher train error and test error [4]. As there are different versions of ResNet model and the main difference is the depth of them. So, here we used ResNet18 and ResNet50 to check if this situation exists in our project. ResNet uses network layers to fit a residual mapping rather than provide a desired underlying mapping directly.

For ResNet18, the accuracy increased to 25% in 15 epochs (left Figure). However, if we choose ResNet50, the accuracy 87% in 6 epochs (right Figure).

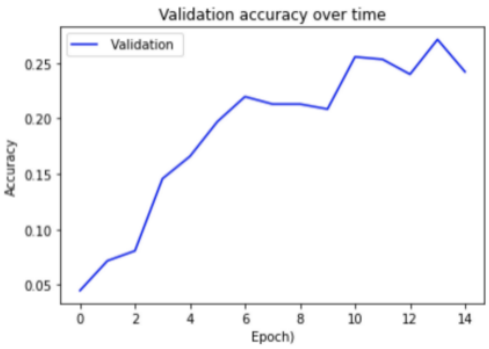


Figure 3.1 - ResNet-18 Validation Accuracy

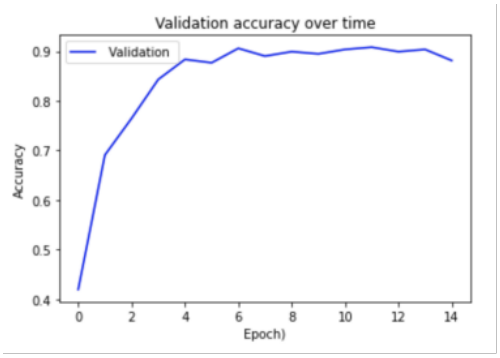


Figure 3.2 ResNet-50 Validation Accuracy

And it can prove that our baseline (GoogLeNet) did not come to the 'deeper worse' phenomenon. And due to the simpler structure of GoogLeNet, we still decided to use it as our baseline.

4.1.3 Other Networks

Except the AlexNet, GoogLeNet, and ResNet. We also tried the DenseNet and Inception V3. However, in this project, the performance for these networks is not as good as GoogLeNet. Moreover, GoogLeNet has sampler structure than these networks. So, we select the GoogLeNet as our baseline.

4.2 Network Modification

Based on the GoogLeNet, we create 5 different versions by modifying the structure of the network. Below is the performance for these 5 versions. And the detailed illustration and discussion will be in the discussion part.

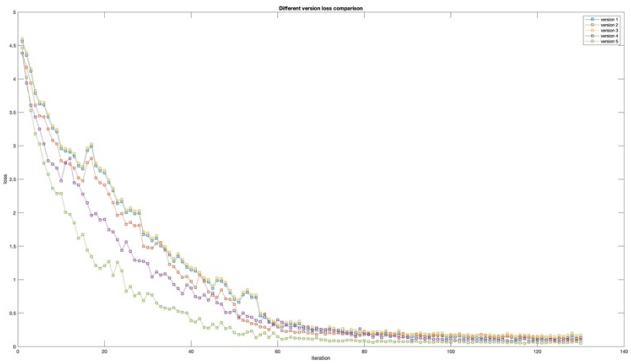


Figure 4 - Loss Comparison

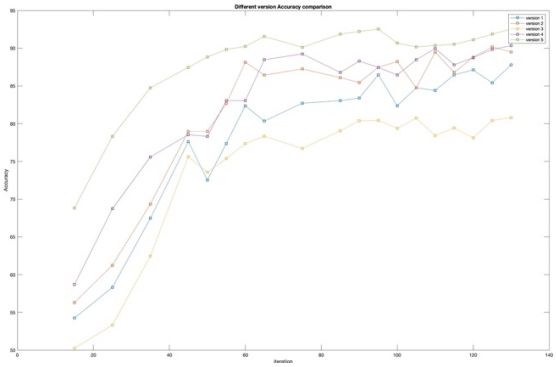


Figure 5- Accuracy Comparison

5 Discussion

5.1 Dataset and Preprocessing

Our group discussed the possibility of using k-fold cross-validation rather than arbitrary divided into 80% and 20%. However, we think that k-fold cross validation will increase our training time. And due to this consideration, we did not use k-fold cross-validation in this project.

5.2 Baseline Choosing Discussion

From the previous work, we found that the AlexNet cannot reach high accuracy as we expected, and training in AlexNet is time-consuming. The DenseNet and ResNet both have their own unique features, but the structure for these two networks after complex, which means they are challenging to modify. So, the DenseNet and ResNet are not good options for baseline. Finally, our group selected GoogLeNet as our baseline. We valued its accuracy, inference time, and its

potential for modification.

5.3 Network Modification

As for the network modification process, we choose the backbone of GoogLeNet with use of auxiliary classifiers and various inception blocks.

In **Network Version 1**, we use a relatively deep network structure to do the classification task. Based on this result, we do further modifications on the inception block depth, output channel size, and shortcut connection. From the learning curve of this situation, we notice that such deep network with large parameter size is difficult to train and the convergence speed is also slow. Moreover, at the end of the training, the validation accuracy stops at 85%, which is not as expected. This means a deep network may not always perform well on a specific dataset.

In **Network Version 2**, we disable 3 inception blocks in the middle of the network and use the same training technique to investigate its performance. For this time, the convergence speed of the network increases under the same learning rate, and the final validation accuracy reaches 90%. This result confirms our guess from the previous experiment, that is, a shallow network is more suitable to our dataset. Nevertheless, we notice that the pth file of this network is 194 MB which is too large compared to that of other networks. We decide to reduce the output channel size of our network to decrease the parameter size and see if we can get the same or even better results.

In **Network Version 3**, We reduce all the output channel size of the inception and channel reduction blocks into half of their original size and train the network from the beginning. Unfortunately, the resulting pth file reduces to 75 MB with the expense of the final validation accuracy of just around 80%. This experiment suggests that, by using too small output channel size, a specific network may perform worse on the same dataset. To overcome this effect, we try to use shortcut connection.

In **Network Version 4**, the shortcut connection similar to that of ResNet is implemented in each inception block except the size reduction blocks. After the experiments, the final validation accuracy reaches almost 90%, which confirms the usefulness of shortcut connection on learning efficiency increasing.

In Network Version 5, we turn all the output channel sizes inside the network to their original values and do a comparative study with Version 4. By using this network, we achieve the highest validation accuracy among all our modified networks, which is almost 93%. Nonetheless, the pth file has a size of 222 MB. This is obviously a trade-off between smaller parameter size and higher classification accuracy.

6 Conclusion and Future Work

6.1 Conclusion

In this project, we test several neural networks and compared their performance on fruit image classification. By observing the loss and accuracy diagram we determined the baseline for this project. After that we preprocess the data and modified the structure of the network to get a better performance. Through this project, we learned how to build a pipeline for fruit image classification, and we also understand the structure and training process of different neural networks.

6.2 Future Work

During the experiment, we find that the learning rate scheduler is not intelligent enough. The learning rate scheduler will decay the learning rate in the pre-set function. However, to get a good performance, we need to find the relationship between the learning rate and the changes in training loss. So, in future, we can write a function to change the learning rate by calculating the difference in training loss.

7 Reference

[1] Yu-Dong, Z., Dong, Z., Chen, X., Jia, W., Du, S., Khan, M., & Shui-Hua Wang. (2019). Image based fruit category classification by 13-layer deep convolutional neural network and data augmentation. Multimedia Tools and Applications, 78(3), 3613-3632. doi:http://dx.doi.org/10.1007/s11042-017-5243-3

[2] Lu Z, Li Y (2017) A fruit sensing and classification system by fractional fourier entropy and improved hybrid genetic algorithm. In 5th International Conference on Industrial Application Engineering (IIAE). Kitakyushu, Institute of Industrial Applications Engineers, Japan, pp 293–299

[3] Hastad, J. T. (1987). Computational Limitations for Small Depth Circuits. MIT Press, Cambridge, MA.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. (2016). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778. IEEE. https://doi.org/10.1109/CVPR.2016.90

8 Appendix

8.1 Group Member Contributions

Xu Yang: Divided the dataset into training and validation set. Developed training code for baseline selection. Performed training on GoogLeNet, Inception v3, ResNet, DenseNet and recorded the results. Developed the modified networks, did the training and validation, and recorded the results. Participated in report writing with a focus on the methodology and modified networks discussion parts.

Yeming Chen: Visualize the performance of GoogLeNet, ResNet, DenseNet, and different versions of our pipeline. Help with modifying the networks and debugging. Mainly focused on report writing. Completed the introduction, previous work, conclusion, and future work part. Moreover, helped with the methodology and discussion part.

Hangyu Chen: Mainly focusing on setting up edited pre-trained ResNet18 and ResNet50, do the comparison between them by visulizing their results (accuracy and loss) in colab and mention it in the report.

8.2 Hardware Specification

cpu type: AMD Ryzen 7 5800X 8-Core Processor 3.80 GHz

RAM: 32.0 GB

GPU type: NVIDIA GeForce RTX 3070

RAM: 32.0 GB

pytorch version: 1.9.1