Lab 2: Residual layers, squeeze and excitation

Introduction

In this lab, we are going to study the example of ResNet implementation and implement the ResNet techniques on the AlexNet model. For the individual part, we will implement the squeeze and excitation (SE) techniques on AlexNet model as well. We also do an experiment to find the best place to put the SE layer in the model which gives the best accuracy. The goal of this lab is to learn how to implement a custom architecture from a previous implemented model or from scratch using the PyTorch framework.

ResNet technique is to copy to preserve the information as an identity and adding them back to the output from the basic block layer of a particular architecture. Surprisingly, from the paper “Deep Residual Learning for Image Recognition”, they claim that this technique can solve the degradation problem when the model gets more dept.

Squeeze and excitation techniques also do the copy but not for preserve data. The copied data are squeezed into a smaller tensor before scale back to the original dimension again (use the squeezed data multiply with original data).

ResNet 18 Test

In this step, we do an experiment to train and find accuracy by using CIFAR-10 data sets on the PyTorch’s ResNet 18. We use the implemented training function from PyTorch official site. The parameters that use for training are

- Training for 25 epochs
- Cross entropy as a loss function
- Stochastic gradient descend as optimizer with
    - Learning rate = 0.001
    - Momentum = 0.9

Results of ResNet 18 Test

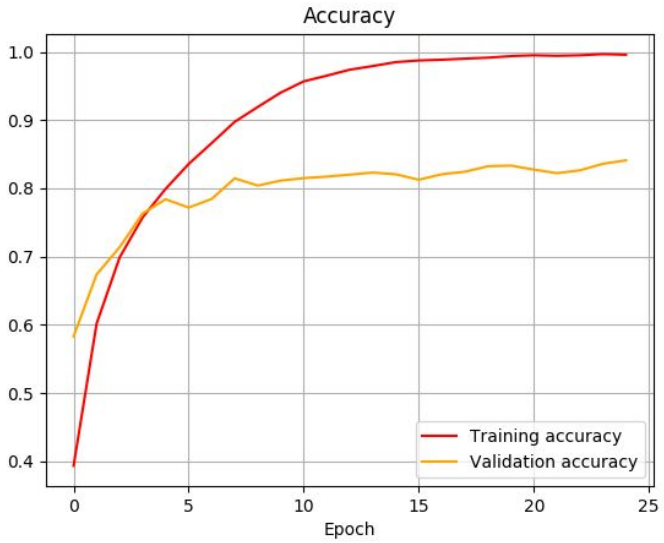


Figure 1. Accuracy and train epochs of original ResNet18

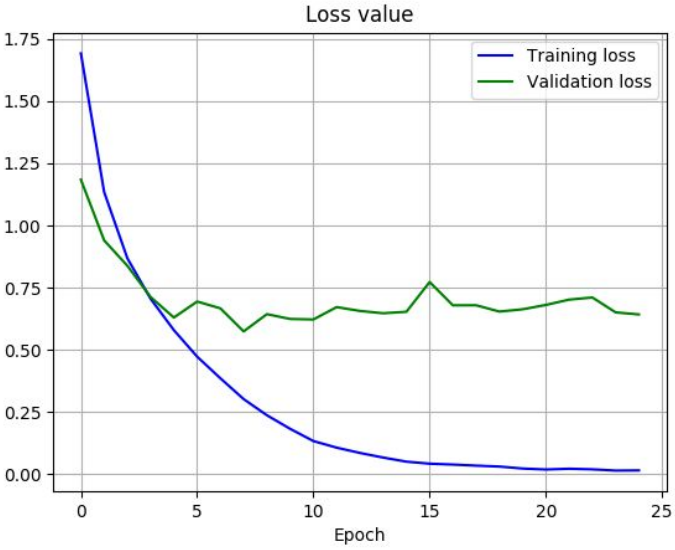


Figure 2. Loss value and train epochs of original ResNet18

Training time = **140 minutes and 22 seconds**
Best validation accuracy = **84.10%**

There is no sign of overfitting since both training and validation accuracy still increase at the end of the experiment. We might see better accuracy on the model if we increase the number of training epochs.

AlexResNet implementation and evaluation

For this step, we study the structure of ResNet and try to understand it (Which I think I get the concept of ResNet technique). After architecture study, we modified the structure of AlexNet to use the ResNet technique which is we called it “AlexResNet”.

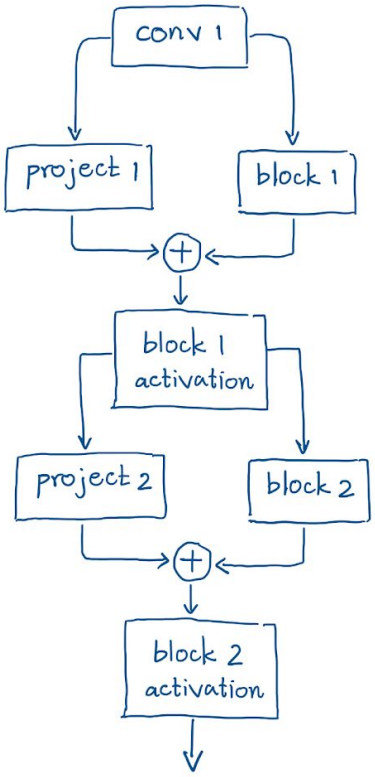


Figure 3. AlexResNet architecture.

From figure 3, we preserve data by copying them to the path which only has a project layer before it passes the block1 layer then we add back after the data passes through the block1 layer. We also do the same on the block2. Preserve the data before passing the block2 layer and add back.

The parameters that use for training are

- Training for 25 epochs
- Cross entropy as a loss function
- Stochastic gradient descend as optimizer with
    - Learning rate = 0.001
    - Momentum = 0.9

Results of AlexResNet evaluation

Figure 4. Accuracy and train epochs of AlexResNet

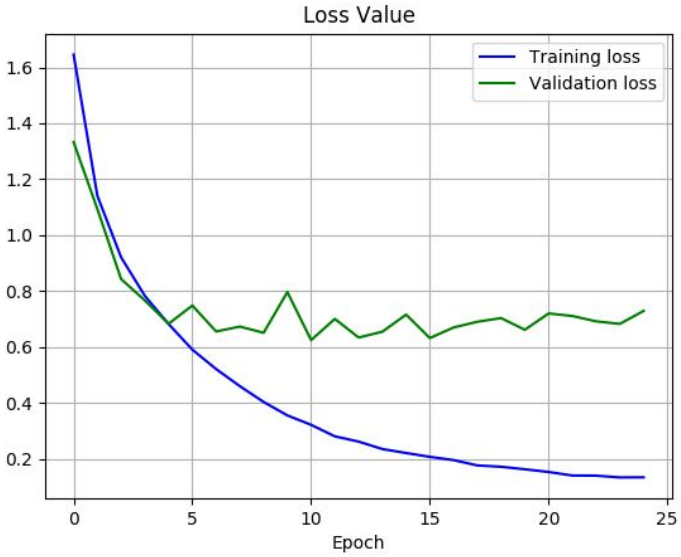


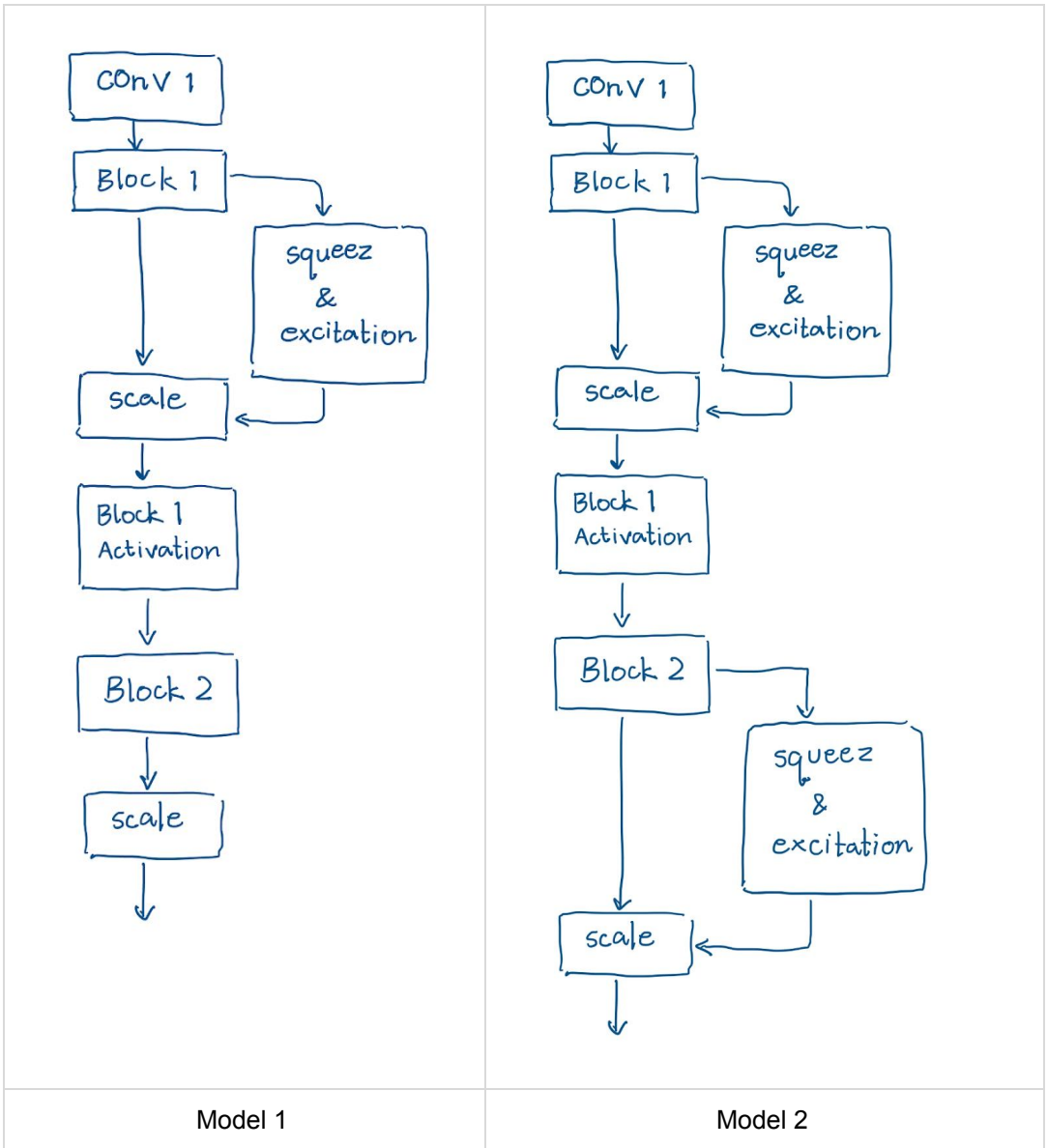
Figure 5. Loss value and train epochs of AlexResNet

Training time = **87 minutes and 38 seconds**
Best validation accuracy = **0.8076**

Compared to the ResNet18 model, AlexResNet has a better training time but has less accuracy. After the 12th epoch, the validation accuracy of the model started to steady. There is no significant evidence of overfitting.

AlexSE implementation and evaluation

In this experiment, we implement the squeeze and excitation on the AlexNet model and observe any changes in model accuracy. We also try to find which is the best approach to implement the squeeze and excitation on the AlexNet model, where we should put the squeeze and excitation block on the model to give the best accuracy. There are three approaches that I have done in this experiment as shown in the figure below.



Model 1 | Model 2

Figure 6. The architecture of each experiment model.

The first model has the squeeze and excitation between block1 and block1 activation. The second has the operation after block2 while the second model has the operation after both block1 and block2.

AlexSE evaluation results

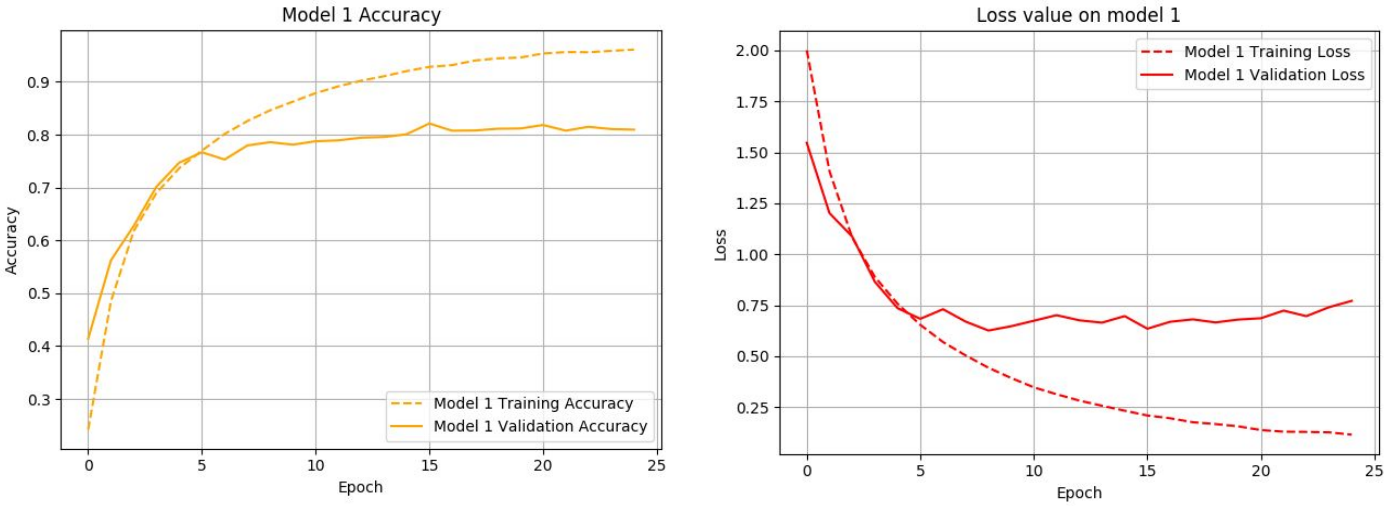


Figure 7. Model 1 Accuracy and 1 Loss value

Training complete in **161m 34s**
Best **model 1's** validation accuracy: **0.821100**

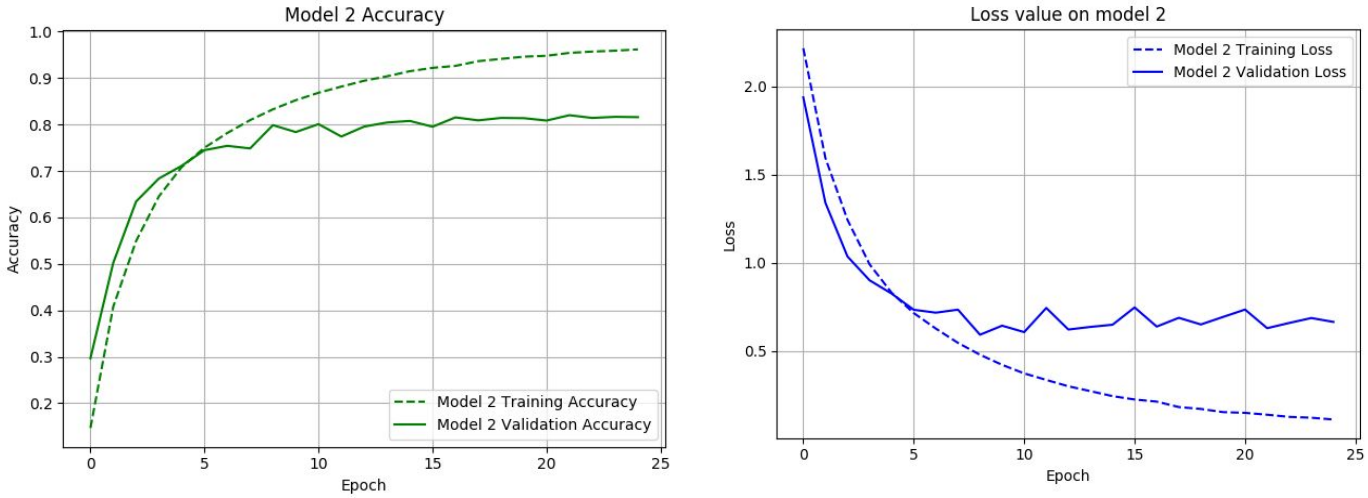
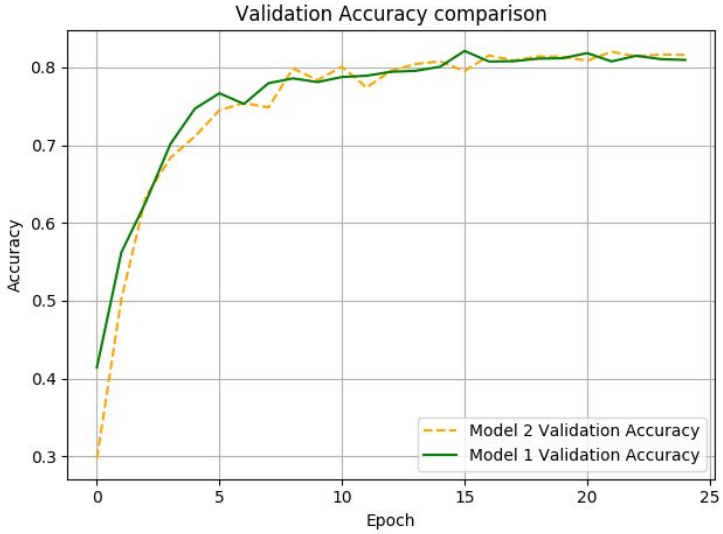


Figure 8. Model 2 Accuracy and Loss value

Training complete in **136m 10s**
Best **model 2's** validation accuracy: **0.820000**

Nearly the end of the training on the model 1, the validation accuracy has slightly decreased. This might be a sign of overfitting in the model 1.
Surprisingly!, model 2 has the most complexity compared to all others models but model 2 training time is less than model 1. The reason is the training time does not only depend on the model complexity but it may depend on the current workload of the server.



When we do a validation accuracy comparison, both models do not have any significant differences. At nearly the end of evaluation, the model 1 has a little accuracy drop when compared to the model 2 which is the evidence of overfitting. If we continue training, the model 1 might lose its accuracy due to overfitting.

To justify which model is better, it is still hard to justify from only 25 epochs of training. We may increase training epochs to observe which model will lose its accuracy first. If to justify the better model from current information that we have, **I agree that model 2 is a little better** than the first model since it has no evidence of overfitting.

Conclusion

In this lab, I learned a lot and understand the concept of Pytorch programming, modify the architecture of the previously implemented network and build the network from scratch. Most importantly, I have learned that I shouldn't do the training on the night before the day of the report submission since the workload of the training server is high.

For the next step, In the AlexSE implementation and evaluation, we may implement another architecture which takes the output after the block1 through squeeze and excitation operation and scale back before the block2 activation and evaluate its accuracy. We may experiment with hyperparameters as well such as learning rate and momentum.