Exercise #1

Goal:

For the first exercise, as I am participating in a Kaggle competition project that requires image classification, I decided to build a preliminary Convolutional Neural Network model. To extend my knowledge and practice on CNN, I will implement the 8 layer structure mentioned in AlexNet introduced in class.

sample data from 10 categories

The dataset is from a Kaggle competition from a program called BTTAI that I am participating in right now. We are asked to train an image classification model that distinguishes plant/specimen pictures from New York Botanic Garden. The training dataset has 80000+ entries, which is huge and can take up a long time to train. To inspect the preliminary results, I've decided to start with only 10% of the training dataset and the validation dataset. I used the .flow_from_dataframe function in tensorflow to link the metadata and the image files, and the ImageGenerator to apply data augmentation to prevent overfitting. As a result, the images are prepared as (224,224) sized, 32 batches.

The 2 models I used in this exercise are 1) basic CNN, 2) AlexNet. Basic CNN contains 3 convolutional layers, each followed by a max pooling layer, a flatten layer, and 2 dense layers. This structure is inspired by the basic model that tensorflow provided. AlexNet contains 5 convolutional layers, each followed by a batch normalization layer, and the first 2 followed by a max pooling layer. After the flatten layer, AlexNet has 3 dense layers, in between are 2 dropout

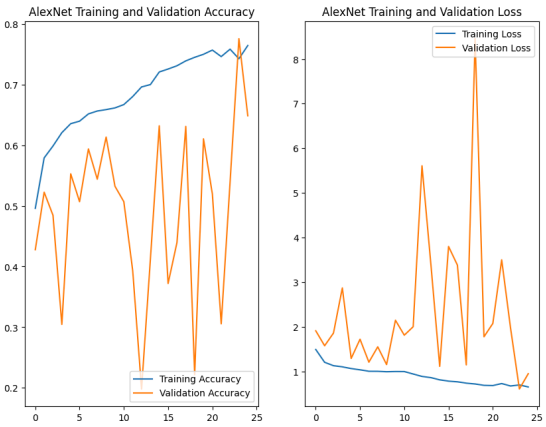
layers. Since tensorflow is not built in AlexNet, this structure is inspired by the AlexNet architecture and some available models.

For evaluation, because it's a Kaggle competition, the test dataset does not contain labels to evaluate the model's final performance. Thus, I used the validation dataset to evaluate the model's performance, and performed predictions on the test dataset. I generated Accuracy & Loss plots as well as confusion matrix to extract insights from the training result for epoch changing and examination of predictions on different categories. After submitting the test file, I will also receive results from the competition to see the real test accuracy.

Here is a training results chart:

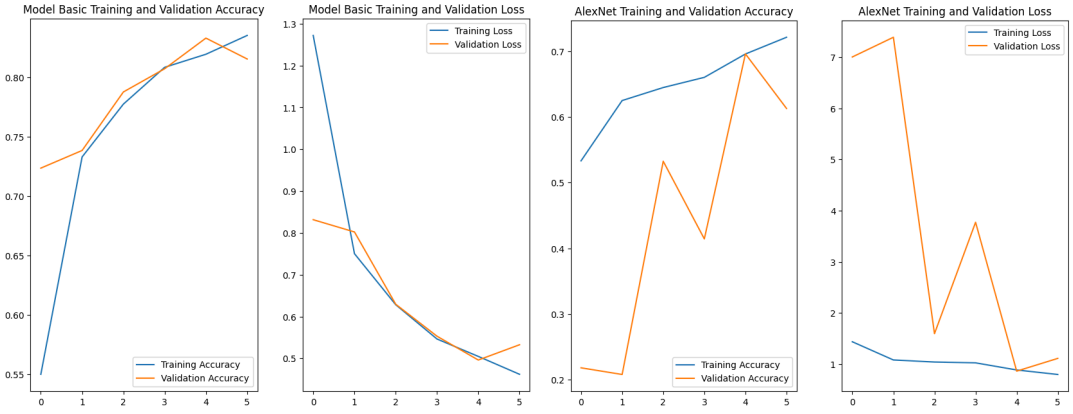
Model	Dataset size	Epochs	Accuracy
Basic	10%	15	0.10
AlexNet	10%	25	0.54
AlexNet	20%	10	~0.60
Basic	25%	6	0.815
AlexNet	25%	6	0.613

The Basic model first performs no better than random guessing, so I continued with AlexNet. AlexNet starts with a much higher accuracy after the first epoch, which is very promising. However, as shown on the left, the accuracy rate for validation is very unstable. The trend is growing, but its peak is very random, so I've decided to do around 5-10 epochs which will also save time. The second iteration of AlexNet uses 20% of the data, and the first epochs seemed to yield better accuracy. Unfortunately it crushes midway. To conclude the exercise, I decided to use a larger portion of the dataset, and train both models one more time and compare the final results.

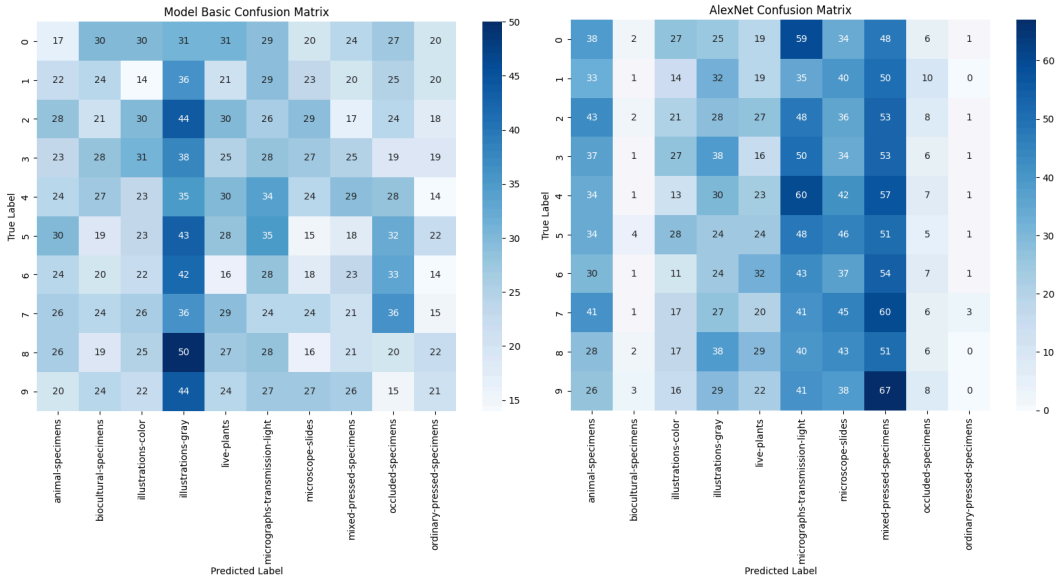


The final training result for Model Basic is better than I expected, and the trend shows slight overfitting, so it seems to be a good point to stop. The final loss is 0.53, and the final

accuracy is 0.82. The Loss and Accuracy plots show good training progress. The confusion matrix, on the other hand, shows that the model does not tend to predict the right labels the most. “Illustrator-gray”, for example, is the label that the model tends to predict the most, especially for occluded specimens. They do appear similar, so for further data preprocessing and model modification, I might consider training separate models to distinguish these two categories separately from the others. AlexNet, on the other hand, does not show significant improvement in accuracy when fed with a larger dataset. The validation appears unstable as before, and the confusion matrix shows that the model’s prediction is very biased with low preference on category 1,8,9. I have no idea why this happens, and am leaning towards not using AlexNet for future training, as this architecture is not exactly what was proposed in the paper but a variation, so it’s hard to tune it or extract further information efficiently.



Model Training Accuracy & Loss (left: Model Basic; right: AlexNet)



Confusion Matrix (left: Model Basic; right: AlexNet)

Because neural networks are powerful themselves, and training/ tuning can be really time consuming, I think the best stretch for my future implications are: 1. try combining multiple models together to perform ensemble methods, and the models don't necessarily need to be neural networks; 2. implement some built-in neural networks in tensorflow like Inceptions and VGGNet, which are said to be suitable for large-scale image classification; 3. apply different models like transformers, transfer learning, which might be covered in class that I can learn from.