DEEP LEARNING

A. Using an Existing Model (Cycle GAN)

In implementing Cycle GAN, I read the images from the directory I defined as the first step. The next step is I preprocessed the images by random flipping of left and right, resizing the images with the shape of (100, 100) and normalize the images into the range -1 to 1. I only took the data of 2000 for training and 1000 for testing because it would take more execution time when I processed all the images just only for one epoch and I didn't get a good result after waiting for 10 hours.

The next step I did is that I built blocks which are used in the Cycle-GAN generators and discriminators. Then, I built the generators, discriminators and the models. Creating the callbacks I found is also necessary to be implemented so that it can save the images periodically.Then, create the loss function for evaluating adversarial loss, defining the loss function for the generators, for the discriminators, creating the GAN Model, calling the callback and fitting into the model.

The training stopped after 71 epoch because of the run time of Google Colab, but the checkpoint of 69 and 70 already look quite well which I believe if the fitting model wasn't interrupted, it would get better as the loss function value is reduced.

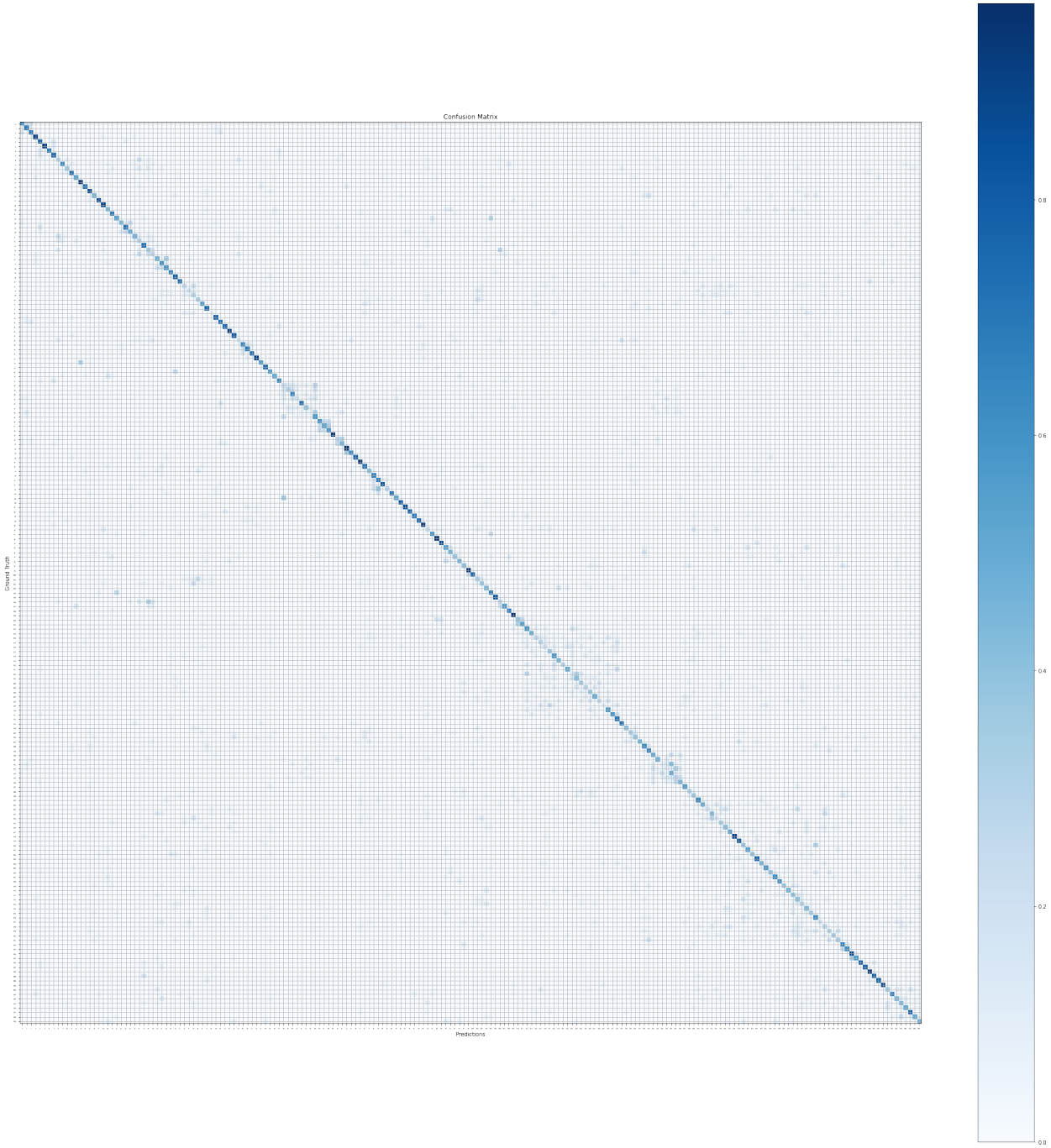
B. Fine-Grained Classification

There are 200 classes of birds that my model should predict. The first step I did is creating a dataframe by reading four txt files. The next step is that we need to split the dataset we've created into three datasets such as the training, validation and test set. The training image is identified with 1 in the column called "is_training_image". I splitted the training dataset into a training and validation set with 25% probability. Otherwise, my generator will only produce less than 200 classes in the validation dataset. It would also affect the accuracy score later on. After that, I created the image generator and used it to create training, validation and test generators. I also shuffled the images when I splitted the images to prevent the model from being biased to only predict certain classes.

Then, I created a model using a pretrained model called Inception V3. Inception V3 has been proved to be computationally efficient in number of parameters and computational cost (memory and other resources). I got the weights from Inception then I use global average pooling 2D to get the average of the features and feed it into 200 classes. But, I dropped the layers up to 70% and connected it to 512 nodes in a hidden layer before passing it into 200 classes. I chose the optimizer Adam with the learning rate of 0.001 as the starting point which will decay little by little until the 30th epochs.

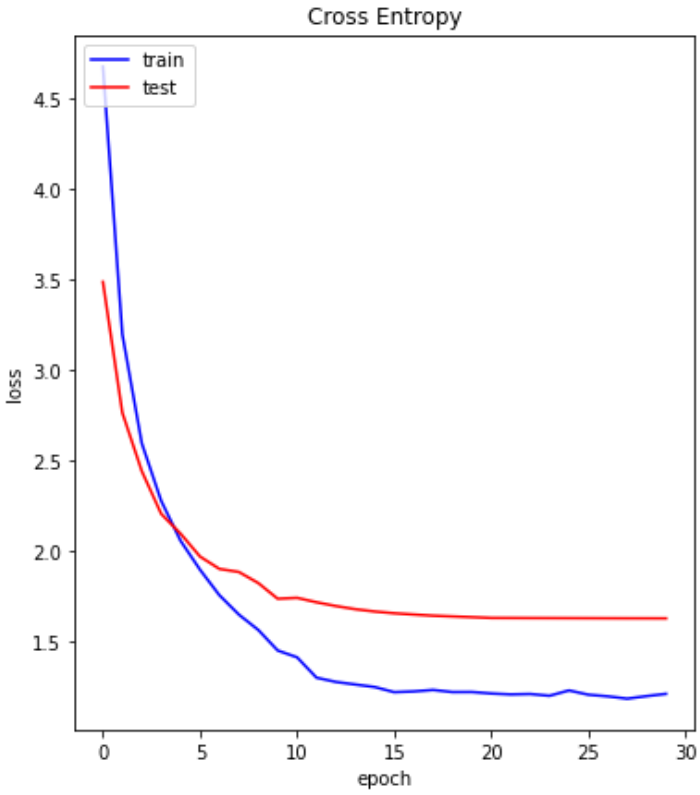
I also implemented the fine-tuning for 300 layers of the model. The way I do it is that I unfreeze the base model and set the bottom layers to be untrainable. Then, I optimize it using SGD with a little learning rate. The best result I could achieve so far is 55% by taking the weight of the transferable learning of the previous model on the 30th epoch to 40th epoch.

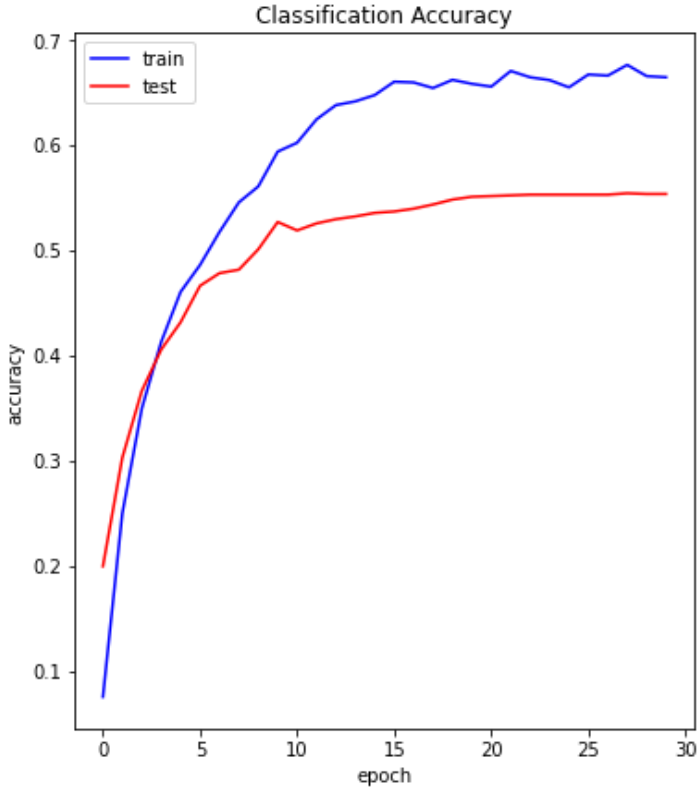
Here is the confusion Matrix shown as follows:



B.1. Confusion Matrix

Here is the cross entropy and classification accuracy can be depicted as follows:





B.2. Cross Entropy an Classification Accuracy

Here is the classification report shown as follows:

precision	recall	f1-score	support
0,678571	0,633333	0,655172	30
0,6	0,7	0,646154	30
0,588235	0,714286	0,645161	28
0,72973	0,9	0,80597	30
0,407407	0,785714	0,536585	14
0,909091	0,909091	0,909091	11
1	0,695652	0,820513	23
0,56	0,777778	0,651163	18
0,258065	0,275862	0,266667	29
0,730769	0,633333	0,678571	30
0,47619	0,333333	0,392157	30
0,689655	0,769231	0,727273	26
0,666667	0,533333	0,592593	30
0,725	0,966667	0,828571	30
0,869565	0,714286	0,784314	28
0,714286	0,892857	0,793651	28
0,705882	0,444444	0,545455	27
0,722222	0,866667	0,787879	15
0,787879	0,896552	0,83871	29
0,382353	0,448276	0,412698	29
0,7	0,7	0,7	30
0,608696	0,538462	0,571429	26
0,619048	0,448276	0,52	29
0,615385	0,727273	0,666667	22
0,555556	0,5	0,526316	30
0,538462	0,466667	0,5	30
0,236842	0,3	0,264706	30
0,611111	0,758621	0,676923	29
0,25	0,333333	0,285714	30
0,296296	0,266667	0,280702	30
0,451613	0,466667	0,459016	30
0,358974	0,608696	0,451613	23
0,472222	0,586207	0,523077	29
0,703704	0,655172	0,678571	29
0,6	0,8	0,685714	30
0,793103	0,766667	0,779661	30
0,333333	0,241379	0,28	29
0,277778	0,166667	0,208333	30
0,195122	0,275862	0,228571	29

0,47619	0,333333	0,392157	30
0,607143	0,566667	0,586207	30
0,560976	0,766667	0,647887	30
0,285714	0,068966	0,111111	29
0,727273	0,8	0,761905	30
0,628571	0,733333	0,676923	30
0,741935	0,766667	0,754098	30
0,651163	0,933333	0,767123	30
0,694444	0,833333	0,757576	30
0,357143	0,166667	0,227273	30
0,666667	0,666667	0,666667	30
0,677419	0,7	0,688525	30
0,851852	0,766667	0,807018	30
0,870968	0,9	0,885246	30
0,708333	0,566667	0,62963	30
0,884615	0,766667	0,821429	30
0,53125	0,566667	0,548387	30
0,75	0,5	0,6	30
0,62069	0,642857	0,631579	28
0,205882	0,233333	0,21875	30
0,473684	0,310345	0,375	29
0,465116	0,666667	0,547945	30
0,384615	0,166667	0,232558	30
0,821429	0,766667	0,793103	30
0,52381	0,366667	0,431373	30
0,090909	0,05	0,064516	20
0,4	0,6	0,48	30
0,580645	0,6	0,590164	30
0,571429	0,533333	0,551724	30
0,529412	0,6	0,5625	30
1	0,966667	0,983051	30
0,32	0,266667	0,290909	30
0,424242	0,466667	0,444444	30
0,659091	0,966667	0,783784	30
0,612903	0,633333	0,622951	30
0,875	0,777778	0,823529	27
0,725	0,966667	0,828571	30
0,647059	0,733333	0,6875	30
0,428571	0,413793	0,421053	29
0,588235	0,666667	0,625	30
0,512195	0,7	0,591549	30

0,862069	0,833333	0,847458	30
0,8	0,266667	0,4	30
0,88	0,733333	0,8	30
0,545455	0,521739	0,533333	23
0,735294	0,833333	0,78125	30
0,896552	0,866667	0,881356	30
0,96	0,8	0,872727	30
0,875	0,7	0,777778	30
0,888889	0,8	0,842105	30
0,756757	0,933333	0,835821	30
0,26087	0,2	0,226415	30
0,692308	0,6	0,642857	30
0,763158	0,966667	0,852941	30
0,764706	0,866667	0,8125	30
0,444444	0,533333	0,484848	30
0,482759	0,466667	0,474576	30
0,52381	0,37931	0,44	29
0,666667	0,466667	0,54902	30
0,48	0,4	0,436364	30
0,878788	0,966667	0,920635	30
0,8	0,8	0,8	20
0,236842	0,3	0,264706	30
0,314286	0,366667	0,338462	30
0,5	0,433333	0,464286	30
0,419355	0,684211	0,52	19
0,925926	0,833333	0,877193	30
0,285714	0,266667	0,275862	30
0,68	0,566667	0,618182	30
0,677419	0,7	0,688525	30
0,74359	0,966667	0,84058	30
0,458333	0,366667	0,407407	30
0,607143	0,566667	0,586207	30
0,3	0,6	0,4	20
0,517241	0,5	0,508475	30
0,388889	0,241379	0,297872	29
0,296296	0,266667	0,280702	30
0,2	0,172414	0,185185	29
0,407407	0,366667	0,385965	30
0,377778	0,586207	0,459459	29
0,538462	0,466667	0,5	30
0,692308	0,3	0,418605	30

0,535714	0,5	0,517241	30
0,142857	0,133333	0,137931	30
0,352941	0,413793	0,380952	29
0,4	0,344828	0,37037	29
0,5	0,266667	0,347826	30
0,289474	0,366667	0,323529	30
0,454545	0,5	0,47619	30
0,28	0,233333	0,254545	30
0,333333	0,1	0,153846	30
0,485714	0,566667	0,523077	30
0,580645	0,6	0,590164	30
0,423077	0,733333	0,536585	30
0,6875	0,733333	0,709677	30
0,48	0,4	0,436364	30
0,5	0,333333	0,4	30
0,354839	0,366667	0,360656	30
0,933333	0,466667	0,622222	30
0,527778	0,633333	0,575758	30
0,526316	0,666667	0,588235	30
0,454545	0,517241	0,483871	29
0,5	0,466667	0,482759	30
0,2	0,1	0,133333	30
0,16	0,133333	0,145455	30
0,340909	0,5	0,405405	30
0,228571	0,266667	0,246154	30
0,424242	0,466667	0,444444	30
0,53125	0,566667	0,548387	30
0,45	0,310345	0,367347	29
0,37931	0,366667	0,372881	30
0,482759	0,666667	0,56	21
0,535714	0,5	0,517241	30
0,5	0,172414	0,25641	29
0,393939	0,433333	0,412698	30
0,190476	0,133333	0,156863	30
0,277778	0,333333	0,30303	30
0,324324	0,413793	0,363636	29
0,363636	0,533333	0,432432	30
0,870968	0,9	0,885246	30
0,675676	0,862069	0,757576	29
0,324324	0,4	0,358209	30
0,484848	0,533333	0,507937	30

0,545455	0,4	0,461538	30
0,685714	0,8	0,738462	30
0,583333	0,466667	0,518519	30
0,62963	0,586207	0,607143	29
0,631579	0,4	0,489796	30
0,473684	0,62069	0,537313	29
0,580645	0,62069	0,6	29
0,6	0,4	0,48	30
0,777778	0,466667	0,583333	30
0,285714	0,4	0,333333	30
0,26087	0,4	0,315789	30
0,272727	0,3	0,285714	30
0,309524	0,433333	0,361111	30
0,5	0,333333	0,4	30
0,463415	0,633333	0,535211	30
0,666667	0,153846	0,25	26
0,2	0,275862	0,231884	29
0,391304	0,3	0,339623	30
0,625	0,344828	0,444444	29
0,357143	0,333333	0,344828	30
0,512821	0,666667	0,57971	30
0,575758	0,633333	0,603175	30
0,627907	0,9	0,739726	30
0,708333	0,566667	0,62963	30
0,941176	0,8	0,864865	20
0,888889	0,8	0,842105	30
0,783784	0,966667	0,865672	30
0,851852	0,793103	0,821429	29
0,888889	0,8	0,842105	30
0,964286	0,9	0,931034	30
0,47619	0,333333	0,392157	30
0,8	0,666667	0,727273	30
0,481481	0,433333	0,45614	30
0,333333	0,433333	0,376812	30
0,625	0,5	0,555556	30
0,571429	0,8	0,666667	30
0,625	0,5	0,555556	30
0,736842	0,466667	0,571429	30
0,546427	0,546427	0,546427	0,546427
0,555947	0,550213	0,541856	5794
0,554722	0,546427	0,53964	5794

C. Language Model

The preprocessing of the language model starts from reading the text by lines and appending it into an array. I also remove the punctuations and put the sentences into an array. Here, I use N-Gram to predict the next words. N-Gram allows us to see the sequences of the text and predict words based on the corpus we feed into the system during training. In this process, I also tokenize the sentences in the array, I declared.

The next step is to generate the padding sequences and obtain variables of predictors and targets. Then, I built the LSTM network which implements Recurrent Neural Network (RNN) to see the connections between nodes by seeing what's before and after. I added an embedding to the network so that the input information can be mapped from a high dimensional to a lower dimensional. Then, I added 100 nodes to an LSTM hidden layer and dropped out the nodes for 20%. Then, I used softmax and the input is the length of the words given.

Here is an example of the context: "Walking down the shore"

```
Walking On The Shore Building The Ones Of The Floor Of The Cubbyhole The Wall Of His Floor And The Restless Glass
```

D. Reflecting on the Paper Reading Workshop

Implementing the high-level paper reading technique which was taught during the class before attending the workshop helped me to get the overall idea of the two papers reviewed during the workshop. Focusing on the details for the second time made me spot some of the drawbacks of the two papers such as the inconsistency of the paragraphs and chapters and lacking explanation of the loss choice for the Cycle-GAN. During the workshop, I managed to learn that some descriptions and motivations are irrelevant and not cited for the Dropout paper. It's also not quite reasonable and clear for the author to implement Cycle-GAN on RBMs either. For the Cycle-GAN paper, the research also lacked characteristics distribution of the training datasets that makes the image translation between horse and zebra not quite well.

Apart from all the drawbacks of the papers, I learned how snowballing can be implemented to understand more concepts of the theory and the techniques used in the work of the two authors. I could go to the citations and read more papers from what the authors cited. Moreover, the two papers are quite clear on their concept and architecture so that it's easy for me to follow along and reproduce the work. I understand how Cycle-GAN can be implemented in many applications such as object transfiguration and dropout can help to reduce overfitting in many cases in our hidden layers. As a future deep learning specialist, the workshop gives me motivation to keep up-to-date with the works of other researchers and how I can easily understand all the progress in the domain. The workshop and the discussion with others made me aware that I can't just accept the author's word and be more critical about the math behind the scene, the datasets they used, and the motivation of their work. It reinvigorates the spirit of innovation inside me from understanding other people's work to get the idea of the current pattern of the deep learning trends.