**50.039 Deep Learning Small Assignment**

**Report on Code**

**Naik Hiong Chiang 1002415**

**Albert Suryanto 1002391**

Change all the paths in the DeepLearningProject.py file (P.S. if the zip files are downloaded and extracted, the paths would all be relative and hence no change is needed.)

* 1. Path to JPEG images and ImageSets at line 465
  2. Path to pretrained\_model\_name\_path at line 502 and 522

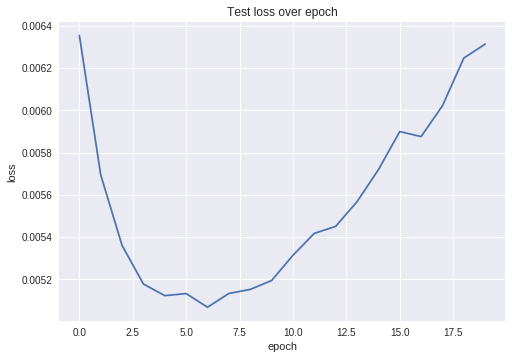
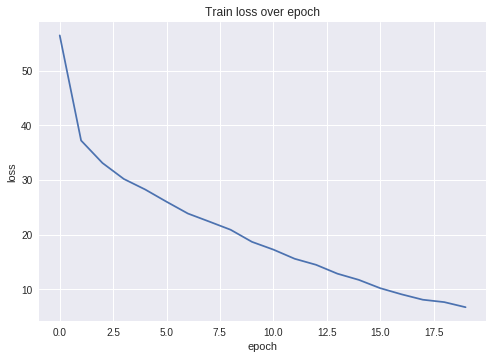
2 different data augmentations are used as reflected in the graphs and results below. The first data augmentation is random rotation of the image by a maximum of 15 degrees, represented by ***transform1***. The second data augmentation is a five crop, represented by ***transform2***. Both transforms utilize a resize and a crop to ensure a fixed dimension into the pretrained model of Resnet18, followed by the training of only the last 2 layers. Reason being the first few layers are trained to detect rudimentary features such as edges etc. The training of the last 2 layers are for high level feature detections and classification of the different dataset that the Resnet18 was trained on(ImageNet). Sigmoid is used for the loss as it is a multi-label dataset and the class probability should be independent from one another, hence softmax is not used.

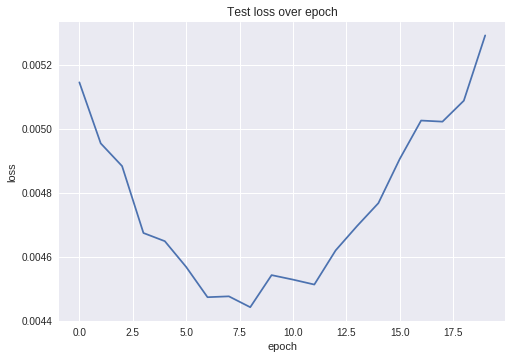
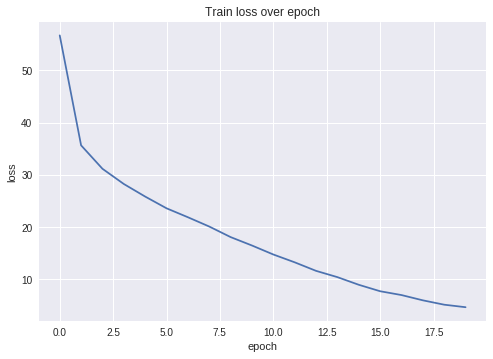
As we are training our model for multi-label classification, the ground label will have one-hot-encoding structure. Thus, the loss will be based on the Binary Cross Entropy Loss. This is so as an image can have multiple labels present in it. Hence cross-entropy-loss over 20 classes is not the right way. Binary cross entropy loss is used as it is able to minimize the 20 binary classifiers.

The optimizer used is Stochastic Gradient Descent with learning rate of 0.01 and a momentum of 0.9. We train our model with these both transforms for 20 epochs and a batch size of 16.

These are the train and test loss for each transform (transform1 and transform2)

**Transform1**



**Transform2**

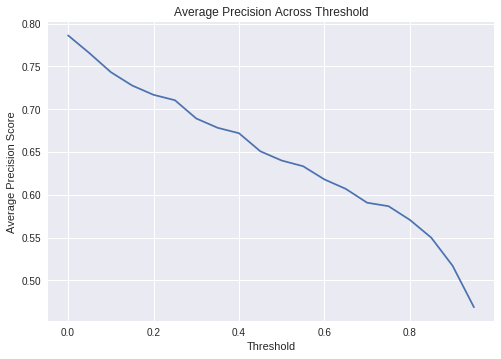
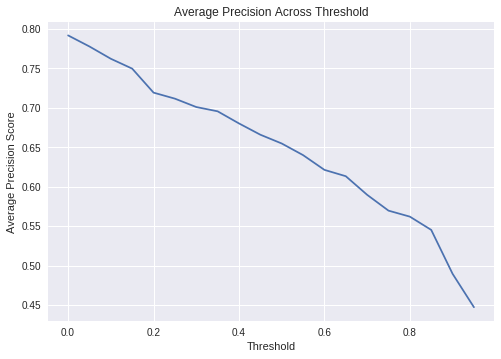
From the train and test loss over 20 epochs graph, we discover that there is an overfitting in both transforms. As we see from the test loss curve for both transforms, the curve has reached minimum point for an epoch (epoch 6 and epoch 8 for transform1 and transform2 respectively) and the loss increases from that epoch onwards. Meanwhile, the training loss for both graphs still continue to decreases. Thus we only save the pretrained model for both transforms when the test loss is at minimum point.

We use our pretrained model to make a prediction on validation dataset. As the nature of the validation dataset label is a binary classifier over 20 classes, we are using precision method, instead of accuracy method, to check our model performance. In multi-label classification, a misclassification is no longer a hard wrong or right. A prediction containing a subset of the actual classes should be considered better than a prediction that contains none of them, i.e., predicting two of the three labels correctly this is better than predicting no labels at all. If accuracy is used, the true positive rate is very low and the true negative rate is high as one image might not contain a lot of classes out of the 20. Hence, accuracy would not be an informative measure as it is an unbalanced problem and precision is favoured.

where, P(k) is the precision at a cutoff of k images, and delta r(k) is the change in recall that happened between cutoff k-1 and cutoff k.

**Average Precision over Different Threshold Graph**

Transform1 Transform2



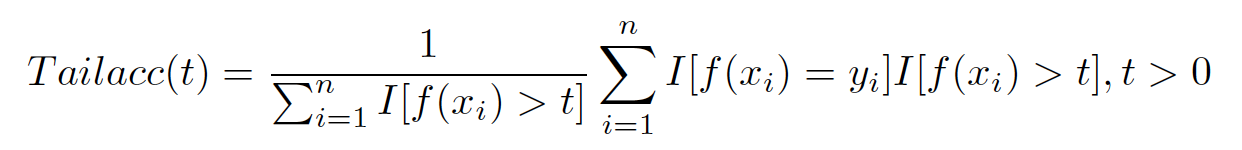
**Average Precision over Different Threshold Table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Threshold** | **Transform1** | **Transform2** |  | **Threshold** | **Transform1** | **Transform2** |
| **0.05** | 0.78623 | 0.79183 |  | **0.55** | 0.64002 | 0.65476 |
| **0.10** | 0.76546 | 0.77777 |  | **0.60** | 0.63339 | 0.64002 |
| **0.15** | 0.74331 | 0.76220 |  | **0.65** | 0.61794 | 0.62132 |
| **0.20** | 0.72765 | 0.74970 |  | **0.70** | 0.60692 | 0.61327 |
| **0.25** | 0.71680 | 0.71926 |  | **0.75** | 0.59073 | 0.58967 |
| **0.30** | 0.71051 | 0.71162 |  | **0.80** | 0.58668 | 0.56961 |
| **0.35** | 0.68915 | 0.70100 |  | **0.85** | 0.57068 | 0.56200 |
| **0.40** | 0.67829 | 0.69557 |  | **0.90** | 0.54993 | 0.54517 |
| **0.45** | 0.67193 | 0.68020 |  | **0.95** | 0.51712 | 0.48951 |
| **0.50** | 0.65081 | 0.66590 |  | **1.00** | 0.46864 | 0.44714 |

As the threshold increases, this will result in less true positive and also less in false negative (TP, and FN decreases). However, the decrease in FN will be slower than TP and this will result in increase of Recall over increase in threshold, and thus the decrease in change of Recall is faster over increase in threshold. As the result, we observe that the average precision decreases as the threshold increases which we see this pattern in our Average Precision over Different Threshold graph above

At threshold 0.5, for transform1 and transform2, the average precision is 0.65081 and 0.66590 respectively and both values are above 0.5. This indicates that our models are doing relatively well

From top-50 images, we also calculate the tail-accuracy for every class over 20 values of thresholds, from 0 to max confidence level of each class



The average tail-accuracy table of the pretrained models for both transforms are very consistent and very high. Most classes have an average tail-accuracy close to 1.0. This indicates that both pretrained models perform very well on the multi-label classification

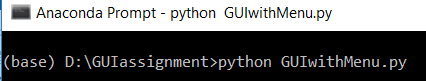
**Average Tail-Accuracy Table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Classes** | **Transform1** | **Transform2** |  | **Classes** | **Transform1** | **Transform2** |
| **Aeroplane** | 1.0 | 1.0 | **Dining Table** | 0.84 | 0.84065 |
| **Bicycle** | 1.0 | 1.0 | **Dog** | 1.0 | 1.0 |
| **Bird** | 1.0 | 1.0 | **Horse** | 1.0 | 1.0 |
| **Boat** | 1.0 | 1.0 | **Motorbike** | 1.0 | 1.0 |
| **Bottle** | 0.98 | 1.0 | **Person** | 1.0 | 1.0 |
| **Bus** | 1.0 | 1.0 | **Potted Plant** | 0.96072 | 0.92 |
| **Car** | 1.0 | 1.0 | **Sheep** | 1.0 | 0.98 |
| **Cat** | 1.0 | 1.0 | **Sofa** | 0.82067 | 0.94 |
| **Chair** | 0.98 | 0.96 | **Train** | 1.0 | 1.0 |
| **Cow** | 0.92 | 0.92 | **TV Monitor** | 1.0 | 1.0 |

Disclaimer:  
There is no enough space to put 40 graphs of tail-accuracy of each class. We upload a of 40 graph images in the folder ‘Tail Accuracy’. Note that the y-axis is supposed to be ‘tail accuracy score’ and the title should be tail accuracy for {class name}.

**Using the GUI**

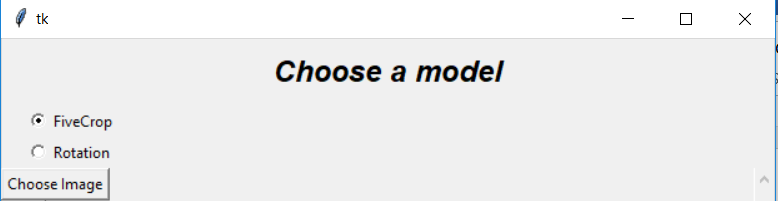
1. Change all the paths in the GUIwithMenu.py file (P.S. if the zip files are downloaded and extracted, the paths would all be relative and hence no change is needed.)
   1. Model\_path on line 285 and 299 – the paths to the best models trained.
   2. Path on line 506, 508, 591, 593, 656 and 658 – the paths to the dictionary file (json) output by the deep learning code.
   3. Image path on 527, 608, 673 - the paths to the JPEG image files.
2. Start the GUI python file on Anaconda Prompt



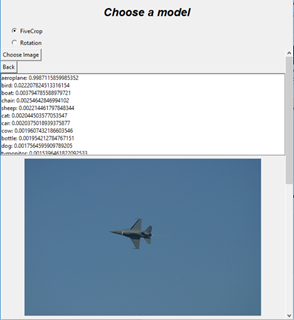
1. The main page will pop up. Choose whether you wish to submit a photo to obtain predictions or to see the precomputed predictions on the validation set.



1. In submit:
   1. Choose a model and an image that you wish to submit.

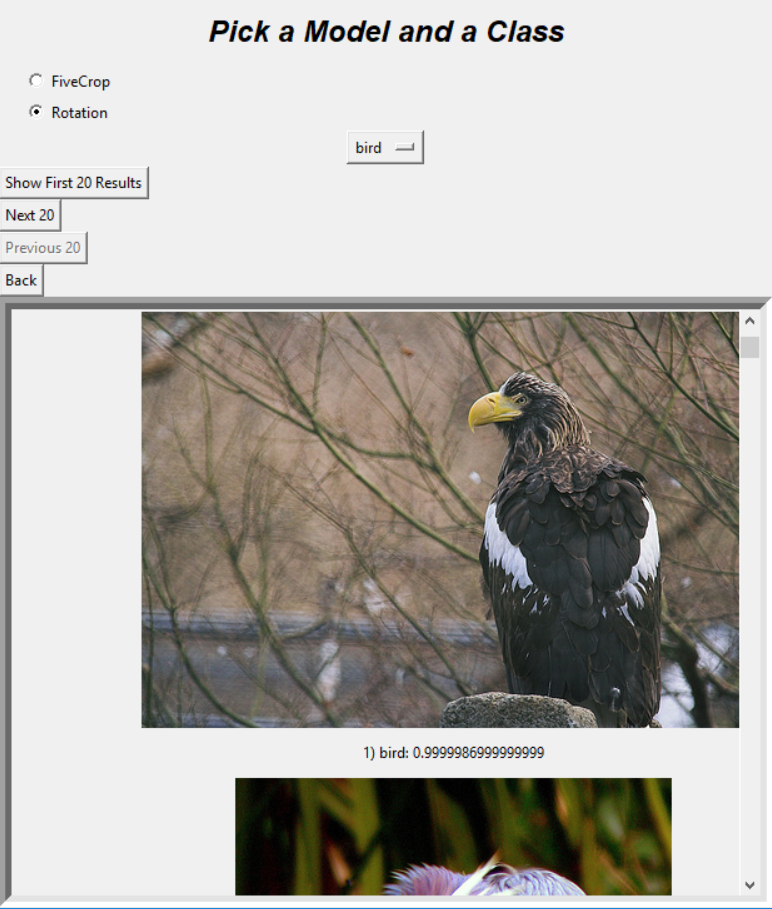


* 1. Predictions sorted on the predictions of each class. The predicted scores are scrollable.

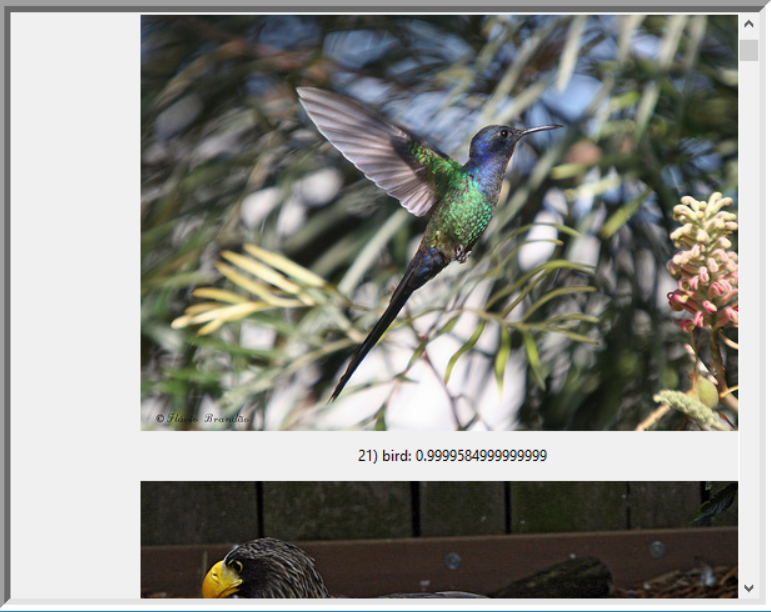


* 1. Repeat for different images. Click back to go back to the main page.

1. In View Precomputed Scores:
   1. Choose a model, a class and click ‘Show First 20 Results’.



* 1. A scrollable results list of 20 images will appear, the results list includes the images and the prediction of that class for each image, sorted in order of highest precision to lowest precision.
  2. Clicking on the ‘Next 20’ will display the 20 results for the proceeding 20 images.



* 1. Clicking on the ‘Previous 20’ will display the preceding 20 images.
  2. The ‘Next 20’ and ‘Previous 20’ will be disabled depending on conditions such as the end or start of the dataset and also whether the ‘Show First 20 Results’ have been clicked.

Disclaimer: In Submit, the image is processed based on the best model chosen to come up with predictions. In show precomputed scores, there is no processing of data to ensure that the GUI will not lag. Instead, it just reads data from the saved files from the Deep Learning code (not reflected in the code as we post processed the data elsewhere to ensure readability)