Deep Learning Mini Project

Wei Letong 1002965 | Wu Tianyu 1002961

Overview

An object detection model with Pascal Voc 2012 dataset (20 target classes) is built by transfer learning. Implementation details for the model are listed below.

Dataset

Three values are returned for customized __getitem__ method: PIL image, label, and filename. Label is a 1-d tensor of shape 20. In this way, each image appears exactly once in customized dataset. Filename is returned to keep track of each image, which will be used in the top-5 and bottom-5 image task. XML file is used for loading class labels. A simple recursion is implemented to deal with XML files.

Transfer Learning

Renset-18 with pre-trained parameters loaded is used as model before training. The fully connected layer is overwritten to be linked to 20 nodes to suit the task.

Loss Function

BCEWithLogitsLoss is used to average over the binary loss for 20 classes for a single batch input. The task is a multi-label classification problem, therefore essentially binary loss should be used for all 20 classes. BCEWithLogitsLoss suits perfectly. BCEWithLogitsLoss combines a Sigmoid layer and BCELoss in one single function and has higher numerical stability by taking advantage of log.

Average Precision Measure

Method in sk-learn for precision measure is sklearn.metrics.average_precision_score. This method returns the approximated area under the precision-recall curve (integral approximation).

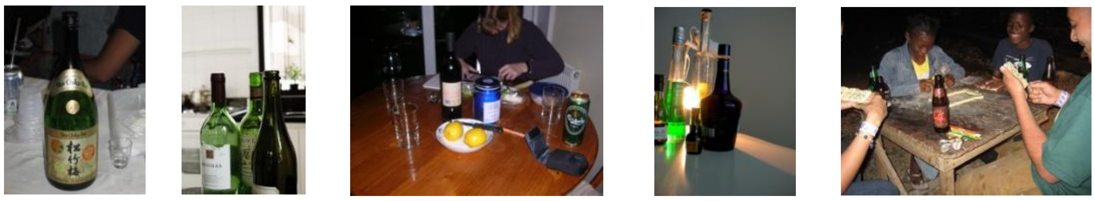
Accuracy vs Precision

Accuracy takes into account both true positive and true negative. In this task, negative data has a portion way higher than positive data. Therefore, accuracy will be influenced mostly by negative data. What matters more in this task is to successfully predict the positive label, not the negative one since the model is for object detection. Moreover, prediction on negative data tends to be considerably high all the time, therefore accuracy vary very little and cannot reflect the effectiveness of model.

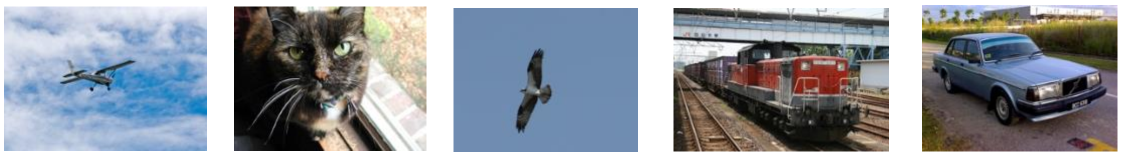
Top-5 and Bottom-5 for 5 Random Classes

All the graphs below are obtained after the model is trained for 5 epochs. As can be seen, top 5 highest score images for each class are already very accurate.

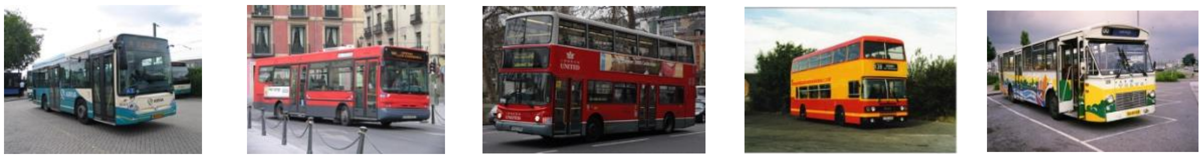
Bottle Class
Top-5



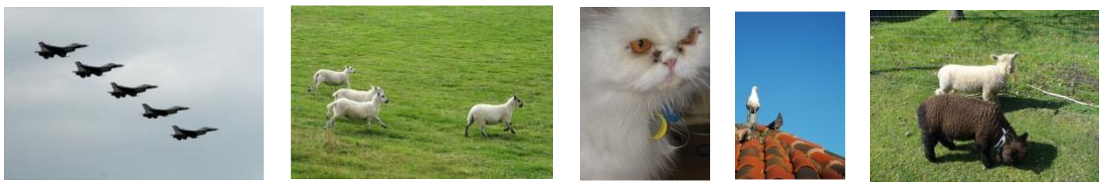
Bottom-5



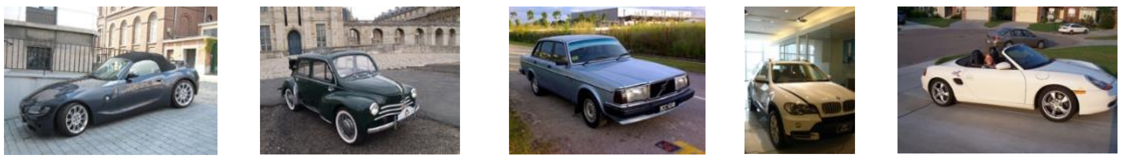
Bus Class
Top-5



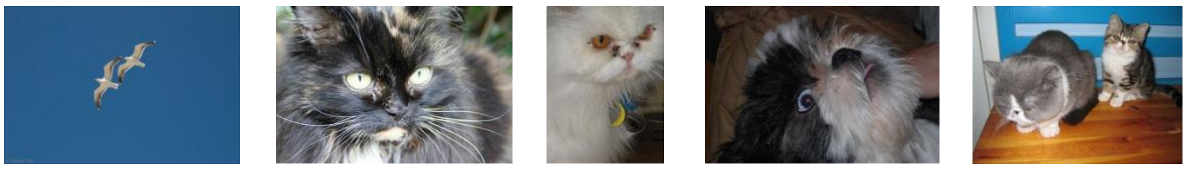
Bottom-5



Car Class
Top-5



Bottom-5



Cat Class
Top-5

Bottom-5

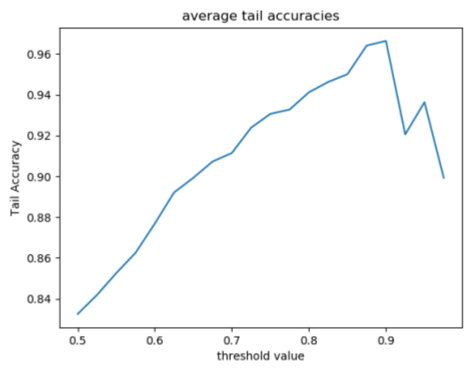
Chair Class
Top-5

Bottom -5

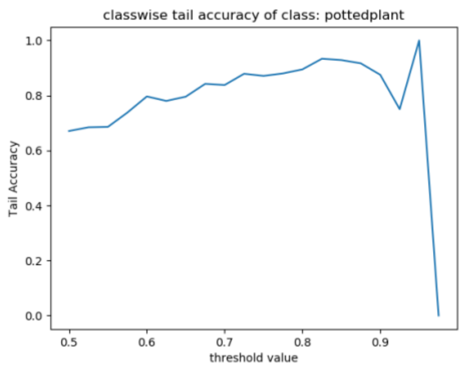
Tail Accuracy

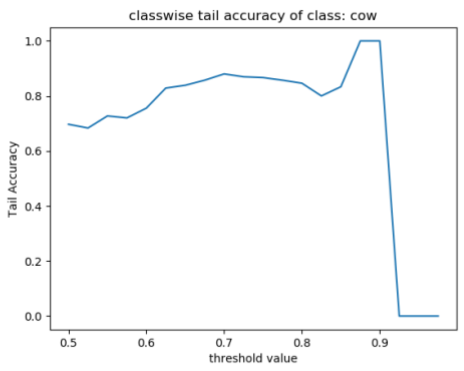
$$Tailacc(t) = \frac{1}{\sum_{i=1}^{n} I[f(x_i) > t]} \sum_{i=1}^{n} I[f(x_i) = y_i] I[f(x_i) > t], t > 0$$

Tail accuracy is the generalization of precision measurement. Tail accuracy with t set to the threshold used in binary classification model is equivalent to normal precision measurement. Below is the average tail accuracy with respect to t. All graphs are obtained with 5 epochs.

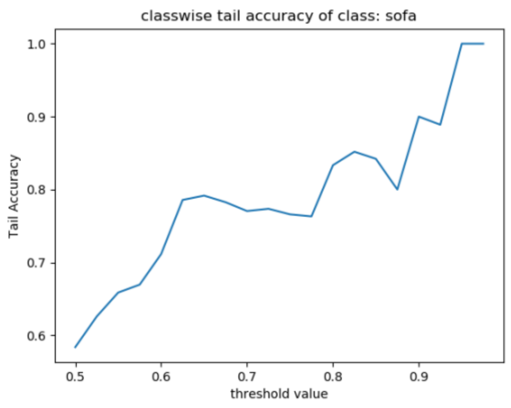


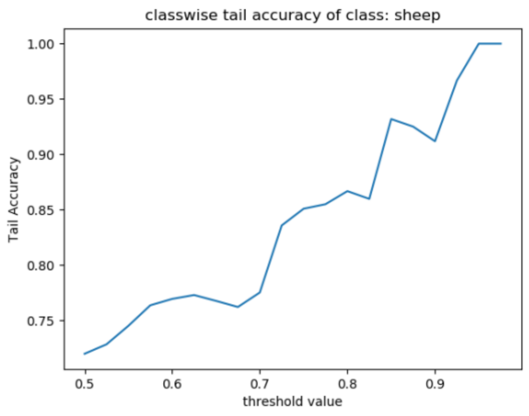
Tail accuracy for some of the classes are attached below. Classes potted plant and cow accounts for the drop in the average tail accuracy. When value t becomes larger than the highest-scoring image for each class, accuracy will drop to 0 by default.



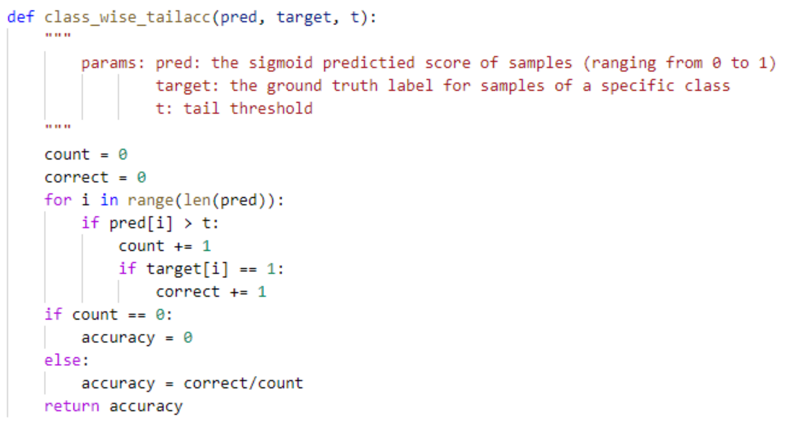


Tail accuracy for class sofa and class sheep is the typical tail accuracy where tail accuracy increases generally with respect to the increase of value t.





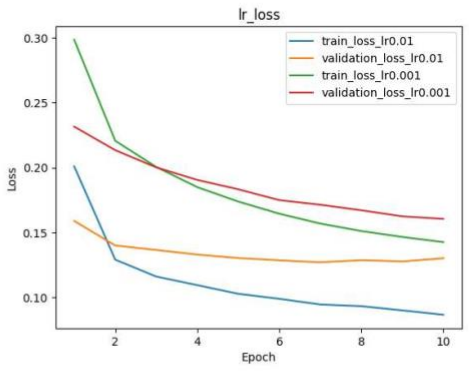
The simple code snippet for calculating tail accuracy is attached below.

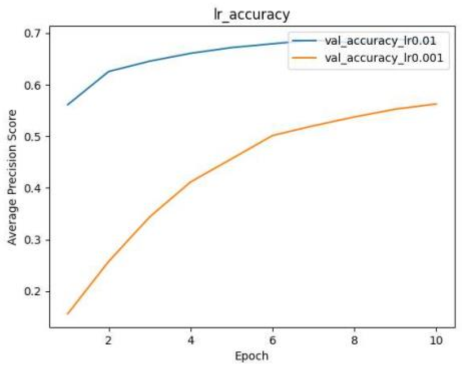


```python
def class_wise_tailacc(pred, target, t):
    """
        params: pred: the sigmoid predictied score of samples (ranging from 0 to 1)
                target: the ground truth label for samples of a specific class
                t: tail threshold
    """
    count = 0
    correct = 0
    for i in range(len(pred)):
        if pred[i] > t:
            count += 1
            if target[i] == 1:
                correct += 1
    if count == 0:
        accuracy = 0
    else:
        accuracy = correct/count
    return accuracy
```

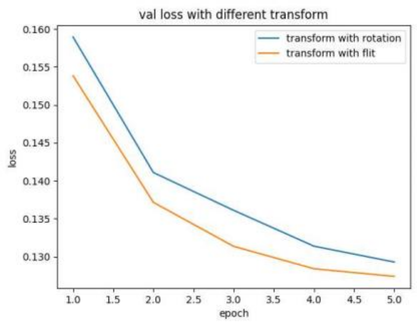
Hyperparameter tuning

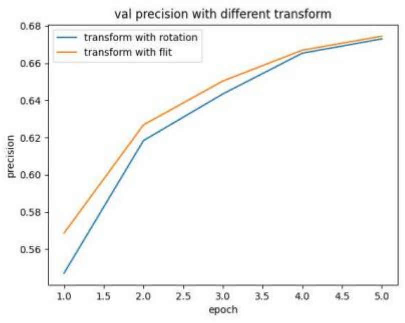
Learning rate is one hyperparameter. Comparison between training losses, validation losses, validation average precisions are made between learning rate 0.01 and 0.001. The results suggest that learning rate 0.01 trains the model faster than 0.001.





Data Augmentation for Validation





Two transforms (rotation and flip) are applied to the validation set separately. The performance of flipping transformation is better than that of rotation transformation.

Reproduction Instruction

All code except for hyperparameter tuning and data augmentation is included in final.py, code for learning rate tuning is in lr.py, code for data augmentation is in transform.py, please make sure that:

· The final.py is in the same directory as the PASCAL data folder (VOCdevkit) and please do not change the directory inside VOCdevkit folder

· Create a TOP folder under the same directory as final.py and make sure that it is empty before running (for saving the top scored images)

· The parameters of the best model will be saved under the same directory as final.py and will be used later, please do not change its directory

· All the plots will be saved under the same directory as final.py

· Number of epochs can be changed by setting num_epochs in the main function. Current setting is 5 for final.py for convenience of reproduction.

· Please make sure above requirements are satisfied by both lr.py and transform.py.