50.039 SMALL PROJECT

Li Xueqing | 1002182
Zeng Yueran | 1002207

Overview

In this project, we build an object detection model with Pascal Voc 2012 dataset by transfer learning. With a GUI as the frontend, the user is able to select or upload an image to have it recognized. And it also enables the user to browse through the top-ranked images predicted by the model.

Code Implementation

Dataloader

By parsing the XML file, the annotation of each image is read and we converted it to binary tensor with length 20 corresponding to each class. The images are transformed by two sets of augmentations: *FiveCrop* and *RandomHorizontalFlip*

Deep Neural Network Model

The deep learning model has simultaneous output over 20 classes so transfer learning is preferred. We choose resnet18 with pre-trained parameters and modify the fully-connected layer to 20 out_features.

Criterion for Loss

As one image may have multiple labels, using cross entropy loss over 20 classes would not meet our goal. As a result, we choose *BCEWithLogitsLoss*, which combines a sigmoid layer and the BCELoss in one single class. It is more numerically stable than using plain Sigmoid and BCELoss. And it measures the binary cross-entropy between the output and target, which are numbers from 0 to 1.

Training

The following are the hyperparameters used in training and validation:

- Batch size: 64
- Number of epochs: 50
- Learning rate: 0.02
- Device using: AWS p2.xlarge

Training procedure:

1. Dataloader load training and validating dataset
2. Train and evaluate on each epoch, record loss, and accuracy
3. Save model when lowest validation loss is found
4. Finish training

Average precision measure and Data Augmentation

Accuracy vs. Precision

Accuracy = (TP + TN) / (TP + TN + FP + FN)
Precision = TP / (TP + FP)

Accuracy takes all data into account. However, accuracy paradox can occur where there is a large class imbalance (in our case, the class *person* around 2000 images while the class *sheep* and *cow* only has around 100 images), the model can simply predict the value of the majority class for all predictions and achieve high accuracy. In this case, such a model is not useful for our prediction.

On the other hand, precision shows how precise the model is out of the data that are predicted positive, how many of them are actually positive.

To get the average precision measure, we utilize the API *sklearn.metrics.**average_precision_score()***, where it calculates the average precision from prediction score. The value is 0 to 1, the higher the more accurate.

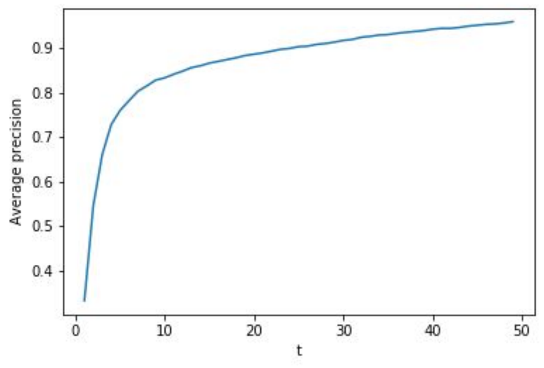


Figure 1. AP for *FiveCrop* as augmentation

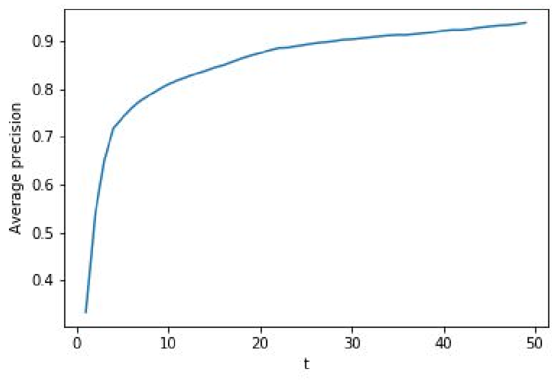


Figure 2. AP for *RandomHorizontalFlip* as augmentation

Data Augmentation
The AP on the validation set for *FiveCrop* is 0.959 and for *RandomHorizontalFlip* (flip with a probability of 0.5) is 0.932 after training for 50 epochs. The performance of *FiveCrop* is better as it largely increases the training data compared with *HorizontalFlip*, while preventing overfitting at the same time as the model never sees two exact inputs.

Evaluation

We compute the accuracy of the predictions in the upper tail. The accuracy is computed as the true predictions over all predictions filtered by a threshold. With the threshold from 0 to 1.0 with an interval of 0.05, we loop through the validating over the 20 t's. Tail accuracy reaches the maximum of 0.8351 at t=0.45.

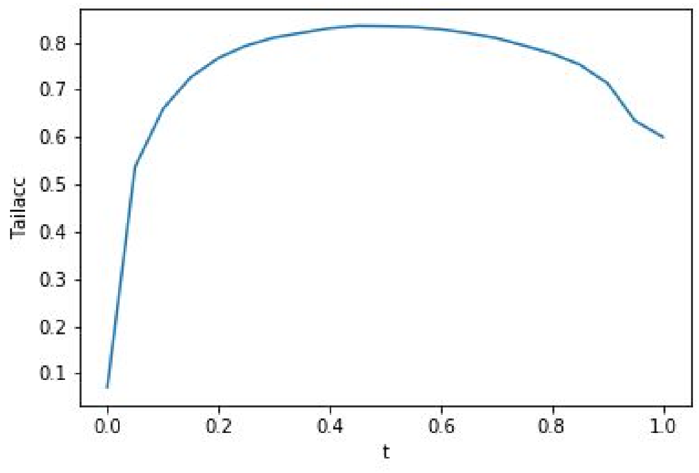


Figure 3. Tailacc with respect to t

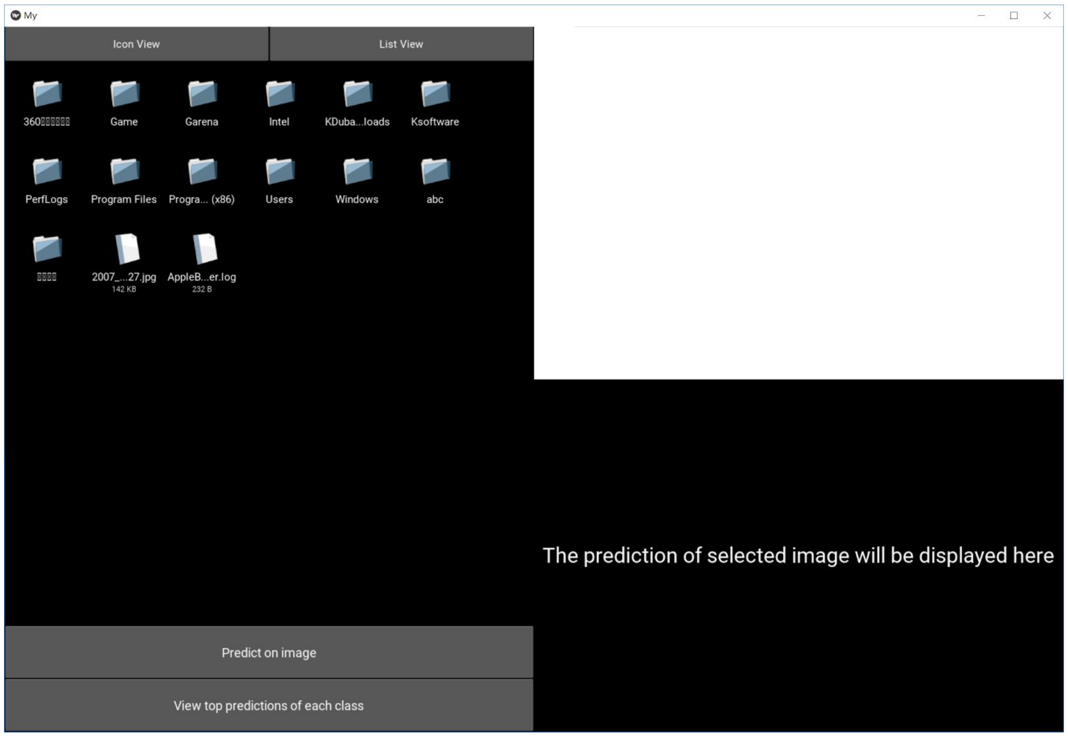
Demonstrator

To open the demonstrator, run the python code **using cpu**: *demonstrator.py*

Before running the demonstrator, please make sure the following files are under your current directory and do not alter the name of files:

- *voc_code.py* : It includes the function for single image prediction
- *demonstrator.kv* : The display settings for the GUI
- *voc_model.pkl* : The model used for image prediction
- *rank_wempty.txt* : The pre-computed predictions for the validation set
- *data/VOCdevkit/VOC2012/JPEGImages/*.jpg* : Images of validation set (make sure the path!)

Image prediction

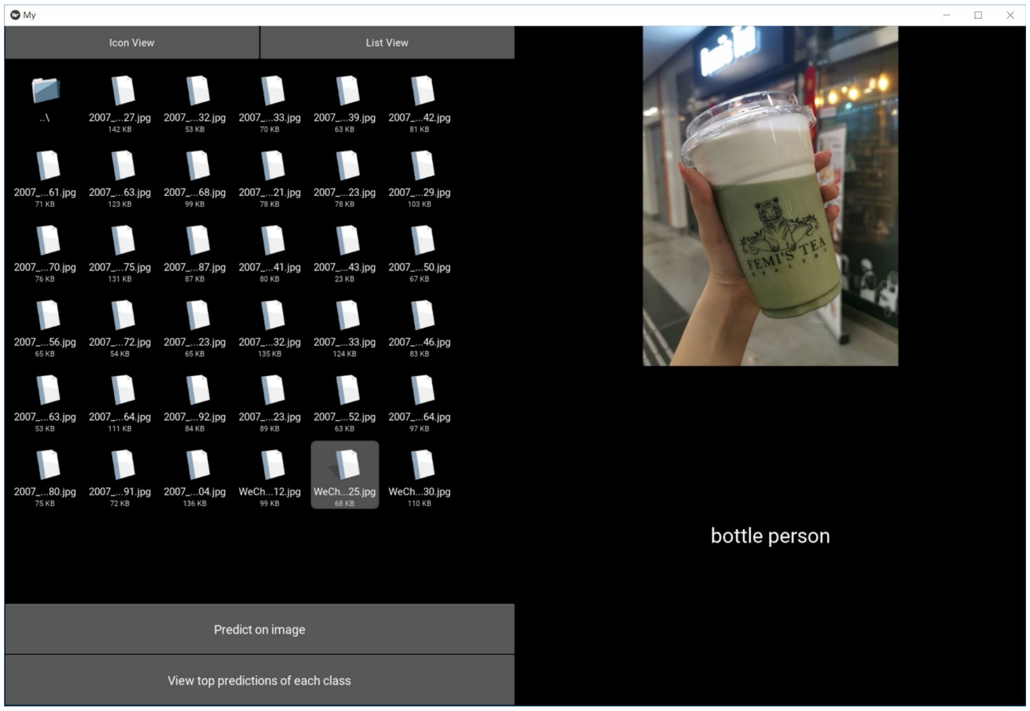


The prediction of selected image will be displayed here

Figure 4. Screen display before prediction

In this screen, the user could make a prediction on a custom image following the procedure:

1. Select the image from the FileChooser on the left side. There are two types of views available for the FileChooser: the icon view and the list view.
2. Then click the button *"Predict on image"* to predict.
3. The image selected and the prediction generated will be displayed on the right side.



bottle person

Figure 5. Screen display after prediction on a custom image

View validation set

Press the button *"View top predictions of each class"* to enter the next screen. In this screen, the user could view images ranked by the prediction performance for each class.

For each class, there is a list of images that are predicted to have such class label. Each image has a probability for the prediction of this class label. These images are then sorted in descending order according to the probabilities, whose names are saved in the text file. A higher probability represents a higher rank, interpreting a better performance for this image's prediction compared with others.

Press the button *"Display"* to view the highest rank of images for each class (images with highest prediction probabilities).

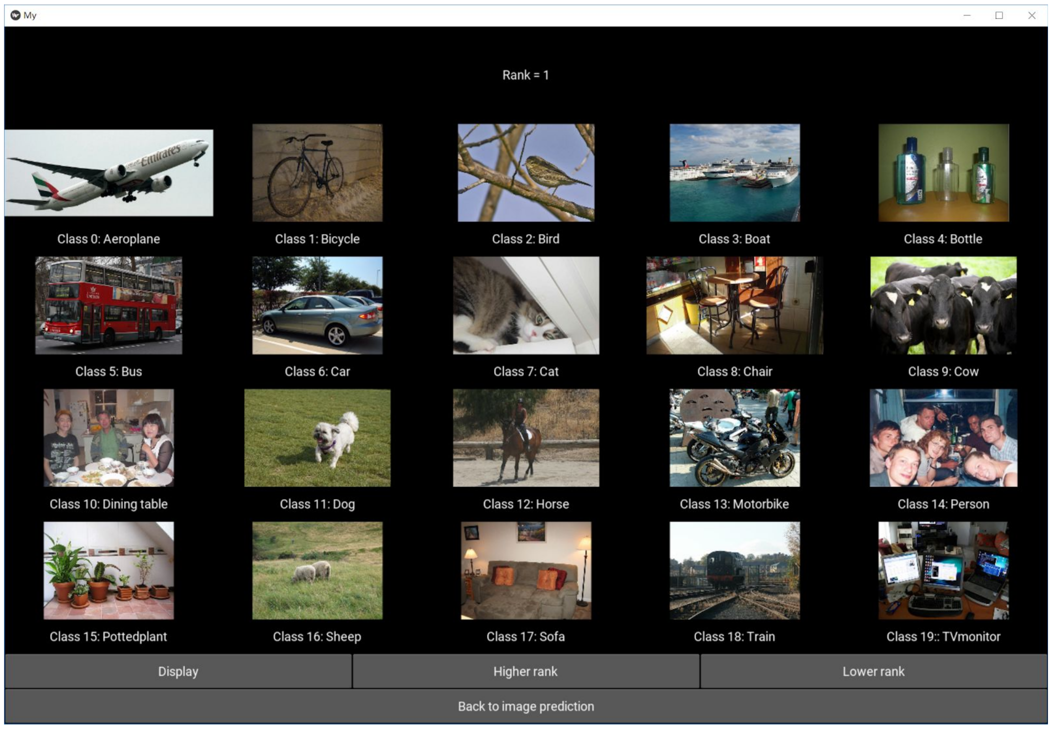


Figure 6. Screen display for the highest rank

The user could be able to view images with higher rank or lower rank by pressing the button *“Higher rank”* or *“Lower rank”*.

If the user wants to return to the first screen for image prediction, kindly press the button *“Back to image prediction”*.