Deep Learning Project 1 - PascalVOC

Goh Yu Jin (1002157)
Singapore University of Design and Technology
yujin_goh@mymail.sutd.edu.sg

Clemence Goh (1002075)
Singapore University of Design and Technology
clemence_goh@mymail.sutd.edu.sg

ABSTRACT

Multi-label classification with images from Pascal VOC 2012 as a dataset. The Pascal VOC 2012 dataset contains images which belong to a mixture of 20 different classes. Each image can have multiple class labels and therefore it is more challenging than single label classification problems. The goal of the project would be to classify images new unseen images which belong to a mixture of the 20 classes using a deep learning model. Through training a deep learning model, we have been able to predict labels with a high average precision of 86% across all classes.

KEYWORDS

image, pascalvoc2012, multilabel, prediction, classification, perceptron

1 INTRODUCTION

We have been tasked to develop a deep learning model capable of classifying images from the Pascal VOC 2012 dataset, which contains images of multiple class labels. The dataset contains images that are labelled with a mixture of 20 different classes. Through the use of transfer learning, we are to develop a model that could accurately determine the classes relevant to the images displayed.

We are therefore, required to produce the following items:

(1) a dataloader for pascal VOC train and val datasets
(2) a deep learning model with simulataneous inputs for all 20 classes
(3) loss for minimizing
(4) average precision measure on validation set
(5) a demonstrator which allows
   - predict on a single image
   - to browse in a GUI the top-ranked predictions for each class on the validation dataset

2 LIBRARIES

Developing the model has required us to use the following libraries as follows:

(1) pytorch
(2) numpy
(3) PIL.Image
(4) torch.utils.data
(5) torchvision
(6) vocparseclslabels
(7) sklearn

3 DATALOADER AND DATASET CLASS

For the purpose of our project, a custom Pytorch Dataset class has been created for it. This dataset class will take in text files from the Pascal VOC dataset named with the respective class labels, containing the names of the image files and a indicator whether it belongs to that specific class. For example, a -1 in the planes.txt file would indicate that the image file does not belong to the plane class. At initialization, the dataset will extract out all of the file names listed in the text files and encode an array of length 20 for it's label. the 20 binary encodings will denote the 20 possible classes the image belongs to, where 1 in index 5 represents that it belongs to class 5 and 0 if it does not. When an item is extracted from the dataset class, it will iterate down the list of images and return the respective PIL image file of the image and its labels.

The dataloader class will load in the data from either the training or validation set. When called, it will shuffle the data to ensure a more representative set of datapoints is used each time.

4 MODEL

4.1 Model Selection

From our experiments we have decided to use a Resnet34 model which takes in an imput size of 224 by 224 with a RGB channel. We have selected the Resnet34 model as it had performed better than the Resnet18 model due to the increased number of layers to approximate a more complicated input. This was determined by training and validating the Resnet18 model and the Resnet34 model on the same parameters, after which the Resnet34 had obtained a better validation score.

4.2 Data Augmentation

Data Augmentation was used to make the model more robust to similar inputs. Thus for the training set we have included three different augmentations, specifically being random crop, horizontal flips and color jitter. Random crop and horizontal flips were adjusted so that the classifier would be able to learn from target classes at different positions on the inputs. Additionally, colour jitter was included to randomly change the brightness, contrast, hue and saturation of the image.

The parameters used in training for brightness, contrast, hue and saturation were left at 0.1 and images had the shorter dimension resized to 224 before being cropped to a square image of 224.

In the validation set, we simply resized the shorter dimension to 224 and cropped it into a square of 224.

Both images were transformed into tensors before applying a normalization with parameters according to the Pascal VOC 2012 dataset as listed in Table 1

4.3 Optimizer and learning rate scheduler

The stochastic gradient descent algorithm was used to optimize the models parameters by updating the weights in the direction of the steepest gradient descent. A initial learning rate of 0.1 was used as well as a momentum of 0.9.

Channel	Mean	Standard deviation
1	0.485	0.229
2	0.456	0.224
3	0.406	0.225

Table 1: Normalization parameters

After every 10 steps, the learning rate of the model will be multiplied by a constant of 0.1. Thus, decreasing the learning rate to assist in in approaching a minima.

4.4 Loss

The BCEWithLogitsLoss was the loss we had used for our model to determine how accurate our model was in predicting the output. The BCEWithLogitsLoss first takes in the 20 outputs of the model and passes it into a logistic sigmoid function, thereby mapping its values to the range of 0 to 1. It then applies a cross entropy loss function on the individual on the outputs of the sigmoid together with the ground truth labels. Finally, the mean of the 20 losses are obtained to be minimized by the optimizer.

Figure 1: train/val loss vs epoch

4.5 Precision

Precision graph report and evaluation Precision was a more accurate evaluation of the model's accuracy as most of the images had 1-4 classes, which meant that most of the labels were left as 0. An accuracy of above 90% could have been obtained easily by labelling all the outputs as 0. Hence, precision is a better metric. The model with the highest precision is saved during training.

4.6 Tail accuracy

As expected, the tail accuracy of the model improved with an increasing threshold values. As we raise the threshold value of t, higher confidence scores were required to be classified as correct. Thus, the precision of the model would increase as only labels with higher confidence scores than the threshold would be predicted as the specified class. Therefore, Labels with a higher confidence level generated by the model were generally classified correctly as seen in figure 1.

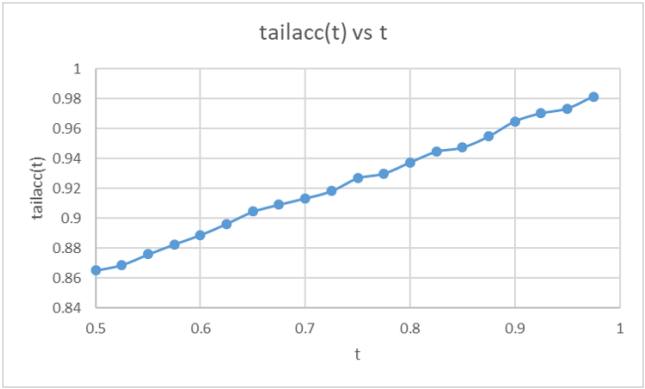

Figure 2: tail accuracy vs threshold, t

5 DEMONSTRATOR

5.1 Pre-processing the data

After training the model, the state dict of the model is saved and loaded into the classifier used in the demonstrator. To prepare all the images, the images present within the validation set which have ground truth labels of 1 for that class are run through the classifier, and their results stored within a json file under the name of "data.json" for later use with the GUI.

5.2 Single Image prediction

The GUI accepts an uploaded image from the user and runs it through the classifier loaded with the state dict from our earlier model. The user is then directed to a page which shows the class that the image most likely belongs to from the 20 classes which our model has been trained on.

5.3 Precomputed predictions

In this part of the GUI, 20 classes are shown with images ranked in descending order in terms of their scores calculated from the classifier. By clicking on any of the classes, the complete list of images pre-computed from the validation set within that class will be shown to the user.