Sai Harsha Nagabothu

Dinakar Burgala Nithin

Dr. Feng Luo

Deep Learning- CPSC 8100

**Deep learning model to detect images with cyberbully actions**

**ABSTRACT**

In Phase one, We developed a model which focuses on classifying a person’s action or behavior from a image. In the second phase we improved the model accuracy from the phase one and our approach for this phase typically started with the implementing the bounding box to detect the object and further extended to detect multiple objects in a single image. Finally, We present our views and model approach in the below document.

**Introduction**

In general, if you want to classify an image into a certain category, you use image classification. On the other hand, if you aim to identify the location of objects in an image, and, for example, count the number of instances of an object, you can use object detection. There is, however, some overlap between these two scenarios. If you want to classify an image into a certain category, it could happen that the object or the characteristics that are required to perform categorization are too small with respect to the full image. In that case, you would achieve better performance with object detection instead of image classification.

**Implementation**

Our implementation for phase two consists mainly of three steps

Collecting and labelling the data to train the model.

* Training the Model on Training data
* Recognizing the object on Testing data

As the data was already collected and classified, In order to detect the object we started by manually labeling the objects in images for the given dataset through implementing the bounding box technique using the LabelImg application[7] to get the corresponding coordinates and the labels for the images.



Figure 1. Labeling the images from dataset using labelimg

After labelling the images, we get the coordinate position of the person depending upon the person action in the image in a XML file.

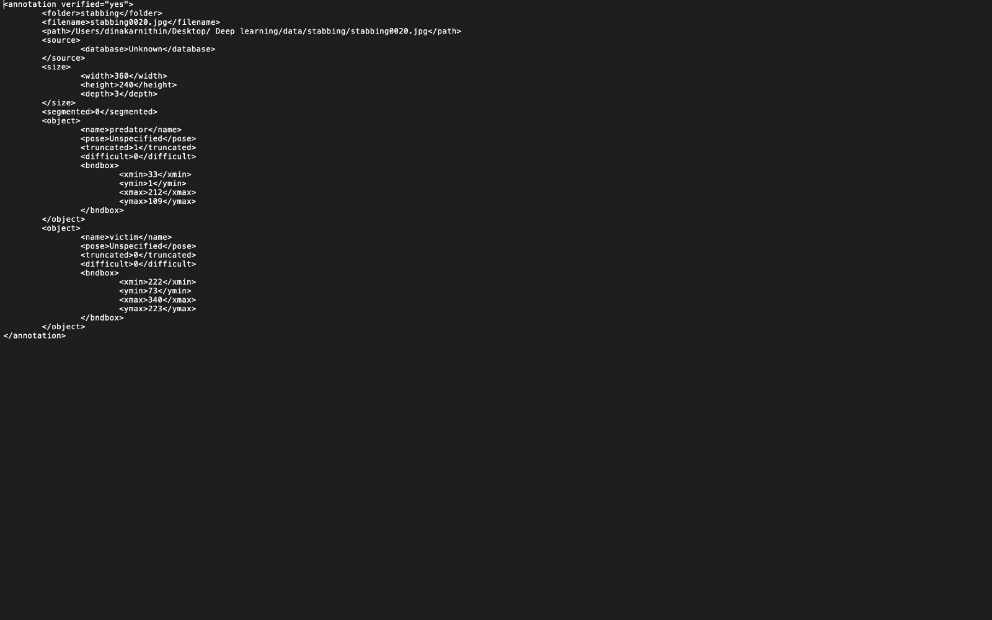


Figure 2. Co-ordinates and labels for the image in an XML file

We generate the co-ordinates for all the images and store them in the respective labels folder, which helps the model to detect the object.

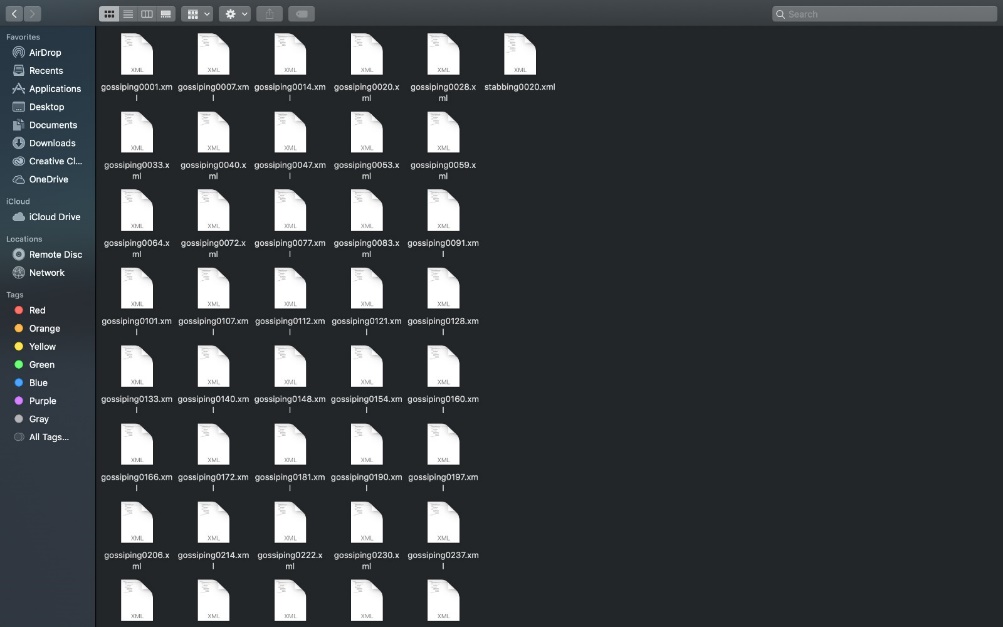


Figure 3. XML files for the respective images

So, The available data is trained on the model and to get the performances such as average error rate, model average and prediction accuracy are measured. And it is tested on the other images whether the trained action is predicting on the test images.

**Training**

From the Yolo coordinates obtained the Yolo model is trained from the following steps.

We need to create configuration files. The first configuration file contains

Classes =1

Train = train.txt

Valid = test.txt

Names = obj.names

Backup = backup/



Figure 4. Sample images used to train the model

The train.txt and test.txt are the files generated from running a script which generates train and test text files which are picked randomly from the Yolo format text files. Obj.names is the second configuration file contain the label for detection. The backup will contain the weights generated by training on the images. After preparing the configuration files, it’s time to train YOLO using the labelled images. The training part is done by running the command.

*./darknet detector train cfg/obj.data cfg/yolo-obj.cfg darknet19\_448.conv.23.*

The training output is obtained in the following format *2: 2.950644, 15.939886 avg, 0.001000 rate, 2.813000 seconds*, 2.950644 indicates the total loss, 15.939886 is the average loss error, which should be as low as possible. As a rule of thumb, once this reaches below 0.060730 average, you can stop training, 0.001000 rate represents the total time spent to process this batch.

After training for Eight hours we stopped. The training is stopped when average reaches 0.06000.

**Results**

From training we obtain .weight files, which will be results obtained from training to test on the inputs. We choose the recent weight which is generated to get the best accurate result on a new input. To run it on the new input the command

./*darknet detector test cfg/obj.data cfg/yolo-obj.cfg yolo-obj1000.weights data/testimage.jpg*

is used. After running the following command the result will be obtained.

**REFERENCES**

[1] <https://arxiv.org/pdf/1711.01467.pdf>

[2] [https://arxiv.org/pdf/1603.05027.pdf](https://arxiv.org/pdf/1711.01467.pdf)

[3] <https://arxiv.org/pdf/1512.03385.pdf>

[4] <https://github.com/facebook/fb.resnet.torch/tree/master/pretrained>

[5] <https://github.com/rohitgirdhar/AttentionalPoolingAction>

[6] <https://towardsdatascience.com/review-resnet-winner-of-ilsvrc-2015-image-classification-localization-detection-e39402bfa5d8>

[7] <https://github.com/tzutalin/labelImg>