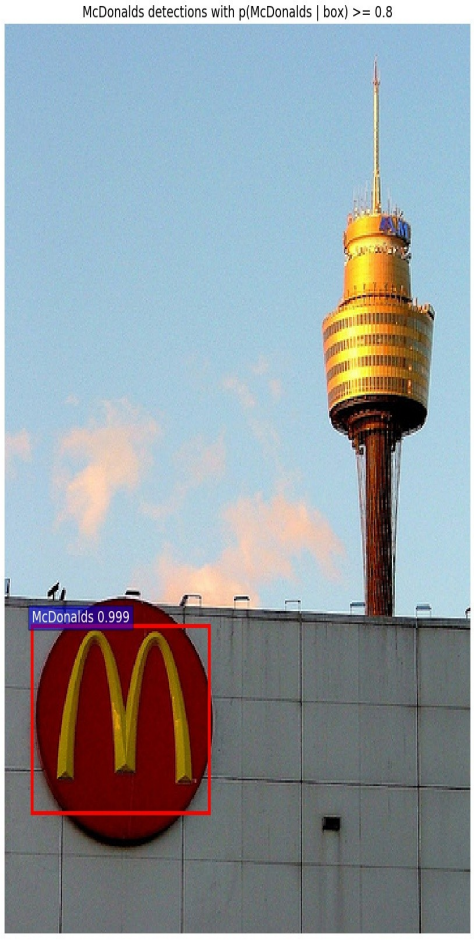
LOGO DETECTION AND CLASSIFICATION

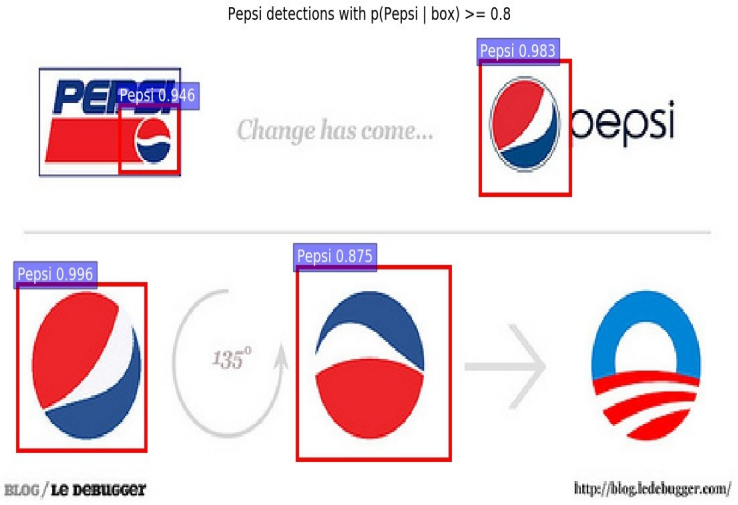
INTRODUCTION

The aim of the project is to detect and classify the logo in a image using Deep Learning. Deep learning model is developed which first detects logo in a given image by drawing bouding boxes around the given image and than classify that logo.

On evaluating the model on test dataset it obtains the logo detection accuracy of 77.03% and logo classification accuracy 92.3%.

Caffe Faster RCNN is used for logo detection and classification.





Due to low computational resources we trained out model on 10000 iterations only. Used AWS g2.2x GPU instance for training. The output of all the test images in the form .jpg files is in **output_images** folder.

Faster R-CNN is an object detection framework based on deep convolutional networks, which includes a Region Proposal Network (RPN) and an Object Detection Network.

SCRIPTS AND OUTPUT

1. The folder **output_images** contains the final output images of all the test images with logo bounding boxes detected and their category classification.

2. **convert_flickr.py :-** Creates the training dataset. It creates Annotations in the form of .xml files and ImageSets containing names of training images.

3. demo.py and **get_accuracy.py :- Demo.py** is used for evaluating the performance of images. It saves the images with logo object and logo class detected in .jpg format. It also creates an additional file out_labels.txt. On running **get_accuracy.py** we get the accuracy of logo classification.

DATASET

The dataset contains 809 annotated images for training and 270 annotated images for testing.The training *set* contains 810 annotated images, corresponding to 27 logo classes/brands (30 images for each class). All images are annotated with bounding boxes of the logo instances in the image. Multiple logo instances per class image is allowed.

The test *set* consists of 270 images. There are five images for each of the 27 annotated classes, summing up to 135 images that contain logos. Beside this the query set contains 135 Flickr images that do not depict any logo class. So removed those 135 images. Now the test set is left with 135 images only.

APPROACH

Finetuned the dataset on Pretrained Faster RCNN ZF network on end2end algorithm model (ZF_faster_rcnn_final.caffemodel).

There are total of 28 classes (27 classes + 1 background class) and 112 output for bounding box regressor. Renamed the last fully connected layer such that the weights of the layer will be initialized randomly instead of copying from pre-trained model (actually copying from pretrained model will cause error). Finetuned pre-trained Faster RCNN model and snapshot at iteration 0. Renamed the layer back to their original names and then train the model for 10000 iterations. Due to low computing resource the model is trained on 10000 iterations only.

Implementation

1. Setup Faster RCNN from the github repositery (https://github.com/rbgirshick/py-faster-rcnn).

2. Convert the dataset. Create three separate folders named Annotations, ImageSets and JPEGImages. The script(**convert_flickr.py**) converts the dataset into .xml annotation files (Annotation folder) and train.txt file (ImageSets folder). Copy all the images in the JPEGImages folder.
We already provided the converted dataset suitable for training in the folder **training_dataset.**

3. Update the **factory.py** file in **py-faster-rcnn/lib/datasets/** or replace it with our **factory.py**. The purpose of **factory.py** file is to get all sets of whole dataset.

4. Add the dataset python file . Add the new **logo.py** file in py-faster-rcnn/lib/datasets/ folder. The purpose of **logo.py** is to read a part of whole dataset , such as train set. Add **logo_eval.py** file inside the py-faster-rcnn/lib/datasets/ folder.

5. Trained the model from fine tuning a pre-trained Faster R-CNN model because a pre-trained model contains a lot of good low level features. Used the ZF-net network.

6. Update the number of output in final layer. We have 28 classes (27 class + 1 background class) and 112 (28*4) output for bounding box regressor. Rename the last fully connected layer.

TRAINING

1. Rename the last fuly connected layer.

2. 1st fine tune. (`./tools/train_net.py --gpu 0 --weights data/faster_rcnn_models/ZF_faster_rcnn_final.caffemodel --imdb logo_train --cfg experiments/cfgs/config.yml --solver models/logo/solver.prototxt --iter 0 )`

3. Rename the layers back to their original names. Set the parameters in solver.prototxt. Due to low computation resources we trained our model on lowest minimum parameters.

4. 2nd fine tune. Set the paths to 1st fine tuned model to get it's weight. (`./tools/train_net.py --gpu 0 --weights output/logo/train/zf_faster_rcnn_iter_0.caffemodel --imdb logo_train --cfg experiments/cfgs/config.yml --solver models/logo/solver.prototxt --iter 10000 )`

TESTING

1. Copy and replace the **demo.py** in **py-faster-rcnn /tools/** with our **demo.py**.

2. Set the path of test_data variable to **flickr_logos_27_dataset_query_set_annotation.txt** (test data) and im_name variable to images location. A **out_labels.txt** file will be generated.

3. In **get_accuracy.py** set the name of **flickr_data_namestest** variable to **out_labels.txt** and **flickr_data_test** variable to **flickr_logos_27_dataset_query_set_annotation.txt.** Run **get_accuracy.py** file to get the accuracy of classification. Our model accuracy of classification is 92% for images in which logo is detected.

RESULTS

Out of the 135 total test images our model is able to detect logo in 106 images. Now on these 106 images we got a logo classification accuracy of 92.55%. So the results obtained for classification are very good. All the images are inside the **output_images** folder.

The logo object detection can be increased further by training the dataset on higher number of iterations. Due to lower number of iterations our model is not able to detect images on all the images.