Artificial Intelligence

Image Classification & Object Localization

Durga Akhil M.	01FB15ECS097
Gadicherla Sameer	01FB15ECS101
Ganesh K.	01FB15ECS104
Rahul R. Bharadwaj	01FB15ECS366

REPORT

Abstract:

To build a Convolutional Neural Network and train using PASCAL VOC dataset.

Objectives

Our goal is to do the following:

- Using the PASCAL VOC2010 dataset, build a classifier. The classifier will classify 20 across classes using Keras, CNN based models.
- Perform detecting object localization: predict bounding box for each image.
- Evaluate the performance and experiment with transfer learning

Approach: (Please see README for source code details)

Preprocessing:

The following steps were followed:

- The Parser reads the entire data set and returns the images and its respective bounding boxes (coordinates) and classes.
- We then use the coordinates to crop out the ROIs and write it into a folder with class label as name.
- Through this method, we do not discard images with multiple classes and also simplify the parsing process during training.

Training:

- Our CNN is a modified version of VGG16.
- Our model was trained using 40,000 images of PASCAL VOC dataset for 100 epochs.
- We had installed Tesorflow GPU and trained overnight on PASCAL VOC dataset for 1000 epochs. But the script crashed before saving the model.
- Our Layers are shown in Figure 3.

Localization:

- Localisation problem is solved using sliding window with varying sizes.
- The sliding window aims at determining the location of an object in the test image and label it with the right class.
- The window initially starts with a size of 32×32. Then with a stride length of 16, it traverses through the image.
- Once the window goes through the entire image, its size is increased by 32 pixels.
- Each window is feed into the predictor to get class along with confidence.
- We chose sliding over all other methods as it was easier to implement.

- Output is shown in Figure 1, 2.

Transfer Learning:

Transfer learning helps in reusing the features learnt by a well-trained CNN.

- Our base Neural Network consists of Resnet. We use pretrained weights [link].
- We then Define the Region Proposal Network(RPN) built on the base layers.
- Training is done on the PASCAL VOC dataset for 100 epochs.
- We then tested it on various images and got the right labels with an average confidence of 95%.

Keras Layers

Our basic model is similar to VGG-16 model but with 10 layers. The following
type of layers has been used:

1. *Conv2D Layers:* This layer creates a convolution kernel that is convolved with the layer input over a single spatial (or temporal) dimension to produce a tensor of outputs. ReLU activation has been applied to the outputs of these layers.
2. *MaxPooling2D Layer*s: Max pooling operation for spatial data. It is also referred to as a down sampling layer. This basically takes a filter and a stride of the same length. It then applies it to the input volume and outputs the maximum number in every sub region that the filter convolves around.
3. *ZeroPadding2D Layers:* This layer can add rows and columns of zeros at the top, bottom, left and right side of an image tensor.
4. *Dropout Layers:* The idea of dropout is simplistic in nature. This layer “drops out” a random set of activations in that layer by setting them to zero. It makes sure that the network isn’t getting too “fitted” to the training data and thus helps alleviate the overfitting problem. An important note is that this layer is only used during training, and not during test time.
5. *Dense Layers:* The model ends with several Dense Layers. One of the Dense Layer being the Softmax Layer. Softmax layer's purpose is converting any vector of real numbers into a vector of probabilities (nonnegative real values that add up to 1). Within this context, the probabilities correspond to the likelihoods that an input image is a member of a particular class.

Gallery:

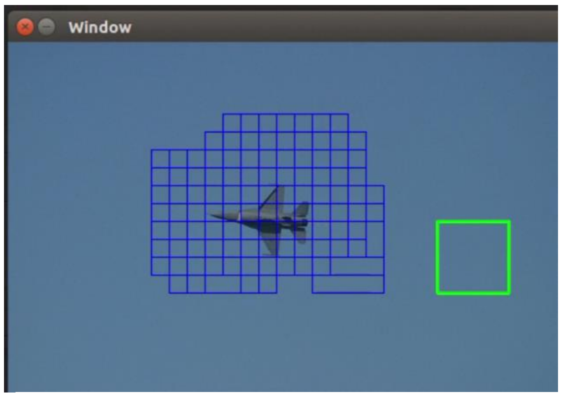


Figure 1 Running Sliding window. Blue represents object might be present and green represents current window

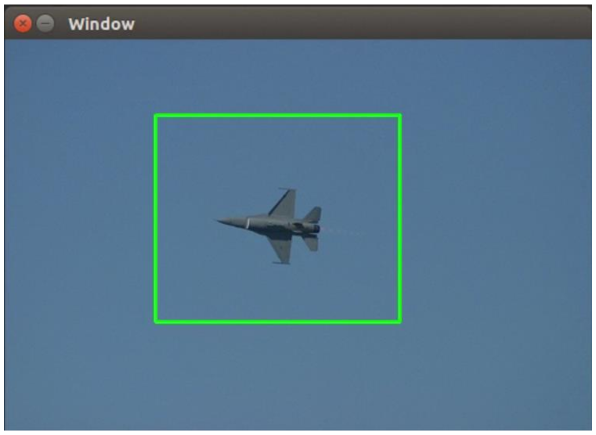
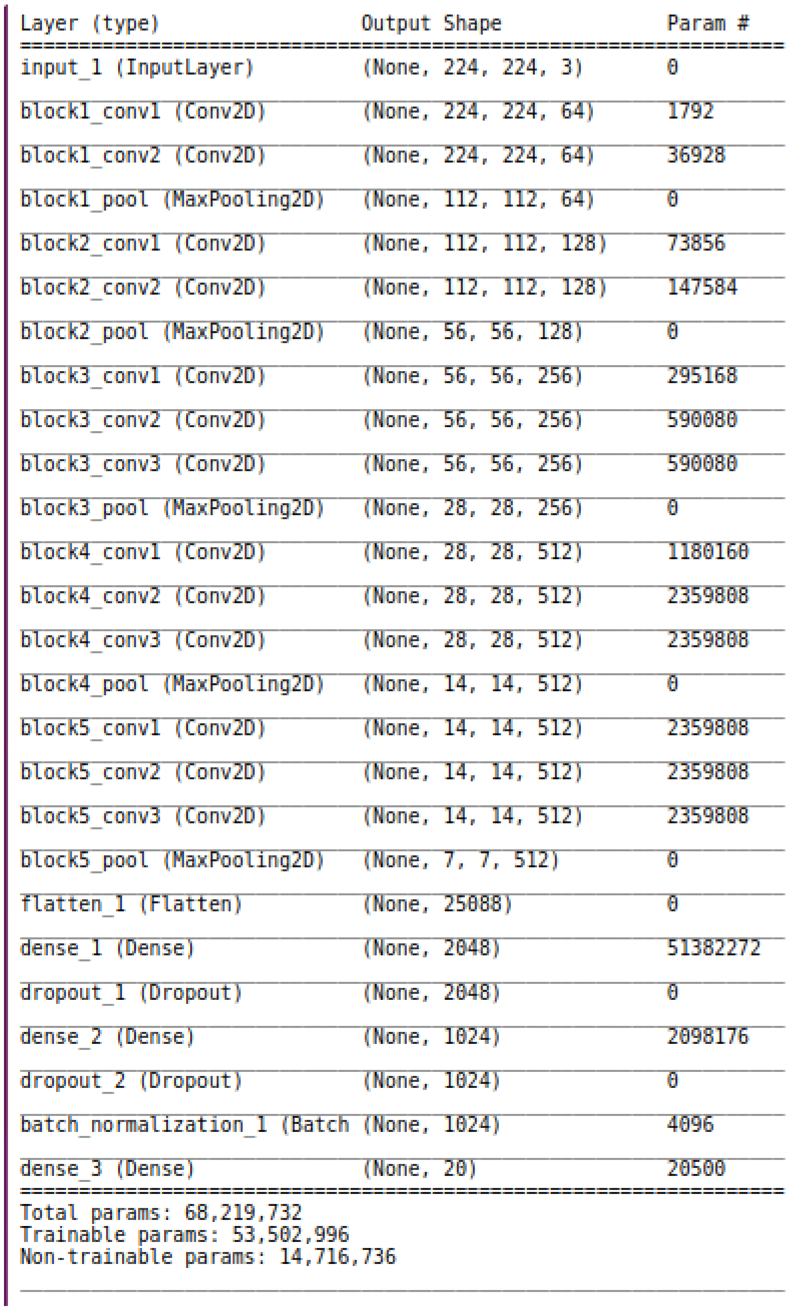
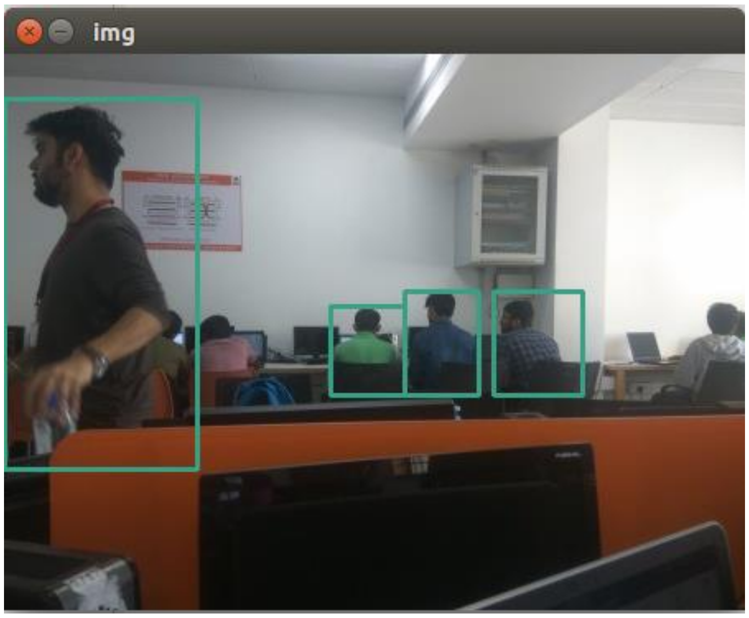


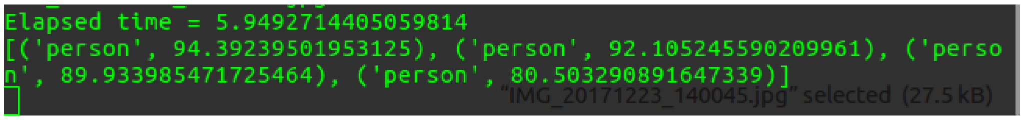
Figure 2 Final image location predicted



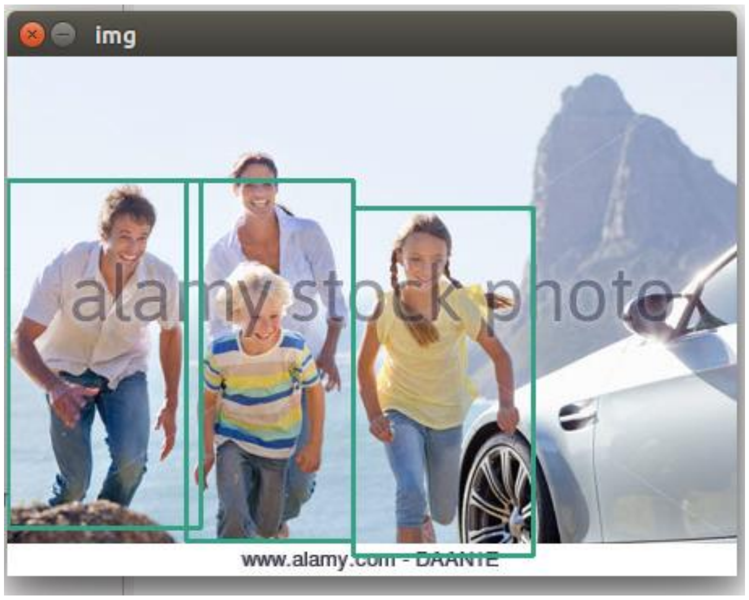
```
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0         
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0         
_________________________________________________________________
dense_1 (Dense)              (None, 2048)              51382272  
_________________________________________________________________
dropout_1 (Dropout)          (None, 2048)              0         
_________________________________________________________________
dense_2 (Dense)              (None, 1024)              2098176   
_________________________________________________________________
dropout_2 (Dropout)          (None, 1024)              0         
_________________________________________________________________
batch_normalization_1 (Batch (None, 1024)              4096      
_________________________________________________________________
dense_3 (Dense)              (None, 20)                20500     
=================================================================
Total params: 68,219,732
Trainable params: 53,502,996
Non-trainable params: 14,716,736
_________________________________________________________________
```

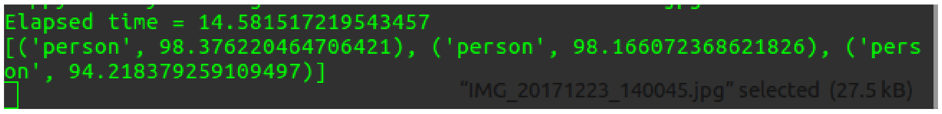
Figure 3 The Model Summary





```
Elapsed time = 5.9492714405059814
[('person', 94.39239501953125), ('person', 92.105245590209961), ('perso
n', 89.933985471725464), ('person', 80.503290891647339)]
                                            "IMG_20171223_140045.jpg" selected (27.5 kB)
```





```
Elapsed time = 14.581517219543457
[('person', 98.376220464706421), ('person', 98.166072368621826), ('pers
on', 94.218379259109497)]
                                        "IMG_20171223_140045.jpg" selected (27.5 kB)
```

Figure 4 Output for Transfer Learning with class and confidence

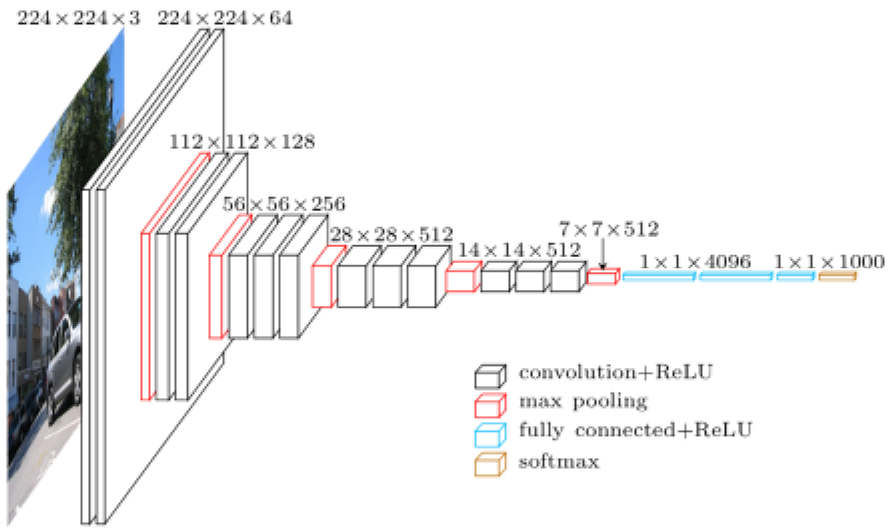


Figure 5 VGG16 Layers

Result Analysis

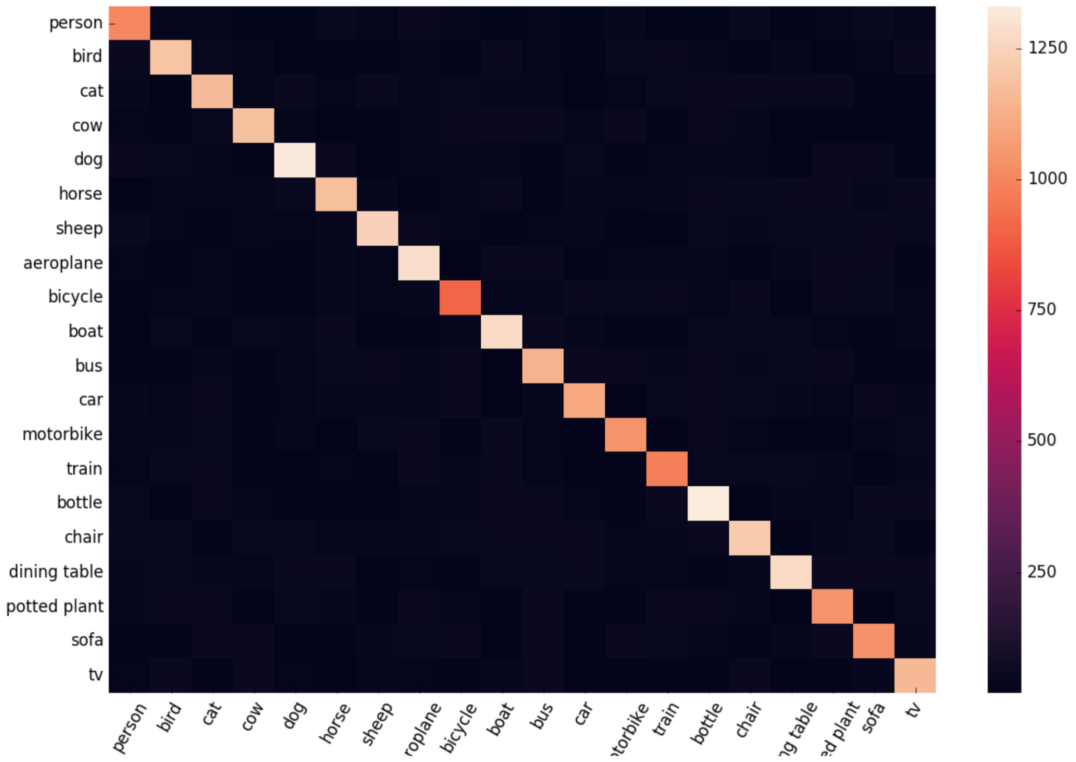


Figure 6 Confusion Matrix with 50% accuracy. Legend shows number of test cases.

Part A:

- Accuracy: 51%
- Epochs: 100
- Training Samples: 40,158

Part B:

- Time per image: 180 Seconds
- Accuracy: 85% (IOU)
- Average Confidence: 90%

Part C:

- Accuracy: 95%
- Epochs: 100
- Training Samples: 40,158
- Time per image: 5 Seconds
- Average Confidence: 98%

Acknowledgement:

- This project is developed as part of end of semester ESA for Artificial Intelligence Course.
- We would like to thank our professors, Anantharaman Iyer and Shruti Kaivalya
- The F-RCNN Code is based on this git repository.
- Our VGG16 modifications were based on this git repository.