04/20/2017

CSC 720 – ARTIFICIAL INTELLIGENCE II

Final Project Report
Spring 2017

Submitted by:

Nikhil Sharma
Unity ID: nsharm12
Student ID: 200131458
NCSU

ABSTRACT

The backpropagation algorithm has played a central and key part in Neural Networks since its inception. Convolutional Neural Networks in particular have significantly advanced the state-of-the-art in many different artificial intelligent tasks like object detection, speech recognition, machine translation, etc. Given enough computational power, Deep Learning can perform complicated AI tasks. However, there are certain limitations when it comes to dealing with real-world data. This project aims at exploring the power (and limitations) of Deep Neural Nets in one such task, where the aim is to combine different object detectors and comment on the performance of it.

Contents

1 Problem Statement

The objective can be divided into two parts

1) Training Detectors:
    - Human Upper Body Detector: Train a Convolutional Neural Network (CNN) detector that is capable of detecting and localizing a human upper body in a given image
    - Object Detector: Train a CNN to detect and localize objects in a given image. Objects here refer to any of those found in day-to-day life, like Hats, Bottles, Cars, etc. For the current problem, three object categories have been considered, viz. Cars, Dogs, and Background.
      In fact a part of the objective is to see how difficult it gets for a single detector to classify an object as the number of object categories increases.

2) Combining the Detectors:
    - The idea here is to see how the detectors could be combined to do multiple detections in an image. For instance, if you were to look up for images which had a person wearing a red shirt standing near his car, then you could run the upper body detector to detect whether or not a person is present in the image, find the color of the cloth they are wearing (the technique of which is discussed later), and find whether or not there is a car in the picture. Note that in the current approach, the car could be anywhere in the picture, and not necessarily near the person.
      Further, the focus currently is on the CNN part and evaluating its performance. Ways to parse a given sentence and extract the different detections to be performed have not been touched.

The detectors were trained in Torch's Deep Learning environment, with Lua scripting language.

2 Upper Body Detection

The objective is to design and train a CNN detector that would be able to tell whether or not a given image has a human upper body, and if there is, it would localize the region.

2.1 Design of the CNN

The network is trained to classify a given image into two classes: Upper Body and Background.

The architecture of the entire network is shown in the image below.

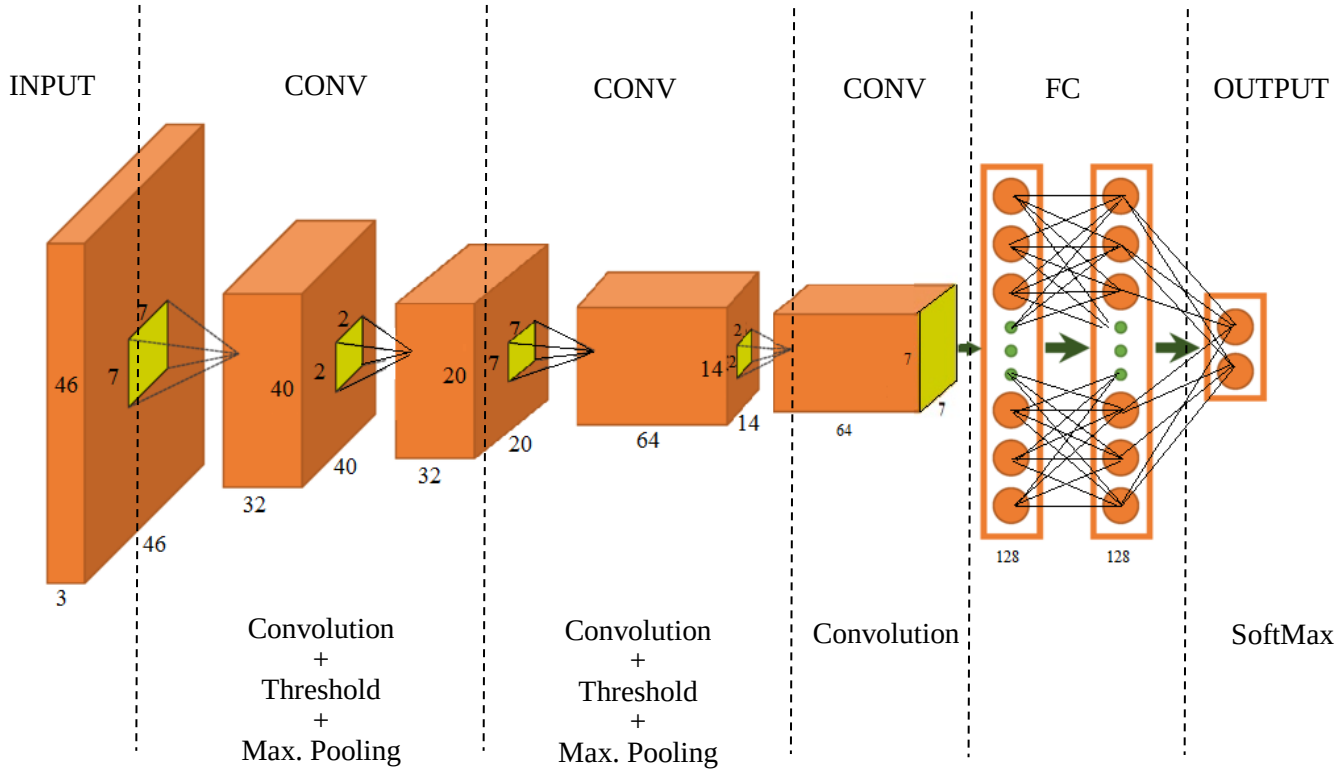


Figure 2.1-1 CNN architecture

* CONV refers to Convolutional Layer

* FC refers to Fully Connected Layer

2.1.1 Input Layer

The input is a 8 bit image, of dimension 3 × 46 × 46

Sample input images are shown in the later sections

2.1.2 CONV Layer 1

This layer has three stages:

- Convolutional Layer: A 7 × 7 convolutional kernel is applied to the input image, with zero padding and stride equal to 1.
  32 features are extracted from the input image.

- Thresholding: The output from the previous layer is thresholded at 0, which introduces the non-linearity in learning the features.

- Max Pooling Layer: The output from the previous stage is max-pooled by a 2 × 2 sliding window.

2.1.3 CONV Layer 2

This layer is similar to the previous convolutional layer, except that in this stage, 64 features are extracted from the 32 features of the previous stage.

2.1.4 CONV Layer 3

This layer has just the convolutional layer, and does not include the thresholding and max-pooling operations. At this stage, 128 features are extracted from the 64 features of the previous stage's output.

2.1.5 FC Layer 1

This is a linear stage that connects (fully) 128 neurons from the previous stage to 128 neurons in the next stage.

2.1.6 FC Layer 2

This layer reduces the 128 neurons to 2 output values, which are passed through a softmax function to compute the log probabilities. These values are the final outputs of the network, and are basically the confidence scores for each of the two classes.

2.2 Training the CNN

2.2.1 Network Parameters

- Learning rate = 1e-3
- Learning rate momentum = 0.1
- Batch size = 128
- Training method = Stochastic Gradient Descent

2.2.2 Data

The images from training and testing have been collected from about 3 places on the web.

- INRIA Person Dataset (http://pascal.inrialpes.fr/data/human/):
  This dataset contains about 4k images of upright people, which were in turn collected from images and video sequences of people on the streets. It also provides bounding box co-ordinates which indicates where exactly a person is located in a given image. These co-ordinates have been used to crop out the upper bodies from all the images, and used as training data for class Upper Body.
  It also has about 4k background images, which are images from the streets without a person present in any of them. These images have been used as training data for class Background.

- CUHK 03 (http://www.ee.cuhk.edu.hk/~xgwang/PS/dataset.html):
  This dataset was collected from the Chinese University of Hong Kong campus. It was sourced from videos acquired by a fixed camera on campus, that captured people as they moved around. The dataset itself comes with about 8k cropped images of the entire body. These have further been cropped into two halves, and the upper half has been used as training data for class Upper Body.

- Background Dataset: (http://dags.stanford.edu/projects/scenedataset.html and http://3dvis.ri.cmu.edu/data-sets/localization/)
  These are the datasets from Stanford and CMU for scene recognition. A subset of these images that had no person in them was separated. Out of those 400 images of size 640 × 480, about 28k images of size 46 × 46 were generated and used for training the Background class.

On total, the dataset had about 42k images, 10k of which were of upper body, and the rest 32k of background. The reason for choosing such high number of background images when compared to upper body images lies in the very sense of what a background is: it is literally anything that has no human body in it, and as such, there are a wide variety of images that the network needs to learn as Background.

The dataset was divided into two sets: one for training and one for testing, in the ratio of 80:20 respectively. No cross validation techniques were used, and as such there were no validation dataset splits.

Shown below are some samples of the data (of both the classes)

Figure 2.2-2 Some Training Samples

The training data consists of about 8k samples of class Upper Body and about 26k samples of class Background.

The images were normalized to zero mean and unit variance before training. No other pre-processing was done on the images.

The test data consists of about 2k samples of class Upper Body and about 6k samples of class Background.

2.2.3 Results

It took about 140 iterations for the network to converge, which was observed from the accuracy of the training set. Figures below plot the value of accuracy (%) over the 140 iterations

Figure 2.2-2 Training Accuracy

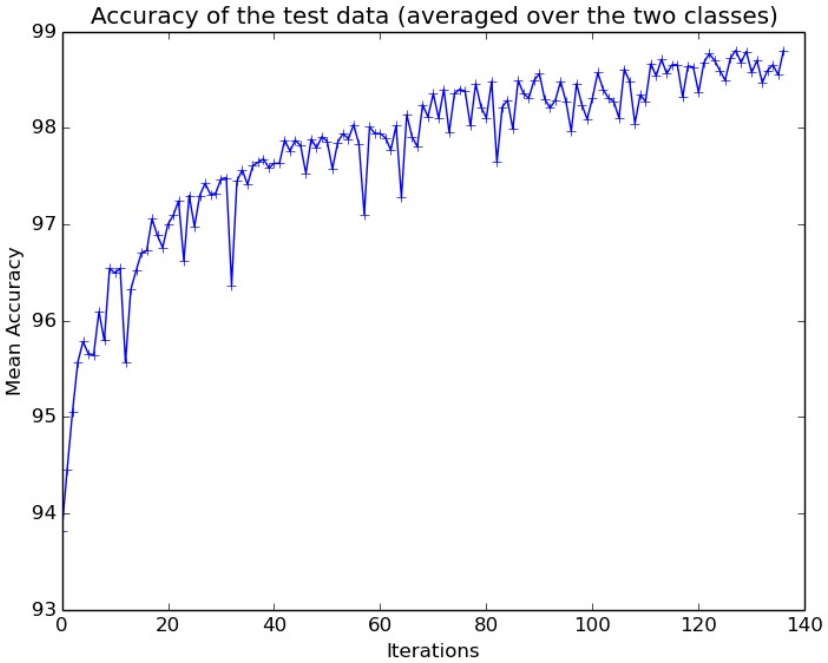


Figure 2.2-3 Testing Accuracy

The figure below gives an insight of the converged values. It shows the final confusion matrix, the classification accuracies over the training set and the test set, and the time it took to train and test one sample.

```
==> doing epoch on training data:
==> online epoch # 137 [batchSize = 128]
 [==================== 35201/35312 ===========>.]  ETA: 906ms | Step: 8ms
==> time to learn 1 sample = 8.1482129057555ms
ConfusionMatrix:
[[    8415     132]   98.456%   [class: body]
 [     116   26537]]  99.565%   [class: backg]
 + average row correct: 99.010187387466%
 + average rowUcol correct (VOC measure): 98.105677962303%
 + global correct: 99.295454545455%
==> saving model to /home/nikhil/myCode/learning/Torch/AI2/model3-PreTrained-Ext
raImg/results/modelTestV2.net
==> testing on test set:
 [==================== 8705/8829 =============>.]  ETA: 0ms | Step: 23h59m
==> time to test 1 sample = 2.1952222300651ms
ConfusionMatrix:
[[    2141      45]   97.941%   [class: body]
 [      59    6459]]  99.095%   [class: backg]
 + average row correct: 98.51813018322%
 + average rowUcol correct (VOC measure): 96.891421079636%
 + global correct: 98.805147058824%
```

Figure 2.2-4 Training Results

The accuracies reported on the test set are a good measure of how the detector would perform on real world images, since the training and testing images themselves are representative of the real world dataset. In particular, the following points about the dataset are true:

- They were taken from an ordinary pedestrian camera, and not a high resolution camera. This rules out the possibility of the network performing well only on detailed images.
- They have been collected at different times of the day, which leads to a variety of lighting conditions in the images; some of them are brightly lit, while some are poorly lit.
- No post-processing was done on the images to make them more detailed, or suitable for feature extractions. The network was trained with the raw captures directly.

Examples showing the detections on sample images are given after the next section.

2.3 Detecting Upper Bodies

2.3.1 Image Pyramids

The network was trained on images of size 46 × 46, which becomes the size of the detector. However, images that we need to test on might be of different sizes, and the size of upper bodies in them might be way larger than 46 × 46, in which case the detector would fail to detect them. To address this issue, an image pyramid is constructed, where each level of the pyramid is a scaled down version of the input image. Starting with the original dimensions of the image, a series of scales are generated, and the detector is run at each scale. This way, there would be a scale at which an upper body in an image would be scaled down to a size close to 46 × 46, and will thus be detected.

Choosing the right values of scales is tricky. While I was able to get proper results on a good number of images with scales {0.3, 0.15, 0.1}, there were some cases were no detection was successful with these scales, and addition of scales {0.5} and/or {0.4} helped get the detections right.

Figure 2.3-4(a) Detection when 0.5 was included in the list of scales

Figure 2.3-3(b) Detection when 0.4 was included in the list of scales

When the scales used were {0.3, 0.15, 0.1}, no upper body was detected. When a scale of 0.5 was added to the list of scales, an upper body was detected, but it was incorrect. When a scale of 0.4 was added to the list, the upper body was correctly detected.

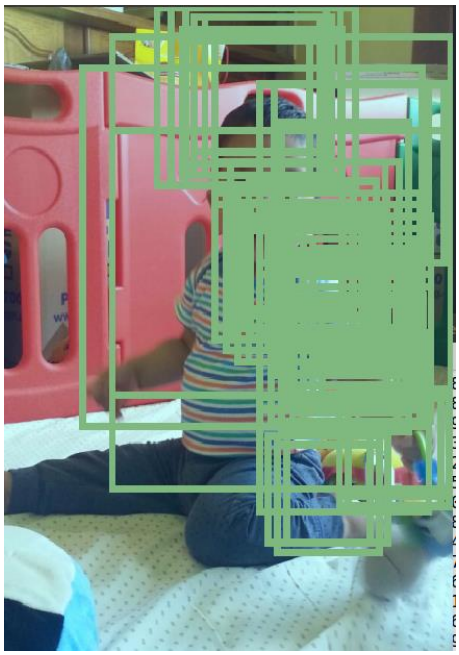


Figure 2.3-2(a) Detection when 0.5 and 0.4 were included in the list of scales

Figure 2.3-2(b) Detection when scales were just 0.3, 0.15, 0.1

2.3.2 Sliding Window

The images that the network was trained on were such that an upper body almost entirely covered the image. However, if were to test the detector on an image where an upper body is in some corner, scaling the image alone will not help. To surmount this, a sliding window detector is used. It is basically a setup where the detector window slides across the image, and at each instance, performs a detection.

2.3.3 Non-Maximum Suppression

Since we run the detections across different scales and positions, there will many detections of the same object, when the results are combined. To address this, the technique of non-maximum suppression is used. It is fairly straightforward: if two detections overlap by more than a certain threshold, keep the detection with the higher score and discard the other. When this technique is applied for all the images, we are left with only the potential upper body detections.

$$Overlap\ score\ of\ two\ detections\ A\ and\ B = \frac{Area\ (A\ \cap\ B)}{Min(Area\ of\ A,\ Area\ of\ B)}$$

This score is compared against an overlap threshold, which is pre-set.

The images that follow show some examples of the number of detections before and after non-maximum suppression.

Figure 2.3-3(a) Before NMS

Figure 2.3-3(b) After NMS

Figure 2.3-4(a) Before NMS

Figure 2.3-4(b) After NMS

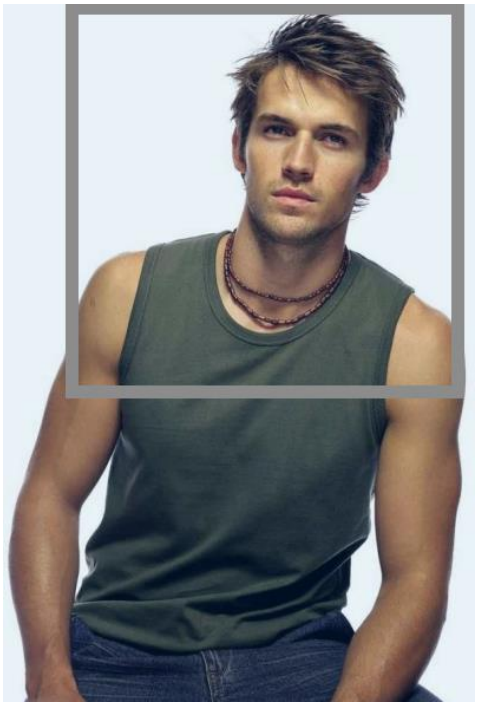
The overlap threshold can have a drastic effect on the resulting detections. Specifically more when it comes to detecting multiple upper bodies in an image.

Figure 2.3-5(a) Before NMS

Figure 2.3-5(b) After NMS, with threshold 0.2. Due to the low threshold value, the detection window for the person in the middle gets suppressed

Figure 2.3-5(c) After NMS, with threshold 0.9. With the high threshold value, the detection window does not get suppressed and the person in the middle is detected

2.3.4 Some More Results



Figures 2.3-6(a-h) Upper Body Detections with Image Pyramid, Sliding Windows, and NMS

Figure 2.3-7 The detector fails to identify the person on the right

Figure 2.3-8 An example where the detector performs poorly

Figure 2.3-9 An example that shows it is easy to fool the detector. Anything that looks like an upper body will be detected, and this is one of the limitations on a CNN. Unlike humans, it has no other knowledge to impart while detecting an object, and its inference is based solely on the computations on the pixels

3 Object Detection

The objective is to design and train a CNN detector that would be able to detect Cars and Dogs in a given image, if present.

3.1 Design of the CNN

The network is trained to classify a given image into three classes: Car, Dog, and Background.

The architecture of the entire network is shown in the image below.

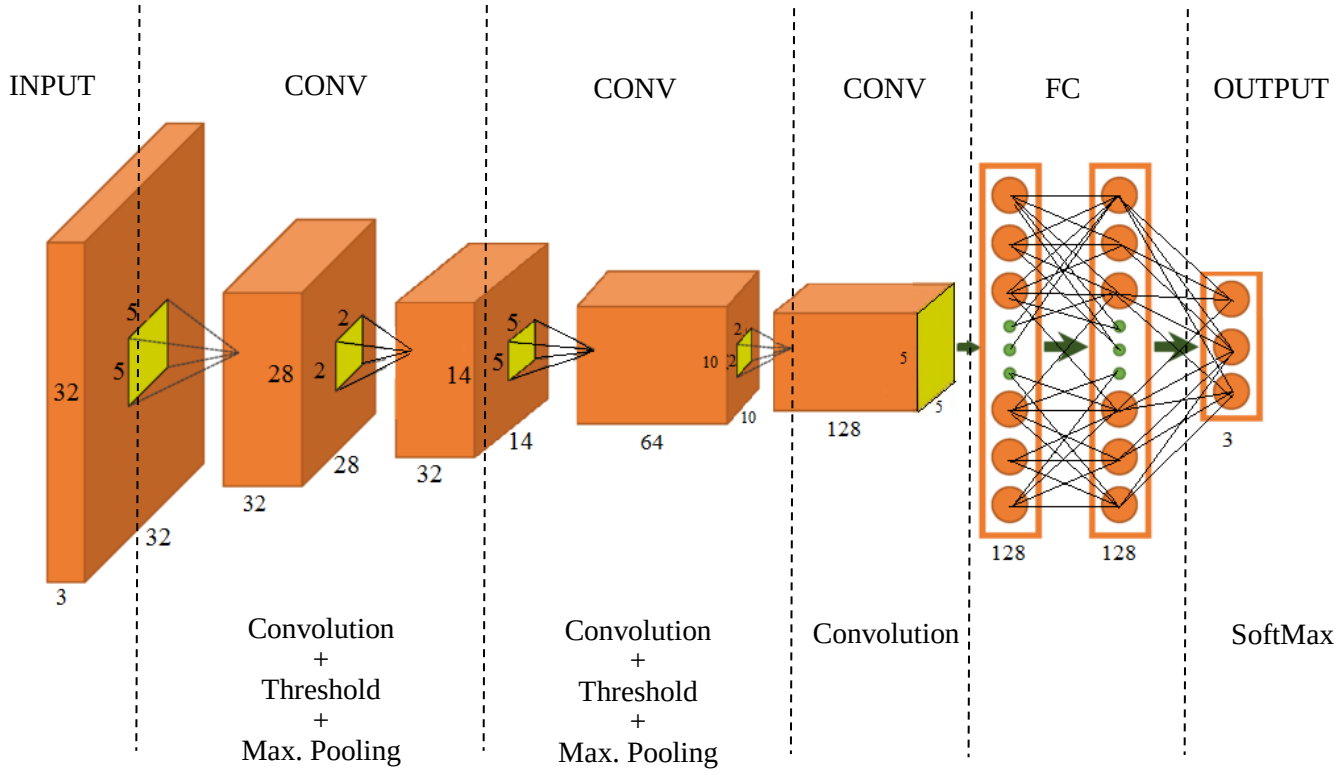


Figure 3.1-1 CNN architecture

* CONV refers to Convolutional Layer

* FC refers to Fully Connected Layer

3.1.1 Input Layer

The input is a 8 bit image, of dimension 3 × 32 × 32

Sample input images are shown in the later sections

3.1.2 CONV Layer 1

This layer has three stages:

- Convolutional Layer: A 5 × 5 convolutional kernel is applied to the input image, with zero padding and stride equal to 1.
  32 features are extracted from the input image.

- Thresholding: The output from the previous layer is thresholded at 0, which introduces the non-linearity in learning the features.

- Max Pooling Layer: The output from the previous stage is max-pooled by a 2 × 2 sliding window.

3.1.3 CONV Layer 2

This layer is similar to the previous convolutional layer, except that in this stage, 64 features are extracted from the 32 features of the previous stage.

3.1.4 CONV Layer 3

This layer has just the convolutional layer, and does not include the thresholding and max-pooling operations. At this stage, 128 features are extracted from the 64 features of the previous stage's output.

3.1.5 FC Layer 1

This is a linear stage that connects (fully) 128 neurons from the previous stage to 128 neurons in the next stage.

3.1.6 FC Layer 2

This layer reduces the 128 neurons to 3 output values, which are passed through a softmax function to compute the log probabilities. These values are the final outputs of the network, and are basically the confidence scores for each of the two classes.

3.2 Training the CNN

3.2.1 Network Parameters

- Learning rate = 1e-3
- Learning rate momentum = 0.1
- Batch size = 128
- Training method = Stochastic Gradient Descent

3.2.2 Data

The images from training and testing have been collected from the following places on the web.

- CIFAR -10 (https://www.cs.toronto.edu/~kriz/cifar.html):
  This dataset consists of contains of about 60k images in 10 classes, with about 6k images per class. For the current tasks, the images that correspond to the classes Automobile and Dogs have been taken, which account to 12k images. These images are already tightly contain the object, and no further cropping is required.

- Background Dataset:
  For the background class, a subset of the images used in the upper body detection's background category were taken. Out of 30k images, about 7k were found to not contain either an automobile or a dog. It was difficult to gather background images for this task when compared to the case of upper body detection.

The dataset was divided into two sets: one for training and one for testing, in the ratio of 5:1 respectively. No cross validation techniques were used, and as such there were no validation dataset splits.

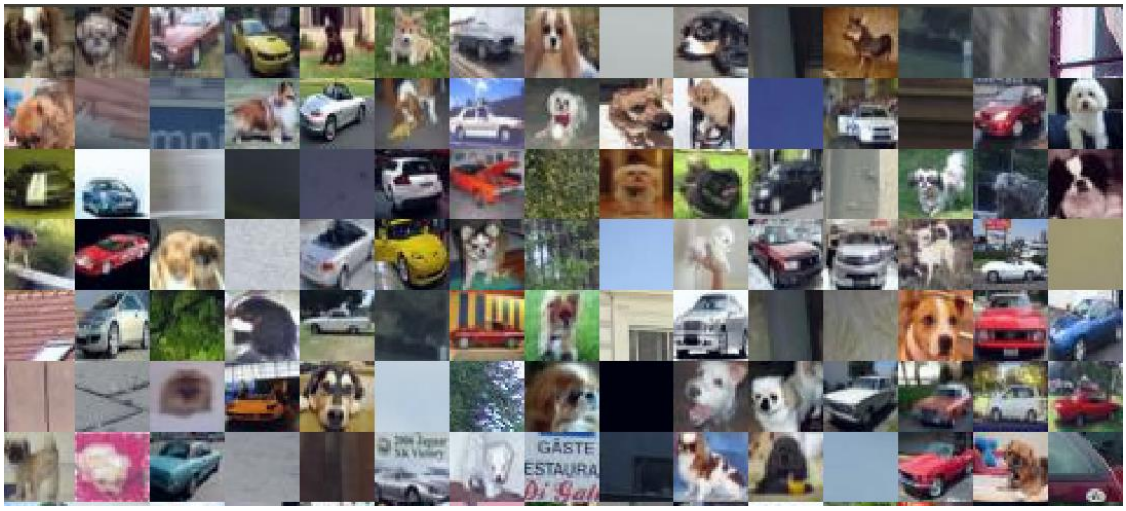


Figure 3.2-1: Some Training Samples

The training data consists of about 5k samples each of class Automobile and Dog, and about 6k samples of class Background.

The images were normalized to zero mean and unit variance before training. No other pre-processing was done on the images.

The test data consists of about 1k samples of each class.

3.2.3 Results

It took about 70 iterations for the network to converge, which was observed from the accuracy of the training set. Figures below plot the value of accuracy (%) over 140 iterations

Figure 3.2-2 Training Accuracy

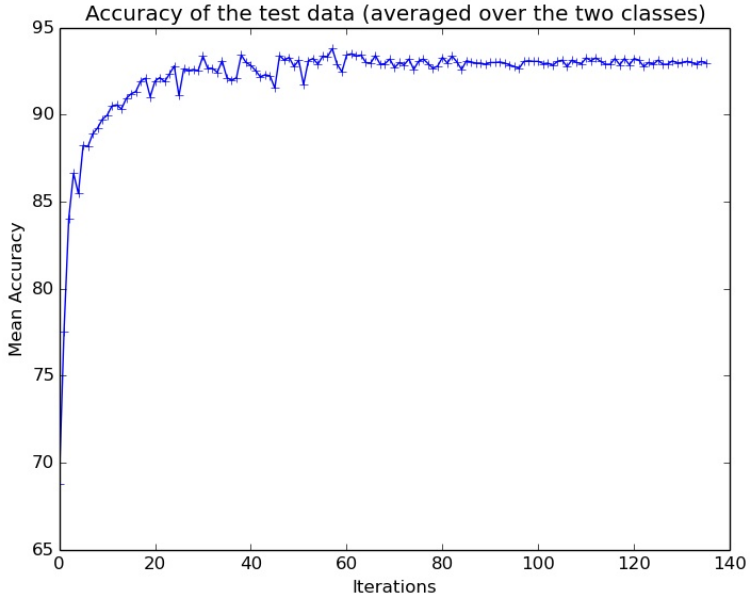


Figure 3.2-3 Testing Accuracy

The figure below gives an insight of the converged values. It shows the final confusion matrix, the classification accuracies over the training set and the test set, and the time it took to train and test one sample.

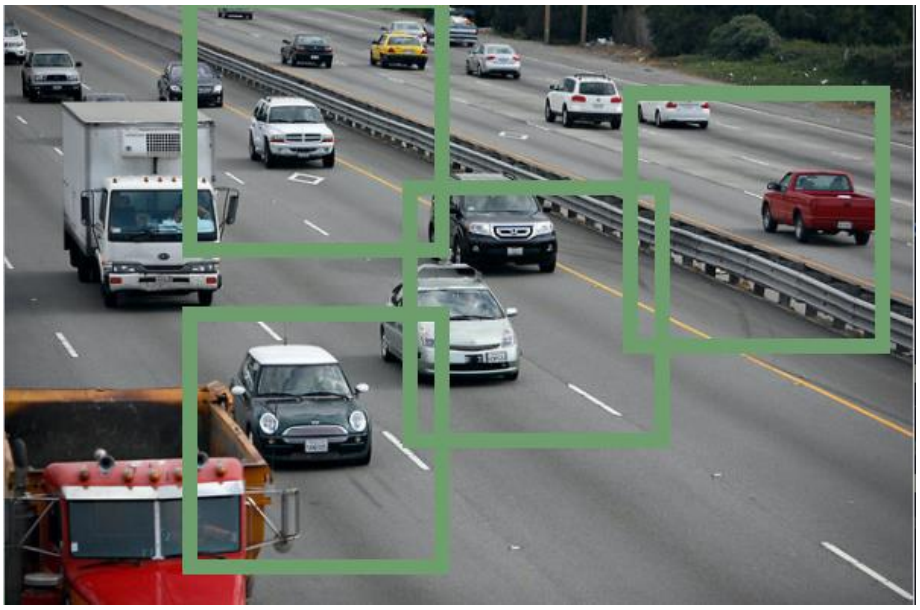
```
==> time to learn 1 sample = 3.6826363642504ms
ConfusionMatrix:
[[    5000       0       0]   100.000%  [class: automobile]
 [       0    5000       0]   100.000%  [class: dog]
 [       0       0    5700]]  100.000%  [class: background]
 + average row correct: 100%
 + average rowUcol correct (VOC measure): 100%
 + global correct: 100%
==> saving model to /home/nsharm12/AI2/cifar-10-batches-py/model5_rgb/results/model.net
==> testing on test set:
 [==================== 2991/3000 =============>.]  ETA: 10ms | Step: 1ms
==> time to test 1 sample = 1.1639306545258ms
ConfusionMatrix:
[[     968      26       6]   96.800%   [class: automobile]
 [      21     962      17]   96.200%   [class: dog]
 [      56      85     859]]  85.900%   [class: background]
 + average row correct: 92.966667811076%
 + average rowUcol correct (VOC measure): 86.812222003937%
 + global correct: 92.966666666667%
```

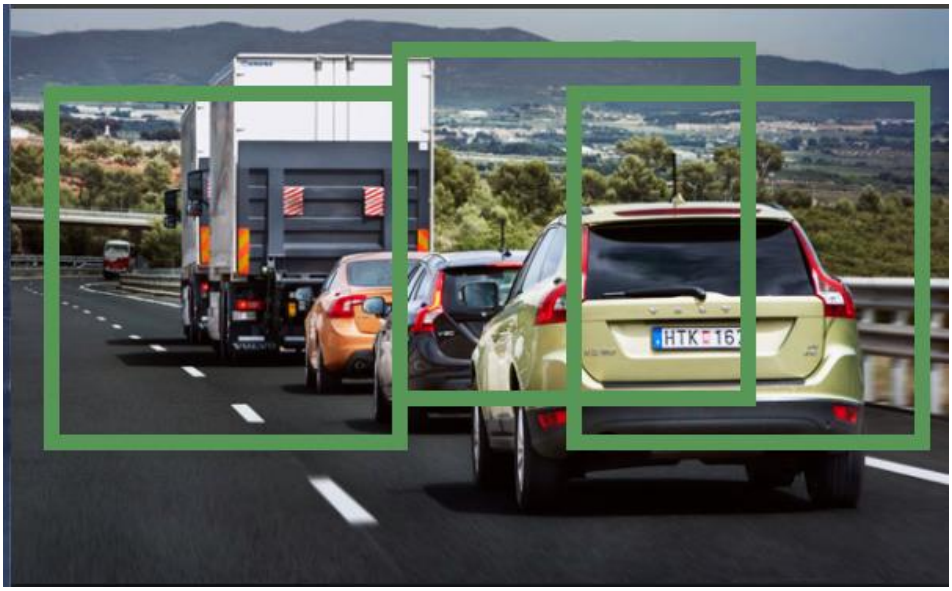
Figure 3.2-4 Training Results

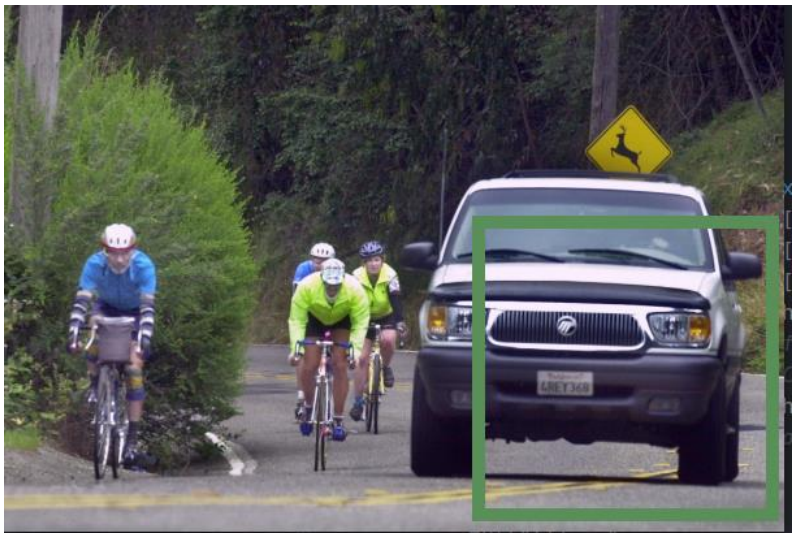
As seen from the results, overfitting occurred during training: while the training accuracies went up to 100 %, the accuracies on the test set were not close.
One potential reason for the low accuracy in the background class is the low number of training images of that class (just 7k, instead of 30k for the upper body detector).

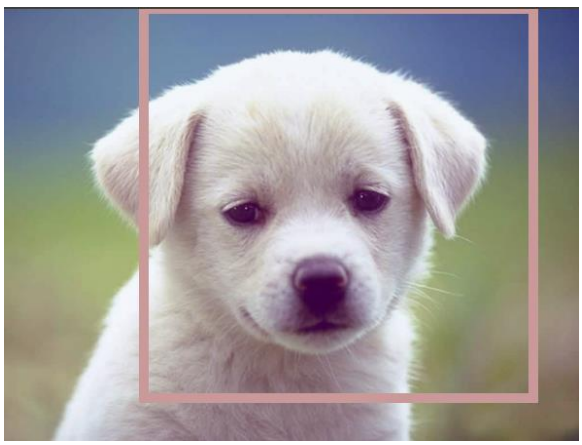
3.3 Detecting Objects

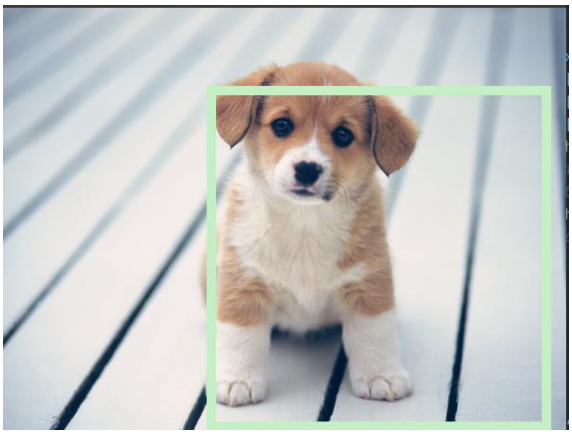
The techniques of Image Pyramid, Sliding Window, and Non-maximum Suppression discussed earlier were used, and the results are shown below







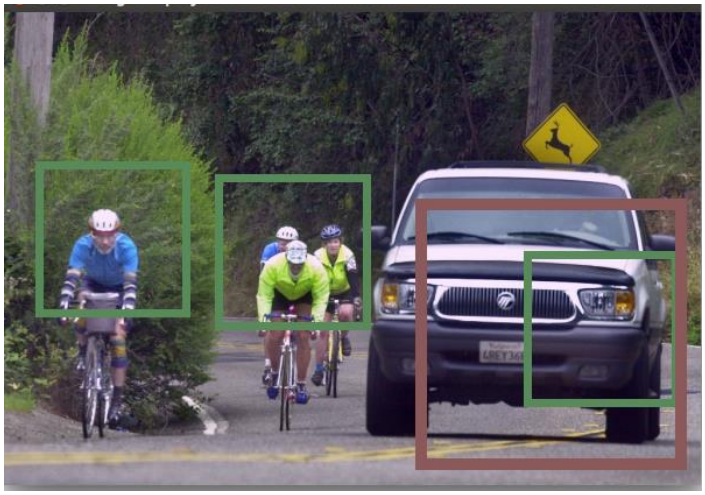


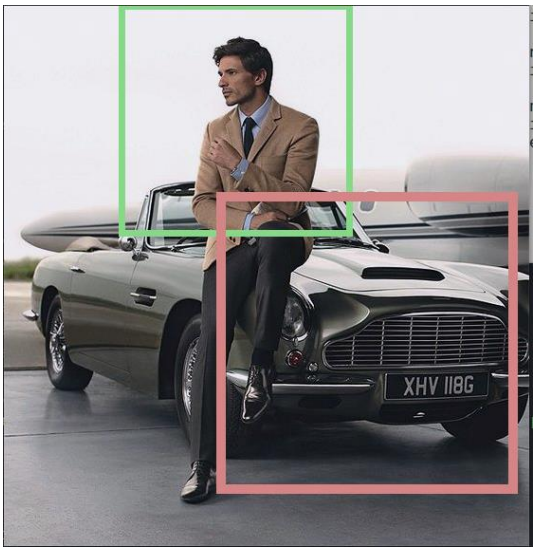


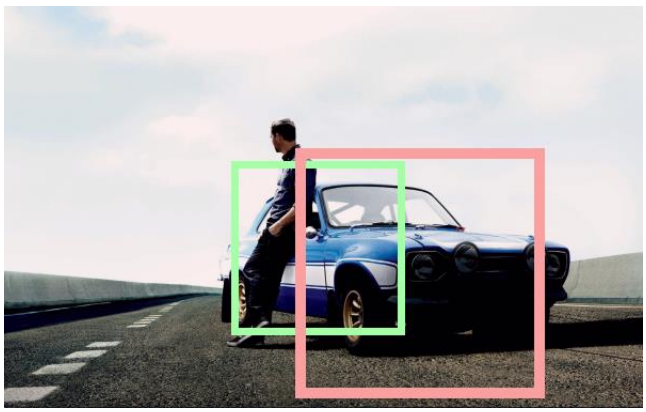
Figures 3.3-1(a-f) Cars and Dogs detections with Image Pyramid, Sliding Windows, and NMS

4 Combining Detections

This section discusses the results of running the detectors trained earlier on a single image, and combining the detections. For instance, if the search query (after parsing) was to look for *cars* and *people*, the following kind of images would show up

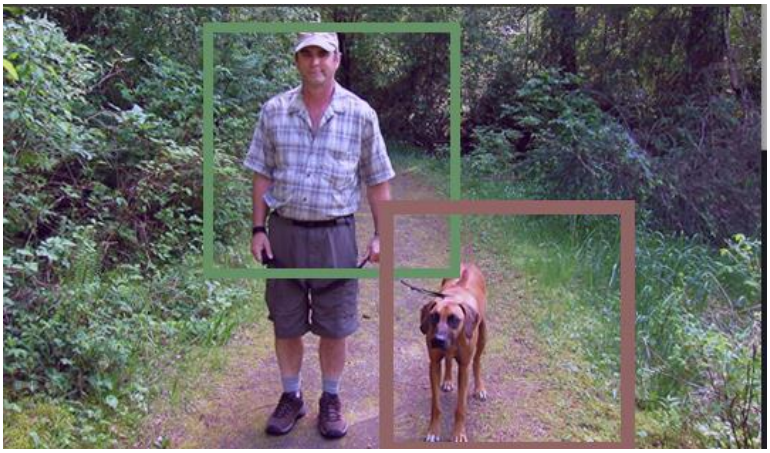






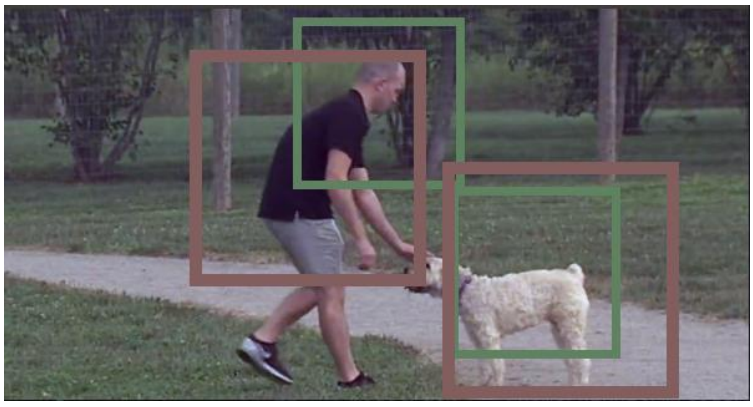
Figures 4-1(a-d) Green boxes are Upper Body detections, and red ones are Car detections

With queries that look for *dogs* and *people*:



Figures 4-2: Green boxes are Upper Body detections, and red ones are Dog detections

*Figures 4-3(a): Detections include False Positives**



*Figures 4-3(b): Detections include False Positives**

* Figures 4-3 (a) and (b) indicate a potential problem with combining results of multiple detectors. As seen in both the images, the upper body detector correctly identifies the upper bodies, and so does the dog detector. However, they falsely identify each other too.
The dog detector in figure 4-3 (a) for instance identifies the upper body to be of class dog. This comes from the fact that the detector does not actually try to learn what a dog looks like. Instead, it tries to learn the differences between the categories dogs, automobiles, and background. The confidence score that the detector outputs for each detection is actually its belief w.r.t the other categories, and not independent of them.

As such, when a human body is given to it in frontal pose, it finds the features therein to be very close to those of a dog than those of an automobile or a background.

4.1 Further Detections

The detectors can further be used to answer queries that are more specific, like ‘person in black shirt with his car’, or ‘people with hat’, and so and so forth.

One such combination tested as part of this project was tagging apparel color with the upper body. Should an upper body be detected in an image, a small patch of it (of size 32 × 32) will be cropped from the center, the color would be extracted from that.

To extract the color, the following technique was used:

- Run k-means on the patch, and group the colors into two clusters. The dominant color of the apparel would be taken as the one (of the three) with highest number of points in the cluster.
- The dominant color is then compared against a database of colors, and the closest one is outputted as the color of the apparel.
- To measure the closeness between two colors, the Delta-LAB measure was used. To do so, both the colors are converted from RGB space to LAB space, and the Euclidean distance between them is found. Closer the colors, lesser is the Euclidean distance. The database of colors consisted of LAB values of the colors black, blue, green, orange, yellow, red, gray, and white.

Figure 4.1-1(a): Upper Body

Figure 4.1-1(b): Patch

Figure 4.1-1(c): Clustering

Sample detections using this technique are shown below

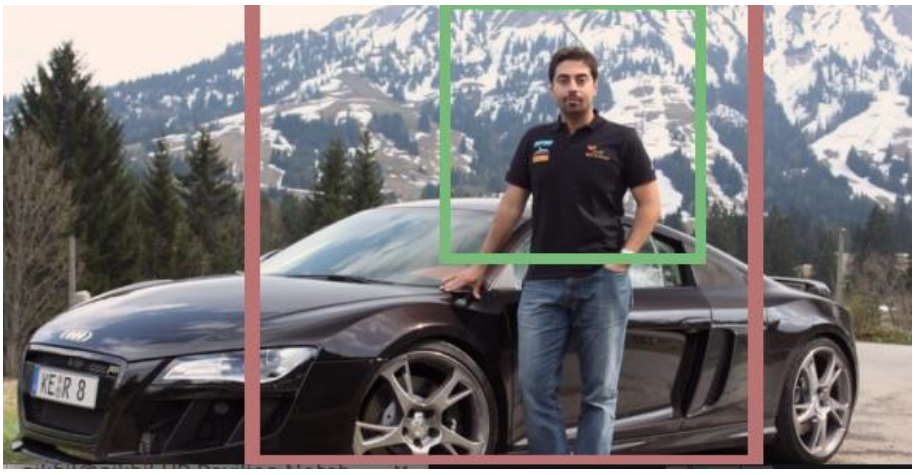


Figure 4.1-2: Upper Body and Car Detected
Color Tag: Black

Figure 4.1-3: Upper Body and Car Detected
Color Tag: Black

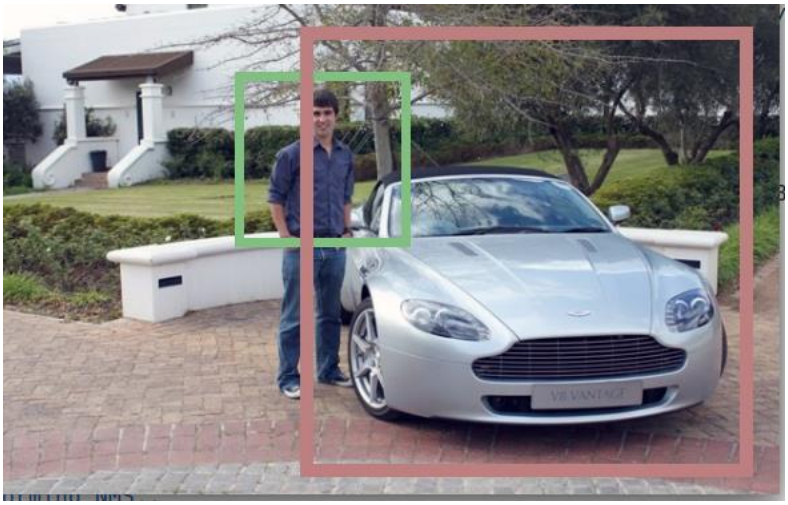


Figure 4.1-4: Upper Body and Car Detected
Color Tag: Blue

5 Conclusions

With the right type and good amount of data, convolutional neural networks can be really good at image classification tasks. However, making the detectors work for a variety of images (which differ in size and spatial location of the objects within the image) requires an appropriate set of image scales, and a good non-maximum suppression technique.

As observed, choosing these values can be tricky and is often not detector invariant. The scales and overlap thresholds that work for one detector might not work for another detector. This introduces some manual trials to set the appropriate factors, which can get tedious. Nonetheless, with the right set of values, the detectors could be powerful.

Identifying multiple objects in one image requires choosing one of the two options: train individual detectors and cascade their results, or train one single detector with multiple classes as targets. The problem with the former, as seen previously, is that the individual detectors learn not the true objects, but the difference between the classes. This might result in false positives of a class that one detector is trained on, by another detector.

The latter method can result in low detection accuracies as the number of classes increase. To verify this, the object detector model used above was retrained on 5 classes, viz. airplane, automobile, cat, dog, and horse.

```
==> testing on test set:
 [==================== 4996/5000 =============>.]  ETA: 5ms | Step: 1ms
==> time to test 1 sample = 1.4713657855988ms
ConfusionMatrix:
[[     888      39      40      14      19]   88.800%   [class: airplane]
 [      34     909      30      14      13]   90.900%   [class: automobile]
 [      41      20     654     218      67]   65.400%   [class: cat]
 [      23      11     211     681      74]   68.100%   [class: dog]
 [      30       8      58      95     809]]  80.900%   [class: horse]
 + average row correct: 78.819999694824%
 + average rowUcol correct (VOC measure): 66.328020691872%
 + global correct: 78.82%
```

Figure 5-1: Results of 5-classes of CIFAR-10 data, on the test set

The corresponding training accuracies had reached 100 % for each class, which means that the network underwent a lot of overfitting. The values from the confusion matrix show that the network could not properly separate a cat from a dog, which is reasonable given the visual features they have. Had this been a classification of just a cat and a dog, the network would have learnt the tiny differences between the two of them. However, with the other classes present, the features of dogs and cats are a lot more similar to each other than any of the other 3 categories. In particular, as the correlation between the features of the classes increases, the accuracy decreases and it becomes difficult to identify those classes.

6 Future Scope

The background images used for training both the detectors could be improved upon. Currently, they were generated by sliding a window over the 640 × 420 image, which meant that there was some correlation in the training images of that class.

As discussed in reference [1], the images which were classified as false positives could be regarded as the *hard* ones to learn, and the network can be retrained with the addition of these false positives as negative training data.

There are some papers, like [2] and [3], that give better methods to reduce multiple detections, instead of using non-maximum suppression. These methods, as said by the paper, can help avoid the problems discussed so far with overlap criteria and threshold value.

The overfitting seen in Object Detection can be eliminated by using techniques like Dropout or PCA, which essentially reduce the dimensionality of the data. A good alternative to working with CNNs would be to extract principle features from the training data, and then train a SVM classifier.

Another way of reducing the false positives or redundant number of true positives could be by using a sliding window during the training process itself. As the detector slides over a training sample, all instances except for the one where the window exactly fits the object can be used as negative training samples.

Figure 6-1(a)

Figure 6-1(b)

Figure 6-1(c)

Figure 6-1(d)

Figure 6-1(e)

Figure 6-1(a-e): As the window slides over the image at a given height, the corresponding sub-images can be noted as negative samples (or positive) based on the location of the true window of the object. In these examples, all the red windows can be fed as negative samples, and the green window as positive sample

7 References

[1] A. Shrivastava, A. Gupta, and R. Girshick “Training Region-based Object Detectors with Online Hard Example Mining”, in *CVPR 2016*.

[2] A. Neubeck, L. Van Gool, “Efficient non-maximum suppression”, in *ICPR 2006*

[3] R. Rothe, M. Guillaumin, and L. Van Gool, “Non-maximum suppression for object detection by passing messages between windows”, in *ACCV. Springer, 2014*