Matlab Pedestrian Detection

1.0 Scope

An implementation of the Convolutional Neural Network (CNN) was used to conduct the detection of pedestrians in photos and videos. The project expected to be implemented in Matlab had to achieve high performance and accurate level of recognitions. Hence, the MatConvNet toolbox was implemented as it efficient and able to run the state-of-the-art CNNs [1].

Furthermore, due to the limited resources (training data and hardware) as well as the time constraints on the project, a pre-trained CNN model was fine-tuned to apply to the specific purpose of pedestrian detection. This is because the convolutional networks generalize learnt features surprisingly well to various recognition tasks, regardless if they were trained to perform recognition on different elements. This means that we were able to achieve accurate detections while using a relatively small set of data.

2.0 MatConvNet-Vgg

Initially, a pre-trained *Matconvnet-vgg-f* was applied. This is because it appeared to be a good compromise between accuracy and time performance. The authors claimed it was able to process 2482.7 images per second and achieve the top-1 error score of 41.4 [2].

2.1 Fine-Tuning

The *matconvnet-vgg* network consists of two fully connected convolutional layers which consist of convolution, ReLU, normalization and pooling. On top of them there are additional convolutional-ReLUs, activations and classifiers combined with a softmax function used for the categorization of the extracted features [3].

The final classification layer was replaced with a new layer. This is because MatConvNet was trained for 1000 different categories on millions of images. However, the features learnt by the network are only a starting point and the new classification is binary: people and not-people (background). Softmax was replaced to train the new classifications.

For better generalization of features, a normalization layers are replaced and new droput layers are inserted for the learning phase. The dropout layers randomly turn on and off. When they are active they discard a set of features. Furthermore the network contained old normalization technique. A new more accurate batch normalization was introduced.

Drop out layers were introduced to randomize the connections in the layers. This was done to stop over-fitting and a wider range of features adjusted during the training process [3]. The layers would randomly turn on and off. Furthermore old normalization techniques were replaced with newer batch normalizations. This is because they permit the detection of high-frequency features with a big neuron response, while damping responses that are large in a local neighbourhood [3, 4]. This resulted in 30% accuracy improvement.

2.2 Learning & Data Augmentation

The CNN learns the features by iterating through the entire dataset *n* number of epochs. At the end of each epoch it evaluates the accuracy of detections based on the test set. The learning data was split into 80% training and 20% test. This resulted in optimal results [3].

On each epoch the CNN grabs the image-label sets in batches. Different transformations were applied to further improve the feature generalization of the learning set. This included flipping, image distortion (through colour manipulations), rotations and zooming in and out. Those transformations were applied at random so that they would vary in each epoch. However, only a small (~3%) improvement was noticed in the detection accuracy.

The data was also expanded to contain as many images of pedestrians as negative samples. Those included cropped out backgrounds and images from different training datasets. This resulted in an enormous improvement, particularly after apply negatives relevant to the pedestrian scenario such as cars, streets, grass, buildings and trees.

Finally, the learning rate was adjusted on each epoch. Initially the rate was high so that the network would learn more features. However, to prevent over-fitting, the rate decayed with each epoch from 0.0001 to 0.00001. This resulted in improved accuracy of detection.

2.3 Sliding Window

To detect people in the image, a sliding window algorithm was used. The algorithm cuts the image into small overlapping windows. Those windows are submitted to the trained CNN to evaluate whether there is a pedestrian there or not. If positive, a border is drawn on the image around the window. The algorithm was optimised in following ways:

- Pre-processing the window locations through matrix operations which results in 1D loop. Removing the nested loops resulted in a 30% speedup.
- Window sizes are scaled by two. This reduced the number of windows.
- Stride of half the window size resulted in optimal performance vs. accuracy trade-off.
- If no detection is made, the next window is skipped. This reduced in 40% speedup without compromising the detections.
- The acceptance threshold on the detection was set above 75%.

2.4 Non-max Suppression

Non-max suppression was applied multiple times at different stages. It was applied for each set of bounding boxes per sliding window size. This resulted in the selection of the most confident detections. Then, it was applied for the final time when all the different sliding window size bounding boxes are added to a single list before they are drawn on the output image. This reduced the number of false positives.

2.4 Results

The combination resulted in an acceptable pedestrian detection. In 40 test pictures of 640x480, the CNN detected 80% of the pedestrians. However, the algorithm averaged 22s per picture and resulted in 30% of false positives.

3.0 HOG-SVM CNN

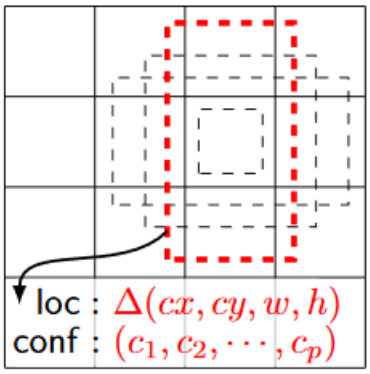
To further improve the detection accuracy, a more heavily pre-trained *Matconvnet-vgg-m* was applied. With 1212.5 images/second it is considerably slower however it achieves 36.9 top-1 error. The network was fine-tuned and trained in the same manner as outlined in section 2.0. This resulted in a 92% detection rate with 5% false positives. However, the average time it took to use the sliding window was 50s per image.

Hence, a HOG-SVM detector was trained on the same dataset. It extracted the Histograms of Orientated Gradients as features and submitted them to a binary linear classifier Support Vector Machines. This resulted in a sliding window detector averaging 1.1s. However, it was not accurate and detected only 50% of the pedestrians. Hence, Matlab's Vision Toolbox pre-trained People Detector was fine-tuned [5]. This is because it was considerably faster as it was optimised for Matlab and was trained on a larger dataset. It was further was improved on by adding a Gaussian smoothing (σ=2) to each image before extraction of HOG features. This resulted in an accurate detection of people in an average of 0.2s.

However, HOG-SVM resulted in many false positives. Hence, it was used as region proposal method. With no threshold set on the minimum acceptance score, the outputs of the HOG-SVM detector served as inputs to the fine-tuned CNN. This resulted in a 90% detection rate and averaged 1s with 10% false positives.

Single Shot MultiBox Detector (SSD)

Finally, the optimal solution was achieved through Samuel Albanie's [6] implementation of the SSD [7]. Small convolutional filters were applied to feature maps through which SSD predicts classifications, scores and region locations. This means that it achieves an excellent time performance. The filters were applied for a fixed set of default bounding boxes and for various different scales of the feature maps. This resulted in high detection accuracy even on low resolution images. These design features lead to simple end-to-end training similar the method in section 2.0. Hence, the pre-trained *ssd-pascal-vggvd-300* was trained in the similar manner. In particular, the image transformations resulted in more noticeable improvements of detection accuracy [6, 7]. However the learning rate was kept constant. Furthermore, the initial categories in the network were kept and not replaced. This resulted in 98% detection accuracy with an averaged time of 0.7s per image and 5% false positive rate.



SSD cannot be implemented in MatConvNet without Matlab's Vision Toolbox [5, 6].

Video Detection

Pedestrian detection in videos was achieved through the combination of the background subtraction and CNN. The videos were read frame by frame. Each frame was converted to gray-scale and subtracted from the previous frame. A binary image was obtained by applying a threshold (>50) on the output. The resulting frame only contained pixels that changed between the frames. Through the sliding window method, the regions of significant change (sum of pixels above threshold) were submitted to the trained CNN to evaluate the detection. Hence, the network only had to compare a handful of regions in the entire frame resulting in a considerable performance increase. The algorithm was further optimised by preforming image erosion and dilation on the binary image. This was done to minimise the regions marginal movements such as trees or clouds.

References

[1] MatConvNet. MatConvNet: CNNs for MATLAB. Retrieved May 21, 2018 from http://www.vlfeat.org/matconvnet/

[2] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Return of the Devil in the Details: Delving Deep into Convolutional Nets. Proceedings of the British Machine Vision Conference 2014 (2014). DOI: http://dx.doi.org/10.5244/c.28.6

[3] Andrej Karpathy. 2018. CS231n: Convolutional Neural Networks for Visual Recognition. Retrieved May 28, 2018 from http://cs231n.github.io/neural-networks-2/

[4] Andrej Karpathy. 2018. CS231n: Convolutional Neural Networks for Visual Recognition. Retrieved May 28, 2018 from http://cs231n.github.io/convolutional-networks/

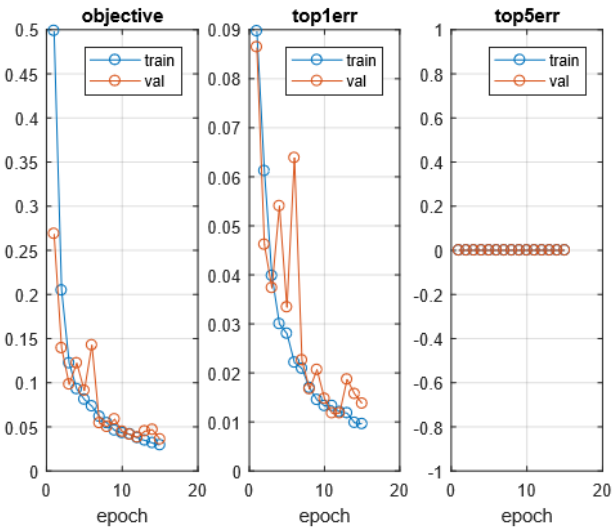
[5] MATLAB. ClassificationModel. Retrieved May 30, 2018 from https://au.mathworks.com/help/vision/ref/vision.peopledetector-system-object.html

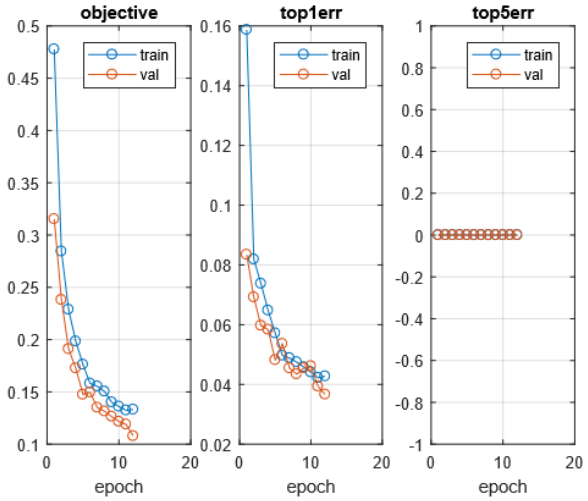
[6] Albanie. albanie/mcnSSD. Retrieved June 1, 2018 from https://github.com/albanie/mcnSSD

[7] Liu et al. 2016. SSD: Single Shot MultiBox Detector. (December 2016). Retrieved May 29, 2018 from https://arxiv.org/abs/1512.02325

Appendix

Training of the Matconvnet-vgg-m and Matconvnet-vgg-f





SSD Architecture [7]