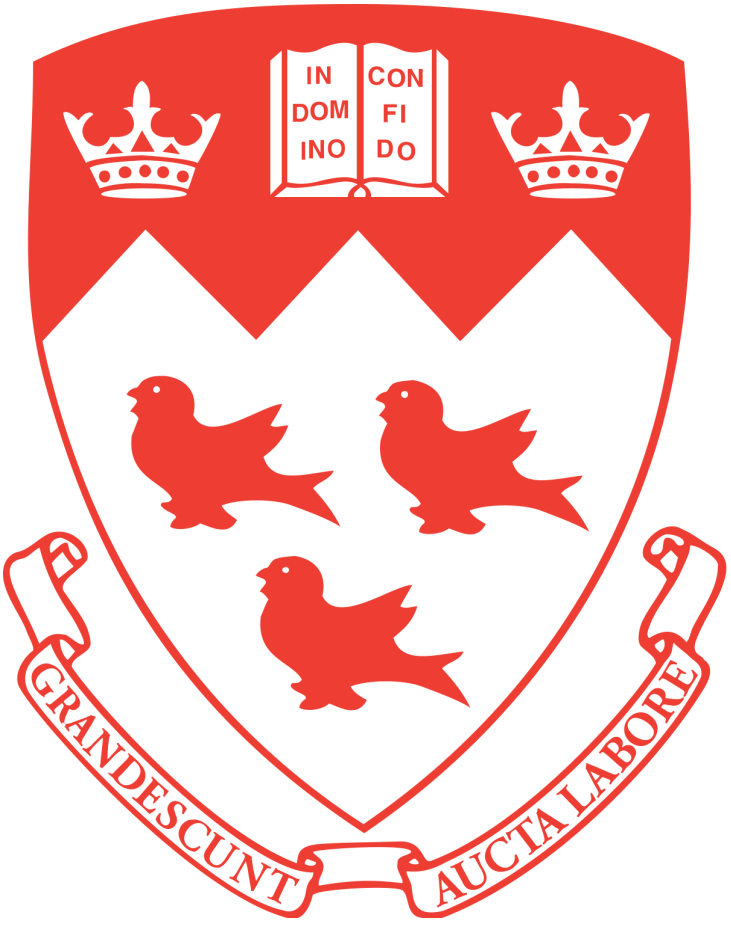
ECSE 415, Fall 2018
Introduction to Computer Vision
Final Project Report
Due date: December 2nd, 12AM

Team Members:
Aljulanda Al Abri: 260739353
Karl Doumar: 260733160
Lea Moukarzel: 260657230
Sadman Shahariar: 260715707
Omar Yamak: 260662616



2.2 Classification Grading

Description of the contents of the dataset (number of samples and image size, for each label) 5 points

The loading of the images was done by creating a dictionary with the label and image titles, and loaded the grey images using a for loop and appending the images along with its name and label. The approach in regards to the contents of the dataset was to use a subset of all the images, this was more feasible then trying all the images which would require a lot of resources.

Number of Sample=11,000 (1000 from each class of images)
Image Size= 50x50

Explanation of feature extraction method. 5 points

We used HoG for feature extraction. HoG computes magnitude-weighted edge orientation histograms to describe the appearances and shapes within an image by the distribution of the intensity gradients. These features allow us to detect corners and edges, that contain a lot of information about objects, as the gradients will be high in these regions. We chose to use the HoG descriptor because in addition to having an easy and fast implementation, it extracts features at all locations of the image rather than just around the keypoints (SIFT).

In order to extract HoG features, we first compute the gradients of the image (magnitude and orientation). Then, we divide the image into cells (4x4 cells in our case) and we compute the histogram of weighted gradient orientations. Orientations are used rather than magnitudes because they are not affected by intensity changes, but the gradient orientations are weighted by their magnitudes, and assigned to a bin depending on their values (1 out of 8 bins in our case). Finally, we normalize the histograms to their L2 distance with overlapping blocks and compute the HoGs for every block in the image.

These features are very commonly used in SVM classification and other machine learning based methods for object detection.

Explanation of how the feature extraction parameters were selected. 5 points

We used a cell size of 4x4, a block size of 4x4 and 8 bins for the extraction of our HoG features. Large cell sizes and small block sizes increase the miss rate. We had to find cell and block sizes to optimize the performance (i.e, minimize the miss rate). We found that a cell size of 4x4 and a block size of 4x4 is the best trade-off. We increased the number of bins to 8 because it decreases the number of false positives.

Description of cross-validation method. 10 points

Cross-validation is a method used to evaluate the performance of a classifier. [1] It consists of splitting the randomly shuffled data set into K folds, and performing a set of operations on each fold. For every fold, the group in question would be the test data set and the other groups would be the train data set. The classifier would then be fitted on the training data set then tested on the test data set. These steps are performed on each fold (in our case, 10 times) and the results are summarized by calculating the average accuracy, precision and recall across validations. This method was performed two times, once to evaluate the performance of our SVM classifier, and once to evaluate the performance of our kNN classifier. One of the advantages of this approach is that it matters less how the data gets divided. There is a relationship between variance and k, as k is increased the variance of the resulting estimate is

reduced. On the other hand, a disadvantage to this method is that the training algorithm has to be rerun from scratch k times, by consequence it takes k times the computations for each evaluation.

Evaluation of performance and interpretation of results (from Section 2.1). 15 points
- Average precision and recall across validations.

For our KNN classifier, we got an average precision of 72.06% and an average recall of 70.44%. For our SVM classifier, we got an average precision of 58.73% and an average recall of 58.73%.

- Are these values consistent with the accuracy?

The accuracy for KNN is 67% +/- 11%. The accuracy for SVM is 57 +/- 9%. Considering the standard deviation, the average precision, recall and accuracy are in the same range and the values are consistent.

- Are they more representative of the dataset?

The accuracy alone does not take into considerations the false positives nor the false negatives. It simply computes the number of correct predictions over the number of total predictions. Therefore, computing the precision and recall would bring about a better representation of the data set and a better evaluation of the classifier's performance.

- In what situations would you expect precision and recall to be a better reflection of model performance than accuracy?

We compute precision and recall using: $precision = \frac{TP}{TP + FP}$, $recall = \frac{TP}{TP + FN}$
where TP is the number of true positives, FP is the number of false positives, and FN is the number of false negatives.
We compute accuracy using: $accuracy = \frac{number\ of\ correct\ predictions}{number\ of\ total\ predictions}$

As mentioned before and as we can see from the formulas, accuracy doesn't take into consideration the false positives and false negatives that may be detected. So, precision would be a better reflection of model performance than accuracy when the number of false positives is high and recall would be a better reflection of model performance than accuracy when the number of false negatives is high.

- From the confusion matrix: are any of the classes difficult for your classifier? Discuss

We know that motion and occlusion may be potential hurdles to overcome when classifying images. From the confusion matrices for each classifier, we can see that the class that has the hardest time getting classified (most false positives) was the pedestrian class (for kNN) and the single unit truck class (for SVM). Indeed, when using KNN, some pedestrians would be classified as bicycles or backgrounds, increasing the number of false positive in the pedestrians predictions. kNN had the most amount of difficulty differentiating between backgrounds and pedestrians mispredicting 61 times The same thing happens with SVM and the single_unit truck class as many articulated_truck, bus and work_vans are classified as single_unit_trucks, increasing the number of false positives in the single_unit_truck class. But the most difficulty that the SVM classifier had was distinguishing between cars and pickup_trucks, mispredicting 49 times. We can also see that our classifiers had some difficulties distinguishing between pedestrians and bicycles and backgrounds, it's understandable as these are very small objects which is tough to distinguish from the background.

3.2 Localization and Classification Grading

Description of the contents of the dataset (e.g., number of samples and bounding box size for each label, contents, etc.) 5 points

The number of samples used was 11000, we used this amount for efficiency considering the amount of time and GB of RAM needed to test on a greater sample size. In addition, the bounding box is essentially the window used to search for objects within our dataset. The size for the box size that we used was (20x20),(50x50),(100x100), and (200x200). The size of the contents was 11000, we used 1000 from each category.

We also created another sub-dataset from the training images in order to create a background classifier which detects only background, we wanted this to be very precise so we used 10,000 background images and 7000 random images from other classes and labelled them as background and not background.

Description of localization method. 10 points

Localization entails being able to locate key features in one image that belong to a certain class, this class has distinguishing features that separate itself from the others. Each image has an action that can be put into a certain classification.

We started our localization by making a classifier that is able to recognize the background in a given window, we chose to use SVM classifier. We had initially set up both KNN and SVM Classifier, then after further testing we saw that SVM Classifier gave better results, so we used SVM. To train this classifier by taking 10,000 background images and 3000 random images from 10 other classes which we put in a folder named "abc" the folder contains two other folder, "background" containing the background images and "positives" which contains 3000 random images from the rest of the classes.

We then took the image and passed sliding windows of length 20, 50,100 and 200. Each of the window were then passed on to compute their HoG features then the classifier was allowed to predict if the window chosen was a background or not. The windows that were predicted "not backgrounds" were then passed on to a an array with the starting and ending coordinates of the window. This gave us about 50 to 60 windows for each object in the image. So then we passed on these windows to a K-Means algorithm with k=3, which returned only 3 boxes representing the centers of the overall window distributions. If there were less than 3 objects in the image then the centers returned would be next to each other. So then we took the windows returned by k-means and determined the euclidean distance between the centers of the windows. If the distance between the windows were 3 times the average window size then we merged the windows by taking the top left most coordinate and the bottom right most coordinate of the boxes. Then we passed on the boxes on to an array called final and plotted the results stored in final on the actual image passed.

Description of your cross-validation approach. 5 points

A subset of the localization training data was taken. Then localization was performed on that subset (user can choose the size of the subset). Then the subset is partitioned into 10 sets. In addition, the metrics (DICE) are calculated for each one of the 10 sets independently. The mean of the subsets were then considered as a performance indicator. We noticed that the performance indicator showed pretty consistent results above 25% till the last validation set, where our performance was bad bringing down the overall mean to around 20%

Evaluation of localization performance and interpretation of results (from Section 3.1). 25 points

- Compare the accuracy, prediction and recall of using your localizer + classifier vs. using classification data + classifier. Is there a difference, why/why not?

Using localizer + SVM classifier, we got an accuracy of 4.16%, a precision of 23.1% and a recall of 4.167%.

Using classification data + KNN classifier, we got an accuracy of 67% +/- 11%, a precision of 72.06% and a recall of 70.44%.

Using classification data + SVM classifier, we got an accuracy of 57 +/- 9%, a precision of 58.73% and a recall of 58.73%.

As seen in the numbers , our accuracy, predictions, and recall are relatively low. This is expected because our localizer had a mean DICE coefficient of less than 20% in best cases. When looking at the generated boxes in the images, we can see that there's at least one box that localized on or near an object. Many of these boxes do not cover the whole object so when we sent it through the classifier it will predict a wrong result. Since our non-deep learning classifier performed relatively well (70% accuracy), we expect it to perform well at least here if the localization was accurate.

- Should the 'background' label of the classifier be included when evaluating the performance of the localizer, why/why not?

We don't think the background label should be included. To begin with, we train our localizer to distinguish between background and not-background. Every box should localize around an object that is not a background. Now the box might be correctly around an object, but for some reason the classifier predicts a "background" for that object. This is why we this background shouldn't be included, because sometimes the localizer is correct, but because of the accuracy of the classifier we get a bad result to what should've been a good result.

4. Deep Learning Bonus

Schematic of architecture. 1 point

Uses a Convolution Neural Network using the TensorFlow framework for classification.

first we load the data using one hot encoding,

Secondly, we analyze the data

Next, we construct the deep neural network model using 3 convolutional layers:

1. 32-3 x 3 filters
2. 64-3 x 3 filters
3. 128-3 x 3 filters
4. Additionally, there are 3 max pooling layers of size 2x2 each

After the 28x28 images pass through these filters, our data is downsampled to 4x4. (from the max pooling layers).

then we train and test the model,

Finally, we make predictions of the test data.

Description of training. 2 points

We first define our training iterations (50), a learning rate(0.001) with a fixed batch size(128). Next we use placeholders: x is an input placeholder and y holds the label of the training images in a form of a matrix

We then define convolution and max pooling functions with certain weights and bias parameters after the data passes these layers, we flatten the output and connect the flattened neurons with each neuron in the next layer.

In the last layer, we have 11 neurons to classify 11 labels,

Finally we apply the adam optimizer

To train: we launch the graph and run the session to execute the variables and run the tensor. Then we use two for loops, one of training iterations and one for batch size. Finally we feed the placeholders x and y the actual data; they then return the loss and accuracy.

Evaluation of performance (as described in the relevant task's section). 1 point

Max Training accuracy: 99.219%

Max Testing accuracy: 70.545%

Since training accuracy is higher than testing accuracy, it seems like the model was over fitting. Looking at the graphs of loss vs epochs for training and testing (figure 1 & 2), the gap between training and testing indicate signs of overfitting. In order to improve performance it would be possible to introduce a dropout layer.

Secondly, using larger images could provide better results however it would increase computation time tremendously.

Increasing number of training iterations could also provide better results, however, from our data it seems that testing accuracy plateaus at 70.545%..

Description of validation. 3 points

To test the model we define two nodes: correct_predicition and accuracy, After each training iteration, we pass the 11000 test images(that it has not seen in the training phase) and labels to keep track of performance. This is done by feeding the placeholders (x & y) the unseen data. Plots for training and validation accuracy are shown in figure 1 and 2.

Comparison with the methods from Sections 2 and 3. 1 point

It was expected that the deep learning method produce much better results than the non deep learning methods of sections 2 & 3, however, due to overfitting the deep learning model, the validation accuracy of both methods seemed similar.

Appendices

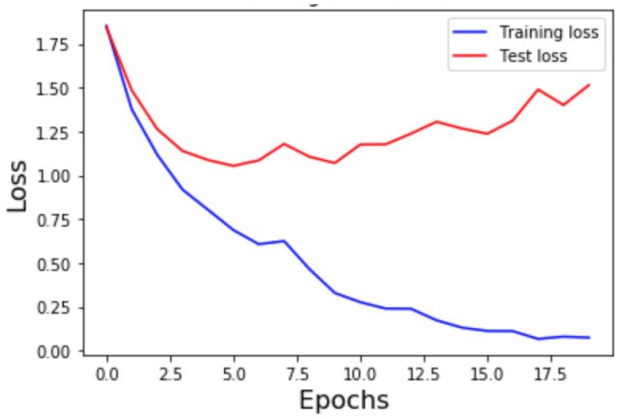


Figure 1:Training and test loss -Deep learning implementation

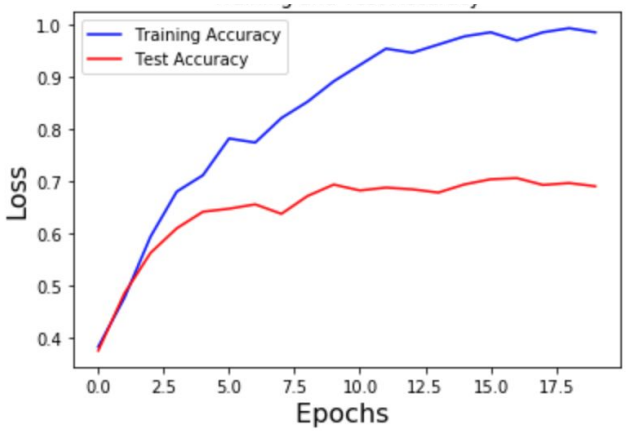


Figure 2: Training and test accuracy -Deep learning implementation

References

[1] Cross-validation: evaluating estimator performance,
https://scikit-learn.org/stable/modules/cross_validation.html

[2] Image Classification with Localization,
https://datalya.com/blog/machine-learning/image-classification-with-localization