A Comparison of Three Different Algorithms for Object Classification (Option A)

Bundhoo Shaheen, Shin Ku Young, Nie Raymond
University of Waterloo
Waterloo, Ontario, Canada
sbbundho@edu.uwaterloo.ca, ky2shin@edu.uwaterloo.ca, rhnie@edu.uwaterloo.ca

Abstract

This project is about learning various object classification algorithms and analyzing their performance and viability. The execution plan is as follows: after researching through various algorithms that optimize object detection and classification, notable algorithms will be compared on their relative strengths. From this project, each group member expects to become familiar with object detection and gain basic knowledge to implement an object detection algorithm. In addition, every member expects to understand the importance and application usages of object detection and learn the different algorithms mentioned in the research papers. To evaluate our results, the program will attempt to identify the message that is delivered by people who are using American Sign Language.

1 Introduction

Object detection refers to identifying the presence of particular objects in an image. In the last few decades, object detection has become a very important topic to explore due to its many applications in different industries. It is currently one the most researched topic in computer vision. Object detection is useful in different fields face detection, autonomous vehicles, anomaly detection, early stage cancer detection using MRI scans and more.

Object detection can be broken down into 4 main sub problems (Verschae, R. and Ruiz-del-Solar, J. (2015)):

1. Object classification: determining whether or not an object is present in an Image
2. Object localization: spotting an object in an image, that is, determining the location of the object instance in the image.
3. Object recognition: determining if a specific object is present in an image
4. View and pose estimation: determining the view and pose of the object

Please note that for the final project we did not review the following papers (as specified in the proposal): 1. *The PASCAL Visual Object Classes (VOC) Challenge* and 2. *Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation*. Instead we focused on the following paper, that was more relevant to our project and focus: *Handwritten Recognition Using SVM, KNN and Neural Network* by Hamid, N., and Sjarif, N.

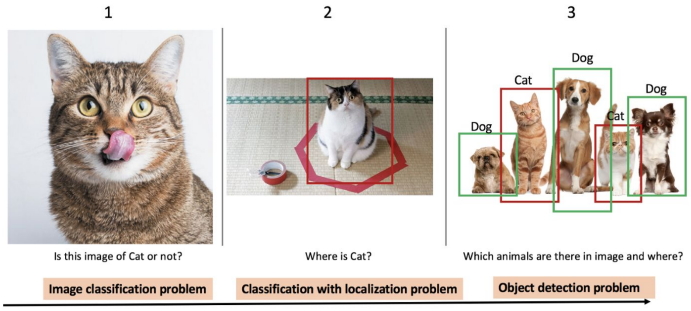

Figure 1: The first three subproblems in Object Detection

Due to time constraint, we decided to focus on the first subproblem, object classification by studying three different models which we have already been introduced to during the CS 489 course. Our objective for this project is to study these models in depth and look at their real-life applications, which is beyond the scope of CS 489.

2 Related Works

Initially, object detection was restricted to traditional image processing methods. One common tool that was used is OpenCV which has many computer vision algorithms implemented and is compatible with Python. As an general introductory approach described in the paper *Study on Object Detection using OpenCV Python*, objects can be classified and detected by finding features and attempting to match those features to regions in the source image. The general process described is to read in an image, understand the features that may be useful in detecting the desired objects and using a feature matching technique to detect these objects in the source image. Specifically images can be read in as coloured, or grayscaled. Depending on the nature of the dataset, it may not be important to keep the colour or alpha channels in which case a grayscale image would be more computationally efficient. The paper then describes that features that can be extracted using built in algorithms in OpenCV like Haar-like Feature extraction algorithm or blob detection or edge detection using Canny Edge filter. Lastly feature matching can be done using a Brute-Force matcher or a FLANN based matcher to try and detect these features in the source image.

Early works on object detection were based on template matching and image processing (Fischler, M. A., and Elschlager, R. (1973)). These methods do not involve any machine learning or deep learning models. Template matching method is a simple technique that defines a template, which is usually a shape or a sub-image, and uses the template to try and detect whether or not a region of the source image matches with the template. A method of measuring how well the template matches with a specific region of the source image is to calculate the correlation coefficient described with the formula,

$$p = \frac{\Sigma_x \Sigma_y (A_{xy} - A_{mean})(B_{xy} - B_{mean})}{\sqrt{\Sigma_x \Sigma_y (A_{xy} - A_{mean})^2 (\Sigma_x \Sigma_y (B_{xy} - B_{mean})^2}}$$

where A and B are matrices of the image, and x and y are the spatial coordinates of the image. A high correlation coefficient may represent a potential region of interest (ROI). While template matching is commonly used for detecting simple objects such as printed characters and numbers, this technique suffers when the source images lack shape consistency or have differences in viewing angles and orientation.

Although traditional image processings yield good results with good images, performance drops when images are noisy. Hence, better algorithms are required to detect and recognize objects in images. Years later, statistical models, such as SVMs, kNNs, Adaboost, Neural Networks, among others (Viola and Jones, 2001) were introduced in solving image detection problems. At a high level, most of these algorithms use somewhat the same technique as image processings but are more flexible to changes and hence noise in the data. For instance, the first layer of a convolutional neural network may focus on detecting edges of an image, similar to how Canny Edge filter of OpenCV works.

An SVM classifier can yield good results when used with a feature extractor. Histogram of Oriented Gradients (HOG) is a common feature descriptor that is often used followed by an SVM classifier to classify images. A feature descriptor is a representation of an image that simplifies the image by extracting useful information and throwing away extraneous information. In their paper, Dalal and Trigg, described different features of HOG descriptor such as fine-scale gradients, fine orientation binning, relatively coarse spatial binning, and high-quality local contrast normalization in overlapping descriptor blocks and how they all important to achieve good results. HOG divides an image into small spatial regions ("cells"). For each cell it computes a local 1-D histogram
of gradient directions or edge orientations over the pixels of the cell. These gradients are then saved in a feature vector, which is then fed into a conventional SVM for classification.

Over the last few years, there has been constant improvement in neural network architecture for object detection due to their high accuracy performance. Convolutional neural networks have been optimized to yield better results. Convolutional neural networks have been the gold standard in object classification and they are now used solve object localization and detection problems.

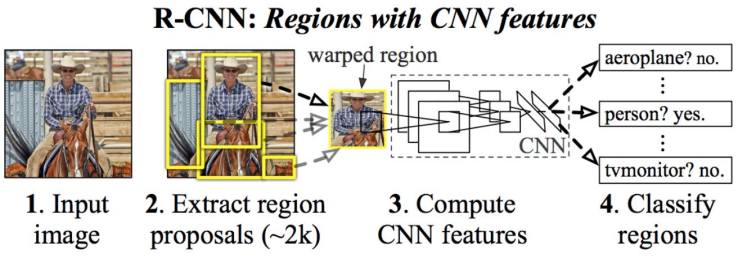

Figure 2: A simplified R-CNN design

An R-CNN extends to a traditional CNN by introducing region proposals in addition to feature extraction. Region proposals are areas of interest where feature extraction should occur. To extract these regions of interest, mechanism such as Selective Search are used. At a high level, Selective Search looks

at an image through windows of different sizes and for each size, it tries to group together adjacent pixels based on the texture, colour and intensity. The proposed regions of interest are then fed into a CNN for feature extraction followed by classification.

Region proposal generation uses SS or CPMC to convert an input image to extract candidate regions, which are regions across the image that are likely to contain objects. The candidate regions are described by CNN features before classification. To transfer a pre-trained CNN, such as the ImageNet ILSVRC, for object detection, the last few layers which are specific to the classification task are removed. This results in a "beheaded" CNN that outputs relatively generic features. The inputs are then fed into SVM training, which divides regions to positive and negative. The bounding boxes are refitted to detected objects by using CNN-based regressor. For further improvement, the initial probability of a region being foreground or background could be changed. For example, inputs with more background regions can be assigned with 75% probability of background region and 25% probability of the foreground region.

One notable disadvantage of R-CNN is the need to recompute the whole CNN from scratch for each evaluated region. This affects greatly on the model's performance, which could be improved. SPP-CNN addresses this issue by factoring the CNN in two parts, where ϕ_{cov} contains the convolutional layers pooling information from local regions, and ϕ_{fc}. Since convolutional layers encode local information, this can be selectively pooled to encode the appearance of an image subregion instead of the whole image. Then a function *g* can be derived to map the feature coordinates back to image coordinates.

It takes a huge amount of time to train an R-CNN. Moreover, the selective search algorithm is a fixed algorithm with no learning happening whatsoever at this stage. Fast R-CNN is the successor of the original R-CNN algorithm with several improvements. The fast R-CNN network works by first processing the whole image with several convolutions and max pooling layers to produce a convolution feature map. Next, we have a region of interest (RoI) pooling layer that extracts a feature vector from the feature map. This results in a improvement in terms of time complexity when compared to the R-CNN since previously we had a forward pass of the CNN for every proposed region. The feature vectors extracted from the RoI pooling layers are fed into multiple fully connected layers and two final output layers. The first is a softmax layer that estimates the class of the object detected, and the second is a layer that outputs four numbers that represent the bounding box position for the object in the source image. Fast R-CNN has several advantages. It has higher detection quality compared to R-CNN and SPPnet. Moreover, training a fast R-CNN is a single-stage process using a multi-task loss function and training can update all network layers.

Faster R-CNN was introduced by replacing selective search that is used to generate region proposals, with a region proposal network (RPN). Thus Faster R-CNN is composed of two networks: RPN and a network using these proposals to detect objects. RPN saves significant time in generating these regions it is designed to share most of its computation with the object detection network. Anchors, fixed-sized rectangles, are fed to the RPN to generate regions of interest. For each anchor, the RPN

predicts the probability of the anchor containing an object in general and refine the anchor by moving and resizing the anchor to the right position.

Initially, research in deep learning was limited since these models required high computational power. However, in the past few years, with improvements in hardware capabilities such as availability of cloud computing and GPUs, training a neural network has become relatively easy and fast. As a result, object detection has been vastly improved in terms of accuracy and performance due to the extensive research accomplished with neural networks.

Although neural networks have high potential of yielding good results, they might not always lead the performance board. The performance of the chosen model often depends on the complexity and amount of training data available. When classifying the MNIST handwritten digits, in their paper, Hamid and Sjarif has shown that simpler models Support Vector Machine (SVM) and K-Nearest Neighbour (kNN) have produced better results than an MLP Neural Network.

3 Proposed Works

For this project, we used Python, openCV, Scikit-learn, and keras to run read the data and build and run classification models. The dataset we used is the ASL American Sign Language dataset, that was downloaded from Kaggle. Each image in this dataset is 200 by 200 pixels (40000 pixels in total). The dataset contains a total of 87000 images. Out of these, 80% was used for training, while 20% was used to testing each model. The different models we studied are: Support Vector Machine (SVM), k-Nearest Neighbours (kNNs) and Convolutional Neural Network (CNN).

Support Vector Machine (SVM)

Support Vector Machine is a supervised machine learning technique that is used for classification, regression and outlier detection. SVM is less sensitive to the size and the number of the training dataset, unlike neural networks. Originally SVM was developed as the binary classifier where it assigns only two possible label, -1 and 1 when fed in a test example. The two different classes are separated by a hyperplane (called the decision boundary) based on the properties of the training samples.

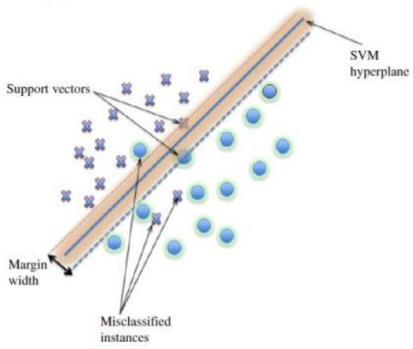

Figure 3: An SVM hyperplane separating two different classes of data

The SVM in scikit-learn support both dense and sparse sample vectors as input. Scikit-learn provides three different models to perform multiclass classification: SVC, NuSVC and LinearSVC. For this project, we used SVC to classify our dataset.

```
clf=svm.SVC(decision_function_shape='ovo')
clf.fit(x,y)
```

Figure 4: SVM classifier where, x is the training features, and y is the corresponding training labels.

K Nearest Neighbours (kNNs)

A kNN classifier is the simplest image classification algorithm. The model does not learn any specific features from the images, unlike SVM or CNN. Given a test data point, the algorithm computes a distance function between the feature vectors of test data point and all the data points it was trained on. The classifier then classifies the test data point by majority vote among the k closest examples (neighbours).

We trained a kNN classifier with different values of k, ranging from 1 to 10. We also computed the accuracy of the classifier and kept track the best k, that is, the value of k that produced the highest accuracy. We then re-trained a classifier with the best k, followed by evaluating the model using our test dataset.

Convolutional Neural Networks (CNN)

The third and last model that we implemented is a convolutional neural network. We built and trained a simple CNN using Python, and keras.

```
my_model = Sequential()
my_model.add(Conv2D(64, kernel_size=4, strides=1, activation='relu', input_shape=target_dims))
my_model.add(Conv2D(64, kernel_size=4, strides=2, activation='relu'))
my_model.add(Dropout(0.5))
my_model.add(Conv2D(128, kernel_size=4, strides=1, activation='relu'))
my_model.add(Conv2D(128, kernel_size=4, strides=2, activation='relu'))
my_model.add(Dropout(0.5))
my_model.add(Conv2D(256, kernel_size=4, strides=1, activation='relu'))
my_model.add(Conv2D(256, kernel_size=4, strides=2, activation='relu'))
my_model.add(Flatten())
my_model.add(Dropout(0.5))
my_model.add(Dense(512, activation='relu'))
my_model.add(Dense(n_classes, activation='softmax'))

my_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=["accuracy"])
```

Figure 4: Simple CNN written in keras

4 Results

As expected, our CNN model performed the best out of the three, with an accuracy of 88% followed by the kNN classifier (82%) and the SVM classifier (73%). The validation accuracy graph for the CNN is shown below.

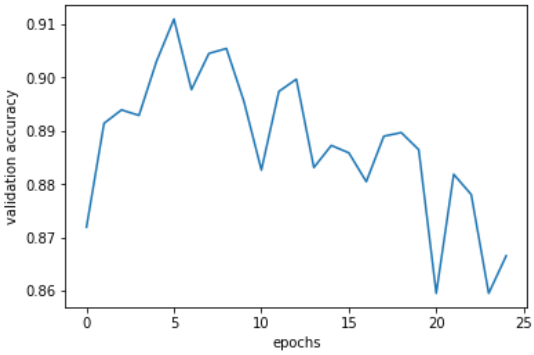

Figure 4: The graph above shows the validation accuracy performed on a test data set per number of epochs. The graph shows that after 10 epochs, the validation accuracy starts to decrease. This hints us that the model is overfitting and has a relatively fast learning rate.

Model Comparison

The pros and cons for each model are summarized in table below.

Model	Pros	Cons
kNN	- Easiest model to implement - kNN can be used to directly classify multiclass datasets	- Hard to find a good distance function for kNN - Testing is slow since the algorithm computes the distance between a test point to each single training data point
SVM	- Kernel-based framework, powerful and efficient - Does not require a lot of data to yield good results - Convex optimization, a global minimum solution can be found	- Cannot classify a multiclass dataset directly; need to build the model using several binary(two-class) SVM models
CNN	- More robust to noisy data compared to SVM and kNN - Parametric	- Computationally expensive: require powerful and expensive hardware - Large amount of training data needed

Table 1: Comparison between kNN, SVM and CNN as image classifier models.

All three algorithms performed pretty well on our chosen dataset. Currently, the convolutional neural network is the state of the art in object classification and detection.

5 Conclusion

Choosing an image classifier model highly depends on the dataset. If the dataset is simple and does not contain a lot of variation or noise, simple algorithms like the kNN or SVM classifier can yield fairly good results. However, for a more noisy dataset, a more complex model like a CNN is preferred. It is also important to note that due to the complexity of a CNN, the later requires a large amount of training data to learn important features.

Lastly, as part of this project, each team member had had the opportunity to learn about complex models like the fast R-CNN and Faster R-CNN, that builds on the knowledge gained from CS489. As there are still ongoing researches in this field, we believe that we still have a lot to learn and explore. The next step would be to implement a faster R-CNN for object recognition and detection. Another interesting research paper we came across is the Mask R-CNN paper by Girshick, Dollar, Gkioxari and He. A Mask R-CNN is extends the Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition.

6 References

1. Gupta, B., Chaube, A., Negi, A., and Goel, U. *Study on Object Detection using OpenCV Python*. 2017.
2. Hu, W., Gharuib, A. and Hafez, A. *Template Match Object Detection for Inertial Navigation Systems*. 2011.
3. Hamid, N., and Sjarif, N. *Handwritten Recognition Using SVM, KNN and Neural Network*. 2017.
4. Russakovsky, O., Deng, J., Su, Hu., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A., and Li, F. *ImageNet Large Scale Visual Recognition Challenge*. 2015.
5. Girshick, R. *Fast R-CNN*. 2015
6. Wang, R. *Edge Detection using Convolutional Neural Networks*. 2016.
7. Verschae, R. and Ruiz-del-Solar, J. *Object Detection: Current and Future Directions*. 2015.
8. Fischler, M. A., and Elschlager, R. *The representation and matching of pictorial structures.* 1973.
9. Viola, P., and Jones, M. *Rapid Object Detection Using A Boosted Cascade of Simple Features*. 2001.
10. Lenc, K., and Vedald, A. *R-CNN minus R*. 2015.
11. Ren, S., He, K., Girshick, R., and Sun, J. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.* 2016.
12. Dalal, N., and Triggs, B. *Histograms of Oriented Gradients for Human Detection*. 2005.
13. Everingham, M., Van Gool, L., Williams, C., Win, J., and Zisserman, A. *The PASCAL Visual Object Classes (VOC) Challenge.* 2010.