CSC-345 Big Data and Machine Learning Coursework:

Object Detection

By:
Jumaira Miller
983101

1 Introduction

In recent years, the quantity of image data has been growing at an exponential rate alongside the advancement in graphical technology. This is a result of cheaper production costs of cameras accompanying an ever-growing variety of devices (mobile phones, watches, doorbells, security cameras, intelligent eyewear). The growing amount of data allows for more accurate image analysis using Machine Learning. One such image analysis technique is Object Detection. The applications of object detection can range in a variety of computer vision tasks, including motion tracking, face detection and recognition, and image annotation. As object detection applications become more prevalent in technology, the associated problem of object classification by computers continues to be one of the significant areas of focus in the Machine Learning community.

The aim of this project is to explore different methods for object detection to resolve to an optimal solution. The methods will be implemented in Python on Jupyter Notebook; thus, the relevant required libraries and packages include TensorFlow, NumPy, Sci-Kit Learn, Sci-Kit Image, and Matplotlib. The exploration will be conducted on a subset of the CIFAR-10 dataset, where the training dataset contains 10,000 images, and the testing dataset contains 1,000 images for each of the ten object categories. The ten categories of objects are as follows: *Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck.*

The proposed classification algorithm for the solution to this problem is to use a Convolutional Neural Networks (CNN) to detect objects within images. The results from the CNN model will be compared against that of a Support Vector Machine (SVM). The CNN produced a constant training accuracy of 100% and a testing accuracy of 61.9%. The result of the SVM model scored 60% when applied to the testing data.

2 Methodology

2.1 Convolutional Neural Network

While other supervised models such as Neural networks have been around and improved on for 50 years, CNNs have demonstrated significant development regarding accuracy and efficiency. CNN's are far more capable at extracting an efficient set of features at a far lower memory and training demand [1]. For this reason, CNN was the chosen solution for this project.

The provided dataset to conduct this project, a subset of CIFAR-10, was in an unsuitable format and so some measure of preprocessing was needed. Initially, the dataset was in a format that had the dimensions first (height, width, and colour channel, respectively), followed by the image data. Both the SVM and CNN models were unable to handle such input, so we transposed the data; This restructured the data such that the image data appeared before the image dimensions, allowing for the classification model to understand the shape of our training and testing data. Furthermore, the labels were also converted to binary matrix vectors to further help the model understand the data we are trying to capture.

The processed training datasets can now be passed into our defined CNN model so that the features can be extracted. After this, we can then verify our trained model against test data.

Our proposed model can be divided into two stages (feature extraction and classification), with specific layers that contribute to these stages:

- **<u>Feature Extraction:</u>**

    - **Input Layer** - This convolutional layer takes our processed image data as input and extracts features from it. The image data, in this case, is represented as a height and width of 32 pixels, and 3 RGB colour channels. At this layer, a ReLU activation function is applied. The activation function works by taking in the calculated value (which is the product of a prediction and its weight) and checking if it is negative; In which case, the ReLU function will then re-evaluate the negative values to 0. The padding is applied to the layer to ensure the spatial dimension of the output is the same as the initial input.
    - **Convolutional Layers** - These layers are where the features were extracted. A kernel size of 3x3 was used whilst also applying the same ReLU activation to each convolution layer. In the proposed model, there is a total of five convolutional layers. At each level, the number of filters is doubled to extract more features at each level for a more well-defined solution.
    - **Pooling Layers -** These layers were utilized to reduce our feature maps and thus reducing the number of parameters considered, and as a result reducing the computational expense. These were used after the first two layers. We specifically use Max Pooling which makes sure that we get the most prevalent feature space from the feature maps.

- **Classification:**
    - **Fully Connected Layers -** These layers were used to begin classifying the images into the ten defined categories. A total of two layers were applied whilst also gradually reducing the number of filters we applied to the subsequent inputs. Again, a ReLU activation function was applied. We found that by adding these two layers, the accuracy of the model was increased.
    - **Output Layer** - Finally, we classified the image data into the ten defined categories. Thus there are only ten filters. Furthermore, we used a SoftMax activation function at this level, instead of a ReLU activation function. This is because the output produced can be considered a series of probabilities, which in this case, is the probability that a given image is part of a specific category.

Once the model was defined as described above, we compiled it using an Adam optimizer to improve the performance even further. When compiling the model, we also set input parameters for the metrics and loss data to categorical cross-entropy and categorical accuracy, respectively. These functions are beneficial in showing us the loss in accuracy when we have two or more labels. One hundred epochs are used to train the model whilst continuously validating the data using the testing dataset. The resulting accuracy and loss curves are visualized for analysis of the model's performance, which is evaluated in section 3 of this report.

2.2 Support Vector Machine

The second proposed model, and the model for which we are evaluating our proposed solution against, is the Support Vector Machine (SVM), which is a supervised learning model used for classification problems [2]. Before defining the model, some data preprocessing was required. First, we had to standardize both training and testing datasets [3]. This involved centring and scaling the data at an arbitrarily chosen midpoint. For the SVM, the preprocessing of the data had an additional step – extracting the histogram of orientated gradients (HOG) features. This is because, wherein the CNN model features are implicitly and continuously extracted by the filters in the convolutional layers, for SVM we need to extract them explicitly. For this, we utilize a function from the Sci-Kit Learn package – hog. This method returned 324 extracted features from each of the images in each dataset. This allows the model to solely focus on the classification element of the problem.

After the data preprocessing, the model was trained on the standardized datasets to then attempt to classify and derive an accuracy scored from the said data. Unlike with the CNN, there was no active validation attempt made on the SVM model. Overall, this model was relatively simple to implement as it is much less complicated than a CNN. On top of this, the model compiled and computed the results much faster.

3 Experimental Results

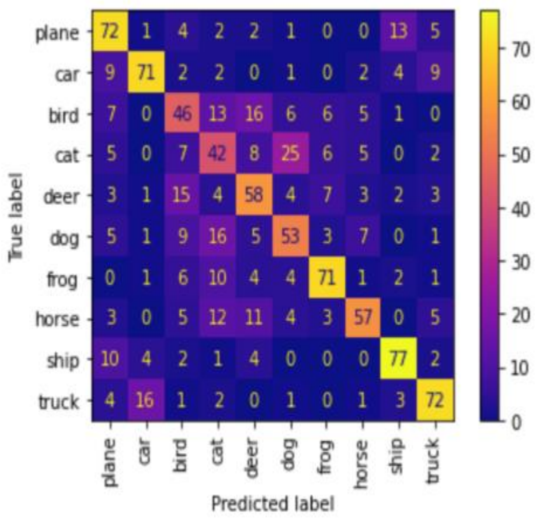
After training both the CNN and the SVM models, we were left with two sets of exciting results. CNN had a training accuracy of 100% and a testing accuracy of 61.90% after 100 epochs. As seen in figure 1 below, there was a steep climb in training accuracy. As a result, we see a very quick plateau and subsequent levelling out of the training accuracy. The accuracy climb was so steep that after about 20 epochs the accuracy was starting to plateau at around 95%. After 40 epochs, the accuracy was already at 100%. All subsequent epochs, of course, had an accuracy of 100%. This is an apparent case of overfitting, for which there are several reasons we could explore.

The first insight is that majority of the CNN layers' task was to feature extract. The model would only continue to take in more and more information with each epoch.

The next cause for overfitting is that there was no dropout layer. This is a crucial feature that is used as a method of regularization [4] to prevent or mitigate the issue of overfitting. Without this component, we keep fitting the training data without purposefully ignoring some aspects of it.

Finally, the dataset itself may have contributed to overfitting the data as it is a relatively small dataset. After a certain number of epochs, there is not much new information being taken in for the model to learn from. Moreover, as we are continuously validating the data with our test dataset, this effect is only exacerbated. In hindsight, it would have been prudent to show more consideration to these factors to prevent such a concerning training score.

Compared to the CNN model, the SVM reported a score of 60% accuracy. Even after a few runs, this value did not change. Whilst CNN boasts a higher accuracy score; the confusion matrix provides an interesting insight. The CNN had a more significant variance in accuracy scores per category, going as low as 42% for the *cat* category and reaching a maximum of 77% for the *ship* category. In contrast, the SVM had a more even distribution in accuracy scores with most values being within the 60-70% range. From this, we can infer there is some measure of inconsistency in performance for the CNN model, even though it is more powerful. These insights can be seen in confusion matrices for the SVM and CNN, shown below in figures 1 and 2.



True label \ Predicted label	plane	car	bird	cat	deer	dog	frog	horse	ship	truck
plane	72	1	4	2	2	1	0	0	13	5
car	9	71	2	2	0	1	0	2	4	9
bird	7	0	46	13	16	6	6	5	1	0
cat	5	0	7	42	8	25	6	5	0	2
deer	3	1	15	4	58	4	7	3	2	3
dog	5	1	9	16	5	53	3	7	0	1
frog	0	1	6	10	4	4	71	1	2	1
horse	3	0	5	12	11	4	3	57	0	5
ship	10	4	2	1	4	0	0	0	77	2
truck	4	16	1	2	0	1	0	1	3	72

```
32/32 [==============================] - 0s 9ms/step - loss: 6.0327 - accuracy: 0.6190
Test set loss: 6.03, test set accuracy: 61.90%
```

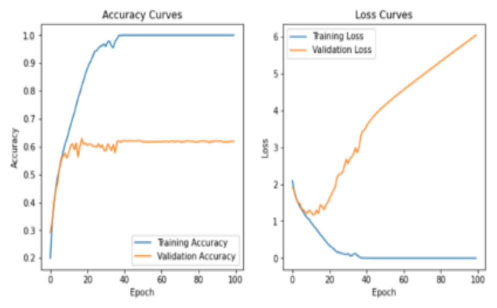
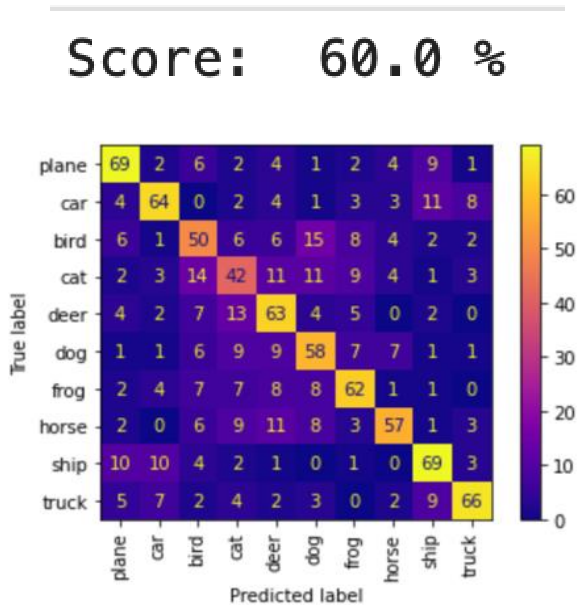


Figure 1: Results showing Confusion Matrix, Test Accuracy Score, and the Accuracy & Loss curves for the CNN model



```
Score:  60.0 %
```

True label \ Predicted label	plane	car	bird	cat	deer	dog	frog	horse	ship	truck
plane	69	2	6	2	4	1	2	4	9	1
car	4	64	0	2	4	1	3	3	11	8
bird	6	1	50	6	6	15	8	4	2	2
cat	2	3	14	42	11	11	9	4	1	3
deer	4	2	7	13	63	4	5	0	2	0
dog	1	1	6	9	9	58	7	7	1	1
frog	2	4	7	7	8	8	62	1	1	0
horse	2	0	6	9	11	8	3	57	1	3
ship	10	10	4	2	1	0	1	0	69	3
truck	5	7	2	4	2	3	0	2	9	66

Figure 2: Results showing the Test Accuracy Score and the Confusion Matrix for the SVM model

4 Conclusion

From the experimental results above, we can come to a reserved conclusion. The overall performance of CNN, in our case, is better than that of the SVM. However, we must also acknowledge that the overall distribution of accuracies, as shown in figure 2 above, was more uniform in SVM compared to CNN. As such, the overall performance of the SVM was more consistent. Furthermore, SVM computed results at a faster run-time for the given dataset, when compared to the time taken to compile the CNN model. The reality is, however, that the SVM will likely not scale up as well as the CNN. Even with the limited dataset size, the CNN model still provided a higher maximum and the minimum accuracy score when compared to the SVM. Whilst unstable, we got a high yield score. With even more testing, applied to the full CIFAR-10 dataset, we would still likely get an even higher score for CNN of over 70% [5].

In conclusion, the above report highlights the interesting results seen by both the SVM and the CNN model. One cannot concretely say whether CNN truly is the better solution. However, purely on a performance basis, the CNN did outperform the SVM (although marginally), and thus we propose this to be the model solution to our classification problem.

To demonstrate the power of the CNN more, we could have mitigated the overfitting issue, and provide more significant variance in our data, such as video frames or different kinds of augmented images. This would be a measurable start regarding future bodies of work.

References

[1] R. K. C. R. S. Hijazi, "Using Convolutional Neural Networks for Image Recognition," *Cadence*, pp. 7-8, 2015.

[2] R. Pupale, "Support Vector Machines(SVM) — An Overview," Towards Data Science, June 16 2018. [Online].] Available: https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989. [Accessed December 2020].

[3] SciKit Learn, "sklearn.preprocessing.StandardScaler," N.A.. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn.preprocessing.StandardScaler.transform. [Accessed December 2020].

[4] J. Brownlee, "A Gentle Introduction to Dropout for Regularizing Deep Neural Networks," Machine Learning Mastery, 3 December 2018. [Online]. Available: https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/. [Accessed December 2020].

[5] TensorFlow, "Convolutional Neural Network (CNN)," 10 September 2020. [Online]. Available: https://www.tensorflow.org/tutorials/images/cnn. [Accessed December 2020].