CSCM45 Big Data and Machine Learning

Coursework 2: Object Recognition

**Introduction**

We have been instructed to use an image dataset set for this experiment, which is a subset of CIFAR-10 [3]. The aim, of the experiment, is to try and find out what machine learning technique or combination of techniques, creates the best prediction accuracies. The following techniques we used are principle component analysis (PCA), linear discriminant analysis (LDA), support vector machine (SVM) and a neural network (NN). We have a benchmark of 44.68% provided. We will also feed the data directly into a NN for a direct comparison along with SVM. Another metric that we used for comparison is the time it takes to compute, as we can assume that using a form of dimension reduction will lower the accuracy. However, the possible faster time it takes to train the model might be a justifiable trade-off. With experimenting with different optimisers and hidden layers within the NN, the SVM method, using a poly kernel, came out the best with an average of 55.90% accuracy which took 47 seconds while PCA into a NN come out the worst with an average of 21.89%. We found that using both PCA and LDA together created the worst results.

**Method**

**Extracting the data from the Dataset**

We first need to extract the data from the two provided 4D arrays, one of size 32x32x3x10000, the training data, and the other 32,32,3,1000, the testing data. In order to be able to effectively use the data a provided function, from the SKLearn library, SKImage was used to extra the features. The features become a numpy array shaped 324, 10000 with the same done to the testing set, which created a numpy array shaped 324, 1000.

**Initial Benchmarks**

We first fed the extracted features into a NN. Doing this was to provide an initial benchmark for comparison. The NN settings were changed using a different number of layers, optimisers and activations to try and find the best outcome for the data. Feeding the data directly into SVM was also done. Again changing the kernel from linear, sigmoid and poly as well as changing the penalty of error.

**Dimensionality Reduction Techniques**

We will use both PCA and LDA; these are ranked highly as some of the best tools for this purpose [5]. Dimensionality reduction will help with the curse of dimensionality [4,6]. First by themselves, fitting and then testing the data. We will then feed the reduced data then into a NN and into SVM, to see what generates the best results. Both forms of dimensionality reduction will also be used together to see what kind of impact this has on the outcomes. For PCA it was decided to only use 100 components (fig 1).

**Supervised learning Techniques**

As we have provided labels for the data, we will use supervised learning to do the predicting. As we now have initial benchmarks, we will use a series of combinations to see what creates the best results. The experiments are PCA into NN; PCA into SVM; LDA into NN; LDA into SVM; PCA into LDA into NN; PCA into LDA into SVM. In order to inform where the training might get stuck, we will use a confusion matrix.

**Optimisation**

To find out the best outcomes, we tried using different optimisers for the NN and SVM. Optimisations included changing the activation on the hidden layers, the number of hidden layers as well as the number of nodes on each layer. As well with the SVM, multiple kernels were changed to see what generated the best results.

**Results**

When feeding the data directly into the NN, the best results found was using just one hidden layer with 200 nodes. This method, on average, took about 126 seconds to complete with a 52% accuracy (fig 2). However, the data directly into SVM took 47s and had an accuracy of 55.90% (fig 3).

When we fed PCA's data into the NN, at the NN optimised settings, one layer and 200 nodes, it took 280s and only had an accuracy of 21.80% (fig 4). The results show that not only is using PCA with a NN less accurate but it also takes longer to compute it. However, feeding the PCA into SVM did create faster speeds, 17s, it also produces worse prediction results, 18.70% (fig 5).

LDA into the NN optimised settings generated results around the 37% mark and took 105s (fig 6). As the results show, this is defiantly quick that just the NN but quite a considerable drop off with accuracy. However, when we optimised the NN for the LDA data, one layer of 4 nodes, the results were better. 54% accuracy and 100s. So using this method would defiantly be useful, but the model did struggle with predicting cats and deer (fig 7). While feeding it through the SVM, the speed was incredible. It was completed in 1.5s and had 35.90% accuracy. So the accuracy was not as good, but for the speed, it could be a justifiable the choice (fig 8). For PCA into LDA into NN or SVM, we found the results were quite poor but feeding the data into the NN done better than using SVM. NN had 29% and 118s while SVM had 25.20% and took 1.6s.

Discussion.

[discussion on why the results might have been this way.]

**Conclusion**

After using a range of different dimensionality techniques and then feeding them into a NN or an SVM, we found some great insights. The processes included: PCA into a NN and SVM; LDA into a NN and SVM; PCA into LDA into NN and PCA into LDA into SVM. As we expected, the dimensionality reduction did lower accuracy, but the speed gains were not enough to warrant using them. Nevertheless, the method that does the best, in terms of accuracy and speed, was using just the SVM. A close contender behind is using LDA into a NN with the NN with it optimised for the LDA data. Both these methods not only beat the benchmark score, but it also was the quickest. The quickest again being the SVM.

An area that this experiment didn’t explore in detail was the effects of epoch levels within the NN and the penalty of error within the SVM algorithm. Although we did initially change these settings, the majority of the experiment looked at just reducing the dimensionality of the data and how many layers with nodes the NN was most effective at. In future this would need more of a focus on. Also changing the number of components in the LDA and PCA. This was changed, but not as in depth as the others.

To summarise, even through optimising, there was no best one fits all. In order to get good results with LDA into the NN, the best results came from having one layer and four nodes, but this would give poor results for the data directly into a NN. Directly into a NN tended to do better when again, there were fewer layers but more nodes within the layers. However, these methods alone are not suitable for this task. Research suggests that using a Convolutional Neural Network (CNN) would be a better NN for completing this task [1,2]. Also, most research suggests using PCA for this task along with a CNN, so more research needs to be done into this as PCA was the weakest option within these experiments.

**References**

[1] Chauhan N. Introduction to ANN.<https://towardsdatascience.com/introduction-to-artificial-neural-networks-ann-1aea15775ef9> (accessed: 04/12/19)

[2] Chansung P. CIFAR-10 Image Classification in TensorFlow, <https://towardsdatascience.com/cifar-10-image-classification-in-tensorflow-5b501f7dc77c> (accessed: 03/12/19).

[3] Krizhevsky A, Nair V, and Hinton G, CIFAR-10 subset <https://www.cs.toronto.edu/~kriz/cifar.html>

[4] Quora. Is PCA supervised or unsupervised? <https://www.quora.com/Is-PCA-supervised-or-unsupervised> (accessed: 04/12/19)

[5] Silipo R, Widmann M. 3 New Techniques for Data-Dimensionality Reduction in Machine Learning. <https://thenewstack.io/3-new-techniques-for-data-dimensionality-reduction-in-machine-learning/> (accessed: 04/12/19)

[6] Wikipedia. Curse of dimensionality<https://en.wikipedia.org/wiki/Curse_of_dimensionality> (accessed: 04/12/1987)

**Appendix**

Fig 1: Fig 2: Fig 3:

A picture containing screenshot

Description automatically generated A screenshot of a cell phone

Description automatically generated A close up of a piece of paper

Description automatically generated

Fig 4: Fig 5: Fig 6:

A picture containing electronics

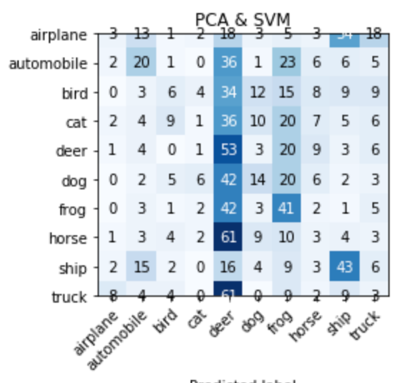
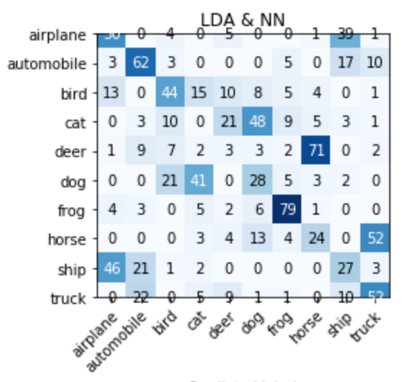
Description automatically generated  

Fig 7: Fig 8:

