Rachel Newton-John - 2006652

# CSC-345 Coursework

## Object Recognition Using Machine Learning Methods

## Table of Contents

# Introduction

The CIFAR100 dataset is a set of 60,000 32x32 labelled images, belonging to a set of 100 classes and 20 superclasses, and split into a set of 50,000 training images and 10,000 test images. It is commonly used for machine learning and neural network experiments in object recognition and image classification. This report compares the use of support vector machines and convolutional neural networks to classify these images, highlights the advantages in both accuracy and practicality that a CNN has in this application, and shows that an SVM is not an appropriate tool for image classification beyond very clearly delineated and distinct images.

# Method

## Support Vector Machine

The first method is the use of a support vector machine fit to an extracted set of histogram oriented gradients from the training dataset.

The SKLearn hog feature extraction is first used to generate a flattened array of feature vectors for the training dataset.

The extracted feature vectors are very large, occupying a 50,000 by 15,876 array for the training data and taking up approximately 6GB of space. Principal Component Analysis is used on the extracted feature vectors to transform them to a set of dimensions across which the variance of the data is maximized, in descending order of significance. The dataset can then be made more manageable by discarding the least significant dimensions of the data with a threshold defined by how much of the original data's variance it is acceptable to lose. The amount of data it is acceptable to discard and it's impact on the accuracy of the SVM is one thing that will be explored.

There is a practical limitation to this approach. The extracted HOG feature dataset is so large that in this instance, the PCA cannot be trained on the whole set within the memory constraints of the system it is being trained on; instead, it is fit to a subset consisting of 45,000 of the initial 50,000 images.

A support vector machine is then trained on the dimensionally reduced data, to classify the samples to the provided categories. The support vector machine is made using the sklearn.linear_model.SGDClassifier class rather than the sklearn.svm.SVC class due to it's improved scalability in training on large datasets. The classifier is given an upper limit of 5000 epochs when training on the data – it can take enough time to run this many iterations that it is a significant hinderance to the practical usability of this approach, so ability to converge within this limit will be a criterion on which it is evaluated.

## Convolutional Neural Network

The second method being used to classify the images is a convolutional neural network, implemented using the tensorflow keras libraries.

Other than reordering the axis of the array to fit the format the network expects, the training data is not pre-processed before the network is trained on it.

The network is comprised of three 2D convolutional layers followed by two dense neuron layers.

The convolutional layers have 400, 200, and 100 neurons from from input to output. The final convolutional layer uses a sigmoid activation function to aid in classification when outputting to the dense layers for final mapping.

Each of the convolutional layers are followed by a dropout layer and a max pooling layer. The dropout layer is intended to help limit the network's overfitting to the training material, by deactivating a random subset of connections between the adjacent layers on every epoch of training.

The 200 neuron convolutional hidden layer also has l2 regularization applied to it to further reduce overfitting, with a regularization factor of 0.0014. This factor was arrived at experimentally by observing it's impact on the output curves of the model. The decision to not include a regularization term on other layers was also arrived at by observing that their presence seemed to make the output less accurate and stable which having minimal impact on the difference between training and testing loss.

The pooling layers are intended to make feature detection less sensitive to orientation and reduce computational complexity by downsampling the data.

The convolutional layers are then flattened and followed by a pair of dense layers with 200 and 100 neurons to classify the output and map it to the training categories.

# Results

The performance of these models will be evaluated against a benchmark of 39.43% across the 20 superclasses, and 24.49% against the 100 finer classes.

To illustrate the accuracy and inaccuracy of the models, one-vs-all confusion matrices are generated for every class, and an average of these results is presented for each label set on each model in addition to interesting regions of the full confusion matrices.

## Support Vector Machine

Multiple SVM's were trained on the PCA transformed data, with dimensionality preserving from 62% to 98% of the original data's variance, in 4% increments.

With the superclass labels, the support vector machine achieved an average accuracy of 21.7% with 62% to 90% of the original data's variance, with no appreciable difference in accuracy from different amounts of variance. With 94% of the original dataset's variance accuracy improved to 23.2%, however with 98% of the original data's variance the SVM failed to converge within the 5000 iteration limit and accuracy dropped to 18%. For testing, the dimensionality reduction preserving 94% of the original variance was used.

With the 100 finer labels, the SVM failed to converge within the 5000 iteration limit at any dimensionality. For testing, 94% of the original data's variance was preserved and the SVM was trained up to the 5000 iteration limit.

This training took 1.5 hours to complete (5336 seconds). Though not directly a metric that the model is being assessed on in this report, the sheer length of time this took to complete compared to the CNN is noteworthy enough to warrant inclusion.

The SVM classifier has a sensitivity and specificity of 0.119 and 0.991 on the fine labels and 0.233 and 0.960 on the superclasses.

| **Average One-Vs-All Confusion Matrix** | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Fine Classes** | | | | **Superclasses** | | | |
| | | Predicted Value | | | | Predicted value | |
| | | Positive | Negative | | | Positive | Negative |
| Actual Value | Positive | 11.9 | 88.1 | Actual value | Positive | 116.6 | 383.4 |
| | Negative | 88.1 | 9811.9 | | Negative | 383.4 | 9116.6 |

Table 1: Average One-vs-All Confusion Matrix values

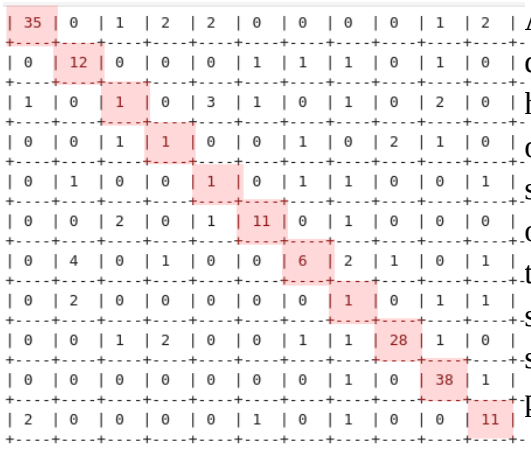


*Figure 1: A region of the fine category confusion matrix for the SVM*

A small area of the confusion matrix for the fine classes is displayed in figure 1 on the left, with the true positives highlighted in red, to illustrate a curious behaviour. The rate of positives is not uniform, but rather there are specific samples that it is very good at classifying accurately, while others are no better than random chance. Furthermore, taking one-vs-all matrices for every class in the model, the sensitivity of individual classes reaches as high as 0.44 in some instances. This behaviour is not observed as prominently in the coarse dataset, though it is still present.

## Convolutional Neural Network

The CNN achieved an accuracy of 47.4% across the 20 superclasses and 36.0% on the 100 finer classes. This is significantly higher than the benchmark results and over double the accuracy of the SVM, with a sensitivity and specificity of 0.360 and 0.994 on the fine classes, and 0.474 and 0.972 on the superclasses.

With training of these models being hardware accelerated, training either of these models consistently took less than 9 minutes, over an order of magnitude faster than the SVM trained on fine labels. The time taken also did not scale with the size of the set of labels it was being trained against, unline the SVM.

| **Average Confusion Matrix** | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Fine Classes** | | | | **Superclasses** | | | |
| | | Predicted Value | | | | Predicted value | |
| | | Positive | Negative | | | Positive | Negative |
| Actual Value | Positive | 35.99 | 64.01 | Actual value | Positive | 237.2 | 262.8 |
| | Negative | 64.01 | 9835.99 | | Negative | 262.8 | 9237.3 |

Table 2: Average One-vs-All Confusion Matrix Values

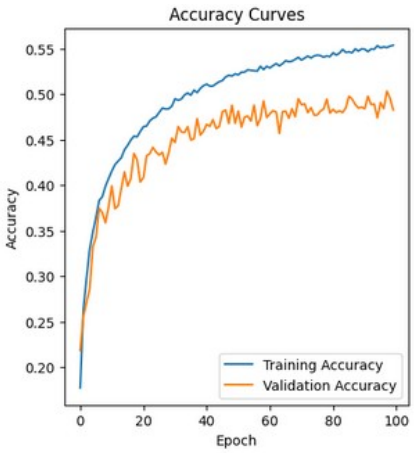


*Figure 2: Training and validation accuracy curves over time for the CNN on the coarse labels*

Despite the presence of regularization measures such as dropout layers and l2 regularization, as well as pooling between layers, the model still overfits to the training data to a degree, as shown in the figure 2 to the left. This could likely be reduced with further experimentation with the model's structure, and with hyperparameter optimization.

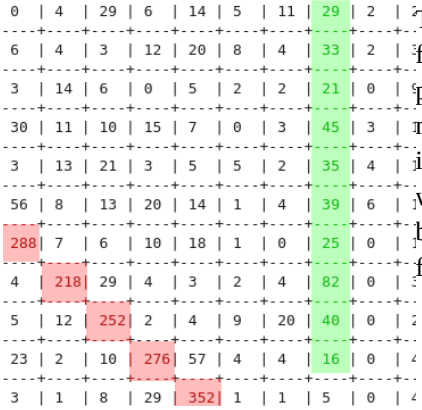

*Figure 3: A region of the coarse category confusion matrix for the CNN*

The region of the confusion matrix for the CNN with the coarse labels shown in figure 3 has the true positives highlighted in red, and highlighted in green is a peculiar section showing that the CNN is very prone to assigning images from most categories to the 14th category, non-insect invertibrates. This category includes crabs, spiders, and worms, all of which are vastly different animals with completely dissimilar sets of features. An intuitive line of reasoning is that because of this, this has become treated as a sort of “other” category, which false positives are biased towards.

*Figure 4: A region of the CNN confusion matrix for the fine labels*

The confusion matrix for the fine labels doesn’t show the same phenomenon. There are specific misclassifications which occur frequently, shown in figure 4, but no categories which draw a disproportionate amount of false positives in the same way. This can also be verified by checking the one vs all confusion matrices for the separate categories.

# Conclusion

Object recognition does not seem to be an appropriate or effective application of support vector machines beyond rudimentary, broad classification. The implementation explored in this report has several shortcomings which could improve it’s effectiveness. Performing PCA on the full training data rather than the truncated set could result in a more effective dimensionality reduction, however this would have higher hardware requirements. Using an SVM with a sigmoid kernel rather than a linear SVM would likely result in more effective classification, however, the SVM implementation with a sigmoid function in sklearn is computationally far more expensive than the SGDclassifier and scales much more dramatically with the size of the dataset it is trained on, making it’s use very impractical.

This consideration of what tradeoffs are and are not acceptable with SVM’s is moot when convolutional neural networks would seem to be far more effective, with lower memory requirements, and the ability to be accelerated using off the shelf hardware to the extent that the network in this report was able to be trained in an order of magnitude less time than the SVM’s used without any compromises being made to the training data due to memory limitations. The CNN also showed significant sensitivity to the parameters used in it’s creation; while the parameters used in this network are functional and present a classifier that is accurate in comparison to the benchmark result, further optimization is likely possible with little to no additional computational expense.

Future work should evaluate methods of reducing CNN overfitting, such as image preprocessing to feed a network rotations and mirrored versions of the image set in addition to the images as provided, as well as assessing different methods of reducing overfitting within the model itself through structural changes and hyperparameter optimization.

Based on the line of reasoning presented about the rate of false positives in category 14, another area to explore would be whether the inclusion of an explicit “other” category in datasets could result in more accurate classification.