Swansea University

Big Data and Machine Learning

Coursework 1

VERNON T. (1907240)
12-12-2023

Table of Contents

Introduction

Overview

The problem at hand is to apply a machine learning algorithm to classify low resolution 32x32 colour images into object categories. There are two sets of labels provided for training, and these are coarse (20 categories) and fine (100 categories). To limit the scope of this problem I will only focus on one of these label sets, that being coarse labels. I chose this because I do not have access to a high level of computing power, which would be needed for training a model to recognise finer detail at a similar level of accuracy [1]. The two solutions that I have chosen to compare are the use convolutional neural networks, and then the use of feature extraction using HOG (Histogram Oriented Gradients) alongside a fully connected neural network.

Method

I am using Visual Studio Code with .ipynb support for my IDE. For the kernel, I've opted to use Anaconda for ease of library installation, with Python 3.10.13. The following libraries are imported:

- NumPy – for working with numerical data.
- Matplotlib – for showing images and data visualisation.
- TensorFlow – for the creation of neural network models
    - The tensorflow-directml plugin was also used speed up training by leveraging the graphical processing power of AMD 7640S APU.
- Scikit-Image – for the HOG feature extraction

Convolutional Neural Networks

This is the step-by-step process that I took to train the convolutional neural network model.

- **Load the training images, test images, and csv containing the label data** as NumPy arrays.
- **Transpose the training data** to move the image index as the first item in the array.
    - This changes the shape of the data from (32, 32, 3, 50000) to (50000, 32, 32, 3).
- **Flatten the training data** by first reshaping it to (50000, 3072).
- **Normalise the training data** by dividing it by 255.
- **Reshape the training data** to the correct format to be taken as input into the model (50000, 32, 32, 3).
- **Build the model layers using TensorFlow.**
    - This required a lot of experimentation with the number of convolution, dense and max pooling layers to maximise the models plateau accuracy. I ended up settling on the following:

```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.MaxPooling2D((2, 2)))
model.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu'))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.MaxPooling2D((2, 2)))
model.add(tf.keras.layers.Conv2D(256, (3, 3), activation='relu'))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.MaxPooling2D((2, 2)))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(512, activation='relu'))
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dense(20, activation='softmax'))
```

- **Compile the model.**

- I opted for the Adam optimiser because it seems like a great all-rounder optimiser for computer vision [2]. I also used the sparse-categorical-crossentropy loss function because it's great for classification models [3].

- **Train the model.**
  - I opted for 24 epochs, as within my testing the model's accuracy and loss stabilised well before reaching this number.

HOG and Fully Connected Neural Networks

This is the step by step process that I took to train the fully connected neural network with HOG feature extraction. I did this after the previous steps for the CNN, so the data is already partially pre-processed.

- **Go through each image** in the training dataset and **generate its equivalent HOG feature** data and add it to a new array.
  - I used 2,2 for pixels per cell, and 1,1 for cells per block.
- **Normalise the feature data** with the standard scalar function from sklearn.preprocessing.
- Create a fully connected sequential neural network model with the following layers:

```
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(512, activation='relu'))
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.Dense(20, activation='softmax'))
```

- **Compile the model.**
  - Using the Adam optimiser, and the sparse-categorical-crossentropy loss function, just like before.
- **Train the model** with the hog features array, and the coarse labels training array.
  - With 24 epochs and a batch size of 64, to match the CNN settings.

Results

Convolutional Neural Networks

This is the graph documenting the accuracy and loss values during the training process for every consecutive epoch. The model reaches a final accuracy level of 0.9625 after 24 epochs (figure 2).

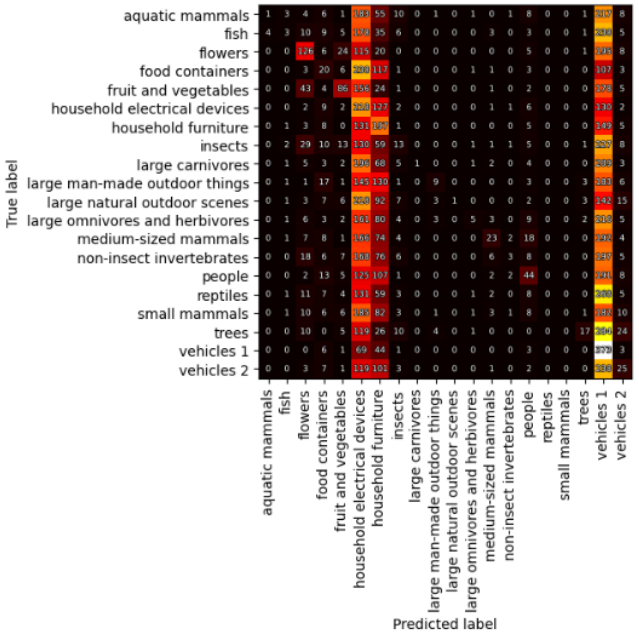

Figure 1

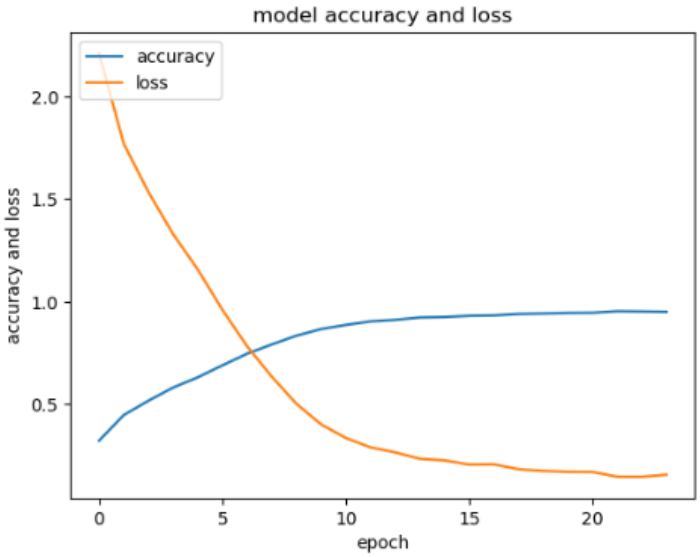

Figure 2

Looking at the confusion matrix (figure 1), we can see that the model performs very poorly in almost all scenarios. The model mistakenly labels most of the objects as under “household electrical devices” or “vehicles 1”. Using these values in the confusion matrix we can calculate per-label accuracy (figure 3). Vehicles 1 at 74.6% is not bad, but with household electrical devices at 43.6% and the majority of the rest well below 15%, this highlights just how poor the model performs. The average of all of these is just 11.7%.

```
aquatic mammals : 0.002
fish : 0.006
flowers : 0.252
food containers : 0.04
fruit and vegetables : 0.172
household electrical devices : 0.436
household furniture : 0.394
insects : 0.026
large carnivores : 0.002
large man-made outdoor things : 0.018
large natural outdoor scenes : 0.002
large omnivores and herbivores : 0.01
medium-sized mammals : 0.046
non-insect invertebrates : 0.006
people : 0.088
reptiles : 0.0
small mammals : 0.0
trees : 0.034
vehicles 1 : 0.746
vehicles 2 : 0.05
```

Figure 3

HOG and Fully Connected Neural Networks

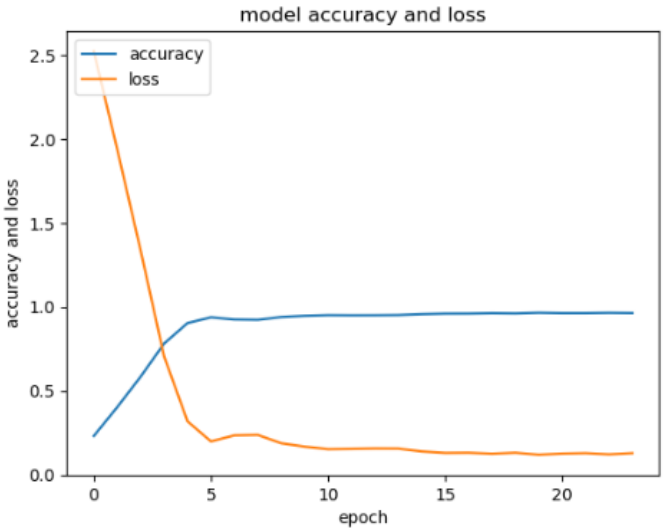

Figure 4

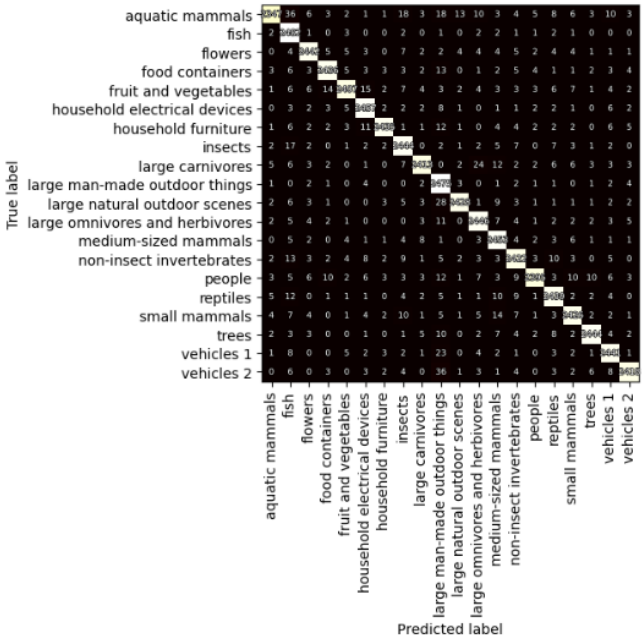

Figure 5

Here are the accuracy/loss and confusion matrix plots (figure 4). The model ended with an overall accuracy of 0.9629. We can see that the model was very fast to train and reach an equilibrium, arguably by just 10 epochs. This is many less epochs than the CNN took to reach a similar level of overall accuracy. Training the FCNN was also much faster, as each epoch only took 5 seconds (as opposed to FCNN at 13 seconds). Looking at the confusion matrix (figure 5), we can see that the model is very, very accurate for all object classes, with a beautiful straight white line down the diagonal, signalling that for each object, the model predicts its corresponding true label with almost perfect accuracy. This is shown if we calculate the accuracy for each object class (figure 6). As you can see, the model predicts every single object class correctly with at least 93.1%, averaging out at 97.3% overall, a stark contrast to the poultry 11.7% of the CNN model.

```
aquatic mammals : 0.9388
fish : 0.9928
flowers : 0.9768
food containers : 0.9744
fruit and vegetables : 0.9628
household electrical devices : 0.9828
household furniture : 0.974
insects : 0.9776
large carnivores : 0.9652
large man-made outdoor things : 0.99
large natural outdoor scenes : 0.9712
large omnivores and herbivores : 0.9784
medium-sized mammals : 0.9812
non-insect invertebrates : 0.9688
people : 0.958
reptiles : 0.9756
small mammals : 0.9704
trees : 0.9776
vehicles 1 : 0.9764
vehicles 2 : 0.9672
```

Figure 6

Conclusion

I can conclude from my results that the use of HOG feature extraction to train fully connected neural network for object recognition is far more effective than using the original colour images to train a convolutional neural network. This is easily proven by comparing the overall accuracies of either of the models, with CNN at 11.7%, and HOG at 97.3%, and can be further emphasised by visualising the confusion accuracies of each category for each model in a boxplot (figure 7). There is a chance that the convolutional neural network result could be the result of overfitting, but even after training for far less epochs, the model still showed a very poor testing accuracy.

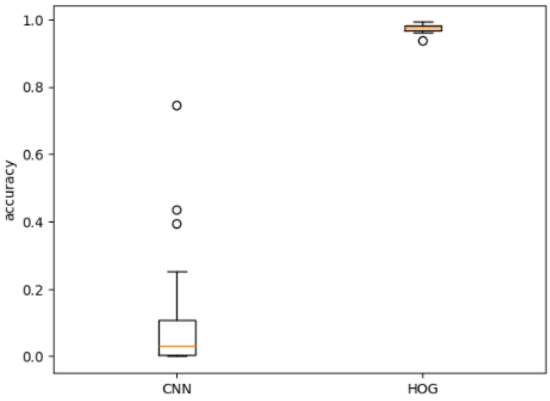

Figure 7

Bibliography

[1] K. G. K. L. G. F. M. Neil C. Thompson, "MIT INITIATIVE ON THE DIGITAL ECONOMY RESEARCH BRIEF," 2020. [Online]. Available: https://ide.mit.edu/wp-content/uploads/2020/09/RBN.Thompson.pdf. [Accessed 8 December 2023].

[2] J. Brownlee, "Gentle Introduction to the Adam Optimization Algorithm for Deep Learning," 13 January 2021. [Online]. Available: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/. [Accessed 8 December 2023].

[3] A. Kumar, "Analytics Yogi," 28 October 2020. [Online]. Available: https://vitalflux.com/keras-categorical-cross-entropy-loss-function/. [Accessed 12 December 2023].