CS 342: Assignment II
Kaggle Competition: The Nature Conservancy

Joe Meagher

April 10, 2017

Abstract

Can machine learning techniques be used to monitor commercial fishing vessels?

When provided with a training set of images taken from aboard an active commercial fishing vessel, can Artificial Neural Networks correctly identify fish contained in a test set of similar images? Here Multi Layer Perceptron and Convolutional Neural Network models are applied to this problem. The analysis shows that overfitting to non-representative features of the data is a challenge in this context, but some progress can be made with a naive application of these methods.

1 Introduction

Here I attempt to present a model solution to the second assessed coursework submitted by students on CS342: Machine Learning module at The University of Warwick. This assessment was based on the submissions made to the Kaggle Competition run on behalf of the Nature Conservancy and the accompanying written report.

2 Data Exploration

This competition ranks models on their ability correctly assign images to one of eight categories. The categories are 'ALB', 'BET', 'DOL', 'LAG', 'NoF',

'OTHER', 'SHARK', and 'YFT'. Performance on this task is assessed using the negative multinomial log loss metric, defined as

$$\ell = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log p_{ij}.$$

Here, N is the number of samples, M is the number of categories, p_{ij} it the probability given by the model that the i^{th} sample belongs to the j^{th} category, $y_{ij} = 1$ when the i^{th} sample belongs to the j^{th} category and 0 otherwise, and log is the natural logarithm.

Training and testing datasets are provided, of size 3777 and 1000 respectively. It is our models performance in classifying the testing dataset on which we are ranked in the Kaggle Leaderboard.

Consider some simple models against which we can benchmark any future models. The first model distributes images uniformly at random across each of the categories. This is a multinomial distribution [1] with $p = \frac{1}{M}$. For this model $\ell = -\log(\frac{1}{8}) \approx 2.08$. The next model considered is again based on a multinomial distribution, but this time $p_{ij} = \frac{n_j}{N}$ where n_j is the number of samples of category j in the training set of images.

The distribution of the training dataset across classes is illustrated in Figure 1. If this multinomial classifier was applied to the training dataset then we would find that $\ell = -\frac{1}{3777}\sum_{i=1}^{3777}\sum_{j=1}^{8} y_{ij}\log\left(\frac{n_j}{N}\right) = 1.61274$. When the classifier is applied to the testing dataset and submitted to Kaggle it receives a score of 1.64611. This is close to the score on the testing dataset, so I conclude that the training dataset is representative of the testing set. We also now have a benchmark against which to compare our models.

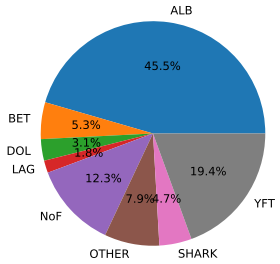
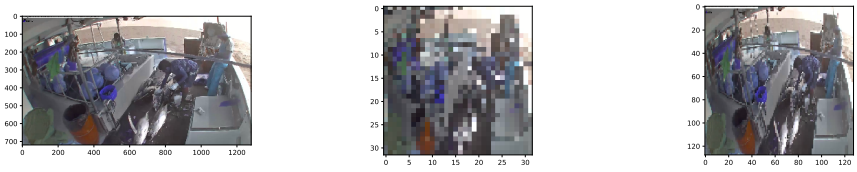


Figure 1: Pie chart illustrating the proportion of training examples belonging to each category.

During a preliminary data exploration some further points were noted. These should be kept in mind as models are developed.

- This is a relatively small dataset, and some classes have very few examples. a thorough treatment of this data would include data augmentation techniques.
- Due to the class imbalance, care must be taken during cross-validation to ensure that each class is appropriately represented in both training and validation splits.
- There is a lot of variation across the images provided. Resolution, lighting, weather conditions, and type of boat are just some sources of variation. It may be the case that models overfit to these non-representative features.
- Some images seem to be mislabelled.
- The position and orientation of fish varies from image to image.
- Fish seem of different species seem to have reasonably different shapes and colours.

3 Feature Engineering



(a) Original Image (b) 32 x 32 pixel Image (c) 128 x 128 pixel Image

Figure 2: Comparison of images at varying resolutions

Feature engineering involves transforming variables into a new feature space. It is hoped that in the new feature space the problem at hand becomes easier to solve [1]. In this context feature engineering includes reducing the size of images (see Figure 2) and converting them to greyscale, and while elementary, these are important to the eventual performance of our model.

Identifying the fish in these images is an object detection problem. It has already been noted that fish could potentially be characterised by their

colour and shape, and that this is a relatively small dataset and so data augmentation would be appropriate. These observations are reflected in the feature engineering techniques chosen.

3.1 Image Thresholding

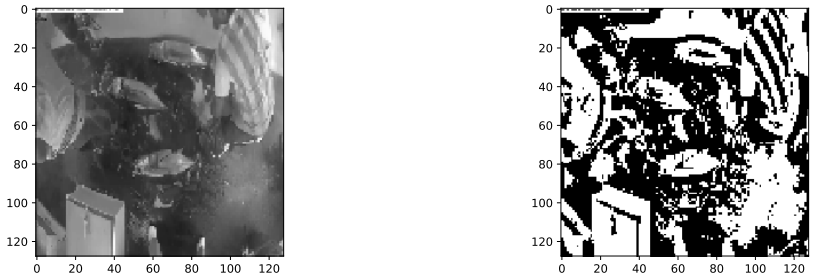


Figure 3: Comparison of a Greyscale Image and its binary representation given by a Gaussian Adaptive Thresholding algorithm.

Image thresholding, as applied in this context, is a segmentation technique which involves creating a binary representation of an image. This technique is most effective at segmenting images of high contrast. It is hoped that thresholding images in the training and test datasets will make the shape of fish in the image clearer, which may improve classification algorithms. The thresholding technique used here is an Adaptive Gaussian Threshold of block size 33 applied to 128×128 Greyscale Images. This method was chosen as lighting conditions can vary within an individual image, and the block size should capture whole fish, providing the binary representation illustrated in Figure 3. Note that Otsu's thresholding method was not applied here as it is best suited to images where a histogram of the pixel values forms a bimodal distribution. OpenCV provides descriptions and implementations of the thresholding techniques described here.

3.2 Histogram of Orientated Gradients

Histogram of Orientated Gradients (HoG) is an object recognition technique which has been successfully applied to pedestrian detection [2]. HoG is designed to describe the objects in an image by a feature vector of of the

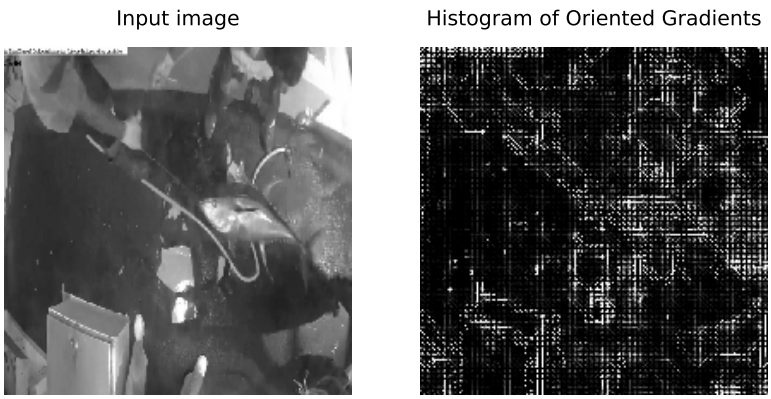


Figure 4: Comparison of greyscale image with it's Histogram of Orientated Gradients.

intensity and direction of the gradient is small regions of the image. HoG calculates the magnitude and direction of the gradients of each pixel in a cell of a specified size, It then computes a histogram of the gradients of that cell by a weighted vote over the specified orientations within that cell. The HoG for each cell can then be normalised over neighbouring blocks to make it less dependent on the luminance of the image. An implementation of HoG is illustrated in Figure 4. HoG was implemented here using the scikit-image library.

3.3 Data Augmentation

Data Augmentation techniques can be applied to a dataset to artificially create 'new' samples. Techniques such as cropping, flipping, rotating and jittering images are common [3]. Data augmentation could also be used to create a more 'balanced' dataset, in terms of the number of samples in each class. Here, random rotations in multiples of 90° are employed to generate new samples. A potential benefit of augmenting the dataset in this way is that it may go some way to preventing models from overfitting to non-representative features of the data, such as the type of boat.

There are a number of alternative feature engineering techniques which may be suitable for this type of problem, Principal Components Analysis or

Scale Invariant Feature Transform to name just two. However when considering feature engineering techniques it is important to ensure that they relate to the problem at hand.

4 The Multi Layer Perceptron

Now that an initial data exploration and feature engineering has been performed, consider how to go about solving this image classification problem.

Consider an image $\mathbf{X}$, where the vector $\mathbf{y}$ indicates which category it belongs to. Say that a human classifying $\mathbf{X}$ involves applying some function $f(\cdot)$ to $\mathbf{X}$, such that

$$f(\mathbf{X}) = \mathbf{y}.$$

Then an appropriate classifier is one which estimates a function $f^*(\cdot)$ which produces the same results as, though is not necessarily equivalent to, $f(\cdot)$. One method of approximating this complex non-linear function $f^*(\cdot)$ is to implement a Multi Layer Perceptron (MLP) model.

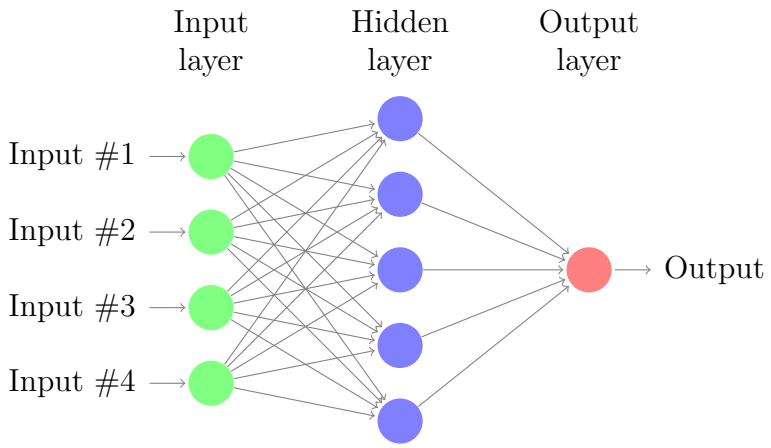


Figure 5: An example of the structure of an MLP

Details on MLP models, a form of Deep Feedforward Network, can be found in [1] and [3], however a brief summary will be provided here. A MLP is fed a feature vector $\phi(\mathbf{X})$ as its input. $\phi(\mathbf{X})$ is a vector containing some transformation of the original image, i.e. resized, Greyscale, HoG, etc. This input is fed to a number of hidden layers of a given number of nodes where the inputs are passed through some activation function before being passed

on to the next layer. The final layer of the model is the output layer, which is the models prediction. A MLP is a fully connected network and is illustrated in Figure 5

The activation function of nodes in the MLP allow the development of models describing complex, non-linear relationships using much simpler 'building blocks'. A typical activation function is the rectified linear unit where for a vector of inputs $\mathbf{x}$ a node outputs $\max\{0, \mathbf{w}^\intercal \mathbf{x}\}$, where $\mathbf{w}$ is the vector of weights associated with that node. Given a set of training examples, $\mathbf{w}$ is found by Backpropogation [1] [3].

There are a number of parameters that can be adjusted when optimising a MLP, these can be experimented with using a Grid Search Cross-Validation Algorithm.

When working with images, a MLP does not preserve the grid structure of the input.

Experiments naively using MLP models to classify images found that the best performing model was one estimated using 10-fold Stratified Cross-Validation on 32×32 pixel rgb colour images with the default parameters associated with the scikit-learn MLP implementation, MLPclassifier(). This model achieved a kaggle score of $\ell = 1.14059$, which would have been among the Top 20 assignment submissions. Note that MLPClassifier().predict returns a binary prediction for classification and so MLPClassifier().predict_proba is needed to produce probabilistic class predictions.

The MLP was not implmented here for the HoG or thresholded image feature vectors. This was because the assignment submissions suggested that these techniques performed worse than models using raw pixel values. This was probably due to the level of clutter in the images. If images were cropped to only contain fish these techniques may offer superior performance.

A MLP was trained on a set of randomly rotated images. This model performed horrendously and had a kaggle score of $\ell = 18.37$. While it is possible that some mistake was made in the implementation of the MLP, this was interpreted as evidence that the MLP is overfitting to non representative features which allow it to improve on the multinomial model outlined earlier. This is consistent with having a model using low resolution images performing relatively well.

5 The Convolutional Neural Network

Convolutional Neural Networks (CNN) offer an alternative method of estimating a function $f^*(\cdot)$ for classification. Like the MLP, the CNN is a feedforward network and relies on Backpropogation to estimate weight parameters. Unlike the MLP, a CNN takes advantage of the grid-like topological structure of images, uses the same small set of parameters across the whole convolutional layer, and is not a fully connected network [3].

There are a number of different types of layers that can be included in a CNN. The CNN implemented in this analysis used convolutional, zero padding, max pooling, and fully connected layers.

An intuitive way to consider convolution is to think of it as matrix multiplication where each row is the same as those adjacent to it but shifted one column to the left or right as appropriate. Applying convolution maintains relationships between adjacent pixels.

The output of convolution is smaller than its input. Zero padding layers effectively maintain the size of output as equivalent to the input. This means that a CNN of arbitrary depth can be constructed.

Max pooling is a method which encourages the model to be invariant to small transformations of the image by passing on only the maximum value of a set of pixels.

There are many ways of regularising a CNN to prevent overfitting such as early stopping and the use of dropout layers.

Dropout layers set some proportion of the output units for each batch to 0 in that layer. This effectively means that half of the features identified are discarded. Repeating this over all batches ensures that a variety of features are identified.

There are many variations that can be made to the construction of a CNN and describing the subtleties of these variations is beyond the scope of this report. Interested readers are strongly encouraged to refer to [3] when studying this material.

Students were provided with a script which contained an implementation of a CNN for this image data. An implementation of this model on 32×32 pixel images using a 10 fold Stratified Cross Validation yielded a kaggle score of $\ell = 1.20582$. Experimenting with various model architectures was subject to time and memory constraints. To get models running in any reasonable period of time students had to train their models by accessing the computers GPU. When trying to find an optimal CNN it is probably best to

base experiments on a demonstrably good architecture, such as [4].

6 Discussion & Conclusions

This report shows that naively implementing MLP and CNN classifiers for the problem posed by the Nature Conservancy on Kaggle provides models which perform better than a multinomial model based on the proportion of instances in the training dataset. This is encouraging. The implementations used here were probably overfitting to non-representative features, however there were many relatively straightforward steps that could have been taken to improve this. Using higher resolution images, augmented datasets, and larger, deeper networks, could all have resulted in improved performance. Time and computing constraints prevented these avenues being explored. It is clear however that Neural Networks offer a general approach which can be adjusted to tackle many specific problems.

Feature engineering did not improve the performance of classification models in this case. It is possible that feature engineering techniques such as HoG and Image thresholding would perform better when trained and tested on images cropped to only contain fish, rather than the images provided which included lots of clutter.

In fact, it is quite likely that in order to compete at the top of the Kaggle Leaderboard, a more nuanced approach to the problem is required. Models which first identify small regions of each image which are likely to contain a fish prior to running a classification algorithm on identified regions seem most promising.

Having developed an understanding the theory underlying and challenges associated with Artificial Neural Networks it is hoped that students are now in a position to further explore this active area of research with confidence.

References

[1] Christopher M Bishop. Pattern recognition. *Machine Learning*, 128:1–58, 2006.

[2] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005.*

CVPR 2005. IEEE Computer Society Conference on, volume 1, pages 886–893. IEEE, 2005.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[4] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.