CS342 - Assignement 2

University: u1406032
Kaggle: CS342_TheoWillYouMarryMe
Department of Physics, University of Warwick, Coventry CV4 7AL, United Kingdom

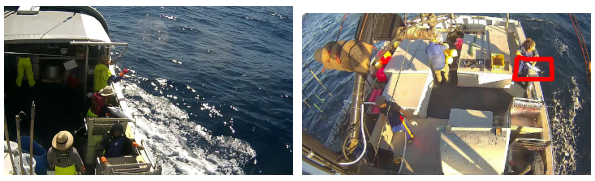
This paper presents an analysis of the methods used and the results obtained as part of the Kaggle competition "Nature Conservancy"[1]. It was found that mixing thresholding, HoG and PCA as preprocessing step for a single layer MLP gave the best results. CNNs were also tested, but overfitting and a lack of computational time made it challenging to set up. A discussion regarding the next possible steps for improvement is then made.

I. Task 1: Preliminary Analysis

The data of the competition came under the form of CCTV-like pictures taken onboard boats, from a multitude of angles. The pictures weren't all the same size, with width and height ranging between 700 to 1300 pixels, and were overall of poor quality. This meant that in most cases, little to no information was lost when rscaling up to about half the initial size. Additionally, the lighting and colouring of pictures were often of bad quality, and a fair subset of the pictures were shot using night vision, making colouring very hard to use as feature. Also, drops could often be found on the lens of the camera.

Somewhere in those pictures, a fish had to be found and classified. In some occurences, the fishes were in a centered position, but for the most part, the fishes occupied only a small part of a picture with ussually very complex backgrounds. Additionally, fishes were found various pespectives, and often, were partially covered by obstacles, with sometimes only a small part coming out.

Nevertheless, some key features could already be observed which, assuming they were correctly exctracted, could provide very high results. Taking for example the very atypical shape of the *Dolphinfish* and *Moonfish*, which makes them especially recognisable. The *shark* also possesses key features, for example the long but thin body, the gills and the infamous *shark* fin. On the other hand, the three tunas are very similar to each other, only the long fin of the *yellowfin tuna* can easily discern it from the two others.

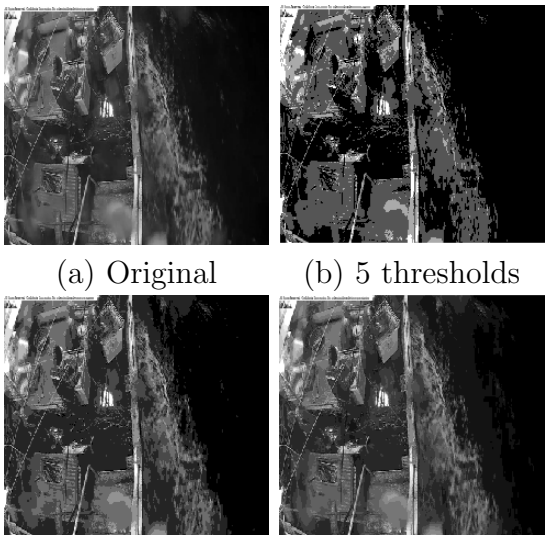


(a) (b)

Figure 1: (a) shows an example of a picture I am not personally capable of finding the fish. (b) shows an example where the fish occupies are very small part of the image, and additionally a fisherman is in the way

II. Feature Extraction

In this context, any sort of raw image classification appears impossible, and it is therefore necessary to apply some feature



(a) Original (b) 5 thresholds

(c) 10 thresholds (d) 15 thresholds

Figure 2: With 5 thresholds, the appears a bit extreme, while with 15 thresholds, it can barely be noticed

extraction to the pictures. With this in mind, various methods were implemented and are discussed.

Thresholding was the first method to be implemented and used, and was found to be extremely useful to "de-noise" pictures and remove small variations due to for example: poor camera, some characteristic of the scene, slight variations in the colour of the sea, or small features such as water puddles. Applying less than 5 thresholds was found to transform the image too much, losing large amount of information, while more than 15 seemed to not change the picture enough, failing to reduce the complexity. Finally, about 10 thresholds was chosen as a good middle-ground between loss of details and gain in uniformity.

A similar method to thresholding is Histogram of Gradients (HoG). This method allows to have additional information on the scene by providing the gradients per block of picture. It was found that HoG worked really well when applied to thresholded pictures, since noise would've been removed and therefore the gradients calculated could be more descriptive of the "skeleton" of the scene. Additionally, it was found that, since the pictures had been rescaled to 400x360, anything over 2x2 gradient blocks made the images recognisable, 2x2 blocks with 4 gradients orientations were used.

Given the intrinsic high dimensionality of pictures, an interesting field of methods which was looked at dimensionality reduction methods. These could obviously not be used for direct unsupervised classification, knowing the complexity of the pictures and the fact that the fishes tended to only occupy a small area. On the other hand, reducing the data to few hundreds dimensions could make it much easier for other classification method to train. With this context in mind, three methods were tested: PCA, LLE and ISOMAP. The results are discussed in the next section.

Additionally, given the pretty small amount of data, especially for a CNN to be efficient, data augmentation was implemented, using open source code written by **github** user *aleju* [2]. With this tool, about 4000 more images were created from flipping, cropping, adding noise, blurring, etc., and all these parameters were tested to check that the resulting images were still classifiable.

A key way to perform well on this task would be to be able to find the fish and crop everything else. Therefore, the match_template function from `skimage.features`, was tested to find features discussed earlier, but it was giving very poor results, even when the feature matched was clear. Another matching method is SIFT, which appeared great. Unfortunately, I was unable to get it to work due to installation errors.

After this preliminary analysis, it was decided to resize all the pictures to 400x360 resolution; the resolution was found by trial

and error by testing various resolution and checking if the scene still made sense after HoG, which requires blocks to be a minimum of 2x2. On the other hand, for the CNN, it was decided that with 200x200, most key features were still recognisable and therefore the network should be able to extract them.

III. Results

Initially, as a showcase example of what failure looks like, the raw pixels were used as input data for an MLP. Various hidden layer sizes, batch sizes, learning rates and regularizations were tested, but the results were always the same: the MLP would not fit at all, usually having a slight biase toward one of the classes, no matter the input. The classification probabilities were all about the same, at around 15%, which implies a nearly uniform distribution throughout the classes.

Next step was to implement and MLP using preprocessed inputs. Initially, the preprocessing was divided in 4 steps: resize, grayscale, apply 10-threshold and apply PCA. To test the structure of the neural network and the parameters, K-fold validation was used, with K varying from 10 to 20. An important issue was the number of parameters involved e.g. the number of components of PCA, the batch size, learning rate, etc.
All of these are more or less interconnected, therefore a grid search would be needed to get a general idea of the combination to chose[1]. Unfortunately, a grid search was too computationally expensive, especially since all the tests were ran from my laptop. Therefore, cuts were made, and parameters were tested singularly using K-fold, and the best structure at each step was recorded for the next step.

[1]In theory, more efficient searches could be used, but these would have to be manually implemented, and would take too long

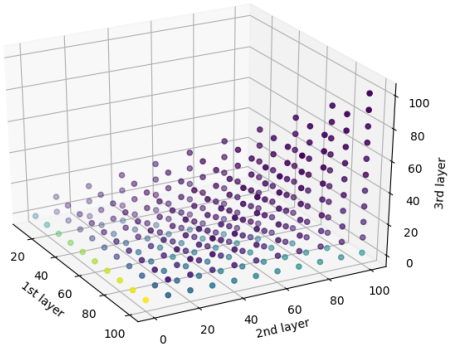


Figure 3: Representation of the score with the different layer sizes. On can clearly see that the more complex models have very low (dark) performance, and the simplext model, 100-0-0, has the highest.

Initially, a base structure of 3 hidden layers using a *logistic* activation function was chosen. This was based mostly on intuition as a structure from which it should be able to evolve, the reasoning was that it wasn't so small that it might go completely past the point, and it wasn't too large that it would take years to fit. The *logistic* function was chosen to allow "soft" classification as a first step, which could later be replaced with possibly a *relu* function.

Having decided a 3 hidden layers structure, a basic *3-dimensional* grid search was executed to get an idea of the performance, and a place to start improving from, and 200 components were intuitively decided as a good place to start for PCA. The range of the search was 10-100, with a step of 10, for each layer. It was found that the best performances were obtained with 100-10-10 structure *(See Fig.3)*. This was especially interesting since the 3 layers were at an extrema: the first, size 100, was the highest value of the grid search, and the two others, both 10, were at the lowest. Therefore, it was hypothesised that the best score would actually be with only one layer, and simulations were ran. The results from these successfully confirmed

the single layer hypothesis, which was therefore kept for later tests, and, after extending the range, showed that the best scores were obtained with a single hidden layer of size 130, with validation accuracy of around 65%.

Having found a good structure, testing the various DR methods was the next step. PCA only requires the number of dimensions to be reduced to, while LLE and Isomap additionally require the number of neighbours. A grid search was executed to find the best method and parameters, and the MLP was found to consistently score higher when PCA was being used, with number of component of 240. It is interesting that both LLE and Isomap gave impressive results when asked for a reduction to *3-dimensional* space, generating a circular shape with clear categorisation of the scene in the pictures (e.g. see view, top view, side view..).

Next, it was thought that HoG applied to a threshold image would likely give better results, since the threshold would successively extract the skeleton of the picture, for which the gradient could provide much more meaningful results. This was therefore applied, and showed impressive results, with the performance score dropping by more than `0.4` down to 1.26.

As final task, the implementation of a CNN was required. This step required a lot of further reading, given my initially poor knowledge. The activation functions and loss functions on the other hand were kept constant, *relu* for all the middle layers, *softmax* for the output layer and *cross entropy* for the loss; these are conventional activation structures for image classification found in the litterature, and the cross entropy loss is the one used by the competitions to score the models, so the choice was simple. Additionally, the pictures were kept at 360x360 resolution,

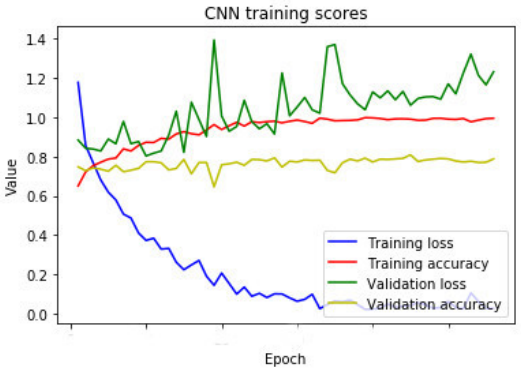


Figure 4: Textbook example of overfitting, with the training loss going to zero and the validation loss increasing.

but this made for very long training times which were not computationally affordable.

The first model, contained 6 layers, 3 convolutional layers with decreasing number of filters and 3 dense layers, of sizes 1024, 512, and 8 to give an 8 class output. This gave peculiar results. More specifically, it was noticeable that the CNN would most of the time pick a class, and give it a very high certainty, while giving very low probability for the others. This resulted in very low scoring, even if the accuracy was of around 60%.
To solve this issue, and executive decision was made to reduce the image size to 200x200, which was chosen by looking at the picture and checking whether the key features and generally the fish were still recognisable. In addition, the CNN structure was completely remodelled, after realising that the number of filters should increase with depth of the CNN, while the size of the convolution should decrease.

This lead to a model which seemed to work well and was much faster to train. The training resulted in what can only be called "textbook" overfitting *(See Fig.4)* (Theo, I give you the authorisation to use this plot for future years).

Therefore, as suggested in a paper[3] by N.Srivastava et. al, *dropouts* were added, with probability of 25%. Additionally, it

was decided that the training data would only consist of augmented data and not the real samples, so that the validation could really be used as a measure of how the model would perform on the unknown data, and overfitting could easily be diagnosed in a major difference between training and validation results.

IV. Discussion

Below, you may find my progression graph, together with the appropriate comments and dating. Overall, 27 submissions were made.

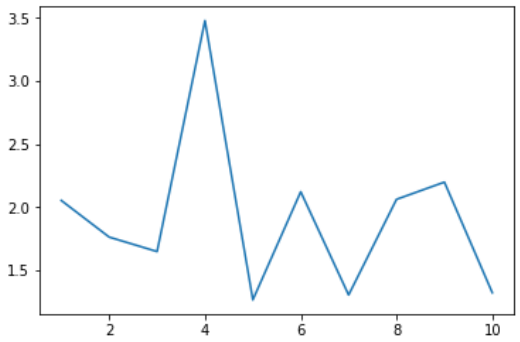


Figure 5: Progression graph. The x axis corresponds to the number in the list, and the y axis the Kaggle score.

1. 7 Mar. Used MLP on raw data, no engineering applied.
2. 9 Mar. Images were grayscaled, then a 10 threshold was applied and the result was flattened and used for PCA with 200 output components. This was then inputted in an MLP with one hidden layer of size 100.
3. 9 Mar. Same as previous, but with hidden layer of size 130.
4. 11 Mar. First attempt using a CNN. The structure was 6 layers, 3 convolutional and 3 dense: 200 filter, 3x3 convolution, 80 filters, 3x3 convolution, 80 filters, 3x3 convolution. This was showing accuracy of about 0.63 in validation, but gave too strict classifications.
5. 12 Mar. Same as #3, but additionally using HoG after thresholding, which gave the best performance, with score of 1.26725
6. 13 Mar. Tried making an ensemble from the previous MLP and the CNN, couldn't achieve a good balance
7. 14 Mar. CNN using 200x200 grayscaled images as input, with 3 conv. layers, this time as 64, 128, 128 and 2 dense layers.
8. 14 Mar. Ensemble of previous CNN with best MLP, this time providing much better results. The balancing was done as such: if both have same answer, give very high probability to that class, otherwise, trust the MLP, which tends to give more uniform probabilities which score high on cross entropy loss.
9. 15 Mar. More CNN structures failing, tried increasing number of convolutional layers, decreasing number of dense layers.. Only one record is shown for spacing reasons.
10. 16 Mar. MLP made using Keras instead of MLPClassifier. Same structure and preprocessing as the other MLP, but more options to play around with.

Continuing this projects, a main task would be to train a model to recognise whether a fish is present in a smaller block of the image. This would allow to then train another CNN to only have to recognise a fish in a much nicer context, and not have to analyse a complex picture. To train the first CNN, online datasets of fishes could be used instead of having to extract from the given pictures.

Obviously, to get higher scoring, ensembles would be required. This would include the already discussed CNN, methods further models using feature extraction, dimensionality reduction, clustering and classification, including for instance K-means applied to reduced data, or SVMs.

References

[1] Competition hosted by kaggle, Web. Visited 15 Mar. 2017, `https://www.kaggle.com/c/the-nature-conservancy-fisheries-monitoring`

[2] Code taken from user *aleju*, Web. Visited 15 Mar. 2017, `https://github.com/aleju/imgaug`

[3] N. Srivastava, G.E. Hinton, A. Krizhevsky et al, Journal of Machine Learning Research, **15** (2014).