Assignment 2 - Shehel Yoosuf - 1425964

Kaggle Username: cs342shehel
Kaggle Display name: CS342_1425964

Abstract:
The goal of 'The Nature Conservancy Fisheries Monitoring' kaggle competition was to identify and classify fishes categorised into 8 labels. The dataset was found to be imbalanced, very noisy and hard to extract relevant features. Some feature engineering methods like data augmentation and thresholding were used among others to try and alleviate this problem. However, by the end of my work, it was clear that fish localization was necessary to improve accuracy to reasonable levels. From the methods I have tried and failed, I have found that training CNNs to identify fishes and to separate the fish in question from the background has been found to be the most appropriate way to tackle this classification problem given our current tools and technologies.

Task 1

1.1 Data Exploration

Perhaps the most important fact I learnt a bit too late was about 'knowing your data'. The provided data set pose multiple issues that makes any straightforward approaches unfavourable. Below are some of the factors that makes it so.

1.1.1 Data distribution

Fig1 shows how 3777 images in the full training dataset are distributed. It is highly uneven especially LAG which only has 67 images compared to 1719 in ALB. This meant that a good set of augmented data is necessary as well as stratified split of train and test data during validation.

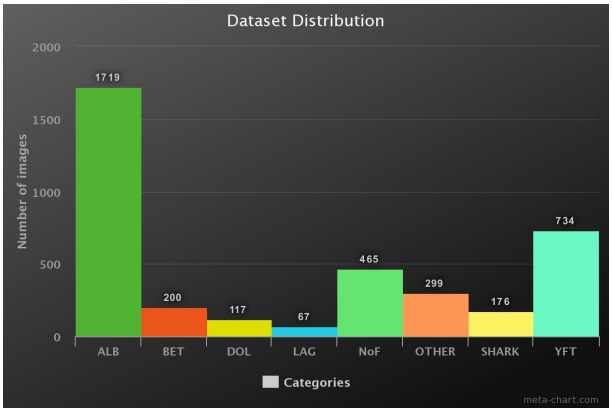
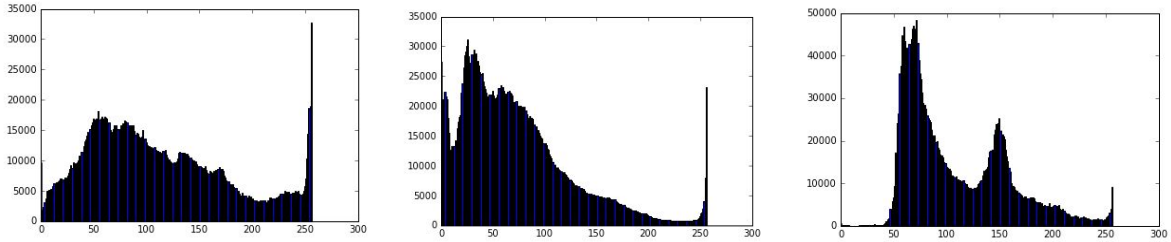


Fig1 Training data distribution

1.2 Pixel distribution



The above histograms show the intensity distribution of 3 randomly selected images from the dataset. It is clear that it is not uniform and therefore, hard to find distinguishing patterns between pixels in such a set. This inconsistency is due to several reasons like pictures captured under different weather conditions, illumination, people, ocean, different boats and other such elements that add noise making the raw pixel values as a whole almost useless. Therefore, feature engineering is essential in the process of identifying the fishes before any classification can be done.

1.3 Others

Another minor observation was that not all images were of the same size. Therefore, resizing was necessary for both standardisation and performance.

Task 2

2 Feature Engineering

Feature engineering techniques were chosen that would best alleviate the problems with the dataset discussed above.

2.1 Data Augmentation

A simple and effective augmenting script by CodeBlox software was used for transformations including rotation, translation and horizontal flipping. The number of transformations done was inversely proportional to the original distribution in that ALB was untouched and LAG was transformed the most. This process somewhat balances the dataset and avoids overfitting by generalizing the data.

2.2 Thresholding

Thresholding was chosen to tackle the poor representation raw pixel values provide. Since thresholding is an image segmentation method, it is roughly able to extract shapes possibly giving better representations for use in MLPs.

Implementation of Otsu's method in Scikit-image was used to implement the thresholding. It gave a simple way to separate the foreground from the background cutting down a lot of noise analysed in 1.1.2.

2.3 Histogram of Oriented Gradients (HOG)

HOG is a feature descriptor used for object detection. HOG excels in human detection who usually form a coherent shape in a scene. HOG algorithm cannot be directly applied to this problem and it was a poor choice given that it wasn't complimented by an object localizer that could potentially separate the fish from its noisy background,

Unsurprisingly, HOG performed poorly with MLPs as the noise gives a false representation of the image. I also experimented with other feature descriptors like SURF and ORB which all gave similar results.

Task 3

3 MLP with raw pixels Best Score - [Best Score - 2.08521]

Multi Layer Perceptron (MLP) trained on raw pixels confirmed the findings in 1.2 providing a best score of 2.08521. This score was reached after using cross validation to tune the hyper parameters. Stratified K fold was used to split the data.

10 hidden units with 4 hidden layers, 'adam' solver and the rest were kept at default. Training this model used up too much time for underwhelming results, so further efforts were not made to squeeze out the best performance.

Task 4

4 MLP with Feature Engineering [Best Score - 1.48581]

MLPs trained on using a larger dataset through data augmentation and a better representation of the images through Otsu's thresholding, were able to provide great improvements over raw pixel representations. Blurring and equalizing the histogram also seemed to improve scores in certain cases, especially with Otsu.

4.1 Tuning Hyperparameters

10 fold Stratified cross validation was used to monitor the performance. Stochastic gradient descent was used as it opened up room for adjusting other parameters. Models were trained on 320x320 images and cross validation took quite some time depending on the size of the dataset. To overcome this convergence problem, an alpha of 0.0008 was used along with an adaptive learning rate and early stopping set to true. This also helped with the problem of overfitting posed by the nature of the dataset, All that remained was picking the hidden layer sizes which pointed to simpler network models (10 neurons and 2 hidden layers for the best score)

On hindsight, expecting MLP to perform well given the complicated image scenes was wrong. Reasonable models can only be achieved by detecting and separating the fish from the background. A CNN trained to identify and bind the fishes before classification would have been a better feature engineering approach.

Task 5

5 Convolutional Neural Networks (CNN) [Best score - 1.25142 (Raw pixel)]

CNNs was perhaps the most interesting part of the assignment because of the flexibility and the computational time required to outperform the best MLP I was able to build. I started off with CNNs with the base script provided in Lab 6. Improving was then an incremental process as I learnt more of the functions and parameters that could be adjusted. These steps are described below.

5.1 Base Model Design

Most of the design decisions were through trial and error with some 'rule of thumb' guidelines. Models with different input sizes were tried with 128x128 images giving the best results.

5.1.1 Batch Size and Epoch

Batch size was set to 64 or 128 in models that could run it. It isn't clear which is better because not all models worked with 128 batch size. Epochs were set at 100 but it rarely crossed 50 with the early stop based on validation set.

5.1.2 Convolutional Layers

A rule that seemed to work consistently was to increase the depth as the image dimensions are reduced through the layers so as to keep the representational power. In practice, 2x2 seemed to work better in this case. Standard 2x2 max pooling was utilised.

5.1.3 Fully connected Layer

Larger number of neurons in the initial dense layer provided better results indicating complex features are being learnt in this stage.

5.2 Overfitting

5.2.1 Dropout

Dropout constrains the network from being too reliant on certain neurons by removing units with some probability helping with generalization. This was the first regularisation technique used and it also did give a visible boost in performance during the initial stages. However, its contribution wasn't obvious during later stages where more regularisation techniques were added and removing or reducing probability of some dropout even helped increase the scores. It seemed to do best in the fully connected layer with the dropout probability being always less than 0.5.

5.2.2 L2 Regularisation and Batch Normalization

L2 regularisation and Batch Normalization significantly improved the performance of the convolutional layers. More normalizations between the layers tends to work very well which can be due to the fact that the dataset is limited and the network was overfitting.

5.2.3 Weight Initialisations and Zero padding

Although reasons are not clear, ‘He_normal initializations’ performed better than the others during tests and it has been shown to work well with ReLU activations. Zero padding was utilised to build models with deeper layers while being constrained by the input shape of the image.

Task 6:

6 Progression Graph

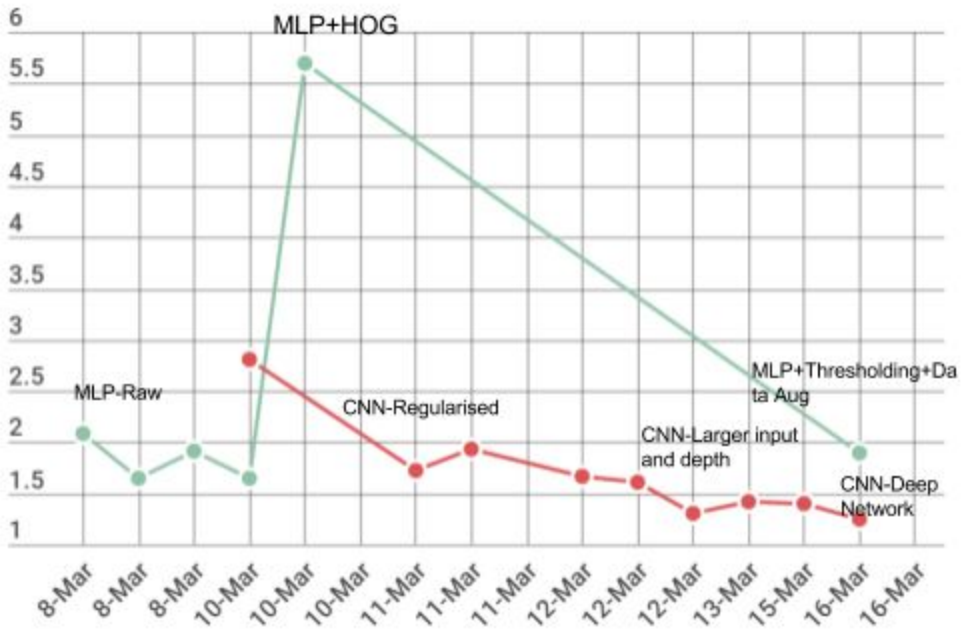


Fig 3

.

The graph generalizes a total of 24 submissions and separately shows how my models have progressed over time. In the case of MLPs, it starts with raw pixel values, goes through feature engineering and finally ends up with the top score after fully understanding the hyper parameters and adjusting it. CNN also took a similar course beginning with a 3 convolutional layer model to a very complex and deep one. However, my initial approach to the problem was fundamentally flawed in that I tried to find the best method or tool to solve the problem rather than following a more data driven approach. This is evident from the progression graph which is not consistent in reducing the loss and also the complicated CNN model only took me so far. Poor time management didn’t help either.

Nevertheless, the assignment covered a lot of ground and I was able to gain a broad understanding of the best of both machine learning and computer vision. As for future work, the first thing I would look at is object localization. Another great place for improvement would be to look between the layers and see what the CNN sees in each step. This will give a better awareness of how the network design can be improved. And finally, incorporating all that into an ensemble model should give reasonable results.