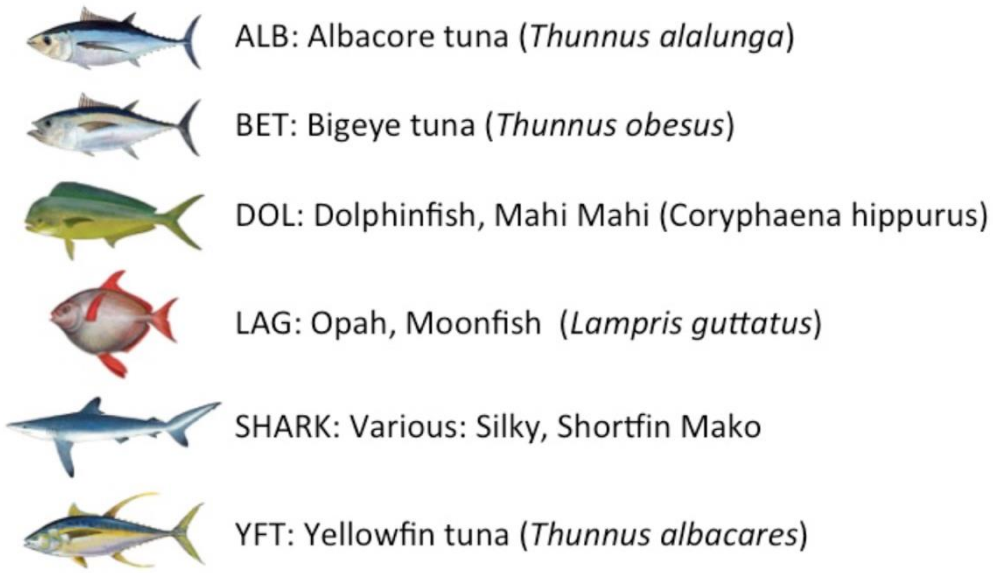
A Report on Detecting and Classifying Species of Fish

Project Members:

Priyanka Bandekar (NUID: 001055485)
Prasham Shah (NUID: 001057833)

Medium Blog

Table of Contents



Fish images are not to scale with one another

3. Evaluation Metrics

- The performance for each of the model is evaluated based on the logarithmic loss score for multi-class classification problem.

- The formula is given as:

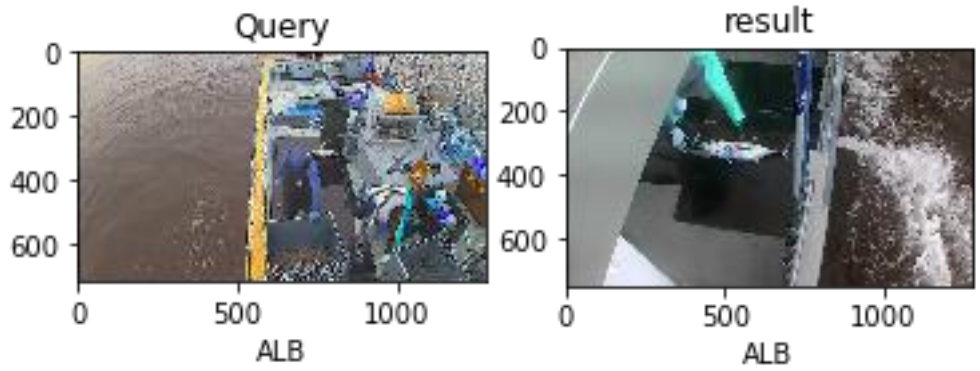
$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij}),$$

- where N is the number of images in the test set, M is the number of image class labels, log is the natural logarithm, yij is 1 if observation ii belongs to class j and 0 otherwise, and pij is the predicted probability that observation i belongs to class j.

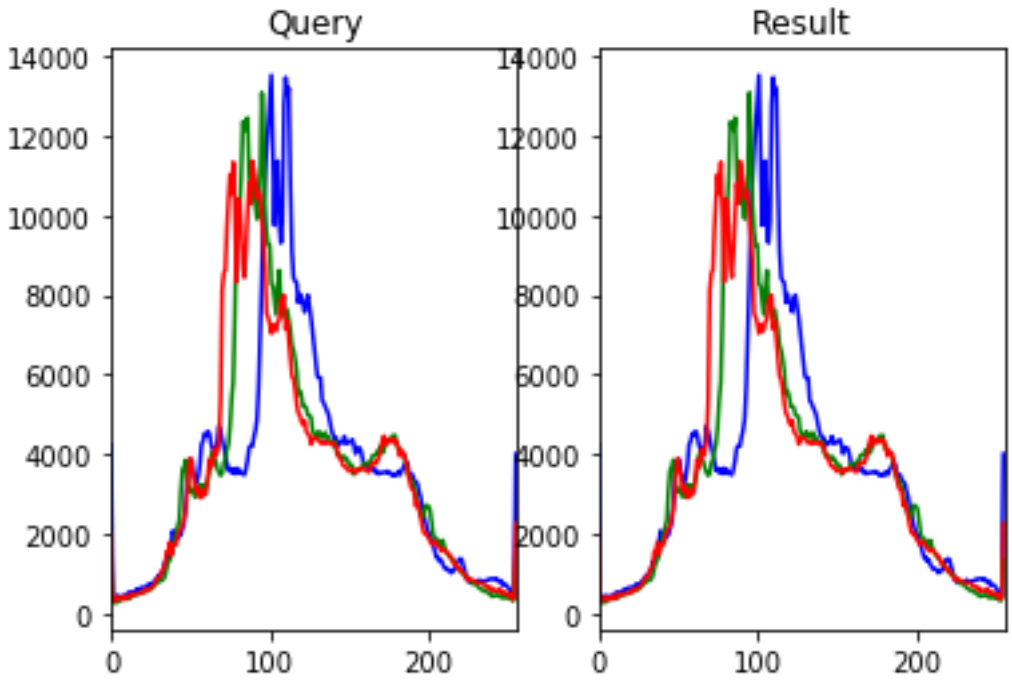
4.Exploratory Visualization

- For the purpose of feature extraction, we extracted the color histogram from the raw input images as features. Histograms are a very powerful technique for plotting the distribution of colors in each of the image by the frequency of each of the pixel values in their respective color channels. Euclidean distance metric was used here to compute the color histogram for each of the input image from the training set and retrieve the histogram of the result for the most similar image.

- It is assumed that similar images will have similar color histogram distribution. For instance, let us take a randomly chosen image and the result returned which is given as follows:



- The images have the same labels of a fish belonging to the class ALB, but they appear quite different. Their color histograms, however, look very similarly distributed.



5.Models Employed

5.1 Convolutional Neural Network

5.1.1 Data Preprocessing

- The input training images are read in using the CV2 library. Each data point is first sampled down from 1280X720 pixels to 480X270 pixels, then converted to numpy unsigned integers (0-255).

5.1.2 Splitting the data

- We then split the data into training and validation sets with a test size of 0.23. The labels/classes for each of the image are also one-hot encoded in the binary form.

5.1.3 Building and compiling the model

- We first setup the CNN model wherein all the input data is first normalized and converted into float32 by the activation layer. It is then followed by a group pf 3 convolutional layers with zero padding and max-pooling layers to extract more features.

- The filter size for the convolutional layers is first set to 36 and then to 72 for a gradual learning process, considering that the fish to be categorized are located in the specific portions of an image.

- The kernel size is set to 5X5 and then reset to 3X3 for the exact same reason. The activation function used here is ReLU since it does not activate all the neurons at the same time. This means that the neurons will only be deactivated if the output of the linear transformation is less than 0.

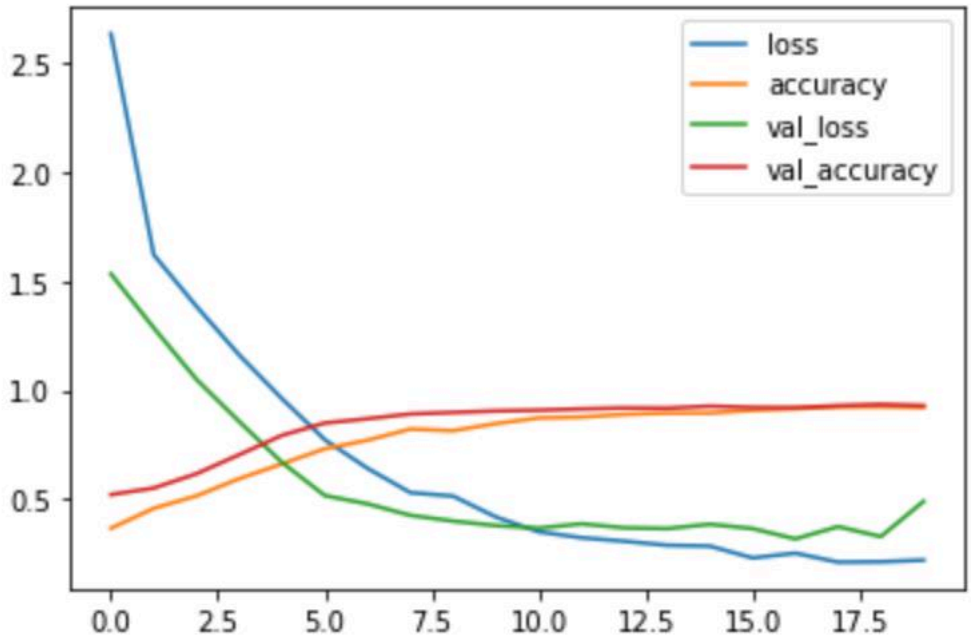
- Finally, the matrix is flattened, and the data are further passed through densely connected layers for feature learning. They are then finally passed through a layer of 8 perceptron for output using a softmax activation function.

- We use a softmax activation function because it is best suited for returning the probabilities of a datapoint belonging to a particular class, basically for a multi-class classification.

- For model compilation, the Adam optimizer is attempted as it is a gradient-based optimization of stochastic functions, thus including the advantage of both RMSprop and AdaGrad optimizers.

5.1.4 Model Performance

- The model is fitted to the training data with number of epochs set to 10 and an early stopping mechanism. The purpose of the callback function is to prevent overfitting in the model. If we would have ran the model for 24 epochs, this is the output that we would have got,



- According to our model, the training time is approximately 11 hours. It can be observed from the log and the plot above that loss and val_loss decreased over-epochs and for val_loss, it converged to approximately 0.35-0.45.

- The accuracy converged to approximately 89% based on the validating dataset. We get a logloss of approximately 0.39 means that the right class is attributed to a probability of 0.677 in average, which is quite high considering this is a multi-class classification problem with 8 classes.

5.2 VGG16

5.2.1 Architecture

- VGG16 is a pretrained model available in Keras, Tensorflow and many more Deep Learning libraries. It was developed by Oxford's Visual Geometry group. It comprises of convolutional layers of size 3X3, max pooling layers and fully connected block.

- Layers employed here:

    1. **Convolution**: It is a layer that performs a convolutional operation, creating several smaller picture windows to go over the data.

    2. **MaxPooling**: Pooling is a down-sampling operation that reduces the dimensionality of the feature map. MaxPooling operation grabs the maximum pixel value given an image of NXN size.

    3. **Dropout**: The Dropout layer is a mask that nullifies the contribution of some neurons towards the next layer and leaves unmodified all others. Dropout layers are important in training CNNs because they prevent overfitting on the training data. If they aren't present, the first batch of training samples influences the learning in a disproportionately high manner. This, in turn, would prevent the learning of features that appear only in later samples or batches.

    4. **Flatten**: Flattening transforms a two-dimensional matrix of features into a vector that can be fed into a fully connected neural network classifier.

    5. **Dense**: They are fully connected layers where the weights of the convolution layers are mapped to the right labels based on some activation function, we have used softmax here.

- Activation functions

- The activation function is a mathematical "gate" in between the input feeding the current neuron and its output going to the next layer. It can be as simple as a step function that turns the neuron output on and off, depending on a rule or threshold.

    1. **ReLU activation** - A ReLU layer performs a threshold operation to each element of the input, where any value less than zero is set to zero.

    2. **Softmax activation** – It is a function that takes as input a vector of K real numbers and normalizes it into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers.

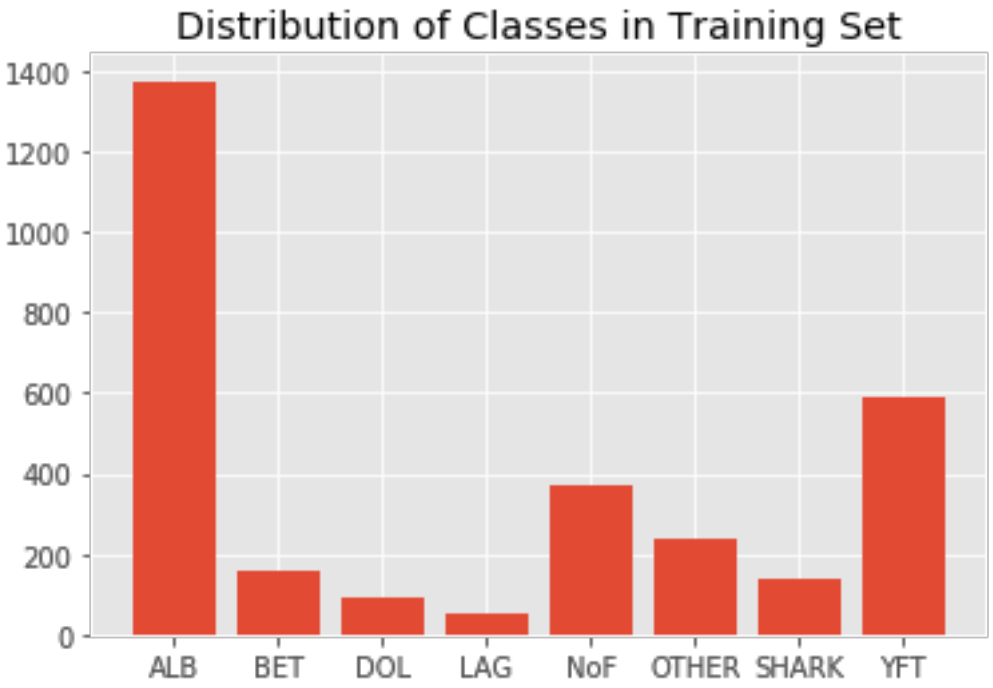
- Optimizers

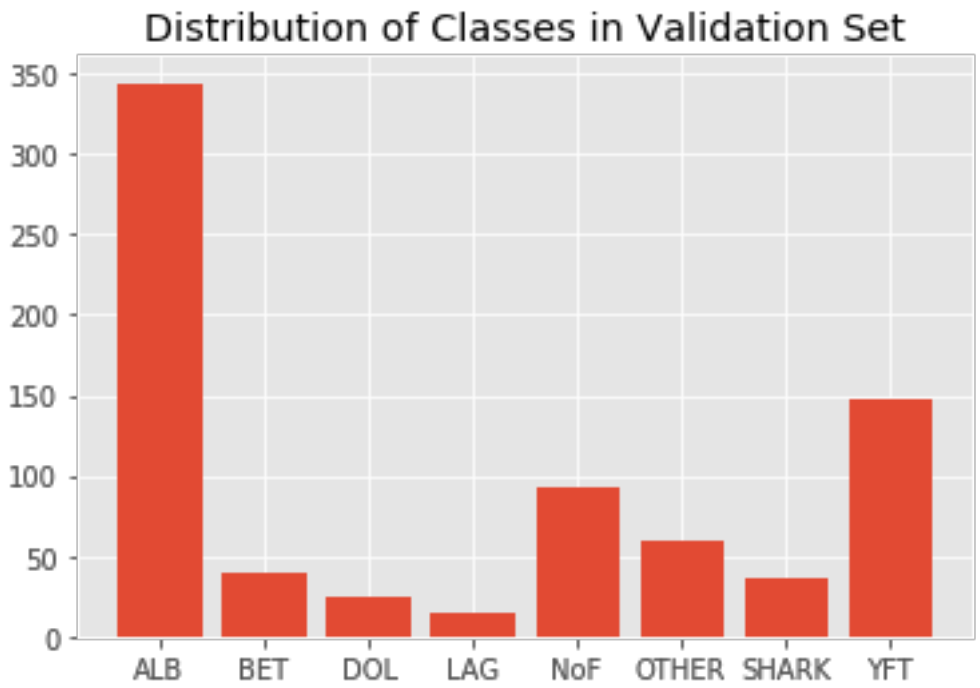
1. **Adam** - Adam uses an adaptive learning rate and is an efficient method for stochastic optimization which only requires first-order gradients with little memory requirement. It combines the advantages of Adagrad optimizer to deal with sparse gradients and the ability of RMSProp optimizer to deal with non-stationary objectives.

5.2.2 Data Preprocessing

- The VGG16 net expects the input color channel to be (B, G, R). But python needs the input color channel to be (R, G, B). We thus had to convert the images from (R, G, B) to the (B, G, R) format.

- The dataset provided on Kaggle had images of a mixed sizes, so here we had to resize the images to fixed image size of 150X150X3. For the need to have a validation data set, we just randomly placed 3019 images in the training dataset and 758 images in validation dataset.





- Keras ImageDataGenerator was also used to obtain the data in batches and process them with their appropriate labels. Validation data was used to obtain the validation accuracy and compute the loss during training.

5.2.3 Splitting the data

- The dataset has been split into training and validation sets. Around 3019 images have been stored in the training set directory and 758 images have been stored in the validation set directory.

5.2.4 Model building

- The VGG16 model has been provided by the Oxford's Visual Geometry group. We are making use of the model for training our preprocessed training image features and validating it using our preprocessed validation image features.

- Feature extraction operations have been performed on the training and validation sets. Before training the mode, we are label encoding the training and the validation set labels.

- The VGG16 model comprises of 5 blocks. Each block is composed of 3 convolutional layers, followed by a MaxPooling layer. The output is then flattened and passed to a mix of Batch normalization, Dense and dropout layers. The dropout layer has been set to 0.5. The output of the previous layers finally connects to the Dense layer where softmax activation function is used.

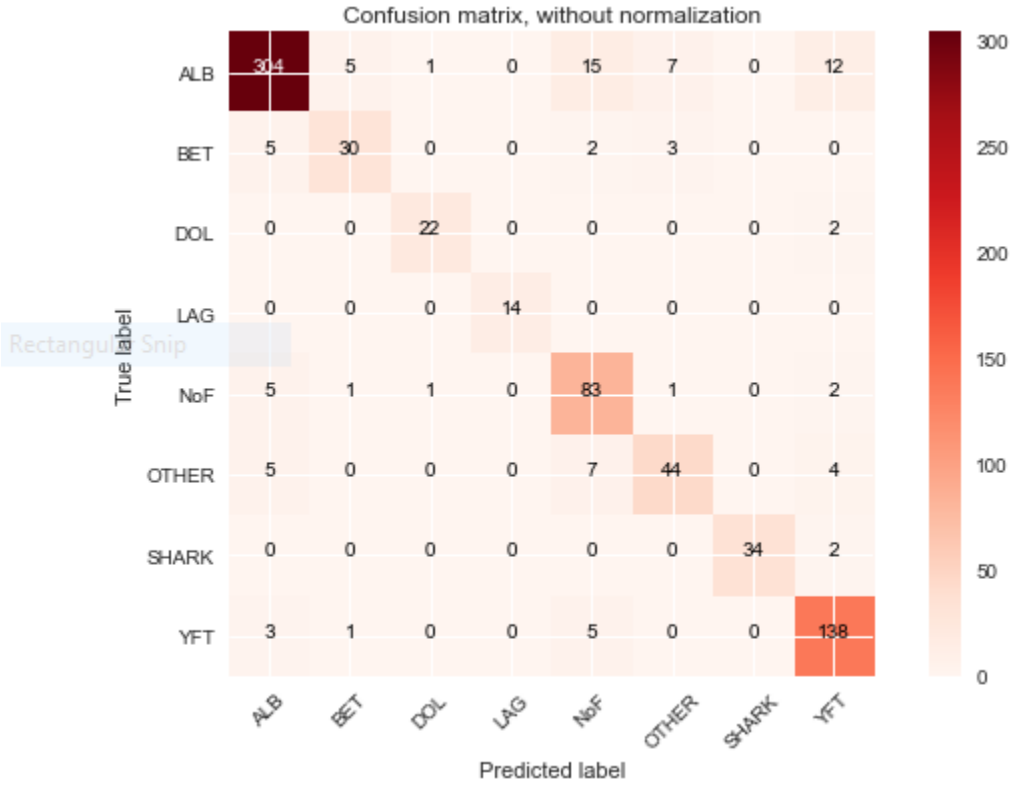
- We compile the model using adam optimizer and compute the accuracy metrics. We are using loss as categorical_crossentropy as it is a multiclass classification problem. We also make use of an ImageDataGenerator in increase the model robustness. We also make use of an EarlyStopping object to prevent overfitting in the model.

5.2.5 Model Performance

- We fit our model to the training features and use the validation set features to validate the model performance. The model is trained for 5 epochs (due to the computational power limitations) and a very good validation accuracy of 88.39% is obtained.

- It has been observed that the log loss of this model is around 1.3, which can be greatly improved by performing Data Augmentation. The accuracy of the model can further be bolstered by plotting a confusion matrix and check how the model did on predicting the validation set features.

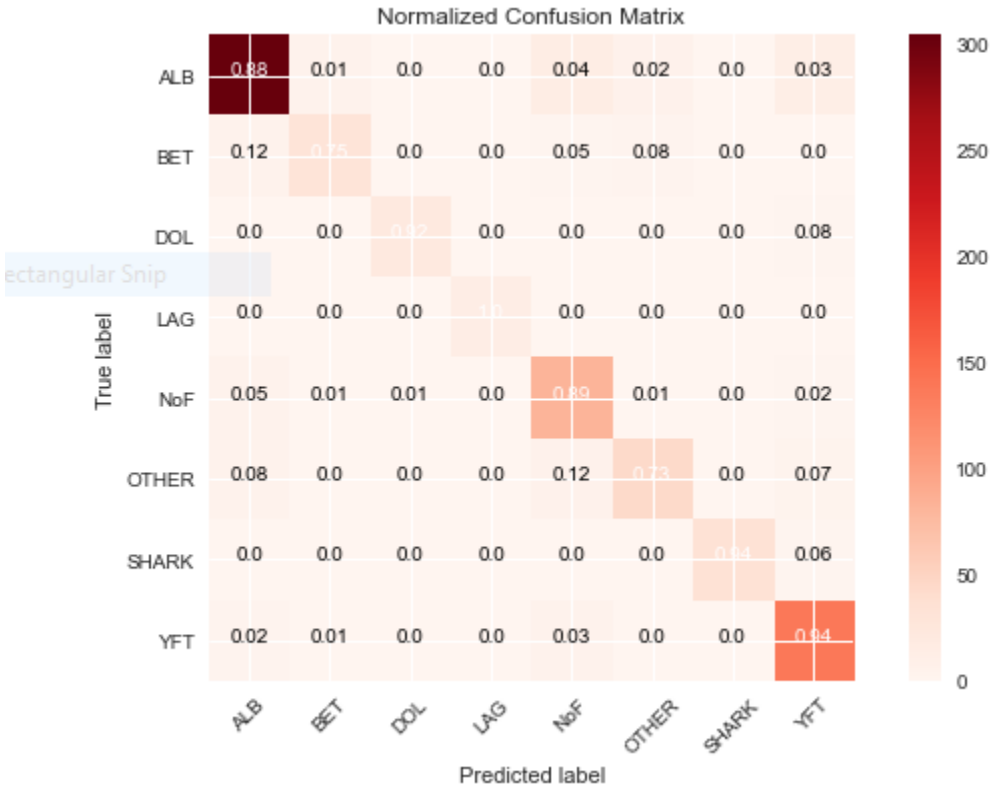
- The model had 669 labels that were correctly predicted out of 758 samples. We can see that it did a nice job as it also had a validation accuracy of 88.4%. The confusion matrix in a non-normalized form is shown below.



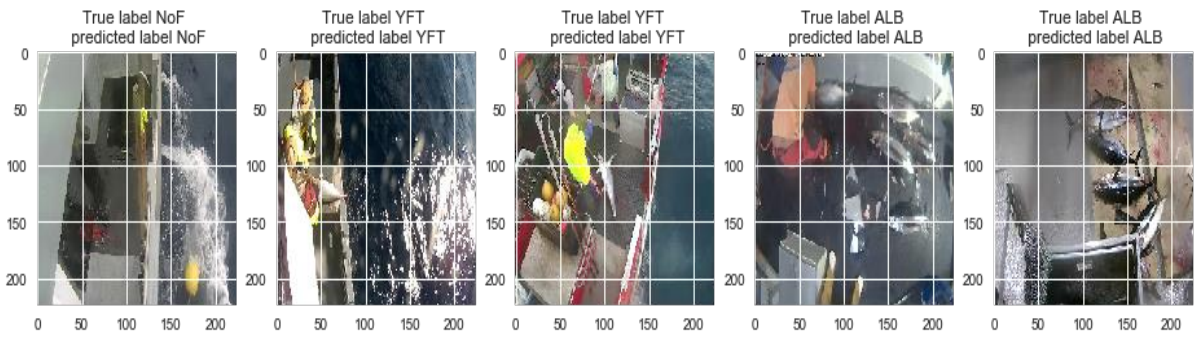
- We can see from the above confusion matrix, that the model does an excellent job in predicting the fish labels ALB (304) and YFT (138). This could be because of the

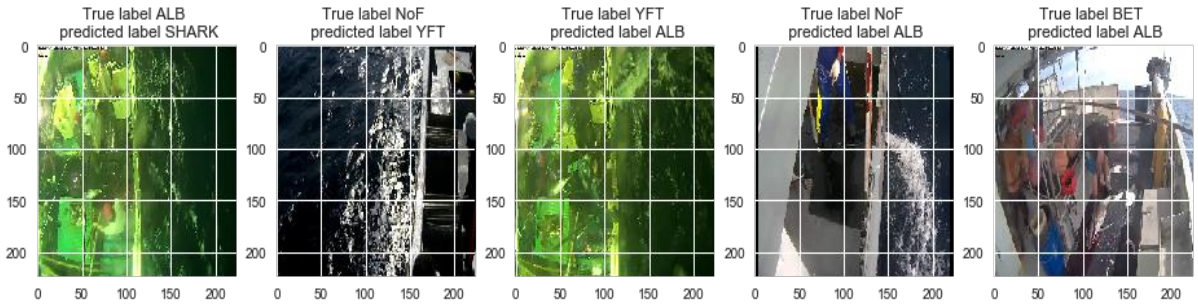
imbalances in the dataset as the training data could have had more images of ALB and YFT fish labels as compared to the other ones.

- A normalized confusion matrix has also been generated and is plotted below.



- We have also displayed some of the correctly and incorrectly predicted fish labels by the model to see if there was any pattern between them.





- We can deduce that the model assigned the class labels YFT and ALB to most of the incorrectly predicted images. This could have occurred because of less amount of training data images for other fish classes.

- The problem can easily be overcome by generating more data for other kinds of fishes.

5.3 K-Nearest Neighbors

5.3.1 Data Preprocessing

- The first step here was to extract raw pixels from the images. The color histogram of the images was built based on the Euclidean distance metric.

- The image labels were then encoded using the LabelEncoder () function.

5.3.2 Splitting the data

- The features were then split into train and test set features, labels were split into train and test set labels, with a sample size of 0.25 and a random state value set to 42.

5.3.3 Model Building and Performance

- The K-Nearest Neighbor classifier was trained with number of neighbors parameter set to 2 as it gave us the best result after training our model for different values of K. We fit our model to the training data and evaluated its performance.

- It gave us a log loss score of 1.51. We were certain a very efficiently developed CNN model could beat this log loss score. So, a good CNN model would be the one having a log loss score less than 1.51.

6. Conclusion

- The CNN model used at the beginning of this report performed the best with a log loss score converging between 0.35 and 0.45. The next better performing model after thar was VGG16 which had a log loss score of 1.3. Finally, the K-Nearest Neighbor model gave us a log loss score of 1.51.

- The log loss score of the VGG16 model can further be improved by performing Data Augmentation techniques. There is also a need to generate data for fishes besides those belonging to Albacore Tuna and Yellowfish Tuna, since majority of the training data samples comprise of these two fish labels.

- Another approach called bounding box can be utilized wherein we determine the precise location of the fish in our images and then classify them. This approach would require better image quality and camera placements in the boat.

7. References

1. CS231n: Convolutional Neural Networks for Visual Recognition. (n.d.). Retrieved November 11, 2020, from https://cs231n.github.io/transfer-learning/

2. Home. (n.d.). Retrieved November 11, 2020, from https://www.fast.ai/

3. Chollet, F. (n.d.). The Keras Blog. Retrieved November 11, 2020, from https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

4. Siddiqui, Shoaib & Salman, Ahmad & Malik, Imran & Shafait, Faisal & Mian, Ajmal & Shortis, Mark & Harvey, Euan. (2017). Automatic fish species classification in underwater videos: Exploiting pretrained deep neural network models to compensate for limited labelled data. ICES Journal of Marine Science. 75. 10.1093/icesjms/fsx109.

5. https://github.com/shawnhan108/nature-notebook/blob/master/The%20Nature%20Conservancy%20Fisheries%20Monitoring/The%20Nature%20Conservancy%20-%20Fisheries%20Monitoring.ipynb

6. https://www.kaggle.com/c/the-nature-conservancy-fisheries-monitoring/data