Machine Learning Engineer Nanodegree

Capstone Project

Haaris Jalal
January 2018

Table of Contents

List of Figures and Tables

Definition

Project Overview

Seafood is one of the main sources of protein for most of the sea neighboring nations. Countries like Japan, South Korea and the Americas are very much into open water fishing. However, excess of fishing including illegal fishing over the years have caused major problems to the marine ecosystem. In addition to overfishing, coastal pollution, habitat destruction, global warming and acidification also contribute to the ongoing destruction of the marine ecosystem.

Fishing activities have altered and led to the destruction of marine ecosystems through both direct and indirect effects. In terms of direct effects, the fisheries remove the results of about 8% of the global primary production in the sea. In most cases, overfishing becomes a serious threat when anglers tend to go for the species with high natural longevity and low reproductive rate. In addition to direct effects, indirect effects can have an important impact on the marine ecosystem as well. Many nearshore ecosystems have altered through destruction of benthic biogenic habitat. Dredging, trawling, long-hauling, and igniting explosives have removed the emergent sessile organisms that provide critical structural habitat on otherwise relatively featureless sea floors.[1]

The Nature Conservancy is one of the organization who are working to protect the wildlife across the globe. One of the areas that they are trying to protect is the marine environment. To combat the problem of overfishing in open waters, the organization decided to incorporate different monitoring devices such as cameras and recording devices on fishing boats. Through these devices, they were able to record footage of anglers and the different species of fish caught on a daily to weekly basis. However, because there are hours of footage that is unsorted, identifying the different species of fish is both time consuming and tedious.[2]

The organization then turned towards the "***Kaggle (www.kaggle.com)***" community to develop an efficient method to classify the different species of fish. Using different machine learning algorithms it will be possible to expedite the classification process with a certain amount of accuracy. In this way, the conservancy can monitor the species that face endangerment.

Datasets for the Project

The Nature Conservancy has provided with a few datasets consisting of training and testing images. For the purpose of this project, we will use the following datasets.[3]

- Train Dataset:
    - The train dataset consists of images classified in eight different folders.
    - The categories are:
        - Albacore Tuna
        - Bigeye Tuna
        - Dolphinfish
        - Moonfish
        - Shark
        - Yellowfin Tuna
        - Other (meaning that there are fish present but not in the above categories)
        - No Fish (meaning that there are no fish in the picture)
- Test Dataset:

- The test dataset consists of all the test images that we need to classify correctly.

Figure 1: Fish Images

Problem Statement

The task is to identify the species of fish in the test data set based on the images given in the train data set. I have decided to go through the strategy of applying deep learning techniques for image classification. The deep learning techniques have been very common in the past few years, as they have outperformed all the previous traditional approaches in many ways.

Two of the methods that are prominent in deep learning are using Dense Neural Networks and Convolutional Neural Networks. In addition to the above two, there is a process called transfer learning. Transfer learning refers to the process of using weights from pre-trained networks on large datasets. The advantage of using pre-trained networks is that the network does not need to train from the beginning. Some of the pre-trained networks are:

- VGG-19
- ResNet-50
- Inception
- Xception

For this project, I will be using the traditional approach as well as the transfer learning approach.

Metrics

For this project, I will be evaluating my model using the multi-class logarithmic loss. The formula for log-loss evaluation is:

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

$where,$

$N\ is\ the\ number\ of\ images\ in\ the\ test\ set.$

$M\ is\ the\ number\ of\ image\ class\ labels.$

$log\ is\ the\ natural\ algorithm.$

$y_{ij}\ is\ 1\ if\ the\ obervation\ i\ belongs\ to\ class\ j\ and\ 0\ otherwise.$

$p_{ij}\ is\ the\ predicted\ probability\ that\ the\ observation\ i\ belongs\ to\ class\ j$

Multiclass logarithmic loss classifier penalizes incorrect predictions. As an example, if the class label is ***"1"*** and the prediction is ***"0"***, then the log-loss will have a high value. On the other hand, if the class label is ***"1"*** and the predicted label is ***"1"***, then the log-loss will have a smaller value. Our main aim is to minimize the log-loss function as much as possible keeping in mind the capability of the processor.

Log-loss takes into account the uncertainty of the prediction based on how much it varies from the actual label. The graphical representation of the range of possible log-loss values given a true representation resembles that of an exponential decay. As the predicted probability approaches "1", log-loss slowly decreases. As the predicted probability decreases, the log-loss increases rapidly. Log-loss is famous for penalizing both types of errors (Type I and Type II) and especially those predictions that are confident and wrong.[4]

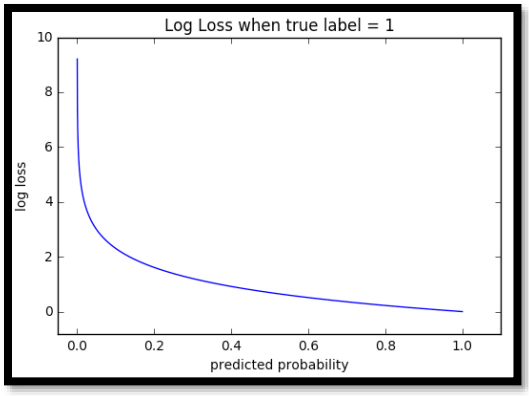


Figure 2: Log Loss Graph

In this project, I will try to reduce the log-loss as much as possible. The reasons and justification for choosing log-loss is that it provides higher punishment for cases where you predict with high confidence but you are wrong. In a business sense, it translates into trying to avoid making errors that might result in large missed financial opportunities. For this project, we needed high penalization. The image needs to classify into one of the eight classes mentioned. If the probability of two or more classes are very near then it will be impossible to place them in the specific class in which they belong. In addition, the competition from where this project is depends on this metric for classification purposes.

Analysis

Data Exploration

To create this dataset, TNC used hours of footage from number of fishing ships and sliced them into many images. The input is a set of raw images so, it is necessary to preprocess the images. Given below is a histogram for the set of different images.

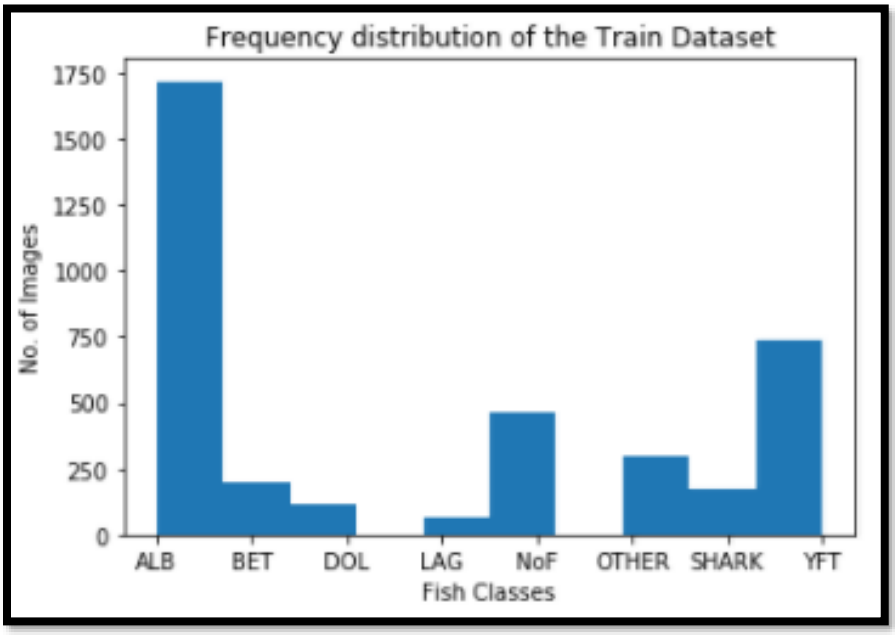


Figure 3: Frequency Table for classes of fish

The Albacore Tuna is the most consumed fish in the marine industry. This is probably the reason why the quantity of this type of fish is maximum among the others.

The dataset is completely categorical in nature. Each image in the training and the test set has only one category of fish. However, small fish are present in some of the pictures as well. These fish were bait for the bigger fish. The images are real life images and any good deep learning algorithm should be able to classify them correctly.

Dataset Features

Total number of images in the train folder: 3777 images

Total number of images in the test folder: 1000 images

Total Classes of fish: Eight classes [ALB, BET, DOL, LAG, NoF, OTHER, SHARK, YFT]

Shape of the image tensor: 224 x 224 x 3

Exploratory Visualization

Once, the dataset is uploaded it will be interesting to see the visualization of the training set and the validation set.

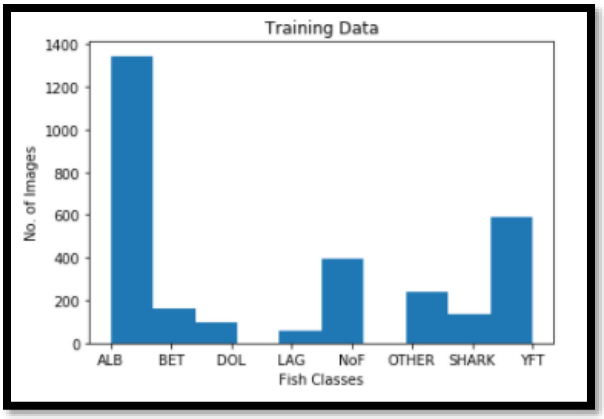


Figure 4: Frequency Table (Train Set)

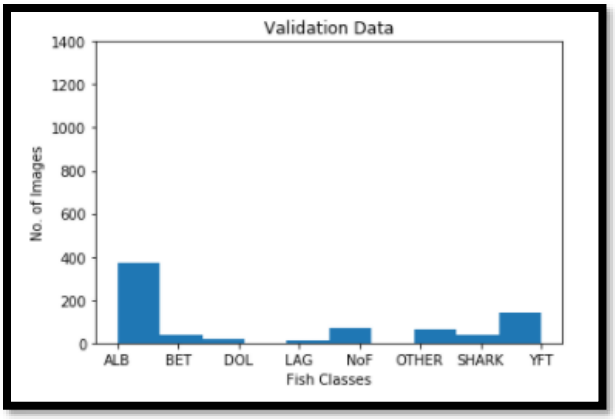


Figure 5: Frequency Table (Validation Set)

The above graphs show the distribution of the training data and the validation data. We will use the train test split for model evaluation and calculating the log-loss ratio for the entire dataset.

Using Color Histograms

Using histograms is a fun way to calculate the RGB ratio of the images. One way for calculation of similarities is using the Minkowski Distance metric. Minkowski distance is a metric in a normed vector space that is a generalization of both the Euclidian Distance and the Manhattan Distance.

The formula for Minkowski Distance is:

$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}}$$

Below are the resultant histograms with their corresponding images. For the report, I used only two images and generated their histograms.

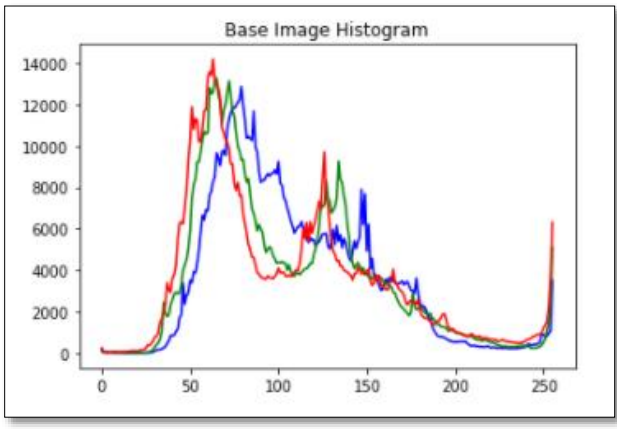


Figure 6: Color Histogram Base Image

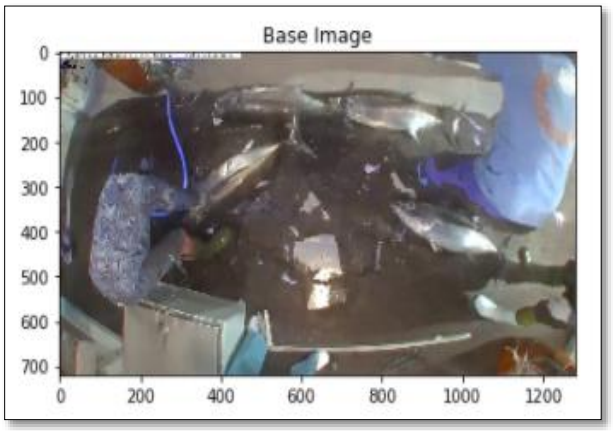


Figure 7: Actual Photo Base Image

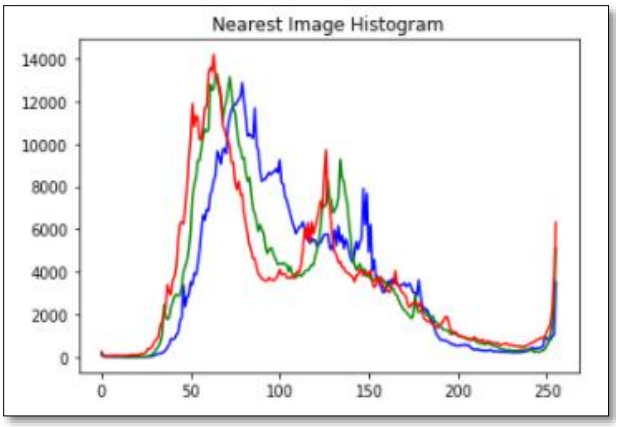


Figure 8: Color Histogram nearest Image

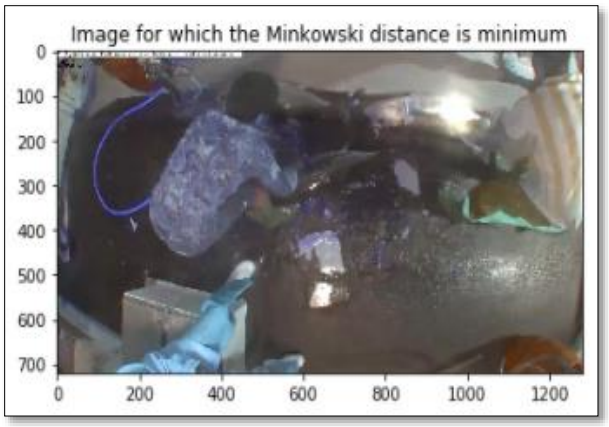


Figure 9: Actual Photo nearest Image

The images and their histograms depict the following:

1. Figure 6 represents the color histogram for figure 7 which is the actual image
2. Figure 8 represents the color histogram for figure 9 which is the nearest image
3. Even though the histograms are the same, the images are different in actuality. Figure 7 shows ***"ALB"*** while figure 9 shows ***"SHARK"***
4. As a conclusion, histograms are not enough for classifying the images properly. One of the reasons is that the histograms do not consider the depth of the object.

Algorithms and Techniques

For classification purposes, there are many different types of algorithms and techniques available. For this project, I will be using Convolutional Neural Networks and Transfer Learning for classification.

Convolutional Neural Network

In machine learning, a convolutional neural network is a class of deep, feed-forward artificial neural networks that help in analyzing visual imagery. The CNNs use different variations and combinations of multilayer perceptrons that require minimal processing. The network learns the filters dynamically and this is a major advantage. They have applications in image recognition, video recognition, recommender systems and natural language processing.[5]

Transfer Learning

Transfer learning is a machine learning method where a model developer for a task is reused as the starting point for a model on the second task.[2] There are two very common approaches to transfer learning.

- Develop Model Approach
- Pre-trained Model Approach

In the develop model approach method, the first step is to select the source task. Generally, we select a related predictive modeling problem with large data in which there is some kind of relationship between the variables. Next, one develops a skillful model that must be better than a naïve model to ensure that feature learning is completed. The model fit on the source task acts as the starting point for a model on the second task of interest. In the end, some fine-tuning gets the refined output from the model.

The pre-trained approach is generally used when the data is not large enough or when the processing time is short. In this approach, select a pre-trained source model from the available models. The pre-trained model then acts as the starting point for a model on the second task of interest. This might involve using all or just a few parts of the model depending on the technique. The model is then tuned so that it may result in the best fit.[2]

Naïve Bayes Model

In machine learning, naïve Bayes classifiers are a family of simple probabilistic classifiers based on Bayes theorem application with strong independence assumptions between the features. It is a simple technique for constructing classifiers. The advantage of Naïve Bayes is that it only requires a small number of training data to estimate the parameters necessary for classification.[3]

I will be using this approach for the benchmark model.

Benchmark

For the benchmark, I selected the Naïve Bayes Algorithm. This is a simple algorithm and it gives satisfactory results. The benchmark value was the value of log-loss. The log-loss from the application of Bayes Model on the training data is:

$$logloss = 19.5228$$

```
In [89]: log_loss(y_valid, predicted)
Out[89]: 19.522768821729727
```

This log-loss value is quite high. My aim would be to reduce this value to less than five. Image flattening is a prerequisite for the Naïve Bayes algorithm. Once complete, Naïve Bayes algorithm conducts the classification of images.

Methodology

Data Preprocessing

We perform preprocessing on the data before any in-depth analysis. Data Preprocessing describes any type of processing performed on raw data to prepare it for another processing procedure. It transforms the data into a format that the user finds easy to work on.[4]

Training a good convolutional network takes quite a bit of time and very good processing requirements. Unfortunately, my laptop processing power is not that high. In this project, I will be training a small convolutional network but then I will be using one of the transfer learning algorithms to try to get better results.

In order to prepare the dataset for implementing the algorithm, I require the following steps:

1. Scale the images according to specific dimensions. In my case, the image was read in the BGR format with col: 224, row: 224, and depth: 3.
2. Once the images are read, it is important to divide them using train_test_split. We divide images into following subsets of the data.
    a. X_train: This contains the training features

b. X_valid: This contains the validation features
c. y_train: This contains the training labels
d. y_valid: This contains the validation labels

3. Once the division is complete, it is important to normalize the features. CNN processes the normalized features in a much faster way. Here I am talking about zero mean centering.

Now the preprocessing of the dataset is complete, it is essential to go towards the implementation.

Implementation

A Convolutional Neural Network (CNN) consists of one or more convolutional layers preceding one or more fully connected layers. Its architecture takes advantage of the 2D structure of an input image. CNN are advantageous over the traditional dense neural networks. They are easier to train and require fewer parameters than dense neural networks in comparison.[6]

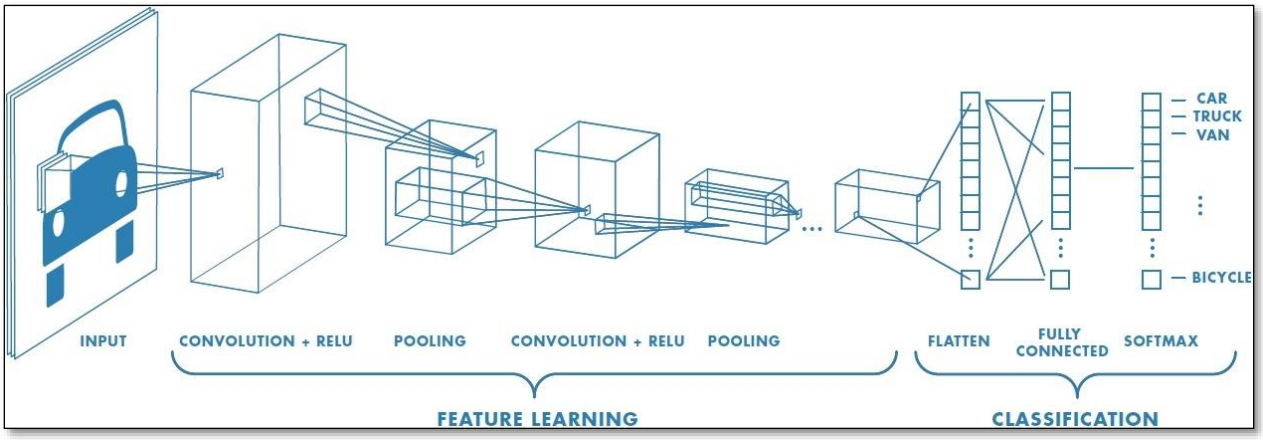


Figure 10: A Typical CNN neural network

A CNN consists of a number of convolutional layers, subsampling layers and optionally fully connected layers. The input to the convolutional layers are the height, width of the image and the number of channels e.g., there are three channels for an RGB format and two channels for a Grayscale format[6].

The developed model consists of a number of convolutional layers, dense layers at the end and the beginning and dropout layers. The number of epochs chosen were five. I wanted to increase my number of epochs but the system performance would go too slow.

Initially, I started out with one epoch and used dropout of 0.5 near the end of the convolution network. However, the log-loss and accuracy for that particular case was,

$$Log\ Loss = 1.4425$$

$$Accuracy = 51\%$$

After the results, I decided that I needed to change and adjust some of the layers. I reduced the number of dropout layers and increased the epochs. The results were quite good and so was the accuracy.

$$Log\ Loss = 0.4577$$

$$Accuracy = 85.05\%$$

The final structure of the CNN model and how it was implemented is available in *appendix A*. Various libraries were chosen for this analysis. The major libraries were keras and sklearn. Both these libraries assisted in calculation of different parameters like log-loss, CV score, image classification, generating the convolutional layers and much more. All the steps involved in creating the model are described in *appendix A*. In addition, the other hyper-parameters are available in *appendix B*. All the methodologies are also present in the structure.

I faced many coding complications during this entire process. For one, I did not know how to use the InceptionV3 module and had to rely on the online help for the analysis. In addition, once I had predicted the probabilities, there was an issue transferring it back into labels. That one was also done using the online help.

The hyper-parameters used for this model are below.

Table 1: Hyper-Parameters for the selected model

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 222, 222, 16)	448
activation_3 (Activation)	(None, 222, 222, 16)	0
conv2d_7 (Conv2D)	(None, 222, 222, 32)	4640
max_pooling2d_5 (MaxPooling2	(None, 111, 111, 32)	0
conv2d_8 (Conv2D)	(None, 111, 111, 64)	18496
max_pooling2d_6 (MaxPooling2	(None, 55, 55, 64)	0
dropout_4 (Dropout)	(None, 55, 55, 64)	0
conv2d_9 (Conv2D)	(None, 55, 55, 128)	73856
max_pooling2d_7 (MaxPooling2	(None, 27, 27, 128)	0
conv2d_10 (Conv2D)	(None, 27, 27, 256)	295168
max_pooling2d_8 (MaxPooling2	(None, 13, 13, 256)	0
dropout_5 (Dropout)	(None, 13, 13, 256)	0
flatten_2 (Flatten)	(None, 43264)	0
dense_3 (Dense)	(None, 8)	346120

Total params: 738,728
Trainable params: 738,728
Non-trainable params: 0

Initially, I had used the dense layer in the beginning and after reading the reviewer's comments, I figured out that I had made an error. The final dense layer needs to be added in the end. This is done so that the purpose of the CNN is not defeated.

The following images are the results for the model trained via neural networks. Out of the four images below, three of the images classify properly. Only one image has an incorrect label.

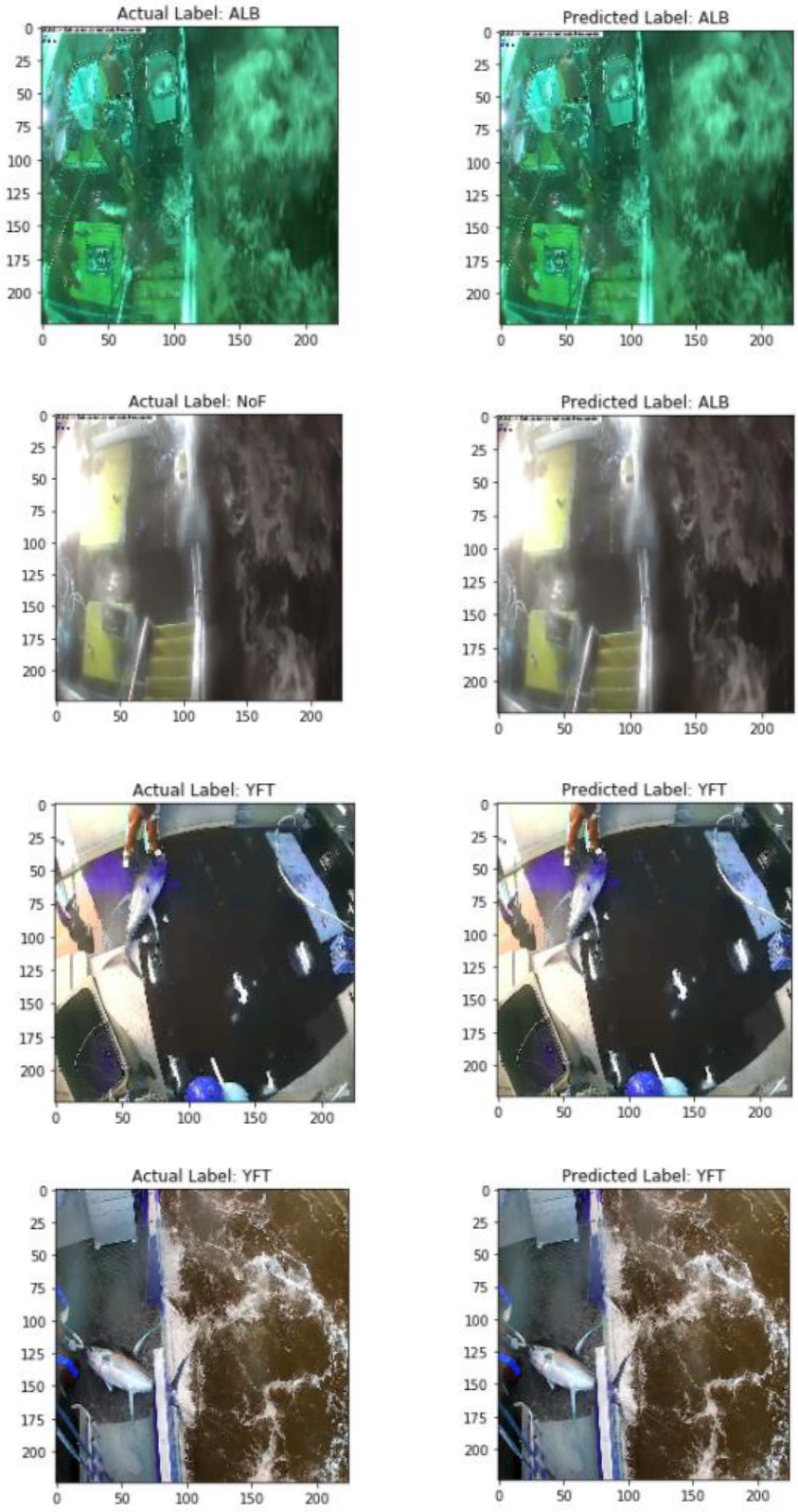


Figure 11: Differences between the Actual Labels and the Predicted Labels

Refinement

Using Transfer Learning for calculation of log-loss

One of the methods used for refinement was transfer learning. Transfer learning allows us not to start from scratch and use one of the pre-trained models to classify the images correctly. For transfer learning, I will be using the inceptionV3 model and using the help as provided in the Keras Documentation page.[7]

InceptionV3 is trained for the ImageNet Large Visual Recognition Challenge using the data from 2012. This is a standard task in computer vision, where models try to classify entire images into 1000 classes, like "Zebra", "Dalmatian", and "Dishwasher".[8]

I wanted to test the code as explained in the original documentation of Keras. I decided to start with the original weights as well. The log-loss through the Inception model did not defeat the log-loss for the initial model. Keeping the original weights allowed me to see where the inception model stood. I decided to use this approach as my initial model actually performed quite well.

For comparison purposes, I ran the pre-trained algorithm. The final log-loss after five iterations came out to be,

$$Log\ Loss = 1.325$$

This log-loss is not as good as the earlier models but it is quite good where less processing power is required. I utilized this method as to compare the results with the initial model. On seeing the results, I decided to go with my original model.

Results

Model Evaluation, Validation and Justification

Different models produced different results. The selection of the model is dependent on the Log-Loss value. The smaller the value, the better the model.

The following were the different log-loss scores for different models used.

Table 2: Comparison Results

Models	**Log – Loss**
Benchmark (Gaussian Naïve Bayes)	19.523
Trained CNN	0.46
InceptionV3 (Transfer Learning)	1.36

For each of the models, the main aim was to get a better log-loss score. If the processing power was not an issue, there was a possibility for implementing data augmentation. There is a chance that the trained CNN might have done small amounts of overfitting. However, the accuracy was less than 90%, which deduces that not much overfitting may have taken place.

To validate the models, I created the confusion matrices in the benchmark and the CNN models with the validation sets.

Confusion Matrix (Benchmark Model)

For the benchmark model, I decided to use the Gaussian Naïve Bayes model. The advantages for this model are:

- These models are simple and easy to use
- If the Naïve Bayes conditional independence assumption holds, it will converge at a faster pace
- It needs less training data
- Highly scalable
- For both binary and multiclass classification problems.

For the confusion matrix, I compared the results between the true labels in the validation set and the predicted labels generated by the model.

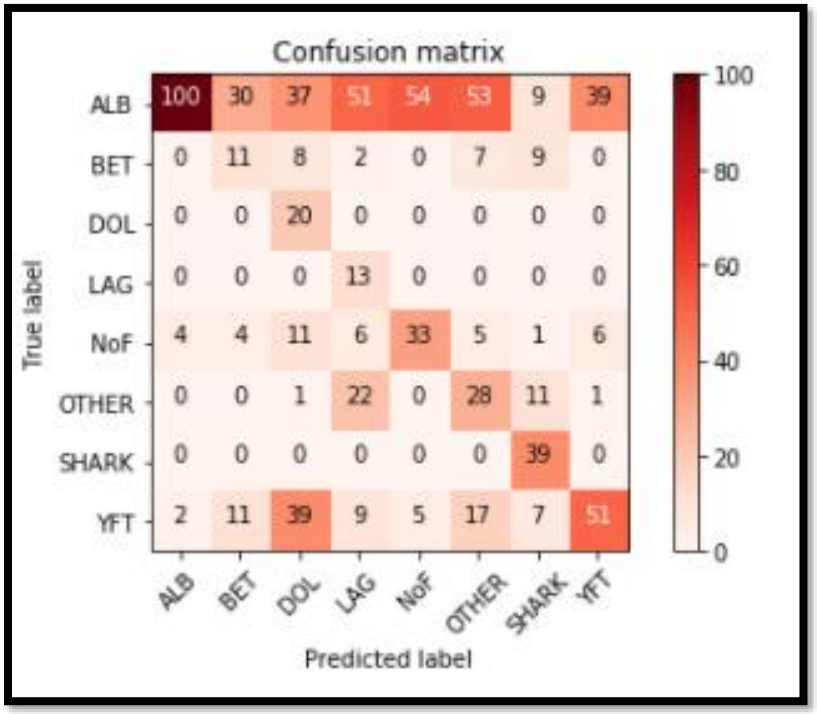


Figure 12: Confusion Matrix - Naive Bayes

This confusion matrix shoes that among the validation set pictures, most did not classify correctly. As an example, the "ALB" category has 373 pictures in actuality and the algorithm predicted only 100 correctly. The error rate for this algorithm in this scenario was quite high. The accuracy for the predictions was,

$$Accuracy = 38\%$$

This model is the benchmark for this experiment. However, I will not recommend using this model for further classification for this dataset.

Confusion Matrix (CNN Model)

The CNN model was the main model for this experiment. I also used transfer learning using InceptionV3 to improve the log-loss. However, the log-loss was better with the traditional CNN approach. Figure 11 shows the correctly classified and the incorrectly classified images through this method.

In order to check for the validity, I created the confusion matrix for the model's output. This confusion matrix shows better prediction compared to the previous confusion matrix.

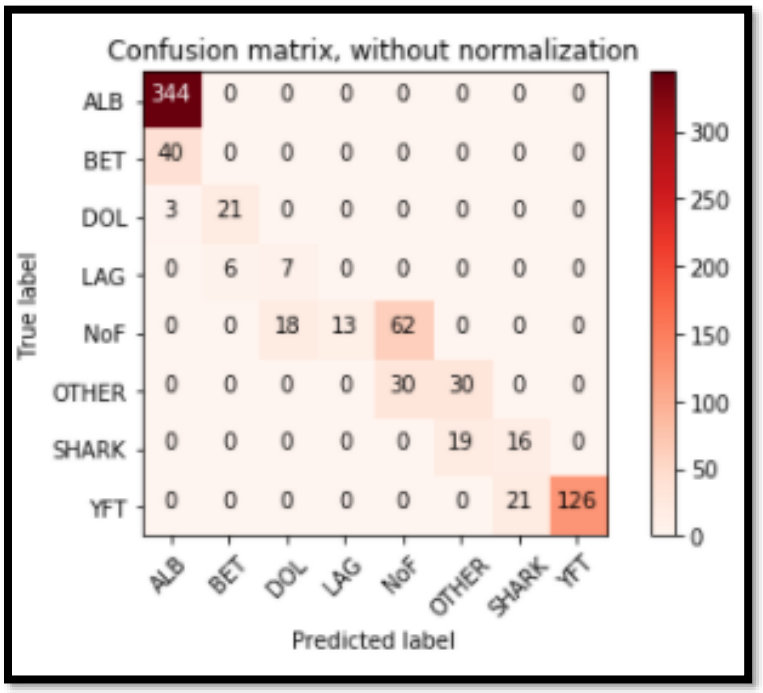


Figure 13: Confusion Matrix - CNN

Compared to the benchmark algorithm, as an example, all the images of "ALB" classify correctly as "ALB". Only 40 images of "BET" and three images of "DOL" wrongly classify as "ALB".

Robustness of the model

The model created by CNN is quite robust and dependable. However, it can be made more robust with additional parameters and epochs. Image augmentation can also help in making the training model better. Image augmentation creates more images and then the model trains on all the images. This is a better approach; however, based on my system's capabilities, the image augmentation would require quite a large amount of time and processing speed. I tried it earlier and the system was suspended.

Table 3: Robustness of the CNN-Model

No. of Iterations	Random State	Log-Loss Score	Accuracy
Iteration 1	42	0.4507	85.05
Iteration 2	80	0.4315	85.714
Iteration 3	15	0.5982	86.5

Average Log-Loss Score = 0.493

Average Accuracy Score = 85.75

In addition, the result from the InceptionV3 model is also quite good. The model's log-loss value is higher than the trained CNN for the iterations and it is robust. It is robust because the log-loss continues to be hovering around the same value. InceptionV3 consists of large number of images in its database. On

adding new images, there is a higher probability of the algorithm performing better than the traditional CNN.

Table 4: Robustness of the InceptionV3 Model

No. of Iterations	Shuffle	Log-Loss Score
Iteration 1	False	1.3643
Iteration 2	True	1.5416
Iteration 3	True	1.4415

Average Log-Loss Score = 1.45

The traditional CNN can have different layers added and removed to incorporate any additional images as well. Given the right processing power, I believe that this model can produce better results than any transfer learning approach. The reason for better performance of this model are:

1. We ourselves will select the images for training the model. This automatically reduces to only the relevant images. On the other hand, transfer learning uses a database with multitude of images from different areas.
2. We can select the number of layers and optimizers in the conventional CNN. This allows us to tweak the results and the process in our favor.

Conclusion

Free form Visualization

One of the most important criteria in this experiment was the log-loss. This log-loss along with the accuracy of the good CNN model is below.

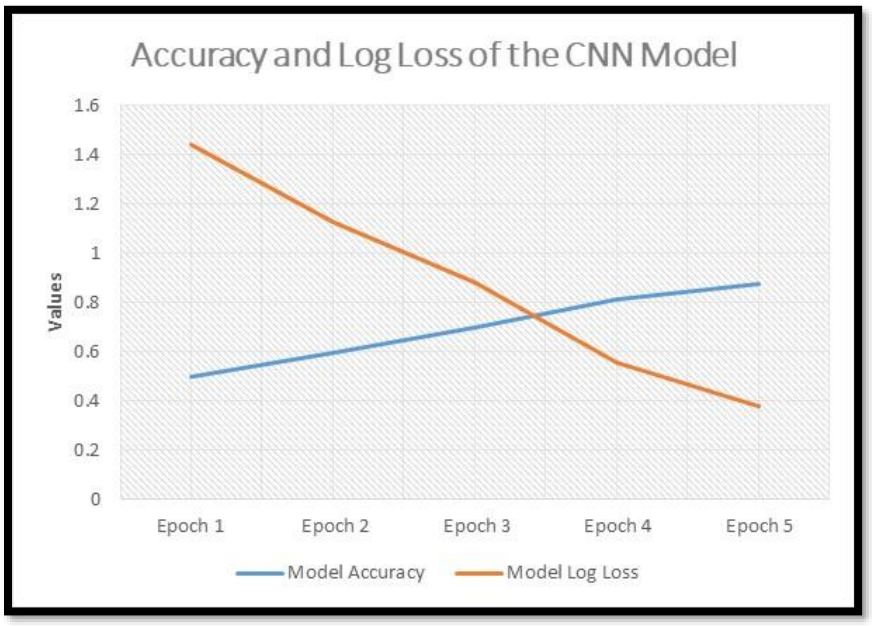


Figure 14: Accuracy and Log-Loss for the CNN Model

The visualization depicts that as the log-loss decreases, the accuracy of the model increases. This is true because in case of Gaussian Naïve Bayes, the accuracy was quite low and the log-loss was quite high.

Reflection

In summary, the problem was to classify the images of fish into separate categories. The metric used for this purpose was the log-loss metric. I started with reading the images into the data and splitting them into RGB format. For the first analysis, I used the color histograms. The concept was that if the Minkowski distances between two histograms were less, then they would be identical. After checking it with the help of two images, it turns out that histograms are not that good a metric for classification.

Afterwards, I decided to use a benchmark model. I decided on using Naïve Bayes because of some advantages as discussed in the previous section. After setting the benchmark model, I trained two neural networks (one of them based on transfer learning) to generate the models that give more than 80% accuracy on the validation dataset.

The log-loss value for both the latter models were quite low.

The problems I faced were plenty for this project. When I had to restart my algorithm, I had to wait for quite a while for the model to run. I could not apply image augmentation to this model as that would increase my dataset size and may cause my computer to crash. I tried my best while working with the CNN models so that they would extract the necessary parameters form the images for proper classification.

The one thing that I found interesting is the vast application of transfer learning and CNNs in everyday life. The vastness of the image databases available from where the algorithms train is fascinating. While researching on this project I came across different algorithms for a viable solution. My major interest is to understand the concepts more deeply so that I can enroll myself in the A.I. nanodegree program soon.

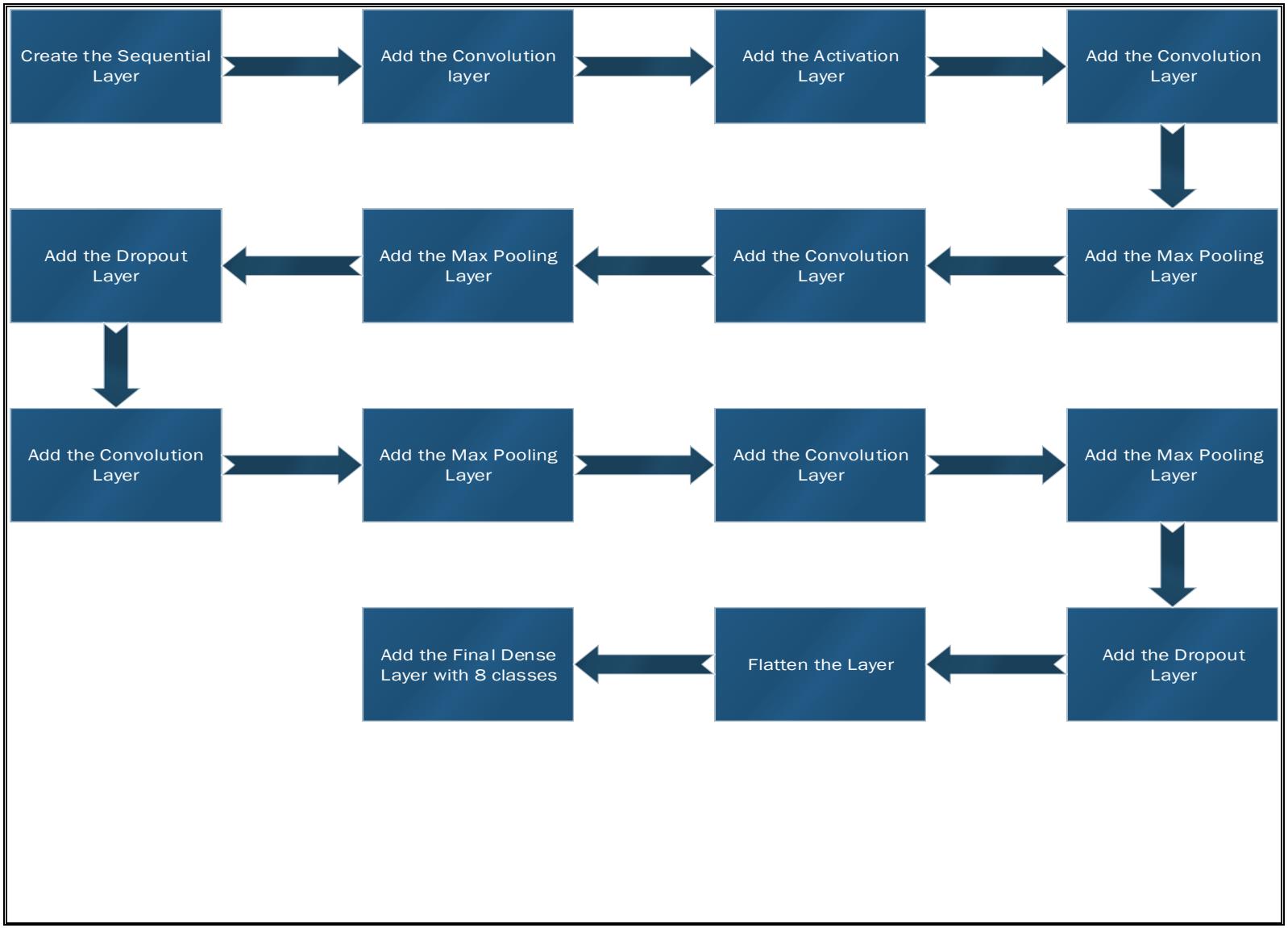
Improvement

There is some level of improvement required for the project. One thing would be to introduce image augmentation. Another technique that I am interested in was reinforcement deep learning. I will want to study about that approach in the next nanodegree that I plan to take.

I am sure that there are many other solutions for better classification. Addition of more convolution layers and less dropouts can make the algorithm more powerful and dependable.

Attaching the submission files with this report as well.

Appendix A: Flowchart for Convolutional Neural Network



Appendix B: Other Hyper-Parameters

Layer (type)	Output Shape	Param #
activation_3 (Activation)	(None, 224, 224, 3)	0
conv2d_5 (Conv2D)	(None, 224, 224, 32)	2432
conv2d_6 (Conv2D)	(None, 224, 224, 32)	25632
max_pooling2d_5 (MaxPooling2	(None, 112, 112, 32)	0
conv2d_7 (Conv2D)	(None, 112, 112, 64)	18496
conv2d_8 (Conv2D)	(None, 112, 112, 64)	36928
max_pooling2d_6 (MaxPooling2	(None, 56, 56, 64)	0
conv2d_9 (Conv2D)	(None, 56, 56, 128)	73856
conv2d_10 (Conv2D)	(None, 56, 56, 128)	147584
max_pooling2d_7 (MaxPooling2	(None, 28, 28, 128)	0
conv2d_11 (Conv2D)	(None, 28, 28, 256)	295168
conv2d_12 (Conv2D)	(None, 28, 28, 256)	590080
max_pooling2d_8 (MaxPooling2	(None, 14, 14, 256)	0
flatten_2 (Flatten)	(None, 50176)	0
dense_4 (Dense)	(None, 256)	12845312
dropout_4 (Dropout)	(None, 256)	0
dense_5 (Dense)	(None, 64)	16448
dropout_5 (Dropout)	(None, 64)	0
dense_6 (Dense)	(None, 8)	520
activation_4 (Activation)	(None, 8)	0

Total params: 14,052,456
Trainable params: 14,052,456
Non-trainable params: 0

References

1. Botsford, L.W., J.C. Castilla, and C.H. Peterson, *The Management of Fisheries and Marine Ecosystems.*
2. Brownlee, J. *A Gentle Introduction to Transfer Learning for Deep Learning*. 2017 [cited 2018 January 16]; Available from: https://machinelearningmastery.com/transfer-learning-for-deep-learning/.
3. *Naive Bayes classifier - Wikipedia.*
4. Rouse, M., *data preprocessing.*
5. *Convolutional Neural Network*. Available from: https://en.wikipedia.org/wiki/Convolutional_neural_network.
6. Tutorial, U. *Convolutional Neural Network*. Available from: ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/.
7. *Keras Documentation.*
8. *Tensorflow Documentation.*