Machine Learning Engineer Nanodegree

Capstone Project

Girish Sharma
September 9th, 2017

I. Definition

Project Overview

Nearly 50% of the world depends on seafood for their main source of protein. In the Western and Central Pacific, where 60% of the world's tuna is caught, illegal, unreported, and unregulated fishing practices are threatening marine ecosystems, global seafood supplies and local livelihoods. The Nature Conservancy is working with local, regional and global partners to preserve this fishery for the future.

The Conservancy is looking to the future by using cameras to dramatically scale the monitoring of fishing activities to fill critical science and compliance monitoring data gaps. Although these electronic monitoring systems work well and are ready for wider deployment, the amount of raw data produced is cumbersome and expensive to process manually.

The Nature Conservancy starts a competition on Kaggle to develop algorithms to automatically detect and classify species of tunas, sharks and more that fishing boats catch, which will accelerate the video review process. Faster review and more reliable data will enable countries to reallocate human capital to management and enforcement activities which will have a positive impact on conservation and our planet.

Datasets

For this competition we are given 3777 images for training and 13153 images for testing captured from boat cameras of various angles.

Eight target categories are available in the dataset
1. Albacore tuna
2. Bigeye tuna
3. Yellowfin tuna
4. Mahi Mahi
5. Opah

6. Sharks
7. Others (meaning that there are fish present but not in the above categories)
8. No Fish (meaning no fish in the picture).

The dataset was compiled by The Nature Conservancy in partnership with Satlink, Archipelago Marine Research, the Pacific Community, the Solomon Islands Ministry of Fisheries and Marine Resources, the Australia Fisheries Management Authority, and the governments of New Caledonia and Palau.

Problem Statement

For this competition, we have to detect which species of fish appears on the boat, based on images captured from boat cameras of various angles.
Our goal is to predict the likelihood of fish species in each picture from the given Eight categories:

1. ALB - Albacore tuna
2. BET - Bigeye tuna
3. YFT - Yellowfin tuna
4. DOL - Mahi Mahi
5. LAG - Opah
6. SHARK - Sharks
7. OTHER - Others (meaning that there are fish but not from the above categories)
8. NoF - No Fish (meaning no fish in the image)

Metrics

The Evaluation Metrics used for this Kaggle competition is multi class logarithmic loss.

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

Where N is the number of images in the test set, M is the number of image class labels, log is the natural logarithm, y_{ij} is 1 if observation i belongs to class j and 0 otherwise, and p_{ij} is the predicted probability that observation i belongs to class j.

The submitted probabilities for a given image are not required to sum to one because they are scaled prior to being scored (each row is divided by the row sum). In order to avoid the extremes of the log function, predicted probabilities are replaced with $max(min(p,\ 1-10^{-15}),\ 10^{-15})$.

II. Analysis

Data Exploration

The dataset was compiled by The Nature Conservancy consisting of 3777 training images and 13153 testing images captured from boat cameras of various angles.

The labels included 6 species of fish as well as one “No fish” and one “Other” label.

Picture describing various classes of Fish for this project

There are several challenges present with this dataset like all images are of differed in size and had been taken at different times of day.

Image taken in daylight

Image taken at Night

Also there are some images in different categories which looks exactly the same that even a human observer could found it difficult to differentiate those images..

Exploratory Visualization

1. There are lots of images with significant variations in the color intensities as the images were taken from different boats, different angles and at the different times of the day. There are only few thousand images available for the training and this can be a difficult task to classify these images correctly.

The Color images have 3 channels R-Red, G-Green, B-Blue. The histograms below provides the color distribution of the image. Red, Green and Blue line represents the respective channels.

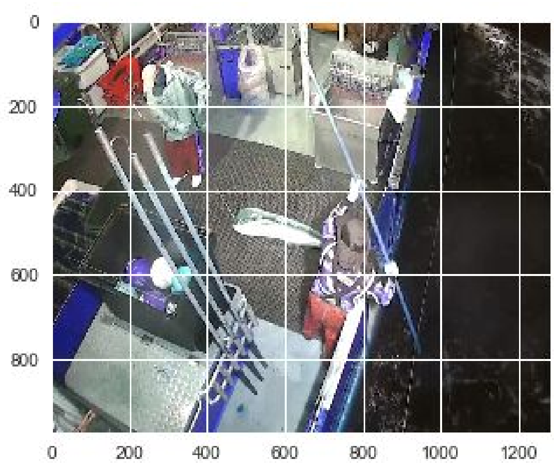
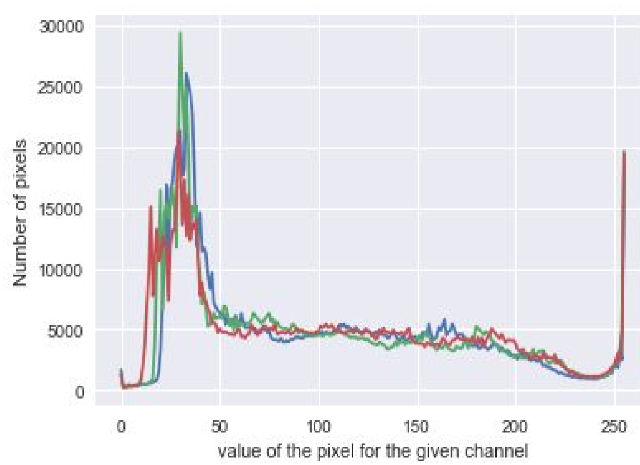
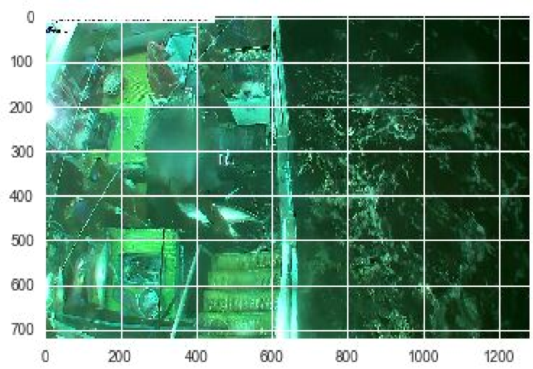


Image Size - (974,1280)





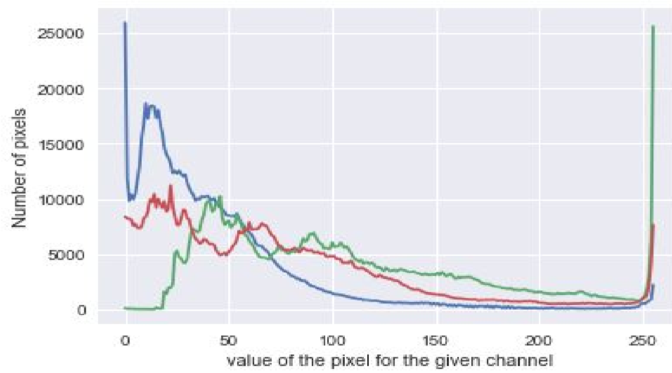
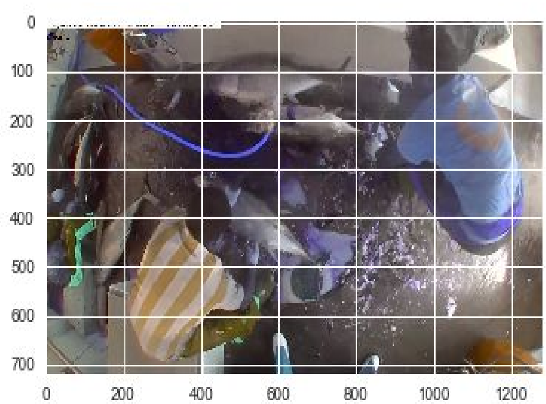


Image Size - (720,1280)



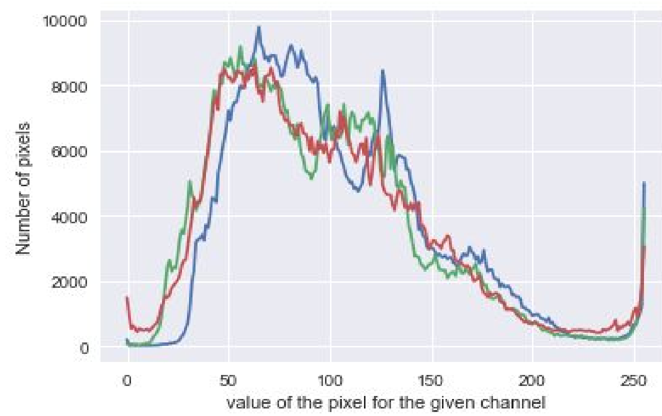


Image Size - (720,1280)

2. It can be seen from the pictures that fish is present in only a small part of the image and the rest is useless stuff which we do not need to classify the fish.

3. The training data provided for this competition was not distributed uniformly among the different classes of the fishes.

Class of Fish	Number of samples
ALB	1719
BET	200
DOL	117
LAG	67
NoF	465
OTHER	299
SHARK	176
YFT	734

Table: - No of Fishes in each class of training set

It may be problematic, as we have a large number of samples from 'ALB' class and very few samples from 'LAG' class the model may find it difficult to make a correct prediction for the later class and predict an image as 'ALB' more often.

Algorithms and Techniques

1. **Deep Learning** - Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms.

2. **Convolutional neural network** - In machine learning, a convolutional neural network (CNN or ConvNet) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery. A CNN consists of an input and an output layer, as well as multiple hidden layers. The hidden layers are either convolutional, pooling or fully connected. We give CNN an input and it learns by itself that what features it has to detect. We won't specify the initial values of features or what kind of patterns it has to detect.

   Various Layers:-

   - **Convolutional** - Also referred to as Conv. layer, it forms the basis of the CNN and performs the core operations of training and consequently firing the neurons of the network. It performs the convolutional operation over the input.

   - **Pooling layers** -Pooling layers reduce the spatial dimensions (Width x Height) of the input Volume for the next Convolutional Layer. It does not affect the depth dimension of the Volume.

   - **Fully connected layer** - The fully connected or Dense layer is configured exactly the way its name implies. It is fully connected with the output of the previous layer. Fully connected layers are typically used in the last stages of the CNN to connected to the output layer and construct the desired number of outputs.

   - **Dropout layer** - Dropout is a regularization technique for reducing overfitting in neural networks by preventing complex co-adaptations on training data. It is a very efficient way of performing model averaging with neural networks. The term "dropout" refers to dropping out units (both hidden and visible) in a neural network.

   - **Flatten** - Flattens the output of the convolutional layers to feed into the Dense layers.

3. **Activation Functions -** In CNN, the activation function of a node defines the output of that node given an input or set of inputs. Some activation functions are:

    - **Softmax** - The softmax function squashes the output of each unit to be between 0 and 1, just like a sigmoid function. It also divides each output such that the total sum of the outputs is equal to 1.

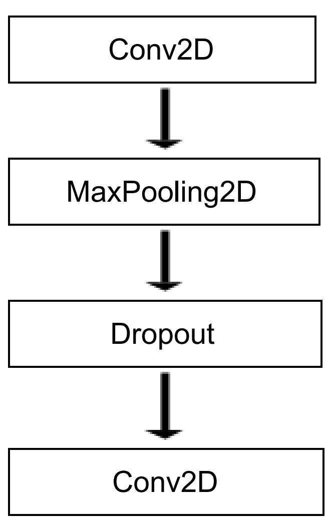
    - **ReLu** - A ReLu (or rectified linear unit) has output 0 if the input is less than 0, and raw output otherwise. i.e, if the input is greater than 0, the output is equal to the input.

4. **Transfer Learning** - In transfer learning, we take the learned understanding and pass it to a new deep learning model. We take a pre-trained neural network and adapt it to a new neural network with different dataset.
   For this problem we use VGG-19 neural network.
    - **VGG-19** - This is a 19-layer network used by the VGG team in the ILSVRC-2014 competition.

Benchmark

1. For this competition, the benchmark score provided by the Kaggle is 1.68824. It is multi-class log score and anything less than this score is considered better than the benchmark.
2. A CNN made from scratch is also used as a benchmark model for measuring the performance of transfer learning approach.

   Model architecture for Benchmark:-



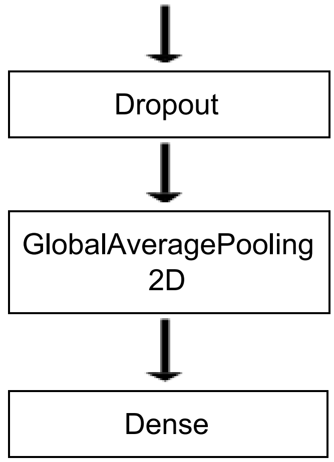


fig:- Benchmark model Architecture

- **Conv2D** - Two Conv2D layers were used, the first layer has 16 and the second layer has 32 filters. Kernel size and strides were set to 2 and 1 respectively and both of the layers use a 'relu' activation function.

- **MaxPooling2D** - One MaxPooling2D layer with pool size and strides both equal to 2.

- **GlobalAveragePooling2D** - One GlobalAveragePooling2D layer was used before the Dense layer.

- **Dropout** - Two dropout layers both having the value of 0.3.

- **Dense** - One fully connected dense layer with 8 nodes (equal to the number of classes of fish) and 'softmax' activation function was used at the end.

After defining the model architecture It was trained on the training set with validation split of 20% and the best weights were saved during the training process. After training, predictions were made on the test set and the Benchmark model archives a multi-class log loss score of **2.00267**.

III. Methodology

Data Preprocessing

When using Keras CNNs, it requires a 4D array (or 4d tensor) as input, with shape **(nb_samples, rows, columns, channels)**, where nb_samples is the total number of sample (images), and rows, columns, and channels correspond to the number of rows, columns, and channels for each image, respectively.

The input images provided in the dataset differed in their dimensions so we need to resize them and convert them into a 4-D tensor. The path_to_tensor function in the script takes a string valued file path to a color image as input and returns a 4-D tensor which is suitable for supplying to a Keras CNN. The function loads the image and first resize it to 224 x 224 pixels. Then, the image is converted to an array which is then resized to a 4-D tensor.

Each tensor corresponds to an image and have a shape (1, 244, 244, 3).
Where nb_samples - 1, rows - 224, columns - 224, channels - 3 (color images have 3 channels, for Red, Green and Blue color).

Implementation

1. First, an initial **benchmark CNN** model architecture was defined and then trained on the pre-processed training data using validation split of 0.2. The training data is divided into batches of 20 images and the best weights are saved using checkpointer.Then the labels of the test images were predicted and converted into a .csv file for submission.
The benchmark model achieves a Multi-class log loss score of **2.00267**.

2. For Model 2 the **transfer learning** was used to create a CNN using **VGG-19** bottleneck features. VGG-19 architecture without the last fully connected layer was used to extract the features from the images. The extracted features were fed to a small fully connected model and predictions were made. It takes around 5-6 hours for extracting the features from the images and making the predictions.
The model did not perform well and get a score of **2.2886**.

3. I tried to reduce the number of layers in VGG-19 by removing 11 layers from the end as I thought that it may be a case where Model 2 was giving more emphasis to the other details than the fish in the image. When I try to extract the features of the images this time my machine ran out of memory and gives a **Memory Error** after running the process for around 3 hours.

4. After the failing of the Model which uses transfer learning and which I supposed will be the solution to this problem I shifted back to make a better model from scratch.
This time I used a combination of Conv2D, MaxPooling, Dropout, GlobalMaxPooling and Dense layers and able to get a Multi-class log loss score of **1.65209.**

Given below is the architecture of my final model.

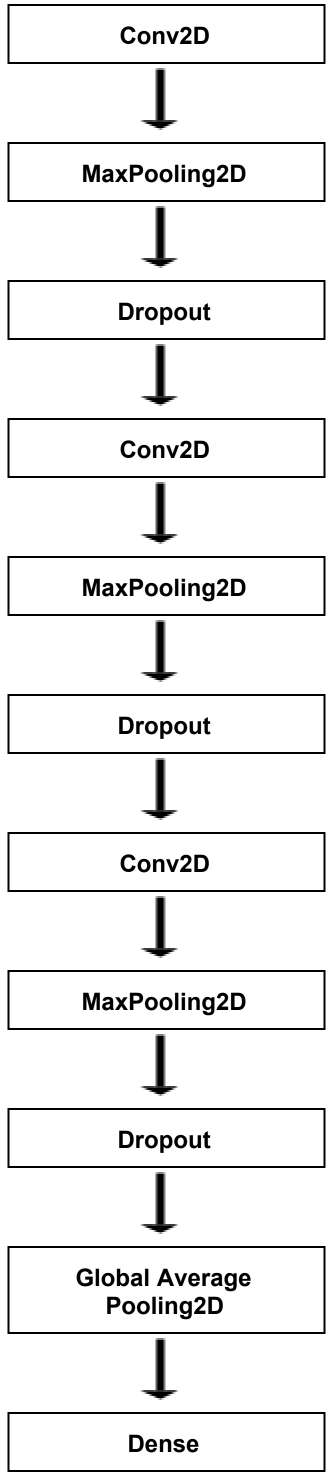


Fig. Architecture of Final Model

Refinement

1. Model4 - Final model shows a significant improvement when compared with the benchmark model. I use 3 Conv layers with increasing number of filters from 16 in first, 32 in second to 64 filters in the last conv layer. After each Conv and MaxPooling, a Dropout layer with probability 0.2 was inserted. This model was able to achieve a multi-class log loss score of **1.65209.**

2. Model5 - Then, I double the number of filters in each Conv layer and increase the value of first and second dropout layer so that it takes less time for training. I train this model for only 5 epochs and able to get a score of **1.56079**.

3. Model6 - Seeing this improvement, I further try to improve my model, this time changing the value of all the dropout layers back to 0.2 and using 'rmsprop' optimizer instead of 'adam'. I train this model for 10 epochs and get a multi-class log loss score of **1.51518**.

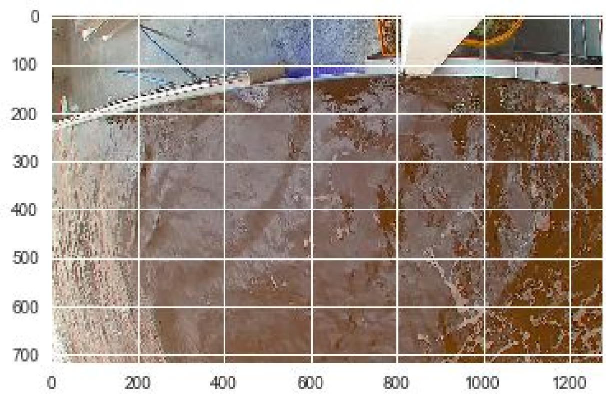
IV. Results

Model Evaluation and Validation

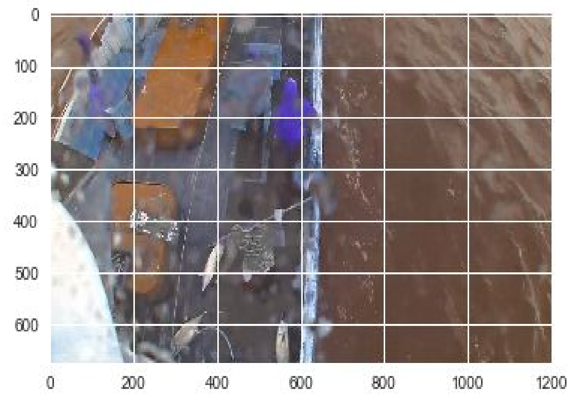
The final model outperforms the benchmarks and other models used in this project. It takes nearly half of the time taken by the model which uses transfer learning approach (with extracted VGG-19 features), and make better predictions.

The parameters of this model were carefully adjusted so that it can make better predictions. It can be reflected by the multi-class log loss score achieved by the model.

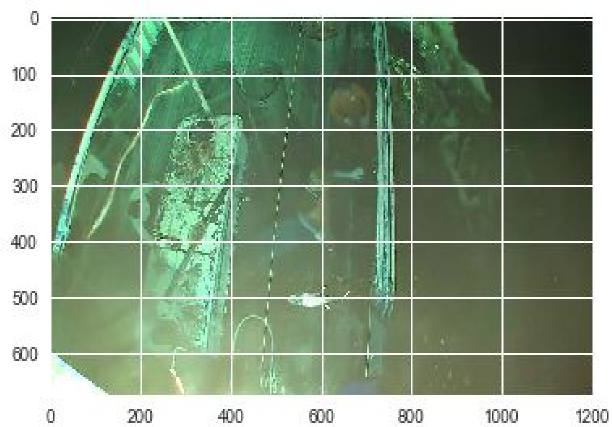
For this competition, Kaggle provides a very large testing set with 13153 images in it. All the images are captured from different boat cameras at different angles and at different times of the day. The dimensions and the quality of the images differed significantly.



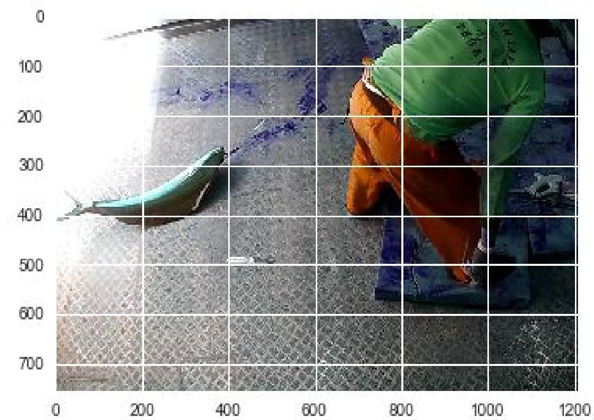
Size - (718, 1276)



Size - (675, 1200)



Size - (675, 1200)



Size - (755, 1208)

Predictions are made on the testing set and then submitted to Kaggle. The score provided by Kaggle for the submission was considered as the norm for measuring the robustness of a model.

Justification

The final model performs better than the benchmark and other models as well. Where benchmark model gets a multiclass log loss score of 2.00267, the final model made a significant improvement of 0.5 (approx) and get a score of 1.51518.

Scores of different Models used in the project:-

Model	Multi-class log loss Score
Benchmark Model	2.00267
Extracted VGG-19 features	2.28866
Extracted VGG-19 features (last 11 layers removed)	MEMORY ERROR
Model4	1.65209
Model5	1.56079
Model6	1.51518

Table:- Multiclass log loss Score of each model

The model does not expect any image of a particular size or taken from a particular angle, it can automatically resize the image and extract the features from it to make its predictions.
It was able to get a decent score when we submit the predictions made by the model on the provided test data set, given that the test dataset has a large number of images of different sizes, taken from different angles at different times of day, and of different quality.

V. Conclusion

Free-Form Visualization

The data provided for this project are images taken from the boat camera at different times of day and from different angles. The images differ in their sizes and quality, some images are captured in very poor lighting conditions and some pictures are very bright and clear. The training images are not distributed uniformly among the different classes, 'ALB' class has nearly half of the images of the whole training set.

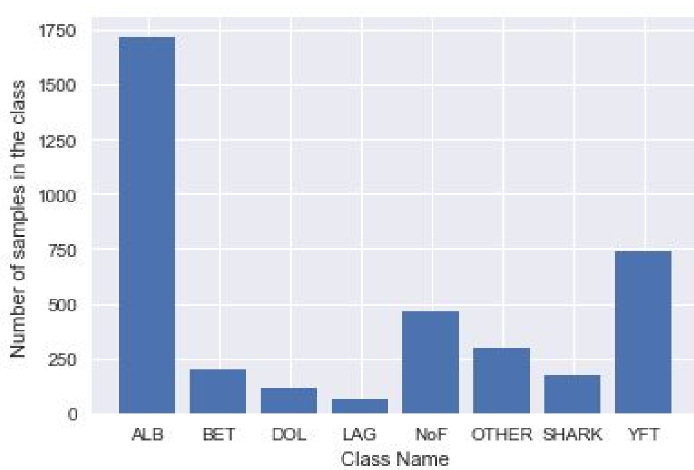


Fig. Shows number of training samples in each class

As we have a large number of samples from 'ALB' class and very few samples from 'LAG' class the model may find it difficult to make a correct prediction for the later class and predict an image as 'ALB' more often.

Reflection

First I build a benchmark architecture from scratch and then try to improve my results using transfer learning technique with the VGG-19 model. Initially, when I proposed this project I thought that transfer learning was really easy to implement and always outperform the other methods, but as I proceed through my project I got to know that I was wrong. It is not easy to apply Transfer learning approach to any solution until you have a clear understanding of it and a lots of hands on practice. It is computationally very expensive to train big models on CPUs and requires GPUs which can run several processes parallelly.

Then I shifted back from transfer learning to create my own model architecture and made improvements to it by fine tuning the parameters.

The final model performs better than my benchmark model as well as better than Kaggle's benchmark for this competition and now I have a better understanding of some concepts about CNN and deep learning and definitely, I will look forward doing some more projects to learn more about them.

Improvement

This was my first experience using CNN and I found that they require a lot more computing power than any traditional approaches and we really need to take care of our memory consumptions, this is really a take away from the project as I ran out of memory several times.

The transfer learning part, as well as the final model, could be improved significantly if I use a more powerful machine and run some more experiments using various architectures. Data argumentation could also be used to increase the number of training samples and the model could be trained for more numbers of epochs.

I still think that transfer learning approach can outperform my final model. Maybe other architectures like VGG-16 and Resnet-50 with carefully adjusted parameters by experimentation can produce a better multi-class log loss score.

References

- https://www.kaggle.com/c/the-nature-conservancy-fisheries-monitoring/submissions?sortBy=date&group=all&page=1
- https://www.conserveca.org/
- https://www.nature.org/
- http://cs231n.github.io/convolutional-networks/
- https://en.wikipedia.org/wiki/Dropout_(neural_networks)
- http://cs231n.github.io/transfer-learning/
- https://www.analyticsvidhya.com/blog/2016/10/tutorial-optimizing-neural-networks-using-keras-with-image-recognition-case-study/
- https://keras.io/
- Udacity's MLND - Deep Neural Networks, Convolutional Neural Network
- https://www.oreilly.com/ideas/the-hard-thing-about-deep-learning
- Google