Humpback Whale Identification Challenge using Convolutional Neural Networks

Pooja Kudav, Piyush Mahatkar

Department of Computer Science, The Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas

Abstract: Identifying Humpback whale based on its fluke can be an easy task, given their anatomy and features. But identifying a type of whale amongst thousands of its kind is very challenging. Moreover, we need to find its identifiable characteristics using a machine learning computer program. In this competition, we are challenged to build a model which identifies the whale based on the photograph of its fluke. The happy whale database has total of 9850 train images and 15000 test images. Total classifications of whales are 4250 (whale id's provided). What makes this such a challenge is that there are only a few examples for each of 4,000+ whale-Ids.

***Index Terms*—Convolutional Neural Networks, Classification, VGG-16, Deep learning, Transfer Learning.**

I. Introduction

A lot of efforts have been taken since a long time to recover the whale population and their preservation. Even after these efforts it has been a has a tough time adapting to warming oceans and efforts to compete daily with the fishing industry for food.

Steps are being taken to preserve whales population. scientists are using photo surveillance systems to keep track of the ocean activity. They use the shape of whale flukes and unique identification spots found in images to identify species of whales they're analyzing. Since last 40 years, this work was being done manually.

The challenge here is to build a model to detect and analyse the whale species in images. The HappyWhale database of 25,000+ images is collected from variety of sources. This algorithm is going to help research team to open new understanding for marine whale population dynamics.

This project will help people and researchers to identify the whale as soon as the picture is taken.

II. Dataset Description

Training data consists of 4251 classes out of one class is "new_whale" which is a default class. The whales which don't belong in any of the 4250 classes would be classified as new whale. There is imbalance in the amount of images per class. It varies from 1 to 810 images per class. Considering the computing requirements we have considered 30 most significant classes from the training dataset. The bar graph in fig 3.a shows the class imbalance in graphic form. A careful analysis of the dataset lead to the conclusion that data augmentation would play an important part in training the model.

III. Related works

A. Right whale Recognition

This challenge on Kaggle took place in the year of 2015. The winner of the challenge deepsense.ai team used Convolutional Neural networks in their solution. They could come up with an efficient model with very little training data which was provided.

They used image augmentation strategies such as rotation, rescaling, color perturbation. Apart from that they used manual approach to achieve good quality passport size photos by manually assigning the whales in images of the training data a bounding box around their heads.

They built the model by:

1. The used CNNs to build head localizer
2. Also built a head aligner using CNN
3. Trained the CNNs on passport like photos of whales.
4. Then they average and tuned their predictions.

They trained their model for 500 epochs and it took them hours of training. Thus building an award winning model.

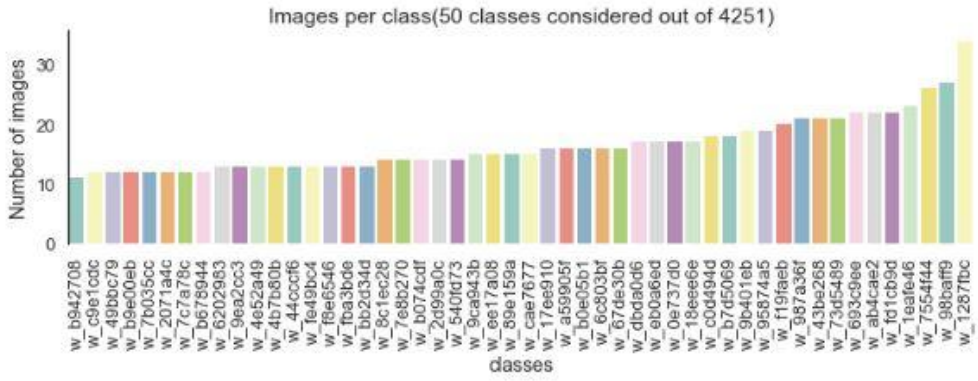

Fig 3.a: The above bar graph the distribution of images per class considering top 50 classes.

IV. Preprocessing Techniques

We dropped the "new_whale" class as it was not a specific class. The images which don't belong to any of the classes are supposed to be classified as new whale .We used libraries such as keras, opencv2 and skiimage for preprocessing. We scaled the number of images of each class to 60 by performing image augmentation.

We have used a few image augmentation techniques like:

1. Rotation- rotated images at 10 degree angle.

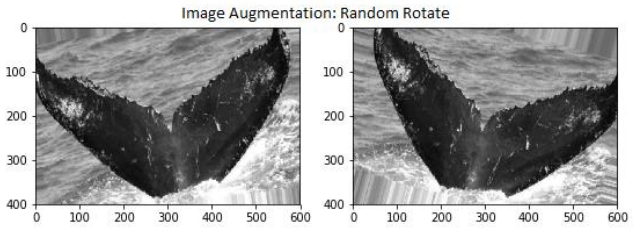

Fig 4.a Rotation

2. Horizontal flip- flipped the image on the middle axis.

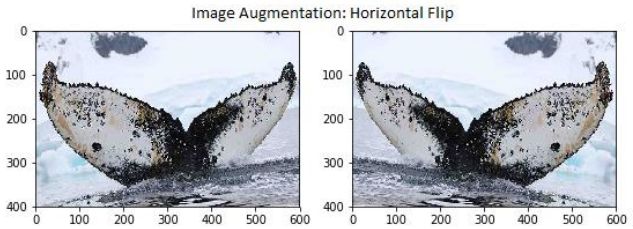

Fig 4.b Flipping

3. Zoom- zoomed in and out the images

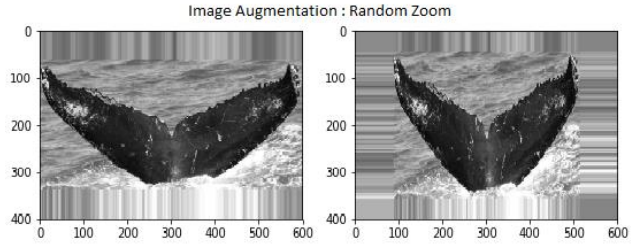

Fig 4.c Zooming

4. Shift

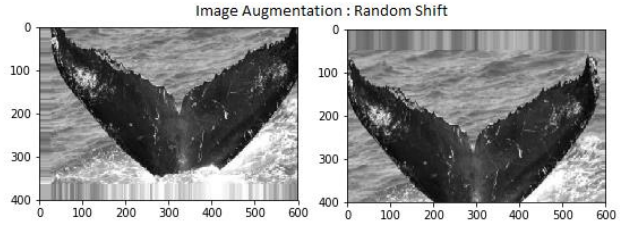

Fig 4.d Shifting

Other filters like Gaussian Blur, Adaptive thresholding and image segmentation were implemented but these filters did not provide expected increase in accuracy.

V. Proposed Solution

Convolutional Neural Nets (CNNs):

CNNs are made up of neurons with certain weights and biases. They differentiate the input array, part by part. Parts can be a sub-matrix of original matrix. These parts can also be called as features. It finds differentiated features by roughly matching the positions in two images, these models get very efficient and stable by finding the similarities than complete matrix matching schemes.

Pooling:

Pooling technique takes large images and shrink them down while preserving the most relevant information in them. We are using Maxpooling for our model. Max pooling select a small window of the array and finds the maximum value to assign to the output. Generally, the window size is 2-3 pixels.

ReLU:
The activation function used in this model is ReLU. Relu stands for rectified linear unit function. This layer reassigns a number, so that the values are consistent with the model. For a negative number it assigns a value of 0 and for positive number it assigns 1. This helps the CNN model to keep learning values and weights within the specified range, rather than getting blown to infinity.

Softmax layer:
This is an activation function which is generally used as the last layer in training models. This function returns the probabilities of each class, for any input data. The output is in the form of float array values between 0 and 1.

Dropout:
This layer selects some neurons randomly and temporarily deactivates the neuron for a specific training phase. Their contribution in training is temporally stopped for the forward pass, thus no weight update is done for those neurons during the backward pass.

Keras:
Keras is a high-level neural networks API, written in Python and capable of running on top of Tensorflow, CNTK and Theano.

There are various pre-trained models like VGG-16, ResNet, Inception etc. we have used the VGG-16 model which is pre-trained on ImageNet weights and added our layers at the end.

Transfer Learning:
Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.

Proposed Model:

Layer (type)	Output Shape	Param #
vgg16 (Model)	(None, 4, 4, 512)	14714688
flatten_11 (Flatten)	(None, 8192)	0
dense_21 (Dense)	(None, 512)	4194816
dropout_11 (Dropout)	(None, 512)	0
dense_22 (Dense)	(None, 30)	15390

Total params: 18,924,894
Trainable params: 4,210,206
Non-trainable params: 14,714,688

Fig.5.a Proposed Model

VGG-16 model summary:

Layer (type)	Output Shape	Param #
input_11 (InputLayer)	(None, 128, 128, 3)	0
block1_conv1 (Conv2D)	(None, 128, 128, 64)	1792
block1_conv2 (Conv2D)	(None, 128, 128, 64)	36928
block1_pool (MaxPooling2D)	(None, 64, 64, 64)	0
block2_conv1 (Conv2D)	(None, 64, 64, 128)	73856
block2_conv2 (Conv2D)	(None, 64, 64, 128)	147584
block2_pool (MaxPooling2D)	(None, 32, 32, 128)	0
block3_conv1 (Conv2D)	(None, 32, 32, 256)	295168
block3_conv2 (Conv2D)	(None, 32, 32, 256)	590080
block3_conv3 (Conv2D)	(None, 32, 32, 256)	590080
block3_pool (MaxPooling2D)	(None, 16, 16, 256)	0
block4_conv1 (Conv2D)	(None, 16, 16, 512)	1180160
block4_conv2 (Conv2D)	(None, 16, 16, 512)	2359808
block4_conv3 (Conv2D)	(None, 16, 16, 512)	2359808
block4_pool (MaxPooling2D)	(None, 8, 8, 512)	0
block5_conv1 (Conv2D)	(None, 8, 8, 512)	2359808
block5_conv2 (Conv2D)	(None, 8, 8, 512)	2359808
block5_conv3 (Conv2D)	(None, 8, 8, 512)	2359808
block5_pool (MaxPooling2D)	(None, 4, 4, 512)	0

Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0

Fig 5.b: Model Description of VGG-16 model without top layers

VI. EXPERIMENTAL RESULTS AND ANALYSIS

We ran our basic CNN as well as the pre-trained VGG-16 model on the non-preprocessed dataset. It was found to give poor accuracies when compared to the models who were fed preprocessed data. The following graphs show how our model does on the dataset which was not preprocessed.

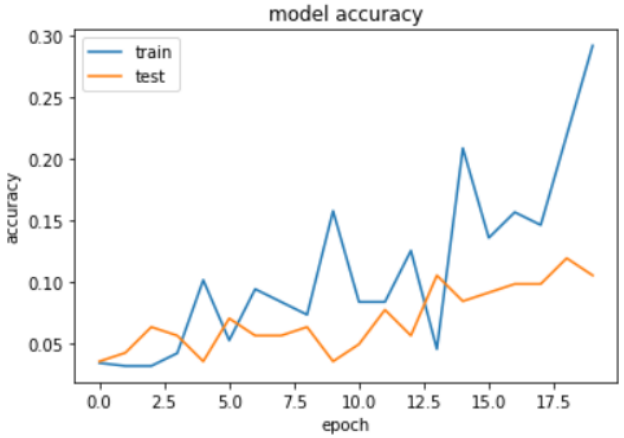

Fig 6.a: Model accuracy of basic CNN without preprocessing

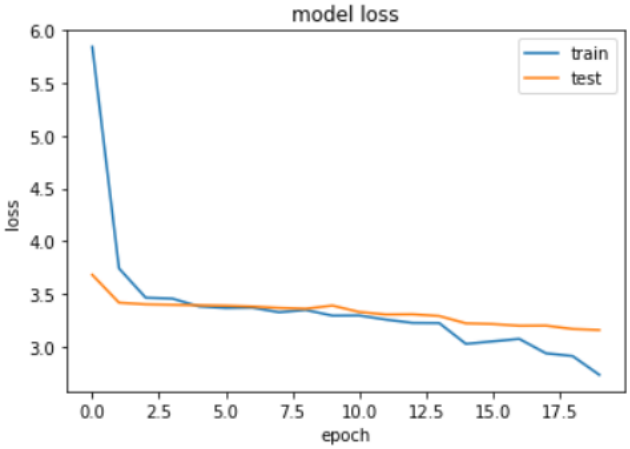

Fig 6.b: Model loss of basic CNN without preprocessing

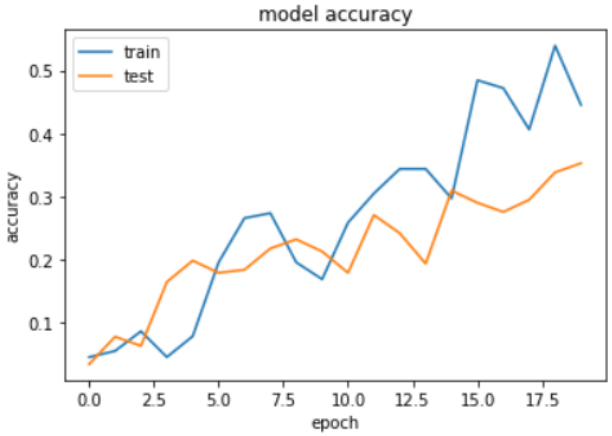

Fig 6.c: Model accuracy of the pre-trained model on non-preprocessed dataset.

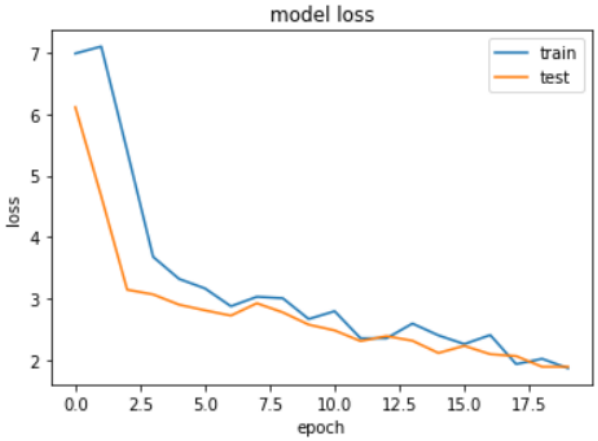

Fig 6.d: Model loss of the pre-trained model on non-preprocessed dataset

We trained our model on preprocessed data. We tried two approaches.

First, we created our own CNN from scratch. It was a series of two convolution and Maxpool layers with two dense layers at the end. The fig 6.e and 6.f shows the accuracy and loss of the model on training and test dataset.

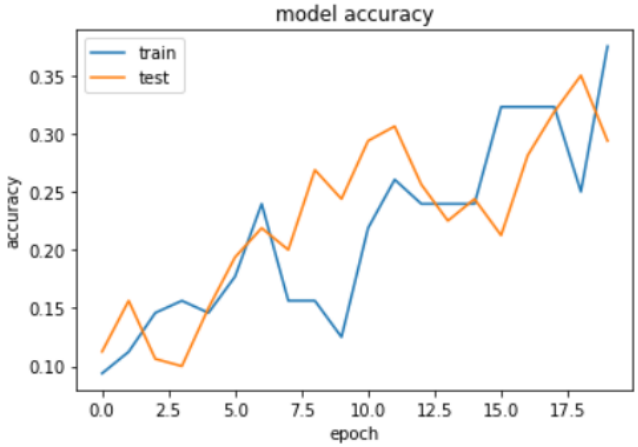

Fig 6.e: Accuracy of basic CNN model with preprocessing done on the dataset.

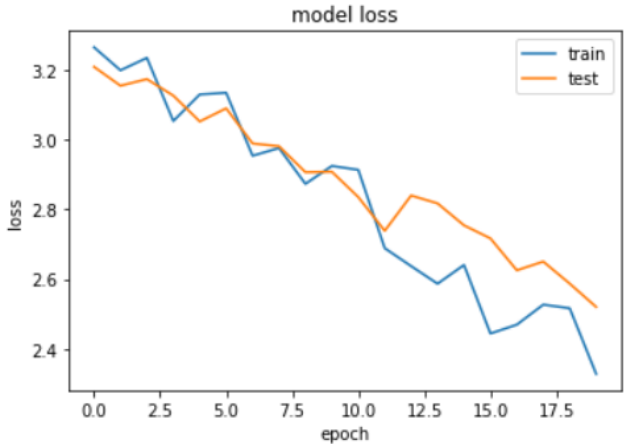

Fig 6.f: Loss of basic CNN model with preprocessing done on the dataset.

Second, we made the use of a pre-trained model VGG-16 and used ImageNet weights, added our own dense layers and could achieve a greater surge in the accuracy as compared to the model which was made from scratch. The figure 6.g and 6.h shows the accuracy and loss of the model on training and test dataset.

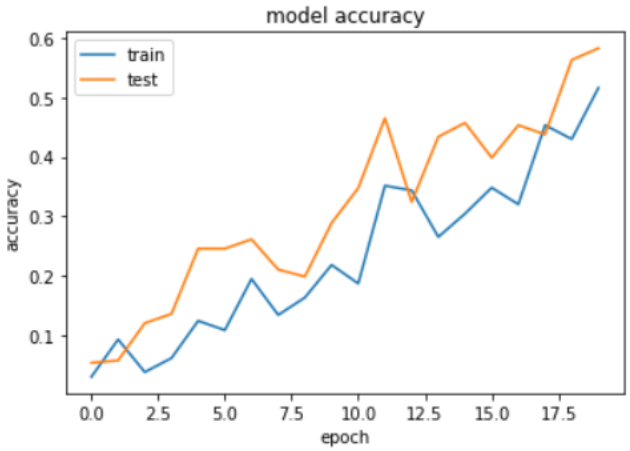

Fig 6.g: Accuracy of the VGG model on the dataset

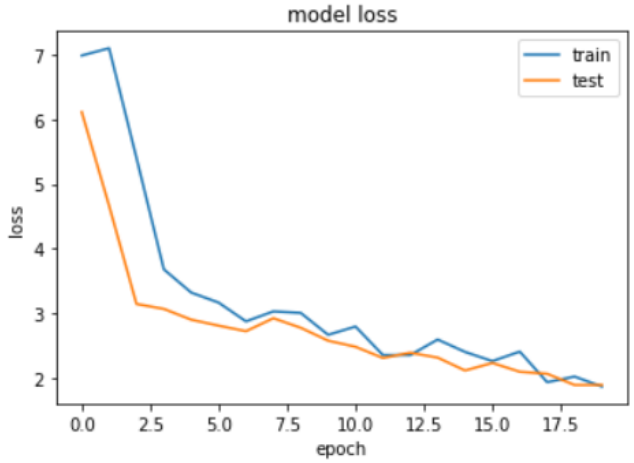

Fig 6.h: Loss for Pretrained VGG-16 model.

The following table shows the comparison of the accuracies obtained on the above models.

	Model	Accuracy
1.	Pre-trained VGG-16 model on preprocessed dataset	55.47%
2.	Basic CNN model on preprocessed dataset.	29.38%
3.	Pre-trained VGG-16 model on non-preprocessed dataset	35.27%
4.	Basic CNN model on the non-preprocessed dataset.	10.49%

VII. Future Work

We have implemented our CNN on 30 most significant classes, further we will train the model with all images from 4250 classes. This requires a powerful GPU..We can also deploy CNNs which localize the tail and learn its shape.

VIII. Conclusion

We have preprocessed our data using image augmentation techniques, implemented our CNN from scratch and used VGG-16 pre-trained model on our dataset which caused accuracy to increase tremendously. Although our accuracy is decent, we could say that image augmentation along with the concept of transfer learning helped us in increasing the accuracy of the model.

IX. Contribution Of Team Members

Pooja Kudav worked on image augmentation, VGG model and CNN. Piyush Mahatkar worked on image augmentation and CNN.

References

[1] "Which whale is it, anyway? Face recognition for right whales using deep learning," *deepsense.ai Blog*, 04-Dec-2017. [Online]. Available: https://blog.deepsense.ai/deep-learning-right-whale-recognition-kaggle/. [Accessed: 24-Apr-2018].

[2] "Plotting on a large number of facets¶," *Plotting on a large number of facets - seaborn 0.8.1 documentation*. [Online]. Available: https://seaborn.pydata.org/examples/many_facets.html. [Accessed: 27-Apr-2018].

[3] "Applications," *Applications - Keras Documentation*. [Online]. Available: https://keras.io/applications/. [Accessed: 20-Apr-2018].

[4] Simonyan, Karen, Zisserman, and Andrew, "Very Deep Convolutional Networks for Large-Scale Image

[5] Recognition," *[1409.1556] Very Deep Convolutional Networks for Large-Scale Image Recognition*, 10-Apr-2015. [Online]. Available: https://arxiv.org/abs/1409.1556. [Accessed: 26-Apr-2018].

[6] H. H. Aghdam and E. J. Heravi, "Convolutional Neural Networks," *Guide to Convolutional Neural Networks*, pp. 85–130, 2017.