Udacity Capstone:
Humpback Whale Identification – Can you identify a whale by its tail?

Definition

Project Overview

For the past 40 years the means of identifying a whale for scientific purposes has relied on checking images by hand and comparing to those already identified.

The most common method of identification is by looking at the whales' flukes (tail) whose characteristics are unique with each adult whale, similar to a fingerprint of a human.
The image below shows the anatomy of a humpback whale's flukes.

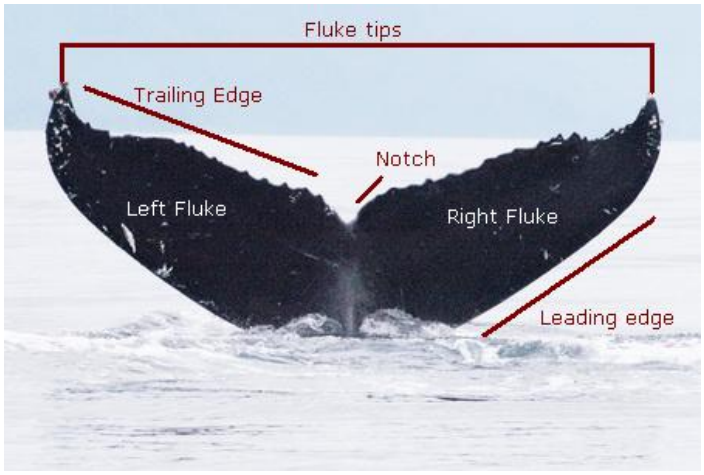


Fig 1 Humpback Whale flukes [http://www.alaskahumpbacks.org/matching.html]

Because of this manual effort, there is a large backlog of untapped data which is unavailable simply because of the time needed to identify a whale by hand. If there was a way to detect if a whale has been identified previously or if the whale is a new identification, this would save many man hours of work and allow the larger volumes of information at citizen science sites such as happywhale.com to be utilised.

Dataset

Kaggle as of 25th January 2019 has a dataset of humpback whales which has been collected from happywhale.com [https://www.kaggle.com/c/humpback-whale-identification/data]
The data is broken into four sets as follows:

- **train.zip** - a folder containing the training images
- **train.csv** - maps the training Image to the appropriate whale Id. Whales that are not predicted to have a label identified in the training data should be labelled as new_whale.
- **test.zip** - a folder containing the test images to predict the whale Id
- **sample_submission.csv** - a sample submission file in the correct format

The training dataset contains over 25,000 images of whales of varying file sizes and quality in jpg format. The train.csv file consists of two columns showing the filename of each of the images in the train set and a unique whale identification Id if the whale is known, or new_whale if the whale is unknown

train

Image	Id
0000e88ab.jpg	w_f48451c
0001f9222.jpg	w_c3d896a
00029d126.jpg	w_20df2c5
00050a15a.jpg	new_whale
0005c1ef8.jpg	new_whale
0006e997e.jpg	new_whale
000a6daec.jpg	w_dd88965
000f0f2bf.jpg	new_whale
0016b897a.jpg	w_64404ac
001c1ac5f.jpg	w_a6f9d33

Fig 2. Sample data from train.csv

There is also a list of images in a test folder which are part of the Kaggle competition that can be submitted but do not have a results csv, and so can't be used in testing the validity of the detection model. This would have to be done by splitting the training data accordingly.

Problem Statement

Can we identify from a new image of a whale's fluke whether we have seen this whale before?

The Kaggle competition associated with this dataset requires submissions to provide up to five possible identifications for each image. As such, this is what we will measure our success upon. In this instance we can only use the training dataset to measure our success.

The solution requires image recognition of up to the five best possible responses. With the problem being one of image recognition; a Convolutional Neural Network (CNN) would be an appropriate model with the final layers equalling the number of existing whales that have been identified. With an output based on sigmoid, we can select the images which the CNN outputs as the most likely match. If the likelihood of a whale selection is extremely low then we can include a label as a 'New Whale'. It is most likely that the use of Transfer Learning by using a pre-existing CNN model such as ResNet50 will be of significant benefit.

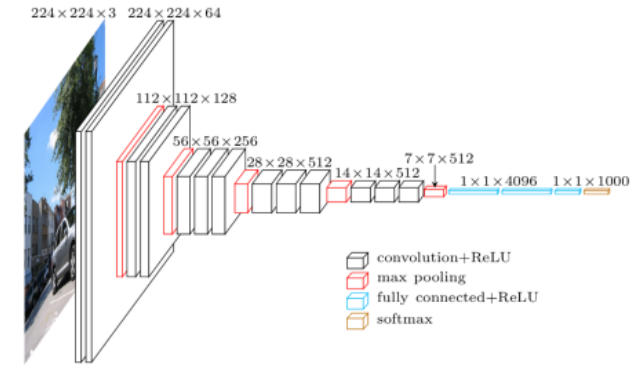


Fig 3. CNN Example with softmax final output layer. Note our proposal is to use sigmoid since we require more than a 0/1 output. [Image Source: Alexis Cook https://alexisbcook.github.io/2017/global-average-pooling-layers-for-object-localization/]

The diagram below shows the proposed workflow

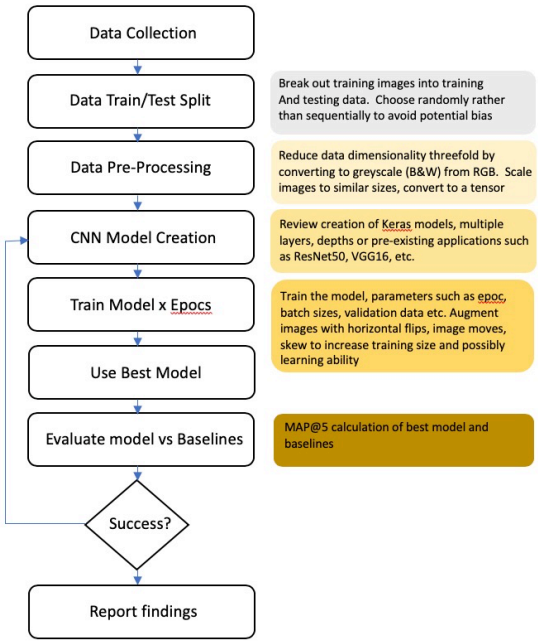


Fig 4. Proposed Workflow

Metrics

The Kaggle competition referenced for Humpback Whale Identification explicitly lists how they would evaluate submissions. Here it is stated they would evaluate according to the Mean Average Precision @ 5 (MAP@5)

'where U is the number of images, $P(k)$ is the precision at cutoff k, n is the number predictions per image, and $rel(k)$ is an indicator function equalling 1 if the item at rank k is a relevant (correct) label, zero otherwise. ', [https://www.kaggle.com/c/humpback-whale-identification#evaluation]

$$MAP@5 = \frac{1}{U}\sum_{u=1}^{U}\sum_{k=1}^{min(n,5)} P(k) \times rel(k)$$

For example, if the correct label is A for an observation, the following predictions all score an average precision of 1.0.

[A, B, C, D, E]
[A, A, A, A, A]
[A, B, A, C, A]

We will compare our model against a simple model where each image is defined as a newly detected whale. We will also measure accuracy alongside MAP to see if we can predict more whale identifications within a maximum of 5 proposals. Can our model identify whales any more accurately than this and how precise will the model be?

Analysis

Data Exploration

As previously discussed, the training dataset consists of over 25000 images of whale flukes and a train.csv file which lists the name of these images against a unique whale ID if known, or 'new_whale' if the whale has not been seen before.

Below is a breakdown of the label ranges seen in the train.csv.

total whales 25361
'new_whale' count 9664
Percentage of whales labelled as 'new_whale' 38.0
None 'new_whale' count 15697
number of whales labelled only once 2073
total number of unique whale Id's seen once 2073
number of whales labelled more than once 13624
total number of unique whale Id's seen more than once 2931
number of whales identified two or less times 4643
number of whales identified more than two times 11054
number of whales identified five or less times 8299
number of whales identified more than five times 7398

Exploratory Visualization

There are 9664 whales in the training dataset labelled as 'new_whale' which you can see in the bar chart below highly skewers the dataset. Were we to train on a dataset containing this proportion of 'new_whale' data I believe this would be less useful. Since we are more concerned with identifying existing whales, it is proposed to remove these from the training set.

Similarly, in the training set there are some whales which have only been identified once (2073). In the final Kaggle competition these would be useful to detect. However, I am evaluating a model on the training data only and a single whale will either be in the Train or Test set of my data. As such these whales would to some extent be the equivalent of a new whale. I will omit these images from the train/test sets, but for a Kaggle submission they would need to be included.

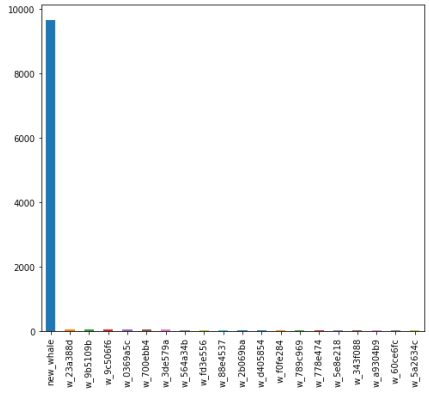


Fig 6. Volume of 'new_whale' files

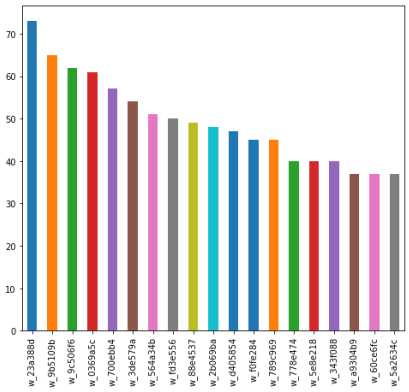


Fig 7. Top 20 whales by number of images once 'new_whale' removed

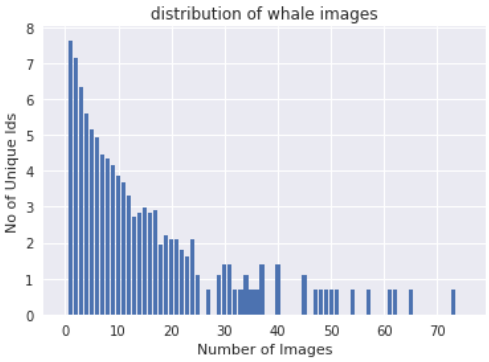


Fig 8. Count of whale IDs vs the number of images of that Whale ID. (excluding 'new_whale') Log scale

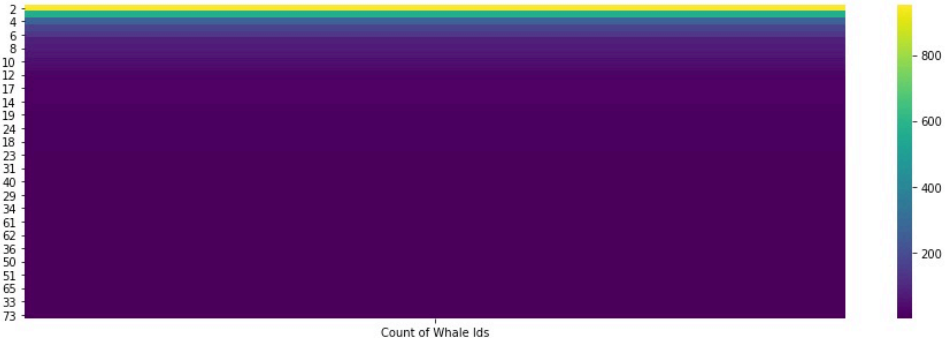


Fig 9. Whale ID detection. Majority of whales have only been sighted twice

From the analysis, you can see that there are only 2931 whales from the whole of the training data which have been logged more than once. Those will be used in our Train/Test split. The majority of those IDs have only been detected twice as can be seen in the heatmap. Again, these will be difficult to predict in our train set though valid for a full identification method.

Algorithms and Techniques

Based on the proposed workflow diagram shown in Fig2 the following techniques will be applied:

One Hot Encoding: We know that we need to have an output based on one of 2931 whale Id labels. In order to feed this into the CNN and return the output effectively the labels would need to be one hot encoded. We can then use the one hot encoded 'y' labels for training and our outputs. From here we can convert back to the actual label names when showing the predictions.

Shuffle and Split Data: Of the selected data we will split this 80/20 between Train and Test sets. The Train set would be additionally split between Train and Validation data. During training of the CNN, a validation group is required as the CNN weights are modified. The Test set will be used to determine the model's success.

Image pre-processing: Each image will need to be sent to the CNN in a specific shape/size. This will be in the shape of a 224x224x3 tensor. Using OpenCV library will assist with this, a

number of techniques will be used to modify the image before fitting the model. For example, does adding padding to the image rather than rescale improve the model? Does a bounding box help?

Model Fit: Its most likely that the CNN to use with be based upon *Transfer Learning*. That being we'll use an existing model such as ResNet50 but strip off the final output layer. From here we will add new layers and an output layer of 2931 to match the number of whale IDs we are predicting. We will fit the model initially not adjust any of the weights of the existing ResNet50 model, only adjusting the layers we added. Following this is may be worthwhile to allow the last layer weights of ResNet50 to be modified during training to see if this improves performance.

Benchmark

The proposal will be to create a Convolutional Neural Network with a sigmoid output matching the number of whale Id's as output. Since we have removed the 'new_whale' Id and any other Id we have seen only once, the output shape for the CNN will be 2931. From these 2931 outputs we will choose the top 5 highest likelihood. If any of these likelihoods is an extremely low value (low likelihood) we will replace that label with 'new_whale' which is more probable. This will allow is to 'pop' in a 'new_whale' in our predictions where our confidence level is low.

We use the MAP@5 calculation to provide a mean average precision score. Since we know a base method of choosing 'new_whale' each time on our full Train dataset would yield a 38% accuracy, can our model improve upon this. Since the test data does not contain 'new_whale' data any success in whale ID detection would be an improvement.

Methodology

Data Pre-processing

Consolidate Data Reference

To simplify the lookup of file names, file paths and whale ID's I used one data frame as the primary reference based on the train.csv file

Image	Id
0000e88ab.jpg	w_f48451c
0001f9222.jpg	w_c3d896a
00029d126.jpg	w_20df2c5
00050a15a.jpg	new_whale
0005c1ef8.jpg	new_whale

before

Image	Id	Image_path
0000e88ab.jpg	w_f48451c	/home/ubuntu/kaggle/train/0000e88ab.jpg
0001f9222.jpg	w_c3d896a	/home/ubuntu/kaggle/train/0001f9222.jpg
00029d126.jpg	w_20df2c5	/home/ubuntu/kaggle/train/00029d126.jpg
00050a15a.jpg	new_whale	/home/ubuntu/kaggle/train/00050a15a.jpg
0005c1ef8.jpg	new_whale	/home/ubuntu/kaggle/train/0005c1ef8.jpg

After

Fig 9. Modified DataFrame based on train.csv file and Image Paths

From this dataset I removed all the entries where the Id was labelled as 'new_whale' or those which had an Id count of one or less. This could then be used as the dataset for the remainder of the project.

One Hot Encoding

One Hot Encoding was applied to the Id column to generate a 'y_ones_hot' label list and the encoding was validated to ensure accuracy.

```
The first whale is identified  14 times
```

Image	Id	Image_path
0000e88ab.jpg	w_f48451c	/home/ubuntu/kaggle/train/0000e88ab.jpg
0af805558.jpg	w_f48451c	/home/ubuntu/kaggle/train/0af805558.jpg
1c351b88e.jpg	w_f48451c	/home/ubuntu/kaggle/train/1c351b88e.jpg
6f7abb1be.jpg	w_f48451c	/home/ubuntu/kaggle/train/6f7abb1be.jpg
77a44bf94.jpg	w_f48451c	/home/ubuntu/kaggle/train/77a44bf94.jpg
79c77838d.jpg	w_f48451c	/home/ubuntu/kaggle/train/79c77838d.jpg
9064d5875.jpg	w_f48451c	/home/ubuntu/kaggle/train/9064d5875.jpg
9fc84d2ae.jpg	w_f48451c	/home/ubuntu/kaggle/train/9fc84d2ae.jpg
c1ec12eb6.jpg	w_f48451c	/home/ubuntu/kaggle/train/c1ec12eb6.jpg
c64e5e861.jpg	w_f48451c	/home/ubuntu/kaggle/train/c64e5e861.jpg
c9df69a69.jpg	w_f48451c	/home/ubuntu/kaggle/train/c9df69a69.jpg
db0699767.jpg	w_f48451c	/home/ubuntu/kaggle/train/db0699767.jpg
e2f1b6c4a.jpg	w_f48451c	/home/ubuntu/kaggle/train/e2f1b6c4a.jpg
e3f2dbd25.jpg	w_f48451c	/home/ubuntu/kaggle/train/e3f2dbd25.jpg

```
y_ones_hot of first entry on entry 2810 is 1.0
now lets check all the images to make sure the encoding is ok
0000e88ab.jpg 0 2810 1.0
0af805558.jpg 584 2810 1.0
1c351b88e.jpg 1433 2810 1.0
6f7abb1be.jpg 5821 2810 1.0
77a44bf94.jpg 6239 2810 1.0
79c77838d.jpg 6362 2810 1.0
9064d5875.jpg 7630 2810 1.0
9fc84d2ae.jpg 8429 2810 1.0
c1ec12eb6.jpg 10272 2810 1.0
c64e5e861.jpg 10527 2810 1.0
c9df69a69.jpg 10747 2810 1.0
db0699767.jpg 11660 2810 1.0
e2f1b6c4a.jpg 12083 2810 1.0
e3f2dbd25.jpg 12133 2810 1.0
```

Fig 10. One Hot Encoding validation

Split the Data into Train and Test

The data was split between Train/Test with an 80/20 split and the Train set was again split 80/20 between Train/Validation sets for fitting the model. The 'y' labels were taken from the one hot encoding previously applied.

Train count	Validation count	Test count
8719	2180	2725

Image Standardisation Prior to Fitting

For a CNN to receive an image as input it needs to be converted into a tensor of the same dimensions each time. For my CNN model I chose an input of 224x224x3 dimensions. I did consider reducing data dimensionality by converting imaged to gray scale. However, ResNet50 input requires a 3rd dimensionality and repeating the content of a gray image to reflect this shape did not seem worthwhile.

When reviewing sample images, it's possible to see that they are of various sizes which would need to be adjusted.

Reshape

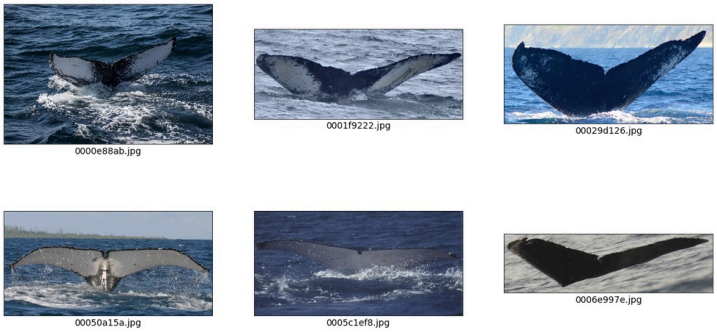


Fig 11. Sample images unchanged

Padding

The question being, would reshaping these images to the same size be sufficient, or would setting the width to be identical, then padding the missing dimensions with blank data provide more precision?

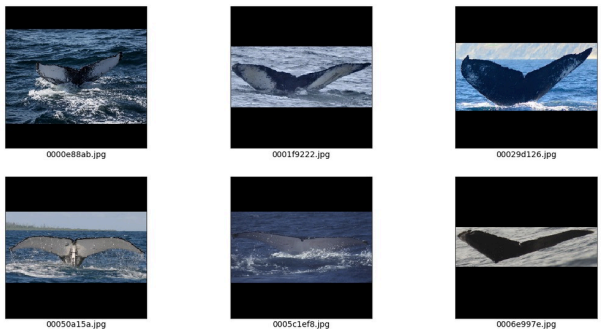


Fig 12. Sample images with width 224 and padding applied

Bounding Box

Lastly, could it be possible that a bounding box would provide improvements? Using OpenCV image Thresholding techniques as described at [https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html]

Defining a boundary (once the image was converted to gray), between the colour of the sea and the dark image of the whale's fluke allowed us to create a simplistic bounding box with some success.

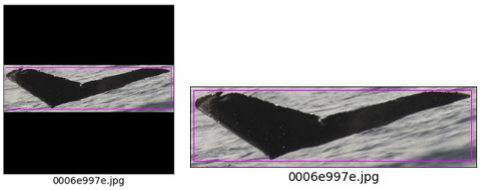


Fig 13. Sample image with bounding box with and without padding

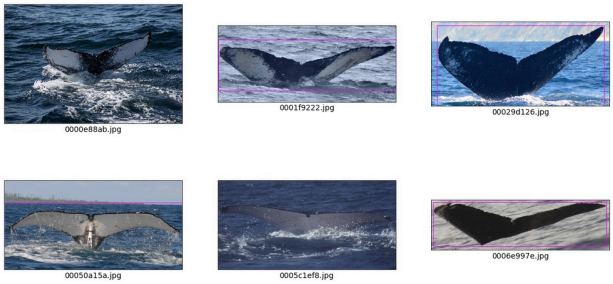


Fig 14. Additional bounding box examples

Implementation

We have the data split into Train, Validation, Test sets with the images converted to a tensor and our Y labels encoded with One Hot encoding. We ran the CNN fit process multiple times with the various image manipulations namely:

- Resize to 224x224x3
- Resize to 224x224x3 with padding
- Resize to 224x224x3 with bounding box
- Resize to 224x224x3 with padding and bounding box

Transfer Learning

The CNN model to implement would be a Transfer Learning model where we take a pre-existing CNN model (ResNet50), and weights (ImageNet), which have been trained on over

14 million images. We can then remove the final dense layers of this model which are used for classification and add our own. Once this new CNN is created, we can fit using our Training and Validation data to adjust the weights of our final dense layer. We explicitly locked the pre-trained portion of the model so that our fitting cannot adjust those weights, only the weights of the layers we added.

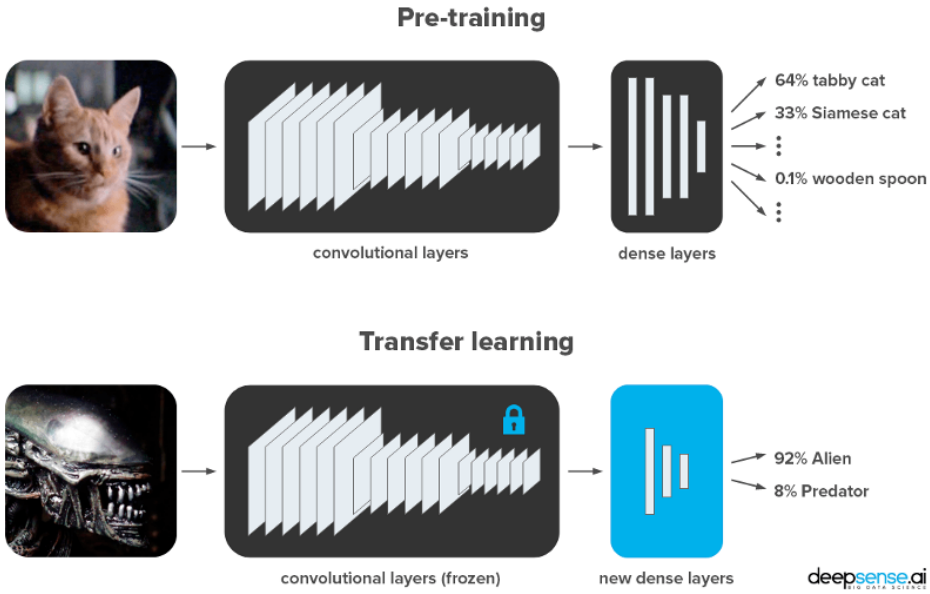


Fig 15. Transfer Learning with frozen CNN ResNet50 Layer.
[https://medium.freecodecamp.org/keras-vs-pytorch-avp-transfer-learning-c8b852c31f02]

In the figure above the dense layers are predicting the likelihood of two objects. In our model we will be predicting the likelihood of 2931 Whale IDs and choosing only those 5 with the highest likelihood.

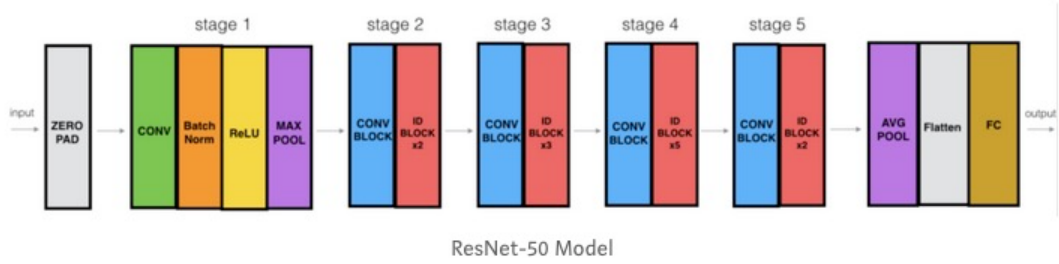
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Model)             (None, None, None, 2048)  23587712
_________________________________________________________________
global_average_pooling2d_1 ( (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               1049088
_________________________________________________________________
dense_2 (Dense)              (None, 2931)              1503603
=================================================================
Total params: 26,140,403
Trainable params: 2,552,691
Non-trainable params: 23,587,712
_________________________________________________________________
```

```
mymodel.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['accuracy'])
print ("How many weights in the model are trainable?", len(mymodel.trainable_weights))
```

```
How many weights in the model are trainable? 4
```

Fig 16. Transfer Learning model with the ResNet50 model weights untrainable, only the Dense layers we added with the final 2931 output.

Following this we tested a model where the last stage of the ResNet50 model was trainable followed by an even simpler output layer.



ResNet-50 Model

Fig 17. ResNet50 model, Setting stage 5 and our replacement output layer as trainable

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Model)             (None, None, None, 2048)  23587712
_________________________________________________________________
global_average_pooling2d_9 ( (None, 2048)              0
_________________________________________________________________
dense_10 (Dense)             (None, 2931)              6005619
=================================================================
Total params: 29,593,331
Trainable params: 7,060,339
Non-trainable params: 22,532,992
_________________________________________________________________
```

```
mymodel.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['accuracy'])
print ("How many weights in the model are trainable?", len(mymodel.trainable_weights))
```

```
How many weights in the model are trainable? 6
```

Fig 18. Transfer Learning model with some ResNet50 model weights trainable. Increase in Trainable parameters and weights.

Fitting the Model to our Data

When updating the weights of the new model we used the Train and Validation data for this purpose. Alongside the basic image preparation and conversion to a tensor we can augment the Train data so that as the number of iterations/epochs occurs the model is using images which have been adjusted. This to some extent increases our volume of data we can train on. This will be worthwhile as it allows increase the number of images associated with specific Whale IDs. Images taken by different people of the same whale flukes will vary in multiple ways, augmenting the data to some extent will replicate this. Keras' ImageGenerator function was used augment the Train images, Validation images were not augmented apart from rescaling the image data by dividing by the largest value in an image (255).

The following parameters were chosen for augmentation as a base:

Augmentation	Initial Value	Justification
rescale	1/255	Rescale image data between 0 and 1 improves efficiencies
rotation_range	5	Rotate the image by up to 5 degrees
width_shift_range	0.2	Move image left/right by up to 20%
height_shift_range	0.2	Move image up/down by up to 20%
shear_range	0.02	Adjust shear of image by 2%
Zoom_range	[0.9, 1.25]	Adjust zoom in/out by -10% to 25%
Horizontal_flip	False	Flip image horizontally
Vertical_flip	False	Flip image vertically
Fill_mode	nearest	'nearest': aaaaaaaa|abcd|dddddddd Extending image to boundary if adjusted
Brightness_range	None	Increase/Decrease image brightness by up to 50%

Table 1. Initial image augmentation settings

Fitting did not require a very large number of epochs since most of the model weights could not be adjusted.

Prediction

Prediction was made, and the model's success was calculated. Because the Training data showed that 38% of the total images were of 'new_whale', it makes statistical sense with a MAP@5 calculation to always define one of the five selections as 'new_whale' since the highest possible probability of the 5th selection being correct is 20% even though this is highly unlikely.

By calculating how many whales were accurately predicted in the Test set and the Mean Average Precision @5, this would show what improvements can be made over a base prediction of just selecting 'new_whale' as the first and only choice.

Refinement

Image to Tensor

There were a number of areas where refinement could be made. The first as previously described was with the initial conversion of the images to a tensor. I tried to fit the model using just a reshape to 224x224x3 as well as using padding and boundary box combinations. Surprisingly the less adjustments made here the more whale IDs were predicted. As such for the tensor portion I just used the resizing of the image to 224x224x3

Image Augmentation

Various image augmentation parameters were considered including image flip, shear, rotation, shift, scale, brightness. The reasoning to flip horizontally being that a fluke could be in either position depending on when the photograph was taken. Similarly, if the whale fluke image was taken from the front or the back of the whale this would affect the identification. The issue with this is that it could cause potential misclassification. The position, zoom etc. were of a reasonable range which didn't not appear to affect the predictions and increased the training set.

Optimizer Hyperparameters

While fitting the model I reviewed a number of documents to see if there were specific learning rate step sizes and came across a number of papers namely:

Fig 19. Learning Rate vs Time to Train
https://medium.com/octavian-ai/which-optimizer-and-learning-rate-should-i-use-for-deep-learning-5acb418f9b2

The original paper [Adam: A Method for Stochastic Optimization: https://arxiv.org/abs/1412.6980] noted the following:
'*The default value of 1e-8 for epsilon might not be a good default in general. For example, when training an Inception network on ImageNet a current good choice is 1.0 or 0.1.*'

Those values are set as defaults in the Keras Optimizers, and changing these did not improve the MAP@5 score.

Alongside the optimizer hyperparameters, changes to batch sizes when fitting would determine how often the weights are adjusted. Discussions recommended using as large a batch size as possible before making weight changes. [Please Explain Why Batch Sizes matter: https://forums.fast.ai/t/please-explain-why-batch-size-matters/9045/2]

As such various fits were made with batch sizes between 32 and 512, and epochs ranging from 10 to 50. Batch sizes greater than 512 were not possible with the compute resources available.

For these iterations the model and weights were saved so they could be reused in the future without having fit again at a later date.

Results

Model Evaluation and Validation

As previously discussed, the Kaggle evaluation would use Mean Average Precision @ 5 as its evaluation model. Prior to calculating MAP I was interested in maximising the greatest number of Whale IDs correctly identified within 5 recommendations. Since predicting no whale IDs accurately would lead to a worse MAP score. From the various image to tensor changes, image augmentations and hyperparameter adjustments the following accuracies were found when predicting Whale ID from the Test set of 2725 images.

Parameters	Whale IDs correctly identified in first 5	Accuracy
Tensor Image 224x224x3 Batch Size 32, epochs 40, Image Augmentation base	138	5.06
Tensor Image 224x224x3 Batch Size 32, epochs 40, Image Augmentation brightness_range=[0.5, 1.5]	159	5.83
Tensor Image 224x224x3 Batch Size 64, epochs 40, Image Augmentation base	151	5.54
Tensor Image 224x224x3 Batch Size 64, epochs 40, Image Augmentation brightness_range=[0.5, 1.5]	160	5.87
Tensor Image 224x224x3 + Padding Batch Size 32, epochs 40, Image Augmentation base	79	2.90
Tensor Image 224x224x3 + Padding + Boundary Box Batch Size 32, epochs 40, Image Augmentation base	71	2.60

Table 2. Model with only output layer trainable

When Using the model where the last layer of ResNet50 was trainable and using a simpler output layer, the number of epochs required was reduced by half to achieve similar results. Using the base rescaled tensor and using the augmentation parameters previously yielding the better results the following improvements were found. With these higher accuracy results the MAP@5 score was also calculated.

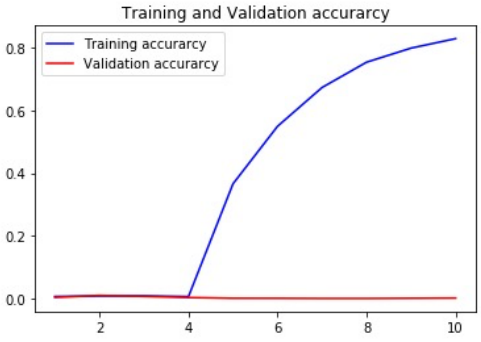
Parameters	Whale IDs correctly identified in first 5	Accuracy	MAP@5
Tensor Image 224x224x3 Batch Size 32, epochs 10, Image Augmentation base	179	6.57	0.0070
Tensor Image 224x224x3 Batch Size 256, epochs 10, Image Augmentation base	173	6.35	0.0064
Tensor Image 224x224x3 Batch Size 512, epochs 10, Image Augmentation base	**188**	**6.90**	**0.008**
Tensor Image 224x224x3 Batch Size 512, epochs 20, Image Augmentation base	189	6.94	0.00785

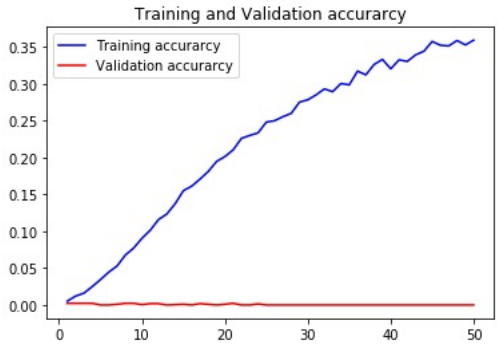
Table 3. Model with ResNet50 Layer5 trainable alongside output layer

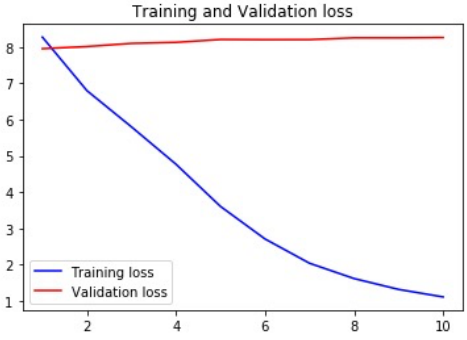
Justification

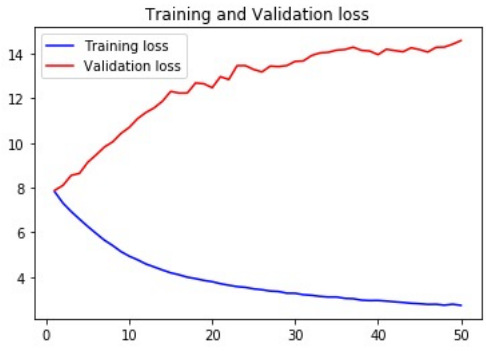
From the various models and hyper parameters tested the optimum model was when modifying the ResNet50 weights during training with a batch size of 512. The number of epochs whether 10 or 20 did not make a significant improvement, and therefore I would choose the model running for 10 epochs as the preferred model with the slight improvement of MAP score and reduced compute overhead.

Training/Validation accuracy and loss looked to indicate overfitting, though I could not seem to make improvements to what is seen below. Some models were significantly worse in that the validation loss significantly increased over time.









Selected Model

None Selected Model

Fig 20. Train/Validation accuracy/loss for selected model and a non selected model

Conclusion

We can see that the model does predict with some level of success the identification of a previously seen whale. Though the accuracy rate is low I believe this is in part due to the high volume of whales where there are only 2 or 3 images. If I reduced the train and test to a rate where there were more images of each whale, accuracy would be higher. With the training sets provided I believe the model is successful.

Free-Form Visualization

I was interested to see how my accuracy and map scores would change if I increased the number of offered whale IDs.

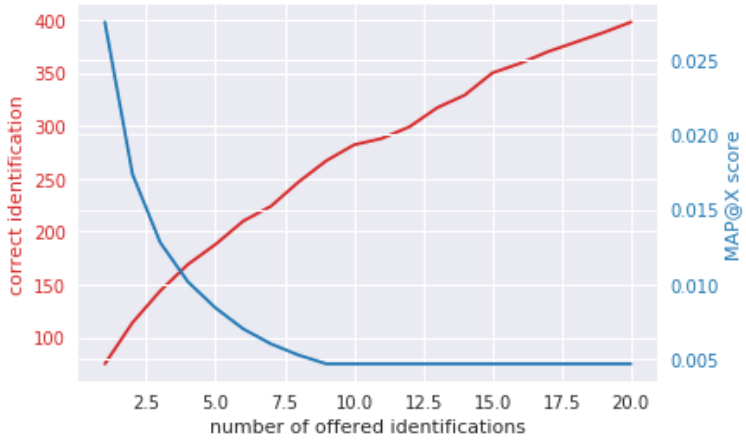


Fig 21. Identification success vs MAP@X scores as the list of proposed identifications increases

As the figure shows, as the number of identification offers increases, our accuracy increases, though our level of precision drops. An optimum compromise appears at the crossover point of offering 4 identifications.

Reflection

Image evaluation is of interest as it's something I'll have to look at in the future and has been one of the highlights of the Udacity course. Looking into the image pre-processing such as bounding boxes, image padding took some time to create and with the model built I was surprised at the little difference they made to the fitting results. I did create a simplistic CNN model by hand (not in the python code supplied) to confirm its effectiveness and that model failed miserably failing to predict any whales whatsoever. I wanted to do this to reinforce the usefulness of using pre-existing models and weights such as ResNet50 and ImageNet are.

I was also surprised at the effectiveness of allowing the last convolutional layer of the ResNet50 model to have its weights modified, as opposed to adding the complexity to the output layer. This significantly increased fit times and improved performance.

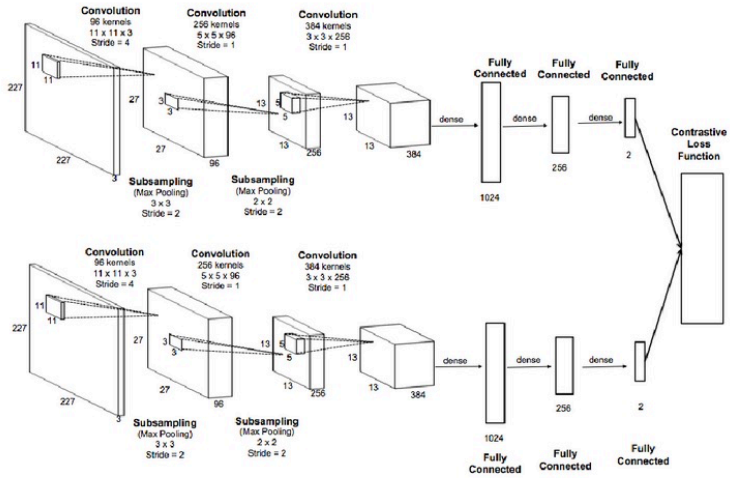
I deliberately kept away from the Kaggle discussion boards when proposing the project. What was interesting is the confirmation that the methodology I had was in line with others. Particularly with regards removing the 'new_whale' labels from the training set. It appears that a large group of successful submissions have applied the dame methodology.

Improvement

In order to improve the model, it would be interesting to spend more time isolating the whale fluke from the rest of the image.

Looking at solutions such as You Only Look Once (YOLO) or other object identification methods as a pre-processing step could help isolate the useful data.

Additionally, it appears that there is a growing interest in Siamese Neural Networks where two or more identical subnetworks with equal weights are used before connecting to a final fully connected layer. These models are reported to be very useful in 'One-shot' learning where there is only one sample per class [Koch, Gregory, Richard Zemel, and Ruslan Salakhutdinov. "Siamese neural networks for one-shot image recognition." ICML Deep Learning Workshop. Vol. 2. 2015.]
In the case of the whale IDs there are many which match these criteria.



Example for a siamese network (source: Rao et al.)

References

https://arxiv.org/abs/1412.6980
https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning
https://medium.com/octavian-ai/which-optimizer-and-learning-rate-should-i-use-for-deep-learning-5acb418f9b2
https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf
https://www.kaggle.com/xhlulu/exploration-and-preprocessing-for-keras-224x224
https://www.kaggle.com/suicaokhoailang/resnet50-bounding-boxes-0-628-lb
https://flyyufelix.github.io/2016/10/08/fine-tuning-in-keras-part2.html
https://github.com/fchollet/deep-learning-models/blob/master/resnet50.py
https://stackoverflow.com/questions/41378461/how-to-use-models-from-keras-applications-for-transfer-learnig/41386444#41386444
https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html
https://jhonatandasilva.com/bias-in-ai/
https://forums.fast.ai/t/please-explain-why-batch-size-matters/9045/5
https://shaoanlu.wordpress.com/2017/05/29/sgd-all-which-one-is-the-best-optimizer-dogs-vs-cats-toy-experiment/
https://medium.com/octavian-ai/which-optimizer-and-learning-rate-should-i-use-for-deep-learning-5acb418f9b2
https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/
https://www.quora.com/How-do-I-extract-a-particular-object-from-images-using-OpenCV
https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html
http://www.jussihuotari.com/2018/01/17/why-loss-and-accuracy-metrics-conflict/
https://github.com/benhamner/Metrics/blob/master/Python/ml_metrics/average_precision.py#L3
https://jdhao.github.io/2017/11/06/resize-image-to-square-with-padding/