**Right Whale Recognition using Convolutional Networks**

Andrei Polzounov, Ilmira Terpugova, Deividas Skiparis, Andrei Mihai

{andrei.polzounov, ilmira.terpugova, deividas.skiparis, andrei.mihai}@est.fib.upc.edu

Facultat d'Informàtica de Barcelona, Universitat Politècnica de Catalunya

**Abstract**

Blah blah blah.

Keywords:

**Introduction**

There are fewer than 500 North Atlantic right whales (Eubalaena glacialis) remaining in the world. The species is highly endangered and is considered as such by the U.S. and Canadian governments. Recognizing individual whale specimens is important if we are to help the species recover to sustainability. Recognizing individual specimens from shipborne or helicopter imagery is a tedious task for marine biologists. To the best of our knowledge automated “face” recognition techniques have not previously been proposed for recognizing right whales. In this report we summarize our analysis of using convolutional neural networks to recognize individual North Atlantic right whales.

**Convolutional Neural Networks**

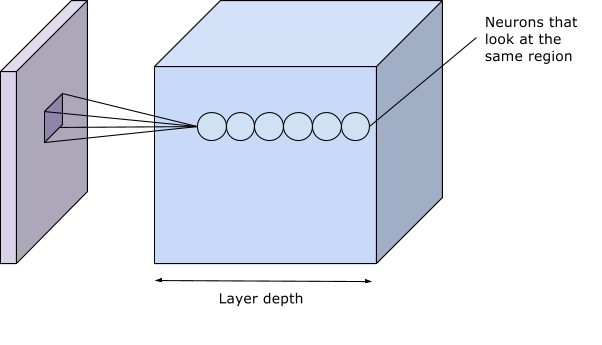
**Citations:** Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., & Rabinovich, A. (2014). Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*.

Mahendran, A., & Vedaldi, A. (2014). Understanding deep image representations by inverting them. *arXiv preprint arXiv:1412.0035*.

**Background:**

In the last few years deep convolutional neural networks have been seeing an explosion in literature and on the internet. They differ from traditional networks by making the explicit assumption that the input data is an image. This allows convolutional or deep networks as they are more commonly called draw another inspiration from nature – receptive fields of vision. The deep network can focus on specific parts of the image using a convolution of the image to focus on specific features of the image.



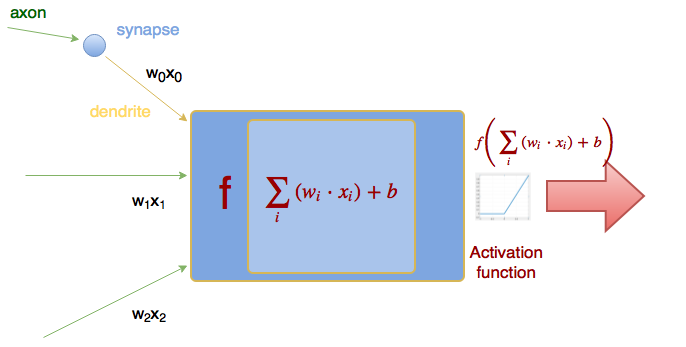


Figure X: Receptive field of CNN vs receptive field of biological neuron

Convolutional networks consist of convolutional layers – which act as receptive fields, followed by pooling layers – which decrease the amount of features and pixels the next convolutional and pooling layers can focus on. These convolutional + pooling layers are stacked many times until finally connected to some sort of classical neural network with some hidden layers and then finally the output layer as the classifier.

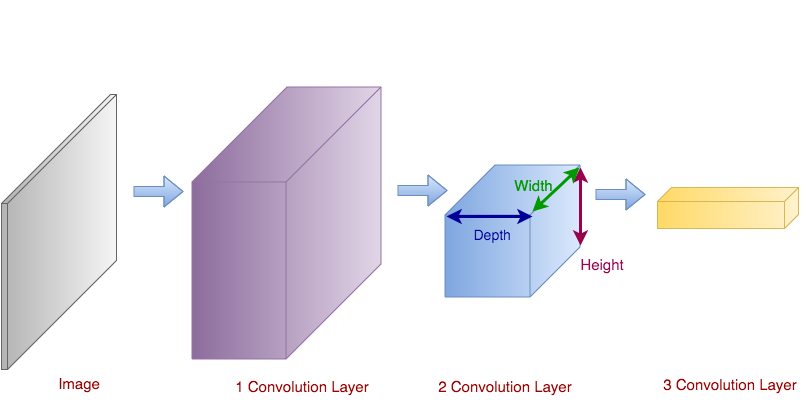


Figure X: ConvNets arrange the neurons in 3 Dimensions (width, height, depth)

The theory is that recognizable features will be close to each other – this allows us to create these receptive fields by convolutional layers. The other option is to create a conventional neural network in which the first neuron layer is connected to every single pixel of the image. This will potentially result in a higher descriptive ability of the network at the cost of greatly increased complexity. Also the neural network will not exploit the fact that pixels very far from each other are not likely to be part of the same feature.

***Successful Applications of Deep Networks:***

Deep networks have been very successful in classifying many different kinds of objects. Some of the best networks are able to classify as many as 22000 different categories learned from a set of 15 million images [ImageNet]. Despite highly optimized code and 3 high performance video cards – the ImageNet network takes about 5 days to train according to the authors.

Another very successful network is outlined in a paper from Google – the GoogLeNet [Google]. It employs many classical computer vision techniques along with the raw computational power of CNN (convolutional neural networks). Some of the novel techniques utilized in the architecture from Google are using very stacks of very small convolutional layers in order to abstract features away from each other and using parallel streams of pooling data which causes some connections to be sparser.

***Problems and possible future improvements of CNNs:***

Is this section relevant?

The main way of increasing model complexity and computational potential is by adding more layers to the topology of the network. This can be problematic because signals have to be extremely strong in order to make it through all the layers. This can be solved by having some neurons skip several layers forward into the network – creating a sparse topology. This is in fact more akin to how neurons fire in biological systems.

It is still not well understood in literature on what exactly creates a better topology for CNNs. A possible, yet very computationally costly solution is use genetic algorithms to rearrange the topology in an automatic way. The exciting potential of this strategy is that a good neural network can “evolve” to solve difficult problems. An unfortunate side effect of the above is the fact that power consumption and computation cost continues to increase with more complicated models and more hyperparameters. A way of speeding up computation and of controlling wasted electricity is using special built hardware such as FPGAs (field programmable logic arrays) for the task of searching through the evolutionary search space. This could potentially result in ground-breaking work only being accessible to large and established tech companies, because no one else can afford expensive FPGAs.

**Expected features for detecting individual whales**

Citation:

North Atlantic Right Whale Catalog. (1997). Retrieved December 27, 2015, from http://rwcatalog.neaq.org

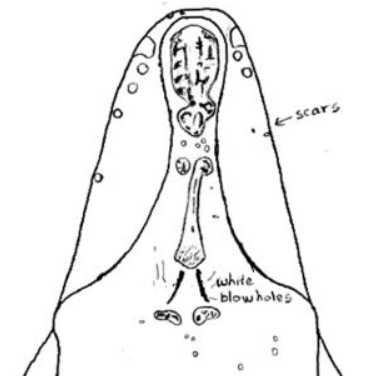


Figure X: Unique callosity pattern of an individual whale specimen

The feature that we are most interested is called the callosity pattern which includes the facial markings on top of the head of the whale and the white markings above the blowholes. These features are unique to each whale.

Some of the features that we decided to ignore were the shape of the tail, dorsal fins and side flippers which may also have been useful, but would have increased the complexity of the detection.

**The dataset**

From the original dataset [cite dataset] of 11469 images only 4542 images were labeled, the labels included 427 unique individuals. Some of the labels only included a single image of the whale. We could not process a dataset this sparse and decided to extract a new dataset – α-whales from the labeled data. Taking only whales which have 20 or more labeled images. This gave us a set of 924 images of 38 unique whales, which is what we based our classifier on.

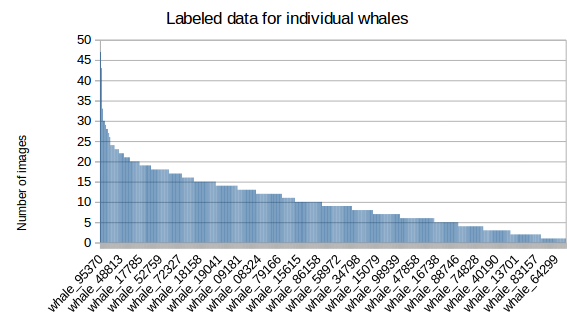


Figure X: Histogram of label distribution in original Kaggle data

**Dataset Preprocessing**

Citation:

Piccardi, Massimo. "Background subtraction techniques: a review." *Systems, man and cybernetics, 2004 IEEE international conference on*. Vol. 4. IEEE, 2004.

The raw images from the dataset were a very large size and resolution. Operating on such images would require massive processing power. In addition much of the visible area of each image was taken up by the water. A large amount of noise with respect to the ROI (region of interest) was added by the waves and splashes around the whale.



Figure X: Image of whale with noise from water

To preprocess the data we had tried to segment the ROI of the whale from the water. We managed to discard the majority of the water pixels by segmenting the histogram of the saturation of the image.

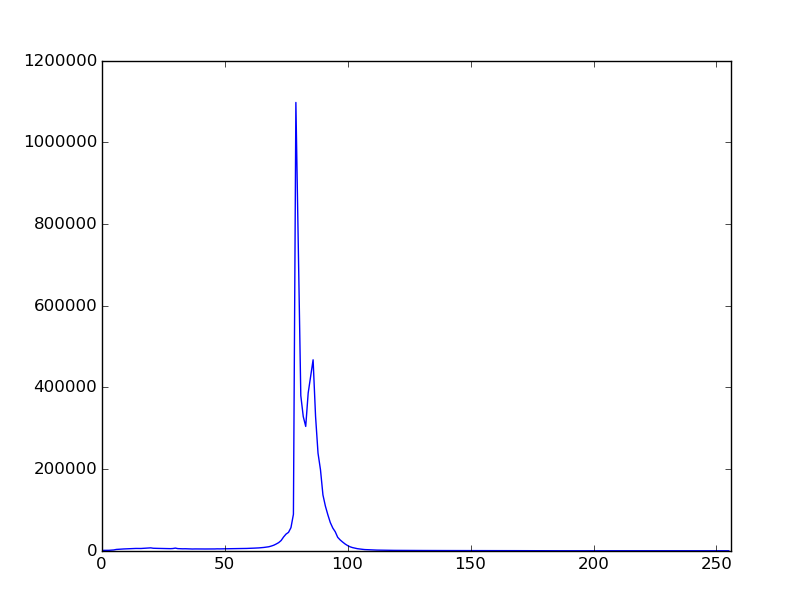
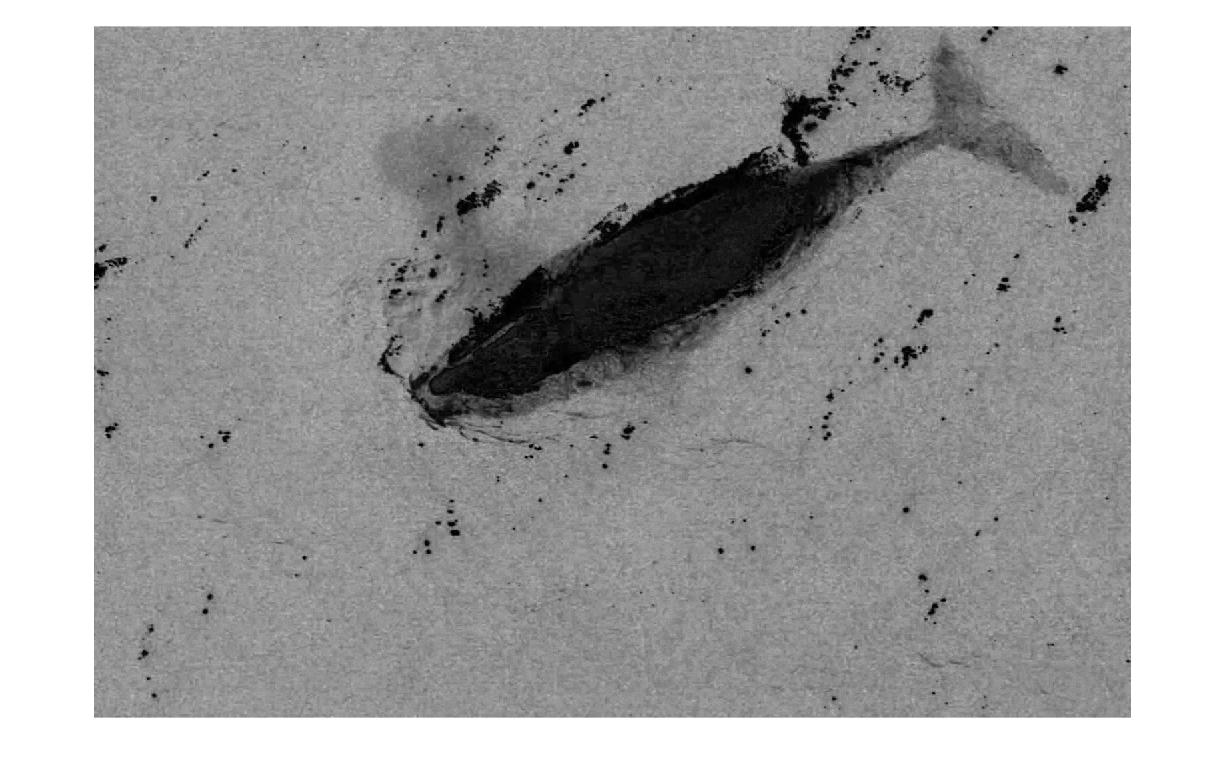


Figure X: Saturation from image and histogram of saturation channel

Ideally the saturation histogram should have two visible peaks (as above). The first and greater maxima is a marker for the water pixels which make up the majority of the image. The second and lesser maxima is therefore expected to be the whale or the foreground of the image. We can threshold the image using the minima that can be found between the two local maxima points – which would leave us with the pixels corresponding to the whale and the surrounding noisy pixels of waves/splashes.

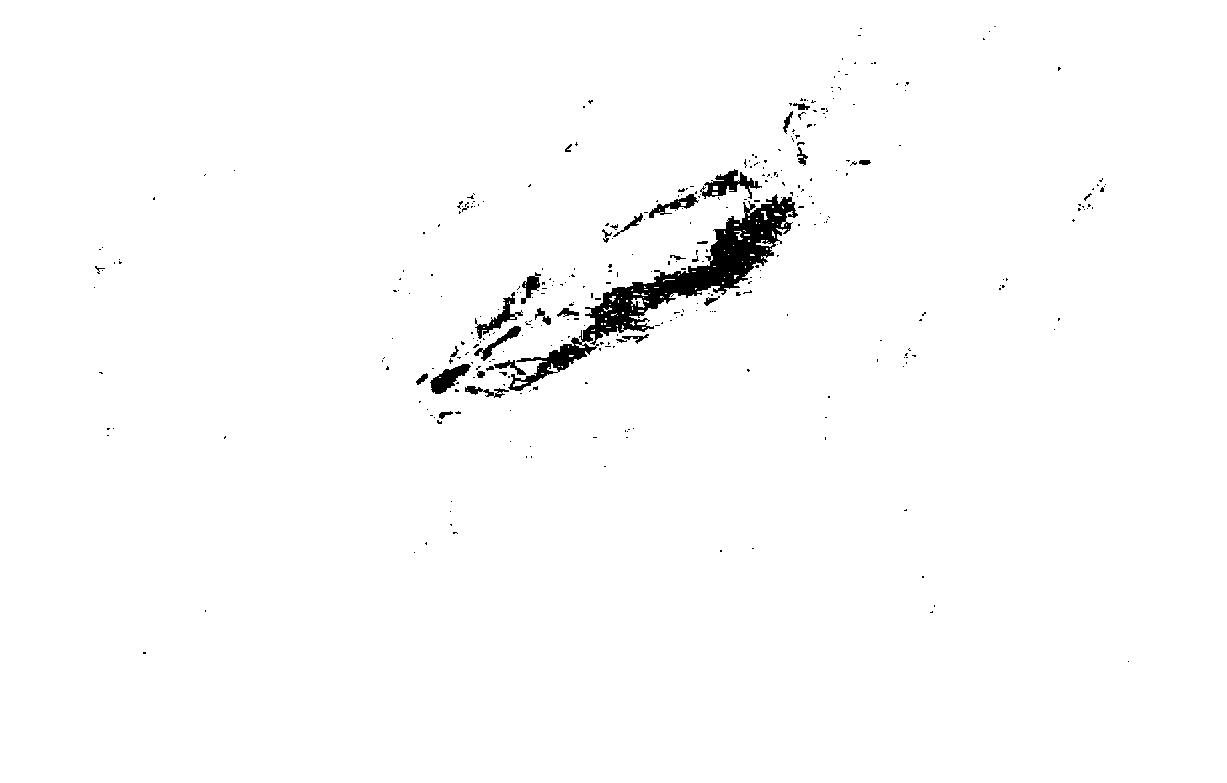


Figure X: Extracted mask of ROI from thresholding the saturation channel

Using this ROI polygon we can fix yet another problem with the data – the fact that the whales all face random directions. A solution to this particular problem is to enscribe an ellipse into the ROI polygon. The major axis of this polygon will roughly coincide with the major axis (head to tail) of the whale. Knowing the angle of rotation we can use an affine transformation to rotate the image to our desired location. The whale will now be facing either up or down.

The entire process works well about 4 out of 5 times. The other times the noise in the water will mess up the histogram which will create an incorrect ROI which in turn will result in a misleading rotation and crop of the image. Even in the successful runs of the preprocessor we were left with the problem of not knowing whether the whale was facing up or down – which mean that we could not further crop out the remainder of the whales body since the face is the most interesting feature.

Since we were not able to reliably preprocess the raw images using conventional Computer Vision techniques we turned to manual preprocessing of what we desired the output to be:

* Rotate the image to have the whale facing right.
* Cut out a square area with the whale dead center.
* Resize image to 256x256

These steps ensured that we obtained the “passport” photos of that the CV preprocessor should have made.

**Convolutional Neural Network models used in classification**

As part of the project we have evaluated many different CNN models. The biggest constraint of trying new models and tuning the associated hyper-parameters were training time, processing power and lack of sufficient computer memory. Due to design of Google TensorFlow a large chunk of memory is pre-allocated to be used in its computation graph [Ref: tf api]. None of our models could fit into 2 GB GPU memory that was available, which could have provided a huge speedup over training on CPU.

All of the model topologies are outlined in Appendix X.

1. DumbNet – our first model (retroactively named DumbNet) was completely of our own design. It consisted of 5 convolutions each followed by a pooling layer. It was based on theoretical understanding of stacking convolutional and pooling layers. The model relied too much on pooling (after every convolution) and many of the neurons died out from over-saturation. We concluded that the model was not complex enough as it failed to converge while training.
2. AlexNet [Ref Krizhevsky U of T] – this network was based on a successful network from literature which Alex Krizhevsky et al used to win the ImageNet ILSVRC2010 competition. The network was difficult to work with due to the non-standard convolution and pooling layers which changed the size of the output image in a way very different from other networks. This network did not converge either.
3. VGGNet [Ref: very deep... oxford] – successful net from a team from Oxford University, it won first place in the ILSVRC2014 competition. The net operates on the use of small stacked convolutions with fewer pooling layers in between. The authors argue that a stack of three 3x3 convolutions activated by ReLU (rectified linear unit) [ref: relu] activations can be more discriminative than a single 7x7 convolution. The problem we experienced with our interpretation of the VGG network was of the massive computational cost of running it. The network consists of 13 convolutional layers, 5 pooling layers and 2 wide fully connected layers before finally coming to the classifier neuron layer. Our computers were not able to reasonably run the network.
4. DeepSenseNet – this network was inspired by the winner of this Kaggle competition [ref: Deep Sense blog]. The authors must have themselves drawn inspiration from VGG as the network seems to be a simpler version of that one. We were able to obtain adequate results from this CNN after about 12 hours of training. We used an exponentially decaying learning rate and normalizations of the activation levels of every convolutional layer. The results were over-fit to the training data with about 80% classification accuracy on the training set and 15% on the validation set. However the result was enough to prove statistical significance of the classifier.

**Results**

To set the baseline for neural networks, kNN classifier with cropped images has been used. The cropped and rotates containing only the nose of the whale were used (image size 256 x 256. See above for details).

Firstly kNN classification was applied to raw feature vectors i.e. vector of unrolled image pixel values. A number of different k values have been used, namely k=1, 3, 5, 7 and 9. Euclidean distance was used as a measure of similarity. The same train + validation data split as for ANN has been applied to test the kNN classifier, so this way results would be comparable.

The accuracy achieved with this setup was relatively poor – the best accuracy was for k=1. Full table below.

In order to improve accuracy, a couple of different dimensionality reduction and feature extraction techniques have been performed. The first one was PCA. After applying PCA, the size of the feature vector in the PCA feature space has been reduced nearly 80 times (from 65536 to 831 features). The number of principal components has been further reduced to only account for around 92% of variance in the data. The resulting FV for an image was [1 200].

Results indicate there was a slight increase in accuracy after PCA analysis. The accuracy has increased to for k=1. Nevertheless, the performance of such classifier is far for satisfactory.

Further feature extraction has been performed using Linear Discriminant Analysis (LDA). Using the combination of PCA and LDA has shown significant improvement in the results. The accuracy has increased to for k=1. Through experimental trials the accuracy has been further increased by changing similarity measure to ‘Chebychev distance’. It is hypothesized, that in LDA feature space Chebychev distance gives advantage over Euclidean, as it only takes into account the most significant feature. With the latter setting, the accuracy was for k=9

Below is the full table of kNN classification results:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| kNN classification results.. #Runs = 20 per k | | | | |
| **k** | **RAW** | **PCA** | **PCA+LDA** | **PCA+LDA (chebychev)** |
| k=1 | 0.2097 | 0.2419 | 0.5726 | 0.5968 |
| k=3 | 0.1290 | 0.1371 | 0.5484 | 0.6048 |
| k=5 | 0.1290 | 0.1452 | 0.5242 | 0.6129 |
| k=7 | 0.1290 | 0.1210 | 0.4677 | 0.5806 |
| k=9 | 0.0968 | 0.1129 | 0.4919 | 0.6210 |

TODO: results of CNN

**Successes**

We can demonstrate that for simple problems such as the MNIST handwritten digit recognition (see Appendix X), we can easily outperform the baseline kNN approach using a CNN. Even though we have failed to demonstrate better performance classifying our α-whales dataset using a CNN vs the kNN we can still produce statistically significant results of about 14% (look into proving statistical significance) classification accuracy on the validation set. This is much greater than a random guess of 1/38 = 0.0263, but not as high as the 20% value from kNN.

We have been able to show that many of the filters that we trained with our convolutional layers are in fact detecting the callosity patterns of the whales with different levels of activations (close to zero or closer to one) and with different types of filters. Some of the filters act as high pass filters – which act edge detectors and some as low pass smoothing filters.

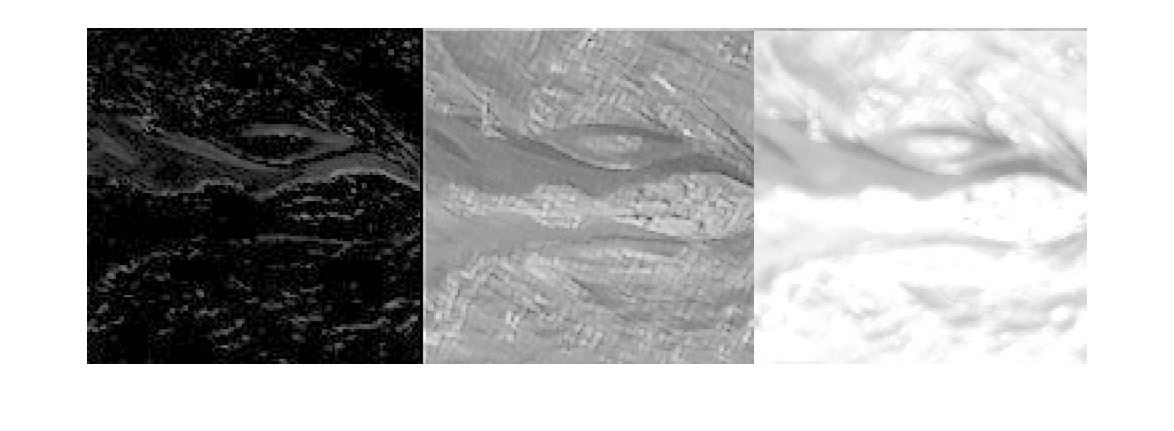


Figure X: Output of a sample neuron layer.

1. High pass filter. Low activation levels overall (close to zero mean) does not show the callosity pattern at all and the activated patterns are mostly noise from the waves. Perhaps the outline of the whale could be useful to deeper layers.
2. Low pass filter. Average activation levels (as seen by the gray levels), very clear callosity pattern and the features from the waves have been smoothed.
3. Low pass filter. Very high overall activation. Similar features to (b) with a good callosity pattern but with slightly more noise – this filter is probably still useful.

Another success of the project is to note that we have qualitatively verified that stacks of small convolutions such as the ones proposed by [ref: Oxford VGG] are easier to work with and converge much faster than networks with large convolution kernels. We suspect that many successful projects in the short term will adopt a similar approach, in fact the winner of the Kaggle competition – Deep Sense used a simplified VGG net.

**Shortcomings**

Our DeepSenseNet was overfit to the training data, this can be easily interpreted from the high classification accuracy on the training set and low accuracy on validation data. Another major problem is the arbitrary validation set, we did not use k fold cross-validation as it would require multiplying training time by the multiplier k. Additionally we have observed saturation in some of the convolutional layers even after normalizing each one of them.

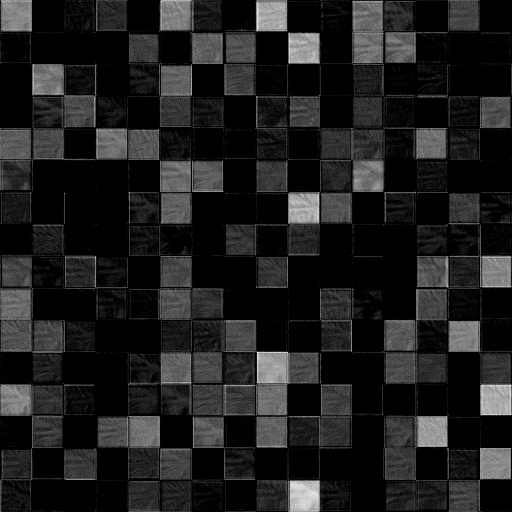


Figure X: Output of a convolutional layer

Each one of the squares in the above image corresponds to a feature detected by a feature in the conv net. Many of the squares are completely black meaning the neurons are dead and cannot be brought back even with very strong multipliers. This is because of a known property of the ReLU activation function of multiplying negative result by zero [ref to some paper]. Other activation functions have other downsides also considered during the design phase of this project.



Figure X: Saliency maps of two distinct images

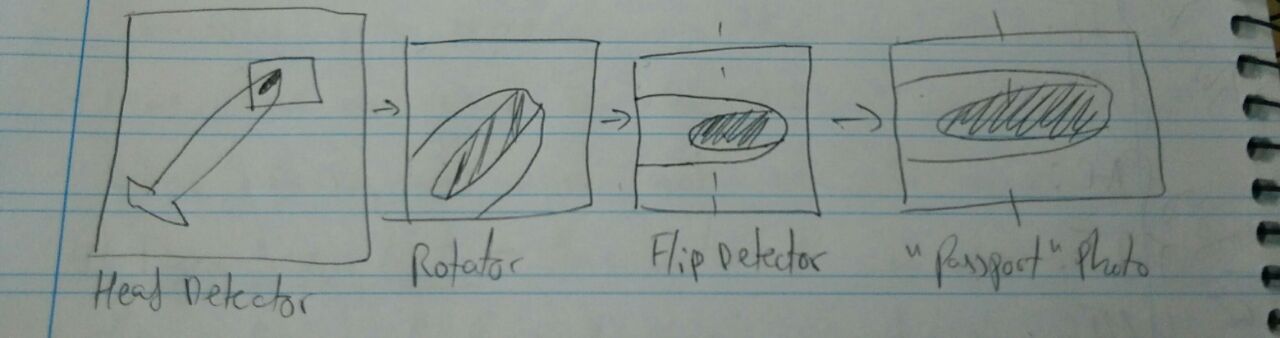
(drawn as heat map with red contributing more to classification than blue)

To analyze the features that our network was learning from we drew a saliency map of a sample image. The results were disappointing as the hot areas of the saliency maps were showing that noisy areas were contributing more to classification than the desired features.

**Possible Improvements**

***Preprocessing***

We were unable to perfect our preprocessing algorithms using conventional computer vision techniques. A solution to utilize computational intelligence to the whale detector. We could take the full set of 11000 images and manually tag the location of the whales head on some of them. Using this supervised data we can train a “whale-head-detector” This algorithm combined with our working rotation algorithm will give us a whale that is roughly aligned to the horizontal axis. At this point the whale will be facing either left or right and we can train another classifier to flip the images to the correct orientation.

Figure X: Proposed whale image preprocessing system

This working preprocessor can then be run on the full set of data – labeled and unlabeled. Ideally this larger dataset should be more trainable without overfitting the data then our α-whales dataset.

***Augmenting the training data***

After obtaining the full training data we would be left with 4500 labeled images of 427 whales – still not enough data to detect some of the whales. We can expand the labeled images set using some augmentation techniques. For every image we would run it through some filters which are meant to signify the variance between the images of the set and then use filter outputs as additional training data. Some of the filters we would use: low pass filter (smoothing), high pass filter (edge detection or sharpening) and various affine transforms such as rotation about the horizontal axis, scaling and pixel-wise shift.

***Improved hardware***

A huge weakness of our project was the lack of cross validation and the constraints imposed on us not being able to train more complex networks in a reasonable amount of time. Training on a video card with a large amount of memory or on the cloud would enable us to iterate through hypothesis much faster.

**Conclusion**

We have confirmed that CNNs classify images that pose significant difficulty to untrained humans. While the results were less than ideal they were sufficient to show the feasibility of the solution. We conclude that the problem is too computationally intensive to solve using unprocessed imagery. Therefore we propose a way of chaining machine learning driven preprocessing system. We have compared many different neural topologies and we concur with the trend in literature of using small kerneled stacked convolutions. Ultimately we must note that the heavy computational requirements remain a limiting factor for the usefulness of deep learning techniques in image recognition.