1. Theory

Baseline - kNN    [Deividas]

In order to compare our neural network performance, we have built a kNN classifier. This classifier builds a model from training images by arranging them in a d-dimensional space by some distance measure. In order to classify a test image, it then finds k training images, which are closest to a test image and uses most frequent label as the solution[1].

Each image in the set is represented by a feature vector (FV). FV in the simplest form could be a vector of unrolled grayscale image pixel values. For an image with dimensions, the FV is.

As presented later in the results, the performance of kNN classifier on raw data was very poor. So in order to achieve better classification accuracy of the kNN classifier, two different dimensionality reduction and feature extraction techniques were used:

1. Principal Component Analysis (PCA) - Aims to explain variance in data by transforming the data to a new set of uncorrelated features, the principal components (PCs)[2]. PCs have much lower dimensionality and preserve all of the original variance. They are ranked (descending order) by the amount of variance they explain, so removing last n PCs, most likely noise and unimportant data is scrapped.
2. Linear Discriminant Analysis (LDA) – a feature extraction technique originally developed by R. A. Fisher in 1936. The algorithm is based on searching for a linear combination of variables that best separate two classes. A generalized version of LDA [3] has been used, which allows working with multiple classes. LDA is similar to PCA in the sense, that they both try to extract linear combinations of variables which best explain the data [4]. However LDA attempts to model differences between different classes of data, which seems to be a perfect tool for trying to distinguish between faces of different whales.

[1]- Coomans, D., & Massart, D. L. (1982). Alternative k-nearest neighbour rules in supervised pattern recognition: Part 1. k-Nearest neighbour classification by using alternative voting rules. Analytica Chimica Acta, 136, 15-27.

[2] - Jolliffe, I. (2002). Principal component analysis. John Wiley & Sons, Ltd.

[3]- Ji, S., & Ye, J. (2008). Generalized linear discriminant analysis: a unified framework and efficient model selection. Neural Networks, IEEE Transactions on, 19(10), 1768-1782.

[2]- Martínez, A. M., & Kak, A. C. (2001). Pca versus lda. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 23(2), 228-233.

  CNN    [Deividas]

  **Backpropagation    [Andrei P]**

Citation: Mitchell, T. (1997). *Machine Learning*. New York: McGraw-Hill.

The backpropagation algorithm is used while training to propagate the error corrections from the output layer down through all the connected layers of neurons. This will ensure that the output result on the next epoch of training will provide a better score.

The main idea of the algorithm is using gradient descent to calculate the required changes of the current weights to reach a value with a minimal error difference from the required output. If the output vector was defined to be a dot product of the weights vector and the input vector the following equation would be found:

The training error of this linear unit (preceptor without the thresholding function) can be calculated by many functions – a simple way is summing the squared difference (t – refers to ideal output):

It is important to realize the correspondence of the weights to the training error will be some sort of hyperparabola – the specific shape of which will of course depend on the dataset. Here is an image of one such imagined parabola with only two weights. The global minima of the parabola will correspond to the best possible weights.

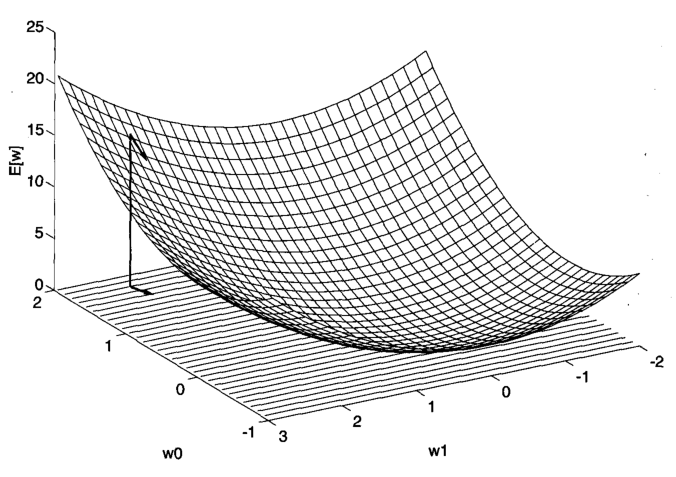


Figure X: Possible parabola graph of neuron errors w.r.t. the weight values

TODO: redo diagram in MATLAB (it’s a screenshot from the textbook)

Taking the gradient of the error vector with respect to each of the weights:

If one were to evaluate with the current weights – the result would be a vector showing the steepest increase of slope away from the current point. The negative of that would be the steepest decrease. Therefore new weights can be updated with the following delta:

is the learning rate. Setting to a greater value can result in a quicker convergence, however it can overshoot the global minima, while setting it too low will result in a very slow training process.

Gradient descent is not guaranteed to reach the global minima, however this is less of a problem when N is large for N-dimensional data as there will be more “routes” of sliding down towards the minima with a larger number of weights. Backpropagation is simply applying gradient descent recursively from the outermost layer all the way through the network.

**Adam Optimizer**   [Andrei P]

Citation: Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

An improvement on the classical gradient descent algorithm would be to maintain inertia while descending towards the minima. The simple gradient descent algorithm can be prone to radically changing directions of descent if the data is noisy. 1st and 2nd order moments can be added to increase the chance of staying on the true path while training.

The Adam (adaptive moment estimation) algorithm is described by (Kingma & Ba, 2014). It is a state of the art method for training. It utilizes two extra variables while calculating the weight updates:

– exponential moving average of the gradient ()

– exponential moving average of the elementwise squared gradient ()

On the first iteration both of the vectors are set to zeroes. After wards they will be calculated using their corresponding exponential decay rates and epsilon .

On each iteration of the Adam gradient descent first the biased moment estimates are found:

The next step is to compute bias corrected moment estimates by dividing by one minus β to the power of t:

Finally the new weights can be recalculated:

  Softmax  [Andrei M]

  Dropout   [Andrei M]

  Sigmoid, relu   [Andrei M]

2. Dataset

  Size of classes (bar chart) [Andrei P]  
  **Expected features** [Andrei P]

Citation:

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

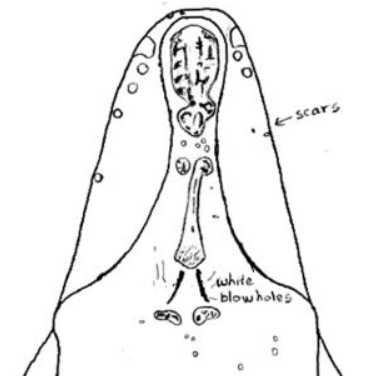


Figure X: Possible facial features of an individual whale specimen

The features that we are most interested in are the facial markings on top of the head of the whale and the white markings above the blowholes. These features are generally very different from whale to whale and they seemed like the best features to try to base our detection on.

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

**Dataset Preprocessing** [Andrei P]

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.

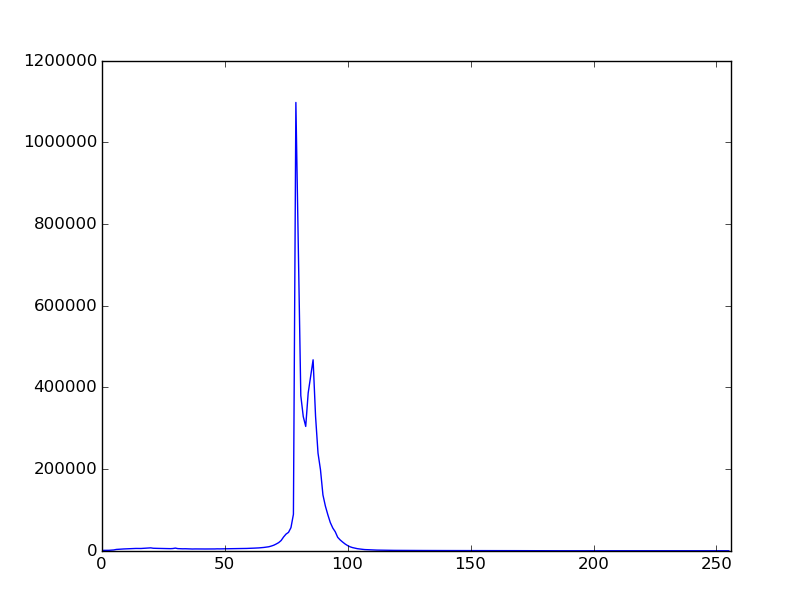
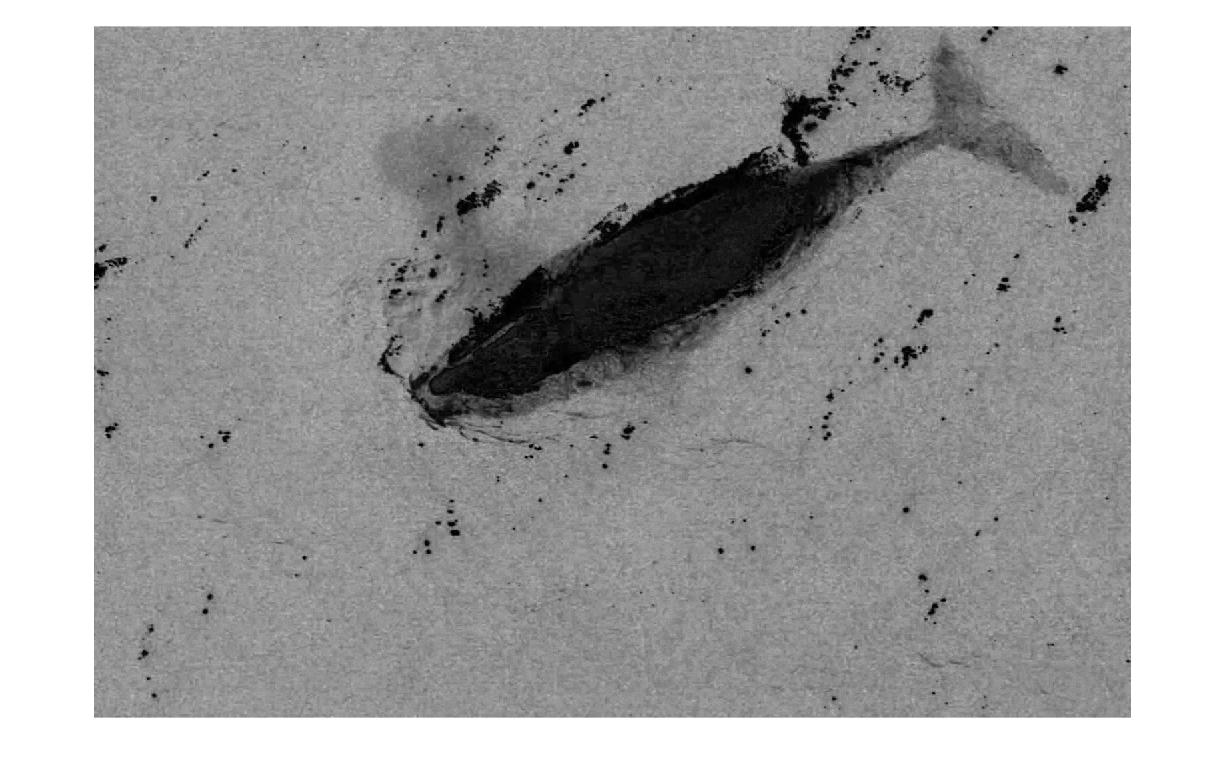


Figure X: Saturation from image and histogram of saturation channel

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.

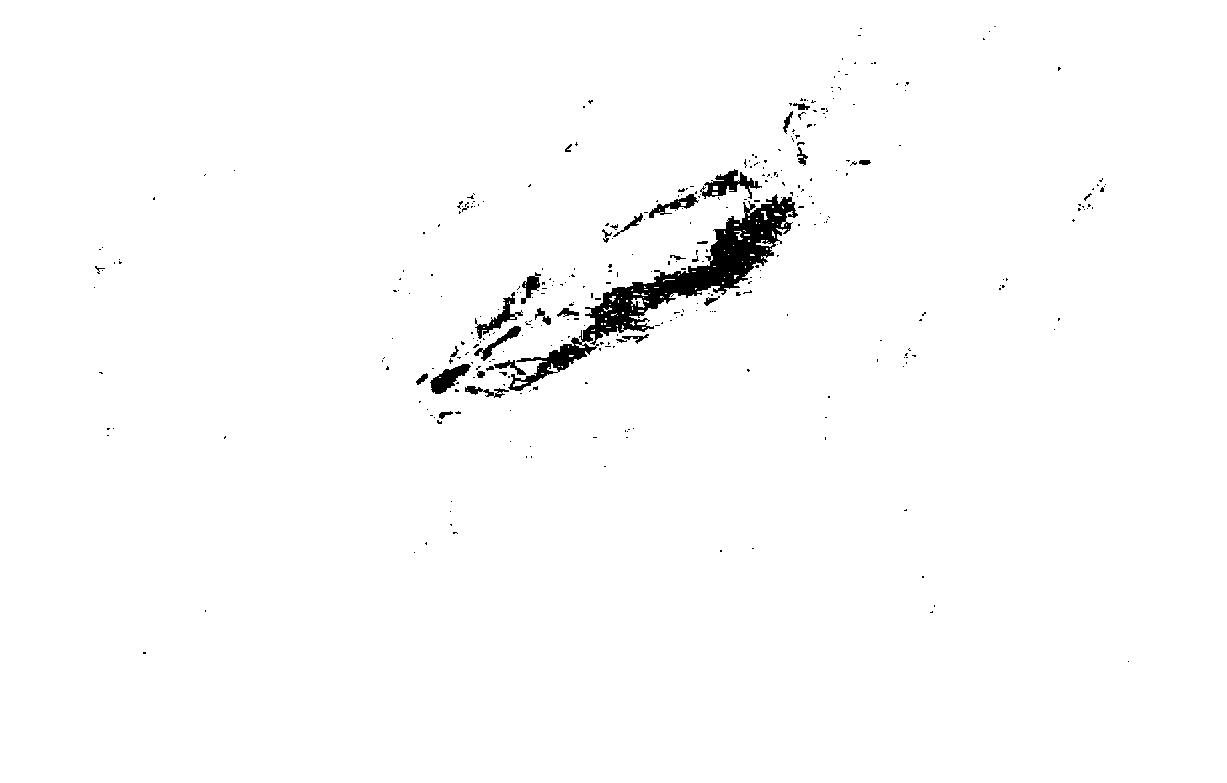


Figure X: Extracted mask of ROI from thresholding the saturation channel

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.

A possible solution we could have implemented would have been to train a simple classifier on this dataset which could have detected the position of the whales head. We could have then used that information to crop out the smallest possible images of the whale heads on the entire dataset.

  Noise (image of filters from CNN)   [Ilmira]  
  New

dataset

     -  Data augmentation, shift      [Andrei P]

**ZCA Whitening Transformation** [Andrei P]

Citation: Krizhevsky, Alex, and Geoffrey Hinton. "Learning multiple layers of features from tiny images." (2009).

We have tried using the zero component analysis whitening algorithm outlined by Krizhevsky and Hinton [?]. This transformation maximally decorrelates the axis of multi-dimensional data from each other while maintaining the important image characteristics. Krizhevsky and Hinton claim that this sort of preprocessing will result in faster convergence while training deep neural networks.



Figure X: Image after ZCA whitening

 3. Experiment

   MNIST example   [Ilmira]  
   Test set      [Ilmira]  
   Validation     [Ilmira]

   k-NN baseline      [Ilmira]

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 |

VGGNet   [Andrei P]

      -  compute number of parameters)

   Alexnet   [Ilmira]

      -  table with parameters tested

   DumbNet    [Ilmira]  
      -  table with parameters   
      -  image of filters

      -  plot of entropy

 4. Analysis

   Why it doesn't work

   What we would do if we were rich and had a lot of GPU