UNIVERSITY OF WARWICK

Machine Learning in Computer Vision - 15 Month Report

Matt Smith

December 6, 2016

1 INTRODUCTION

We have been working on two main projects over the last few months. We were invited to be a co-author of a paper, Cytometry Part A journal which publishes papers on cell characteristics. This was in response to placing in the top 10 of the HER2 scoring contest `http://www2.warwick.ac.uk/fac/sci/dcs/research/combi/research/bic/her2contest/`. The paper is due to be submitted by the end of the year. The outline of our contribution to the challenge is dicussed in section 2.

We will also target a special issue of the Institution of Engineering and Technology (IET) journal which will focus on computer vision technqiues for animal biometrics `http://digital-library.theiet.org/files/IET_CFP_CV_ABIO_FINAL.PDF`. The deadline for the paper is 31st January. For this we go back to the work on the right whale competition issued by Kaggle. We have some novel techniques for normalization of whales in aerial photographs with applications which we strongly believe will generalise to similar machine learning tasks in computer vision. The method and preliminary results are discussed in section 3.

Regarding future research, I have enhanced my skills in parallel programming by attending a 1-week intensive course on programming with GPUs (`http://people.maths.ox.ac.uk/gilesm/cuda/`). Neural networks layers are embaressingly parrallel and this course has given me fundamental training to test new layers for machine learning tasks. Specifically I have prototyped a 3D version of fractional max pooling (fmp), which we eventually would like to test on medical datasets to see if it's performance gains are as noteworthy as it can be in 2D. This layer was originally introduced by my previous supervisor, Dr. Ben Graham, so I will hopefully be able to collaborate effectively with him on this task. We dicuss the layer and our proposed applications in section 4. Dr. Graham is also looking at releasing a Torch7 verision of his Spatially-Sparse convolutional neural networks in the near future which could also be a potential avenue for research [9].

I attended the British Machine Vision Conference (BMVC) in York this September gone which lasted four days. I had invaluable experience networking and listening to fellow deep learning researchers in vision, which took the lion's share of the poster sessions and talks.

Regarding frameworks, I have been using Torch7 [5] heavily over the last 14 months and have recently moved to TensorFlow [2]. I believe diversifying will only improve the scope for what I will be able to research in the future.

2 HER2 SCORING IN BREAST CANCER HISTOLOGY IMAGES

2.1 OVERVIEW

In this machine learning challenge the aim was to accelerate machine learning development for automating scoring of human epidermal growth factor receptor 2 (HER2) in whole-slide-images (WSIs) of breast cancer histology slides. HER2 over expression acts as as a significant prognostic in growth of malignant cells. The aim of is to produce state of the art algorithms would eventually reduce the workload of a pathologist. In this contest there are 52 pairs of WSIs with linked ground truth data of which contains two labels; the HER2 score and a percentage score given by the pathologists. The evalution is based on the classification accuracy of 28 WSI pairs of unlabled test data. Descriptions of the first of the two labels are given in 2.1 along with typical regions corresponding to each of the scores in 2.1, please see e.g. [22] for further details of the scoring process.

Staining score	Staining Pattern	Staining Assessment
0	No membrane staining observed or lies in less than 10% of the tumour cells	Negative
1	Faint, weak membrane staining in some proportion of the tumour cells	Negative
2	Feeble, non-uniform membrane staining in greater than 10% of tumour cells	Equivocal
3	Vigorous, intense membrane staining is observed in more than 10% of tumour cells	Positive

Table 2.1: Criteria for scoring HER2 based on cell membrane staining

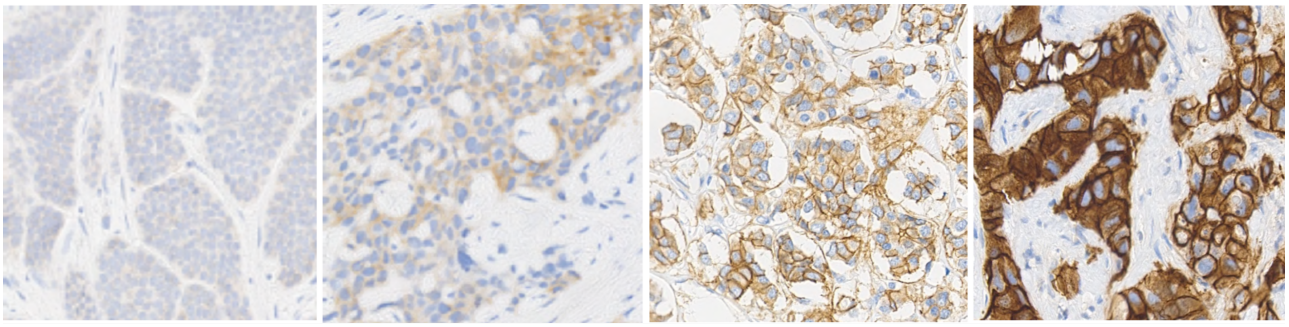


Figure 2.1: Patch examples in order of increasing score.

Each histology sample image is taken using a Hamamatsu NanoZoomer C9600 scanner, which generates a gigapixel image in order of $100,000 \times 80,000$ pixels. The file is stored in a lowpass pyramid structure, where the highest resolution image is convolved with an appropriate smoothing filter and subsampled by a factor of two. This smoothing and subsampling procedure is repeated several times, creating on average 10 images, each of which is $1/4$ of the size of the previous. We use openslide to easily access different parts of the pyramid structure [7].

For a given WSI we generate several samples of subsections or patches of the image from a given resolution. This is then passed through a convolutional neural network to compute our scores. We now go on to outline our procedure.

2.2 Classification algorithm

The RumRocks classification algorithm has two principal stages: preprocessing and classification. Both stages employ 2D convolutional neural networks [16] [15]. We achieved a top 10 position in the competition which was evaluated based the accuracy of our predictions over the 28 test images.

Firstly, the given test WSI image i is processed using a DCNN [21] and a CNN, as well additional image processing techniques, to generate a set of images P_i^N of size N containing patches of width w and height h from level L^* of an image pyramid. In the final stage, we use r random subsets of size n, $P_i^n \subset P_i^N$, to represent the WSI's regions of interest, each of which creates a tensor of size $(n \times w \times h \times 3)$. The r subsets are separately passed through a final CNN to generate the HER2 scores which are averaged over r. The three networks are denoted DCNN_1, CNN_1 and CNN_2 respectivley.

2.3 Preprocessing

Preprocessing a WSI test image i is as follows (for a visualisation see figure 2.2):

- Using a low resolution representation of the image (level L from the image pyramid), we pass it through a DCNN_1 to generate a binary mask.
- The binary mask is split into equally sized disjoint square patches each of size $w' \times h', w' = h' = w(2^{(L-L^*)})^{-1}$. Padding is added as necessary to ensure the image size is a multiple of w'. Binary patches are discarded if they are less than 50% in area of interest.
- The subsampled patch coordinates are translated to level L^* of the pyramid, where they are used to extract regions of interest from the high resolution version.
- The high resolution regions are discarded or kept using CNN_1 which aims to further reduce any noise present.
- Selected patches P_i^N are passed through to the classification stage.

2.3.1 Training and hyperparameter choice

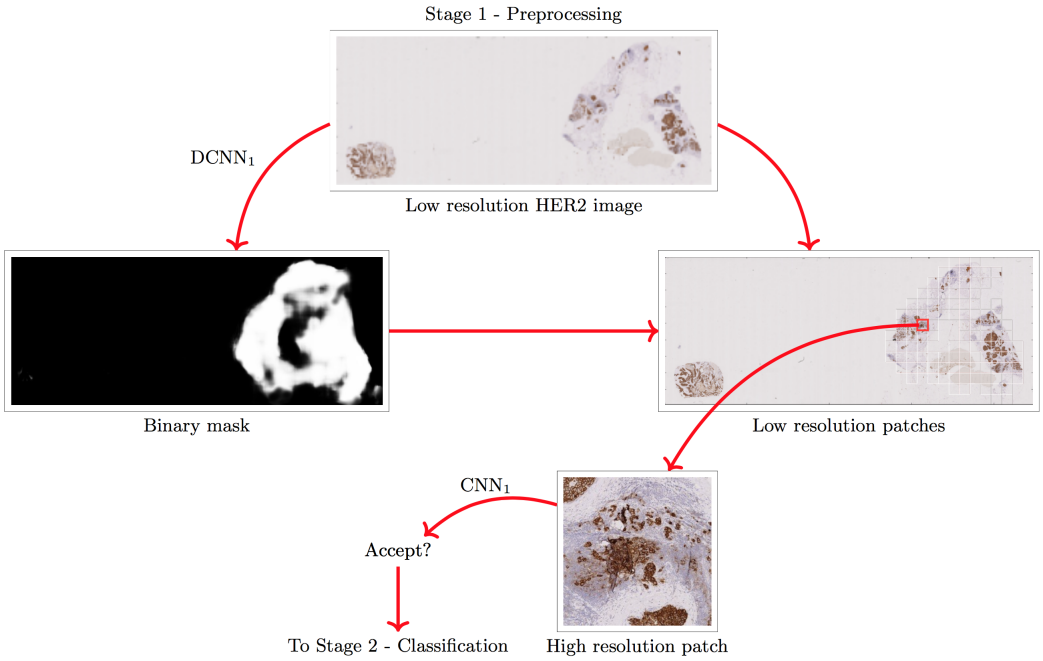
For DCNN_1, we use handcrafted binary masks from $L = 7$ of each image in the training set, with the HE image as a reference guide. Training masks are generated using image preprocessing techniques. The network learns a mapping between the input image and binary output mask as this step is required to automate the removal of irrelevant artefacts and background. The distribution of the region of interest is largely distinct compared to the irrelevant/control artefact, hence it can be be learned and and removed.

For CNN_1, we generate hand made binary labels from L^* for around 2000, $(128 \times 128 \times 3)$ example patches. The network learns to accept or reject a patch based on its contents. It is useful to have an additional filter to remove any further noise which the binary mask is unable to remove.

We augment images before they are fed into the networks using small translations, rotations, scaling and reflections in order to reduce overfitting. Both networks are trained using the mean squared error loss function and Adam optimizer [13].

2.4 Classification

The classification of a test WSI image i proceeds as follows:



Network architectures:

$$\text{DCNN}_1 = \{\text{Down}_1 -, ..., - \text{Down}_6 - \text{Up}_1 -, ..., - \text{Up}_5 - \text{C}_{2D}3/1 - \text{Sigmoid}\}$$
$$\text{CNN}_1 = \{\text{Down}_1 -, ..., - \text{Down}_7 - \text{Reshape} - \text{FC} - \text{Sigmoid}\}$$

$$\text{Down}_\text{n} = \{16\,\text{nC}_{2D}3/1 - \text{BN} - \text{ReLU} - \text{MP}3/2\}$$
$$\text{Up}_\text{n} = \{16(6-\text{n})\text{C}_{2D}3/1 - \text{BN} - \text{ReLU} - \text{UP}2\}$$

Figure 2.2: Stage 1 - Preprocessing: Steps to generate patches of interest from WSI. The notation of network structure is similar to [8]: {a-b-c} denotes a network with 3 layers where the input is passed through a then b and finally c. $fC_{iD}k/s$ denotes an i dimensional convolutional layer with kernel size k in each dimension, a stride of s and number of filters f. Similarly MP$k/s$ is a two dimensional max-pooling layer with kernel size k and stride s. Other layer notations; BN = batch normalization, UP2 = upsampling layer by factor of 2, FC = fully connected layer, ReLU and Sigmoid are layers of rectified linear activation units and sigmoid activation units respectively.

- Draw and augment n patches with random reflecions and small translations without replacement from P_i^N to generate P_i^n.
- Pass tensor P_i^n through CNN_2 to output the two predicted scores.
- Repeat the above two steps r times and average predictions.

2.4.1 Network structure

To help with optimization and due to impressive performances on the ImageNet dataset, the CNN_2 structure includes residual layers [10] (see figure 2.4). They argue that by allowing layers to be structured in a residual mapping deep networks are easier to optimise. It is proposed that deep networks should produce no higher training error than its shallower counterpart, however degradation problems exist in training where shallower networks obtain lower training errors. This indicates an optimisation issue which residual

"blocks" aim to address. Combining many of these blocks essentially generates another feed forward network structure but with shortcut connections which allow units to skip one or more layers. A residual block is shown visually in figure 2.3. Formally we can define a block as

$$\mathbf{y} = f(\mathbf{x}, \{\mathbf{W_i}\}) + \mathbf{I}\mathbf{x}, \tag{2.1}$$

where in figure 2.3 $f = \mathbf{W_2}\sigma(\mathbf{W_1}\mathbf{x})$, σ is an activation function (e.g. ReLU) and $\mathbf{I}$ is an identity function. The number of weight matrices for this block can be of arbitrary length, as long as the output f is conformable with $\mathbf{X}$ for an element-wise operation, which in this case is addition, this is the shortcut connection. If this is not the case then a linear projection $\mathbf{W_s}$ can be used instead of $\mathbf{I}$ in the shortcut stage to match the dimensions of f.

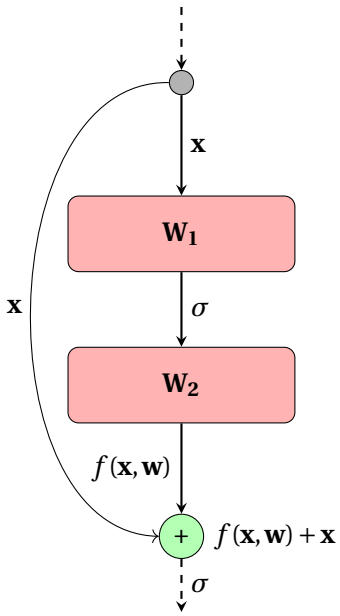
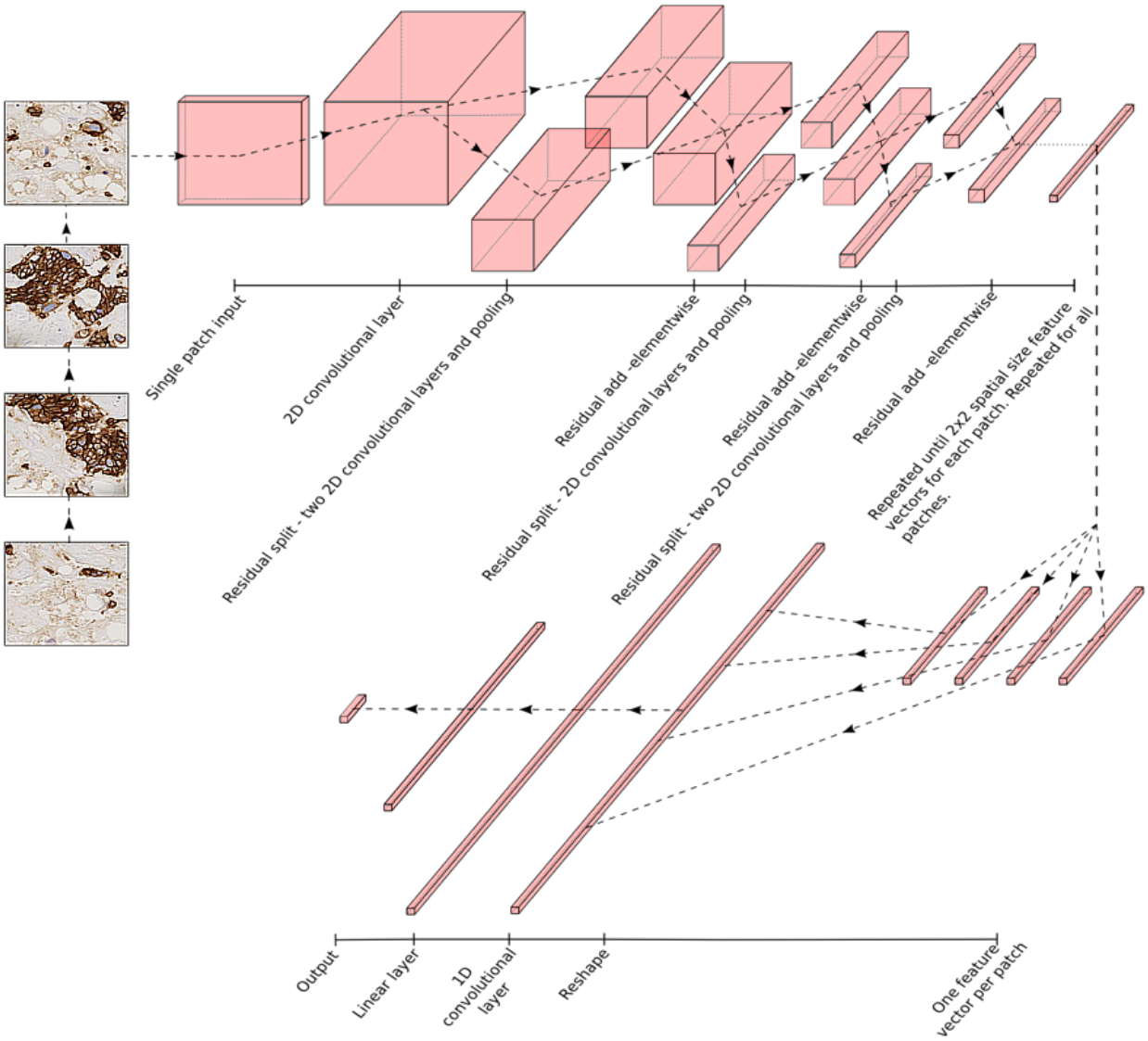


Figure 2.3: Visualisation of a residual block

We exploit the batch dimension in order to feed in multiple patches from the same example WSI simultaneously using the same network; figure 2.4 depicts the network receiving $n = 4$ patches. Instead of creating n separate predictions for the score of each patch and averaging the predictions, we reshape the tensor to a vector once the spatial size has been significantly reduced and forward it through a 1D convolution and linear layers. Intuitively these final layers look for particular combinations of information in each patch instead of a weighted average. The latter method could be problematic if a low proportion of patches contain the information necessary to correctly classify the true score. Furthermore, by drawing n samples from N_i r times we are able to reduce sampling bias during testing.

The final output tensor is a vector of size 4 which represents the score using an ordinal binary encoding using the first 3 elements (final score is the sum of these elements), the last is used for the percentage stain score. This enables us to penalise a misclassification of 0 (true class 3) more than a misclassification of 1 (true class 2) as well as keeping the output range $\in (0,3)$.

We believe by exploiting the batch dimension and looking for non-linear combinations of patches through 1D convolutional and linear layers, we are creating a novel network which could have a wide range of uses for single class large images where voting is required. It could be a future task to look at how the responsiveness of high level neurons change depending on the particular combinations of patches, as well as potentially applying the network to more WSI datasets to show its potential. Although not done here, it would be of interest to see the effect on performance of this batch voting technique and if time permits this would be a good experiment to try.



Network architecture:

$$\text{CNN}_2 = \{16C_{2D}3/1 - \text{BN} - \text{ReLU} - \text{resBlock}_1 - ,..., - \text{resBlock}_7 - \text{Flatten} - C_{1D}1/1 - \text{BN} - \text{ReLU} - \text{FC} - \text{ReLU} - \text{FC} - \text{Sigmoid}\}$$

$$\text{resBlock}^1{}_n = \{16nC_{2D}3/1 - \text{BN} - \text{ReLU} - \text{MP}3/2\}$$
$$\text{resBlock}^2{}_n = \{16nC_{2D}3/2 - \text{BN}\}$$
$$\text{resBlock}_n = \{(\text{resBlock}^1{}_n \oplus \text{resBlock}^2{}_n) - \text{ReLU} - 16nC_{2D}3/1 - \text{BN} - \text{ReLU}\}$$

Figure 2.4: Stage 2 - Classification: Architecture of CNN_2 with $n = 4$. $\oplus$ denotes element-wise addition between two tensors.

3 Right Whale Recognition

3.1 Motivation

It is of upmost importance to researchers and conservationists to monitor the status of individuals in a threatend population. Accurate monitoring of the distribution and abundance of animal species over time is a key ingredient to successful nature conservation [28] [3]. Human observation is expensive and limits the breadth of data collection [29]. Image sensors are increasingly being used in biodiversity monitoring, with each study generating many thousands or millions of pictures. Autonomous ariel photography is becoming an increasingly useful technique regarding animal biometrics [14] [19]. Efficiently identifying species and individuals captured by each image is a critical challenge for the advancement of this field [30].

We are targetting an IET journal which focuses on animal biometrics in computer vision. Last year we entered an online competition hosted by Kaggle which required creating an algorithm to distinguish between individual north atlantic right whales in aerial photographs. An example of a typical photograph is given in figure 3.1.

Figure 3.1: Typical input

We revisit this challenge and improve on some of our techniques; using a combination of interesting methods to locate, normalize and classify ariel photographs of these whales. The main methods of interest, based on location and normalization include; a histogram matching algorithm to normalize for sea/whale color to aid location, a principal axes based technique to normalize for the whale's direction and a semi-supervised method for making up for limited labelled data. We believe these methods can be used very effectively in numerous ariel vision tasks.

The competition involves 447 unique right whales for which we were asked to create a "face recognition" system. There are 4543 training and 6925 test images. Evaluation of the test set is based on the multi-class logarithmic loss

$$\text{logloss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{i=1}^{M} y_{ij}\log(p_{ij}), \tag{3.1}$$

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

In modern face recognition, the normal approach consists of four stages: detect, aligin, represent and classify [26]. Arguably the best feature when making a classifier for this particular species of whales are their heads which contain rough patches of skin. We therefore proceeded to tackle the challenge in two main stages for a given photo, both involving the use of deep learning methods.

- Location and normalization - An algorithm is used to reduce the dimensionality of the image and

adjust the orientation and distance to the whale's head. This generates a passport photo of the whale as well as a semantically segmented mask.

- Classification - The head shot is then passed through a classification stage to identify the whale by outputting a probability mass function over the 447 whales.

3.2 Whale Normalization

It is helpful to remove variation inputs before giving them to a deep learning algorithm [17]. Using a graphics editor we hand label 700 images generating a mask for each one such that we have a data set $D = (\{X_1, Y_1\}, \{X_2, Y_2\}, ..., \{X_{700}, Y_{700}\})$. An example of a labelled pair is given in figure 3.2.

Figure 3.2: Example of $\{X, Y\}$ pair.

We distinguish between head, body and sea by labelling the head red, the body yellow and everything else black. Having two colors for distinguishing parts of the whale enables us to infer the direction of the whale. We attempt to normalize the i^{th} image X_i for several other factors; orientation, scale and location by passing it through a series of processing stages as well as a fully connected network, denoted F_1 [18]. Semantic segmentation tasks are mainly driven using labels relying on convNets [11]. Our network generates a predicted mask M_i which is processed further by using a combination of other image processing techniques inlcuding; principal axis and affine transforms, generating our desired passport photos.

3.2.1 Normalization Algorithm

- Denoting the raw i^{th} image as X_i^0, we rescale it to $w \times h \times c$ (width × height × channel) using bilinear interpolation and we use histogram matching to normalize the cumulative distribution function (cdf) of each YUV channel of each image, yeilding X_i^1. Histogram matching [6] each channel of the YUV color space of each image to the respective YUV channels of a predetermined target image T, helps normalize for illumination and color. T and it's histogram (density plot) is given below in fig 3.3.

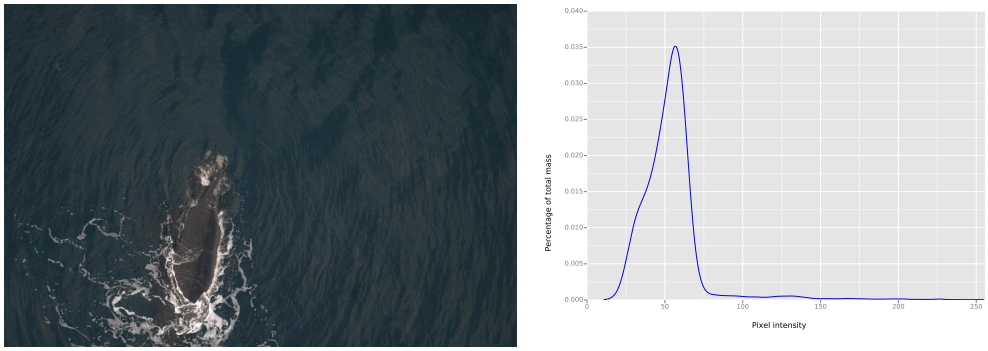


Figure 3.3: Target image and it's density

We intially tried to simulate images by modelling the joint distribution of YUV cdf mappings. Possible candidate we discessed included non-parametric models such as a Gaussian process. We ended up modelling the parameters $\Theta = (\theta_1, \theta_2, ..., \theta_7)$ in the following function

$$f(x;\Theta) = \theta_1 + \theta_2 \tanh\left(\frac{x-\theta_3}{\theta_4}\right) + \theta_5 \tanh^{-1}\left(\frac{x-\theta_6}{\theta_7}\right)$$

with the aim to simulate from the distribution of Θ and generate mutliple mappings for each photo. The reason for using a linear combination of Tanh and Tanh^{-1} being with certain constraints on Θ, we can ensure monotonicity, which is desirable property for cdfs. However, we found that learning a single exact mapping , i.e. mapping each image to T would prove much cheaper computationally.

We use histogram matching to reduce sea and whale colors down to a much smaller color range, in order to help the learning algorithm in the presence of limited labelled data. YUV is a color space which splits image information into luminance (Y) and color (U and V). From an RGB system we can calculate the equivalent YUV values via

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \tag{3.2}$$

Given a reference and target single channel array x_1 and x_2 a histogram matching method is explained in algorithm 1. An illustration of how a single value is remapped using this algorithm is shown in 3.4. X_i^1 is then transformed such that pixel values between [0,1].

Function *Histogram Match (x_1,x_2)*
  Result: Returns modified version of x_1 such that it has the same cdf as x_2.
  F_1 = cdf(x_1);
  F_2 = cdf(x_2);
  Initialize $M()$;
  for *i = 0,1,...,255* **do**
    $\text{M(i)} = \underset{j}{\text{argmin}} \left| F_1(i) - F_2(j) \right|$
  end
  for *pixel in* x_1 **do**
    pixel = M(pixel)
  end
  return x_1;

Algorithm 1: Histogram matching algorithm

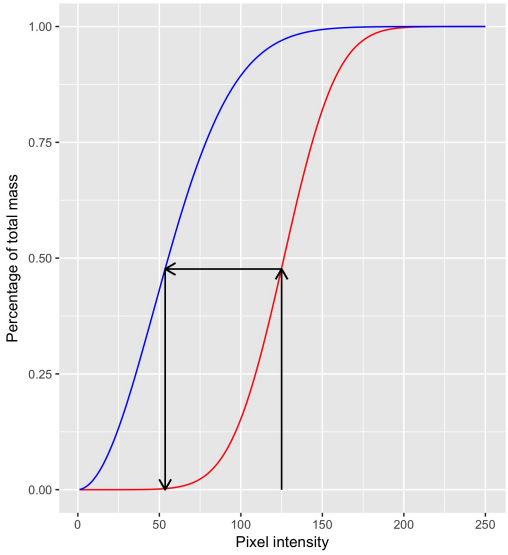


Figure 3.4: Histogram matching illustration: All pixel values in the source image (red cdf) equal to 125 are changed to 54. Both images have the same amount of mass less than these respective values before the transformation.

- X_i^1 is passed through F_1 denoted by

$$F_1 = \text{down}_0 -, ..., -\text{down}_4 - 3\text{C}_{2D}3/1 - \text{sigmoid}, \tag{3.3}$$

where

$$\begin{aligned} \text{conv}_\text{n} &= \{(48+32\text{n})\text{C}_{2D}3/1 - \text{BN} - \text{ReLU}\} \\ \text{down}_\text{n} &= \{\text{conv}_\text{n} - \text{MP}3/2\}, \end{aligned}$$

generating predicted mask M_i^0. The ground truth, Y_i is rescaled to $h'' \times w'' \times c'' = 19 \times 29 \times 3$ for calculation of the error via

An illustration of the network is given in figure 3.5 along with tensor dimensions at each stage.

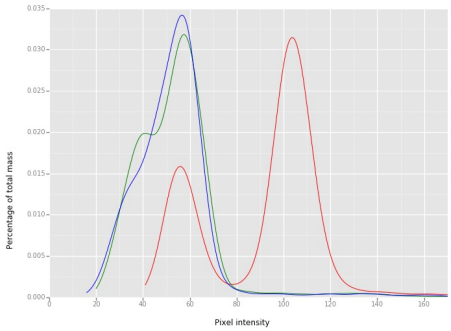
- M_i^0 is thresholded, blurred and the single largest elipse is found from the resulting contours of the image, generating M_i^1. This removes the possibility of having two whales as well as removing noise which would affect calculation of moments in the following step. Regarding the competition, it is reasonable to assume that the whale which takes up most of the image is the intended whale to be classified. An example of this is given in figure 3.6.

(a) X_i^0. (b) M_i^0. (c) M_i^0 with chosen ellipse outlined. (d) M_i^1.

Figure 3.6: Ellipse fitting is used to find single objects.

- The principal axes, red and green centroids of M_i^1 are calculated. For a single channel image function

(a) Source, destination and target images.



(b) Source (red), destination (green) and target (blue) densities.

$I(x,y)\, x = 0,1,...,w-1, y = 0,1,...,h-1$, its raw moments m_{ij} can be calculated as

$$M_{ij} = \sum_x \sum_y x^i y^j I(x,y), \tag{3.4}$$

and the coordintes of the centroid can be calculated as $\bar{x} = \frac{M_{10}}{M_{00}}$ and $\bar{y} = \frac{M_{01}}{M_{00}}$. Given 3.4 we can obtain the covariance matrix of a single channel

$$\text{cov}\big[I(x,y)\big] = \begin{bmatrix} \mu'_{20} & \mu'_{11} \\ \mu'_{11} & \mu'_{02} \end{bmatrix}, \tag{3.5}$$

where

$$\mu'_{20} = M_{20}/M_{00} - \bar{x}^2,$$
$$\mu'_{02} = M_{02}/M_{00} - \bar{y}^2,$$
$$\mu'_{11} = M_{11}/M_{00} - \bar{x}\bar{y}.$$

The principal axes, u_1 and u_2 can be found via the singular composition of 3.5

$$\text{cov}\big[I(x,y)\big] = U\Sigma V'.$$

Denote the centroids of the red and green channels as m_r and m_g, we can calculate the direction in which the mask of the whale is pointing $p = m_r - m_g$. Only if u_1 and p are pointing in different directions then we flip the direction of u_1.

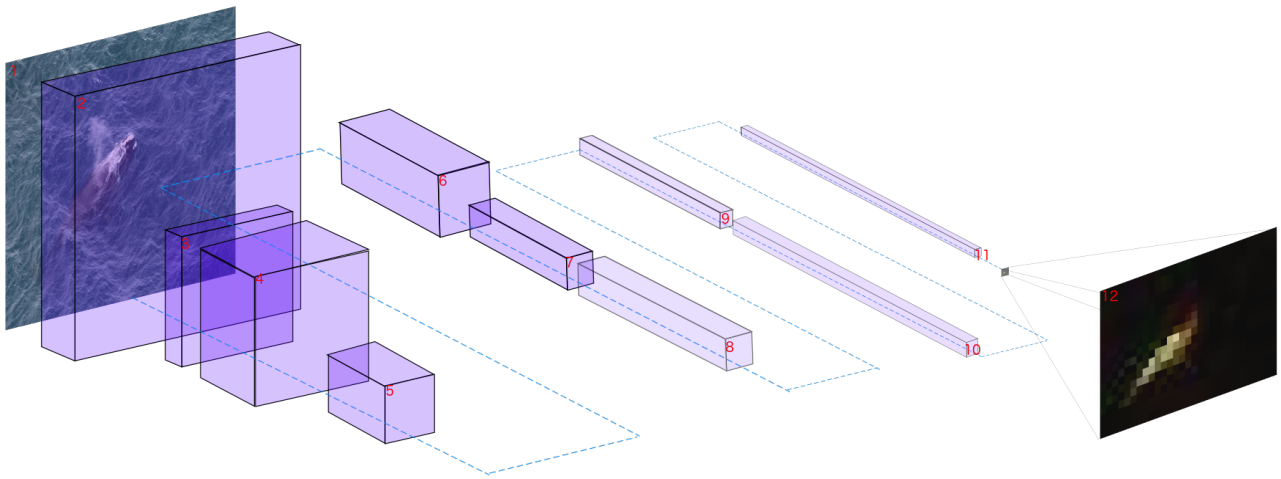
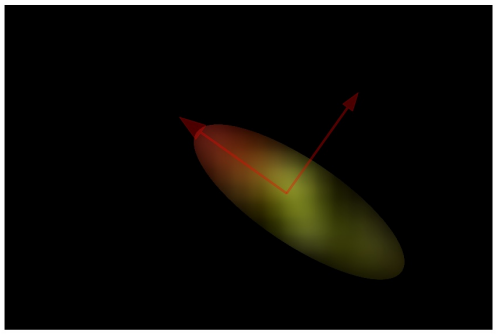


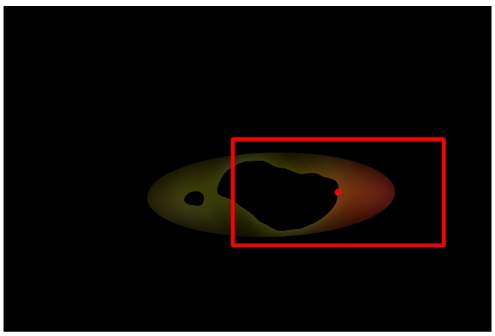
Figure 3.5:

No.	Size ($w \times h \times c$)	No.	Size ($w \times h \times c$)	No.	Size ($w \times h \times c$)
1	$900 \times 600 \times 3$	5	$225 \times 150 \times 80$	9	$38 \times 57 \times 144$
2	$900 \times 600 \times 48$	6	$225 \times 150 \times 112$	10	$38 \times 57 \times 176$
3	$450 \times 300 \times 48$	7	$113 \times 75 \times 112$	11	$19 \times 29 \times 176$
4	$450 \times 300 \times 80$	8	$113 \times 75 \times 144$	12	$19 \times 29 \times 3$

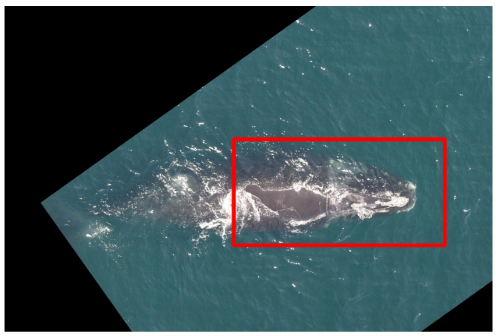
(a) X_i^0.



(b) M_i^1 with arrows depicting the principal axes.



(c) M_i^2 with area of interest highlighted.



(d) X_i^2 with area of interest highlighted.

(e) X_i^3

Figure 3.7: Rotation and cropping.

The angle of rotation, θ is now given by

$$\theta = \text{atan2}(u_{12}, u_{11})$$

- We rotate M_i^1 by θ around m_r. Then we select highest intensity red regions using thresholding to find the head, generating M_i^2. The mode of the red channel of M_i^2 is then calculated and an area of interest is formed around it to generate the coordinates of the head shot area of interest with dimensions $w' \times h' \times c'$.

- We apply the same affine transform to the original X_i^0, producing X_i^2 and using the coordinates generated in the previous step we obtain our headshot photo X_i^3. A single example of the rotation and cropping stages is shown in figure 3.7. We also illustrate the whole process for four images given in figure 3.2.1.

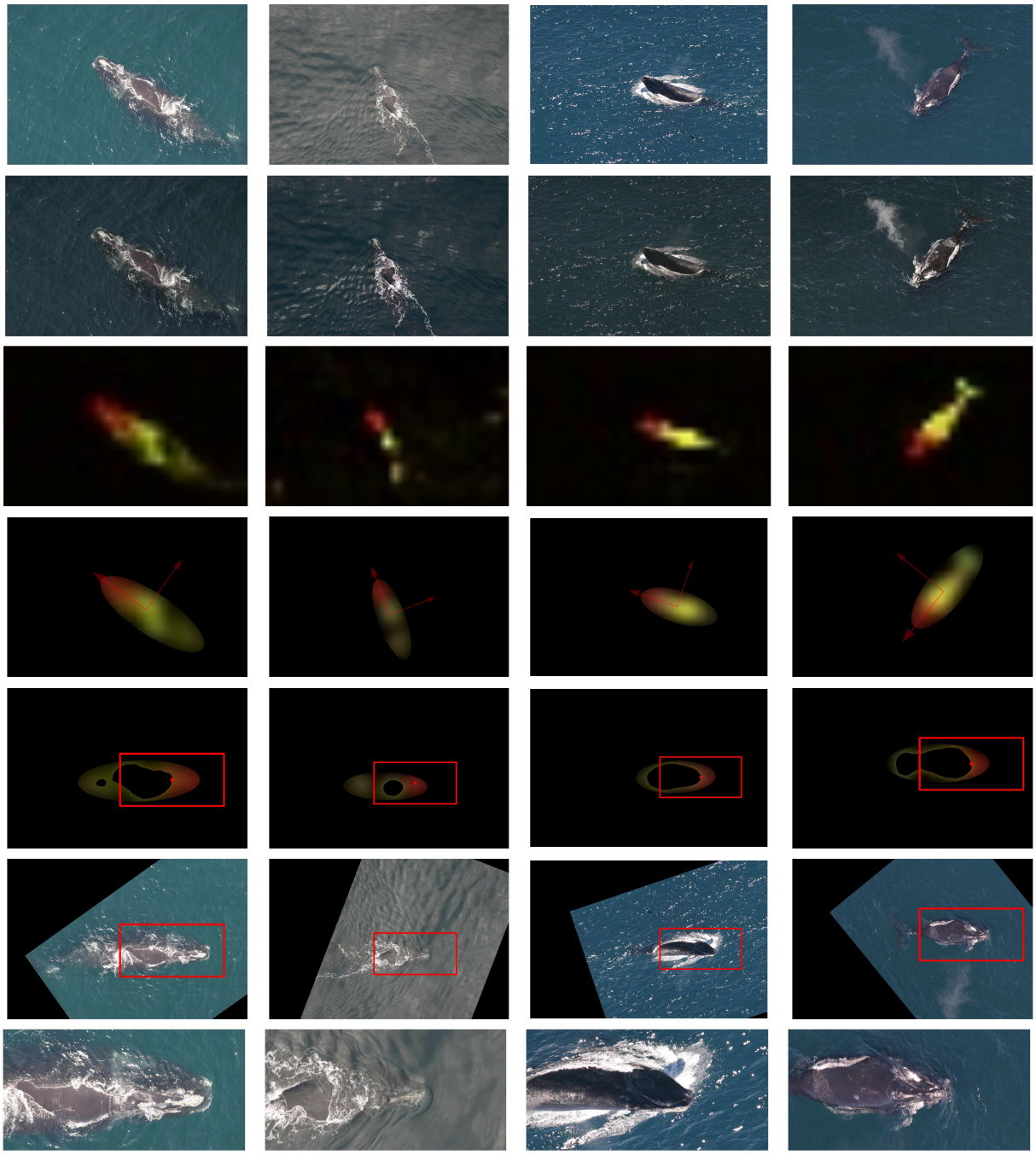


Figure 3.8: Four examples of the processing pipeline for the normalization stage - from top to bottom; $X_i^0 - X_i^1 - M_i^0 - M_i^1 - M_i^2 - X_i^2 - X_i^3$.

3.2.2 Training and hyperparameter choice

To artificially increase the size of the dataset we augmented the training set by 100 fold by combinations of shears, flips, rotations and translations as show in fig 3.9.

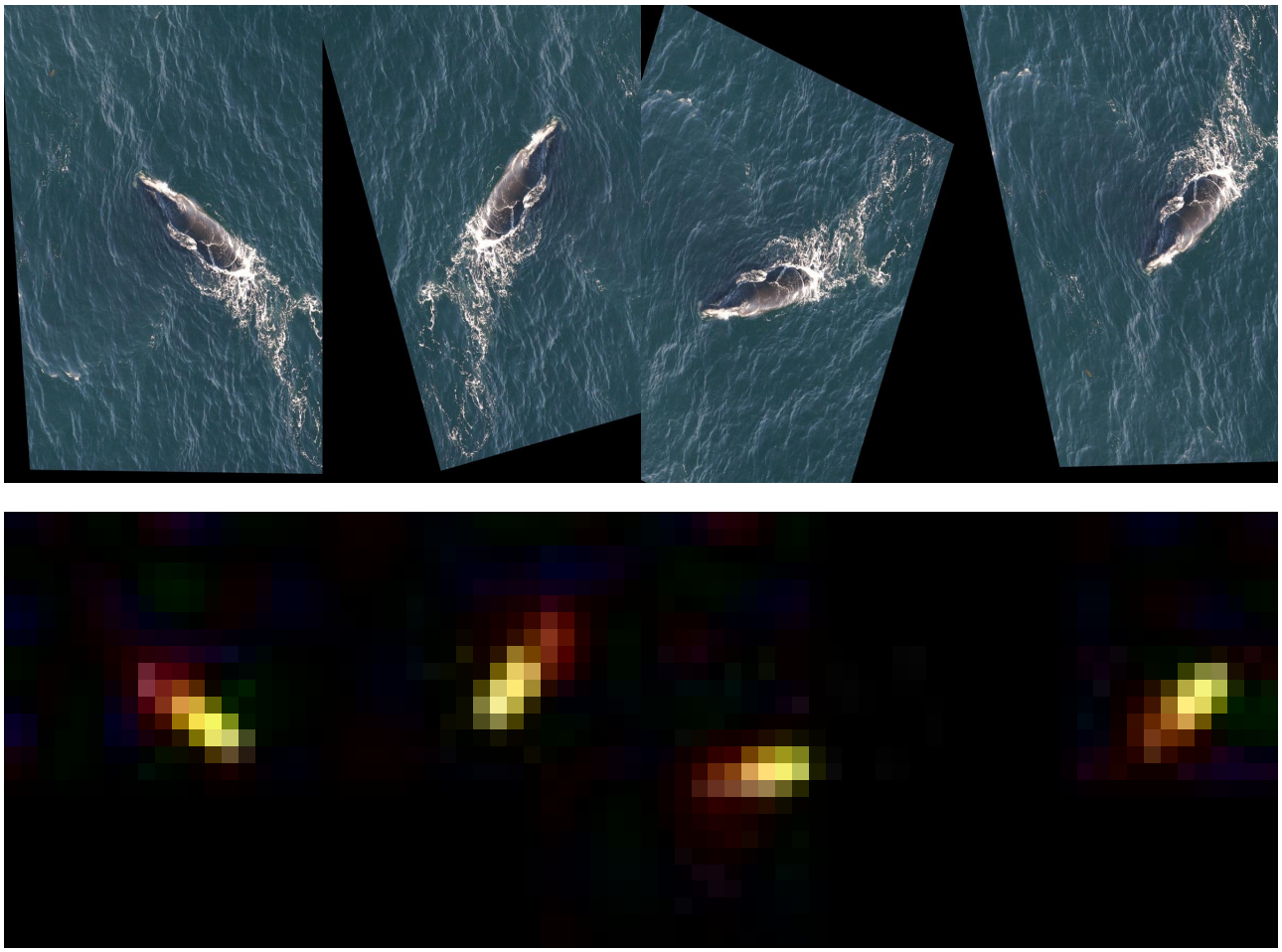


Figure 3.9: Four augmentation examples of the same image and its mask.

Model selection includes choosing the output dimensions of X_i^3 as well as the structure of the network given by 3.3. We choose our model by splitting our training set into a sub train test set via a 4:1 split, i.e. we have approximatley 150 test images. We use the mean squared error (mse) loss function and Adam optimizer to learn the parameters of the network. Togther with the mse we also look at a variant of the Sørensen-Dice coefficient to compare model performance [24] given by

$$\mathrm{QS}(Y, \hat{Y}) = \frac{2|Y \cap \hat{Y}| + 1}{|Y| + |\hat{Y}| + 1}. \tag{3.6}$$

The dice score is a popular performance measure in image segmentation to compare similarities of two objects. To compute the dice score we need to convert each channel of Y_i and $\hat{Y}_i$ into a binary tensors. This requires the latter to be thresholded, which we choose to be 0.5, as the output of the final layer is continuous (sigmoid) and the scores are then averaged over each of the RGB channels. If time permits we would like to do a ROC equivalent for the dice score at multiple thresholds to give a more robust view of performance. We can see from 3.10 and 3.11 that histogram matching has a positive effect on test performance.

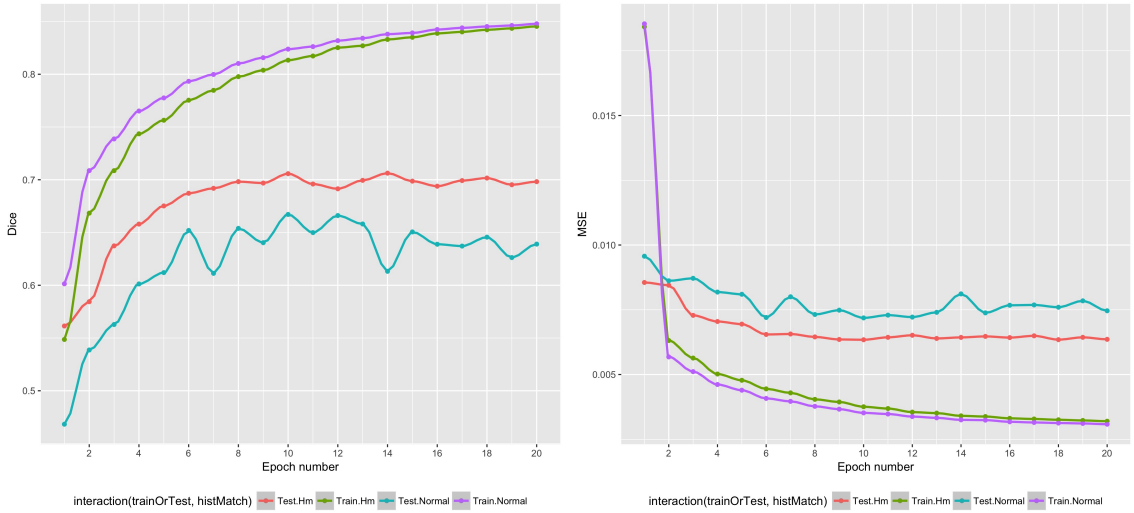


Figure 3.10: Histogram matching effect on train and testing performance metrics; dice score (threshold of 0.5) and mean squared error.

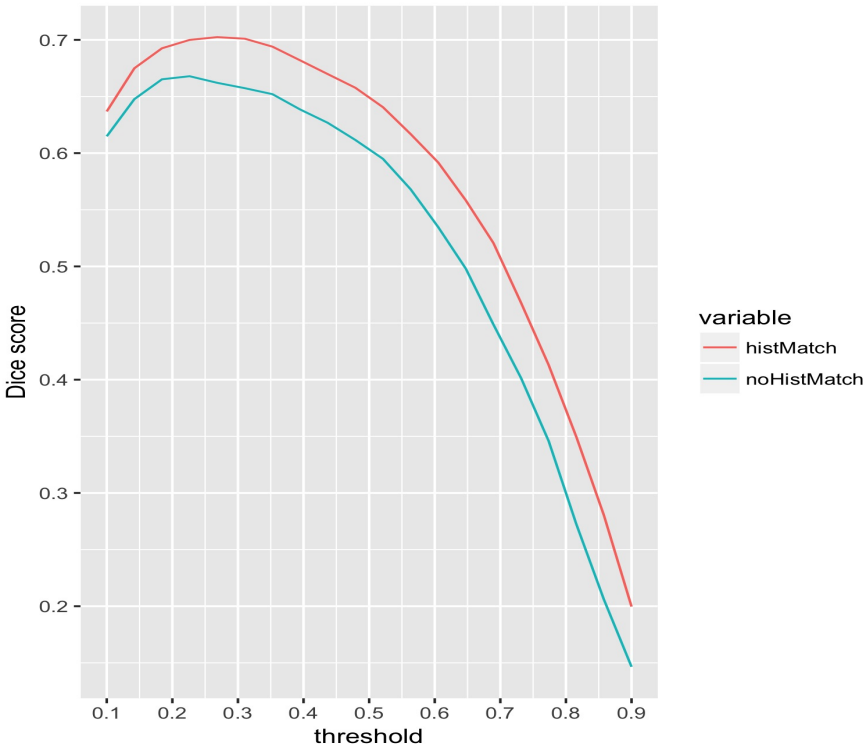


Figure 3.11: Histogram matching effect on dice score at final epoch for multiple threshold values.

3.3 Whale Classification

Each headshot X_i^3 of size $w' \times h' \times c'$ is then fed into the final classifier F_2 who's structure is given by,

$$F_2 = \mathrm{b}_1 - , \ldots, - \mathrm{b}_6 - \mathrm{reshape} - \mathrm{linear} - \mathrm{ReLU} - \mathrm{linear} - \mathrm{SoftMax}, \tag{3.7}$$

where

$$b_n = \{16nC_{2D}3/1 - BN - ReLU - MP3/2\},$$

outputing a 447-dimensional vector, simulating a pmf over the individual whales. We use the cross entropy loss function and Adam optimizer. Again data augmentation techniques are used including small rotations. Results using our new method of localization are to be finalized.

3.3.1 Areas for improvement

We describe a few parts of the task which could have been improved, with possible outlines of solutions;

- Probably the biggest improvement to train F_1 and consequently F_2 would be to have more and higher quality labels. Our hand labelling is not perfect and we can mislabel accidently or subjectively parts of the ocean we think are whale and vice versa which are not true. We have proposed an alternative solution to time consuming labelling via a form of semi-supervised learning.

  Semi-supervised learning (SSL) is arguably good for limited good quality labels. SLL is halfway between supervised and unsupervised learning and works on the assumption that similar data points have similar labels. Training deep networks requires a large quantity of segmentation ground-truths which result from tremendous human annotation efforts and costs [11]. The intial bottleneck for the task at hand is this lack of ground truths and it is not plausible to learn a huge number of parameters in deep networks reliably in this scenario [31]. Moreover our hand drawn masks are not perfect. The standard approach in SSL in our case is to update the model of our network by iteratively inferring and refining hypothetical segmentation labels using weakly annotated images in an EM type fashion, however there is no guarantee of convergence to a good solution [11].

  We attempted SSL for mask prediction but found a negative effect on our dice score. We do however believe that it could be unfair to test this via the dice score as the distribution of our true hand labels are different to the distribution of the semi-supervised labels. We will attempt to clarify whether SSL helps with the classification stage if time pemits.

- The possibility of having more than one whale in an ariel photograph, especially when it is ambiguous regarding which whale has been considered for the label, can be problematic. Dealing with this using a separating algorithm, which would effectively split an image which contains say two whales into two images where each image would then go on to be processed as normal. The resulting output probability mass functions could then be averaged to get an estimate with lower variance.

- We attempt to normalize for the direction of the whale and the size of the head in our final passport photo. We have not normalized for perspective, which could be of value as it is used in top performing algorithms such as deepface [26].

- For this species of whales it is generally aknowledge that the barnacles on their heads are the most distinguising features. However it may be that there can be some critical information lost in disregarding the rest of the whale. Improvements can be made through a potential second classifier which would look at the whole whale (if not occluded) and a weighted ensemble of a seperate body and original head classifier could be learnt through cross validation.

- Our choice of T, the histogram matching image, is largely subjective and we chose this as it had a relativley smooth histogram. It would be of interest to see how dice scores and resulting classification scores differ when choosing a different image, perhaps with a multimodal histogram.

- We did not want to punish the network as much for making an incorrect prediction between two

labels, head and body than for being in the sea. Hence distance-wise red and yellow are closer than yellow-black, however red-black and red-yellow have the same distance. We might try to improve this by making the head and body colors closer and see if this has a positive effect on our algorithm.

3.3.2 Advantages over other methods in animal biometrics

Succesful methods for the challenge included regression of coordinates of a box around the whale's head, see for example [1]. The output of this and many other top scoring methods typically constrain output to only four sets of coordinates for a whale's head. For future tasks there maybe a need to find multiple animals in a photo and it is not uncommon to find these animals with others; this is where a pixel wise network is at a large advantage as the predicted mask can be used to generate multiple passport photos of the whale. Furthermore it may be of use to certain groups (conservationists, biologists, zoologists) to know which part of the image is head or body and this method can be easily extended to label other parts of an animals anatomy e.g. fins, tail etc. If it is of interest to semantically label at a higher resolution, this can be acheived through deconvolutional layer usampling, perhaps by using a U-net type architecture [23].

4 Fractional Max Pooling in 3D

4.1 Motivation

As discussed, a future project will be to make a 3 dimensional version of Fractional Max Pooling (FMP) layer originally designed by [8]. In a two dimensional setting, convNets have traditionally used max pooling layers or a strided convolutional layer. FMP has been shown to generate superior performances in CIFAR-10 and CIFAR-100 [25] [8] [4]. It has been shown on three dimensional data sets that capturing temporal/depth as well as spatial information can lead to gains in performance [12] [27] [20]. It naturally follows that the structure should be extended to work in a 3D setting, for use in medical, video and other computer vision tasks where it would be natural to use 3-dimensional learners.

4.2 Description

A 2D max pooling functions typically have pooling regions of $\alpha \times \alpha$, $\alpha \in 2,3$. An example where $\alpha = 2$ is shown in figure 4.1.

$$X = \begin{pmatrix} 4 & 1 & 2 & 10 \\ 6 & 8 & 4 & 3 \\ 3 & 1 & 4 & 5 \\ 8 & 9 & 9 & 10 \end{pmatrix} \qquad MP2/2(X) = \begin{pmatrix} 8 & 10 \\ 9 & 10 \end{pmatrix}$$

Figure 4.1: Max pooling operation applied to matrix

Pooling layers aim to reduce the dimensionality of the task at hand by keeping strong activations from previous layers. Furthermore by discarding a significant portion of the data at each pooling layer in the network a large degree of invariance to translations and elastic distortions is generated. When $\alpha = 2$ these layers would discard 75% of data, however this data reduction can have performance limitations. With FMP the regions α are allowed to take non-integer values. There are two proposed types of FMP, disjoint and overlapping and we shall focus on the latter.

For simplicity consider a cube of dimensions $N_{in} \times N_{in} \times N_{in}$. From this input tensor we wish to generate an output tensor $N_{out} \times N_{out} \times N_{out}$ such that $N_{in}/N_{out} \in (1,2)$. We divide the input cube into N_{out}^3 pooling regions P_{ij}.

To generate pooling regions, let $(a_{i=0}^{N_{out}}), (b_{j=0}^{N_{out}})and(c_{k=0}^{N_{out}})$ be an increasing sequence of integers startingat 1 and ending with N_{in} and all increments equal to one or two. We define our 3D pooling regions as

$$P_{i,j,k} = [a_{i-1}, a_i] \times [b_{j-1}, b_j] \times [c_{k-1}, c_k]$$

The sequences are pseudorandom if they take the form

$$a_i = \text{ceiling}(\beta(i+u)), \qquad \beta \in (1,2), u \in (0,1).$$

Consider the 1-D FMP case where $N_{in} = 15$, $N_{out} = 10$. Firstly we must find β, which can be thought of as our average stride. Recall that the number of output neurons N_{out} when using a kernel of width k, stride β and input neurons N_{in} is given by

$$N_{out} = (N_{in} - k)/\beta + 1. \tag{4.1}$$

Solving 4.1 for β with we obtain $\beta = 13/9$. We present the case of $u = 0.2$ where we give the sequence, its differences and pooling regions respectively by

$$\left(a_{i=0}^{10}\right) = (1,2,4,5,7,8,9,11,12,14,15) \tag{4.2}$$

$$(a_i - a_{i-1})_{i=1}^{10} = (1,2,1,2,1,1,2,1,2,1) \tag{4.3}$$

$$\left(P_{i=1}^{10}\right) = (\{1,2\},\{2,4\},\{4,5\},...,\{14,15\}). \tag{4.4}$$

$$\tag{4.5}$$

Denote the function fmp($\boldsymbol{X}$) taking a random vector $\boldsymbol{X} \in \mathbb{R}^{15}$. Using our sequence in equation 4.2 we obtain

$$\text{fmp}(\boldsymbol{X}) = \left[\max(P_1), \max(P_2), ..., \max(P_{10})\right]', \tag{4.6}$$

where individual elements of FMP ($\boldsymbol{X}$) and the partial derivatives wrt. each input can be computed as follows

$$[\text{fmp}(\boldsymbol{X})]_j = \max_{k \in P_j} X_k$$

$$\frac{\partial [\text{fmp}(\boldsymbol{X})]_j}{\partial X_i} = \begin{cases} X_i & \text{if } i = \arg\max_{k \in P_j} X_k \\ 0 & \text{otherwise.} \end{cases}$$

4.3 Applications

I have prototyped a toy example in CUDA and aim to finish the full version once we have finished the whale paper. Apart from a software release of the layer, we will also aim to apply the layer to 3D medical data and this may possibly include revisiting the CT lung cancer dataset we previously worked on, LUNA16 `https://luna16.grand-challenge.org/results/`. We would attempt to improve on the performance of our algorithm through a swap of max pooling layers for fmp layers. The performance can be measured for the two tracks of the challenge including; false positive reduction as well as the nodule detection track. There are two main journals that we would like to target with this work; Medical Image Analysis, published by Elsevier and Transactions on Medical Imaging, published by IEEE.

REFERENCES

[1] Kaggle - right whale competition - 1st place solution. `http://blog.kaggle.com/2016/01/29/noaa-right-whale-recognition-winners-interview-1st-place-deepsense-io/`. Accessed: 2016-12-05.

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[3] Stephen T Buckland, David R Anderson, Kenneth P Burnham, Jeff L Laake, David L Borchers, and Len Thomas. Introduction to distance sampling estimating abundance of biological populations. 2001.

[4] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[5] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011.

[6] Dinu Coltuc, Philippe Bolon, and J-M Chassery. Exact histogram specification. *IEEE Transactions on Image Processing*, 15(5):1143–1152, 2006.

[7] Adam Goode, Benjamin Gilbert, Jan Harkes, Drazen Jukic, Mahadev Satyanarayanan, et al. Openslide: A vendor-neutral software foundation for digital pathology. *Journal of pathology informatics*, 4(1):27, 2013.

[8] Benjamin Graham. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, 2014.

[9] Benjamin Graham. Spatially-sparse convolutional neural networks. *arXiv preprint arXiv:1409.6070*, 2014.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[11] Seunghoon Hong, Hyeonwoo Noh, and Bohyung Han. Decoupled deep neural network for semi-supervised semantic segmentation. In *Advances in Neural Information Processing Systems*, pages 1495–1503, 2015.

[12] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013.

[13] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] L Koh and S Wich. Dawn of drone ecology: low-cost autonomous aerial vehicles for conservation. 2012.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[17] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[18] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[19] Julien Martin, Holly H Edwards, Matthew A Burgess, H Franklin Percival, Daniel E Fagan, Beth E Gardner, Joel G Ortega-Ortiz, Peter G Ifju, Brandon S Evers, and Thomas J Rambo. Estimating distribution of hidden objects with drones: From tennis balls to manatees. *PLoS One*, 7(6):e38882, 2012.

[20] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015.

[21] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.

[22] Emad A Rakha, Sarah E Pinder, John MS Bartlett, Merdol Ibrahim, Jane Starczynski, Pauline J Carder, Elena Provenzano, Andrew Hanby, Sally Hales, Andrew HS Lee, et al. Updated uk recommendations for her2 assessment in breast cancer. *Journal of clinical pathology*, pages jclinpath–2014, 2014.

[23] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[24] Thorvald Sørensen. {A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons}. *Biol. Skr.*, 5:1–34, 1948.

[25] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[26] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.

[27] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497. IEEE, 2015.

[28] Jan C van Gemert, Camiel R Verschoor, Pascal Mettes, Kitso Epema, Lian Pin Koh, and Serge Wich. Nature conservation drones for automatic localization and counting of animals. In *Workshop at the European Conference on Computer Vision*, pages 255–270. Springer, 2014.

[29] Ben G Weinstein. Motionmeerkat: integrating motion video detection and ecological monitoring. *Methods in Ecology and Evolution*, 6(3):357–362, 2015.

[30] Xiaoyuan Yu, Jiangping Wang, Roland Kays, Patrick A Jansen, Tianjiang Wang, and Thomas Huang. Automated identification of animal species in camera trap images. *EURASIP Journal on Image and Video Processing*, 2013(1):1, 2013.

[31] Xiaojin Zhu. Semi-supervised learning. In *Encyclopedia of machine learning*, pages 892–897. Springer, 2011.