Automatic Baby Feet Segmentation for Gestational Age Estimation

Mukul Rathi
School of Computer Science, University of Nottingham

***Abstract*— The objective is to localise and segment a baby's foot within an image, using Computer Vision and Deep Learning. This is the first step to automatically calculate the gestational age of babies using photographs of their feet. A Fully Convolutional Network (FCN) was trained using Stochastic Gradient Descent to classify each pixel as either part of a foot (white) or not (black), therefore segmenting the image and generating prediction masks. These predictions are compared to the ground truth mask using the Jaccard Index. The ground truth masks for each image were generated based on the outline of the foot, using the foot's 43 landmarks as reference. The FCN was able to segment the images very accurately, with a median Jaccard Index of 0.8561. Most notably, in some cases, the FCN was able to learn complex representations and even outperformed the ground-truth mask by producing segmentations that accounted for gaps between toes. This highlighted the immense potential of FCNs.**

I. INTRODUCTION

A. Problem: Gestational Age Estimation

The primary objective of the research is to estimate the gestational age of a baby at birth, thus diagnosing whether a baby is extremely pre-term, very pre-term, moderately pre-term, term, or late term. This can then be used as a guideline to decide the clinical treatment necessary. For example, moderately pre-term babies might need incubating, while extremely pre-term babies might also need to receive medication. The main methods of estimation can be categorised as follows:

1) Manual: The primary method of gestational age estimation is the use of ultrasound imaging. This is a non-invasive method of monitoring the growth of the baby during pregnancy, and is used by doctors to ensure the baby is developing normally. The advantages of this method include the early diagnosis of any potential problems at a foetus stage, which means clinical treatment or an altered course of action can be determined even before the baby is born. However, in developing countries, the cost of acquiring an ultrasound scanner is a major barrier, and so this method of estimation is unavailable for a large fraction of the world's population.

Another method of manually estimating the gestational age is through the Ballard Maturational Assessment [4]. This involves categorising each of the physical and neuromuscular characteristics of the baby and assigning a sub-score of -1 to 5, based on which category's description most aptly describes that characteristic displayed by the baby. The sub-scores are summed to give an overall score, which is then used to calculate the baby's gestational age at birth. Figures 1 and 2 illustrate this. The advantage of this method is the lack of medical specialised equipment required, thereby removing the cost barrier for medical practices in developing countries. However, the main shortcoming of this estimation is the subjectivity of the assessment, which can result in an inaccurate diagnosis if the clinician is not skilled or experienced enough.

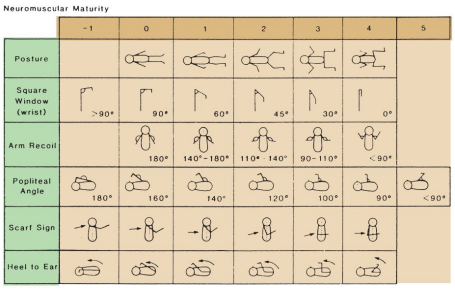

Fig. 1: *The scoring of the neuromuscular characteristics of the baby for the Ballard Maturation Assessment.* [2]

Physical Maturity

	-1	0	1	2	3	4	5
Skin	sticky friable transparent	gelatinous red, translucent	smooth pink, visible veins	superficial peeling &/or rash. few veins	cracking pale areas rare veins	parchment deep cracking no vessels	leathery cracked wrinkled
Lanugo	none	sparse	abundant	thinning	bald areas	mostly bald	
Plantar Surface	heel-toe 40-50 mm: −1 < 40 mm: −2	>50mm no crease	faint red marks	anterior transverse crease only	creases ant. 2/3	creases over entire sole	
Breast	imperceptible	barely perceptible	flat areola no bud	stippled areola 1-2 mm bud	raised areola 3-4 mm bud	full areola 5-10 mm bud	
Eye/Ear	lids fused loosely: −1 tightly: −2	lids open pinna flat stays folded	sl. Curved pinna; soft; slow recoil	well-curved pinna; soft but ready recoil	formed & firm instant recoil	thick cartilage ear stiff	
Genitals male	scrotum flat, smooth	scrotum empty faint rugae	testes in upper canal rare rugae	testes descending few rugae	testes down good rugae	testes pendulous deep rugae	
Genitals female	clitoris prominent labia flat	prominent clitoris small labia minora	prominent clitoris enlarging minora	majora & minora equally prominent	majora large minora small	majora cover clitoris & minora	

Maturity Rating	
Score	weeks
−10	20
−5	22
0	24
5	26
10	28
15	30
20	32
25	34
30	36
35	38
40	40
45	42
50	44

Fig. 2: *The scoring of the physical characteristics of the baby for the Ballard Maturation Assessment.* [2]

2) Automatic: This method of estimation involves photographing images of the baby and inputting this into a computer, which then will use deep learning, the forefront of machine intelligence, to automatically output an estimation of the gestational age of the baby. The advantages of this are that it does not require any specialist knowledge or any medical equipment, as the estimation could occur with an easily available smartphone app. The outreach of this technology is aided by the growing prevalence of smartphones in developing countries. This lowers the barrier of adoption of this technology, thus enabling medical practices in developing countries to use this to estimate the gestational ages of babies more effectively, and thus result in a better

course of clinical treatment if the baby were to be born significantly pre-term or post-term.
This is the method that will be researched, as part of improving gestational age estimation.

II. LITERATURE REVIEW

A. Segmentation

1) Using Support Vector Machines: The traditional method of segmentation and classification is to manually choose features from the image, which can then be input into classification algorithms such as logistic regression and Support Vector Machines (SVMs). However, the performance of these algorithms is heavily reliant on the features selected, and as such the performance can be severely hampered by bad feature selection. The features for each image differ - a good feature for one image may not be a good feature for another image. Thus, the process of manually selecting images is time-consuming, and also prone to subjectivity, as the judgement of whether a feature is good or not varies from person to person. These judgements could also be erroneous, since what constitutes a good feature is not always immediately apparent. A SVM with a Gaussian Kernel could be used, with the kernel defined as:

$$k(x, x') = \exp(\frac{- \parallel x - x' \parallel^2}{2\sigma^2})$$

Picking each of the landmarks x'_j would still be a point of contention, although for the initial training the 43 landmarks supplied in the training set could be used. However, this method relies on these landmarks for testing as well, since the features $f_j^{(i)}$ extracted from the image $x^{(i)}$ are computing the similarity function between the input and the landmarks. Thus, this method of segmentation is not feasible for automatic detection as it still requires selecting good landmarks, and could be hampered if these landmarks are not well chosen.

2) Sliding Window Classifier [9]: This method of segmentation is often used in text detection. It involves first training a classifier to detect whether there is a foot in the image or not by supplying lots of positive and negative examples. Having trained the classifier, a window (patch size pre-determined) is taken and run over the image with a stride hyperparameter (the amount the window shifts from patch to patch) adjusted to balance performance and computational cost. For each patch the window moves over, the classifier is run on the patch to detect the presence of a foot or not. Having done this, the window size is increased and the process of classification is repeated. This process is done for varying window sizes.
The advantages of such a segmentation method is that it preserves the spatial encoding of the foot, and it does not require manually extracting landmarks from the image for segmentation. However, this method can be computationally expensive, as several different sizes of windows have to be run to detect all sizes and orientations of feet in the image. Moreover, it would struggle to scale to images with a greater number of pixels, although this could be alleviated through image pre-processing and standardardising the size of the image. The more pressing disadvantage is the accuracy of segmentation - the sliding windows classifier only segments the foot in a rectangle, as opposed to truly segmenting the foot round its outline.

3) Convolutional Neural Networks (CNNs): These allow for automatic feature extraction from the image, which ensures that the features extracted from the image always capture enough data, and are useful for the segmentation and classification. They do this through a combination of convolution, ReLU and Pooling layers.
Each convolution layer consists of multiple filters passing over patches of the input image, with a stride hyperparameter, akin to the sliding windows classifier. Each filter has its own weights and the filter computes the dot product between the pixel values of the patch and the weights to output a real number for each patch, with an overall feature map outputted once it has passed over each of the patches of the image; the image is often padded with zeros to ensure the dimensionality of the feature map matrix is the same as the height and width of the image. The weights are learnt throughout training, so the filters end up activating (producing a different output) when it recognises a certain feature, (such as an edge) as shown in Figure 3. The feature maps are then stacked in the depth dimension, and this is then output to the ReLU layer. The ReLU layer consists of Rectified Linear Units which apply the activation function:

$$f(x) = max(0, x)$$

to the output of the convolution layer element-wise, to introduce non-linearities - this allows for the CNN to generate a more complex hypothesis.
The pooling layer downsizes the input dimensions to make the output more manageable. Like the convolution layer, filters are run over the input image, however, unlike the convolution layer, each feature map is treated separately. For each patch the filter outputs a real number: for max pooling, it outputs the maximum pixel value in the patch; for average pooling, it outputs the mean pixel value of the patch. Since this is a fixed function, there are no weights required, and unlike the convolution layer there is no zero-padding, since dimensionality is being reduced, not preserved.
The image is passed through a sequence of convolution, ReLU and pooling layers to extract a feature vector. The fully connected layers are the equivalent of a standard neural network's layers, and operate in a similar fashion, with the softmax function, defined as:

$$\sigma(x) = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}, \quad \text{for j = 1,2...K. and } x \in \mathbb{R}^K$$

applied to output probabilities of objects being present in the image, with the highest probability selected as the prediction.
The hierarchical nature of CNNs means that the features in subsequent convolution layers become more advanced, processing from edges to shapes to outlines of objects. This means that the the final set of features extracted by the CNN

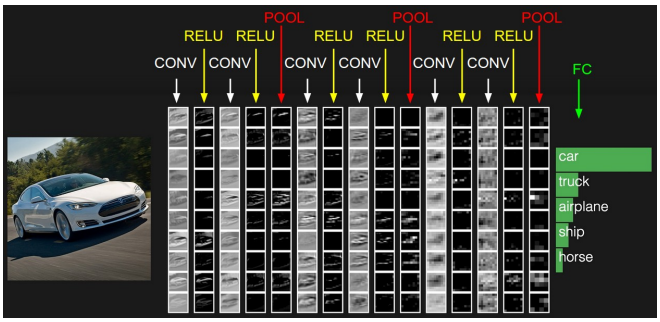

Fig. 3: *A visualisation of a CNN extracting the features from the image (shown by the visual representations of each node) and then classifying the image (probabilities of classes shown in green).*[1]

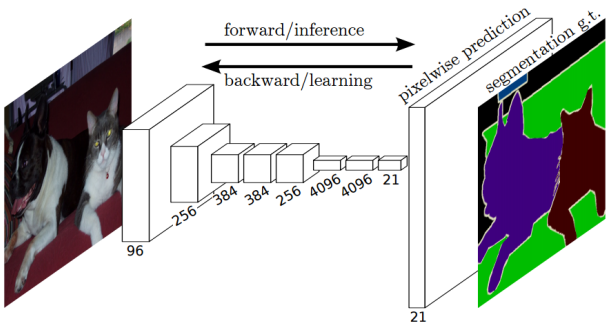

Fig. 4: *A visualisation of a FCN extracting the features from the image and then providing a pixel-wise classification to produce a heatmap of the image, with the objects segmented.).* [6]

can be very complex, and represent the structure and data contained by the image far better than any manual feature representation.

However, CNNs simply classify the image as either containing the foot or not, with no spatial encoding (where the foot is located in the image). This is useless for segmentation, as we already know that there is a foot in the image and are more concerned about the relative position of the foot, so that we can segment it.

4) Fully Convolutional Networks (FCNs): FCNs are a modification of CNNs and replace the neuron units in the fully-connected layers of the CNN with 1x1 convolutions. This preserves the spatial encoding of the output, and also means the FCNs can input variably-sized images, unlike the CNN which requires a fixed size input. The FCNs also include deconvolution layers, which upsample the output through a fractionally-strided convolution so that it has the same dimensionality (height and width) as the input image. This allows for a heat-map of the input image - providing a binary pixel-wise classification, so allowing us to segment the foot by selecting only the pixels inside the foot (which have value 1) and ignoring the pixels of the background (which have value 0). The pixel-wise classification thus allows for a much more accurate segmentation as it can capture the outline of the foot, unlike the sliding windows classifier. This can be seen in Figure 4, where the FCNs capture the outline of the cats very accurately.

Therefore, in this research we will focus on using Fully Convolutional Networks as a means of segmenting the image of the baby's foot.

B. Deep Learning Architecture

The deep learning framework being used to build the FCN model and subsequently train and test the model is the Caffe framework. FCNs are based off the VGG network architecture [7]. VGG networks differ from the traditional LeNet architecture by using small filter patches/kernels, replacing a single convolution layer where filters have 7x7 kernels with 3 convolution layers with filters using 3x3 kernels instead. This makes the network deeper, so it can produce more complex representations. It also contains fewer parameters whilst still

Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The ReLU activation function is not shown for brevity.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Fig. 5: *This figure shows the structure of the VGG Convolutional Neural Network architecture [7].*

providing the same receptive field, making the network more computationally efficient and decreasing the chances of overfitting (where the network fits the training data too well, and thus fails to generalise well to new input). VGG networks also differ from LeNet networks with regards to pooling: rather than pooling after every convolution layer, VGG nets pool after several convolution layers, as shown in Figure 5. In particular, the 16 weight layer VGG network architecture [7] is used to initialise the FCN. The FCN also uses Dropout [8] in some of the layers to regularise the output. This helps prevent FCNs from over-fitting the training data (caused by the sheer number of parameters as shown in Figure 5). As mentioned earlier, the FCN architecture also foregoes the fully connected layers, instead using 1x1 convolutions, and adds a deconvolution layer to upsample the output. Figure 6 shows the resultant FCN architecture being used [6], ignoring

the ReLU layers for brevity:

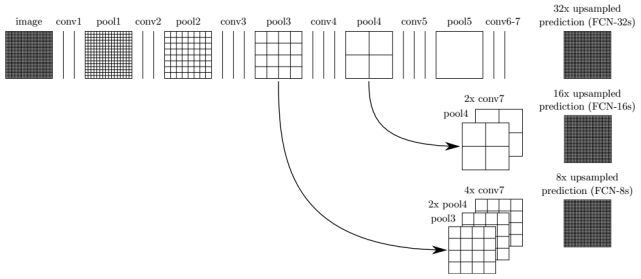

Fig. 6: *This figure shows the structure of the FCN architecture being used - the FCN-16s architecture in particular.* [6]

III. DATA PRE-PROCESSING

The dataset supplied to train and test the model consists of 1355 JPEG images of babies' feet, with the location of 43 landmarks on the feet also supplied for each image. An example of this, is shown in Figure 7.

First, the images were converted to PNG, a lossless format which better preserves the pixel-wise information necessary to train the network. The images' resolution varied from 885x1328 to 11,345x7563 and therefore had to be resized and standardised to under 400x400 pixels, to ensure a consistent input that is not too large, as this can be computationally expensive. The landmarks' pixel coordinates were resized and standardised by the same factor as its corresponding image, to ensure that they were still aligned with the images.

Fig. 7: *An example of the image supplied, with the landmarks then superimposed as yellow points.*

Next, to supply training data to the FCN, a ground-truth mask was generated for each image by plotting the outline of the foot using the landmarks, and using that as a decision boundary: all pixels outside the foot were set to 0, whilst the pixels contained in the foot were set to 1 (set to 255 during visualisation - so foot appeared white on black background for each mask).

Finally, the images were shuffled, along with their respective masks, and were then split into a training set with 60% of the samples and a test set with 40% of the samples. The images and masks were then converted to LMDB files, in preparation for input into the network. For the training set, the model had both the images and the ground truth masks available to learn from, whereas in the test set it only saw the images, with the ground truth masks later used to evaluate the performance of the FCN.

A second set of ground-truth masks was generated later, with these masks tracing the outline of the foot, but also including a gap between the largest toe and the second toe. This was done after evaluating the performance of the FCN on the first set of masks, with the aim of providing a more accurate outline of the foot for the model to learn from. This would result in further improvements on the model's performance on the segmentation task. An example comparing the two sets of masks is shown in Figure 8 and Figure 9. These were also converted to LMDB files, and were input, independent of the first set of masks, to the network for training. The FCN was reinitialised using the same initial weights as the first set of masks, to allow for a valid comparison between the performance of the network trained with each set of masks.

Fig. 8: *An example of the polygon decision boundary (in blue) used to generate the mask with (right) and without (left) without a gap between toes.*

Fig. 9: *An example of a mask with (right) and without (left) without a gap between toes.*

IV. METHODOLOGY

A. Training the FCN

The Fully Convolutional Network was trained using Stochastic Gradient Descent with momentum [5]. The weights of each layer were regularised using L2 regularisation (where an extra term $\lambda \parallel \Theta \parallel^2$ is added to the cost function $J(\Theta)$ to regularise the weights) with $\lambda = 5 * 10^{-4}$, alongside the aforementioned Dropout implementation. The Stochastic Gradient Descent (with momentum) update rule is as follows:

$$\Theta_{t+1} = \Theta_t - \Delta\Theta_t$$

$$\Delta\Theta_{t+1} = \alpha \frac{\partial}{\partial\Theta}(J(\Theta)) + \gamma\Delta\Theta_t$$

Gradient descent, shown in Figure 10, uses the fact that the gradient is zero at a minimum to iteratively adjust the weights Θ down the gradient $\frac{\partial}{\partial\Theta}(J(\Theta))$ in the direction of

the nearest minimum of the cost function. The cost function $J(\Theta)$ quantifies how close the output of the network is to the actual value, with greater error resulting in a greater cost. The gradients are calculated using backpropagation, where the error of the FCN's output is propagated from the output layer back through the network, layer by layer, to the second layer (the input layer has no error). Having calculated the error, denoted as $\delta_j^{(l)}$ for node j in layer l, the gradient with respect to the weights can then be calculated.

Stochastic gradient descent calculates the cost function for one example at a time, and optimises the weights with respect to that example, and perform a few passes through the shuffled training set. The advantage of stochastic gradient

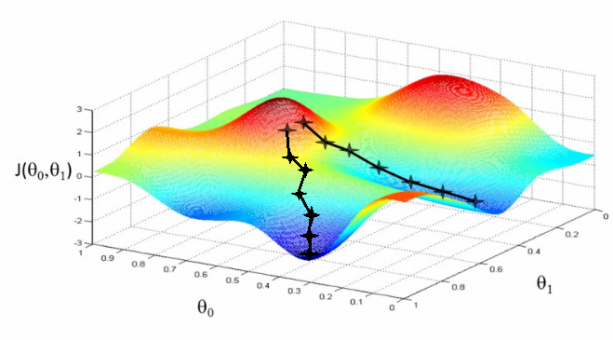

Fig. 10: *A visualisation of gradient descent.* [3]

descent is that computing the cost over the entire training set can be computationally expensive, and iterating using only one example at a time results in the FCN learning much more quickly. Momentum also accelerates the convergence as it is a velocity vector $\Delta\Theta$, that uses the directions of the previous gradients to continue converging in the general direction of previous updates, rather than oscillating along the cost function surface.

The hyperparameter α controls the learning rate, that is the size of the steps taken down the gradient whilst updating the weights. The hyperparameter γ controls the momentum, balancing the trade-off between the current update's descent direction with the previous updates' general direction.

To train the FCN, the hyperparameters were set as $\alpha = 1 * 10^{-4}$ and $\gamma = 0.9$. The mini-batch size was 1, and the cost was averaged over 20 iterations and output every 20 iterations, for debugging purposes, as the cost could be checked to see whether it was indeed decreasing or whether convergence had been achieved.

The training consisted of 4 batches of 10,000 iterations, with a snapshot of the weights saved after each batch of iterations. To optimise training of the FCN, three layers (score59, score-pool4 and score-pool3) were activated in turn, with the other two deactivated (learning rate set to zero) for the each of first three batches, and for the final batch of training all of the three layers were activated.

The weights were initialised for the first batch of training, and each snapshot of the weights was then input for the next batch of training, to be improved upon. The snapshots of weights were then used in testing, as they could be used to generate multiple outputs, which could be compared to visualise the improvement of the FCN throughout training.

The FCN was trained initially with the first set of masks, then the weights were re-initialised and the FCN was trained with the second set of masks, with the same training and test set. This was done to ensure as valid a comparison between the performance of the two sets of masks.

B. Post-processing

The Fully Convolutional Network had a tendency to over-fit in the latter stages of training, with regions of false positives, often caused by hands in the image or other skin present. To further improve the performance of the network, a post-processing algorithm was applied to the outputs, leaving only the largest connected region of white pixels (the baby's foot) as white, and the remaining white pixels set to zero (black), thus eradicating any false positives. Figure 11 shows an example of this process. The performance of the FCN after post-processing can also then be compared and contrasted with the performance when post-processing has not been applied to the outputs.

Fig. 11: *An example of an output before (left) and after (right) post-processing - note the lack of false positives.*

V. RESULTS AND EVALUATION

A. Evaluation Metric

The evaluation metric used was the Jaccard Index. This statistic is used to compare the similarity of two sets of data. It is defined, for sets A and B, as:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}, \quad 0 \leq J(A,B) \leq 1$$

More concretely, the Jaccard index could be seen as:

$$\frac{|TruePositives|}{|FalsePositives + FalseNegatives|}$$

For each of the FCN outputs, we defined the set A as the white pixels in the mask, and set B as the white pixels in the predictions. The Jaccard Index can then be calculated for each of the outputs.
Another common evaluation metric is the Jaccard distance, which is complementary to the Jaccard Index and defined as:

$$d_J(A, B) = 1 - J(A, B), \quad 0 \leq d_J(A, B) \leq 1$$

Since some researchers prefer the Jaccard Distance as a measure of similarity, the Jaccard Distance was also calculated for each of the outputs of the network.

B. Table of Results

For each set of results, the mean, median, minimum and maximum Jaccard Index and Jaccard Distance were computed, to give an overall evaluation of the performance of the Fully Convolutional Network, with comparisons between the sets of results highlighted. Table I and Table II summarise these metrics before and after post-processing, respectively.

TABLE I: Output with First set of masks (before post-processing)

		Output of FCN after # iterations:			
		10,000	*20,000*	*30,000*	*40,000*
Jaccard Index	Mean:	0.6329	0.7035	0.7313	0.7873
	Median:	0.7393	0.7937	0.807	0.854
	Max:	0.9447	0.9519	0.9507	0.9588
	Min:	0	0	0	0
Jaccard Distance	Mean:	0.3671	0.2965	0.2687	0.2127
	Median:	0.2607	0.2063	0.193	0.146
	Min:	0.0553	0.0481	0.0493	0.0412
	Max:	1	1	1	1

By comparing the mean and median Jaccard Index for the outputs after 10,000, 20,000, 30,000 and 40,000 iterations, (Table I) the FCN's performance can be seen to improve significantly with training, from 0.6329 to a final result of 0.7873, which shows how accurate FCNs are. However, as the minimum Jaccard Index of 0 shows, the FCN cannot segment some test set images, highlighting potential for further improvement.

Comparing Tables I and II, the numbers in bold highlight a significant improvement in performance for the outputs after 30,000 and 40,000 iterations after post-processing. This confirms that the removal of false positives (which were not present in the first 20,000 iterations of training) does indeed improve the performance of the FCN.

Adding a gap between the largest toe and the second toe to the mask improved the performance of the FCN when comparing the outputs earlier in training (10,000, 20,000 and 30,000 iterations) as both the median and mean Jaccard Index

TABLE II: Output with First set of masks (After Post-processing)

		Output of FCN after # iterations:			
		10,000	*20,000*	*30,000*	*40,000*
Jaccard Index	Mean:	0.6329	0.7034	**0.7393**	**0.7892**
	Median:	0.7393	0.7937	**0.8163**	**0.8561**
	Max:	0.9447	0.9519	0.9507	0.9588
	Min:	0	0	0	0
Jaccard Distance	Mean:	0.3671	0.2966	0.2607	0.2108
	Median:	0.2607	0.2063	0.1837	0.1439
	Min:	0.0553	0.0481	0.0493	0.0412
	Max:	1	1	1	1

TABLE III: Output with Second set of masks (After Post-processing)

		Output of FCN after # iterations			
		10,000	*20,000*	*30,000*	*40,000*
Jaccard Index	Mean:	0.6788	0.7431	0.7672	0.8141
	Median:	0.7508	0.8269	0.8274	0.8524
	Max:	0.9135	0.9467	0.9425	0.9491
	Min:	0	0	0	0
Jaccard Distance	Mean:	0.3212	0.2569	0.2328	0.1859
	Median:	0.2492	0.1731	0.1726	0.1476
	Min:	0.0865	0.0533	0.0575	0.0509
	Max:	1	1	1	1

are higher for the second set of masks (shown in Table III compared to the first set of masks (Table II).

However, comparing the outputs after 40,000 iterations, the mean Jaccard Index is higher for the second set (0.8141 vs 0.7892) but the median Jaccard Index is lower than the first set (0.8524 vs 0.8561). Furthermore, the maximum Jaccard Index for each of the outputs of the second set is slightly lower than their counterparts for the first set of masks, suggesting the addition of a gap does not significantly improve the performance of the FCN for the easily segmented examples.
The minimum Jaccard Index is the same for all outputs of the FCN across both sets of masks (0) suggesting the addition of the gap does not improve performance for the examples the FCN finds difficult to segment. It seems in these cases it would perhaps be better to provide more training examples for the FCN to learn from, since the network struggled with only a handful of images.

C. Test Set Visualisations

In order to better evaluate the Fully Convolutional Network's performance, and perform error analysis on the test set, the test set was visualised. The visualisation consisted of: the original image, the ground truth mask, and the outputs of the Fully Convolutional Network when the weights learnt at 10,000 iteration intervals were supplied to the model for testing on the test set. The outputs give a sense of how the

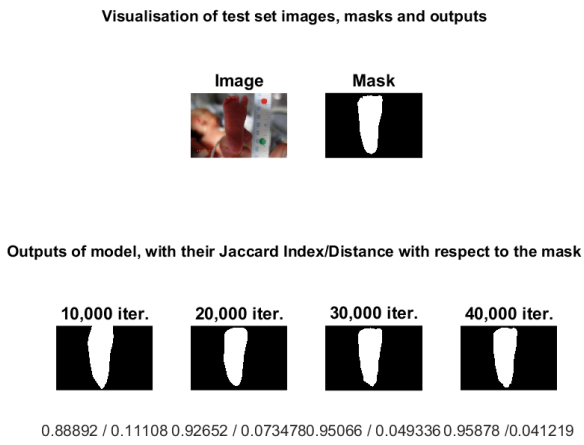

Fig. 12: *An example of a good output of the FCN when trained on the first set of masks.*

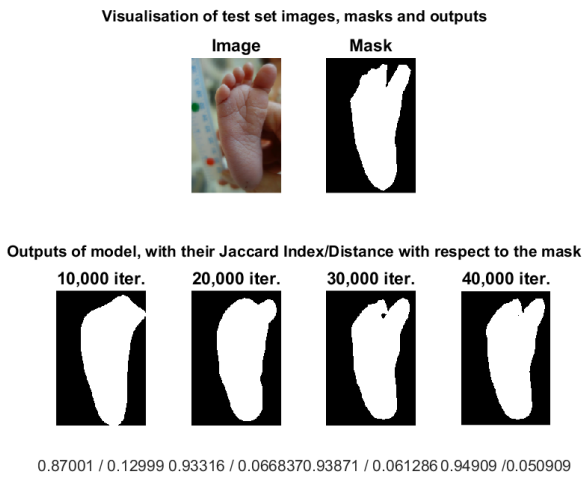

Fig. 13: *An example of a good output of the FCN when trained on the second set of masks.*

FCN is improving as training progresses, with the Jaccard Index and Jaccard Distance values are provided as reference. These visualisations, shown in Figure 12, Figure 13, Figure 14 and Figure 15 provide meaning to the evaluation metric.

In Figure 12, the image is easy for the model to segment as it takes up a large proportion of the image and is a straightforward shape for the FCN, evidenced in the output after 10,000 iterations being a good fit already when compared to the ground truth mask.

For Figure 13, the outputs show the model initially underfitting, but then as training continues, it learns to identify the gap between the large toe and second toe. The FCN has segmented the image very well, helped by the fact that the foot only has one gap between its toes, making segmentation easier.

Figure 14 shows that the FCN has under-fit the data and not included the gap in the foot between the second and third toes. This highlights the restricted nature of the ground truth mask in that it does not do a good job in fitting the data, as this image didn't have a gap between the large toe and second toe. The failings of the ground truth mask reflect in the performance, and thus a focus for future work is to improve the ground truth mask.

Figure 15 highlights the inadequacy of the ground truth mask, since there are multiple gaps between the toes of the foot, the major point it highlights is the FCN's ability to learn features that humans could not have expected it to learn. Here, the output of the FCN initially under-fits, but then learns to identify gaps between the toes. The segmentation of the image thus is excellent, matched only by manually

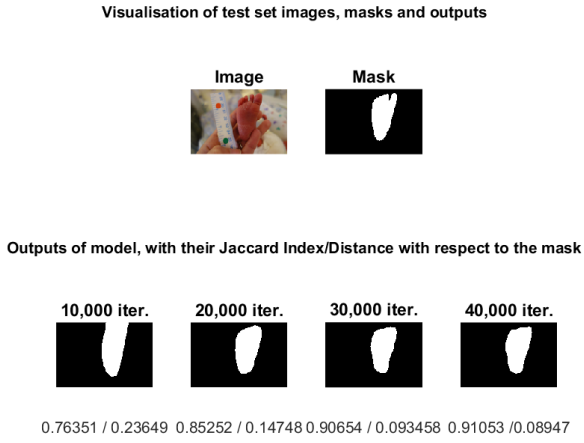

Fig. 14: *An example of a poor output of the FCN when trained on the second set of masks.*

drawing the outline of the foot as a decision boundary.

Figure 16 highlights the progress the FCN has made on an obscure test set example. As the ground truth mask highlights, the foot takes up only a small fraction of the total image, and is thus difficult to localise, evidenced by the outputs after 10,000 and 20,000 iterations of training. However, the FCN can be seen to learn to identify small images, and, given more examples of such images, could, after more training, segment even these tough test examples.

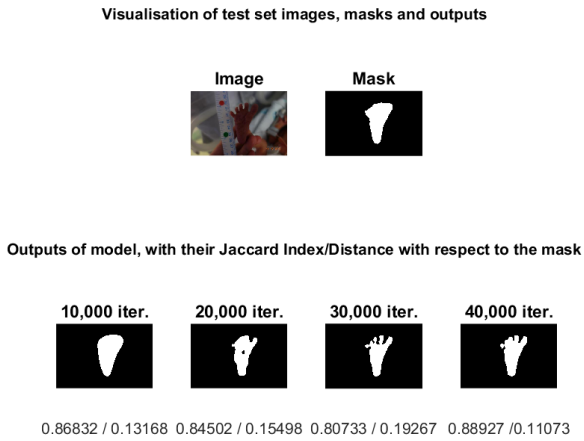

Fig. 15: *An example of a great output of the FCN when trained on the first set of masks. The FCN has actually outperformed the ground truth mask - an example of the FCN learning unexpected features of the image.*

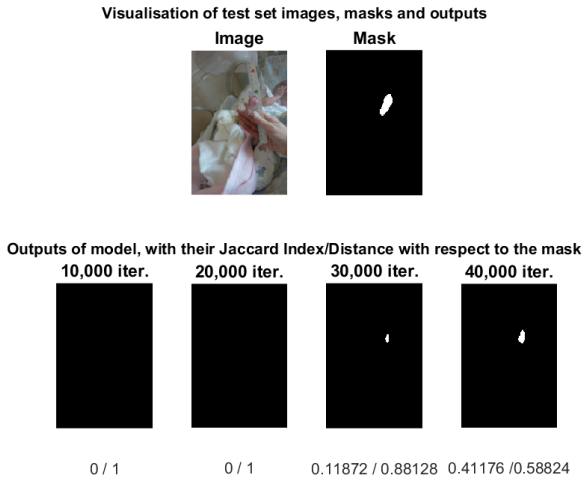

Fig. 16: *An example of the FCN learning and improving during training on the first set of masks.*

VI. CONCLUSION

Convolutional Neural Networks are an area of deep learning that has shown real potential and state-of-the-art performance thus far on computer vision tasks, especially for classification. Fully Convolutional Networks extend this performance to the computer vision challenges of localisation and segmentation. The Fully Convolutional Network performed very well, with its segmentations being highly accurate, with a median Jaccard Index of 0.8561 after training on the first set of masks highlighting this.

The FCN highlights the immense potential of deep learning, as Figure 15 shows, as the model's representations of the foot demonstrated true self-taught learning, another way that the FCN outperforms the other methods of segmentation.

The test-set visualisations highlighted the inadequacy of the current ground truth masks, and it is highly probable that with much more accurate ground truth masks, the network will be able to learn a more complex representation of the foot, so could perform the task of segmentation to human-like levels for most masks.

It is also possible that a larger training set, with thousands or tens of thousands of images would cause the model to improve its performance. A higher quality of training data would also be advantageous, since Figure 16 and later in the report, Figure 19, where the images are blurry or only a small fraction consists of the foot, highlight lower-quality data that could hinder the FCN's training.

On the other hand, having a mixture of high and low quality data could make the model's representations and segmentations more robust, and could extend its outreach to poorer areas as the cameras would not have to be as high-resolution, and so cheaper camera technology could be used. The relative frequencies of the data could be manipulated so the examples the network struggles to segment appear slightly more often when training, as it is with these examples with large error that the model will see the greatest amount of learning as documented by LeCun et al [5].

VII. FUTURE WORK

A. Improving Ground Truth Masks

By improving the ground truth mask, the performance of the Fully Convolutional Network will increase, since the quality of the training data the network learns from is higher, and more detailed, allowing the FCN to better segment the foot in the image. The Jaccard Index will also increase, since the ground truth mask better fits the foot, so more of it will intersect with the output of the model. A couple of potential methods of improving the mask are given:

1) Level Sets: One method of tracing the outline of the baby's foot more accurately is using level sets. The level set function is initialised and reshaped iteratively so, when considering the 3rd dimension, the pixels inside the baby's foot's are set to 2, and the pixels outside the foot are set to -2, and the boundary (outline of the foot) is the zero-crossing of the function. To aid this convergence, an edge detector function is run such that the landmarks are set to 0, with the surrounding pixels set to 0.0001 and the pixels surrounding those are set to 0.0005. This ensures the function shrinks slower around the landmarks, to ensure it converges in the

shape of a foot.

The outputs of the Level Set Function are currently erroneous for the most part, with a correct outline being the exception rather than the rule. Some sample outputs are shown below in Figures 17, 18 and 19. To improve the output, there are two possible courses of action: tweaking the hyperparameters of the level set function for better convergence or instead improving the edge detection function so it highlights the edges of the foot, but not any surrounding objects like rulers or hands. The latter is more likely to result in better mask decision boundaries being output.

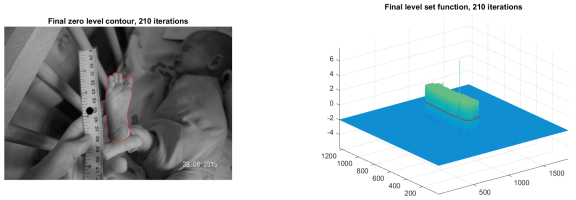

Fig. 17: *An example of a decent output using the level sets function.*

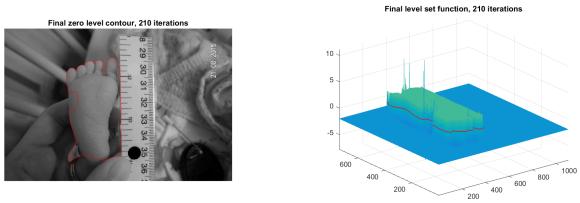

Fig. 18: *An example of an erroneous output using the level sets function - note the edge of the ruler detected.*

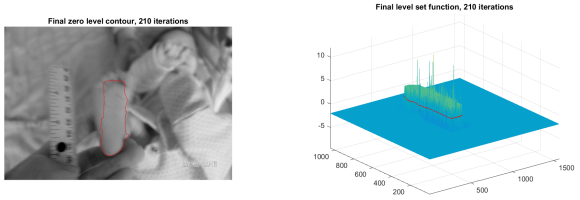

Fig. 19: *An example of a poor output using the level sets function - note the edges of the foot not detected. This is due to the blurriness of the image - highlighting low-quality input data.*

2) K-means Clustering: Another method of segmenting the baby's foot in the model is to first isolate the outline of the baby's foot by setting the background to black then converting the colour map of the image from RGB to HSV. Given that the baby's foot (which has a pinkish-red hue) contrasts with the background, the K-means clustering (an unsupervised machine learning algorithm) was run with 3 clusters - the foot, the background and the gaps between the toes. This should cluster the foot pixels, which can then be set as white, while the background and gaps clusters could be set to black. This would result in masks that takes into account of any gaps between toes for each image of the baby's foot.

Figure 20 shows the variability of the outputs of the clustering. The issue with the K-means clustering approach is the HSV colourmap of the image - the pixels of the foot do not consistently have similar hues on the photo, one of the possible causes being the lighting and shadows in the image. Therefore, the algorithm is less likely to perform as well, since the clusters are not as well-defined, and as such the outputs of the clustering are more often erroneous and actually worse than the standard mask used in the research. Adapting the colour map, or perhaps post-processing the image such that black pixels surrounded by white pixels were turned white could improve the mask output.

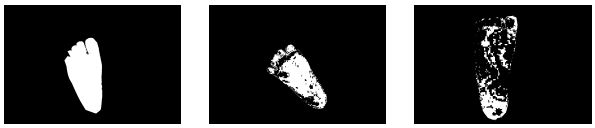

Fig. 20: *Examples of a good clustering output (left), a typical clustering output (middle) and a bad clustering output (right).*

VIII. ACKNOWLEDGEMENTS

The author gratefully acknowledges the contributions of Michel Valstar and Mercedes Torres Torres for their guidance and helpful advice, especially regarding methods of further improving the performance of the Fully Convolutional Network.

IX. REFERENCES AND BIBLIOGRAPHY

REFERENCES

[1] Cs231n: Convolutional neural networks for visual recognition. `http://cs231n.stanford.edu/`. Accessed: 2016-08-19.

[2] Maturational assessment of gestational age (new ballard score). `http://www.respiratoryupdate.com/members/Maturational_Assessment_of_Gestational_Age_New_Ballard_Score.cfm`. Accessed: 2016-08-19.

[3] Tuning the learning rate in gradient descent. `http://blog.datumbox.com/tuning-the-learning-rate-in-gradient-descent/`. Accessed: 2016-08-19.

[4] J. L. Ballard, K. K. Novak, and M. Driver. A simplified score for assessment of fetal maturation of newly born infants. *The Journal of pediatrics*, 95(5):769–774, 1979.

[5] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[6] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[7] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[8] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[9] C. Wojek, G. Dorkó, A. Schulz, and B. Schiele. Sliding-windows for rapid object class localization: A parallel technique. In *Joint Pattern Recognition Symposium*, pages 71–81. Springer, 2008.