Clinical Scene Segmentation with Tiny Datasets

Thomas J. Smith[1] Don Sharkey[2] John Crowe[3] Michel Valstar[1]
University of Nottingham
Computer Vision Lab[1], School of Medicine[2], Faculty of Engineering[3]
{Thomas.Smith3, Don.Sharkey, John.Crowe, Michel.Valstar}@nottingham.ac.uk

Abstract

Many clinical procedures could benefit from automatic scene segmentation and subsequent action recognition. Using Convolutional Neural Networks to semantically segment meaningful parts of an image or video is still an unsolved problem. This becomes even more apparent when only a small dataset is available. Whilst using RGB as the input is sufficient for a large labelled dataset, achieving high accuracy on a small dataset directly from RGB is difficult. This is because the ratio of free image dimensions to the number of training images is very high, resulting in unavoidable underfitting. We show that the addition of superpixels to represent an image in our network improves the semantic segmentation, and that superpixels can be learned to be detected by Convolutional Neural Networks if those superpixels are appropriately represented. Here we present a novel representation for superpixels, multi-channel connected graphs (MCGs). We show how using pre-trained deep learned superpixels used in an end-to-end manner achieve good semantic segmentation results without the need for large quantities of labelled data, by training with only 20 instances for 23 classes.

1. Introduction

Clinical procedures on newborns can be some of the most important events in a human's life as they can make the difference between a person surviving or not. Resuscitation is one of the most common procedures performed on newborns directly after delivery. Because of the fragility of babies, it is a more complex procedure than for adults, and the paediatric staff have to follow a regimented protocol. The protocol consists of a sequence of actions, usually carried out by more than one person and involving multiple items of equipment. Add to this the increasing pressure upon staff, and it becomes clear that there is ample opportunity for mistakes to be made. Given the current state-of-the-art in action recognition, it should be possible to perform automatic scene segmentation and action recognition to monitor this procedure. This would be of high value as it would allow one to establish best practice, could be used as a tool in training healthcare professionals, and could function as an early warning system when clinicians deviate from standard protocol. A similar system has been developed for detection of CPR performed on children. In this study, they use a dataset populated with simulated CPR videos [14]. However they only detect between performing CPR and not performing CPR.

To use state-of-the-art deep learning techniques it is widely accepted that an abundance of training data is required, with datasets such as MS Coco including 330k images of 80 semantic segmentation classes [11]. Collecting datasets of even a moderate size is difficult when it concerns newborn babies, as it requires a lengthy ethics approval and one always has to balance the future benefit of a new technology with the added risk of changing the operational environment of clinicians, even if that change involves only small changes in procedures and hardware. And even if we could collect large amounts of imagery, it would then need to undergo lengthy manual annotation of semantic segments for the networks to be able to learn from it.

Semantic Segmentation is the process of segmenting a scene into meaningful segments [7]. Each segment must belong to a single class that exists in a dictionary of possible objects. Multiple smaller elements may need to be joined together to form a single complex segment. An example of this is a car, as a car is made up of several parts such as the body, windows and wheels. However, in regards to the pixels in an image, these elements can be split even further into a collection of similar pixels. This is a type of over-segmentation that is commonly referred to as *superpixels*. Superpixels are created using man-made algorithms to cluster similar pixels, generally using a combination of colour and location similarity as well as size and shape constraints. If it would be possible to learn the relationship between pixels and superpixels from a large dataset of images using Convolutional Neural Networks (CNNs), we could then construct an end-to-end architecture where the learned superpixels feed into the segmentation network.

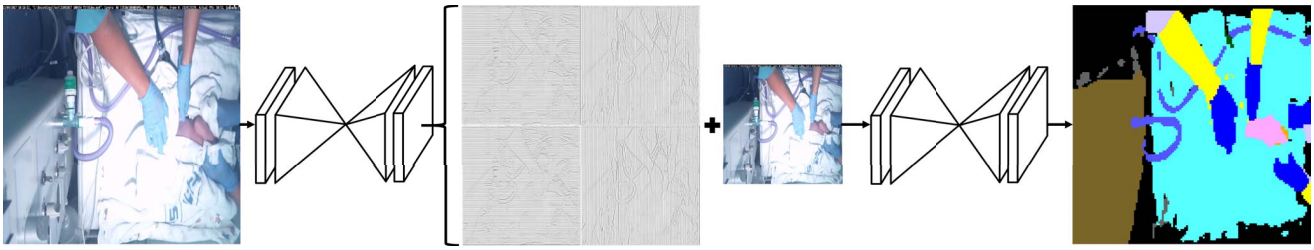

Figure 1. Our end-to-end network architecture takes an RGB image of shape $512 \times 512 \times 3$, and predicts the MCG representation of the deep learned superpixels in the first hourglass. Then these predicted MCGs plus the original RGB image are fed into a second hourglass where the network learns the semantic segmentation masks for each class and predicts them as grayscale masks of shape 512×512. These masks can then be combined back together to show all classes at ones as an RGB representation where each class has its own colour.

Whilst the network that learns superpixels could use a very large, unsupervised dataset for training, the greatly reduced input complexity of superpixels would allow the segmentation part of the network to learn useful models with a much smaller dataset.

Learning superpixels from RGB data is surprisingly hard. In this paper we show why a straightforward approach of treating each superpixel as a class doesn't work, and propose a novel representation for learning superpixels that is based on Markov Blankets [5]. We then propose a novel network that learns to do end-to-end superpixel and semantic segmentation. We show that by using learned superpixels when analysing a single niche task, as little as 20 training instances is sufficient to perform semantic segmentation of clinical procedures, where we can successfully detect 10 classes to the point where they could be successfully applied to action recognition and scene analysis.

In summary, the contributions of this paper are:

- We propose for the first time to learn superpixel oversegmentation
- We propose to use learned superpixels to do semantic segmentation, and show that this can be done using end-to-end trained Convolutional Neural Networks
- We demonstrate that good domain-specific results can be obtained with as little as 20 training images for semantic segmentation

2. Related Work

Below we discuss relevant related work in superpixel oversegmentation and deep-learned semantic segmentation.

2.1. Superpixels for segmentation

In 2003 Ren and Malik [16] proposed learning a classification model for segmentation. This is an early example of using superpixels to perform scene segmentation. In their work they merge the superpixels by using Gestalt grouping cues for the features of a linear classifier. The method worked well, however some parts of the scene would still be split into multiple super-regions rather than one segment per object. Thus complete segmentation was not achieved.

More recently superpixels have been used in video segmentation. Examples of this are; figure/ground video segmentation via low-risk sparse learning by Gu et al. [6], video segmentation using spectral clustering on superpixels by Bhatti et al. [4], and video object segmentation using multiple random walkers with a GMM restart rule by Heo et al. [8].

The work of Gu et al. [6] used superpixels to perform foreground and background segmentation. They use an adaptive dictionary which remembers n previous frames, where each frame has the superpixels that correspond to the foreground and background stored separately using the L2ECM [10]. The representation is learnt using the Inexact Augmented Lagrange Multiplier (IALM) method [18].

Bhatti et al. [4] use superpixels as part of a novel solution combining colour, optical flow and saliency maps. They make use of several superpixel maps, where each map has a different resolution of superpixels. These are then merged to form a more detailed map. Where there is more consistency in the image the superpixels are larger, and they are smaller around more detailed objects. They then combine the saliency, merged superpixels and optical flow maps together to create a foreground separation map. Finally, they use this new map in conjunction to the saliency and optical flow maps to create a final segmentation mask. At the time this achieved the best average score for segmentation on the SegTrack v2 dataset [9].

2.2. Deep learned semantic segmentation

One of the most popular learnt segmentation methods is the Fully Convolutional Network (FCN) developed by Long et al. [12] in 2015. In their work, an end-to-end network is used to produce pixel-by-pixel predictions. This produces both segmentations and classification. This is achieved by use of their skip architecture which makes use of both the deep coarse semantic information and the shallower appearance information.

Another variation on the learnt segmentation is the

Learned Watershed (LW) method by Wolf et al. [17]. This method expands on work such as that by Bai and Urtasun [2] by going a step further and building an end-to-end CNN that produces segmentations instead of the original energy maps used. LW keeps the original watershed algorithm unchanged by starting with seeds and iteratively adding the best pixels for each seed from their queues, whilst prioritising closer pixels.

In summary, superpixels have previously been used for semantic segmentation with some success, but always by using hand-crafted superpixel methods and combining superpixels to form super-regions/segments. The most successful semantic segmentation approaches have been based on deep learning that make predictions directly from RGB. However, such techniques would require very large datasets. Deep leaned superpixels have not been proposed before, let alone used for semantic segmentation. Making use of deep learned superpixels instead of hand-crafted techniques would allow end-to-end learning. In this work we show how this can be achieved.

3. Multi-channel Connected Graphs

Learning to detect a superpixel is non-trivial. A straightforward approach would be to set each superpixel to be its own class. However, when doing so, training the networks we experimented with did not converge to anything meaningful. A different representation is thus required. We propose to use a representation that shows for each pixel's neighbours whether they are included in the same superpixel or not. We call this Multi-channel Connected Graphs (MCGs).

The MCGs work similarly to a Markov blanket by representing the relationship between neighbouring pixels. To obtain two-dimensional images that can be used as the target in e.g. an hourglass network, we use four channels to represent the four directions where an immediate neighbour can be: up, right, down and left.

Using superpixels as an example use case, each pixel in the algorithm's output mask carries the information of whether it belongs to the same superpixels as any of its four neighbours. This means every pixel in our two-dimensional data space carries connectivity information for the four relative directions. To represent this in a way where we can use standard CNN methods, we transform the problem to multi-channel connectivity maps, one indicating connectivity in each direction (up, right, down, left).

To transform from a superpixel mask to a MCG representation is simple. First, there must be four channels of the same width and height as the superpixel mask. These channels must be initialised to zero. Next, the superpixel mask must be iterated over pixel by pixel, whilst checking if the current pixel is in the same superpixel as its neighbours. For each of the pixel's four neighbours, the respective masks are updated by setting the cell at the current pixel's coordinates.

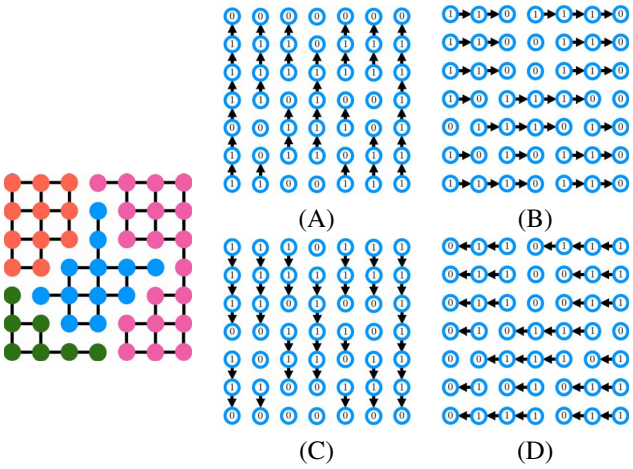

Figure 2. The left multi-coloured image represents a small example of superpixels where each superpixel is assigned a different colour. The lines between pixels represent that the pixels are in the same superpixel as each other. To the right are four MCGs representing the superpixels in the left image. Here we are using four directions: up (A), right (B), down (C), and left (D). Each of the graphs consist of ones and zeros, where one indicates that the given pixel is in the same superpixel as the pixel in the direction of the graph, and zero represents otherwise. To help illustrate this further we have included arrows where the graph is activated and the direction of the activation. For example, if we take the origin to be the upper left corner, then the pixel $(3, 3)$ belongs to the orange superpixel. This pixel is in the same pixel as it's upper and left neighbours, thus graph A and D at $(3, 3)$ are set to one. As there are no connections to the right or down of this pixel, graphs B and C are set to zero at $(3, 3)$.

This will be set to a 1 if the two neighbours are in the same superpixel, or otherwise this is left at 0. For the edge cases where it would be indexing out of the superpixel mask, there are two options. Edge cells would be set to 0 if there is no evidence that they are connected to the same theoretical superpixel. They would be set to 1 if you assume that the superpixels expand past the frame. In our work we use the first assumption, but when working with video it might be better to use the second one.

In figure 2 we show a small example of how to represent a superpixel mask in the MCG format, but instead of using numbers as the unique identifiers for each superpixel, different colours and connecting lines are used.

The MCG methodology could easily be expanded with a larger neighbourhood, such as eight neighbours, to include the diagonal neighbours or even 24 to cover neighbours of neighbours. However, expanding the number of channels could cause issues with memory space.

4. Learning Superpixels with MCG

To create a deep learned superpixel predictor that can be used in a wide range of applications, we need to have a suf-

ficiently large dataset to train our network with. As we are using a large CNN network we are using an unsupervised approach to training as it would be impractical to do anything else. We chose the DAVIS [15] dataset as it consists of 120 HD videos and has a large variation of scenes, ranging from sheep in a field to cars driving around a race track.

As we use relatively large images for training at 512×512 we have cut down the number of images used from the dataset to three per video: the first, middle and last frames. This gives us the largest variation per video whilst keeping the final dataset at a reasonable size for training.

For each of the three frames we run the hand crafted superpixel algorithm, Simple Linear Iterative Clustering (SLIC) [1], multiple times with varying numbers of target superpixels, ranging from 500 to 5000 superpixels in steps of 500. Varying the number of target superpixels changes the size constrain of the superpixels. The other SLIC parameters were set to $\sigma = 1$, $compactness = 30$. This means there was approximately 3000 images for training and 1000 for validating the network.

The images were first cropped to 1080×1080 pixels by cropping 420 pixels from the left and right sides of the frame. Next, we resized them to 512×512 pixels because this is a power of two. This works more appropriately for the network, so that when down-sampling via max-pooling with a 2×2 kernel there is always the ability to halve the resolution again.

The network architecture we adopt is the highly successful hourglass network introduced by Newell et al. [13]. To adapt the network we have changed the input of the network to allow input of $512 \times 512 \times 3$. This allows us to input an RGB image at a high resolution to reduce information loss when down-scaling the images and superpixel masks. We have also added two up-scaling layers before the output of the network to bring the resolution back to $512 \times 512 \times 4$.

When using the MCG representation, the model did not predict the superpixels ground truths but instead abstracted to super-regions. This is a consequence of learning the superpixels at varying size constraints. It does this whilst keeping good boundary separation.

5. Clinical Use Case Dataset

The Newborn Resus dataset consists of 70 videos of newborn babies receiving care to ensure that they are in good health. Some of the babies in the dataset require resuscitation or stabilisation performed by the nurses and doctors. This dataset consists of two different scenarios, firstly full term babies born by caesarean section (C-section) after a normal pregnancy, and babies born by 'any route' who may need stabilisation or resuscitation.

The videos taken for the 'any route' scenario were collected under different conditions to the C-section scenario. The main differences were the bed the babies are placed on, the colour of the resuscitaires tubes, and the angle at which the video was shot. The C-section videos were shot top down whereas the 'any route' videos were shot at about 45° from the bottom of the bed.

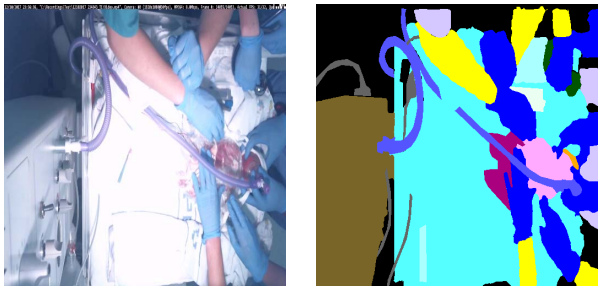

Figure 3. An example frame from the C-section scenarios with ground truth data generated using the custom image annotation tool. Key for ground truth colours can be found in table 4.

To create the tiny dataset to train and evaluate our model, we take 50 frames from the 35 'good' videos. To achieve a good distribution throughout the dataset, the number of frames from each video was inversely proportional to the number of instances from the scenario, resulting in a 30:20 split of C-section to 'any route' frames.

A custom annotation tool was used to pixel-wise semantically label each frame. The image annotation tool makes use of SLIC superpixels to speed up annotation, allowing the user to select the superpixel corresponding to each class. The SLIC parameters can be adjusted to suit the current data one is working with. If the images one is working with aren't properly segmented by the superpixels, it also allows the use of polygons for annotation. See figure 3 for example frame and ground truth data.

6. End-to-End Semantic segmentation with Deep Learned Superpixels

Our end-to-end network architecture uses two hourglass networks. The first hourglass is used to predict the deep learned superpixels, by taking in the RGB frame and outputting the MCGs. This first hourglass is pre-trained and then left unfrozen to allow the superpixels to update with the different niche tasks. The MCG output plus the original RGB is then fed into the second hourglass to predict x grayscale masks, where x is the number of classes. Initially to prove that the concept would work we started with five classes: Gloves, Bed, Baby, Pipes, Stethoscope, and an unknown class. As a baseline, we used mean squared error loss for the semantic segmentation sub-network and trained the network in a traditional multi-class classification (all classes at once).

To determine the best format for training with such a small dataset, we changed several variables; loss function, transfer learning vs. all at once, and adaptive vs. static un-

known class for transfer learning.

The loss functions that we compared are MSE, and Lovász-Softmax loss [3]. Lovász-Softmax loss, also known as the Jaccard loss, is an approximation of the intersection over union metric used to determine the accuracy of the network. Berman et al. showed that their loss function was able to improve detection of small objects and this property of Jaccard loss function is applicable to the clinical dataset as there are many small pieces of equipment.

Transfer learning is common practice in deep learning, however unlike common practice we aren't bootstrapping the semantic segmentation from a well known classification dataset such as MS Coco. To pre-train our custom network, it would take a long time and potentially not help the learning of these niche datasets, because the objects being detected can be vastly different to the objects in the large datasets. We use transfer learning to train one class at a time and iteratively append new classes to the previously trained models until all classes are learnt.

As part of the transfer learning we tested the use of an adaptive unknown class. The static unknown class is predefined before training and is learnt as a class, whereas the adaptive unknown is set to all classes except the current and previously learned classes. Thus on the next iteration of the network, we can initialise the weights of the new output layer to be the same as the unknown class because the new class exists in the previous unknown class. This means that the network only has to learn what isn't in the unknown and what is the new class.

To train the network we partitioned the 50 frames into three sets with 20 training, 20 test, and 10 validation instances. There are no frames from the same video spanning multiple sets. We then trained the network with 14 pipelines, table 1, to compare the previously mentioned network parameters. When looking at the results in table 1 the best performing pipeline is M_1 using transfer learning, descending order, and MSE. However, when manually inspecting the prediction of the network it was noted that frames that differ greatly from the majority performed poorly, and these brought down the average results.

We then refined the dataset to remove the frames that came from the babies born by 'any route' as these videos consisted of completely different lighting and work space setup. To replace these images we added additional frames from the 'good' videos to create a more consistent dataset. The test set can be seen in figure 4. As it can be seen in this figure, the majority of the input frames have the same exposure and use the same work space setup. On row nine of figure 4 the first input image isn't as brightly lit due to the staff turning off the light and the performance of the network is drastically reduced.

The pipelines were then rerun with the newly refined dataset and the results can be seen in table 2. When comparing the performance of the two datasets it can be clearly seen that the second refined dataset has a higher average performance across all pipelines. However, now the gains difference between the pipelines has decreased. To help evaluate this, we combine the F1 score with the percentage of incorrect pixels by combining them with equal weighting, one-to-one. The results of this can be seen in table 3. When looking at the top four performing pipelines it is now clear that the ordering of the classes does not have a large impact on the performance of the network. M_2 (descending order) performs the best but is within the margin of error with K_2 (ascending order), and F_2 (random order). The adaptive unknown class and the use of transfer learning results show in both cases to be better than static unknown class, and multi-class learning respectively. Unfortunately, the Jaccard loss isn't consistently better than MSE, and often the performance between the two loss functions is similar. In the ordering pipelines the Jaccard loss actually performs worse than MSE in both F1 and percentage of incorrect pixels.

The network using the pipeline M_2 has performed remarkably well with a relatively high F1 score and a low percentage of pixels to update, considering that it is only trained with 20 instances. Next we trained a model using the parameters from the M_2 pipeline but have increased the number of classes from six to 23. The performance for each class and the average performance can be seen in table 4. At first glance, the average performance looks poor with the lower F1 score of 0.437 and average percentage of in-

ID	Pipeline	Loss Function	F1 Score	% Incorrect Pixels
A_1	MCL RO PU	MSE	0.556	28.80%
B_1	MCL RO PU	Jaccard	**0.578**	**24.86%**
C_1	TL RO PU	MSE	0.46	**17.68%**
D_1	TL RO PU	Jaccard	0.537	18.86%
E_1	TL RO AU	MSE	0.551	17.97%
F_1	TL RO AU	Jaccard	**0.556**	19.65%
G_1	TL SF PU	MSE	0.558	**18.11%**
H_1	TL SF PU	Jaccard	**0.581**	21.34%
I_1	TL SF AU	MSE	0.578	19.67%
J_1	TL SF AU	Jaccard	0.569	21.34%
K_1	TL AO AU	MSE	0.587	27.61%
L_1	TL AO AU	Jaccard	0.565	23.27%
M_1	TL DO AU	MSE	**0.622**	28.20%
N_1	TL DO AU	Jaccard	0.571	**22.69%**

Table 1. Comparison of the average performance of the test set with the 14 pipelines in regards to F1 score and percentage of incorrect pixels. Trained using the original dataset.
(MCL) Multi-Class Learning, (TL) Transfer Learning, (RO) Random Order, (SF) Stethoscope First, (AO) Ascending Order by number of pixels in class, (DO) Descending Order by number of pixels in class, (PU) Predefined Unknown, (AU) Adaptive Unknown. **BOLD** best in subset.

ID	Pipeline	Loss Function	F1 Score	% Incorrect Pixels
A_2	MCL RO PU	MSE	0.591	17.60%
B_2	MCL RO PU	Jaccard	**0.633**	**15.14%**
C_2	TL RO PU	MSE	0.695	**17.81%**
D_2	TL RO PU	Jaccard	0.701	18.66%
E_2	TL RO AU	MSE	0.625	17.83%
F_2	TL RO AU	Jaccard	**0.727**	18.49%
G_2	TL SF PU	MSE	0.706	**17.47%**
H_2	TL SF PU	Jaccard	0.699	18.24%
I_2	TL SF AU	MSE	**0.710**	17.88%
J_2	TL SF AU	Jaccard	0.694	18.05%
K_2	TL AO AU	MSE	0.726	17.38%
L_2	TL AO AU	Jaccard	0.701	18.80%
M_2	TL DO AU	MSE	**0.727**	**17.03%**
N_2	TL DO AU	Jaccard	0.716	17.48%

Table 2. Comparison of the average performance of the test set with the 14 pipelines in regards to F1 score and percentage of incorrect pixels. Trained using the refined dataset with image frames from videos that have similar lighting and work space setup.
(MCL) Multi-Class Learning, (TL) Transfer Learning, (RO) Random Order, (SF) Stethoscope First, (AO) Ascending Order by number of pixels in class, (DO) Descending Order by number of pixels in class, (PU) Predefined Unknown, (AU) Adaptive Unknown. **BOLD** best in subset.

ID	F1	% Incorrect Pixels	Score	Rank
M_2	0.727	17.03%	0.779	1
K_2	0.726	17.38%	0.776	2
N_2	0.716	17.48%	0.771	3
F_2	0.727	18.49%	0.771	4
I_2	0.710	17.88%	0.766	5
G_2	0.706	17.47%	0.766	6
H_2	0.699	18.24%	0.759	7
C_2	0.695	17.81%	0.759	8
J_2	0.694	18.05%	0.757	9
L_2	0.701	18.80%	0.757	10
D_2	0.701	18.66%	0.757	11
B_2	0.633	15.14%	0.741	12
E_2	0.625	17.83%	0.723	13
A_2	0.591	17.60%	0.707	14

Table 3. Comparing the pipelines using the refined dataset and then ordering them best to worst by score, where the score is a weighted sum of the F1 score and percentage of incorrect pixels. Here we weight the two metrics one-to-one.

correct pixels being 24.458%, which is worse than the six class problem we had been working with previously.

When the distribution of classes, figure 5 , and the confusion matrix, figure 6, is examined it can be seen that not all classes exists in the training set, meaning that the network would never be able to predict these unseen classes. This arised due to the nature of the dataset because each video

Class	F1	Precision	Recall	% Incorrect Pixels
Unknown	0.672	0.564	0.833	14.03%
Gloves	0.795	0.812	0.780	15.62%
Bed	0.857	0.831	0.884	22.19%
Baby	0.762	0.779	0.745	22.97%
Pipes	0.676	0.791	0.590	23.36%
Stethoscope	0.504	0.580	0.445	23.45%
Arms	0.739	0.623	0.909	23.97%
Hat	0.320	0.515	0.232	23.98%
Machines	0.865	0.987	0.770	24.21%
Syringe	1.000	1.000	1.000	24.21%
Blue Towel	0.000	0.000	1.000	24.70%
Scissors	0.000	1.000	0.000	24.71%
Electric Patches	0.000	1.000	0.000	24.71%
Mobile	0.000	1.000	0.000	24.72%
Plastic Bag	0.000	1.000	0.000	24.72%
Packaging	0.134	0.747	0.074	24.75%
Umbilical Cord Clamp	0.000	1.000	0.000	24.75%
Pink Jacket	1.000	1.000	1.000	24.75%
Wires	0.163	0.363	0.105	24.75%
Name tag	0.000	1.000	0.000	24.75%
Umbilical Cord	1.000	1.000	1.000	24.75%
Clothing	0.570	0.508	0.648	24.75%
Airway Opener	0.000	1.000	0.000	24.75%
Average	0.437	0.787	0.479	23.46%

Table 4. The 23 classes shown with their respective F1, precision, recall and incorrect pixel percentages.

doesn't follow an exact procedure, meaning that some of the objects do not appear in some videos. Additionally, only taking a few frames for each video drastically reduces the probability of the majority of classes being present in every frame. The performance of the classes that exist in the training sets are a lot better than the average. The network misclassifies classes that are very similar to the unknown or bed classes. We hypothesise that this is due to the difficulties that arise by the over-exposure of the dataset, making most classes white-washed. Despite this, the network still manages to detect almost all classes in the training set. The prediction of the network on the test set can be seen in 4.

7. Conclusion and Future Work

To conclude, we have shown that the MCG representation firstly allows the superpixels to be learned by a CNN. Secondly, we have shown that the learned superpixels used in a end-to-end manner reduce the number of training instances

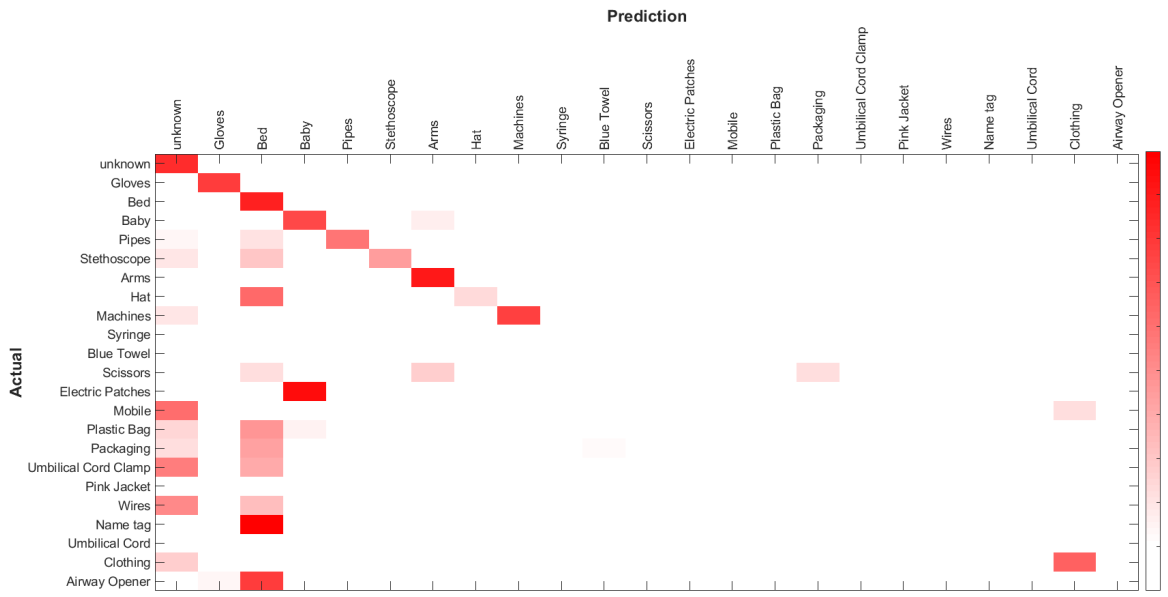

Figure 4. Confusion matrix for the refined dataset. Prediction is across the horizontal axis, and ground truth is across the vertical axis. The darker the colour, the higher the count.

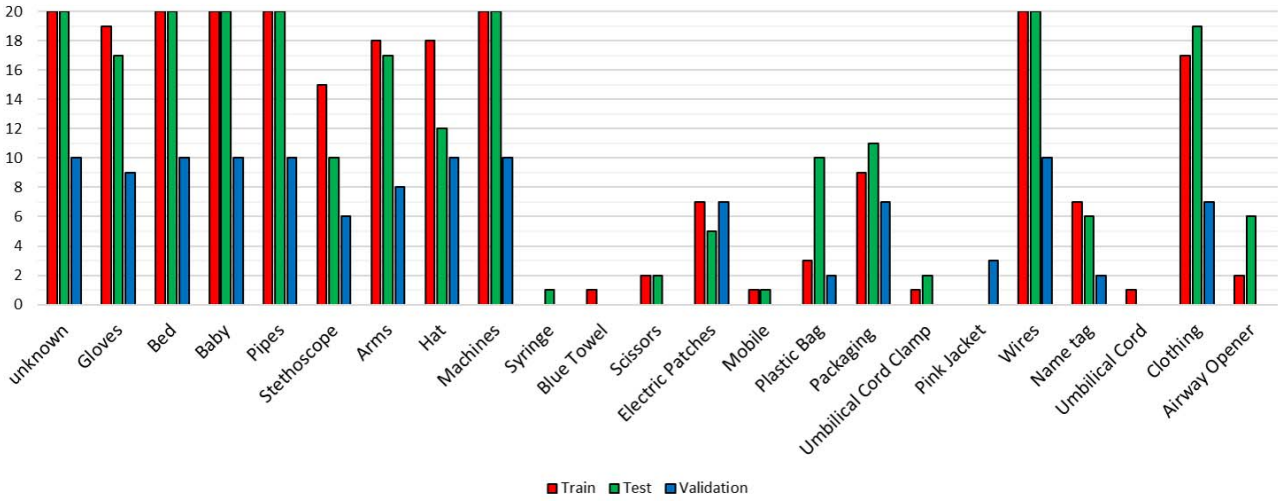

Figure 5. Class occurrences for the refined dataset between training, test, and validation sets.

required to get good performance on niche dataset without having a pre-trained network to bootstrap onto. Doing this shows that small niche datasets should be able to take advantage of the current state-of-the-art deep learning networks without having to have thousands of training instances.

We plan to add more classes and instances of each class to the dataset whilst keeping the number of training instances low to fully annotate the dataset with full scene semantic segmentation. This will then be used to train another network to detect the series of actions performed to help them evaluate the procedure. A possibility would be to test pre-training the network on a large dataset with a wide variety of objects and see if this improves performance. Another possibility is to test augmenting the input data for brightness and saturation to see if with limited data this issue can be over come.

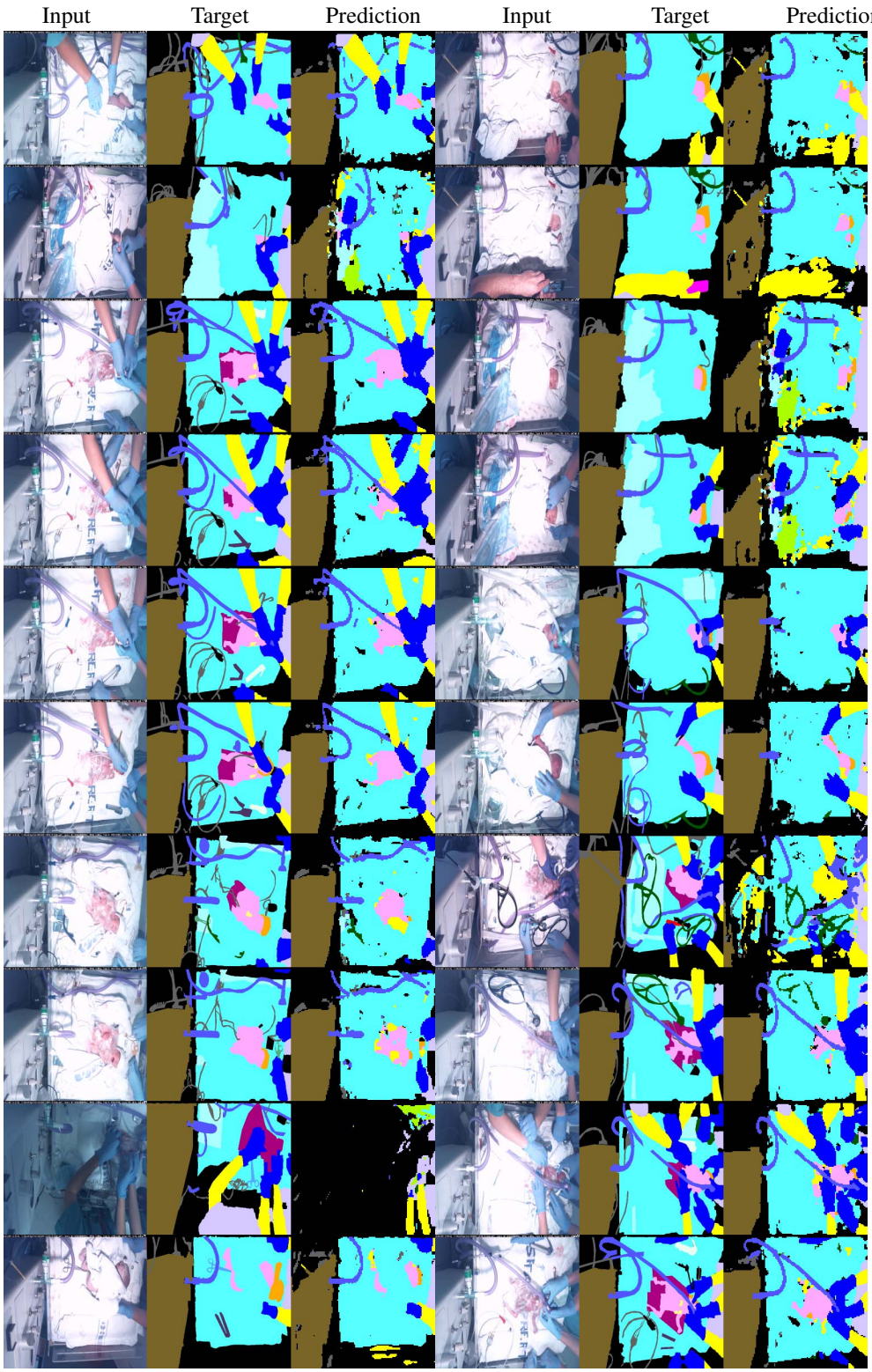

Figure 6. Full test set of images, from left to right: input image, ground truth and prediction. Performance for images with similar exposure performs well, however the underexposed image on row nine performs poorly. Key for ground truth and prediction colours can be found in table 4.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[2] M. Bai and R. Urtasun. Deep watershed transform for instance segmentation. *arXiv preprint arXiv:1611.08303*, 2016.

[3] M. Berman, A. Rannen Triki, and M. B. Blaschko. The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4413–4421, 2018.

[4] A. H. Bhatti, A. Rahman, and A. A. Butt. Video segmentation using spectral clustering on superpixels. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 869–873. IEEE, 2016.

[5] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[6] S. Gu, J. Wang, L. Pan, S. Cheng, Z. Ma, and M. Xie. Figure/ground video segmentation via low-rank sparse learning. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 864–868. IEEE, 2016.

[7] Y. Guo, Y. Liu, T. Georgiou, and M. S. Lew. A review of semantic segmentation using deep neural networks. *International journal of multimedia information retrieval*, 7(2):87–93, 2018.

[8] M. Heo, W.-D. Jang, and C.-S. Kim. Video object segmentation using multiple random walkers with gmm restart rule. In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2016 Asia-Pacific*, pages 1–5. IEEE, 2016.

[9] F. Li, T. Kim, A. Humayun, D. Tsai, and J. M. Rehg. Video segmentation by tracking many figure-ground segments. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2192–2199. IEEE, 2013.

[10] P. Li and Q. Wang. Local log-euclidean covariance matrix (l 2 ecm) for image representation and its applications. In *European conference on computer vision*, pages 469–482. Springer, 2012.

[11] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[12] 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.

[13] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, pages 483–499. Springer, 2016.

[14] M. A. Panicker, H. Frigui, and A. W. Calhoun. Identification of cardio-pulmonary resuscitation (cpr) scenes in medical simulation videos using spatio-temporal gradient orientations. In *2015 International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pages 365–369. IEEE, 2015.

[15] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017.

[16] X. Ren and J. Malik. Learning a classification model for segmentation. In *null*, page 10. IEEE, 2003.

[17] S. Wolf, L. Schott, U. Köthe, and F. Hamprecht. Learned watershed: End-to-end learning of seeded segmentation. *arXiv preprint arXiv:1704.02249*, 2017.

[18] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Low-rank sparse learning for robust visual tracking. In *European conference on computer vision*, pages 470–484. Springer, 2012.