Ultrasound Nerve Segmentation

Implemented in Keras

Sneha Ravikumar
Northeastern University
ravikumar.s@husky.neu.edu

Abstract—This paper describes an approach to segment the Brachial Plexus using Deep Learning. It involves predicting which pixels of an ultrasound image contain the brachial plexus. The dataset is from 2016 Kaggle competition. Deep learning requires the ability to learn features automatically from the data, which is generally only possible when lots of training data is available. This paper uses the U-NET architecture - a network and training strategy that relies on the strong use of data augmentation to use the available annotated samples more efficiently [1]

I. Business Value

A. *Patient Pain Management post surgey*

Even the bravest patient cringes at the mention of a surgical procedure. Surgery inevitably brings discomfort, and oftentimes involves significant post-surgical pain. Currently, patient pain is frequently managed with narcotics that bring a bevy of unwanted side effects. Indwelling catheters that block or mitigate pain at the source can be used as an alternate thus reducing the dependency on narcotics and speed up patient recovery.

B. *Medical Requirements*

Other applications of onsist of border detection in angiograms of coronary, surgical planning, simulation of surgeries, tumor detection and segmentation, brain development study, functional mapping, blood cells automated classification, mass detection in mammograms, image registration, heart segmentation and analysis of cardiac images

II. Dataset and Evaluation

Data was obtained from one of the hosted 2016 competitions.It contains ultrasound images acquired on human necks, and the aim is to segment a collection of nerves called the Brachial Plexus (BP). The data set contains a training set that has been segmented by trained volunteers, and a test set as shown in the figure. The region of interest No obvious feature is present and does not have a clear boundary against surroundings.

Also, as per many candidates, there is some bad data. There are masks that have not been identified right.

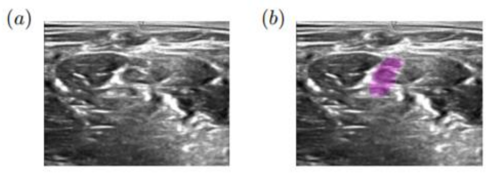


Figure 1. One example of (a) the medical ultrasound images in the dataset, and (b) segmentation of the image by trained human volunteers. The segmented nerves are represented in red.

.

This competition is evaluated on the mean Dice coefficient. The Dice coefficient can be used to compare the pixel-wise agreement between a predicted segmentation and its corresponding ground truth.
If X is the predicted set of pixels and Y is the ground truth, then Dice Coefficient is evaluated as below.

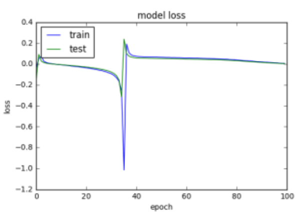
$$\frac{2 * |X \cap Y|}{|X| + |Y|}$$

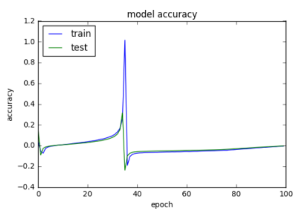
III. Network Architecture

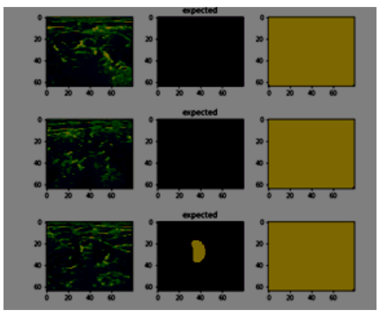
To understand how different layers make a difference to the model, there are 4 models built with the chosen optimizer (Adam with lr =1e-5) and Dice Coefficient score as the the accuracy measure.

A. ***NEURAL NETWORK WITH FULLY CONNECTED LAYERS***

The 64x80 image input was flattened, fed to the network as a 1-D array with a dense layer (5120 neurons) input. It has one hidden layer (512 neurons) and another fully connected layer (5120 neurons).

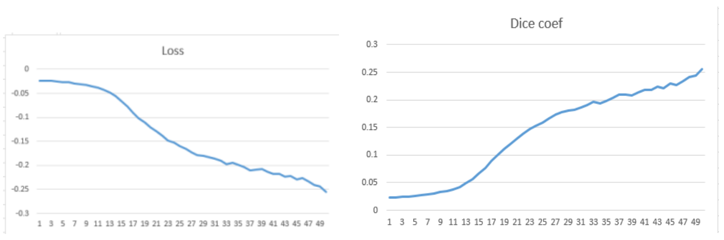


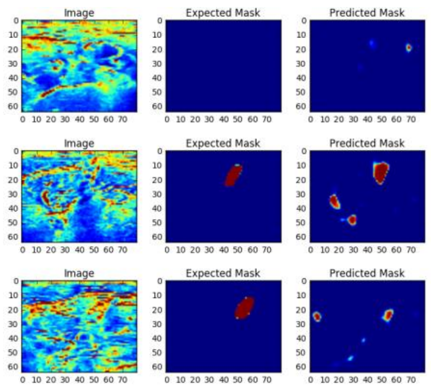




B. U-NET WITH A SINGLE CONTRACTING AND EXPANDING LEVEL

to understand the importance of additional convolutions and depth of the network, this model was built by modifying the U-Net model to have only one contracting and one up-sampling layer keeping all the other hyperparameter consistent.





C. U-NET MODEL

This model consists of two paths –
The contracting path and the expansion path as shown in figure below and proven to be best for segmentation problems [2]
The keras implementation was obtained using Marko Jocic's kernel. This model gave the best dice coefficient (0.69)

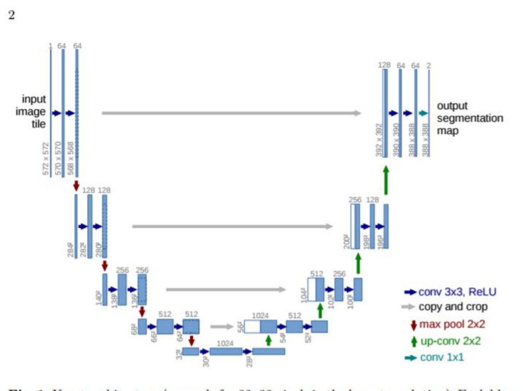
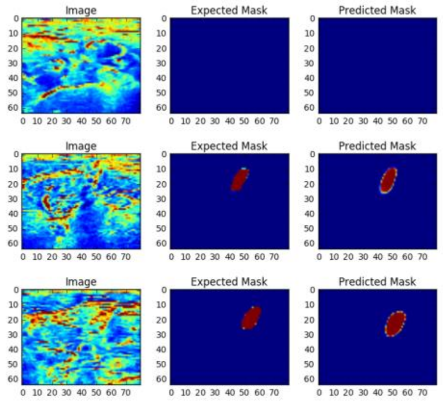


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Results with the validation set using this model



IV. SUMMARISING THE ARCHITECTURE

X_train = 1000 Training images
Y_train = 1000 Training images, with mask value
X_test = 200 test images
Output of the predict() function, returning a mask for each image is also a scaled image saved as n-d array in a .npy file
The 'Train Dice Coefficient' and 'Dice Coef Loss' is the value after 100 epochs for each model

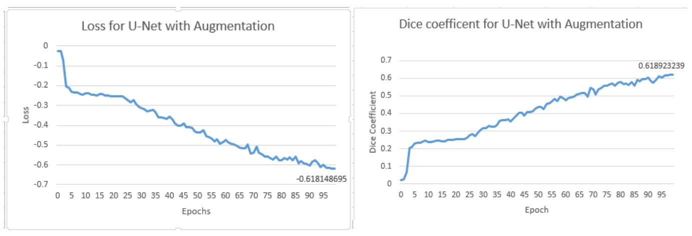
MODEL	AVG. TIME PER EPOCH	DICE COEFFICIEBNT	INPUT TYPE	INPUT SIZE
MLP	5.8S	1.01	1-D ARRAY	5120
SIGLE LAYER UNET	271.6 S	0.4652	2-D ARRAY	64x80
U-NET	498.3 S	0.6967	2-D ARRAY	64x80
AUGMENTED U-NET	600.8 S	0.6189	2-D ARRAY	64x80

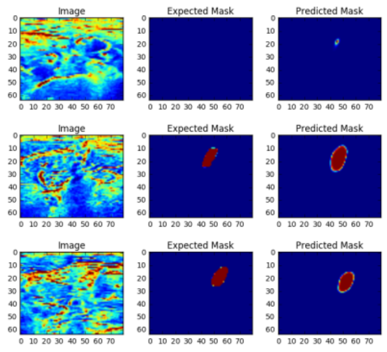
D. U-NET WITH IMAGE AUGMENTATION

To make the most of our few training examples, the images were "augmented" via various random transformations, so that the model would never see twice the exact same picture. This helps prevent overfitting and helps the model generalize better.
The following transformations were made:

a. Horizontal Flip
b. Vertical Flip
c. Image Rotation

I observed that the convergence was slower in case of augmentation, but the prediction was more accurate.





.V. FUTURE WORK

The following can be tried to improve the performance of the network.

1. To remove the images without mask but similar to images containing masks from training data while training
2. Training using bigger training images
3. Using deconv layers instead of Up-sampling layer
4. Trying different types of activation functions.
5. Using bigger kernel size on convolutional layers.
6. Using weighted average of predictions from multiple networks
7. Post processing of the results

VI. REFERENCES

[1]U-Net Image Segmentation Paper
https://arxiv.org/pdf/1505.04597.pdf
[2] Kernels from www.kaggle.com