ULTRASOUND NERVE SEGMENTATION

Mini Project Phase II

Presented by:
Sakshi Kelshikar-30
Shruti Rajput-52
Mansi Sambare-55

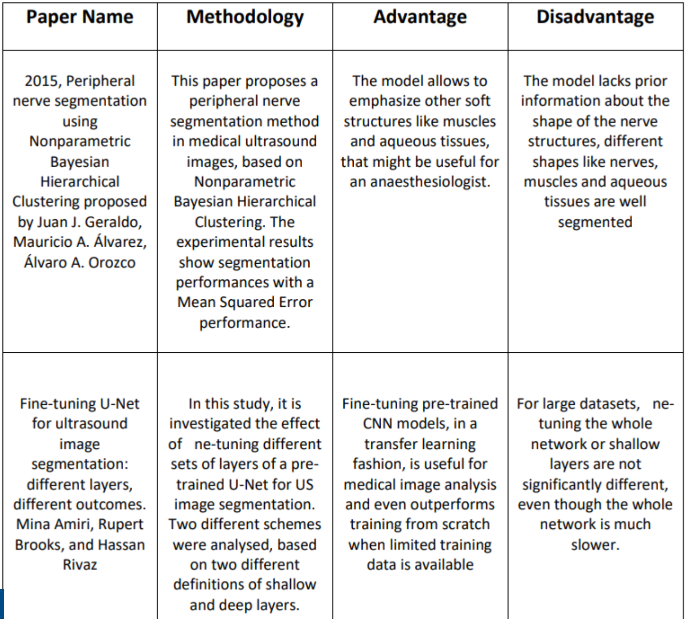
Guided by: Prof.Rachana Dhannawat

Contents

Introduction

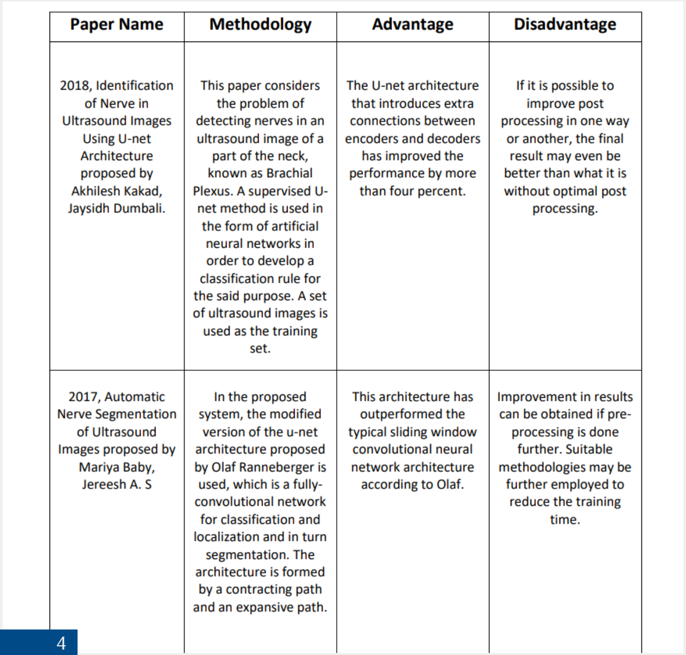
- The accurate ultrasound nerve segmentation has attracted wide attention, for it is beneficial to ensure the efficacy of regional anesthesia, reducing surgical injury, and speeding up the recovery of surgery. However, because of the characteristics of high noise and low contrast in ultrasonic images, it is difficult to achieve accurate neural ultrasound segmentation.
- Ultrasound images provide some information on the anatomy of the particular patient under surgery, and it is possible to use this information for improving regional anesthesia.
- A supervised method called U-net is used in the form of artificial neural networks in order to develop a classification rule for the said purpose. A set of ultrasound images is used as the training set, while another set of similar images is used as the test set, and is used for cross-validation of the resulting classification.

Literature Survey



Paper Name	Methodology	Advantage	Disadvantage
2015, Peripheral nerve segmentation using Nonparametric Bayesian Hierarchical Clustering proposed by Juan J. Geraldo, Mauricio A. Álvarez, Álvaro A. Orozco	This paper proposes a peripheral nerve segmentation method in medical ultrasound images, based on Nonparametric Bayesian Hierarchical Clustering. The experimental results show segmentation performances with a Mean Squared Error performance.	The model allows to emphasize other soft structures like muscles and aqueous tissues, that might be useful for an anaesthesiologist.	The model lacks prior information about the shape of the nerve structures, different shapes like nerves, muscles and aqueous tissues are well segmented
Fine-tuning U-Net for ultrasound image segmentation: different layers, different outcomes. Mina Amiri, Rupert Brooks, and Hassan Rivaz	In this study, it is investigated the effect of ne-tuning different sets of layers of a pre-trained U-Net for US image segmentation. Two different schemes were analysed, based on two different definitions of shallow and deep layers.	Fine-tuning pre-trained CNN models, in a transfer learning fashion, is useful for medical image analysis and even outperforms training from scratch when limited training data is available	For large datasets, ne-tuning the whole network or shallow layers are not significantly different, even though the whole network is much slower.

Literature Survey

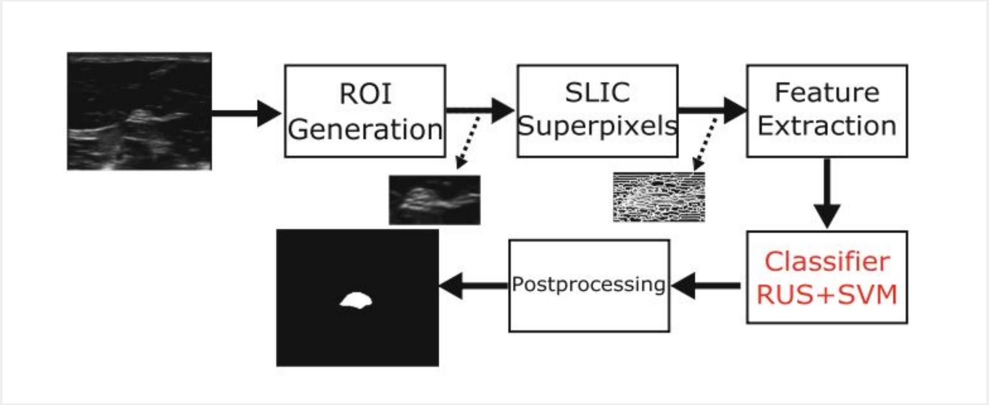


Paper Name	Methodology	Advantage	Disadvantage
2018, Identification of Nerve in Ultrasound Images Using U-net Architecture proposed by Akhilesh Kakad, Jaysidh Dumbali.	This paper considers the problem of detecting nerves in an ultrasound image of a part of the neck, known as Brachial Plexus. A supervised U-net method is used in the form of artificial neural networks in order to develop a classification rule for the said purpose. A set of ultrasound images is used as the training set.	The U-net architecture that introduces extra connections between encoders and decoders has improved the performance by more than four percent.	If it is possible to improve post processing in one way or another, the final result may even be better than what it is without optimal post processing.
2017, Automatic Nerve Segmentation of Ultrasound Images proposed by Mariya Baby, Jereesh A. S	In the proposed system, the modified version of the u-net architecture proposed by Olaf Ranneberger is used, which is a fully-convolutional network for classification and localization and in turn segmentation. The architecture is formed by a contracting path and an expansive path.	This architecture has outperformed the typical sliding window convolutional neural network architecture according to Olaf.	Improvement in results can be obtained if pre-processing is done further. Suitable methodologies may be further employed to reduce the training time.

Existing system

- Even though there are many systems that have been developed till now using different technologies like Deep learning, deep neural networks,Deep Adversarial Networks and U-net neural networks, approaches like Random Under-Sampling (RUS) and Support Vector Machine (SVM) classifier using algorithms such as Principal Component Analysis–(PCA), the accuracy of the models are low as compared to the one using Convolutional Neural Network(CNN).
- The overall accuracies of these methods indicated that the proposed model outperforms all the other methods in the classification.
- CNN tends to be a more powerful and accurate way of solving classification problems.
- In SVM after training and testing the accuracy, classification on some systems the data had a problem of hardware capacity to process it

Block Diagram

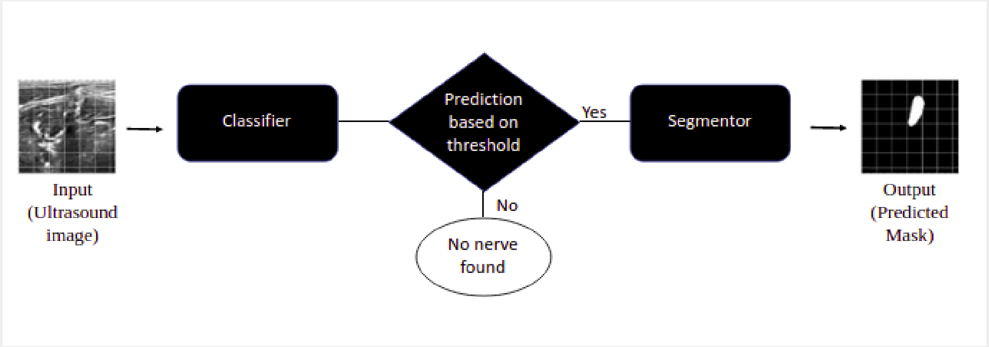


Proposed System

The project focuses on the approach based on image processing using deep learning algorithm CNN for identification of plant diseases. In biomedical application, for the detection of the diseases, it is very essential to have the boundary detail of the acquired image of the organ under observation. Thus it is very essential to extract the edges of the images. Power is one of the main parameters that have to be considered while dealing with biomedical instruments. The biomedical signal processing instruments should be capable of operating at low power and also at high speed. In order to segregate the images into different levels or stage, we use convolutional neural networks for classification.

- Dataset: In our proposed system, we are using a dataset consisting total of 47 subjects and 120 images and masks per subject resulting in a total of 5635 images to train on.We are using CNN algorithm to train and test the model.
- Model: The proposed model will fetch input in the form of ultrasound image from the user and it is given to classifier.If nerve is detected, it is passed on to the segmentor and mask as output is predicted, else 'No nerve found' is displayed

Block Diagram



Hardware and Software Requirements

- Hardware Requirements
  - Processor – Intel Core i3 or above with 1.5 GHz speed
  - RAM of 500 MB and above for all devices
  - Free storage memory capacity of more than 100MB

- Software Interface Requirements
  - Windows/ android/ Linux/ mac/ or any other operating system
  - Mozilla Firefox / Google chrome / opera mini / UC browser or internet explorer
  - Internet connectivity

Implementation

- Dataset The data has been provided by kaggle as part of one of their competitions which requires a model for nerve segmentation. The task was to segment a collection of nerves called the Brachial Plexus (BP) in ultrasound images. A large training set of images where the nerve has been manually annotated by humans was provided.

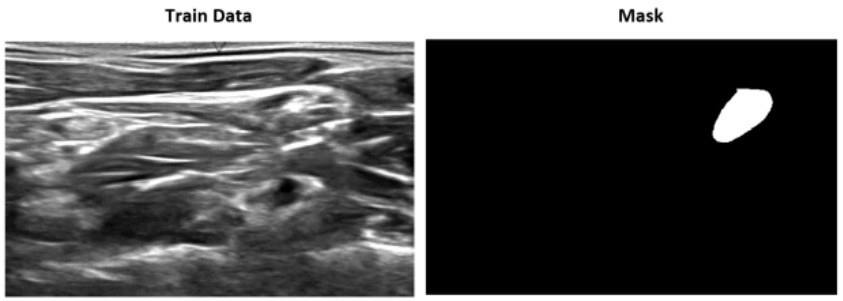
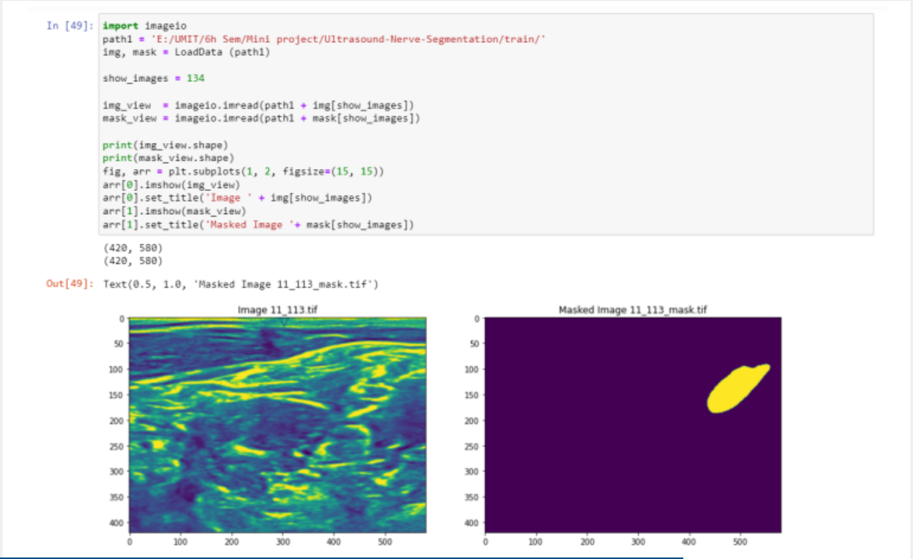


Figure 8: Dataset

Implementation

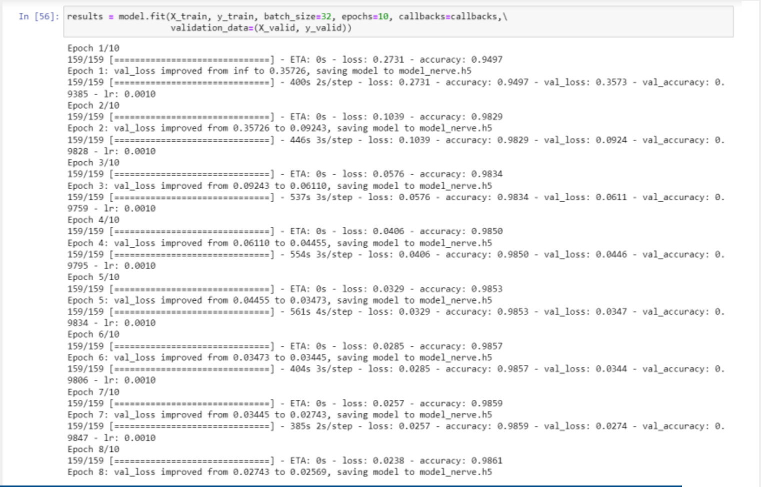
- We read a black-and-white image as a 3d image. After that we convert it into numpy array and display the input image and its masked image along with its shape



```
In [49]: import imageio
         path1 = 'E:/UMIT/6h Sem/Mini project/Ultrasound-Nerve-Segmentation/train/'
         img, mask = LoadData (path1)

         show_images = 134

         img_view  = imageio.imread(path1 + img[show_images])
         mask_view = imageio.imread(path1 + mask[show_images])

         print(img_view.shape)
         print(mask_view.shape)
         fig, arr = plt.subplots(1, 2, figsize=(15, 15))
         arr[0].imshow(img_view)
         arr[0].set_title('Image ' + img[show_images])
         arr[1].imshow(mask_view)
         arr[1].set_title('Masked Image '+ mask[show_images])

         (420, 580)
         (420, 580)

Out[49]: Text(0.5, 1.0, 'Masked Image 11_113_mask.tif')
```

Accuracy

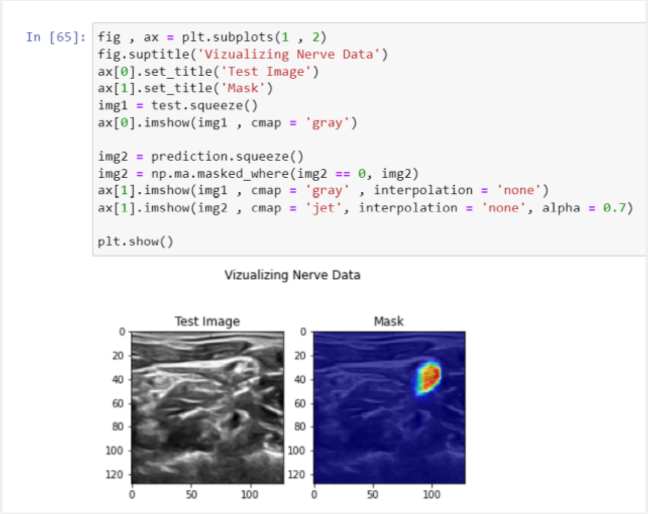
- The two losses both loss and val loss are decreasing and the two acc : acc and val acc are increasing. So this indicates the modeling is trained in a good way. The val acc is the measure of how good the predictions of your model are. So in this case, it looks like the model was trained pretty well after 10 epochs, and the rest training is not necessary.



```
In [56]: results = model.fit(X_train, y_train, batch_size=32, epochs=10, callbacks=callbacks,\
                             validation_data=(X_valid, y_valid))

         Epoch 1/10
         159/159 [==============================] - ETA: 0s - loss: 0.2731 - accuracy: 0.9497
         Epoch 1: val_loss improved from inf to 0.35726, saving model to model_nerve.h5
         159/159 [==============================] - 400s 2s/step - loss: 0.2731 - accuracy: 0.9497 - val_loss: 0.3573 - val_accuracy: 0.
         9385 - lr: 0.0010
         Epoch 2/10
         159/159 [==============================] - ETA: 0s - loss: 0.1039 - accuracy: 0.9829
         Epoch 2: val_loss improved from 0.35726 to 0.09243, saving model to model_nerve.h5
         159/159 [==============================] - 446s 3s/step - loss: 0.1039 - accuracy: 0.9829 - val_loss: 0.0924 - val_accuracy: 0.
         9828 - lr: 0.0010
         Epoch 3/10
         159/159 [==============================] - ETA: 0s - loss: 0.0576 - accuracy: 0.9834
         Epoch 3: val_loss improved from 0.09243 to 0.06110, saving model to model_nerve.h5
         159/159 [==============================] - 537s 3s/step - loss: 0.0576 - accuracy: 0.9834 - val_loss: 0.0611 - val_accuracy: 0.
         9759 - lr: 0.0010
         Epoch 4/10
         159/159 [==============================] - ETA: 0s - loss: 0.0406 - accuracy: 0.9850
         Epoch 4: val_loss improved from 0.06110 to 0.04455, saving model to model_nerve.h5
         159/159 [==============================] - 554s 3s/step - loss: 0.0406 - accuracy: 0.9850 - val_loss: 0.0446 - val_accuracy: 0.
         9795 - lr: 0.0010
         Epoch 5/10
         159/159 [==============================] - ETA: 0s - loss: 0.0329 - accuracy: 0.9853
         Epoch 5: val_loss improved from 0.04455 to 0.03473, saving model to model_nerve.h5
         159/159 [==============================] - 561s 4s/step - loss: 0.0329 - accuracy: 0.9853 - val_loss: 0.0347 - val_accuracy: 0.
         9834 - lr: 0.0010
         Epoch 6/10
         159/159 [==============================] - ETA: 0s - loss: 0.0285 - accuracy: 0.9857
         Epoch 6: val_loss improved from 0.03473 to 0.03445, saving model to model_nerve.h5
         159/159 [==============================] - 404s 3s/step - loss: 0.0285 - accuracy: 0.9857 - val_loss: 0.0344 - val_accuracy: 0.
         9806 - lr: 0.0010
         Epoch 7/10
         159/159 [==============================] - ETA: 0s - loss: 0.0257 - accuracy: 0.9859
         Epoch 7: val_loss improved from 0.03445 to 0.02743, saving model to model_nerve.h5
         159/159 [==============================] - 385s 2s/step - loss: 0.0257 - accuracy: 0.9859 - val_loss: 0.0274 - val_accuracy: 0.
         9847 - lr: 0.0010
         Epoch 8/10
         159/159 [==============================] - ETA: 0s - loss: 0.0238 - accuracy: 0.9861
         Epoch 8: val_loss improved from 0.02743 to 0.02569, saving model to model_nerve.h5
```

Result

- After training the segmentation model for 10 epochs we got a validation score of 0.98 Let's visualize a couple of predictions by the model on the validation set. We can see that our model can segment nerves from ultrasound images accurately.



```
In [65]: fig , ax = plt.subplots(1 , 2)
         fig.suptitle('Vizualizing Nerve Data')
         ax[0].set_title('Test Image')
         ax[1].set_title('Mask')
         img1 = test.squeeze()
         ax[0].imshow(img1 , cmap = 'gray')

         img2 = prediction.squeeze()
         img2 = np.ma.masked_where(img2 == 0, img2)
         ax[1].imshow(img1 , cmap = 'gray' , interpolation = 'none')
         ax[1].imshow(img2 , cmap = 'jet', interpolation = 'none', alpha = 0.7)

         plt.show()
```

Future Scope

- Improvement in results can be obtained if preprocessing is done further. Suitable methodologies may be further employed to reduce the training time.
- A newly proposed network needs to have a wide range of adaptability, and we have to test its performance on other datasets.
- As an improved network, its working mechanism needs to be strictly proved by mathematics to prove that our interpretation of its optimization mechanism is in accordance with scientific basis.

Conclusion

- We have developed a system based on the deep learning method for segmentation of ultrasound images. We have chosen the best possible combination of methodologies to improve the results. Improvement in results can be obtained if preprocessing is done further. In our paper, we propose an improved u-net network called a mini-u-net network.
- As compared to the original u-net and the 4 concatenations of u-net on the Kaggle 2016 neural image dataset, it was found that the mini-u-net works better.On comparison between the performance of the mini-u-net and the networks in the two papers, we have found that u-net can also achieve better results while keeping the dataset unchanged. However, there are still some problems with mini-u-net.

Thank You!