Efficient Detection of Brachial Plexus in Ultrasound Images using Machine Learning Algorithms

Deepak Sharma, Lalit kumar

Abstract: ***Humans can identify objects from an image. Only better knowledge can help to identify specific objects from the field in which we are working. The present work aims at detection of collection of nerves called the Brachial Plexus in ultrasound images. Ultrasound Images has been used for its low prices and low risks but they poses some challenges to detecting the Brachial Plexus in ultrasound images.***

Keywords: Mobile Adhoc Network (MANET), AODV Protocol, Black Hole, cooperative black hole.

I. INTRODUCTION

Surgery oftentimes involves post-surgical pain. Patient fears or back out from the surgical procedure by mentioning. Surgery surely can brings pain, and discomfort. Managing pain involves the use of narcotics which have several unwanted side effects; under or overdosing respiratory side effects and sedation. One way to manage pain with less dependency on narcotics by using of indwelling Catheters that deliver anesthetic. Pain management catheters block the pain at the source. These catheters are inserted in the area around the nerves that carries sensation from the surgical site. It is therefore imperative to accurately identify nerve structures in order to effectively insert the catheters. The dataset for this project is taken from Kaggle competition "Ultrasound Nerve Segmentation". It consists of ultrasound images of the neck, from which nerve structures can be identified to improve the placement of the catheter. The ultrasound imaging is a powerful tool for visualizing the Brachial Plexus due to its superior advantages, e.g., portability, real-time imaging, low-cost and free of radiation. The routine work done by manual can be time-consuming and production rate of work will be limited. The automated techniques can reduce the workload and enhance the efficiency. The main objective is to build a predictable model to detect the Brachial Plexus in Ultrasound Images of nerve structure of neck, proposing a solution to ignore the surgical procedures and consumption of narcotics by patients. The generated trained model can be useful in several ways. For example, it can be integrated with ultrasound apparatus to automatically show the area of interest.

Revised Manuscript Received on October 30, 2019.

* Correspondence Author
Deepak Sharma*, Research scholar, MD University, Rohtak, India. Email: erdeepaksharmabwn@gmail.com
Lalit Gaur, PG student, C-DAC, GGSIP University, New Delhi, India. Email: lalitgaur45@gmail.com

II. LITERATURE REVIEW

In this section we are highlighting some of the best known work in this field and provided a well analyzed and brief review about that work. In this section, we have discussed many different approaches and studied their methodologies to achieve desired results.

P.A.Venktachalam et al. [1] proposed a system for processing of Abdominal Ultrasound Images Using Seed based Region Growing Method. The objective of this work is to build an Ultrasound processing Model for the showing the affected regions in abdomen by implementing histogram equalization and region growing method. As a Data Set for the model more than 20 images of abdominal US Images were taken from General Hospital, Ipoh and Fathimah Hospital, Ipoh, Malaysia. Results of this research experiment shows a clear image of an affected regions in abdomen & margins. Also it is concluded that It cannot detect automatically, but grayscale gradient improving slowly, which was depend on threshold values, can be viewed which was helpful. To meets the expectations of radiologist, it was operated manually to change the threshold values.

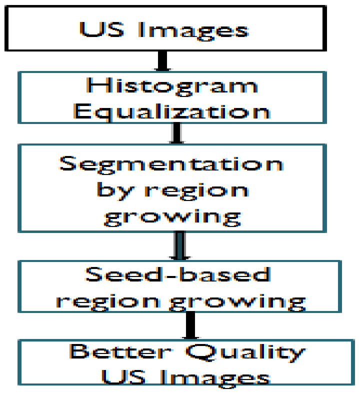

Fig. 1. Approach used [1]

Hitesh Garg et al. [2] proposed a system for the Segmentation of Thyroid gland in Ultrasound Images using neural network. The Objective of their work is to develop an automated approach for segmentation of targeted area which is thyroid gland.

More than ten Ultrasound Images from five patients were used as a Data Set in this project. The approached used in this project for segmentation takes texture as the criteria and intensity of pixels. In this proposed approach FeedForward neural network is used for intensity based classification at the end by first removing the noise then various pixel based features are extracted to calculate texture. It shows that neural network got better results.

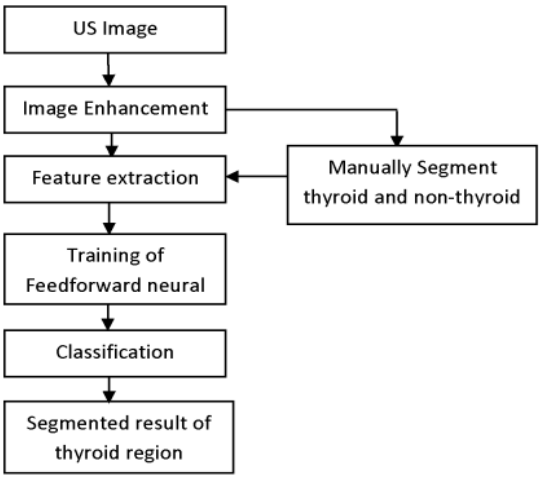
US Image
Image Enhancement
Feature extraction
Manually Segment thyroid and non-thyroid
Training of Feedforward neural
Classification
Segmented result of thyroid region

Fig. 2. Approach used [2]

Ashnil kumar et al. [3] proposed a system for Plane Identification in Fetal Ultrasound Images Using Saliency Maps and Convolutional Neural Networks. The objective of this work is to propose an approach for the classification, in Fetal Ultrasound images, of anatomical planes. 7568 Ultrasound images are used as dataset in this work. These Ultrasound Images were taken at the Nepean Centre for Perinatal Care, University of Sydney from 185 fetuses. In this work an efficient approach was proposed for the classification in 2D Fetal Ultrasound images of anatomical planes, which was integrated the most salient points in images with raw Ultrasound images features to compensate for the characteristics of Ultrasound images of Fetal images. Originally the method was designed for classifying different facial planes, but in this approach the method was designed for classifying different anatomical regions in contrast to the baseline.

Gustavo Carneiro et al. [4] proposed a system for the Segmentation of the Left Ventricle of the Heart from Ultrasound Data Using Deep Learning Architectures and Derivative-Based Search Methods. The objective of this work is to propose a new technique of supervised learning for segmentation of left ventricle of heart in US Images. 400 annotated images of diseases cases are used as dataset for this work (from 12 sequences) and also another normal cases of 80 annotated images (from 2 sequences) are used as extended dataset. In this technique, it produces the LV segmentation without constraint (temporal) and severe reduction of the training set size showed robustness. In this work some issues that can affected badly on supervised learning techniques are requirement of large train annotated dataset, and the processes which are deeply complicated. After getting the results, the proposed technique it is robust to larger training dataset and gradient descent and Newton's method search processes showed a reduction of up to tenfold in the search complexity.

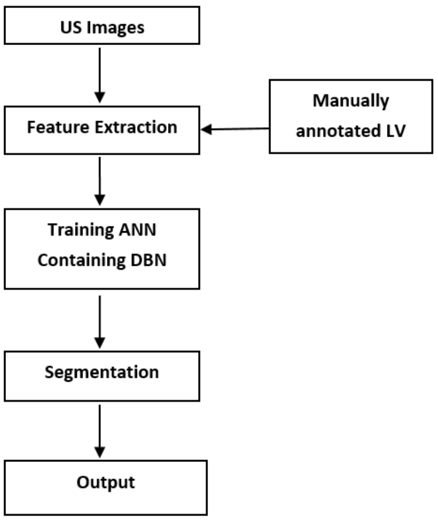

Fig. 3. Approach used [4]

Olaf Ronneberger et al. [5] proposes a system for U-Net: Convolutional Networks for Biomedical Image Segmentation. The objective of this work is to modify FCN architecture and get more accurate segmentations by extending the architecture. The training dataset of 30 images which is used in this work provided by EM segmentation challenge. The proposed architecture gets good results on different biomedical images. It will need very few annotated images by applying data augmentation with elastic deformations, and has less training time. By using data augmentation with elastic deformations, only 10 hours on a NVidia Titan GPU (6 GB) for very few annotated images.

III. PROPOSED APPROACH

In this section, we present our approach towards our objective. The goal of this project is to identify a collection of nerve structures called the Brachial Plexus (BP). Given an ultrasound image, highlight or annotate the area in the image where the BP is located. By applying deep learning in computer vision to recognize the Brachial Plexus in ultrasound images. In doing so, First we have to design a deep neural network. After designing a deep neural network we have to train this neural network from a dataset of ultrasound images. Lastly we have to create a script that accepts images as input, and outputs the same images with annotated Brachial Plexus, if present.

A. Idea behind the proposed approach

The idea behind the proposed approach is that in this semantic image segmentation problem, a deep neural network will be used to predict the labels for each pixel in the image. Yann Lecun introduced convolutional networks from which most state of the art image recognition techniques are derived. In the dataset, the pixels are roughly similar all throughout the image. There is no clear groupings or sharp boundaries with which an untrained eye can clearly identify the brachial plexus.

It can be theorized that, in order to classify a pixel as to whether or not, it belongs to a brachial plexus, the immediate surrounding pixels need to be taken into account. The larger the patch around the target pixel, the better chance of classifying the pixel correctly. Translating this concept to a convolutional network, it would mean larger patch size. However, performing convolutions with a large patch is very computationally expensive.

B. Working principle

Image pyramid can be used to generate a series of down scaled images. Using the same patch size across the images, the patch will create a contextual window around the pixel. The patch on the smaller scaled down image will have a blurred larger view version of the image, while the patch on the original full size image will have a higher resolution view of the image. Image pyramid will allow the use ofa smaller sized patch while having a larger receptive field on the image. Applying inception architecture will increase the model size while utilizing computational efficiency and low parameter count. The overall neural network is shown in Figure 5. Image pyramid is generated, which are then fed to layers of convolutional network, followed by inception layers. The outputs of inception layers are gradually up-sampled, where needed, to the original input image size, and then concatenated. The concatenated layers are then fed to a series of convolutions. The final layer is a 1x1 convolution with a depth of 2, one for each positive and negative class. The classification layer uses softmax to generate the prediction output mask.

Three versions of inception modules will be used for this algorithm:

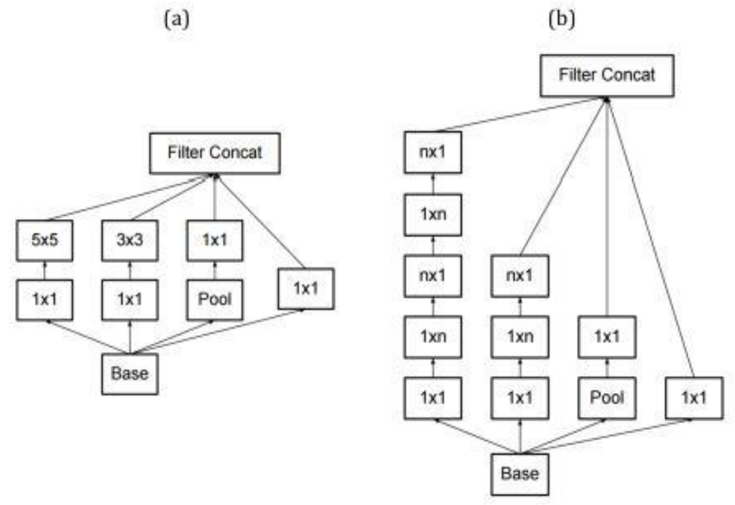

Fig. 4. Inception modules: Version (a) is the original inception module. Versions (b) and (c) are newer versions of inception. For version (b), an n=7 is used for this project.

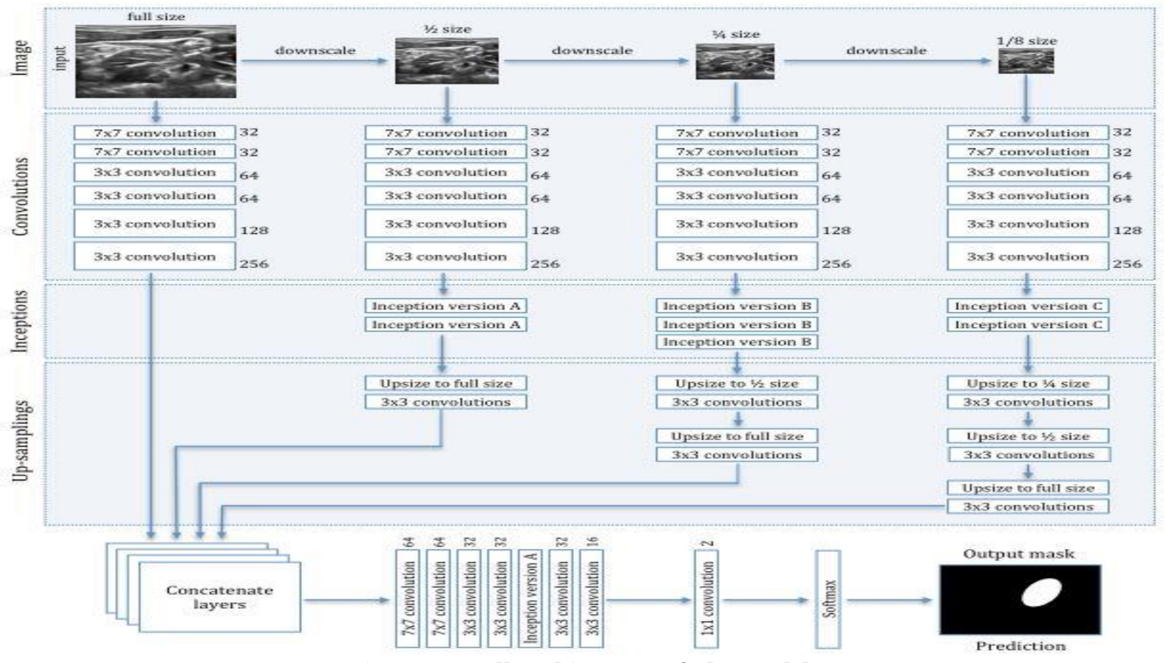

Fig. 5. Overall architecture of the model.

C. Data Set Exploration

There are 5,635 images for training and 5,508 images for testing. The images are grayscale with dimensions 580x420 pixels and are noisy. Training images have masks to indicate where the BP is present, while there is none for testing images. In the training images, only 2,323 images have positively identified the BP. There are inaccuracies in the annotations of the BP. There is no prefect ground truth or gold standard. There are very similar images but with conflicting annotations; one image has positively identified the BP,however the other has none (see Figure 7). There are also very similar images that have positive annotations, but the area where they are annotated differ in shape/area, although the annotations are located approximately in the same region (see Figure 8).The classes are severely imbalanced; consisting of 1.2% positive class and 98.8% negative class. This class imbalance is addressed later in the Refinement section.

Where the BP is present in an image, the BP annotations has the following characteristics:

Groups of images are taken from the same patient. Images that come from the same patient are highly correlated (see Figure 6). Some images are very similar, but they are not exactly the same.

Very similar images are expected to have similar annotations. However, due to human error during manual annotation of the dataset, there are very similar images that have conflicting annotations. One image has the BP annotation, while another very similar image has none. The annotations are not very accurate. There are similar images that have varying annotations; the shapes and sizes are different (see Figure 8). However, they are located in the same general location (see Figure 9). Some annotations of similar images vary dramatically as can be seen in images from patient 23 (see last row of Figure 8).

.

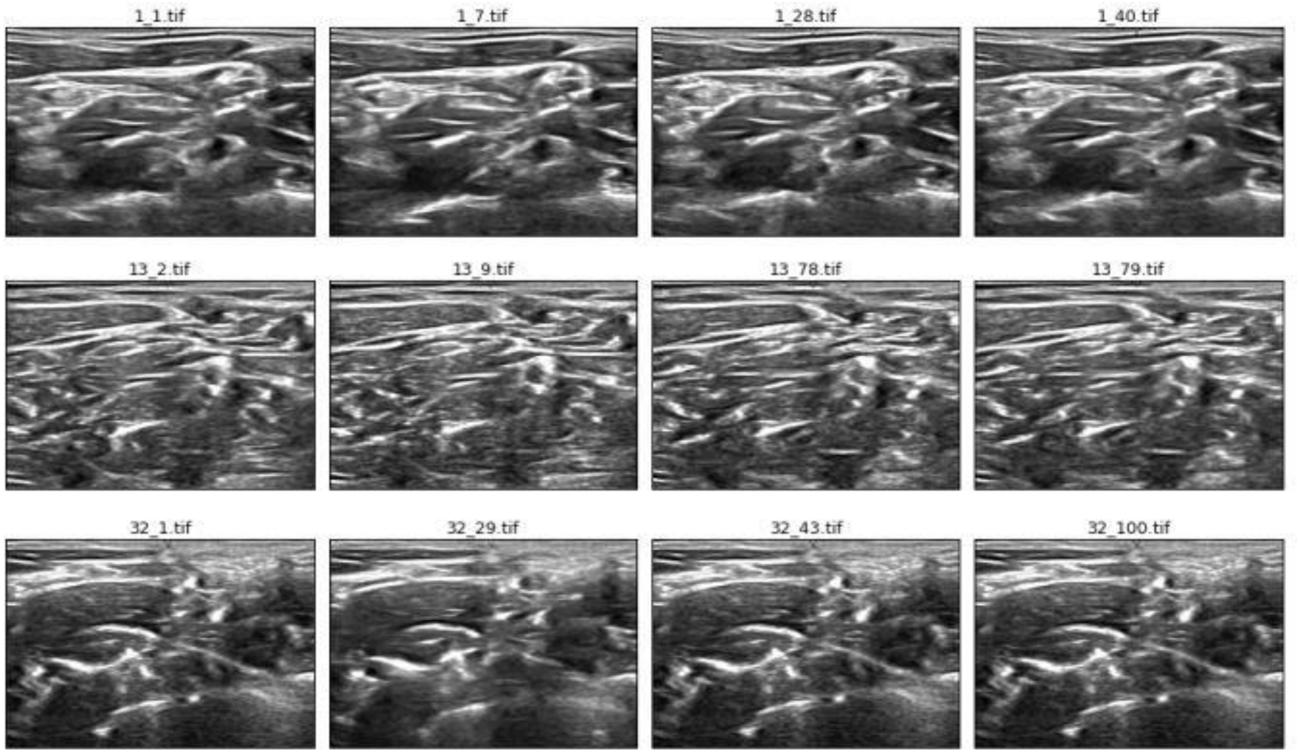

Fig. 6. High correlation of images.

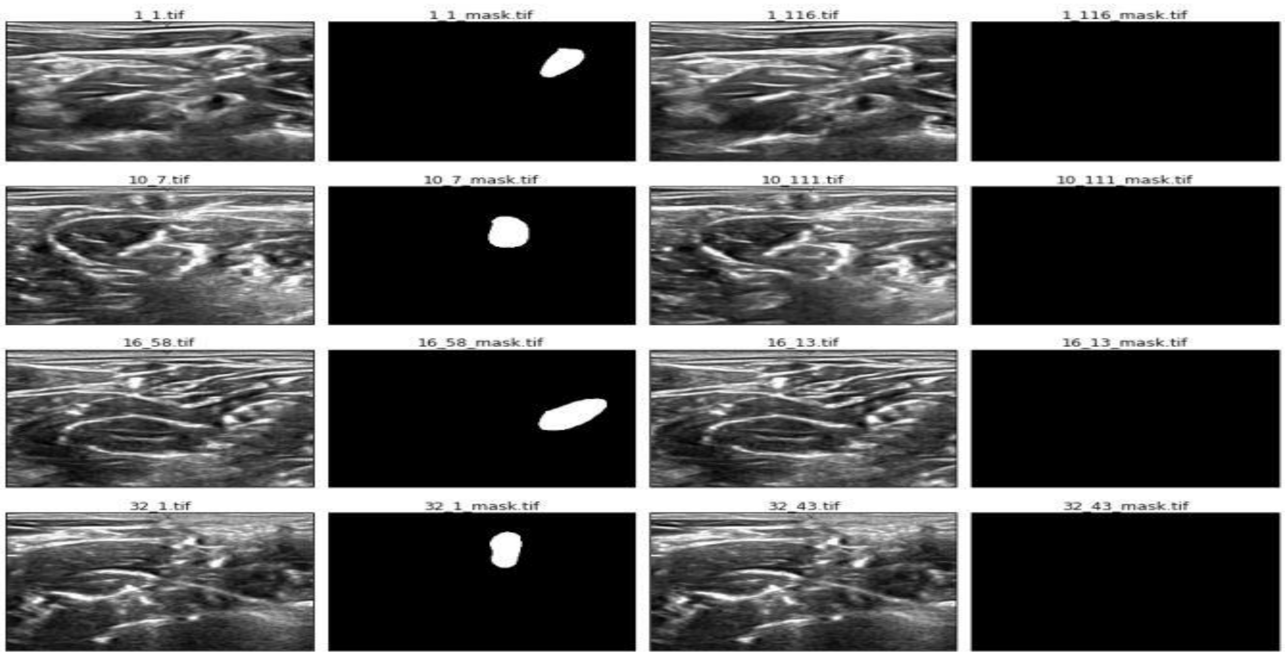

Fig. 7. Conflicting annotations

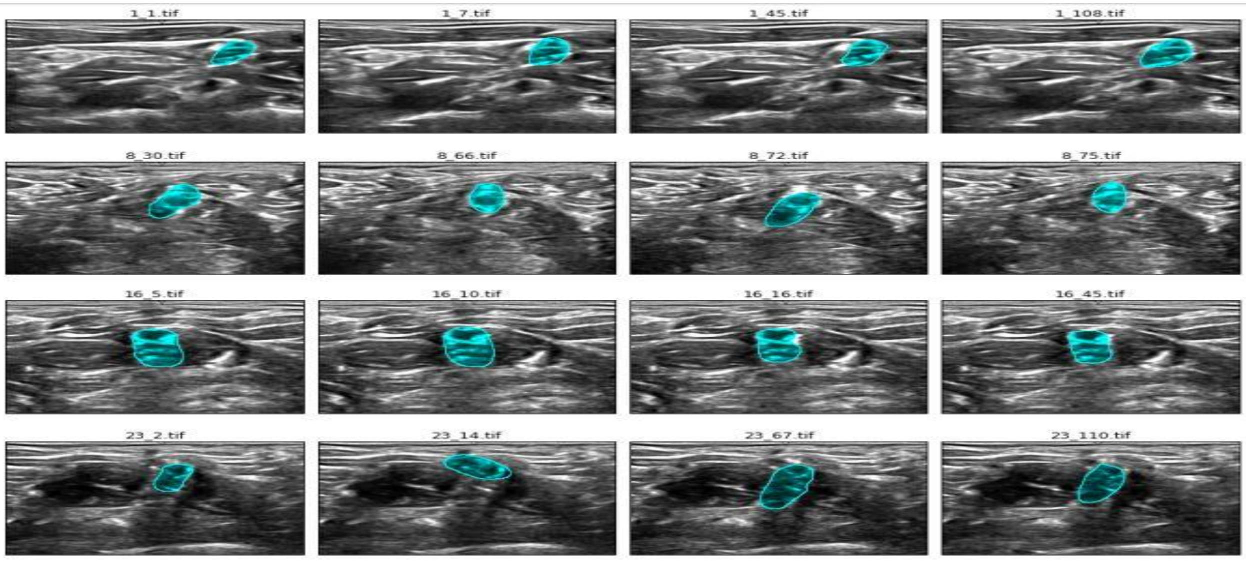

Fig. 8. Similar images with varying annotations

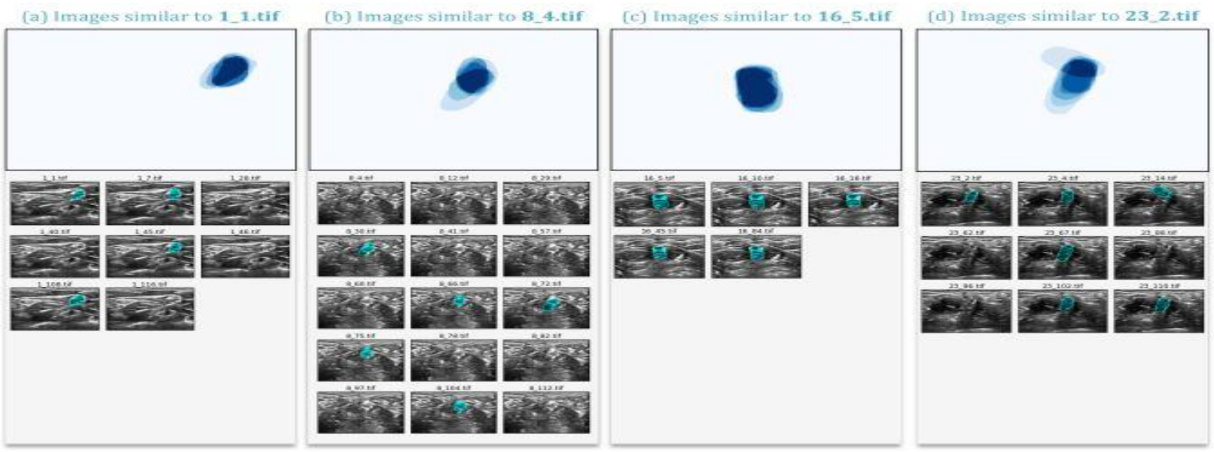
(a) Images similar to 1_1.tif
(b) Images similar to 8_4.tif
(c) Images similar to 16_5.tif
(d) Images similar to 23_2.tif

Fig. 9. Average annotations of similar images.

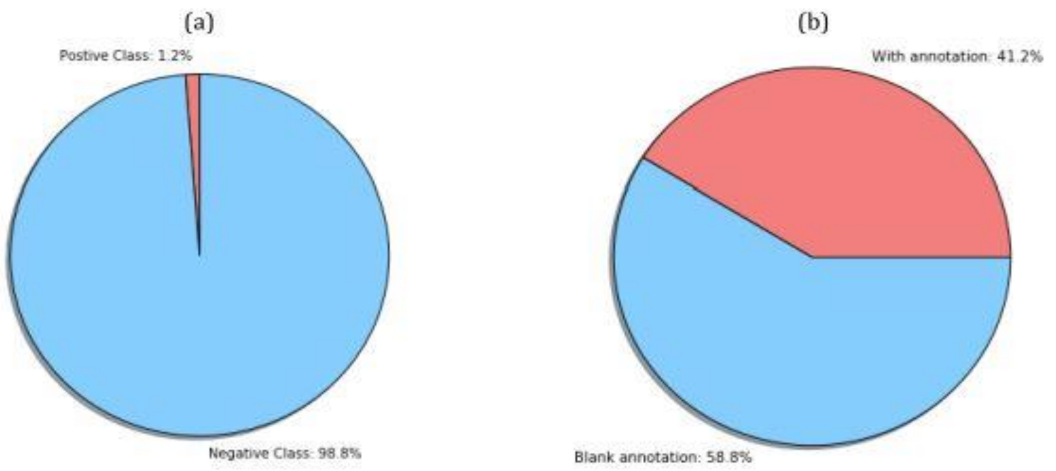

Fig. 10. Distribution of classes. (a) Shows the classes are severely imbalanced (b) shows majority of the images has no annotations.

The Figure 10 show the distribution of classes of the dataset. In terms of total number of pixels, it is notable that the negative class greatly outnumbers positive class. The majority of images are devoid of an annotation. Accuracy can also be used as the metric of choice, however, a quick look at the dataset shows that there is a huge imbalancebetween positive and negative classes; therefore it would make accuracy a not very reliable metric for this case. To set a benchmark for measuring the performance of the model, all pixels are set to positive class and thenthe F1 score is computed against the ground truth of the validation set. The resulting F 1 score is 0.0362. Randomly setting the pixels to positive and negative classes resulted in a lower F1 score of ≈0.0356.

D. Flow charts

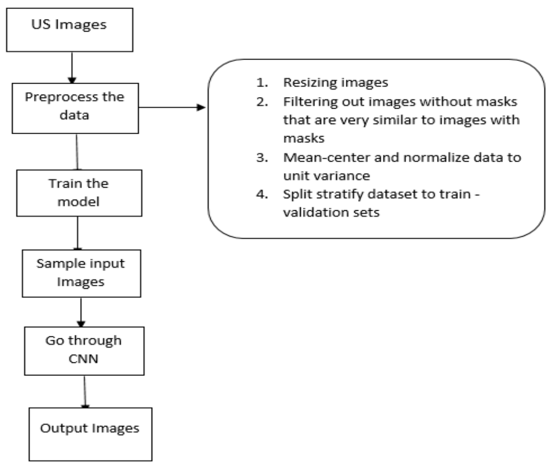
US Images
Preprocess the data
Train the model
Sample input Images
Go through CNN
Output Images
1. Resizing images
2. Filtering out images without masks that are very similar to images with masks
3. Mean-center and normalize data to unit variance
4. Split stratify dataset to train - validation sets

Fig. 11. Flow graph of Proposed System

IV. IMPLEMENTATION AND RESULTS

This section explains how the new proposed approach has been implemented.

A. All about the Data Set

The train dataset can be downloaded from: https://www.kaggle.com/c/ultrasound-nerve-segmentation/download/train.zip. Data pre-processing involves 4 sub-processes. These processes are resizing images, filtering out images without masks that are very similar to images with masks, Mean-centre and normalize data to unit variance, split stratify dataset to train-validation sets. The pre-processing code is in preprocess.py. This code handles all the needed pre-processing steps. The ultrasound images are large files and are noisy. To reduce the noise and have faster training, the images are downsized to 96x128 pixels using inter-area interpolation. As shown in Exploratory Visualization section, several images are very similar but have conflicting masks. This will greatly negatively affect the learning process. To mitigate the negative impact, images without masks, but with very similar images that have masks, are filtered out (images without masks and have no similar images are retained). The reason for dropping these images is, they most likely have similar annotation with their annotated counterpart, and were overlooked during manual annotations (it is more likely to miss to annotate an image than accidentally annotating random location of the image).

For similar images with varying annotations (in terms of shapes and sizes), they mostly cover the same general area of the image, and have common intersections. These images are retained for training to force the model to figure out what is common between the annotations. The similarity of the images are measured using normalized cross-correlation with a threshold of 0.7 (similaritythreshold can be configured in config.py).

To have better training performance, the dataset is centered at zero mean and normalized to unit variance. This was done by subtracting the mean and then dividing by the standard deviation of the dataset. The dataset is split to 80% train and 20% validation sets. The split is stratified based on the presence of mask. The images with the same patient ID are highlycorrelated, thus the split is also stratified based on patient ID.

B. Implementation of the Proposed Model

Tensorflow is used as the main deep learning library.

The entry point of the code is train.py, where the model is also implemented. The train and validation dataset (output generated from preprocess.py) are fetched. The model function andthe dataset are passed to run_training() function which handles the execution of the training process. The run_training() function, which is located in engine.py, reads the latest checkpoint file. This allows the training to resume from previous runs in case it was interrupted. The script saves a checkpoint atthe end of every epoch.

The compile_model() function generates the tensor graph for training. It uses the supplied make_model() function to create logits. It also handles generating of tensor graph for the loss function and the optimizer. The loss_and_predict() function computes the loss value using cross entropy. It flattens the logits and target labels tensors before computing the softmax and cross entropy value. Since both computations for prediction and cross entropy requires flattening of the logits, the same segment of the tensor graph is used for this operation. For efficiency, both loss and prediction is returned by a single call of this function. Adam optimizer is used for learning with exponential learning rate decay. The initial learning rate, decay rate, and decay steps are set in config.py. Gradient clipping is applied to stabilize the loss function. The gradient clipping value is also set in config.py. The do_training () function executes the training loop. Each loop constitutes 1 epoch. Each epoch runs several iterations, where every iteration step constitutes 1 mini-batch of training dataset. The training dataset is divided into mini-batches. At the beginning of each epoch, the train dataset is shuffled so each mini-batch contains different images every epoch. At the end ofeach epoch, validation is performed using a process similar to training, except that it is using the validation dataset. The number of epochs to run mini batch sizes for training and validation are set in config.py. In every iteration step, the number of correct and incorrect predictions (true/false positives, true/false negatives) is recorded for metrics gathering. At the end of every epoch, F1 scores for training and validation are computed using np_f_beta_score() function which can be found at lib/stats_tools.py. The scores are saved in output/training_log.csv file. Checkpoint files are saved for every epoch at output/checkpoints/. Since checkpoint files can easily take up disk space, a limit is set as to the number of check points to keep, which can be configured in config.py. Older checkpoints are automatically deleted when the limit is reached. The checkpoint point file that has the highest F1 validation score will not be deleted. The training ends when the number of set epochs to run is exhausted or when the user presses CTRL+C in the command line. Running train.py again will resumethe training from the last saved checkpoint. Training takesapproximately ≈35 inutes per epoch on Amazon AWS g2.2xlarge instance with GPU support. A pre-trained model, that ran for 21 hours with 35 epochs. The model hyper parameters were adjusted iteratively. The initial model was trained using default parameters and learning rate of 0.01. The loss was dropping too fast within the first epoch. Several learning rate values were tried out, decreasing in order of magnitude, until a gradual descend of loss was obtained.

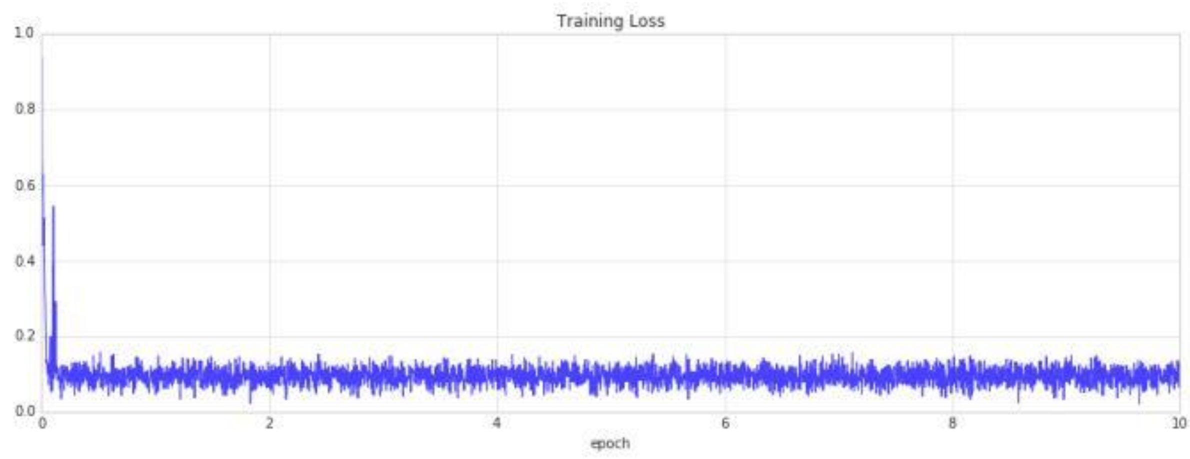

Fig. 12. Training Loss with learning rate set to 0.01. (The learning rate dropped too swiftly in just a few iterations.

C. Results

Due to class imbalance, the model preferred to always predict negative class. To correct this, weighted class was applied using median frequency balancing, where the weight of a class is the ratio of the median of class frequencies in the entire training set divided by the class frequency. The resulting weights are normalized so that it will be between 0.0 and 1.0. This impliesthat the positive class will have a weight of 1.0 (since the positive class has smalle frequency) while the negative class will have a weight <<1.0 (approximately ≈0.03). The class weights are multiplied to the logits before the loss is computed. The model overfitted very easily. Dropout layers were then applied at the end of the inception layers and at the concatenated layer. Dropout keep rate was gradually adjusted from 0.8 down to 0.5. To further improve the overfitting issue, L2 regularization was applied. Several regularization strengths were tried starting from a weak value of 10^{-6}, gradually increasing in order of magnitude until the model underfits.

The learning rate that produced reasonable result is around 0.00001. Higher learning rate converged to higher loss value while with lower learning the training was not able to converge after several epochs. Exponential learning rate decay of 0.98, decayed at every epoch, helped stabilized the loss convergence at the later stage of training.

With limited training data, the model very easily overfits. The model had high F1 score on training set, but performed badly on validation set. Applying dropout and L2 regularization minimized overfitting. A dropout keep rate of 0.5 and L2 regularization strength of 0.005 yielded reasonable results, which reduced the gap between training and validation F1 scores.

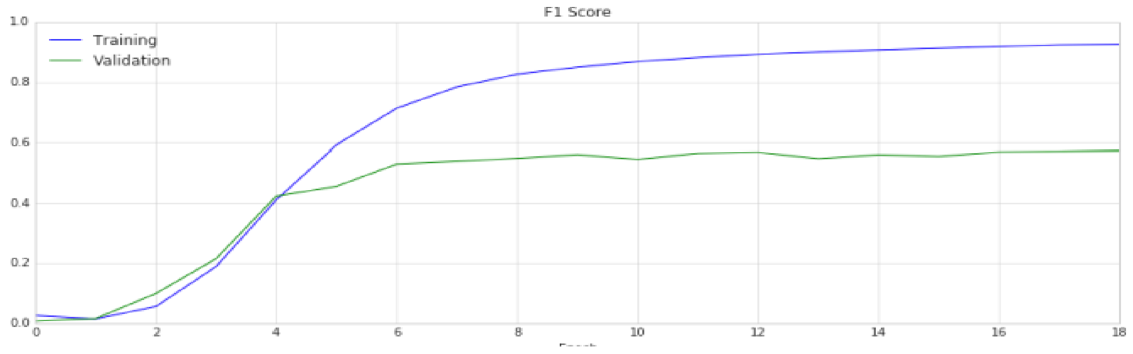

Fig. 13. Overfitting model. (There is a big gap betweentraining and validation, which indicates that the model is overfitting.)

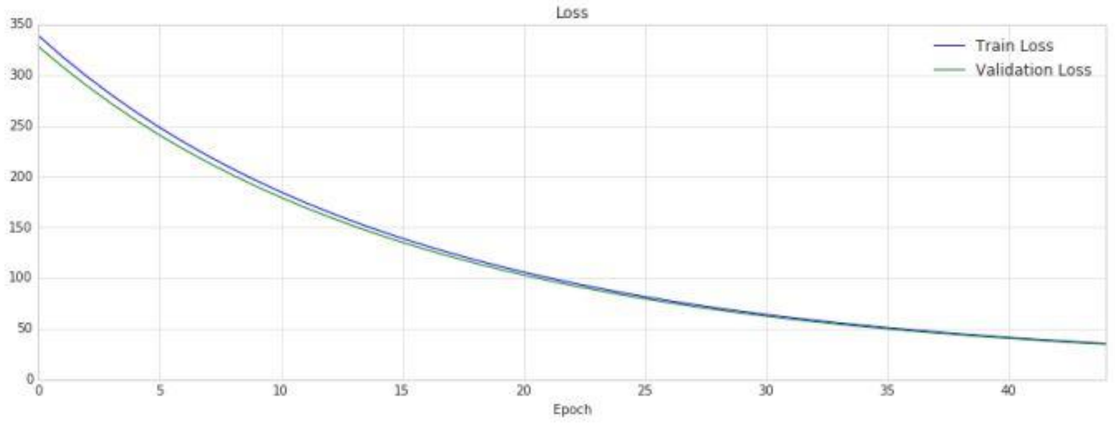

Fig. 14. Loss graph of the final model

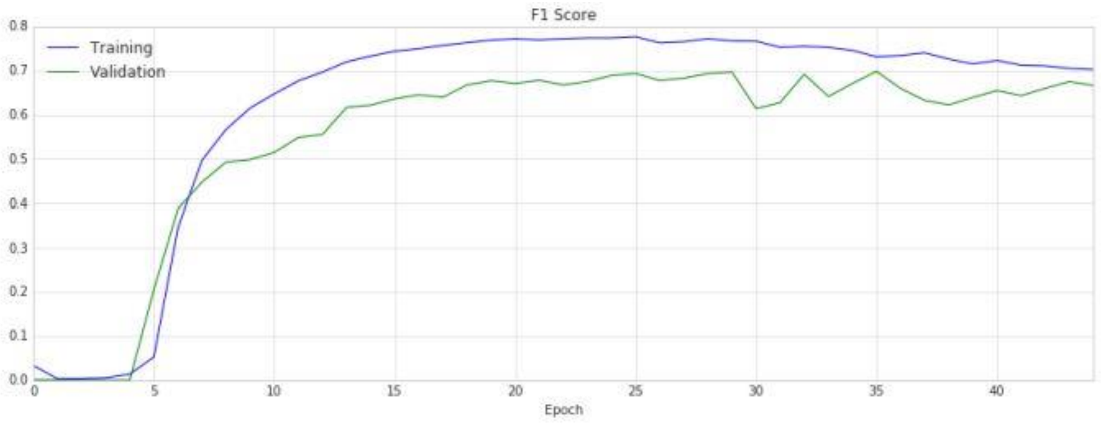

Fig. 15. F1 score graph of the final model.

The final model was chosen using early stopping. The checkpoint with highest validation score was chosen as the final model. The validation set uses images that were not included during training. It generally has lower score compared to training, which reflects more accurate performance metric of the model. To verify the robustness of the model, inference was run using the test dataset. This dataset does not have accompanying ground truth that can be compared with to produce a quantitative measure. It is rather difficult for untrained eyes to make a judgment as to whether or not the predicted annotations are accurate. As a non-medical professional, we can look over and again the training set and pick up general pattern of nerve structures. However, this would still be not very reliable and consulting with a trained medical professional would be best to verify the result.

Some samples predictions using the test dataset are shown in Figure 15 for qualitative assessment. The final model yielded an F1 score of 0.698 on validation set which is significantly higher than the benchmark 0.0362. Given that the training data has a good portion of inaccuracies, the predictions of the model is sufficient enough to point the location of nerve structures, or the absence thereof. The predicted area of the image shows the general location of the brachial plexus.

V. CONCLUSION AND FUTURE SCOPE

A. Conclusion

As a non-medical person, the most challenging part of the project is not having knowledge on what brachial plexus exactly looks like. Even after going through all the sample images, it is still difficult to identify the nerve structures in the ultrasound images without medical training. It is interesting to note that the machine learning model was able to identify nerve structures after training within hours, while an untrained person would not be able to figure out the pattern from the same set of images. Some images from validation and test set that were not used in training were evaluated. It can be seen that the model performs well on most images. However, there are also cases where it misses. The area highlighted in cyan is the ground truth while the area outlined in yellow the prediction produced by the model. The model predictions are almost identical to the ground truth.

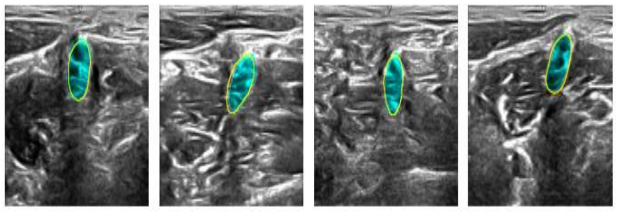
Fig. 16. Good performance of the model

From left to right: (a) the model prediction mostly missed the ground truth, (b) the model hit the ground truth but also annotated an extra area, (c) the model was not able to detect brachial plexus but there actually exists, and (d) the model was able to find brachial plexus where human annotator missed out.

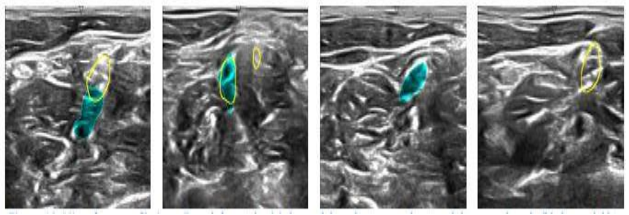
Fig. 17. Missed out predictions.

The yellow highlights are the prediction of the model. The test dataset has no accompanying ground truth to compare the results with. Consultation from trained ultrasound professional is needed.

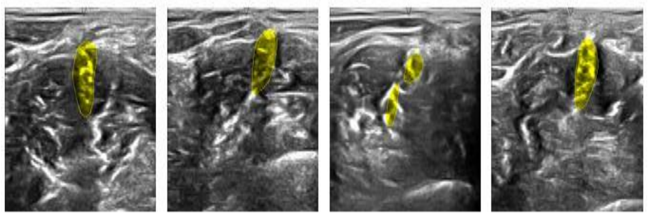
Fig. 18. Inference on test dataset.

B. Future Scope:

The accuracy of the result can be greatly improved with more training data with higher accuracy. In addition, data augmentation may also help improve the result. Using a different neural architecture, such as U-net may have faster inference time. Using another deep learning library, such as Theano or Torch, may improve training time. Significant improvement on training time would be very useful to make the model run on a real time application in video input, or run natively in less powerful devices such as on mobile platform. Scheme is lost which also result in false detection of black hole node.

REFERENCES

1. P.A.Venktachalam, Ahmad Fadzil Mohd Hani, Umi Kalthun Nag Lim Eng Eng: "Processing of Abdominal Ultrasound Images Using Seed based Region Growing Method", Proceedings of International Conference on Intelligent Sensing and Information Processing, IEEE, 2004.
2. Hitesh Garg, Alka Jindal: "Segmentation of Thyroid gland in Ultrasound Images using neural network", Fourth International Conference on Computing, Communications and Networking Technologies, IEEE, 2013.
3. Ashnil kumar, Pradeeba sridar, Ann Quinton, R. Krishanan Kumar, Dagan Feng, Ralph Nanan, Jinman Kim : "Plane Identification in Fetal Ultrasound Images Using Saliency Maps and Convolutional Neural Networks" , 13th International Symposium on Biomedical Imaging (ISBI),IEEE,2016.
4. Gustavo Carneiro, Jacinto C. Nascimento, António Freitas: "The Segmentation of the Left Ventricle of the Heart From Ultrasound Data Using Deep Learning Architectures and Derivative-Based Search Methods", IEEE, 2012.
5. Olaf Ronneberger, Philipp Fischer, and Thomas Brox: "U-Net: Convolutional Networks for Biomedical Image Segmentation" MICCAI, Springer, 2015.

AUTHORS PROFILE

Deepak Sharma is currently pursuing a Ph.D. in Computer Science at M. D. University, Rohtak. He has completed his M.tech from C-DAC: Centre for Development of Advanced Computing, Ministry of Communications and Information Technology, Government of India affiliated from Guru Gobind Singh Indraprastha University, Delhi. His main research areas include Data mining, Mobile Adhoc Network (MANET), wireless sensor network (WSN) and Internet of things (IoT).

Lalit Kumar has successfully completed her master's degree in Computer Science and Engineering from C-DAC, Noida affiliated to Guru Gobind Singh Indraprastha University, New Delhi. His research dimension covers computer networking, image processing and network security.