Major Project Report

On

"ULTRASOUND NERVE SEGMENTATION USING DEEP LEARNING"

Submitted in partial fulfillment of the

Requirements for the award of the degree of

Bachelor of Technology

In

Computer Science & Engineering

By

P. Harika Reddy – 17R21A05G3
Md Nizamuddin– 17R21A05F2
Md Abdul Mujeeb – 17R21A05F1

Under the guidance of

R. Anusha
Assistant Professor

Department of Computer Science & Engineering

MARRI LAXMAN REDDY GROUP OF INSTITUTIONS
MLR INSTITUTE OF TECHNOLOGY
(UGC AUTONOMOUS)
Affiliated to JNTUH,Approved by AICTE
Laxman Reddy Avenue, Dundigal, Hyderabad-500 043, Telangana, India
NBA NATIONAL BOARD OF ACCREDITATION
A NAAC

2021

MARRI LAXMAN REDDY GROUP OF INSTITUTIONS
MLR INSTITUTE OF TECHNOLOGY
(UGC AUTONOMOUS)
Affiliated to JNTUH,Approved by AICTE
Laxman Reddy Avenue, Dundigal, Hyderabad-500 043, Telangana, India
NBA NATIONAL BOARD ACCREDITATION
NAAC

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the project entitled "**ULTRASOUND NERVE SEGMENTATION USING DEEP LEARNING"** has been submitted by **P. Harika Reddy (17R21A05G3), Md Nizamuddin (17R21A05F2), Md Abdul Mujeeb (17R21A05F1)** in partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering from MLR Institute of Technology, Hyderabad. The results embodied in this project have not been submitted to any other University or Institution for the award of any degree or diploma.

Internal Guide **Head of the Department**

External Examiner

MARRI LAXMAN REDDY GROUP OF INSTITUTIONS
MLR INSTITUTE OF TECHNOLOGY
(UGC AUTONOMOUS)
Affiliated to JNTUH,Approved by AICTE
Laxman Reddy Avenue, Dundigal, Hyderabad-500 043, Telangana, India
NBA NATIONAL BOARD ACCREDITATION
NAAC

Department of Computer Science & Engineering

DECLARATION

We hereby declare that the project entitled **"ULTRASOUND NERVE SEGMENTATION USING DEEP LEARNING"** is the work done during the period from **March 2021 to June 2021** and is submitted in partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering from MLR Institute of Technology, Hyderabad. The results embodied in this project have not been submitted to any other university or Institution for the award of any degree or diploma.

P. Harika Reddy	**17R21A05G3**
Md Nizamuddin	**17R21A05F2**
Md Abdul Mujeeb	**17R21A05F1**

Department of Computer Science & Engineering

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned our efforts with success. It is a pleasant aspect that I now have the opportunity to express my guidance for all of them.

First of all We would like to express my deep gratitude towards my internal guide **R. Anusha, Assistant Professor, Department of CSE** for her support in the completion of our dissertation. We wish to express our sincere thanks to **Dr. N. CHANDRASHEKHAR REDDY,** HOD, Dept. of CSE and also principal **Dr. K. SRINIVASA RAO** for providing the facilities to complete the dissertation.

We would like to thank all my faculty and friends for their help and constructive criticism during the project period. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve goals.

P. Harika Reddy	**17R21A05G3**
Md Nizamuddin	**17R21A05F2**
Md Abdul Mujeeb	**17R21A05F1**

Department of Computer Science & Engineering

ABSTRACT

Today medical field has provided us enormous facilities that has been never thought before. There has been done many improvements in the field of surgery, medicine, X-Rays and many more. But some areas still want some improvements so that patients don't need to face any type of difficulty or pain. We are trying to highlight the difficulties in the treatment that are based on the ultrasound images. So in this paper our main focus is to improve the treatments based on ultrasound scans which is used widely in medical field due to vast area of application and cost effectiveness. These ultrasound scans are very important to detect any kind of injury of disease in human body because it used to scan the internal tissues of the body. 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.

LIST OF FIGURES

INDEX

Chapter 1

Introduction

1.1 Overview

Surgery oftentimes involves post-surgical pain. Patient fears or back out from the surgical procedure by mentioning. Surgery surely can bring 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 it’s 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.

1.2 Purpose of the project

This is to improve pain management through the use of indwelling catheters that block or mitigate pain at the source. Pain management catheters reduce dependence on narcotics and speed up patient recovery. Accurately identifying nerve structures in ultrasound images is a critical step in effectively inserting a patient's pain management catheter.

In this project, we are challenged to build a model that can identify nerve structures in a dataset of ultrasound images of the neck. Doing so would improve catheter placement and contribute to a more pain free future.

1.3 Motivation

Brachial plexus is the main sensory and motor nerve of the upper limb. Blocking brachial plexus can mitigate the pain for the surgery of upper limb. Ultrasound is a noninvasive and real-time imaging technology which is widely used to guide the process of brachial plexus block. Accurately segmenting brachial plexus in ultrasound images is a critical step in effectively inserting a patient's pain management catheter. Manual segmentation of brachial plexus is time-consuming and highly variable. Doctors are desperate for auto-segmentation to save the time and reduce variation. However, auto-segmentation of ultrasound is extremely difficult for a number of reasons including low image quality, anatomically inadequate images, edge blur and indistinguishable characterization.

This project will ease the work of the doctor in finding the brachial plexus nerve of the patient and medicate the patient without any side effects.

Chapter 2

Literature Survey

2.1 Existing System

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.

A system for processing of "Abdominal Ultrasound images based on Region growing method" is proposed by Venktachalam.PA. An ultrasound processing model for showing the affected areas in abdomen by implementing histogram equalization and region growing method is the main objective.[2]

Gustavo Carneiro et al 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.[3]

2.2 Disadvantages of Existing System

1. Noise present in the ultrasound images lower the accuracy of the model.

2. Low accuracy of the model doesn't predict correct location of nerve.

3.Takes more time in predicting the location of nerve in the ultrasound images.

Chapter 3
Proposed System

3.1 Proposed System

In proposed system, as data of the system we are using “Ultrasound nerve images” dataset. Pre-processed data is present in the dataset, which means that the noisy data is removed. Along with the noisy data, the blur ultrasound images are also removed from the ultrasound image dataset to reduce the error in the output. Our main idea is to construct a Deep Neural Network for Biomedical Image Segmentation, in this case the ultrasound scan images of the nerves near the areas of the Neck. To Train the Neural Network with not the raw ultrasound nerve images available but to pre-process the data with the aim to reduce noise in the image in order to make it clearer for the Deep Neural Network to understand the ultrasound nerve image. To keep improving the Accuracy of the model by identifying potential parameters which are a bottleneck. To improve prediction time of the Neural Network by training efficient models that take significantly less time to predict when compared to conventional models.

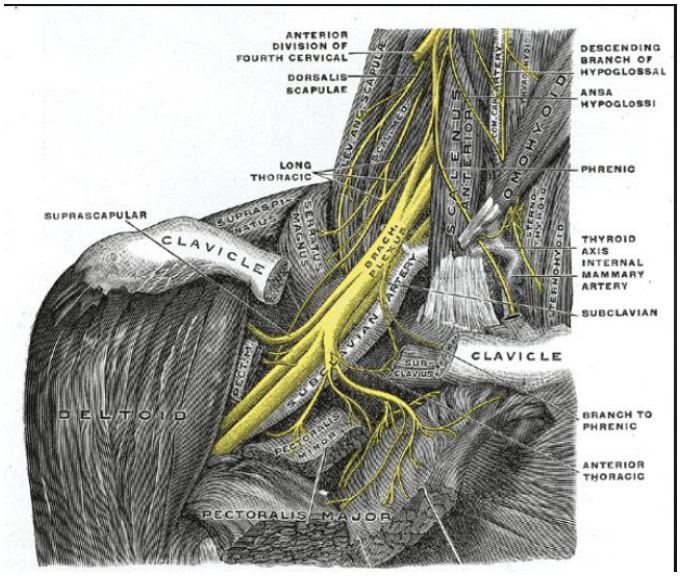


Fig 3.1.1 Brachial plexus nerve

The above picture represents the brachial plexus nerve in the neck part of the human body.. The network of nerves extends from the spinal cord, through the cervicoaxillary canal in the neck, over the first rib, and into the armpit. For body parts like chest, shoulder, arm, forearm and hand it supplies afferent and efferent nerve fibers.

3.2 Advantages of Proposed System

- Clear Images prone to give High accuracy with significantly less data but more information due to clarity in dataset
- Increase in Accuracy due to Image Noise reduction and pre-processing of the ultrasound images of nerves
- Fast prediction due to availability of clear data for training the model for prediction.

3.3 System Requirements

➢ **Software Requirements**

1. Python 3 or above (with IDE)
2. Pip to install python packages
3. Python libraries numpy, scikit images, keras, tensorflow
4. Ultrasound Nerve Images Dataset

➢ **Hardware Requirements**

1. Minimum 4GB RAM (8GB recommended)
2. Windows/UNIX based OS

Chapter 4

System Design

4.1 Proposed System Architecture

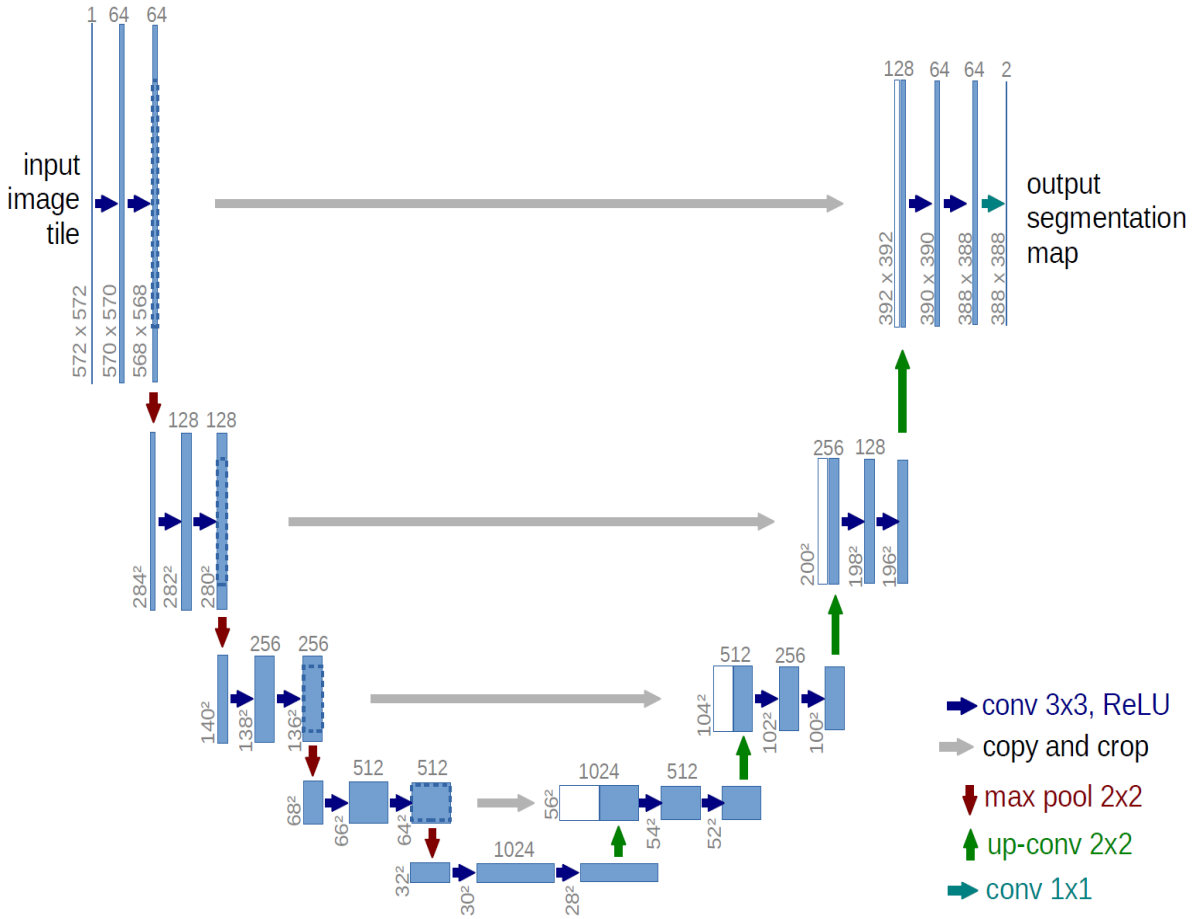


Fig 4.1.1 Architecture

The architecture of the project is U-Net Convolutional Networks for Biomedical Image Segmentation. It is fully convolutional. It is also called skip connection of Encoder-decoder architecture on the same level. There will be 3x3 grid vectors displacement with smooth deformations. Gaussian distribution is used for sampling the distribution [4]. We will train the system with the help of the results that we get from the nerve segmentation. All the data that we get as the result of the nerve segmentation will be loaded into the machine learning algorithm

4.2 Modules

Based on the methodology, we divided the project into 3 modules. They are

1) Preprocessing module

2) Model Training module

3) Prediction module

1. **Pre Processing**

The term preprocessing itself means processing the data before performing actions or training. The ultrasound images are generally noisy and it is very difficult to identify the brachial plexus nerve.[5] Hence, preprocessing is one of the main module in which the noisy data, duplicate data and unwanted data is removed from the ultrasound images. Before sending as training data for training the model process, the data is properly up-sampled. To stretch the low resolution picture, “Unpooling” is used and to up-sample the Image to a Higher resolution for more clarity “Convolution” is used.

Unpooling

Unpooling is the first type of upsampling which takes the idea of pooling[6]. If we consider the below example, the 4x4 is minimized into 2x2 with highest number within the colored boxes, that method of processing is called pooling. Maximizing the 2x2 into 4x4 grids with highest number occupying the same place and rest of the boxes are filled with zeros.

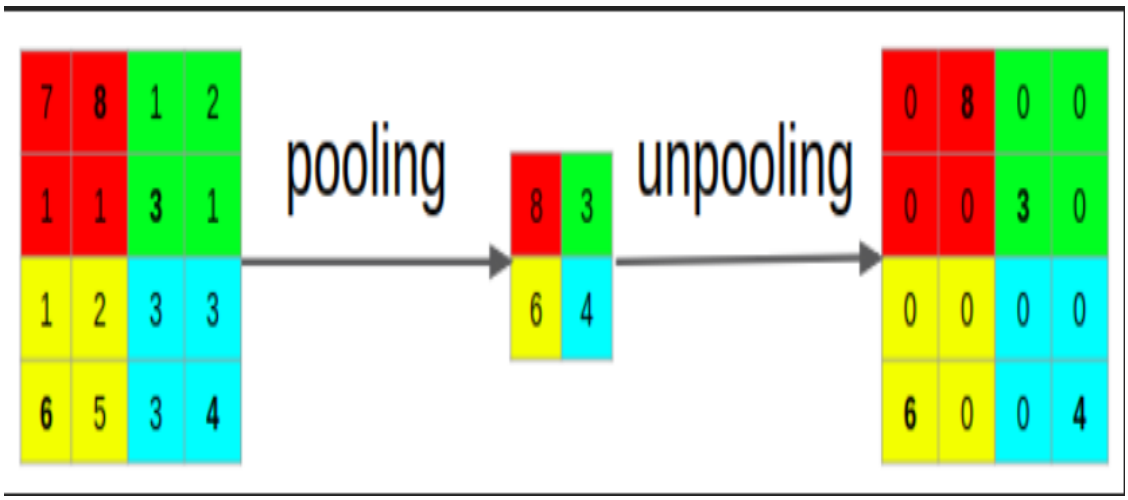


Fig 4.2.1 Pooling & Unpooling

Convolution

Convolution can be explained easily by the below picture. Minimizing the image within a maximizing part of the actual image is called convolution. Deconvolution means maximizing the image of the minimized part of the actual image[7].

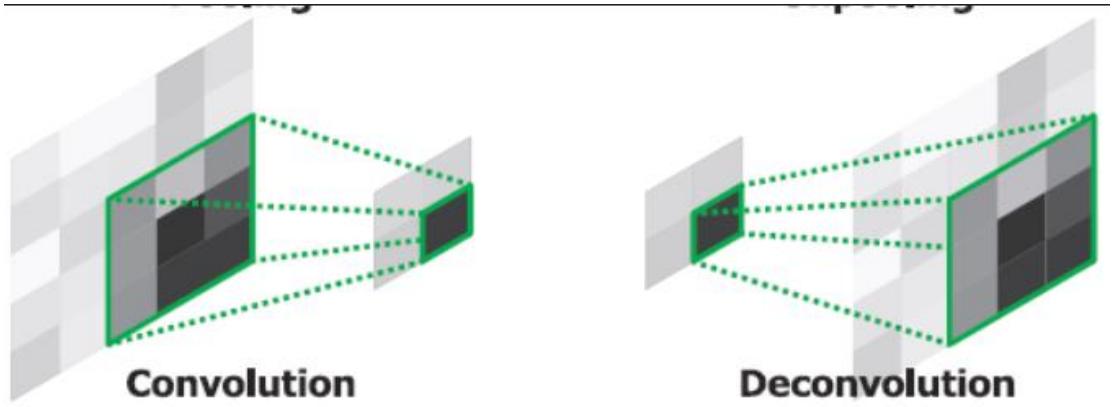


Fig 4.2.2 Convolution

2. Model Training

The dataset taken is “Ultrasound Nerve Segmentation Images”. There are two different type of images within the dataset. The first one is ultrasound images and the other one is black and white image. That white dark shade indicates the brachial plexus nerve.

3. Prediction

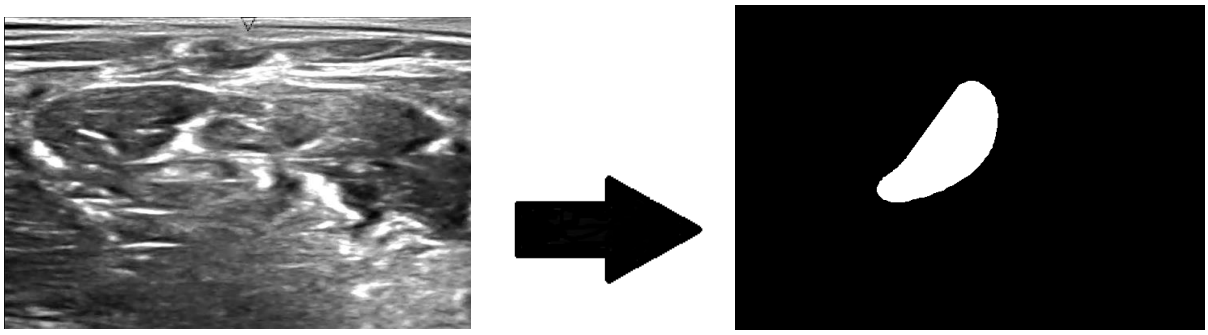


Fig 4.2.3 Predicting nerve location

As shown in the above picture. The input data of ultrasound image around the neck part is taken, this results in predicting the Brachial plexus nerve by having 80% of the training data and 20% of the testing data. The white shaded region indicates the area of brachial plexus nerve of the given input ultrasound image.

4.3 UML Diagrams

a) Activity Diagram

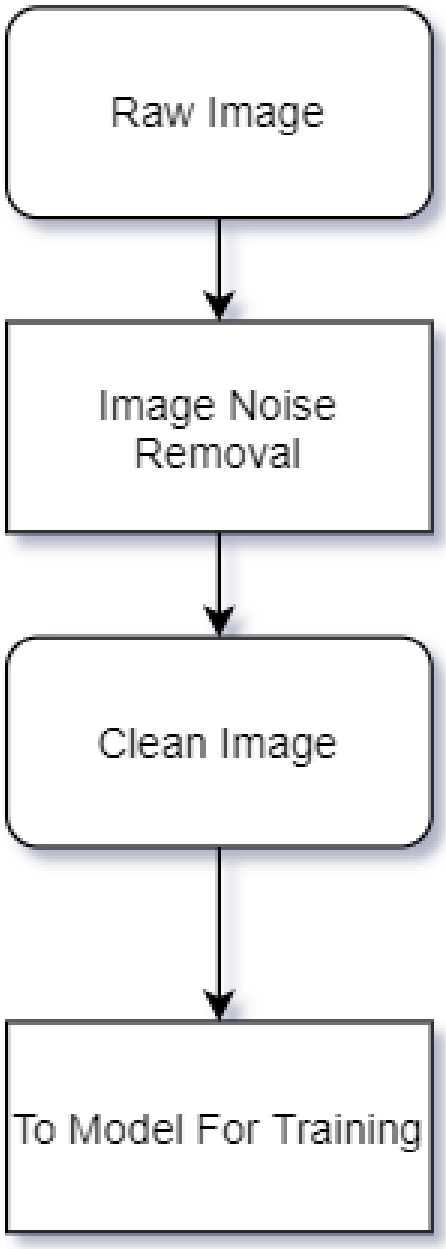


Fig 4.3.1 Preprocessing

b) Activity Diagram

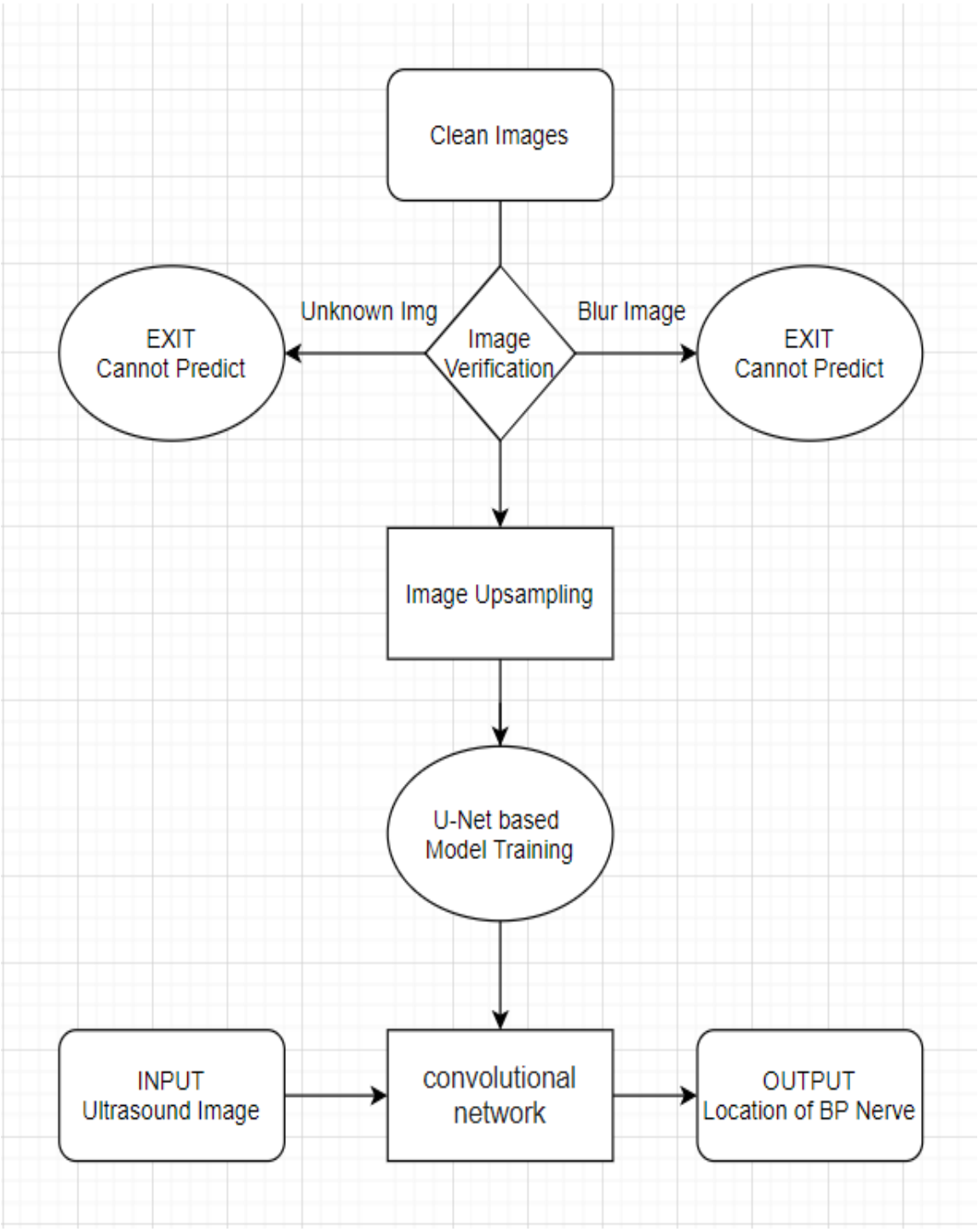
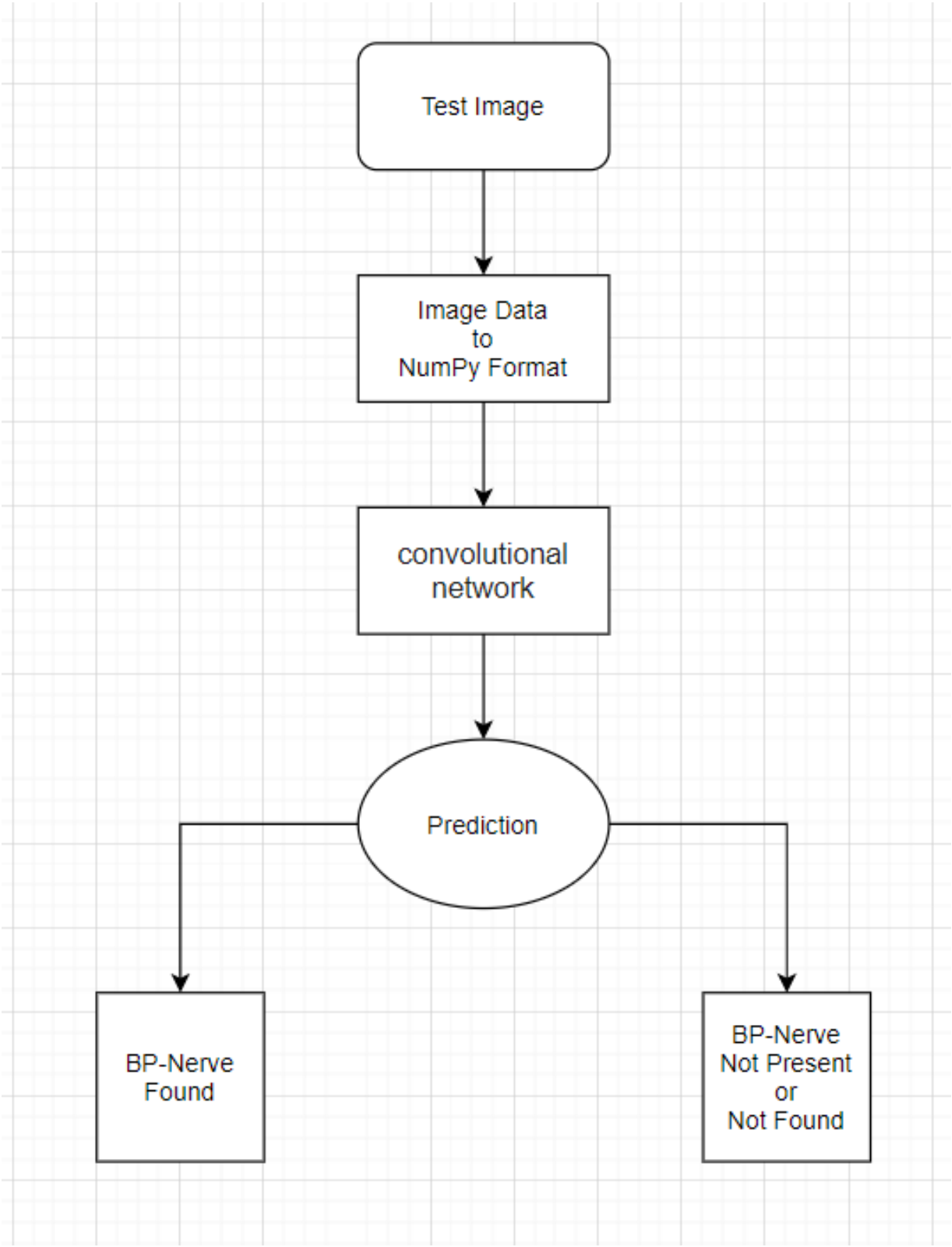


Fig 4.3.2 Model Training

c) Activity Diagram



4.3.3 Prediction

Chapter 5

Implementation

5.1 Algorithms

Algorithm	Use-case
U-Net Convolutional Network	Architecture
Image Segmentation	Image -> Understandable Format
Run-Length Encoding	To convert to binary format
Convolutional 2D Layer Transpose	Training Deep Learning Model

Fig 5.1.1 Algorithms used

5.2 Implementation Steps

- The idea is to construct a Deep Neural Network for Biomedical Image Segmentation, in this case the ultrasound images of the nerves near the areas of the Neck.
- To Train the Neural Network with not the raw ultrasound nerve images available but to pre-process the data with the aim to reduce noise in the image in order to make it clearer for the Deep Neural Network to understand the ultrasound nerve image.
- To keep improving the Accuracy of the model by identifying potential parameters which are a bottleneck
- To improve prediction time of the Neural Network by training efficient models that take significantly less time to predict when compared to conventional models.
- Keras model is developed using core Tensor flow components
- Model is supplied with test data, the kind of Images it hasn't encountered before

- Clear Images, if they contain BP Nerve, an White spot on Black Background is shown which depicts the position of the BP nerve in the Ultrasound Image.

Fig 5.2.1 Ultrasound Images

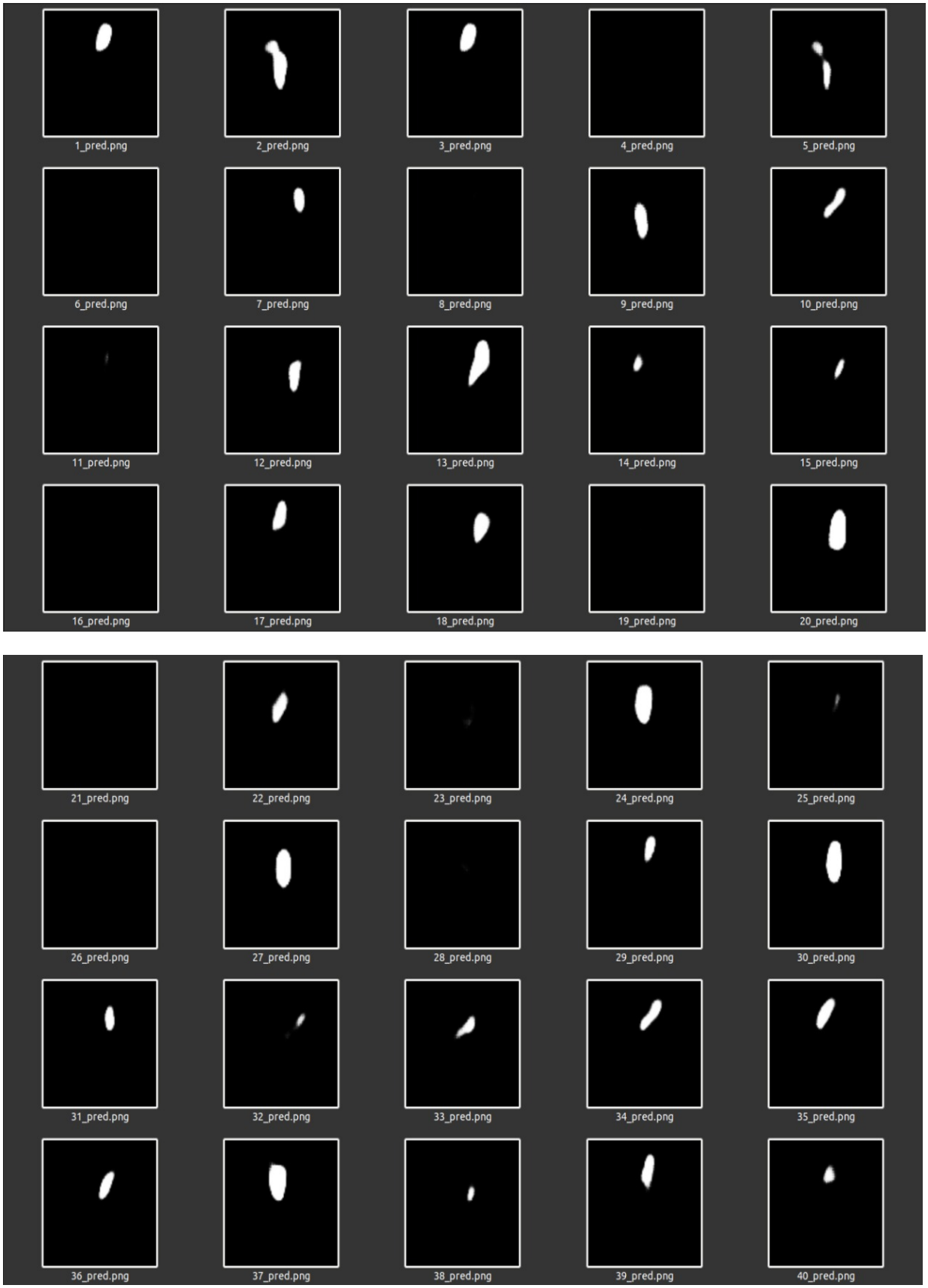


Fig 5.2.2 Predicted Nerve Location

5.3 Source Code

1. **Data.py:**

```
from __future__ import print_function

import os

import numpy as np

from skimage.io import imsave, imread

data_path = 'raw/'

image_rows = 420

image_cols = 580

def create_train_data():

train_data_path = os.path.join(data_path, 'train')

images = os.listdir(train_data_path)

total = len(images) // 2

imgs = np.ndarray((total, image_rows, image_cols), dtype=np.uint8)

imgs_mask = np.ndarray((total, image_rows, image_cols), dtype=np.uint8)

i = 0

print('-'*30)

print('Creating training images...')

print('-'*30)

for image_name in images:

if 'mask' in image_name:

continue

image_mask_name = image_name.split('.')[0] + '_mask.tif'

img = imread(os.path.join(train_data_path, image_name), as_gray=True)

img_mask = imread(os.path.join(train_data_path, image_mask_name), as_gray=True)

img = np.array([img])
```

```
img_mask = np.array([img_mask])

imgs[i] = img

imgs_mask[i] = img_mask

if i % 100 == 0:

print('Done: {0}/{1} images'.format(i, total))

i += 1

print('Loading done.')

np.save('imgs_train.npy', imgs)

np.save('imgs_mask_train.npy', imgs_mask)

print('Saving to .npy files done.')

def load_train_data():

imgs_train = np.load('imgs_train.npy')

imgs_mask_train = np.load('imgs_mask_train.npy')

return imgs_train, imgs_mask_train

def create_test_data():

train_data_path = os.path.join(data_path, 'test')

images = os.listdir(train_data_path)

total = len(images)

imgs = np.ndarray((total, image_rows, image_cols), dtype=np.uint8)

imgs_id = np.ndarray((total, ), dtype=np.int32)

i = 0

print('-'*30)

print('Creating test images...')

print('-'*30)

for image_name in images:

img_id = int(image_name.split('.')[0])
```

```
img = imread(os.path.join(train_data_path, image_name), as_gray=True)

img = np.array([img])

imgs[i] = img

imgs_id[i] = img_id

if i % 100 == 0:

print('Done: {0}/{1} images'.format(i, total))

i += 1

print('Loading done.')

np.save('imgs_test.npy', imgs)

np.save('imgs_id_test.npy', imgs_id)

print('Saving to .npy files done.')

def load_test_data():

imgs_test = np.load('imgs_test.npy')

imgs_id = np.load('imgs_id_test.npy')

return imgs_test, imgs_id

if __name__ == '__main__':

create_train_data()

create_test_data()
```

2. **Predict.py**

```
import sys

ind1 = sys.argv[1]

ind2 = sys.argv[2]

i1='data/test/' + str(ind1) + '.tif'

i2='preds/' + str(ind1) + '_pred.png'

i3='data/test/' + str(ind2) + '.tif'

i4='preds/' + str(ind2) + '_pred.png'
```

```
import matplotlib.pyplot as plt

import matplotlib.image as mpimg

import numpy as np

from PIL import Image

fig , ax = plt.subplots(2,2)

ax[0 , 0].set_title(i1)

ax[0 , 1].set_title(i2)

ax[1 , 0].set_title(i3)

ax[1 , 1].set_title(i4)

img1 = mpimg.imread(i1)

ax[0,0].imshow(img1)

img2 = mpimg.imread(i2)

ax[0,1].imshow(img2)

img1 = mpimg.imread(i3)

ax[1,0].imshow(img1)

img2 = mpimg.imread(i4)

ax[1,1].imshow(img2)

plt.show()
```

3. **Summary.py**

```
# load and evaluate a saved model

from __future__ import print_function

import os, pickle

from skimage.transform import resize

from skimage.io import imsave

import tensorflow as tf

import numpy as np
```

```
from keras.models import Model
from keras.layers import Input, concatenate, Conv2D, MaxPooling2D,
Conv2DTranspose
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint
from keras import backend as K
from data import load_train_data, load_test_data
from numpy import loadtxt
from keras.models import load_model
from data import load_train_data, load_test_data
import numpy as np
model = load_model('model.h5', compile=False)
model.summary()
```

4. **Train.py**

```
from __future__ import print_function
import os, pickle
from skimage.transform import resize
from skimage.io import imsave
import numpy as np
from keras.models import Model
from keras.layers import Input, concatenate, Conv2D, MaxPooling2D,
Conv2DTranspose
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint
from keras import backend as K
from data import load_train_data, load_test_data
```

```
K.set_image_data_format('channels_last')  # TF dimension ordering in this code
img_rows = 96
img_cols = 96
smooth = 1.
def dice_coef(y_true, y_pred):
y_true_f = K.flatten(y_true)
y_pred_f = K.flatten(y_pred)
intersection = K.sum(y_true_f * y_pred_f)
return (2. * intersection + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) + smooth)
def dice_coef_loss(y_true, y_pred):
return -dice_coef(y_true, y_pred)
def get_unet():
inputs = Input((img_rows, img_cols, 1))
conv1 = Conv2D(32, (3, 3), activation='relu', padding='same')(inputs)
conv1 = Conv2D(32, (3, 3), activation='relu', padding='same')(conv1)
pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
conv2 = Conv2D(64, (3, 3), activation='relu', padding='same')(pool1)
conv2 = Conv2D(64, (3, 3), activation='relu', padding='same')(conv2)
pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
conv3 = Conv2D(128, (3, 3), activation='relu', padding='same')(pool2)
conv3 = Conv2D(128, (3, 3), activation='relu', padding='same')(conv3)
pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)
conv4 = Conv2D(256, (3, 3), activation='relu', padding='same')(pool3)
conv4 = Conv2D(256, (3, 3), activation='relu', padding='same')(conv4)
pool4 = MaxPooling2D(pool_size=(2, 2))(conv4)
conv5 = Conv2D(512, (3, 3), activation='relu', padding='same')(pool4)
```

```
conv5 = Conv2D(512, (3, 3), activation='relu', padding='same')(conv5)
up6 = concatenate([Conv2DTranspose(256, (2, 2), strides=(2, 2),
padding='same')(conv5), conv4], axis=3)
conv6 = Conv2D(256, (3, 3), activation='relu', padding='same')(up6)
conv6 = Conv2D(256, (3, 3), activation='relu', padding='same')(conv6)
up7 = concatenate([Conv2DTranspose(128, (2, 2), strides=(2, 2),
padding='same')(conv6), conv3], axis=3)
conv7 = Conv2D(128, (3, 3), activation='relu', padding='same')(up7)
conv7 = Conv2D(128, (3, 3), activation='relu', padding='same')(conv7)
up8 = concatenate([Conv2DTranspose(64, (2, 2), strides=(2, 2),
padding='same')(conv7), conv2], axis=3)
conv8 = Conv2D(64, (3, 3), activation='relu', padding='same')(up8)
conv8 = Conv2D(64, (3, 3), activation='relu', padding='same')(conv8)
up9 = concatenate([Conv2DTranspose(32, (2, 2), strides=(2, 2),
padding='same')(conv8), conv1], axis=3)
conv9 = Conv2D(32, (3, 3), activation='relu', padding='same')(up9)
conv9 = Conv2D(32, (3, 3), activation='relu', padding='same')(conv9)
conv10 = Conv2D(1, (1, 1), activation='sigmoid')(conv9)
model = Model(inputs=[inputs], outputs=[conv10])
model.compile(optimizer=Adam(lr=1e-5), loss=dice_coef_loss,
metrics=['accuracy'])
return model
def preprocess(imgs):
imgs_p = np.ndarray((imgs.shape[0], img_rows, img_cols), dtype=np.uint8)
for i in range(imgs.shape[0]):
imgs_p[i] = resize(imgs[i], (img_cols, img_rows), preserve_range=True)
```

```
imgs_p = imgs_p[..., np.newaxis]

return imgs_p

def train_and_predict():

print('-'*30)

print('Loading and preprocessing train data...')

print('-'*30)

imgs_train, imgs_mask_train = load_train_data()

imgs_train = preprocess(imgs_train)

imgs_mask_train = preprocess(imgs_mask_train)

imgs_train = imgs_train.astype('float32')

mean = np.mean(imgs_train)  # mean for data centering

std = np.std(imgs_train)  # std for data normalization

imgs_train -= mean

imgs_train /= std

imgs_mask_train = imgs_mask_train.astype('float32')

imgs_mask_train /= 255.  # scale masks to [0, 1]

print('-'*30)

print('Creating and compiling model...')

print('-'*30)

model = get_unet()

model_checkpoint = ModelCheckpoint('weights.h5', monitor='val_loss',

save_best_only=True)

print('-'*30)

print('Fitting model...')

print('-'*30)
```

```
model.fit(imgs_train, imgs_mask_train, batch_size=32, epochs=20, verbose=1,
shuffle=True,
validation_split=0.2,
callbacks=[model_checkpoint])
print('-'*30)
print('Loading and preprocessing test data...')
print('-'*30)
imgs_test, imgs_id_test = load_test_data()
imgs_test = preprocess(imgs_test)
imgs_test = imgs_test.astype('float32')
imgs_test -= mean
imgs_test /= std
print('-'*30)
print('Loading saved weights...')
print('-'*30)
model.load_weights('weights.h5')
print('-'*30)
print('Predicting masks on test data...')
print('-'*30)
imgs_mask_test = model.predict(imgs_test, verbose=1)
np.save('imgs_mask_test.npy', imgs_mask_test)
print('-' * 30)
print('Saving predicted masks to files...')
print('-' * 30)
pred_dir = 'preds'
if not os.path.exists(pred_dir):
```

```
os.mkdir(pred_dir)
for image, image_id in zip(imgs_mask_test, imgs_id_test):
image = (image[:, :, 0] * 255.).astype(np.uint8)
imsave(os.path.join(pred_dir, str(image_id) + '_pred.png'), image)
model.save("model.h5")
if __name__ == '__main__':
train_and_predict()
```

Chapter 6

Testing

The dataset is having 5000+ ultrasound images of the patients neck. The dataset is divided into 80% training data and 20% testing data which means 4000+ ultrasound images for the training purpose and the 1000+ ultrasound images for the testing purpose. The training data contains original image and also the masked image which shows the location of the brachial plexus nerve in the ultrasound image. But in the testing data only the ultrasound image is given to the model to which it should predict the location of the nerve in the given ultrasound image.

Fig 6.1 Training data

Fig 6.2 Testing data

Chapter 7

Result

Our project successfully executed and predicted the Brachial plexus nerve from the ultrasound image of 98% accuracy. The Project describes

1. **Accuracy**

```
> Epoch 20/20
141/141 [==============================] - 1232s 9s/step - loss: -0.5779 -
    accuracy:
0.9789 - val_loss: -0.4044 - val_accuracy: 0.9780
```

Fig 7.1 Accuracy of model

2. **Nerve predicted through the Ultrasound Images**

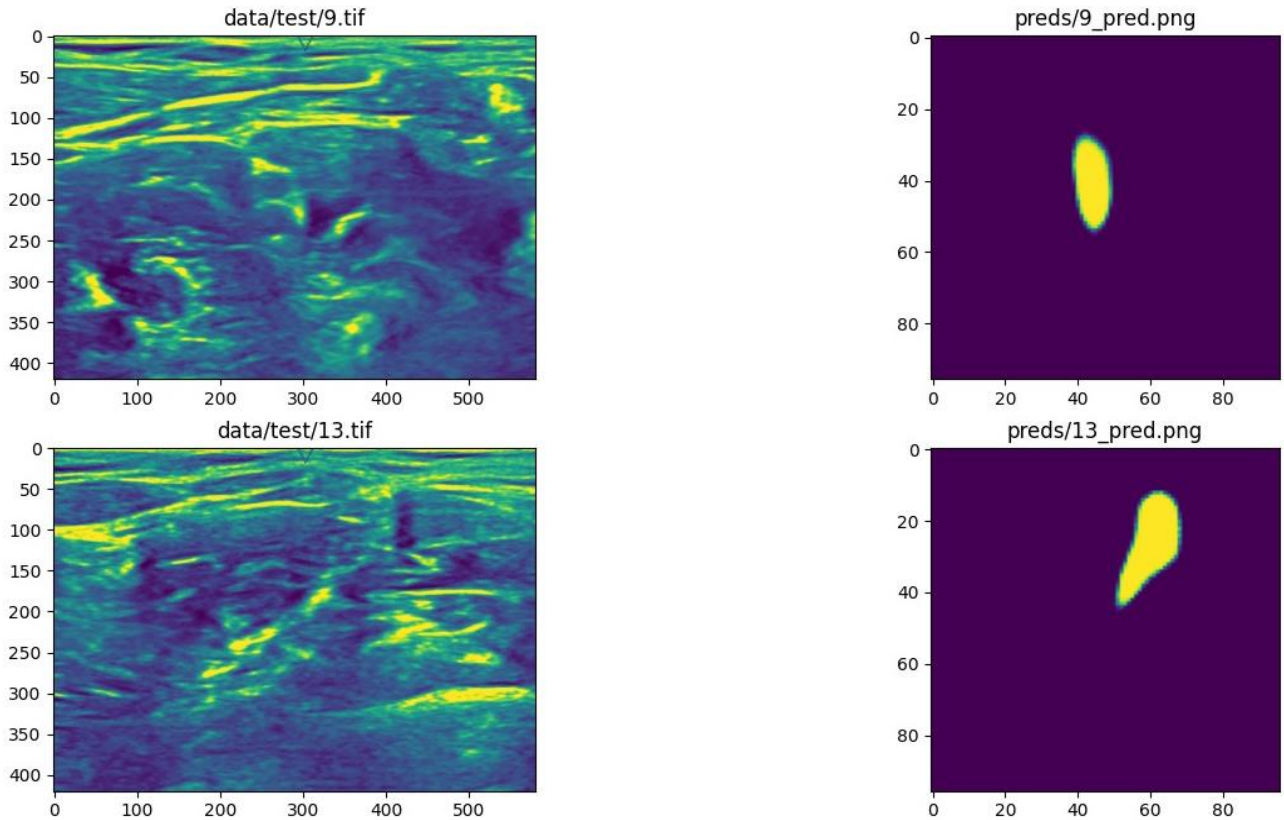


Fig 7.2 Predicted Nerve Location

Chapter 8

Conclusion

One major disadvantage of these images is that they include huge amount of noise so doctors face difficulty in finding the exact location of the nerve where they have to inject the medicine to operate. These pictures are not clear enough to find the nerve at once so they have to inject needle very times. With this application they can find the nerve very easily because it includes the segmentation of these nerves in ultrasound images.

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. This application is further extended to train the system with this data so that it can be used worldwide.

Chapter 9

Future Enhancement

Some of the drawbacks can be resolved in the future to make the model more accurate and efficient.

- 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.

Chapter 10

References

[1] https://www.kaggle.com/c/ultrasound-nerve-segmentation

[2] 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.

[3] Efficient Detection of Brachial Plexus in Ultrasound Images using Machine Learning Algorithms Deepak Sharma, Lalit kumar

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox: "U-Net: Convolutional Networks for Biomedical Image Segmentation" MICCAI, 2015.

[5] https://en.wikipedia.org/wiki/Brachial_plexus

[6] https://medium.com/jun-devpblog/dl-12-unsampling-unpooling-and-transpose-convolution-831dc53687ce

[7] https://www.tutorialspoint.com/dip/concept_of_convolution.htm

[8] https://byjus.com/physics/accuracy-precision-measurement/

[9] https://medium.com/analytics-vidhya/precision-and-recall-in-machine-learning-c8a1b9638eeb

[10]TeemuKanstrén,https://towardsdatascience.com/a-look-at-precisionrecall-and-f1-score-36b5fd0d

[11]International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-9 Issue-1, October 2019

[12] https://www.ijeat.org/wp-content/uploads/papers/v9i1/A1612109119.pdf