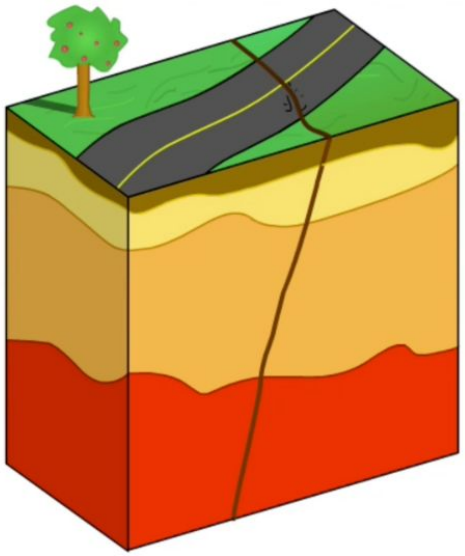
Information Systems Project

Fracture Detection From Seismic Images



Team 28

-Mohammed Fayiz Parappan

-Sneha Kumari

-Ronit Wanare

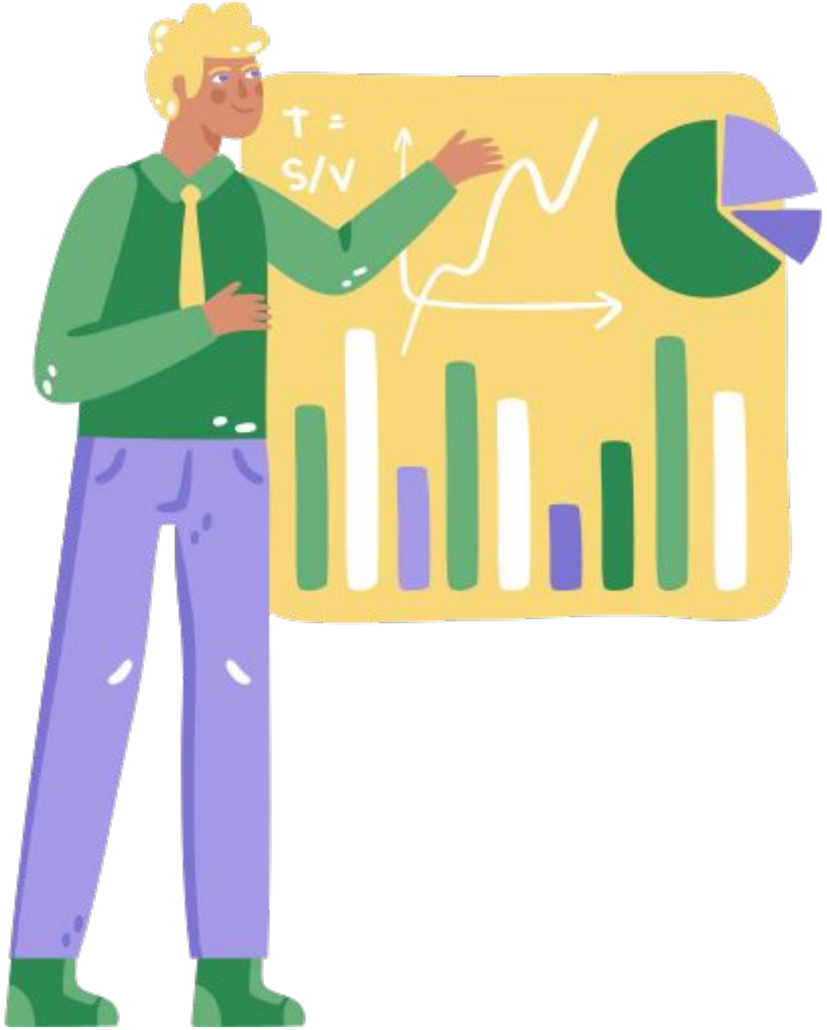
Table of Contents

Project Overview

Project Title: Fracture Detection from Seismic images

Problem Statement:

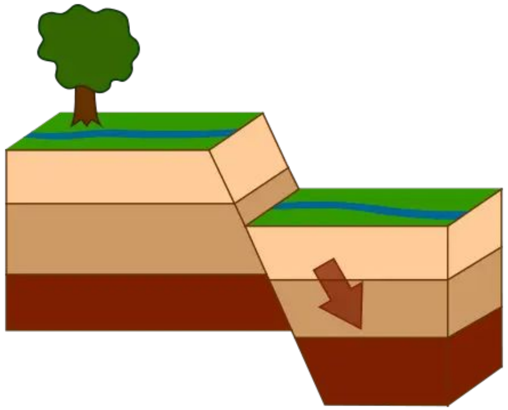
- Fracture/Fault detection from seismic images is an integral task both for geoscience research and industries involved in oil extraction.
- Present detection technique involve tremendous task of printing inline seismic data on large sheets and raw observation across pixel columns on the sheet.
- Large number of workforce is required for faster detection.
- High Human error



This results in significant cost of resources and time to identity faults. Present development in the field of deep learning and image segmentation have opened possibilities of segmenting fault layers.

System Description

A model to detect and present fault/fracture layer using deep learning techniques



Pre-Mid Sem Milestones

The Following are the milestones which we achieved by the end of mid-sem:-

- Performing **Transfer learning** on the prefitted U-net model(fault sec 3D) on synthetic seismic image dataset to extract features out of our model and to increase model accuracy.

- Designing and framing **custom Unet model** as per user Input annotations, such as taking filter size, no.of Convolution layers and filter number as Inputs from the user.

- Designing and framing GUI for **Es based coherence algorith**

Post-Mid Sem Milestones

The Following are the milestones which we have achieved post midesem:-

- Designing of 3 **UI** pages for different algorithms of Fault detection and custom training model:

  - Train 3D net using custom Unet model
  - Deep Learning based 2D fault Detection
  - Deep Learning based 3D fault Detection

- **Integration** of custom Unet model and 3D fault detection supporting real-time access to user to operate fault sec 3D.

- Transfer Learning For Fault Detection on **Real Seismic image (K-G Basin)**.

Post-Mid Sem Milestones

The Following are the milestones which we have achieved post midesem:-

- Minute changes suggested by ONGC:
  - In the 2D result view tab, the result view could be in the right sub-plot and the seismic volume could be in the left subplot.
  - In the 2D result view tab, when a subplot is being zoomed, the other sub-plot is not zooming in the exact same way.
  - Visualization menu could be shifted to the right of import

Post-Mid Sem Milestones

The Following are the milestones which we have achieved post midesem:-

- Implemented “Save File” option to save files after fault segmentation is generated.
- Denoising the Seismic data using LRTV algorithm
- Integrating denoising feature in GUI to allow user to inspect denoised block of image as per provided dimensions in real time through plots.
- Implementing small changes in the Train 3D Unet page1 - removal black screen, resizing whole page.
- Code Documentation

Languages/Libraries used-

-Python (via Jupyter Notebook)
-Tensorflow (for deep learning model)
-Segyio (for handling standard seismic data)
-Matplotlib (to plot results)
-Numpy

Numerous Libraries are used in the UI design code files such as tqdm, segpy, ray, sklearn, gurobipy and many more...

Software:-

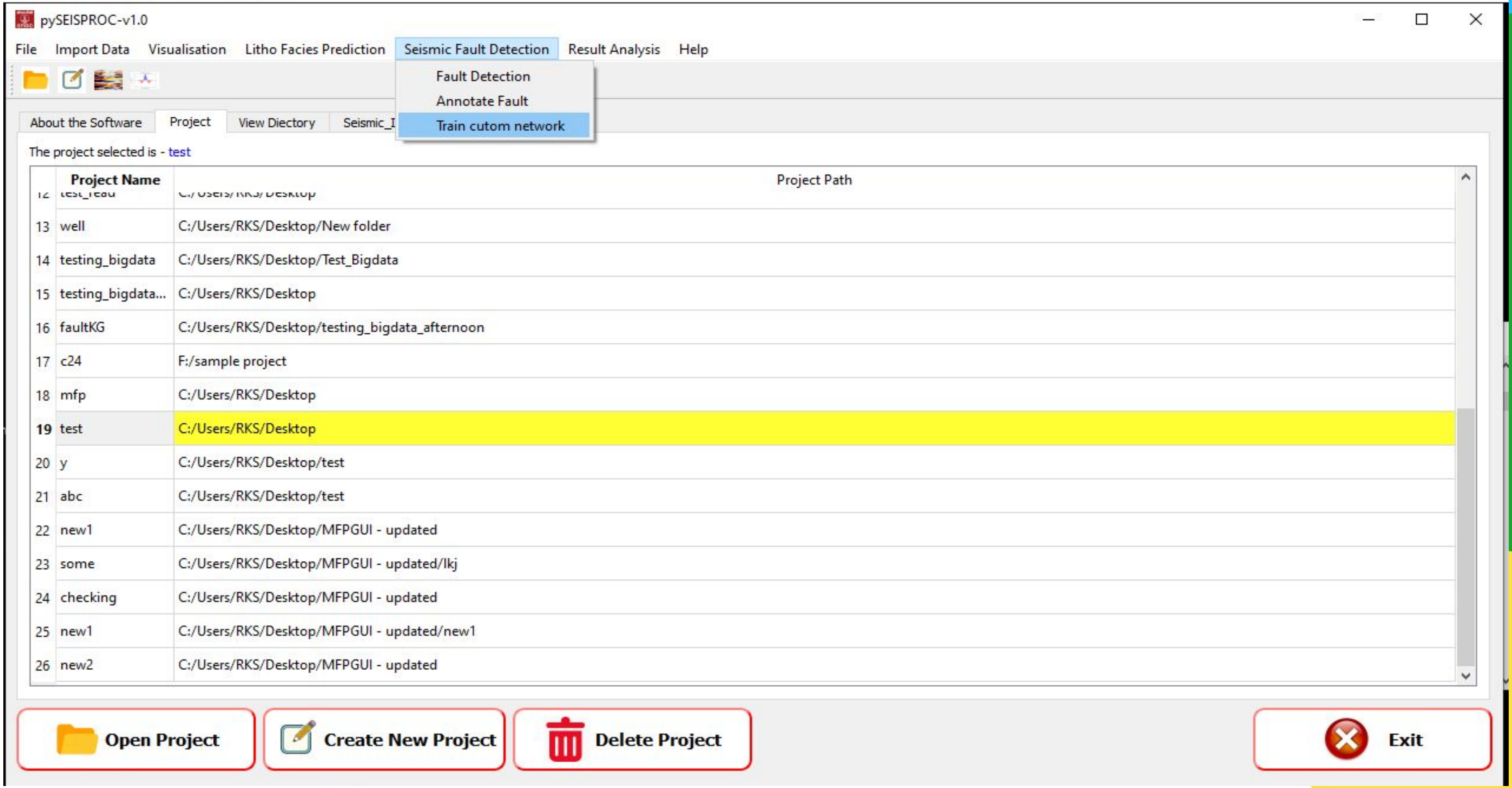
-Qt Designer (for UI model)
-Spyder(for ide purpose to edit code)
-Notepad++(for ide purpose to edit code)
-Cuda compilation tools, release 11.2, V11.2.67
Build cuda_11.2.r11.2/compiler.29373293_0

Operating System:-

-Windows

WindowsOS

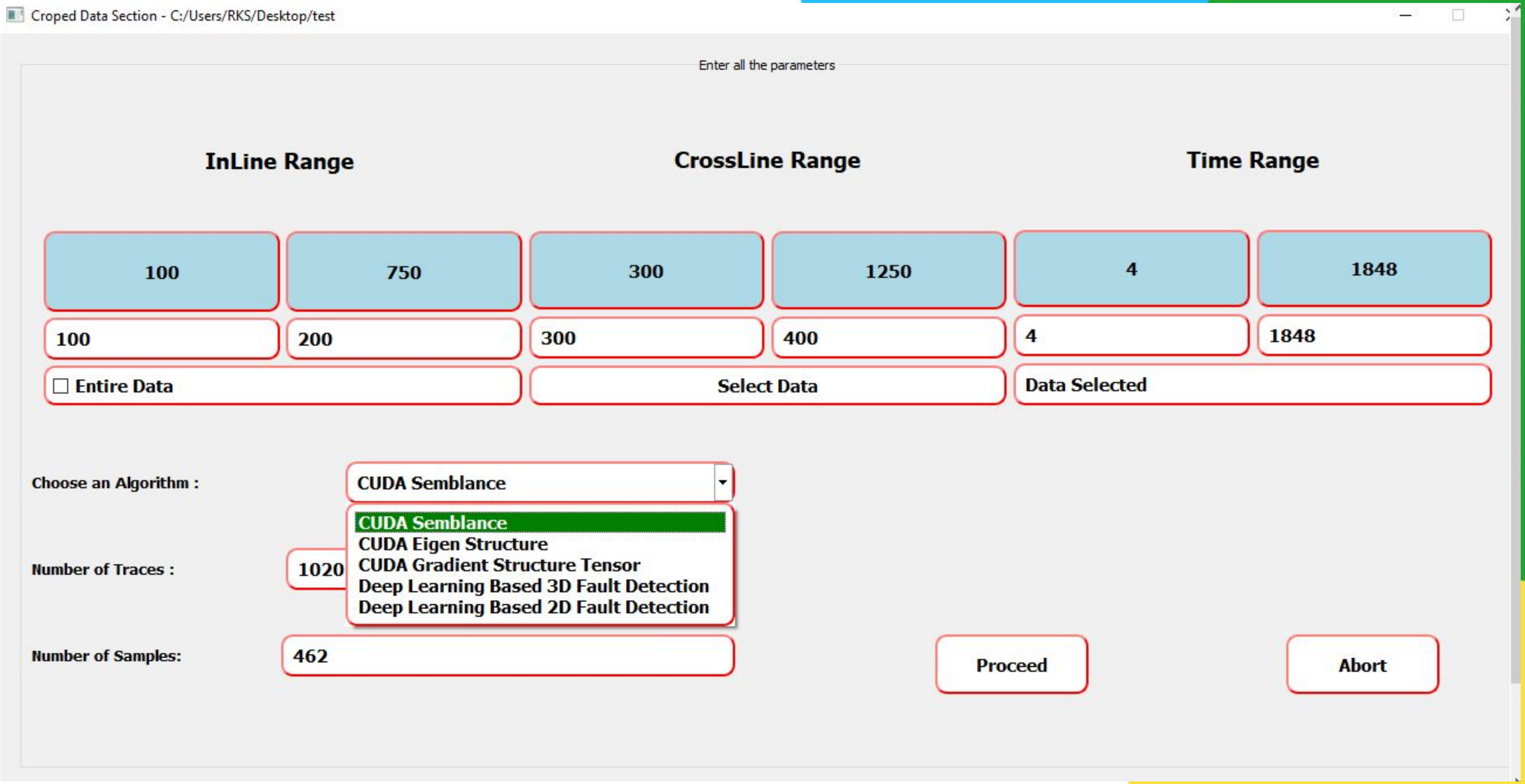
MAIN PAGE


pySEISPROC-v1.0
File
Import Data
Visualisation
Litho Facies Prediction
Seismic Fault Detection
Result Analysis
Help
Fault Detection
Annotate Fault
Train cutom network
About the Software
Project
View Diectory
Seismic_I
The project selected is - test
Project Name
Project Path
13 well C:/Users/RKS/Desktop/New folder
14 testing_bigdata C:/Users/RKS/Desktop/Test_Bigdata
15 testing_bigdata... C:/Users/RKS/Desktop
16 faultKG C:/Users/RKS/Desktop/testing_bigdata_afternoon
17 c24 F:/sample project
18 mfp C:/Users/RKS/Desktop
19 test C:/Users/RKS/Desktop
20 y C:/Users/RKS/Desktop/test
21 abc C:/Users/RKS/Desktop/test
22 new1 C:/Users/RKS/Desktop/MFPGUI - updated
23 some C:/Users/RKS/Desktop/MFPGUI - updated/lkj
24 checking C:/Users/RKS/Desktop/MFPGUI - updated
25 new1 C:/Users/RKS/Desktop/MFPGUI - updated/new1
26 new2 C:/Users/RKS/Desktop/MFPGUI - updated
Open Project
Create New Project
Delete Project
Exit

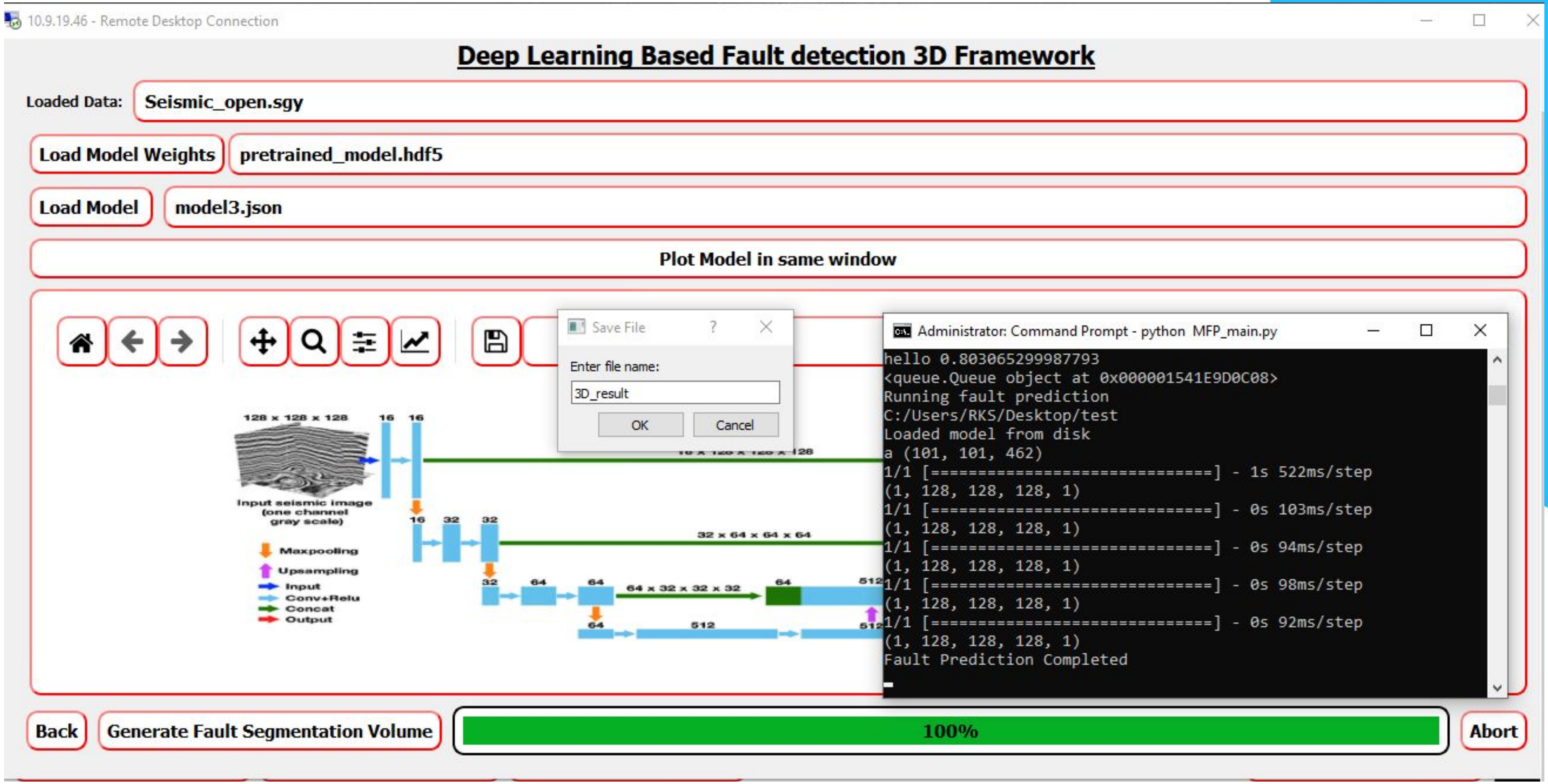
Croped_Data.UI

Page 1 -

Added new items in dropdown menu for Algorithm Selection and integrated it with the new page for Eigen Structure algorithm, Deep Learning based 2D and 3D fault detection pages.

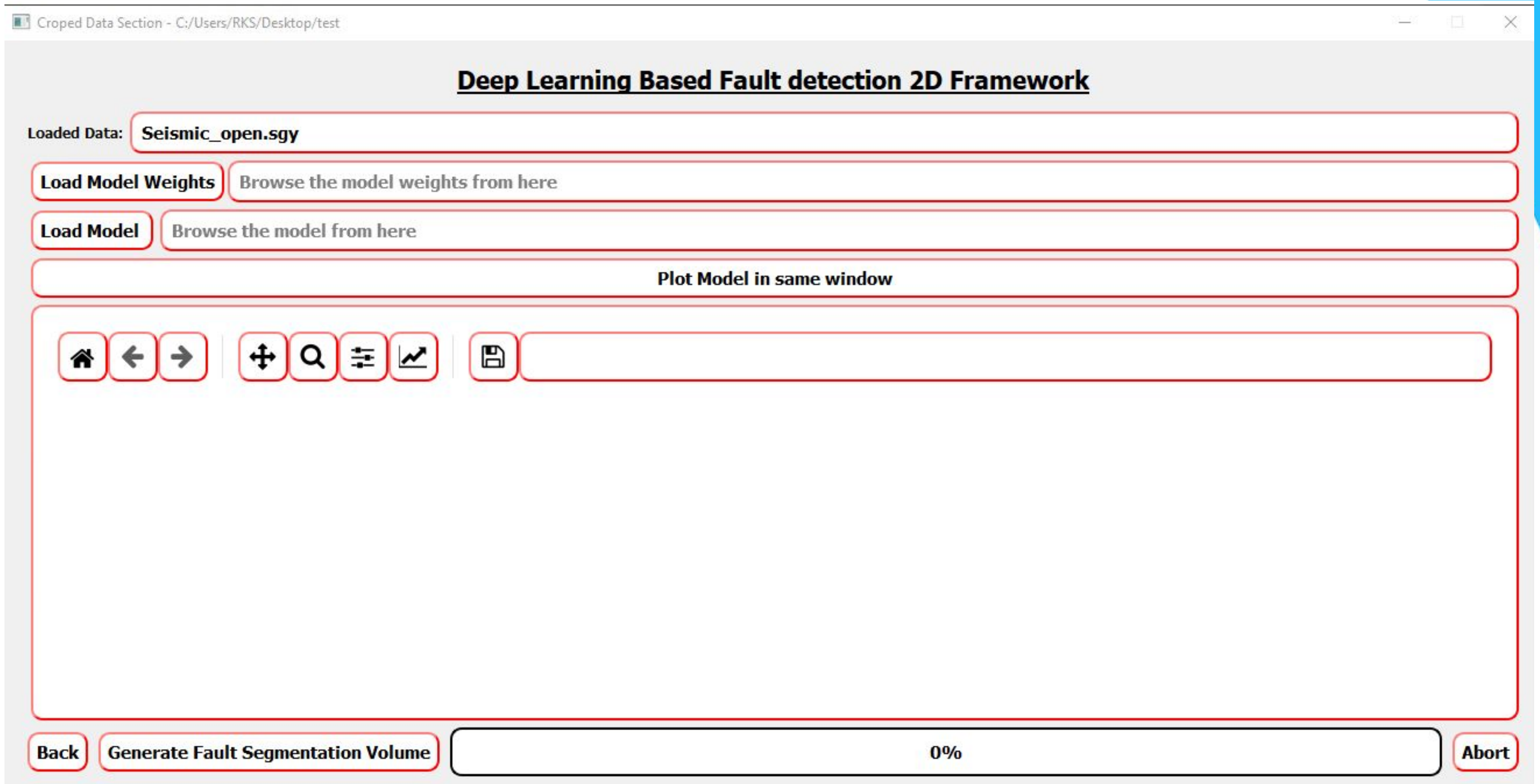


Deep Learning based 3D fault Detection

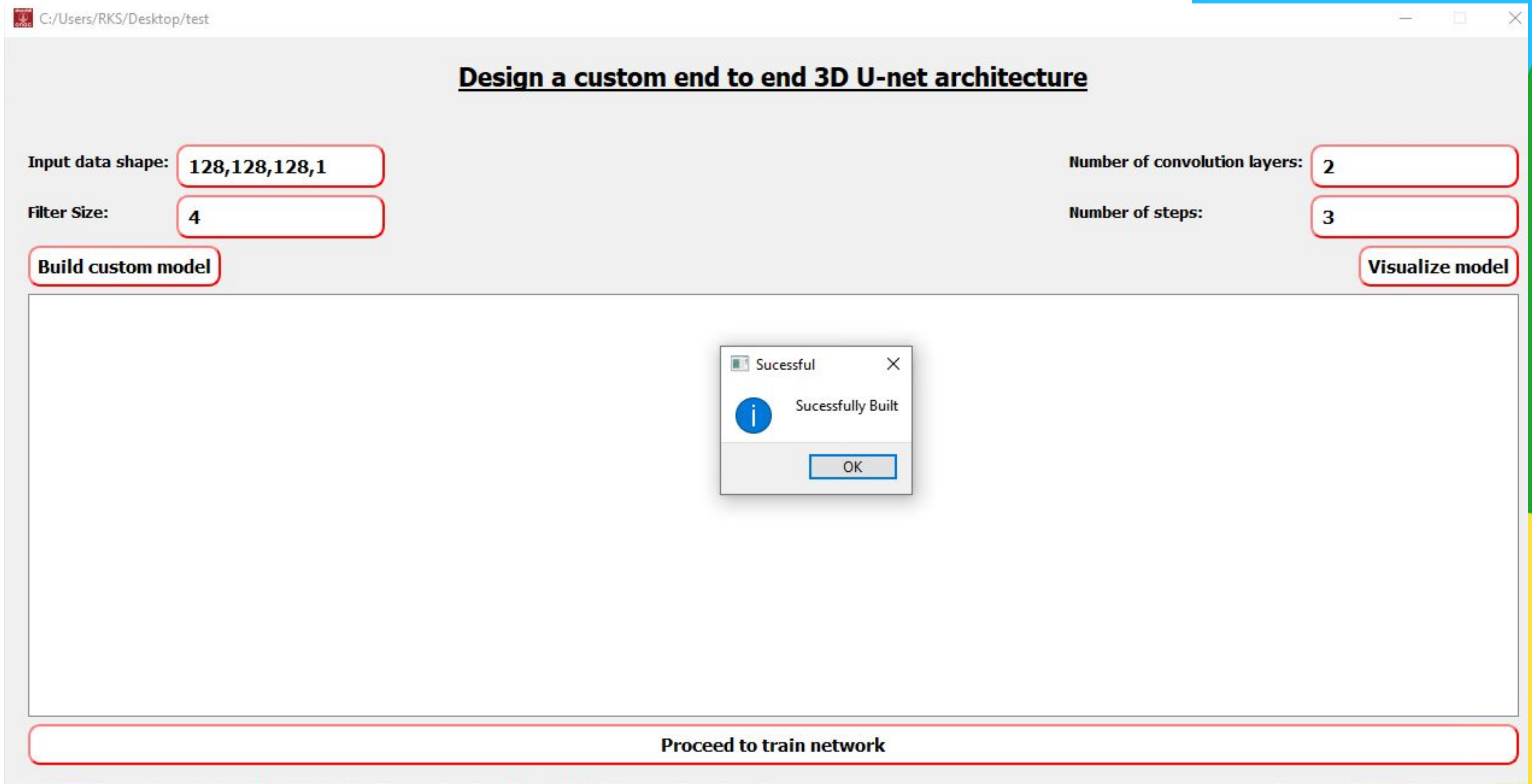


Deep Learning based 2D fault Detection

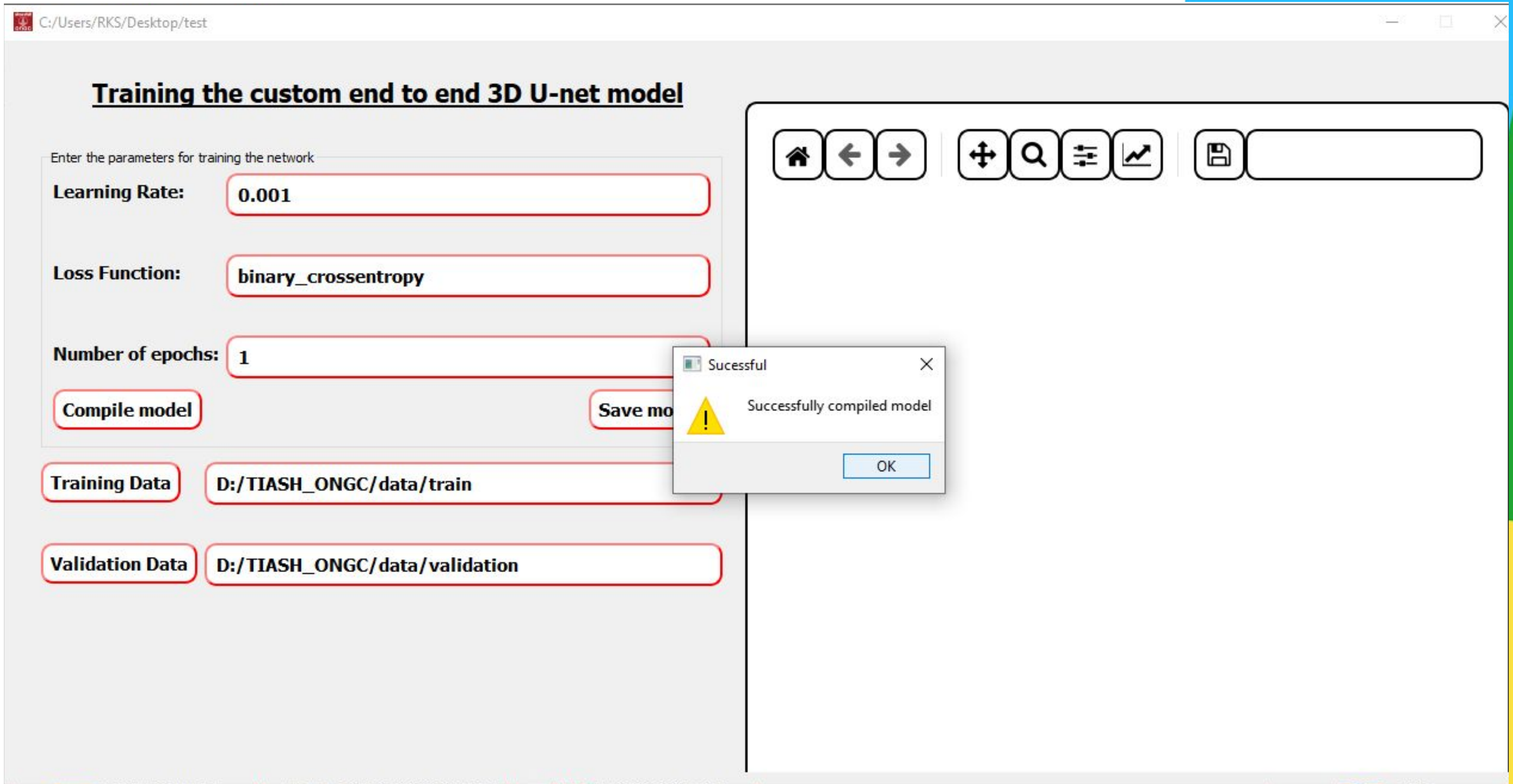
~partial integration left



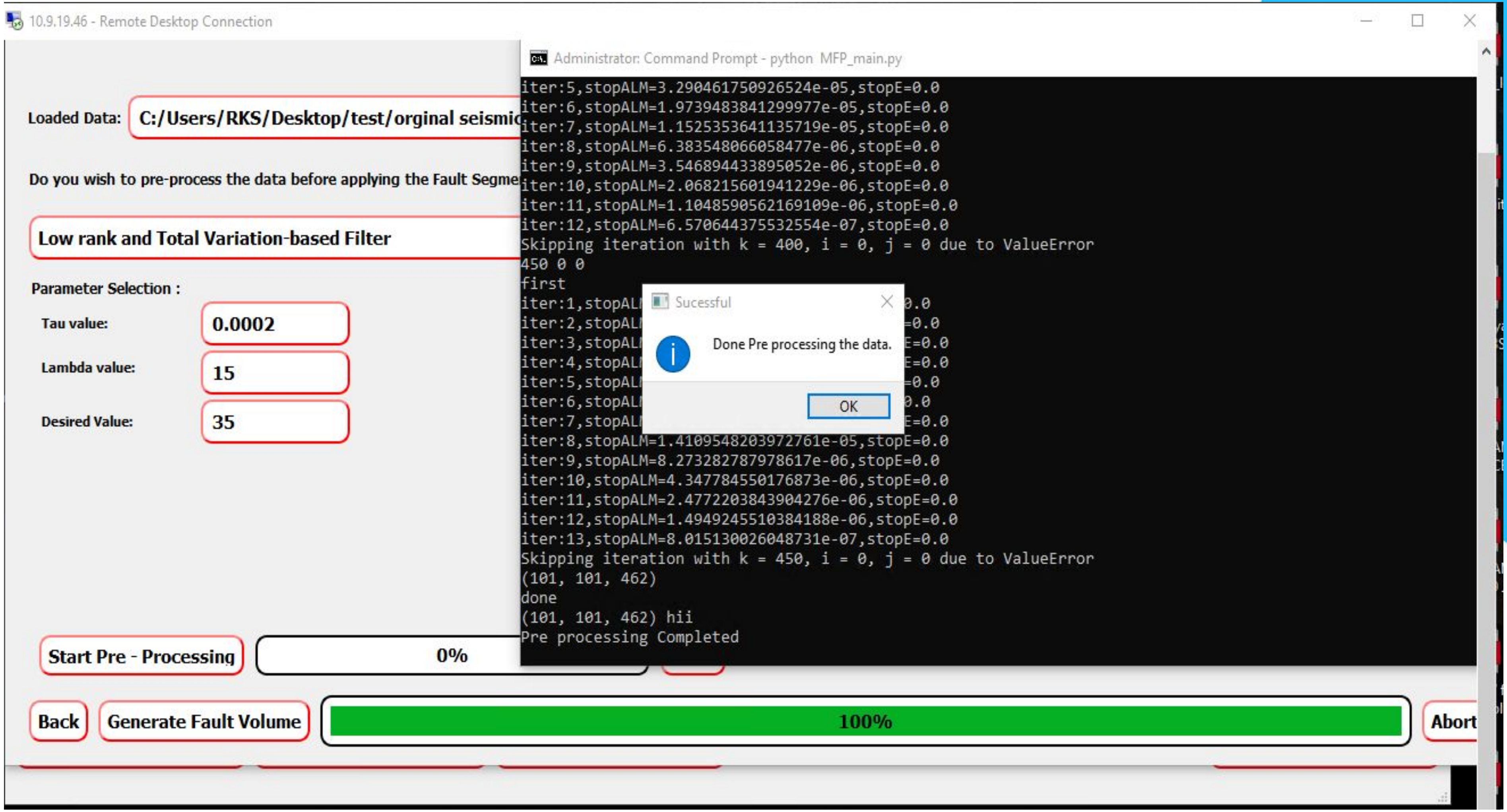
Train 3D net using custom Unet model



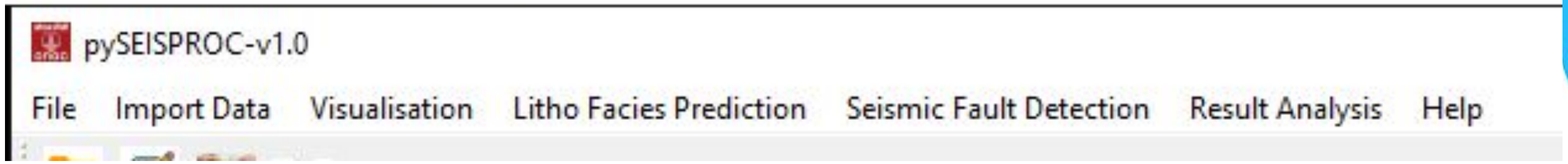
Train 3D net using custom Unet model

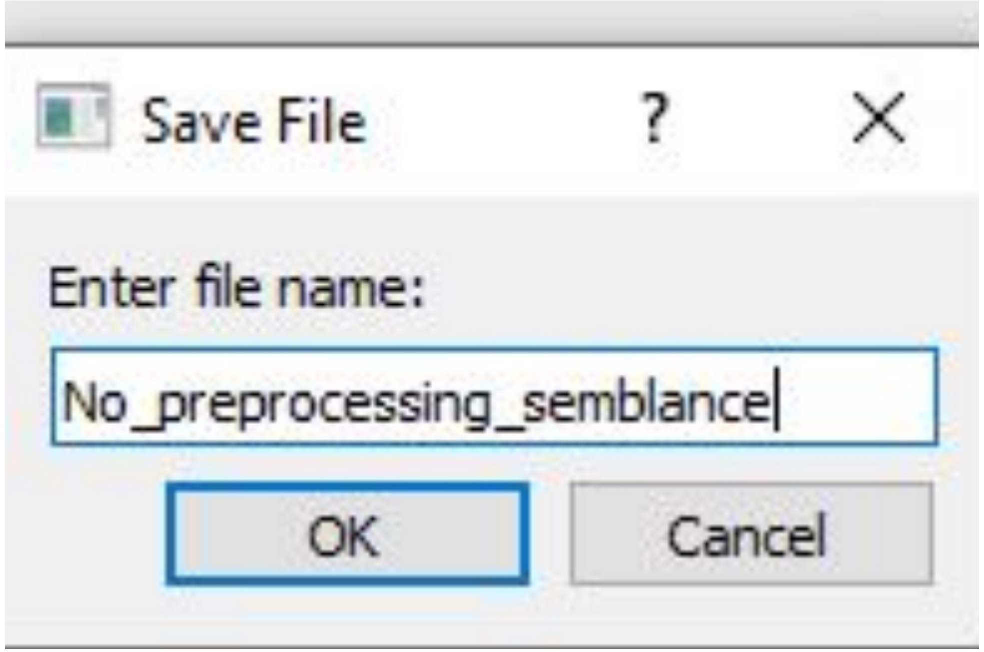


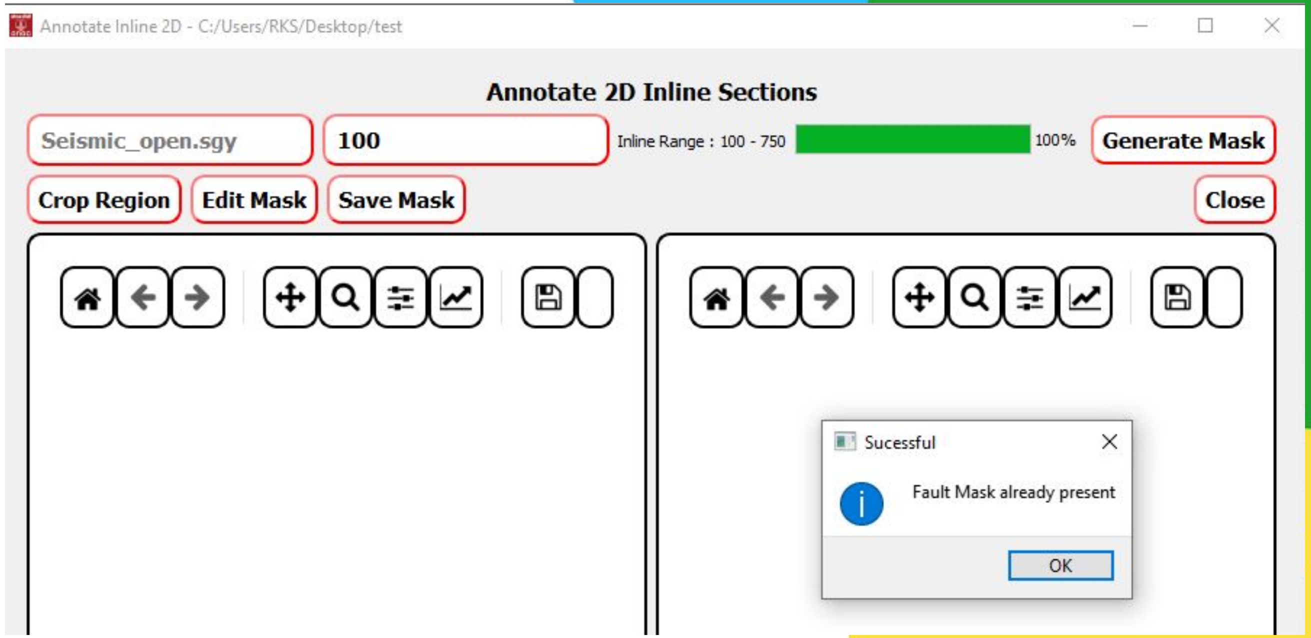
Data PreProcessing



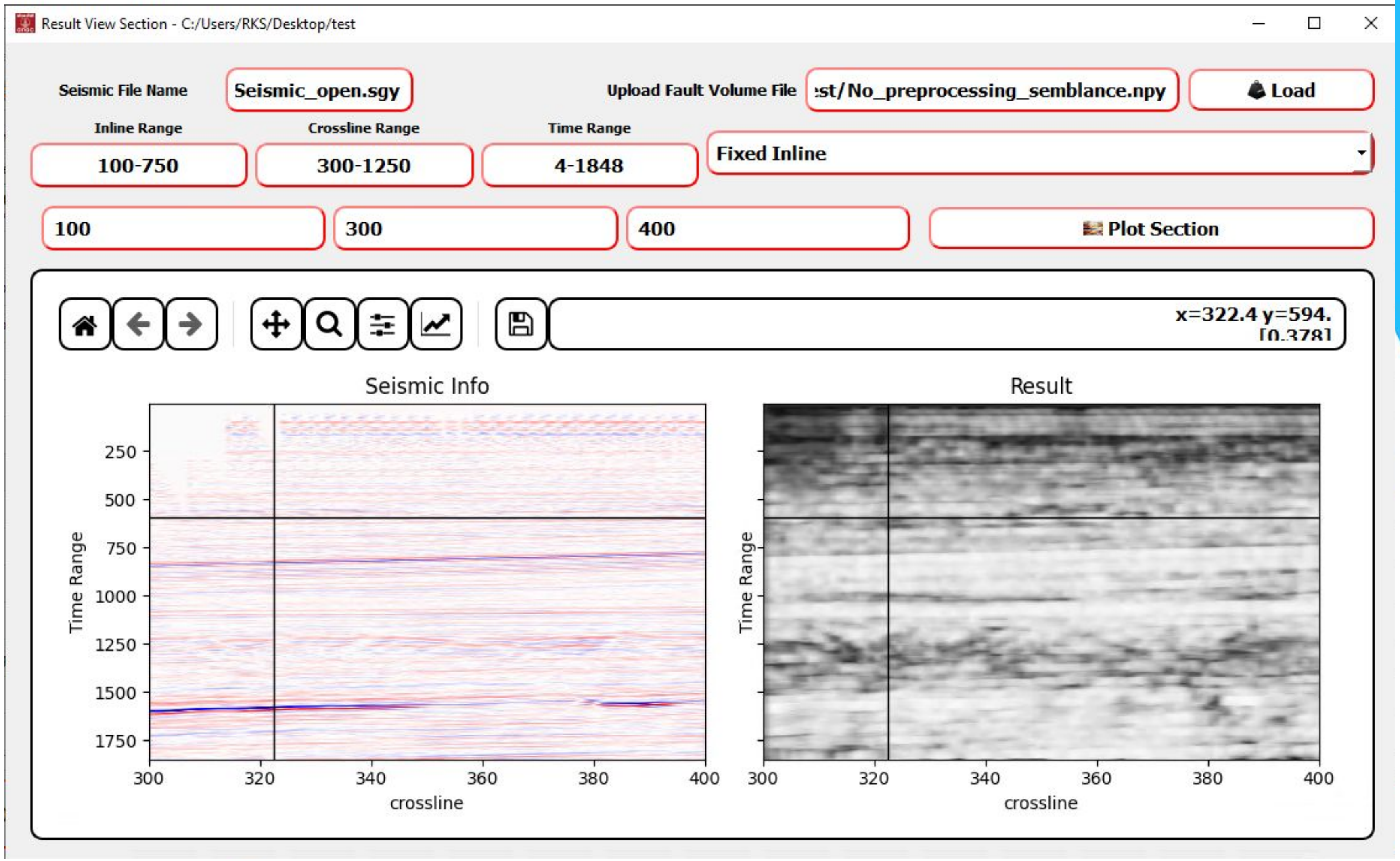
Save File, Visualization Tab, Annotate 2D Inline







2D Viewer



Result View Section - C:/Users/RKS/Desktop/test
Seismic File Name
Seismic_open.sgy
Upload Fault Volume File
:st/No_preprocessing_semblance.npy
Load
Inline Range
100-750
Crossline Range
300-1250
Time Range
4-1848
Fixed Inline
100
300
400
Plot Section
x=322.4 y=594.
Seismic Info
Result
Time Range
250
500
750
1000
1250
1500
1750
300
320
340
360
380
400
crossline

Code Snippet - Save File

CropedDataPy.py

This chunk of code is used to enable save file option at various places.

```
2d.py ×   Croped_DataPy.py ×   sectionview_2DPy.py ×
#SAVE FILE (Rename)**********************************************************************************
base_directory = self.Base_directory
file_name = "Semblance_Fault_volume.npy"

# Search for the file in the base directory
file_path = None
for root, dirs, files in os.walk(base_directory):
    if file_name in files:
        file_path = os.path.join(root, file_name)
        break

#If the file was found, get the new file name from the user and rename the file
if file_path:
    new_file_name, ok = QtWidgets.QInputDialog.getText(self, "Save File", "Enter file name:")
    if ok:
        new_file_path = os.path.join(os.path.dirname(file_path), new_file_name+".npy")
        if os.path.exists(new_file_path):
        # # If the renamed file already exists, ask the user if they want to replace it
        #     reply = QtWidgets.QMessageBox.question(self, "File Exists", "The file already e
        #     if reply == QtWidgets.QMessageBox.Yes:
            os.remove(new_file_path)
        #     else:
        #         QtWidgets.QMessageBox.warning(self, "Error", "Could not save file. File wit
        os.rename(file_path, new_file_path)
```

Code Snippet - 2D viewer

SectionView2DPy.py.py

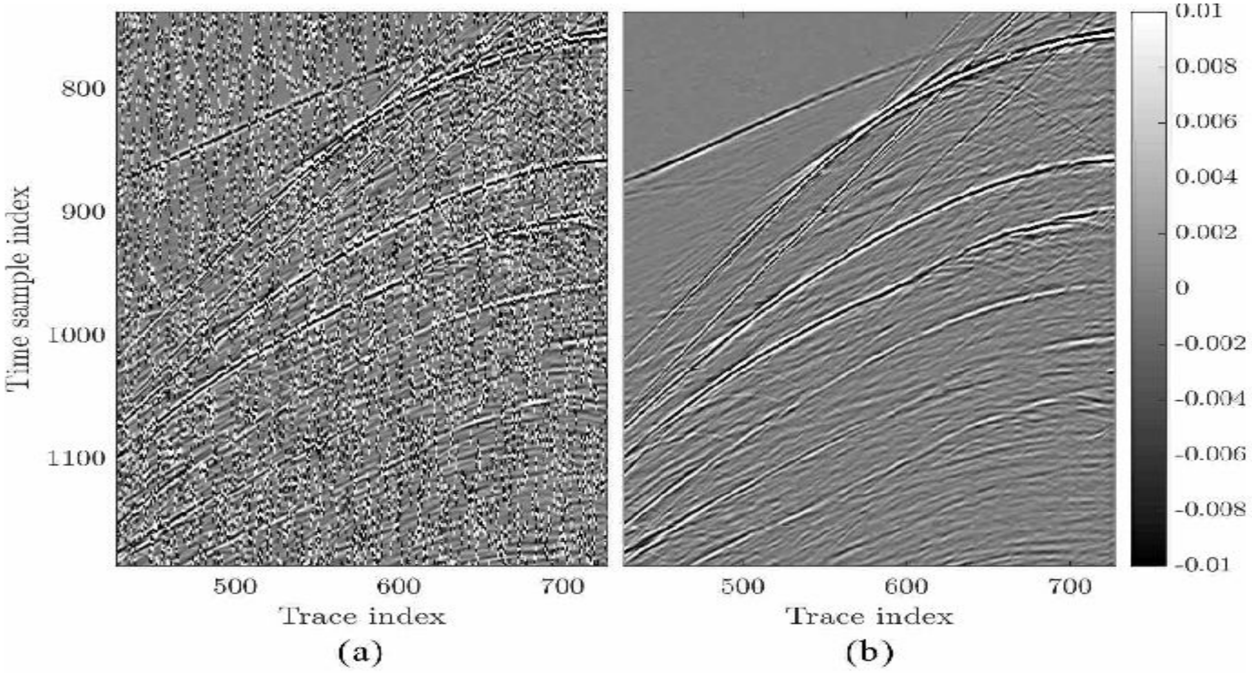
ataPy.py × sectionview_2DPy.py ×

```python
def update():
    self.progressBar.hide()
    # self.stackedWidget.setCurrentIndex(1)
    original_seismic=self.result_queue.get()
    self.graphicsView_3.figure.clf()
    ax3,ax4=self.graphicsView_3.figure.subplots(ncols=2,sharey=True,sharex=True)

   # ax3=self.graphicsView_3.figure.subplots(ncols=1,sharey=True)

    ax3.imshow(original_seismic.T,interpolation='gaussian',cmap='seismic',aspect='auto',extent=[int(self.

    ax3.set_ylabel('Time Range')
    ax3.set_xlabel('crossline')
    ax3.set_title('Seismic Info')
    # ax3.set_ylim(time_range[0],time_range[1])
    #self.graphicsView_3.figure.tight_layout()
    self.graphicsView_3.canvas.draw()

    xlineStart = int((int(self.lineEdit_11.text()) - crossline[0])/int(self.step_xline))
    xlineEnd = int((int(self.lineEdit_12.text()) - crossline[0])/int(self.step_xline))
    inlineSession = int((int(self.lineEdit_10.text()) - inline[0])/int(self.step_inline))
    newData = data.transpose(0,2,1)
    print(newData.shape)
    if newData.shape[0] < inlineSession or newData.shape[2] < xlineEnd:
        msg = QtWidgets.QMessageBox()
        msg.setIcon(QtWidgets.QMessageBox.Warning)
        msg.setText("Fault volume not present for the input range. Fault volume shape" + str(data.shape))
        msg.setWindowTitle("Error")
```

Denoising the Seismic data using LRTV algorithm

Why denoising?

Denoising is a critical step in processing seismic data in the oil and gas industry. Seismic data is acquired by generating sound waves and recording the echoes that bounce back from the subsurface layers of the earth. However, the recorded data is often contaminated with noise, which can distort the signal and make it difficult to interpret.



LRTV(linear regression with total variation regularization)

The LRTV (Linear Regression with Total Variation regularization) algorithm is a method for denoising 3D seismic data. It is a type of model-based denoising algorithm that uses a combination of linear regression and total variation regularization.

Local regression involves fitting a model to a small neighborhood of each point in the image, which helps to preserve local structures and patterns. Total variation regularization penalizes the total variation of the image, which helps to smooth out noise and preserve sharp edges and features.

How LRTV works ?

Here's how the LRTV algorithm works:

1. The seismic data is first divided into small patches.
2. For each patch, a linear regression model is fit to the data using nearby patches as predictors.
3. The residual between the data and the regression model is then computed.
4. The LRTV algorithm applies total variation regularization to the residuals to suppress noise while preserving edges and fine details.
5. Finally, the denoised patch is obtained by adding the regularized residuals to the regression model.

Approach for Implementing Denoisization using LRTV:-

1. Reading the .sgy and converting it to .mat file as we can use various libraries which makes it more flexible to use as compared to numpy.

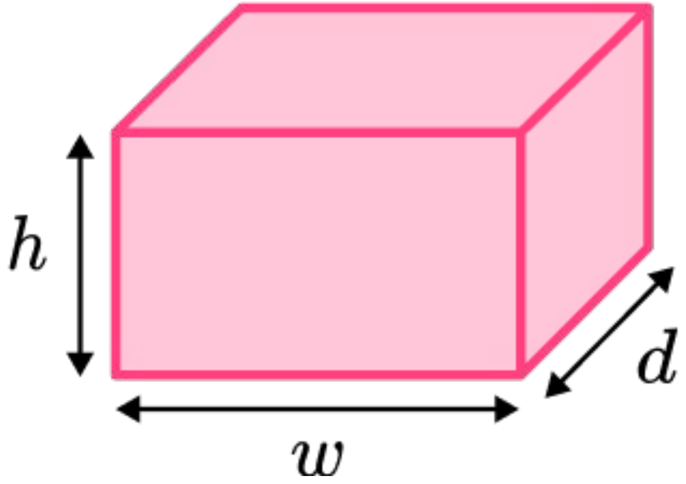
```python
import numpy as np
import matplotlib.pyplot as plt
import numpy as np
from segpy.reader import create_reader
import matplotlib.pyplot as plt
import scipy
import numpy as np
from scipy.io import savemat

with open('D:\IS project\Seismic_data.sgy', 'rb') as file:
    # The seg_y_dataset is a lazy-reader, so keep the file open throughout.
    seismic = create_reader(file, endian='>')  # Non-standard Rev 1 little-endian
    print(seismic.num_traces())
sel_trace=[]
'''Multi trace    (function) inline_numbers: Any
inline=seismic.inline_numbers()
xline=seismic.xline_numbers()

print(inline,xline)
print(len(inline))
print(len(xline))
inline=np.array(inline)
xline=np.array(xline)
seis_vol=[]
```

```python
for i in range(len(inline)-50):
    inline_start_index=inline[i]
    synthetic_seismic=np.zeros([len(seismic.trace_samples(1)),len(xline)])
    for j in range(len(xline)):
        xline_start_index=xline[j]
        tpl=(inline_start_index,xline_start_index)
        trace_index=seismic.trace_index(tpl)
        trace=seismic.trace_samples(trace_index)
        trace=np.array(trace)
        t=np.arange(0,len(trace),1)
        synthetic_seismic[:,j]=trace
    seis_vol.append(synthetic_seismic)
print(seis_vol)
print(type(seis_vol))
seis_vol = np.array(seis_vol)
savemat("segpy.mat",{'arr': seis_vol})
```

2. Reading the .mat file and move along the time axis of the data taking win_size as spatial dimensions of the data along with adjustable weights for proper denoization.

```python
import numpy as np
from numpy.linalg import norm
from lrtv import LRTV


import scipy.io as sio
import matplotlib.pyplot as plt


inp=sio.loadmat("D:\\IS project\\segpy.mat")
inp=inp[list(inp.keys())[-1]]
inp=(inp-inp.min())/(inp.max()-inp.min())
#noisy_org=inp+0.1*np.random.randn(inp.shape[0],inp.shape[1],inp.sh
#noisy = noisy_org[0:142,0:142,:]
noisy=inp[:100,162:,:300]

p,M,N=noisy.shape

win = [M,N]
channel_size = 20
# new_m = M - win_size
# new_n = N - win_size
# new_p = p - channel_size
tau = 0.0015
# lmbda = 15
overlap = 50
lmbda=20/np.sqrt(M*N)
r=35
```

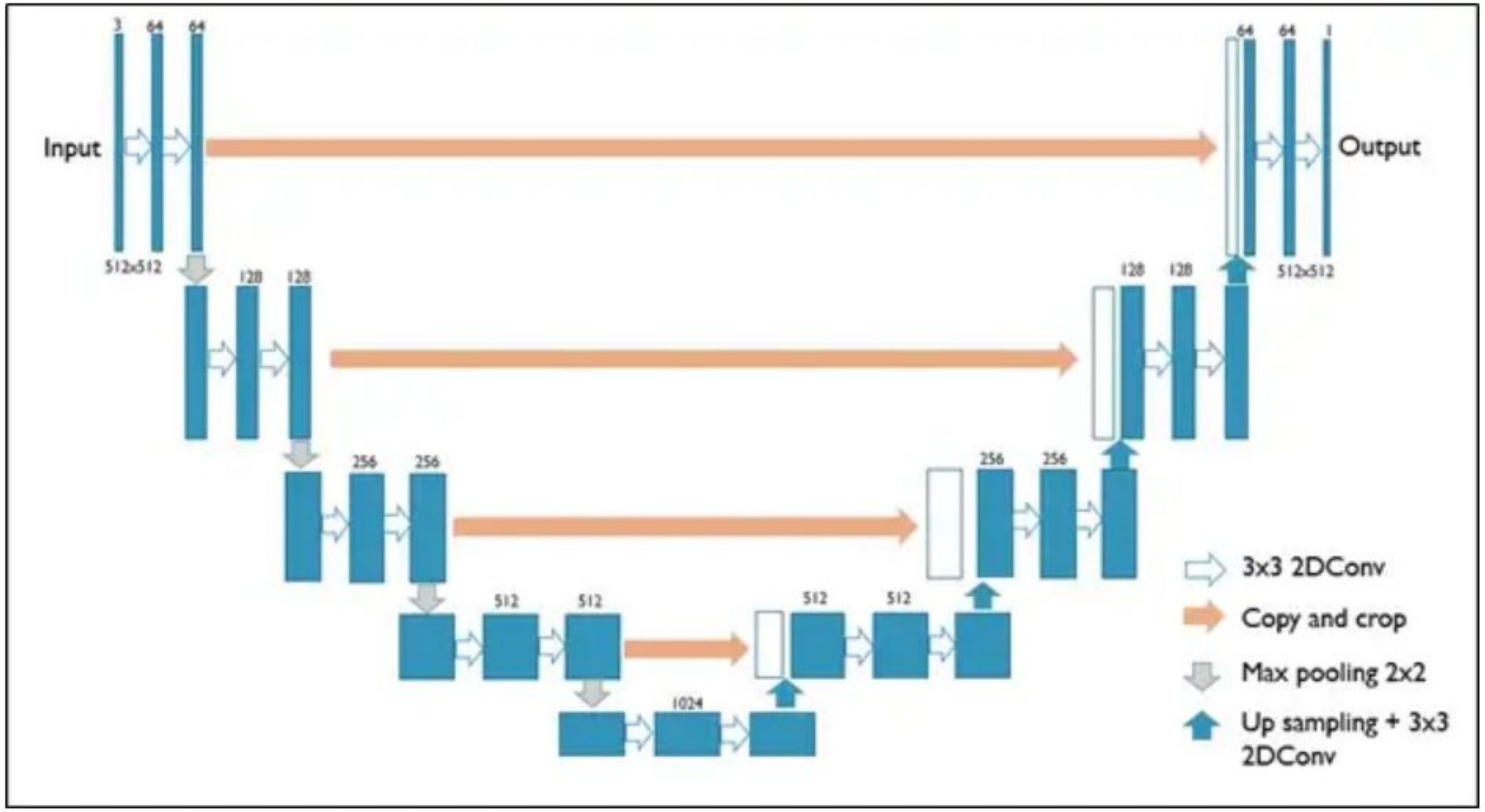
```python
stride = 10

denoised = np.zeros((p,M,N))

weights = np.zeros((p,M,N))
incremental_win = np.ones((p,win[M],win[N]))

for i in range(0, M-1, win[M]):
    for j in range(0, N-1, win[N]):
        for k in range(0, p-1, channel_size):
            try:
                print(k, i, j)
                noisy_patch = noisy[k:k+channel_size, i:i+win[M], j:j+win[N]]
                if k > 0:
                    print("first")
                    noisy_patch = noisy[k-stride:k+channel_size, i:i+win[M], j:j+win[N]]
                    denoised[k-stride:k+channel_size, i:i+win[M], j:j+win[N]] = LRTV(noisy_patch, tau, lmbda, r)
                    weights[i:i+win[M]-1, j:j+win[N], :] =  weights[i:i+win[M]-1, j:j+win[N], :] + incremental_win
                    output_image = LRTV(noisy_patch, tau, lmbda, r)
                    denoised[:k+channel_size, i:i+win[M], j:j+win[N]] = denoised[:k+channel_size, i:i+win[M], j:j+win[N]] + output_image
                else:
                    print("second")
                    denoised[:k+channel_size, i:i+win[M], j:j+win[N]] = LRTV(noisy_patch, tau, lmbda, r)
                    weights[i:i+win[M]-1, j:j+win[N], :] =  weights[i:i+win[M]-1, j:j+win[N], :] + incremental_win
                    output_image = LRTV(noisy_patch, tau, lmbda, r)
                    denoised[:k+channel_size, i:i+win[M], j:j+win[N]] = denoised[:k+channel_size, i:i+win[M], j:j+win[M]] + output_image
            except ValueError:
                    print(f"Skipping iteration with k = {k}, i = {i}, j = {j} due to ValueError")
                    continue
```



Custom U-NET

U-Net framework for image segmentation

- -Originally developed for segmentation in medical images



U-Net Framework (Image by Rachel Zhiquing Zheng, see Reference)

Custom Model Framing as per user Annotations

The custom unet model provides the flexibility to user to optimize the model in terms of following ways:-

1. The user can provide number of **convolution layers** along with **number of filters** in each layer by keeping constant dimension of 2X2 for max_pooling layer.

2. User can according choose the **downsampling rate**,which gives access to optimize model performance as per user.It also **gives an error** if the parameters provided by the user are not computationally possible.

Code Main Highlights

The following are the main highlights of our Custom-Net3D which allows the flexible approach for model design:-

Convolution block :-

```
def conv_block(input,num_filters,num_conv):
    """
    Construct a convolutional block with a specified number of convolutional layers

    Args:
        input (tensorflow.Tensor): The input tensor to the convolutional block.
        num_filters (int): The number of filters in convolutional layer.
        num_conv (int): The total number of convolutional layers in the block.

    Returns:
        tensorflow.Tensor: The output tensor of the convolutional block.
    """
    x = Conv3D(num_filters,3,padding="same",activation = 'relu')(input)
    for i in range(num_conv-1):
        x = Conv3D(num_filters,3,padding="same",activation = 'relu')(x)
    return x
```

2. Encoder block

Encoder block :-

Pool Size is fixed :2

```
def encoder_block(input,num_filters,num_conv):
    """
    Construct an encoder block, which consists of a convolutional block followed by max pooling.

    Args:
        input (tensorflow.Tensor): The input tensor to the encoder block.
        num_filters (int): The number of filters in the first convolutional layer of the block.
        num_conv (int): The total number of convolutional layers in the block.

    Returns:
        tuple: A tuple containing the output tensor of the convolutional block and the max pooling layer.
    """
    x = conv_block(input,num_filters,num_conv)
    p = MaxPooling3D(pool_size=(2,2,2),strides=(2,2,2))(x)

    return x, p
```

3. Decoder block :-

```
def decoder_block(input, skip_features, num_filters, num_conv):
    """
    Implements a single decoder block operation for convolutional neural networks used in tasks like image segmentation.

    Parameters:
    input (tensor): input tensor to the decoder block
    skip_features (tensor): skip connection features from the corresponding encoder block
    num_filters (int): number of filters to use in the decoder block
    num_conv (int): number of convolutional layers to use in the decoder block

    Returns:
    tensor: output tensor from the decoder block
    """

    # Upsample the input tensor using UpSampling3D layer with size 1
    x = UpSampling3D(size=1)(input)

    # Apply a 3D transposed convolutional layer to x tensor
    # with num_filters filters, kernel size of (2,2,2), stride of 2, and padding set to "same"
    x = Conv3DTranspose(num_filters, (2, 2, 2), strides=2, padding="same")(x)

    # Concatenate skip_features tensor with x tensor along the depth axis
    x = Concatenate(axis=4)([skip_features, x])

    # Apply num_conv convolutional layers with num_filters filters in each layer
    # to the concatenated tensor x using the conv_block function
    x = conv_block(x, num_filters, num_conv)

    # Return the output tensor from the decoder block
    return x
```

4. The Error trigger function:-

```
import math

def PosDown(input_shape):
    """
    Returns the number of times a given input shape can be divided by 2.

    Parameters:
    input_shape (tuple): A tuple of integers representing the input shape.

    Returns:
    int: The number of times the input shape can be divided by 2.
    """
    l = input_shape[0]
    ans = 0

    # Iterate over the log base 2 of the first element of the input shape
    for i in range(int(math.log(l, 2))):
        # Check if the length is even
        if l % 2 == 0:
            ans += 1
            l /= 2
        else:
            # If the length is odd, break out of the loop
            break

    return ans
```

Transfer Learning For Real Fault Detection

Pre- midsem:- Fault Detection on synthetic fault detection

Continual Learning:- Transfer of Learnt Knowledge

- Dataset 1:- Synthetic 2d seismic images
- Task 1:-Fault detection from synthetic
- Dataset 2:-Real Seismic images (K-G Basin)
- Task2:- Real Fault Detection

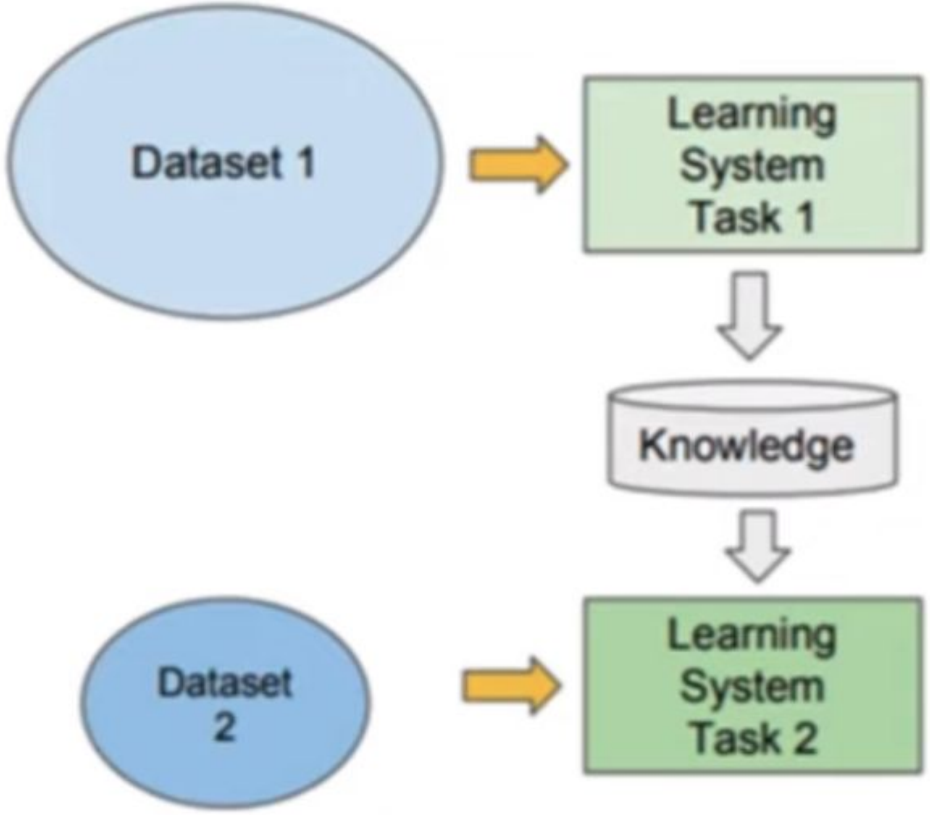
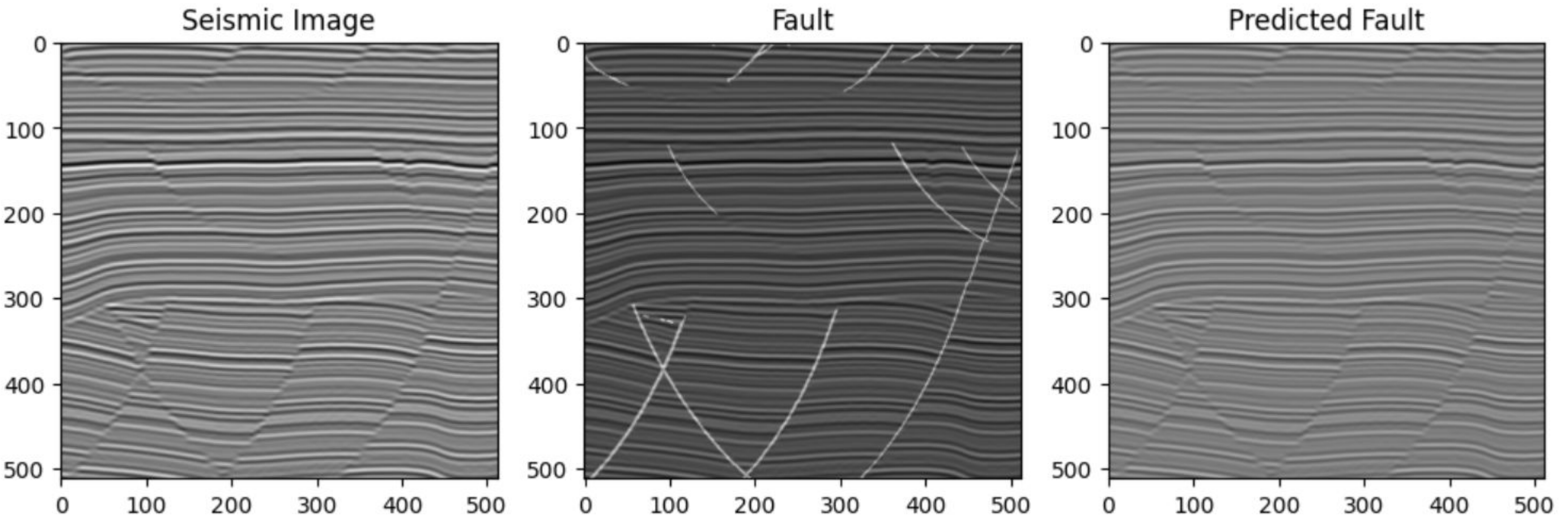


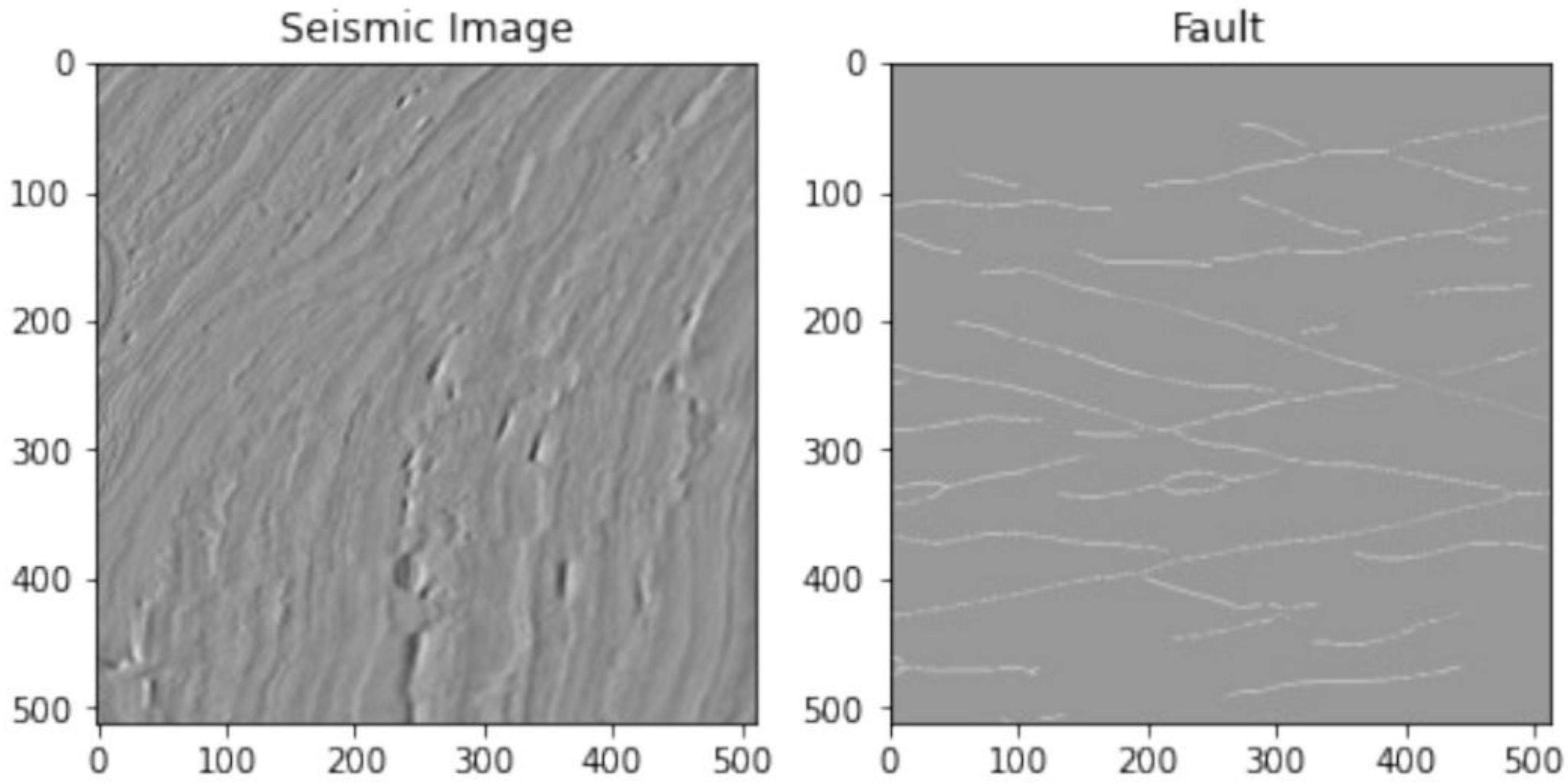
Figure 1: Transfer learning scheme

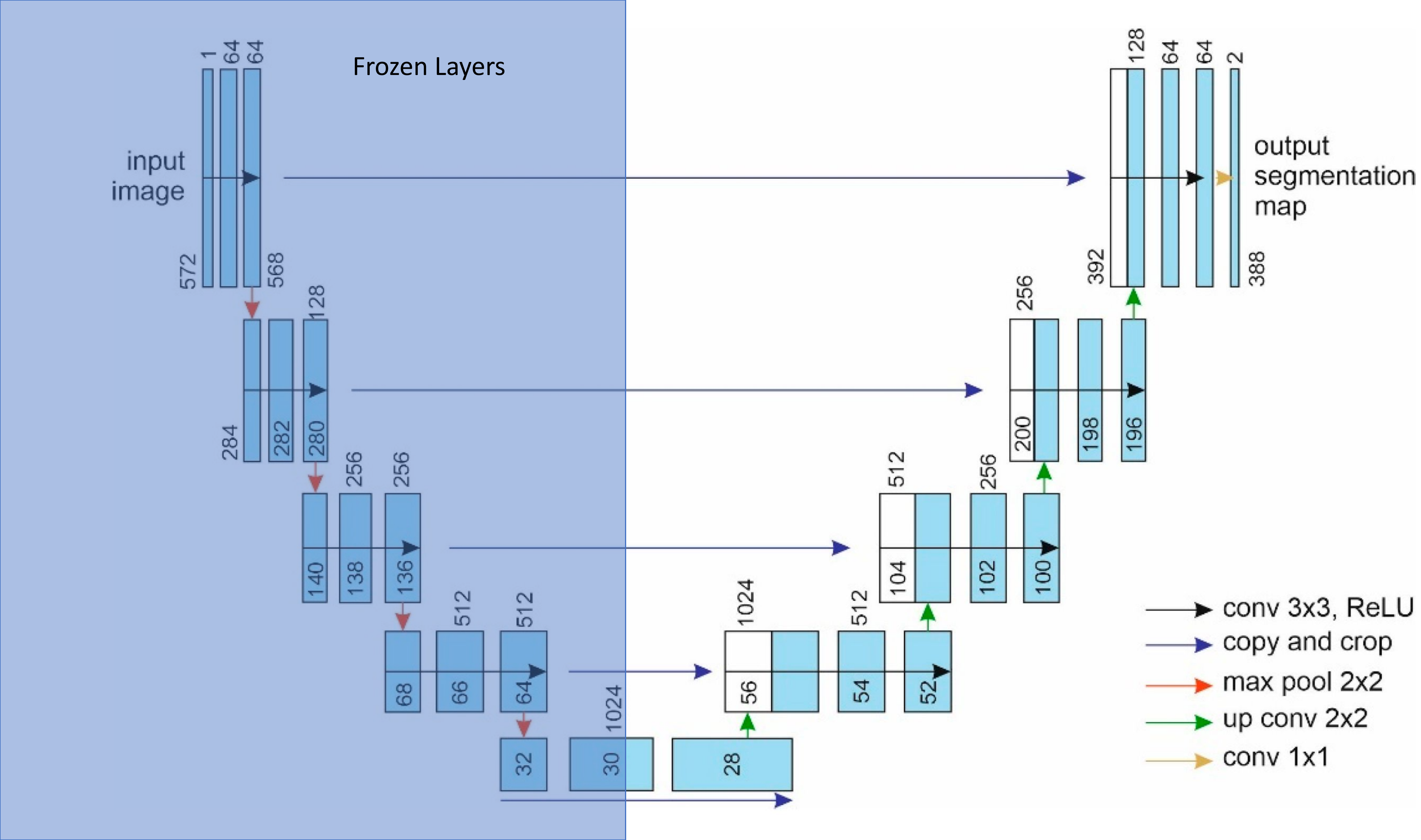
What we have?



- Synthetic Model

What we want?




Frozen Layers
input image
output segmentation map
1
64
64
572
568
128
284
282
280
256
256
140
138
136
512
512
68
66
64
1024
32
30
28
1024
56
512
54
52
512
104
256
102
100
256
200
198
196
128
64
64
2
392
388
conv 3x3, ReLU
copy and crop
max pool 2x2
up conv 2x2
conv 1x1

Code To Freeze and Unfreeze

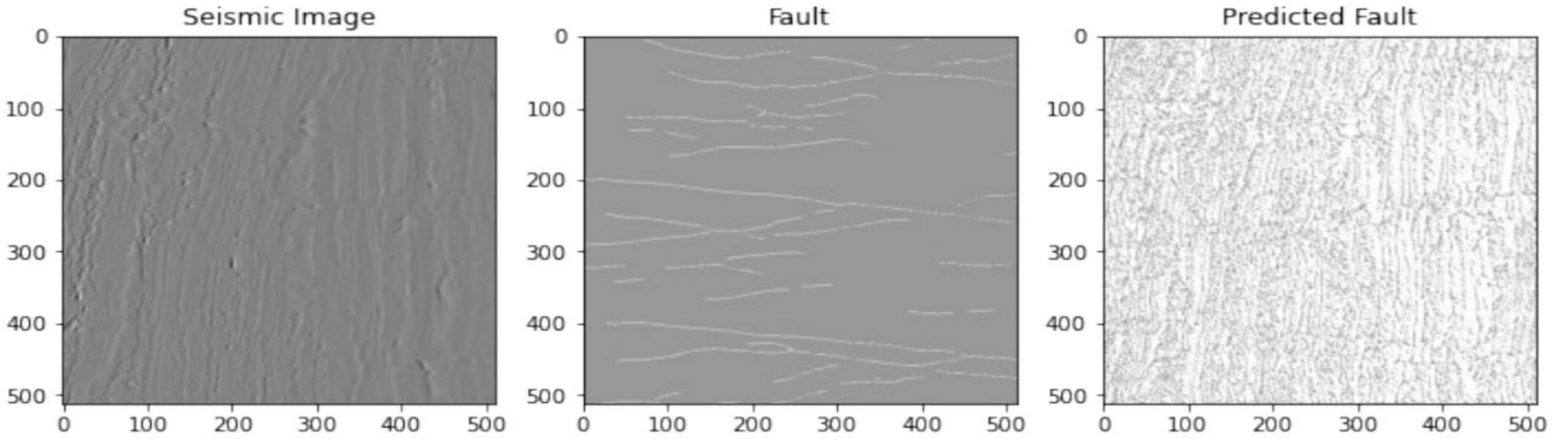
```
#Unfreezing all parameters of model
for param in model.parameters():
    param.requires_grad = True

#Verifying all parameters are unfreezed and are capable to update
weights during learning
for child in model.children():
    print(child)
    for param in child.parameters():
        print(param.requires_grad)

FeatureMapBlock(
  (conv): Conv2d(1, 64, kernel_size=(1, 1), stride=(1, 1))
)
True
True
ContractingBlock(
  (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
  (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
```

Training Model on Unfreezed Weights

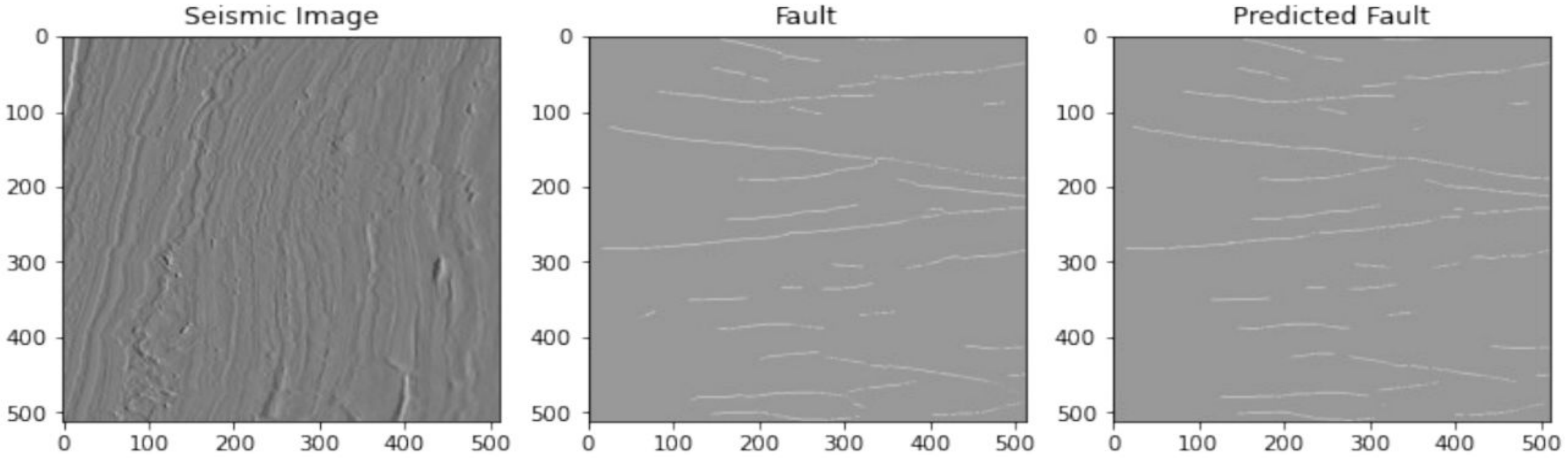
```
def train():
    # Load data into data loader with specified batch size and
shuffling
    dataloader = DataLoader(
        dataset,
        batch_size=batch_size,
        shuffle=True)

    # Initialize UNet model with specified input and output dimensions
and move to GPU if available
    unet = UNet(input_dim, label_dim).to(device)

    # Initialize optimizer for UNet with specified learning rate
    unet_opt = torch.optim.Adam(unet.parameters(), lr=lr)

    # Initialize variables for keeping track of training progress
    cur_step = 0
    s=0
    train_losses = []
    train_accs = []

    # Loop through each epoch
    for epoch in range(n_epochs):
        # Loop through each batch in the data loader
        for real, labels in tqdm(dataloader):
            # Get the current batch size
            cur_batch_size = len(real)

            # Move the real and labels tensors to the GPU if available
            real = real.to(device)
            labels = labels.to(device)

            ### Update U-Net ###

            # Reset gradients for the UNet optimizer
            unet_opt.zero_grad()

            # Get the predicted output of the UNet on the current
batch
            pred = unet(real)

            # Calculate the loss between the predicted output and the
ground truth labels
            unet_loss = criterion(pred, labels)
            train_losses.append(unet_loss.item())

            # Backpropagate the loss and update the UNet parameters
            unet_loss.backward()
            unet_opt.step()
```

At 0th Epoch

```
Epoch 0: Step 0: U-Net loss: 54.097686767578125, Train Accuracy:
23.785400390625
```

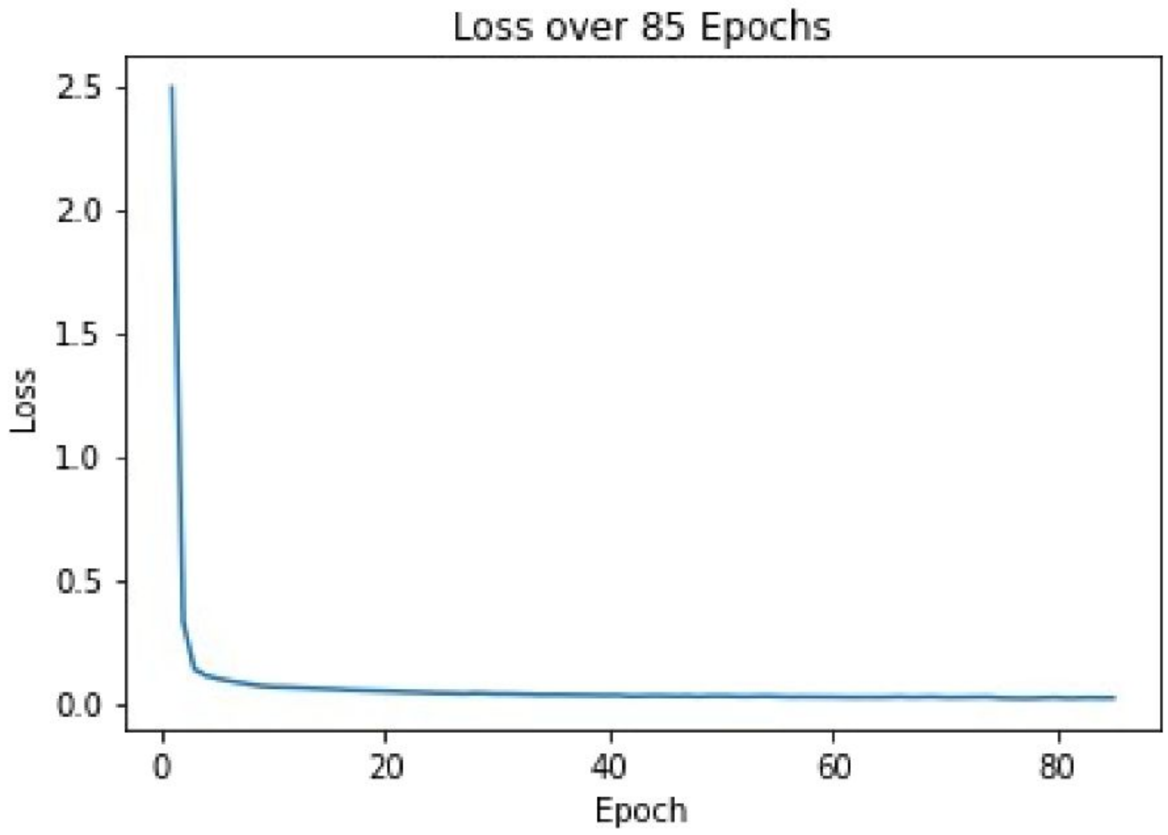


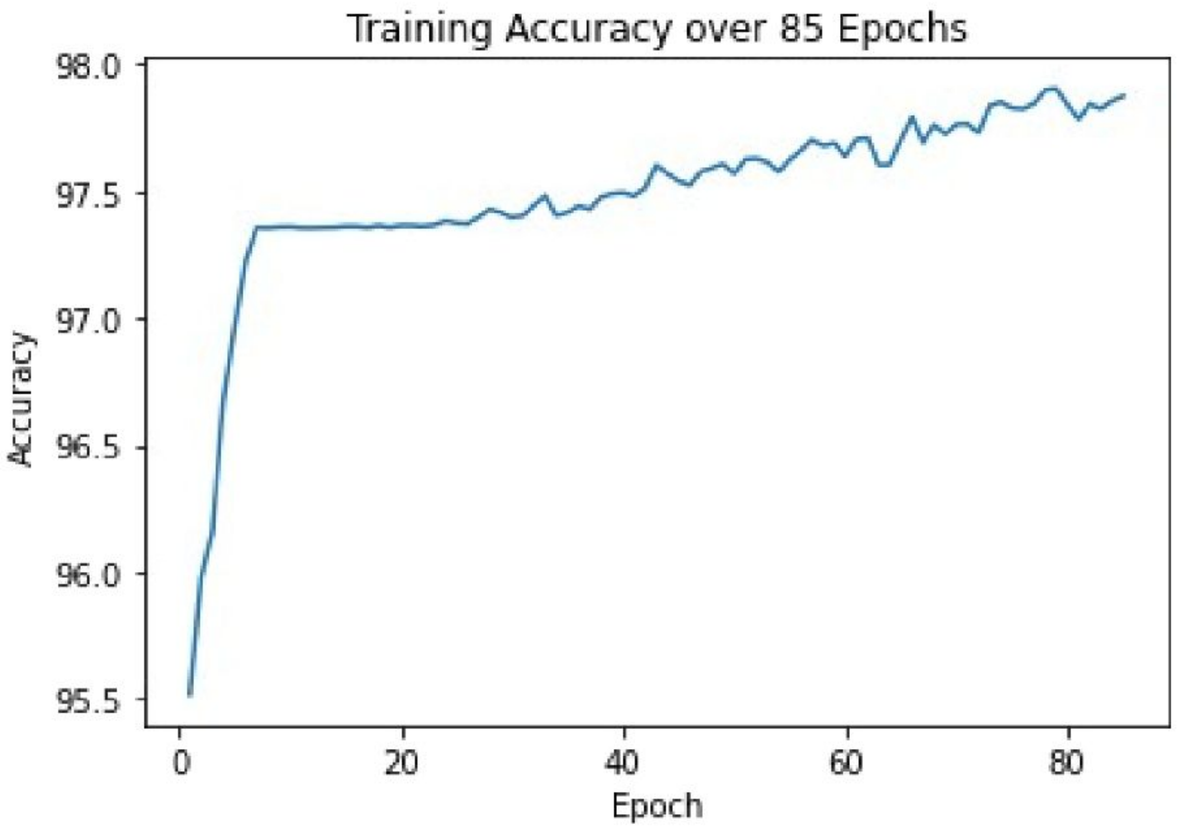
Final Epoch



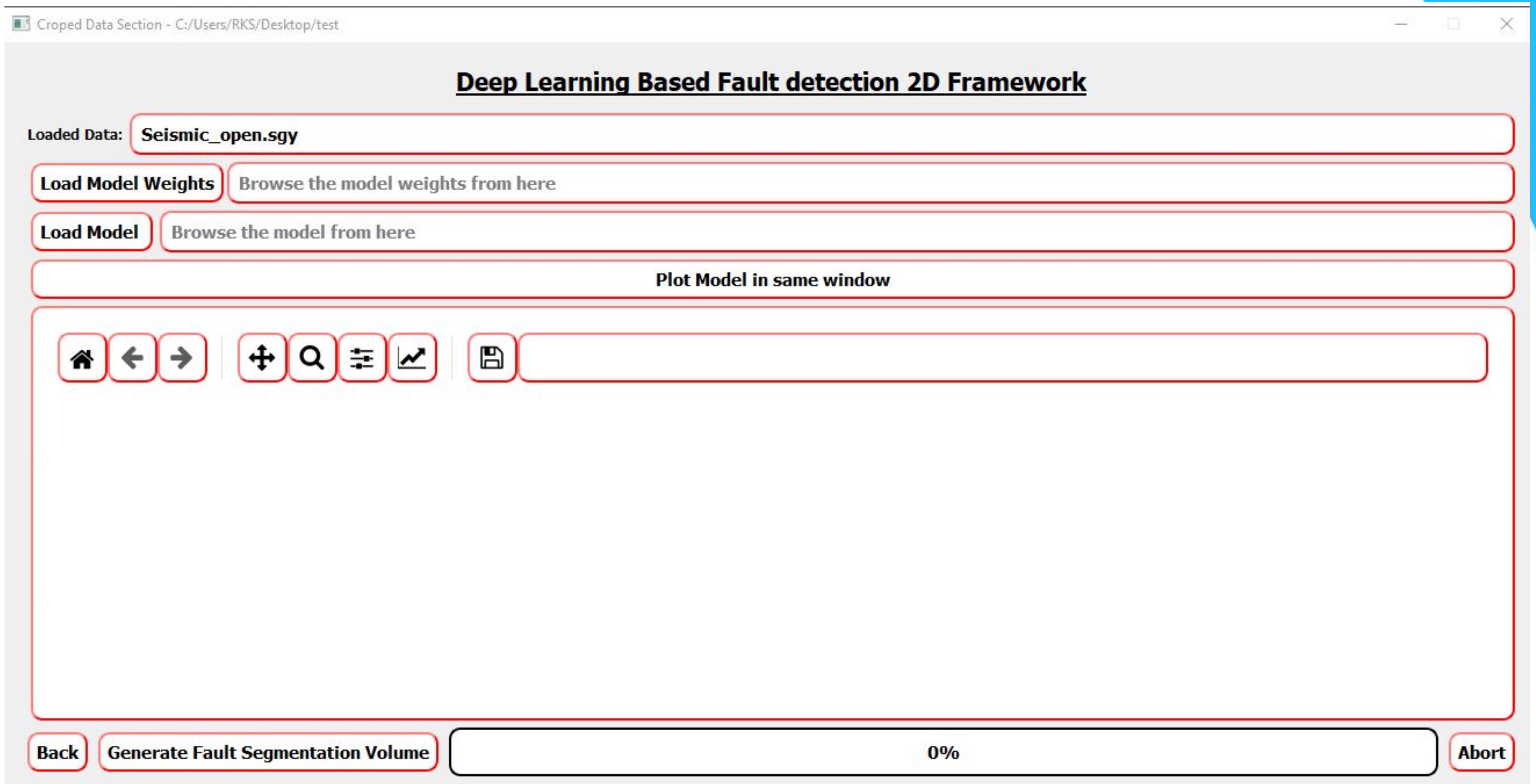

```
Epoch 82: Average Train Accuracy: 98.65017026472734
```

Loss Curve And Accuracy Curve





Deep Learning based 2D fault Detection



Future work:

- Save options in Annotate 2D
- Complete integration of the 2D Fault detection UI page
- Train 3D Unet - Loss & accuracy curve printing (training + validation)
- 2D network testing page - Custom 2D Unet model

Thank you

~ Team 28