**Improving the performance of seismic fault detection using Convolutional Neural Network**

*Himanshu Badera Abhishek Goyal*

*Danilo Meza Subhash Mekapati*

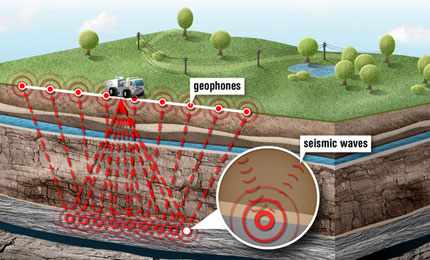
*Nikhil Gorantla Praveen Dasari*

**Abstract**

Seismic surveys are one of the primary mechanisms used in oil and natural gas exploration, both onshore and offshore. It has lowered the cost of exploration and allowed the discovery of reserves that were previously undetected by traditional means. The process of surveying can be divided into two major steps: data acquisition and seismic interpretation. Data acquisition utilizes energy sources and an array of sensors to record the seismic waves that travel through the earth. This data is then processed and prepared for seismic interpretation. Fault detection is one of the most important aspects of this process, as it directly contributes to the likelihood of finding hydrocarbons in the subsurface. In the traditional approach, faults are detected as reflection discontinuities or abruptions and manually tracked in post-stack seismic data, which is laborious and time-consuming. To improve efficiency, a variety of automatic fault detection methods have been proposed, with deep learning-based methods receiving the most attention. Current methods require very large, labeled datasets to train the model, which could be impractical for smaller companies that have difficulty obtaining enough data. To overcome this limitation, we use a transfer learning strategy to allow the use of smaller training datasets for automatic fault detection. We begin by training a deep neural network on synthetic seismic data. The network is then retrained using real seismic data. We employ a random sample consensus (RANSAC) method to obtain real seismic samples and automatically generate corresponding labels. Three real 3D examples are analyzed to show how retraining the network with a small number of real seismic samples can greatly improve the fault detection accuracy of the pre-trained network models.

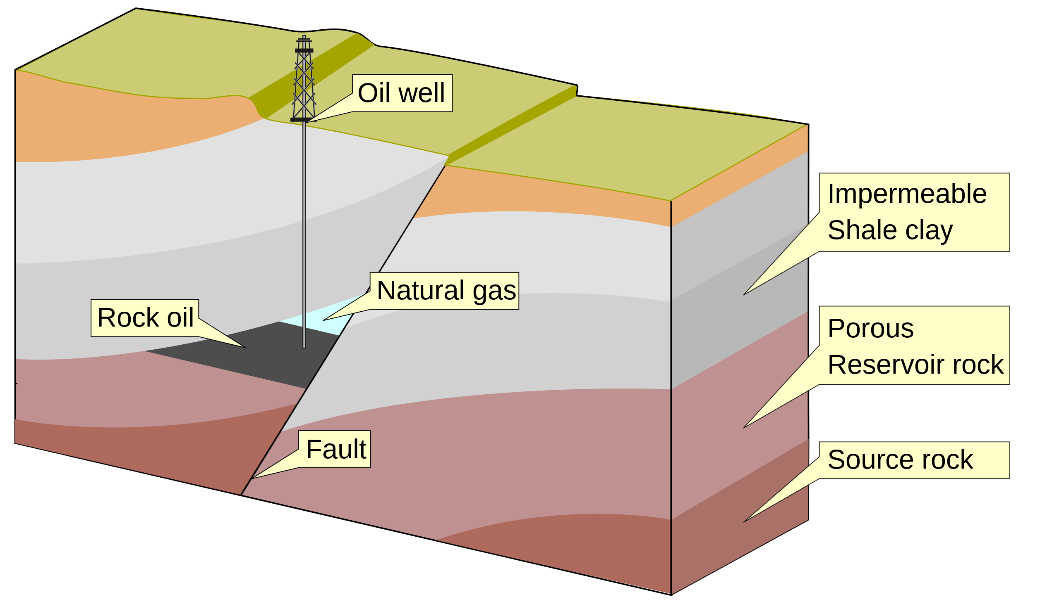
**Introduction**

A seismic survey is conducted above the area of interest by using reflective seismology, an exploration method in which an energy source is used to send a signal or wave into the subsurface. These waves then interact with the rock layers at different speeds and depending on the physical properties they can reflect, refract, or diffract in different ways. These waves are then received by geophones that are placed at the surface and the travel time is recorded. After processing this data, a seismic interpreter will then have the task of mapping geological faults.



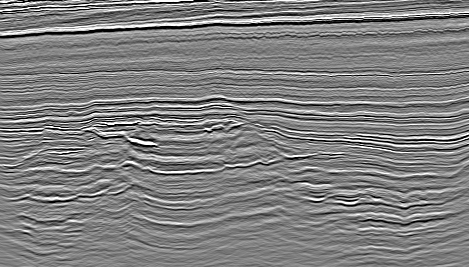
*Figure 1: Seismic Waves*

Faults are geological structures created by the combination of many processes such as tectonic plate movement, gravity, and overpressures. These are cracks or planes in which blocks of rock slip across, and they can have many sizes ranging from meters to kilometers. Faults are important for oil and gas exploration because they may act as a natural trap for hydrocarbons and can also help in the migration of these. Additional to the discovery of hydrocarbons, fault mapping is an essential step in the reservoir characterization process, which aims to provide an optimal understanding of the reservoir’s internal architecture and help calculate key economic indicators.



*Figure 2: Seismic Faults*

Traditional methods for fault detection require a seismic interpreter to trace and label faults manually. This process consumes substantial amounts of time and can be considered an inefficient process as it could take anywhere from weeks to months to label faults within a typically sized area of interest. To improve the interpretation efficiency, numerous researchers have proposed various methods. Since faults generate discontinuities and abruptions, some methods use statistical models to try to identify faults. However, fault detection using only a few physical attributes is limited and not very effective at correctly detecting faults. Therefore, the best detection mechanisms use machine learning and deep neural networks (DNNs) to achieve fast and reliable results in fault detection.

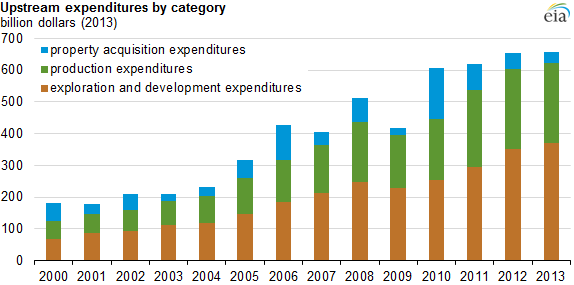


*Figure 3: An example of a seismic image (Source: Force Competition)*

The convolutional neural network (CNN) is one of the most common DNNs, and it has proven to be very effective at many tasks. However, to properly train a CNN, we need a large number of processed and labeled samples, which can be in the thousands or millions. Seismic data is very expensive to acquire so having large datasets to work with is usually out of reach for smaller companies in the industry. So, overcoming this limitation is of paramount importance, as it has a direct impact on the costs of exploration. In this paper, we are going to use a strategy that utilizes synthetic data and a small sample of real data to train a CNN based on the U-Net architecture to automatically detect faults. In real-world applications, it would mean that a seismic interpreter would only need to provia de few labeled samples of a dataset to train the network, and then the model should be able to automatically detect and label the rest of the faults in the same dataset.

**Business Problem**

For oil and gas producers, upstream expenses can be divided into the following categories. Exploration and development include expenditures related to searching for and developing the facilities and infrastructure to produce reserves. Production includes costs associated with extracting oil and natural gas from the ground once the field has been developed. Property acquisition includes costs incurred to purchase proved and unproved oil and natural gas reserves. As we can see from the chart published by the U.S. Energy Information Administration, the costs of exploration and drilling make up a large portion of their yearly expenditures.



***Figure 4: Source:****U.S. Energy Information Administration, based on*[*Evaluate Energy*](http://www.evaluateenergy.com/)*database*

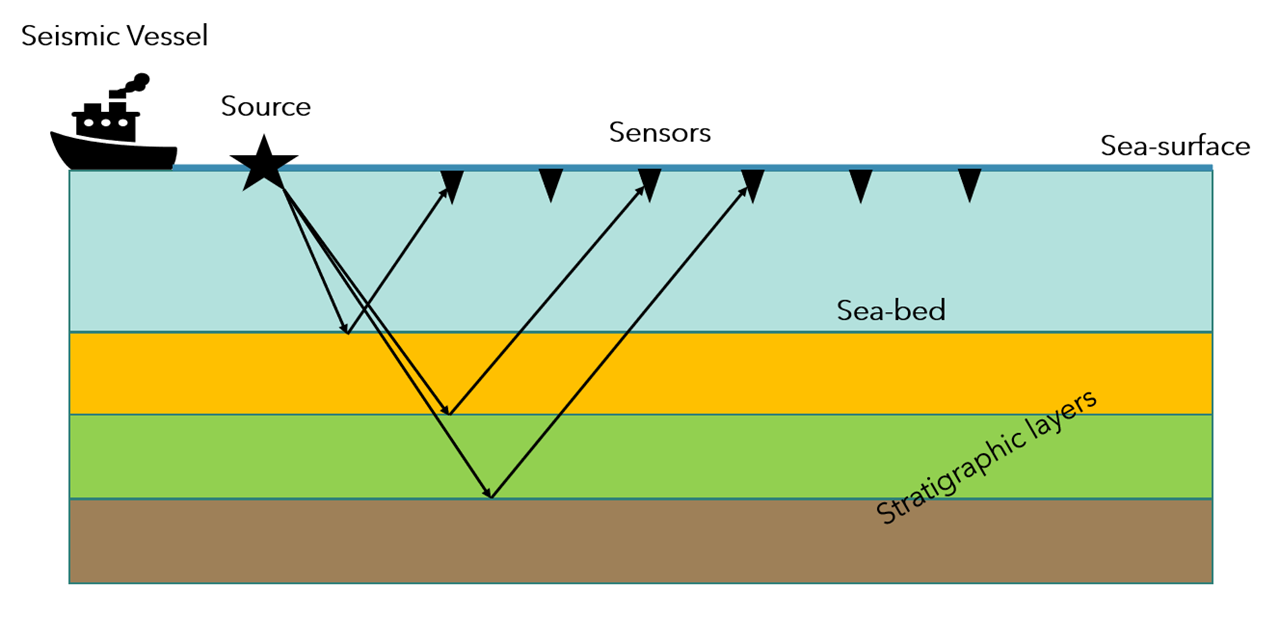
Cost-savings strategies are of high importance to producers as they will result in significant improvements to their profits. The possible economic potential benefits of our model are:

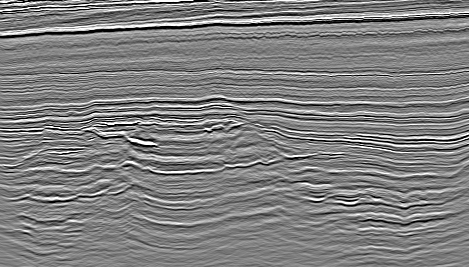
* **Significant reduction of labor and time.** Seismic interpreters using traditional methods can take weeks or months to map a typical dataset. By implementing CNNs, we can reduce that time to days and the only investment would be in computing capacity.
* **Reduction of dry holes.** The improved fault detection effectiveness could reduce the number of dry holes because of higher quality mapping and better reservoir characterization.
* **Large datasets not needed.** 3D mapping can cost between $40,000 to $100,000 per square mile or more. Purchasing large datasets can be very expensive. This model would not require the use of large datasets for training, resulting in significant savings versus traditional DNNs.
* **Faster reserves.** By accelerating the process of exploration, we can potentially cover more areas of exploration each year and increase the chances of developing reserves. The increase in reserves can give more economic value to the company.

Next, we introduce the methodology of the proposed strategy for seismic fault detection. Finally, we conclude our findings at the end of this paper.

**Literature Review**

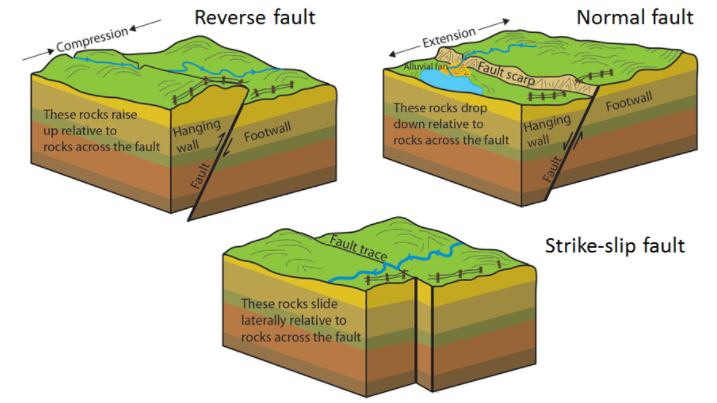
A seismic image provides a structural snapshot of the Earth's subsurface at one point in time. A 'source' sends a sound wave to the subsurface, which travels through the Earth's layers at different speeds and is reflected, refracted, or diffracted along the way. In seismic imaging, we record and stack the waves reflected from different geological layers to create a 2D or 3D image. Because different geological layers have different physical properties, the waves are reflected at the boundary between layers due to density contrast. There are various types of waves, but in imaging, we are mostly concerned with P-waves (compressional waves). The image below shows an example of seismic acquisition and the final image after collecting all of the waves.

*Figure 5: Seismic Image*



*Figure 6: An example of a Seismic image*

Faults are geological structures formed by a variety of physical processes such as pressures, gravity, plate tectonics, and so on. These are the cracks or planes that a block of rocks will slide across. Faults come in a variety of sizes, ranging from a few meters to miles. The **San Andreas Fault** is an example of strike-slip faulting on a massive scale (also called Transform Fault).



*Figure 7: Different types of Faults*

**Significance of Fault Mapping in seismic data**

Fault mapping is an important aspect of seismic exploration because careful analysis of faults can help determine whether there is a chance of finding an oil/gas reservoir in the subsurface. Two points are critical during the early exploration phase:

* Faults may act as conduits for hydrocarbon migration
* it may aid in the trapping of oil in place.

The mapping of faults during the oil/gas development phase is critical in making economic analyses because faults influence the hydrodynamics of a reservoir by changing the fluid permeability. This will have a direct impact on the volumetric of an oil/gas play as well as the mechanical engineering aspect of oilfield development. The presence of faults in the shallow subsurface poses a drilling risk. Correct identification of faults will allow drilling bits to be guided in such a way that any faulted region is avoided as much as possible.

Finally, large-scale fault mapping aids in understanding regional geodynamic processes occurring on Earth. This is essential for comprehending natural hazards such as earthquakes, volcanoes, landslides, and so on.

As we can see, there are numerous advantages to fault identification, particularly in hydrocarbon exploration. As a result, considerable effort has been expended in seismic exploration to accurately identify and map the faults. Manual mapping of faults, on the other hand, is a time-consuming process that can take days or weeks even in a small survey area.

Now we will see some Machine Learning and/or Deep Learning methods on how to identify faults faster.

**Data**

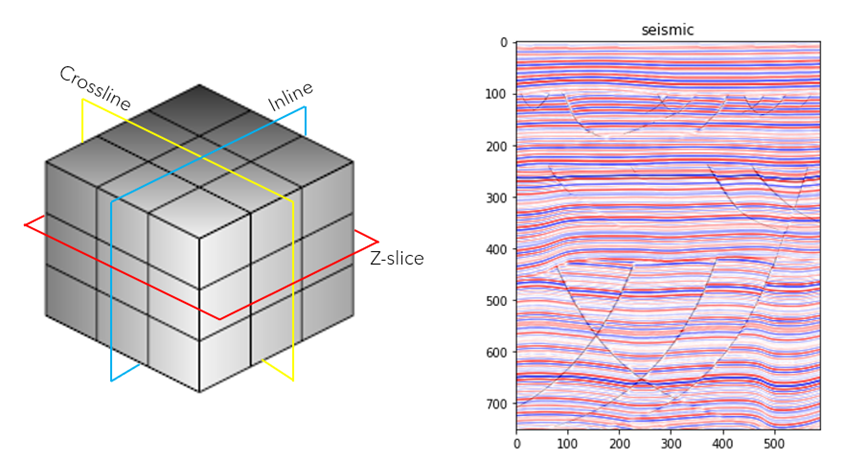
With advances in Deep Neural Network technology, it is possible to train seismic images to create a model that can identify faults in seismic data. In this article, we will walk you through a Deep Learning Framework that can predict faults from seismic data. With advances in Deep Neural Network technology, it may be possible to train seismic images to create a model that can identify faults in seismic data. For this study, we used synthetic seismic data provided by Force Competition. The Ichthys Seismic dataset is provided courtesy of Geoscience Australia and is available under a CC BY 4.0 license.

It has a very well-expressed polygonal fault system in the overburden, which is locally intersected by larger planar faults in the survey's eastern part, and a deeper faulted section in the Jurassic that is dominated by more diffuse but human mappable fault zones.

To employ a synthetic model trained algorithm that accurately maps deep and shallow faulting and approximates human interpretation.

**Data Exploration**

This study's data is in the SEG-Y format, which is the industry standard for seismic data (SEG stands for Society of Exploration Geophysicists). This is a specialized data format, and we used a Python library called 'segyio' that was designed to read it. Because we are working with a 3D dataset, the segyio can easily convert it to a 3D NumPy array. A 3D grid structure example is shown below, along with three planar surfaces: inline, crossline, and z-slice. These are the 2D surfaces that are commonly used in the seismic industry to visualize data.

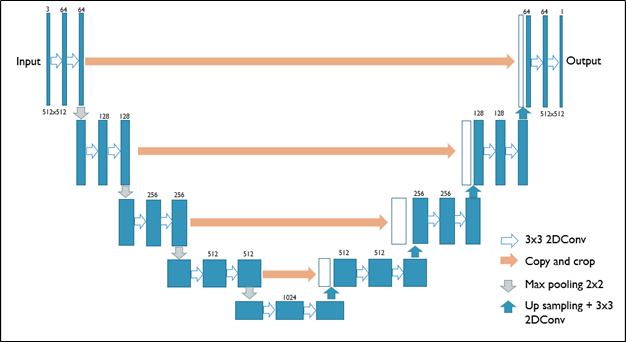


*Figure 8: A 3D seismic grid with an example of a 2D seismic display along the Inline direction with Fault overlay*

**Methodology**

In this project, we have used Convolutional Neural Network (CNN), which is a computationally efficient model with special convolution and pooling operations for analyzing 3D Seismic images. The fault mapping task is a type of image segmentation task. A U-Net framework is appropriate for this type of task because we want the neural network's output to be a full resolution image.

A general schematic of the U-Net framework is shown below:



*Figure 9: U-Net Framework*

The U-Net is divided into two distinct flow paths:

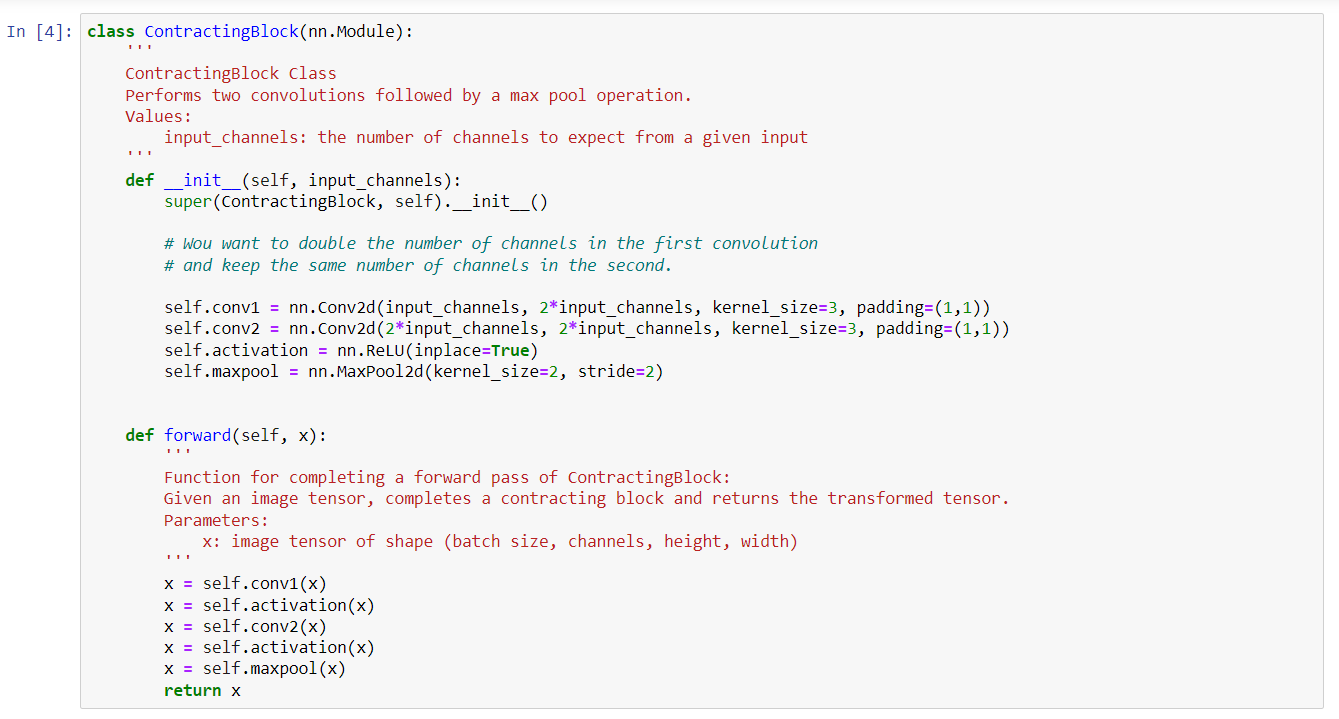
1. a forward contraction path involving several downsampling steps.

This is the U-Net encoder section, which includes two 3x3 convolutions, a ReLU, a 2x2 max pooling, and a stride of 2 for downsampling. The original U-Net implementation employs 'unpadded' convolutions, which result in smaller final output size.

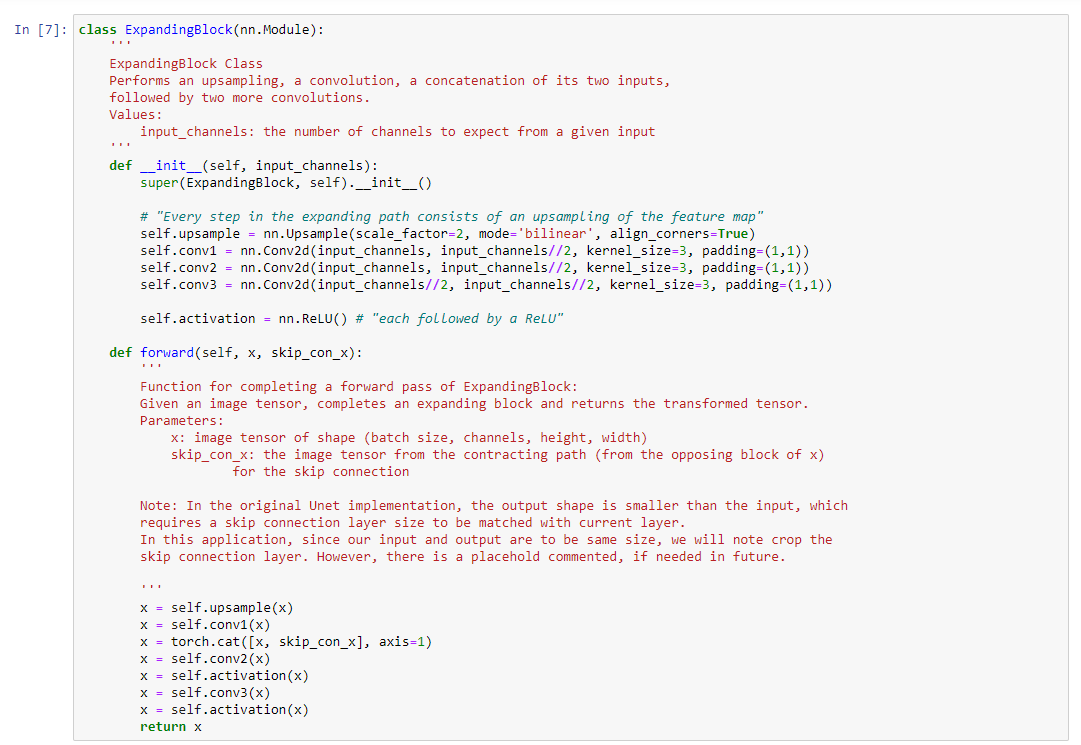
1. a path of expansion in reverse, involving several upsampling steps.

This is the decoder section, which includes an upsampling of the feature map, a 3x3 convolution, and a concatenation of a feature map from the previous contracting block, followed by three 3x3 convolutions with ReLU activation.

The code snippets for these two steps are below:



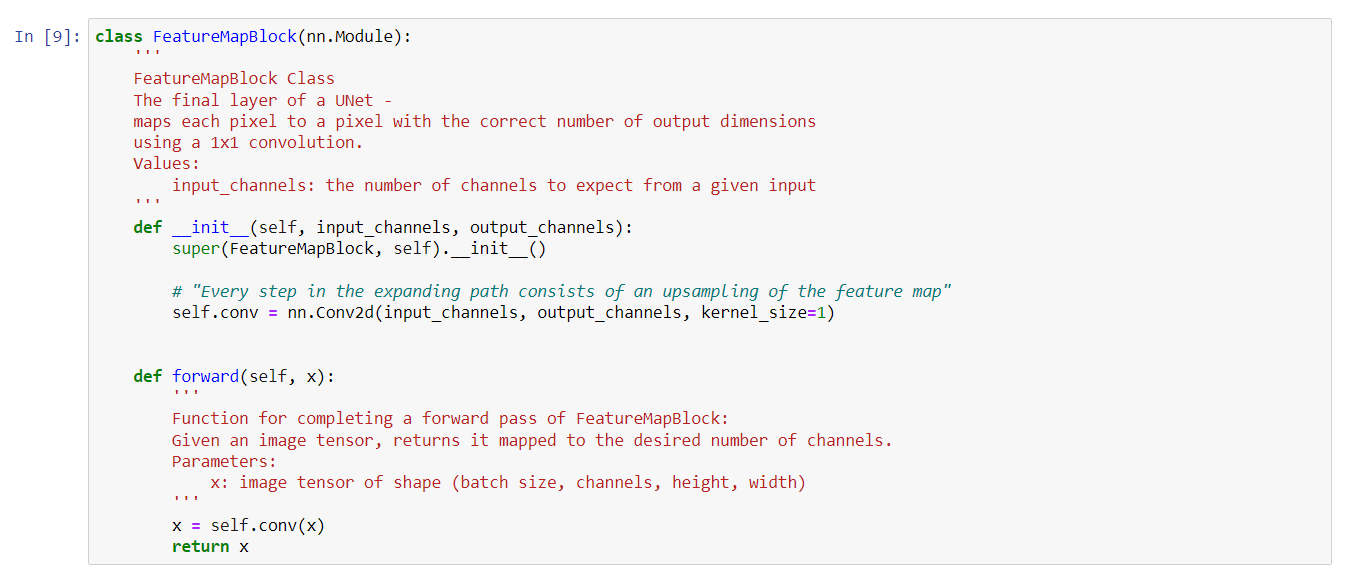
*Figure 10: Contracting Block Architecture*



*Figure11: Expanding Block Architecture*

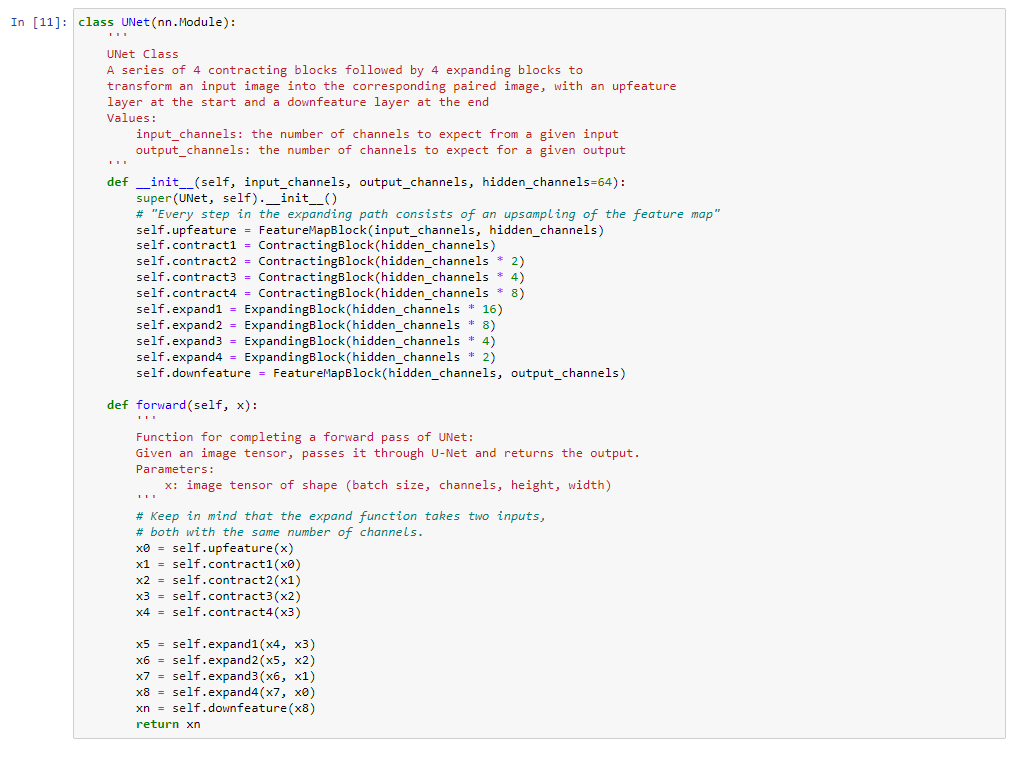
In the original U-Net implementation, the output shape is smaller than the input, necessitating the use of a skip connection layer size that corresponds to the current layer. The skip-connection layer should be cropped to match the size of the layer after upsampling and convolution in this case. In addition, the padding must be set to 1 to ensure that the final output shape matches the input shape.

Next, we need a final code block that will output tensors with the same input size.



*Figure 12: Feature Map Block*

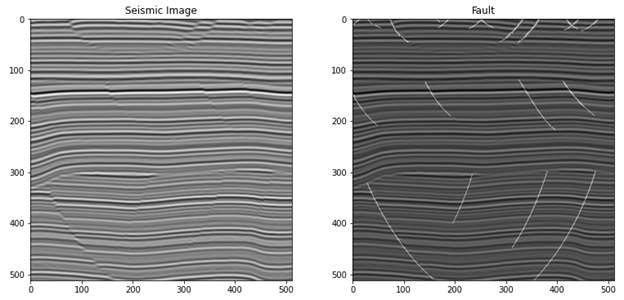
In total, the final U-Net block has four contraction and four expansion blocks, as well as a feature map block at the network's beginning and end.



*Figure 13: UNet Architecture*

**Training Data**

For this study, we will use two volumes of data: a seismic cube and a fault cube, with the seismic cube serving as training data and the fault cube serving as label data. The fault data consists of a series of manually mapped surfaces with values between 0 and 1. The image on the left shows a 2D seismic display in the inline direction, while the image on the right shows the same display with faults overlaid.



*Figure 14: A seismic example with fault overlay*

*As we can see the left image has the actual appearance of faults compared to the right.*

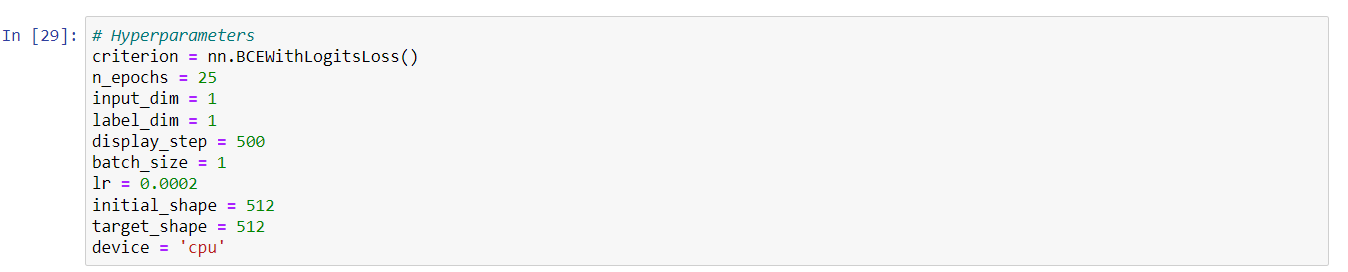
The original input data provided has the following shape: (101, 589, 751). There are 101 inlines, 589 crosslines, and 751 samples in total. Because the seismic images are not RGB, they can be treated as single-channel grayscale images. The number of inlines can be considered the batch size, and along with the inline, we will have a 2D image with the dimensions 589 x 751 pixels. The original input data provided has the following shape: (101, 589, 751). There are 101 inlines, 589 crosslines, and 751 samples in total. Because the seismic images are not RGB, they can be treated as single-channel grayscale images. The number of inlines can be considered the batch size, and along with the inline, we will have a 2D image with the dimensions 589 x 751 pixels.

The desired size of the input tensor is: (batch size, channels, height, width).

As a result, our input tensors have the form (101, 1, 589, 751), where 1 denotes a single channel. However, the odd number size initially caused some issues, prompting me to crop the input volume to obtain a shape of (101, 1, 512, 512)

**Model Training**

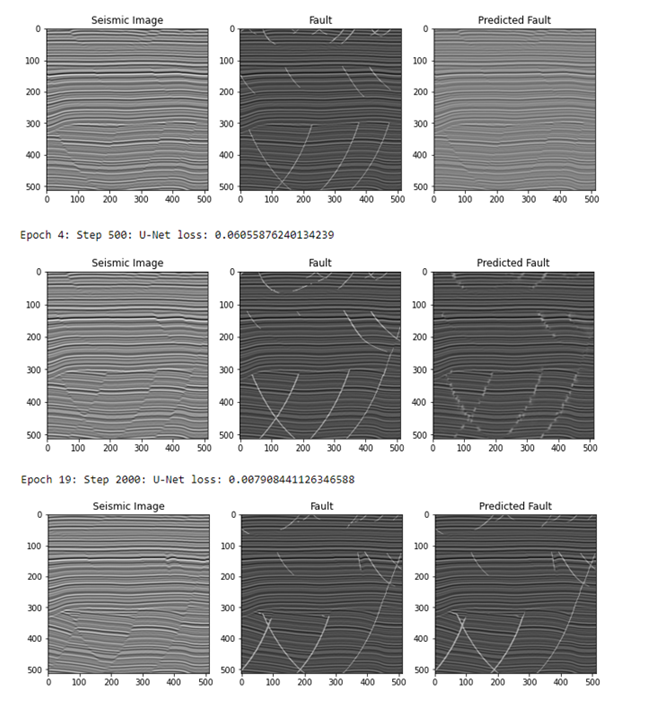
The following are the general training parameters. We have trained the model with 25 epochs, but we'll see that for this dataset, the model begins to pick out faults very effectively at around 10 epochs. We ran the model on an NVIDIA GeForce RTX 2060 Super with 8GB memory, one batch at a time.



*Figure 15: Hyperparameters Used*

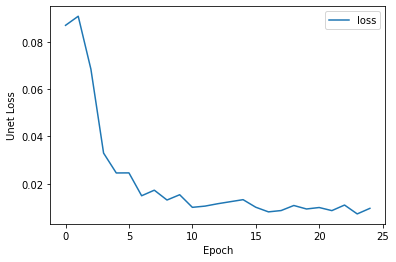
**Result**

The images below were gathered at random at three different epochs. In Epoch 4, the model is already beginning to detect faults, and by Epoch 19, it has successfully mapped the fault.



*Figure 16: Result Faults*

When we look at the model performance, we can see that the loss function drops dramatically within the first 5 epochs and then stabilizes around the 15th epoch. This rapid decrease in model loss is most likely due to the use of clean synthetic data for training.



*Figure 17: Plot between UNet Loss vs Epoch*

**Conclusion and Recommendations**

This small experiment demonstrated the power of deep learning and how it can be used to map faults relatively quickly when the input data is noise-free. However, in the real world, seismic data are very messy and full of noises, which can significantly impair model accuracy. If we train the model with a wide variety of seismic datasets from various basins all over the world, the model can generalize quite well.

**Acknowledgments**

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals. I would like to extend my sincere thanks to all of them. It has been a great honor and privilege to present a project for this course. We are highly indebted to Dr. Mohamad for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing the project. His constant guidance and willingness to share his vast knowledge made us understand this project and its manifestations in great depths ad helped us to complete the assigned tasks on time. I would like to express my gratitude to my parents and friends for their kind support and encouragement which helped us in the completion of this project. My thanks and appreciations also go to my colleagues in developing the project and people who have willingly helped us out with their abilities. It helped us to increase our knowledge and skills.

**Nomenclature**

* **CNN:** Convolutional Neural Network
* **SEG-Y:** Society of Exploration Geophysicists
* **Planar surfaces**: inline, crossline & z-slice
* **Architecture:** U-Net
* **ReLU:** Rectified Linear Activation Unit
* **Concatenation:** Increases the precision of learning

**References**

1. <https://github.com/sgautam666/CNN-Based-Seismic-Fault-Prediction>
2. <https://github.com/bolgebrygg/Force-2020-Machine-Learning-competition>
3. <https://www.linkedin.com/pulse/force-fault-mapping-competition-exploring-data-ben-lasscock/?trk=public_profile_article_view>
4. <https://towardsdatascience.com/seismic-fault-prediction-with-deep-learning-2935704c9b48>
5. [https://www.sciencedirect.com/topics/engineering/convolutional-neural-network#:~:text=Convolutional%20neural%20network%20(CNN)%20is,related%20problems%20by%20analyzing%20images.](https://www.sciencedirect.com/topics/engineering/convolutional-neural-network%23:~:text=Convolutional%20neural%20network%20(CNN)%20is,related%20problems%20by%20analyzing%20images.)
6. [https://library.seg.org/doi/10.1190/geo2017-0666.1#:~:text=We%20formulate%20the%20seismic%20fault,result%20for%20the%20central%20point.](https://library.seg.org/doi/10.1190/geo2017-0666.1%23:~:text=We%20formulate%20the%20seismic%20fault,result%20for%20the%20central%20point.)
7. <https://doi.org/10.3390/en14123650>
8. <https://doi.org/10.1093/jge/gxy015>
9. <https://doi.org/10.1016/j.cageo.2021.104776>
10. <http://www.xeek.ai>
11. <https://github.com/bolgebrygg/Force-2020-Machine-Learning-competition/tree/master/fault_mapping_competition>
12. <https://www.evaluateenergy.com>
13. <https://nextcloud.drgaff.net/index.php/s/7mXn8mSbjxdZyr8>

**Appendix**

1. Abstract ………………………………………………………………......…..……… 1
2. Introduction …………………………………………………………………...…… 1-2
3. Business Problem ………………………………………………………….….…… 3-4
4. Literature Review ……………………………………………………….….……… 5-6
5. Data ……………………………………………………………………………...… 7-8
6. Methodology …………………………………………………………...………… 8-12
7. Training the Model ……………………………………………………...……… 12-14
8. Result …………………………………………………………………………… 14-15
9. Conclusion ………………………………………………..………………………… 16
10. Acknowledgement …………………………………………………………..……… 16
11. Nomenclature ……………………………………………………………….……… 17
12. Reference …………………………………………………………………………… 17
13. Appendix …………………………………………………………………...…… 17-18