Machine learning of lineaments

from magnetic, gravity and elevation maps

by

Mohammad Amin Aghaee Rad

B.Sc., Sharif University of Technology, 2016

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

February 2019

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, a thesis/dissertation entitled:

Machine Learning of Lineaments from Magnetic, Gravity and Elevation Maps

submitted by Mohammad Amin Aghaee Rad in partial fulfillment of the requirements for the degree of Master of Science

in Computer Science

Examining Committee:

David Poole, Computer Science

Supervisor

Jim Little, Computer Science

Supervisory Committee Member

Abstract

Minerals exploration is becoming more difficult, particularly because most mineral deposits at the surface of the earth have been found. While there may be a lot of sensing data, there is a shortage of expertise to interpret that data. This thesis aims to bring some of the recent advances in AI to the interpretation of sensing data. Our AI model learns one-dimensional features (lineaments) from two-dimensional data (in particular, magnetics surveys, maps of gravity and digital elevation maps), which surprisingly has not had a great deal of attention (whereas getting two-dimensional or zero-dimensional features is very common). We define a convolutional neural network to predict the probability that a lineament passes through each location on the map. Then, using these probabilities, cluster analysis, and regression models, we develop a post-processing method to predict lineaments. We train and evaluate our model on large real-world datasets in BC and Australia.

Lay Summary

Finding geological linear features (e.g. lineaments or faults) is important for mineral exploration. Building on advances of artificial intelligence in the last few years, this thesis explores how to find such linear features automatically. The input is aerial surveys of magnetism, gravity and elevation. An expert has interpreted some maps also with lineaments, and the aim is to predict lineaments in different parts of the world. This work is evaluated on parts of BC and Australia.

Preface

This thesis is largely based on collaborative work between UBC and Minerva Intelligence Inc. I collaborated with Professor David Poole on the UBC team, and Clinton Smyth and Dave Farrow on the Minerva team. During completion of this work, we had a great deal of active discussion among both teams. Knowledge of machine learning algorithms and their implementations were provided by the UBC team, and real-world datasets, as well as expert knowledge about geology and mineral exploration, was provided by the Minerva team. Below is a description of specific contributions of each member in this project.

Professor Poole, my academic supervisor and supervisor of this work, introduced this research topic to me. I consulted with him on various aspects of my work. We had meetings at UBC and Minerva discussing my progress, analyzing the obtained results, and deciding on the next steps. He guided me through all the steps, making sure I was on the right track.

Mr. Smyth, CEO of the Minerva Intelligence Inc, introduced this problem to us and provided us with dataset, expert knowledge, and intuitions on why some ideas may work or not work on their dataset. He participated in meetings with Minerva team in their company.

Mr. Farrow, a technical advisor at Minerva Intelligence Inc, met with us approximately every week. In these meetings, I presented the progress of the work and shared the results. We investigated the results and they provided helpful thoughts and feedback. After I prepared the content of this thesis and wrote up its first draft, Professor Poole gave me his helpful feedbacks.

Table of Contents

List of Tables

List of Figures

Acknowledgements

First of all, I would like to thank to my supervisor Professor David Poole for his guidance, great support and kind advice through Master research studies. It was a real privilege and an honor for me to share of his knowledge but also of his extraordinary human qualities.

My acknowledgment also goes to Minerva team members, Clinton Smyth, Dave Farrow, Chris Ahern, Todd Ballantyne and Sam Cantor.

Last but not least, my deep and sincere gratitude to my parents and my sisters for their continuous love, and support.

This thesis is dedicated to my parents.

For their endless love, support and encouragement.

Chapter 1: Introduction

Resource extraction, including mineral exploration and mining, is an important part of Canada's economy. There is currently a great interest in finding the minerals that are needed for batteries, such as cobalt and nickel. Arguably the lack of minerals is the biggest impediment to cutting the price of electric vehicles and weaning society off fossil fuels.

One important problem in mineral exploration is finding various geological features. Some geological features play crucial role in understanding mineral deposits and guide mineral explorers by providing predictive mineral maps. In research, published in science journal Ore Geology Reviews, Ying Chao Liu et al. [1] found that all major gold deposits are controlled by geological faults, but small fault systems are more likely to lead to gold than larger ones.

Recognizing geological linear structures such as bedrock fault and shear zones, thrust faults, fractures, lithological contacts and fold structures is essential when exploring for mineral deposits. In order to find such structures on Earth, geologists use remote sensing methods. The term "remote sensing" refers to the process of getting information about the Earth from sensors on satellites, drones or other aircraft. [2]. Many remote sensing techniques and solutions for producing an interpreted map from aerial photographs in geology been studied [3] [4] [5]. Remote sensing data is undergoing an explosive growth and traditional remote sensing processing techniques are not fast enough to process all of data [6]. In a recent study, Ma et al. [6] review the analysis of remote sensing big data and related challenges using current methods.

There are a few semi-automated methods which have a good performance on remote sensing and aerial feature extraction. Middleton et al. [7], for example, proposed a method to detect deformed zones on a regional scale from remotely sensed magnetic and digital elevation datasets. In another study [8], a semi-automated-algorithm using image and signal processing techniques to find land cover change detection is suggested. However, these methods are not accurate enough to be used directly. Geology experts still need to find, merge and modify the results of these methods based on their expert knowledge to generate an interpreted map as an output. An interpreted map is a two-dimensional aerial image or three-dimensional map of some region's surface in which some features are indicated.

Artificial intelligence has great potential in many applications. Among various AI methods, machine learning and deep learning are used in many applications, such as fraud detection [9], face recognition [10], spam filtering [10], stock trading [12], text categorization and some aspects of minerals exploration. Recently, machine learning techniques have been used for detecting important features on satellite images and for classifying land covers from remote sensing datasets [9]. In this thesis we use deep learning for training a neural network model to predict existence of lineaments (one dimensional feature, such as faults) from aerial images.

Chapter 2: Background and Related Works

In this chapter, we provide background information for a reader to understand the rest of the thesis. We also describe algorithms and models used in the rest of this thesis.

2.1 Background in Geology

In this section, we have a quick review on geological definitions that are necessary to understand the datasets we use to train our machine learning model. Most of these definitions are extracted from a report by Sven Tirén [10]:

- **Topography** is "the configuration of the curves of any surface" [10], or simply speaking, it refers to the shape of Earth's surface.
- **Remote sensing** refers to the process of getting information about the Earth from sensors on satellites, drones or other aircraft.
- A **model** of something is a representation that can be used to make predictions about that thing. In particular, in this work are trying to train a neural network model to predict existence of geological features on parts of the Earth.
- A **lineament** is a one-dimensional feature on the surface of the Earth. For example, faults are lineaments, and the edge of a river is a lineament. We will model lineaments as splines.
- **Aerial images** are images taken from an aircraft or other flying objects. These images provide two-dimensional or three-dimensional information from various modalities (e.g. light sensors, magnetic sensors).

Geologists and geophysics find lineaments of a region by looking at aerial images of that region collected from remote sensing such as magnetic map, gravity map, etc. In this work, we use the same images geologists use to find lineaments. Although we have used these images as input black boxes for training our model, it might be helpful to briefly understand what each layer corresponds to. The types of maps used in this dissertation include the following:

- A **magnetic map** measures spatial changes in the magnetic field in an area. These aerial maps are gathered from a magnetic sensor installed on a helicopter.
- An **RTP map** is a processed magnetic map taking into account the elevation of aircraft and the location of the North and South poles. Reduction to the pole (**RTP**) is a standard magnetic data processing method.
- A **first vertical derivate** (**1vd**) quantifies the change in a signal as a function of survey height. 1VD maps are used to enhance weaker and less visible signals. As well as the RTP map, we use the first vertical derivative of the RTP map.
- A **gravity map** presents gravity anomalies. A gravity anomaly refers to changes between the observed gravity on the surface of the Earth.
- A **digital elevation map** is two-dimensional map of height on surface of the Earth.

2.2 Machine Learning

Machine learning is a field of artificial intelligence (AI) that uses statistical methods to give computer systems the ability to be trained from data. Due to the fact that systems can learn from data, identify patterns and makes decisions themselves, machine learning is considered as a branch of artificial intelligence. Nowadays, banks and other businesses in the financial industry use

machine learning to prevent frauds [11]; oil and mineral companies use machine learning for the detection of oil spills in satellite radar images [12].

There are various machine learning methods adopted for different purposes. Deep learning, supervised and unsupervised learning such as regression, clustering, and classification are some of the main approaches. We mainly use three approaches in this thesis. Firstly, we use artificial neural networks to build and train a model for making predictions on the existence of lineaments in each location on the map. Secondly, we use clustering methods to convert our predictions to lineaments. Finally, we use polynomial regressors to convert clusters into polynomials and curves. More details are discussed further in chapter 3.

2.2.1 Deep Learning and Neural Networks

Deep Learning is a branch of machine learning. Artificial neural networks are inspired by the structure and function of the brain (for instance neurons and synapses). Deep learning refers to artificial neural networks that have more than one hidden layer between the input and output layer. In Figure 1, you can see a sample neural network with two hidden layers, three input units in the input layer, and two output units in the output layer.

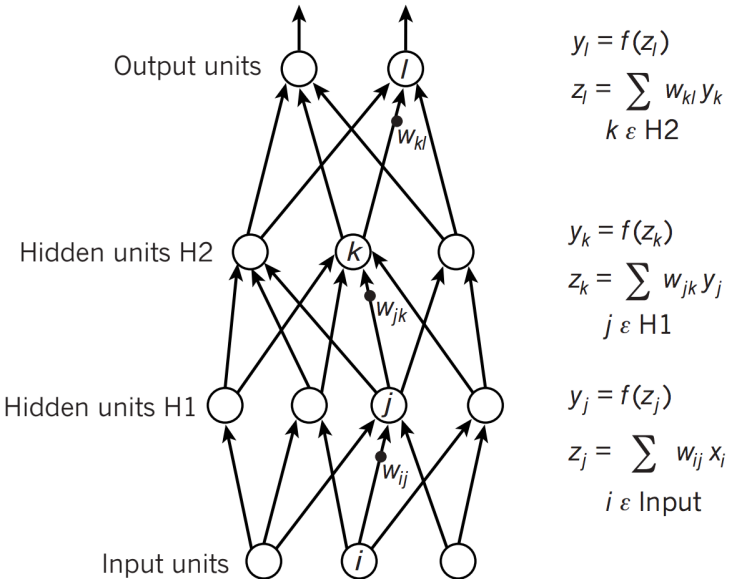

Figure 1 Sample Neural Network taken from [13]

Today, deep learning is used in many applications. Some applications are automatic colorization of black and white images [14] [15], automatic text generation [16] [17], object classification and detection in photograph [18] [19] [20], and remote sensing to classify crop types [21]. In this work, we also inspired by a convolutional network for biomedical image segmentation called U-Net [22].

2.2.1.1 U-Net

The U-Net is convolutional network architecture (i.e. artificial neural network with convolutional layers) for segmentation of images [22]. Their target is to convert a raw medical image in input layer into a segmentation map (with the same size of input) in the output layer.

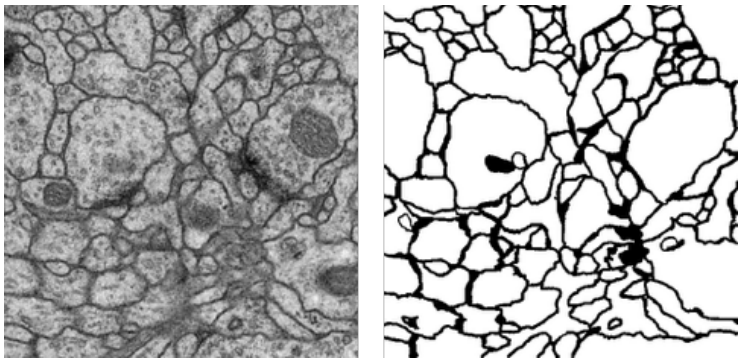

Figure 2 Input and output of U-Net [22]

They proposed combining the features of different layers to calculate the final segmentation directly at the lowest layer using an 11-layer u-shaped network architecture called U-Net which was designed to leverage both high- and low-level features. You can see their architecture in Figure 3:

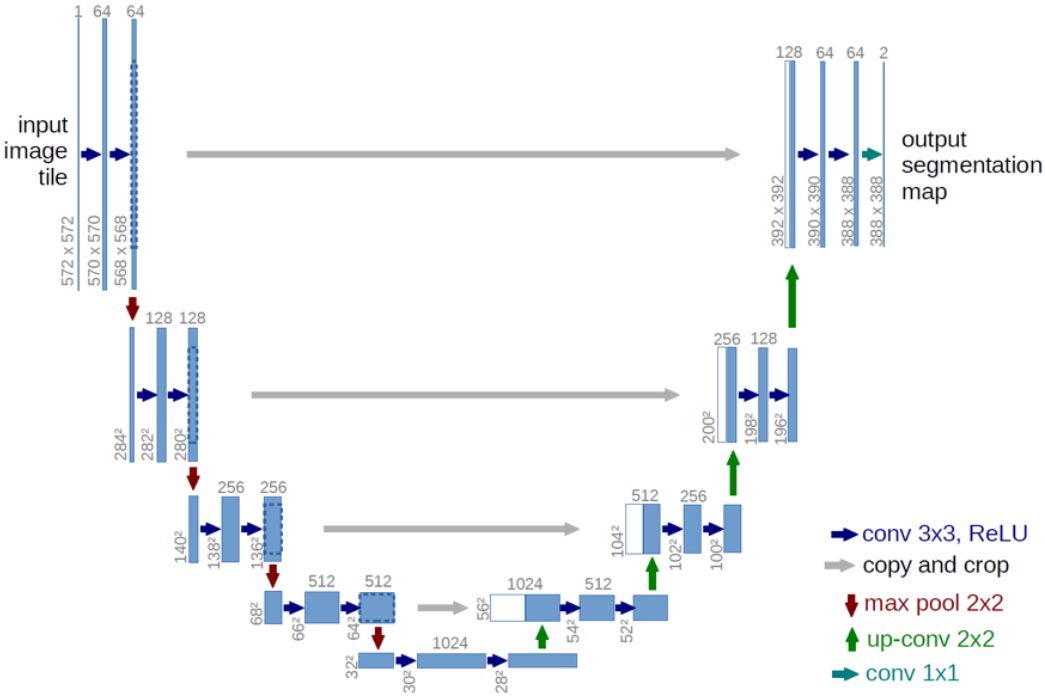

Figure 3 U-Net architecture taken from [22]

There are some similarities between their segmentation task and our problem:

- The output layer is a 2D image with the same size of the input layer.
- The output is a binary (0 or 1) map in which 1 indicates existence of some feature. In the original application of a U-Net, features are boundaries between different cell segments.

One big challenge in training artificial neural networks, is obtaining enough data. A U-Net is a convolutional neural network that needs lots of data to be trained. One well-known solution to this problem, is data augmentation. Data augmentation involves feeding the neural net with extra training data and merging it with the original training set. In U-Net, they designed an augmentation method in their pre-processing which they call random elastic deformation. They use random elastic deformation to generate more training samples by adding small changes to the training data. They also break large images into smaller patches and feed the neural net with those small patches. Initially we added some modifications to U-Net and trained it with our dataset. However, the results were not promising due to many differences between U-Net and the problem of finding lineaments in images. A non-expert person can easily find boundaries (mainly darker pixels) and generate more training samples by looking at the input. In our problem, lineaments are not necessarily visible in the input images. They are interpreted by geology experts. We used basic ideas about data generation and the neural network structure used in U-Net and then designed our own structure. More details about our structure and differences between U-Net and our problem will be discussed in chapter 3.

2.2.2 Clustering

Data clustering algorithms try to find clusters of objects so that objects in one cluster are more similar (in some sense) to each other than to those in other clusters. The similarity measure could be spatial distance, connectivity, or any other function. There are different categorizations for clustering algorithms such as density-based, centroid-based and distribution-based methods [23]. DBSCAN [24] is one of the most common data clustering algorithms that groups together data points that are closely packed together and ignores outliers in low-density areas.

2.2.2.1 DBSCAN

Density-based spatial clustering of applications with noise (DBSCAN [24]) is a density-based clustering algorithm. Given a set of points in some space, it groups together points that are closely packed together (uses spatial distance as similarity function). It marks those points that lie alone in low-density regions as outliers (whose nearest neighbors are not close enough).

DBSCAN is chosen in this paper mainly because of the following two reasons:

1) DBSCAN does not require one to specify the number of clusters before starting, as opposed to other clustering methods such as k-means.
2) DBSCAN can find arbitrarily shaped clusters (linear shapes, circular shapes and other curves). It can also find a cluster completely surrounded by a different cluster

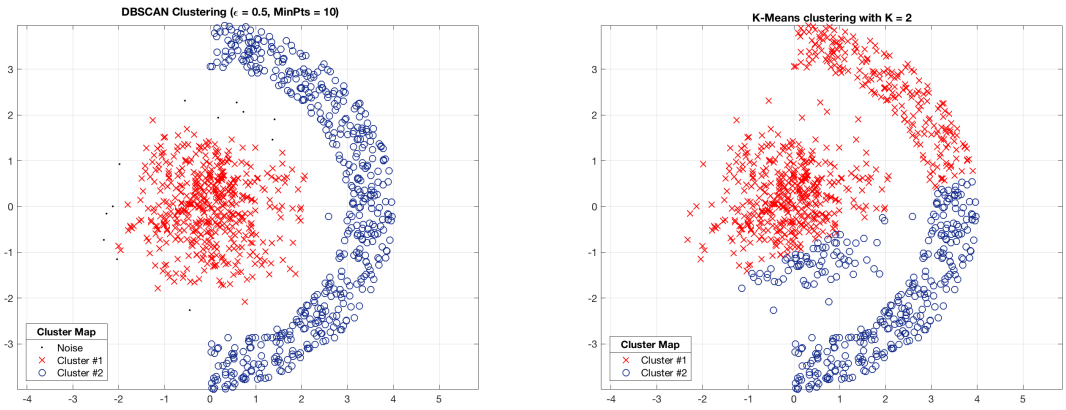

Figure 4 Comparing DBSCAN with another famous clustering method (K-means). Both methods were run over the same dataset in MATLAB.

Now we describe a density-based algorithm for discovering clusters in large spatial databases with noise or DBSCAN [24]. Consider the following definitions, where the algorithm has two parameters $MinPts$ and $\boldsymbol{\mathcal{E}}$:

- $d(p,q)$ is **spatial distance** between two data points p, q. The most common distance metric used is Euclidean distance.
- The $\boldsymbol{\mathcal{E}}$-neighborhood of a point $\boldsymbol{p}$, denoted by $N_{\mathcal{E}}(p) = \{q \in D \mid d(p,q) < \mathcal{E}\}$
- Point $\boldsymbol{p}$ is a **core point** if it has $MinPts$ points is its $\mathcal{E}$-neighborhood.
- A point $\boldsymbol{p}$ is directly reachable from point $\boldsymbol{q}$ with parameters $\boldsymbol{\mathcal{E}}$ and $MinPts$ if:
  - $p \in N_{\mathcal{E}}(q)$
  - $MinPts < |N_{\mathcal{E}}(q)|$
- A point $\boldsymbol{q}$ is reachable from $\boldsymbol{p}$ if there is a path $\boldsymbol{p_1}, \ldots, \boldsymbol{p_n}$ (starting from $\boldsymbol{p}$ and ending up in $\boldsymbol{q}$) where each $\boldsymbol{p_{i+1}}$is **directly reachable** from $\boldsymbol{p_i}$.
- A point $\boldsymbol{p}$ is a **noise point** if it is not reachable from any other point in the dataset, and is not a core point

Given a set of points and $MinPts$ and $\boldsymbol{\mathcal{E}}$ parameters, this algorithm finds core points first (that is it finds all points in dataset which have at least $\boldsymbol{MinPts}$ number of points in their $\boldsymbol{\mathcal{E}} - neigbourhood$. It finds all points reachable from each core points. The goal is to assign all reachable points to the same cluster. You can find more details about this algorithm in [24]. In the next chapter, it is explained how we modify this method to convert predictions in a probability map into clusters.

2.2.3 Polynomial Regression

In regression problems, the goal is to construct a function to predict values for a continuous output variable. Formally, given a set of variables $X(x_1, x_2, \dots, x_n) \in \mathbb{R}^n$ as input and a continuous output variable Y, the target is to find a function $f(X)$ that predicts a value close to Y. This mapping could be linear:

$$\boldsymbol{y = \beta_0 + \beta_1 x}$$

or it could be anything else such as a polynomial of degree m:

$$\boldsymbol{y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_m x^m}$$

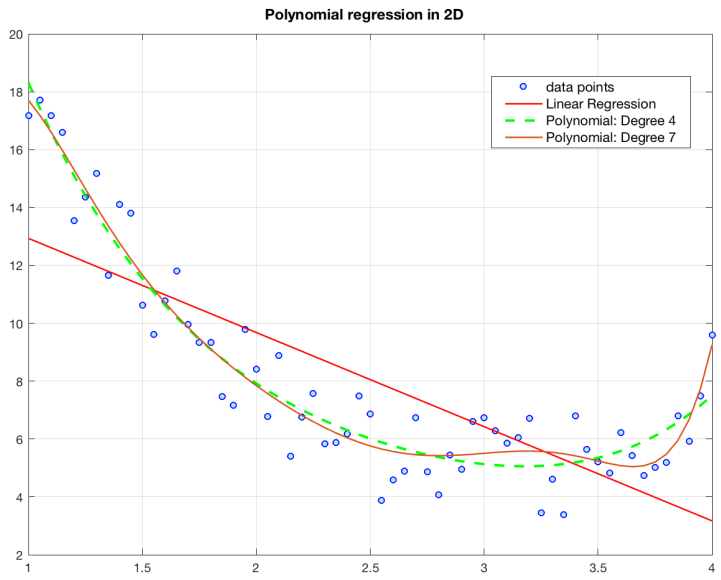

Figure 5 Comparing polynomial regression of degree 1,4,7

For n data points (i.e. number of dataset) and degree m, the polynomial regression problem can be expressed in form of matrix in terms of a matrix $\boldsymbol{X}$, a response vector $\vec{y}$, a parameter vector $\vec{\beta}$. The matrix of equations is:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ 1 & x_3 & x_3^2 & & x_3^m \\ & \vdots & & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^m \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \vdots \\ \beta_m \end{bmatrix}$$

Using the ordinary least squares estimation and having the precondition $m < n$, a vector of estimated polynomial coefficients (i.e. solution of this problem) is:

$$\hat{\vec{\beta}} = (X^T X)^{-1} X^T \vec{y}$$

In chapter 3, we use polynomial regression to convert set of points in clusters into polynomials.

Chapter 3: Lineament Learning Model

Given a set of aerial images of some part of the Earth, the problem is to find lineaments on that part of the Earth. Aerial images including magnetic maps, gravity maps and digital elevation maps are inputs of our model. Table 1 lists all input images we use in the training procedure. A map of lineaments interpreted by an expert is also given and we consider it as our model's desired output. We want to use these images and labels to train a model to predict lineaments in other areas in the world. We divide input images and interpreted image into a training set and a test set. We train our model using the training set. Then our model makes predictions on the test set and we compare the predictions with the interpreted lineaments.

In this chapter we provide information on how we develop our lineament predictor model, what problems each part of the model solves, and how each part use other parts' outputs or provide inputs for other parts.

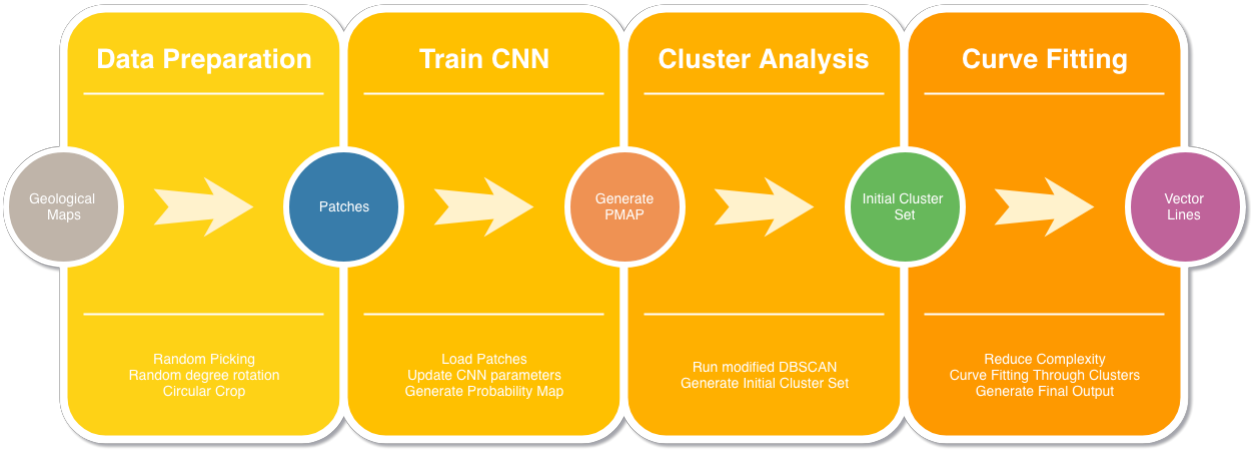

Figure 6 Workflow of Lineament Learning Model

3.1 Data Preparation

In order to train our model, we use eight aerial images as our inputs layer. These are the same images that geologists use to label lineaments on maps based on their experience. These images

are listed in Table 1. Each pixel in these images represent a 50-meter by 50-meter square on the Earth.

Table 1 List of aerial images as input layer

Image 1	RTP map	Reduction to the pole map
Image 2	1VD of RTP map	First vertical derivative of RTP map
Image 3	Digital Elevation map	Map of height on surface of the Earth
Image 4	Isostatic gravity map	A map that presents gravity anomalies
Image 5	RTP HGM map	RTP map of high-resolution magnetic field of a region
Image 6	RTP HGM_RS	Processed-by-geologist version of RTP, HGM_RS refers to residual shallow horizontal gradient
Image 7	RTP HGM_RD	Processed-by-geologist version of RTP, HGM_RD refers to residual deep horizontal gradient
Image 8	RTP HGM_RI	Processed-by-geologist version of RTP, HGM_RI refers to residual intermediate horizontal gradient

Background information about these layers are briefly provided in Section 2.1. In our work, we want to use exactly the same images that geologists use to find lineaments. These input image can be considered as input black boxes provided by geologists.

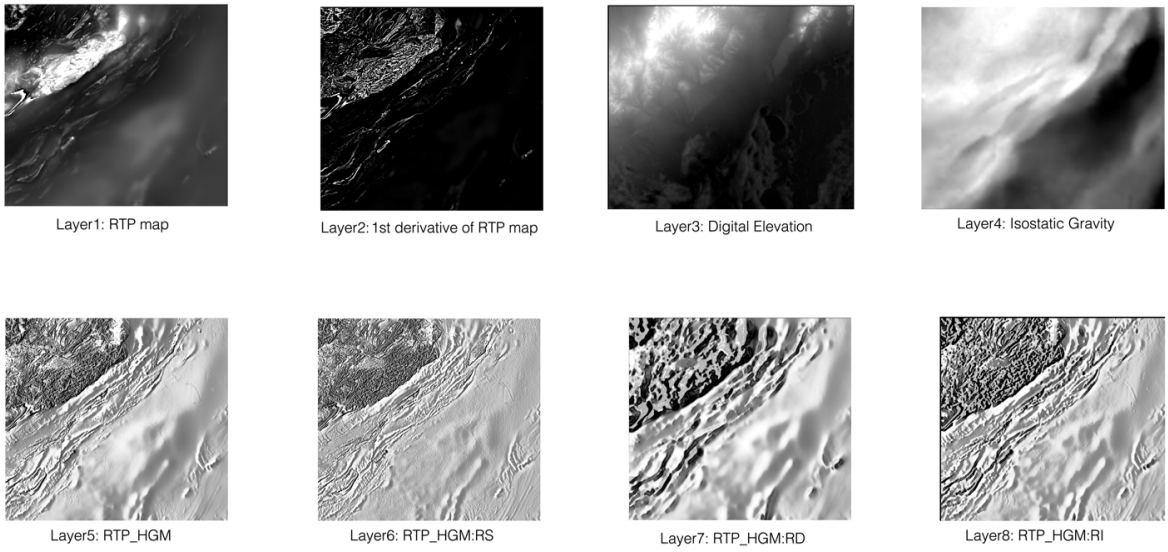

Figure 7 Input images for training from Australia Dataset

We also have an interpreted-by-expert lineament image. This image is a vector file in which lines were drawn by experts as lineaments. We convert the interpreted-by-expert vector image into a black and white image. In Figure 8 Interpreted lineaments (expected output) on the Australia Dataset is illustrated.

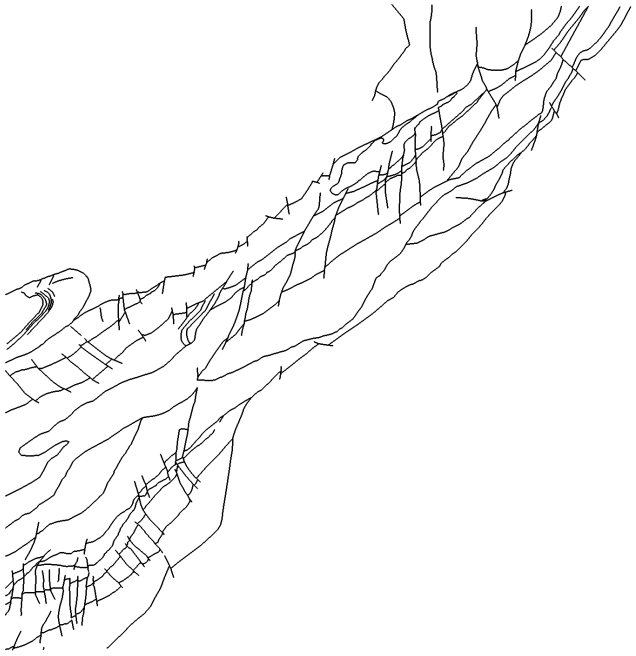

Figure 8 Interpreted lineaments (expected output) on the Australia Dataset

A typical use of convolutional networks is to recognize visual patterns directly from pixel images and to predict a label for those images. For instance, labels could be bicycle, cat, dog, car, … or a combination of those labels. C. Szegedy et al. [25] propose a deep convolutional neural network architecture to detect objects in images that achieves the new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14) [26]. In our model, the desired output should include localization which is an assignation of a prediction (the probability that a lineament is close the pixel) to each pixel of the image.

Thousands of training images are typically required to properly train a neural network. This amount is difficult to achieve in geology (because acquiring data is expensive; input images are

obtained from satellites or regional flying and the output map requires interpretation by experts). We generate many training samples out of the few input images we are given. As Ciresan et al. [27] suggest, we train our model using a sliding-window to generate more training samples and to make our model **translation invariant**. Our model predicts a label (the probability that a lineament is close the pixel) for each pixel by providing a neighborhood of the pixel. We define these local regions as **patches** which are $W \times W \times 8$ matrices where W is patch size (because we have eight types of input images).

In order to generate even more samples to train our model, and to make our model **rotation-invariant** (since aerial images could be captured from flying on any direction) we rotate every patch by r radians where $r \in Uniform(0, 2\pi)$. A target value or label for each patch derives from the center pixel of that patch. We want to have patches that are independent of rotations. So, we apply a circular mask filter to each patch as it is demonstrated in figure below.

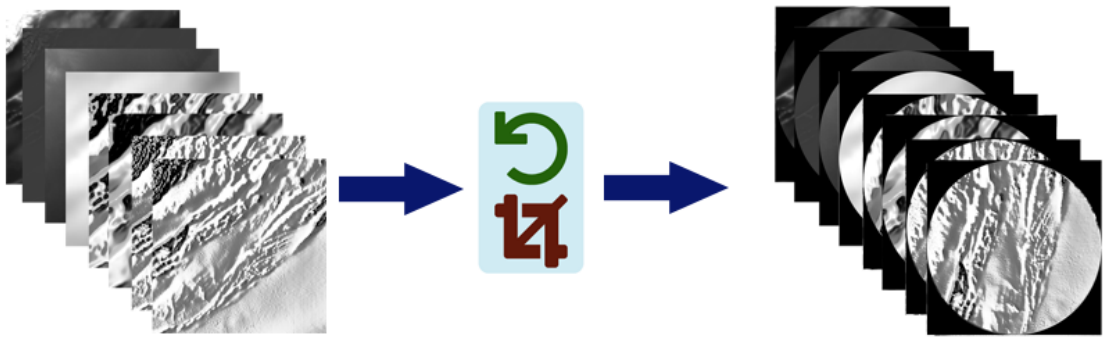

Figure 9 Rotation and Crop filters over every patch

Each patch in original dataset has a label y that corresponds to whether the center pixel of the patch has a drawn-by-expert lineament passing through it:

$$p = \begin{cases} 1: & A\ lineament\ drawn\ by\ expert\ passes\ through\ the\ center\ point \\ 0: & No\ lineaments\ drawn\ by\ expert\ passes\ through\ the\ center\ point \end{cases}$$

Treating lineaments as 1-pixel wide lines does not work well for a number of reasons. Some areas in images could be noisy or could have not enough clue for experts to consistently draw a lineament. It is still acceptable to mineral explorers who use these interpretations if the model predicts the existence of lineaments with a few hundred meters accuracy. Accordingly, we expand interpreted lines using a probability distribution around the lines and convert the binary map into a probability map. The distribution should have a high probability at the center of the lines and low probabilities in farther pixels. We use a Gaussian distribution with σ, μ parameters for expanding lineaments. with minimum value of τ (values below this cutoff threshold will be zero). So, labels p are now a real number $0 \leq p \leq 1$. We have implemented this process inside a **Expand** function that has three parameters:

1. The interpreted-by-expert map of lineaments
2. The distribution function which is a Gaussian function **G(σ, μ)** where **σ=1**
3. A cutoff threshold τ

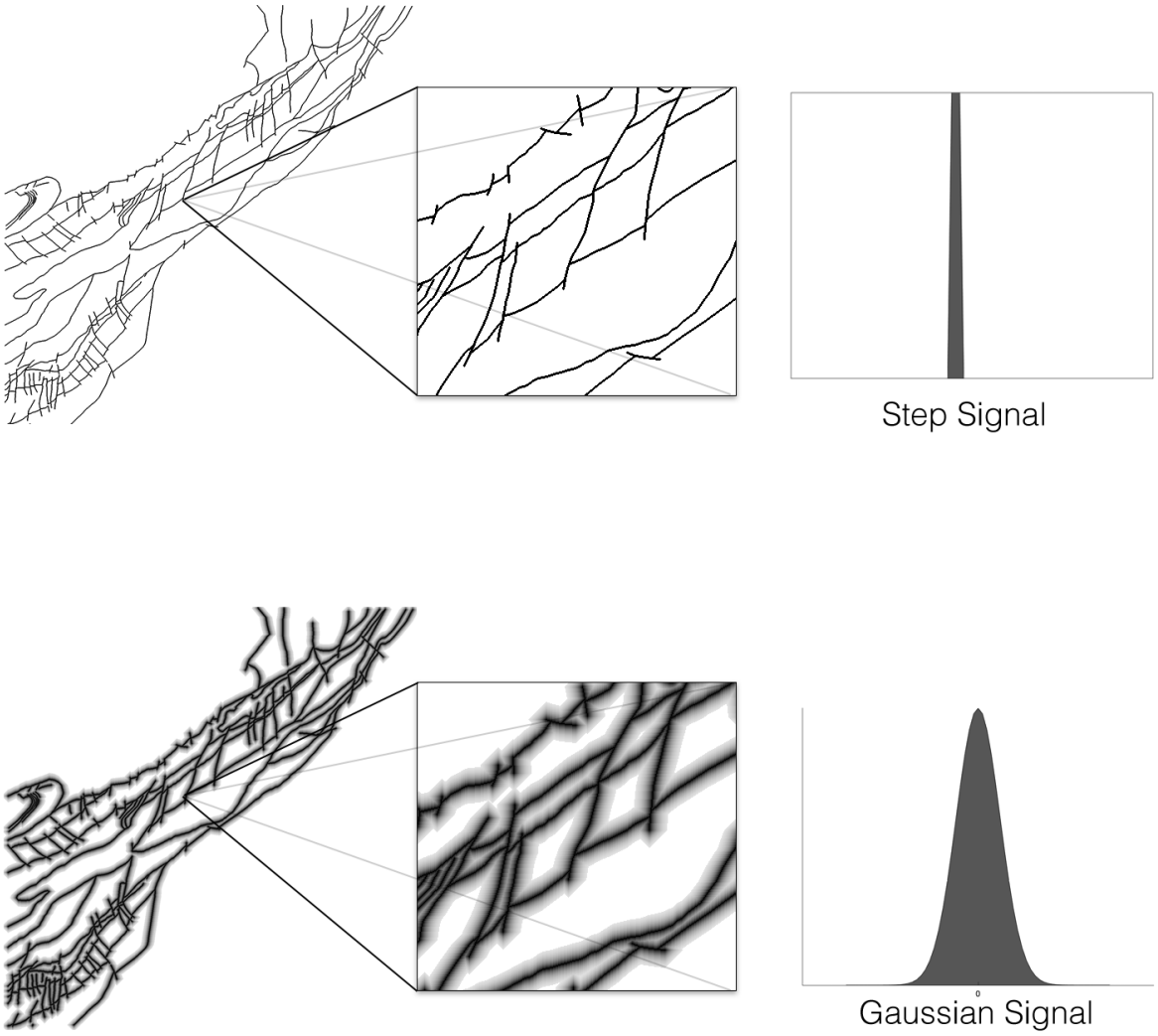

Figure 10 Comparing expanded lineaments using Gaussian Signal with interpreted map

After expanding lineaments using the Gaussian distribution, a sliding-window moves randomly on the map and generates the training samples. Here is the workflow for preparing dataset with a given window size W:

Data Preparation (W = Window size, I = Input images $D_1 \times D_2 \times 8$, O = interpreted map):

$\breve{E}_{D_1 \times D_2}$ = Expand (O, G(σ=1, μ), τ)

Repeat:

Choose (i, j) randomly from $\left[\frac{W}{2}, D_1 - \frac{W}{2}\right] \times [\frac{W}{2}, D_2 - \frac{W}{2}]$

Patch $X_{W \times W \times 8}$ = extract $W \times W$ windows from I around (i, j)

Rotate X by r radians where $r \in Uniform(0, 2\pi)$

Crop X circularly

Label $y = \breve{E}(i, j)$

Generate $[X_{W \times W \times 8}, y]$

In this method the sliding window moves randomly on $\left[\frac{W}{2}, D_1 - \frac{W}{2}\right] \times [\frac{W}{2}, D_2 - \frac{W}{2}]$ to avoid hitting the boundaries. At each iteration, this method generates a pair $[X_{W \times W \times 8}, y]$. The first element in that pair includes data from the processed input images and the second element is the desired output.

3.2 Train Convolutional Neural Network

During this work, we tested different neural networks with different structures from very simple models to more complicated ones like U-Net. Initially, we were trying networks with many convolutional layers that apply convolution operations to the inputs, passing results to the next layer. The convolution emulates the response of an individual neuron to visual stimuli. In our experiment, however, we realized that complex neural networks with many convolutional layers are not effective in this problem. This is the main reason why we decided to simplify our model.

The network architecture for windows size $W = 49$ is illustrated in Figure 11. It starts from the input layer on the left which is consist of patches of size $W \times W \times 8$. Then we have a convolution

layer that creates a 3×3 convolution kernel that is convolved with the layer input to produce the output. A rectified linear unit (ReLU) is applied to the outputs of the convolutions.

In order to reduce the dimensionality and to allow generalization in patches, we use 6×6 max pooling operations, which combine the outputs of neuron clusters at one layer into a single neuron in the next layer. We use a flatten layer that reshapes and merges previous hidden layers in the network into a single one-dimensional array. Finally, we use a fully connected neural network with two hidden layers with ReLU activations and one output layer of size one with Sigmoid activation.

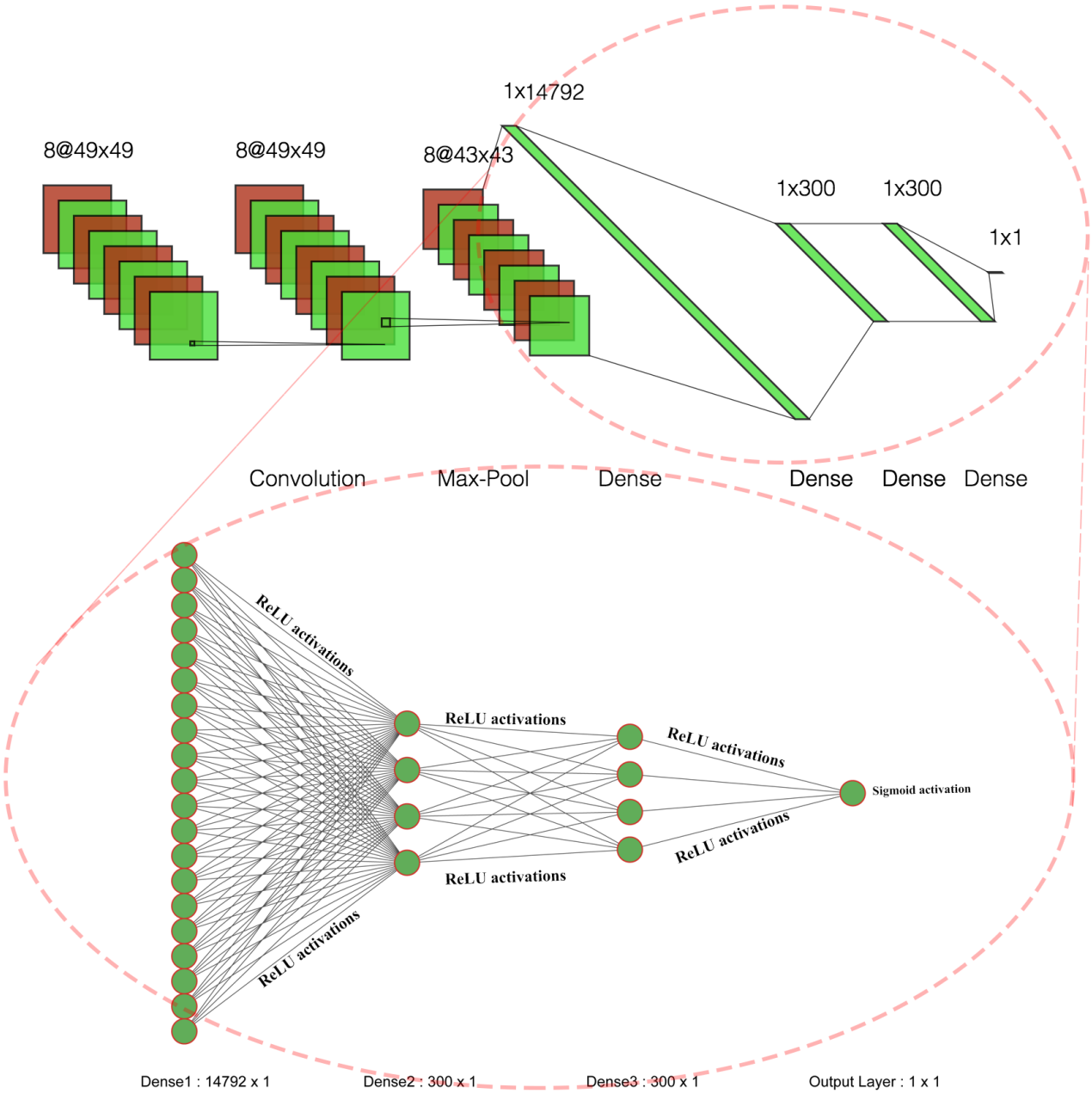

Figure 11 Lineament Learning Architecture

Since our output value is a probability value that corresponds to existence of lineament in the center pixel of input patch, we use cross-entropy or log loss function for training. This loss function measures the performance of a classification model whose output is a probability value between 0

and 1. Cross-entropy loss increases as the predicted probability or $\hat{y}$ diverges from the actual label of y. The cost function is computed by taking the average of all cross-entropies in the sample. For example, suppose we have generated N samples $\{(X_1, y_1), (X_2, y_2), \dots, (X_N, y_N)\}$. The cross-entropy loss function is:

$$Loss = -\frac{1}{N}\sum_{n=1}^{N}[y_n . \log \hat{y}_n + (1 - y_n).\log(1 - \hat{y}_n)]$$

This function reaches to its minimum when $\hat{y} = y$.

After training our convolutional neural network, we can make a probability map or **Pmap** for any region using our trained parameters. We predict values on the valid areas only. All non-valid areas such as non-interpreted areas or out-of-boundaries are **`NaN`** (Not a Number). A Pmap is a two-dimensional map with size $D_1 \times D_2$ (the same size as the input geological maps) where each pixel has a probability value between zero and 1. These values are our model's prediction on the probability of lineament existence on that pixel. A sample Pmap is illustrated in Figure 12 where probabilities are visualized by shading of red where higher values are more red and lower values are blacker. Non-valid areas with label `NaN` are also shown with black. We evaluate the output on valid areas only.

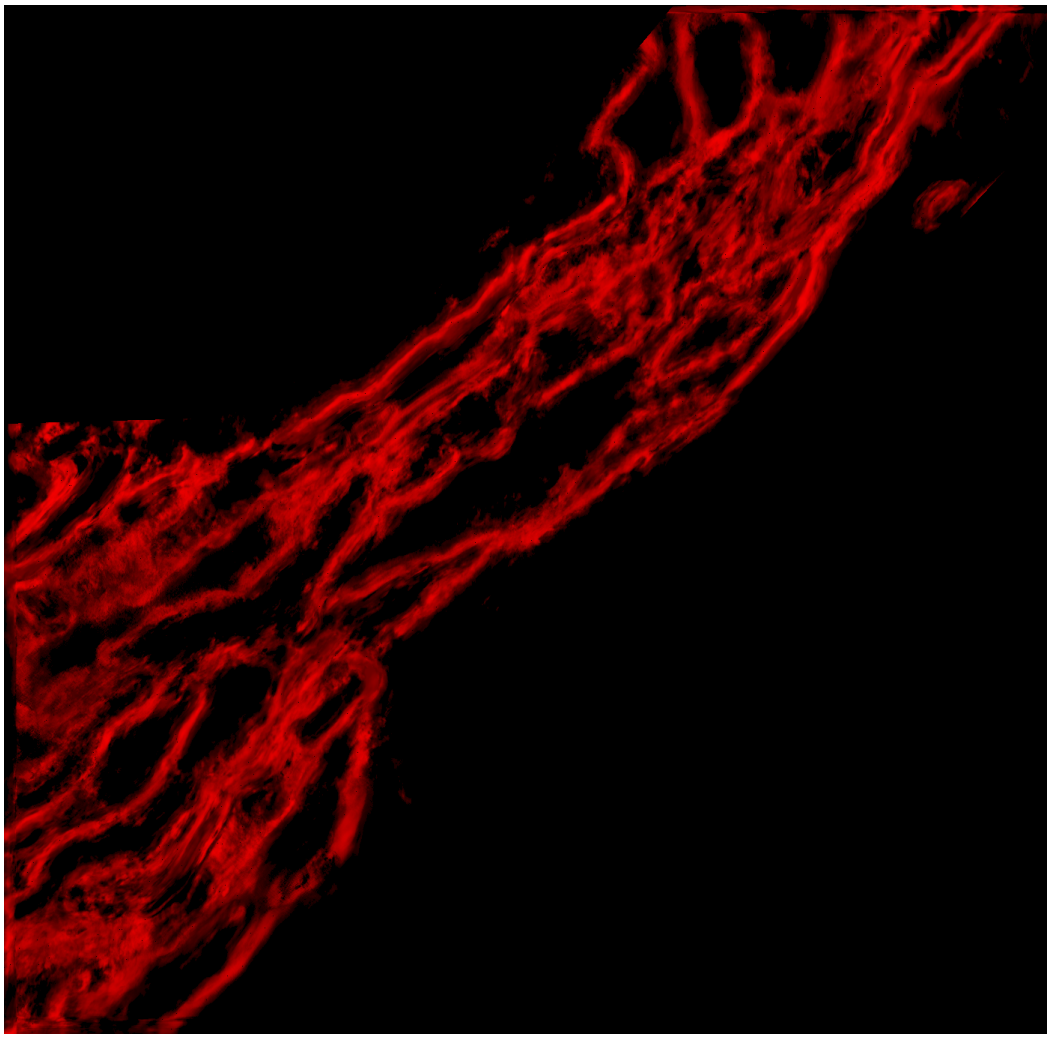

Figure 12 Probability map for the Australia dataset with W=45. Higher probabilities for a lineament are in red.

3.3 Cluster Analysis

The final product of lineament learning should be lines or curves similar to the interpreted map in Figure 8. However, Pmaps are probability maps that have probability of fault existence in every location. In this section and the following section, we discuss our post-processing methods to convert Pmaps into lines and polynomials.

In the previous chapter, the DBSCAN clustering algorithm was briefly discussed. In the original DBSCAN method, a core point are points that have $MinPts$ points in its ϵ-neighbourhood. We add a little modification into the original method. We call it P-DBSCAN or Probability-DBSCAN. In P-DBSCAN, lineament points have probabilities. So, we sum up the probabilities of lineaments points in ϵ-neighbourhood. Figure 13 shows the ϵ-neighbourhood of the blue point in the original DBSCAN on the left and P-DBSCAN on the right.

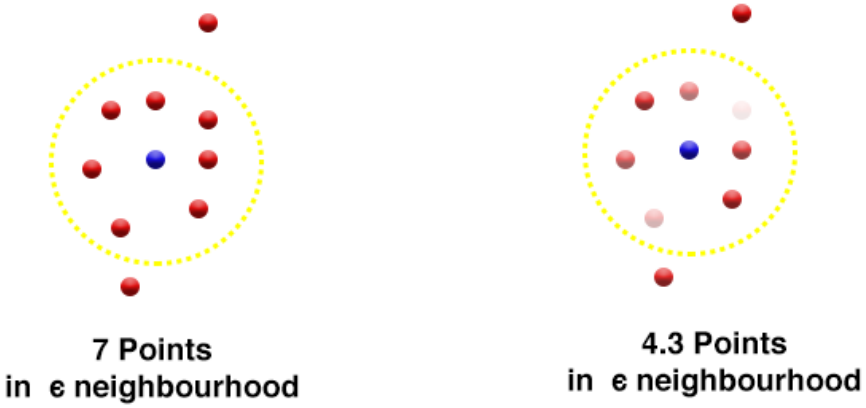

Figure 13 Original DBSCAN versus P-DBSCAN. In P-DBSCAN, we sum up the probabilities of lineaments points in ϵ-neighbourhood.

We give the lineament points distributed on a Pmap to P-DBSCAN. P-DBSCAN is also a clustering algorithm that finds clusters. A sample output with $\epsilon = 3.3$ and $MinPts = 20$ is illustrated in Figure 14. In this figure, each cluster is demonstrated with a random color. DBSCAN removes outliers by assigning no label to them. So, the nebulous and less confident areas in a Pmap are removed after running the clustering step.

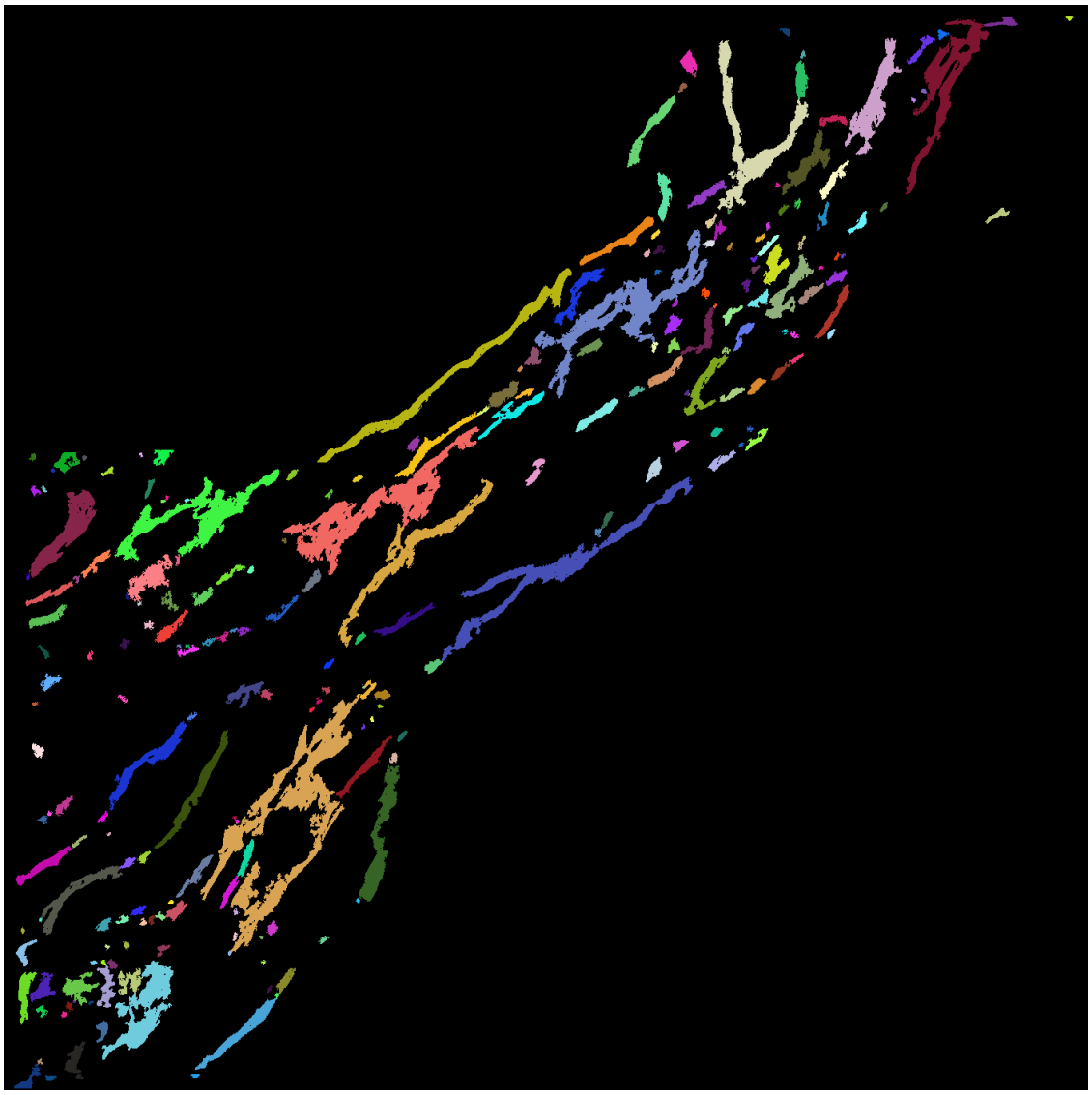

Figure 14 Running P-DBSCAN on Pmap with $\epsilon = 3.3$ and $MinPts = 20$

3.4 Curve Fitting

In the final stage of our method, we convert output clusters of P-DBSCAN into curves and generate vector files as output. Given that each cluster consists of multiple points with two-dimensional features (i.e. (x, y) values in a 2D image), one option is to fit a linear regression model through each cluster and then replace that cluster with the best model. However, this makes the output

complex for the user looking at the results since there are many small curves. In Figure 15, an example of converting two sets of clusters into curves is illustrated. In this figure, at the top there are 15 clusters each of which we want to fit a line. The number of curves could be decreased by merging some of those clusters, as shown in the bottom of the figure.

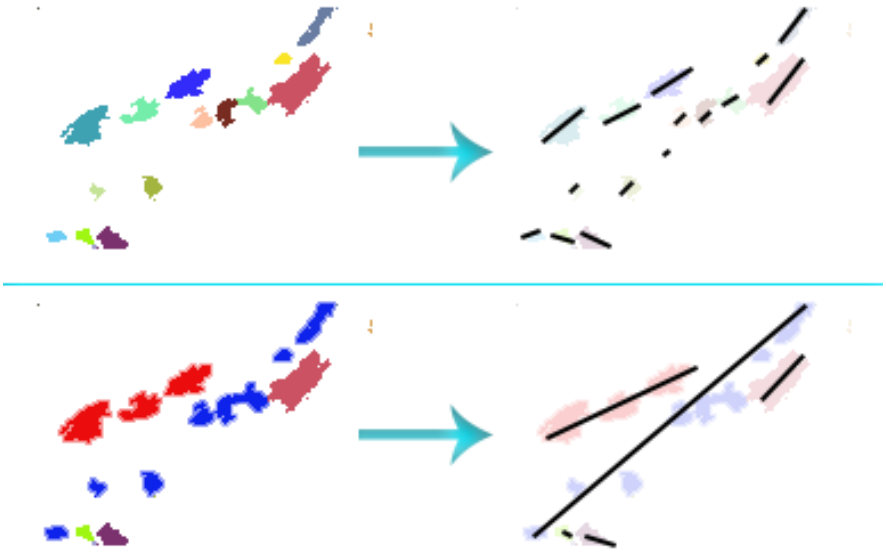

Figure 15 Comparing a model with many clusters versus one with some of the clusters merged

We use the following method to merge clusters so that the total number of curves get reduced:

```
LimitedCurves (Clusters C = {c_1, c_2, …, c_K}, maximum clusters M, L, λ ):
    While |C| > M do:
        Choose C_i from C with the smallest size
        {c'_1, c'_2, …, c'_L} = L closest clusters to C_i
        Best = (c_i ∪ c'_1)
        For c_x ∈ {c'_1, c'_2, …, c'_L}:
            If Error (c_x ∪ c_i , λ) < Error (Best, λ) then:
                Best = (c_i ∪ c_x)
        Merge (Best) and update C;
```

In this method, function Error(cluster) returns mean square error of a best curve passing through a cluster. It fits linear, degree 2, degree 3 to degree 7 polynomials and choose the best curve based on our error function which relies on mean square error. Higher degree polynomials always have lower errors that lower degrees. However, we prefer simpler curves and simpler output. We choose low degree polynomials unless its error is greater than a threshold $\boldsymbol{\lambda}$. We use the following method to find the error of fitting a polynomial:

```
Error (Clusters C, Threshold λ):
    For d = 2 to 7:
        P = fit degree d polynomial to C
        E = mean square error for P
        If E < λ:
            Return E
    Return E
```

Starting from $\boldsymbol{k}$ clusters, this method merges two clusters after each iteration and reduces the total number of clusters until it reaches $\boldsymbol{M}$. The value of $\boldsymbol{M}$ is the desired number of clusters. On each iteration, it chooses the smallest cluster. We prefer removing smaller clusters and having larger clusters through which a single polynomial is fitted. Our method tests $\boldsymbol{L}$ nearby clusters one by one and chooses a cluster with a minimum curve error if those clusters are merged together. This method only tests $\boldsymbol{L}$ nearby clusters to merge nearby clusters in early steps. Then the clusters grow in size and get closer to each other.

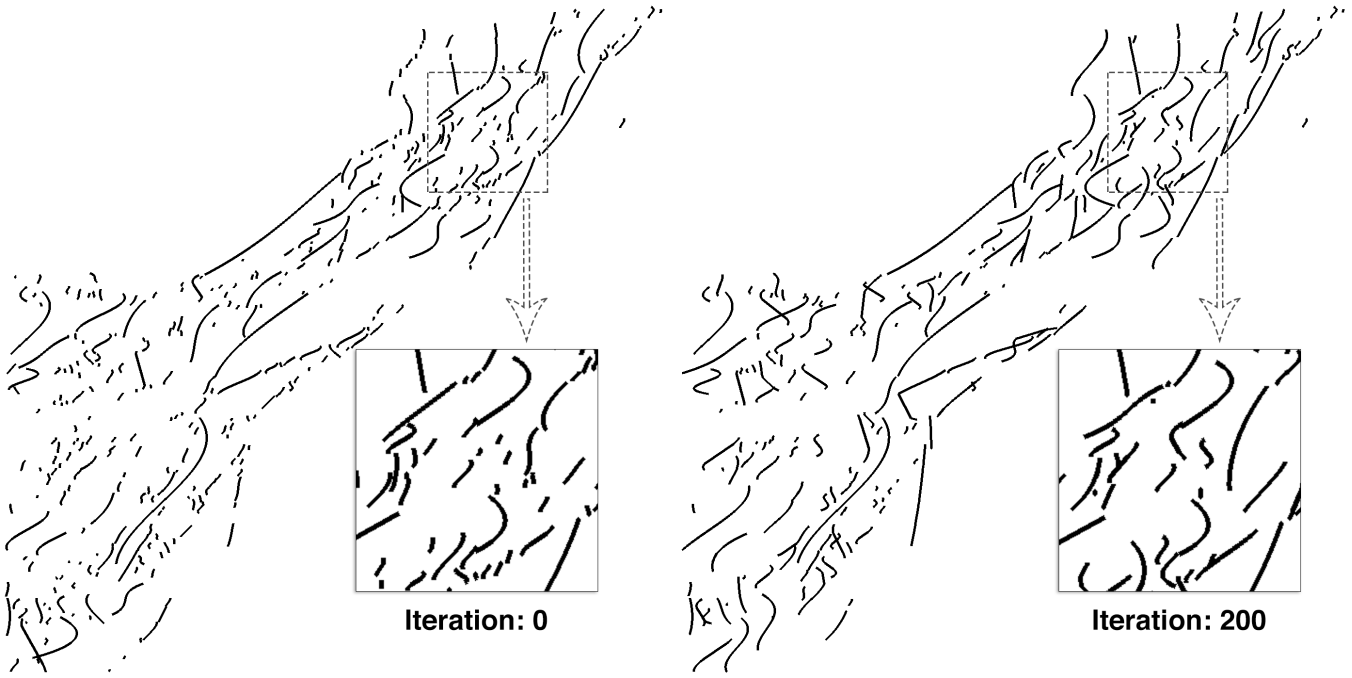

Figure 16 Best Curves method iteration 0 and 200 on Australia dataset

The curves shown in Figure 16 are outputs of our limited curve process initially and after 200 iterations.

Chapter 4: Empirical Results

4.1 Datasets

We trained and tested our model on the real datasets provided by our partner, Minerva Intelligence. Each contains the input 8 layers as presented in 3.1 and the interpreted fault lines. These datasets are:

1) **Australia Dataset**: 106.5×106.8 kilometers or 2130×2136 pixels (each pixel is 50 meters by 50 meters on the ground) from a relatively flat area in Australia.
2) **Quest Dataset**: 253.9×205.8 kilometers or 5078×4116 pixels (each pixel is 50 meters by 50 meters on the ground) from a mountainous region in center of British Columbia, Canada.

Each dataset is a region with a valid mask. The masks are publicly available and usually included in the dataset. We convert these masks into two separate masks for training and testing our model on with 66% for training and 33% for testing. The usual way in learning IID (independent and identically distributed) models is to randomly dedicate some part of datasets for testing. However, since the data is not IID, and because we want to test the output predictions in a two-dimensional region, we picked testing areas regionally. The mask is converted into training and testing mask as it is shown in Figure 17.

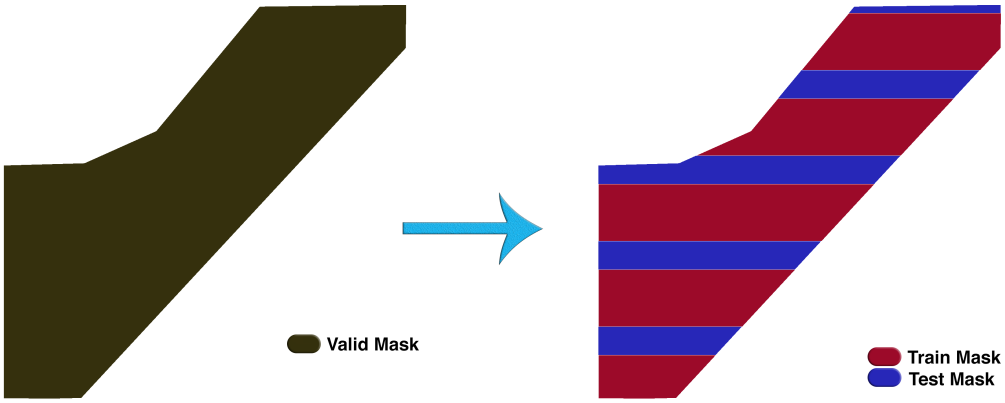

Figure 17 Converting valid mask into training and testing masks. Blue areas show the test set and red area shows the training set. $\frac{W+1}{2}$ wide boundary is set between these areas.

Then we remove lines of widths $\frac{W+1}{2}$ from the boundaries between the testing and training areas. So, the sliding window doesn't use the information in the testing area while it is in the training process. In the training process, we only use patches that are inside the training mask. After training, we generate the probability map on the whole area including both the training and the testing regions. We evaluate the outputs for the training areas, the testing areas and the whole area separately.

4.2 Model Evaluation

Define the following notations:

1) $\boldsymbol{p(x,y)}$ is the desired output for the probability of lineament existence
2) $\widehat{\boldsymbol{p}}(x,y)$ is the predicted value from the model
3) $\boldsymbol{L^+ = \{(x,y): p(x,y) > \tau\}}$
4) $\boldsymbol{L^- = \{(x,y): p(x,y) < \tau\}}$

We define the positive error and the negative error as below:

$$Positive\ Error = \frac{1}{|\boldsymbol{L^+}|} \sum_{(x,y)\in \boldsymbol{L^+}} (p(x,y) - \hat{p}(x,y))^2$$

$$Negative\ Error = \frac{1}{|\boldsymbol{L^-}|} \sum_{(x,y)\in \boldsymbol{L^-}} \hat{p}(x,y)^2$$

Thus, the positive error is mean square error of predictions on area where lineaments exist. The negative error is the mean square error of the predictions that are supposed to be zero. Consider an extreme predictor which always predicts $\widehat{\boldsymbol{p}} = \mathbf{1}$. The negative error computes the error on the regions that the desired output should be very low. Thus $\widehat{\boldsymbol{p}} = \mathbf{1}$ predictor has a very high negative error. Another extreme predictor with $\widehat{\boldsymbol{p}} = \mathbf{0}$ on the other hand, a negative error equal to zero.

4.3 Experiments

Three different types of models are trained:

1) Australia Models: These models are only trained on the Australia dataset

2) Quest Models: These models are only trained on the Quest dataset
3) Mixed Models: These models are trained on a mixed map of both the Quest and the Australia datasets

For data preparation, we use data preparation method discussed in Section 3.1. The mixed models are trained with a combination of Quest and Australia datasets. That means we feed the mixed models with the data samples generated from the both datasets. For each type, we train various network structures with window size starting from $W = 7$ pixels to $W = 51$ pixels in increments of 4. Due to limitations on hardware memory and time limit, we did not train models with larger window sizes.

In Figure 18 the positive error and the negative errors for these variations are illustrated. The train, the test and the total errors are also included for these models.

By looking at evaluation graphs in Figure 18:

1) By looking at the total errors in this figure, we see that they decrease by increasing the window size and converge or reaches to its minimum approximately in range $25 \leq w \leq 41$. For example, Quest model on Quest dataset reaches to its minimum when $w = 35$ but Mixed model on Australia dataset reaches to its minimum when $w = 23$.
2) Models that were trained on the Australia dataset, do not work well on the Quest dataset and vice versa. The Australia on Quest plot shows a high total negative error when total positive error is quite low. The Quest on Australia plot also shows a high total positive error. By looking at the Mixed on Australia and Mixed on Quest plots, we realize that the

Mixed Models (that trained on a combined maps of both datasets), have quite similar performance comparing to Australia on Australia and Quest on Quest models.

3) Models on Australia datasets suffers from an overfitting problem. In the Australia on Australia plot for example, there is a gap between test error and train error when window size is small. Increasing window size helps the model.

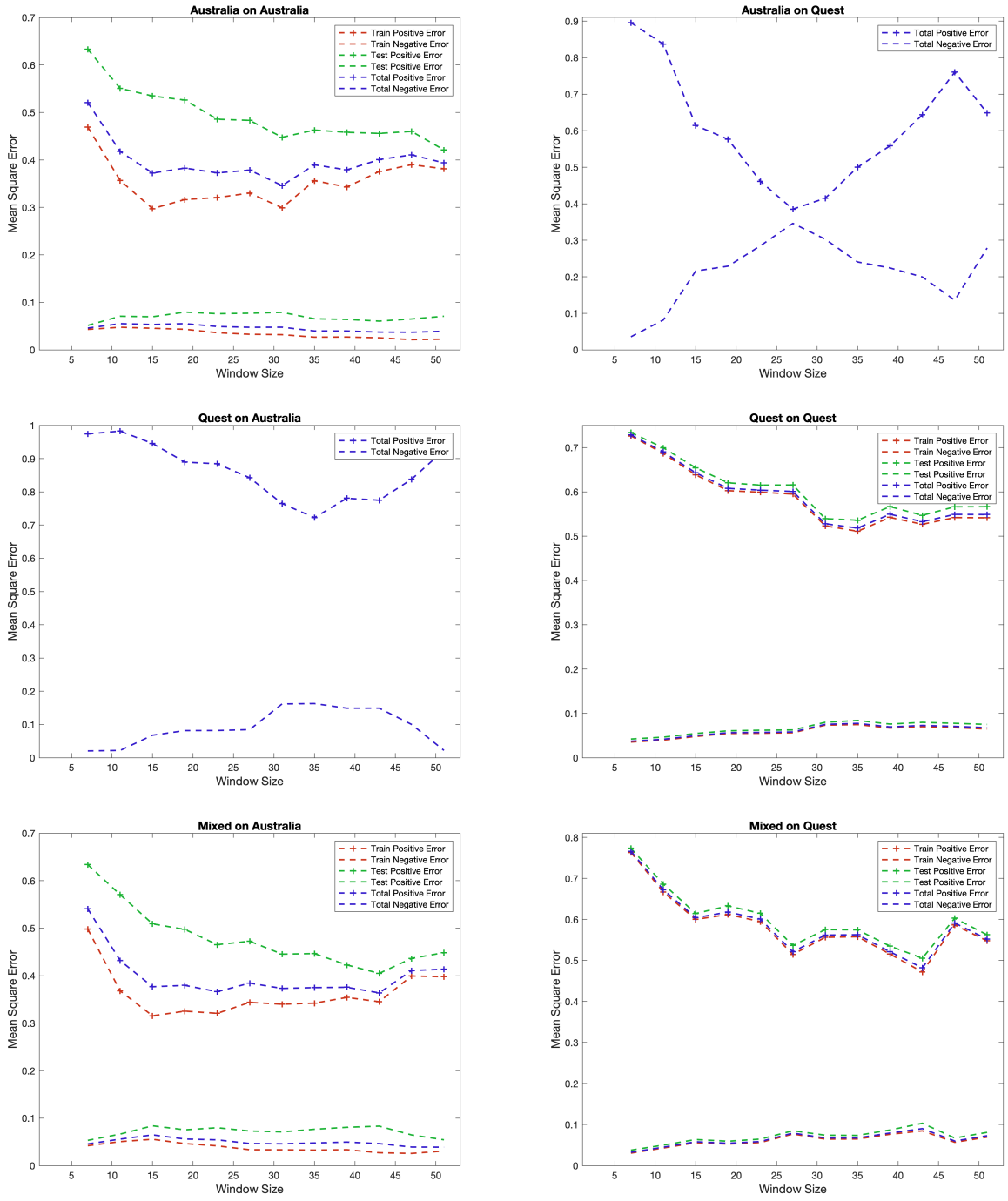

Figure 18 Training, Testing and Total positive and negatives errors for different models

In Figure 18, all plots show quite high positive errors. However, having a reasonable low negative error makes this method work. Because, most areas on the desired output is equal to zero. A model

with a high negative error is not working well on most areas on the map. For example, the Australia model has a high negative error but low positive error on the Quest dataset when is high when $w = 27$.

We have a very low error on dark areas and high errors near the lineaments (the positive error). Our predictions near the lineaments are noisier. The performance of our predictor is very well on negative error. So, we run P-DBSCAN and after that do the regression to find lineaments in the noisy predictions. Figure 19 shows mean square errors for these models on the whole region of the datasets in a single plot. In these plots, we are not separating the positive and negative errors.

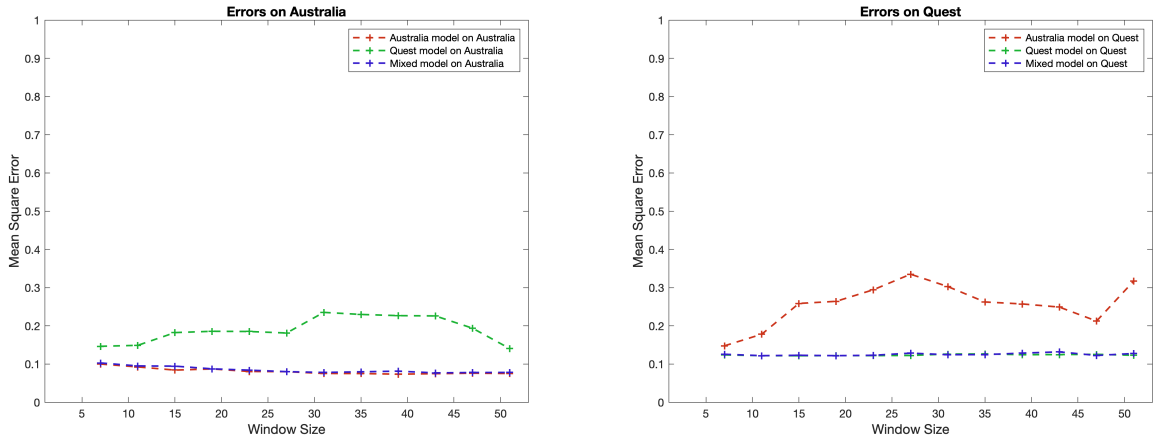

Figure 19 Total errors for the Australia, the Quest and the Mixed models on the whole datasets

Plots in Figure 19 suggest that models trained on the Quest area for example, are not necessarily able to generate a high-quality output the Australia region. Because the error of the Quests models applied on the Australia dataset is greater than the errors for the Australia models on the Australia dataset. This is also true for the Australia models that applied on the Quest dataset. It also suggests that the Mixed models work as well as the Australia models on Australia dataset and the Quest models on the Quest dataset.

4.4 Small Geographic Information System Applet

Although the models could be evaluated using probability maps and error measurements, geologists want to compare output probabilities and lines with interpreted lines visually. They also want to see combinations of input layers underlying our interpretations. We developed a small geographic information system applet in which:

1) User loads both a model and a dataset.
2) User can choose underlying images.
3) User can compare probability map and interpreted lines.
4) User can set a threshold for probabilities.
5) User can run the post-processing methods described in chapter 3.

The models and the applet are available on GitHub[1]. Figure 20 shows some screenshots of the applet in different stages with different setups.

[1] https://github.com/aminrd/LineamentLearning

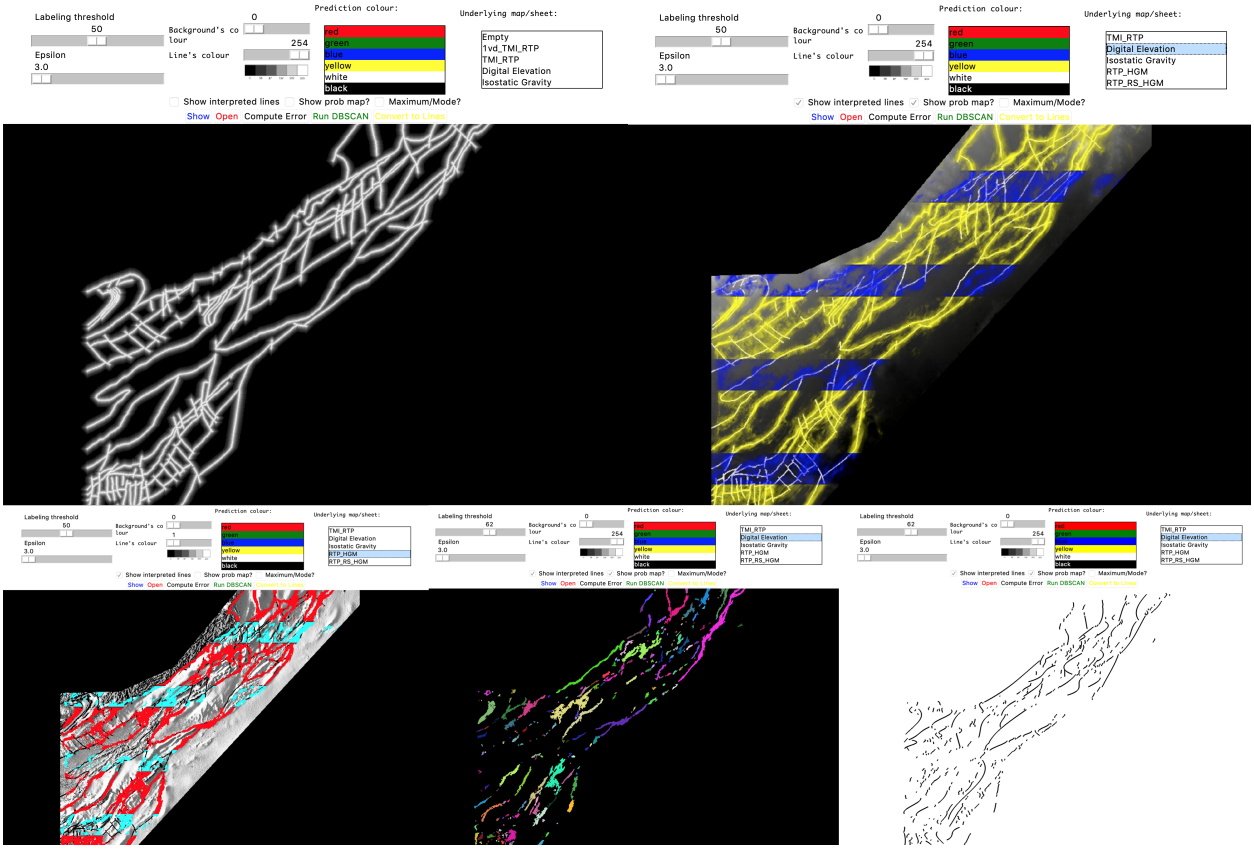

Figure 20 Small GIS applet screenshots

Here is a short review of this applet and its capabilities shown in Figure 20:

1) After opening the applet, the user should open the dataset, the probability map and the trained model. Then applet loads all of them (top-left figure)
2) The labeling threshold is a value between 0 and 100 which is divided by 100 to get a probability. The applet uses this threshold to remove all probabilities below the threshold and only shows values above this threshold. By selecting the "show probability map" checkbox, this threshold is ignored, and the predictions will be more or less transparent based on their probability. User can also decide on the color of probabilities in the training and testing areas (bottom-left figure)

3) By default, the underlying layer is a solid black color that user can change that by choosing options from list of input images or by choosing another solid color (top-right figure)
4) Lines color refers to the color of the interpreted-by-expert lines. User can turn on or off this overlay layer.
5) "Run DBSCAN" button calls our modified DBSCAN method, which is the first step in our post-processing routine. User can set ϵ and $MinPts$ values manually in the applet (middle-down figure)
6) User can convert clusters into polynomials by clicking on the "Convert to Lines" button (bottom-right figure)

Chapter 5: Conclusion

In this thesis we are trying to extract lineaments from aerial images. Magnetic images, digital elevation and gravity maps as well as maps of lineaments interpreted by experts are given.

We trained a convolutional neural network to predict the probability of the existence of a lineament in each location on the map. Then by using these probabilities, clustering analysis, and regression models, we developed a post-processing method to generate an interpretation. The training aims to minimize the binary cross entropy. We defined positive and negative errors for evaluations. We tested our model on real large datasets provided by a mineral exploration company.

In the future, this model is going to be adjusted and implemented in an industrial software for lineament extraction of other areas. Finally, it is worth mentioning that each dataset is interpreted by a geology expert. Personal opinions could affect these interpretations drastically. If we obtain different interpretations by different experts for the same region, we would expect to have a better performance on average. We would like to acquire more interpretations for our dataset and test the models again.

Bibliography

[1] Y. Liu, Z.-X. Li, C. Laukamp, G. West and S. Gardoll, "Quantified spatial relationships between gold mineralisation and key ore genesis controlling factors, and predictive mineralisation mapping, St Ives Goldfield, Western Australia," *Ore Geology Reviews,* vol. 54, pp. 157 - 166, 2013.

[2] L. Ran, Y. Zhang, W. Wei and Q. Zhang, "A hyperspectral image classification framework with spatial pixel pair features," *Sensors,* vol. 17, no. Multidisciplinary Digital Publishing Institute, 2017.

[3] R. G. Ray, "Aerial photographs in geologic interpretation and mapping," U.S. Govt. Print. Off., 1960.

[4] P. D. P. and J. D. Kiser, Aerial Photography and Image Interpretation, 3rd Edition, WILEY, 2012.

[5] T. Lillesand, R. W. Kiefer and J. Chipman, Remote sensing and image interpretation., John Wiley & Sons., 2008.

[6] Y. Ma, H. Wu, L. Wang, B. Huang, R. Ranjan, A. Zomaya and W. Jie, "Remote sensing big data computing: Challenges and opportunities," *Future Generation Computer Systems,* vol. 51, no. 0167-739X, pp. 47-60, 2015.

[7] M. Middleton, T. Schnur, P. Sorjonen-Ward and E. Hyvonen, "Geological lineament interpretation using the object-based image analysis approach: results of semi-automated analyses versus visual interpretation," pp. 135-154, 2015.

[8] Z. Lv, W. Shi, X. Zhou and J. A. Benediktsson, "Semi-Automatic System for Land Cover Change Detection Using Bi-Temporal Remote Sensing Images," *Remote sensing,* vol. 9, no. Multidisciplinary Digital Publishing Institute, p. 1112, 2017.

[9] D. J. Lary, A. H. Alavi, A. H. Gandomi and A. L. Walker, "Machine learning in geosciences and remote sensing," *Geoscience Frontiers,* vol. 7, pp. 3-10, 2016.

[10] S. A. Tirén, "Lineament interpretation Short review and methodology," SSM, 2010.

[11] S. Stolfo, D. W. Fan, W. Lee, A. Prodromidis and P. Chan, "Credit card fraud detection using meta-learning: Issues and initial results," in *AAAI-97 Workshop on Fraud Detection and Risk Management*, 1997.

[12] M. Kubat, R. C. Holte and S. Matwin, "Machine learning for the detection of oil spills in satellite radar images," *Machine Learning,* vol. 30, no. 2, pp. 195-215, 1998.

[13] Y. LeCun, Y. Bengio and G. Hinton, "Deep Learning," *nature,* vol. 521, no. 7553, p. 436, 2015.

[14] R. Zhang, P. Isola and A. A. Efros, "Colorful image colorization," in *European Conference on Computer Vision*, 2016.

[15] M. Richart, J. Visca and J. Baliosian, "Image Colorization with Neural Networks," in *Computer Vision (WVC), 2017 Workshop of*, 2018.

[16] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[17] J. Mao, W. Xu, Y. Yang, J. Wang and A. L. Yuille, "Explain images with multimodal recurrent neural networks," *arXiv preprint arXiv:1410.1090,* 4 October 2014.

[18] C. Szegedy, A. Toshev and D. Erhan, "Deep neural networks for object detection," *Advances in neural information processing systems,* pp. 2553-2561, 2013.

[19] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012.

[20] D. Erhan, C. Szegedy, A. Toshev and D. Anguelov, "Scalable object detection using deep neural networks.," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[21] N. Kussul, M. Lavreniuk, S. Skakun and A. Shelestov, "Deep learning classification of land cover and crop types using remote sensing data," *IEEE Geoscience and Remote Sensing Letters,* vol. 14, no. IEEE, pp. 778-782, 2017.

[22] O. Ronneberger, P. Fischer and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, 2015.

[23] A. Aghaee, M. Ghadiri and M. S. Baghshah, "Active distance-based clustering using K-medoids," *Springer International Publishing,* vol. 9651, pp. 253-264, 2016.

[24] M. Ester, H.-P. Kriegel, J. Sander and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Second International Conference on Knowledge Discovery and Data Mining*, 1996.

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going deeper with convolutions," in *IEEE conference on computer vision and pattern recognition* , 2015.

[26] O. Russakovsky, S. Ma, J. Krause, J. Deng, A. Berg and F.-F. Li, "ImageNet," 2015. [Online]. Available: http://www.image-net.org/challenges/LSVRC/2014/.

[27] D. Ciresan, A. Giusti, L. M. Gambardella and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images," in *Advances in neural information processing systems*, Curran Associates, Inc., 2012, pp. 2843--2851.