A Novel Approach to Automated Coal Petrography Using Deep Neural Networks

Souptik Mukhopadhyay

A Novel Approach to Automated Coal Petrography using Deep Neural Networks

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology
in
Computer Science

by

Souptik Mukhopadhyay
[Roll No: CS-1704]

under the guidance of

Dr. Dipti Prasad Mukherjee
Professor
Electronics and Communication Sciences Unit
Deputy Director
Indian Statistical Institute

Indian Statistical Institute
Kolkata-700108, India

July 2019

To the entire deep learning community

CERTIFICATE

This is to certify that the dissertation entitled **"A Novel Approach to Automated Coal Petrography using Deep Neural Networks"** submitted by **Souptik Mukhopadhyay** to **Indian Statistical Institute, Kolkata**, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science** is a bonafide record of work carried out by him under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.

Dipti Prasad Mukherjee
Professor,
Electronics and Communication Sciences Unit,
Deputy Director,
Indian Statistical Institute,
Kolkata-700108, INDIA.

Acknowledgments

I would like to show my highest gratitude to my advisor, *Prof. Dipti Prasad Mukherjee*, Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata, for his guidance and continuous support and encouragement. He has literally taught me how to do good research, and motivated me with great insights and innovative ideas.

I would also like to thank *Dr. Bhabatosh Chanda*, Professor, Indian Statistical Institute, Kolkata, for his valuable suggestions and discussions.

My deepest thanks to all the teachers of Indian Statistical Institute, for their valuable suggestions and discussions which added an important dimension to my research work. I would like to acknowledge Suman Ghosh, Avishek Shaw and Bikash Santra and all other seniors at the lab for their constant guidance.

Finally, I am very much thankful to my parents and family for their everlasting supports.

Last but not the least, I would like to thank all of my friends for their help and support. I thank all those, whom I have missed out from the above list.

Souptik Mukhopadhyay
Indian Statistical Institute
Kolkata - 700108 , India.

Abstract

This research work is industry sponsored and carried out in collaboration with Tata Steel, India. Its objective is to alleviate a bottleneck in the steel manufacturing pipeline by the application of automated coal petrography. The problem can be defined as generating semantic segmentation of microscopic coal petrography images. We are presented with a heavily imbalanced and weakly labelled dataset having major intensity based interclass confusion.

We have attempted to solve this challenging problem by adopting a deep learning approach to do away with the painful feature engineering process that is often a necessity in classical machine learning. The segmentation task is approached as a pixel level multiclass classification problem. Our novel solution uses five binary U-Net classifiers in accordance with the One-vs-All approach to multiclass classification. These binary classifiers are trained using loss functions having additional regularization terms that we have developed in order to handle the interclass confusion problem. These regularizers have succesfully resolved majority of this confusion. The result obtained by amalgating the output of the binary classifiers is termed as coarse-segmentation and it suffers from both unclassified and misclassified pixels. These errors are corrected using a post processing module having four self-developed image processing algorithms and a fine-segmentation is obtained as the final result. Our solution's performance is benchmarked against two previous approaches based on a Miminum Distance Classifier and a Random Forest Classifier. Our method creates superior segmentations that have greater visual appeal and are more accurate. All experimental results are included to support our claim. It was also observed that our results were nearest to those obtained from the current, non-automated standard procedure used in the industry at present.

Keywords: *automated coal petrography, deep learning, U-NET, semantic segmentation, multiclass imbalance, weakly labelled data, image processing*

Contents

Chapter 1

Introduction

1.1 Introduction

The domain of operations management teaches that the efficiency of operation of a plant or industry is often hurdled by the slowest machine or process in the pipeline. Entire operations is heavily affected by the capabilities of this machine or process. Such hurdles are known as industrial bottlenecks. Industries are always on the lookout for identifying such bottlenecks and alleviating them as doing so will lead to significant improvements in overall performance and revenue gains for the company.

Today Tata Steel, India faces such a bottleneck where the manufacturing pipeline is slowed down heavily due to coal quality estimation tests that are mandatorily performed on batches of coal arriving at their plants. Results of these tests provide estimates of overall quality of the current batch and is used to decide whether to accept or reject the current batch. These tests involve manual analysis of coal petrography images by domain experts known as petrologists. Usually 24 hours or more si required to complete a single batch. The company has proposed the requirement of an intelligent system that can successfully generate segmented images as fast as possible. The maceral classes present have to be identified accurately and the phase fractions have to be calculated.

The branch of science that deals with identification of visible structures of coal is known as Coal Petrography. Images of coal is obtained using a petrographer's microscope and regions belonging to various maceral and mineral classes are identified. This identification is then used for calculating the Phase Fraction which will provide an insight to the overall quality of the coal. There are three, maceral classes to be identified namely Vitrinite, Inertinite and Liptinite. Along with Mineral, these four classes are often found embedded in a background Resin material. The overall research problem hence boils down to obtaining a semantic segmentation of microscopic images into five classes where the decision making process should take into account

intensity, spatial geometry, neighbourhood, textural information, etc.

Till date, existing industry accepted methods of coal maceral classification are mostly manual. A Minimum Distance Classifier was used by Mukherjee and Uma Shankar in 1994 [1]. Mukherjee and Paul used a Random Forest Classifier [2] that has successfully generated segmentations with a fair level of accuracy. However improved accuracies are desired which motivated us to look into the problem from the Deep Learning perspective. Deep Learning has made major breakthroughs in recent years in a wide variety of research domains including Pattern Recognition, Computer Vision, Natural Language Processing to name a few. Deep Neural Networks such as CNN [3], FCNN [4] and U-Net [5] has shown great promise in Medical Image Segmentation, Cell Tracking, Lesion Detection etc.

We propose a novel intelligent approach that has generated considerably better and more accurate segmentation as compared to the Random Forest Classifier and Minimum Distance Classifier. We have used a modified version of U-Net as our classifier. Due to pecularities of the dataset we have chosen to follow the One-vs-All approach and trained five different U-Net binary classifiers, one for each class. Each binary classifier is trained on its own individual dataset created from the original dataset provided by the company. This was done to tackle the major class imbalance present in the vanilla dataset. Each binary classifier is trained using custom loss functions. Binary cross entropy acts as the main loss function added to which are new regularizers developed in house which significantly reduces misclassification rates of each individual classes. The results of 5 individual classifiers are amalgamated to obtain a coarse-segmentation. Demerits of the One-vs-All approach involve that post amalgamation some pixels remain unclassified in the coarse-segmentation. In order to remove unclassified pixels as well as correcting incorrectly classified pixels we take the coarse-segmentation and apply four image processing based correction techniques namely Border Correction, Uniformity Based Correction, Region-based Correction and Shape-based Correction to generate fine -segmentation as our final result.

This novel approach has resulted in significantly better segmentation results as compared to previously mentioned techniques. We provide detailed results and method comparisons to support our claim. We have also provided a new type of neural network architecture that we call Nested-Net as our future work proposal as a continuation of this research.

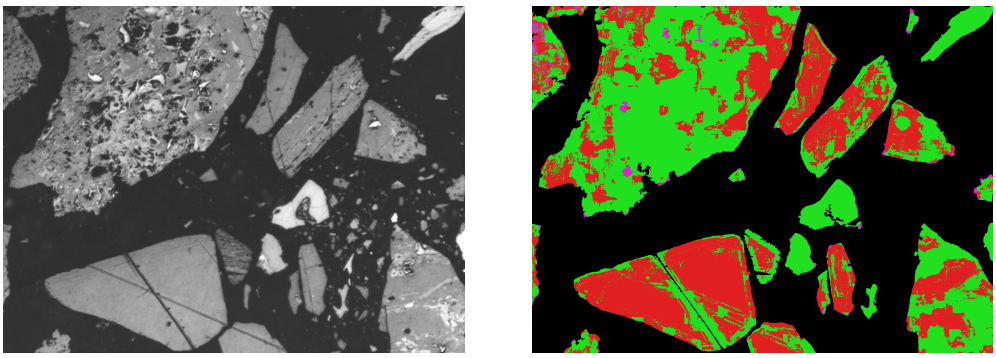
Chapter 2

Problem Statement and Overview of our Solution

This chapter provides a formal definition of the problem at hand and provides a basic overview of the method we adopted to solve it.

2.1 The Problem

The problem is formally defined as follows: given a set of pairs of input and ground truth images, design a machine learning model that when trained on this set is capable of generating an accurate semantic segmentation of petrographic images presented to it. The model should have good generalization capabilities and should not overfit the training set. Provided the desired levels of accuracy is achieved, the model will be deployed and put to use in the industry.



(a) Input Image (b) Desired Segmentation

Figure 2.1: Defining the problem: An example input and its desired output image.

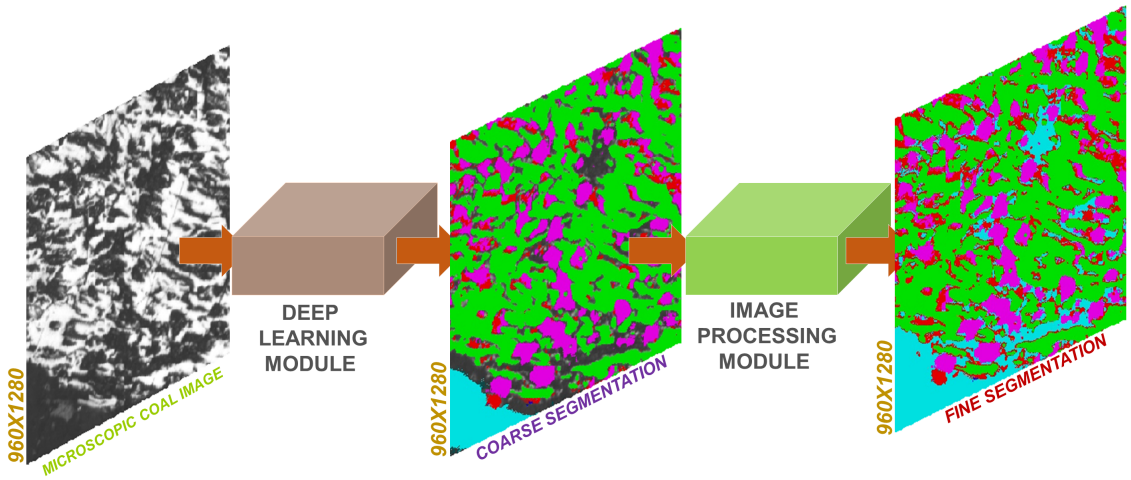


Figure 2.2: Block Diagram of our Deep Learning based solution

2.2 Overview of Our Solution

Figure 2.2 displays our solution strategy in the form of a block diagram. Our strategy can be described as a two step process. The input image is passed through a deep learning module that generates a coarse-segmentation. This coarse-segmentation is then passed through a image processing module which generates a fine-segmentation as the final result. Both the deep learning and the image processing modules have been explained in detail in chapter 6.

2.3 Why U-Net?

We have used U-Net as our main deep learning classifier. During our literature review it was found that U-Net has been shown promising results in medical image segmentation and cell tracking. Both these problems had images having textural intricacies at similar levels to our problem. This motivated us to choose U-Net as our classifier.

Chapter 3

Prerequisites

In this chapter we provide a very brief description of concepts necessary to explain our approach and experiments for easier understanding of the reader.

3.1 Coal Petrography

Viswanathan et al. [6] describes Coal Petrography as the branch of science concerned with the visible structure of coal, the structure may be examined visually by the unaided eye or by an optical microscope. Just as the components of inroganic rock are known as minerals, components of organic rock are known as macerals. The chemical behaviour and reactivity of coal can be evaluated with the knowledge of relative proportions of macerals obtained within a coal sample. Different macerals originate from different plant matter that got trapped during coalification. Different plant matter have different molecular structures which undergo different chemical alterations and hence exhibit difference in their chemical behaviour.

3.1.1 Maceral Descriptions

Vitrinite

Vitrinite maceral group makes up major proportions of most coals. The macerals of this group are derived from plant tissue (e.g. stem, root, bark, leaf). Material from the cell walls and cell contents contributes to various vitrinite macerals [7]. Vitrinite has reflectance values in between Liptinite and Inertinite and have a grayish apperance [2]. Also vitrinite has a more uniform texture compared to other maceral groups.

Inertinite

Inertinite group consists of materials which are relatively inert and undergo less alteration during carbonisation of a coal. They are generally dense, hard or brittle with a high reflectance in incident light [7]. They have lesser uniform texture compared to Vitrinite.

Liptinite

Liptinite maceral group is mostly found in low rank coal and have the lowest reflectance all all maceral groups. They have a more dark grayish appearance than Vitrinite.

Other non-maceral classes

Other than the three macerals described above two other classes need to be identified namely mineral and resin. Mineral deposits are often found within this macerals, and the macerals and mineral are bonded within a background resin material.

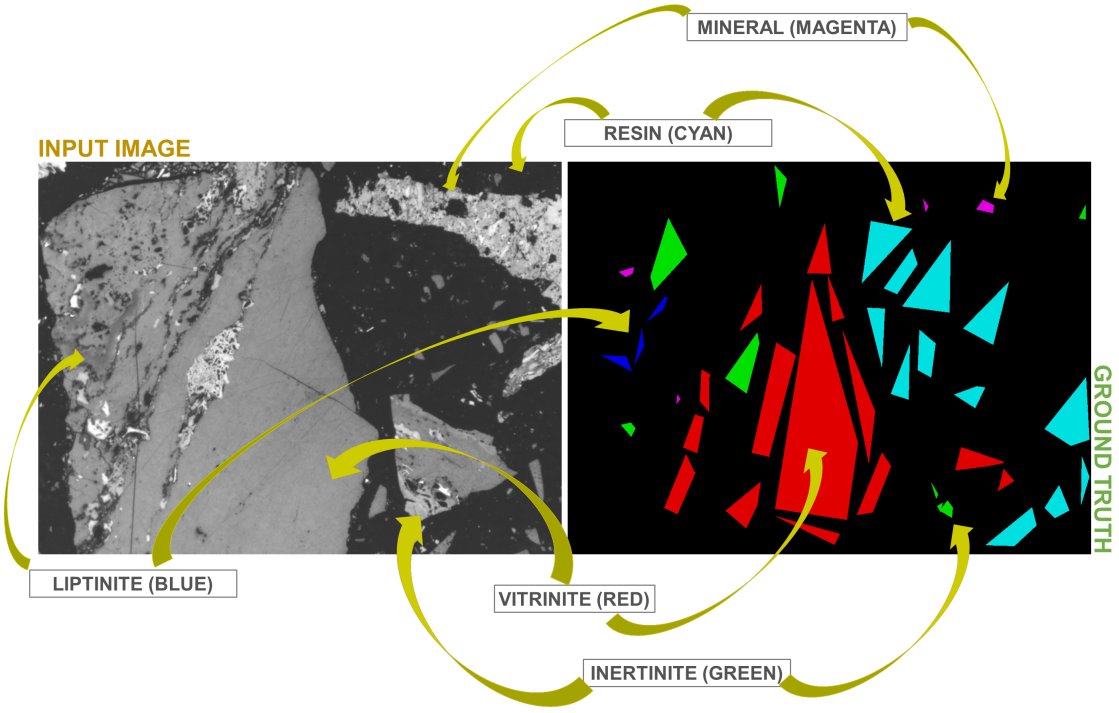


Figure 3.1: The five different classes that needs to be identified.

Phase Fraction

The metric used for calculating the relative proportions of macerals within the sample is known as Phase Fraction (PF) [2]. It is measured as a percentage of the overall pixel area that belongs to the current maceral or non-maceral class in question. It is evaluated for all the classes and the percentages together portray the relative proportion that we are trying to measure. Mathematically:

$$M_i = \left(\frac{S_i}{S}\right) \times 100\% \quad \forall\, i \in M, \tag{3.1}$$

where M_i is the PF and S_i is the pixel area of the i^{th} maceral or non-maceral class. S is the overall pixel area of the image and M is the set of all maceral and non-maceral classes to be identified.

3.2 Semantic Segmentation

Semantic Segmentation is the process of dividing an image into semantically meaningful parts and hence classified each part to one of the predetermined classes, five in our case [8]. This is achieved by allocating each and every pixel of the image to one of the five predetermined classes.

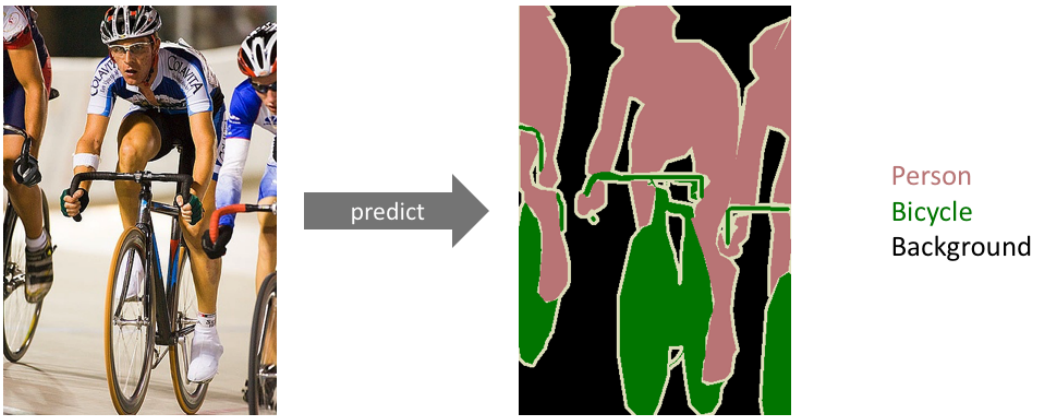


Figure 3.2: Semantic segmentation example, colours represent respective classes [9].

3.3 Convolution, Upsampling, Maxpooling

A brief understanding of this two mathematical operations and how they are performed on images is crucial and hence they are discussed in brief over here.

3.3.1 Convolution

The convolution operation involves sliding a matrix of weights known as the Kernel over an image and extract meaningful features from it [3]. This is similar to applying filters in image processing to extract edges, corners etc. Mathematically the operation is described as follows:

$$V = \sum_{i=1}^{q} \left(\sum_{j=1}^{q} \frac{f_{ij} \times d_{ij}}{F} \right), \tag{3.2}$$

where f_{ij} is the pixel value at (i, j) location and d_{ij} is the corresponding Kernel weight at the same. F is the stride i.e the amount the by which the kernel is shifted during the operation. q is known as the Kernel size. The convolution operation results in an image having dimensions lesser than the original image and is calculated as follows, $q^{'} = \frac{n+2\times p-q}{F} + 1$, where n is the image dimension and p is the size of padding. Padding involves bordering the image with a layer of zeros,is often used to handle odd dimensions. The output image so obtained is often referred to as a feature map.

3.3.2 Transposed convolution

The opposite of convolution is transposed convolution (also known as upsampling) [10] [11]. While convolution results in reduction of image dimensions, transposed convolution results in an output image having increased dimensions.This is usually achieved by techniques such as nearest neighbour, bi-linear or bi-cubic interpolations. Output dimensions are obtained by the equation :

$$q^{'} = F \times (n-1) + q - 2 \times p. \tag{3.3}$$

3.3.3 Maxpooling

This operation simply chooses the pixel value that is maximum among a group of pixels that occur within the pooling filter as the maxpooling kernel slides over the input image. It is mainly used to reduce dimensions of extracted feature maps [3].

3.4 Convolution Neural Networks

Convolution Neural Networks or CNN was developed for image classification [3]. CNN is made up of successive convolution layers having varying kernel sizes. They are followed by conventional hidden (also known as dense) layers that are present in traditional neural networks. Successive convolution layers act as automatic feature extractors while the dense layers serve as conventional neural network classifiers. Typically as we progress deeper into the network, the size of the extracted feature maps from the previous layer goes on decreasing. To ensure that an optimal number of features gets extracted the number of feature maps extracted in is gradually increased. A CNN can dual up as a primitive deep learning based semantic segmentor. A large image is taken and a neighbourhood of appropriate dimensions around the center pixel is chosen. The CNN predicts a class label based on this neighbourhood information. This operation performed over every pixel in the image results in a segmented image. Ciresan et al [12]. used this method for medical image segmentation and won the ISBI 2012 EM Segmentation Challenge.

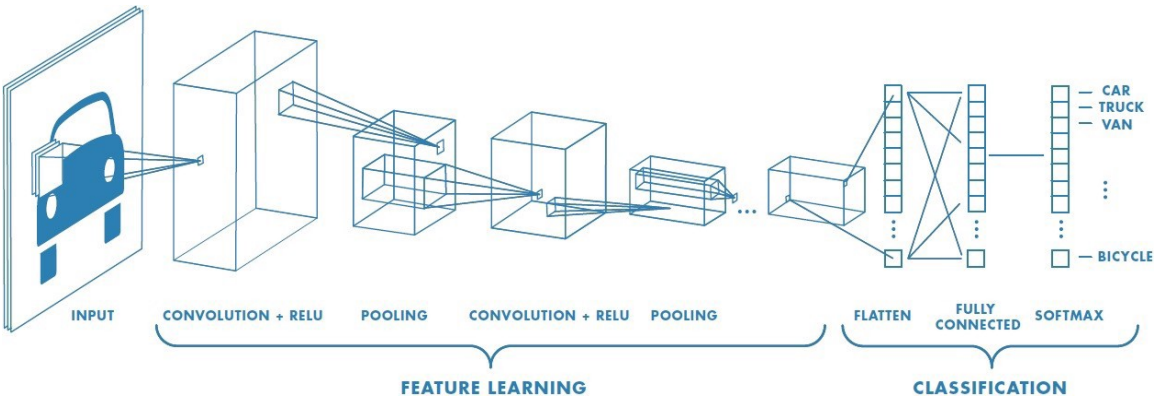


Figure 3.3: How a Convolution Neural Network works[13].

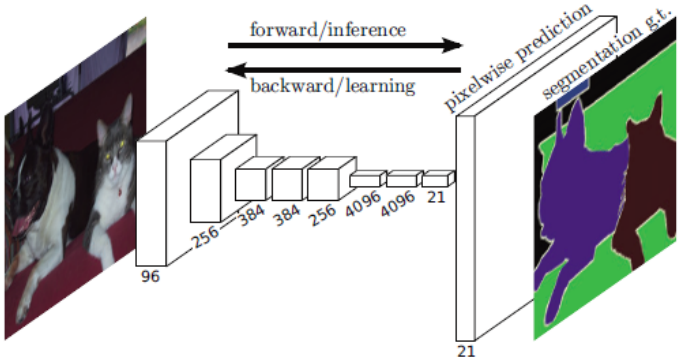


Figure 3.4: How a Fully Convolution Neural Network works [4].

3.5 Fully Convolutional Neural Networks

While CNN may provide a decent segmentation however as the resolution of the input image increases the time requirement to generate it becomes significant. Fully convolutional neural networks [4] do away with the concept of having a sliding window for pixelwise classification. Instead they take the whole image as input and generates segmented images as a whole as output. This is achieved by converting the flat dense layers of CNN into 1×1 convolutions followed by reshaping into a two dimensional image having $H \times W$ dimensions. Due to maxpooling during convolution the resulting image is small hence transpose convolution (also known as upsampling) is applied to reshape the image to have the same dimensions as the original image. Just a single upsampling results in rough segmentation so output of penultimate convolution layers are also upsampled and the final segmentation is obtained by fusing the results of all the upsampling operations using element wise addition. FCNN beat the performance of CNN as semantic segmentors in 2014. In Figure 2.4 the numbers denote the number of filter maps obtained.

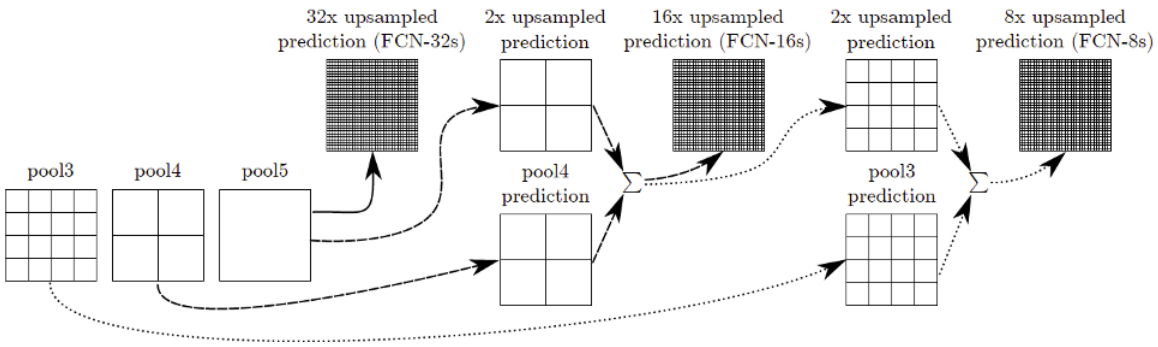


Figure 3.5: Fusing of all upsampled outputs to generate final segmentation [4].

3.6 U-Net

U-Net was introduced in 2015 [5]. It extended the idea of upsampling and fusion of FCNN to develop an encoder-decoder architecture for generating segmentation. First half of the U-Net architecture acts as a convolutional encoder that extracts meaningful features from the input image. The feature maps obtained as an output of the encoder stage is an effective summarization of the information present in the input image. The second half of the U-Net architecture acts as a convolutional decoder that decodes these feature maps and generates the required segmentation. Figure 2.6 describes the UNET architecture. The vanilla U-Net comprises 5 encoding blocks followed by 5 decoding blocks. Each encoding block is of the form conv-conv-pool i.e two convolution layers followed by a maxpooling layer. Transition from one encoding

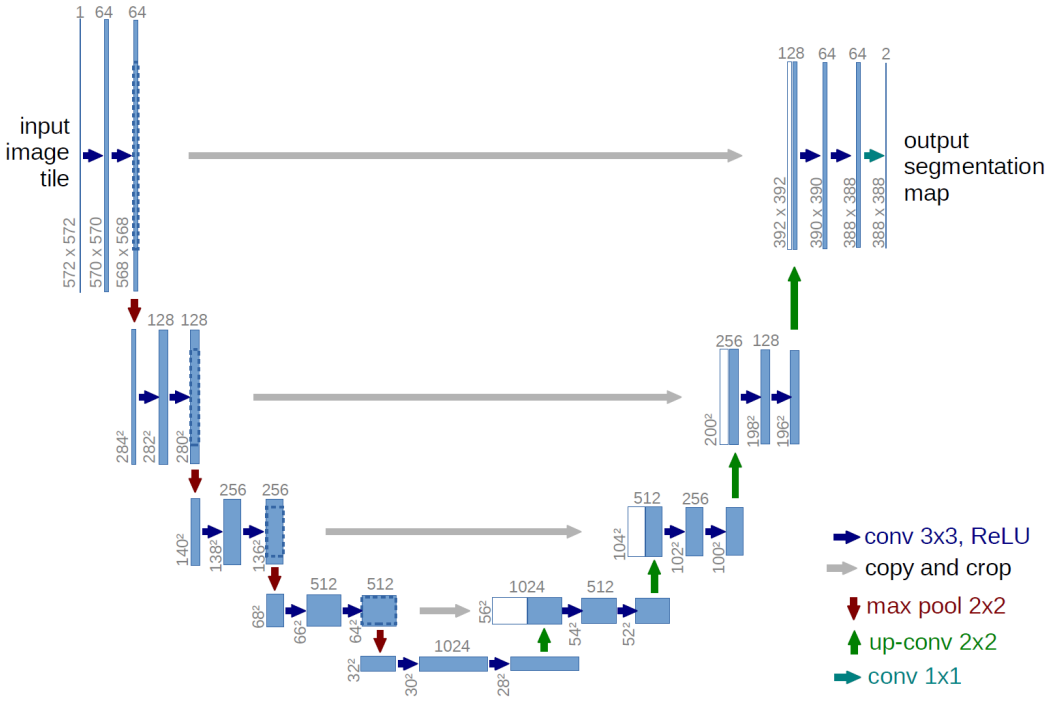


Figure 3.6: Vanilla U-NET Architecture

block to the next reduces image dimensions by half. The feature maps obtained before the maxpooling operation of each encoding block is copied, cropped and passed on to the corresponding decoding block. This ensures enough information is available at each stage of decoder to generate accurate segments. Each deconding block is of the form deconv (also known as upconv-concat-conv-conv. In other words, upsample the feature maps from the previous block, concatenate these decoded feature maps with those passed on from the encoder block corresponding to the current decoder block and fuse them together using two successive convolution operations.

3.7 Losses and Metrics

Neural networks fall in the domain of supervised learning. Training a neural network involves presenting it with a set of data for which the correct output, named as the ground truth is already known to us. The network provides us a predicted output. The difference between the ground truth and the predicted output is known as the loss. Metrics are methods of evaluating the output of the network. While loss is used to train the network, metric is used only for evaluation. A brief description of the loss functions and metrics used is provided in this section.

3.7.1 Binary Cross Entropy

Binary Cross Entropy (BCE) is a loss function derived from information theory [14] [15] . Lets say the ground truth data comes from a distribution known as the true distribution $q(y)$ while the neural network predicts a result that comes from a distribution $p(y)$. Entropy is a measure of uncertainity of a distribution. Thus cross entropy becomes a measure of estimating how far away $p(y)$ is from $q(y)$. The purpose of training is to make $p(y)$ as close to $q(y)$ as possible. Mathematically:

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log\left(p(y_i)\right) + (1 - y_i) \cdot (1 - \log\left(p(y_i)\right)), \tag{3.4}$$

Here N is the total number of datapoints. y_i is the ground truth value of the i_{th} datapoint and $p(y_i)$ is the predicted value for the current datapoint. For the case of images, one datapoint is equivalent to one pixel and the loss is calculated over all pixels present in the image.

3.7.2 Intersection over Union

Intersection over Union is a metric used for evaluating how accurate a segmentation has been predicted [16] [17]. It is the ratio of the intersection of the predicted and ground truth (or target) images over the union of the two images. Greater the ratio, more accurate is the segmentation obtained. Mathematically:

$$IoU = \frac{target \cap prediction}{target \cup prediction}. \tag{3.5}$$

3.8 Training Neural Networks

This section describes in brief how the training of neural networks takes place.

3.8.1 The Gradient Descent Algorithm

Originally developed by Cauchy [18] this algorithm is used extensively for training neural networks. Training data is presented to the network and corresponding predictions are obtained. The value of the loss is calculated by comparing predictions with ground truth values. The objective of gradient descent is to minimize the loss by updating the weights of the network in the direction of steepest descent i.e the gradient. The algorithm converges when we have reached a local minima and no further reduction of the loss value is possible. Mathematically:

$$w_i^{'} = w_i - \eta \nabla(J), \tag{3.6}$$

where η is known as the learning rate which controls how fast the network reaches the local minima, w_i is the weight value, $w_i^{'}$ is the updated value of the weight w_i and J is the calculated loss. Presenting the input to the network and obtaining the prediction is known as forward propagation while calculating loss, and updating weights using equation 2.6 is known as backpropagation. If backpropagation occurs after every forward propagation then it is called Stochastic Gradient Descent. When backpropagation occurs after forward propagating a batch of data and accumulation the net loss then it is known as Mini-batch Gradient Descent. We have used Mini-batch Gradient Descent as our training algorithm.

3.8.2 Regularization

This is a method used to fine tune predictions made by the network. It also helps in better training by reducing overfitting.

Dropout

Dropout regularization [19] is a technique that distributes the decision making process over the entire network. During trainingit may so happen that the weight value associated to a particular hidden neuron may become very high and it may start behaving as the deciding neuron for a particular class. Drop out randomly selects neurons and sets their activations to zero. This ensures the weight of those neuron don't get updated for the current pass. This maintains uniformity within the network.We have developed our own novel regularizers that has helped in reduction of misclassification for certain classes.

3.8.3 ADAM Optimization

This optimization technique speeds up the training process. Instead of using a fixed learning rate η, ADAM [20] uses an adaptive learning rates for different parameters. Adaptability is achieved by using exponentially moving averages computed on gradients of the current mini-batch. We have utilized ADAM optimization to speed up our training process.

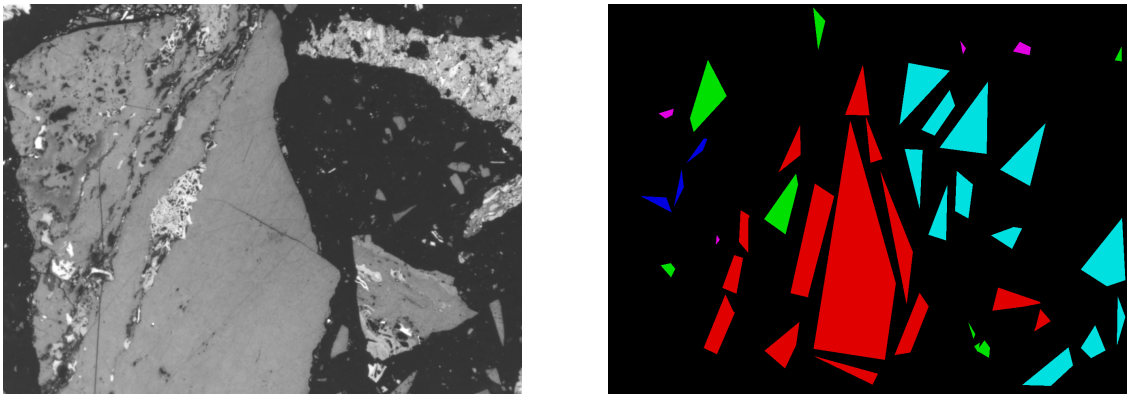
Chapter 4

The Dataset and its Peculiarities

This chapter describes the dataset provided by the company. An example of input and ground truth image is presented here. Several peculiarities were observed within the dataset, which increased the difficulty of the classification task. Challenges imposed on our problem due to such pecularities are elaborated. We also provide a description of data preprocessing techniques that was applied to generate the final training and test datasets.

4.1 General Description

The company provides 5 datasets each having 15-20 high resolution images to train our classifiers and 150-300 test images. All datasets have images dimensions 1920×960. Training images are accompanied by corresponding ground truth while no such labelled data is provided for the test images.



(a) Input Image (b) Ground Truth

Figure 4.1: An example input image and its ground truth image.

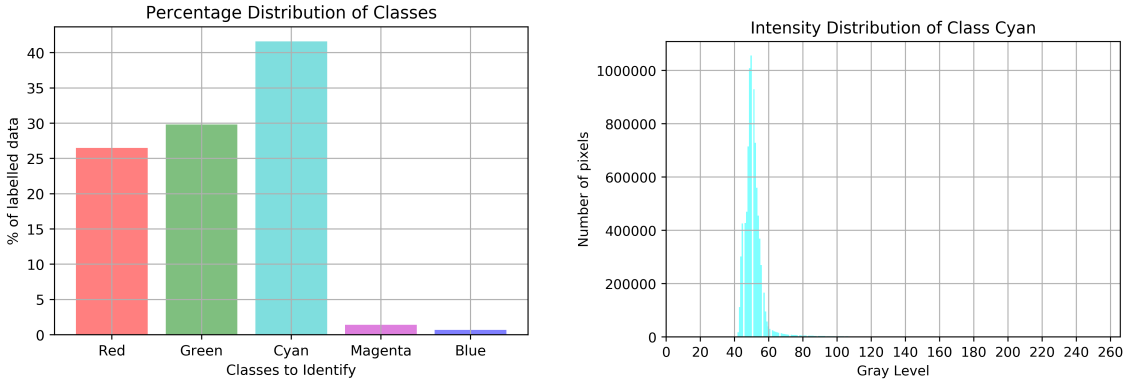
Figure 3.1 displays an example Input Image and its corresponding Ground Truth Image. In the Ground Truth Image, Red, Green, Blue represent Vitrinite, Inertinite and Liptinite maceral classes respectively. Magenta represents mineral and cyan is used to represent background resin. Here onwards the classes will be referred to by their corresponding colours.

4.2 Peculiarity 1: Weakly Labelled Ground Truth

A careful examination of Figure 3.1 reveals that the ground truth has been labelled very weakly. Even within the ground truth image several pixels have not been labelled (all black pixels in Figure 3.1(b). Moreover approximate shapes has been used to mark the classes instead of following exact contours. In certain cases these approximate shapes include regions belonging to different classes giving rise to incorrectly labelled pixels. This poses a major problem from the deep learning perspective. Using a smaller training set leads to a network that cannot predict certain classes whereas taking all images present in our training set leads to overfitting. In such cases the network starts detecting these approximate shapes instead of the original contours.

4.3 Peculiarity 2: Imbalanced Classes

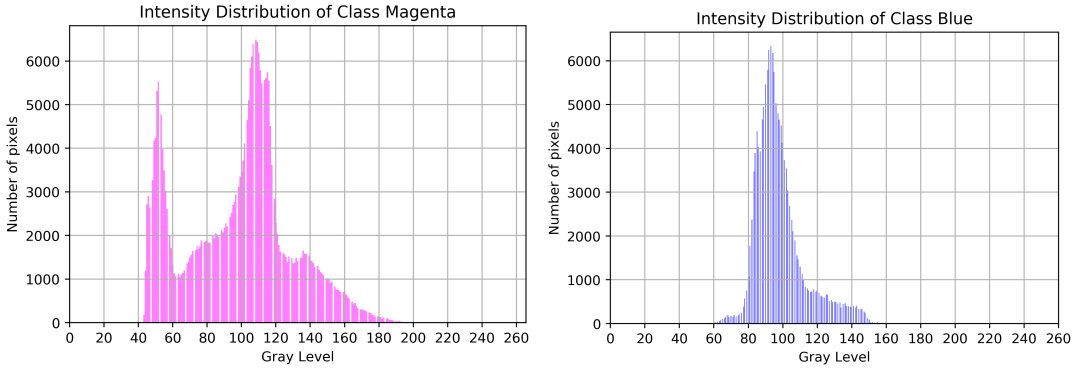
A visual inspection of the training set reveals that heavy class imbalance is present within the dataset. Cyan, Red and Green i.e Resin, Vitrinite, and Inertinite are available in plenty and act as majority classes whereas Magenta and Blue has very few representations and act as minority classes. Figure 3.2(a) compares the total number of available pixels of ground truth for each class. It clearly portrays this imbalance.



(a) Class Imbalance

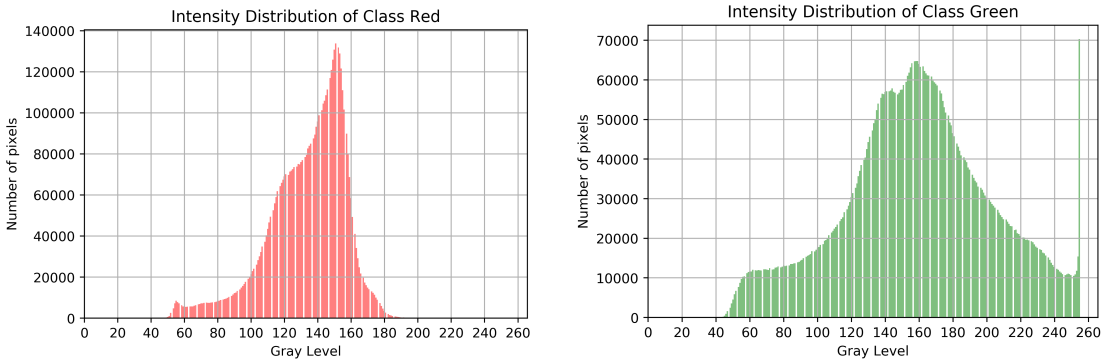
(b) Histogram for Cyan

Figure 4.2: Percentage Distribution of Labelled Data, Intensity Distribution of Cyan.



(a) Histogram for Magenta (b) Histogram for Blue

Figure 4.3: Intensity Distribution of Minority Classes.



(a) Histogram for Red (b) Histogram for Green

Figure 4.4: Intensity Distribution of Majority Classes.

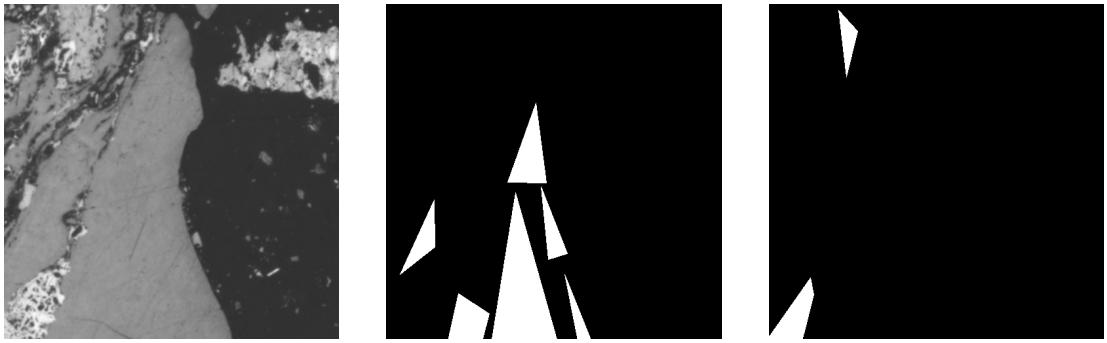
4.4 Peculiarity 3: Confusion between Classes

Figure 3.2, 3.3 and 3.4 shows the intensity distribution of the five classes. It is seen that the peaks of Cyan and Magenta coincide at gray levels $40 - 60$. There is considerable overlap among the Red and Green histograms and some overlap between Red and Blue. These overlaps are a consequence of the weak labeling described earlier and this contributes greatly to misclassification. Moreover it was observed that there is a strict variation of illumination among the 5 datasets. This also contributes to the overlap of histograms. A classifier trained to detect Cyan detects all Magenta regions as Cyan. A classifier trained to detect Magenta detects some regions of Magenta as Cyan. Often the classifiers of Red and Green misclassify each other while the classifier for Blue detect some Red regions as Blue. Two novel regularizers that heavily penalize the neural networks during training for the above mentioned

misclassification were developed. To improve the results even further four image processing based misclassification correction algorithms were also developed.

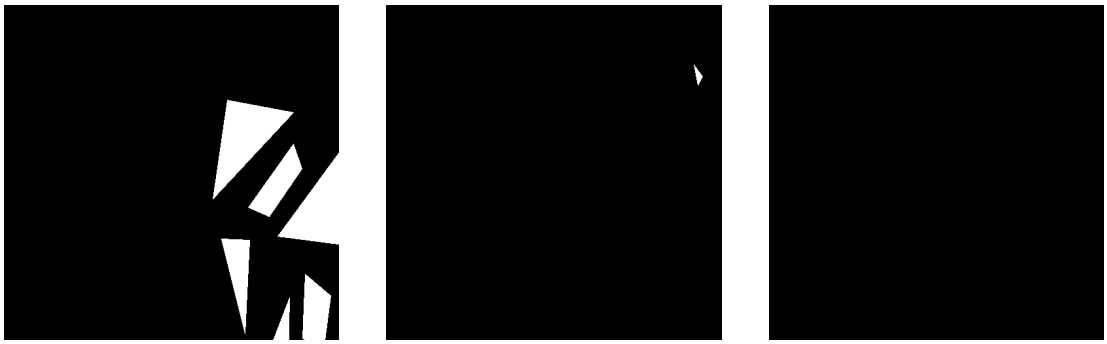
4.5 Data Preprocessing

Our proposed solution imposed the requirement of developing individual datasets for all the classes. This was mainly done to remove class imbalance to a certain extent. The high resolution input images of the training set were divided into patches of size 512×512. The original Ground Truth images were used to construct binary masks of the same size for each class. Figure 4.5 and 4.6 displays a sample input patch and its corresponding masks. The number of training patches for Red, Green, Cyan, Magenta and Blue are 204, 204, 204, 167 and 55 respectively. Note that Blue is not present in this patch, its mask is pure black or in other words all pixel labels are 0.



(a) Input Patch (b) Binary Mask for Red (c) Binary Mask for Green

Figure 4.5: A sample Input Patch and Binary Masks for Red and Green.



(a) Binary Mask for Cyan (b) Binary Mask for Magenta (c) Binary Mask for Blue

Figure 4.6: Binary Masks for Cyan, Magenta and Blue.

Chapter 5

Existing Solutions

In this chapter we provide descriptions of previously existing solutions to the problem at hand. A brief idea regarding these solutions is necessary as we have benchmarked the performance of our solution by comparing with results obtained from these solutions.

5.1 Minimum Distance Classifier Approach

Uma Shankar and Mukherjee came up with this solution in 1991 [1]. It is one of earliest approaches to automated coal petrography. They applied the Minimum Distance Classifier (MDC) [21] [22] to RGB petrographic images. Mukherjee and Ghosh (cite) carried out a comparison of results by applying the same MDC on grayscale images. We intend to do the same and compare all three methods.

An MDC is used to classify unknown image data to classes that minimize the distance between the image data and the class in multi-feature space. The distance is defined as an index of similarity so that the minimum distance is identical to the maximum similarity [22]. The k^{th} class is represented by the mean of gray levels of that class. This mean is calculated from available ground truth information and can be expressed mathematically as:

$$m_k = \frac{1}{T_k} \sum_{i=1}^{T_k} x_i^{(k)},\ k = 1, \ldots, C, \tag{5.1}$$

where m_k = mean of gray levels of ground truth of k^{th} class. $T_{k=}$ total number of ground truth pixels of k^{th} class. $x_i^{(k)}$ is the gray level of the i^{th} pixel of the k^{th} class. There are C classes in total. A test image pixel x is classified to class C_k provided:

$$x \,\epsilon\, C_k \quad iff \quad d_M(x,\, C_k) = min\Big\{ d_M(x, C_i) \Big\},\ \ k = 1, \ldots, C, \tag{5.2}$$

where $d_M(x, C_i)$ is the Euclidean distance of x to class C_k.

5.2 Random Forest Approach

Paul and Mukherjee [2] used a Random Forest Classifier (RF) [23] [24] to obtain better segmentation compared to the MDC. A brief description of the feature extraction process as well as the method is necessary.

5.2.1 Feature Extraction Process

If x was the current pixel that they wanted to classify then a 31×31 neighbourhood is chosen with x as the centre. This provides local neighbourhood texture information that aids the classification process. This neighbourhood was further subdivided into blocks of 3×3. For each block the mean and variance of the 9 pixels present was calculated. Let μ_l, μ_h, σ_l and σ_h denote the lowest and highest values of mean and variance respectively. Dividing $(\mu_h - \mu_l)$ into α and $(\sigma_h - \sigma_l)$ into β equally spaced bins gives them $\alpha \times \beta$ number of bins. A mean-variance histogram was plotted with each pixel in the current neighbourhood getting allocated to some bin based on the mean and variance of it's corresponding 3×3 block. Taking $\alpha = 10$ and $\beta = 10$ a 100 dimensional mean-variance histogram based feature was extracted for each pixel. This feature was fed into the RF to obtain the class label for x.

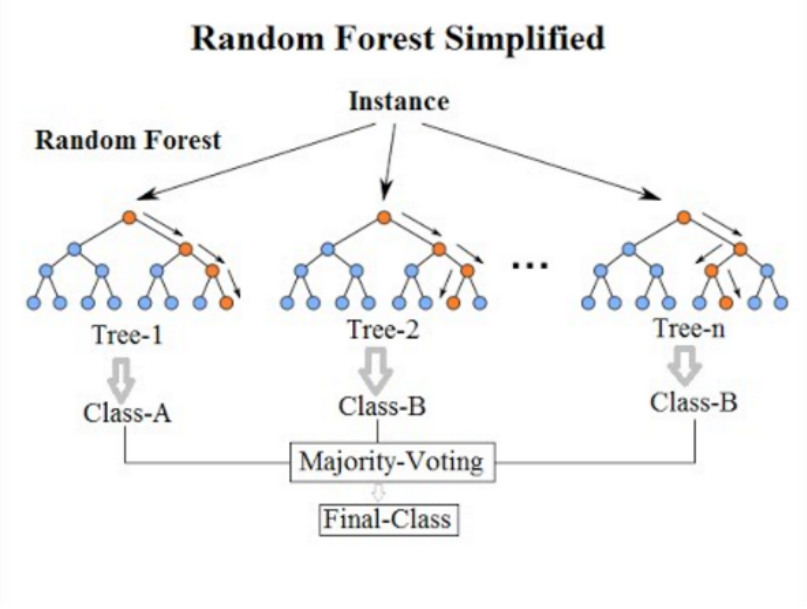


Figure 5.1: How a Random Forest Works [25].

5.2.2 Random Forest

A random forest [23] [24] is an ensemble of n decision trees [26]. A Decision Tree is a classical machine learning classifier that builds a binary tree based on the available information (in the form of features) obtained from the training set. Each node of the Decision Tree makes a decision whether to progress down the left child or the right child of the tree based on the value of a particular feature. The feature that leads to maximum gain in entropy or GINI Index [27] [28] is chose as the deciding feature for the current node. Leaf nodes of decision trees represent class labels. An input x to the tree will traverse down the tree upon a particular path (decided by the magnitude of the features of x) and will eventually end up at some leaf node. The class label associated with this leaf node is the class label allocated to x. For their case a pixel is allocated a colour based on its 100 dimensional feature described above.

The random forest is made up of n such decision trees. Each such tree is build using a subset of the entire training set by repetitive sampling with replacement known as bootstrapping [29]. A subset of the total available features is used to make the decision making process of each individual tree. An input x is passed to all n trees, Each tree independently predicts a class label for x. Majority Voting is carried out to allocate the final class label to x.

Each pixel of the input image was represented by its neighbourhood's mean-variance histogram. The extracted features were passed through the random forest to obtain the class label. Hence the segmented image is created.

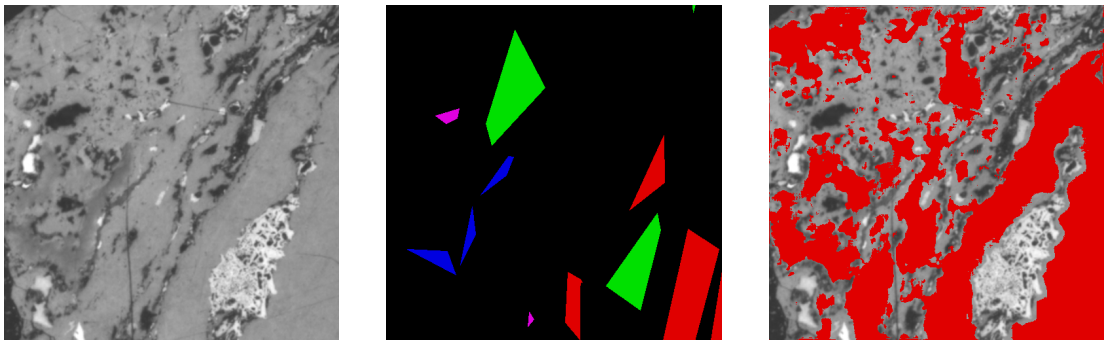
Chapter 6

Proposed Solution

This chapter provides an elaborate description of our contribution to the research problem at hand. The motivation behind adapting the One-vs-All approach as the correct solution strategy is described first. It is followed up by our methodology, network architecture and training details. The pecularities of the dataset that were described earlier are taken up one by one and the novel solutions that we developed for resolving them are detailed.

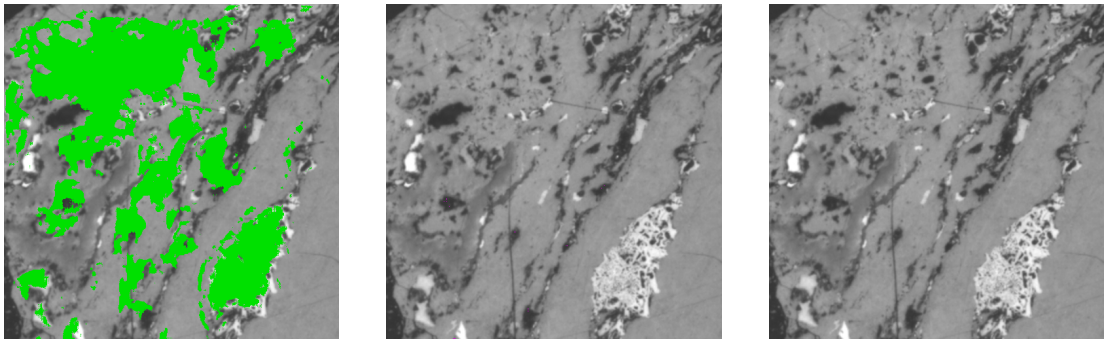
6.1 Motivation behind adopting the One-vs-All Approach

Several paradigms to achieving multiclass classification exists. One such paradigm is using a single multiclass classifier. In the nascent stages of the research, only three of the five datasets were available. A dataset comprising 204 patches of size 512×512 were created. Neural networks, being inherently multiclass classifiers, our intial resolution was to use a single U-Net to perform the classification task. We tried to train our U-Net on this dataset of 204 images. It was capable of detecting the majority classes, but it failed miserably while detecting the minority classes of Magenta and Blue. The heavy class imbalance (refer to Figure 3.2(a)) present in the dataset was identified as the root cause of the problem. As the minority classes make up less than 1% of the dataset, a neural network that is completely incapable of detecting them is still 99% accurate. Hence, gradient descent happily converges to such a local minima every time the model is trained. Figure 6.1 provides a visual elaboration of this problem. Even after experimenting with loss functions such as Focal Loss [30] that has been specifically designed to handle class imbalances, the problem persisted.



(a) Input Patch (b) Ground Truth (c) Red Prediction

Figure 6.1: Predictions on smaller dataset: Input Ground Truth and Red Prediction (Cyan being absent in this example has not been shown).

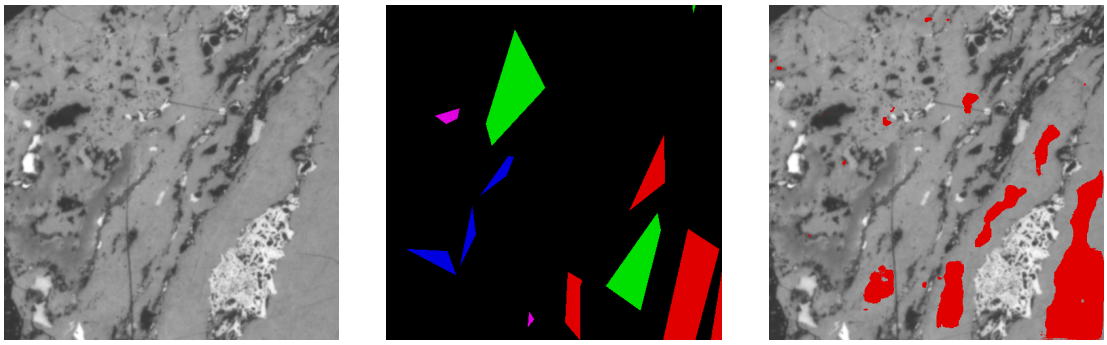


(a) Green Prediction (b) Magenta Prediction (c) Blue Prediction

Figure 6.2: Predictions on smaller dataset: Green, Magenta and Blue Prediction (Cyan being absent in this example has not been shown).

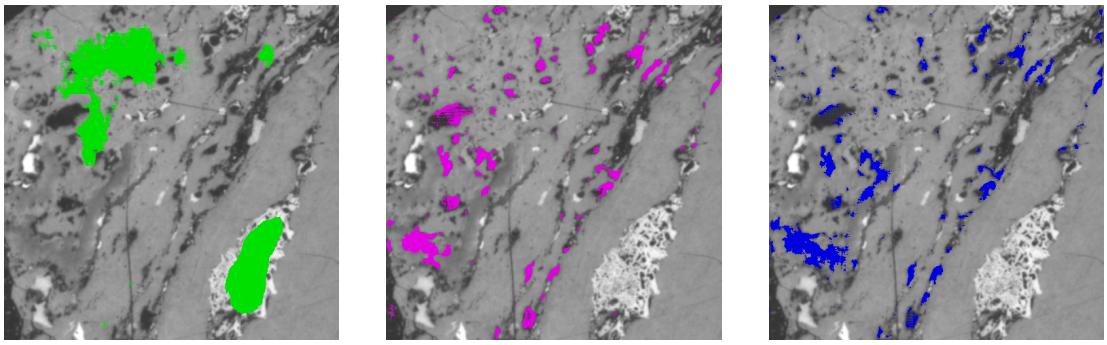
Later two more datasets arrived and a larger dataset of 371 patches were created. These new datasets contained slightly more representation of minority classes however even now the net minority class representation remained less than 3%. The U-Net was now trained on this larger dataset. It was seen that minority classes were now being detected. However the network lost its generalization capabilities for the majority classes. It started overfitting the majority classes of Red, Green and Cyan. It had learned the approximate shapes used to represent the classes in the ground truth instead of the original contours. The weak labelling of the dataset described previously was identified as the root cause for this problem. Figures 6.3 and 6.4 provides the corresponding visual elaboration.

One interesting fact to note is that in addition to these problems both networks still suffer from the third and most important peculiarity of confusion between classes. Varying illumination among the datasets as well as weak labelling jointly contributes



(a) Input Patch (b) Ground Truth (c) Red Prediction

Figure 6.3: Predictions on smaller dataset: Input Ground Truth and Red Prediction (Cyan being absent in this example has not been shown).



(a) Green Prediction (b) Magenta Prediction (c) Blue Prediction

Figure 6.4: Predictions on larger dataset: Green, Magenta and Blue Prediction (Cyan being absent in this example has not been shown).

to this confusion. These three problems increases the difficulty of the task manifolds. Hence we were motivated to eliminate this paradigm in favour of the One-vs-All approach.

The One-vs-All approach uses several binary classifiers, one for each class. Each classifier now classifies whether a particular pixel belongs to the current class or not, for example the classifier for Blue would say whether a pixel is Blue or Not-Blue. The predictions of the individual classifiers are assembled together to obtain the overall multiclass classification. Several advantages of using such a solution is apparent.

6.1.1 Merits of the One-vs-All Approach

- It provides us with a divide and conquer strategy. The previously difficult problem is now divided into 5 comparatively simpler problems. Each problem

can be solved to a high degree of accuracy individually.

- 5 individual datasets were created to train the 5 binary classifiers. The majority class classifiers were trained on the smaller datasets of 204 patches each. This ensured that model generalization capabilities were preserved. The dataset for Magenta was brought down to 167 patches while that of Blue had only 53 patches. This reduces the Magenta to Non-Magenta and Blue to Non-Blue imbalance ratios significantly and the corresponding binary classifiers can detect them successfully. Thus the class imbalance problem is resolved.

- Each binary classifier can now be trained on independent loss functions to resolve the inter-class confusion problem. We developed two novel regularizers and added them to the BCE loss and this successfully resolved the inter-class confusion to a certain degree.

6.2 Methodology

Our methodology is described next. Figure 6.5 neatly describes the workflow. Following the One-vs-All approach, 5 independent binary U-NET classifiers have been trained. Each classifier is trained on its own individual training set made up of 512×512 patch, binary mask pairs (refer to Figures 3.5 and 3.6). The classifiers are trained using individual loss functions, details of which are provided later. The output of the 5 classifiers is amalgamated to generate a segmented image. This segmentation is coarse and contains both unclassified as well as misclassified pixels. Coarse-segmentation is elaborated in section 6.4.4. A detailed study of coarsely segmented images was carried out and four image processing algorithms were developed to generate the Fine-segmentation as our final result.

6.2.1 U-Net Architecture

Our U-Net architecture (refer to Figure 6.6) differs from the Vanilla U-Net architecture previously described (Figure 3.10). Much like the original our net has the same overall 5 layers of encoder-decoder structure. However we have used padding to maintain image dimensions within each encoder/decoder block. In the vanilla architecture successive convolution layers go on reducing the image dimensions. Also we used far less number of feature maps compared to the original. The number of feature maps generated post convolution remains constant within a block. We have extracted 16 feature maps in the first layer and have progressively doubled the number moving down the layers. 256 feature maps having dimensions 32×32 are obtained as the output of the encoding half of the U-Net.

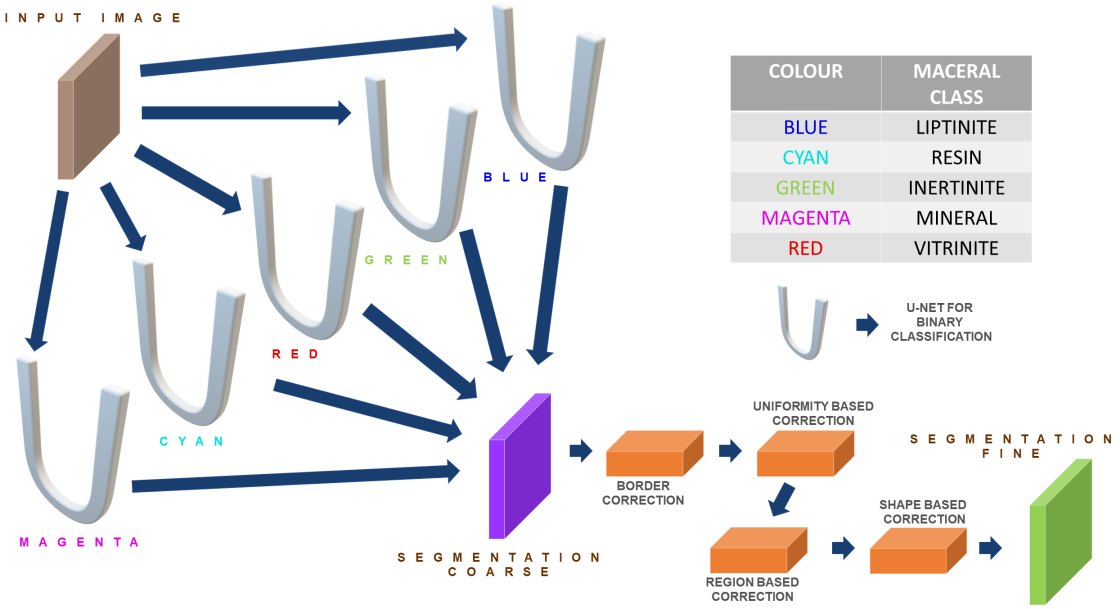


Figure 6.5: Methodology of proposed solution.

In the decoding stage, 50% of feature maps are copied from the output of the corresponding encoding block and the rest 50% is obtained by decoding the previous layer. Theses maps are stitched together by two successive convolution layers and the same number of feature maps are generated as output.

6.2.2 Details of Training

We used python 3.6 as our programming language. The neural networks were developed using Keras (version 2.2.4) [31] and Tensorflow (version 1.13.1) [32] . These two deep learning libraries provide extensive GPU support to speed up training by parallelization. Our system comprises an Intel i7-7770HQ CPU, a 4GB Nvidia GeForce GTX 1050Ti GPU, 16 GB of ram and an A-DATA Nvme SSD for data storage. Dropout layers were added at the end of encoding block of each U-Net. Each model was trained for 100 epochs with an earlystopping patience of 7-10. Training time varied from 20 minutes to 2 hours approximately.

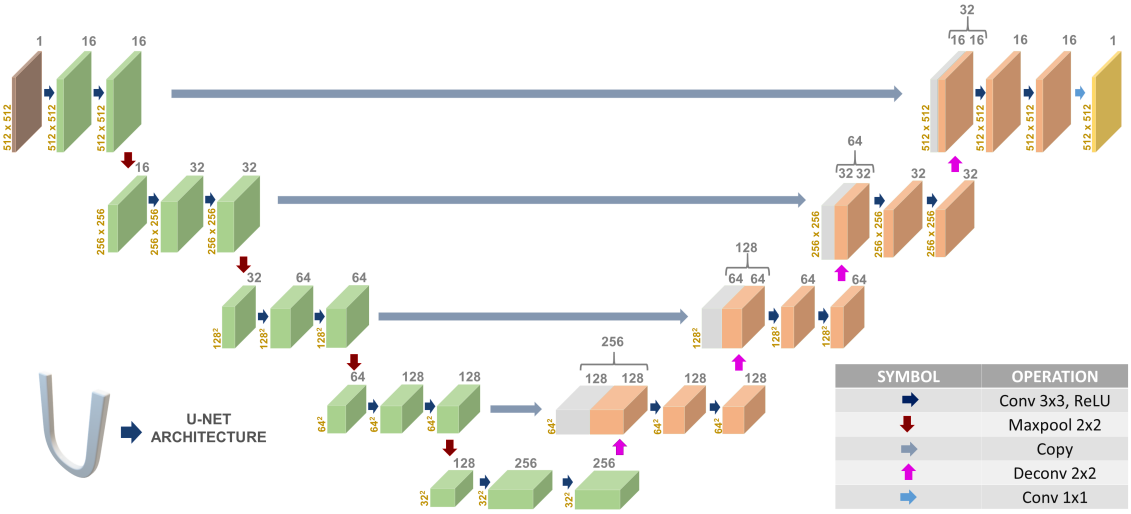


Figure 6.6: Architecture of our U-Net binary classifier.

6.3 Area based Regularization for Cyan-Magenta Confusion

We developed this innovative area based regularization technique to tackle the Cyan-Magenta confusion problem. The problem is described followed by our solution and obtained results reported.

6.3.1 The Cyan-Magenta Confusion

Figure 4.2(b) and Figure 4.3(c) plots the histograms of Cyan and Magenta classes respectively. It is seen that all ten lakh pixels marked as Cyan in the Ground Truth have a gray level between 40-60. The histogram of Magenta has two peaks of which the smaller peak of approximately six thousand pixels also lie within the same gray level range as above. This overlap of peaks imply that any classifier that uses gray levels to decide class labels will get confused among these two classes. It is also noted that both these classes are devoid of any texture and appear as plain black regions. Hence even the spatial neighbourhood information extraction of convolution neural networks will be at a loss while distinguishing these two classes.

We trained two individual U-Nets to detect these two classes, using BCE (equation 2.4) as our loss function. The resulting U-Net for Cyan detected Cyan regions accurately but additionally misclassified all Magenta regions as Cyan. The U-Net for Magenta detection detected Magenta accurately but detected certain regions of Cyan as Magenta.

6.3.2 Area based Regularization

We pondered on resolving this confusion. Certain domain knowledge was available that Magenta (i.e Mineral deposits) occurs in small quantities and is often embedded in other maceral classes namely Red (Vitrinite) and Green (Inertinite). While Cyan is the background resin that embeds the above mentioned macerals. In other words pixel area of Magenta regions will be far less than those of Cyan. Careful inspection of the Ground Truth revealed that this is indeed the case.

Therefore if we can somehow penalize the Cyan neural network whenever it predicts smaller regions (having smaller pixel area) as Cyan, it will gradually learn not to do so. The exact opposite is applicable for Magenta, we make the neural network learn to predict only smaller regions as Magenta. This is achieved by adding an area based regularization term to the previous loss function. These terms are described mathematically as follows:

$$R_{cyan}(A_c) = \frac{1}{n_c}\left\{ \sum_{A_i<A_c} A_i \Big/ \sum_{i=1}^{n_c} A_i \right\}, \tag{6.1}$$

$$R_{magenta}(A_m) = \frac{1}{n_m}\left\{ \sum_{A_i>A_m} A_i \Big/ \sum_{i=1}^{n_m} A_i \right\}, \tag{6.2}$$

where n_c and n_m are the total number of connected components labelled as Cyan and Magenta by the networks. A_i is the pixel area of the i^{th} connected component. $\sum_{A_i<A_c} A_i$ and $\sum_{A_i<A_c} A_i$ represent the sum of pixel areas of all connected components that have $A_i < A_c$ and $A_i > A_m$ respectively. Here A_c and A_m are area thresholds for the corresponding classes. The loss functions used to train Cyan and Magenta classes now become:

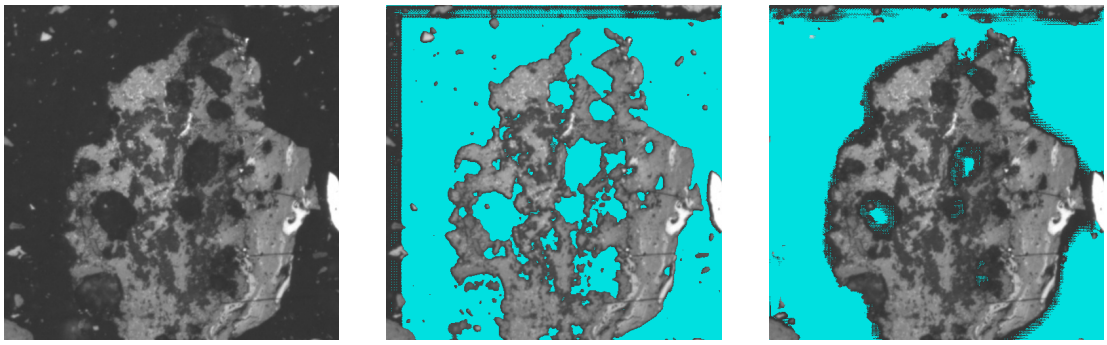
$$J_{cyan}(p, q, A_c) = H_p(q) + \lambda_1 \times R_{cyan}(A_c), \tag{6.3}$$

$$J_{magenta}(p, q, A_m) = H_p(q) + \lambda_2 \times R_{magenta}(A_m), \tag{6.4}$$

where λ_1 and λ_2 are hyperparameters that control the weightage assigned to the regularizing terms. p and q represent predictions and ground truth respectively.

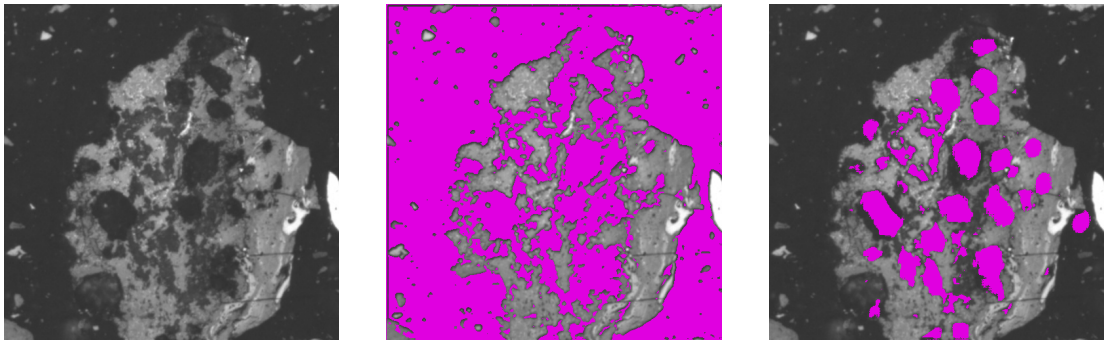
An intuitive understanding of the above equations is provided next. Consider the Cyan network. If it predicts smaller regions (having pixel area $< A_c$) as Cyan the numerator of $R_{cyan}(A_c)$ goes on increasing. This increases $J_{cyan}(p, q, A_c)$ in return. The only way for Gradient Descent to minimize the value of $J_{cyan}(p, q, A_c)$ is if it can reduce the value of $R_{cyan}(A_c$ and hence forcing the network to modify its weights in such a manner that it no longer predicts smaller regions as Cyan. A similar explanation is applicable for the Magenta network. It learns not to predict larger regions as

Magenta. Figure 5.7 and 5.8 displays the benefits of adding this regularization term in the obtained segmentation.



(a) Input Image (b) Ony BCE (c) BCE + Regularization

Figure 6.7: Predictions of Cyan Classifier without and with Area Regularization.



(a) Input Image (b) Only BCE (c) BCE + Regularization

Figure 6.8: Predictions of Magenta Classifier without and with Area Regularization.

6.4 Intensity Based Regularization: Improves Detection of Blue

We developed a second regularization term based on pixel intensities. It improved the detection of Blue compared to vanilla BCE.

6.4.1 Magenta - Blue Confusion

Figure 3.3(a) and 3.3(b) reveals that the second peak of Magenta coincides with the peak of Blue at gray levels of 100-120. Networks trained with vanilla BCE as loss detects Magenta regions as Blue.

6.4.2 Red - Blue Confusion

Figure 3.4(a) and 3.3(b) reveals that the peak of Blue also coincides with the sub-peak of Red at similar gray levels of 100-120. Networks trained with vanilla BCE sometimes detects certain red regions as Blue hence.

6.4.3 Intensity Based Regularization

Weak labelling was identified as the major cause for both these confusions. A second regularization term was developed. Our objective being penalizing the neural network if it predicts a pixel as Blue, that has a gray value not lying within the range of (b_l, b_u). Mathematically this term takes the form:

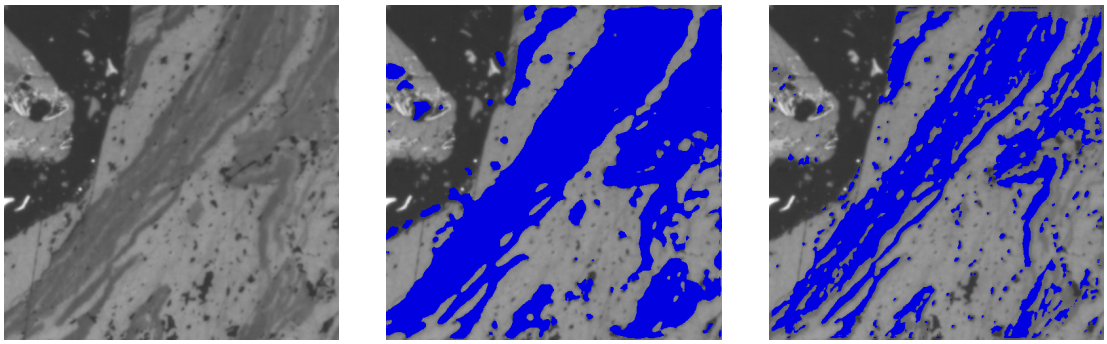
$$R_{blue}(b_l, b_u) = \frac{1}{n_b}\left\{ \sum_{i=1}^{n_b} A_i^{'} \Big/ \sum_{i=1}^{n_b} A_i \right\} \tag{6.5}$$

and the corresponding loss function takes the form:

$$J_{blue}(p, q, b_l, b_u) = H_p(q) + \lambda_3 \times R_{blue}(b_l, b_u) \tag{6.6}$$

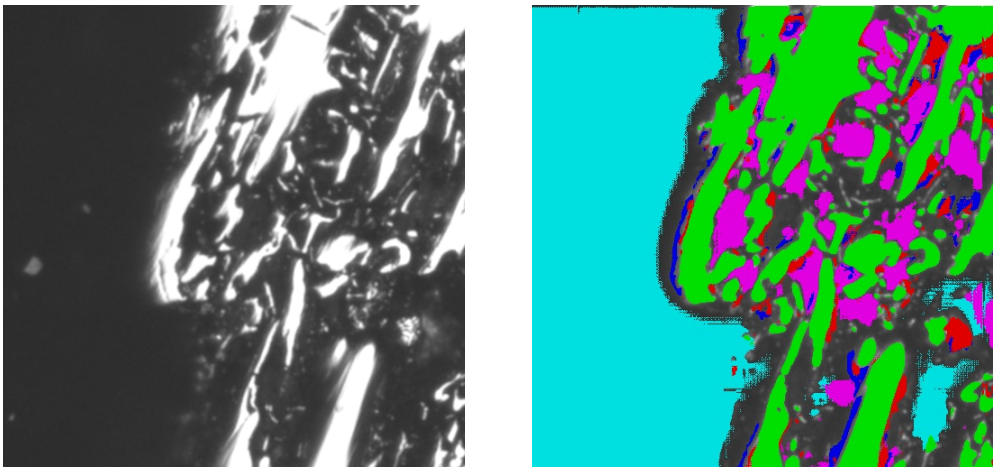
where n_b is the total number of Blue connected components, A_i is the area of the i^{th} connected component, $A_i^{'}$ is the pixel areas within this connected component that does not lie within the gray level range of (b_l, b_u). λ_3 is a scaling hyperparameter similar to λ_1 and λ_2. Other symbols have their usual meaning.

Intuitively, having such a loss function prompts Gradient Descent to change the weights of Blue predictor in such a way that more refined and accurate contours are predicted. Figure 5.8 demonstrates the advantage of using intensity based regularization technique.



(a) Input Image (b) Only BCE (c) BCE + Regularization

Figure 6.9: Predictions of Blue Classifier without and with Intensity Regularization



(a) Input Image (b) Coarse Segmentation

Figure 6.10: Defining the problem: The Coarse Segmentation

6.4.4 Coarse Segmentation

The majority classes of Red and Green were trained using vanilla BCE losses only. Certain confusion among them prevailed. Altough we tried to develop similar regularizers the results obtained were inferior compared to their BCE counterparts. These confusions are resolved using novel image processing algorithms described later. The output of the five indivual classifiers are amalgamated to generate a partialy segmented image that we name as Coarse Segmentation. Figure 5.9 demonstrates a sample and its coarse segmentation.

6.5 Effects on weight updation in Backpropagation

As we have added new regularization terms to BCE loss, (refer to equations 6.1-6.6, the weight update equations during backpropagation will change. The mathematical derivation is elaborated next. We have followed the same notation used by Zhifei Zhang [33].

6.5.1 Forward-propagation Equations

The following equations can be used to describe the forward propagation operation. They are described as follows:

6.5.2 Parameter Definitions

$$k^x_{p,q} \rightarrow \ Convolution\, Kernel\, Weights,\ b^x_p \rightarrow Bias\, value, \tag{6.7}$$

where x represents the x^{th} layer, p is the number of feature maps in the x^{th} layer, q is the the number of kernels or the number of feature maps to be extracted in the $(x+1)^{th}$ layer.

6.5.3 Convolution

The convolution operation is described mathematically by the following equations: For the 1^{st} convolution layer,

$$C^1_{1,p} = \sigma(I * k^1_{p,q} + b^1_p), \tag{6.8}$$

$$C^1_{1,p}(i,j) = \sigma\Big(\sum_{u=-1}^{1}\sum_{v=-1}^{1} I(i-u, j-v)\cdot k^1_{p,q}(u,v) + b^1_p\Big), \tag{6.9}$$

where $\sigma(x) = max(x, 0)$ is the ReLU activation function, I is the input image, $C^1_{1,p}$ is the feature map obtained. $C^1_{1,p}(i,j)$ is the value of the $(i,j)^{th}$ pixel in $C^1_{1,p}$. $k^1_{p,q}(u,v)$ is the weight value at the $(u,v)^{th}$ location of $k^1_{p,q}$. $I(i-u, j-v)$ is the $(i-u, j-v)^{th}$ pixel of I.

In general for the x^{th} convolution layer the equations will be:

$$C^x_{p,q} = \sigma(C'^{\,x}_{p,q} * k^x_{p,q} + b^x_p), \tag{6.10}$$

$$C'^{\,x}_{p,q} = \Phi(C^x_{p,q}),\ where\ \Phi\,(x)\ \epsilon\,\{0,\ C^x_{p,q}\}\ chosen\, randomly, \tag{6.11}$$

$$C^x_{p,q}(i,j) = \sigma\Big(\sum_{w=1}^{p}\sum_{u=-1}^{1}\sum_{v=-1}^{1} C'^{\,x}_{w}\cdot k^x_{w,q}(u,v) + b^x_w\Big), \tag{6.12}$$

where previously mentioned symbols have their usual meaning, w represents the w^{th} feature map. $k^x_{w,q}(u,v)$ represents the kernel weight at the $(u,v)^{th}$ location of the w^th kernel of $k^x_{p,q}$.

6.5.4 Maxpooling

The maxpooling operation in our case is described mathematically as:

$$S_q^x = max\Big(C_q^x(2i, 2j),\ C_q^x(2i-1, 2j-1),\ C_q^x(2i-1, 2j),\ C_q^x(2i, 2j-1)\Big), \quad (6.13)$$

where S_q^x is the result of the maxpooling operation using a 2×2 kernel having stride 2. q is the number of feature maps.

6.5.5 Transpose Convolution

The transpose convolution operation is described as follows:

$$C_l^x(i, j) = \sum_{0 \le (i-u) \le L} \Big(\sum_{0 \le (j-v) \le L} C_l^{x-1} \cdot k_l(u, v) \Big), \quad (6.14)$$

where $C_l^x(i, j)$ is the $(i, j)^t h$ pixel of the l^{th} feature map in the output of the deconvolution, $k_l(u, v)$ is the kernel weight at location (u, v) of the l^{th} feature map.

6.5.6 Back-propagation Equations

We derive the effect on weight updation during back-propagation for the final $1x1$ convolution layer here. For previous layers the equation will be the same as thsi, multiplied by some additional terms that occur as consequences of the chain rule.

Consider equation 6.3. Let us try to compute the partial derivative of $J_{cyan}(p, q, A_c)$ with respect to the predicted pixel $p(y_i)$. The corresponding ground truth pixel is y_i (say). The derivative is expressed as:

$$\frac{\partial J_{cyan}(p, q, A_c)}{\partial p(y_i)} = \frac{\partial H_p(q)}{\partial p(y_i)} + \lambda_1 \cdot \frac{\partial R_{cyan}(A_c)}{\partial p(y_i)}, \quad (6.15)$$

where,

$$\frac{\partial H_p(q)}{\partial p(y_i)} = -\frac{1}{N} \left\{ y_i \cdot \frac{1}{p(y_i)} + (1 - y_i) \cdot \Big(1 - \frac{1}{p(y_i)}\Big) \right\}. \quad (6.16)$$

Next consider equation 6.1. Let y_i belong to the s^{th} connected component A_s. Then:

$$A_s = \sum_{t=1}^{\beta} y_t \ \ where \ \ p(y_i) \, \epsilon \, \{y_1, y_2, \ldots y_\beta\}, \quad (6.17)$$

Therefore,

$$\frac{\partial R_{cyan}(A_c)}{\partial p(y_i)} = \frac{1}{n_c}\left\{\frac{\sum_{s=1}^{n_c} A_s - \sum_{A_s \leq A_c} A_s}{\left(\sum_{s=1}^{n_c} A_s\right)^2}\right\}. \tag{6.18}$$

Now the derivative of ReLU is either 0 and 1, assuming that the input to ReLU is greater than 1 it's derivative will be 1

Now we know that,

$$p(y_i) = \sigma\left(C_{p,1}^F\right) = \sigma(C_{p,1}^{'F-1} * k_{p,1}^{F-1} + b_p^{F-1}) \tag{6.19}$$

where F represents the final layer. $F-1$ is the penultimate 1×1 convolution layer having weights $k_{p,1}^{F-1}$.

Therefore the final gradient with respect to $k_{p,1}^{F-1}$ becomes:

$$\frac{\partial J_{cyan}(p,q,A_c)}{\partial k_{p,1}^{F-1}} = \frac{\partial J_{cyan}(p,q,A_c)}{\partial C_{p,1}^F} \cdot \frac{\partial C_{p,1}^F}{\partial k_{p,1}^{F-1}}. \tag{6.20}$$

So the final weight update rule can be expressed as:

$$\Delta k_{p,1}^{F-1} = -\eta \cdot \left\{ -\frac{1}{N}\left\{ y_i \cdot \frac{1}{p(y_i)} + (1-y_i) \cdot \left(1 - \frac{1}{p(y_i)}\right)\right\} + \lambda_1 \cdot \frac{1}{n_c}\left\{\frac{\sum_{s=1}^{n_c} A_s - \sum_{A_s \leq A_c} A_s}{\left(\sum_{s=1}^{n_c} A_s\right)^2}\right\}\right\}. \tag{6.21}$$

Similarly back-propagation equation can be derived for $J_{magenta}(A_m)$ as well.

6.6 From Coarse to Fine : Image Processing based Correction Algorithms

Figure 6.10(b) reveals that the coarse-segmentation is indeed coarse. This segmentation,is accurate in identifying the classes yet it cannot be accepted as the final result due to major shortcomings.Four image processing algorithms were to overcome these shortcomings .

6.6.1 Shortcomings of the Coarse Segmentation

1. It has unclassified pixels. There are two major reasons for this.

(a) The principal reason behind this is the One-vs-All approach itself. We have used five independent classifiers that predict that a particular pixel belongs to the current class or not. Hence there will always be certain pixels that will be rejected by all five classifiers. Such pixels will hence remain unclassified.

(b) It was also seen that some neural networks trained with our self-developed loss functions often predicted low values of probability (below the probability threshold chosen) for border pixels of the detected region. Such pixels hence remain unclassified.

2. Figure 6.7(c) and 6.8(c) shows that even after using regularization some regions of Magenta still remains classified as Cyan. Magenta classifier sometimes leaves out few pixels surrounding the detected regions.

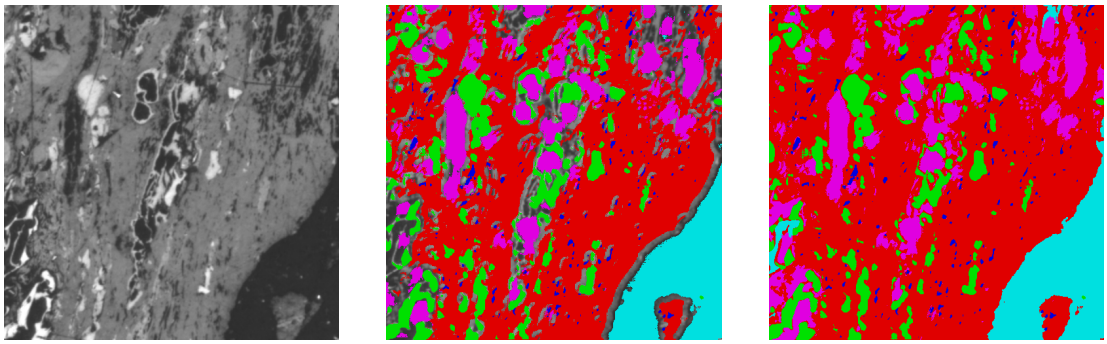
3. We discussed five confusions in overall. Two of the above has been solved. Three more confusions namely the Red-Green confusion, Red-Blue and Magenta-Blue confusions remains to be solved. Pixels affected by these confusions needs to be corrected.

6.6.2 Image Processing based Algorithms

Four algorithms were developed to deal with the shortcomings of the coarse-segmentation.

6.6.3 Border Correction

It was observed that the classifiers of Red and Cyan were leaving a small border of unclassified pixels around the periphery of their predicted regions. This was not the case for the other three classes. We resolved this by a very simple algorithm:



(a) Input Image (b) Before Correction (c) After Correction

Figure 6.11: Applying Border Correction to Coarse Segmentation

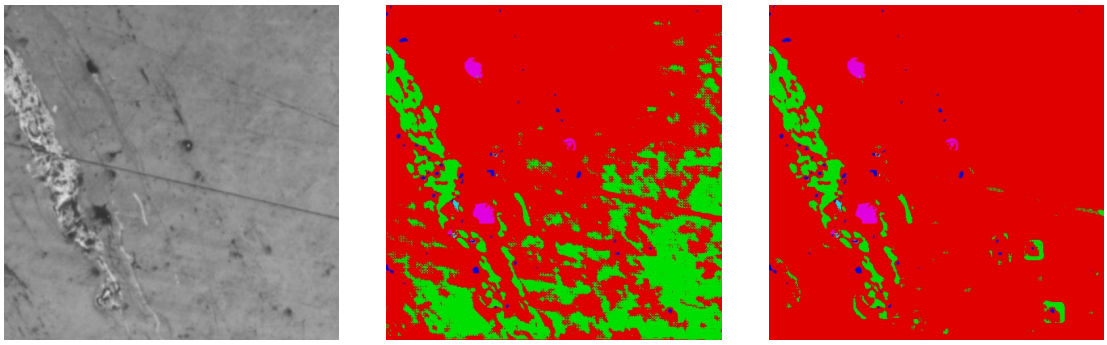
Algorithm 1 Border Correction

Input: X - Segmented Image, c_t - Graylevel threshold for Border Correction
Output: $X^{'}$ - Border Corrected Segmented Image

```
X' ← copy X
while c is a pixel in X' do
    if c is unclassified then
        if c_graylevel < c_t then
            c_label ← cyan
        else
            c_label ← red
        end if
    end if
end while
return X'
```

6.6.4 Uniformity based Correction

We have elaborated about the Red-Green confusion before. This algorithm was developed to resolve this confusion. It was observed that Red (Vitrinite) has a more uniform texture having mid-range graylevels. while Green (Inertinite) had a more non-uniform texture. The maceral itself is nearly white and has nodes of mineral deposits embedded within it. These nodes are nearly black. Hence a neighbourhood around a Red pixel would have a smaller standard deviation compared to that of a Green pixel. This information was exploited to determine the class label of a confused pixel with a decision based on the standard deviation of it's neighbourhood. This standard deviation threshold is chosen midway between the mean of neighbourhood standard deviations of all pixels marked Red and Green in our dataset respectively. Figure 5.12 demonstrates the effect of Uniformity Based Correction.



(a) Input Image (b) Before Correction (c) After Correction

Figure 6.12: Applying Uniformity based Correction

Algorithm 2 Uniformity Based Correction

Input: X - Segmented Image, Y - Input Image R - Red Prediction, G - Green Prediction, n - neighbourhood size, u_t - Std dev threshold for uniformity correction
Output: $X^{'}$ - Uniformity based Corrected Image

```
X' ← copy X
while c is a pixel in X do
    R_c ← Red Prediction at c
    G_c ← Green Prediction at c
    W_c ← Neighbourhood of size n centered at c, made from Y
    if R_c is True and G_c is True then
        c_stddev ← Calculate standard deviation of W_c
        if c_stddev < u_t then
            c_label ← Red
        else
            c_label ← Green
        end if
    end if
end while
return X'
```

6.6.5 Region based Correction

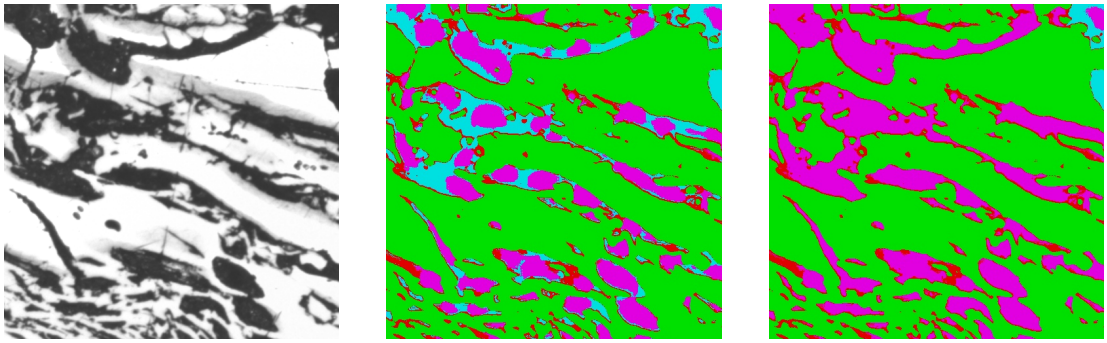
A close inspection of Figure 6.7(e) and 6.8(e) will reveal that certain Magenta pixels remain undetected while certain regions of Magenta are still detected as Cyan. This algorithm was developed to eliminate both these problems. This correction operation should be applied post Border Correction. The Border Correction operation would fill unclassified Magenta pixels as Cyan. Note that our Magenta Classifier only misses out small regions near larger, correctly detected Magenta regions. Hence these newly marked Cyan pixels always lie within the neighbourhood of some Magenta region. This algorithm takes up a Magenta region (say B_m) one at a time. It then takes a pixel x within B_m and checks for cyan pixels within a 21×21 neighbourhood centered at x. If a cyan pixel y is detected, region growing is applied from y to determine the connected component B_c it belongs to. If the area of B_c is less than the area threshold A_c (refer equation 5.1) then this region is actually Magenta, and all pixel labels are corrected from Cyan to Magenta. This is repeated for all pixels present in B_m. This algorithm also corrects the reverse case as well. If we find that the pixel area of B_c is much larger than A_c and that of B_m is less than A_m(refer equation 6.2) then B_m is actually Cyan and all pixels present here are relabelled as Cyan. Figure 6.13 demonstrates the benefits of Region based Correction.

Algorithm 3 Region Based Correction

Input: X - Segmented Image, C - Cyan Prediction, M - Magenta Prediction, A_t - area threshold for region based correction

Output: $X^{'}$ - Region based Corrected Image

$X^{'} \leftarrow$ copy X
$C_c \leftarrow$ Connected components of C
$C_m \leftarrow$ Connected components of M
while C_i is a connected component in C_m **do**
  while m is a pixel in C_i **do**
    $W_m \leftarrow$ Neighbourhood of size 21×21 centered at p , made from X
    $confusion \leftarrow False$, i.e no Cyan in W_m
    while p is a pixel in W_m **do**
      $R_p \leftarrow$ Cyan Prediction at p
      if R_p is $True$ **then**
        $confusion \leftarrow True$
        $C_p \leftarrow$ connected component of Cyan having p
        $A_p \leftarrow$ Area of C_p
      end if
    end while
    if $confusion$ is True **then**
      if $A_p < A_t$ **then**
        $x_{label} \leftarrow$ Magenta, $\forall\, x \,\epsilon\, C_p$
      else
        $x_{label} \leftarrow$ Cyan, $\forall\, x \,\epsilon\, C_i$
      end if
    end if
  end while
end while
return $X^{'}$



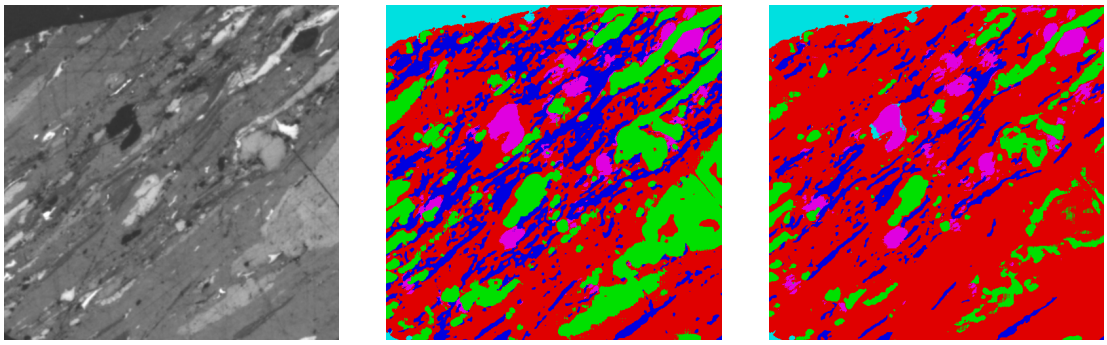
(a) Input Image (b) Before Region Correction (c) After Region Correction

Figure 6.13: Applying Region based Correction

6.6.6 Shape based Correction

Figure 4.3(a) and 4.3(b) displays that the second peak of Magenta coincides with the peak of Blue. This leads to confusion between these two classes with each being misclassified as the other. Figure 4.3(b) and 4.4(a) reveals that Blue has some overlap with Red as well. Hence some Red regions are detected as Blue. We also have certain poorly illuminated ground truth images that makes the Blue classifier misclassify certain Cyan regions as Blue.

It was noted that most Blue regions occur as long thin strands while Magenta usually have a more roundish shape and Cyan and Red have no such shape based distinctive property. This property was exploited to develop our final correction algorithm which we describe next. Area-over-perimeter-square is a good measure of how thin and long a shape is. The shape of the Blue ground truth regions were studied and a threshold value of area-over-perimeter-square is determined, say B^{Th}_{aops}. We take the predictions of all Red, Magenta, Cyan and Blue classifiers. We take each connected component of Blue and calculate its area over perimeter square, say B_{aops}. If $B_{aops} < B^{T}_{aops}$ we keep it unchanged else we find the overlaps of B with the corresponding Magenta, Cyan and Red components M, C and R. All pixels in B are corrected to the class which has maximum overlap with B. The algorithm is described in the next page.



(a) Input Image (b) Before Shape Correction (c) After Shape Correction

Figure 6.14: Applying Shape based Correction

Algorithm 4 Shape Based Correction

Input: X - Segmented Image, C - Cyan Prediction, M - Magenta Prediction, B - Blue Prediction, B^T_{aops} - area over perimeter squared threshold for shape based correction

Output: $X^{'}$ - Shape based Corrected Image

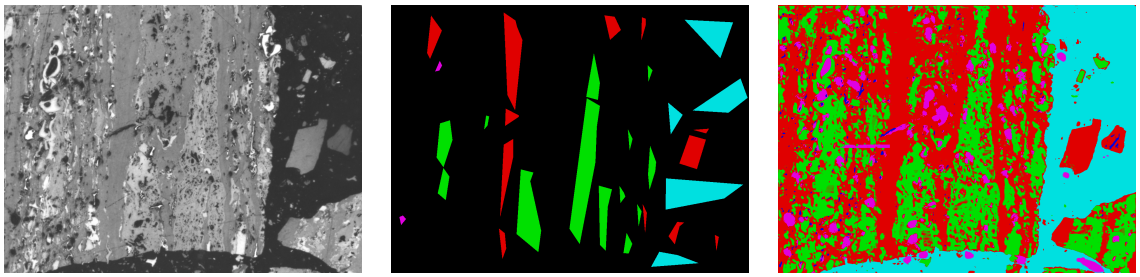
$X^{'} \leftarrow$ copy X
$C_b \leftarrow$ Connected components of B
while C_i is a connected component in C_b **do**
  $B_{aops} \leftarrow$ area over perimeter square of C_i
  if $B_{aops} > B^T_{aops}$ **then**
    $O_r \leftarrow 0$, initializing overlap of C_i with R
    $O_m \leftarrow 0$, initializing overlap of C_i with M
    $O_c \leftarrow 0$, initializing overlap of C_i with C
    while x is a pixel in C_b **do**
      if x in M **then**
        $O_m \leftarrow$ append x
      end if
      if x in C **then**
        $O_c \leftarrow$ append x
      end if
      if x in R **then**
        $O_r \leftarrow$ append x
      end if
    end while
  end if
  $label \leftarrow max(O_r, O_m, O_c)$
  if $label$ equals Red **then**
    $x_{label} \leftarrow$ Red, $\forall\, x \,\epsilon\, C_i$
  end if
  if $label$ equals $Magenta$ **then**
    $x_{label} \leftarrow$ Magenta, $\forall\, x \,\epsilon\, C_i$
  end if
  if $label$ equals $Cyan$ **then**
    $x_{label} \leftarrow$ Cyan, $\forall\, x \,\epsilon\, C_i$
  end if
end while
return $X^{'}$

Chapter 7

Results and Inferences

This chapter elaborates the results obtained during this research work. Figure 7.1 displays an example input image, it's weakly labelled ground truth data and the corresponding segmentation obtained from our Deep Learning based classifier. We have benchmarked our method's performance by comparing with those obtained from a Minimum Distance Classifier(referred to as MDC henceforth) created in accordance with Mukherjee and Uma Shankar's original work [1] and the actual results of the Random Forest Classifier of Mukherjee and Paul's work [2].

A visual comparison of the results obtained from the three methods is carried out first. The pro's and con's of each method is duly noted and how and where our method beats the performance of the other two is reported.The confusion matrices as well as the receiver operating characteristics obtained form the three methods is reported next. Inferences drawn are duly reported. Finally the phase fractions of all five datasets (each having 300-400 test images) have been calculated and compared with the results of the existing non-automated industry standard procedure as well as with the results obtained from the Random Forest based approach.

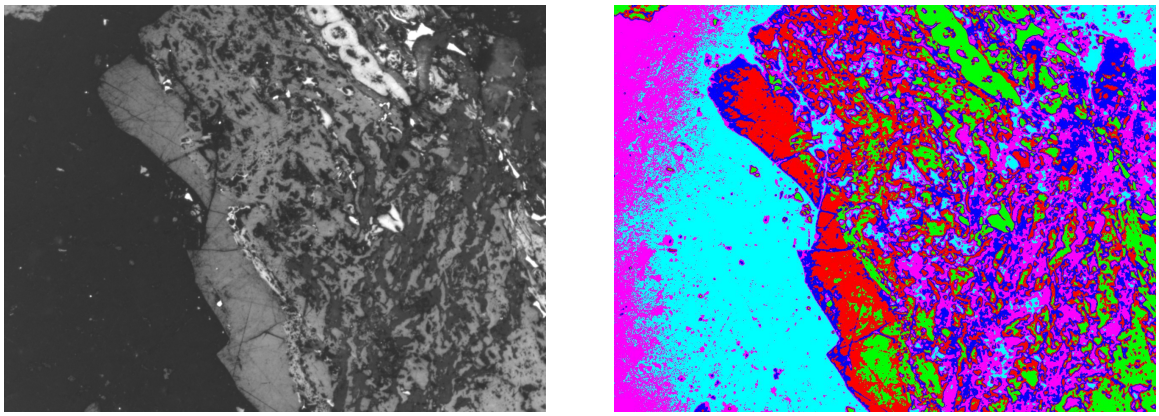


(a) Input Image (b) Ground Truth (c) Obtained Segmentation

Figure 7.1: A sample Input Image, corresponding Ground Truth and segmentation obtained from our Deep Learning based Classifier.

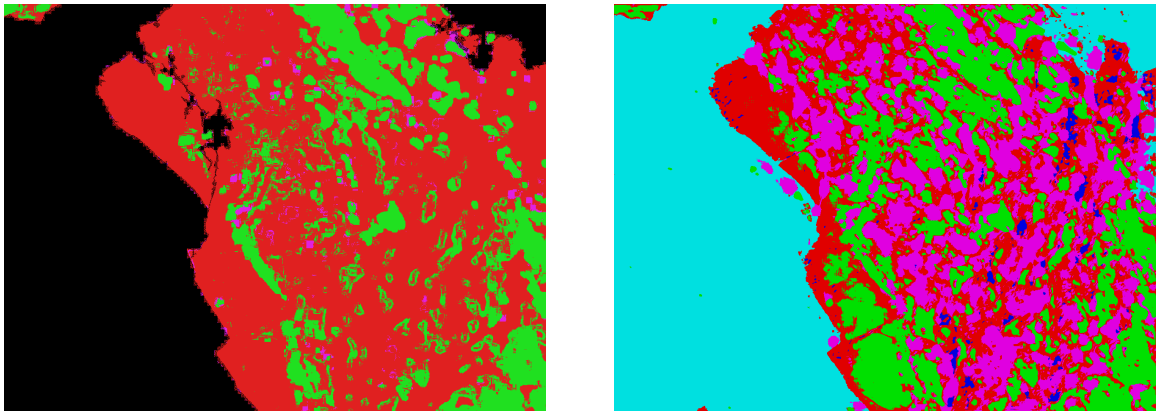
7.1 Visual Comparison

Figure 7.2 and Figure 7.3 shows sample segmentations obtained from the MDC, RF and DL classifers. Figure 7.4 provides another set of examples placed side by side for comparison. Note that in Figure 7.3(a) and Figure 7.4(c) black represents the background Cyan class.



(a) Input Image (b) MDC Output

Figure 7.2: An Input image and corresponding segmentation by MDC

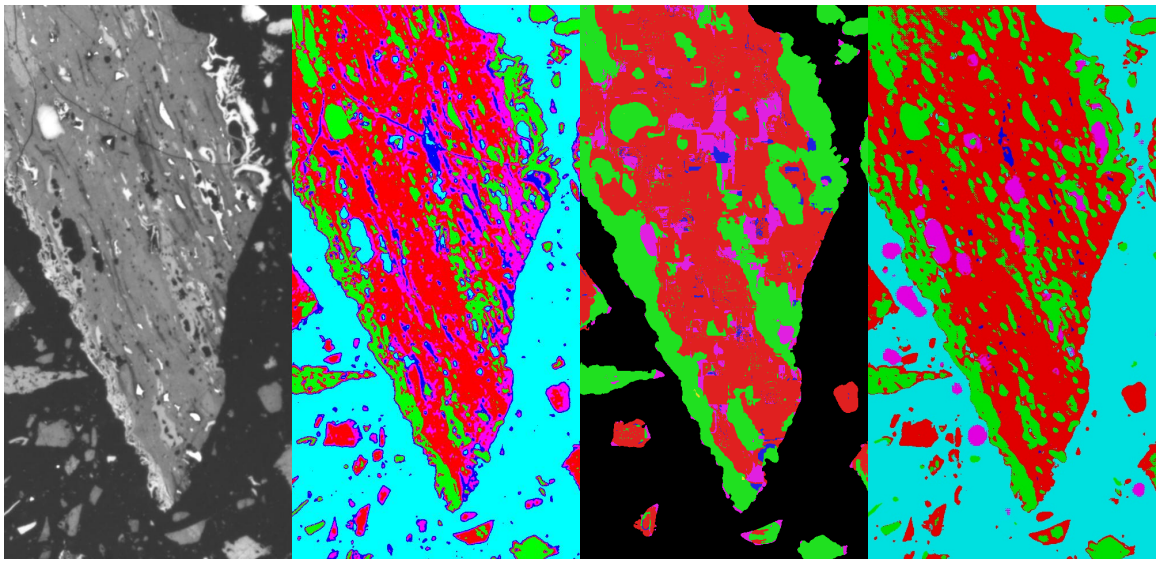


(a) RF Output (black represents cyan) (b) DL Output

Figure 7.3: Corresponding segmentations by RF and our DL Classifier.

7.1.1 Inferences

1. An initial look at Figures 7.2, 7.3 and 7.4 reveals that the MDC output is most intricately detailed while RF output has the least amount of detail. However a detailed inspection reveals that MDC is the least accurate of the three methods.



(a) Input mage (b) Segmentation(MDC) (c) Segmentation(RF) (d) Segmentation(DL)

Figure 7.4: Visual Comparision of segmentations of MDC, RF, and our DL method.

2. The Cyan-Magenta confusion is very distinctive in MDC. Figure 7.2(b) and 7.4(b) shows that a lot of Cyan has been detected as Magenta whereas all Magenta has been detected as Cyan. Moreover most Blue regions have been detected as Magenta and a border of Magenta and Blue surrounds all detected regions. This is undesirable.

3. Figures 7.3(c) and 7.4(c) shows that the RF output is devoid of the Cyan-Magenta confusion. However this result is much less detailed than the other two. It detects majority classes of Red and Green as approximate patches instead of following the exact contours of the regions. Magenta detection is improved compared to MDC but still not upto the mark. However it still suffers from the same Magenta and Blue bordering problem. Moreover small pieces of macerals(check bottom left of each image of Figure 7.4) are not detected by RF. This too is undesirable.

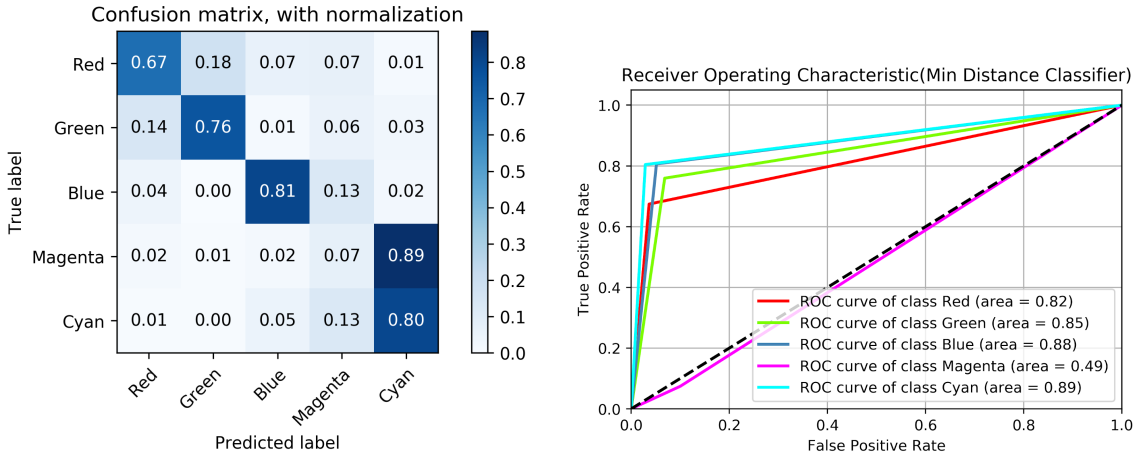
4. Figures 7.3(b) and 7.4(d) shows that our DL classifier generates an output that is marginally less intricate than that of the MDC but it is the most accurate. Cyan-Magenta confusion is eliminated. It shows much superior Magenta detection capabilities, Blue detection is at superior than that of RF and MDC as well. Unlike RF, it does not fail to detect the tiny maceral deposits. Most importantly it does not suffer from the Magenta-Blue bordering problem. These claims are supported mathematically in the next section.

7.2 Confusion Matrices and ROC Curves

Figures 7.5, 7.6 and 7.7 displays the confusion matrices and receiver operating characteristics of MDC, RF and DL classifiers. They provide a mathematical verification of inferences drawn by visual inspection described previously.

7.2.1 Inferences

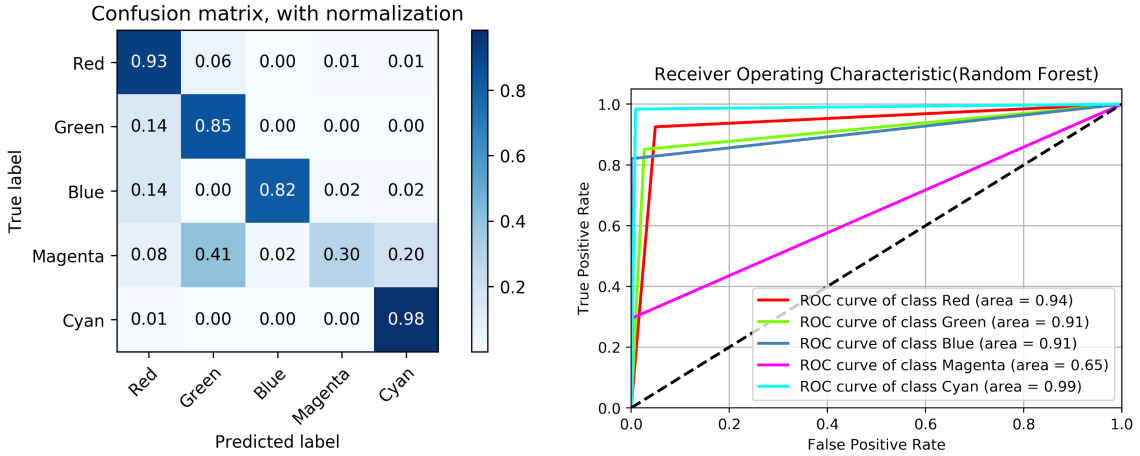
- Comparing Figure 7.5(a) with Figure 7.6(a) and 7.7(a) reveals several interesting facts.7.5(a) and 7.6(b) shows that the MDC classifier completely fails in detecting Magenta. 89% of Magenta has been detected as Cyan.This was observed visually as well.
- Figure 7.6(a) and 7.6(b) shows that performance of RF is superior compared to MDC. Red, Green and Cyan have been detected too a significant level of accuracy. But only 30% of Magenta is correctly detected. About 45% is misclassified as Green and 20% as Blue.
- Figure 7.7(a) and 7.7(b) shows that he DL classifier we developed, performs superior to both MDC and RF. 86% of all Magenta and Blue has been detected accurately. This was the main challenge of this research problem i.e to detect the highly imbalanced minority classes as shown previously in Figure 4.2(a).



(a) Confusion Matrix

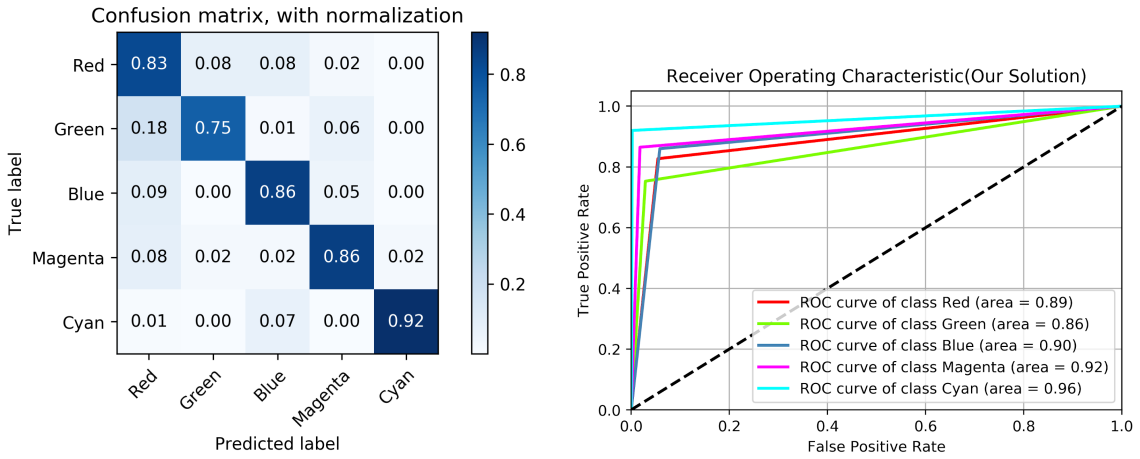
(b) Receiver Operating Characteristic

Figure 7.5: Confusion Matrix and ROC curve of Min Distance Classifier.



(a) Confusion Matrix (b) Receiver Operating Characteristic

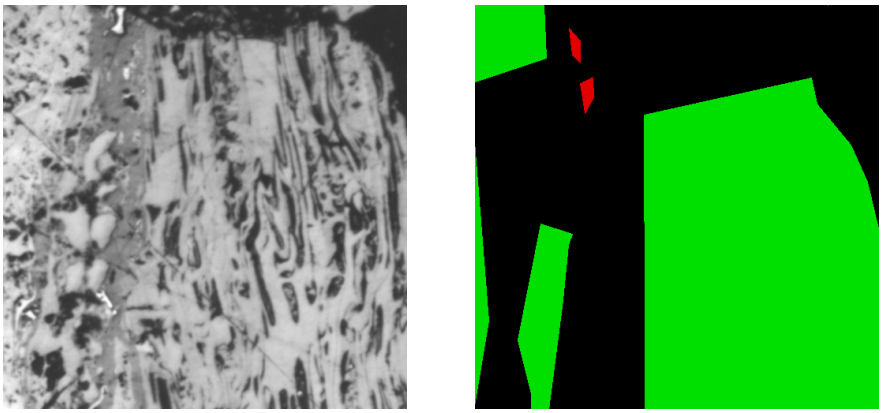
Figure 7.6: Confusion Matrix and ROC curve of Random Forest Classifier



(a) Confusion Matrix (b) Receiver Operating Characteristic

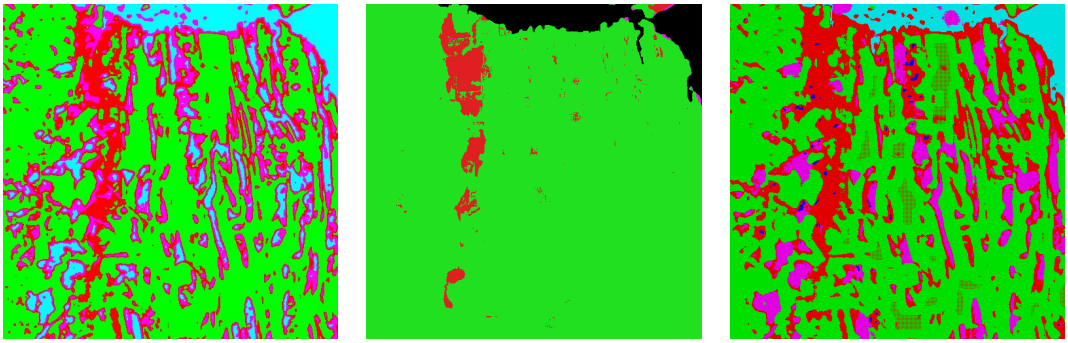
Figure 7.7: Confusion Matrix and ROC Curve of Deep Learning Classifier

- It seems that the accuracy of majority classes of our method is marginally lower compared to RF method, but that is an acceptable amount of decrease when compared to the improvements obtained in detecting the majority classes. The highest decrease is observed in Green where accuracy has dropped down to 75%, this value however is misleading. This is a consequence of the weak labelling which we explain next.



(a) Input Image (b) Ground Truth

Figure 7.8: Explaining the Green Confusion: Input and Ground Truth.



(a) Segmentation(MDC) (b) Segmentation(RF) (c) Segmentation(DL)

Figure 7.9: Explaining the Green Confusion: Segmentations of MDC, RF and DL.

- Figure 7.8(a) shows that Magenta regions(Mineral) is embedded within Green (Inertinite). But weak labelling of Ground Truth (Figure 7.8(b)) marks both Magenta and Green as Green. Figure 7.9(a), 7.9(b) and 7.9(c) shows that neither MDC nor RF has detected these embedded Magenta Regions while our DL classifier has accurately detected them. But as the Ground Truth of all much Magenta Regions has been incorrectly labelled as Green so the confusion matrix of RF will show that the accuracy of RF for Green is more than that of DL. Therefore although the confusion matrix misleads us in believing that RF detects green better in reality our DL classifiers detection of Green is superior.

7.3 Comparing Phase Fractions

Phase Fraction is our final metric of comparison. It has been defined in the prerequisites chapter. This is also the principal method of testing our model on completely

unseen data. Our training images being weakly labeled provided us with a combination of training as well as test data. The following table compares Phase Fractions obtained from the manual industry standard, RF method and our method. Our results have deviated quite a bit. We have 8 threshold values. We were unable to find the optimum set values due to lack of time as this test is run on 1500 images with approximately 5 minutes required for generation of each image. Finding that set will improve our values to a great extent.

Sample	Manual	Random	Deep
Coal A (lower rank)	Vitrinite=75% Inertinite= 18% Liptinite= 3% Mineral = 4%	Vitrinite=69% Inertinite= 16% Liptinite= 9% Mineral = 7%	Vitrinite=71% Inertinite= 7.1% Liptinite= 9% Mineral = 10.5%
Coal B (lower rank)	Vitrinite=47.7% Inertinite= 47.2% Liptinite= 0.3% Mineral = 4.8%	Vitrinite=47.5% Inertinite= 46.1% Liptinite= 0.9% Mineral = 4.5%	Vitrinite=35% Inertinite= 55.9% Liptinite= 2% Mineral = 6.8%
Coal C (higher rank)	Vitrinite=64% Inertinite= 33.6% Liptinite= 0% Mineral = 2.4%	Vitrinite=54.6% Inertinite= 44.3% Liptinite= 0% Mineral = 1.07%	Vitrinite=34% Inertinite= 56.1% Liptinite= 2% Mineral = 7.5%
Coal D (higher rank)	Vitrinite=57.4% Inertinite= 38.2% Liptinite= 0.6% Mineral = 3.7%	Vitrinite=42.5% Inertinite= 54.7% Liptinite= 0.6% Mineral = 2.1%	Vitrinite=39.14% Inertinite= 43.44% Liptinite= 8.1% Mineral = 9.2%
Coal E (lower rank)	Vitrinite=83.8% Inertinite= 8.7% Liptinite= 4.2% Mineral = 3.3%	Vitrinite=94.4% Inertinite= 4.2% Liptinite= 0.4% Mineral = 1.1%	Vitrinite=54.7% Inertinite= 6.8% Liptinite= 32% Mineral = 6.32%

Table 7.1: Phase fractions of different Coal samples

Chapter 8

Conclusion

This research work presented us a live industrial bottleneck problem. Tata Steel had placed the requirement of an Automated Coal Petrography solution capable of generating fast, accurate segmentations while demanding minimal involvement from petrologists. This work plays a pivotal role in speeding up the manufacturing process and hence boost revenue generation.

The dataset of microscopic images provided, posed several challenges in itself. It was heavily imbalanced, weakly labelled, and had 4 distinct types of intensity based inter-class confusion. Minority class detection, accuracy of segmentation, loss of generalization, misclassification and remaining of unclassified pixels were practical problems that originated as a consequence of the above mentioned nature of the dataset.

We were inspired to approach the problem from the Deep Learning perspective as it removes the painful process of feature engineering. This saved us time to focus on perfecting the classification task instead. We chose U-NET as our classification model, as a thorough survey of related work had revealed that the U-NET is a widely used model that has segmented images of similar textural intricacies in the field of medical images, cell tracking etc.

Dataset imposed challenges motivated us to use 5 binary U-NET classifiers following a One-vs-All Approach instead of a single multiclass classifier. This approach provided us with its own set of demerits and shortcomings and was capable of generating only a coarse segmentation. This imposed on us the need of developing a post-processing module. 4 novel image processing algorithms were developed to remove unclassified and correct misclassified pixels. Finally the fine segmentation was obtained as our final result. We benchmarked our performance against two previous approaches to the problem involving a Minimum Distance Classifier and a Random Forest. Obtained results, confusion matrices and ROC curves revealed that our method is significantly superior to these previously existing methods.

Chapter 9

Future Work

We follow up our work with a future work proposal of a novel neural network architecture and training mechanism that we christen as Nested-Net which we envision as being capable of generating accurate segmentations as a single multiclass classifier having no need of any form of post-processing techniques.

9.1 Resolving hurdles in the path of a Single Multiclass Classifier

- First the problem of Weak Labeling is looked into. Weak Labelling can be resolved by asking petrologists to correct whatever little amount of error that persisted in the results obtained from our Deep Learning Classifier. The segmentation obtained post this correction is 100% accurate and every pixel will now have a marked and verified class label. These images will act as our Ground Truth for our Single Multiclass Classifier and we name them as Strongly Labelled Ground Truth data.

- Data Imbalance is looked into next. This is resolved by carefully choosing the images that we want to use as Strongly Labelled Ground Truth data. Proper choosing leads to significant reduction in imbalance and hence a single classifier now will be able to detect both majority and minority classes with ease.

- If we want to incorporate all the properties of our binary classifiers into a single multiclass classifier imposes new challenges. This is elaborated in the next section.

9.2 Proposed Multiclass Deep Learning Solution

In our One-vs-All approach we had three independent regularization terms for the classes of Cyan, Magenta and Blue namely $R_{cyan}(A_c)$, $R_{magenta}(A_m)$, and $R_{blue}(b_l, b_u)$ (refer to equations 6.3, 6.4, 6.5 and 6.6). When we try to create a single multiclass DL classifier then its loss function should have all these 3 terms added. Moreover we desire to eliminate the post processing algorithms. As these algorithms work in a similar fashion as these regularization terms so three additional regularization terms can be created namely $R_{uniform}(u_t)$, $R_{region}(A_t)$, $R_{shape}(B^T_{aops})$ corresponding to Uniformity, Region and Shape based corrections. Border Correction becomes invalid as we are using a single multiclass classifier so every pixel will have some label, i.e no pixel remains unclassified. The resulting loss function of our desired classifier therefore takes the form:

$$\begin{aligned} J_{cyan}(p, q, b_l, b_u, u_t, A_c, A_m, A_t, B^t_{aops}) = H_p(q) &+ \lambda_1 \times R_{cyan}(A_c) + \lambda_2 \times R_{magenta}(A_m) \\ &+ \lambda_3 \times R_{blue}(b_l, b_u) + \lambda_4 \times R_{uniform}(u_t) \\ &+ \lambda_5 \times R_{region}(A_t) + \lambda_6 \times R_{shape}(B^T_{aops}), \end{aligned} \tag{9.1}$$

where all symbols have their usual meanings and as stated before. This is a complicated loss function having 6 scaling factors $\lambda_1, , \lambda_2, \ldots, \lambda_6$ and 7 threshold values namely b_l, b_u, u_t, A_c, A_m, A_t, B^t_{aops} i.e 13 hyperparameters in total. Depending on the values alloted to these 13 hyperparameters the nature of the loss function changes, and finding the optimal set of values of these hyperparameters (that leads to good segmentation post training), manually is a tedious and near impossible task. We thought of automating this process as well which we elaborate in the next section.

9.2.1 Proposal of Novel Architecture and Training Process

We propose a new type of neural network architecture that we name Nested-Net. The architecture of Nested-Net is shown in Figure 9.1. The architecture has two neural networks, the training of which occurs in a nested fashion hence the name. We use the same dataset to train both these networks. The external network is a Convolution Neural Network that takes input images as batches from our coal dataset and gives 13 numbers as output. These 13 numbers are chosen as the 13 hyperparameters of equation 9.1 as mentioned previously. Initially a batch of input images is passed through the CNN and a loss function is created that has the form of equation 7.1 with the output obtained from the CNN used as the hyperparameters. In Figure 9.1, this loss has been represented as $LOSS_{internal}$.Also the top section shows the CNN model that is acting as our external neural network.

The internal neural network is a multiclass U-NET that is then trained on the entire dataset in batches using this loss function. The middle portion of Figure 9.1 shows

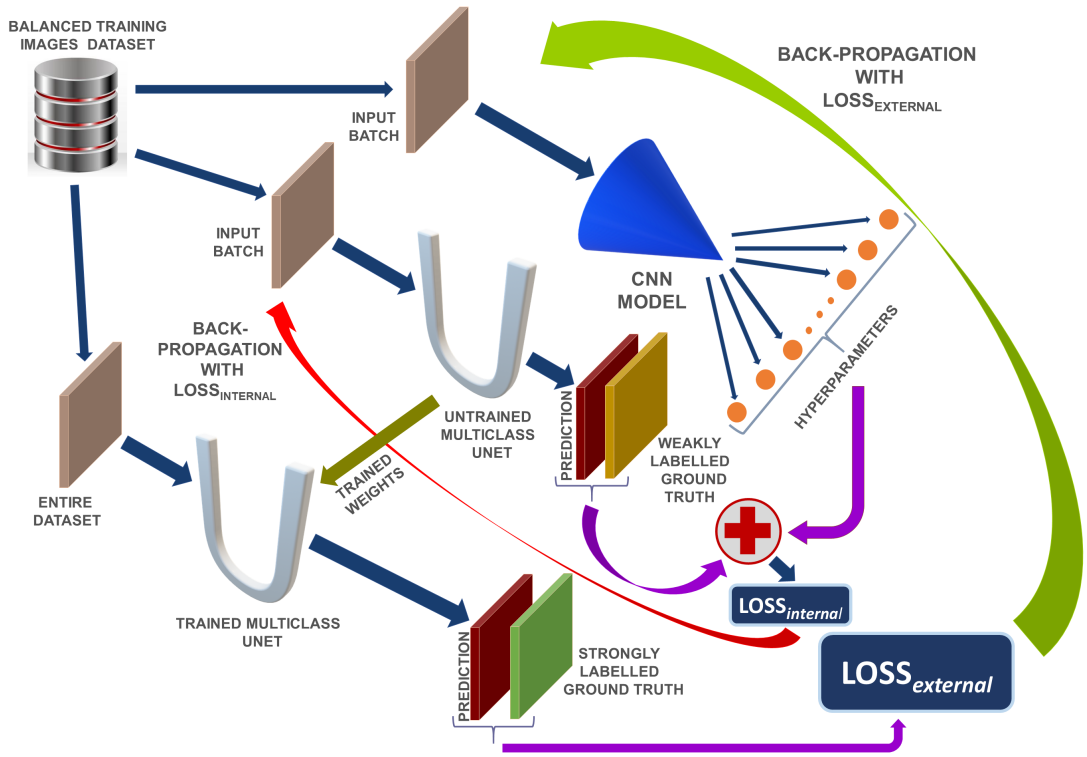


Figure 9.1: Proposed Novel Neural Network Architecture

our untrained multiclass U-NET. A batch of training set images is forward propagated through the untrained U-NET(having randomly initialised weights) to obtain predictions on the current batch. These predictions along with the corresponding weakly labelled ground truth is used to calculate the value of $LOSS_{internal}$. This loss is backpropagted through our multiclass U-NET and weights are updated. In this fashion the untrained multiclass U-Net gets converted into a trained multiclass U-NET. This is represented in the bottom left of the figure.

Next the entire dataset is passed trough our trained U-NET and predictions are obtained. These predictions and the corresponding strongly labelled ground truth data is used to compute simple binary cross entropy loss. This is represented as C. This loss is back-propagated through our external CNN network to update its weight. This network the receives a new batch of training images and it outputs 13 new values of hyperparameters. This gives rise to a new $LOSS_{internal}$ which is used to retrain the internal muticlass U-Net.

The intuition behind this novel approach is that Gradient Descent applied to the external network will try to reduce $LOSS_{external}$.The only way it can do so is if it can make the predictions of the trained multiclass U-Net as accurate as possible. It has no control on the weights of our trained multiclass U-Net. But it can modify the weights of the external network in such a manner such that in the next forward pass through

the external network, 13 new hyperparameter values are generated leading to a more optimal $LOSS_{internal}$ than before. When the multiclass U-NET gets trained with this more optimal $LOSS_{internal}$, its predictions will be ore accurate hence reducing $LOSS_{external}$ as a result. When the external neural network is trained optimal set of hyperparameters have been obtained and the multiclass U-Net has been trained with the optimal $LOSS_{internal}$. Hence the segmentation obtained from it as a result are the best that can be achieved.

The forward and back-propagation mathematics needs to be derived for Nested-Net followed by programming in Tensorflow. All training functions has to be developed as this architecture is a new one. However that is for another day, another research work. As of now we are done!

Bibliography

[1] D. Mukherjee, D. Banerjee, B. U. Shankar, and D. D. Majumder, "Coal petrography: a pattern recognition approach," *International journal of coal geology*, vol. 25, no. 2, p. 155, 1994.

[2] A. Paul, A. Gangopadhyay, A. R. Chintha, D. P. Mukherjee, P. Das, and S. Kundu, "Calculation of phase fraction in steel microstructure images using random forest classifier," *IET Image Processing*, vol. 12, no. 8, pp. 1370–1377, 2018.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[4] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[5] 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*. Springer, 2015, pp. 234–241.

[6] B. Viswanathan, *Energy sources: fundamentals of chemical conversion processes and applications*. Newnes, 2016.

[7] A. George and G. Mackay, "Petrology," in *The Science of Victorian Brown Coal*. Elsevier, 1991, pp. 45–102.

[8] J. Jordan, "An overview of semantic image segmentation," https://www.jeremyjordan.me/semantic-segmentation/, accessed: 2019-06-27.

[9] "Image source: Pascal visual object classes homepage," http://host.robots.ox.ac.uk/pascal/VOC/voc2012/#devkit, accessed: 2019-06-27.

[10] N. Shibuya, "Up-sampling with transposed convolution," https://towardsdatascience.com/up-sampling-with-transposed-convolution-9ae4f2df52d0, accessed: 2019-06-27.

[11] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville, "Adversarially learned inference," *arXiv preprint arXiv:1606.00704*, 2016.

[12] 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*, 2012, pp. 2843–2851.

[13] S. Saha, "A comprehensive guide to convolutional neural networks the eli5 way," https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, accessed: 2019-06-27.

[14] C. E. Shannon, "A mathematical theory of communication," *Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[15] D. Godoy, "A comprehensive guide to convolutional neural networks the eli5 way," https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a, accessed: 2019-06-27.

[16] P. Jaccard, "Étude comparative de la distribution florale dans une portion des alpes et des jura," *Bull Soc Vaudoise Sci Nat*, vol. 37, pp. 547–579, 1901.

[17] J. Jordan, "A comprehensive guide to convolutional neural networks the eli5 way," https://www.jeremyjordan.me/evaluating-image-segmentation-models/, accessed: 2019-06-27.

[18] A. Cauchy, "Méthode générale pour la résolution des systemes d'équations simultanées."

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[21] . J. A. of Remote Sensing, "Minimum distance classifier," http://sar.kangwon.ac.kr/etc/rs_note/rsnote/cp11/cp11-6.htm, accessed: 2019-06-27.

[22] N. Nilsson, "Learning machines: Foundations of trainable pattern-classifying systems. 1965."

[23] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[24] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.

[25] W. Koehrsen, "Random forest image," https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d, accessed: 2019-06-27.

[26] L. Breiman, *Classification and regression trees*. Routledge, 2017.

[27] C. Gini, "Measurement of inequality of incomes," *The Economic Journal*, vol. 31, no. 121, pp. 124–126, 1921.

[28] ——, "Variabilità e mutabilità," *Reprinted in Memorie di metodologica statistica (Ed. Pizetti E, Salvemini, T). Rome: Libreria Eredi Virgilio Veschi*, 1912.

[29] B. Efron, "Bootstrap methods: another look at the jackknife," in *Breakthroughs in statistics*. Springer, 1992, pp. 569–593.

[30] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[31] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[32] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[33] Z. Zhang, "Derivation of backpropagation in convolutional neural network (cnn)," 2016.