2021-08-24

Exploring Convolutional Neural Networks and Transfer Learning for Oil Sands Drill Core Image Analysis

Anzum, Fahim

UNIVERSITY OF CALGARY

Exploring Convolutional Neural Networks and Transfer Learning for Oil Sands Drill Core Image Analysis

by

Fahim Anzum

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN COMPUTER SCIENCE

CALGARY, ALBERTA
AUGUST, 2021

Abstract

An accurate permeability estimate is crucial for effectively characterizing the McMurray oil sands for in situ recovery. Such an estimate is critical to inform the best locations for placing wells and pads and accurately forecast future oil production rates. This fact is becoming significantly important as in situ development moves to areas of increasingly complex geology. The traditional methods of estimating permeability largely do not work well in oil sands because of the core disturbance or the fact that the core is filled with immobile bitumen. Moreover, it is expensive to get physical samples from many different depths at many wells, and the experiments carried out in the labs to measure permeability sometimes are not representative. However, permeability can be estimated from different parameters such as mean grain size (MGS), median grain size, and particle size distribution (PSD). This thesis investigates how convolutional neural networks (CNNs) and transfer learning perform when estimating MGS from the oil sands drill core photos. Three preliminary approaches are explored for classifying core photos based on the facies, including (1) the application of transfer learning on the pre-trained VGG-16 CNN model, (2) fine-tuning a few top layers of VGG-16, and (3) the combination of VGG-16 and traditional machine learning (ML) algorithms. Experimental results achieved by these classification models reveal opportunities to extend these approaches for predicting MGS from core photos. Therefore, the three approaches are then investigated using a library of core photographs with known MGS calculated from PSD to see which one works best. Experimental results exhibit good performance in estimating MGS from core photos using the explored approaches. Overall, the investigation supports that the application of CNNs, and transfer learning is feasible in different oil sands drill core image analysis workflows and more advanced research outcomes can be achieved by further exploration of these techniques in the oil sands research domain.

Acknowledgements

The journey of my graduate studies is supported and guided by my supervisors Dr. Mario Costa Sousa and Dr. Usman Alim. I want to express my immense gratitude to them for spending countless hours discussing and formulating the ideas, improving scholarship applications, and writing recommendation letters for me. Their insightful opinions have helped me to think critically about the research. I am deeply grateful to my supervisors for the mentorship they have provided throughout my M.Sc. study.

I am thankful to Mitacs-Accelerate Graduate Research Internship Program for their financial support, which kept me dedicated to the research. Through this program, I got the opportunity to work closely with an amazing research team at Suncor Energy, the industrial partner of my research project. I am very grateful to Suncor Energy for guiding me and providing me with all the necessary resources to conduct the research. I am also very thankful to Dr. Hamidreza Hamdi for showing me the right research directions, helping me to generate research ideas, providing me helpful resources, and reviewing my research progress throughout my graduate study.

My parents inspired me to stay motivated toward my target. Their encouragement boosted me up to surpass myself and set a new target. I want to thank my parents for upbringing me so that I can complete this milestone with excellence. Finally, I want to thank my beloved wife for supporting me throughout my graduate study. Without her inspiration and emotional support, I could have never been this successful in my graduate studies, and she is the reason for what I have accomplished today.

Last but not least, I would like to share my appreciation for the rest of the examination committee for their insightful and valuable feedback.

— To my lovely wife —

Table of Contents

List of Figures and Illustrations

List of Tables

List of Symbols, Abbreviations and Nomenclature

Symbol or abbreviation	Definition
MGS	Mean Grain Size
PSD	Particle Size Distribution
CNN	Convolutional Neural Network
ML	Machine Learning
DL	Deep Learning
CK	Carman-Kozeny
MAE	Mean Absolute Error
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
CBIR	Content Based Image Retrieval
EOR	Enhanced Oil Recovery
GPUs	Graphics Processing Units
HSI	Hue, Saturation, Intensity
HSV	Hue, Saturation, value
GLCM	Gray Level Co-occurrence Matrix
CCM	Color Co-occurrence Matrix
AD	Alzheimer's Disease
FC	Fully Connected
ONE	Online Nearest-neighbor Estimation
PCA	Principal Component Analysis
PQ	Product Quantization
AI	Artificial Intelligence
KNNs	K-Nearest Neighbors
SVM	Support Vector Machine
RL	Reinforcement Learning
ANN	Artificial Neural Network
tanh	Hyperbolic Tangent
ReLU	Rectified Linear Unit
DLNN	Deep Learning Neural Network
RNN	Recurrent Neural Network
RGB	Red, Green, Blue
ELU	Exponential Linear Unit

SELU	Scaled Exponential Linear Unit
VMI	Visual Mud Index
IHS	Inclined heterolithic stratification
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
ROC	Receiver Operating Characteristic
AUC	Area Under Curve
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
TPR	True Positive Rate
LDS	Laser Diffraction system
HE	Histogram Equalization
AHE	Adaptive Histogram Equalization
CLAHE	Contrast Limited Adaptive Histogram Equalization

Chapter 1

Introduction

Grain size is one of the most fundamental properties of sediment particles that help geologists to understand crucial geological aspects such as permeability, particle size distribution (PSD), and among others. Grain size analysis of drill core, therefore, provides important clues to the depositional settings, sediment provenance, transport history, and geomorphic significance of fluid dynamics in the natural environment ([1], [2], and [3]). The Athabasca oil sands deposit in northeastern Alberta, Canada, is one of the world's largest petroleum reservoirs. Oil sands, also known as "tar sands", are sediments or sedimentary rocks composed of sand, clay minerals, water, and bitumen. The oil is in the form of bitumen, a very heavy liquid or sticky black solid with a low melting temperature. Due to having such large deposits of oil sands, a lot of core drilling is performed. As a result, the cores are also readily available and accessible for geologists to conduct grain size analysis and determine PSD. As the cores are drilled, information related to the cores is recorded, including the core photos. Therefore, as there are plenty of methods to quantify grain size distribution with the availability of grain size data, the photos of the drilled cores paired with the grain size data can be leveraged to determine various crucial geological properties since the laboratory methods of determining these properties are not always representative. With the advancement of convolutional neural networks (CNNs), important information can be obtained using images.

CNN is a sub-field of deep learning (DL) that enables computers or systems to derive meaningful information from digital images, videos, and other visual inputs, and take actions or make recommendations based on that information.

This thesis explores different variations of a CNN-based technique called transfer learning to analyze oil sands drill core photos. These techniques are employed to investigate how they work to classify the core photos based on different facies or rock types and predict mean grain size (MGS) from the core photos. While prior work has analyzed grain size from drill core photos using different machine learning (ML) and DL for conventional drill cores, to the best of our knowledge, there is no published work on predicting facies and MGS from oil sands drill core photos using transfer learning and ML approaches where most textures and grains are rarely visible because of the black bitumen.

1.1 Context and Motivation

Most of today's energy needs are met by fossil fuels like coal, oil, and gas. These unique high-energy fuels are non-renewable resources that took millions of years to form. About 2 billion years ago, marine organisms like algae and microscopic animals and plants died and settled on the ocean floor. In the absence of oxygen, these fossils changed into a substance called kerogen [4]. Under heat and pressure, kerogen gradually changes into oil or gas. The whole process usually takes at least a million years. At a molecular level, oil and gas are hydrocarbons made up of hydrogen and carbon atoms. The constant pressure and movement of the earth's crust squeeze oil and gas through the pores or spaces between rocks. Some oil and gas reach the earth's surface. Often it is trapped beneath the surface by impermeable layers or rock structures. Within the crust, oil or gas deposits build up and form reservoirs [5].

Conventional reservoirs [19] consist of source rock, reservoir rock, and caprock, as shown in Figure 1.1. While the source rock holds the oil and gas's kerogen, the reservoir rock is the porous and permeable rock layers that hold the oil and gas. It is the cap rock that traps

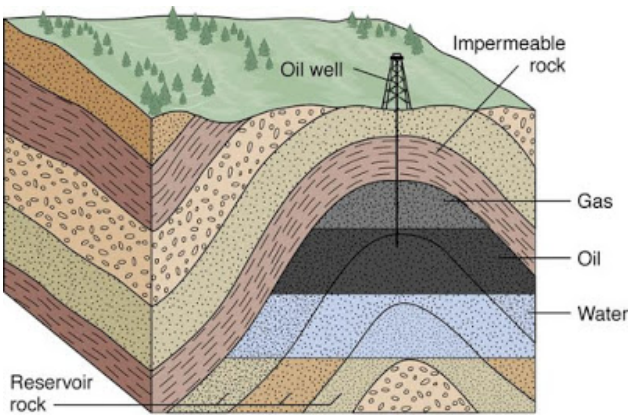


Figure 1.1: Hydrocarbon reservoir [11].

the hydrocarbon in the reservoir. To perform proper well drilling, reservoir rock needs to be porous and permeable. Permeability is a factor that quantifies how hard or how easy it is for the fluid to flow through the reservoir to the oil-producing well. It measures the ability of fluids to flow through rock or other porous media. An accurate permeability estimate is one of the most important parameters to characterize the McMurray oil sands for in-situ recovery. Prior research proposed different correlations to estimate permeability. Although permeability can be calculated using porosity, the results are not reliable as the parameters are uncertain due to the constraint conditions of unconsolidated and homogeneous porous media [100].

The traditional method of estimating permeability in oil sands is a critical task as the core is filled with immobile bitumen [9]. Although lab experiments for estimating permeability are expensive, there is another way to estimate permeability based on the particle size distribution (PSD) index, indicating what sizes of particles are present in what proportions in the sample particle group to be measured [83]. In PSD measurement, the relative particle amount is expressed as a percentage where the total amount of particles is 100%. PSD of a material is an important parameter to determine certain physical and chemical properties of a material. There are several methods of determining the PSD, such as sieve analysis, laser diffraction system, image particle analysis, and particle counting in a Coulter counter [110], [111] and [112]. A brief working principle of sieve analysis and laser diffraction system

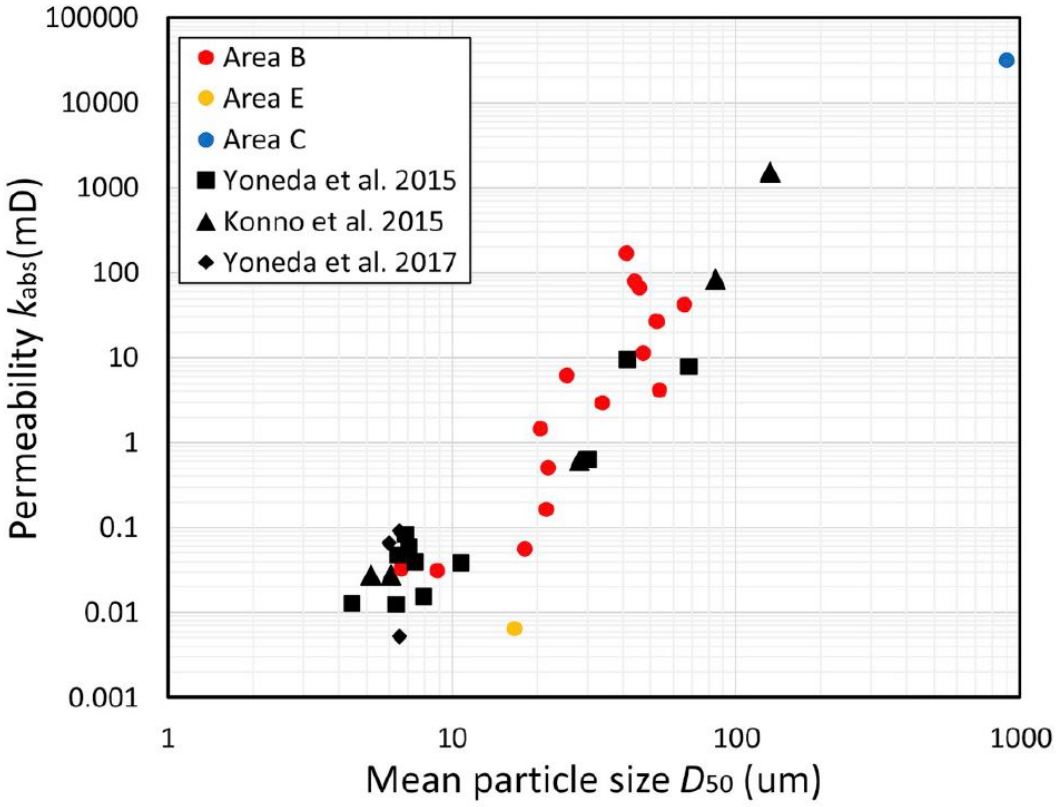


Figure 1.2: Linear correlations between permeability and mean grain size [141].

is presented in Chapter 5. PSD identifies how the porous spaces or particles are close to each other. According to Carman-Kozeny (CK) equation ([24], [25]), permeability can be estimated with the known distribution of pore sizes or particle sizes [28]. Research also shows that the correlation between permeability and MGS is quite high as well [141]. Moreover, it is also a convenient parameter to derive from the PSD. Figure 1.2 shows a nearly linear relationship between permeability and sediment grain size, where the permeability in each case is measured at the in situ stress state. Therefore, we can state that permeability can be also determined if the MGS of the distribution is known.

PSD has been successfully estimated in the mining industry using photographs for a number of years (e.g., [29] [30]). The approach has been used for estimating the PSD of piles of blasted rock or material on conveyor belts. The advantage of this type of material is that the edges of grains can be easily imaged. However, the drill core is more challenging

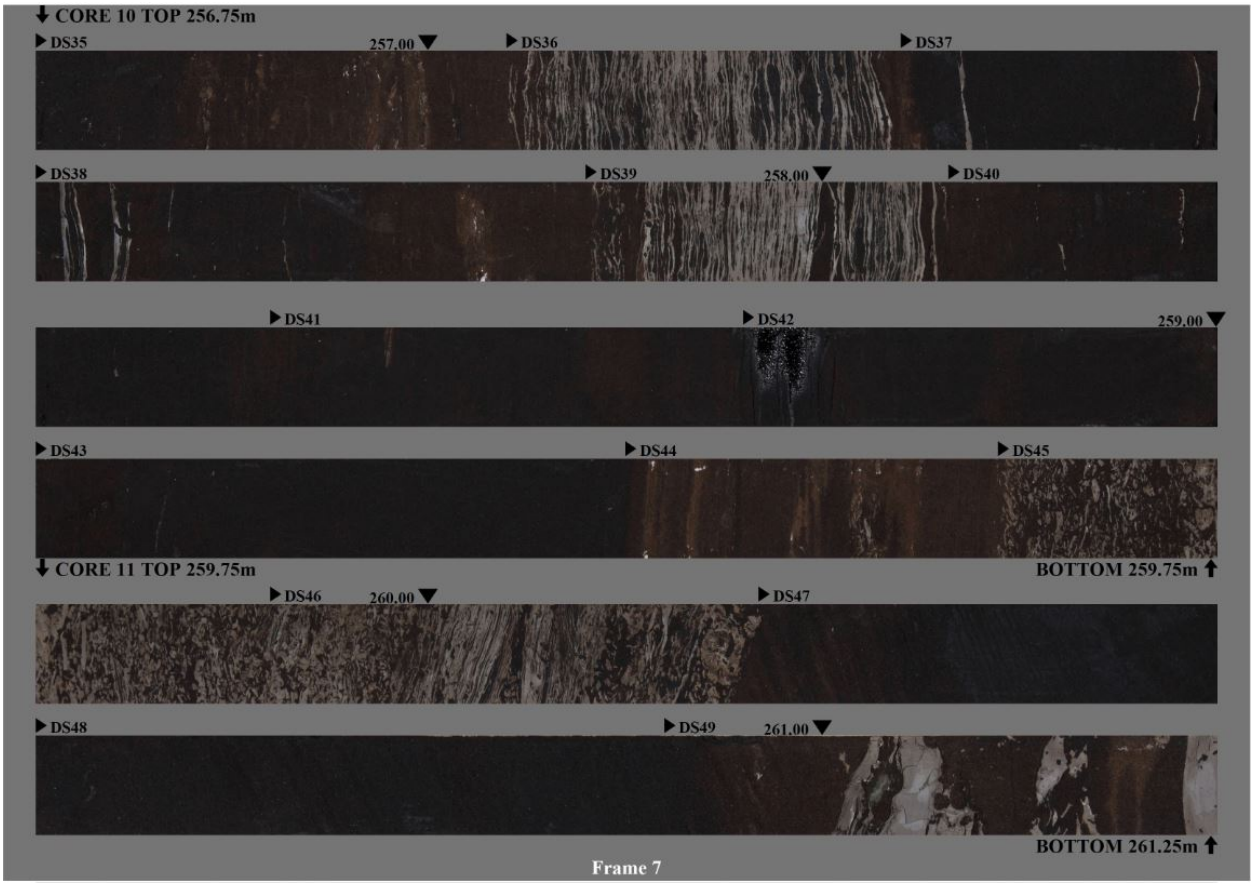


Figure 1.3: Example of core photos collected by Suncor Energy.

because the edges of grains cannot be easily seen on the drill core. Only the part of a grain that is exposed at the surface of the cut core can be imaged. Like an iceberg, the true size of the grain is hidden. Drill core from the Alberta oil sands is particularly challenging since the drill core is also covered in bitumen which interferes with the imaging because parts of the grains are not readily visible. Besides, it is also expensive to get physical samples from many different depths at many wells. For this Thesis, our industry partner and collaborator, Suncor Energy, provided the core photos from the previous cored wells along with the PSD data. An example of some oil sands drill core photos is shown in Figure 1.3.

1.2 Objectives

Different ML and DL techniques have been successfully applied with considerable success in the Geoscience domain for almost two decades. Some examples of the application of ML in the Geoscience research community include seismic-facies classification [84] - [90], CNNs for geological image classification [91], volcanic ash classification using CNNs [92] [93]. However, though DL, CNN, and transfer learning techniques have gained popularity and become established as robust and powerful tools in other scientific fields, these tools are still novel concerning the application within the Geoscience community.

This thesis aims to explore how CNNs, transfer learning, and image-based ML techniques can be utilized for oil sands drill core image analysis. For different experiments, the available data includes a library of raw oil sands drill core photos where each sample is labeled with the MGS calculated from the whole PSD [47]. In order to gain insight about the opportunity to implement the aforementioned techniques to estimate the MGS from core photos, first we aim to investigate the performances of these techniques in core image classification based on the facies or rock types. Relying on the experimental results achieved from this investigation, our goal is to extend the experimental scope for estimating MGS from the core photos. Since extracting useful information from the available oil sands drill core photos are very critical due to their unique characteristics, our objective is not to produce an optimal solution using the explored methods. Rather, our aim is to explore how CNNs, transfer learning, and ML techniques perform with such kinds of data, identify potential challenges, and open doors for extensive future research opportunities in this domain.

1.3 Challenges

Traditional ML applications rely on a set of attributes or features selected or designed by a domain expert. Features are specific characteristics of an object that can be used to distinguish one object from another. For image data, several features can be considered,

such as texture, orientation, and edges. To reduce the dependencies on handcrafted features by the domain expert, DL can be utilized. In computer vision problems, rather than relying on hand-engineered features, CNN models can automatically extract different features from the image data. However, existing DL methods train deep architectures from scratch, which have a few limitations [31] [32]. Training a DL network requires:

- huge amount of annotated training data,
- huge computational resources,
- careful and tedious tuning of many parameters and hyper-parameters, sub-optimal tuning of which can result in overfitting/underfitting and in turn, result in poor performance.

The primary challenge in our work is that the amount of annotated data is not sufficient. Due to having less data than a typical DL model requires, the performance of the DL models is also affected. However, an attractive alternative to training a DL network from scratch is fine-tuning a deep network through transfer learning [33]. In transfer learning, the idea is that a neural network can take the knowledge learned from one task and apply that knowledge to a separate task. For example, if a CNN model has the knowledge to recognize the images of cats and dogs, the same knowledge or a part of that knowledge can be used for a medical image classification task. In transfer learning, the knowledge from a neural network pre-trained on a vast data library is leveraged and applied to another problem with a smaller dataset. In Chapter 3, the concept of transfer learning is briefly explained.

Therefore, to overcome the aforementioned challenge and reduce the dependency on a large dataset, transfer learning is employed along with traditional ML and conventional DL approaches. We show that we can implement the concept of transfer learning combined with the traditional ML approach for the facies classification and for predicting MGS from core photos. We also show that the explored approaches exhibit reasonable classification and prediction results despite having a small dataset.

1.4 Contributions

The main contribution of the thesis lies in exploring CNNs and transfer learning techniques in oil sands drill core image analysis. The overall contribution of this thesis can be outlined as follows:

1. We explore the application of transfer learning on the pre-trained VGG-16 CNN model, fine-tuning a set of layers of the VGG-16 model, and the combination of VGG-16 and traditional ML models (random forest, decision tree) for the classification of oil sands drill core images labeled with facies. Among these three explored approaches, we achieve the highest accuracy of 98.87% for drill core facies classification using the combination of VGG-16 and random forest classifier. In this approach, we train a random forest classifier using the image features extracted by the last convolutional layer block of the VGG-16 pre-trained CNN model. The trained classification model is tested with core photos that it has never seen before. Therefore, the model can perform the facies classification with 98.87% accuracy on newly taken core photos.Therefore, with this accuracy, the geologists can also use the model for the oil sands drill core facies classification tasks. Although the best accuracy is achieved by combining VGG-16 and random forest classifier, experimental results obtained by all the explored approaches demonstrate good classification performances. Overall, this investigation provides an insight into the further implementation of these approaches for more challenging problems such as estimating MGS from oil sands drill photos. We describe the implementation of these explored approaches for facies classification and present the experimental results in Chapter 4.

2. After getting insights from the core image classification task, the aforementioned approaches are explored to see how they perform to predict the MGS from the core photos. A traditional ML regression model (random forest regressor) is also applied with these three approaches. To train the random forest model, the original pixel

values of the images, Gabor filter responses, and Sobel filters are used to extract the features from the core photos. However, similar to the core image classification, the combination of VGG-16 and random forest regression model for estimating MGS from core photos demonstrate the best prediction result. Experimental results show that, this explored method estimates the MGS from oil sands drill core photos with a mean absolute error (MAE) of 11.59, which is acceptable based on the overall distribution (mean = 153.99, standard deviation = 56.05) of the available dataset. We describe the explored approaches along with their prediction performance in Chapter 5. The explored approach might be useful for the geologists since it would provide them with a new technique to estimate MGS from core photos. Moreover, determining MGS from core photos can also be helpful to get important information about permeability and PSD.

Overall, to the best of our knowledge, this thesis explores different approaches based on CNN, transfer learning, and ML for the first time to analyze the oil sands drill core images. Therefore, despite the primary objectives, the overall outcome of this thesis will benefit future researchers by providing a benchmark and opening the scopes for conducting more extensive research in this domain.

1.5 Thesis Outline

The rest of the thesis is organized in the following manner. In Chapter 2, the applications of ML and DL techniques in the Geoscience domain are briefly discussed followed by different applications in content-based image retrieval (CBIR) and rock image classifications problems. Moreover, we discuss how a CNN model pre-trained on a huge library of natural images can be utilized to a problem that is associated with a completely different domain such as medical imaging and CBIR. In Chapter 3, we briefly describe the oil and gas exploration and production life cycle, followed by an extensive review of ML, DL, CNN, and transfer

learning. In Chapter 4, we present the three explored approaches based on transfer learning and traditional ML to perform the classification of drill core images based on facies types. Chapter 5 extends the experiments conducted in Chapter 4 to investigate how these explored approaches can be accumulated to predict the MGS from the core photos. Finally, Chapter 6 concludes the thesis by presenting the summary of the contribution and discussing the limitations, followed by describing possible future research directions.

Chapter 2

Literature Review

Applications of CNN and ML have demonstrated notable success in a wide variety of research domains, including computer vision [101], text recognition [102], speech recognition [103], object detection [104], medical image analysis [107], biometric technology [105], online social media [106], among others. Although traditional ML and DL approaches are gaining popularity in Geoscience research and reservoir engineering, the application of CNN and transfer learning is yet to be explored in these areas. At the beginning of this chapter, the application of ML and DL techniques in additional research related to the Geoscience and petroleum industries is presented. Moreover, we review some previous works on the application of transfer learning in other application domains such as medical imaging and content-based image retrieval (CBIR). We review these works to demonstrate how the concept of transfer learning effectively works for different applications, although the models are pre-trained on the data that are different from the target application. The limitations of the prior works are also discussed while reviewing them.

2.1 Machine Learning and Deep Learning in Geoscience

De Lima et al. explored the application of transfer learning to facilitate the analysis of uninterpreted images of fossils, slabbed cores, or petrographic thin sections [91]. The pre-

trained MobileNetV2 [94] and InceptionV3 [95] were successfully employed to perform the classification of microfossils, petrographic photomicrographs, and rock and mineral hand sample images. For the microfossil image classification tasks, only the pre-trained models' classification layers were trained on a dataset of 1850 qualified images belonging to seven different fusulinids (index fossils for the Late Paleozoic). After training the models, accuracy for the test data (10% of the dataset) of 100% was obtained for both models. The experimental results strongly support that, even though the dataset's volume is significantly small, state-of-the-art performance can be achieved by employing transfer learning. The researchers utilized these pre-trained CNN models to classify 1521 images of six different rock types in the same work. For this experiment as well, both the CNN models exhibited significantly good classification performances. The accuracy scores for the pretrained MobileNetV2 and InceptionV3 were 98% and 97%, respectively. For core image classification, several hundred feet of labeled cores from Mississippian limestone in Oklahoma (data from [97] and [98]) is used, and a small sample of only 285 images was selected for the classification. For this experiment, an accuracy of 100% was achieved using MobileNetV2 and an accuracy of 97% using the pretrained InceptionV3.

Zhou et al. introduced a CNN regression model for predicting permeability using the data collected from the Jacksonburg-Stringtown oil field, West Virginia, a potential carbon storage site and enhanced oil recovery (EOR) operations field [99]. In this work, geological feature images were produced by converting five parameters from the geophysical well logs. These five parameters were gamma rays, bulk density, the slope of the gamma rays and bulk density, and shale content. Each of the feature images was labeled with permeability. Therefore, the designed CNN model considered the feature images as inputs and the permeability as the target output. To improve the prediction performance and to reduce overfitting, the initial weights and biases of the proposed CNN model were searched using a genetic algorithm [35] and particle swarm optimization [36]. To evaluate the performance of the proposed model, several metrics were considered, such as root mean error, average absolute

error, and maximum absolute error. Root mean error determines the overall performance of the model, whereas the average and maximum absolute errors are used to determine the error range of the predicted result. After evaluating the overall performance of the model, the root mean error was valued at 114.78 for the training data and 125.31 for the testing data. While the maximum absolute error was 69.73 for both the training and testing process, the average absolute error for the training process was 5.21 and 12.17 for testing data. Since the errors between true and predicted permeability follow a normal distribution, the mean and standard deviation are taken from a Gaussian model. In this work, to compare the results more concisely, the permeability is plotted on a logarithmic scale. A comparison between the core permeability and the permeability predicted by the CNN regression model shows a mean of 6.6101 with a standard deviation of 24.3534. Overall the experimental results demonstrated that the proposed CNN model performed very well in constructing the relationship between the well log data and the permeability.

Panda et al. proposed a model by modifying the CK model to estimate permeability using both the bulk physical properties (porosity and tortuosity) and the statistics of the PSD of an unconsolidated permeable medium [10]. In this paper, the authors mentioned this aspect; although many models were proposed to estimate permeability, limited attention was given to the impact of the PSD on the permeability estimate. As the CK model was not applicable for a media consisting of mixed particle sizes, many modifications over the CK model were proposed to improve the estimation of permeability. However, the key limitation of these modified models was that these models required independent determination of various parameters. Therefore, estimating these parameters in actual permeable media became cumbersome and frequently empirical. The model proposed in [10] used detailed information of the permeable medium, such as PSD statistics, where the variation in PSD statistics predicted the variation in permeability estimate. The statistical parameters could be estimated from core samples, outcrops, drill cuttings, or thin sections of cores. By validating the proposed model against experimental data, the authors used it to investigate the nature

of permeability-porosity relationships. They also determined the parameters that influenced the permeability estimate the most. However, the proposed method failed to work when the permeability is below 1 μm^2. In such cases, the model over-predicted permeability.

Mauricio et al. developed a state-of-the-art DL algorithm to directly predict permeability from 2D images by leveraging the computational power of graphics processing units (GPUs) [45]. In this work, the authors built two different neural network architectures to estimate the permeability from 2D images where the first network is composed of seven convolutional layers followed by two dense layers, and the second network is inspired by the U-Net architecture [96]. According to the authors, to the best of their knowledge, this is the first work where laboratory-measured permeability data for 135 different rock samples from 11 reservoirs were used as labels. Also, this is the first time the input to the DL systems is a high-resolution image rather than a segmented image that allows the DL system to deduce the relevant features by itself. The preliminary results showed that the proposed approach was a viable means for efficiently determining permeability. After training the DL models, they demonstrated superior performances by accurately predicting the desired properties within a fraction of seconds, speeding up the simulation workflows. The average error of the predictions was only 11.69% for permeability, where the predictions were produced instantaneously. Therefore, all the related field-development workflows that depend on permeability estimation were expedited. In another research, Eric et al. demonstrated and tested a DL workflow for the automated extraction of bioturbation data from a core photo dataset [46]. The authors proposed a method for extracting image tiles from core photos along a grid and referencing each tile with collected sedimentary data that allowed users to quickly generate thousands of labeled training images for a machine learning (ML) model. The trained deep learning model could automatically predict whether a core photo contained bioturbation or not with up to 88% accuracy.

Lepisto et al. introduced the classification problem of the rock textures from rock sample images where two types of features were considered in the process of rock texture classifi-

cation [12]. These were the textural features and the spectral features. For spectral feature extraction, the HSI model was considered where hue (H) describes a pure color, saturation (S) gives the measure of the degree to which a pure color is diluted by white light, and intensity (I) is decoupled from the image's color information. In this work, the authors used only hue and intensity information for spectral features extraction. On the other hand, texture features were calculated from the gray-level co-occurrence matrix as several texture features such as contrast and entropy can be calculated from this matrix. For testing purposes, 118 rock images were acquired, where the size of the rock sample was 300 x 300 mm and the image size was 1430 x 1430 pixels. The rock samples represented two typical rock texture types. In type I, there were 54 samples of relatively homogeneous rock texture with small color variations. These samples were divided into four classes where all the samples had similar textures but different colors. In type II, 64 samples of non-homogeneous rock textures had strong differences in texture and color. The difference could also be significantly noticed within the samples of this type. These samples were divided into three classes employing these features. The classification of the texture samples was based on the k-nearest neighbor method and the value of k was experimentally selected to be 3. Results showed that, in the case of homogeneous textures classification, spectral features (mean hue and mean intensity) gave the best results whereas, in non-homogeneous textures classification, the texture features obtained the best results (contrast and entropy). In this work, a new approach to the classification and analysis of non-homogeneous rock textures was presented. The non-homogeneous textures were divided into blocks so that different areas could be considered separately. Therefore, the co-occurrence matrix gave better classification results.

For classification and retrieval of natural rock images, the use of texture granularity was considered in a study presented by Lepisto et al. [13]. The main purpose of this work was to find images with grains (phenocrysts) of a particular size and color from a rock image database. Color analysis tools were used combined with morphological operators for the recognition of the texture grains. The experimental results showed that the images with grain

could be distinguished from the other rock images using the proposed model. In another paper, Kong combined both the color and texture features for image retrieval [14]. The color features were extracted using HSV (Hue, Saturation, Value) color space, and the work of texture feature extraction was obtained by using gray-level co-occurrence matrix (GLCM) or color co-occurrence matrix (CCM). A satisfactory result was achieved by conducting experiments on the actual images and verifying the integrated feature's superiority over the single feature.

Overall, we reviewed some of the prior works on the applications of transfer learning, CNN, and traditional ML in the Geoscience research domains. We also reviewed the application of transfer learning for core image classification and the application of CNN for estimating permeability from log data. However, these approaches were implemented to build classification, or regression-based models for the data originated from the conventional cores, where the feature extraction workflow from the data is not critical. Unlike the conventional cores, for the oil sands drill core of Athabasca oil sands, the core images are very critical to handle due to the unique geological characteristics. Therefore, although the applications of transfer learning, CNNs, and traditional ML are not new for the conventional cores, there are still rooms for extensive research to explore the opportunity to employ these concepts in the challenging oil sands drill core image analysis domain.

2.2 Transfer Learning in Other Application Domains

While traditional DL techniques require a huge library of training data to train the models, extremely powerful computational resources, and optimal parameter and hyperparameter tuning to avoid overfitting/underfitting, instead of training a DL from scratch, Lisa et al. introduced transfer learning as the improvement of learning in a new task by the transferring knowledge from a related task that has already been learned [34]. Transfer learning allows the reuse of existing DL models to novel classification and regression problems with limited

data. While traditional ML and DL algorithms address the isolated task, transfer learning attempts to change this by developing methods to transfer knowledge learned in one or more source tasks and use it to improve learning in a related target task. It has been shown that CNNs are excellent feature learners [38] and can generalize image features given a large training set. Qayyum et al. showed that transfer learning could even achieve comparable or better results than training a CNN model with randomly initialized parameters [107]. In this section, we review some prior works that show that CNN models pre-trained on over 14 million natural images belonging to 1000 classes have been successfully employed in completely different domains - medical image classification, and CBIR tasks to name a few.

Khan et al. investigated the use of layer-wise transfer learning technique on a state-of-the-art CNN architecture named VGG-19 [16] that has been pre-trained on natural images, to classify medical images for improved diagnosis of Alzheimer's Disease (AD) [39]. The network was fine-tuned with layer-wise transfer learning where only a pre-defined group of layers were trained on MRI images. To reduce the dependency on large training data, the authors employed an intelligent filtering approach that chose the most useful images by calculating the image entropy. They also provided class activation maps to demonstrate how the proposed model focused on discriminative image regions that were neuropathologically relevant and could help the healthcare practitioner in interpreting the model's decision-making process. Khan et al. observed that, instead of training all the layers of the pre-trained model, the best possible result was achieved by re-training only a few top layers. However, the proposed approach did not consider noisy training images. In real-life cases, data could be messy that might result in difficulties in determining important feature information from the data.

Alex et al. trained a large deep CNN named as AlexNet that could classify 1.2 million high-resolution images of 1000 different categories [15]. AlexNet has 60 million parameters and 650,000 neurons. It consists of five convolutional layers, where some are followed by max-pooling layers and three fully connected (FC) layers with a final 1000-way softmax.

Overfitting in the FC layers is reduced by employing the "dropout" regularization method. Experiments demonstrated promising performance on a highly challenging dataset using purely supervised learning. The paper reporting AlexNet is considered one of the most influential ones published in computer vision; it has spurred many more papers published employing CNNs and GPUs to accelerate deep learning [40]. In another study, Simonyan et al. investigated the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting [16].

Wan et al. explored a DL framework, CNN to learn feature representations from images and their similarity measures towards CBIR tasks [18]. CBIR is a well-studied computer vision technique to search for relevant images from large databases. In CIBR, the search is based on different image features like color, texture, shape, or any other features derived from the image itself. In another work, Xie et al. showed that image classification and retrieval are not different since both tasks could be tackled by measuring the similarity between images [20]. In this work, a new algorithm named Online Nearest-neighbor Estimation (ONE) was proposed to utilize Principal Component Analysis (PCA) and Product Quantization (PQ) approximation, GPU parallelization to scale the proposed algorithm up to large-scale image search.

2.3 Chapter Summary

The analysis of previous research on ML and DL techniques for enhanced image analysis and image classification problems in diverse research domains is the focal point of this chapter. In this chapter, the research gaps are identified. In summary, the earlier works primarily focused on rock image classification based on texture, color, and spectral features. Also, many different image classification and retrieval techniques were proposed using traditional image-based ML and DL approaches in various research domains such as improving disease diagnosis by medical image analysis, analyzing different critical geological properties, etc. A few prior

pieces of research on the application of CNN and transfer learning in the geoscience and petroleum industries are also discussed in this chapter. We briefly discussed the application of transfer learning on the pre-trained CNN model and the implementation of the CNN model for core image classification and permeability prediction. However, as mentioned earlier, these applications were primarily focused on the dataset where the feature extraction workflow is not critical compared to the oil sands drill core images. Since the CNN and transfer learning techniques can automatically extract useful features from images, we aim to investigate if we can leverage this for our oil sands drill core image data. To the best of our knowledge, the application of transfer learning, CNN, and traditional ML techniques have not been explored yet to estimate MGS from oil sands drill core photos where the textures and grains are rarely visible because of the black bitumen. Therefore, to fill the research gaps, in the subsequent chapters, we explore these approaches for the core image classification and to estimate MGS from core photos, compare the performances of the explored approaches and propose the best model, identify potential challenges to manipulate the dataset, and propose possible future research directions.

Chapter 3

Preliminaries

This chapter presents the oil and gas operating cycles and different activities associated with each phase. Then we provide a brief overview of different ML and DL techniques. We also present the conventional architectures of traditional DL and CNN models and describe different components and terminologies of each layer of a CNN model. Moreover, in this chapter, we describe different activation functions used to train a traditional DL model. Finally, we end this chapter by presenting a brief overview of the transfer learning concept.

3.1 Oil and Gas Operating Cycle and Activities

In oil and gas industries, exploration and production companies find hydrocarbon reservoirs, drill oil and gas wells, extract these raw materials, and sell them to be refined by other companies into products such as gasoline [6]. Typically, the life cycle of extracting and processing hydrocarbons from a typical oil and gas reservoir include four stages: Exploration, Development, Production, and Abandonment, as shown in Figure 3.1.

Exploration is a method used by petroleum geologists and geophysicists for searching for oil and gas resources in subsurface earth by reviewing existing geological and geophysical data to learn about the potential reservoir. The exploration process consists of locating oil and gas reserves using seismic surveys and drilling wells. Seismic survey or seismology

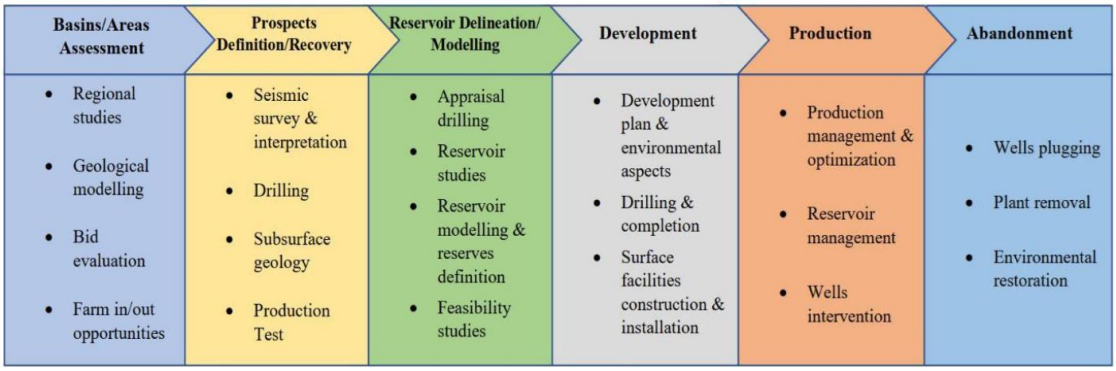


Figure 3.1: The oil and gas operating cycle and activities (Treccani-Petroleum Encyclopaedia) (Adapted from [8]).

are performed by creating and recording substantial vibration on the earth's surface via explosives or machinery [82]. As shown in Figure 3.2, As shown in Figure 3.2, the seismic waves travel down the surface, and the reflected waves are analyzed at the surface to identify layers of rock that trap oil and natural gas reservoirs. The result of a seismic survey is essentially a picture of the various rock layers used to identify geological structures that may contain oil and natural gas resources [7]. Exploration is an expensive, risky, and time-consuming process. The expenditures associated with this process are usually valued at millions of dollars and the oil and gas discovery process may take substantial time even though sometimes the explorers may find nothing at all after the exploration process.

If the seismic data analysis shows a geological structure that could contain oil and gas resources, an exploration well is drilled to confirm the presence of oil and gas resources. A well-logging tool is lowered into the well to acquire geological data and understand the presence of reservoir, reservoir fluid characteristics, among other properties. Core analysis is performed at this stage from the core samples to obtain detailed petrophysical data. After the successful exploration drilling, appraisal drilling is performed to reduce the uncertainty or possibility of losses about the size of the oil and gas field and its properties. In this phase, more wells, in addition to the exploration wells are drilled to collect more information and samples from the reservoir. Another seismic survey with higher resolution is repeated to get a better image of the reservoir. Data from the seismic survey and wells assist the geologists

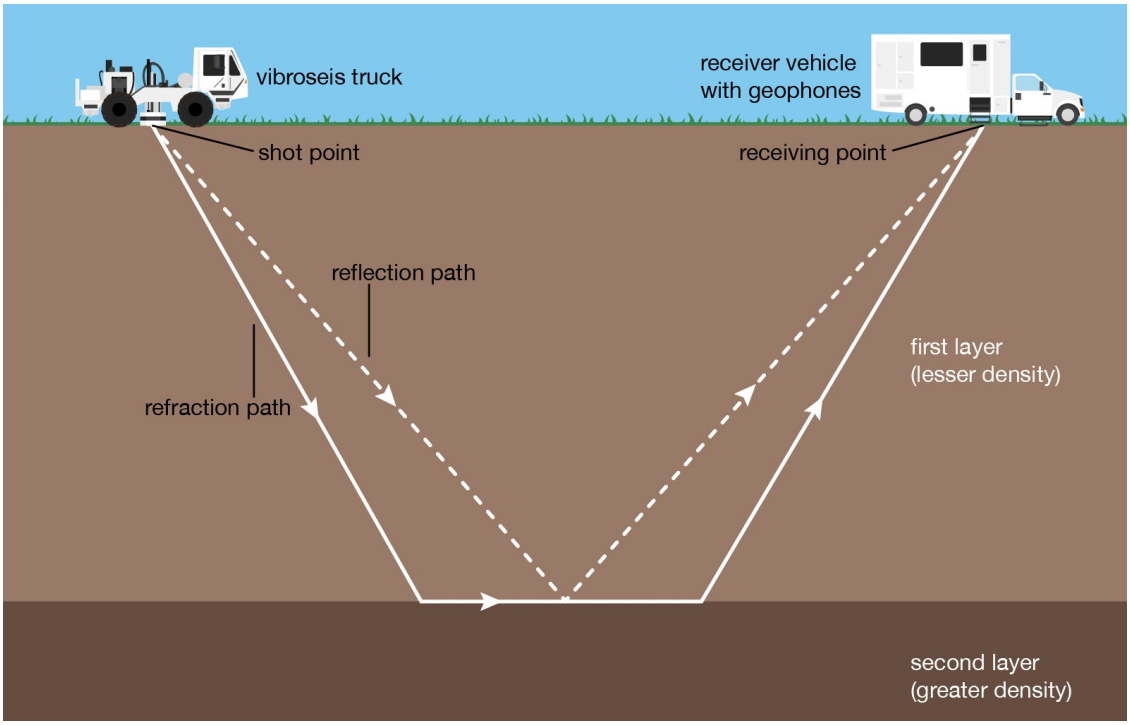


Figure 3.2: Seismic survey
(Britannica, The Editors of Encyclopaedia. "Seismic survey". [accessed 2021 July 26]. https://www.britannica.com/science/seismic-survey.).

and reservoir engineers in understanding the reservoir better and comprehending specific characteristics about the reservoirs, such as how much oil and gas might be in the reservoir and how fast oil or gas would move through the reservoir.

After the successful appraisal phase and identifying potentially viable fields, the development stage is initiated before the full-scale production. The engineers determine the number of wells needed to meet production requirements and the method of extraction of the liquid hydrocarbons. At the beginning of the development stage, the geologists, geophysicists, and reservoir engineers create a development plan and identify the number of wells that need to be drilled to produce the oil or gas. The drilling engineers decide on the best design for the production wells and the project engineers assist in building the planned facilities such as, deciding what production facilities are required to process the oil or gas before it is sent to the refinery or customer. Finally, the logistic engineers decide the best route to export the oil and gas. The development phase also may cost hundreds of billion dollars and it typically

may last for 5 to 10 years to develop an oil or a gas field. This fact may vary based on the location, size of the facilities, and the number of required wells.

In the production stage, liquid hydrocarbons extracted from the well are separated from the non-saleable components such as water and solid residuals. Natural gas is often processed on-site while oil is piped to a refinery before being offered for sale. Finally, once all of the accessible oil and natural gas reserves in a field have been produced, the project can be decommissioned, meaning that wells are plugged by setting mechanical or cement plugs in the wellbore (i.e., drill hole including the open hole or uncased portion of the well) at specific intervals to prevent fluid flow. The plugged wells are then abandoned, the infrastructure removed, and the site cleared to remove any debris.

3.2 Machine Learning Overview

ML is a sub-field of artificial intelligence (AI) that imitates human intelligence by learning from the surrounding environment. In this era of big data, ML provides the tools and technology that can be utilized to extract meaningful insights from the data. Although ML is a field within the Computer Science domain, it differs from traditional computational approaches. Unlike traditional computational algorithms, instead of being explicitly programmed, ML algorithms are trained on data inputs and use statistical analysis to output values that fall within a specific range. Because of this, ML facilitates computers in building models from sample data to automate decision-making processes based on data inputs. There is much data today generated not only by people but also by computers, smartphones, and other sensors or devices. This reality will only continue to grow in the years to come. Traditionally, humans have analyzed data and adapted systems to the changes in data patterns. However, as the volume of data grows, it surpasses the ability for humans to make sense of it and manually write those rules. Therefore, we will turn increasingly to automated systems that can learn from the data and, importantly, the changes in data to adapt to a shifting

landscape. Techniques based on ML have been applied successfully in diverse fields ranging from pattern recognition, computer vision, computational biology to medical applications. In ML, we use data to answer questions. Here "using data" refers to "training" the system and "answering questions" means "making predictions". Therefore, to connect these two concepts, we use data to train an ML model and make it increasingly better and more powerful in predictions. Later on, this predictive model can be deployed to predict data that has never been seen before.

Arthur Samuel, a pioneer in the field of computer gaming and AI, coined the term "Machine Learning" and defined it as a "field of study that gives computers the ability to learn without being explicitly programmed" [56]. A more modern definition of ML, given by Tom Mitchell is "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" [57]. For instance, we assume that we want to create an ML model that can distinguish between the images of cats and dogs. The examples that the model uses to learn are collected in a "training set". In this example, the training set is a library of images of cats and dogs. In a training set, each training example is called a training instance or sample. Comparing this example with Tom Mitchell's definition, the task T is to perform classification between cat and dog for an unknown image, the experience E is the training data, and the performance measure P needs to be defined; for example, the ratio of correctly classified images.

3.2.1 Types of Machine Learning

Based on whether or not the ML systems are trained with human supervision, they are of three types: supervised learning, unsupervised learning, and reinforcement learning [37]. In supervised learning, the training data contains the desired solutions called "labels" [59]. Therefore, each sample in the training set has a label. A typical supervised learning task is *classification.* The above-mentioned example that can identify cats and dogs from images is

an example of supervised learning. Here, the model is trained with many example images along with their *class* ("Cat" or "Dog"), and it must learn how to classify a new image. Another typical task is to predict a target numeric value, such as the price of a house, given a set of features (number of rooms, size in square feet, geographic location, year built, etc.). This type of task is called *regression*. Training this system would require many examples of houses, including both the features and their labels i.e., their prices. Some of the most widely used supervised learning algorithms are k-Nearest Neighbors (KNN) [60], Support Vector Machine (SVM) [65], Linear Regression [61], Logistic Regression [62], Decision Trees and Random Forests [64], and Neural networks [63].

Unlike supervised learning, in unsupervised learning, training data is unlabeled [59]. Here, the system needs to find patterns in the data from the training examples. Detecting unusual credit card transactions for fraud prevention is an example of unsupervised learning. Here the system is trained with typical credit card transactions. When the system sees a new transaction, it tries to identify whether or not this instance is an anomaly. Some unsupervised learning algorithms are k-Means clustering [66], and Principal Component Analysis (PCA) [67].

Reinforcement learning (RL) [41] is a type of ML technique where an "agent" explores an "environment". Based on the "environment", the "agent" takes a certain action, and based on that action, it receives "reward" or "penalties". It must then learn by itself what is the best strategy, called a "policy", to get the most reward over time. For example, RL is implemented to train an obstacle detection and avoidance robot [42]. Given the environment with and without different types of obstacles, if the robot hits the obstacle it gives itself negative rewards. Over time the robot learns from its experience and will be able to identify which actions lead to the best rewards. In a paper, Balaji et al. demonstrated how the AWS DeepRacer learns to drive by itself using RL [58]. It is to note that, neural network can be unsupervised learning and reinforcement learning as well (see [127] and [128]).

3.3 Introduction to Deep Learning

DL is a sub-filed of ML concerned with algorithms inspired by the structure and function of the brain called artificial neural network (ANN). Like how human beings learn from experience, a DL algorithm repeatedly performs a task, each time tweaking it a little to improve the outcome. DL is the key technology behind driverless cars, enabling them to recognize a stop sign or distinguish a pedestrian from a lamppost. In DL, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance [43]. A variety of complex tasks can be achieved using these models, such as classification and regression using ANN and different computer vision tasks by CNN.

DL architecture is the mechanism of representation of a pattern in a hierarchical manner [68]. The hierarchy starts from the input layer and ends with the decision layer. In a neural network, there can be one or more hidden layers. A neural network with only one hidden layer is called a shallow network. A deeper neural network contains more than one hidden layer. Biological neurons or simply neurons are the fundamental units of the brain and nervous system, the cells responsible for receiving sensory input from the external world via dendrites, process it, and give the output through axons as shown in Figure 3.3. Figure 3.4 represents the general model of ANN which is inspired by a biological neuron, also called

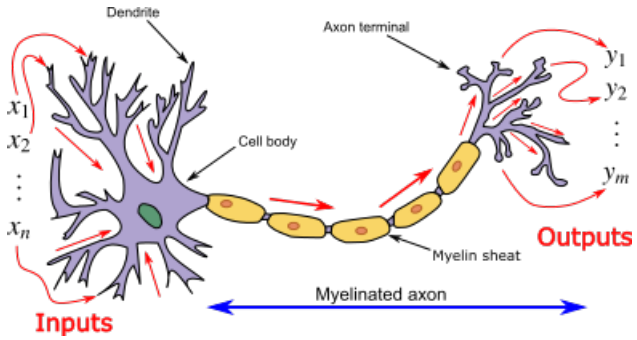


Figure 3.3: Biological neuron model
(Wikipedia contributors. (2021, May 26). Biological neuron model. [accessed 2021 July 27]. https://en.wikipedia.org/w/index.php?title=Biological_neuron_model&oldid=1025272997).

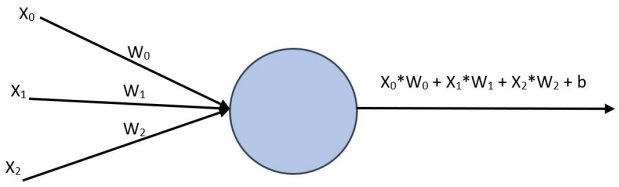


Figure 3.4: Perceptron model.

perceptron. If the input to a perceptron is x, the result of the perceptron z is defined by:

$$z = w * x + b \tag{3.1}$$

Here, w is the weight of the perceptron and b is the bias [69]. Similarly, if a neuron receives multiple input values represented as $x_0, x_1, x_2, \ldots, x_n$, each of these inputs is multiplied by a connection weight represented as $w_0, w_1, w_2, \ldots, w_n$ followed by summing the products as follows:

$$x_0 * w_0 + x_1 * w_1 + x_2 * w_2 \ldots x_n * w_n = \sum_{i=0}^{n} x_i * w_i \tag{3.2}$$

Equation 3.1 performs linear transformation of the inputs. In order to introduce non-linearity, a non-linear activation function as follows is applied to the result of the perceptron obtained by Equation 3.1 [69].

$$F = g(z) \tag{3.3}$$

Although activation functions can be both linear and non-linear, in DL architecture linear activation functions are rarely applied because the combination of linear activation is also a linear function and a linear activation can not model non-linear data. On the other hand, non-linear activation functions with the linear transformation in the perceptron of the hidden layer can approximate a function to describe the data [70].

In Equation 3.3, g can be sigmoid, hyperbolic tangent (tanh), Rectified Linear Unit (ReLU), and other non-linear activation functions. Sigmoid, tanh and ReLU activation functions are defined in Equations 3.4, 3.5, and 3.6 respectively [69], [70]. Moreover, Figure

3.5 depicts the behavior of sigmoid, tanh and ReLU activation functions.

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{3.4}$$

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.5}$$

$$\text{ReLU}(x) = \max(0, x) \tag{3.6}$$

The sigmoid activation function plays an important role in the context of logistic regression. Logistic regression is a technique to predict an outcome of binary classification problems. The sigmoid activation function takes the weighted sum of the input features as input and outputs the probability value of the outcome. According to Equation 3.4, for any value of x, the sigmoid(x) function will output a value within the range from 0 to 1. On the other hand, the tanh(x) activation function output is from -1 to 1. Therefore, the mean of the activated results of tanh(x) function would always be more centered to zero when compared to sigmoid(x). Moreover, the benefit of the hyperbolic tangent activation is that the gradient of the hyperbolic tangent is larger than the gradient of the sigmoid activation function. Thus, the hyperbolic tangent is almost always preferable to the sigmoid activation function [72]. ReLU(x) activation function outputs the activated value within the range from 0 to ∞. Negative values are clipped to zero in the ReLU(x) activation function.

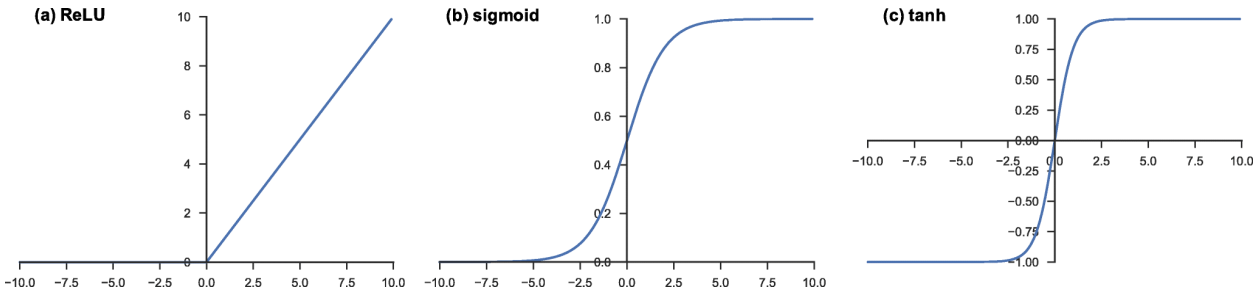


Figure 3.5: Activation functions commonly applied to neural networks: a) rectified linear unit (ReLU), b) sigmoid, and c) hyperbolic tangent (tanh)
(Yamashita, R., Nishio, M., Do, R. K. G., Togashi, K. Convolutional neural networks: an overview and application in radiology. *Insights Imaging* 9 (4), 611–629 (2018).).

ReLU(x) activation function is faster to compute than sigmoid(x) and tanh(x) because of its low computational complexity.

In a shallow network, many perceptrons are present in a hidden layer where multiple hidden layers are stacked one after another to form a deeper network. Nodes of the input layer are connected to all the nodes in the hidden layer in a deep neural network. If all the nodes in a hidden layer are connected to all the nodes in the next hidden layer or output layer, this hidden layer is called a fully connected (FC) layer.

Several methods can be applied to initialize the weights of the nodes in the hidden layers, including: normal distribution, Xavier weight initialization method [108], and He weight initialization method [109] to name a few. The initialized weights are updated after each epoch using the backpropagation technique [73]. An epoch indicates the number of passes of the entire training dataset the ML or DL algorithm has completed. Considering a supervised classification task, each of the sample inputs is assigned a class label also called ground truth. Thus, the result of the decision layer after the activation is compared with the ground truth.

Initially, as the model weights are initialized, in the beginning, the model can not distinguish between right and wrong predictions. This stage is where the learning comes in. The idea is that the model needs to 'understand' when the computational prediction is wrong, which is calculated by some form of 'loss'. This loss depends on the problem, but it typically involves minimizing the difference between the predicted output and the ground truth value. In a classification task, the cross-entropy loss function is widely used. It is also called logarithmic loss or log loss. In this loss function, each predicted class probability is compared to the actual class desired output (0 or 1), and a loss/score is calculated that penalizes the probability based on how far it is from the ground truth value. Cross-entropy is used to adjust the model weights while training, and it is defined by:

$$L_{CE} = -\sum_{i=1}^{\mathrm{n}} t_i \cdot \log(p_i) \tag{3.7}$$

where n is the number of classes, t_i is the ground truth label, and p_i is the predicted probability for the *i-th* class.

For binary classification task, we use binary cross-entropy defined as:

$$L_{BCE} = -\sum_{i=1}^{2} t_i \cdot \log(p_i) = -[t \cdot \log(p) + (1-t) \cdot \log(1-p)] \tag{3.8}$$

where t_i is the ground truth label taking a value 0 or 1, and p_i is the predicted probability for the *i-th* class. The overall loss of a model is calculated as the average of differences between all predictions and ground truth observations. This calculation is called the cost function. Loss function mainly applies for a single training set compared to the cost function, which deals with a penalty for several training sets or the complete batch. Therefore, the loss is calculated numerous times for a single training cycle, but the cost function is only calculated once. Based on the loss function shown in Equation 3.8, binary cross-entropy is often calculated as the average cross-entropy across all the data examples defined by

$$J(W) = -\frac{1}{N}\sum_{i=1}^{N}[(t_i \cdot \log(p_i) + (1-t_i) \cdot \log(1-p_i))] \tag{3.9}$$

for N data points, where t_i is the ground truth label taking a value 0 or 1, and p_i is the predicted probability for the *i-th* class, and *J(W)* is the function that calculates the overall loss for the whole training example. In *J(W)*, W is the set of model weights.

Unlike a classification problem, if the model needs to predict a continuous value as output, the appropriate loss would be the mean squared error that tries to minimize the squared difference between the actual value and a predicted value. We define this loss function in Chapter 5. To classify drill core photos based on facies, we use binary cross-entropy as a loss function. On the other hand, mean squared error is used for the prediction of MGS from core photos.

After defining the loss function, loss optimization and model training need to be employed. The loss optimization attempts to find a set of weights W, that minimizes the

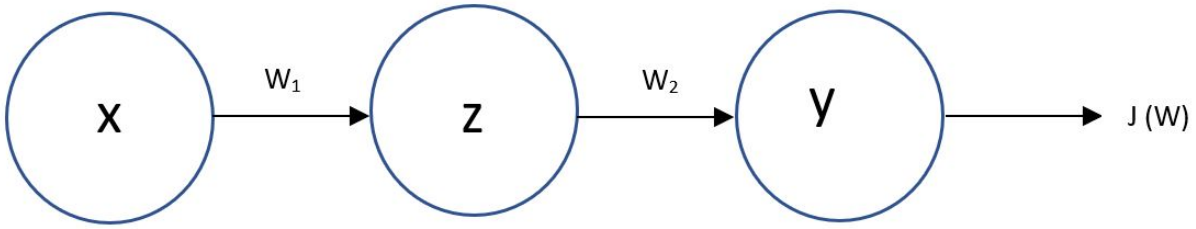


Figure 3.6: Neural network with three neurons and associated weights.

calculated loss. If there is only one weight component, it is possible to plot the weight and the loss on a 2-dimensional graph and then choose the weight that minimizes the loss. However, for most deep neural networks with multiple weight components, visualizing an n-dimensional graph is critical. Instead, the gradient is calculated based on the loss function as follows:

$$\text{Gradient} = \frac{\partial J(W)}{\partial W} \tag{3.10}$$

Here *J(W)* is the loss function, and W is the model weight. The gradient is a commonly used term used in optimization and ML that measures the change in all weights regarding the change in errors. A gradient can be considered as the slope of a function. The higher the gradient, the steeper the slope, and the faster a model can learn. However, if the slope is zero, the model stops learning. In mathematical terms, a gradient is a partial derivative with respect to its inputs. Weights of each layer are updated based on the gradients of the loss function. The goal is to minimize the loss function so that the difference between the decision layer output and the ground truth label is as low as possible to produce optimal prediction results. The weights are updated until a certain number of epochs or the target loss is achieved or the update of the weights does not change the loss for a certain number of epochs. The process of calculating this derivative is known as backpropagation that is calculated based on the chain rule of calculus [17]. Considering a neural network shown in Figure 3.6, the derivative or the gradient explains how a small change in the first set of

weights (w_1) affects the final loss ($J(W)$).

$$\frac{\partial J(W)}{\partial w_1} = \frac{\partial J(W)}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial w_1} \tag{3.11}$$

The effect of the change in weights w_2 on the overall loss can also be determined similarly. Finally, the new weights are calculated as follows:

$$W_{new} = W_{current} - \eta \frac{\partial J(W_{current})}{\partial W_{current}} \tag{3.12}$$

Here, η denotes the learning rate of the model. Learning rate is an important factor to consider when training a deep neural network. As the model travels to find an optimal set of weights, it needs to update its weights by some factor. If the learning rate is too small, the model can either run for an exponentially long period or get trapped somewhere known as the global minimum. If the factor is too large, then the model might miss the target point and then diverge. However, an adaptive learning rate can be used to reduce the chances of such problems. Here, the factor changes based on the current gradient, the current weights' size, and other factors that can affect where the model should go next to find the optimal weights. Overall, this is a typical way to train a DL model.

3.3.1 Types of Deep Learning Architectures

There are several DL architectures focused on the paradigm of supervised ML. Deep Learning Neural Network (DLNN), Recurrent Neural Network (RNN), and CNN are commonly used for supervised ML. The conventional architecture of a DLNN is shown in Figure 3.7

FC layers with a different configuration of nodes are stacked one after another in between the input layer and the decision layer in a DLNN architecture. The input layer usually takes raw data as inputs. It also takes handcrafted features extracted from the original data as input as well. A DLNN designed for binary classification contains a decision layer with only one node. For a multi-class classification problem, the number of nodes in a decision layer

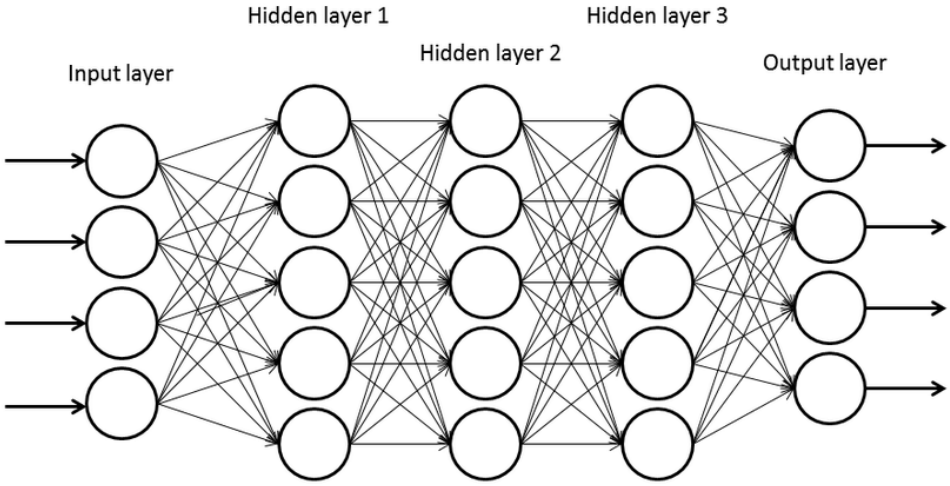


Figure 3.7: Conventional architecture of a deep learning neural network (Miralles-Pechuán, L., Rosso, D., Jiménez, F., García, J. M. (2017). A methodology based on Deep Learning for advert value calculation in CPM, CPC and CPA networks. *Soft Computing,* 21(3), 651-665.).

is equal to the number of class labels. Similar to binary classification, a DLNN contains only one node in the decision layer designed for a regression task. The sigmoid function is applied as an activation function in the decision layer for binary classification. Whereas, for multi-class classification tasks, Softmax activation is used to determine the prediction probabilities of each of the class labels [81]. Moreover, for a regression problem, the linear activation function is often used in the decision layer.

CNN is a DL architecture that can take images as inputs, assign weights or importance to various aspects in the image, and differentiate one from the other [21]. CNNs typically consist of several pairs of convolutional and pooling layers, followed by a number of FC layers, and finally a 'softmax' layer, or regression layer, to generate the output labels. The conventional architecture of a CNN including different layers is shown in Figure 3.8.

- Input Layer: An image that the CNN takes as an input is nothing but an n-dimensional array of pixel values. An n-dimensional array of values is called a tensor. A tensor can be a vector (1-dimensional array of values) or a matrix (2-dimensional array of values) that represents all data types. All values in a tensor hold identical data types

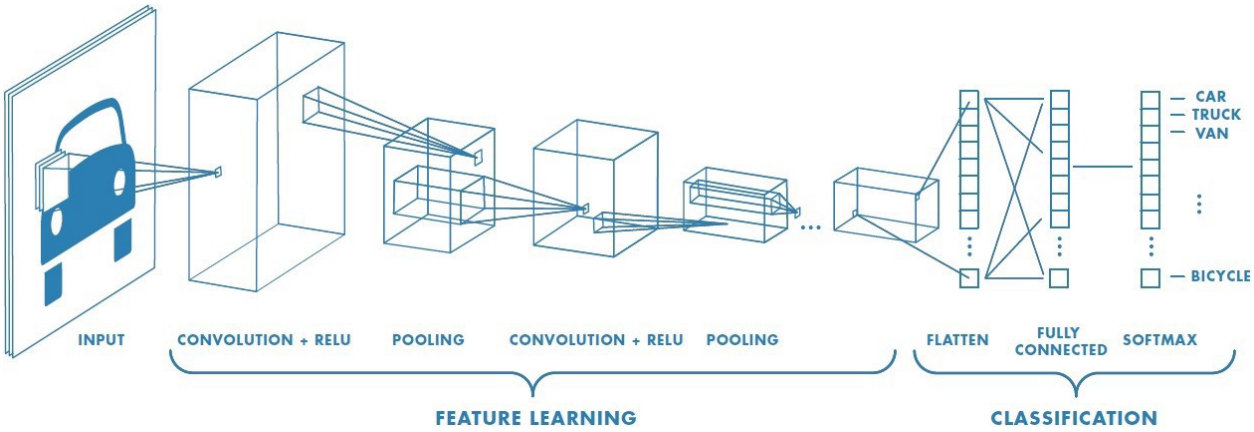


Figure 3.8: Conventional architecture of a convolutional neural network [21].

with a known (or partially known) shape. The input layer of a CNN has no learnable parameters since this layer has nothing to learn. In CNN, the input image has an arbitrary number of channels. For a gray-scale image, it has only one channel. Whereas for an RGB image, each input has three color channels indicating Red (R), Green (G), and Blue (B) and the values of each pixel can range from 0 to 255.

- Convolutional Layer: Convolutional layers also known as Conv layers consist of a number of filters which can be visualized as 2-D matrices of numbers. This layer can use an input image and a filter to produce an output image by convolving the filter with the input image. The objective of this convolution operation is to extract different types of features from the input image. CNNs need not be limited to only one convolutional layer. Conventionally, the first convolutional layer captures low-level features such as edges, color, and gradient orientation. With added layers, the architecture adapts to the high-level features, giving us a network that has a wholesome understanding of images in the dataset, similar to how we would. Features from the input tensors are extracted using the convolution operation defined by the convolutional kernel, the number of filters, and the stride size. Stride is the number of pixels shifting over the input matrix. If the stride is 1, the filter is moved 1 pixel at a time. When the stride is 2, we move the filters to 2 pixels at a time. Stride controls how the filter convolves

around the input. When the stride is 1, the filter convolves around the input volume by shifting one unit at a time. The amount by which the filter shifts is the stride. If the input tensor is a 2D matrix (I), the convolutional kernel is usually a 2D kernel (K). A 2D convolution on the index (i, j) can be defined as follows:

$$F[i,j] = (I * K)[i,j] = \sum_m \sum_n K[m,n] I[i-m, j-n] \tag{3.13}$$

Here the symbol (*) denotes the convolution operation [68]. After a valid convolution operation, the dimension of the feature map is reduced. However, if the dimension of the feature map and the input matrix are identical after the convolution, a padding operation is performed to add rows and columns evenly in the input matrix. Stride size is another hyperparameter of the convolutional layer. Usually, the convolutional kernel is shifted by one step by setting the stride size to 1. However, if the stride size is set to larger than 1, the convolution kernel is shifted accordingly to perform convolution operation and down-sampling simultaneously.

To introduce non-linearity to the feature map, the ReLU activation function is commonly used in CNN architecture due to less computational complexity and better optimization than sigmoid and tanh activation functions. Several other activation functions were also proposed to replace ReLU, such as Leaky ReLU [76], Parametric ReLU [77], Exponential Linear Unit (ELU) [78], and Scaled Exponential Linear Unit (SELU) [79]. However, due to the instability in different CNN architectures, these activation functions are not widely adopted by the research community.

As CNN learns in the convolutional layer, certainly there are weight matrices. Learnable parameters can be calculated by multiplying the shape of width m, height n and account for all filters k. Therefore, several parameters in a convolutional layer would be *((m*n)+1)*k)*; added 1 because of the bias term for each filter.

- Pooling Layer: The pooling layer is responsible for reducing the dimension of the

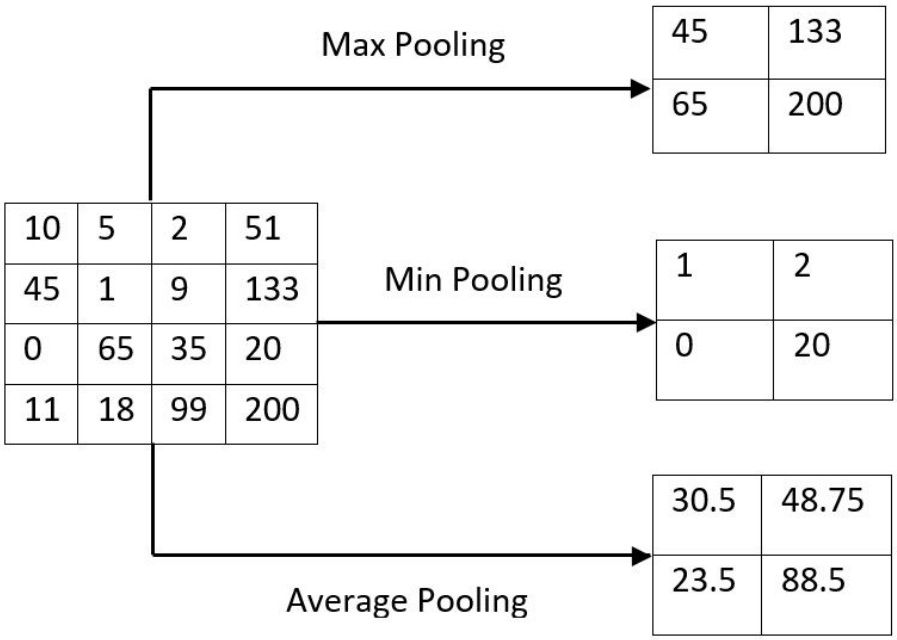


Figure 3.9: Examples of max pooling, min pooling and average pooling operation on 4 x 4 matrix using 2 x 2 kernel with a stride size of 2.

convolved feature. It is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training the model. There are three types of pooling: max pooling, min pooling, and average pooling. Max Pooling and Min pooling return the maximum and minimum values respectively from the image portion covered by the filter. On the other hand, average pooling returns the average of all the values from the image portion covered by the filter. An example of a pooling operation is illustrated in Figure 3.9. The pooling layer also does not have any learnable parameters.

- FC Layer: FC layers have every node connected to every output from the previous layer. Conv layers and pool layers break the input images into features. FC layer takes the convolution/pooling process results and uses them to classify the image into a label in a simple classification example.

- Softmax Layer: Other than these four layers, there is another layer called the softmax

layer for multi-class classification problems. To complete the CNN, it needs to be able to make predictions. The softmax layer is an FC layer that uses the softmax activation function to make the prediction tasks. The softmax activation function turns arbitrary real values into probabilities.

DLNN and CNN models are trained and optimized using the optimization method using backpropagation algorithms [74], where unknown weights for each layer are iteratively updated to minimize a specific cost function. Typically, the weights are initialized with a random set of values. However, the large number of weights typically associated with a CNN requires a large number of training samples so that the iterative backpropagation algorithm can converge properly. Having a limited number of training samples can result in the algorithm being stuck at a local minimum [75], which will result in sub-optimal classification performance. An alternative to randomized weight initialization is transfer learning or fine-tuning, where the weights of the CNN are copied from a network that has already been trained on a larger dataset.

3.3.2 Transfer Learning

In computer vision, transfer learning is a popular method since it allows to build an accurate model in a time-saving way. Instead of learning from scratch, transfer learning allows us to take a network trained on a different domain for a different source task and adapt it for our domain and our target task. Transfer learning is usually expressed through the use of pre-trained models. A comprehensive review of pre-trained models' performance on computer vision problems using data from the ImageNet (Deng et al. 2009) [15] challenge is presented by Canziani et al. [80].

Transfer learning aims to improve learning in the target task by leveraging knowledge from the source task. Researchers noted that the parameters learned by the layers in many CNN models trained on images exhibit a common behavior - layers closer to the input data tend to learn general features, such as edge detecting/enhancing filters or color blobs. Going

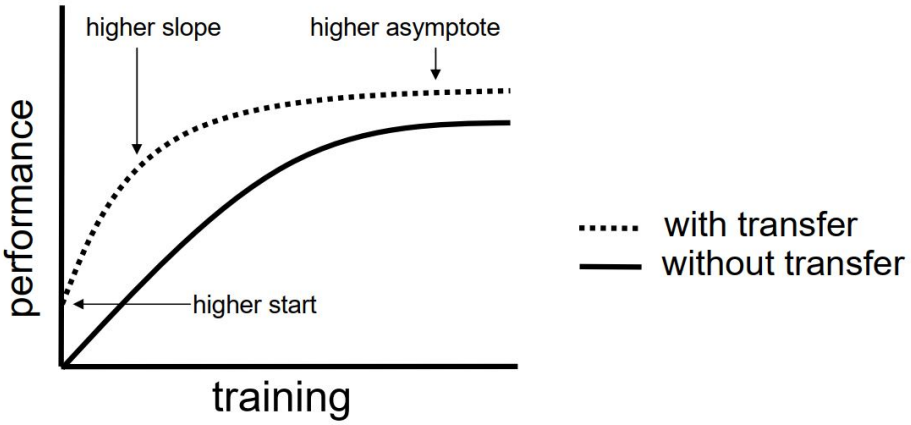


Figure 3.10: Three ways in which transfer might improve learning
(Torrey, L. and Shavlik, J., 2010. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques* (pp. 242-264). IGI global).

deeper into the network model, the extracted features from the previous layers are utilized to construct more data-specific features such as faces, feathers, or object parts [33] [113]. These general-specific CNN layer properties are important factors to be considered for the implementation of transfer learning [114].

There are three common measures by which transfer might improve learning. First is the initial performance achievable in the target task using only the transferred knowledge, before any further learning is done, compared to the initial performance of an ignorant agent. Second is the time it takes to thoroughly learn the target task given the transferred knowledge compared to learning it from scratch. The third is the final performance level achievable in the target task compared to the final level without transfer. Figure 3.10 illustrates these three measures.

If a transfer method decreases performance, then a negative transfer has occurred. One of the major challenges in developing transfer methods is to produce positive transfer between appropriately related tasks while avoiding negative transfer between less related tasks.

3.4 Chapter Summary

Presenting a brief overview of different types of ML and DL models is the primary objective of this chapter. In this chapter, we determined how the architecture and functionality of biological neurons are related to the conventional architecture and training process of a traditional DL and CNN model. Along with discussing how a neural network model learns, we described the pros and cons of different optimization techniques and determined the most superior one to use in a certain type of problem. In the following chapters, we discuss how we explore different CNN, transfer learning, and traditional ML approaches to answer questions associated with this thesis.

Chapter 4

Machine Learning and Computer Vision for Facies Classification

Core facies analysis is fundamental as it allows us to understand the depositional theory, provides the understanding of petroleum systems and allows us to understand how to optimize resource extraction. However, core facies analysis is a highly specialized and time-consuming process mainly done by geoscientists; it requires a significant amount of resources. This chapter explores different approaches based on CNNs, transfer learning, and traditional ML to automate the facies analysis process that would save geologists' time by generating consistent facies classification. Moreover, the experimental results provide insight and motivation for implementing the explored techniques for further analysis of oil sands drill core photos and predicting the MGS from the core photos, as presented in Chapter 5.

4.1 Background

Just as the type of vegetation indicates the type of climate in a region, similarly, geological facies represent rock types that indicate the depositional setting. Facies analysis indicates the depositional setting, which improves the understanding of petroleum systems leading to more optimal resource extraction decisions. In geology, facies represent the overall characteristics

of a rock unit that reflect its origin and differentiate the unit from others around it. In terms of different physical characteristics such as sedimentary structure, and grain size, facies can be of different types such as sedimentary facies, lithofacies, and seismic facies. The physical and organic characteristics found in these rock units usually provide insights into different processes and systems that may have occurred in the region. Combining several facies with physical models and other geological data can help provide informative low-dimensional models of the geologic region, leading to better insights regarding the geology of the region.

An accurate estimate of facies is essential for a better understanding of geological variation in a reservoir. The interpretation of facies may rely on the geological context described by geologically interpretive features and petrophysical rules. A facies is primarily defined based on the texture, mineralogy, grain size, visual mud index (VMI), among other attributes. Ideal sources for facies classification are core samples of rocks extracted from wells. However, obtaining the core samples from different depths at many wells is a costly and time-consuming process. However, core samples can be collected from the previously cored wells and each sample can be labeled based on a petrophysical rule. Since the conventional process of manually assigning facies by human interpreters is very tedious and time-consuming, several alternative approaches have been proposed to address facies classification. Several works can be found where different ML and DL techniques have been implemented for facies classification based on wire-line logging measurements [115–118]. Through the logging technique, a detailed description of rock formations at different depth levels is possible to obtain by measuring a wide variety of rock properties. For every well, a set of seven scalar attributes are available to consider in a logging technique [116] [117] [119] [120]. The available attributes are as follows:

- *Gamma ray* measures natural formation radioactivity;
- *Resistivity* measures the subsurface ability to impede the flow of electric current;
- *Photoelectric effect* measures electrons emission of a facies illuminated by light rays;

- Neutron-density porosity difference and average neutron-density porosity are measurements correlated to facies *density*;
- *Nonmarine/marine indicator* is a binary flag attributed by experts to distinguish between marine and nonmarine facies upon data inspection;
- *Relative position* is the integer index of each layer's depth starting from 1 for the top layer and increasing with depth.

In an ML workflow for facies classification using well log data, the seven attributes above are concatenated to produce the feature vector and passed through the ML classification algorithm. In this chapter, a few approaches based on CNN, transfer learning, and ML are explored to perform the classification of facies on the drill core image data. These approaches are explored to get an insight into whether or not they can be implemented for oil sands drill core image analysis workflow to estimate the MGS, PSD, or permeability. Experimental workflow for estimating facies from core photos is shown in Figure 4.1. In this workflow, first, we prepare the data for the classification task. The data preparation phase includes cropping the core photos from the core slabs [1] and labeling the core photos with facies for classification. In the next phase, we select different classification models followed by training the models with the data prepared in the previous phase. Finally, we evaluate the performance of the explored models based on different performance evaluation metrics and present a comparison among the explored models. In the following section, we briefly describe the phases shown in Figure 4.1.

4.2 Methodology

For the classification of facies using the core photos, three methods are explored. To investigate whether or not the concept of transfer learning can be implemented to analyze the oil

[1] Collected by Suncor Energy, the industry partner on this research.

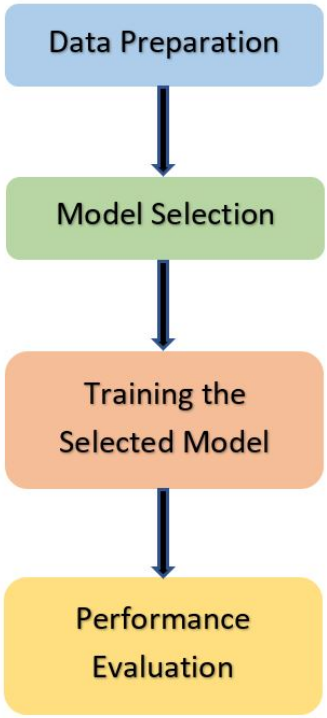


Figure 4.1: Experimental workflow of the facies classification task.

sands drill core images, in the beginning, a traditional transfer learning technique using the VGG-16 pre-trained CNN is used. In this approach, all layers associated with the feature extraction from data are kept frozen, meaning that these layers are not retrained on the target data. Only the layers associated with the prediction or the classification tasks are re-trained with the available training data. As the second approach, the pre-trained VGG-16 is fine-tuned. Here, instead of keeping all the feature extraction layers frozen or non-trainable, one or more convolution layer blocks are re-trained on the available training data. Finally, as the third approach for the facies classification task, a combination of transfer learning with the pre-trained VGG-16 CNN model and the traditional ML model is used. Here, the VGG-16 is used to extract features from the drill core dataset, and a few traditional ML models such as random forest, and decision tree are used for the classification task. In the end, the experimental results achieved from these three approaches are compared.

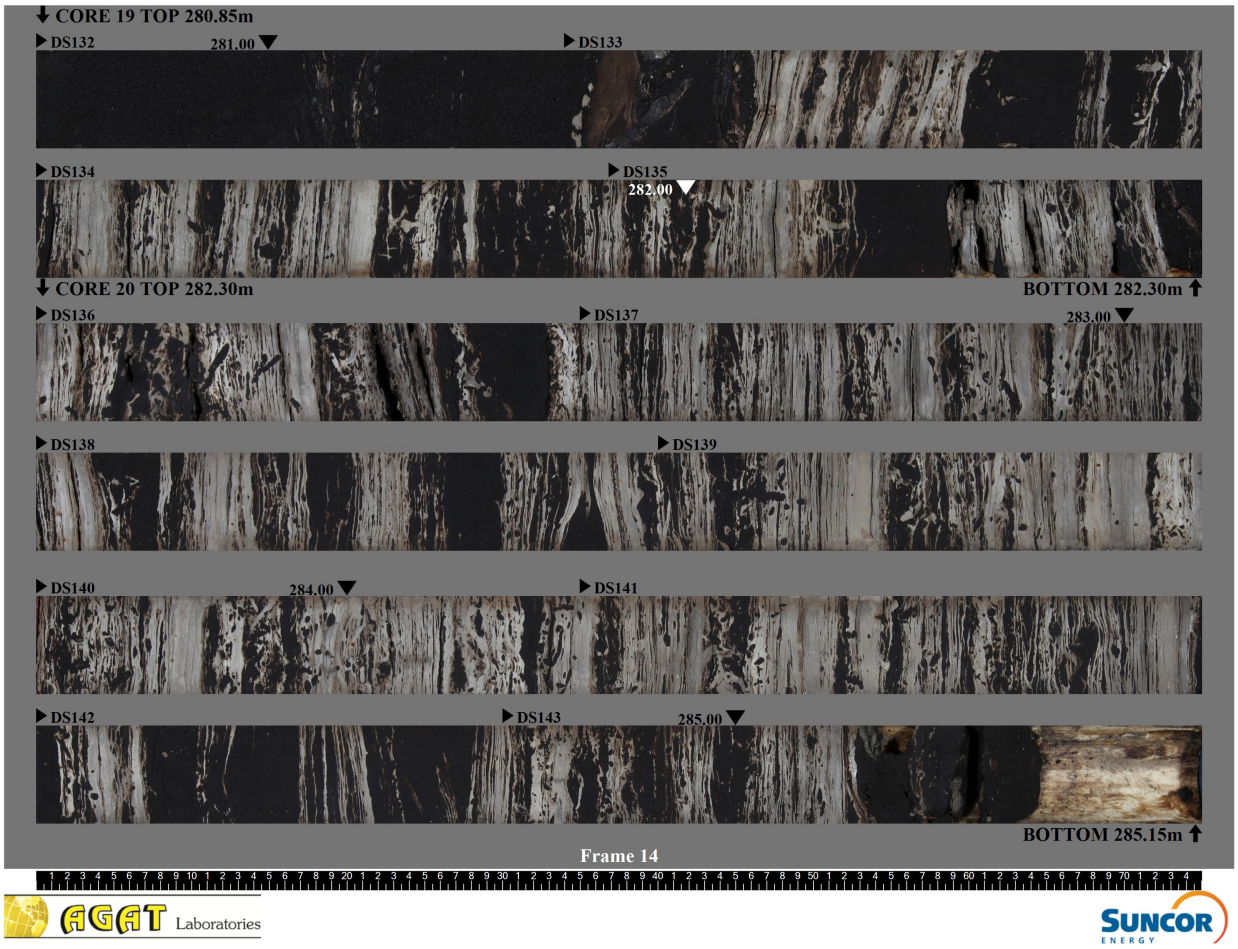


Figure 4.2: Photo of a collection of core samples collected by Suncor Energy.

4.2.1 Data Preparation

Coring is an essnetial process in oil and gas exploration. It refers to the collection of samples from deep inside the earth's crust to determine the existence of oil or natural gas. A rock formation suspected of containing oil and gas is drilled using a special core bit. The coring bit is hollow, allowing it to collect a cylindrical formation sample, called a core. The core is then removed from the borehole and checked for signs of oil and natural gas. Then the drillers store the core samples in the core boxes and log the time and date of drilling, including the depth, core recovery, and possible losses.

Suncor Energy [121], a Canadian integrated energy company based in Calgary, Alberta, that specializes in the production of synthetic crude from oil sands, performed well drilling

(a) Core sample from DS132 to DS133.

(b) Core sample from DS134 to DS135.

(c) Core sample from DS136 to DS137.

(d) Core sample from DS138 to DS139.

Figure 4.3: Example of the cropped photos of core sample.

and collected the core samples from the Athabasca oil sands. Athabasca oil sands are large deposits of bitumen or extremely heavy crude oil located in northeastern Alberta, Canada near Fort McMurray (56.7267° N, 111.3790° W). The Athabasca deposit is the largest known reservoir of crude bitumen globally and the largest oil sand deposit in Alberta [122]. After storing the core samples in the boxes, they are taken to the laboratories, and the photos of the core slabs collection are captured (Figure 4.2). As mentioned earlier, information about the drilled well, possible depth, core ID, among others, are also recorded at this stage. Since the photos contain a collection of core slabs, each core sample is cropped based on the interval points annotated in the core slab photos, as shown in Figure 4.2. For example, starting from DS132 to DS133 indicates one core sample in the collection of core sample photos. The interval starting and ending points have been set to indicate a core sample the PSD is calculated from, which is discussed in Chapter 5. Figure 4.3 shows the photos of four core samples cropped from the photo containing the collection of core slabs shown in Figure 4.2.

As all the core photos are cropped, these are labeled with facies by a domain expert in Suncor Energy. The facies is defined by visual mud index (VMI). Here, the VMI indicates the proportion of mud present in the sample. Based on the different percentages of VMI, different types of facies can be obtained. If the VMI is less than 5%, the facies is classified as F1 also known as sandstone. Similarly, if the VMI is within the range of 5% - 15%, the facies is F2 or Sandy Inclined Heterolithic Stratification (IHS), for 15% - 30% VMI the facies

Facies	Rock Type	VMI
F1	Sandstone	0-5%
F2	Sandy IHS	5-15%
F3	IHS	15-30%
F4	Muddy IHS	30-70%
F5	Mudstone	70-100%
F10	Breccia	Variable VMI

Table 4.1: Classification of facies based on visual mud index.

is F3 or IHS, and so forth up to 100% VMI. Table 4.1 represents the correlation between different facies types and VMI. In addition, Figure 4.4 shows some core samples labeled with different facies types provided by Suncor Energy.

Geologists always want to identify the existence of good quality rock vs poor quality rock so that the wells can be sited in places where the presence of good quality rock is proven. F1 is considered a good quality rock as these have lots of sands and bitumen whereas the other facies such as F3 to F10 contain lots of interbedded sands and muds that make the production critical.

The available data contains the core images labeled with F1, F3, and F4 for the facies classification. Since F1 is a good quality rock and F3 and F4 are the poor quality rock types, F3 and F4 images are concatenated as a single class labeled as "not F1". Therefore, the explored classification models predict if a core sample image is either an F1 or "not F1". Suncor Energy provided a total of 225 core photos labeled as F1, and after concatenating F3 and F4, we have 220 photos labeled as "not F1". In the next section, we briefly present different explored classification models along with the experimental results.

4.2.2 Model Selection

In this section, we present different classification models we explored to classify core photos belonging to either F1 or not F1 category. We briefly describe the model architectures, motivations behinds employing these models in our facies classification exploration workflow, and how we use these models for our core image classification tasks.

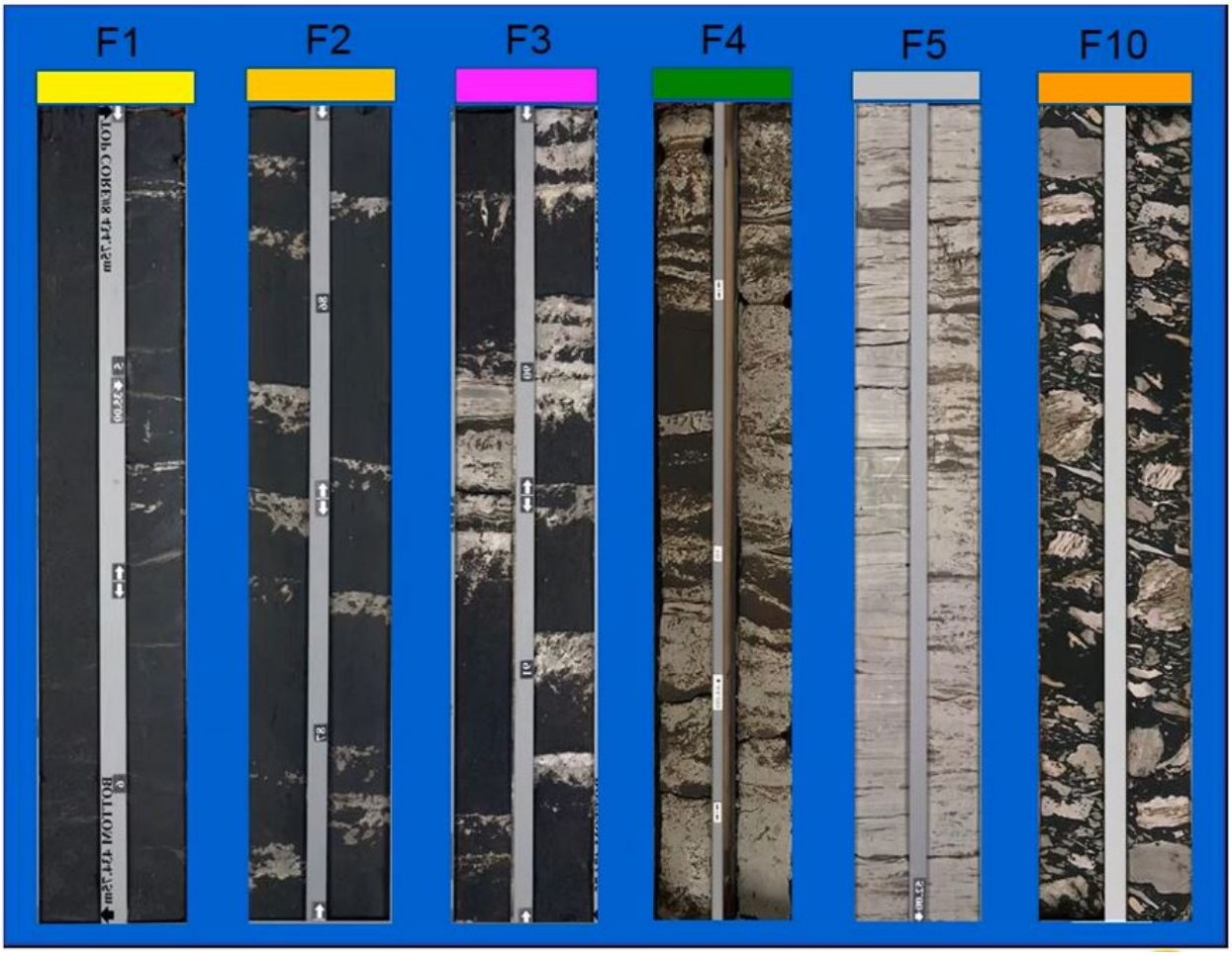


Figure 4.4: Core samples labeled with different facies provided by Suncor Energy.

Approach 1: Transfer Learning on VGG-16

VGG is a CNN with a specific architecture that was proposed by Simonyan et al. [16]. The VGG group participated in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) and submitted the VGG model for competing in object localization and image classification tasks. ImageNet is a dataset of over 15 million labeled high-resolution images belonging to about 22,000 categories. VGG-16 model was trained on over 14 million ImageNet data belonging to 1000 categories. It outperformed other models with 92.7% top-5 test accuracy [2]. This VGG-16 model was trained for weeks and used NVIDIA Titan Black GPUs. The reason behind following the VGG architecture is not only the high-accuracy but

[2]VGG-16 model won first and second place in the 2014 ILSVC

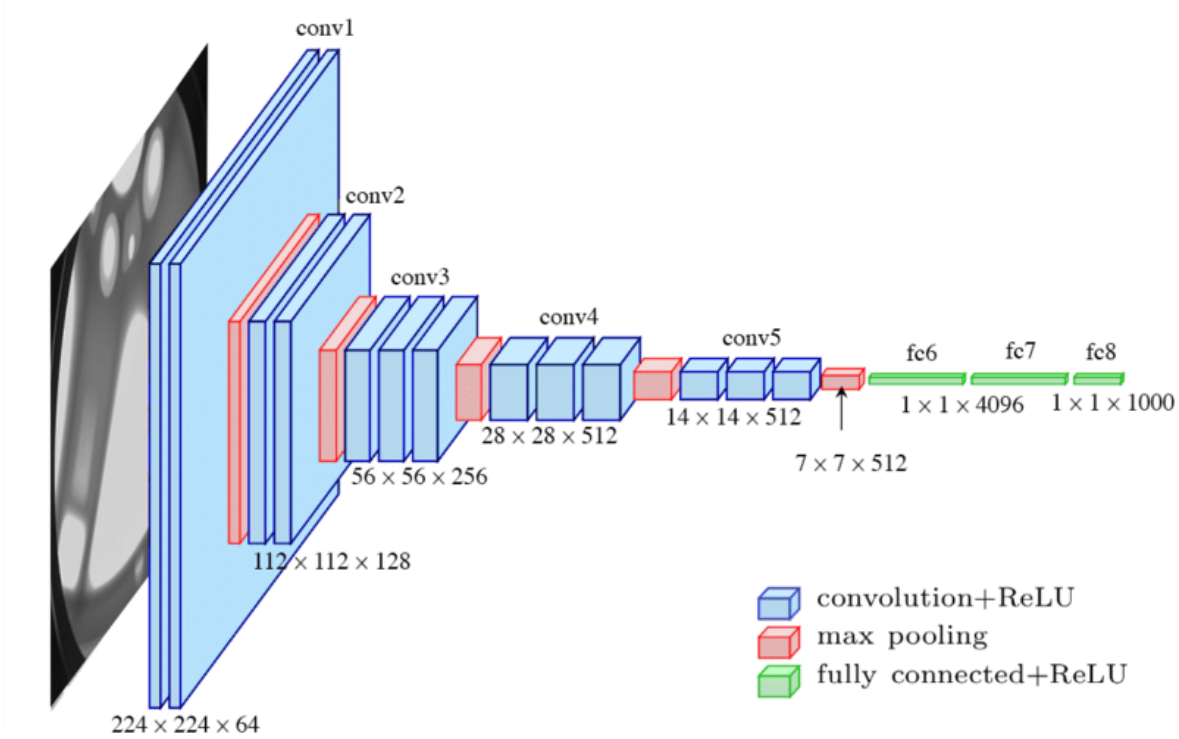


Figure 4.5: Architecture of VGG-16
(Ferguson, M., Ak, R., Lee, Y. T. T., Law, K. H. (2017, December). Automatic localization of casting defects with convolutional neural networks. In *2017 IEEE international conference on big data (big data)* (pp. 1726-1735). IEEE).

also the efficiency, and more importantly, adaptability to other image classification problems than ImageNet [39]. VGG architecture has been implemented to address problems in different research domains and demonstrated success compared to many other pre-trained models. Furthermore, despite having some drawbacks, the VGG model is very good at extracting features from the data due to having many convolutional layers.

VGG has two different architectures: VGG-16 and VGG-19 with 16 and 19 layers, respectively. We employ the VGG-16 architecture for the core image classification stage. Its network starts with two convolutional layers, followed by a max-pooling one. The collection of convolutional layers and the max-pooling layer is called a convolutional layer block. For the first two convolutional layer blocks, it follows the same combination. Unlike the first and second convolutional layer blocks, where they have the combination of two convolu-

tional layers followed by a max-pooling layer, the rest of the model architecture contains the combinations of three convolutional layers followed by a max-pooling layer. Overall, the VGG-16 model is designed with five convolutional layer blocks. Finally, after the five convolutional layer blocks, at the top of the model, it contains three FC layers where the last FC layer produces the model's output. Figure 4.5 shows the architecture of the VGG-16 model. Its weights are available on different platforms (e.g. Keras) and can be used for further analysis [50].

The key idea behind the architecture is to increase the depth of the network by adding more convolutional layers while keeping other network parameters fixed. The number of trainable parameters is managed by keeping the convolution filter size very small (e.g., 3 X 3) throughout all layers. The width of the layers (i.e. number of channels) increases as we progress through the network to later layers. The increase in the number of channels in later layers is significant since they capture more complex features, for which a larger receptive field is required [33]. The convolution stride is fixed to one pixel due to the filters' small size (i.e., 3 X 3). All the hidden layers utilize the Rectified Linear Units (ReLU) activation function. During training, the input to the VGG-16 is a fixed-size (224 X 224 X 3) image, where (224 X 224) represents the width and height of the input image and 3 represents the three color channels: R = Red, G = Green, B = Blue.

To employ transfer learning on VGG-16, first, we pick which layer of VGG-16 to use for feature extraction. Since the VGG-16 is pre-trained on images belonging to 1000 categories, the very last classification layer ('fc8' in Figure 4.5) is not very useful. Instead, we depend on the last convolutional layer block ('conv5' in Figure 4.5) as the features extracted in this layer retain more generality compared to the final layer. Therefore, the VGG-16 model preloaded with weights trained on ImageNet is instantiated, not including the FC or top layers ('fc6', 'fc7', and 'fc8' in Figure 4.5). As the model is instantiated, we freeze all the convolution blocks. Freezing prevents the weights in a given layer from being updated during training. After that, the layers associated with the classification tasks are defined

to generate predictions from the block of features. Firstly, the extracted features from the last feature extraction layer are flattened so that the pooled feature map is converted into a 1-dimensional vector passed to the FC layer. Since the model predicts whether the input core image belongs to either F1 or "not F1" facies type, a sigmoid activation function is used for this binary classification in the last classification layer. Table 4.2 shows the overall summary of the VGG-16 model, the output shape of each layer, and the number of parameters. In this table, all the layers (i.e., from the first to the last convolutional layer block) are utilized as the pre-trained layers of the VGG-16 model. Based on the features extracted from these layers, only the top layers are associated with the classification of core photos. Note that the symbol (*) denotes the layers we introduce to the original model architecture for the core image classification task. In this model, the number of trainable parameters is 262,401 and the number of non-trainable parameters is 14,714,688, where the total number of parameters in VGG-16 is 14,977,089.

After creating the model, to compile it we use binary cross-entropy as the loss function. The equation of binary cross-entropy is given in Chapter 3, Equation 3.7. Another parameter required to compile the model is the optimizer. Here, we use Adam for the model optimizer. Adaptive Momentum Estimation also called Adam is an optimization algorithm that can be used to update network weights iteratively based on training data [123]. Adam is a popular algorithm in the field of DL as it achieves good results fast since it combines the properties of RMSProp and Stochastic Gradient Descent with Momentum (SGDM) [123]. The third parameter required to complete the model compilation is the evaluation metric. In this approach, we evaluate the performance of the model based on the overall accuracy. Later in this chapter, we elaborate more on the evaluation metrics used to monitor the performance of the classification model.

Layer (Type)	Output Shape	Number of Parameters
input_1 (InputLayer)	[(None, 64, 64, 3)]	0
block1_conv1 (Conv2D)	(None, 64, 64, 64)	1792
block1_conv2 (Conv2D)	(None, 64, 64, 64)	36928
block1_pool (MaxPooling2D)	(None, 32, 32, 64)	0
block2_conv1 (Conv2D)	(None, 32, 32, 128)	73856
block2_conv2 (Conv2D)	(None, 32, 32, 128)	147584
block2_pool (MaxPooling2D)	(None, 16, 16, 128)	0
block3_conv1 (Conv2D)	(None, 16, 16, 256)	295168
block3_conv2 (Conv2D)	(None, 16, 16, 256)	590080
block3_conv3 (Conv2D)	(None, 16, 16, 256)	590080
block3_pool (MaxPooling2D)	(None, 16, 16, 256)	0
block4_conv1 (Conv2D)	(None, 8, 8, 512)	1180160
block4_conv2 (Conv2D)	(None, 8, 8, 512)	2359808
block4_conv3 (Conv2D)	(None, 8, 8, 512)	2359808
block4_pool (MaxPooling2D)	(None, 4, 4, 512)	0
block5_conv1 (Conv2D)	(None, 4, 4, 512)	2359808
block5_conv2 (Conv2D)	(None, 4, 4, 512)	2359808
block5_conv3 (Conv2D)	(None, 4, 4, 512)	2359808
block5_pool (MaxPooling2D)	(None, 2, 2, 512)	0
flatten (Flatten)*	(None, 2048)	0
dense (Dense)*	(None, 128)	262272
dense_1 (Dense)*	(None, 1)	129

Table 4.2: Summary of the VGG-16 architecture after introducing new layers.

Approach 2: Fine-Tuning VGG-16

In the first approach, we only train a few layers on top of the VGG-16 base model. The weights of the pre-trained network are not updated during training. However, in the second approach, we train or fine-tune the weights of the last convolutional layer block of the pre-trained VGG-16 model along with training the top layers associated with the classification task. Here, the training process forces the weights to be tuned from generic feature maps to features explicitly associated with the dataset. In this approach, instead of fine-tuning the whole pre-trained model weights, only one or a small number of top layers should be trained. In most CNN models, the higher up a layer is, the more specialized it is. The first few layers learn the generic features such as vertical lines, horizontal lines, orientation, etc. that generalize to almost all types of images. As we go higher up the layers, the features

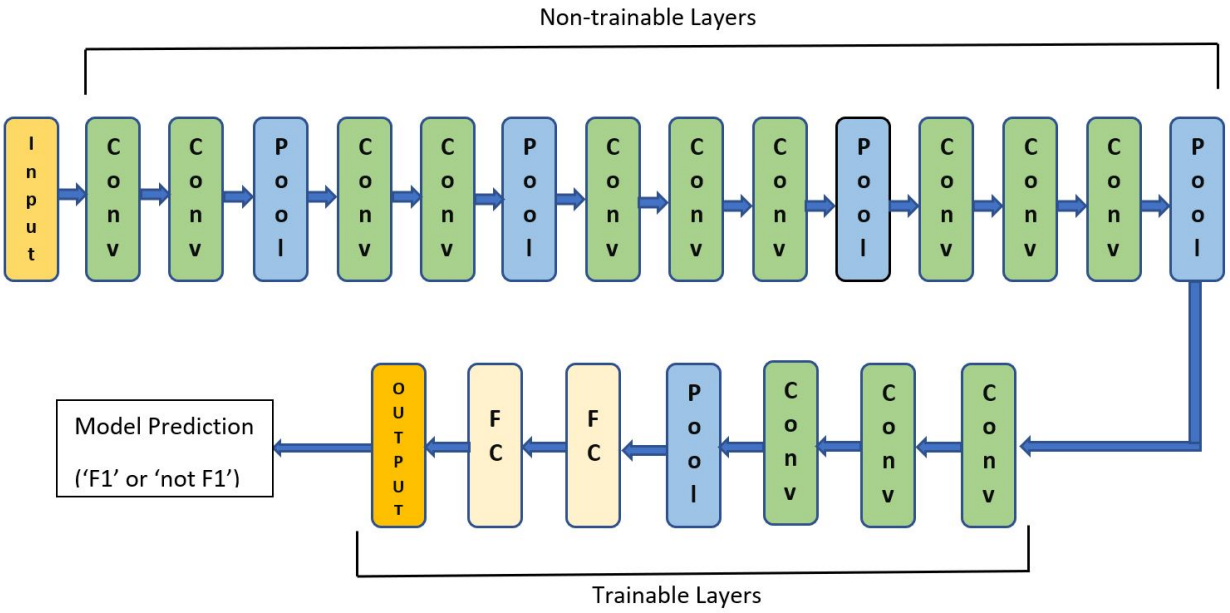


Figure 4.6: Block diagram of the fine-tuned VGG-16 model.

are increasingly more specific to the dataset on which the model is trained. Therefore, fine-tuning aims to adapt these specialized features to work with the new dataset rather than overwrite the generic learning. In this approach, we un-freeze the last convolution block of the VGG-16 model. Therefore, unlike the first approach, instead of only training the classification layers, we train the fifth convolution block as well. In this fine-tuned VGG-16 model, the number of trainable parameters is 7,341,825 and the number of non-trainable parameters is 7,635,264. Finally, for compiling the model, the same parameter setting is used as the first approach. Figure 4.6 shows the block diagram of the fine-tuned VGG-16 model explored in this approach.

Approach 3: Combination of VGG-16 and Traditional Machine Learning Model

In this approach, first, we employ transfer learning on VGG-16 to extract features from the core photos. Features are obtained from the last convolution block of VGG-16 (See Figure 4.7). We train different ML classification models to classify the core photos based on the facies types using the extracted features. As the traditional ML approaches, we explore the

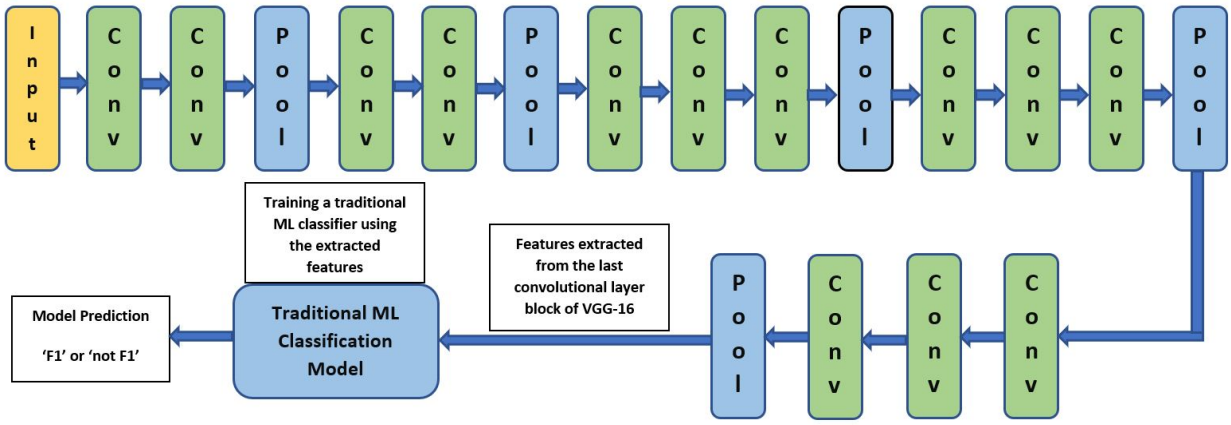


Figure 4.7: Block diagram of the approach based on the combination of VGG-16 and a traditional machine learning classification model.

random forest and decision tree classifiers. Random forest is one of the most used supervised ML algorithms because of its simplicity, diversity, and flexibility as it can be used for both classification and regression. Most of the time, random forest produces reasonably good results even without any hyper-parameter tuning [145]. One of the biggest problems in ML is over-fitting. However, even if the random forest model is trained with many decision tree models that may have low bias and high variance, after averaging the results obtained by the decision tree models based on the majority votes, the random forest shows low bias and low variance, meaning that the model does not overfit [143] [144]. Moreover, random forest is faster than most of the traditional ML algorithms since each of the decision tree in the random forest are trained in parallel with subsets of data and features.

Decision trees are the building blocks of a random forest model as shown in Figure 4.8. The decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences. Decision tree creates a flowchart structure where each internal node represents a 'test' applied to an attribute, with each branch representing the outcome of the test and each leaf node representing a class label or the decision taken after computing all attributes [124]. The paths from the root to the leaf represent classification rules. A classification problem dealing with data not linearly separable means that a single

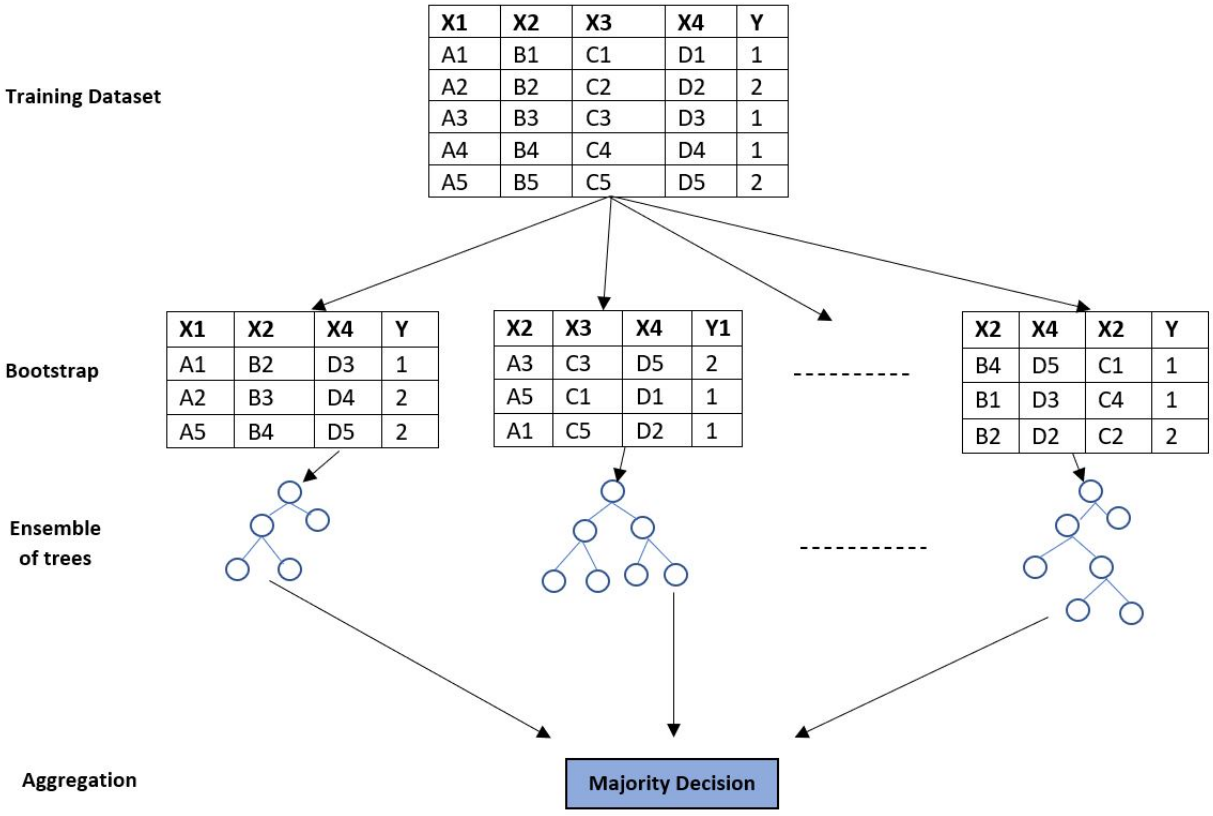


Figure 4.8: Implementation of random forest classifier on a dataset that has four features (X1, X2, X3, and X4) and two classes (Y = 1 and 2). Random forest classifier is an ensemble method that trains several decision trees in parallel with bootstrapping followed by aggregation. Each tree is trained on different subsets of training samples and features (adapted from [53]).

line cannot be drawn through the data to classify the points. Decision tree approximates a non-linear boundary by drawing several axis-aligned boundaries through the data to separate the data points. Random forest, on the other hand, is a model made up of many decision trees. Instead of just simply averaging the prediction of trees, a random forest model does the random sampling of training data when building trees, and considers random subsets of features when splitting nodes. Random forest combines hundreds or thousands of decision trees, trains each one on a slightly different set of observations, splitting nodes in each tree considering a limited number of the features. The final predictions of the random forest are made by averaging the predictions of each tree. Random forest adds additional randomness to

the model while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model and reduces over-fitting. On the other hand, as a decision tree algorithm grows deep, it suffers from over-fitting. However, random forest prevents over-fitting by creating random subsets of the features and building smaller trees using those subsets. Afterward, it combines the sub-trees.

Due to being superior to other traditional ML classification models, the better performance of random forest classifier compared to the decision tree in core image classification is also reflected by the obtained experimental results. We present the experimental results achieved by all the aforementioned explored approaches in Section 4.3 of this chapter.

4.2.3 Training the Models

The pixel values in the image data must be scaled before providing the images as input to a deep neural network model during the training or evaluation of the model. Therefore, we re-scale the pixel values from 0-255 to the range 0-1 preferred for neural network models. Scaling data to the range of 0-1 is traditionally referred to as normalization. As we have all the images and corresponding labels, we split the data into a training and a testing set before training the models. Testing set is used to evaluate how the trained model performs on the data that the model has never seen before. We further split our training set into K number of subsets called folds. We iteratively fit the models K times, training the data on K-1 of the folds and evaluating the K-th fold (called the validation data). This procedure is called K-fold cross-validation. This method generally results in a less biased model compared to other methods since it ensures that every observation from the original dataset has the chance of appearing in training and validation. For our core image classification, we implement 10-fold cross-validation (K = 10), meaning that the available dataset is divided into ten equal parts, and the following process runs ten times, each time with a different validation set. The process has four main steps:

1. Take the group as a validation dataset.

2. Take the remaining groups as a training dataset.

3. Fit the model on the training set and evaluate it on the validation set.

4. Retain the evaluation score and discard the model

In this process, at the very end of the training, we average the performance on each of the folds to come up with final validation metrics for the model.

As we have the dataset and the models, we train the models for an upward bound of 20 epochs in 64 batches, meaning that the data is passed through the network 20 times. Batch size refers to the number of training examples utilized in one iteration. Along with training the models, we use model checkpoint to save the models and the weights in a checkpoint file so that the models or weights can be loaded to continue the training from the state saved.

In terms of computing resources, the experiments were performed using a 9th Generation Intel Core i7 - 9750H, with six-core processors @ 2.6 GHz, 16 GB RAM running on Windows 10 Operating System and NVIDIA GeForce RTX 2060 GPU. For programming, we used Python 3.8.5 and we utilize Scikit-learn, TensorFlow 2.3.1 platform and Keras 2.4.3 libraries for ML and neural network workflow. We also use python's scientific computing package NumPy, pandas for data analysis and manipulation, and Matplotlib for data visualization.

4.3 Experimental Results

The performances of the approaches as mentioned above were evaluated in terms of accuracy, ROC curve (i.e., Receiver Operating Characteristic), AUC score (i.e., Area Under the ROC Curve), and confusion matrix. We also analyze each model's precision, recall, and f1-scores, although these metrics are generally considered while evaluating an imbalanced classification. The recall, precision, and f1-score metrics are reported by determining the macro-average for each class. The precision, recall, and f1-score metrics are determined for each class to

	Actual Class: F1	Actual Class: not F1
Predicted Class: F1	True Positive (TP)	False Positive (FP)
Predicted Class: not F1	False Negative (FN)	True Negative (TN)

Table 4.3: Confusion Matrix.

calculate the macro-average, and the corresponding unweighted average is measured. In this section, we describe different evaluation metrics that we investigated to evaluate the performance of the explored approaches. Moreover, we present the model comparisons based on the experimental results.

4.3.1 Evaluation Metrics

In classification problems, a confusion matrix is a table that summarizes the number of correct and incorrect predictions with count values and is broken down by each class. A confusion matrix provides an insight into the errors being made by a classifier and, more importantly, the types of errors that are being made. A confusion matrix overcomes the limitation of using a classification accuracy alone, where classification accuracy only shows the ratio of correct predictions to total predictions made, which can be sometimes misleading as it hides the detail we need to understand the performance of a classification model better. Especially, classification accuracy is not an ideal evaluation metric when dealing with an imbalanced dataset. However, a confusion matrix can provide a better understanding of the classifier's overall performance, as a few other evaluation metrics can be derived from this as well.

The confusion matrix for the classification of core photos based on facies where the class labels are F1 and not F1 is shown in Table 4.3. The top row represents the actual class whereas the left column represents the labels predicted by the classifier. Here we assume that the class F1 is positive, and the class "not F1" is negative. Therefore, for a sample data point, if both the actual class and the predicted class are F1, it is called true positive (TP), meaning that a positive instance (F1) has been correctly predicted as positive (F1).

If the actual class is F1 (i.e., positive) but the classifier predicted that instance as "not F1" (i.e., negative), it is a false negative (FN). Similarly, if both the actual class and the predicted class are "not F1" (i.e., negative), then it is called a true negative (TN), meaning that a negative instance (i.e., not F1) has been correctly predicted as negative (i.e., not F1). On the other hand, a negative (i.e., not F1) instance predicted as positive (i.e., F1) by the classifier is marked as false positive (FP). In the confusion matrix, TP and TN are the correct predictions whereas, FP and FN are the wrong predictions by the classifier. FP and FN are also called Type-I and Type-II errors, respectively. Any classification model aims to reduce Type-I and Type-II errors. Since our dataset is balanced, we only focus on the classification accuracy defined by [44]:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} \tag{4.1}$$

where the proportion of the total number of correct predictions to all predictions.

When building a classification model with a balanced dataset, the model does not get biased based on the different categories we have in the binary classification problem. However, to have a more detailed classification model performance, three other evaluation scores can be obtained from a confusion matrix: recall, precision, and f1-score. These three evaluation metrics are very important to consider while dealing with imbalanced datasets. Out of the total positive (F1) actual values, the metric recall determines how many values the model correctly predicted as positive (F1). On the other hand, precision determines, out of total predicted positive (F1) results, how many results were positive (F1). The metric recall is sometimes also called true positive rate (TPR), or sensitivity, and Precision is also called Positive Predictive Value. Recall and precision are defined as follows [44]:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.2}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.3}$$

For a problem statement, if the Type-I error or the FP value needs to be reduced, precision must be used for the evaluation metric, whereas if the Type-II error or the FN value needs to be reduced, recall must be used. For a problem statement, if both FP and FN are equally important and both the Type-I and Type-II values need to be reduced, then f1-score has to be considered to achieve the most accurate prediction, which is defined by [44]:

$$\mathrm{f}_1 - \mathrm{score} = 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{4.4}$$

ROC curve and AUC score are two important metrics for binary classification problems. There are two ways to solve a classification problem statement: the first one is based on class labels, and the second one is based on probabilities. For a binary classification problem, where there are two output values or class labels (e.g., 0 and 1), by default, the threshold value is 0.5, meaning that if the prediction of the classification model is greater than or equal to 0.5, the predicted output label would be 1 and 0 otherwise. For the drill core image classification task, we represent the class F1 as 1 and class not F1 as 0. Therefore, if the classifier's prediction is greater than or equal to the threshold value, the predicted class is F1, and if the prediction is less than the threshold value, the predicted class is "not F1". However, in the case of probabilities, we find the suitable class label by selecting the optimal threshold value instead of the default threshold value of 0.5. The accuracy might not be the highest with the optimal threshold value, but it balances between the TPR and FPR. This optimal threshold value can be determined using the ROC curve, which is a probability curve that plots the TPR against FPR at various thresholds. The AUC measures the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve. The higher the AUC, the better the performance of the model at distinguishing between the two classes. When AUC equals 1, the classifier correctly distinguishes between all the positive and negative class points. If, however, the AUC equals 0, the classifier predicts all positive classes as negative and vice versa. To construct the confusion matrix for the core image

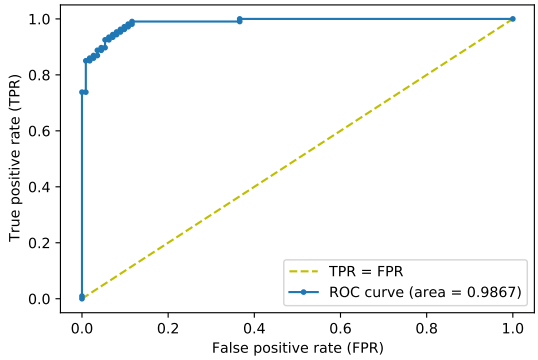
Model	Accuracy	Recall	Precision	F1-score
Transfer Learning on VGG-16	93.15	94.00	94.00	94.00
Fine-tuning VGG-16	96.30	98.00	98.00	98.00
VGG-16 and Random Forest Classifier	**98.87**	**99.00**	**99.00**	**99.00**
VGG-16 and Decision Tree	94.52	96.38	93.52	94.84

Table 4.4: Summary of classification performance of the explored methods on oil sands drill core dataset.

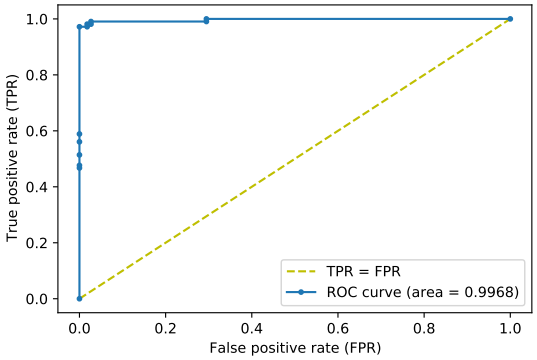
classification, we first determine the optimal threshold value based on the ROC curve and AUC score and then observe the number of correct and incorrect predictions of the explored classification models using the confusion matrix. Moreover, to observe whether or not the VGG-16 models are over-fitting, we also observe the learning rate curves.

4.3.2 Model Comparison

Table 4.4 summarizes the overall classification performances of the three explored methods based on the accuracy, recall, precision, and f1-score. The first explored approach, where transfer learning on the pre-trained VGG-16 model is employed, shows 93.15% and 94.00% accuracy for the recall, precision, and f1-score. Better classification performance is demonstrated by the second explored method where the last convolution block along with the classification layers of the pre-trained VGG-16 model is trained on the drill core dataset. This method shows 96.30% and 98% accuracy for recall, precision, and F1 score. In terms of training a traditional ML classification model with the features extracted from the last convolutional layer block of VGG-16, the best performance is achieved by the random forest classification model that also outperforms the other explored methods. This method shows the classification accuracy of 98.87% and the recall, precision, and f1-score of 99.00%. However, the performance of the decision tree is not as good as random forest which is understandable due to the reasons stated while comparing the characteristics of random forest and decision tree before in this chapter. In every explored approach, the accuracy is calculated by averaging the accuracies achieved at the end of every training in cross-validation.



(a) ROC curve and AUC score for transfer learning on VGG-16.

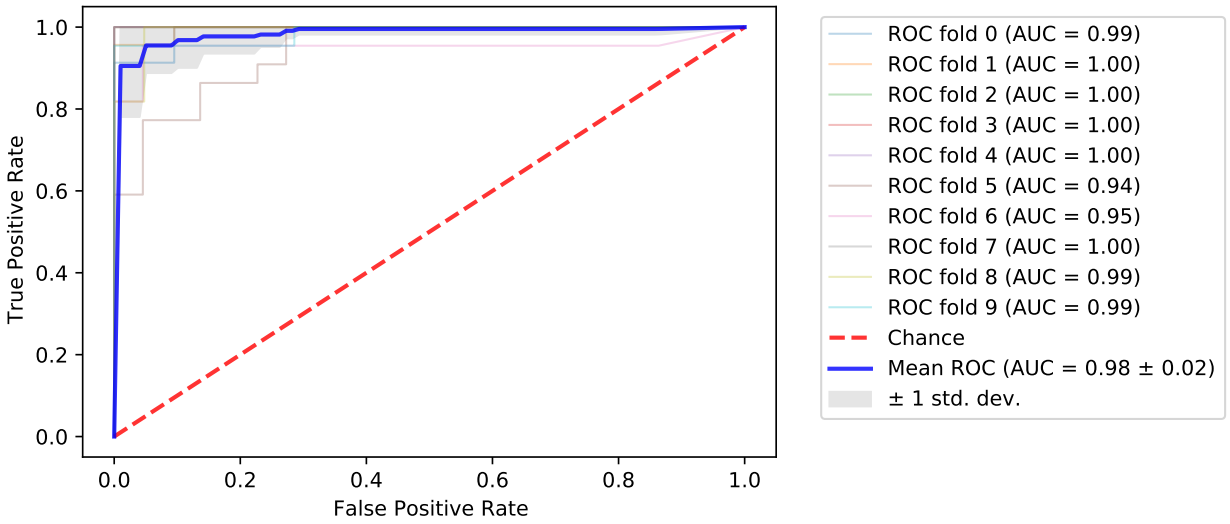


(b) ROC curve and AUC score for the fine-tuned VGG-16.

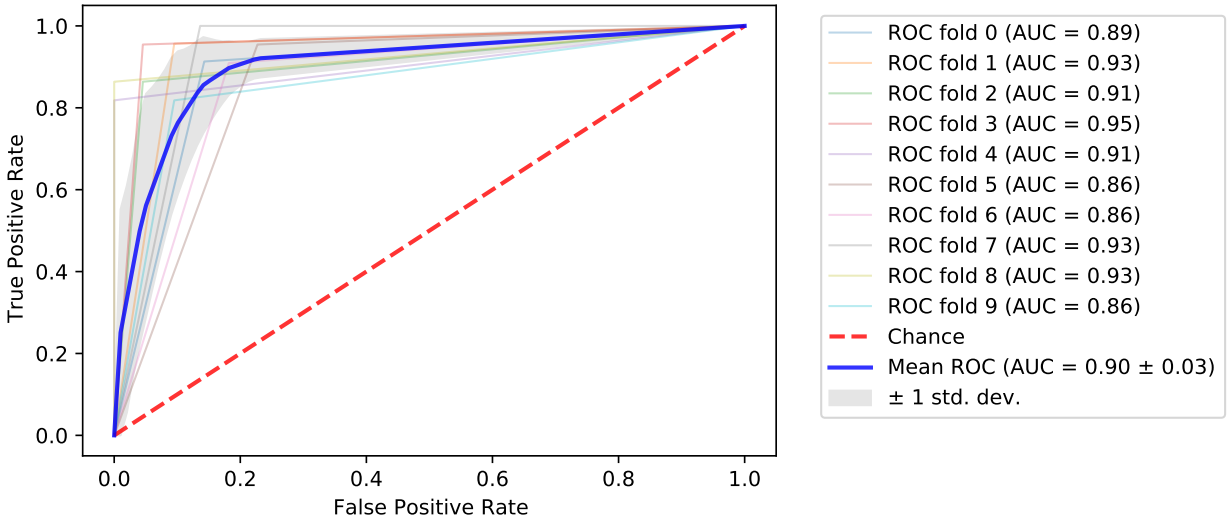
Figure 4.9: ROC curve and AUC score for the explored VGG-16 models.

Figure 4.9 shows the ROC curves and AUC scores for the explored VGG-16 methods. The probability measurement in the TPR vs FPR shows the separability among classes. The more area under the ROC curve is the more the capability of the classifier to show the sensitivity to separate the classes correctly. For the first explored approach, the AUC of ROC is 0.9867 and 0.9968 for the second approach, where the fine-tuning is done on VGG-16. Figure 4.10 shows the ROC curves and AUC scores for the combination of transfer learning on VGG-16 and the explored traditional ML models (random forest and decision tree). It shows the ROC curves for each fold and the mean ROC is plotted along with them. For random forest, we see that the AUC score ranges from 0.94 to 1.00, and for decision tree the score ranges from 0.86 to 0.93. Therefore, the mean AUC for the random forest is around 0.98 and for the decision tree, the score is 0.90. These scores also prove that the random forest is performing better than the decision tree model for core image classification.

Table 4.5 shows the confusion matrices for the explored approaches. The confusion matrix in 4.5a shows that the transfer learning on VGG-16 correctly predicts 47 F1 labeled data among the total of 48 actual F1 labeled test data. On the other hand, it correctly predicts 38 “not F1” labeled data among the total number of 40 test data labeled as “not F1”. Therefore, from the total of 88 test data, the transfer learning on VGG-16 wrongly predicts



(a) ROC curve and AUC score for the combination of VGG-16 and random forest.



(b) ROC curve and AUC score for the combination of VGG-16 and decision tree.

Figure 4.10: ROC curve and AUC score for the combination of VGG-16 and the traditional machine learning models.

	Actual: F1	Actual: not F1
Predicted: F1	47	1
Predicted: not F1	2	38

(a) Confusion Matrix for transfer learning on VGG-16.

	Actual: F1	Actual: not F1
Predicted: F1	47	1
Predicted: not F1	1	39

(b) Confusion Matrix for fine-tuned VGG-16.

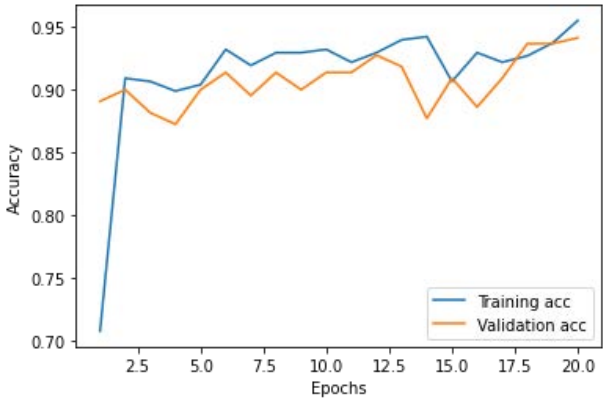
	Actual: F1	Actual: not F1
Predicted: F1	47	1
Predicted: not F1	0	40

(c) Confusion Matrix for the combination of VGG-16 and random forest.

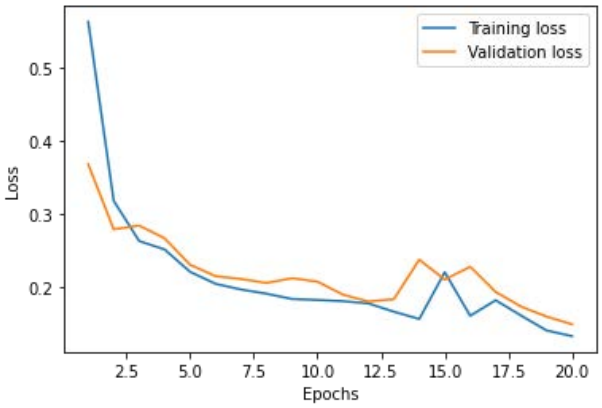
Table 4.5: Confusion matrix of the explored approaches for core image classification.

three test images. However, the second and third explored methods do slightly better than the first method in reducing the number of wrong predictions. Tables 4.5b and 4.5c show that using the fine-tuned VGG-16 model, the number of wrong classification is two, and the combination of VGG-16 and random forest classifier wrongly predicts only one test data among 88 test data. For all the explored methods, it is observed that one of the F1 labeled data is misclassified as "not F1" by all the explored methods. However, although the first and second approaches wrongly predict only two and one "not F1" labeled data as F1, the third approach can correctly predict all the not F1 labeled images. Confusion matrices here show the prediction performance of the explored models based on the test dataset that the models have never seen before.

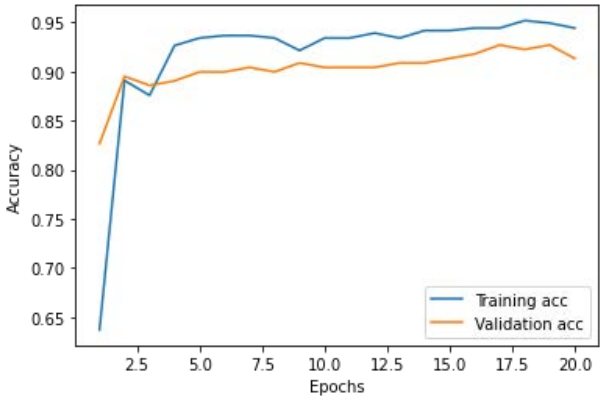
Figure 4.11 and Figure 4.12 demonstrate the average learning curves of the transfer learning on VGG-16 pre-trained CNN and the fine-tuned VGG-16 model respectively with 10-fold cross-validation on our oil sands drill core image dataset. For each value of K, the training and validation accuracy and loss over epochs demonstrate whether the model is generalizing or memorizing. If the training and validation loss gradually decreases and the training and validation accuracy gradually increase over epochs, a generalized pattern is



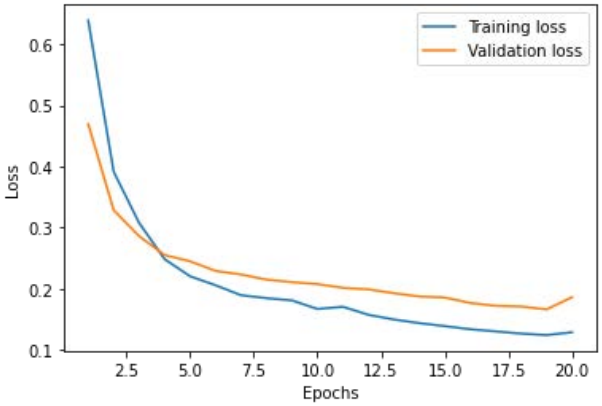
(a) Average training and validation accuracy in K-fold cross validation (fold 1).



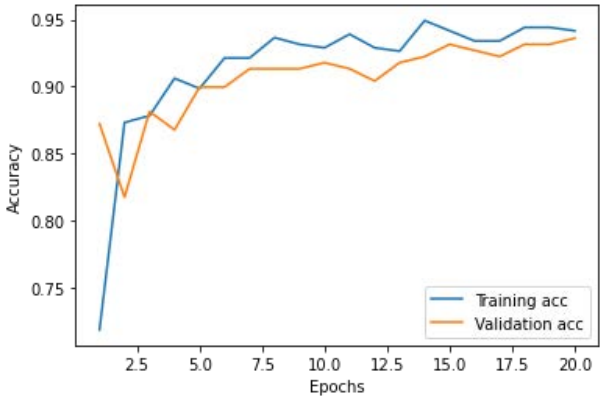
(b) Average training and validation loss in K-fold cross validation (fold 1).



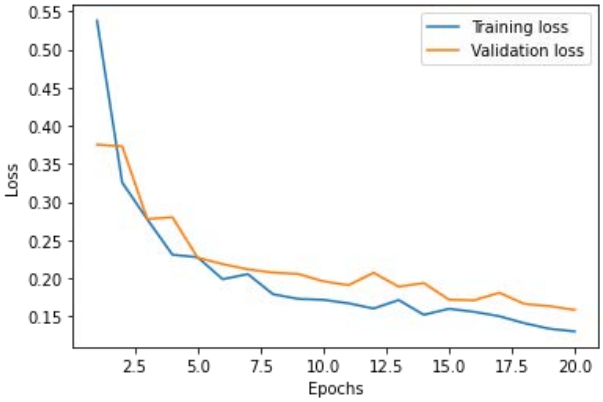
(c) Average training and validation accuracy in K-fold cross validation (fold 2).



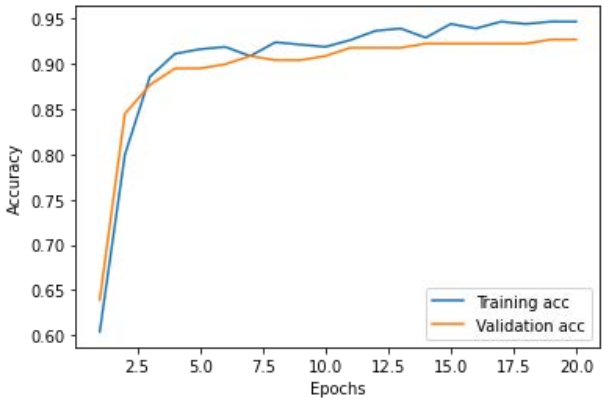
(d) Average training and validation loss in K-fold cross validation (fold 2).



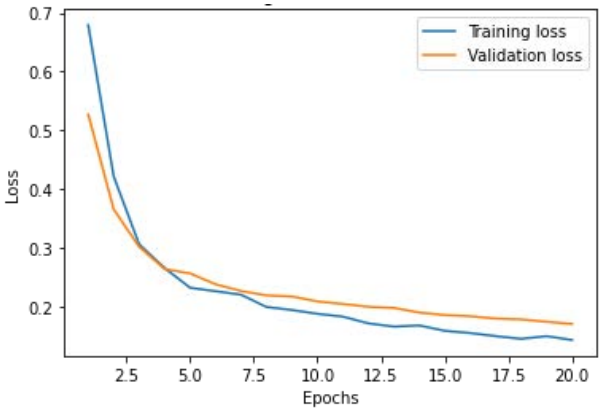
(e) Average training and validation accuracy in K-fold cross validation (fold 3).



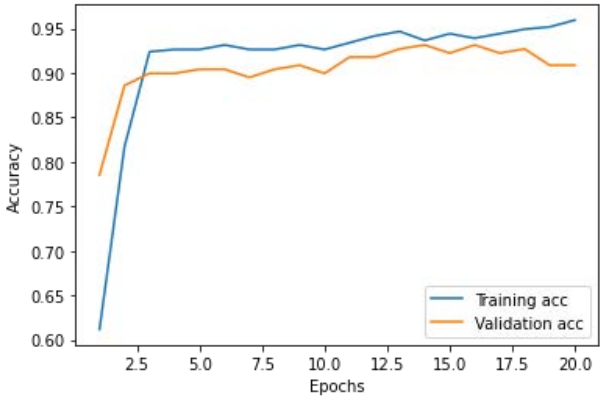
(f) Average training and validation loss in K-fold cross validation (fold 3).



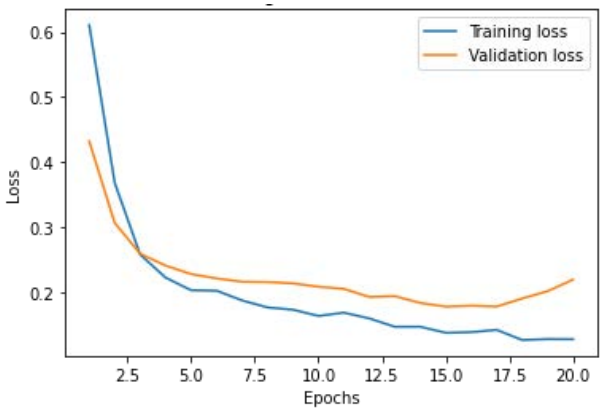
(g) Average training and validation accuracy in K-fold cross validation (fold 4).



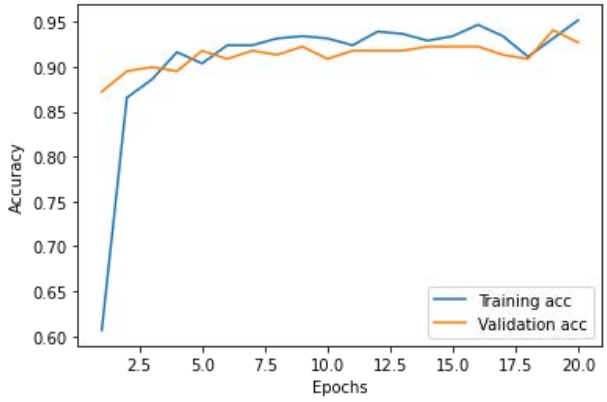
(h) Average training and validation loss in K-fold cross validation (fold 4).



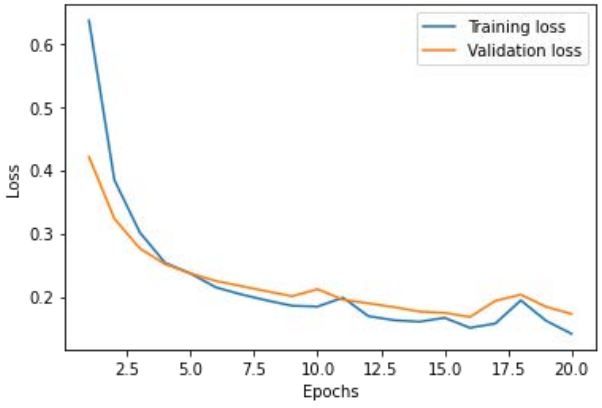
(i) Average training and validation accuracy in K-fold cross validation (fold 5).



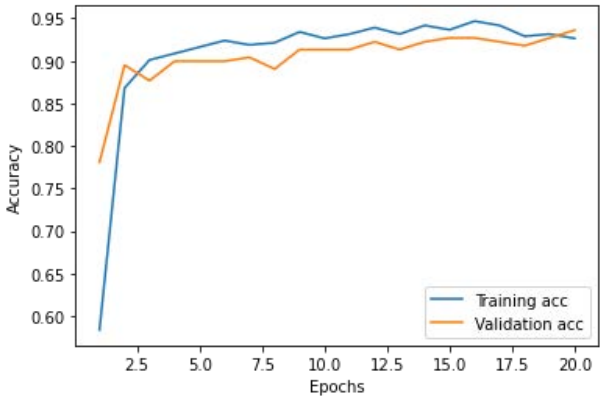
(j) Average training and validation loss in K-fold cross validation (fold 5).



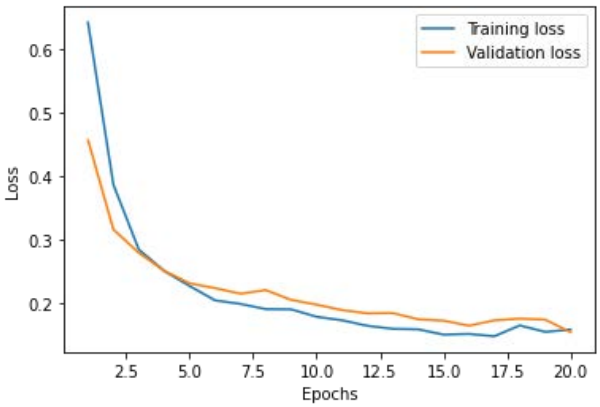
(k) Average training and validation accuracy in K-fold cross validation (fold 6).



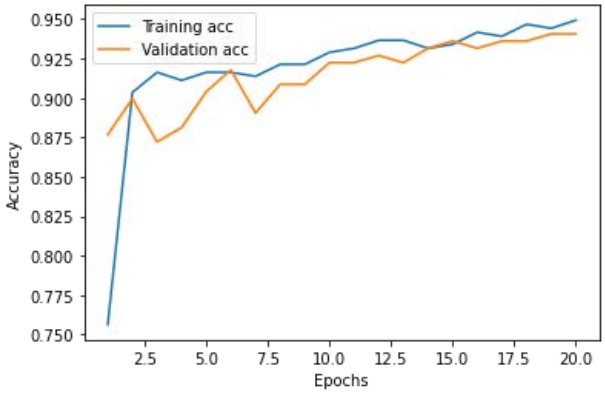
(l) Average training and validation loss in K-fold cross validation (fold 6).



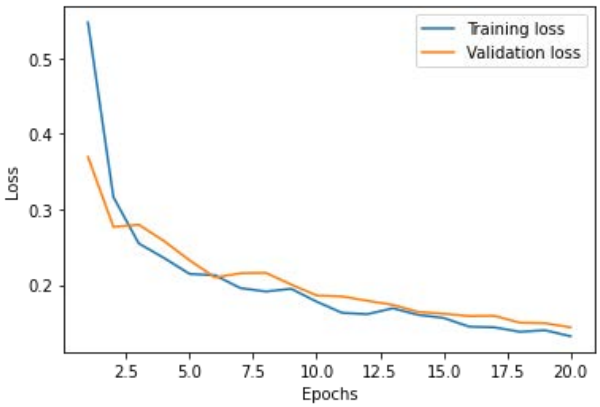
(m) Average training and validation accuracy in K-fold cross validation (fold 7).



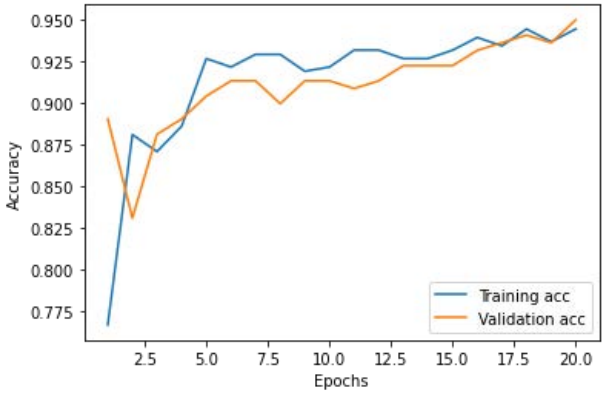
(n) Average training and validation loss in K-fold cross validation (fold 7).



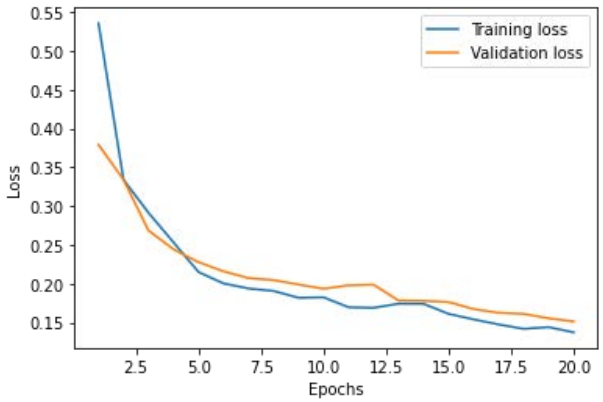
(o) Average training and validation accuracy in K-fold cross validation (fold 8).



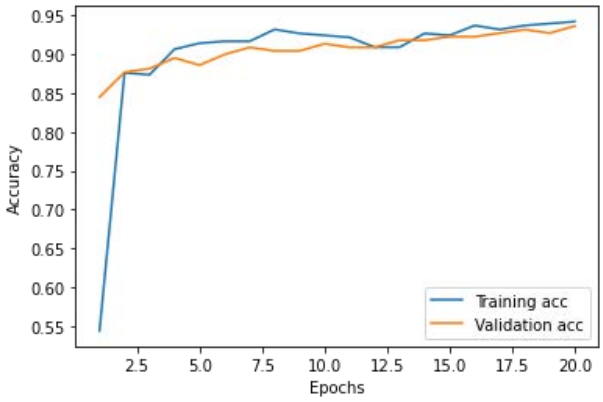
(p) Average training and validation loss in K-fold cross validation (fold 8).



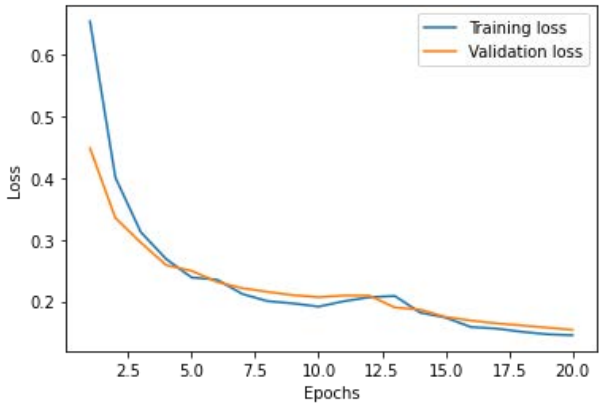
(q) Average training and validation accuracy in K-fold cross validation (fold 9).



(r) Average training and validation loss in K-fold cross validation (fold 9).



(s) Average training and validation accuracy in K-fold cross validation (fold 10).



(t) Average training and validation loss in K-fold cross validation (fold 10).

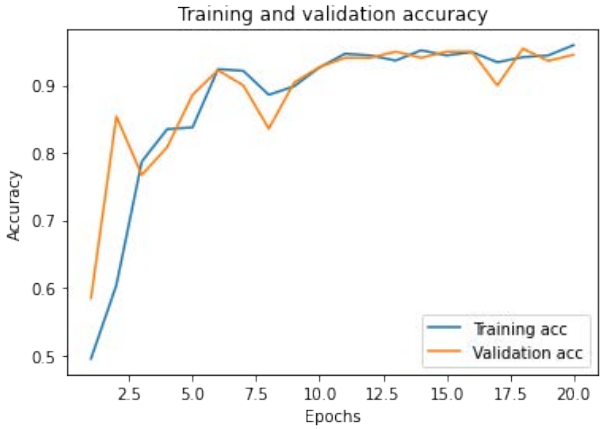
Figure 4.11: Average learning curves of transfer learning on VGG-16 with 10-fold cross validation.

learned by the model. Thus, the model over-fitting can be identified from the learning curve.

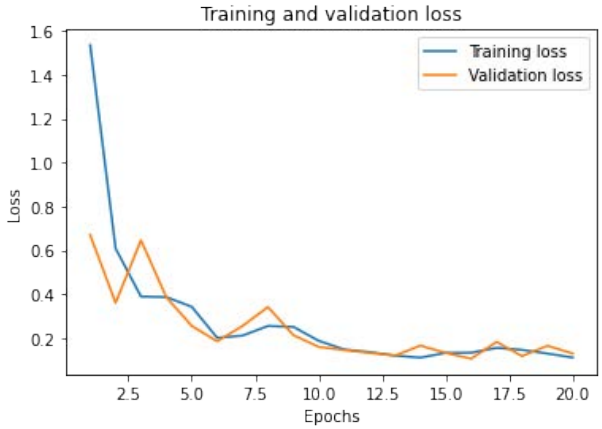
Observing the learning curves for both the explored approaches, it can be seen that the direction of the validation loss for both the models is downward over epochs. Moreover, for both the models, the difference between the validation and training accuracy/loss is small for each value of K. Therefore, it also determines that the models have not suffered from overfitting. Since every observation from the original dataset has the chance of appearing in both training and validation due to employing 10-fold cross-validation, the experimental results also reflect the reliability of the explored models.

4.4 Summary

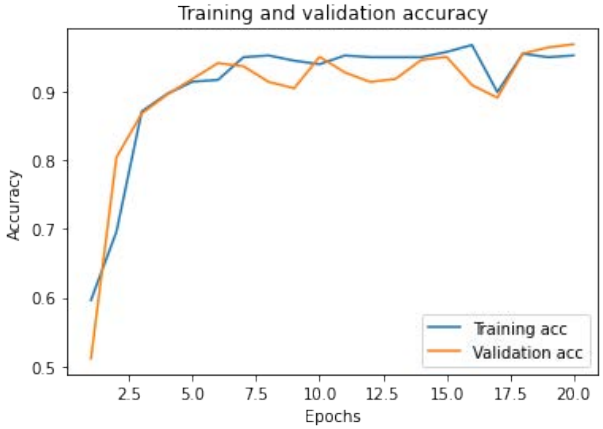
In this chapter, the primary objective is to explore how CNN, transfer learning, and the combination of transfer learning and ML can be used to analyze the oil sands drill core images and how they fit in this domain. We investigated three approaches based on transfer learning on VGG-16 pre-trained CNN model, fine-tuning the VGG-16 model, and a combination of transfer learning on VGG-16 model and traditional ML models (random forest and decision tree) for the classification of drill core image data based on two types of facies. To



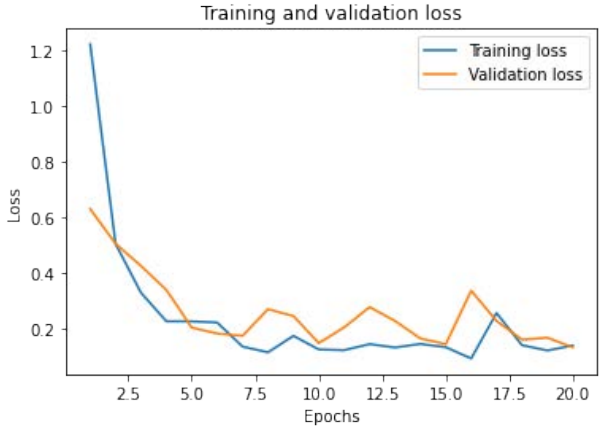
(a) Average training and validation accuracy in K-fold cross validation (fold 1).



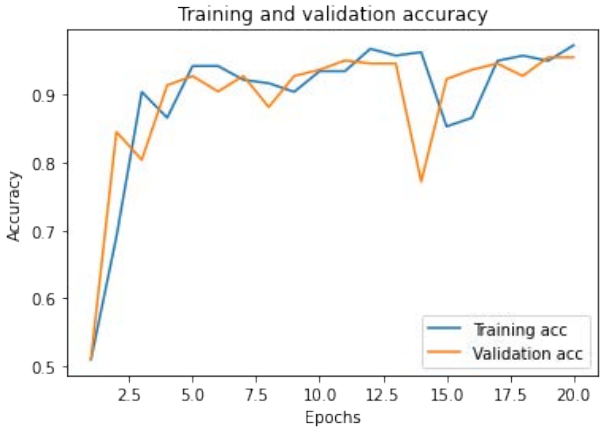
(b) Average training and validation loss in K-fold cross validation (fold 1).



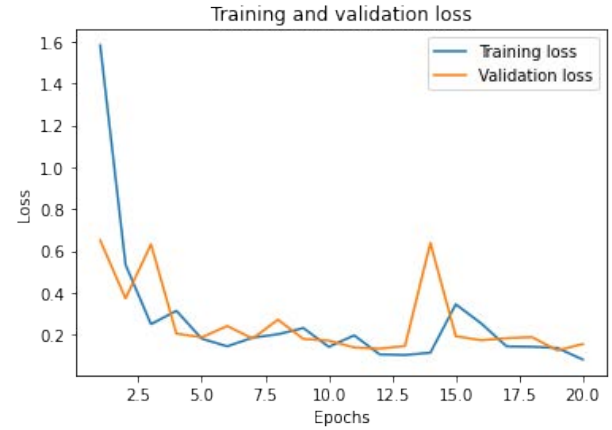
(c) Average training and validation accuracy in K-fold cross validation (fold 2).



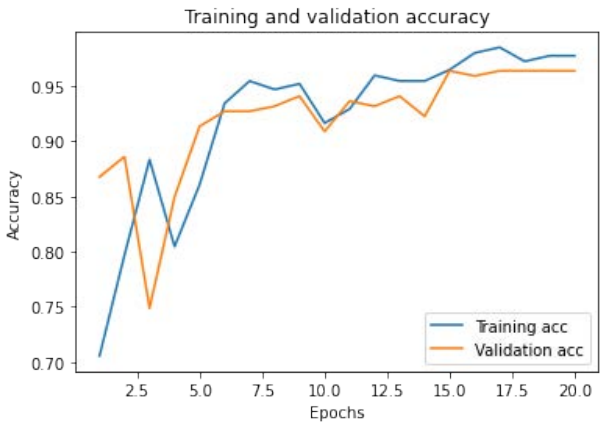
(d) Average training and validation loss in K-fold cross validation (fold 2).



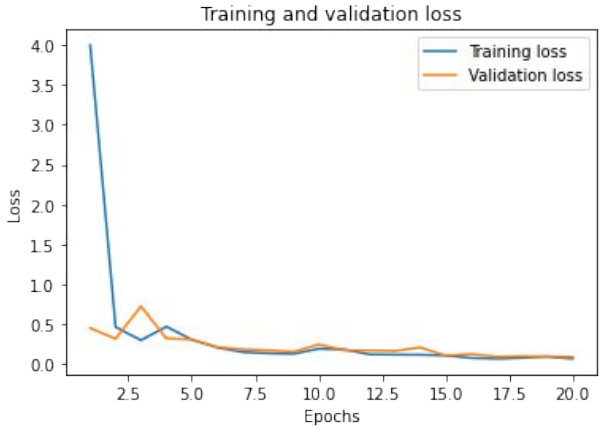
(e) Average training and validation accuracy in K-fold cross validation (fold 3).



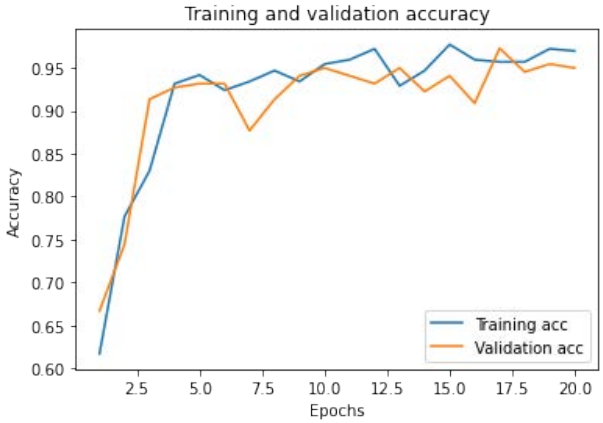
(f) Average training and validation loss in K-fold cross validation (fold 3).



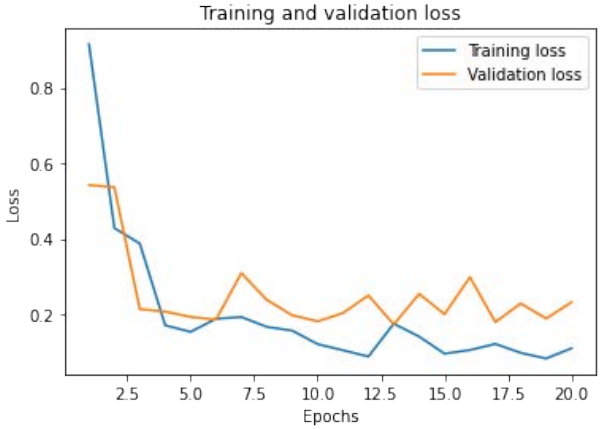
(g) Average training and validation accuracy in K-fold cross validation (fold 4).



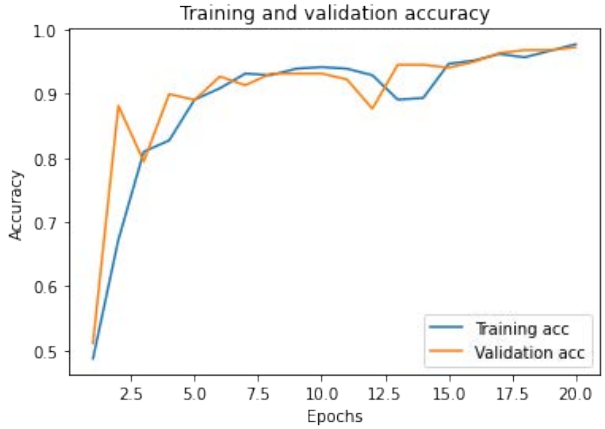
(h) Average training and validation loss in K-fold cross validation (fold 4).



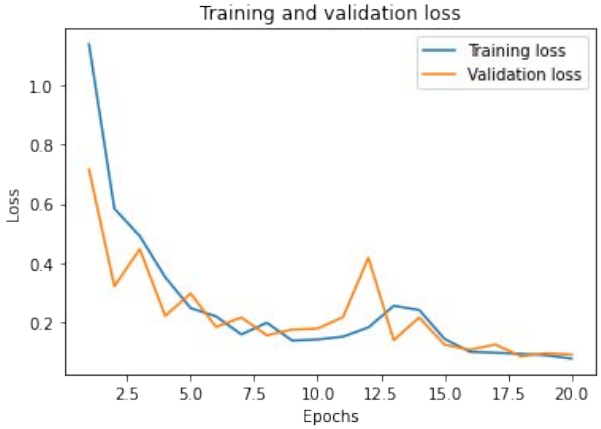
(i) Average training and validation accuracy in K-fold cross validation (fold 5).



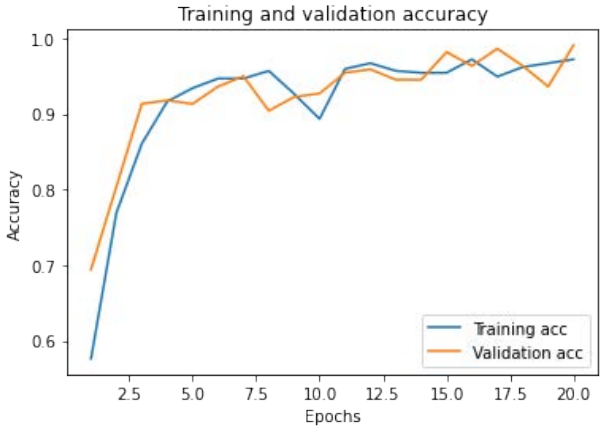
(j) Average training and validation loss in K-fold cross validation (fold 5).



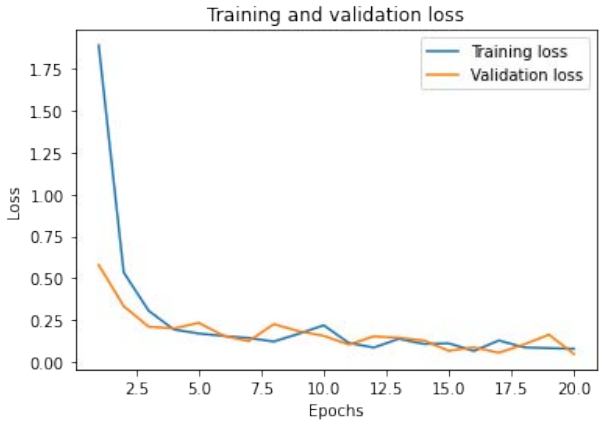
(k) Average training and validation accuracy in K-fold cross validation (fold 6).



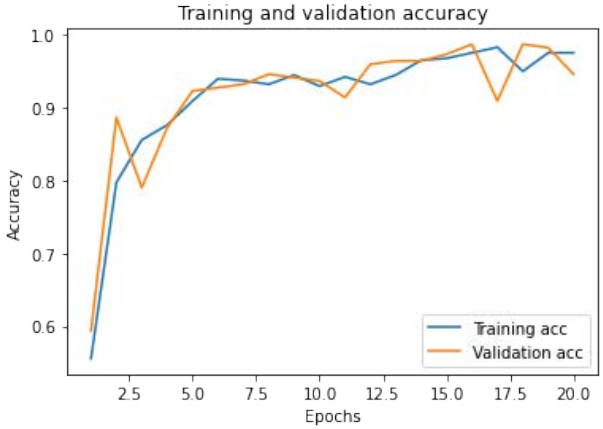
(l) Average training and validation loss in K-fold cross validation (fold 6).



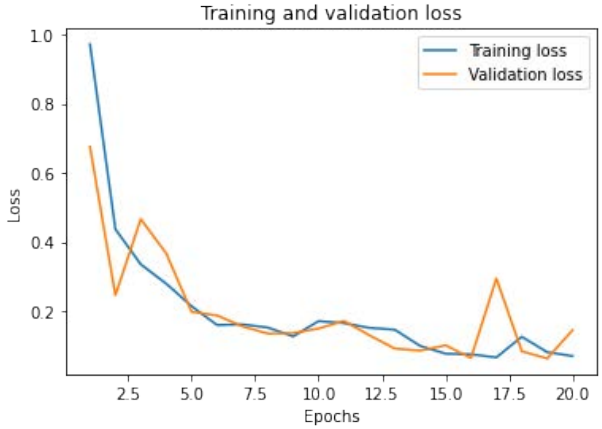
(m) Average training and validation accuracy in K-fold cross validation (fold 7).



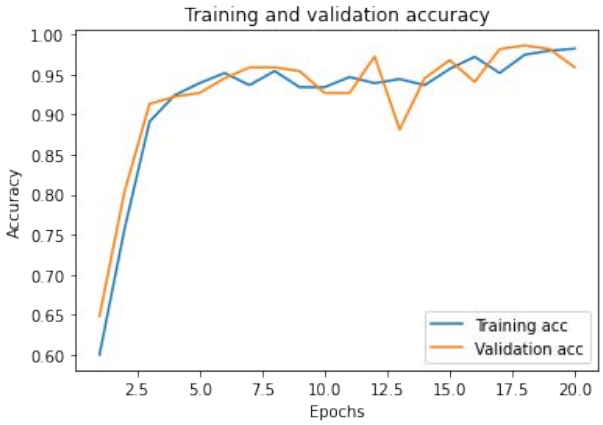
(n) Average training and validation loss in K-fold cross validation (fold 7).



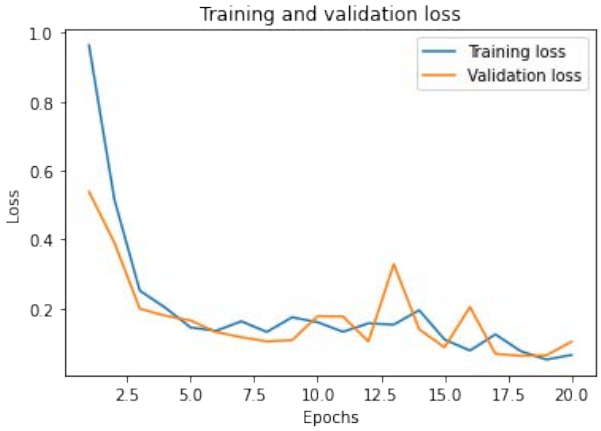
(o) Average training and validation accuracy in K-fold cross validation (fold 8).



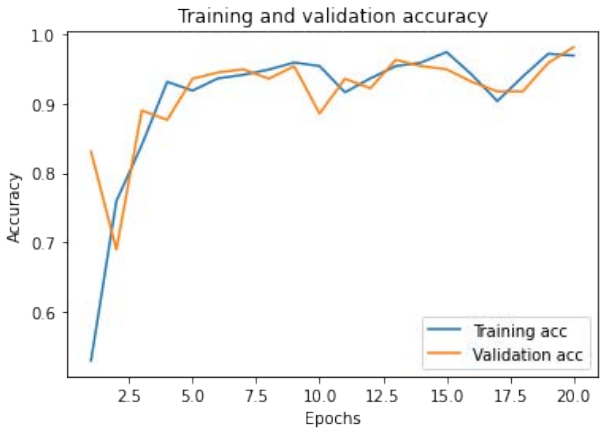
(p) Average training and validation loss in K-fold cross validation (fold 8).



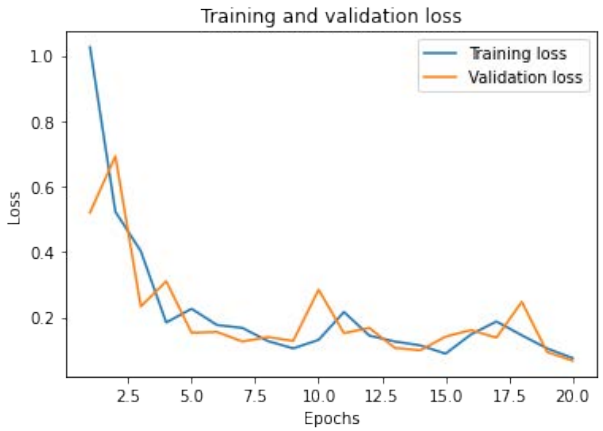
(q) Average training and validation accuracy in K-fold cross validation (fold 9).



(r) Average training and validation loss in K-fold cross validation (fold 9).



(s) Average training and validation accuracy in K-fold cross validation (fold 10).



(t) Average training and validation loss in K-fold cross validation (fold 10).

Figure 4.12: Average learning curves of fine-tuned VGG-16 with 10-fold cross validation.

evaluate the performance of these different models, we described different types of classification model evaluation metrics such as accuracy, recall, precision, f1-score, ROC curve, AUC score, confusion matrix, and average model learning curve. Based on these evaluation metrics, we analyzed how the explored methods perform for the drill core image classification. The experimental results support that the explored methods are convenient for implementing drill core image analysis tasks. Based on the outcome, we achieve clear insights and motivations to explore further these approaches for estimating the MGS from oil sands drill core photos. Though transfer learning, ML, and DL approaches have been implemented for core image analysis in the previous research [91], to the best of our knowledge, transfer learning on VGG-16 and the combination of transfer learning and traditional ML approaches have never been explored before for the classification and to predict MGS, permeability, or PSD for the images that are collected from the oil sands drill core where most of the grains are obscured by the black bitumen. In the following chapter, we describe how the explored methods described in this chapter can be employed for the MGS estimation and we describe the overall performance of the models by observing different performance metrics.

Chapter 5

Estimating Mean Grain Size From Core Photos

In Chapter 4, we investigated how different forms of transfer learning on pre-trained CNN models and traditional ML algorithms work in drill core image analysis such as classifying core photos based on facies. In this chapter, we investigate how the approaches explored in Chapter 4 work in estimating MGS for core photos. MGS provides important information about the permeability and PSD of a core sample. As PSD is an essential parameter in reservoir characterization, and permeability can also be predicted by determining PSD, successfully estimating MGS using ML and DL techniques would be useful for geologists. In the following sections of this chapter, we first describe how PSD can be calculated using different laboratory-oriented methods, followed by how the MGS is derived from PSD to train the explored ML and CNN models. Then the experimental results after training the explored models on the drill core dataset are described and analyzed based on performance evaluation metrics. Finally, at the end of this chapter, we discuss the challenges and potential research opportunities of image-based ML and DL methods in the Geoscience domain.

5.1 Particle Size Distribution

As discussed in Chapter 4 (Subsection 4.2.1), Suncor Energy (our industry partner in this research) collected the core samples and recorded different information about the samples based on experimental laboratory results. Moreover, the photos of the core samples associated with the lab tests were also taken. Among different experiments, one of these was to calculate the PSD from core samples. The PSD of geomaterials is correlated and interrelated with various physical properties and mechanical behavior. It is used in the estimation of hydraulic properties [131], evaluation of the degree of crushing [132], estimation of water retention curves [133], soil classification [134], and determination of desertification potential [135], among others. In a given sample, there exist various sizes of particles. Some particles are of very small size, and some particles have a large size. Using the PSD, we can quantify and understand the percentage of various particles present in the whole sample. There are different ways of determining the PSD or the percentage of particles present in a sample. In the following sections, we describe two laboratory-oriented methods of grain size analysis.

5.1.1 Method 1: Sieve Analysis

Sieve analysis is a technique used to determine the PSD [49]. In grain size analysis workflows, sieve analysis is used for coarse material of particle size greater than 75 μm. Different sizes of sieves are used in sieve analysis for determining the PSD. Here each sieve has square-shaped openings of a specific size. The sieve separates the larger particles from the smaller ones, distributing the soil sample in two quantities. The grains with diameters larger than the size of the openings are retained by the sieve, while smaller diameter grains pass through the sieve. As shown in Figure 5.1, a sieve shaker is used to vibrate the sieve stack for a specific period of time. Vibration allows irregularly shaped particles to reorient as they fall through the sieves. Additionally, agitation of the sieves serves to break apart weak agglomerates,

Figure 5.1: Sieve shaker equipment and wire cloth sieves
(Sieve analysis. n.d. [accessed: 2021 July 10].
https://pharmahub.org/app/site/collections/excipients/testmethods/Sieve Analysis.pdf).

allowing for a more reliable measurement of the PSD. The PSD serves as an indication of flowability. Samples with a broad size distribution tend to be poorer flowing than those with a narrow size distribution. In sieve analysis, results are reported as the differential weight percent retained on each sieve as well as the cumulative weight percent less than the sieve size. Following are the procedure and example results obtained by sieve analysis.

Procedure

1. Record the mass of the sample to be used.

2. Stack sieves from smallest to largest, starting at the bottom, with the pan below the smallest sieve.

3. Add the sample to the top sieve.

4. Tighten the equipment to ensure that the sieve stack is held firmly in the shaker assembly.

5. Set the sieve shaker to vibrate for 15 minutes.

Results

1. Carefully weigh the mass retained on each sieve and in the pan.

2. Organize the results in a table that provides columns for the retained sieve number, size, and mass fraction. Here the mass fraction is defined as the mass retained on a particular sieve divided by the initial sample mass.

5.1.2 Method 2: Laser Diffraction System

Laser diffraction system (LDS) utilizes the diffraction patterns of a laser beam that pass through any object having a size ranging from nanometers to millimeters and measure geometrical dimensions of a particle [136]. It captures PSD information by measuring scattering intensity as a function of the scattering angle, wavelength, and polarization of light based on application scattering models. LDS does not require any calibration and it offers several advantages over the traditional methods of grain size analysis that include ease of use, fast operation, broad dynamic size range. Therefore, over the past two decades, LDS has replaced conventional methods, such as sieving and sedimentation to size particles smaller than a few millimeters.

The process of measuring the PSD using LDS begins with generating a monochromatic beam by a light source that passes through a series of optical components that results in converting the raw beam into an expanded, collimated beam that illuminates particles in the scattering volume. The particles scatter light generates unique angular scattering patterns that are transformed into a spatial intensity pattern. A photodetector array then detects this intensity pattern. A photocurrent is subsequently processed and digitized, followed by

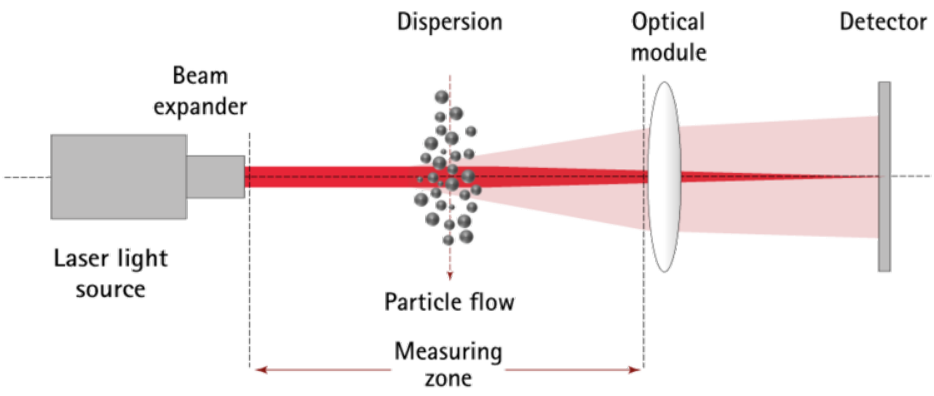


Figure 5.2: The basic optical system of a laser diffraction particle size analyzer (Laser diffraction. n.d. [accessed: 2021 July 10]. https://www.sympatec.com/en/particle-measurement/sensors/laser-diffraction/).

creating an intensity flux pattern that is then converted in a PSD. The basic optical system of a laser diffraction particle size analyzer is shown in Figure 5.2.

After comparing the results achieved by both LDS and sieve analysis, it is found that LDS makes fast calculations that are easier to recreate after a one-time analysis and it does not require a large sample size. Furthermore, LDS results in having better precision than sieve analysis and any other methods for particle measurement.

After determining the PSD using the sieve analysis or LDS, MGS can be calculated from PSD sample as follows:

$$\bar{x}_a = \frac{\sum f m_m}{100} \tag{5.1}$$

where for a PSD sample, $\bar{x}_a$ is the MGS for each sample, f is the frequency in per cent, and m is the mid-point of each class interval in metric (m_m) [47].

The oil sands drill core PSD data collected by Suncor Energy contains the PSD determined using the sieve analysis and LDS for each core sample. Each sample has a well ID indicating the wells the samples are collected from, the length of the sample, and the interval points of the core samples. For each core slab photo, the sample interval points are also marked. Therefore, to prepare the images for the use of ML and CNN models, the original images are cropped based on the sample interval points, and for each cropped core sample

image, we then have the PSD. To calculate the MGS from PSD, the cumulative probability distribution for each sample is first calculated. A cumulative distribution derived from sieve analysis and LDS data consists of plotting the cumulative mass percentage finer against the midpoint of each size interval. The cumulative mass percentage finer is the sum of all mass percentages in size ranges smaller than and including the current size range. As the cumulative distribution is obtained, we derive the probability distribution from it, then calculate the MGS for the sample using Equation 5.1. Therefore, instead of having the whole PSD, we now have the MGS for each sample along with the sample photo. Furthermore, we train the explored models with this dataset so that the models estimate the MGS using the core photos.

5.2 Data Preprocessing

Data preprocessing is a fundamental step in any ML and DL pipeline. The quality of the data and the useful information derived from the data directly affect the learning ability of a model, and it can manipulate the overall performance of a model. Therefore, it is essential to preprocess the data before feeding it into the model. Since we are dealing with image-based ML and computer vision techniques, the data are images. Before training an ML or CNN model with images, different types of image pre-processing can enhance the performance of the models. In the following, a few image-preprocessing techniques that are implemented before training the models for estimating MGS from core photos are described.

One important constraint in some ML algorithms, such as CNN, is the need to resize the images in the dataset to a unified dimension. It implies that the images must be preprocessed and scaled to have identical widths and heights before passing them to the learning algorithm. Conventionally the VGG-16 pre-trained models are trained on images that have the dimension of 224 X 224 X 3. Therefore, to perform classification or prediction using the pre-trained VGG-16, input image data has to be resized to the VGG-16 specified dimension.

However, different image dimensions can also be used to utilize a pre-trained network for transfer learning or fine-tuning some layers of the pre-trained model. There are two common reasons to use different image dimensions. One of the reasons is that if the target problem image dimensions are considerably smaller than what the CNN was trained on, increasing the size introduces too many artifacts and dramatically hurts the loss or the accuracy of the model. Another reason is when the target problem images are high resolution and contain small objects that are hard to detect. Therefore, resizing to the original input dimensions of the CNN may reduce the accuracy. Finally, keeping the image dimensions small reduces the number of parameters in a CNN model resulting in faster training and reduced memory consumption. Considering different scenarios, therefore, updating the shape of the input data before performing transfer learning results in superior model performance.

One of the most important and commonly used image preprocessing techniques is histogram equalization (HE) to improve contrast in images. A histogram is a graphical representation of the intensity of distribution of an image. It represents the number of pixels for each intensity value considered. HE is a way of stretching the histogram to include all ranges in the image histogram. HE increases the global contrast of many images, especially when close contrast values represent the usable data of the image. Through this adjustment, the intensities can be better distributed on the histogram, thus allowing for areas of lower local contrast to gain a higher contrast. HE accomplishes this by effectively spreading out the most frequent intensity values. The method is useful in images with backgrounds and foregrounds that are both bright or dark. A color histogram of an image represents the number of pixels in each type of color component. HE cannot be applied separately to the Red, Green, and Blue components of the image as it leads to dramatic changes in the image's color balance. However, if the image is first converted to HSI color space, then the algorithm can be applied to the luminance or value channel without resulting in changes to the hue and saturation of the image.

As in HE, it increases the global contrast that results in creating the image either too

(a) Core sample 1 before implementing CLAHE.

(b) Core sample 1 after implementing CLAHE.

(c) Core sample 2 before implementing CLAHE.

(d) Core sample 2 after implementing CLAHE.

Figure 5.3: Implementation of CLAHE on core photos.

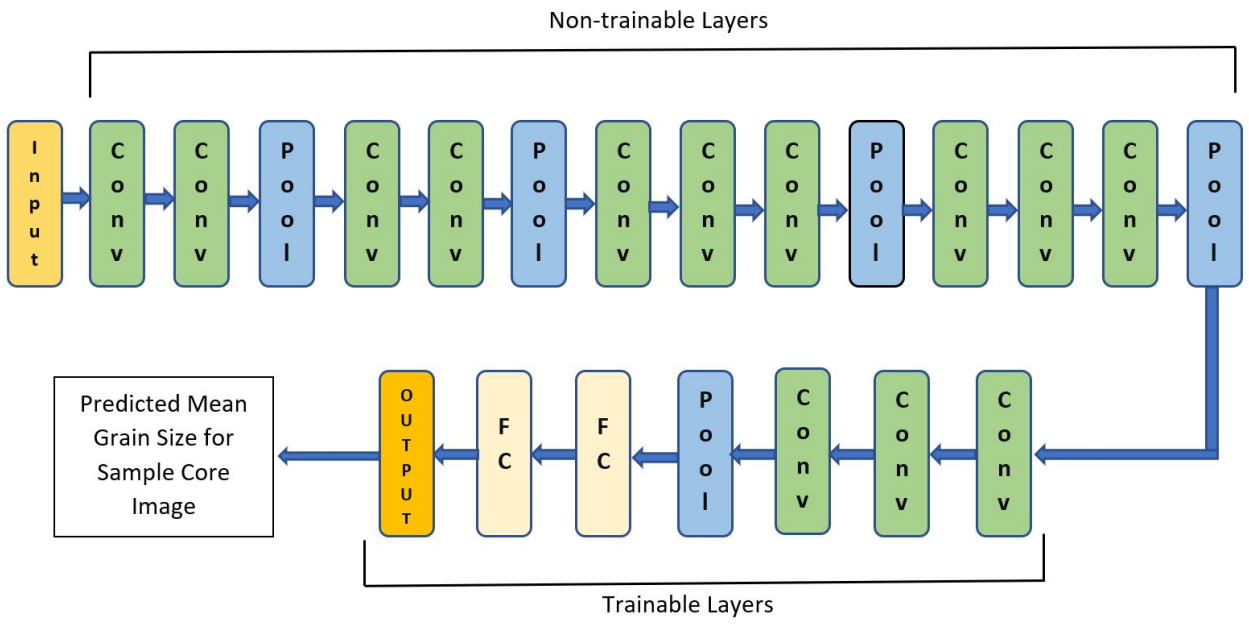


Figure 5.4: Block diagram of the explored fine-tuned VGG-16 regression model for estimating MGS from core photos. The last convolutional layer block of the VGG-16 model is trained along with the top layers associated with the prediction task. Rest of the convolutional layer blocks are kept frozen so that the pre-trained weights remain non-trainable.

dark or too bright. Contrast limited adaptive histogram equalization (CLAHE) solves this limitation of HE by operating on small regions in the image rather than the entire image [130]. CLAHE is a variant of adaptive histogram equalization (AHE). In AHE, it divides the image into small regions called tiles. Within each tile, the histogram is then equalized. The limitation here is that, if the image has a lot of noise, it gets amplified during this process. However, CLAHE prevents this by limiting the amplification. As the CLAHE operates on each tile, the neighboring tiles are combined using linear interpolation to remove any artificial boundaries. Unlike HE, CLAHE can also be applied to color images. Since CLAHE performs adaptive histogram equalization by limiting contrast and it enhances the definitions of edges in each region of an image, we utilize this technique for the drill core image equalization where the grains are rarely visible because of the dark bitumen. An example of core photos before and after implementing the CLAHE technique is shown in Figure 5.3. Note that,

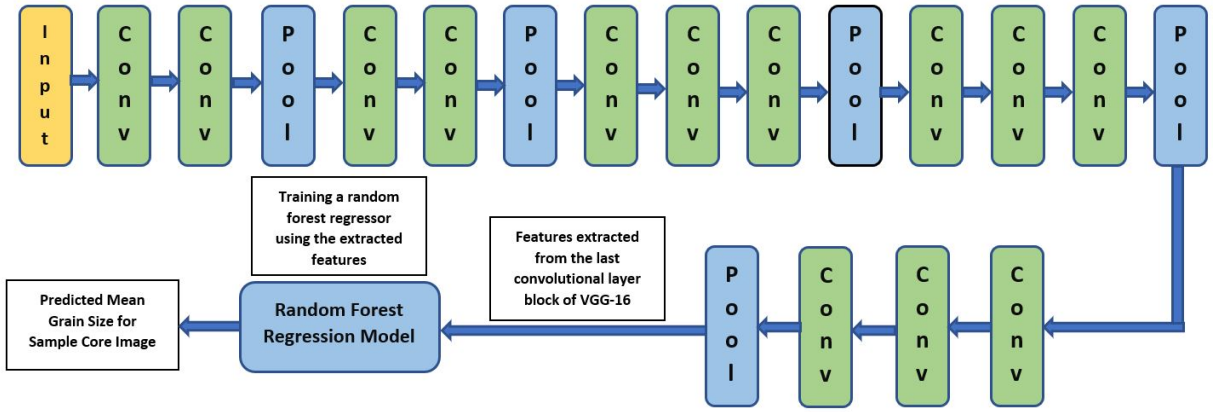


Figure 5.5: Block diagram of the combination of VGG-16 and random forest regression model for estimating MGS from core photos. A random forest regression model is trained with the features extracted from the last convolutional layer block of VGG-16.

after implementing the CLAHE technique, the image quality is improved, and both the dark and light segments of the images are balanced.

5.3 Training the Models

In Chapter 4, we described three approaches explored for the core image classification where we achieved good classification performances by the models. Therefore, we explore the same models to predict the MGS from the core photos. Figure 5.4 shows the block diagram of the explored fine-tuned VGG-16 model for estimating MGS from core photos. Moreover, the block diagram of the combination of VGG-16 and random forest regression model is shown in Figure 5.5.

Along with the previously described three approaches, we also explore the traditional ML-based technique for MGS prediction from core photos. Unlike DL, traditional ML models require hand-engineered features to be extracted from the data which is generally done by the domain experts. Based on the hand-engineered features, the ML models are trained according to the specific task. However, we explore the implementation of the traditional ML approach

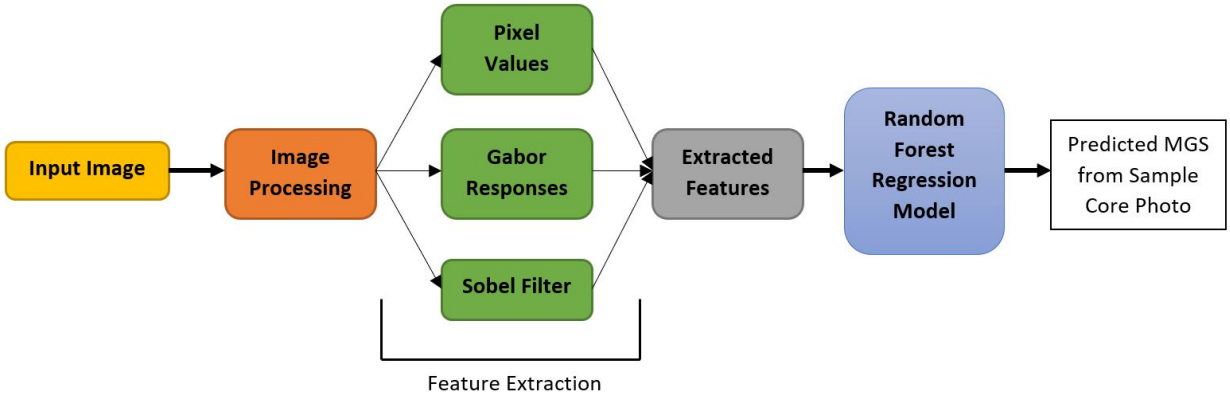


Figure 5.6: Block diagram of training a random forest regression model using the extracted core image features.

for core image analysis by extracting the features without the supervision of a domain expert. Here, we train a random forest regression model with different features extracted from the core photos considering the pixel values, Gabor filters, and Sobel filters. For the first type of feature, we use the original pixel values of the images. The second type of feature used in this workflow is the responses from the Gabor filters. Gabor filters are orientation-sensitive filters, used for texture analysis [137]. Frequency and orientation representations of the Gabor filter are similar to those of the human visual system. Different Gabor features are generated by manipulating different parameters in the Gabor filter function [138]. Finally, another feature that we use along with the pixel values and Gabor filter responses is the Sobel filter. The Sobel operator or Sobel filter is used in image processing, particularly within the edge detection algorithms that create images emphasizing edges [139]. Overall, in this explored method, a total of 5,435,392 features are extracted from the training dataset that is then used to train the random forest regression model. The block diagram of training a random forest regression model with the features extracted using the techniques mentioned above is shown in Figure 5.6. For both the traditional ML approach and for the combination of VGG-16 and random forest approach, the parameters and hyper-parameters for the random forest regression model are selected using Randomized Search CV [125] that performs a randomized search on hyperparameters. Randomized Search CV is very useful when there are many

Criteria	Classification Models	Regression Models
Evaluation metric	Accuracy	Mean absolute error
Activation	Sigmoid	Linear
Loss	Binary cross-entropy	Mean squared error
Model output	Core facies ('F1' or 'not F1')	Mean grain size

Table 5.1: Summary of modifications between the classification models and the regression models.

parameters to try and the training time is very long. In contrast to Grid Search CV [126], not all parameter values are tried out, rather a fixed number of parameter settings is sampled from the specified distributions. Randomized Search CV selects the best hyperparameters for any classification or regression model and it is computationally faster than Grid Search CV. Overall, to estimate the MGS from core photos, we explore four methods and compare the results to see how each of them performs.

Unlike a classification task, in regression-based ML and DL models, the activation functions and model evaluation metrics are different. For the MGS estimation task, we utilize linear activation, mean squared error (MSE) to compute model loss, and for the evaluation metric, we use mean absolute error (MAE). Modifications between the explored classification models described in Chapter 4 and the regression models are presented in Table 5.1. In the following section, we discuss MSE and MAE. Before passing the images to train the models, like the classification task, we re-scale the pixel values of the images into a unified range. Therefore, the previous pixel values ranging from 0 to 255 are converted into values ranging from 0 to 1 which is preferred for any ML and CNN model.

After that, as we have all the images and corresponding MGS, we split the entire data into ten equal folds for 10-fold cross-validation as described in Chapter 4. We also reserve 20% from the available data only to test the explored models. These reserved data are not used for training the model. Testing the models with unseen data provides a better understanding of the overall performance of the models. Then we train the models for an upward bound of 100 epochs with 64 batch size, meaning that the data is passed through

the network 100 times. In terms of computing resources, we utilize the same configuration described in Chapter 4 (Subsection 4.2.3).

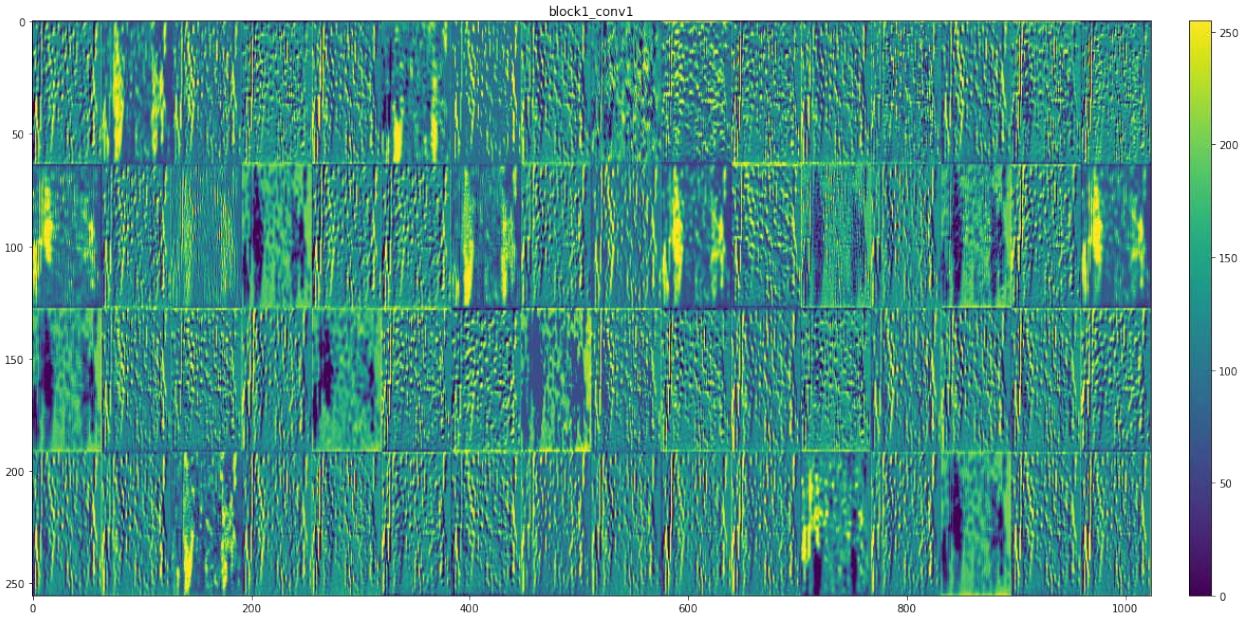
5.3.1 Visualizing the Intermediate Activations of VGG-16

Intermediate activations are helpful to understand how the convolutional layers transform the input. The visualization of intermediate activations depicts the output feature maps computed after various convolution and pooling layers in a network. For input, the output of a layer is often called its activation or the output of the activation function. This workflow gives a view into how the input is decomposed into different filters learned by the network. Each (R,G, B) channel encodes relatively independent features, so the proper way to visualize these feature maps is by independently plotting the contents of every channel as a 2D image.

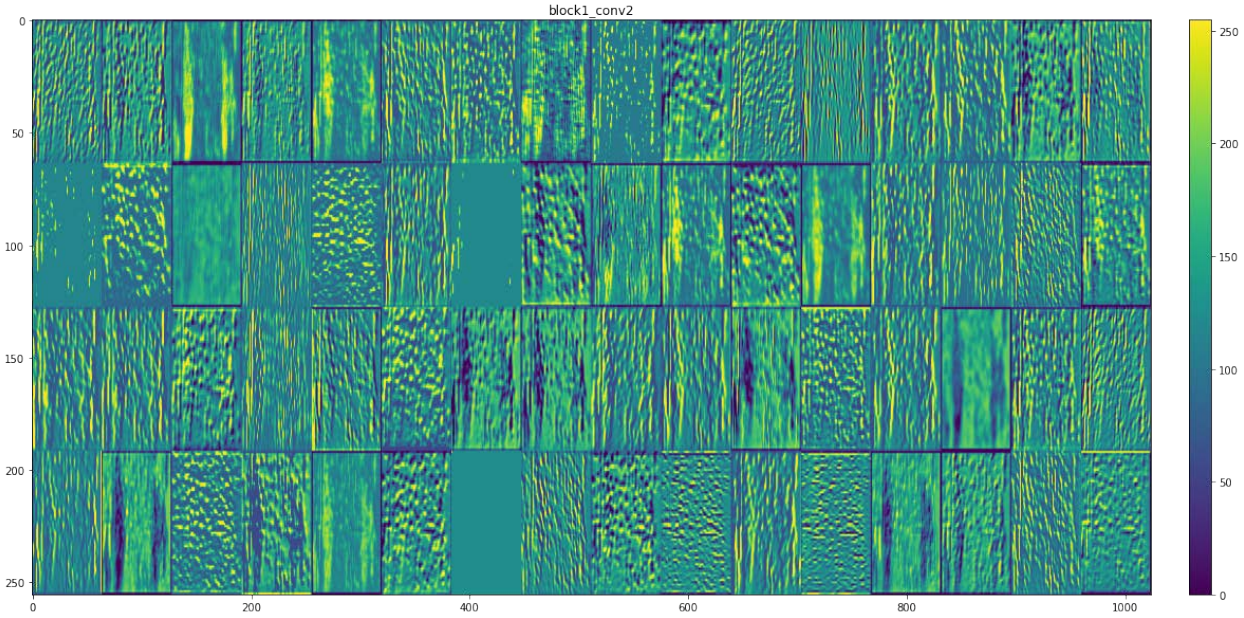
Our VGG-16 model has 13 convolutional layers. In every convolutional layer block, the model has a max-pooling layer. The initial convolutional layers generally capture everything from the image. However, as we go deeper into the network, the model tries to concentrate on more abstract features from the images that are crucial for the prediction. Figure 5.8 shows the activations of the first convolutional layer block of VGG-16 on an input core image shown in Figure 5.7. Figure 5.7 shows a sample image of oil sand drill core, where the black regions indicate the sands. Sands have enough porosity to be saturated with the bitumen which make them black. On the other hand, the brown regions contain mud or clay.

Figure 5.9 shows the activations of the last convolutional layer block of VGG-16 on the same input image. According to the model architecture of VGG-16 (see Table 4.2), the first and second convolutional layers have output shapes of 64 X 64 X 64, meaning that these

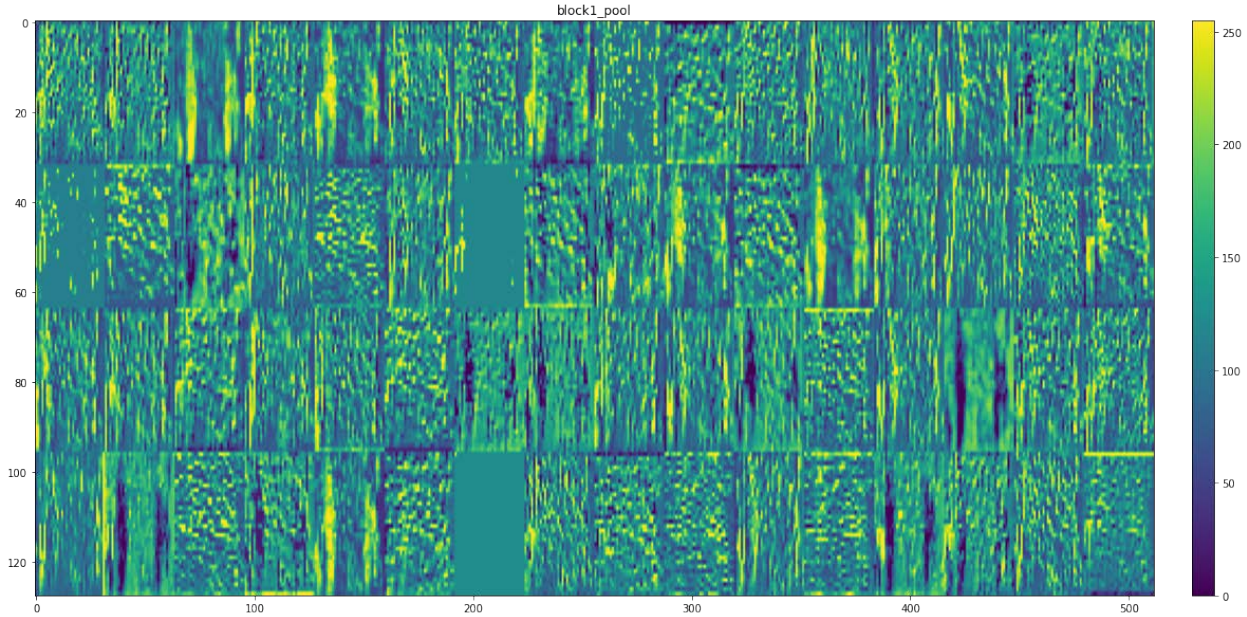
Figure 5.7: A sample input image



(a) Activations of the first convolutional layer of the first convolutional layer block.



(b) Activations of the second convolutional layer of the first convolutional layer block.

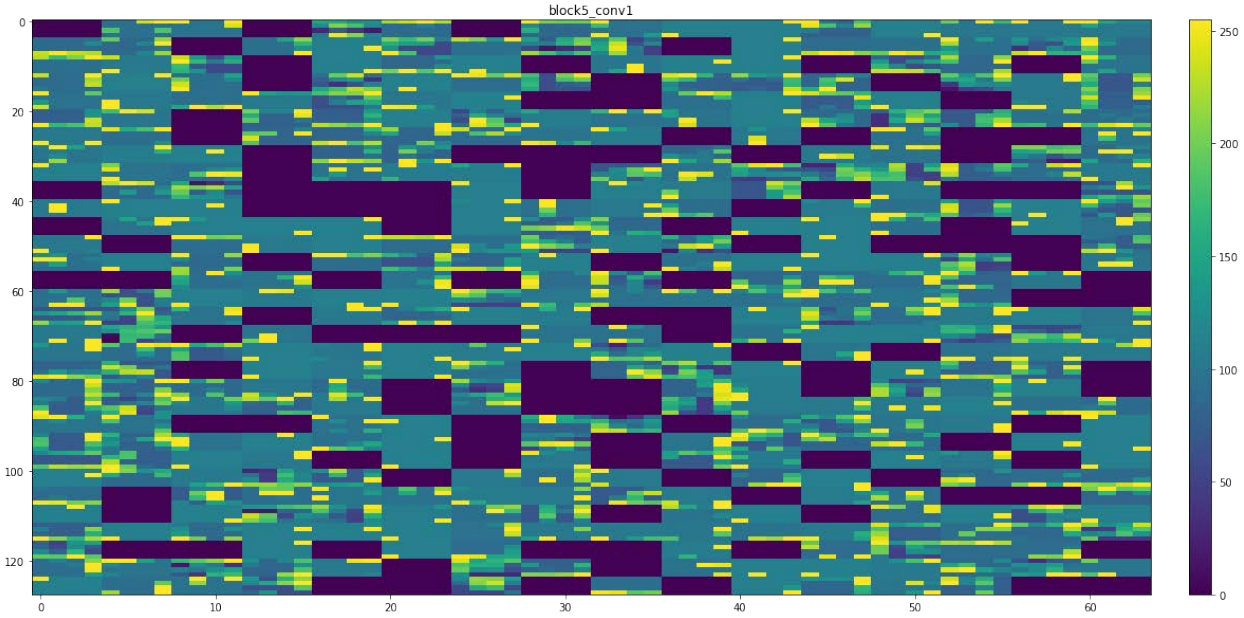


(c) Activations of the max-pooling layer of the first convolutional layer block.

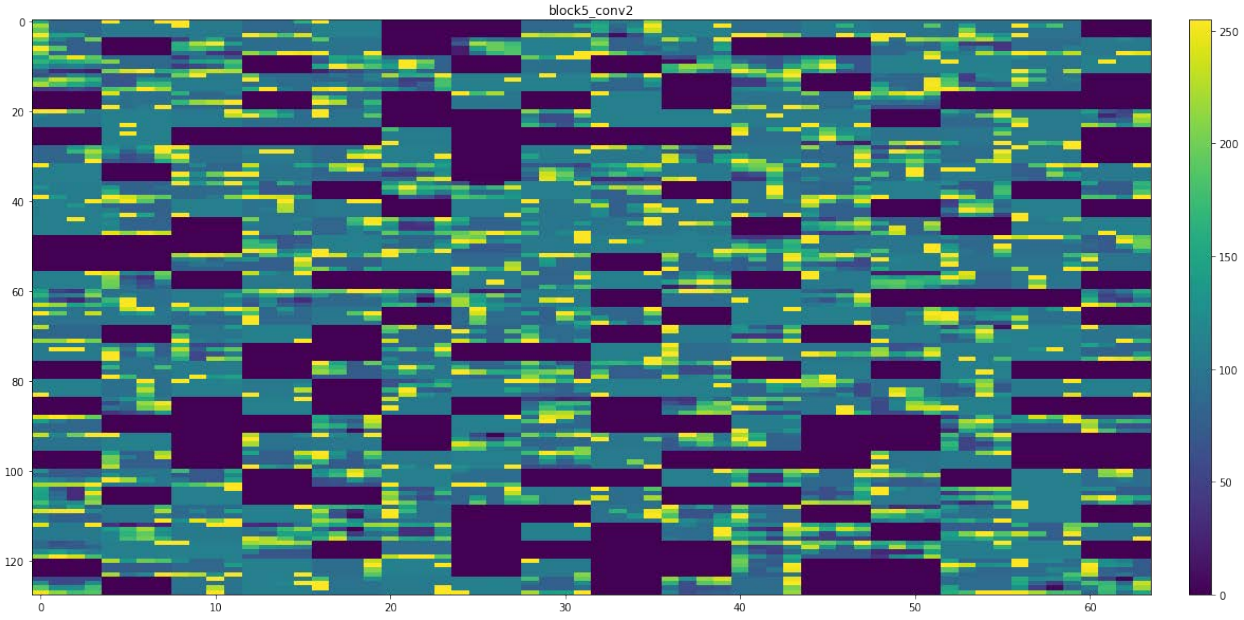
Figure 5.8: Activation of the first convolutional layer block of VGG-16.

layers have 64 filters that are applied on each channel of the input image with the dimension of 64 X 64. The max-pooling layer of the first convolutional layer block has an output shape of 32 X 32 X 64, meaning that 64 filters are applied on the output of the previous layer and an output with a reduced dimension is produced.

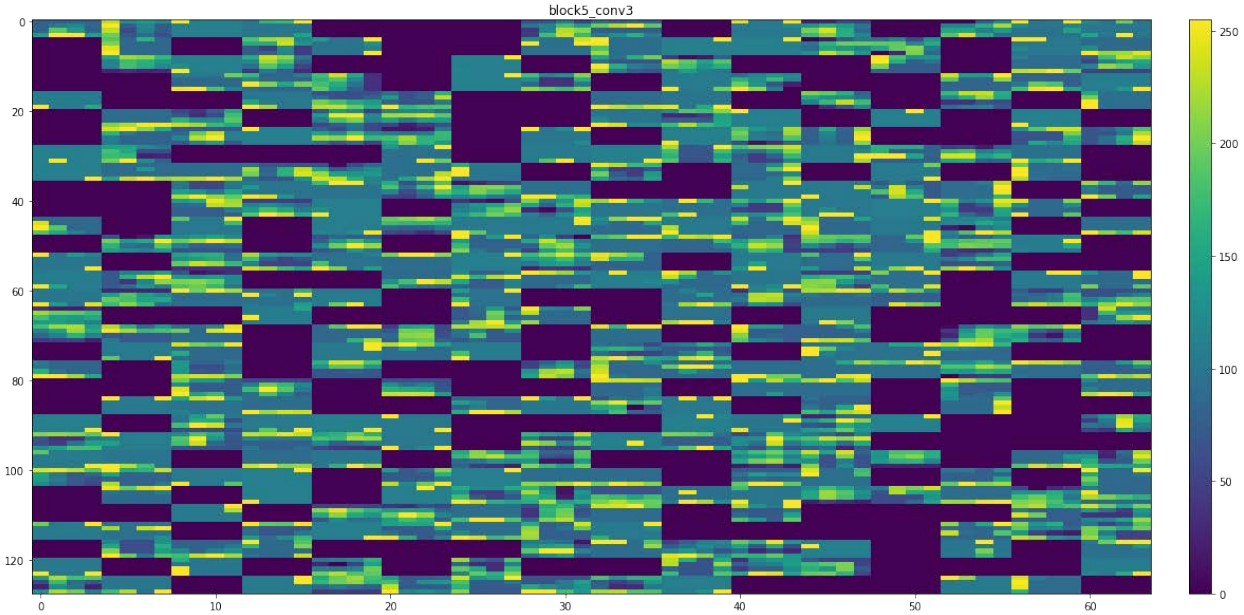
Note, the first convolutional layers do not skip any information from the image; some of the filters are capturing the vertical edges, while other filters are capturing horizontal edges; some filters are distinguishing between the dark regions (sands) and the lighter (e.g., mud, clay) regions from the core photo. Note that, in the first convolutional layer block, all the filters are activated and are not left blank. However, as we go deeper into the network, there are several filters that are not activated and are left blank (see Figure 5.9), meaning that, these filters do not have any new data to learn new information. Therefore, some filters there are not activating at all as there is nothing more to learn at that point. For these cases the outputs of the activation are shown in dark violet color. Further, it is not straightforward to visually interpret the last layer activations as they become increasingly abstract. They encode higher-level concepts that carry increasingly less information about



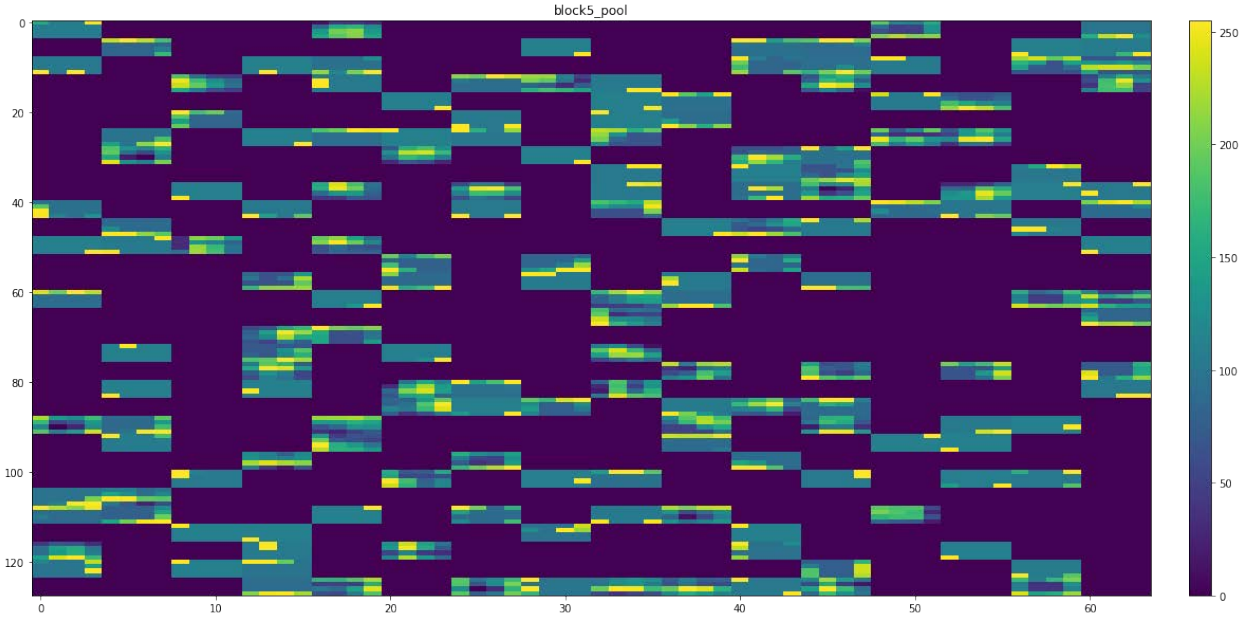
(a) Activations of the the the the the the the the first convolutional layer of the the the the the the the the last convolutional layer block.



(b) Activations of the the the the the the the second convolutional layer of the the the the the the last convolutional layer block.



(c) Activations of the the the the third convolutional layer of the the the last convolutional layer block.



(d) Activations of the the max-pooling layer of the last convolutional layer block.

Figure 5.9: Activation of the last convolutional layer block of VGG-16.

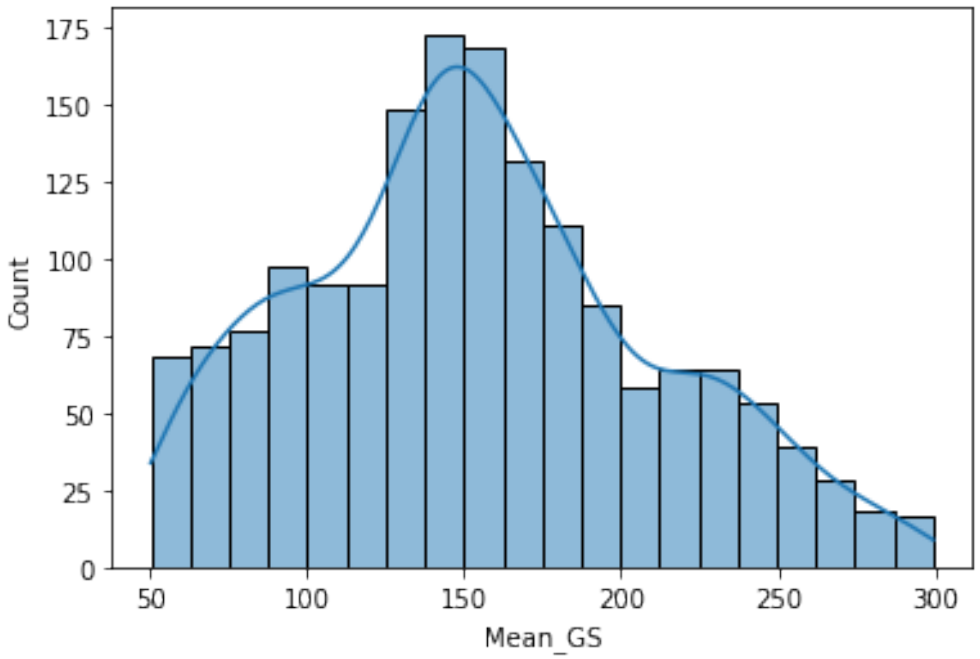


Figure 5.10: Distribution of the mean grain size in the available dataset.

the visual contents of the image and more information related to the prediction of the image.

5.4 Experimental Results

To estimate the MGS from core photos, the inputs of the models are the cropped images of core samples and the outputs are the MGS derived from the PSD. Here the number of available data is 1659. We reserve 332 core images to test the trained models. The rest of 1327 data are used for the 10-fold cross-validation. The mean and standard deviation of the available MGS values are 153.99 and 56.05 respectively. The minimum MGS value in the dataset is 113.61 and the maximum MGS is 299.47 where 75% of the values are below 188.28, 50% of the values are below 150.48, and 25% of the values are below 113.61. Figure 5.10 depicts the overall distribution of the MGS in the available dataset.

5.4.1 Regression Loss Functions and Model Evaluation Metrics

The most commonly used metrics to evaluate the performance of a classification model are accuracy, recall, precision, and f-score, as described in Chapter 4 (Subsection 4.3). However, to evaluate the performance of a regression model, different metrics are used. Some of the most frequently used metrics are mean square error (MSE), root mean square error (RMSE) and mean absolute error (MAE). These evaluation metrics are also used as loss functions.

In a typical regression-based ML model, the model produces continuous values as its predicted outputs where the primary objective is to keep these predicted values closer to the actual values or the ground truth. If the actual values are denoted as y and the predicted values are denoted as $\hat{y}$, the error is computed as follows:

$$\text{Error} = y - \hat{y} \tag{5.2}$$

This error is also called residual error. The ideal condition is that the residual error is zero, meaning that the model predicts all values correctly which is rarely possible for a regression model. Residual errors are then used to calculate different types of errors mentioned above.

Mean Absolute Error (MAE)

MAE is the sum of absolute/positive errors of all values. While computing MAE, the direction of the errors is not taken into consideration. Even if the difference between the actual value and the predicted value is negative, MAE only considers the positive values of all errors. After taking the sum of all absolute errors, the mean or average is calculated. MAE is calculated as follows:

$$\text{MAE} = \frac{\sum_{i=0}^{n} |y_i - \hat{y}_i|}{n} \tag{5.3}$$

where y_i is the actual value for a sample i, and $\hat{y}_i$ is the predicted value for that sample.

Mean Square Error (MSE)

MSE is the most commonly used regression loss function that computes the average squared distance between the actual and predicted values. Another variant of MSE is RMSE which is obtained by computing the square root of MSE. MSE and RMSE are defined by:

$$\text{MSE} = \frac{\sum_{i=0}^{n} (y_i - \hat{y}_i)^2}{n} \tag{5.4}$$

$$\text{RMSE} = \sqrt{\frac{\sum_{i=0}^{n} (y_i - \hat{y}_i)^2}{n}} \tag{5.5}$$

Comparing MAE with MSE, MAE is more robust to outliers whereas, MSE is sensitive to outliers. Since MSE squares the error, the value of the error increases a lot if it is greater than 1. If there is an outlier in the data, the value of the error becomes high, and the squared error becomes much higher. As a result, large errors have a relatively greater impact than the smaller errors on the total square error, meaning that the total square error grows as the total error is concentrated within a decreasing number of increasingly large individual errors [129]. Therefore, the model with MSE as a loss function gives more weight to outliers than a model with MAE as the loss function. A model with MSE as a loss function adjusts to minimize any case with outliers at the expense of other common examples, which reduces the model's overall performance. On the contrary, MAE is less biased for higher values and does not necessarily penalize large errors.

5.4.2 Model Comparison

For predicting MGS from core photos, we use MSE as the loss function. Along with that, as the regression-based models' performance evaluation metric, we observe the MAE at the end of each epoch as MAE determines how big of an error can be expected from the prediction on average. Since MSE is significantly larger than MAE, to make it on the same scale as MAE, RMSE is also determined.

Model	RMSE	MAE	Percentage Error
Transfer Learning on VGG-16	46.33	36.25	23.54
Fine-tuning VGG-16	49.29	38.28	24.86
VGG-16 and Random Forest Regressor	**16.89**	**11.59**	**7.53**
Traditional Machine Learning with Random Forest	51.60	40.14	26.06

Table 5.2: Summary of the performances of the explored methods on oil sands drill core dataset to estimate mean grain size.

Table 5.2 shows the overall performance of the explored models in terms of RMSE, MAE, and percentage errors. Experimental results show that similar to the facies classification results presented in Chapter 4, the combination of the pre-trained VGG-16 and the random forest regression model exhibits the best prediction performances compared to the other explored models. In this method, the MAE is 11.59 meaning that, on average the model prediction is approximately 7.53% off from the actual value. On the contrary, the performance achieved by implementing the traditional ML approach is the worst among the explored approaches where the MAE is 40.14 and the RMSE is 51.60, which is understandable as this model is trained on hand-engineered features only. Transfer learning on the VGG-16 pre-trained CNN model and fine-tuning the VGG-16 model show similar prediction performance as shown in Table 5.2. Figure 5.11 shows a scatter plot demonstrating the overall prediction results achieved by the combination of the VGG-16 pre-trained model and the random forest regression model. In this plot, the actual MGS values are given on the X-axis, and the Y-axis represents the predicted MGS by the model. The straight line represents the ideal line for the model. The scatter plot shows that the correlation between the model predicted MGS and the actual MGS is nearly linear. This aspect also supports that the explored model demonstrates reasonable prediction performance in estimating MGS from core photos.

According to the scatter plot, being the points close to the ideal line means a good prediction. As the distance between the points and the ideal line increases, the prediction has a larger error than the one that is closer to the ideal line. We can note from the scatter plot that the distance between the points and the ideal line is large for the MGS values

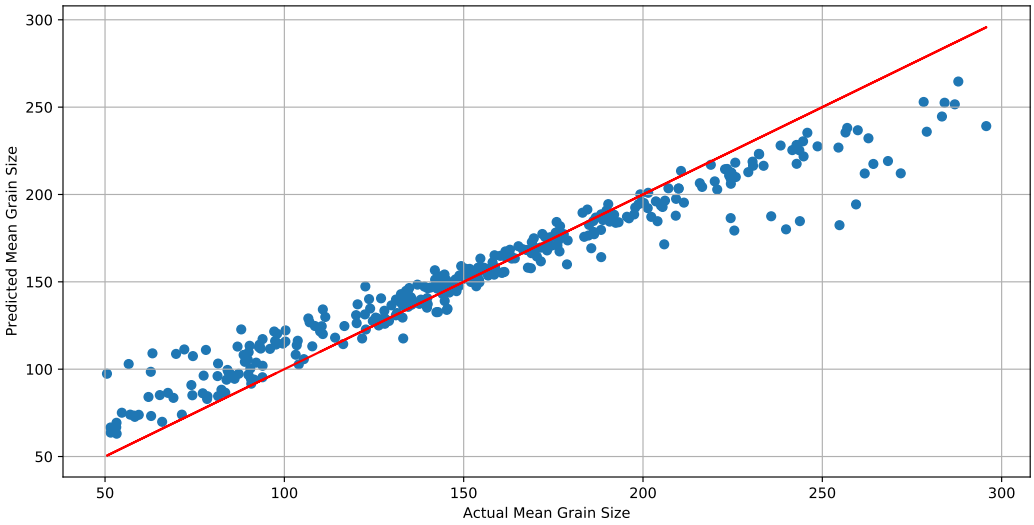


Figure 5.11: Model prediction performance of the the combination of VGG-16 and random forest regression model.

ranging from around 200 to 299. The poor prediction performance by the model for this range is understandable since we do not have a large volume of data within this range (also shown in Figure 5.10). However, the model demonstrates better performance for the rest of the data since we have a decent amount of training data there.

5.5 Discussion and Summary

In this section, we first describe the primary challenges and present some possible future directions to overcome them. Finally, we present the summary of this chapter.

5.5.1 Discussion

The primary objective of this chapter is to explore the opportunity to employ image-based ML and CNN models to estimate the MGS of oil sands drill core images. After implementing several approaches to address this problem, we obtain some valuable insights that can be utilized for further investigation in different aspects of the Geoscience domain based on the

Figure 5.12: Example of broken samples.

experimental results. Research shows that MGS is one of the most critical factors determining PSD and permeability [140]. Therefore, the proposed approach based on the combination of transfer learning and random forest can assist the geologists in calculating MGS using the photos of the previously cored wells which can also provide significant information about the PSD and permeability and save hours of laboratory experiments. After analyzing the experimental results it can be understood that, although the volume of the available labeled dataset is not significantly large that a CNN or ML model requires, the overall performances of the models are well acceptable. A few challenges that were encountered in the conducted experiments are discussed here. Furthermore, we present some reasonable suggestions that can be considered in the future extension of this work.

The primary challenge in this work is associated with data availability. In an ML or DL workflow, data play a vital role since they contain the most important information about the overall problem that is utilized to train the model. However, in oil sands drill core images, significant portions of the grains are not visible as the layer of black bitumen hides them. Therefore, even after performing the image pre-processing steps, learning the discriminating features from the data was critical for the models. Moreover, the size of the collected core samples is not of the same size. Therefore, every core sample image has a different dimension which is challenging to handle while preparing the image to train the ML or CNN models. Furthermore, in some cases, when the images were captured in the laboratory, some unnecessary contents were added to them. For example, we have some data that contain written texts on the images that hide the primary contents of the data. There are also some data where the PSD was calculated for a broken sample, as shown in Figure 5.12. In these cases, the image contains a large proportion of core slabs in between

the rock samples. We attempted to address some of these critical issues by conducting image processing. For some issues, we discarded some data from the dataset. Therefore, the total number of data got reduced as well. However, with the superior ability of the CNN model to extract valuable information from the data and with the robust prediction by the random forest model, we achieved insightful results to analyze the drill core images. Since analyzing the oil sands drill core images is critical, and to the best of our knowledge, no prior research is found to implement the proposed method to address this problem, geoscientists can benefit from it. Moreover, the experimental result obtained by our proposed method can be considered as a benchmark for future research. As the further extensions of this work, researchers can explore different model architectures, the different combinations of hyperparameters and compare the performances of the explored methods with our proposed approach. Moreover, visualization of different activations would provide insights to future researchers about fine-tuning the deeper layers of the CNN model. Furthermore, it would also provide information about understanding the best layer to extract features from and train the random forest model with these extracted features.

5.5.2 Chapter Summary

In summary, in this chapter, we investigated the opportunity to adapt the explored methods presented in Chapter 4 with the MGS estimation task. Similar to the classification task, we observed that the combination of VGG-16 and the random forest model outperformed the other explored approaches. Based on the overall investigation, we gained valuable insights suggesting the extensive scope of further research in this domain using the explored methodologies. We discussed some challenges encountered in the workflow and presented some explored solutions attempted to handle them. In the next chapter, we present a further elaborated discussion about more possible future extensions of this research.

Chapter 6

Conclusion

In this chapter, first, we present a summary of the contributions of the thesis. Then we conclude the thesis by describing a few limitations of this thesis, followed by discussing several possible future extensions and research scopes with this work.

6.1 Summary

This thesis investigates how CNN, transfer learning, and ML techniques perform in the oil sands drill core image analysis. We explored the traditional ML approach (random forest), transfer learning on pre-trained CNN model (VGG-16), the combination of random forest and VGG-16 in both the classification and regression-based tasks. Manual feature extraction techniques were implemented to apply and test traditional ML techniques. On the other hand, using transfer learning, features are automatically extracted based on the pre-trained weights. Overall, the summary of the contributions of this thesis is as follows.

1. We investigated three approaches based on transfer learning on the pre-trained VGG-16, fine-tuning a set of layers in VGG-16, and training traditional ML models (random forest, decision tree) with the features extracted from the last convolution layer block of pre-trained VGG-16 model to predict the VMI facies from the drill core photos. In

this task, the drill core photos are labeled as two facies types based on the VMI values. The core photos are labeled as either F1 or not F1. Therefore, the facies prediction task is a binary classification problem. Classification performances are evaluated based on several evaluation metrics, such as accuracy, recall, precision, f-score, ROC curve, and AUC score. Experimental results showed that the combination of VGG-16 and random forest classifier outperformed the other investigated approaches. With this approach, an accuracy of 98.87% was achieved. The recall, precision, and f-score achieved using this approach were 99.0% for all. Fine-tuning the transfer learning on VGG-16 demonstrated very similar classification performance compared to the VGG-16 and random forest. Here, the explored model showed the accuracy, recall, precision, and f-score of 97.73%, 98.0%, 98.0%, and 98.0% respectively. However, transfer learning on the pre-trained VGG-16 demonstrated slightly less than 97% accuracy, recall of 96%, and 97% for both precision and f-score. By exploring different ML and transfer learning techniques, we learned that these techniques can also be explored in comparatively complicated tasks such as estimating MGS, PSD, or the permeability of drill core photos.

2. Since we achieved good classification accuracy with the models predicting facies from drill core photos, we implemented the same techniques on the extended dataset to investigate how they perform to predict the MGS from the core photos. Here the output is MGS which is a numeric value. Therefore, the explored models are regression-based models. Along with the approaches explored in the classification task, we implemented a traditional ML technique (random forest). Since manual feature extraction needs to be performed before training a traditional ML model, we used original pixel values, responses from different combinations of Gabor filters for each image pixel, and the Sobel filters to extract the features from the core photos. Before training the random forest model, the hyper-parameters were chosen based on the Randomized Search CV to obtain the best hyper-parameters. This approach allows the model to be configured

with the optimal algorithmic settings. Before training the models, we performed image preprocessing using CLAHE. Experimental results showed that the combination of VGG-16 and the random forest method outperformed the other explored techniques with the MAE value of 11.59. We also found that only the random forest model did not perform well as the features were extracted using the manual feature extraction techniques without the supervision of any domain experts. However, the other explored methods provided reasonable prediction performance for the dataset used.

6.2 Limitations and Future Work

In this thesis, we conducted two types of experiments: performing classification of core photos based on facies and the regression-based task for predicting MGS from the core photos. In both cases, we explored different models that achieved acceptable performance. Since, to the best of our knowledge, there is no published work where transfer learning, the combination of transfer learning and traditional ML have been employed in the oil sands drill core image analysis, this thesis provides important insights and motivation for future researchers to further explore these techniques in the Geoscience research domain. In this thesis, our aim was not to produce the best solution for the facies classification or predicting the MGS from core photos. Instead, our primary objective was to explore different image-based ML and transfer learning techniques to understand whether or not these techniques are suitable to implement in the oil sands drill core image analysis. In the oil sands drill core samples, most of the grains remain hidden by the black bitumen. Therefore, it is critical for an ML or DL model to predict based on these types of core photos as the discriminating features are rarely determinable. However, by applying some image processing techniques, the explored approaches exhibited acceptable prediction performances. Therefore, our explored experimental setting can be considered as a baseline setting for future research.

However, there are a few limitations in the work presented in this thesis, where we only employed transfer learning on the pre-trained VGG-16 model. Although this pre-trained model is very simple in architecture and performs very well to extract features from different types of datasets other than the ones the model was trained on, VGG-16 contains a large number of parameters. Although most of these parameters do not need to be trained in transfer learning, the VGG-16 model is computationally slow and takes a large amount of disk space. Another limitation is that we considered the raw images as model input data in this thesis and performed very little pre-processing.

Although we aimed to understand how the models perform with the raw core images captured during the sample collection phase in the laboratories, different types of data pre-processing stages need to be performed to improve the prediction performance of the models. Moreover, the laboratory collected raw images are not high-quality images, and the images contain plenty of unnecessary information that adversely affects the feature learning from the images during the model training process.

There are several future research scopes where the different ML and DL techniques can be utilized to analyze drill core photos. Research shows that permeability has a close correlation with the grain size [141]. Therefore, as the following future extension of this thesis, different ML and DL models can be trained with different parameter settings to predict grain size using the core photos to estimate the permeability of the drill core. However, a larger database needs to be accumulated before reliable fit parameters and variability can be predicted [142]. Moreover, the explored approaches for facies classification can be extended to solve multi-class classification as well. The explored DL and ML models can be re-trained with drill core images belonging to more than two facies by setting the activation function as Softmax instead of Sigmoid in the classification layer and changing the number of neurons as the number of output classes instead of using a layer with a single neuron used for binary classification. Other than facies classification, these techniques can be used to classify drill core photos based on grain size too, such as coarse grain, fine grain, and so on.

Moreover, extensive care should be taken while collecting core images. Since data contains the most critical information in the ML workflow, capturing unnecessary information and photos should be avoided for better model performances. Training models with high-resolution images can also enhance the model performance. Therefore, to utilize the effectiveness of DL models, an extensive library of high resolution and accurately labeled image datasets can be obtained from the already cored wells. Since laboratories have the access to the already drilled core slabs, re-capturing the core photos carefully and training the models can result in better model performances. However, acquiring high-resolution images may not be feasible due to the expense and time restriction. In this scenario, if the low-quality images are the only feasible option, extensive image processing has to be performed so that the models can effectively learn from the data. Different CNN models can be explored after extending the library of images to estimate both the MGS and PSD standard deviation from the core photos.

Although MGS is the most important parameter for the PSD, accumulating standard deviation with MGS can make the determination of PSD simpler and more accurate. As the volume of the database increases, along with employing the transfer learning technique, different CNN models can be built from scratch. Here, the CNN models can be trained entirely on the core photos to learn the core images' features efficiently. Although pre-trained models are trained on millions of images and are very good at recognizing important features, training a CNN from scratch with lots of core photos can also produce accurate prediction performances.

In a nutshell, this thesis paves new ways for doing research and implementing image-based ML, transfer learning, and CNN techniques to address different challenging problems in the oil and gas industries. With more extensive research and exploration, oil and gas industries can leverage the ML and DL techniques, save a lot of time and money, and produce state-of-the-art performance from them.

Bibliography

[1] Folk, R. L., Ward, W. C. (1957). Brazos River bar [Texas]; a study in the significance of grain size parameters. *Journal of sedimentary research,* 27(1), 3-26.

[2] Friedman, G. M. (1979). Differences in size distributions of populations of particles among sands of various origins: addendum to IAS Presidential Address. *Sedimentology,* 26(6), 859-862.

[3] Bui, E. N., Mazzullo, J. M., Wilding, L. P. (1989). Using quartz grain size and shape analysis to distinguish between aeolian and fluvial deposits in the Dallol Bosso of Niger (West Africa). *Earth Surface Processes and Landforms,* 14(2), 157-166.

[4] M. Vandenbroucke and C. Largeau, "Kerogen origin, evolution and structure," *Organic Geochemistry,* vol. 38, no. 5, pp. 719–833, 2007.

[5] J. R. Fanchi, Principles of applied reservoir simulation. *Elsevier,* 2018.

[6] Chen, A., 2020. Exploration production (EP). [Online]. Available: https://www.investopedia.com/terms/e/exploration-production-company.asp. [Accessed: August 24, 2021].

[7] Atlantic Canada's Offshore Oil and Gas Industries. Offshore oil and natural gas life cycle. n.d., [Online]. Available: http://atlanticcanadaoffshore.ca/offshore-oil-gas-lifecycle/. [Accessed: August 24, 2021].

[8] The oil and gas operating cycle and activities – (treccani – petroleum encyclopaedia). n.d., [Online]. Available: http://www.oil-gasportal.com/upstream/basic-concept/?print=print. [Accessed: August 24, 2021].

[9] Alberta Energy Regulator. n.d., Oil Sands [Online]. Available: https://www.aer.ca/providing-information/by-topic/oil-sands. [Accessed: August 24, 2021].

[10] M. N. Panda and L. W. Lake, "Estimation of single-phase permeability from parameters of particle-size distribution," *AAPG bulletin,* vol. 78, no. 7, pp. 1028–1039, 1994.

[11] J.M.K.C. Donev et al. (2019). Energy Education - Oil and gas reservoir [Online]. Available: https://energyeducation.ca/encyclopedia/Oil_and_gas_reservoir. [Accessed: July 26, 2021].

[12] L. Lepisto,¨ I. Kunttu, J. Autio, and A. Visa, "Rock image classification using non-homogenous textures and spectral imaging," 2003.

[13] Lepistö, L., Kunttu, I., Autio, J. and Visa, A., 2004, April. Rock image retrieval and classification based on granularity. In Proceedings of *5th International Workshop on Image Analysis for Multimedia Interactive Services.*

[14] F.-H. Kong, "Image retrieval using both color and texture features," in *2009 International Conference on Machine Learning and Cybernetics,* vol. 4. IEEE, 2009, pp. 2228–2232.

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

[16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556,* 2014.

[17] Wikipedia contributors. (2021, July 14). Chain rule. In *Wikipedia, The Free Encyclopedia.* Retrieved 09:52, July 20, 2021, from https://en.wikipedia.org/w/index.php?title=Chain$_r uleoldid = 1033509971$

[18] J. Wan, D. Wang, S. C. H. Hoi, P. Wu, J. Zhu, Y. Zhang, and J. Li, "Deep learning for content-based image retrieval: A comprehensive study," in Proceedings of the *22nd ACM international conference on Multimedia.* ACM, 2014, pp. 157–166.

[19] J.M.K.C. Donev et al. (2019). Energy Education - Conventional vs unconventional resource [Online]. Available: https://energyeducation.ca/encyclopedia/Conventional_vs_unconventional_resource. [Accessed: August 24, 2021].

[20] L. Xie, R. Hong, B. Zhang, and Q. Tian, "Image classification and retrieval are one," in Proceedings of the *5th ACM on International Conference on Multimedia Retrieval.* Acm, 2015, pp. 3–10.

[21] S. Saha. (2018). A comprehensive guide to convolutional neural networks - the ELI5 way [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Accessed: August 24, 2021].

[22] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks,* vol. 61, pp. 85–117, 2015.

[23] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *arXiv preprint arXiv:1605.07678,* 2016.

[24] P.C. Carman, "Fluid flow through granular beds." *Transactions, Institution of Chemical Engineers, London,* 15: 150-166, 1937.

[25] P.C. Carman, "Flow of gases through porous media." *Butterworths, London,* 1956.

[26] Wikipedia contributors, "Facies — Wikipedia, the free encyclopedia," 2019, *[Online].*

[27] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein et al., "Imagenet large scale visual recognition challenge," *International journal of computer vision,* vol. 115, no. 3, pp. 211–252, 2015.

[28] Alpak, F.O., Lake, L.W. and Embid, S.M., 1999, January. Validation of a modified Carman-Kozeny equation to model two-phase relative permeabilities. In *SPE Annual Technical Conference and Exhibition. Society of Petroleum Engineers.*

[29] Andersson, T., Thurley, M. J., Carlson, J. E. (2012). A machine vision system for estimation of size distributions by weight of limestone particles. *Minerals Engineering,* 25(1), 38-46.

[30] Wipware (2018), Fragmentation and analysis software, *www.wipware.com*

[31] D. Erhan, P.-A. Manzagol, Y. Bengio, S. Bengio, and P. Vincent, "The difficulty of training deep architectures and the effect of unsupervised pretraining," in *Artificial Intelligence and Statistics.* USA: JMLR, 2009, pp. 153–160.

[32] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, "Convolutional neural networks for medical image analysis: Full training or fine tuning?" *IEEE Trans.* Med. Imag., vol. 35, no. 5, pp. 1299–1312, May 2016.

[33] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in Proc. *Adv. Neural Inf. Process.* Syst., 2014, pp. 3320–3328.

[34] Torrey, L. and Shavlik, J., 2010. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques* (pp. 242-264). IGI global.

[35] Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and computing,* 4(2), 65-85.

[36] Bai, Q. (2010). Analysis of particle swarm optimization algorithm. *Computer and information science,* 3(1), 180.

[37] Aurlien Gron. 2017. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems (1st. ed.). *O'Reilly Media, Inc.*

[38] Long, M., Cao, Y., Wang, J., Jordan, M. (2015, June). Learning transferable features with deep adaptation networks. In *International conference on machine learning* (pp. 97-105). PMLR.

[39] Khan, N.M., Abraham, N. and Hon, M., 2019. Transfer learning with intelligent training data selection for prediction of Alzheimer's disease. *IEEE Access,* 7, pp.72726-72735.

[40] Adit, D., 2019. The 9 Deep Learning Papers You Need To Know About (Understanding CNNs Part 3). Engineering at Forward. [Online]. Available: https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html

[41] Sutton, R. S., Barto, A. G. (2018). Reinforcement learning: An introduction. *MIT press.*

[42] Duguleana, M., Mogan, G. (2016). Neural networks based reinforcement learning for mobile robots obstacle avoidance. *Expert Systems with Applications,* 62, 104-115.

[43] Dodge, S., Karam, L. (2017, July). A study and comparison of human and deep learning recognition performance under visual distortions. In *2017 26th international conference on computer communication and networks (ICCCN)* (pp. 1-7). IEEE.

[44] Hossin, M., Sulaiman, M. N. (2015). A review on evaluation metrics for data classification evaluations. *International journal of data mining knowledge management process,* 5(2), 1.

[45] Araya-Polo, M., Alpak, F.O., Hunter, S., Hofmann, R. and Saxena, N., 2019. Deep learning–driven permeability estimation from 2D images. *Computational Geosciences,* pp.1-10.

[46] Timmer, E., Knudson, C. and Gingras, M., 2020. Applying Deep Learning for Identifying Bioturbation from Core Photos. *AAPG Bulletin,* (20,200,828).

[47] Blott, S.J. and Pye, K., 2001. GRADISTAT: a grain size distribution and statistics package for the analysis of unconsolidated sediments. *Earth surface processes and Landforms,* 26(11), pp.1237-1248.

[48] Suncor Energy n.d., accessed 10 July 2021, https://www.suncor.com/en-ca/

[49] Sieve analysis n.d., accessed 10 July 2021, [Online]. Available: https://pharmahub.org/app/site/collections/excipients/testmethods/Sieve Analysis.pdf

[50] Keras applications n.d., accessed 10 July 2021, https://keras.io/api/applications/

[51] Mohammad Masum. A Poor Example of Transfer Learning: Applying VGG Pre-trained model with Keras.

[52] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z., 2016. Rethinking the inception architecture for computer vision. In Proceedings of the *IEEE conference on computer vision and pattern recognition* (pp. 2818-2826).

[53] Misra, S., Li, H., He, J. (2020). Noninvasive fracture characterization based on the classification of sonic wave travel times. In *Machine Learning for Subsurface Characterization* (pp. 243-287). Gulf Professional Publishing.

[54] Kurama, V., A Review of Popular Deep Learning Architectures: ResNet, InceptionV3, and SqueezeNet. (2020). [Online]. Available: https://blog.paperspace.com/popular-deep-learning-architectures-resnet-inceptionv3-squeezenet/. [Accessed: August 24, 2021].

[55] Sik-Ho Tsang. Review: Inception-v3 — 1st Runner Up (Image Classification) in *ILSVRC 2015.*

[56] Samuel, A. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM J. Res. Dev.,* 3, 210-229.

[57] Mitchell, T. M. (1997). Machine learning.

[58] Balaji, B., Mallya, S., Genc, S., Gupta, S., Dirac, L., Khare, V., ... Karuppasamy, D. (2020, May). Deepracer: Autonomous racing platform for experimentation with sim2real reinforcement learning. In 2020 *IEEE International Conference on Robotics and Automation (ICRA)* (pp. 2746-2754). IEEE.

[59] Ayodele, T. O. (2010). Machine learning overview. *New Advances in Machine Learning,* 9-19.

[60] Medjahed, S. A., Saadi, T. A., Benyettou, A. (2013). Breast cancer diagnosis by using k-nearest neighbor with different distances and classification rules. *International Journal of Computer Applications,* 62(1).

[61] Rustam, F., Reshi, A. A., Mehmood, A., Ullah, S., On, B. W., Aslam, W., Choi, G. S. (2020). COVID-19 future forecasting using supervised machine learning models. *IEEE access,* 8, 101489-101499.

[62] Choudhary, R., Gianey, H. K. (2017, December). Comprehensive review on supervised machine learning algorithms. In 2017 *International Conference on Machine Learning and Data Science (MLDS)* (pp. 37-43). IEEE.

[63] Dreiseitl, S., Ohno-Machado, L. (2002). Logistic regression and artificial neural network classification models: a methodology review. *Journal of biomedical informatics,* 35(5-6), 352-359.

[64] Ali, J., Khan, R., Ahmad, N., Maqsood, I. (2012). Random forests and decision trees. *International Journal of Computer Science Issues (IJCSI),* 9(5), 272.

[65] Noble, W. S. (2006). What is a support vector machine?. *Nature biotechnology,* 24(12), 1565-1567.

[66] Ahmad, A., Dey, L. (2007). A k-mean clustering algorithm for mixed numeric and categorical data. *Data Knowledge Engineering,* 63(2), 503-527.

[67] Ding, C., He, X. (2004, July). K-means clustering via principal component analysis. In Proceedings of the *twenty-first international conference on Machine learning* (p. 29).

[68] Bengio, Y., Courville, A., Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8), 1798-1828.

[69] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning.* MIT Press Cambridge, 2016.

[70] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural Networks,* vol. 2, no. 1, pp. 53–58, 1989.

[71] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning,* 2010, pp. 807–814.

[72] Bari, A. S. M. (2020). Kinect-based Gait Recognition Using Deep Learning *(Master's thesis, Science).*

[73] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by backpropagating errors," *Nature,* vol. 323, no. 6088, pp. 533–536, 1986.

[74] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks,* vol. 61, pp. 85–117, 2015.

[75] Atakulreka, Akarachai, and Daricha Sutivong. "Avoiding local minima in feedforward neural networks by simultaneous learning." In *Australasian Joint Conference on Artificial Intelligence.* Springer, Berlin, Heidelberg, 2007.

[76] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *International Conference on Machine Learning,* 2013.

[77] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in Proceedings of the *IEEE International Conference on Computer Vision,* 2015, pp. 1026–1034.

[78] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv:1511.07289,* 2015.

[79] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances in Neural Information Processing Systems,* 2017, pp. 971–980.

[80] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *arXiv preprint arXiv:1605.07678,* 2016.

[81] Dunne, R. A., Campbell, N. A. (1997, June). On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function. In Proc. *8th Aust. Conf. on the Neural Networks, Melbourne (Vol. 181, p. 185).* Citeseer.

[82] Britannica, The Editors of Encyclopaedia. "Seismic survey". *Encyclopedia Britannica,* 7 Nov. 2017, https://www.britannica.com/science/seismic-survey. Accessed 13 June 2021.

[83] Global Analytical and Measuring Instruments. "Particle Size Distribution Dependent on Principle of Measurement". *SHIMADZU Excellence on Science,* https://www.shimadzu.com/an/service-support/technical-support/analysis-basics/lesson02.html. Accessed 13 June 2021.

[84] Meldahl, Paul, et al. "The chimney cube, an example of semi-automated detection of seismic objects by directive attributes and neural networks: Part I; methodology." *SEG Technical Program Expanded Abstracts 1999. Society of Exploration Geophysicists,* 1999. 931-934.

[85] West, B. P., May, S. R., Eastwood, J. E., Rossen, C. (2002). Interactive seismic facies classification using textural attributes and neural networks. *The Leading Edge,* 21(10), 1042-1049.

[86] de Matos, M. C., Yenugu, M., Angelo, S. M., Marfurt, K. J. (2011). Integrated seismic texture segmentation and cluster analysis applied to channel delineation and chert reservoir characterization. *Geophysics,* 76(5), P11-P21.

[87] Roy, A., Romero-Peláez, A. S., Kwiatkowski, T. J., Marfurt, K. J. (2014). Generative topographic mapping for seismic facies estimation of a carbonate wash, Veracruz Basin, southern Mexico. *Interpretation,* 2(1), SA31-SA47.

[88] Qi, J., Lin, T., Zhao, T., Li, F., Marfurt, K. (2016). Semisupervised multiattribute seismic facies analysis. *Interpretation,* 4(1), SB91-SB106.

[89] Hu, S., Zhao, W., Xu, Z., Zeng, H., Fu, Q., Jiang, L., ... Liu, W. (2017). Applying principal component analysis to seismic attributes for interpretation of evaporite facies: Lower Triassic Jialingjiang Formation, Sichuan Basin, China. *Interpretation,* 5(4), T461-T475.

[90] Zhao, T., Li, F., Marfurt, K. J. (2017). Constraining self-organizing map facies analysis with stratigraphy: An approach to increase the credibility in automatic seismic facies classification. Interpretation, 5(2), T163-T171.

[91] De Lima, R. P., Bonar, A., Coronado, D. D., Marfurt, K., Nicholson, C. (2019). Deep convolutional neural networks as a geological image classification tool. *Sediment.* Rec, 17, 4-9.

[92] Shoji, D., Noguchi, R., Otsuki, S., Hino, H. (2018). Classification of volcanic ash particles using a convolutional neural network and probability. *Scientific reports,* 8(1), 1-12.

[93] Civitarese, D., Szwarcman, D., Brazil, E. V. (2019, June). Stratigraphic Segmentation Using Convolutional Neural Networks. In 81st EAGE Conference and Exhibition 2019 Workshop Programme (Vol. 2019, No. 1, pp. 1-5). *European Association of Geoscientists Engineers.*

[94] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the *IEEE conference on computer vision and pattern recognition* (pp. 4510-4520).

[95] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the *IEEE conference on computer vision and pattern recognition* (pp. 2818-2826).

[96] Ronneberger, O., Fischer, P., Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (pp. 234-241). Springer, Cham.

[97] Suriamin, F., Pranter, M. J. (2018). Stratigraphic and lithofacies control on pore characteristics of Mississippian limestone and chert reservoirs of north-central Oklahoma. *Interpretation,* 6(4), T1001-T1022.

[98] Pires de Lima, R., Suriamin, F., Marfurt, K. J., Pranter, M. J. (2019). Convolutional neural networks as aid in core lithofacies classification. *Interpretation,* 7(3), SF27-SF40.

[99] Zhong, Z., Carr, T. R., Wu, X., Wang, G. (2019). Application of a convolutional neural network in permeability prediction: A case study in the Jacksonburg-Stringtown oil field, West Virginia, USA. *Geophysics,* 84(6), B363-B373.

[100] Mauran, S., Rigaud, L., Coudevylle, O. (2001). Application of the Carman–Kozeny correlation to a high-porosity and anisotropic consolidated medium: the compressed expanded natural graphite. *Transport in porous media,* 43(2), 355-376.

[101] Voulodimos, A., Doulamis, N., Doulamis, A., Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience,* 2018.

[102] Ikonomakis, M., Kotsiantis, S., Tampakas, V. (2005). Text classification using machine learning techniques. *WSEAS transactions on computers,* 4(8), 966-974.

[103] Nassif, A. B., Shahin, I., Attili, I., Azzeh, M., Shaalan, K. (2019). Speech recognition using deep neural networks: A systematic review. *IEEE access,* 7, 19143-19165.

[104] Pathak, A. R., Pandey, M., Rautaray, S. (2018). Application of deep learning for object detection. *Procedia computer science,* 132, 1706-1717.

[105] Everson, L., Biswas, D., Panwar, M., Rodopoulos, D., Acharyya, A., Kim, C. H., ... Van Helleputte, N. (2018, May). BiometricNet: Deep learning based biometric identification using wrist-worn PPG. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS) (pp. 1-5).* IEEE.

[106] Guimaraes, R. G., Rosa, R. L., De Gaetano, D., Rodriguez, D. Z., Bressan, G. (2017). Age groups classification in social network using deep learning. *IEEE Access,* 5, 10805-10816.

[107] Qayyum, A., Anwar, S. M., Awais, M., Majid, M. (2017). Medical image retrieval using deep convolutional neural network. *Neurocomputing,* 266, 8-20.

[108] Glorot, X., Bengio, Y. (2010, March). Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the *thirteenth international conference*

on artificial intelligence and statistics (pp. 249-256). JMLR Workshop and Conference Proceedings.

[109] He, K., Zhang, X., Ren, S., Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the *IEEE international conference on computer vision* (pp. 1026-1034).

[110] Hareland, G. A. (1994). Evaluation of flour particle size distribution by laser diffraction, sieve analysis and near-infrared reflectance spectroscopy. *Journal of Cereal Science,* 20(2), 183-190.

[111] Kumara, G. H. A. J. J., Hayano, K., Ogiwara, K. (2012). Image analysis techniques on evaluation of particle size distribution of gravel. *Int. J. Geomate,* 3(1), 290-297.

[112] Eisma, D., Bernard, P., Cadee, G. C., Ittekkot, V., Kalf, J., Laane, R. W. P. M., ... Schuhmacher, T. (1991). Suspended-matter particle size in some West-European estuaries; Part I: Particle-size distribution. *Netherlands journal of sea research,* 28(3), 193-214.

[113] Yin, X., Chen, W., Wu, X., Yue, H. (2017, June). Fine-tuning and visualization of convolutional neural networks. In 2017 12th *IEEE Conference on Industrial Electronics and Applications (ICIEA)* (pp. 1310-1315). IEEE.

[114] Caruana, R. (1995). Learning many related tasks at the same time with backpropagation. In *Advances in neural information processing systems* (pp. 657-664).

[115] Halotel, J., Demyanov, V., Gardiner, A. (2020). Value of geologically derived features in machine learning facies classification. *Mathematical Geosciences,* 52(1), 5-29.

[116] Bestagini, P., Lipari, V., Tubaro, S. (2017). A machine learning approach to facies classification using well logs. In *Seg technical program expanded abstracts 2017* (pp. 2137-2142). Society of Exploration Geophysicists.

[117] Hall, B. (2016). Facies classification using machine learning. *The Leading Edge,* 35(10), 906-909.

[118] Mandal, P. P., Rezaee, R. (2019). Facies classification with different machine learning algorithm–An efficient artificial intelligence technique for improved classification. *ASEG Extended Abstracts,* 2019(1), 1-6.

[119] Bohling, G. C., Dubois, M. K. (2003). An integrated application of neural network and Markov chain techniques to the prediction of lithofacies from well logs: *Kansas Geological Survey Open-File Report* 2003-50, 6 p. Group.

[120] Dubois, M. K., Bohling, G. C., Chakrabarti, S. (2007). Comparison of four approaches to a rock facies classification problem. *Computers Geosciences,* 33(5), 599-617.

[121] Wikipedia contributors. (2021, March 2). Suncor Energy. In *Wikipedia, The Free Encyclopedia.* Retrieved 21:20, June 27, 2021, from https://en.wikipedia.org/w/index.php?title=Suncor_Energyoldid=1009743501

[122] Wikipedia contributors. (2021, June 10). Athabasca oil sands. In *Wikipedia, The Free Encyclopedia.* Retrieved 21:21, June 27, 2021, from https://en.wikipedia.org/w/index.php?title=Athabasca_oil_sandsoldid=1027812817

[123] Kingma, D. P., Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

[124] Wikipedia contributors. (2021, March 20). Decision tree. *In Wikipedia, The Free Encyclopedia.* Retrieved 02:31, June 30, 2021, from https://en.wikipedia.org/w/index.php?title=Decision_treeoldid=1013198756

[125] Bisong, E. (2019). More supervised machine learning techniques with scikit-learn. In *Building Machine Learning and Deep Learning Models on Google Cloud Platform* (pp. 287-308). Apress, Berkeley, CA.

[126] Brownlee, J. (2016). How to grid search hyperparameters for deep learning models in python with keras. línea]. [Online]. Available: https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/. [Accessed: August 24, 2021].

[127] Dike, H. U., Zhou, Y., Deveerasetty, K. K., Wu, Q. (2018, October). Unsupervised learning based on artificial neural network: A review. In *2018 IEEE International Conference on Cyborg and Bionic Systems (CBS)* (pp. 322-327). IEEE.

[128] Baker, B., Gupta, O., Naik, N., Raskar, R. (2016). Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167.*

[129] Willmott, C. J., Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate research,* 30(1), 79-82.

[130] Reza, A. M. (2004). Realization of the contrast limited adaptive histogram equalization (CLAHE) for real-time image enhancement. *Journal of VLSI signal processing systems for signal, image and video technology,* 38(1), 35-44.

[131] Jabro, J. D. (1992). Estimation of saturated hydraulic conductivity of soils from particle size distribution and bulk density data. *Transactions of the ASAE,* 35(2), 557-560.

[132] Lee, K. L., Farhoomand, I. (1967). Compressibility and crushing of granular soil in anisotropic triaxial compression. *Canadian geotechnical journal,* 4(1), 68-86.

[133] Wang, J. P., Hu, N., François, B., Lambert, P. (2017). Estimating water retention curves and strength properties of unsaturated sandy soils from basic soil gradation parameters. *Water Resources Research,* 53(7), 6069-6088.

[134] ASTM, D. (2011). 2487. Standard practice for classification of soils for engineering purposes. In *American Society for Testing of Materials.*

[135] Su, Y. Z., Zhao, H. L., Zhao, W. Z., Zhang, T. H. (2004). Fractal features of soil particle size distribution and the implication for indicating desertification. *Geoderma,* 122(1), 43-49.

[136] Wikipedia contributors. (2021, May 17). Laser diffraction analysis. In *Wikipedia, The Free Encyclopedia.* Retrieved 10:04, July 9, 2021, from https://en.wikipedia.org/w/index.php?title=Laser$_d iffraction_a nalysisoldid$ = 1023569489

[137] Lee, C. J., Wang, S. D. (1999). Fingerprint feature extraction using Gabor filters. *Electronics Letters,* 35(4), 288-290.

[138] Li, W., Mao, K., Zhang, H., Chai, T. (2010, September). Selection of gabor filters for improved texture feature extraction. In *2010 IEEE International Conference on Image Processing* (pp. 361-364). IEEE.

[139] Vincent, O. R., Folorunso, O. (2009, June). A descriptive algorithm for sobel image edge detection. In *Proceedings of informing science IT education conference (InSITE)* (Vol. 40, pp. 97-107).

[140] Detmer, D. M. (1995). Permeability, porosity, and grain-size distribution of selected Pliocene and Quaternary sediments in the Albuquerque Basin. *New Mexico Geology,* 17(4), 79-87.

[141] Yoneda, J., Oshima, M., Kida, M., Kato, A., Konno, Y., Jin, Y., ... Tenma, N. (2019). Permeability variation and anisotropy of gas hydrate-bearing pressure-core sediments recovered from the Krishna–Godavari Basin, offshore India. *Marine and Petroleum Geology,* 108, 524-536.

[142] Wilson, A. M., Huettel, M., Klein, S. (2008). Grain size and depositional environment as predictors of permeability in coastal marine sands. *Estuarine, Coastal and Shelf Science,* 80(1), 193-199.

[143] Breiman, L. (2001). Random forests. *Machine learning,* 45(1), 5-32.

[144] Oshiro, T. M., Perez, P. S., Baranauskas, J. A. (2012, July). How many trees in a random forest?. In *International workshop on machine learning and data mining in pattern recognition* (pp. 154-168). Springer, Berlin, Heidelberg.

[145] Probst, P., Wright, M. N., Boulesteix, A. L. (2019). Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery,* 9(3), e1301.