

UNIVERSITY OF THE
FREE STATE
UNIVERSITEIT VAN DIE
VRYSTAAT
YUNIVESITHI YA
FREISTATA



EFFECTS OF INCREASING THE DIMENSIONS OF CONVOLUTIONAL NEURAL NETWORKS (CNN) ON THEIR PERFORMANCE AND ACCURACY FOR ORE GRADE ESTIMATION FROM DRILL CORE HYPERSPECTRAL DATA

By

Karabo Elsie MAHLAPHA

Student No.: 2018289369

MSc dissertation submitted to the Faculty of Natural and Agricultural Sciences, University of the Free State, in fulfilment of the requirements for the degree of Masters.

Supervisor: Prof. E.J.M Carranza

Department of Geology
University of the Free State
Bloemfontein, South Africa

October 2025

DECLARATION

I, **Karabo MAHLAPHA**, declare that the master's degree research dissertation that I herewith submit for the master's degree qualification in Geology at the University of the Free State is my independent work, and that I have not previously submitted it for a qualification at another institution of higher education.



Karabo Mahlapa

Date: 06/10/2025

SUPERVISOR DECLARATION

I, Prof. Emmanuel John M. Carranza, approve the submitted MSc thesis of Ms. **Karabo MAHLAPHA** for assessment and that the submitted work has not previously, either in part or in its entirety, been submitted to the examiners or moderators.

Signature:

Date: 06/10/2025


Name: Prof. E.J.M Carranza

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Prof. John Carranza, for his guidance and constructive feedback throughout this work. I also thank the Mandela Mining Precinct for the bursary and funding to complete this study. To my labmates at the SAMERDI Research Centre for Advanced Orebody Knowledge, your support and insightful discussions are cherished.

Lastly, I express my heartfelt thanks to my parents for their encouragement, support and belief in me thus far in my academic journey. Their motivation and encouragement sustained me through challenging moments.

ABSTRACT

The applications of deep learning for ore grade estimation from hyperspectral data are increasing. Currently, convolutional neural networks (CNN) are being used to extract deep features from data and learn relationships that exist between hyperspectral and ore grade data. However, these algorithms are still below optimal for ore grade estimation. This study endeavoured to investigate if the accuracy of ore grade estimation can be increased by changing the architecture of these algorithms and improve them for non-image hyperspectral data, for which CNN models are not widely applied. The dimensions of the CNN models were increased from 2D to 3D and the models were trained on spectrogram plots obtained from the short-time Fourier transform of reflectance spectra that were acquired from drill cores of the Olympic dam deposit in Australia. A ResNet-3D model achieved a root mean squared error (RMSE) of 0.238 ppm for Au grade estimation from data that had 75% of the Au grades below 0.625 ppm. This was lower than the 0.531 ppm RMSE achieved by a DenseNet-2D model on the same dataset and the 0.48 Cu% RMSE achieved by a VGG16 2D CNN on a dataset that had 75% of the data points below 1 Cu% reported in the literature. These results imply that the ResNet-3D model achieved higher accuracy of Au grade estimation for individual points within the data and that the under- or overestimation of the Au grades by this model would be lower than the other 2D CNN models when applied to testing data. Thus, the results imply that, in the application of CNN for ore grade estimation in research and in the mining industry, 3D CNN models should be considered.

Table of Contents

DECLARATION.....	ii
SUPERVISOR DECLARATION	iii
Chapter 1 : Introduction	1
1.1 History of Mining Exploration and Resource Estimation	1
1.2 Problem Statement.....	2
1.3 Research Aim and Objectives	4
1.4 Significance of Study.....	5
1.5 Structure Outline	6
Chapter 2 : Literature Review	7
2.1 Convolutional Neural Network (CNN)	7
2.1.1 <i>Architecture of CNN</i>	9
2.1.2 <i>Topology of CNN: Convolutions and Dimensions of CNN</i>	12
2.1.3 <i>Training of CNN</i>	15
2.2 HS Data: Acquisition and Pre-processing.....	16
2.3 Frequency-domain representation of HS Reflectance Spectra	17
2.4 Accuracy and Performance Measurement.....	18
2.5 Hypotheses	20
2.5.1 <i>Hypothesis 1: Relationship of Dimensions with Performance of CNN</i>	21
2.5.2 <i>Hypothesis 2: Relationship of Architecture with Performance of CNN</i>	21
Chapter 3 : Olympic Dam Geology and Data Collation	22
3.1 Olympic Dam Geology and Mineralogy.....	22
3.2 Data Collation.....	24
Chapter 4 : Data Pre-processing and Model Architecture Selection	26
4.1 Data Cleaning	27
4.2 Data Pre-processing	29
4.3 Data Splitting.....	32
4.4 2D and 3D CNN model architecture selection	32
4.4.1 <i>2D CNN Model Architecture Selection</i>	33
4.4.2 <i>3D CNN Model Architecture Selection</i>	36
4.5 Conclusion	39
Chapter 5 : Model Training and Results.....	40
5.1 Correlation Analysis	40
5.2 DenseNet-2D and ResNet-3D Model Training, Results and Comparison.....	45
5.2.1 <i>DenseNet-2D and ResNet-3D Model Training and Results</i>	45
5.2.2 <i>Comparison of the Accuracies of Prediction of DenseNet-2D and ResNet-3D</i>	47
Chapter 6 : Discussion	50

6.1 DenseNet-2D and ResNet-3D Model Accuracies and Performance for Ore Grade Estimation.....	50
6.2 Normalization of Spectrograms for ResNet-3D and DenseNet-2D CNN Models.....	55
6.3 CNN Model Performance on Reflectance Spectra with Subtle Features.....	56
Chapter 7 : Conclusion	59
References	62
Appendices.....	65

TABLE OF FIGURES

Figure 2.1	Diagram illustrating how kernels apply (2D) convolutions to the input and stride along summing up the products to result in a feature map (modified from Yamashita et al. (2018))	13
Figure 2.2	Diagram illustrating how 3-D convolutions are applied to an input to result in a feature map as the result of the multiple convolutions (modified from Rao & Liu (2020))	14
Figure 2.3	Conceptual model of the study. Performance of CNN for ore grade estimation is influenced by the dimensions and architecture of CNN, and the amount and structuring of the data. Therefore, the hypotheses are based on the conditions of HS data and CNN architecture.	20
Figure 3.1	Geological map of the eastern margin of the Gawler craton showing the location of Olympic Dam deposit. The deposit, which is shown by the mine symbol on the map, is located in the Hiltaba Granite Suite of the Gawler silicic large igneous province (map from McPhie et al., 2011).	23
Figure 3.2	Simplified geological map of the Olympic Dam breccia complex showing the lithology and mineralogy of the complex and deposit. Au mineralisation is associated with Cu mineralisation in the deposit. The locations of the drill cores from which the data that were used in this study were obtained are shown on the map (map modified from Prado et al. (2023))	24
Figure 4.1	Ore grade estimation workflow using deep learning (DL) algorithms	26
Figure 4.2	Study workflow showing the steps from data cleaning to model evaluation using accuracy metrics (STFT: Short-Time Fourier Transform, VGG16: Visual Geometry Group 16, ResNet: Residual Networks, DenseNet: Densely Connected Convolutional Networks, FCN: Fully Convolutional Networks, SGCN: Spectral Graph Convolutional Networks, ResNeXt: Residual Networks with Aggregated Transformations, SECF: Squared error cost function, R ² : Coefficient of determination, RMSE: Root mean square error)	27
Figure 4.3	(a) Reflectance spectra of the 300–302 m interval of the RD76 drill core and (b) average reflectance spectra for this interval.	28
Figure 4.4	Average reflectance spectrum for depth interval 300-302m of drill core RD76 and the standard deviation of the original reflectance spectra from which the average reflectance spectrum was computed. Diagnostic absorption and reflectance features in the average spectrum occur at the same wavelength as those occurring in the standard deviation.	30
Figure 4.5	Average reflectance spectra of the 300–302 m interval of the RD76 drill core. (a) 2D spectrogram plots; (b) 3D spectrogram plot	31
Figure 4.6	Model architecture for VGG16 2D CNN that was trained in this study. The training hyperparameter for this model were batch size =32, epoch = 50, optimizer = adam	33
Figure 4.7	Model architecture for a standard 2D CNN that was trained in this study. The training hyperparameter for this model were batch size =32, epoch = 50, optimizer = adam	34
Figure 4.8	Model architecture for a ResNet-2D that was trained in this study. The training hyperparameter for this model were batch size =32, epoch = 50, optimizer = adam	34
Figure 4.9	Model architecture for a DenseNet-2D that was trained in this study. The training hyperparameter for this model were batch size =32, epoch = 50, optimizer = adam	34
Figure 4.10	Model architecture for a 2D FCN that was trained in this study. The training hyperparameter for this model were batch size =32, epoch = 50, optimizer = adam	35
Figure 4.11	Model architecture for a 2D SGCN that was trained in this study. The training hyperparameter for this model were batch size =32, epoch = 50, optimizer = adam	35
Figure 4.12	Model architecture for a 2D ResNeXt that was trained in this study. The training hyperparameter for this model were batch size =32, epoch = 50, optimizer = adam	35
Figure 4.13	Model architecture for a 2D U-Net that was trained in this study. The training hyperparameter for this model were batch size =32, epoch = 50, optimizer = adam	35
Figure 4.14	Architecture of standard 3D CNN that was trained in this study. The training hyperparameter for this model were batch size =16, epoch = 50, optimizer = adam	37

Figure 4.15 Architecture of ResNet-3D model that was trained in this study. The training hyperparameter for this model were batch size =16, epoch = 50, optimizer = adam	37
Figure 4.16 Architecture of DenseNet-3D model that was trained in this study. The training hyperparameter for this model were batch size =16, epoch = 50, optimizer = adam	37
Figure 4.17 Architecture of 3D FCN model that was trained in this study. The training hyperparameter for this model were batch size =16, epoch = 50, optimizer = adam	38
Figure 4.18 Architecture of 3D SGCN model that was trained in this study. The training hyperparameter for this model were batch size =16, epoch = 50, optimizer = adam	38
Figure 4.19 Architecture of 3D ResNeXt model that was trained in this study. The training hyperparameter for this model were batch size =16, epoch = 50, optimizer = adam	38
Figure 5.1 Histogram of Au grade values of the data that were used to train, validate and test the models. The data points are also plotted above the histogram to show their skewness. The quartile cutoff grades are shown by the blue, green and red lines.	41
Figure 5.2 Scatter plot of 3000 Au grade data points that were below 0.625 ppm. These points make up 75% of all the Au grade points in the data. The quartile cutoff grades are shown by the blue, green and red lines.	42
Figure 5.3 Reflectance spectra of corresponding quartiles (Q1, Q2, Q3 and Q4) that the data were split into. The quartile ranges and number of samples in the quartile are shown next to each quartile number on the upper left.	43
Figure 5.4 Continuum removed reflectance spectra of corresponding quartiles (Q1, Q2, Q3, Q4) that the data were split into. The quartile ranges and number of samples in the quartile are shown next to each quartile number on the upper right.	45
Figure 6.1 Illustration of extraction of spatial and spectral features from HS images and the application of this extraction in 3D spectrogram. (a) Hyperspectral image showing the extraction of spatial features in 2D and the extraction of spectral features in 3D. This extraction collectively makes for extraction in 3D. (b) 2D spectrogram representations of reflectance data used in this study showing the extraction of spatial features from frequency and wavelength relationship (c) 3D spectrogram representation used in this study showing the extraction of spatial features from frequency and wavelength relationship, and extraction of spectral feature in a third amplitude dimension.	52
Figure 6.2 Original (left) and Normalized (right) reflectance spectra of 300 – 302 m interval of drill core RD76. Both spectra have the same absorption and reflectance features.	56
Figure 6.3 Continuum-removed and original 3D spectrograms for the interval 300-303 m of drill core RD76.	58
TABLE OF TABLES	
Table 4.1 SECF of each 2D model architecture on the validation dataset	36
Table 4.2 SECF values of each 3D model architecture on the validation dataset	39
Table 5. 1 Table of quartiles and their corresponding Au grade ranges that data was split into	43
Table 5. 2 Table of RMSE of testing data for DenseNet-2D and ResNet-3D	47
Table 5. 3 Table of RMSE of quartile testing datasets for 2D CNN models	48
Table 5. 4 Table of RMSE of quartile testing datasets for 3D CNN models	48

List of Abbreviations

CNN	Convolutional neural network
AI	Artificial Intelligence
HIS	Hyperspectral imaging
ML	Machine learning
OA	Overall accuracy
DL	Deep learning
ANN	Artificial neural network
VGG	Visual geometry group
RMSE	Root mean squared error
SWIR	Shortwave infrared
VNIR	Visible-near infrared
TIR	Thermal infrared
ODBC	Olympic Dam breccia complex
STFT	Short-time Fourier transform
SECF	Squared error cost function

Chapter 1 : Introduction

1.1 History of Mining Exploration and Resource Estimation

Resource estimation in the mining industry has been evolving. It upgraded from the use of classical estimation methods such as polygonal and cross-sectional methods to the use of geostatistical methods that consider the geology of the deposits and, consequently, increase the accuracy of the estimations (Abuntori et al., 2021). However, with the development of artificial intelligence (AI) and surge of deposits data in the last few decades, much research has gone into how these can be leveraged for resource estimation. AI, which is used to analyse big data and predict mineral grades, can be applied through two main methodological approaches: knowledge-driven AI and data-driven AI (Yang et al., 2024).

Knowledge-driven AI makes use of the knowledge of the geology of deposits to guide AI models in the analysis of data for predictions. In some applications, the knowledge is incorporated into AI models as associated mineralogy of the deposit types (Tan et al., 2024) and, for some applications, spectral libraries are built into a model from knowledge of other deposits to guide the AI models in the analysis (Clark et al., 2003; Dalton et al., 2004). Some research has also been done to incorporate geostatistical formulas that account for the geology of the deposit into AI models to code the analysis of the data. For example, Bai & Tahmasebi (2021) incorporated ordinary kriging (OK) formulas into the loss function of a deep neural network to estimate grades of unknown points. This created a hybrid AI model, which not only captured the spatial distribution of the estimated grades but also reduced the number of unknown points required for the training of model. However, the development of data-driven AI has created models that can also capture the spatial distribution of predicted grades and

estimate mineral resources without incorporation of the knowledge of the geology of the deposit prior to application.

Data-driven AI relies on models that learn patterns from input-output relationships in training data to make predictions for new, unseen data. These models can extract spatial and spectral features from data and capture complex linear and non-linear relationships that exist within the data to make predictions (Liu, 2021). This ability of AI makes it advantageous over geostatistics for resource estimation, in that, unlike geostatistics, AI can learn the non-linear spatial trends that may exist in data of structurally controlled deposits like vein deposits and make predictions for the quantity of ore at the points within the vein (Abuntori et al., 2021). However, AI models, whether data- or knowledge-driven, still require improvement in their performance if they are to be used for resource estimation and assimilated in the mining industry.

1.2 Problem Statement

One of the data-driven AI models used for ore grade estimation is convolutional neural network (CNN) (Prado et al., 2023). It is one of the state-of-the-art AI algorithms that can capture non-linear relationships that may exist between two sets of data and use such relationships to make predictions for new data. They have convolutional filters that can extract features from input data and map them to output data to learn the non-linear relationships. Although they are currently also used for any data that has topology and grid pattern, they were designed to analyse image data (Krizhevsky et al. (2012); Yamashita et al. (2018)) for classification. For this reason, they were initially applied as 2D (two-dimensional) CNN with 2D filters that extract data from images, which are represented as 2D arrays of pixels into the algorithm. Most

studies on the use of CNN have investigated how increasing the dimensions of the CNN from 2D to 3D (three-dimensional) affects its performance for hyperspectral image data analysis, where spatial relationships between adjacent pixels are naturally exploited through convolutional operations (Chen et al. (2016); Praveen & Menon (2021); Abubakar et al. (2024)). The performance increases when the dimensions are increased from 2D to 3D because the filters can extract both spatial and spectral features in 3D, which leads to learning of more patterns in the hyperspectral images and labels than in 2D, which extract only spatial features. However, there is inadequate research on how increasing the dimensions of CNN from 2D to 3D affects performance for point spectral measurements from drill cores, where each spectrum is typically processed independently. When individual spectra are transformed into spectrograms for CNN analysis, convolutions operate within each spectrum's spectral-frequency representation rather than across spatially adjacent drill hole samples, representing a fundamentally different architecture from image-based approaches. This research gap is particularly important because reflectance spectra, as non-image data that are broadly used for ore grade estimation, yield high errors of ore grade estimation.

The performance of ore grade estimation by current applications of CNN models to hyperspectral (HS) reflectance data is low. For examples, in a study to investigate the use of a 2D CNN model referred to as VGG16 for Cu grade estimation to reflectance spectra acquired along drill cores, the model achieved a root mean squared error (RMSE) of 0.48 Cu% (Prado et al., 2023). While the model was proven to achieve Cu grade estimations for all the datapoints of data that was used to test the model, this RMSE was high for this dataset. The error implies that the model would over- or under-estimate the Cu% values by 0.48 Cu%, which is not satisfactory for a dataset that had at least 75% of datapoints below 1 Cu% because the over- or under-estimates would be significantly far from the actual values ($1 - 0.48 \text{ Cu\%} = 0.52 \text{ Cu\%}$).

Furthermore, in classification for ore grade estimation, all the 2D CNN models studied by Eichstaedt et al. (2022) and Kutzke et al. (2022) achieved 80–90% classifier accuracies, which can still be improved to higher accuracies. Therefore, the current frameworks of application of CNN for ore grade estimation reported in the literature, are not satisfactory and reliable, and still require optimization to achieve lower errors and higher accuracies of ore grade estimation.

1.3 Research Aim and Objectives

Given the higher errors of grade estimations reported from the current frameworks of application of CNN models, and the lack of research on how increasing the dimensions of CNN from 2D to 3D affects their performance for non-image data, this study endeavoured to investigate how increasing the dimensions of CNN from 2D to 3D would affect the performance for ore grade prediction from HS reflectance data in the frequency domain, one of the frameworks currently used for ore grade estimation. This aim was achieved through the following objective:

- To train and build 2D and 3D CNN models using spectrograms converted from HS reflectance data and compare the accuracies and performance of 2D and 3D CNN for HS reflectance data for ore grade estimation.

Therefore, the research attempted to answer the following question:

1. What are the performances of 2D and 3D CNN models and how do they compare for HS reflectance data in the frequency-domain for ore grade estimation?

1.4 Significance of Study

Most researches comparing the performance of 2D CNN and 3D CNN has applied CNN to hyperspectral imaging (HSI) data. The results of these studies have not only demonstrated 3D as the optimum dimensions of CNN for the imaging data but have also shown how the features extracted from the data are used for their purposes. In a study to compare the performance of some state-of-the-art machine learning (ML) algorithms on lithological classification of the Huaniu mountains in Gansu, China, Liu et al., (2021) found that 3D CNN obtained a higher overall accuracy (OA) of 98.56% for classification of the HS image compared to 2D and 1D CNN, which achieved 96.47% and 94.70% accuracies, respectively. Therefore, since this study endeavours to investigate how the accuracy and performance are affected by these changes for hyperspectral data, it contributes knowledge to the application of 3D CNN for non-image data, which has not been done before. Furthermore, the knowledge is contributed to the context of ore grade estimation, in which the previous studies have not applied 3D CNN models.

Although geostatistical and classical statistical methods are still the only estimation methods widely used in the mining industry, their application is complex and susceptible to errors, especially for structurally controlled deposits. Data obtained from structurally controlled deposits are usually skewed and contain outliers, which would create a nugget effect on the variograms during the application of these methods and make the variograms useless for estimation. In attempting to solve this, the geologist would reduce the number of outlier points, which would lead to under- or overestimation of ore grades. However, deep learning (DL) algorithms, which include CNN, can capture and learn patterns from non-linear data which render them more useful for structurally controlled deposits such as vein deposits. Therefore, this study also contributes to the existing body of knowledge for improving resource estimation and shift mineral exploration to the use of the complicated but robust methods.

The integration of ML/DL with HS data for ore grade estimation is less time consuming, less laborious and less destructive than conventional methods which are also used in ore grade estimation. These conventional methods which include inductively coupled plasma (ICP), x-ray fluorescence (XRF) and Fire assay analysis, require extraction of rocks which may lead to acid drains and renders them more laborious and destructive to the environment compared to the integration of HS data with ML and DL techniques which usually require HS scanning of intact drill core or outcrops to obtain the data, which is analysed within minutes of computational time. Therefore, this research will contribute to the knowledge of the methods that can be used in the mining industry which would not only be fast but also lower mining costs and be less destructive to the environment.

1.5 Structure Outline

The rest of this thesis is structured as follows: In chapter 2, the theory and application of current applications of CNN for ore grade estimation are provided and reviewed. The geology of the deposit from which the data was obtained and how data was collected are explained in chapter 3. The study area chapter is succeeded by the chapter 4 which reports on the methodology that was used to pre-process the data and select the best models for the data representation in the study. The details of training of the 2D and 3D CNN for comparison of accuracy for ore grade estimation are provided in chapter 5, which is followed by the interpretation of the results is provided in chapter 6 and conclusions in chapter 7.

Chapter 2 : Literature Review

Machine learning (ML) and deep learning (DL) algorithms have been used to estimate resources from the integration of hyperspectral (HS) data of rocks with their corresponding geochemical assays. These data are used to train these algorithms by labelling the HS data with their corresponding geochemical data so that these algorithms learn linear or non-linear relationships that exist between these data and use them to make predictions of geochemical grades for new HS data. This is supervised learning, which differs from unsupervised learning, in which the algorithms learn relationships without labelling HS data with geochemical data. A convolutional neural network (CNN) trains and learns relationships by supervised learning. However, the properties of the data and the structure of a CNN influence the performance of this algorithm for ore grade estimation. Therefore, in this chapter, the framework of the applications of CNNs, which includes the structures of CNNs, and the properties of the data are reviewed for ore grade estimation. Some relevant literature on the use of this algorithm for ore grade estimation are also reviewed to generate research hypotheses, which are detailed at the end of this chapter as part of the conceptual model of this study.

2.1 Convolutional Neural Network (CNN)

CNN is a DL algorithm designed for grid-structured data such as images. It uses stacked convolutional layers to automatically learn features at different levels - early layers detect simple patterns like edges or textures, while deeper layers identify more complex, abstract features used for final predictions (Bengio, 2009). These layers, which are referred to as convolutional layers, have filters or kernels which stride along the data applying specific mathematical operations to the data to compute an output of features referred to as the feature

map. These kernels are the structures of the CNNs that enable them to automatically learn the features from the data. The feature map is then input into other types of layers that make up the CNN. These layers, which have different roles from the convolutional layer, are the pooling and fully connected layers. Pooling layers downsample these maps, reducing computation while making the network more robust to small translations. The fully connected layers at the end combine all learned features to produce the final prediction.

CNN's ability to learn features through hierarchical feature extraction, which contributes to the 'deep' learning, was implemented by Le Cun (1989), who built upon Fukushima (1980)'s earlier concepts of local receptive fields and filters in an artificial neural network (ANN), named Neocognitron, that was invented by Fukushima (1980). This improvement to the Neocognitron algorithm was inspired by the mammalian primary visual cortex (PVC) which it resembles. CNN was designed to resemble PVC which has a 2D (two-dimensional) structure and is a spatial map (Goodfellow et al., 2016). For this reason, it was designed to analyse image data (Yamashita et al., 2018). PVC also has simple cells which are characterized by linear functions and complex cells which are affected by a change in the position of their feature and measurement. The activation functions in the convolutional layers and the pooling layers, respectively, were inspired by these cells (Goodfellow et al., 2016). Le Cun (1989) introduced this framework of CNN into Neocognitron as a backpropagation technique that was proposed by Rumelhart et al. (1986) into the CNN and named the CNN LeNet-5 (Le Cun et al., 1995). Since then, CNN has since been applied to images, text, time series data and sensor data for classification applications in face recognition, quality control, medicine, geological mapping and grade and tonnage estimation.

2.1.1 Architecture of CNN

A typical architecture of a CNN has convolutional layers, pooling layers and fully connected layers (Namatēvs, 2018), and the number and successions of the layers may differ among CNN models. The first layer that the data is fed into is the input layer. It is usually succeeded by convolutional layers which are the main layers of a CNN. Convolutional layers have two stages: the convolutional stage and the detector stage. The convolutional stage is where features in the data are extracted using kernels. The kernels in each of the layers extract the features at different levels with the kernels in the initial convolutional layers extracting low-level features and the kernels in the last layer extracting the high-level features. Each kernel produces a feature map which will be an input into the next layer. The next stage of the convolutional layers is the detector stage. The detector stage is the stage by which non-linearity is applied to the data. This stage gives CNN and other ML algorithms the ability to learn complex non-linear patterns in data, importantly, data that have outliers which are particularly common in geological deposit data (Abuntori et al., 2021). This application of non-linearity is achieved by non-linear activation functions in multiple neurons present in the convolutional layers. Some of the common activation functions include the hyperbolic tangent function $g(z) = \tanh(z)$, sigmoid function ($g(z) = \text{sigmoid}(z)$) and the ReLU function ($g(z) = \max(0, z)$) (Krizhevsky et al., 2012). The ReLU function is commonly used in CNN architectures because it decreases the computational time for training. The output of these convolutional layers is then fed into the pooling layers and fully connected layers.

The pooling layers and the fully connected layer are usually the final layers in the architectures of CNNs. The pooling layers are applied for down-sampling of the features which leads to dimensionality reduction of the feature maps. This is another feature of CNN that makes it advantageous over other ML algorithms because it automatically performs dimensionality

reduction, which is crucial for HS data that commonly have redundant features. The pooling layers achieve the down-sampling by extracting patches from features maps and using the maximum value from each patch as the output feature map. The output feature maps of the pooling layers are then fed into the fully connected layers. The fully connected layers are usually 1D (one-dimensional) layers that are fully connected to the activation function of the pooling layers and the output nodes. They learn the representation of the output of the CNN. Their output may be a 1D array of numbers, which each represent the probability of the output falling into a particular class in classification applications such as lithological classification or a scalar which is a discrete measure of the target variable. CNN architectures that output the latter are commonly used for ore grade estimation.

Different architectures of CNN with differing layer numbers and successions have been used for resource estimation and most have proven to estimate ore grade with higher accuracy than other ML algorithms. Eichstaedt et al. (2022) assessed the performance of 10 commonly used ML algorithms for prediction of geochemical properties of 700 km of drill core and the algorithms included three CNN models, NN1, NN3 and NN4, with different architectures. The results showed that the three architectures have an average accuracy of 80% in classification which is higher than the accuracies of the other algorithms that were used in the study. The NN1 architecture, a modified VGG design that showed the highest accuracy compared to the other CNN architectures tested, has 8 layers including convolutional and pooling layers followed by dense layers. Kutzke et al. (2022) further used this model to investigate the effects of increasing the amount of data on the performance of ML algorithms and optimised its parameters. It achieved an accuracy greater than 80-90%. While this model was applied to HS data as reflectance spectra in both of these studies, Prado et al. (2023) applied the another VGG variant to HS data in the frequency domain for ore grade estimation and it achieved an accuracy

of 0.48 Cu% by root mean squared error (RMSE), which implies that model would achieve an error of 0.48 Cu% for predictions for new data and would under- or over-estimate the Cu grades. The conditions of application of the VGG architectures in these studies cannot be compared, as different estimation metrics have been used. Therefore, the difference of application of the models to original reflectance spectra and to reflectance spectra in the frequency domain cannot be analysed and although they demonstrate potential for continuous ore grade prediction, the results from the studies suggest that the current condition of application of the models can be improved since RMSE achieved is high and the accuracy values are lower or equal to 90%.

Other CNN architectures that have been used for ore grade estimation include the ENVI-Net model with 27 convolutional layers as the only set of layers in the architecture. Abdolmaleki et al. (2022) showed that this architecture was able to discriminate ore from waste with an accuracy of 89%. However, only one image of the whole drill core was used. Therefore, the effectiveness of the architecture was not sufficiently determined and its application for ore grade estimation is inconclusive. Another deep learning architecture of layers with a succession of four convolutional layer, two pooling layers and 2 fully connected layers was used by Choros et al. (2022) for the same purpose of ore and waste discrimination. This architecture achieved 76.7% accuracy. (Choros et al., 2022) showed that the model is a promising candidate for application for the determination of ore grade during excavation so that ore may not be diluted with waste during the process. These two studies show that although one architecture may achieve a greater accuracy than the other for the same purpose, the superiority of one over the other may still not be determined as the conditions of application were different.

The different accuracies of estimation with different architectures show that the application of CNN to analysis of HS data for ore grade estimation and other applications is sensitive to its own architecture and that none of the architectures is superior to the other because the condition of application of these architectures may be different. In actuality, this shows that in the application of CNN architectures, the optimal architecture for specific conditions of application for ore grade estimation ought to be determined. This can be done by applying various architectures to HS data under the same conditions of investigation and the architecture with the highest accuracy on training data can be used for those conditions. This approach will be used in this study.

2.1.2 Topology of CNN: Convolutions and Dimensions of CNN

Kernels, by which CNN extracts features, are applied to data through mathematical operations referred to as convolutions (Lee & Kwon, 2016). Convolutions are an element-wise multiplication of kernels, which are usually multidimensional arrays of parameters, by the input data, which are usually multidimensional array of pixels. The element-wise multiplication results in a product which reflect the extent to which the kernel and input data match if their graphs were aligned. As the products are computed from all the elements of the input data, they are summed together to create another array, which is the feature map. The process of how these products are summed together to form the features map is shown by Figure 2.1. The kernel must slide along the whole array of the input to compute these convolutions, and this results in a hyperparameter that is set during training of the model called the stride. In Figure 2.1, the stride is 1 as the kernel slides one unit to apply convolutions to the input and the kernel size is 3×3 .

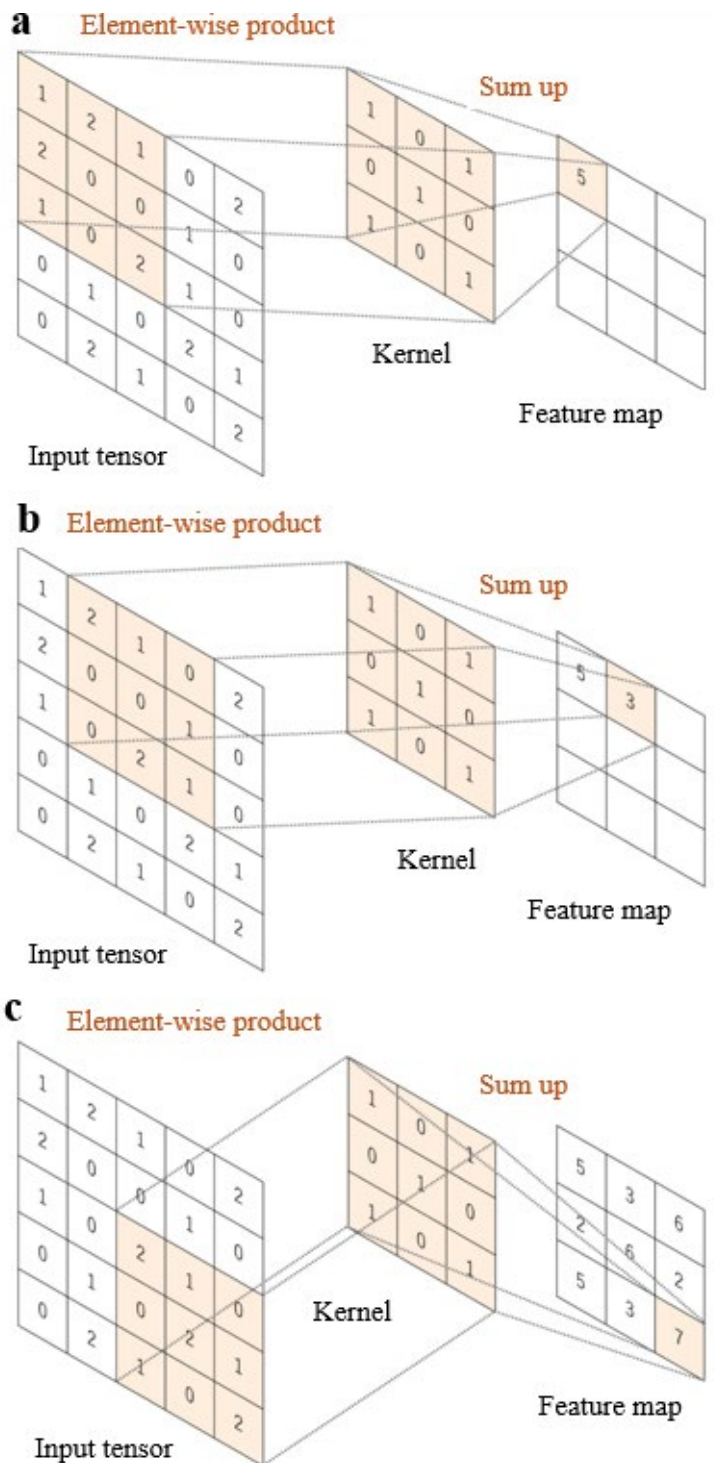


Figure 2.1 Diagram illustrating how kernels apply (2D) convolutions to the input and stride along summing up the products to result in a feature map (modified from Yamashita et al. (2018))

Kernels may have different dimensions. In 1D, the kernels have a size of $N \times 1$, where N can take any number greater than one that represents the number of elements of the array in one dimension and the 1 implies only element of the array in next dimension. A kernel of this size

only strides in the dimension of the N elements. This applies 1D convolutions to the data. To introduce more dimensions to the convolutions and the whole CNN, the kernels take a size of $N \times M$, where N and M can be any numbers greater than one, which imply the number of elements of the array in both dimensions. A kernel of this size strides along the input array in two dimensions and applies convolutions in 2D. The kernel shown in Figure 2.1 applies 2D convolution to the input because it has a size of 3×3 and it strides in both dimensions of the input as can be seen in the figure. From 2D, a third dimension can be applied to the convolution by increasing the size of the kernels to $N \times M \times L$. The L in the size can take any number of from 1 and implies that the kernel can stride in a third dimension. This applies 3D (three-dimensional) convolutions and the feature map that results would be three-dimensional. Figure 2.2 shows how this 3D convolutions can be applied to the input and result in a 3D feature map.

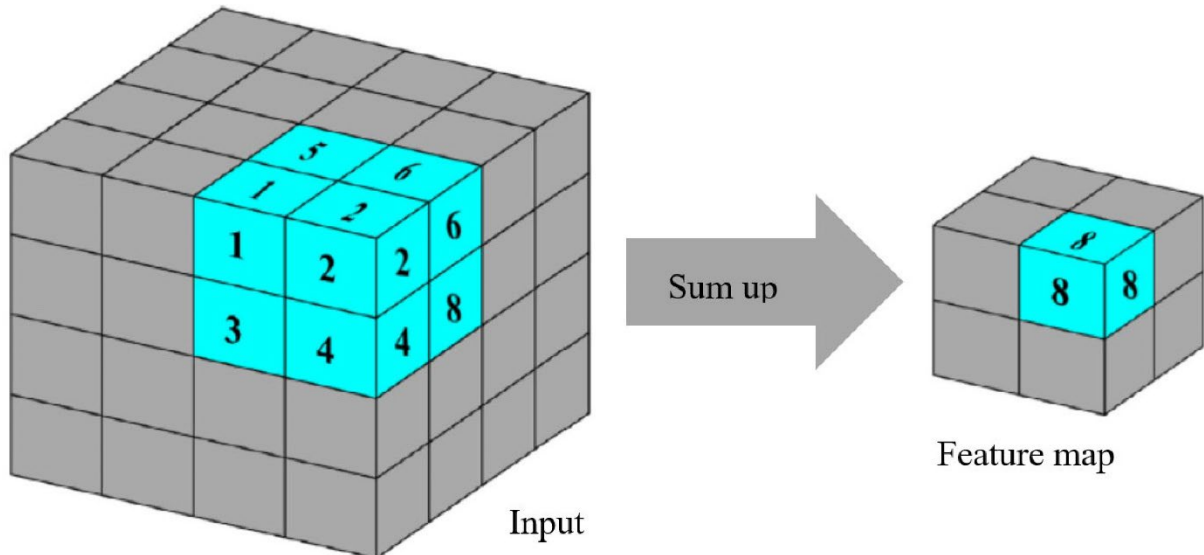


Figure 2.2 Diagram illustrating how 3-D convolutions are applied to an input to result in a feature map as the result of the multiple convolutions (modified from Rao & Liu (2020))

Different dimensions of convolutions enable a CNN to extract different kinds of features. In 1D CNN, spectral features are extracted as kernels convolve along the spectral bands of

individual pixels (Yu et al., 2020). However, 2D CNN extract spatial features (Chen et al., 2016). 3D CNN extracts both spectral and spatial features, which enables more accurate predictions (Chen et al., 2016). Most applications of CNN for ore estimation have been 2D models. The VGG models used by Eichstaedt et al. (2022), Kutzke et al. (2022) and Prado et al. (2023) and the ENVI-Net applied by Abdolmaleki et al. (2022) were 2D CNN. The conditions of application of the VGG model in the Prado et al., (2023) study were different from those of application of the same model in Eichstaedt et al. (2022) and Kutzke et al. (2022), leading to an inconclusion of the factor that led to the lower accuracy. Thus, the dimension of the CNN used in the study will be increased to 3D to investigate its effects on accuracy.

2.1.3 Training of CNN

Prior to the application of CNN, it requires training. This training is conducted so that a CNN may learn features in the training data through two processes of adjusting kernels and weights. These processes enable the CNN to calculate and gage its performance on the training data and adjust the kernels and weights to fit data. The first process is the forward propagation by which the performance is computed. Forward propagation computes predictions and then evaluates them against the true labels using a loss function. The loss function measures the prediction error enabling the computation of the performance of the CNN. The second process is the backpropagation by which the kernels and weights are adjusted. Backpropagation calculates gradients to show how much each weight contributed to the error and then updates the weights through gradient descent to reduce that error. The weights, which are parameters that the CNN learns through this process are then applied to new data to make predictions with accuracy.

2.2 HS Data: Acquisition and Pre-processing

HS data are acquired using HS sensors that may have different spectral and spatial resolutions. The spectral resolution refers to the number of bands (i.e., wavelength ranges) of the electromagnetic spectrum that the sensor acquires data from. The accuracy of resource estimation increases with this spectral resolution. However, since matter are made of different elements at different chemical bonds, which influence their reflectance of radiation, HS data of all material ought to be acquired in the range of their reflectance. Most alteration minerals like phyllosilicates (clays, micas, illite, kaolinite), carbonates, and sulphates reflect radiation in the shortwave infrared (SWIR) range of the electromagnetic spectrum and are effectively detected in these wavelength ranges (Amer et al., 2016). For this reason, all studies on the application of HS data for ore grade estimation used HS data in the SWIR region (Eichstaedt et al., 2022; Choros et al., 2022; Kutzke et al., 2022; Abdolmaleki et al., 2022). This is particularly crucial for ore grade estimation since alteration minerals are key indicators. However, Eichstaedt et al. (2022), Kutzke et al. (2022) and Prado et al. (2023) also used data from the visible-near infrared (VNIR) and thermal infrared (TIR) regions. VNIR is specifically used to detect iron oxides and hydroxides (hematite, goethite, jarosite), while TIR is used to identify rock-forming silicate minerals such as quartz, feldspars, and other non-hydroxylated minerals that don't show diagnostic features in SWIR.

HS sensors can acquire HS data at different scales. It can be acquired at a small-scale using spaceborne, airborne and unmanned aerial vehicles-based sensors or on a large-scale using ground-based sensors. For ore grade estimation, HS data are acquired at a large-scale using ground-based sensors. Choros et al. (2022) scanned a 15-m mine face and cm-length samples of gold ore from the same mine face for HS images to determine if HS imaging data and neural networks could be used to discriminate and predict the amount of ore at the point of excavation.

In contrast, Abdolmaleki et al. (2022), Eichstaedt et al. (2022), Kutzke et al. (2022) and Prado et al. (2023) acquired the HS data from drill cores. Abdolmaleki et al. (2022) acquired a scan of a whole drill core in one HS image for ore grade estimation of parts of the drill core whereas Eichstaedt et al. (2022), Kutzke et al. (2022) and Prado et al. (2023) acquired HS data from numerous parts of drill cores for grade estimation of those individual parts. This acquisition of HS data at large scale increases the spatial resolution at which the HS data is acquired and enables grade prediction at higher accuracy. This is because ground-based sensors (drill core sensors) observe smaller areas as large-scale sensors. Therefore, for ore grade estimation, it is beneficial to acquire HS data at a large-scale.

HS data have high dimensionality and redundancy, which are caused by the contiguous bands of spectra and by noise in the bands that may have been caused by sensor error(s) and atmospheric interference(s) (Kale et al., 2017). These lead to the requirement of pre-processing of HS data to determine and extract useful bands and features. The useful features may be extracted manually using techniques such as principal components analysis (PCA) and independent components analysis (ICA) while noise is reduced using minimum noise fraction (MNF) (Kale et al., 2017). However, in the application of DL methods for analysis of HS data, manual extraction of features before analysis using an algorithm is not required because features are extracted by a DL algorithm as described in section 2.1.2.

2.3 Frequency-domain representation of HS Reflectance Spectra

CNN, as a DL algorithm that does not require manual feature extraction from data, was discovered to directly learn time-frequency features from the frequency-domain representation of data and outperforms ML algorithms, which required feature extraction from such data. This

application of CNN to frequency-domain representation of data as spectrograms was first implemented on speech and audio data for sound classification (Piczak, 2015). In this study, environmental sound data were converted to log-scaled mel-spectrograms, and a deep CNN model consisting of two convolutional layers and two fully connected layers was applied to the spectrograms for classification of the sound. This simple CNN architecture outperformed support vector machines, gaussian mixture and hidden Markov models, which required manual extraction and input of mel-frequency coefficients of such data to the models for classification. Since then, CNN has been applied to frequency-domain representation of data as spectrograms in other industries and research, including mining for ore grade estimation (Prado et al., 2023).

Spectrograms are computed using the short-time Fourier transform of a signal in time-frequency analysis (Gröchenig, 2001). Prior to the Fourier transform, a time-localized windowing is applied to isolate a short segment of the signal and the windowing slides across the whole signal to isolate multiple segments of the signal. For every windowed segment of the signal, the Fourier transform is applied to compute the frequency at each specific time. The computed frequencies at each time are visualised as an intensity plot of the magnitudes of the short-time Fourier transform. This creates time-frequency grids which are spectrograms which can be analysed.

2.4 Accuracy and Performance Measurement

There are numerous metrics that can be used to measure the performance of DL algorithms for analysis of HS data. However, metrics are specific to the purpose of an application of DL algorithms. Some metrics are applicable to classification applications while some are applicable to regression applications (Naser & Alavi, 2020). Most classification metrics make

use of a matrix, referred to as a confusion matrix, which relates correctly and incorrectly classified labels and true labels (Erickson & Kitamura, 2021). From this confusion matrix, classification metrics such accuracy, precision, recall, and F1-score can be generated. These metrics measure overall correctness, correct positive predictions, ability of the model to find all positives and the balance between precision and recall, respectively. However, metrics used in regression applications to predict target values measure the error between the predicted values and the true values. Therefore, in ore grade estimation, where specific target values of ore grade are determined, regression metrics are used to measure the performance of the DL algorithms. Some of the most commonly used regression metrics for ore grade estimation are the correlation coefficient (R), coefficient of determination (R^2), mean absolute error (MAE) and root mean squared error (RMSE).

Most resource estimation studies in the literature have used classes to predict ore grades and the overall accuracy (OA) is the accuracy metric used (Choros et al. (2022); Eichstaedt et al. (2022); Kutzke et al. (2022); Abdolmaleki et al. (2022)). In these studies, ore grade values are grouped into classes and the correctly and incorrectly grouped ore grades are determined for the calculation of OA. While this accuracy metric calculates the number of correct predictions in the whole data, which is useful to measure the performance of DL algorithm on whole testing dataset, it does not take into account how the DL algorithm performs on outliers, which are common in ore deposit data. However, RMSE, another commonly used metric in ore grade estimation (e.g., Prado et al., 2023), takes outliers into account as it measures the mean of the error between predicted values and true values. Nevertheless, this metric can only be used in regression studies. Although these metrics are used in different applications of ore grade estimation, that is, classification and regressions, this review highlights the requirement to use multiple evaluation metrics for each of the conditions of classification as this can overcome the

limitations of one evaluation metric. That is, in classification, it is beneficial to use multiple classification evaluation metrics, while in regression, multiple regression evaluation metrics are beneficial.

2.5 Hypotheses

In this section, hypotheses based on the preceding review of CNN, HS data and performance of ore grade estimation and on how the performance is influenced by structures of CNN and properties of HS data, are postulated. These hypotheses are used to build a conceptual model of this study (Figure 2.3) to investigate the relationships between the performance and the structures of CNN and properties of data.

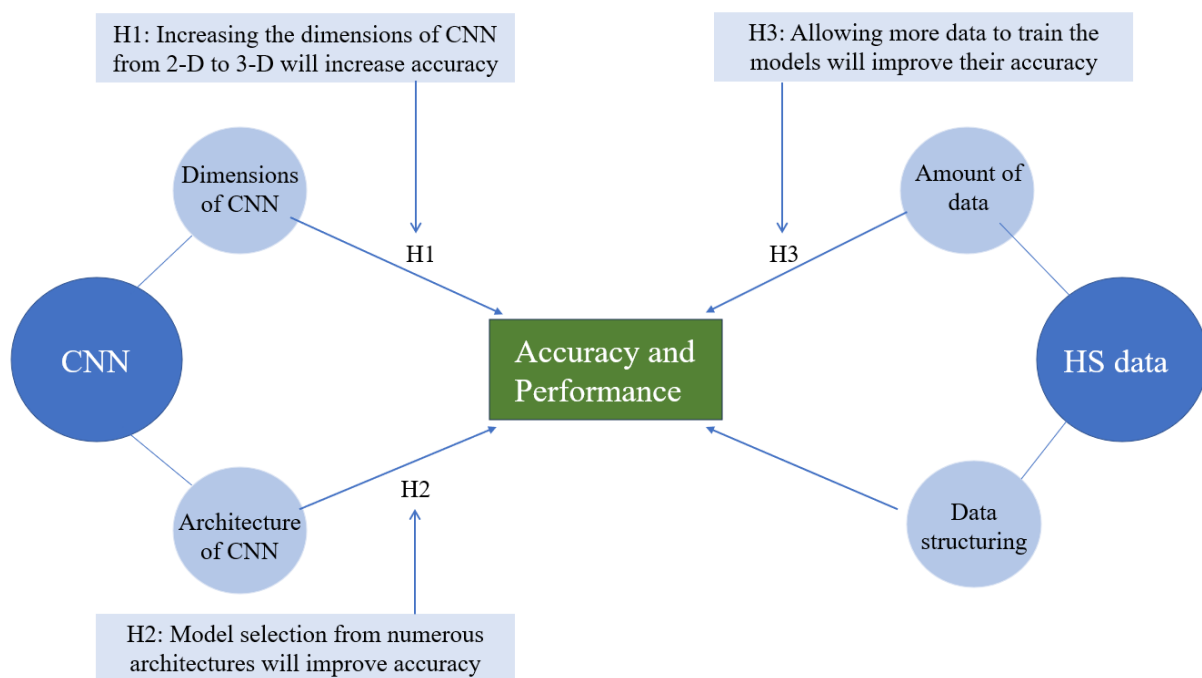


Figure 2.3 Conceptual model of the study. Performance of CNN for ore grade estimation is influenced by the dimensions and architecture of CNN, and the amount and structuring of the data. Therefore, the hypotheses are based on the conditions of HS data and CNN architecture.

2.5.1 Hypothesis 1: Relationship of Dimensions with Performance of CNN

Increasing the dimensions of CNN from 2D to 3D allows the extraction of both spectral and spatial features in image data, as described in section 2.1.2. This increased the accuracy of classification in the studies by Chen et al. (2016), Praveen & Menon (2021) and Abubakar et al. (2024). Therefore, it is hypothesized that increasing the dimensions of CNN from 2D to 3D for analysis of HS reflectance data for predictive ore grade estimation improves the performance of the CNN. However, in the case of this study, where 3D convolution are applied to frequency-transformed data, there is extraction of spatial relationship between different core samples.

2.5.2 Hypothesis 2: Relationship of Architecture with Performance of CNN

Based on the conclusion drawn in the review on the influence of architecture of CNN on the performance in section 2.1.1, the optimal 2D and 3D CNN architectures were investigated and selected from a variety of 2D and 3D architectures and these will be used for comparison in this study. Therefore, the performance of CNN improves from 2D to 3D architecture.

Chapter 3 : Olympic Dam Geology and Data Collation

3.1 Olympic Dam Geology and Mineralogy

The hyperspectral (HS) and geochemical data used in this study were from Olympic Dam deposits. The deposit, located at 30°26'24"S 136°52'23"E in Australia, is the largest single deposit of the Olympic Cu-Au Province and one of the largest Cu-U-Au-Ag deposits in the world. It is situated in the northeastern part of the Gawler Silicic Large Igneous Province (SLIP) which is at the eastern margin of the Mesoproterozoic to Archean Gawler craton, as shown in Figure 3.1. It is hosted by a hydrothermal breccia complex referred to as the Olympic Dam Breccia Complex (ODBC). The breccia complex is surrounded by the Mesoproterozoic Roxby Downs Granite (RDG) from which it was derived (Reeves, 1990). The RDG is part of the Hiltaba Granite Suite, which forms the Gawler SLIP with the Gawler Range Volcanics (GRV). Rocks of this GRV have been found in the ODBC. The breccia complex is overlain by very thick sections of sedimentary facies of laminated mudstones, internally graded sandstones and conglomerates, which may cover an area of up to $\sim 1.5 \text{ km} \times 0.9 \text{ km}$ (McPhie et al., 2011). These sedimentary facies are unconformably overlain by Neoproterozoic to Cambrian sedimentary rocks of the Stuart shelf which overlie the RDG and ODBC.

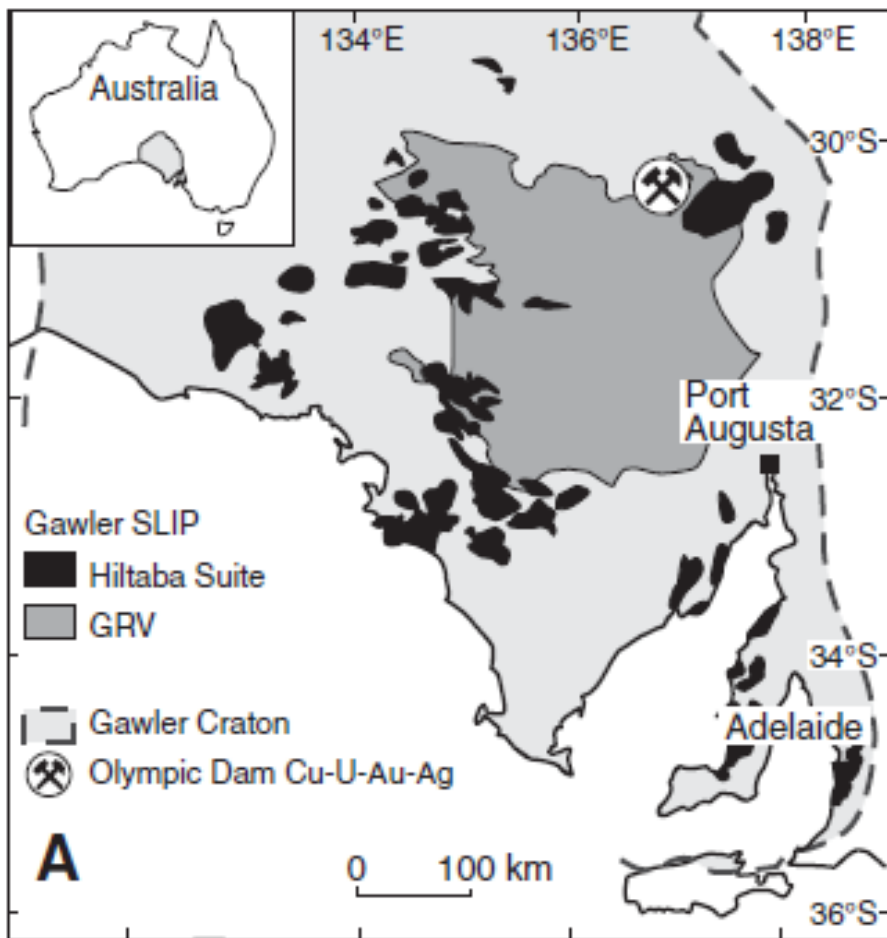


Figure 3.1 Geological map of the eastern margin of the Gawler craton showing the location of Olympic Dam deposit. The deposit, which is shown by the mine symbol on the map, is located in the Hiltaba Granite Suite of the Gawler silicic large igneous province (map from McPhie et al., 2011).

The lithology and mineralogy of the ODBC varies from the outer margins towards the centre of the deposit (Ehrig & Kamenetsky, 2012). The outer margins of the ODBC are characterized by the replacement of biotite with chlorite from the RDG towards the centre, as shown by Figure 3.2. Near the outer margins, occurs the granite-rich breccias, whose hydrothermal alteration assemblage is defined by a reduced Fe oxide mineralogy. The assemblage comprises magnetite, apatite, siderite, chlorite and quartz. Sodic plagioclase and orthoclase are altered to sericite from the outer margins, and the major sulphides are pyrite and chalcopyrite. Towards the complex centre, the Fe oxide is oxidised to form hematite-rich breccias. The hematite is also formed from sericite alteration of the granite-rich breccias. The hematite-rich breccias

have an alteration assemblage comprising hematite, sericite and fluorite, with chalcopyrite, bornite and chalcocite as the major sulphides. This is the main Cu mineralisation, which is also associated with Au mineralisation in the deposit. At the centre of the ODBC, there is a distinct alteration assemblage of hematite, barite and quartz which is defined by lack of mineralisation.

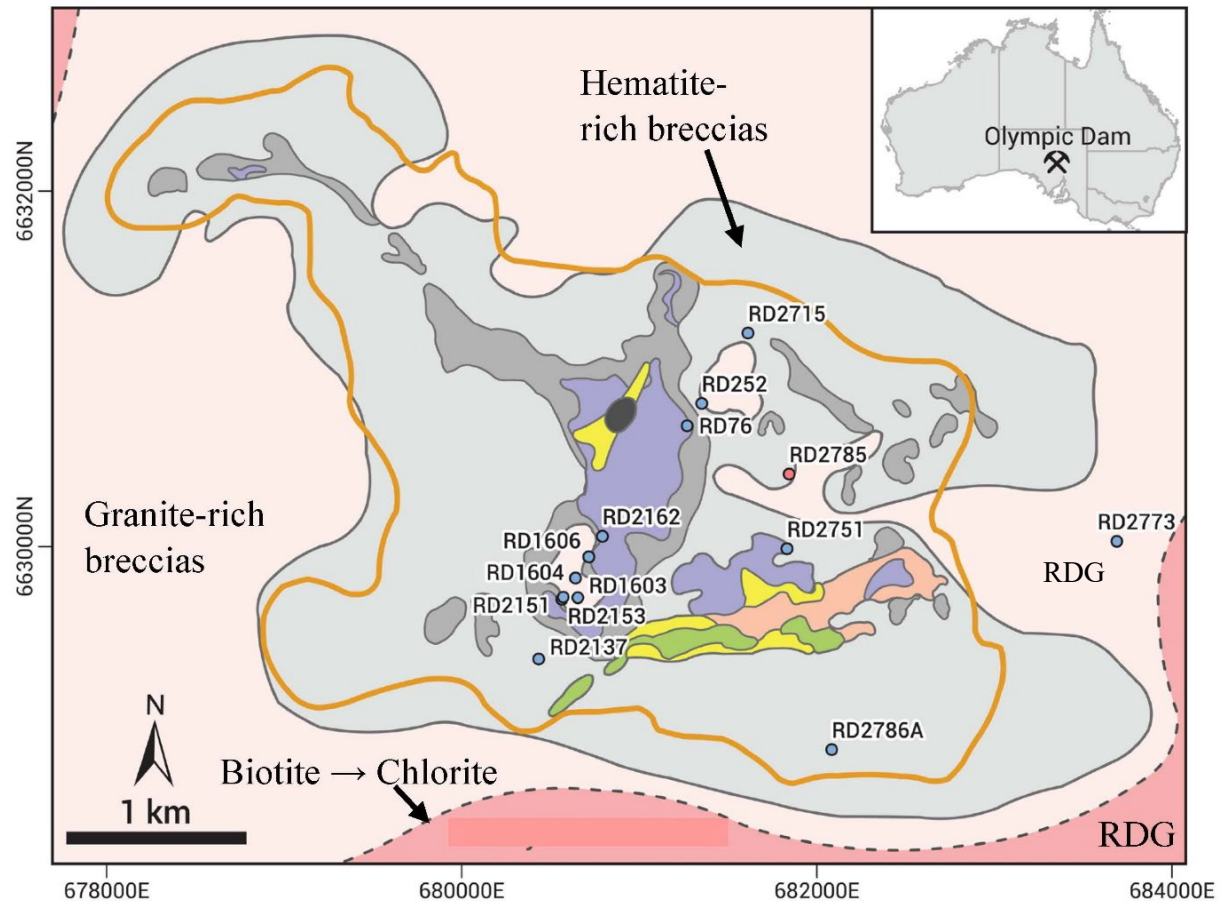


Figure 3.2 Simplified geological map of the Olympic Dam breccia complex showing the lithology and mineralogy of the complex and deposit. Au mineralisation is associated with Cu mineralisation in the deposit. The locations of the drill cores from which the data that were used in this study were obtained are shown on the map (map modified from Prado et al. (2023))

3.2 Data Collation

The data from drill cores (shown in Figure 3.2) that were used in this study were obtained from the AuScope National Virtual Core Library (NVCL) and the South Australian Resources Information Gateway (SARIG). These data include HS data, which were acquired in the VNIR–SWIR (380–2500 nm) and TIR (6000–14500 nm) spectral bands, although only data

from the VNIR-SWIR spectral bands were used for the study. The HS data were acquired using the Commonwealth Scientific Industrial and Research Organisation (CSIRO) Hylogger-3 spectrometer, which was positioned at 8-mm intervals along a drill core to obtain point-measured HS data. The field of view of the spectrometer was 10 mm along the diameter of the core and 18 mm along the drill core's extension.

The data obtained from the SARIG also included the geochemical content of 29 elements that were analysed in samples from the drill cores. However, only data for Au were used for this study. The Au ore grade was acquired using fire assay. These ore grades and the VNIR–SWIR spectral data of the intervals whose grades were acquired were used to train, validate and test the CNN models.

Chapter 4 : Data Pre-processing and Model Architecture Selection

This study investigated how increasing the dimensions of a convolutional neural network (CNN) from 2D (two-dimensional) to 3D (three-dimensional) affects its performance for ore grade prediction from hyperspectral (HS) reflectance data in the frequency domain. Consequently, this aim led to the following research objective:

- To build and train 2D and 3D CNN models using HS reflectance data and compare their performance for ore grade estimation.

This objective has guided the literature review, which led to hypotheses detailed in section 2.5. Therefore, the research methodology that was followed involved a supervised CNN application approach of data collection, cleaning, pre-processing, labelling as shown by the Figure 4.1. Training and evaluation of the CNN models for ore grade estimation was also done as shown by the figure. The parts of the workflow in which the research hypotheses of this study are involved are also shown.

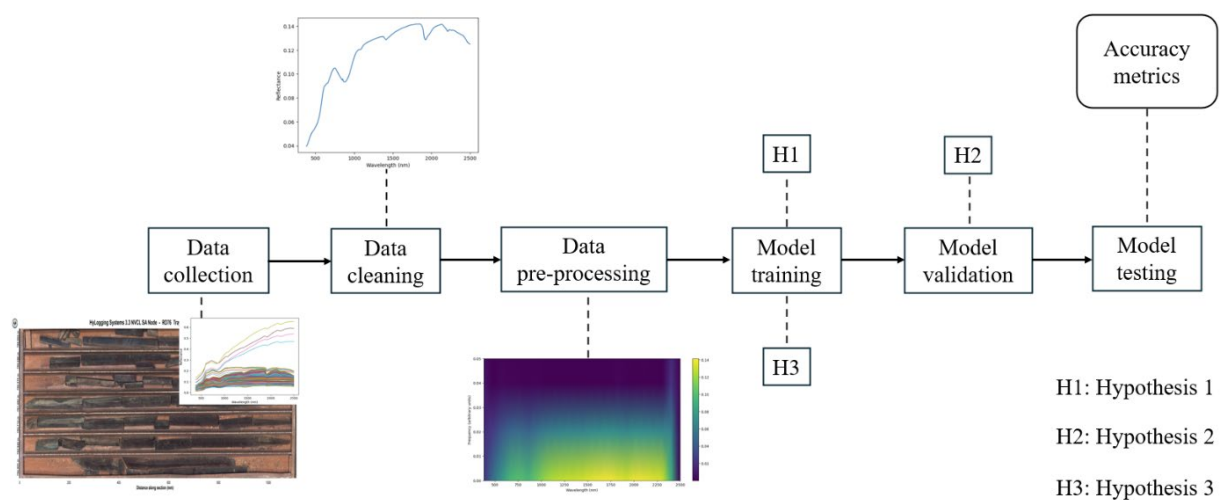


Figure 4.1 Ore grade estimation workflow using deep learning (DL) algorithms

The focus of this study was on model training and evaluation, as highlighted by the positions of the research hypotheses in the ore grade estimation workflow (Figure 4.1). Therefore, the main parts of the workflow for this study are shown by Figure 4.2.

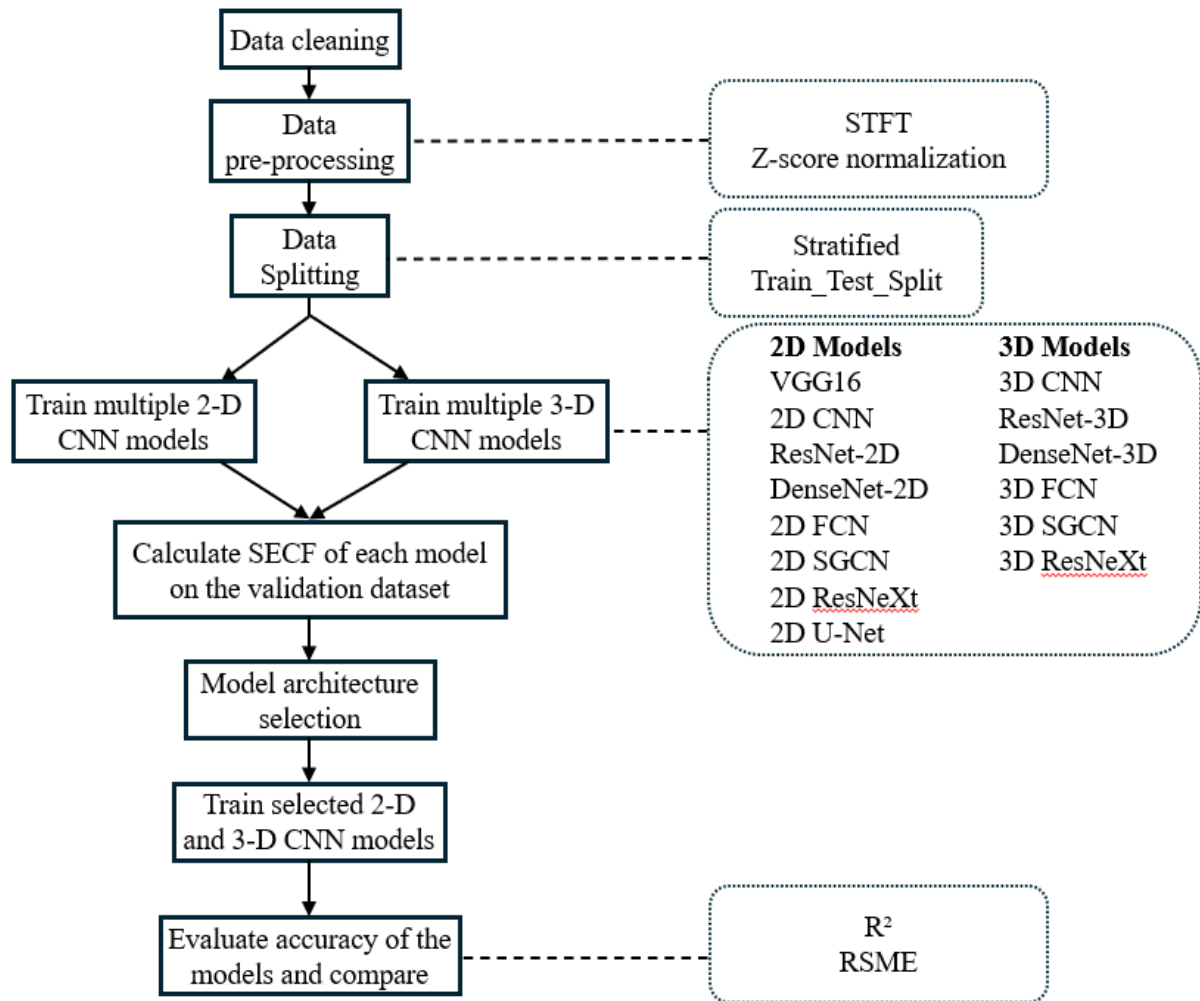


Figure 4.2 Study workflow showing the steps from data cleaning to model evaluation using accuracy metrics (STFT: Short-Time Fourier Transform, VGG16: Visual Geometry Group 16, ResNet: Residual Networks, DenseNet: Densely Connected Convolutional Networks, FCN: Fully Convolutional Networks, SGCN: Spectral Graph Convolutional Networks, ResNeXt: Residual Networks with Aggregated Transformations, SECF: Squared error cost function, R²: Coefficient of determination, RMSE: Root mean square error)

4.1 Data Cleaning

The HS and geochemical data of seven drill cores (RD76, RD2162, RD252, RD2137, RD2153, RD2751, RD2773) that were collated from the AuScope National Virtual Core Library and the

South Australian Resources Information Gateway websites as described in section 3.2 were used in this study. The HS data were obtained as reflectance spectra encompassing the intervals for which samples were analysed for geochemistry. The geochemical assay samples had been acquired at 1m for 90% of the samples, 2m for 2% of the samples and 5m for 8% of the samples. Therefore, the average reflectance spectra for these intervals were calculated, and so that there was only one reflectance spectrum for each geochemical label of the intervals. Approximately 200 spectra were averaged for each interval. Reflectance spectra for the 300–302 m interval of drill core RD76 is shown by Figure 4.3 as example, and the resulting average reflectance spectrum that was calculated from these spectra is also shown in the figure. The Python codes for loading the multiple reflectance spectra and calculating the average reflectance spectra are shown in Appendices A and B, respectively.

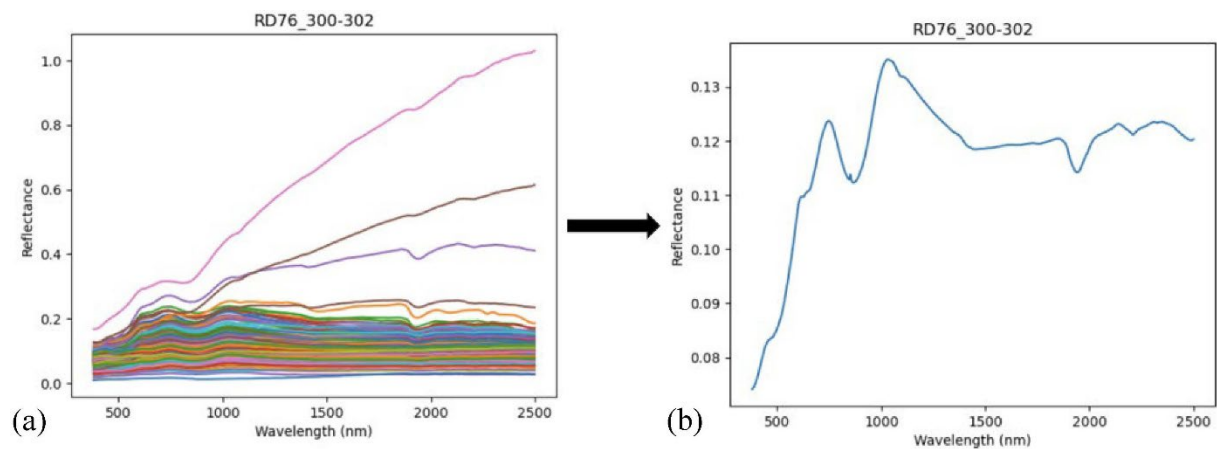


Figure 4.3 (a) Reflectance spectra of the 300–302 m interval of the RD76 drill core and (b) average reflectance spectra for this interval.

Averaging spectra across large depth intervals has the effect of smoothing sharp absorption features and diminishing reflectance and absorption features that occur at the same wavelength. This results in average reflectance spectra that are not diagnostic for the minerals within the rocks from which the spectra are acquired. Therefore, upon calculations of average reflectance

spectra for all the intervals, the standard deviation of the reflectance spectra from which the average reflectance spectra were calculated, were plotted around the average spectra to determine if the original spectra had absorption or reflectance features that were diminished by averaging. Figure 4.4 shows the average reflectance spectrum for the 300-302m interval of drill core RD76 and the standard deviation of original reflectance spectra from which the spectrum was calculated. The figure shows that although the original spectra deviate from the average spectrum by reflectance values, the resulting absorption and reflectance features in the average spectrum occur at the same wavelength as the features appear in the standard deviation of the original spectra. Therefore, this shows that the average spectrum had the diagnostic absorption and reflectance features as they are present in the original spectra, so averaging the spectra across the intervals was a viable approach to obtain only one reflectance spectrum for each geochemical label of the intervals.

Upon calculation of average reflectance spectra, the spectra that had reflectance values greater than one or less than zero were discarded as these were erroneous spectra collected by the Hylogger-3 spectrometer. After data cleaning, 4034 reflectance spectra from the seven drill cores were retained for analysis.

4.2 Data Pre-processing

The average reflectance spectra were transformed into 2D spectrogram plots for 2D CNN and, further, into 3D spectrogram plots for 3D CNN. The 2D and 3D spectrogram plots of the 300–302 m interval of the RD76 drill core are shown in Figure 4.5 as examples.

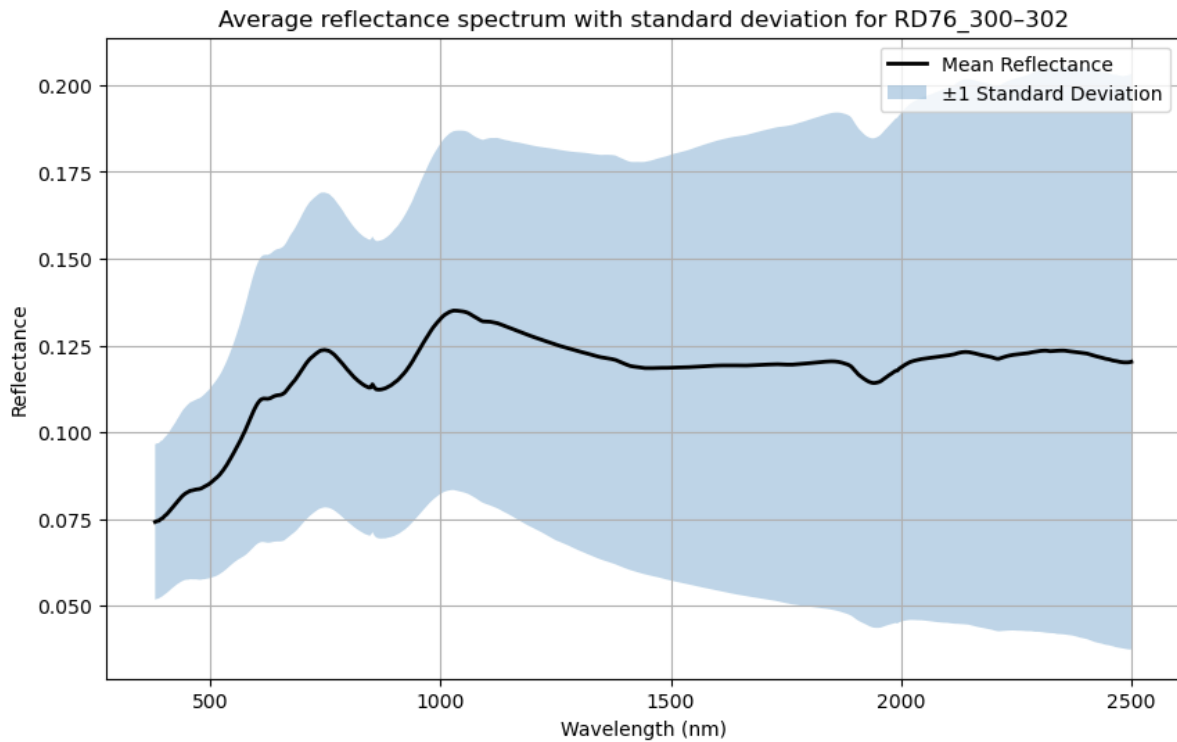


Figure 4.4 Average reflectance spectrum for depth interval 300-302m of drill core RD76 and the standard deviation of the original reflectance spectra from which the average reflectance spectrum was computed. Diagnostic absorption and reflectance features in the average spectrum occur at the same wavelength as those occurring in the standard deviation.

The 2D and 3D spectrograms were calculated using the short-time Fourier transform (STFT) of the average reflectance spectra. The STFT was implemented in code from SciPy library, and the implementation for both 2D and 3D are shown in Appendices C and D. The STFT for calculating both the 2D and 3D spectrograms was computed using the Hann window with a window size of 50 samples and 50% overlap between adjacent windows. The FFT length was set to 50 to match the window size. A sampling frequency of 1 arbitrary unit was set since the data are defined along wavelength rather than over time. In the process, the average reflectance spectra were transformed from a spectral domain to a frequency domain. The 2D spectrogram plots are image-like frequency vs wavelength plots with colour-scaled amplitude and the 3D spectrogram plots are 3D mesh grids of the 2D spectrogram plots with the amplitude expressed as the third dimension of the spectrogram plots. These 3D mesh grids increase the tensor of the

spectrogram plots from a 2D tensor shape of (batch_size, wavelength_bins, frequency_bins, channels) to a tensor shape of (batch_size, wavelength_bins, frequency_bins, depth, channels), making 3D spectrograms volumetric data for the application of 3D CNNs, which cannot be applied to 2D spectrogram plots.

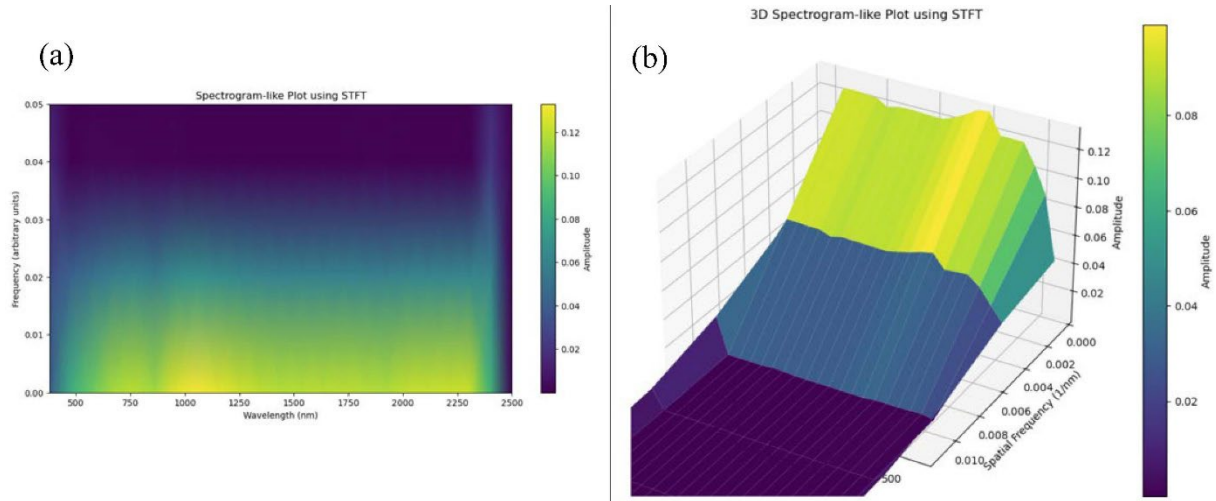


Figure 4.5 Average reflectance spectra of the 300–302 m interval of the RD76 drill core. (a) 2D spectrogram plots; (b) 3D spectrogram plot

After the STFT of the average reflectance spectra were calculated, the 2D and 3D spectrograms were normalized using min-max normalization. This was done to remove any variations that existed in the reflectance spectra data due to the acquisition and calculation of average reflectance spectra for reflectance spectra obtained from samples of different sizes. The different physical dimensions of the samples, different illumination condition during acquisition, temperature effects of different acquisition times and sensor noise are the factors that could have led to variations in reflectance spectra in these data. Therefore, the normalization was applied to stabilize the CNN models by scaling the spectrogram amplitudes, which can differ by orders of magnitude, to the same range. The normalization would ensure that the large magnitude inputs would not dominate the weight updates to stabilize the models.

The min–max normalization, given by equation 4.1, was implemented in python codes shown in Appendices C and D.

$$X' = \frac{X - \min(X)}{\max(X) - \min(X)} \quad (\text{Equation 4.1})$$

4.3 Data Splitting

Splitting of the data into training, validation and testing data was done using python Scikit-Learn's train test split code. All data from the seven drill cores were collectively split into 60% training data, 20% validation data and 20% testing data. In the code, a stratification argument was also used to split the data such that all grade classes of the data were represented in all the three datasets. This was to ensure that the resulting datasets all had widely ranging data that would prevent bias of the models towards the majority ore grade classes. The stratify argument does not allow splitting of the data if there are data samples with less than two samples in the class; therefore, these data samples were discarded. This resulted in an overall dataset of 3852 samples from which stratified splitting was done. Consequently, 2338 samples for training the models, 757 samples for validation and 757 for testing the models resulted from the data splitting. These division percentages remained constant for both 2D and 3D CNN and the python code for the splitting is shown in Appendix E.

4.4 2D and 3D CNN model architecture selection

Different 2D and 3D CNN model architectures were trained and validated using split data to select the best architecture per dimension. Training data were used to train each architecture as described in sections 4.4.1 and 4.4.2. All the models were trained by regression. The architectures were then validated by determining the squared error cost function (SECF) of

each architecture on the validation dataset. The SECF, shown by equation 4.2, calculates the average sum of the squared differences between the predicted and actual values of a dataset and determines how well the model does on the dataset.

$$SECF = \frac{1}{2N} \sum_{i=1}^N (\hat{y}(i) - y(i))^2 \quad (\text{Equation 4.2})$$

The higher the SECF, the lower the performance of the model on the dataset. Therefore, this calculation was used to determine how each model would perform on the testing dataset, based on the value calculated from the validation dataset. For this reason, this SECF was used to determine the best architecture for the study, and the best architecture would be the one with the lowest SECF.

4.4.1 2D CNN Model Architecture Selection

The 2D spectrograms that were split into 2338 training data samples were used to train eight 2D CNN architectures: VGG16, a standard 2D CNN, ResNet-2D, DenseNet-2D, 2D fully convolutional network (FCN), 2D spectral graph convolutional network (SGCN), 2D ResNeXt and 2D U-Net. These architectures are shown by Figures 4.6 to 4.13. The Python codes that were used to code for the creation of these model architectures are shown in Appendices F to M.

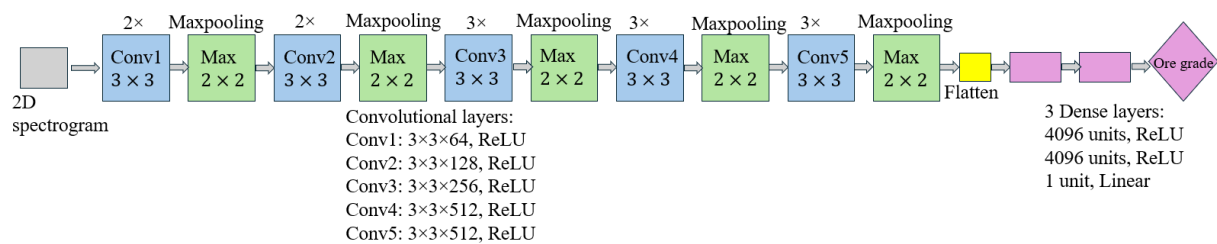


Figure 4.6 Model architecture for VGG16 2D CNN that was trained in this study. The training hyperparameter for this model were batch size =32, epoch = 50, optimizer = adam

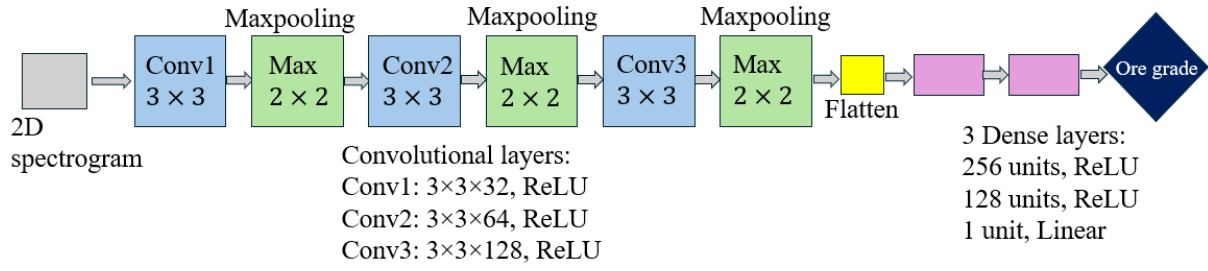


Figure 4.7 Model architecture for a standard 2D CNN that was trained in this study. The training hyperparameter for this model were batch size =32, epoch = 50, optimizer = adam

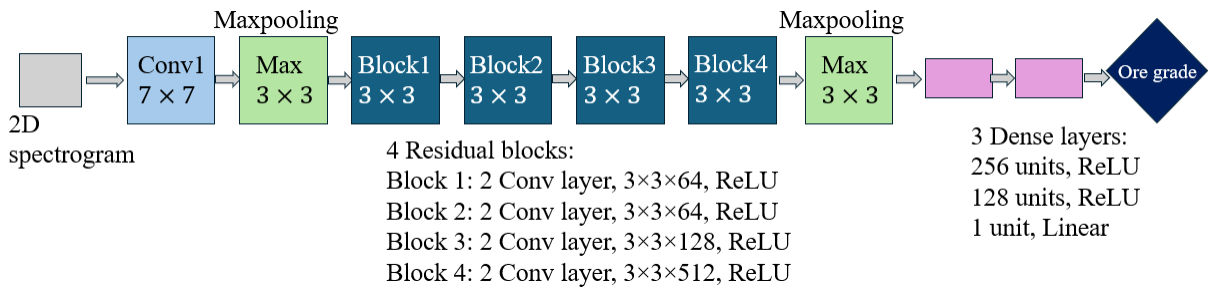


Figure 4.8 Model architecture for a ResNet-2D that was trained in this study. The training hyperparameter for this model were batch size =32, epoch = 50, optimizer = adam

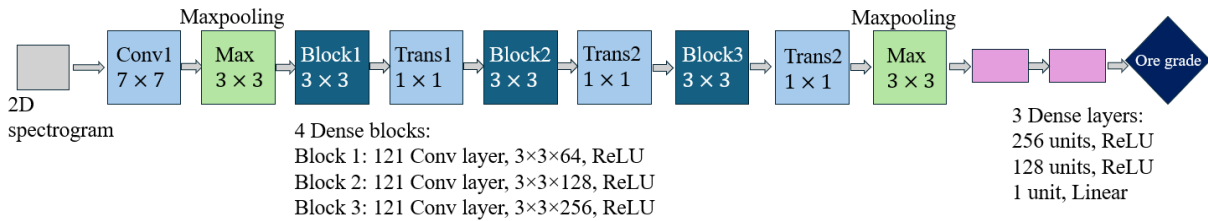


Figure 4.9 Model architecture for a DenseNet-2D that was trained in this study. The training hyperparameter for this model were batch size =32, epoch = 50, optimizer = adam

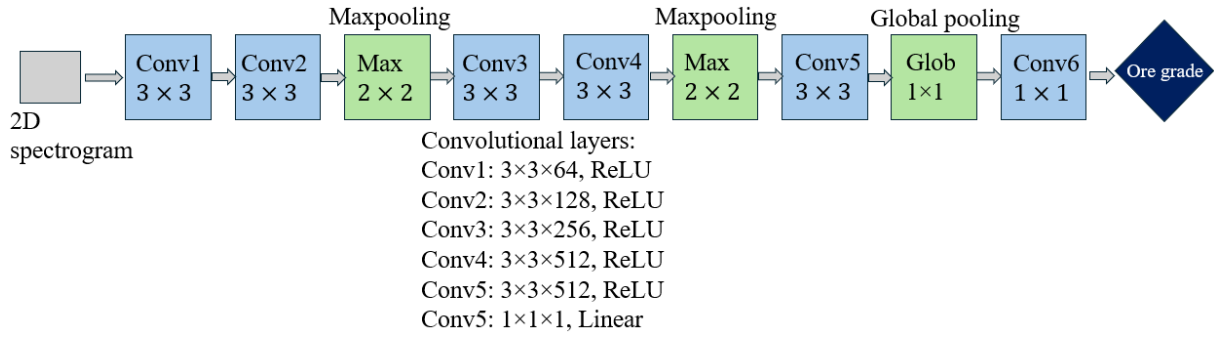


Figure 4.10 Model architecture for a 2D FCN that was trained in this study. The training hyperparameter for this model were batch size =32, epoch = 50, optimizer = adam

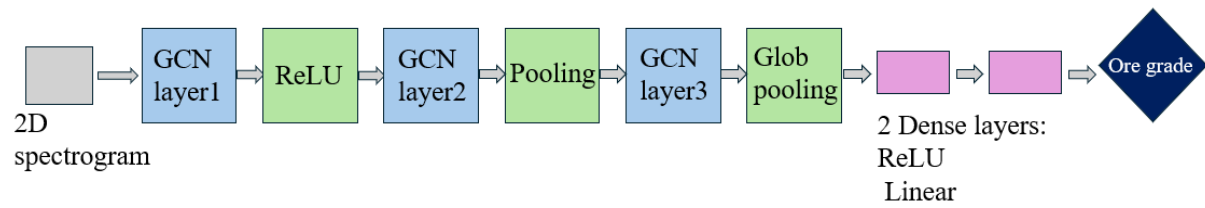


Figure 4.11 Model architecture for a 2D SGCN that was trained in this study. The training hyperparameter for this model were batch size =32, epoch = 50, optimizer = adam

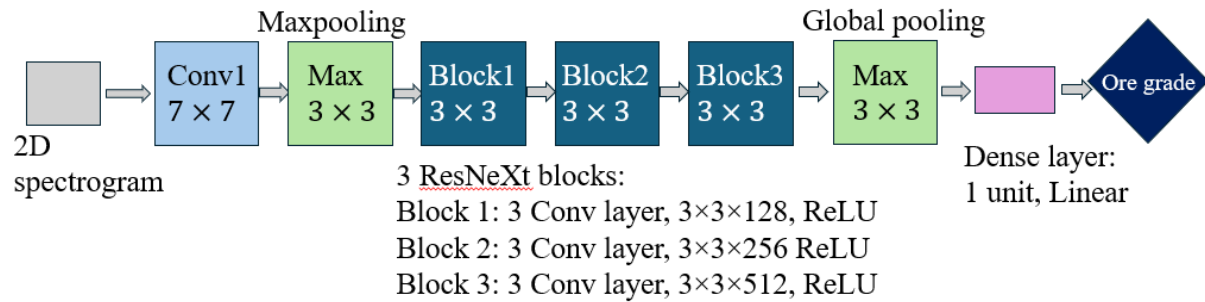


Figure 4.12 Model architecture for a 2D ResNeXt that was trained in this study. The training hyperparameter for this model were batch size =32, epoch = 50, optimizer = adam

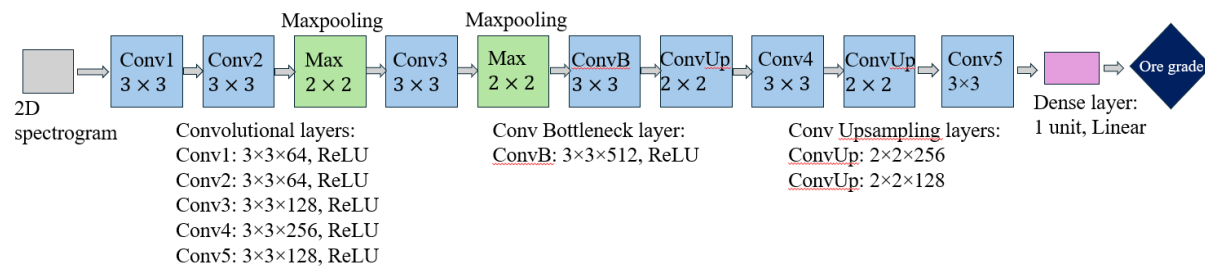


Figure 4.13 Model architecture for a 2D U-Net that was trained in this study. The training hyperparameter for this model were batch size =32, epoch = 50, optimizer = adam

The calculated SECF values for the 2D CNN model architectures are shown in Table 4.1. Based on these values, the DenseNet-2D was chosen as the 2D model architecture for the remainder of this study.

Table 4.1 SECF of each 2D model architecture on the validation dataset

2D CNN architecture	Validation dataset SECF (ppm)
VGG16	0.053
Standard 2D CNN	0.053
ResNet-2D	0.052
DenseNet-2D	0.039
2D FCN	0.053
2D SGCN	0.045
2D ResNeXt	0.056
2D U-Net	0.059

4.4.2 3D CNN Model Architecture Selection

Six model architectures for 3D CNN were trained: standard 3D CNN, ResNet-3D, DenseNet-3D, 3D FCN, 3D SGCN and 3D ResNeXt. These are shown in Figures 4.14 to 4.19 and the codes that were used to create and train them are shown in Appendices N to S. The input to each model architecture was the 3D spectrogram plots obtained as described in section 4.3. The spectrogram plots were input as volumetric spectrogram data discretized over wavelength and frequency with amplitude as a voxel intensity across a third dimension. Therefore, the input of these models was a tensor with a shape of (batch_size, wavelength_bins, frequency_bins, depth, channels) = (2338, 26, 21, 3, 1).

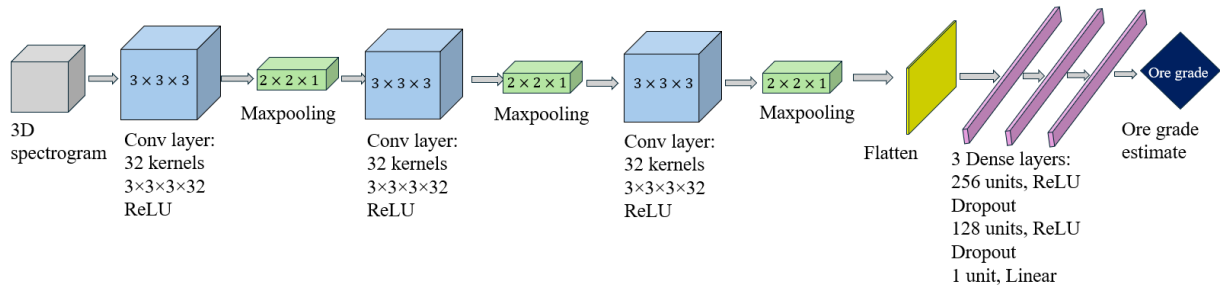


Figure 4.14 Architecture of standard 3D CNN that was trained in this study. The training hyperparameter for this model were batch size =16, epoch = 50, optimizer = adam

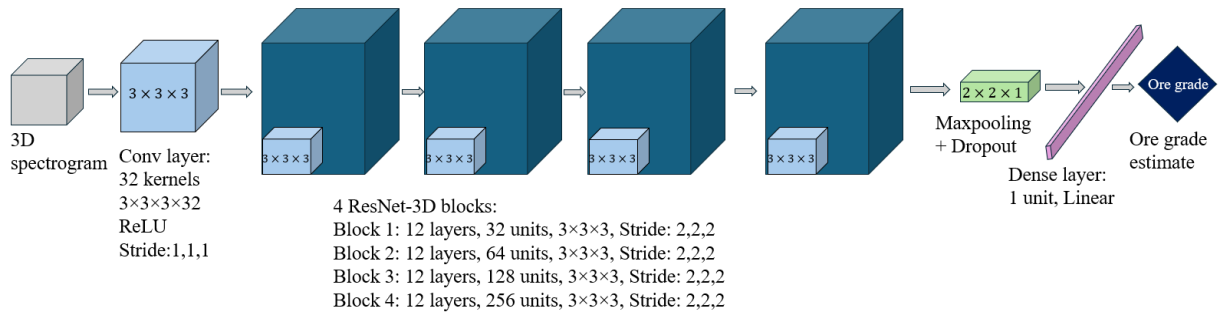


Figure 4.15 Architecture of ResNet-3D model that was trained in this study. The training hyperparameter for this model were batch size =16, epoch = 50, optimizer = adam

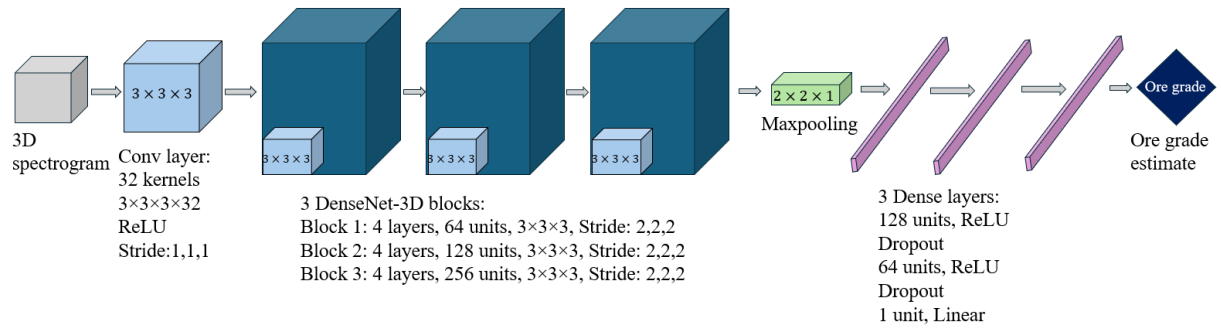


Figure 4.16 Architecture of DenseNet-3D model that was trained in this study. The training hyperparameter for this model were batch size =16, epoch = 50, optimizer = adam

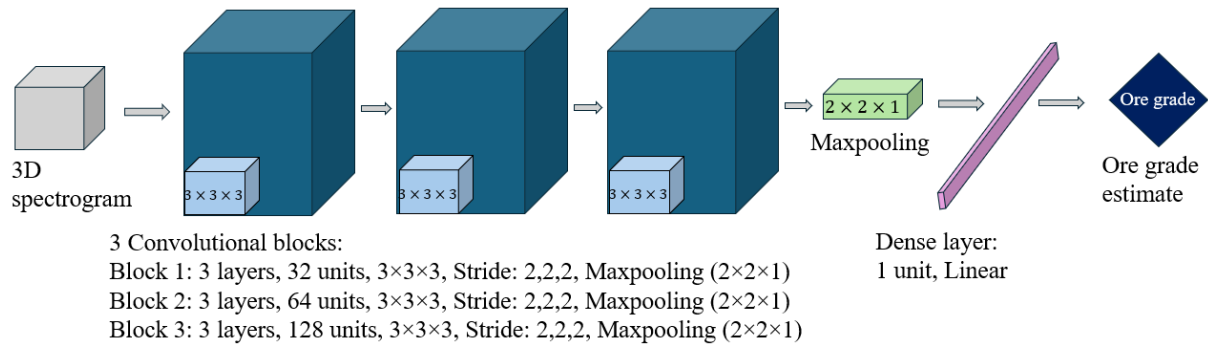


Figure 4.17 Architecture of 3D FCN model that was trained in this study. The training hyperparameter for this model were batch size =16, epoch = 50, optimizer = adam

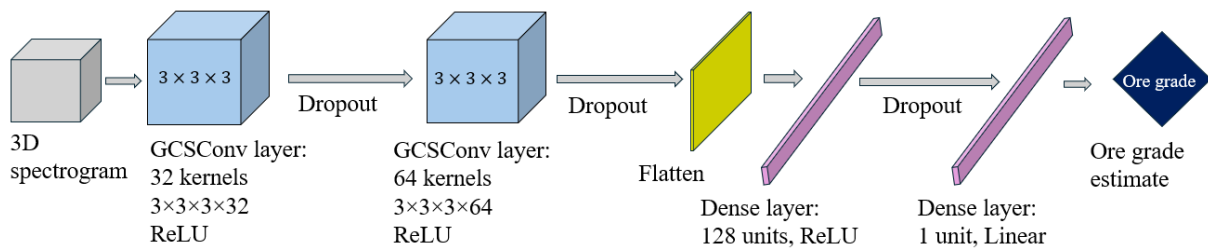


Figure 4.18 Architecture of 3D SGCN model that was trained in this study. The training hyperparameter for this model were batch size =16, epoch = 50, optimizer = adam

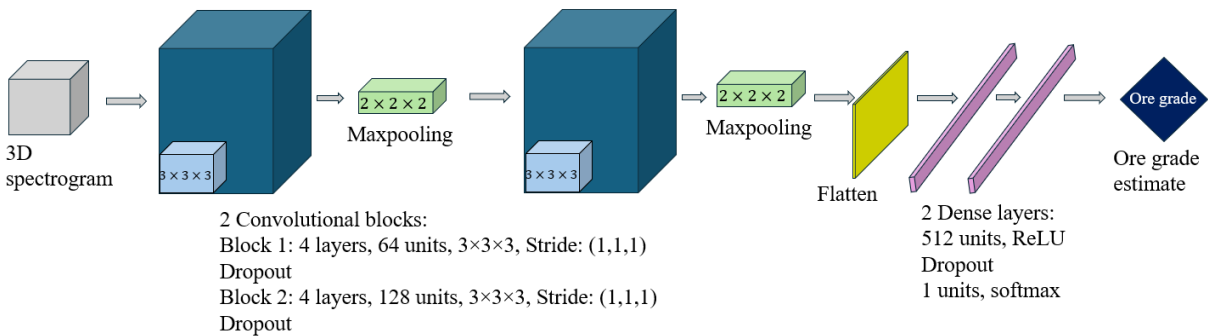


Figure 4.19 Architecture of 3D ResNeXt model that was trained in this study. The training hyperparameter for this model were batch size =16, epoch = 50, optimizer = adam

The calculated SECF values of each of the 3D model architectures are shown in Table 4.2. Based on these values, the ResNet-3D was chosen as the 3D architecture for the remainder of this study.

Table 4.2 SECF values of each 3D model architecture on the validation dataset

3D CNN architecture	Validation dataset SECF (ppm)
Standard 3D CNN	0.030405
ResNet-3D	0.029312
DenseNet-3D	0.030514
3D FCN	0.031078
3D SGCN	0.089430
3D ResNeXt	0.054488

4.5 Conclusion

DenseNet-2D and ResNet-3D were the best 2D and 3D model architectures, respectively, that were selected for comparison of performance of 2D and 3D CNN for HS reflectance data for Au grade estimation. Therefore, the performance of these models was evaluated and compared. The details of this comparison are described in the next chapter.

Chapter 5 : Model Training and Results

5.1 Correlation Analysis

The DenseNet-2D and ResNet-3D models, selected as the best-performing 2D and 3D architectures respectively, were further evaluated to compare their performance for Au grade estimation. However, prior to training and evaluation of these models, dataset correlation was investigated for relationships between features in the reflectance spectra and the Au grade values. This aimed to determine if there were patterns in the features of the spectra that the algorithms learned to make predictions for Au grade during the model architecture selection process or if the predictions that the models made were only by chance. Therefore, to investigate the features of the reflectance spectra that could have been used in making predictions, the Au grade values of the whole dataset were split into classes, and the reflectance spectra of each class were plotted to observe variations in reflectance and absorption features with variations in Au grade values (Prado et al., 2023).

To determine the cut-off grades of the Au grade classes, the distribution and range of the grade values were determined by use of a histogram (Figure 5.1). The histogram showed that the Au grade values had a positive skew and that 3000 (75%) were below 0.625 ppm. Therefore, a scatter plot of the Au grade values below 0.625 ppm was also plotted, (Figure 5.2). The plot showed that the lowest Au grade value of the whole dataset was 0.005 ppm, which is above 0.001 ppm, the high-grade domain-specific cut-off grade for drill core from iron oxide copper-gold (IOCG) deposits and Olympic dam deposit (Zhu, 2016). Therefore, because 75% of the Au grade values of the whole dataset was below 0.625 ppm and because the lowest Au grade value was above the high-grade domain-specific cut-off grade used for IOCGs, the data were

split into quartiles but not according to IOCG and Olympic dam grade benchmarks because Au grade was high in this dataset. Besides, splitting the grade data into quartiles would also ensure that all the Au grade classes would have the same number. Therefore, the splitting of the data into quartiles resulted into quartile class ranges shown in Table 5.1 and Figure 5.2.

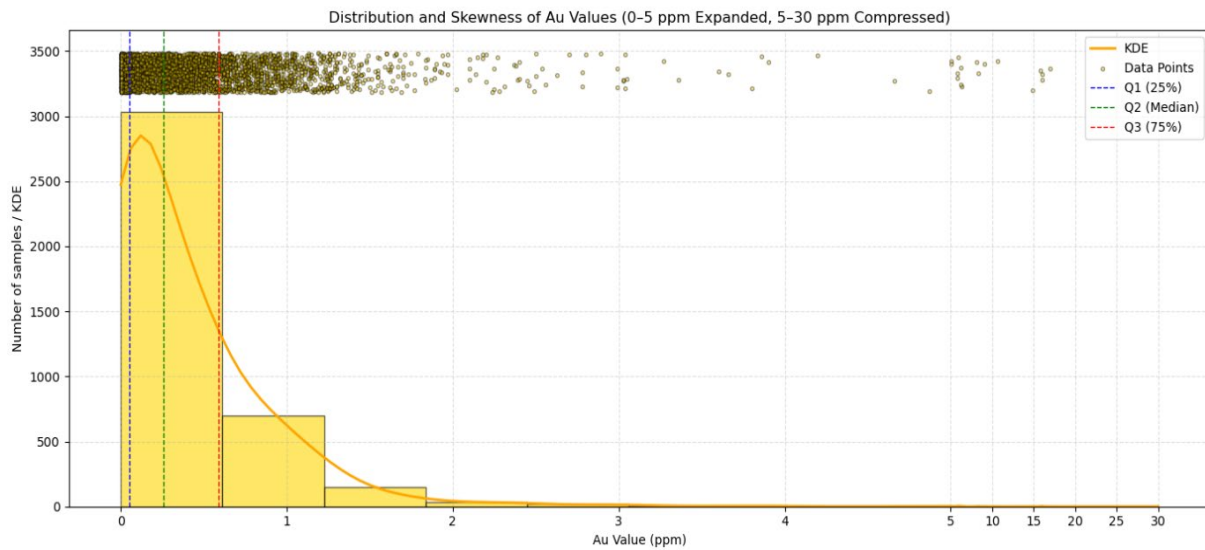


Figure 5.1 Histogram of Au grade values of the data that were used to train, validate and test the models. The data points are also plotted above the histogram to show their skewness. The quartile cutoff grades are shown by the blue, green and red lines.

Reflectance spectra corresponding to Au grade values in each quartile class were plotted to observe trends in the features of the spectra with variation in Au grade (Figure 5.3). In general, reflectance decreases from an average of 0.10 to 0.08 as the Au grade increases from a lower quartile to a higher quartile. This could be expected as Au mineralisation in this deposit is associated with copper sulfides and a host mineralogy of spectrally weak minerals. Therefore, the reflectance would decrease with increasing Au grade.

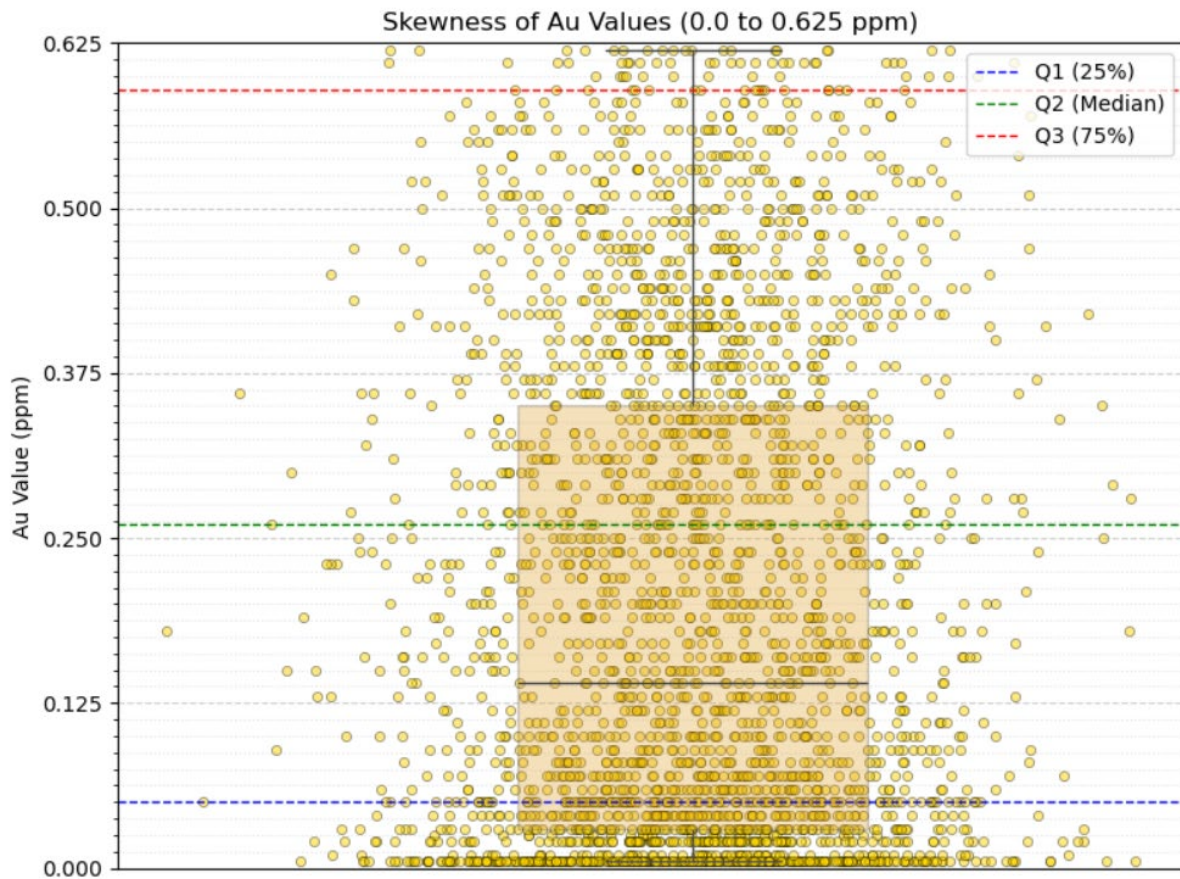


Figure 5.2 Scatter plot of 3000 Au grade data points that were below 0.625 ppm. These points make up 75% of all the Au grade points in the data. The quartile cutoff grades are shown by the blue, green and red lines.

Decrease and increase in some of the spectral features of the samples from the lower quartile to the higher quartiles were also observed. One feature that showed a trend with increasing Au grade values across the quartiles was the broad absorption feature at 800–900 nm shown in Figure 5.3. This feature is characteristic of iron oxide as hematite. Across the quartile classes, it was observed that this absorption feature increased in the spectra as Au grade increased towards the centre of the deposit. This observation correlates with the occurrence of hematite and the association of hematite with Au mineralisation in this deposit. In the deposit, hematite becomes abundant towards the core of the deposit at which high Au mineralisation occurs (Ehrig & Kamenetsky, 2012). The increase in the hematite towards the centre of the deposit is due to replacement of magnetite by the hematite as the magnetite becomes further oxidised.

Therefore, this explains why this 800-900nm absorption feature of hematite increases with increasing Au grade across the quartile classes. Furthermore, hematite replacement of sericite occurs further towards the centre of the deposit, and this also explains why this absorption feature increases with increasing Au grades which are associated with hematite towards the centre of the deposit.

Table 5. 1 Table of quartiles and their corresponding Au grade ranges that data was split into

Quartiles	Quartile Au grade range
Q1	0.005 – 0.05 ppm
Q2	0.05 – 0.26 ppm
Q3	0.26 – 0.59 ppm
Q4	0.59 – 30 ppm

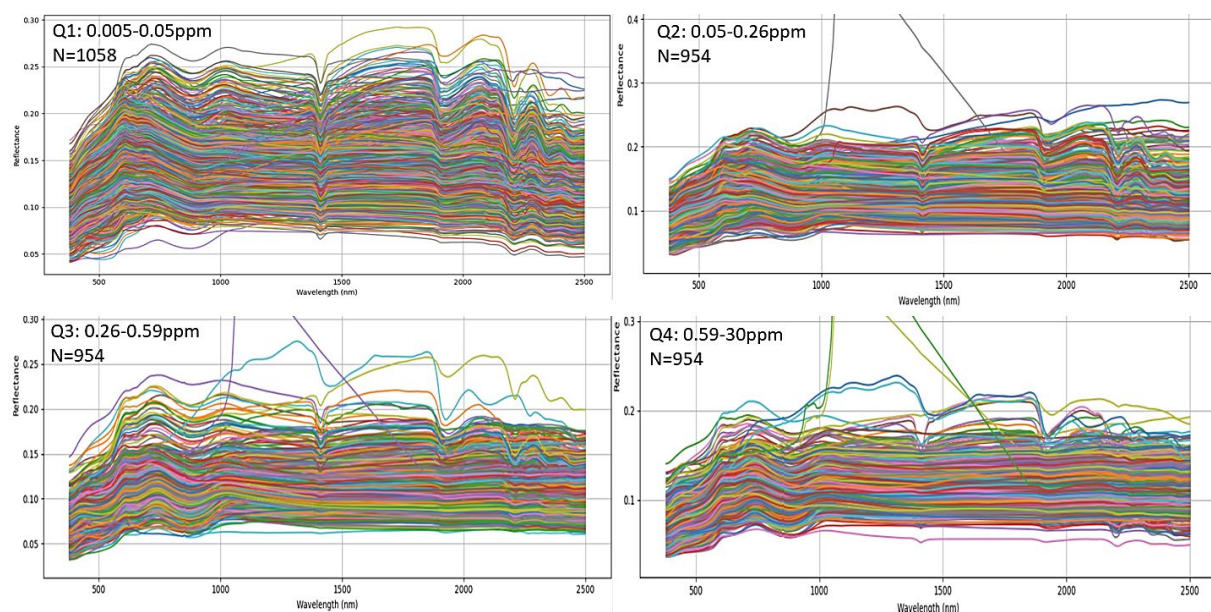


Figure 5.3 Reflectance spectra of corresponding quartiles (Q1, Q2, Q3 and Q4) that the data were split into. The quartile ranges and number of samples in the quartile are shown next to each quartile number on the upper left.

A prominent feature that also shows a change as Au grade increased was the feature at 1400 nm. This spectral feature is caused by OH bonds present in hydrous minerals including sericite, and it decreases with increasing Au grades across the quartile classes as shown in Figure 5.3.

The decreasing trend of this absorption feature from the lowest quartile to the highest quartile correlates with the occurrence of sericite and Au mineralisation in the deposit. Sericite is abundant at the centre of the deposit and is associated with Au mineralisation (Reynolds, 2000). However, hematite alteration of the sericite occurs towards the deposit's core and causes its reduction (Reynolds, 2000). Therefore, this explains why this absorption feature was more pronounced in the lower quartiles of Au mineralisation and gradually disappears with increasing Au mineralisation towards the centre of the deposit. The diagnostic sericite features at 1900 nm (H_2O) and 2200 nm (Al-OH) show the same decreasing trend, confirming the progressive replacement of hydroxyl-bearing sericite by hematite in high-grade zones.

These increasing and decreasing trends in the absorption and reflectance features of the reflectance spectra with increasing Au grade in the quartile classes provide evidence for correlation of spectral features with Au grade in the dataset. Therefore, it is deduced that there are patterns in the reflectance data that the algorithms can learn and use to make predictions for new data.

However, most of the features that were observed to change with increasing Au grade seem to decrease with increasing Au grade values. Therefore, these spectral features were enhanced using continuum removal to make them more pronounced for the algorithms to capture them and use them for prediction, especially for those associated with higher Au grades. The resulting continuum removed reflectance spectra of each quartile are shown in Figure 5.4.

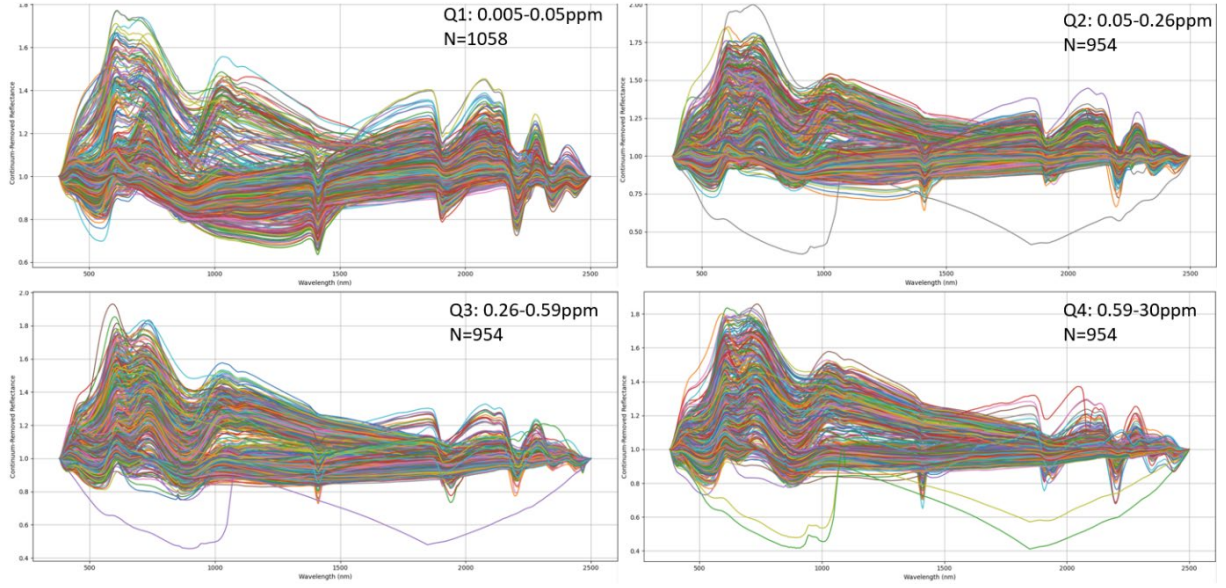


Figure 5.4 Continuum removed reflectance spectra of corresponding quartiles (Q1, Q2, Q3, Q4) that the data were split into. The quartile ranges and number of samples in the quartile are shown next to each quartile number on the upper right.

5.2 DenseNet-2D and ResNet-3D Model Training, Results and Comparison

5.2.1 DenseNet-2D and ResNet-3D Model Training and Results

Continuum removed 2D and 3D spectrograms were used to train the DenseNet-2D and ResNet-3D models that were selected for this study. However, the effect of this continuum removal of the spectra on the accuracy of Au grade estimation for these models was also investigated by comparison of accuracies determined from original reflectance spectra without continuum removal. The continuum was removed from the average spectra that were obtained from data cleaning described in section 4.1 using python code shown in appendix T. Continuum removal divides reflectance at a wavelength by continuum reflectance at that specific wavelength, thus:

$$R_{CR} = \frac{R(\lambda)}{C(\lambda)}. \quad (\text{Equation 5.1})$$

where R_{CR} is the continuum removed reflectance, and $R(\lambda)$ and $C(\lambda)$ are the reflectance and continuum reflectance at a specific wavelength, respectively. After continuum removal, the

spectra were converted to 2D and 3D spectrograms and normalized using min–max normalization as described in section 4.2. The 2D and 3D continuum removed and original spectrograms were then split into 60% training, 20% validation and 20% testing data for each of the datasets using the stratified argument of train-test described in section 4.3. The data were split into these proportions to ensure that there was more data for training the models to make them robust. The training data for continuum removed and original 2D and 3D spectrograms were used to train DenseNet-2D and ResNet-3D models with the same hyperparameters and parameters used to train the model in model architecture selection described in section 4.4. The validation datasets were used to tune the hyperparameters. The accuracy of Au grade estimation of the models was determined by making predictions for the testing data and calculating root mean squared error (RMSE). The results of these tests are shown in Table 5.2.

The original 3D spectrograms model achieved an RMSE of 0.651 ppm, which was higher than 0.342 ppm that the model achieved during model architecture selection without normalization of the reflectance spectra. Therefore, another ResNet-3D model was trained without normalization of the reflectance spectra using the same hyperparameters and parameters as those used with normalized spectra and this model achieved a RMSE of 0.238 ppm. For comparison of this non-normalization on 2D and 3D CNN, another DenseNet-2D model was trained without normalization of the reflectance spectra using the same hyperparameters and parameters and was also tested. The results of these training and testing are also shown in Table 5.2 and their comparisons are explored in the next section.

5.2.2 Comparison of the Accuracies of Prediction of DenseNet-2D and ResNet-3D

DenseNet-2D and ResNet-3D achieved higher performance (i.e., lower RMSEs) for original spectrograms than for continuum removed and normalized spectrograms (Table 5.2). However, for original spectrograms, ResNet-3D had a higher performance than DenseNet-2D model.

Table 5. 2 Table of RMSE of testing data for DenseNet-2D and ResNet-3D

DenseNet-2D RMSE (ppm)			ResNet-3D RMSE (ppm)		
Continuum removed 2D spectrograms	Normalized 2D spectrograms	Non-normalized 2D spectrograms	Continuum removed 3D spectrograms	Normalized 3D spectrograms	Non-normalized 3D spectrograms
0.875	0.738	0.531	0.567	0.651	0.238

Although ResNet-3D achieved a higher performance with a RMSE of 0.238 than all models, this RMSE was still high for a dataset that has a range of 0.000–0.625 ppm for 75% of all Au grade values because if this error was to be subtracted or added to the highest grade value of this range, as shown by equations 5.2 and 5.3, 0.387 ppm would be the predicted Au grade value for the sample and this would under-estimate the Au grade value.

$$0.625 \text{ ppm} - 0.24 \text{ ppm} = 0.387 \text{ ppm} \quad (\text{Equation 5.2})$$

$$0.625 \text{ ppm} + 0.24 \text{ ppm} = 0.865 \text{ ppm} \quad (\text{Equation 5.3})$$

Therefore, this RMSE was too large for the dataset range. Thus, the ResNet-3D model was trained and evaluated on the quartile datasets to determine if this high RMSE was not resulting from the subtle features in the spectral data of the highest Au grade data because it was observed in correlation analysis (section 5.1) that the absorption and reflectance features of these reflectance spectra were not pronounced. These subtle features would not be captured and learned by the models and would cause a high RMSE, especially because these quartiles have high grade values which would result in large differences in predicted and actual grade values

and the RMSE is more sensitive to the large differences of the dataset as the differences are squared before the application of the square root.

The ResNet-3D model achieved a RMSEs of 0.0161 ppm on the Q1 testing dataset, 0.067 ppm on the Q2 testing dataset, 0.091 ppm on the Q3 testing dataset and 0.404 ppm on the Q4 testing dataset. It was evident that the RMSE increased with increasing Au grade across the quartiles. Therefore, this verified that the RMSE of 0.238 ppm that the ResNet-3D model achieved on the representative grade dataset, in which all quartile classes were represented, resulted mostly from the highest grade data in the Q4 range. Therefore, all the model architectures that had been trained and tested in model architecture selection in section 4.4 were trained and tested again on the quartile dataset using the same hyperparameters and parameters to determine if the model accuracy that was achieved by these in model architecture selection also resulted from the highest Au grade data. The RMSEs of all the models on the quartile datasets are shown by Tables 5.3 and 5.4.

Table 5. 3 Table of RMSE of quartile testing datasets for 2D CNN models

	Q1 RMSE (ppm)	Q2 RMSE (ppm)	Q3 RMSE (ppm)	Q4 RMSE (ppm)
ResNet-3D	0.0161	0.067	0.091	0.404
DenseNet-3D	0.0163	0.067	0.092	0.404
Standard 3D CNN	0.0164	0.070	0.092	0.405
3D FCN	0.0164	0.697	0.118	1.722
3D SGCN	0.0267	0.162	0.413	1.069
3D ResNeXt	0.981	0.855	0.599	1.076

Table 5. 4 Table of RMSE of quartile testing datasets for 3D CNN models

	Q1 RMSE (ppm)	Q2 RMSE (ppm)	Q3 RMSE (ppm)	Q4 RMSE (ppm)
DenseNet-2D	0.025	0.063	0.092	2.276
VGG16 CNN	0.015	0.175	0.286	2.270
Standard 2D CNN	0.015	0.062	0.093	2.218
ResNet-2D	0.049	0.118	0.089	2.276
2D FCN	0.016	0.063	0.091	2.270
2D SGCN	0.020	0.068	0.180	2.292

2D ResNeXt	0.019	0.076	0.122	2.207
2D U-Net	0.016	0.063	0.093	2.275

Although the application of the ResNet-3D models and the rest of the CNN models used in this study suggested that the subtle features in the spectra were the cause of the high RMSE of the model to the representative dataset, as evident from the tables 5.3 and 5.4, another factor that could have contributed to the high RMSE is imbalance and scarcity of data in the high Au grade ranges. It is evident from the histogram plotted in figure 5.1 that in the highest Au grade ranges there is sparse density of the Au grades, which suggests that there are less data in those ranges for the models to learn from. Therefore, this could have been a possible contributor to the high RMSE of the ResNet-3D model on the representative datasets.

Chapter 6 : Discussion

Thus far in this study, DenseNet-2D and ResNet-3D were selected as the best architectures for the 2D (two-dimensional) and 3D (three-dimensional) spectrogram representations, respectively. They were compared for Au grade estimation as described in the previous chapter. The results of this comparison are discussed in this chapter.

6.1 DenseNet-2D and ResNet-3D Model Accuracies and Performance for Ore Grade Estimation

Although DenseNet-2D and ResNet-3D models with different primary architectures were compared for comparison of 2D and 3D CNN models for ore grade estimation, the difference in their primary architecture design did not bias the results of this study because they were the best primary architectures for the spectrograms at each dimension, which is a fundamental factor that affects the performance of deep learning models. The selection of these models was done through an evaluation of different architectural designs of 2D and 3D CNN models, of which the models were trained at matching model capacities to ensure the same parameter count of the models at each dimension for a fair benchmark. The multiple model architectures included DenseNet and ResNet in both the 2D and 3D CNN model architectures. In 2D, DenseNet-2D achieved the lowest squared error cost function (SECF) among all the 2D CNN architectures on the validation dataset and in 3D, ResNet-3D achieved the lowest SECF among all 3D CNN architectures on the validation dataset. These were selected as the best architectures for those CNN dimensions. Therefore, since these architectural designs were the best among a range of architectural designs that included both DenseNet and ResNet in 2D and 3D, the comparison of the DenseNet-2D and ResNet-3D models for comparison of 2D and 3D

CNN models is not affected by the difference in architectural design. Rather, the difference in performance between these two models was due to the difference in the dimensions of the CNN models.

The ResNet-3D models outperformed the DenseNet-2D models on non-normalized, normalized and continuum removed spectrograms with root mean squared error (RMSE) values of 0.238 ppm, 0.651 ppm and 0.567 ppm, respectively, which were lower than the RMSEs 0.531 ppm, 0.738 ppm and 0.875 ppm, respectively, that the DenseNet-2D models achieved on 2D spectrograms. This difference in performance of ResNet-3D and DenseNet-2D CNN models on spectrogram representations of HS reflectance data for ore grade estimation is comparable to the differences in performance of 3D and 2D CNN models reported in literature about geological mapping and classification of HS images. In geological mapping and classification, 3D CNN models classify HS images with higher accuracies than 2D CNN models because the former extract not only the spatial correlations of features but also the spectral features in 3D of the images, unlike the latter which extract only the spatial correlation of features (Figure 6.1a) (Liu et al., 2021). Therefore, these results here imply that expression of frequency, wavelength and amplitude in 3D spectrograms enabled access of more features for ResNet-3D models than the expression of these components in 2D for DenseNet-2D models. Thus, the ResNet-3D models extract not only the features from the relationship between frequency and wavelength, but also the features from the amplitude dimension of the 3D spectrograms, which were more than the features extracted only in the relationship between frequency and wavelength in 2D by the DenseNet-2D models. In the spectrograms, the spatial features extracted by CNN models in 2D in classification for HS image would be the relationship between frequency and wavelength, as shown by Figures 6.1b and 6.1c, and the

spectral features extracted from HS images in 3D would be the amplitude in the 3D spectrograms, as shown by Figure 6.1c.

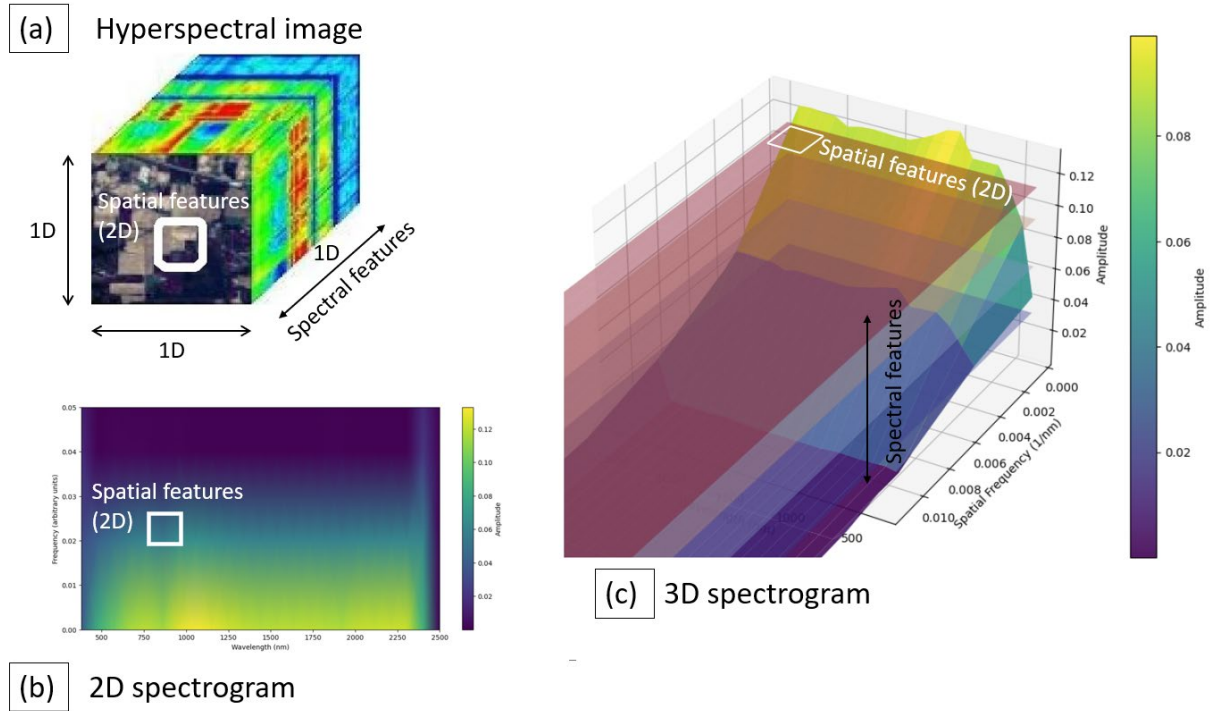


Figure 6.1 Illustration of extraction of spatial and spectral features from HS images and the application of this extraction in 3D spectrogram. (a) Hyperspectral image showing the extraction of spatial features in 2D and the extraction of spectral features in 3D. This extraction collectively makes for extraction in 3D. (b) 2D spectrogram representations of reflectance data used in this study showing the extraction of spatial features from frequency and wavelength relationship (c) 3D spectrogram representation used in this study showing the extraction of spatial features from frequency and wavelength relationship, and extraction of spectral feature in a third amplitude dimension.

Time–frequency analysis further explains the access of more features in 3D spectrograms. In time–frequency analysis, expression of a function f and expression of the frequencies of the function F are two different equivalent representations of the same object (Gröchenig, 2001). This means that a function and the frequencies of the function have different features and enable access of different features. However, when the frequencies are expressed in terms of time, features from both the function and the frequencies are accessed in time-frequency, thereby allowing access of more features in frequency vs time than in frequency only. Applying

this to spectrogram representations of reflectance data, which are computed using the Fourier transform like the spectrograms computed in time-frequency analysis, the frequencies computed from reflectance using the Fourier transform and the reflectance have different features. Expressing the frequencies over wavelength, accesses features from both the frequencies and reflectance, whose features are expressed in amplitude in the spectrograms. Therefore, expression of amplitude as a third dimension in 3D spectrograms enabled the ResNet-3D models to access not only the features in frequency that are accessed in 2D from the relationship of frequency and wavelength but also the features in reflectance that are expressed in amplitude which are not accessed by 2D CNN models because these latter models can only capture features that are in the relationship between frequency and wavelength as shown in Figure 6.1b. Thus, this implies that, to make the estimations, the ResNet-3D models access and use more features for the expression of frequency over wavelength than the DenseNet-2D models.

The performance of the ResNet-3D and DenseNet-2D models were further investigated by training and testing the models on quartile datasets. The 3D and 2D CNN models with different primary architectures that were trained in model architecture selection were also trained and tested on quartile datasets. Most of the 2D models including the DenseNet-2D models achieved lower errors in Au grade estimates than 3D models on the lower Au grade quartile datasets, which had pronounced absorption and reflectance features in the spectra, as shown by RMSEs in Tables 5.3 and 5.4. Conversely, most of the 3D models achieved lower errors in Au grade estimates than 2D models on the higher Au grade quartile datasets, which had subtle features in the spectra as shown in Tables 5.3 and 5.4. These results show that, even though the ResNet-3D models outperformed the DenseNet-2D models on the non-normalized, normalized and continuum removed representative datasets that had all quartiles represented in them, the 2D

models outperformed the 3D models on some of the quartile datasets, especially those in which absorption and reflectance features in the spectra were pronounced. However, these results also show the robustness of the ResNet-3D models and the other 3D CNN models for data in which subtle features existed in the spectra. This suggests that, in the non-normalized, normalized and continuum removed representative datasets, where all quartile datasets were represented in the training, validation and testing datasets, making it harder for the CNN models to identify and capture data with subtle features due to bias towards data with pronounced absorption and reflectance features, the ResNet-3D models and other 3D CNN models were able to capture and make use of the spectra with subtle features than the DenseNet-2D models. Therefore, the ResNet-3D models would be beneficial in a production environment where models are trained on representative datasets and used to estimate Au grades from spectra of unknown Au grade quartile class because these models would estimate Au grade with higher accuracy compared to DenseNet-2D models because they would have learned even the subtle features in some spectra in the representative datasets that they would have been trained on.

The ResNet-3D models outperforming DenseNet-2D models on all representative datasets implies that the former models achieved individual point Au grade predictions that were closer to the actual individual Au grade values that were determined from fire assay. Furthermore, most 3D CNN models outperforming most 2D CNN models on the quartile datasets that had subtle features in spectra than 2D CNN provided further evidence for the robustness of 3D CNN models over 2D CNN models for datasets with subtle features. Therefore, these results provide collective evidence for higher accuracies of ore grade estimation by ResNet-3D models than DenseNet-2D models, and that, in application of CNN for ore grade estimation from drill core HS data in frequency domain, 3D CNN models should be considered. However, when applied to other kinds of datasets for ore grade estimation of other minerals, the best

architectures for the data representation of such datasets should be investigated first because the performance of the architectures may be influenced by the data representation of the datasets.

6.2 Normalization of Spectrograms for ResNet-3D and DenseNet-2D CNN Models

One of the most significant results from the study was that both the ResNet-3D and DenseNet-2D models obtained better performance in Au grade estimation on the non-normalized spectrograms than on the normalized spectrograms. The ResNet-3D and DenseNet-2D models achieved RMSEs of 0.238 ppm and 0.531 ppm on the non-normalized 3D and 2D spectrograms, respectively, and RMSEs of 0.651 ppm and 0.738 ppm on the normalized 3D and 2D spectrograms, respectively. The normalization of the spectrograms was done to remove any variation that existed among the reflectance spectra as a result of differences in acquisition. This was done by scaling the reflectance to the same range, which would stabilize the CNN models when applied to the data. However, this normalization caused an increase in RMSE. Therefore, to determine the reason for the increase in RMSE in the normalized spectrograms, the non-normalized and normalized spectra from the same interval of the same drill core were plotted to determine any difference in the spectra (Figure 6.2). The reflectance spectra were plotted over the spectrograms because any variations in features would be visible in the absorption or reflectance features in reflectance. The two plots showed that there was no difference in shape of the absorption and reflectance features between the normalized and non-normalized spectra but only that the former were scaled to reflectance values between 0 and 1 while the non-normalized reflectance spectrum was plotted in the acquired reflectance values. It is known that reflectance is a measure quantified between 0 and 1. Therefore, it is deduced that, since reflectance was already acquired as values between 0 and 1, normalising the spectrograms by scaling them to between 0 and 1 distorted the features that were already scaled,

which lowered the accuracy of Au grade estimation based on them. Therefore, normalization of spectrograms transformed from reflectance spectra by min-max method should not be considered, because reflectance is naturally scaled to between 0 and 1. Furthermore, this result also highlights that little is known about the effects of normalization and the different techniques of normalization on different kinds of datasets; therefore, future studies should investigate these effects on different kinds of datasets, including HS reflectance data and spectrograms.

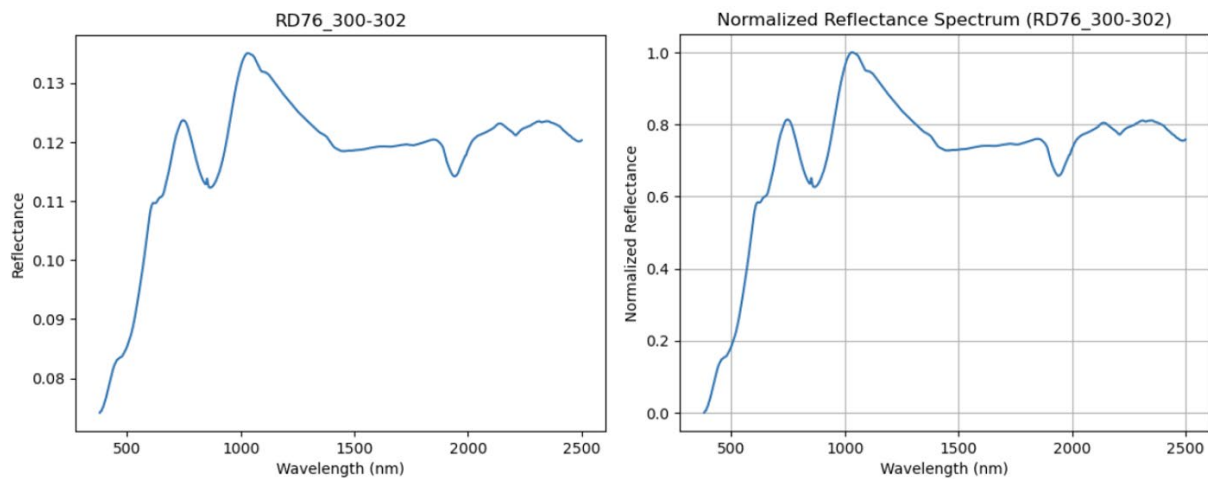


Figure 6.2 Original (left) and Normalized (right) reflectance spectra of 300 – 302 m interval of drill core RD76. Both spectra have the same absorption and reflectance features.

6.3 CNN Model Performance on Reflectance Spectra with Subtle Features

The ResNet-3D and all 3D CNN models outperformed the DenseNet-2D models and all other 2D CNN models on higher quartile datasets with subtle features in spectra. This provides evidence for better accuracy and performance for Au grade estimation by 3D CNN models from spectra with subtle features. However, all the CNN models that were trained in this study, performed poorly on the higher quartile datasets than on the lower quartile datasets. This highlights that CNN models generally perform poorly on spectra with subtle features and that those subtle features cause high RMSEs in ore grade estimation regardless of the dimensions

of CNN models. Therefore, it is necessary to analyse the reflectance spectra prior to the training of models, to not only determine if the spectra and the grades have patterns and relationships that the models can learn and use to make prediction for new data but also to determine if subtle features exist in the spectra that the models cannot capture and learn. Therefore, techniques that could enhance these features should be applied in order to reduce the chances of over- or underestimation of ore grades. Furthermore, the analysis would also be beneficial in a production environment where an acquired spectrum with an unknown grade can be analysed prior to grade estimation to determine if it is likely to result in over- or underestimation of such grade so that other complementary techniques of grade estimation can be applied to prevent over- or underestimation of ore grades.

Moreover, since both the ResNet-3D and DenseNet-2D models obtained higher RMSEs on the continuum removed spectrograms than on the original spectrograms, which was another unexpected result of the study, it highlights that more research must be done on how subtle features in reflectance can be enhanced for spectrograms for accurate estimation of ore grades. Continuum-removal was applied to the reflectance spectra that had subtle features that were observed in the higher Au grade quartile datasets. This was done to enhance features so that they would be pronounced in the spectrograms and consequently be captured by the models to increase the accuracy of grade estimation. However, it decreased the accuracy by increasing the RMSE. Therefore, this showed that continuum-removal does not enhance subtle features in the spectrograms. Instead, it was observed that continuum removal reduced the size of the spectrograms in both 2D and 3D from the original spectrograms (Figure 6.3), which reduced some of the features that were accessible to the models to learn and use to make predictions for new data. Thus, this showed that more studies on how subtle features in reflectance can be enhanced for spectrograms should be done because the continuum removal that was used did

not enhance features in spectra but instead reduced the features in the spectrograms. This would be contributory as these features may be associated with target ore grades in other datasets as they are in this study and they should be enhanced to prevent over- or under-estimation of the target ore grades.

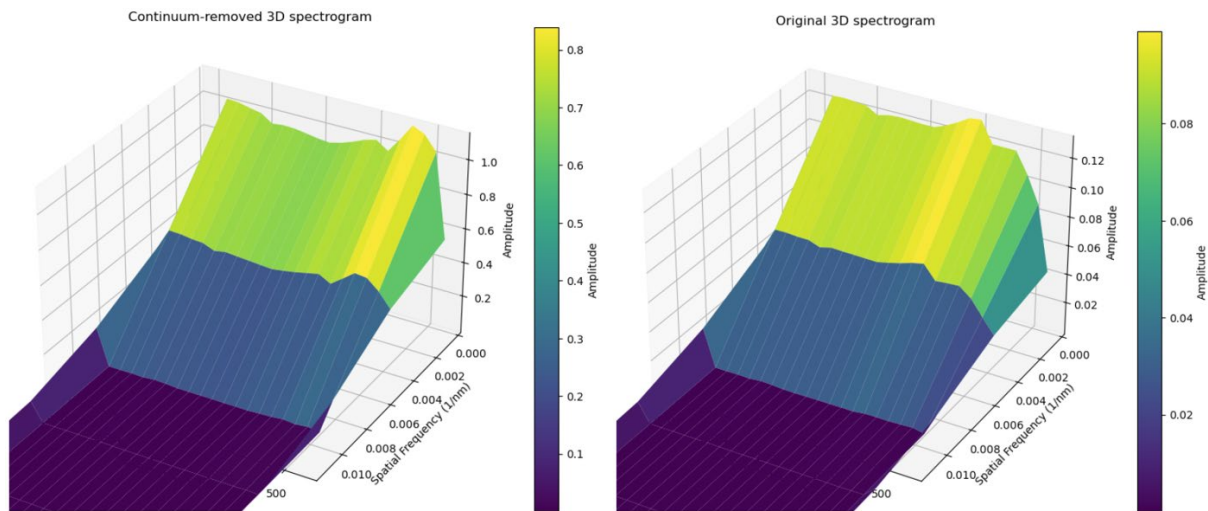


Figure 6.3 Continuum-removed and original 3D spectrograms for the interval 300-303 m of drill core RD76.

Chapter 7 : Conclusion

The effects of increasing the dimensions of convolutional neural network (CNN) from 2D (two-dimensional) to 3D (three dimensional) on the accuracy and performance of ore grade estimation from hyperspectral (HS) reflectance data in the frequency domain were investigated by training and comparing the accuracies of DenseNet-2D and ResNet-3D models on 2D and 3D spectrograms. The results of this comparison were supported by comparing the accuracies of other multiple 2D and 3D CNN models, from which the selection of DenseNet-2D and ResNet-3D models as the best 2D and 3D CNN models for the spectrograms that were used was done in this study. The effects of increasing the dimensions of CNN from 2D to 3D on the accuracy and performance of ore grade estimation were investigated because the accuracy of ore grade estimation by CNN reported in the literature was still not highly satisfactory. Therefore, this study endeavoured to investigate if grade estimation errors can be decreased by increasing the dimension of CNN from 2D to 3D, as this has proven to increase the accuracy of HS image classification. Furthermore, there is inadequate research on how increasing the dimensions of CNN from 2D to 3D affects accuracy from non-image data in which features lack spatial correlation, and this was achieved in this study on spectrograms, which were computed from reflectance data.

As discussed earlier, the ResNet-3D models outperformed the DenseNet-2D models based on lower RMSEs on non-normalized, normalized and continuum removed spectrograms. In addition, the ResNet-3D models and the other 3D CNN models outperformed the DenseNet-2D models and the other 2D CNN on the quartile datasets that had subtle features in the spectra. These results not only showed that increasing the dimensions of CNN from 2D to 3D increases

the accuracy of ore grade estimation from HS reflectance data in the frequency domain, but also that 3D CNN models, compared to 2D CNN models, are more robust for subtle features in some reflectance spectra. The subtle features, which cause poor accuracy of grade estimation, were identified in the higher quartile datasets of the Au grade data used in this study and were not enhanced by continuum removal in the spectrograms for identification by the CNN models. Therefore, while research is required on how these features can be enhanced for capturing by CNN models, as continuum removal failed to enhance the features in the spectrograms, the application of 3D CNN models on the spectrograms for ore grade estimation should be considered because they are more robust for subtle features in the spectrograms and they lower the RMSE of grade estimation.

Furthermore, the results provided evidence for the increase in accuracy by increase in the dimensions of CNN from 2D to 3D for ore grade estimation from non-image data. The spectrograms that were computed from the reflectance spectra did not have any spatial correlation of adjacent points, unlike in HS images which have spatial correlation of adjacent points on objects in the image. Instead, the spectrograms were a wavelength–frequency representation of the reflectance spectra and the features that extracted from the spatial axes in HS images were extracted from the relationship between the wavelength and frequency in the spectrograms. Therefore, the spectrograms did not have spatially correlated features that represent adjacent points of one object. Thus, these results showed that increasing the dimensions of CNN from 2D to 3D does increase accuracy of ore grade estimation from non-image data, though this remains underexplored in the application of HS data.

However, contrary to what is commonly reported in literature, the CNN models that were trained and evaluated in this study, both 2D and 3D, achieved lower RMSEs on the original non-normalized spectrograms. While this study did not confirm the underlying mechanisms by which this happened, one tentative explanation is that because reflectance was quantized between 0 and 1, the min–max normalization that was used in this study to normalize the spectra was unnecessary since it scaled a naturally scaled reflectance to between 0 and 1. Therefore, in the application of the 3D CNN models that achieve higher accuracy of ore grade estimation from spectrogram representation of reflectance spectra, min–max normalization should not be applied. However, research should be done on the stability of these models and other normalization techniques that can be applied to the spectrograms to stabilize the 3D CNN models for ore grade estimation from these spectrograms.

References

- Abdolmaleki, M., Consens, M., & Esmaceli, K. (2022). Ore-Waste Discrimination Using Supervised and Unsupervised Classification of Hyperspectral Images. *Remote Sensing*, 14(24). <https://doi.org/10.3390/rs14246386>
- Abubakar, J., Zhang, Z., Cheng, Z., Yao, F., & Bio Sidi D. Bouko, A. A. (2024). Advancing Skarn Iron Ore Detection through Multispectral Image Fusion and 3D Convolutional Neural Networks (3D-CNNs). *Remote Sensing*, 16(17). <https://doi.org/10.3390/rs16173250>
- Abuntori, C. A., Al-Hassan, S., & Mireku-Gyimah, D. (2021). Assessment of Ore Grade Estimation Methods for Structurally Controlled Vein Deposits - A Review. *Ghana Mining Journal*, 21(1), 31–44. <https://doi.org/10.4314/gm.v21i1.4>
- Amer, R., El Mezayen, A., & Hasanein, M. (2016). ASTER spectral analysis for alteration minerals associated with gold mineralization. *Ore Geology Reviews*, 75, 239–251. <https://doi.org/10.1016/j.oregeorev.2015.12.008>
- Bai, T., & Tahmasebi, P. (2021). Accelerating geostatistical modeling using geostatistics-informed machine Learning. *Computers and Geosciences*, 146. <https://doi.org/10.1016/j.cageo.2020.104663>
- Bengio, Y. (2009). Learning Deep Architectures for AI. *Foundations and trends in Machine Learning*, 2(1), 1–127 <http://www.iro.umontreal.ca/~bengioy>
- Chen, Y., Jiang, H., Li, C., Jia, X., & Ghamisi, P. (2016). Deep Feature Extraction and Classification of Hyperspectral Images Based on Convolutional Neural Networks. *IEEE Transactions on Geoscience and Remote Sensing*, 54(10), 6232–6251. <https://doi.org/10.1109/TGRS.2016.2584107>
- Choros, K. A., Job, A. T., Edgar, M. L., Austin, K. J., & McAree, P. R. (2022). Can Hyperspectral Imaging and Neural Network Classification be Used for Ore Grade Discrimination at the Point of Excavation? *Sensors*, 22(7). <https://doi.org/10.3390/s22072687>
- Clark, R. N., Swayze, G. A., Livo, K. E., Kokaly, R. F., Sutley, S. J., Dalton, J. B., McDougal, R. R., & Gent, C. A. (2003). Imaging spectroscopy: Earth and planetary remote sensing with the USGS Tetracorder and expert systems. *Journal of Geophysical Research: Planets*, 108(12). <https://doi.org/10.1029/2002je001847>
- Dalton, J. B., Bove, D. J., Mladinich, C. S., & Rockwell, B. W. (2004). Identification of spectrally similar materials using the USGS Tetracorder algorithm: The calcite-epidote-chlorite problem. *Remote Sensing of Environment*, 89(4), 455–466. <https://doi.org/10.1016/j.rse.2003.11.011>
- Ehrig, K., McPhie, J., & Kamenetsky, V. S. (2012). Geology and mineralogical zonation of the Olympic Dam iron oxide Cu-U-Au-Ag deposit, South Australia. *Society of Economic Geologists*, 16, 237–267 <https://www.researchgate.net/publication/291217932>
- Eichstaedt, H., Ho, C. Y. J., Kutzke, A., & Kahnt, R. (2022). Performance measurements of machine learning and different neural network designs for prediction of geochemical properties based on hyperspectral core scans. *Australian Journal of Earth Sciences*, 69(5), 733–741. <https://doi.org/10.1080/08120099.2022.2017344>
- Erickson, B. J., & Kitamura, F. (2021). Magician's corner: 9. performance metrics for machine learning models. In *Radiology: Artificial Intelligence*, 3(3). Radiological Society of North America Inc. <https://doi.org/10.1148/ryai.2021200126>
- Fukushima, K. (1980). Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics*, 36(4), 193–202
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press
- Gröchenig, K. (2001). *Foundations of Time-Frequency Analysis*. Birkhäuser.
- Kale, K. V., Solankar, M. M., Nalawade, D. B., Dhumal, R. K., & Gite, H. R. (2017). A Research Review on Hyperspectral Data Processing and Analysis Algorithms. *Proceedings of the National Academy of Sciences India Section A - Physical Sciences*, 87(4), 541–555 <https://doi.org/10.1007/s40010-017-0433-y>

- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in neural information processing systems*, 25 <http://code.google.com/p/cuda-convnet/>
- Kutzke, A., Eichstaedt, H., & Kahnt, R. (2022). Potential of hyperspectral-based geochemical predictions with neural networks for strategic and regional exploration improvement. *Australian Journal of Earth Sciences*, 69(8), 1197–1206. <https://doi.org/10.1080/08120099.2022.2094465>
- Le Cun, Y., (1989). *Generalization and Network Design Strategies* (Technical Report CRG-TR-89-4). University of Toronto Connectionist Research Group.
- Le Cun, Y., Jackel, L. D., Eon Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Mm Uller, U. A., Ss Ackinger, E., Simard, P., & Vapnik, V. (1995). Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, 261(276), 2
- Lee, H., & Kwon, H. (2016). Contextual deep CNN based hyperspectral classification. *2016 IEEE international geoscience and remote sensing symposium, (IGARSS)*, 3322-3325. IEEE.
- Lee, H., & Kwon, H. (2016). Going deeper with contextual CNN for hyperspectral image classification. *IEEE Transactions on Image Processing*, 26(10), 4843-4855. <https://doi.org/10.1109/TIP.2017.2725580>
- Liu, B. (2021). “Weak AI” is Likely to Never Become “Strong AI”, So What is its Greatest Value for us? <http://arxiv.org/abs/2103.15294>
- Liu, H., Wu, K., Xu, H., & Xu, Y. (2021). Lithology classification using TASI thermal infrared hyperspectral data with convolutional neural networks. *Remote Sensing*, 13(16). <https://doi.org/10.3390/rs13163117>
- McPhie, J., Kamenetsky, V. S., Chambefort, I., Ehrig, K., & Green, N. (2011). Origin of the supergiant Olympic Dam Cu-U-Au-Ag deposit, South Australia: Was a sedimentary basin involved. *Geology*, 39(8), 795–798. <https://doi.org/10.1130/G31952.1>
- Namatēvs, I. (2018). Deep Convolutional Neural Networks: Structure, Feature Extraction and Training. *Information Technology and Management Science*, 20(1). <https://doi.org/10.1515/itms-2017-0007>
- Naser, M. Z., & Alavi, A. H. (2020). Insights into Performance Fitness and Error Metrics for Machine Learning. www.mznaser.com
- Piczak, K. J. (2015). Environmental sound classification with convolutional neural networks. In *2015 IEEE 25th international workshop on machine learning for signal processing (MLSP)*, 1-6, IEEE
- Prado, E. M. G., de Souza Filho, C. R., & Carranza, E. J. M. (2023). Ore-Grade Estimation from Hyperspectral Data Using Convolutional Neural Networks: A Case Study at the Olympic Dam Iron Oxide Copper-Gold Deposit, Australia. *Economic Geology*, 118(8), 1899–1921. <https://doi.org/10.5382/econgeo.5023>
- Praveen, B., & Menon, V. (2021). Study of Spatial-Spectral Feature Extraction Frameworks with 3-D Convolutional Neural Network for Robust Hyperspectral Imagery Classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14, 1717–1727. <https://doi.org/10.1109/JSTARS.2020.3046414>
- Rao, C., & Liu, Y. (2020). Three-dimensional convolutional neural network (3D-CNN) for heterogeneous material homogenization. *Computational Materials Science*, 184. <https://doi.org/10.1016/j.commatsci.2020.109850>
- Reeves, J. (1990). Olympic Dam copper-uranium-gold-silver deposit. *Geology of the mineral deposits of Australia and Papua New Guinea*, 15-26
- Reynolds, L. J. (2000). *Hydrothermal Iron Oxide Copper-Gold and Related Deposits: A Global Perspective* (Vol. 1). PGC Publishing.
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088), 533–536. <https://doi.org/10.1038/323533a0>
- Tan, R., Shao, Y., Brzozowski, M. J., Zheng, Y., Xiong, Y.-Q. (2024). Development of a machine learning model to classify mineral deposits using sphalerite chemistry and mineral assemblages. *Ore Geology Reviews*, 169, 106076. [doi.org](https://doi.org/10.1016/j.oregeorev.2024.106076)

- Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. In *Insights into Imaging*, 9(46), 11–62. <https://doi.org/10.1007/s13244-018-0639-9>
- Yang, F., Zuo, R., & Kreuzer, O. P. (2024). Artificial intelligence for mineral exploration: A review and perspectives on future directions from data science. *Earth-Science Reviews*, 258 <https://doi.org/10.1016/j.earscirev.2024.104941>
- Yu, C., Han, R., Song, M., Liu, C., & Chang, C. I. (2020). A simplified 2D-3D CNN architecture for hyperspectral image classification based on spatial-spectral fusion. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 13, 2485–2501. <https://doi.org/10.1109/JSTARS.2020.2983224>
- Zhu, Z. (2016). Gold in iron oxide copper-gold deposits. *Ore Geology Reviews*, 72(P1), 37–42. <https://doi.org/10.1016/j.oregeorev.2015.07.001>

Appendices

Appendix A: Python code for plotting the reflectance spectra from the 300-302 m interval of drill core RD76

```
# Plotting the Reflectance spectra from the 300-302m interval
import glob
import numpy as np
import matplotlib.pyplot as plt

# Plotting the graphs on the same grid
files=glob.glob('RD76_000*.txt')
print(files)

for i in files:
    data=np.loadtxt(i,delimiter=',', skiprows=1)
    x=data[:,0]
    y=data[:,1]
    plt.plot(x,y, label=str(i))
    plt.xlabel('Wavelength (nm)')
    plt.ylabel('Reflectance')
    plt.title('RD76_300-302')

    plt.legend
```

Appendix B: Python code for calculating average reflectance spectra and plotting the resulting spectra

```
# Calculating average reflectance of spectra and plotting the average reflectance spectra
import glob
import os, sys
import spectral
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path
from scipy.signal import stft

input_dir = Path.cwd()/ "RD76_300-302"

files = list(input_dir.glob("*.txt*"))

parts = []
for path in list(input_dir.glob("*.txt*")):
    part = pd.read_csv(path)
    parts.append(part)

length = len(parts)

df = pd.concat(parts, axis=1)

filename = 'RD76_0001_116.txt'
RD76_116 = pd.read_csv(filename, delimiter=',', usecols=[0,1], header=None, skiprows=1)

Wavelength_nm = RD76_116[0]

df["mean"] = df.loc[:, [c for c in df.columns if c!= "Wavelength_nm"]].mean(axis=1)

y = df["mean"]

x = Wavelength_nm
y = df["mean"]

plt.plot(x, y)
plt.xlabel('Wavelength (nm)')
plt.ylabel('Reflectance')

plt.title('RD76_300-302')
plt.show()
```

Appendix C: Python code for STFT of reflectance spectra into 2D spectrogram

```
wavelengths_nm = Wavelength_nm
reflectance_spectrum = y

fs = 1
f, t, Zxx = stft(reflectance_spectrum, fs=fs, nperseg=50)

wavelengths_t = np.linspace(wavelengths_nm.min(), wavelengths_nm.max(), len(t))

plt.figure(figsize=(12, 6))
plt.pcolormesh(wavelengths_t, f, np.abs(Zxx), shading='gouraud', cmap='viridis')

plt.ylim(0, 0.05)

plt.colorbar(label='Amplitude')
plt.xlabel('Wavelength (nm)')
plt.ylabel('Frequency (arbitrary units)')
plt.title('Spectrogram-like Plot using STFT')
plt.show()
```

Appendix D: Python code for STFT of reflectance spectra into 3D spectrogram

```
wavelengths_nm = Wavelength_nm
reflectance_spectrum = y

num_samples = 500
wavelengths_uniform = np.linspace(wavelengths_nm.min(), wavelengths_nm.max(), num_samples)

interp_func = interp1d(wavelengths_nm, reflectance_spectrum, kind='cubic', fill_value="extrapolate")
reflectance_uniform = interp_func(wavelengths_uniform)

fs = 1 / np.mean(np.diff(wavelengths_uniform))

# Apply STFT
nperseg = min(50, len(reflectance_uniform))
f, t, Zxx = stft(reflectance_uniform, fs=fs, nperseg=nperseg)

wavelengths_t = np.linspace(wavelengths_uniform.min(), wavelengths_uniform.max(), len(t))

# plotting
W, F = np.meshgrid(wavelengths_t, f)
Z = np.abs(Zxx) # Magnitude of STFT

# visualization
fig = plt.figure(figsize=(14, 8))
ax = fig.add_subplot(111, projection='3d')

surf = ax.plot_surface(W, F, Z, cmap='viridis', edgecolor='none')

ax.set_ylim(0, f.max() / 10)

ax.set_xlabel('Wavelength (nm)')
ax.set_ylabel('Spatial Frequency (1/nm)')
ax.set_zlabel('Amplitude')
ax.set_title('3D Spectrogram-like Plot using STFT')
fig.colorbar(surf, ax=ax, label='Amplitude')

ax.view_init(elev=30, azim=120)

plt.show()
```

Appendix E: Python code for stratified splitting of data into training, validation and testing datasets

```
import numpy as np
import os
from pathlib import Path
from sklearn.model_selection import train_test_split
from collections import Counter

labeled_data_dir = Path.cwd() / "/Users/2018289369/Documents/Merged files of the data as 3-D data/Labeled_3D_Plots"
labeled_files = list(labeled_data_dir.glob("*.npz"))
X_data, Y_labels, filenames = [], [], []
for npz_file in labeled_files:
    data = np.load(npz_file)
    W, F, Z, label = data['W'], data['F'], data['Z'], data['label']
    X_data.append(np.stack([W, F, Z], axis=-1)) # Shape: (N, M, 3)
    Y_labels.append(label)
    filenames.append(npz_file.name)
X_data = np.array(X_data) # Shape: (num_samples, N, M, 3)
Y_labels = np.array(Y_labels) # Shape: (num_samples,)
label_counts = Counter(Y_labels)
print("Label distribution before removing classes with one sample:", label_counts)
classes_to_keep = [label for label, count in label_counts.items() if count > 1]
indices_to_keep = [i for i, label in enumerate(Y_labels) if label in classes_to_keep]
X_data_filtered = X_data[indices_to_keep]
Y_labels_filtered = Y_labels[indices_to_keep]
filenames_filtered = [filenames[i] for i in indices_to_keep]
label_counts_filtered = Counter(Y_labels_filtered)
print("Label distribution after removing classes with one sample:", label_counts_filtered)
X_train, X_temp, Y_train, Y_temp, filenames_train, filenames_temp = train_test_split(
    X_data_filtered, Y_labels_filtered, filenames_filtered, test_size=0.4, stratify=Y_labels_filtered, random_state=42
)
X_val, X_test, Y_val, Y_test, filenames_val, filenames_test = train_test_split(
    X_temp, Y_temp, filenames_temp, test_size=0.5, stratify=Y_temp, random_state=42
)
print(f"✅ Training set: {X_train.shape}, Labels: {Y_train.shape}")
print(f"✅ Validation set: {X_val.shape}, Labels: {Y_val.shape}")
print(f"✅ Testing set: {X_test.shape}, Labels: {Y_test.shape}")
output_dir = labeled_data_dir / "Split_Data"
output_dir.mkdir(exist_ok=True)
np.savez(output_dir / "train_data.npz", X=X_train, Y=Y_train, filenames=filenames_train)
np.savez(output_dir / "val_data.npz", X=X_val, Y=Y_val, filenames=filenames_val)
np.savez(output_dir / "test_data.npz", X=X_test, Y=Y_test, filenames=filenames_test)
print("\n✅ Data successfully split and saved!")
```

Appendix F: Python code used to train the 2D VGG16 model

```
class SpectrogramDataGenerator(tf.keras.utils.Sequence):
    def __init__(self, dataframe, batch_size, input_shape, shuffle=True):
        self.dataframe = dataframe.reset_index(drop=True)
        self.batch_size = batch_size
        self.input_shape = input_shape
        self.shuffle = shuffle
        self.on_epoch_end()

    def __len__(self):
        return int(np.ceil(len(self.dataframe) / self.batch_size))

    def __getitem__(self, index):
        batch_df = self.dataframe.iloc[index * self.batch_size:(index + 1) * self.batch_size]
        X = np.zeros((len(batch_df), *self.input_shape), dtype=np.float32)
        y = np.zeros((len(batch_df),), dtype=np.float32)

        for i, (_, row) in enumerate(batch_df.iterrows()):
            try:
                arr = np.load(row["filepath"])
                arr = arr.astype(np.float32)

                arr = tf.image.resize(arr[..., np.newaxis], self.input_shape[:2]).numpy()
                X[i] = arr
                y[i] = row["label"]
            except Exception as e:
                print(f"⚠ Error loading {row['filepath']}: {e}")

        return X, y

    def on_epoch_end(self):
        if self.shuffle:
            self.dataframe = shuffle(self.dataframe)

def build_vgg16_regressor(input_shape):
    model = models.Sequential()

    model.add(layers.Conv2D(64, (3, 3), activation='relu', padding='same', input_shape=input_shape))
    model.add(layers.Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(layers.MaxPooling2D((2, 2)))

    model.add(layers.Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(layers.Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(layers.MaxPooling2D((2, 2)))

    model.add(layers.Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(layers.Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(layers.MaxPooling2D((2, 2)))

    model.add(layers.Flatten())
    model.add(layers.Dense(256, activation='relu'))
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(1, activation='linear')) # Regression output

    return model
```

Appendix G: Python code used to train the standard 2D CNN model

```
class SpectrogramDataGenerator(tf.keras.utils.Sequence):
    def __init__(self, dataframe, batch_size, input_shape, shuffle=True):
        self.df = dataframe.reset_index(drop=True)
        self.batch_size = batch_size
        self.input_shape = input_shape
        self.shuffle = shuffle
        self.on_epoch_end()

    def __len__(self):
        return int(np.ceil(len(self.df) / self.batch_size))

    def __getitem__(self, index):
        batch = self.df.iloc[index * self.batch_size:(index + 1) * self.batch_size]
        X = np.zeros((len(batch), *self.input_shape), dtype=np.float32)
        y = np.zeros((len(batch),), dtype=np.float32)

        for i, (_, row) in enumerate(batch.iterrows()):
            try:
                arr = np.load(row["filepath"]).astype(np.float32)
                arr = tf.image.resize(arr[..., np.newaxis], self.input_shape[:2]).numpy()

                arr = np.repeat(arr, 3, axis=-1)

                X[i] = arr
                y[i] = row["label"]
            except Exception as e:
                print(f"⚠ Failed to load {row['filepath']}: {e}")

        return X, y

    def on_epoch_end(self):
        if self.shuffle:
            self.df = shuffle(self.df)

train_gen = SpectrogramDataGenerator(train_df, batch_size, input_shape)

model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=input_shape),
    MaxPooling2D(pool_size=(2, 2)),

    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),

    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),

    Flatten(),
    Dense(256, activation='relu'),
    Dense(128, activation='relu'),

    Dense(1, activation='linear')
])

model.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

Appendix H: Python code used to train the ResNet-2D model

```
train_csv = Path("/Users/2018289369/Documents/Model training - Quartiles - 2D/Q1_Lowest/split_data/train.csv")
input_shape = (128, 128, 3)
batch_size = 32
epochs = 50
save_path = "resnet2d_regression_model_Q1.keras"

def residual_block(x, filters):
    skip = x

    x = Conv2D(filters, (3, 3), padding='same')(x)
    x = BatchNormalization()(x)
    x = ReLU()(x)

    x = Conv2D(filters, (3, 3), padding='same')(x)
    x = BatchNormalization()(x)

    if skip.shape[-1] != filters:
        skip = Conv2D(filters, (1, 1), padding='same')(skip)
        skip = BatchNormalization()(skip)

    x = Add()([x, skip])
    x = ReLU()(x)
    return x

def build_resnet_regression(input_shape):
    inputs = Input(shape=input_shape)

    x = Conv2D(64, (7, 7), padding='same', strides=2)(inputs)
    x = BatchNormalization()(x)
    x = ReLU()(x)
    x = MaxPooling2D(pool_size=(3, 3), strides=2, padding='same')(x)

    x = residual_block(x, 64)
    x = residual_block(x, 128)
    x = residual_block(x, 256)
    x = residual_block(x, 512)

    x = GlobalAveragePooling2D()(x)
    x = Dense(256, activation='relu')(x)
    x = Dense(128, activation='relu')(x)

    outputs = Dense(1, activation='linear')(x)

    model = Model(inputs, outputs)
    model.compile(optimizer='adam', loss='mse', metrics=['mae'])
    return model
```

Appendix I: Python code used to train the DenseNet-2D model

```
train_csv = Path("/Users/2018289369/Documents/Model training - Quartiles - 2D/Q1_Lowest/split_data/train.csv")
input_shape = (128, 128, 3)
batch_size = 32
epochs = 50
save_path = "densenet2d_regression_model_Q1.keras"

def dense_block(x, filters, growth_rate, num_layers):
    for _ in range(num_layers):
        y = Conv2D(growth_rate, (3, 3), padding='same')(x)
        y = BatchNormalization()(y)
        y = ReLU()(y)
        x = Concatenate()([x, y])
    return x

def transition_layer(x, filters):
    x = Conv2D(filters, (1, 1), padding='same')(x)
    x = BatchNormalization()(x)
    x = ReLU()(x)
    x = AveragePooling2D(pool_size=(2, 2), strides=2, padding='same')(x)
    return x

def build_densenet_regression(input_shape=(128, 128, 3), growth_rate=32, num_layers=[4, 4, 4]):
    inputs = Input(shape=input_shape)

    x = Conv2D(64, (7, 7), padding='same', strides=2)(inputs)
    x = BatchNormalization()(x)
    x = ReLU()(x)
    x = MaxPooling2D(pool_size=(3, 3), strides=2, padding='same')(x)

    for i, layers in enumerate(num_layers):
        x = dense_block(x, filters=64*(i+1), growth_rate=growth_rate, num_layers=layers)
        if i != len(num_layers) - 1:
            x = transition_layer(x, filters=64*(i+2))

    x = GlobalAveragePooling2D()(x)
    x = Dense(256, activation='relu')(x)
    x = Dense(128, activation='relu')(x)
    outputs = Dense(1, activation='linear')(x)

    model = Model(inputs, outputs)
    model.compile(optimizer='adam', loss='mse', metrics=['mae'])
    return model
```

Appendix J: Python code used to train the 2D FCN model

```
def build_fcn_regression(input_shape=(128, 128, 3)):
    inputs = Input(shape=input_shape)
    x = Conv2D(64, (3, 3), padding='same', activation='relu')(inputs)
    x = Conv2D(128, (3, 3), padding='same', activation='relu')(x)
    x = MaxPooling2D(pool_size=(2, 2))(x)
    x = Conv2D(256, (3, 3), padding='same', activation='relu')(x)
    x = Conv2D(512, (3, 3), padding='same', activation='relu')(x)
    x = MaxPooling2D(pool_size=(2, 2))(x)
    x = Conv2D(512, (3, 3), padding='same', activation='relu')(x)
    x = GlobalAveragePooling2D()(x)
    outputs = Dense(1, activation='linear')(x)
    model = Model(inputs, outputs)
    model.compile(optimizer='adam', loss='mse', metrics=['mae'])
    return model

class SpectrogramDataGenerator(tf.keras.utils.Sequence):
    def __init__(self, dataframe, batch_size, input_shape, shuffle=True):
        self.df = dataframe.reset_index(drop=True)
        self.batch_size = batch_size
        self.input_shape = input_shape
        self.shuffle = shuffle
        self.on_epoch_end()

    def __len__(self):
        return int(np.ceil(len(self.df) / self.batch_size))

    def __getitem__(self, index):
        batch = self.df.iloc[index * self.batch_size:(index + 1) * self.batch_size]
        X = np.zeros((len(batch), *self.input_shape), dtype=np.float32)
        y = np.zeros((len(batch),), dtype=np.float32)
        for i, (_, row) in enumerate(batch.iterrows()):
            try:
                arr = np.load(row["filepath"]).astype(np.float32)
                arr = tf.image.resize(arr[..., np.newaxis], self.input_shape[:2]).numpy()
                arr = np.repeat(arr, 3, axis=-1) # Convert grayscale to RGB
                X[i] = arr
                y[i] = row["label"]
            except Exception as e:
                print(f"⚠ Error loading {row['filepath']}: {e}")
        return X, y

    def on_epoch_end(self):
        if self.shuffle:
            self.df = shuffle(self.df)

train_df = pd.read_csv(train_csv)
train_gen = SpectrogramDataGenerator(train_df, batch_size, input_shape)
model = build_fcn_regression(input_shape=input_shape)
model.summary()
checkpoint = tf.keras.callbacks.ModelCheckpoint(save_path, save_best_only=False)
model.fit(
    train_gen,
    epochs=epochs,
    callbacks=[checkpoint],
    verbose=1
)
print(f"\n✅ FCN-2D regression model trained and saved as: {save_path}")
```

Appendix K: Python code used to train the 2D SGCN model

```
csv_path = Path("/Users/2018289369/Documents/Model training - Quartiles - 2D/Q1_Lowest/split_data/train.csv")
target_size = (32, 32)
channels = 1
input_dim = target_size[0] * target_size[1]
epochs = 50
batch_size = 32

def build_grid_adjacency(h, w):
    from scipy.sparse import lil_matrix
    adj = lil_matrix((h * w, h * w))

    for row in range(h):
        for col in range(w):
            idx = row * w + col
            if row > 0: adj[idx, idx - w] = 1
            if row < h - 1: adj[idx, idx + w] = 1
            if col > 0: adj[idx, idx - 1] = 1
            if col < w - 1: adj[idx, idx + 1] = 1

    return adj.tocsr()

adj = build_grid_adjacency(*target_size)

class SpectrogramGCNDataset(Dataset):
    def __init__(self, csv_file, **kwargs):
        self.df = pd.read_csv(csv_file)
        super().__init__(**kwargs)

    def read(self):
        output = []
        for i, row in self.df.iterrows():
            spec = np.load(row["filepath"]).astype(np.float32)
            spec = tf.image.resize(spec[..., np.newaxis], target_size).numpy()
            x = spec.reshape(-1, 1)
            y = np.array([row["label"]], dtype=np.float32)
            output.append(Graph(x=x, a=adj, y=y))
        return output

dataset = SpectrogramGCNDataset(csv_path, transforms=GCNFilter())
loader = BatchLoader(dataset, batch_size=batch_size, epochs=epochs, shuffle=True)
```

Appendix L: Python code used to train the 2D ResNeXt model

```
train_csv = Path("/Users/2018289369/Documents/Model training - Quartiles - 2D/Q1_Lowest/split_data/train.csv")
input_shape = (128, 128, 3)
batch_size = 32
epochs = 50
save_path = "resnext2d_regression_model_Q1.keras"

def resnext_block(x, filters, groups=32):
    shortcut = x
    in_channels = x.shape[-1]

    x = Conv2D(filters, (1, 1), padding='same', use_bias=False)(x)
    x = BatchNormalization()(x)
    x = ReLU()(x)

    x = Conv2D(filters, (3, 3), padding='same', groups=groups, use_bias=False)(x)
    x = BatchNormalization()(x)
    x = ReLU()(x)

    x = Conv2D(filters, (1, 1), padding='same', use_bias=False)(x)
    x = BatchNormalization()(x)

    if in_channels != filters:
        shortcut = Conv2D(filters, (1, 1), padding='same', use_bias=False)(shortcut)
        shortcut = BatchNormalization()(shortcut)

    x = Add()([x, shortcut])
    x = ReLU()(x)
    return x

def build_resnext_regression(input_shape=(128, 128, 3), num_blocks=3, filters=64, groups=8):
    inputs = Input(shape=input_shape)

    x = Conv2D(filters, (7, 7), strides=2, padding='same', use_bias=False)(inputs)
    x = BatchNormalization()(x)
    x = ReLU()(x)
    x = MaxPooling2D(pool_size=(3, 3), strides=2, padding='same')(x)

    for i in range(num_blocks):
        x = resnext_block(x, filters, groups=groups)
        filters *= 2

    x = GlobalAveragePooling2D()(x)
    x = Dense(256, activation='relu')(x)
    x = Dense(128, activation='relu')(x)
    outputs = Dense(1, activation='linear')(x)

    model = Model(inputs, outputs)
    model.compile(optimizer='adam', loss='mse', metrics=['mae'])
    return model
```

Appendix M: Python code used to train the 2D U-Net model

```
class SpectrogramDataGenerator(tf.keras.utils.Sequence):
    def __init__(self, dataframe, batch_size, input_shape, shuffle=True):
        self.df = dataframe.reset_index(drop=True)
        self.batch_size = batch_size
        self.input_shape = input_shape
        self.shuffle = shuffle
        self.on_epoch_end()

    def __len__(self):
        return int(np.ceil(len(self.df) / self.batch_size))

    def __getitem__(self, index):
        batch = self.df.iloc[index * self.batch_size:(index + 1) * self.batch_size]
        X = np.zeros((len(batch), *self.input_shape), dtype=np.float32)
        y = np.zeros((len(batch),), dtype=np.float32)

        for i, (_, row) in enumerate(batch.iterrows()):
            try:
                arr = np.load(row["filepath"]).astype(np.float32)
                arr = tf.image.resize(arr[..., np.newaxis], self.input_shape[:2]).numpy()
                arr = np.repeat(arr, 3, axis=-1)
                X[i] = arr
                y[i] = row["label"]
            except Exception as e:
                print(f"⚠ Error loading {row['filepath']}: {e}")

        return X, y

    def on_epoch_end(self):
        if self.shuffle:
            self.df = shuffle(self.df)

def conv_block(x, filters):
    x = Conv2D(filters, (3, 3), padding='same', activation='relu')(x)
    x = Conv2D(filters, (3, 3), padding='same', activation='relu')(x)
    return x

def unet_regression(input_shape=(128, 128, 3)):
    inputs = Input(shape=input_shape)

    c1 = conv_block(inputs, 64)
    p1 = MaxPooling2D((2, 2))(c1)

    c2 = conv_block(p1, 128)
    p2 = MaxPooling2D((2, 2))(c2)

    b = conv_block(p2, 512)

    u1 = UpSampling2D((2, 2))(b)
    m1 = Concatenate()([u1, c2])
    c3 = conv_block(m1, 128)

    u2 = UpSampling2D((2, 2))(c3)
    m2 = Concatenate()([u2, c1])
    c4 = conv_block(m2, 64)

    outputs = Conv2D(1, (1, 1), activation='linear')(c4)

    model = Model(inputs, outputs)
    model.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

Appendix N: Python code used to train the standard 3D CNN model

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv3D, BatchNormalization, Activation, MaxPooling3D, Flatten, Dense, Dropout
from tensorflow.keras.regularizers import l2

X_train = np.load('/Users/2018289369/Documents/3-D models/3-D CNN (standard) on stratified dataset/Labeled_3D_Plots/Split_Data/X_train.npy')
Y_train = np.load('/Users/2018289369/Documents/3-D models/3-D CNN (standard) on stratified dataset/Labeled_3D_Plots/Split_Data/Y_train.npy')

X_train = np.expand_dims(X_train, axis=-2)

model = Sequential([
    Conv3D(32, kernel_size=(3, 3, 3), padding="same", kernel_regularizer=l2(0.01), input_shape=X_train.shape[1:]),
    BatchNormalization(),
    Activation("relu"),
    MaxPooling3D(pool_size=(2, 2, 1)),

    Conv3D(64, kernel_size=(3, 3, 3), padding="same", kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    Activation("relu"),
    MaxPooling3D(pool_size=(2, 2, 1)),

    Conv3D(128, kernel_size=(3, 3, 3), padding="same", kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    Activation("relu"),
    MaxPooling3D(pool_size=(2, 2, 1)),

    Flatten(),
    Dense(256, activation="relu", kernel_regularizer=l2(0.01)),
    Dropout(0.5),
    Dense(128, activation="relu", kernel_regularizer=l2(0.01)),
    Dropout(0.5),
    Dense(1, activation="linear")
])

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
              loss="mean_squared_error",
              metrics=["mae"])

model.fit(X_train, Y_train, epochs=50, batch_size=16, verbose=1)

model.save("3D_CNN_trained_model.h5")
```

Appendix O: Python code used to train the ResNet-3D model

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, regularizers

X_train = np.load('/Users/2018289369/Documents/Model training/Data/Split_Data/X_train.npy')
Y_train = np.load('/Users/2018289369/Documents/Model training/Data/Split_Data/Y_train.npy')

X_train = np.expand_dims(X_train, axis=-1)

def build_resnet_3d(input_shape):
    inputs = keras.Input(shape=input_shape)

    x = layers.Conv3D(32, (3, 3, 3), padding="same", strides=(1, 1, 1),
                      kernel_regularizer=regularizers.l2(1e-4))(inputs)
    x = layers.BatchNormalization()(x)
    x = layers.ReLU()(x)

    x = resnet_3d_block(x, 32)
    x = resnet_3d_block(x, 64, strides=(2, 2, 2))
    x = resnet_3d_block(x, 128, strides=(2, 2, 2))
    x = resnet_3d_block(x, 256, strides=(2, 2, 2))

    x = layers.GlobalAveragePooling3D()(x)
    x = layers.Dropout(0.5)(x)

    outputs = layers.Dense(1, activation="linear")(x)

    model = keras.Model(inputs, outputs, name="ResNet3D_Regression")
    return model

input_shape = X_train.shape[1:]
model = build_resnet_3d(input_shape)

model.compile(optimizer=keras.optimizers.Adam(learning_rate=1e-4),
              loss="mean_squared_error",
              metrics=["mae"])

history = model.fit(X_train, Y_train, epochs=50, batch_size=16,
                    validation_split=0.2, verbose=1)

model.save("resnet3d_regression_model.h5")

print("✅ Training complete! Model saved.")
```

Appendix P: Python code used to train the DenseNet-3D model

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import (
    Input, Conv3D, BatchNormalization, Activation, Dense, GlobalAveragePooling3D, Dropout
)
from tensorflow.keras.regularizers import l2

X_train = np.load('/Users/2018289369/Documents/3-D models/DenseNet-3D CNN on stratified dataset/Labeled_3D_Plots/Split_Data/X_train.npy')
Y_train = np.load('/Users/2018289369/Documents/3-D models/DenseNet-3D CNN on stratified dataset/Labeled_3D_Plots/Split_Data/Y_train.npy')

X_train = np.expand_dims(X_train, axis=1)

def dense_block(x, filters, dropout_rate=0.2, l2_lambda=0.01):
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Conv3D(filters, kernel_size=(3, 3, 3), padding='same', kernel_regularizer=l2(l2_lambda))(x)
    x = Dropout(dropout_rate)(x)
    return x

def build_densenet_3d(input_shape):
    inputs = Input(shape=input_shape)

    x = Conv3D(32, kernel_size=(3, 3, 3), padding='same', kernel_regularizer=l2(0.01))(inputs)
    x = dense_block(x, 64)
    x = dense_block(x, 128)
    x = dense_block(x, 256)

    x = GlobalAveragePooling3D()(x)
    x = Dense(128, activation='relu', kernel_regularizer=l2(0.01))(x)
    x = Dropout(0.3)(x)
    x = Dense(64, activation='relu', kernel_regularizer=l2(0.01))(x)
    x = Dropout(0.3)(x)
    outputs = Dense(1, activation='linear')(x)

    model = Model(inputs, outputs)
    return model

input_shape = (1, 26, 21, 3)
model = build_densenet_3d(input_shape)

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
              loss='mse',
              metrics=['mae'])

history = model.fit(X_train, Y_train, epochs=50, batch_size=16, verbose=1)

model.summary()
```

Appendix Q: Python code used to train the 3D FCN model

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, regularizers

X_train = np.load('/Users/2018289369/Documents/3-D models/3-D FCN CNN on stratified dataset/Labeled_3D_Plots/Split_Data/X_train.npy')
Y_train = np.load('/Users/2018289369/Documents/3-D models/3-D FCN CNN on stratified dataset/Labeled_3D_Plots/Split_Data/Y_train.npy')

X_train = np.expand_dims(X_train, axis=-1)

def build_fcn_3d(input_shape):
    inputs = keras.Input(shape=input_shape)

    x = layers.Conv3D(32, (3, 3, 3), activation='relu', padding='same',
                      kernel_regularizer=regularizers.l2(0.001))(inputs)
    x = layers.BatchNormalization()(x)
    x = layers.MaxPooling3D((2, 2, 1))(x)

    x = layers.Conv3D(64, (3, 3, 3), activation='relu', padding='same',
                      kernel_regularizer=regularizers.l2(0.001))(x)
    x = layers.BatchNormalization()(x)
    x = layers.MaxPooling3D((2, 2, 1))(x)

    x = layers.Conv3D(128, (3, 3, 3), activation='relu', padding='same',
                      kernel_regularizer=regularizers.l2(0.001))(x)
    x = layers.BatchNormalization()(x)
    x = layers.GlobalAveragePooling3D()(x)

    outputs = layers.Dense(1, activation='linear')(x)

    model = keras.Model(inputs, outputs, name="3D_FCN_Regression")
    return model

model = build_fcn_3d(X_train.shape[1:])
model.compile(optimizer=keras.optimizers.Adam(learning_rate=0.001),
              loss='mean_squared_error',
              metrics=['mae'])
model.summary()
history = model.fit(X_train, Y_train, epochs=50, batch_size=16, verbose=1)
# Save the trained model
model.save("3D_FCN_model.h5")
```

Appendix R: Python code used to train the 3D SGCN model

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense, Dropout, Flatten
from tensorflow.keras.regularizers import l2
from spektral.layers import GCSCConv
from spektral.utils import normalized_adjacency

X_train = np.load('/Users/2018289369/Documents/3-D models/3-D SGCN CNN on stratified dataset/Labeled_3D_Plots/Split_Data/X_train.npy')
Y_train = np.load('/Users/2018289369/Documents/3-D models/3-D SGCN CNN on stratified dataset/Labeled_3D_Plots/Split_Data/Y_train.npy')

X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], -1))

num_nodes = X_train.shape[1]
A = np.ones((num_nodes, num_nodes))
A = normalized_adjacency(A)

def build_sgcnn_model(input_shape, A_matrix, l2_reg=1e-4, dropout_rate=0.4):
    X_input = Input(shape=input_shape)
    A_input = Input(shape=(num_nodes, num_nodes))

    x = GCSCConv(32, activation="relu", kernel_regularizer=l2(l2_reg))([X_input, A_input])
    x = Dropout(dropout_rate)(x)
    x = GCSCConv(64, activation="relu", kernel_regularizer=l2(l2_reg))([x, A_input])
    x = Dropout(dropout_rate)(x)
    x = Flatten()(x)
    x = Dense(128, activation="relu", kernel_regularizer=l2(l2_reg))(x)
    x = Dropout(dropout_rate)(x)
    outputs = Dense(1, activation="linear")(x)

    model = Model(inputs=[X_input, A_input], outputs=outputs)
    model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
                  loss='mse',
                  metrics=['mae'])

    return model

model = build_sgcnn_model(X_train.shape[1:], A)
model.summary()

history = model.fit([X_train, np.tile(A, (X_train.shape[0], 1, 1))], Y_train, batch_size=16, epochs=50, verbose=1)

model.save("sgcn_regression_model.h5")
```

Appendix S: Python code used to train the 3D ResNeXt model

```
def build_resnext_3d(input_shape, num_classes):
    weight_decay = 0.01

    model = models.Sequential()
    model.add(layers.InputLayer(input_shape=input_shape))

    model.add(layers.Conv3D(filters=64, kernel_size=(3, 3, 3), padding='same',
                           strides=(1, 1, 1), kernel_regularizer=regularizers.l2(weight_decay)))
    model.add(layers.BatchNormalization())
    model.add(layers.ReLU())
    model.add(layers.Dropout(0.3))

    model.add(layers.Conv3D(filters=64, kernel_size=(3, 3, 3), padding='same',
                           groups=8, kernel_regularizer=regularizers.l2(weight_decay)))
    model.add(layers.BatchNormalization())
    model.add(layers.ReLU())
    model.add(layers.Dropout(0.3))

    model.add(layers.MaxPooling3D(pool_size=(2, 2, 2), padding='same'))

    model.add(layers.Conv3D(filters=128, kernel_size=(3, 3, 3), padding='same',
                           strides=(1, 1, 1), kernel_regularizer=regularizers.l2(weight_decay)))
    model.add(layers.BatchNormalization())
    model.add(layers.ReLU())
    model.add(layers.Dropout(0.4))

    model.add(layers.Conv3D(filters=128, kernel_size=(3, 3, 3), padding='same',
                           groups=8, kernel_regularizer=regularizers.l2(weight_decay)))
    model.add(layers.BatchNormalization())
    model.add(layers.ReLU())
    model.add(layers.Dropout(0.4))

    model.add(layers.MaxPooling3D(pool_size=(2, 2, 2), padding='same'))

    model.add(layers.Flatten())

    model.add(layers.Dense(512, kernel_regularizer=regularizers.l2(weight_decay)))
    model.add(layers.ReLU())
    model.add(layers.Dropout(0.5))

    # Output layer
    model.add(layers.Dense(num_classes, activation='softmax', kernel_regularizer=regularizers.l2(weight_decay)))

    return model
```

Appendix T: Python code for computing continuum removed reflectance spectra for 2D continuum removed spectrograms

```
import glob
import os, sys
import spectral
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path
from scipy.spatial import ConvexHull

input_dir = Path.cwd() / "RD76_300-302"
files = list(input_dir.glob("*.txt*"))

parts = []
for path in files:
    part = pd.read_csv(path)
    parts.append(part)

df = pd.concat(parts, axis=1)

filename = 'RD76_0001_116.txt'
RD76_116 = pd.read_csv(filename, delimiter=',', usecols=[0, 1], header=None, skiprows=1)
Wavelength_nm = RD76_116[0]

df["mean"] = df.loc[:, [c for c in df.columns if c != "Wavelength_nm"]].mean(axis=1)

x = Wavelength_nm.values
y = df["mean"].values

# CONTINUUM REMOVAL

def continuum_removal(x, y):
    points = np.column_stack((x, y))
    hull = ConvexHull(points)

    upper_hull_indices = sorted([v for v in hull.vertices if
                                (v == 0 or v == len(y)-1) or
                                (y[v] >= y[v-1] and y[v] >= y[v+1])])

    upper_hull_x = x[upper_hull_indices]
    upper_hull_y = y[upper_hull_indices]

    continuum = np.interp(x, upper_hull_x, upper_hull_y)

    cr_y = y / continuum
    return cr_y

cr_y = continuum_removal(x, y)

plt.figure(figsize=(10, 6))
plt.plot(x, cr_y, color='black')
plt.xlabel('Wavelength (nm)')
plt.ylabel('Continuum-Removed Reflectance')
plt.title('Continuum-Removed Spectrum: RD76_300-302')
plt.grid(True)
plt.tight_layout()
plt.show()
```