Deep Learning Methods for Elemental Map Prediction from MA-XRF spectra

A Thesis

submitted to the designated
by the Assembly
of the Department of Computer Science and Engineering
Examination Committee

by

Ioannis Georvasilis

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN DATA AND COMPUTER SYSTEMS ENGINEERING

WITH SPECIALIZATION
IN DATA SCIENCE AND ENGINEERING

University of Ioannina
School of Engineering
Ioannina 2023

Dedication

To my family, for their unwavering support and love during my academic journey.

Acknowledgements

I am deeply grateful to Professor Aristidis Likas for his unwavering trust, guidance, invaluable advice, and methodological insights throughout the preparation of this thesis. His continuous and unconditional support has been vital in initiating my journey into the field of Machine Learning research, and I owe the beginning of this endeavor to his mentorship.

I would like to thank my friend and fellow Fanis Gerodimos for his valuable help, support and quality cooperation in the whole process of research and experimentation. Our scientifically exchanging views were crucial for the success of this study. I would also like to extend my appreciation to Professor Dimitrios Anagnostopoulos for entrusting me with the research on Macro-XRF data. His continuous presence and patience in explaining the complex Physics involved have been invaluable to my work. I would also like to thank my friend and fellow Georgios Vardakas for his valuable contribution and enlightening scientific discussions on Machine Learning.

I'd like to offer my sincere thanks to my wonderful friends. Their support and willingness to engage in deep conversations, even when the topics were complex, have been a source of immense comfort and growth. Interacting with them has helped me stay in touch with my humanity, fostering a love for the world free from prejudice and nurturing my commitment to a more just society. My friends have been an essential part of my journey, and I am truly grateful for their presence in my life.

Last but not least, I am deeply indebted to my beloved family - my father, Maximos; my mother, Eleni; and my twin brothers, Thanasis and Julia. You have all been my pillars of strength, and the foundation upon which I have built my aspirations. Your sacrifices and unending belief in me have been the driving force behind my accomplishments. I cherish the moments we've shared, and I am eternally grateful for the immeasurable love and support you have showered upon me throughout this journey.

Table of Contents

List of Figures

List of Tables

Glossary

AI Artificial Intelligence

ML Machine Learning

CV Computer Vision

DL Deep Learning

GT Ground Truth

NLP Natural Language Processing

FCN Fully Connected Network

MLP Multi Layer Perceptron

DNN Deep Neural Network

CNN Convolutional Neural Network

1DCNN 1D Convolutional Neural Network

XRF X-ray Fluorescence

MA-XRF Macroscoping X-ray Fluorescence

SDD Silicon Drift Detectors

EDS Energy Dispersive Detectors

NMF Non-negative Matrix Factorization

ReLU Rectified Linear Unit

SGD Stochastic Gradient Descent

Abstract

Ioannis Georvasilis, M.Sc. in Data and Computer Systems Engineering, Department of Computer Science and Engineering, School of Engineering, University of Ioannina, Greece, 2023.
Deep Learning Methods for Elemental Map Prediction from MA-XRF spectra.
Advisor: Aristidis Likas, Professor.

Macro-XRF (MA-XRF) is a valuable non-destructive technique for investigating the chemical elemental composition of materials. MA-XRF data can be considered as high dimensional images, where a large vector of X-ray wavelengths emitted by excited atoms is computed for each pixel. Through MA-XRF analysis, experts derive Elemental Distribution Maps, which provide essential insights into the spatial distribution of various elements within a given sample. However, the generation of these maps requires significant expertise, analysis and time investment. This work attempts to provide an automated solution to this fundamental problem that is cast as a regression problem with high dimensional inputs and multiple outputs. The dataset under examination comprises paintings created using medieval painting techniques, dating from the 18th to the 19th centuries, with the primary objective of predicting concentrations for twelve specific chemical elements.

To address this problem two popular deep architectures namely, a Fully Connected Neural Network (MLP) and an 1d Convolutional Neural Network, have been employed as multi-output regressors. A novel issue in our approach is the integration of physics-based prior knowledge into these models. This prior knowledge is injected through a predefined layer with constant weights, which accurately encapsulates the spectral signatures of the twelve elements under examination. The experimental results indicate that the enhanced models lead to substantial performance improvement, highlighting the potential of combining deep learning techniques with physics-based insights in the domain of MA-XRF elemental mapping.

Keywords: machine learning, deep learning, deep neural networks, MA-XRF, cnn, elemental distribution maps, physics-based prior knowledge

Εκτεταμενη Περιληψη

Ιωάννης Γεωρβασίλης, Δ.Μ.Σ. στη Μηχανική Δεδομένων και Υπολογιστικών Συστημάτων, Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πολυτεχνική Σχολή, Πανεπιστήμιο Ιωαννίνων, 2023.
Μέθοδοι Βαθιάς Μάθησης για Πρόβλεψη Στοιχειακών Χαρτών από φάσματα MA-XRF.
Επιβλέπων: Αριστείδης Λύκας, Καθηγητής.

Η φασματοσκοπία φθορισμού ακτίνων-X macro (MA-XRF), είναι μια πολύτιμη μη καταστροφική αναλυτική τεχνική που χρησιμοποιείται για τη διερεύνηση της στοιχειακής σύνθεσης των υλικών. Μέσω της ανάλυσης MA-XRF, οι ειδικοί εξάγουν Χάρτες Στοιχειακής Κατανομής, οι οποίοι παρέχουν ουσιαστικές πληροφορίες για τη χωρική κατανομή διαφόρων στοιχείων σε ένα δεδομένο δείγμα. Ωστόσο, η δημιουργία αυτών των περίπλοκων χαρτών απαιτεί σημαντική τεχνογνωσία, ανάλυση και επένδυση χρόνου. Υπό το πρίσμα της αυξανόμενης αξιοποίησης της Μηχανικής Μάθησης, ιδίως της Βαθιάς Μάθησης, για την αντιμετώπιση σύνθετων προκλήσεων που σχετίζονται με το MA-XRF, η παρούσα εργασία περιστρέφεται γύρω από το θεμελιώδες πρόβλημα της πρόβλεψης Χαρτών Στοιχειακής Κατανομής (Elemental Distribution Maps), το οποίο διαμορφώνεται ως πρόβλημα παλινδρόμησης πολλαπλών εξόδων. Το υπό εξέταση σύνολο δεδομένων περιλαμβάνει πίνακες που δημιουργήθηκαν με μεσαιωνικές τεχνικές ζωγραφικής και χρονολογούνται από τον 18ο έως τον 19ο αιώνα, με πρωταρχικό στόχο την πρόβλεψη των συγκεντρώσεων δώδεκα συγκεκριμένων στοιχείων.

Για την αποτελεσματική αντιμετώπιση αυτής της πρόκλησης, η παρούσα μελέτη χρησιμοποιεί δύο ευρέως μελετημένες βαθιές αρχιτεκτονικές: ένα Πλήρως Συνδεδεμένο Νευρωνικό Δίκτυο (MLP) και ένα 1d Συνελικτικό Νευρωνικό Δίκτυο, που χρησιμοποιούνται ως παλινδρομείς πολλαπλών εξόδων. Αυτό που διαφοροποιεί αυτή την έρευνα είναι η ενσωμάτωση πρότερης γνώσης με βάση τη Φυσική σε αυτά τα μοντέλα βαθιάς μάθησης. Η πρότερη γνώση ορίζεται μέσω ενός προκαθορισμένου ενδιάμεσου επιπέδου (Prior Layer) με σταθερά βάρη, το οποίο ενσωματώνει τις φασματικές υπογραφές των δώδεκα στοιχείων που ορίζονται ως στόχοι. Τα σταθερά βάρη στο επίπεδο αυτό χρησιμεύουν ως μια μορφή εξαγωγής χαρακτηριστικών. Η εκμετάλλευση των φασματικών υπογραφών βοηθά τα μοντέλα να εστιάζουν στα κατάλληλα χαρακτηριστικά, βελτιώνοντας έτσι την ικανότητά τους να διακρίνουν και να διαφοροποιούν τα στοιχεία.

Τα πειραματικά ευρήματα της προτεινόμενης προσέγγισης υποδεικνύουν σημαντική βελτίωση της επίδοσης των μοντέλων, υπογραμμίζοντας έτσι τις δυνατότητες συνδυασμού της γνώσης βασισμένη στη Φυσική με τις τεχνικές βαθιάς μάθησης στον τομέα της στοιχειακής ανάλυσης MA-XRF.

Λέξεις Κλειδιά: *μηχανική μάθηση, βαθιά μάθηση, βαθιά νευρωνικά δίκτυα, χάρτες στοιχειακής κατανομής, πρότερη γνώση βασισμένη στη Φυσική*

Chapter 1

Introduction

1.1 Learning Algorithms

A Machine Learning (ML) algorithm is one that can learn from data, which occurs when a computer program improves its performance on a certain set of tasks as assessed by a specific performance metric over time. Machine Learning enables us to tackle issues that would be impossible to accomplish with fixed programs built and designed by humans.

Machine Learning tasks are typically described in terms of how the machine learning system should handle a given example. An example, denoted as $\mathbf{x} \in \mathbb{R}^d$, constitutes a set of d quantitatively measured features extracted from an object or event that we intend the Machine Learning system to process. Machine Learning can be applied to a wide array of tasks, with some of the most prevalent ones encompassing the following:

- **Classification**: In this task, the computer program is required to determine the category to which an input belongs among K possible categories. Usually, the learning algorithm is typically required to generate a function $f : \mathbb{R}^n \rightarrow \{1, \ldots, k\}$. Specifically, the model f outputs the category $\hat{y}$ for a given example x. There are alternative versions of the classification task where the output, $\hat{y}$, represents a probability distribution across categories (i.e. object recognition, where the input is an image and the output is a numeric code identifying the object in the image) or a binary decision in case of two categories (i.e. distinguishing between spam and non-spam emails in the context of email filtering).

- **Regression**: In this Machine Learning task, the computer program is now required to estimate a numerical value given some input. Commonly with classification tasks, the learning algorithm is typically required to generate a function f, differentating in the output space $f : \mathbb{R}^n \rightarrow \mathbb{R}$. Some real-world examples of regression tasks include predicting the price of a house given its characteristics, predicting the weather, or how many degrees the car should turn given the front camera input, etc.

To evaluate a Machine Learning algorithm's abilities, it is essential to establish a numerical measure for assessing its task performance. For tasks such as classification, we often measure the accuracy of the model. **Accuracy** is just the proportion of examples for which the model produces the correct output. We can also obtain equivalent information by measuring the **error rate**, the proportion of examples for which the model produces an incorrect output. The error on a particular example x is 0 if it is correctly classified and 1 if it is not.

Usually, we aim to assess how effectively the Machine Learning algorithm performs on unseen data, as this dictates its real-world deployment potential. Subsequently, we measure the model's performance using a distinct set of data, independent of the data used for training the Machine Learning system.

In certain instances, like a regression task, selecting a performance measure that aligns with the intended behaviour of the system can be challenging since it is not always clear how to determine what should be measured. For instance, when conducting a regression task, we need to determine whether to penalise the system more for frequently committing medium-sized errors or for rarely committing very large ones. These types of design choices are dependent on the application at hand. Below are listed some popular performance measurements:

- **Accuracy** (Classification): It measures the proportion of correctly classified instances out of the total number of instances.
- **Precision and Recall** (Classification): Precision measures the ratio of true positives to the total number of predicted positives, while recall (sensitivity) measures the ratio of true positives to the total number of actual positives.
- **F1-Score** (Classification): The F1-score is the harmonic mean of precision and recall.
- **Mean Absolute Error (MAE)** (Regression): MAE measures the average absolute difference between the predicted values and the actual values.
- **Mean Squared Error (MSE)** (Regression): MSE calculates the average of the squared differences between predicted and actual values.

1.2 Generalization

The primary objective in Machine Learning is achieving good performance on novel, unseen inputs on which our model was trained. **Generalization** is called the ability to perform well on unobserved inputs.

When a Machine Learning model is trained, there is typically access to a training datase, the `trainset`. An essential metric to be computed on this dataset is the **training error**, which represents the error that is aimed to be minimized. What sets Machine Learning apart from optimization is the goal of reducing not only the training error, but also the **generalization error**, which refers to the expected value of the error on a new, unseen input. This is also commonly referred to as the test error. The estimation of a Machine Learning model's generalization error is commonly achieved by measuring its performance on a test set of examples that were collected separately. This is known as the `testset`.

An immediate connection can be observed between the training and test error is that the expected training error of a randomly selected model is equal to the expected test error of that model. For achieving good generalization, two key goals are aimed to be achieved. Firstly, the training error is sought to be minimized to ensure good performance on the data on which the model is trained. Secondly, efforts are made to narrow the gap between the training error and the test error, which quantifies how well the model generalizes to new, unseen data. These two factors correspond to the two key challenges in Machine Learning : **underfitting** and **overfitting**.

- **Underfitting**: Underfitting occurs when a model is too simple to capture the underlying patterns in the training data, leading to poor performance on both the training and test datasets. It signifies that the model hasn't learned the data well and lacks the complexity needed to make accurate predictions.
- **Overfitting**: Overfitting is a challenging issue and occurs where a model becomes excessively complex, capturing noise and random fluctuations in the training data instead of the genuine underlying patterns. This results in excellent performance on the training dataset but poor performance on unseen data (test dataset), as the model fails to generalize effectively.

Model **capacity**, represents the ability of a model to learn and represent complex relationships within the data. We can control whether a model is more likely to overfit or underfit by altering its capacity. There are in fact many ways of changing a model's capacity. Model capacity is not solely determined by the model choice and the number of parameters it comprises. The model defines a family of functions that the learning algorithm can select from while adjusting parameters to minimize a training objective. This concept is referred to as the **representational capacity** of the model, and it often poses a challenging optimization problem. Consequently, the learning algorithm does not seek to identify the optimal function $f^\star$ but rather aims to find one, $\hat{f}$, that substantially reduces the training error.

1.3 Feedforward Neural Networks

Feedforward Neural Networks, or multilayer perceptrons (MLPs), is the subfield of Machine Learning models that made the most significant achievements in recent years in the research area of Artificial Intelligence (AI). The goal of a feedforward network is to approximate some function $f^\star$. For instance, a MLP classifier maps an input x to a category y. The MLP establishes a mapping $y = f(x;\theta)$ and acquires the optimal function approximation by learning the values of the parameters θ. These models are referred to as feedforward because the information passes through the function to evaluate from x, through the intermediate computations used to define f, and finally to the output y.

Feedforward networks build upon the foundational concept of the basic **Perceptron**, who invented from the psychologist F. Rosenblatt from Cornell University [2]. It is known as the "Perceptron" by its inventor, incorporating both analogue and digital signals with a threshold component that transforms analogue signals into digital ones. It is regarded as the initial artificial neural network (ANN). The Perceptron (see Figure 1.1), multiplies each feature, x_i, of the input vector $\mathbf{x}$, with its corresponding weight w_i. The weight w_0, also known as bias, is always multiplied by the number one. Afterward, the total sum passes through the activation function ϕ, as defined in Equation 1.1.

$$\phi = \begin{cases} 1, & \text{if } w_0 + \sum_{i=1}^{d} w_i x_i > 0 \\ 0, & \text{otherwise} \end{cases} \tag{1.1}$$

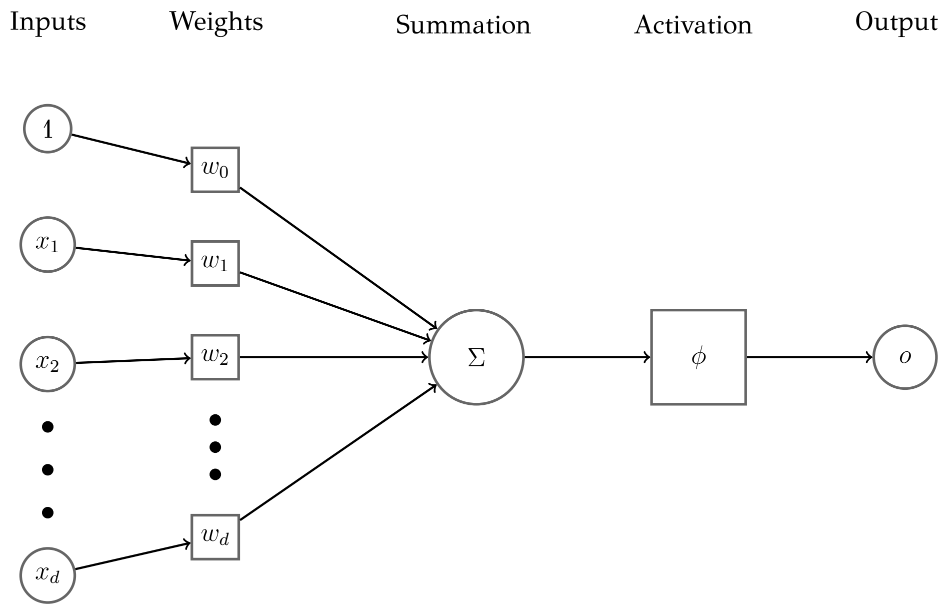


Figure 1.1: F. Rosenblatt's Perceptron.

Expanding upon the foundational Perceptron, feedforward networks extend the Perceptron's capabilities by introducing multiple layers $f^i(\boldsymbol{x})$ of interconnected neurons, allowing for the modeling of more intricate relationships within data. They are referred to as networks since they are commonly composed by combining multiple functions together, the **hidden layers**. The model is associated with a chained structure graph (see Figure 1.2), whereby each layer output feeds into the subsequent hidden layer as an input. The overall length of the chain gives the depth of the model. Each hidden layer of the network is typically vector-valued and the dimensionality of these hidden layers determines the width of the model. It is optimal to perceive feedforward networks as machines for approximating functions that are intended to achieve statistical generalisation, often drawing upon some knowledge gained from our observations.

Non-linear activation functions, such as the Sigmoid, Rectified Linear Unit (ReLU), or Hyperbolic Tangent (tanh), introduce **non-linearity** into the network. They enable neural networks to approximate complex, non-linear functions by capturing non-linear patterns and relationships in data, that are common in real-world data. By applying non-linear activation functions to the weighted sums in the hidden layers, the network gains the capacity to learn and represent intricate relationships.

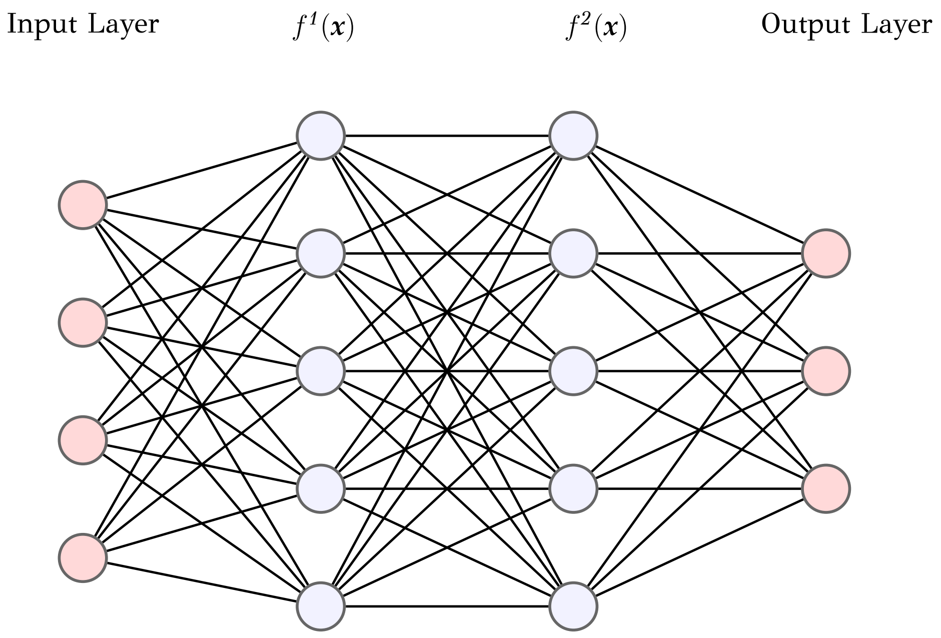


Figure 1.2: Feedforward Neural Network architecture with 2 hidden layers.

1.4 Cost Function Optimization

Nearly all of feedfoward networks are trained by **stochastic gradient descent** or SGD, which is an extension of gradient descent. Gradient descent is an optimization algorithm commonly used in Machine Learning to minimize a cost or loss function. It iteratively updates the parameters of a model to find the minimum of the **cost function**. A cost function, or loss function, denoted as $J(\theta)$, measures the network's effectiveness, similar to the metrics outlined in Section 1.1, by quantifying the difference between predicted and actual outputs. The optimization process of gradient descent, however, necessitates a differentiable cost function. For example, the parameters of MLP for classification tasks are often optimized using the Cross-Entropy Loss cost function:

$$CrossEntropy = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i))$$

where N is the number of training examples, y_i is the true class label for the i-th example (0 for one class and 1 for the other in binary classification), and $\hat{y}_i$ is the predicted class probability (typically obtained using a softmax activation function for multiclass classification).

On the other hand a widely used cost function for regression tasks is the Mean Squared Error (MSE):

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{1.2}$$

where N is the number of training examples, y_i is the actual target value for the ith example, and $\hat{y}_i$ is the predicted value for the i-th example.

The core idea behind SGD is the perspective that the gradient can be seen as an expectation. This expectation can be efficiently approximated using a small subset of samples. More precisely, in each iteration of the algorithm, a minibatch of examples $B = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(m')}, y^{(m')})\}$ can be randomly drawn uniformly from the training set. The size of the minibatch, denoted as m', is usually selected as a relatively small number of examples, typically ranging from 1 to a few hundred always depending on the training set size m.

The estimation of the gradient is defined as:

$$\nabla_\theta \text{Loss}(\theta) = -\frac{1}{b}\sum_{i=1}^{b}\nabla_\theta \text{Loss}(x^{(i)}, y^{(i)})$$

where ∇_θ represents the gradient with respect to the networks parameters θ, Loss(θ) is the loss function, m' is the minibatch size, $(x^{(i)}, y^{(i)})$ represents the i-th minibatch sample with input $x^{(i)}$ and target $y^{(i)}$. $\nabla_\theta \text{Loss}(x^{(i)}, y^{(i)})$ is the gradient of the loss with respect to the model parameters for the i-th minibatch example.

Stochastic Gradient Descent (SGD) plays a vital role in the extension of MLPs to Deep Neural Networks (DNNs), described in Chapter 3. This significance arises from SGD's proficiency in optimizing models within high-dimensional parameter spaces, its regularization capacity, aptitude for escaping local minima, adaptability through dynamic learning rates, suitability for parallel computation, memory efficiency, and applicability in real-time learning scenarios. The adaptability and versatility inherent in SGD render it an indispensable tool for the efficient training and optimization of the intricate architectures and extensive datasets commonly associated with DNNs.

1.5 Thesis Contribution

This thesis makes a notable contribution by proposing the integration of deep learning techniques with physics-based prior knowledge to address the challenge of Elemental Maps Prediction in Macro-XRF (MA-XRF) analysis. By incorporating a predefined layer encapsulating spectral signatures of examined elements, the research significantly enhances model accuracy and effectiveness, offering a promising approach for improving elemental mapping in materials analysis. This innovative fusion of machine learning and domain-specific insights has the potential to advance the broader field of MA-XRF and has practical implications for various applications, including cultural heritage preservation and materials science.

1.6 Thesis Outline

The following is a brief description of the structure of the following chapters:

- Chapter 2: Presentation of Macro X-ray fluorescence, including Image Analysis Techniques and Spectral Properties.
- Chapter 3: Introduction to deep learning, exploring representation learning with deep neural networks, convolutional neural networks (including convolution and architecture), and training techniques.

- Chapter 4: Outline of the proposed method using deep networks (Fully Connected Networks and Convolutional Networks) and incorporating Physics-Based Prior Knowledge.
- Chapter 5: Description of the experimental procedures carried out and analysis of the experimental results.
- Chapter 6: A summary of the thesis, along with ideas for future research and improvements in Elemental Distribution Mapping.

Chapter 2

Macro X-ray fluorescence

2.1 Introduction

Macro X-ray fluorescence (MA-XRF) is a powerful analytical technique that enables non-destructive elemental analysis of large objects and surfaces. It provides valuable information about the elemental composition of materials, allowing researchers to identify and characterize various substances. Macroscopic XRF is closely correlated with hyperspectral imaging, as both techniques involve the acquisition of spectral information. While hyperspectral imaging focuses on capturing detailed spectral data across a wide range of wavelengths, macro XRF specifically targets X-ray wavelengths (Figure 2.1) emitted by excited atoms in a sample. This technique offers several advantages, including its non-invasive nature, high sensitivity, and ability to analyze diverse materials. Moreover, macro XRF finds significant applications in different fields, such as archaeology, art conservation, geology, and environmental science. In the domain of cultural heritage, macro XRF has emerged as a valuable tool for investigating and preserving historical artifacts, paintings, sculptures, and architectural elements. By providing insights into the elemental composition of these cultural treasures, macro XRF aids in authentication, provenance determination, and the identification of restoration materials, contributing to the preservation of our shared heritage.

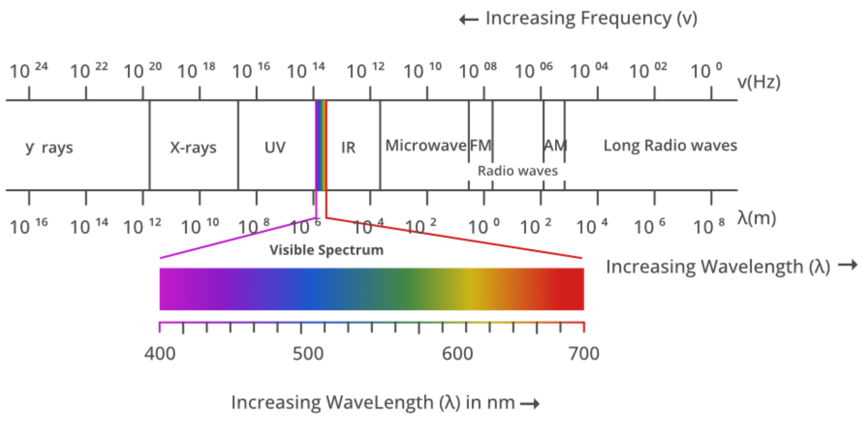


Figure 2.1: The Electromagnetic Spectrum: Illustration depicting the full range of electromagnetic wavelengths.

2.2 Spectro-Scanning techniques for Image Analysis

The spectro-scanning technique plays a crucial role in the analysis of macro-XRF images. The first step in spectro-scanning involves the acquisition of XRF images. XRF instruments emit high-energy X-ray radiation onto the sample surface, causing the atoms within the sample to emit characteristic fluorescent X-rays. These emitted X-rays are then collected and processed to generate a spatially resolved image of the sample's elemental composition.

To obtain XRF images, a scanning mechanism is employed (Figure 2.2). This mechanism consists of an X-ray source and a detector that move together across the sample surface in a controlled manner. The X-ray source emits a focused X-ray beam onto the sample, typically scanning the surface in a grid pattern or along predetermined lines. As the X-ray beam interacts with the sample, the emitted X-rays are captured by the detector which records their energy and intensity. The scanning process is often controlled by precise mechanical systems that ensure accurate positioning and movement of the X-ray source and detector. These systems employ various mechanisms such as stepper motors, linear stages, or robotic arms. The synchronized movement of the X-ray source and detector enables the collection of spatially resolved XRF data, which forms the basis of macro-XRF images.

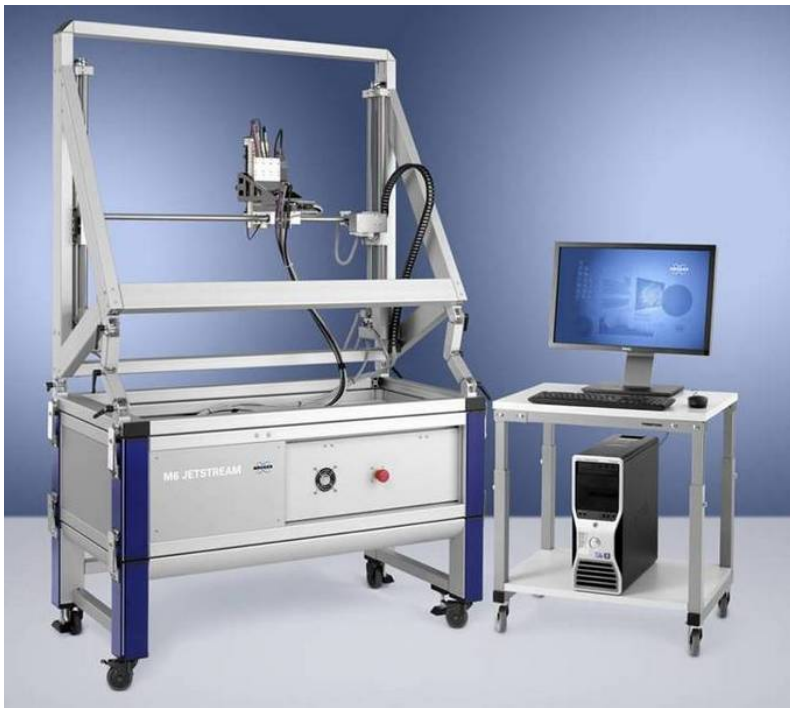


Figure 2.2: The spectro-scanning mechanism - Bruker M6 Jetstream.

Once the XRF data is acquired, it undergoes a series of processing steps to generate meaningful images. Initially, the raw data is pre-processed to remove noise and correct for instrumental artifacts. This includes background subtraction to eliminate signals unrelated to the sample's elemental composition. Next, the processed data is reconstructed into an image format using appropriate software. The XRF data is typically represented as a matrix of intensity values, where each pixel corresponds to a specific spatial location on the sample surface. This matrix can be visualized using various techniques, such as color mapping, where different elemental intensities are assigned different colors, allowing for the visual identification of elemental distributions.

In addition to elemental images, spectroscopic information can be extracted from the XRF data. By analyzing the energy spectrum of each pixel, it is possible to identify the presence of specific chemical elements and quantify their relative concentrations. This spectral analysis provides valuable insights into the composition and structure of the analyzed sample.

2.3 Physics Principles of Macro-XRF

X-ray fluorescence (XRF) spectroscopy is based on the interaction between high energy X-rays and the atoms in a sample, as shown in the figure 2.3. When a sample is exposed to X-rays, the inner-shell electrons of the atoms can be excited to higher energy levels. These excited electrons are inherently unstable and quickly return to their original energy levels, resulting in the emission of characteristic fluorescent X-rays.

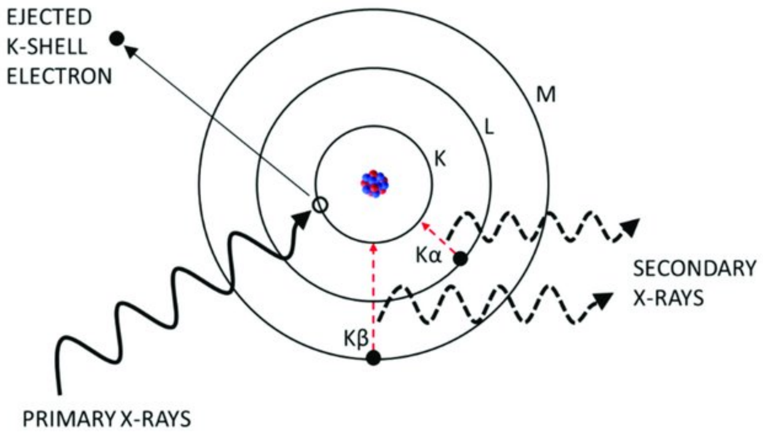


Figure 2.3: Physical mechanisms within an atom for X-ray-fluorescence [1].

The XRF excitation process occurs through two primary mechanisms: X-ray absorption and the photoelectric effect. X-ray absorption involves the absorption of incident X-rays by atoms in the sample, leading to the excitation of inner shell electrons. The probability of absorption (P) can be described by the Beer-Lambert law:

$$P = I_0 \cdot e^{-\mu x} \tag{2.1}$$

where I_0 is the initial intensity of the X-ray beam, μ is the linear absorption coefficient, which depends on the atomic properties and energy of the incident X-rays, and x is the thickness of the sample.

In the photoelectric effect, an incident X-ray photon interacts with an atom's inner shell electron, causing the electron to be ejected from its orbit. The energy (E) of the ejected electron is equal to the difference between the X-ray photon energy (E_{photon}) and the binding energy (E_{binding}) of the electron shell from which it originated:

$$E = E_{\text{photon}} - E_{\text{binding}} \tag{2.2}$$

The emitted characteristic X-rays during the de-excitation process provide valuable information about the elemental composition of the sample. Each chemical element has unique characteristic X-ray energies associated with its electron transitions, enabling the identification and quantification of elements present in the sample.

MA-XRF data can be thought of as high-dimensional images, akin to traditional two-dimensional images but with an added layer of complexity. In this context, each pixel of the image corresponds to a spectrum, which is essentially a large vector of X-ray wavelengths emitted by excited atoms within the sample. This multidimensional data structure captures a wealth of information about the sample's composition and distribution of elements, making it a powerful tool for in-depth analysis and visualization. Just as in traditional image processing, where each pixel represents a color or intensity value, in MA-XRF, each pixel encapsulates a spectrum, offering a rich source of data for researchers to explore and extract valuable insights. Figure 2.4b visualizes the MA-XRF data structure as presented by the authors in [3].

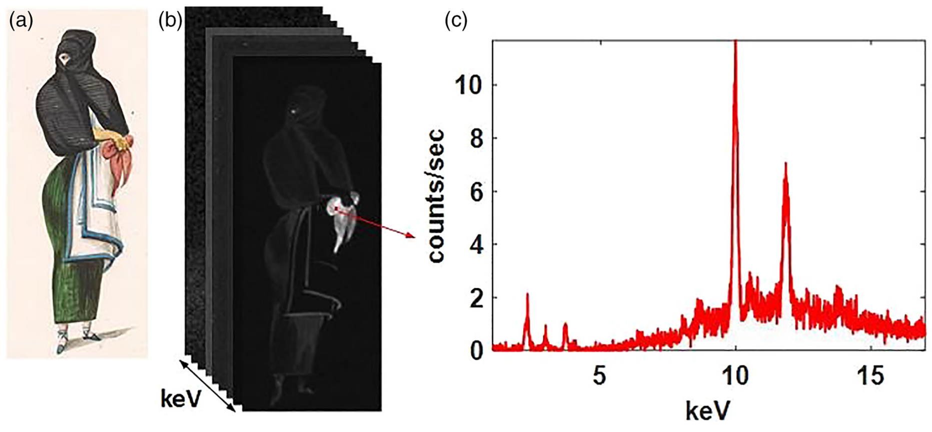


Figure 2.4: MA-XRF Data Representation.

To detect and analyze the emitted X-rays, various types of detectors are employed in macro-XRF instruments. Commonly used detectors include solid-state detectors, such as silicon drift detectors (SDD) or energy-dispersive detectors (EDS) [4], and gas-filled detectors like proportional counters or microcalorimeters. Solid-state detectors [5], operate based on the principle of X-ray energy conversion into electrical signals. When an X-ray photon interacts with the detector material, it produces a charge cloud that is collected by the detector's electrodes, generating an electrical pulse proportional to the X-ray energy. This signal is then processed and analyzed to determine the energy and intensity of the detected X-rays. Gas-filled detectors [6], on the other hand, rely on the ionization of gas molecules by X-ray photons. When an X-ray interacts with the gas, it ionizes the gas atoms, leading to the formation of ion-electron pairs. These ion-electron pairs are collected, resulting in an electrical signal that can be measured and analyzed to obtain information about the X-ray energy and intensity.

The macro-XRF technique considers not only the excitation and emission processes but also the effects of various factors on the resulting X-ray spectra. Factors such as X-ray attenuation in the sample, fluorescence yield, and detector efficiency impact the measured XRF intensities. Understanding and accounting for these factors are crucial for accurate quantitative analysis and interpretation of macro-XRF data.

2.4 Spectrum Properties and Analysis

XRF spectra retain useful properties that are essential for in-depth analysis and understanding. They enable researchers to improve analytical methods, reveal hidden compositional subtleties and, in turn, promote a deeper understanding of complex materials. More precisely, a XRF spectrum represent the distribution of X-ray intensities at different energies emitted by the sample [7]. Each energy corresponds to a specific atomic transition within the sample's constituent elements.

As illustrated in Figure 2.5, the form and intensity of the chemical elements (such as the Ka transition of Fe and Zn) differ, offering essential insights into the existence of elements and their corresponding ratios. XRF spectra have a key property: the pure elemental spectrum signatures are non-linearly combined. To clarify this statement objectively, a pure elemental spectrum can be obtained by either scanning a sample that exclusively contains a specific element or generating synthetic spectra using specialized software that employs statistical analysis and requires input of the necessary parameters for generation [8]. Such a combination or mixture results from the overlapping excitation and emission energies of the different elements.

The process of separating the combination of spectra into its individual elemental spectra is referred to as Spectral Unmixing or Decomposition. Various algorithms and mathematical models have been developed to perform this task [9, 10, 11]. These methods aim to identify the individual elemental contributions based on known pure elemental spectra. By comparing the measured mixed spectrum to a library of pure

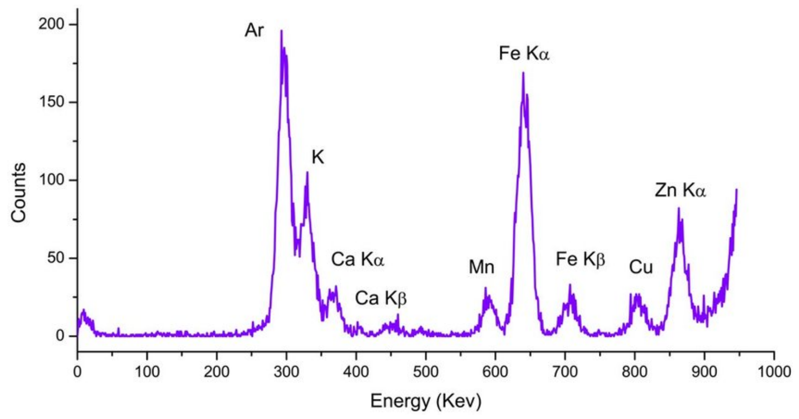


Figure 2.5: Revealing Elemental Composition: Sum Spectrum of Scanned Image.

elemental spectra, it is possible to estimate the relative concentrations of the elements within the sample. Nonetheless, Spectral Unmixing can be a challenging task due to factors such as spectral overlap, limited energy resolution of the detector, and measurement noise. Advanced techniques, such as Non-negative Natrix Factorization (NMF) [12] and constrained Non-linear Least Squares fitting, have been employed to improve the accuracy and reliability of Spectral Unmixing [13].

Interpretation of XRF spectra involves identifying the characteristic peaks and determining the corresponding elements. Each element has unique XRF spectral characteristics such as peak energies and relative intensities. Comparison of the measured spectra with reference spectra or databases of known elemental XRF signatures aids in elemental identification. In addition, software tools and algorithms have been developed to automate the process of element identification and quantification based on spectral analysis [14, 15].

Chapter 3

Deep Learning

3.1 Introduction

Deep Learning has risen to prominence thanks to a confluence of factors that have transformed the landscape of artificial intelligence. The sheer availability of massive datasets generated in the digital age, coupled with significant advancements in hardware capabilities, has paved the way for Deep Neural Networks (DNNs) to emerge. Breakthroughs in algorithmic techniques, from novel architectures to optimization methods, have made it possible to train and deploy deep networks with unprecedented efficiency. Real-world applications across domains such as computer vision (CV) [16, 17, 18], natural language processing (NLP) [19, 20], and autonomous systems, have showcased the transformative potential of deep learning, further fueling its popularity. Additionally, the rise of transfer learning [21, 22], open-source frameworks, a vibrant research community, and industry giants' substantial investments have collectively propelled deep learning into the mainstream, making it an essential tool for tackling complex and diverse challenges in the modern era.

3.2 Deep Neural Networks

3.2.1 Representation Learning

Deep Neural Networks, constitute an expansion of Feedforward Neural Networks characterized by their depth, with multiple hidden layers between the input and output layers (Figure 3.1). In a DNN architecture, each hidden layer consists of a collection of neurons or units, followed by a non-linear activation function, and these layers are densely connected through weighted connections.

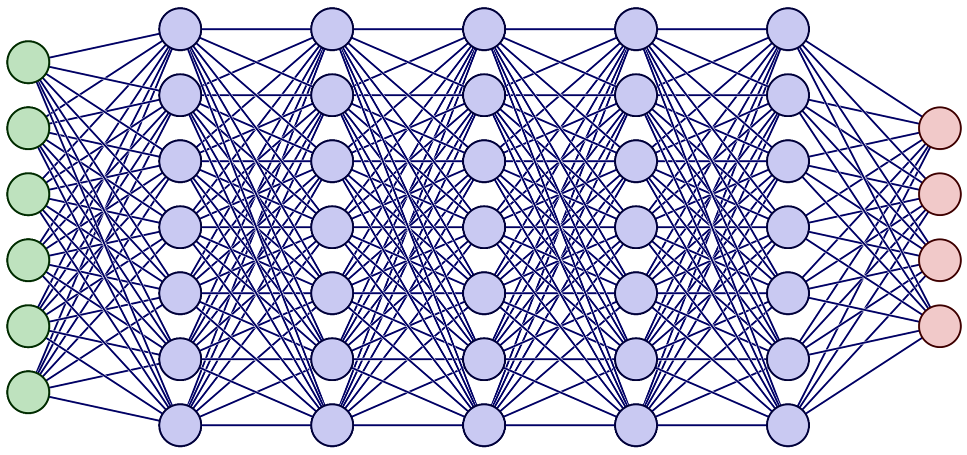


Figure 3.1: A Deep Neural Network consisting of five hidden layers.

Deep networks trained in a supervised manner can be thought as performing a kind of **representation learning**. The hidden layers learn to provide a representation to the output layer, which is typically a linear classifier, responsible for predicting the correct outputs for the given inputs. Training using a supervised criterion inherently causes the representations in each hidden layer to acquire characteristics that facilitate the learning process. To illustrate, inputs belonging to classes that are not linearly separable in their features, probably will become linearly separable in the last hidden layer, through the effective representations derived from the hidden layers outputs.

Supervised training of feedforward networks does not involve explicitly imposing any condition on the learned intermediate features. Other kinds of representation learning algorithms are often explicitly designed to shape the representation in some particular way [23]. However, a tradeoff between preserving as much information about the input as possible and achieving desirable representation properties is encountered in most representation learning problems. Efficiently learning representations of input data is of particular interest, due to its potential to enable unsupervised and semi-supervised learning, as noted in [24], stemming from the availability of abundant unlabeled training data and limited labeled training data.

While supervised training of feedforward networks doesn't explicitly impose conditions on learned intermediate features, other representation learning algorithms, such as **Autoencoders**, are designed with the explicit purpose of shaping representations in a specific manner. Autoencoders, a popular approach in representation learning, employ a deep network architecture (Figure 3.2) that consists of an **encoder** and a **decoder**. The encoder compresses the input data into a lower-dimensional representation, often called a bottleneck layer or latent space, while the decoder aims to reconstruct the input from this representation. This process encourages the network to capture meaningful features in the data, striking a balance between information preservation and the creation of useful representations. Similar to other representation learning paradigms, Autoencoders play a pivotal role in leveraging large volumes of unlabeled data for various machine learning tasks, including unsupervised and semi-supervised learning.

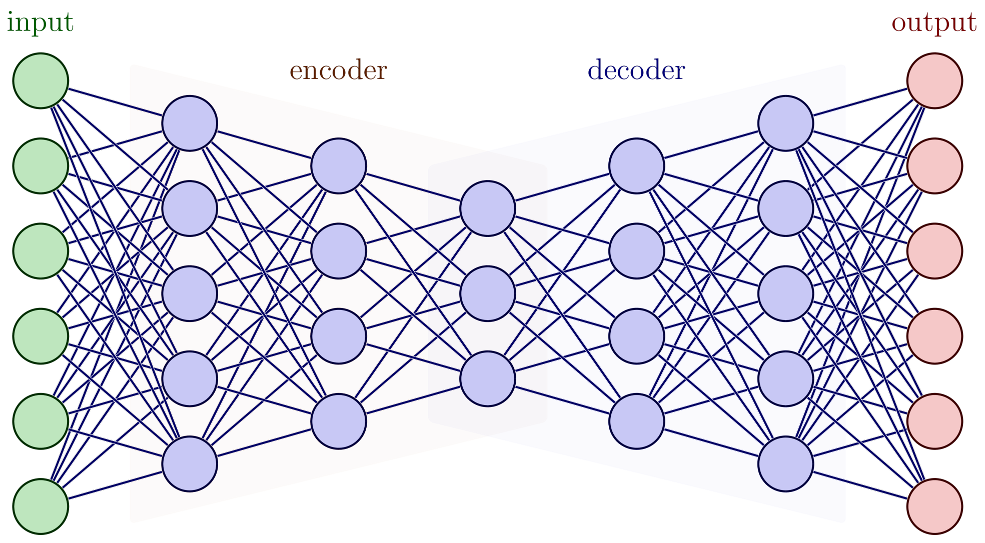


Figure 3.2: The autoencoder architecture.

In summary, representation learning is a key concept in deep neural networks. While supervised training naturally fosters helpful representations, techniques like Autoencoders intentionally shape them. This balance between preserving information and creating useful representations is vital for advancing machine learning, especially in scenarios with ample unlabeled data. Representation learning continues to drive improvements in data-driven decision-making.

3.2.2 Back-Propagation

When a DNN is fed with an input $\boldsymbol{x}$ and produces an output $\hat{\boldsymbol{y}}$, the information flows forward through the network's hidden layers and produces a scalar cost $J(\boldsymbol{\theta})$. This process is called **forward propagation**. The **back-propagation** algorithm [25], allows the information from the cost J to then flow backwards through the network, in order to compute the gradient. Deriving an analytical expression for the gradient is a straightforward task, yet numerically assessing this expression can be computationally demanding. The back-propagation algorithm, on the other hand, accomplishes this efficiently through a straightforward and cost-effective algorithm that automatically computes the gradient.

Neural networks can be observed as computational graphs where the individual computational units or nodes are directed connected with other nodes, all responsible for performing operations, such as summations, multiplications and in general mathematical computations. If a variable y is computed by applying an operation to a variable x, then a directed edge from x to y can be defined. The chain rule of calculus finds application in computing the derivatives of functions created by composing other functions with known derivatives. Back-propagation is an algorithm designed to effectively compute the chain rule, employing a specific sequence of operations that optimizes efficiency.

Given J and a deep neural network with parameters $\boldsymbol{\theta}$, the algorithm iteratively computes the gradients as presented in Equation 3.1.

$$\frac{\partial J}{\partial \boldsymbol{\theta}} = \frac{\partial J}{\partial f^{H+1}} \cdot \frac{\partial f^{H+1}}{\partial f^{H}} \cdot \frac{\partial f^{H}}{\partial f^{H-1}} \cdot \ldots \cdot \frac{\partial f^{2}}{\partial f^{1}} \cdot \frac{\partial f^{1}}{\partial \boldsymbol{\theta}} \tag{3.1}$$

where $\frac{\partial J}{\partial f^{H+1}}$ represents the gradient of the cost function with respect to the network's output (f^{H+1}), and $\frac{\partial f^{H}}{\partial f^{H-1}}$ represents the gradient of the H-th layer output (f^{H}) with respect to the $(H-1)$-th layer output (f^{H-1}). This process is repeated layer by layer, propagating the gradients backward through the network to adjust the parameters $\boldsymbol{\theta}$ and improve its performance during training.

The update rule for parameters $\boldsymbol{\theta}$ of the hidden layer f^{i} in a neural network using backpropagation with gradient descent formulated in Equation 3.2,

$$\theta^{i}_{new} = \theta^{i}_{old} - \alpha \nabla_{\theta^{i}} J(\theta^{i}_{old}) \tag{3.2}$$

where θ^{i}_{new} and the θ^{i}_{old} represents the updated and the current parameters of the hidden layer f^i respectively. α is the **learning rate**, hyperparameter that controls the step size during the update. ∇_{θ^i} represents the gradient with respect to the parameters θ^i and $J(\theta^i_{old})$ the cost function associated with the network, which is typically calculated using the forward and backward pass through the network.

3.2.3 Regularization

A central problem in deep neural networks is how to learn them to perform well not just on the training data, but also on new unobserved inputs. Many Machine Learning strategies are explicitly designed to minimise test error, potentially leading to an increase in training error. These strategies, collectively known as regularization, aim to improve model generalization.

Many regularization approaches are based on limiting the capacity of deep models, by adding a parameter norm penalty $\Omega(\boldsymbol{\theta})$ to the cost function J which was expressed in Equation 3.3.

$$\widetilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \alpha\Omega(\boldsymbol{\theta}) \tag{3.3}$$

The hyperparameter $\alpha \in [0, \infty)$ is responsible for weighting the relative contribution of the norm penalty term, Ω, relative to the standard cost function J, where setting α to 0 results in no regularization. When the training algorithm minimises the regularised objective function $\widetilde{J}$, it reduces both the initial objective function J on the training data and a measure of the parameter $\boldsymbol{\theta}$. Equation 3.4 presents one of the most important parameter norms that are widely used for deep networks regularization, the L^2 regularization also known as ridge regression.

$$\Omega(\boldsymbol{\theta}) = \frac{1}{2}\|\mathbf{w}\|_2^2 \tag{3.4}$$

Another way for a ML model to achieve generalization, as many researchers claim, is to train it on more data, although in practice, the amount of available data is limited. One way to overcame such an obstacle is to create and incorporate fake data into the training set and in this way to increase the diversity in the data. **Dataset augmentation** has been a particularly effective technique for difficult classification problems like object recognition. A commonly used technique in Computer Vision and Natural Language Processing is the generation of new data through transformations applied to existing data. Such transformations could include image manipulations such as rotation, translation, scaling, and flipping, among others. However, it is important to exercise caution to ensure that transformations do not result in misclassifications (e.g. horizontally flipping a ‘6’ may result in a ‘9’).

A modern and promising approach for generating new fake data is through the use of a special category of ML models, the Generative models. These models learn to generate synthetic data that resembles real data but is not actually derived from real-world examples. The two main types of generative models are Variational Autoencoders (VAEs) [26] and Generative Adversarial Networks (GANs) [27]. These models aim to understand the underlying probability distribution of the training data, allowing them to generate new outputs based on what they have learned. Instead, they learn to approximate it through the neural network architecture and optimization processes used during training. The focus is on generating data samples that are statistically similar to real data, rather than providing an analytical probability density function (PDF) formula.

Probably, the most commonly used form of regularization in deep learning is a strategy called **early stopping** and it is popular due both to its effectiveness and its simplicity. Specifically, researchers commonly define a validation subset, which is a portion of the training subset excluded from both model training and testing. The model’s performance is monitored on this validation set during training, and the error or loss on this validation set is referred to as the **validation error**. Early stopping involves terminating the training process when the validation error starts to increase, indicating that the model is beginning to overfit the training data. This prevents the model from learning noise in the data and helps it generalize better to unseen data.

Another widely used technique for achieving better generalization in deep learning is **dropout**. Dropout is a regularization method that involves randomly setting a fraction of the neurons in a neural network to zero during each training iteration. This effectively turns off some neurons, making the network more robust and preventing it from relying too heavily on any particular set of neurons. Dropout encourages the network to learn more robust and distributed features, reducing the risk of overfitting. During inference (when the model is used for predictions), dropout is typically turned off. This simple yet effective technique has proven to be highly successful in improving the generalization of deep neural networks.

In summary, generalization techniques play a crucial role in enhancing the performance of deep learning models. These techniques are not mutually exclusive and can often be combined to further improve a model's ability to generalize effectively on unseen data.

3.3 Convolutional Neural Networks (CNNs)

3.3.1 Introduction

Convolutional neural networks, also known as CNNs are a specialized class of neural networks designed for processing structured data objects [28]. These objects often exhibit a grid-like structure, with examples including time-series data, represented as a 1D grid sampled at regular intervals, and image data, portrayed as a 2D grid of pixels. CNNs have proven highly effective in practical applications. The term "convolutional neural network" signifies that these networks utilize a mathematical operation called convolution, which is a specialized form of linear operation. Essentially, convolutional networks are deep neural networks that incorporate convolution instead of traditional dot product multiplication in at least one of their layers.

3.3.2 Convolutional Layer

In the realm of convolutional neural networks (CNNs), the convolutional layer is a fundamental building block responsible for extracting features or representations from input data in a manner inspired by the mathematical operation of **convolution**. In its simplest form, convolution is the result of sliding a small window, known as a *kernel* or *filter*, over the input data and computing the element-wise dot product at each position. It's important to note that convolutional layers exhibit a unique property known as *weight sharing*, where the same set of weights (kernel) is used at different spatial positions of the input. This operation is visualized in Figure 3.3 for 1D data. Importantly, convolution, as a linear operation, is differentiable, allowing for the use of gradient-based optimization during training.

In practice, a convolutional layer often consists of multiple convolutional *filters* applied in parallel to the input data. Each filter captures different patterns or features from the input. The results from these filters are then combined, typically through element-wise addition, and passed through a non-linear activation function such as the rectified linear unit (ReLU) to form a **convolutional block**.

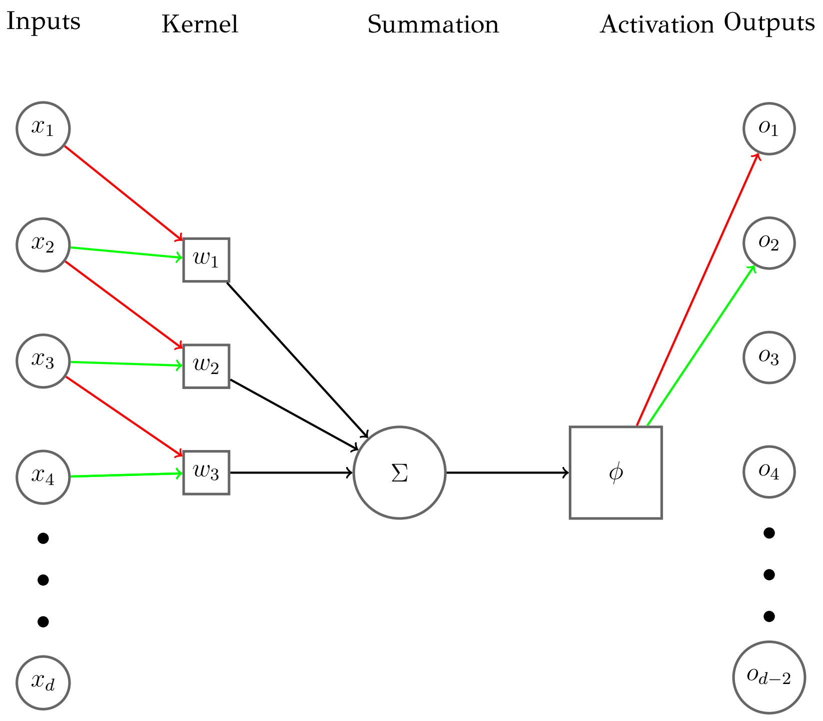


Figure 3.3: The convolution operation

Three crucial parameters influence the behavior of a convolutional layer:

- *Kernel*: The kernel defines the size and shape of the local receptive field. It determines which features the filter extracts from the input.
- *Stride*: Stride specifies the step size at which the kernel slides over the input. A larger stride reduces the spatial dimensions of the output feature map.
- *Padding*: Padding involves adding extra values (often zeros) around the input data to control the spatial dimensions of the output feature map. Padding can be 'valid' (no padding) or 'same' (padding to preserve input dimensions).

The convolutional layer's ability to automatically learn and extract hierarchical features from data makes it a cornerstone of deep learning architectures, especially in computer vision tasks.

3.3.3 Pooling Layer

In CNNs, the pooling layer is a crucial component that serves to downscale feature maps generated by preceding convolutional layers. Pooling operations are employed to reduce spatial dimensions while retaining essential information, leading to more efficient computation and translational invariance.

There are two primary types of pooling commonly used:

- **Max Pooling**: In max pooling, a small window, known as the pooling kernel or filter, slides over the input feature map and selects the maximum value within the window at each position, discarding the rest. This operation focuses on preserving the most prominent features within each region.
- **Average Pooling**: Average pooling computes the average value within the pooling window, resulting in a smoother downsampling process. It provides a form of translational invariance while being computationally less intensive than max pooling.

Pooling layers offer several benefits in CNNs:

- **Dimension Reduction**: Pooling reduces the spatial dimensions of feature maps, which subsequently reduces the computational complexity and memory requirements of the network.
- **Translation Invariance**: By selecting key values, pooling layers make the network less sensitive to small spatial shifts or variations in the input data.
- **Hierarchical Feature Abstraction**: As CNNs progress through multiple pooling layers, they capture increasingly abstract and high-level features from the input.

The key parameters for pooling layers include the size of the pooling window and the stride, which determine how the pooling operation is applied to the input data. Additionally, variants of pooling, such as global average pooling, have been introduced and found applications in various network architectures.

In combination with convolutional layers, pooling layers enable CNNs to efficiently learn hierarchical features from data, making them indispensable in numerous computer vision and deep learning applications.

3.3.4 CNN Architecture

The architecture of CNNs is also characterized by a hierarchical arrangement of layers (Figure 3.4), each designed to capture and transform features from the input data progressively. At its core, a typical CNN consists of three main types of layers: *convolutional layers*, *pooling layers*, and *fully connected layers*.

Fully connected layers, also known as dense layers, are traditionally placed at the end of the CNN architecture. They serve to consolidate the extracted representations and provide the final classification or regression output. These layers are fully connected, meaning that each neuron in a fully connected layer is connected to every neuron in the previous layer, like the classic feedforward neural networks.

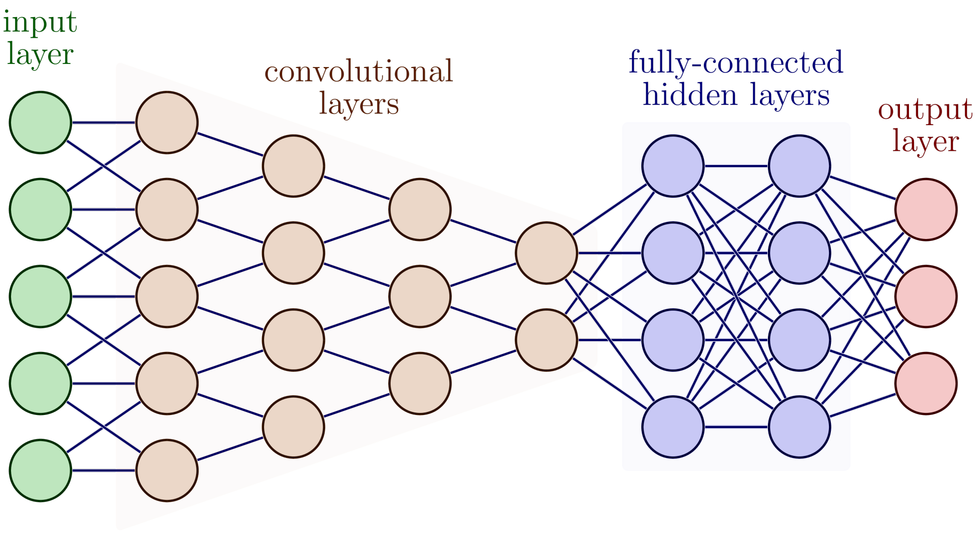


Figure 3.4: A CNN is depicted visually, composed of four convolutional layers and two fully-connected layers. The first layer has a kernel size of 3, while the remaining have a kernel size of 2.

The combination of these layers forms a deep neural network that is particularly effective for tasks involving structured data, such as image recognition and natural language processing. The precise arrangement and complexity of CNN architectures can vary widely based on the specific task and dataset, with deeper networks often capturing more abstract and high-level features.

Chapter 4

Elemental Distribution Mapping with Deep Neural Networks

4.1 Introduction

Elemental distribution mapping or **elemental mapping**, is a crucial analytical technique in the field of materials science and related disciplines. It involves the spatial visualization and quantification of the distribution of chemical elements within a sample, providing invaluable insights into the composition and structure of materials. The utility of elemental mapping lies in its ability to unveil intricate details about the elemental composition of materials, enabling researchers to make informed decisions, optimize processes, and gain a deeper understanding of the underlying physical and chemical properties [29, 30, 31]. By producing high-resolution elemental maps, scientists can identify impurities, defects, or gradients within materials, which is essential for quality control and the development of advanced materials with tailored properties.

However, the practical implementation of elemental mapping in real-world scenarios poses significant challenges. Traditional elemental analysis methods, such as electron probe microanalysis (EPMA) [32] and energy-dispersive X-ray spectroscopy (EDS) [33], require manual operation, are time-consuming, and often demand a high level of expertise. Additionally, these methods may be limited in their ability to handle complex sample geometries and large datasets efficiently. The automation of elemental mapping processes is thus imperative to overcome these challenges.

There is a growing need for automation to tackle these challenges and improve the efficiency of elemental mapping. Deep Neural Networks have emerged as a promising solution for automating complex processes. By harnessing their capabilities, it is possible to significantly enhance the speed and precision of elemental mapping.

4.2 Problem Definition

In the context of this study, elemental mapping is formally defined as a **Multi-output Regression Problem**. This characterization encapsulates the fundamental objective of estimating and quantifying the concentrations of multiple chemical elements within each pixel or region of interest in an image or sample. These concentrations are determined from the counts of specific X-ray fluorescence characteristic transitions, where a high count indicates a high concentration of the corresponding element. Term ”Multi-output” is aptly chosen, signifying that the predictive model must simultaneously handle and predict accurately the overall concentration (or total number of counts) for all the target elements, resulting in a multi-dimensional output space. Specifically, the problem can be formulated as follows:

$$Y = f(X) + \varepsilon$$

where X denotes the input data, consisting of individual spectra obtained from the spectroscanning mechanism and Y the corresponding ground-truth elemental concentrations. $f(\cdot)$ is the mapping function implemented by the Deep Artificial Neural Networks (DNNs) to predict elemental concentrations while ε accounts for the residual error or noise in the predictions.

To address this complex challenge, we employ Deep Artificial Neural Networks (DNNs), recognized for their ability to handle intricate, high-dimensional datasets effectively. The inputs comprise of individual spectra corresponding to each pixel from the spectro-scanned image, while the DNNs are responsible for predicting the element concentrations for the corresponding spectra. The concentration values are non-negative integers, since they are defined as the total counts of the characteristic transitions.

Evaluating the performance of DNNs in the context of elemental mapping hinges on establishing criteria for a "good" network. In practical terms, the quality of a network is assessed based on its capacity to provide accurate estimates of elemental concentrations across the entire concentration range. While identifying and quantifying higher concentrations of elements is generally less challenging, particular attention must be dedicated to lower concentration levels. Lower concentrations pose a challenge because they are often obscured by noise and overlapped by other elements, making their accurate identification and estimation a tough task even for seasoned experts in the field. Thus, the effectiveness of a network is measured, in part, by its ability to accurately estimate elements, particularly in scenarios involving low concentrations.

4.3 Deep Networks Models

The primary Machine Learning models being considered for the prediction of Elemental Distribution Maps are the essential Fully Connected Model (FCN) and a 1D Convolutional Neural Network (1DCNN).

The intuition for studying the FCN arises from the fact that in FCNs first hidden layer, the weights of each neuron are directly connected to corresponding input values from the spectra. This design allows each neuron to act as a spectral signature identifier, recognizing intricate patterns and correlations within the spectral data. The most effective FCN architecture identified through the experimentation process is visually depicted in Figure 4.1. The FCN architecture proposed includes multiple layers. The input layer is customized to accommodate the 4096 total features extracted from the input spectra, as detailed in the dataset description presented in Section 5.1. In addition, the architecture includes three hidden layers. The first hidden layer yields

512 features, while the subsequent two yield 64 features each. The output layer is configured to yield k features, where k corresponds to the total number of elemental concentration values that need to be predicted. Notably, each of the hidden layers employs the ReLU activation function, introducing non-linearity to the intermediate representations within the network.

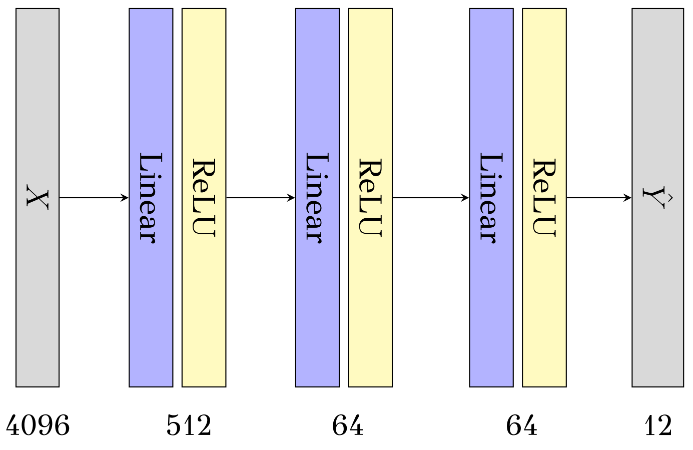


Figure 4.1: The architecture of Fully Connected Network (FCN).

Regarding the proposed 1DCNN architecture, it comprises a sequence of five one-dimensional convolutional layers tailored to effectively extract valuable features from the input spectra, as revealed by the authors' research at [34].

Specifically, the initial convolutional layer takes in a single input channel and yielding an output of 64, employing a kernel size of 5 and a stride of 2 for effective capture of local patterns. Subsequently, three more convolutional layers follow suit, each taking in 64 input channels and producing 64 output channels, utilizing smaller kernel sizes of 3 and a stride of 2 to further enhance the learned representations. Finally, a fifth convolutional layer takes 64 input channels and yields 128, employing the same kernel size and stride like the previous ones. Throughout the network, the ReLU activation function is consistently applied to introduce non-linearity into the intermediate representations.

Additionally, to mitigate overfitting during training, a dropout layer with a dropout probability of 0.05 is incorporated. As is customary in convolutional neural networks, a flattening step follows the convolutional layers to convert the series of representations into a singular vector. Lastly, a linear layer is used for the output layer. The network architecture also integrates max-pooling with a kernel size of 2 and a stride of 2 to reduce the spatial dimensions of the intermediate output representations, facilitating the extraction of prominent features. The overall architecture of the proposed 1DCNN can be seen in Figure 4.2

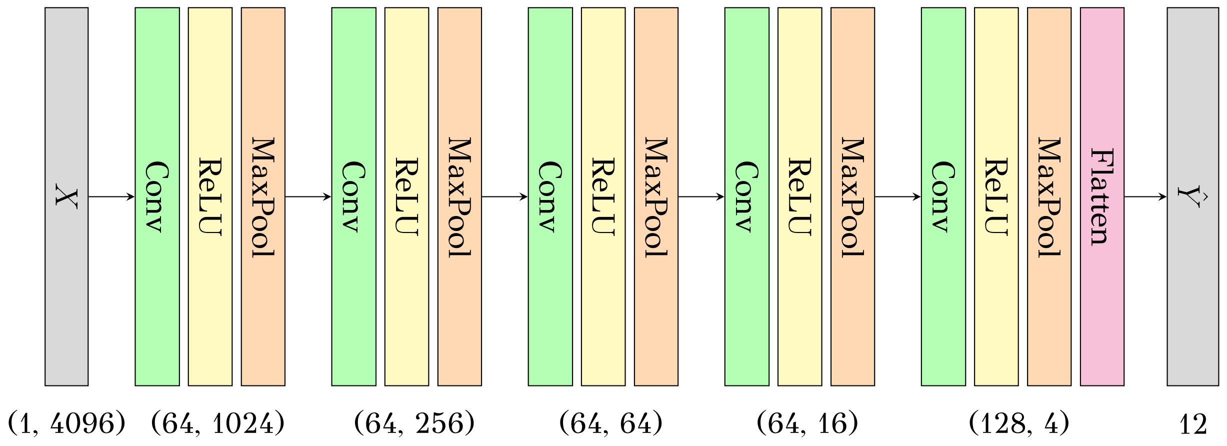


Figure 4.2: The architecture of 1D Convolutional Neural Network (1DCNN).

4.4 Incorporating Physics-based Prior Knowledge

Throughout the experimentation, particularly in evaluating the deep networks detailed in Section 4.3, it was apparent that these networks have encountered difficulty in precisely estimating elemental concentrations in areas of paintings with extremely low levels.

During experimentation a noteworthy observation emerged: during inference, the neural network exhibited a tendency to overlook the low peaks in the input spectra. As a result, significant portion of vital information contained within individual spectra remained underutilized by the trained networks. This can be attributed to the standard training process of neural networks, which relies on gradient-based algorithms to minimize a loss function. When certain peaks, particularly the high peaks, have a greater influence on the loss function, the associated gradients become larger. Consequently, the network prioritizes the learning of patterns associated with these high peaks, inadvertently relegating the significance of the lower peaks.

The significance of lower peaks in MA-XRF analysis cannot be underestimated. These lower peaks, though less noticable, play a crucial role in characterizing the elemental composition of the spectra. To grasp their importance, it's essential to consider the concept of the pure spectrum, which introduced in Section 2.4. As can be deduced from Figure 4.3, pure spectra function as elemental signatures, encapsulating vital information about the height and width of the peaks, as well as the precise energies at which they are positioned. While the high peaks may dominate the overall signal, the lower peaks provide essential fine-grained details that aid in distinguishing between the different elements. In essence, they act as indicators that contribute to the comprehensive understanding of the spectral data, ensuring that no subtleties in the elemental composition go unnoticed.

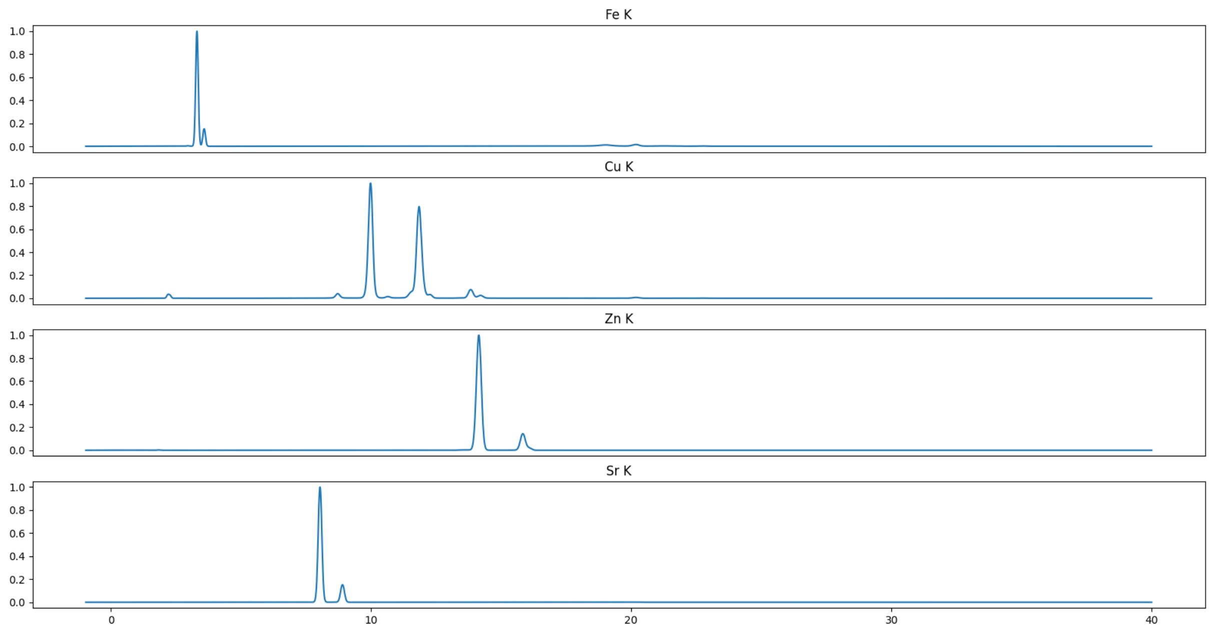


Figure 4.3: Pure spectra visualization of iron (Fe K), copper (Cu K), zinc (Zn K) and strodium (Sr K).

In this study, an innovative approach is introduced that utilizes the concept of pure spectra to enhance the efficiency and accuracy of neural networks in predicting elemental concentrations within a given sample. To the best of current knowledge, this represents the first attempt to incorporate pure spectra for the improvement of elemental concentration estimation.

The rationale behind this approach is based on the intuition of computing the dot product between real spectrum (comprising a mixture of elemental signals) and a normalized pure spectrum. This operation serves as a measure of alignment or similarity between the observed spectra and the idealized elemental signature. The outcome of this procedure offers a numeric indication of a specific element's contribution to the given sample. High dot product outcomes suggest increased concentration of the corresponding element. This is shown in Figure 4.4, which visualize the dot product

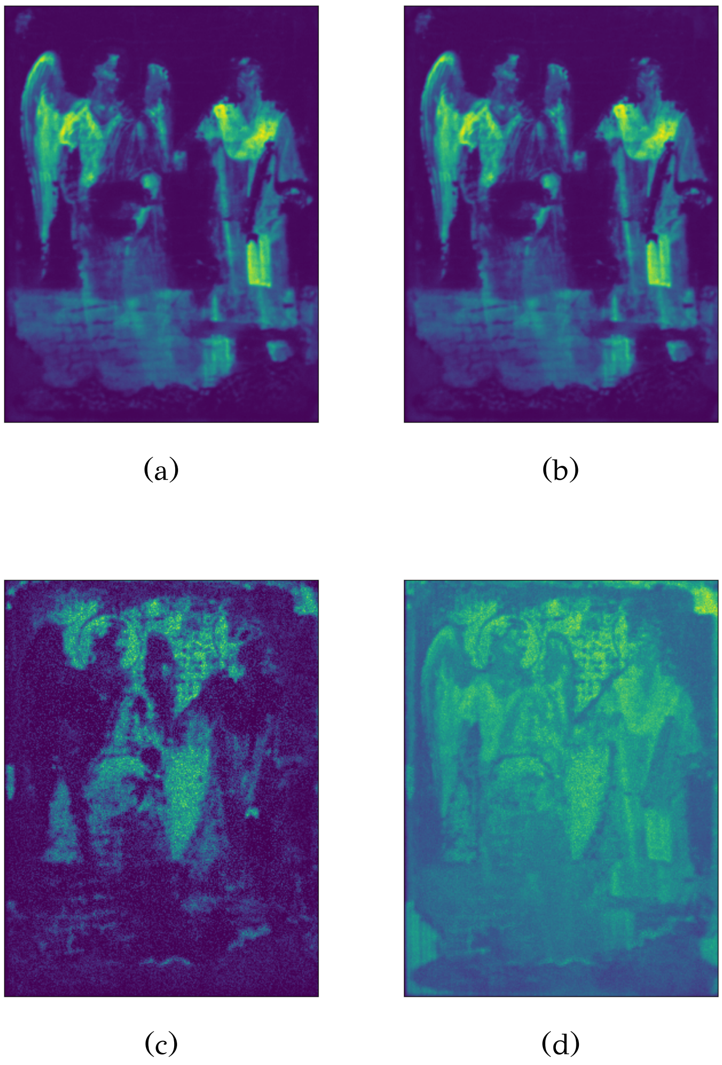


Figure 4.4: Comparison of Expert-Derived Pb (a) and S (c) Distribution Maps with Proposed Dot Product Results (b), (d).

result between the spectra/pixels of an X-ray scanned painting and the corresponding pure spectra of lead (Pb) and sulfur (S).

This idea is implemented by extending the architectures of both the FCN and 1DCNN through the incorporation of an additional layer. Specifically, we introduce the prior layer, which contains pre-existing and untrainable weights $\boldsymbol{W} \in \mathbb{R}^{d \times l}$, where d represents the dimensionality of the input spectra, and l signifies the total number of spectral signatures currently available. In prior layer's weight matrix, each row corresponds to the pure spectral signature of a respective element. The placement of prior layer occurs immediately preceding the initial hidden layer in both networks; however, the forward process varies between the two.

In FCN, the forward process of the prior layer remains consistent with that of traditional linear layers, where it computes the dot product between the input and weight matrix $\boldsymbol{W}$, yielding a l-dimensional vector $\boldsymbol{o} \in \mathbb{R}^l$. Each value in this vector denotes the level of alignment between the input and corresponding weight. Thus, in such a representation effectively indicates the degree to which each element is present in the input.

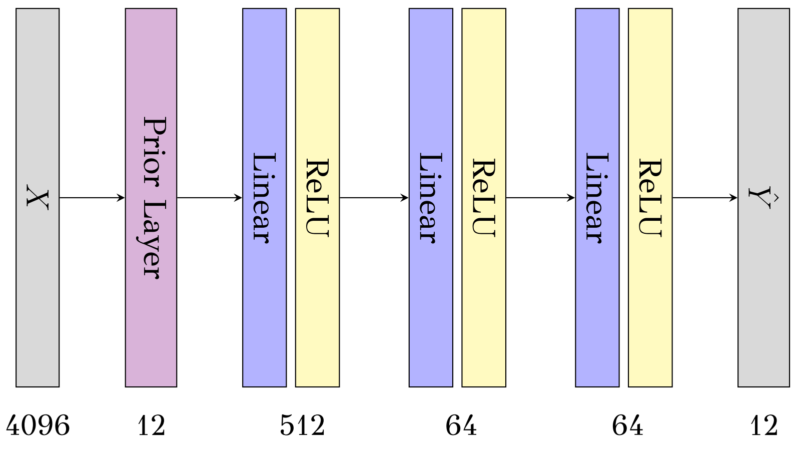


Figure 4.5: The architecture of Fully Connected Network (FCN+) featuring the Prior Layer.

For the 1DCNN, a small modification has been made to the forward process in order to effectively apply convolution operations that occur after the prior layer. These changes have been made so that the multiplicative operations performed between the input and weight values do not result in a sum, as is the case in dot product. Instead, these operations produce distinct values placed in a new vector. Therefore, the prior layer generates l representations or feature vectors, denoted as $o_i \in \mathbb{R}^d$, corresponding to the presence of certain elements within the input spectrum. Subsequently, these l generated representations are concatenated with the input spectrum, resulting in a total of $l + 1$ input representations. They are then fed into the 1DCNN for additional processing during the feed-forward phase.

Figures 4.5 and 4.6 present the overall architecture of FCN and 1DCNN respectively.

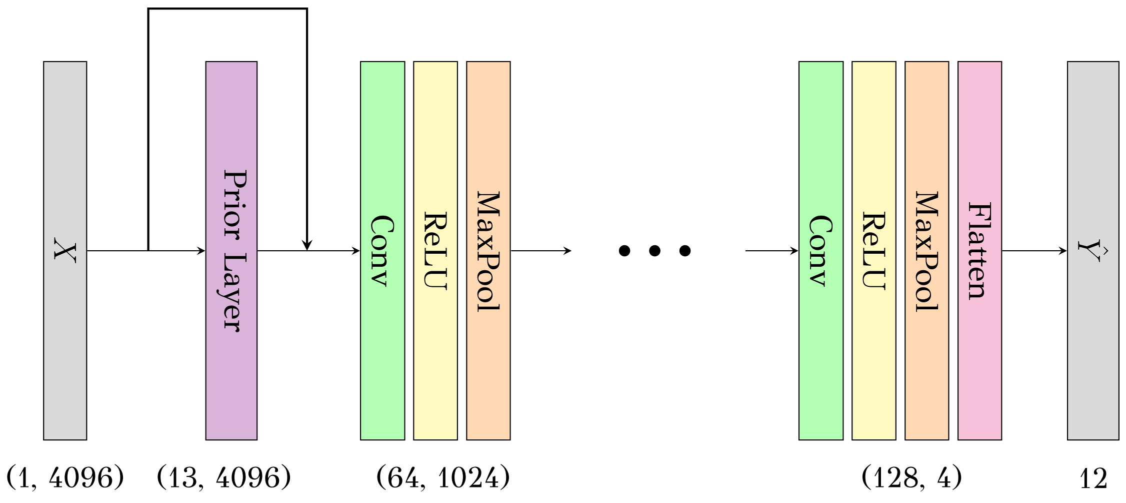


Figure 4.6: The architecture of 1D Convolutional Neural Network (1DCNN+) featuring the Prior Layer.

4.5 Evaluation Measures

In assessing the performance of the elemental mapping models, four evaluation measures have been computed to gauge their effectiveness.

The initial measure for evaluating the models' effectiveness is the **Structural Similarity Index (SSIM)**. The measure is based on the principle that the human visual system is highly adapted to recognize subtle changes in an image's luminance (l), contrast (c), and structure (s) [35]. SSIM ranges from -1 (entirely dissimilar images) to 1 (completely identical images) and is presented as follows:

$$\mathrm{SSIM}(y_{pred}, y_{real}) = \frac{(2 \cdot \mu_{y_{pred}} \cdot \mu_{y_{real}} + c_1) \cdot (2 \cdot \sigma_{y_{pred}y_{real}} + c_2)}{(\mu^2_{y_{pred}} + \mu^2_{y_{real}} + c_1) \cdot (\sigma^2_{y_{pred}} + \sigma^2_{y_{real}} + c_2)}$$

where $\mu_{y_{pred}}$, $\sigma_{y_{pred}}$, $\mu_{y_{real}}$, $\sigma_{y_{real}}$, are the means and standard deviations of y_{pred} and y_{real} respectively. $\sigma_{y_{pred}y_{real}}$ is the covariance of y_{pred} and y_{real}, and c_1, c_2 are constants to avoid instability when the denominator approaches zero.

The second evaluation measure is the **Pearson Correlation** between y_{pred} and y_{real}, so to quantify the linear relationship between the predicted elemental concentrations and the actual ones. It provides information about the strength and direction of the linear association. Certainly, a good value for the Pearson Correlation coefficient (r) is one that is closer to 1, indicating a stronger positive linear relationship between the predicted and actual elemental concentrations, where the predictions closely align with the ground truth values. The Pearson Correlation coefficient (r) is calculated as:

$$r(y_{\text{real}}, y_{\text{pred}}) = \frac{\sum (y_{\text{real}_i} - \bar{y}_{\text{real}})(y_{\text{pred}_i} - \bar{y}_{\text{pred}})}{\sqrt{\sum (y_{\text{real}_i} - \bar{y}_{\text{real}})^2 \sum (y_{\text{pred}_i} - \bar{y}_{\text{pred}})^2}}$$

where y_{real_i} and y_{pred_i} represent individual data points in y_{real} and y_{pred}, respectively. $\bar{y}_{\text{real}}$ and $\bar{y}_{\text{pred}}$ denote the means of y_{real} and y_{pred}, respectively.

An additional method of assessing the precision of the predicted concentrations y_{pred} in contrast to the real values y_{real} is to fit them using a **linear relationship** in the following form:

$$y_{pred} = \alpha \cdot y_{real}$$

A good value for the slope α in the linear relationship between predicted concentrations y_{pred} and actual values y_{real} would typically be close to 1.0. This indicates that the predicted concentrations are highly accurate and align closely with the actual values, following a nearly one-to-one relationship. Values significantly different from 1.0, whether smaller or larger, would suggest a deviation from an ideal fit, indicating potential inaccuracies in the predictions.

Finally, it's essential to consider the use of the **absolute z-score measure**, which gauges the accuracy of the predicted values (y_{pred}) in comparison to the actual values (y_{real}). This measure calculates absolute z-score value individually for each pixel p and each transition k, following the methodology outlined in reference [36]:

$$z_k^p = \frac{|y_{pred}^k - y_{real}^k|}{\sigma_{y_{real}^k}}$$

The calculated z-score values (z_k^p) are then grouped into specific ranges, namely, $(0, 1], (1, 2], (2, 3], (3, +\infty)$. The number of z-score values that fall within the $[0, 1)$ range serves as a crucial indicator of model performance – a higher count in this range implies better model accuracy and closer agreement between predicted and actual values.

4.6 The Objective Function

In the quest to optimize the model's training process, different loss functions were carefully considered, including the conventional $\mathcal{L}_{\text{L1}}$ and $\mathcal{L}_{\text{L2}}$ variants. However, during experimentation, a notable challenge arose: the precise estimation of low-concentration values using conventional loss functions proved to be a difficult task. Despite efforts, it was observed that L1 and L2 loss functions were inadequate in capturing the nuances of the data, particularly when dealing with values at the lower end of the concentration spectrum.

In response to this issue, a different approach was implemented by introducing the $\mathcal{L}_{\mathrm{aL1}}$ loss function into the model training regimen (Equation 4.1). This decision was driven by the recognition of the need for adaptability in the loss function to account for the inherent variability and uncertainty within the dataset. The $\mathcal{L}_{\mathrm{aL1}}$ loss function uniquely addresses these challenges by normalizing the absolute differences between the true target values (y_i) and the model's predictions ($\hat{y}_i$) based on the square root of the true target values.

$$\mathcal{L}_{\mathrm{aL1}} = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y}_i|}{\sqrt{y_i}} \tag{4.1}$$

This adaptive loss function excels in scenarios where traditional loss functions fall short, especially when dealing with low-concentration values. By placing more emphasis on improving accuracy for smaller and more uncertain true values, the $\mathcal{L}_{\mathrm{aL1}}$ loss function has significantly improved the model's performance.

Chapter 5

Experiments

5.1 Datasets

Training Deep Learning models poses challenges not only in enhancing existing methodologies and algorithms but also in harnessing a substantial volume of representative data to achieve optimal generalization performance when confronted with previously unobserved data. A high-quality dataset for deep learning should encompass a diverse range of examples, ensuring it captures the inherent variability in the target domain. Furthermore, the dataset's size and scale should align with the complexity of the problem at hand, striking a balance between computational feasibility and the model's capacity to learn intricate patterns.

For the experiments, five Greek paintings, dating from the 18th to the 19th centuries, were carefully selected (Figure 5.1). This dataset encompasses a total of 589,068 spectra, representing the entirety of pixels within the images as captured by the spectro-scanning mechanism, courtesy of The Ceramics & Composites Laboratory (CCL) within the Department of Material Science and Engineering at the University of Ioannina [37]. Specifically, the dataset comprises:

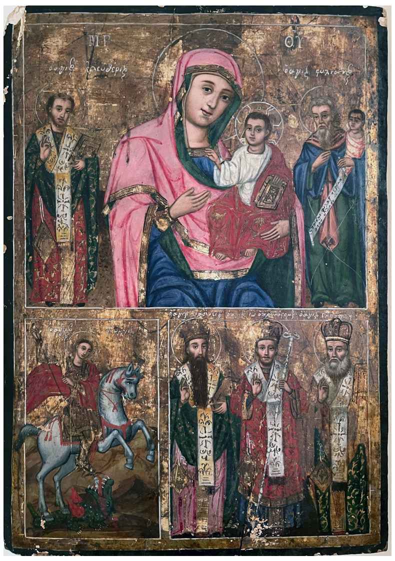
1. A mid-19th-century painting showing a Deesis scene (upper part) along with St Georgios, “the dragon-slayer,” and the Three Hierarchs (lower part).
2. A late 19th-century painting featuring St. Fanourios.
3. A 19th-century painting presenting St. Dionisios.
4. A painting from the mid-18th century depicting the Virgin Mary Hodigitria, commonly known as ’She who shows the way’ [39].
5. An early 19th-century, severely decayed painting depicting St. John the Forerunner and a Hierarch [38].

It is worth noting that these paintings were created using techniques and materials akin to those employed by medieval painters. The process involved applying egg tempera paints and delicate metallic leaves onto gessoed wooden panels [40, 41], distinguishing them from the prevailing oil-based canvas paintings in contemporary Western European art during the same period.

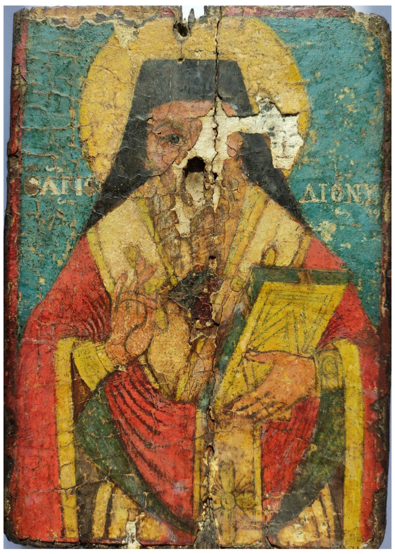
The MA-XRF measurements were carried out utilizing the M6-Jetstream scanner, a product of Bruker [42], which allows scanning of areas up to 80x60 cm^2. Equipped with a 30 W Rhodium X-ray tube, the M6 Jetstream scanner operated with a high voltage of $50kV$ and a current of 600 μA in the present measurements. Notably, no absorption filter was applied to the beam path of the ionization radiation. The incoming X-ray beam from the source was focused using a polycapillary glass optic and directed perpendicularly onto the target surface. Photon detection was performed using a silicon drift detector with a 30 mm^2 active area, providing an energy resolution of 145 eV at the Mn Kα-energy. Each spectrum was comprised of 4096 features. Detailed information regarding the MA-XRF scans and their associated scanning parameters can be found in Table 5.1.

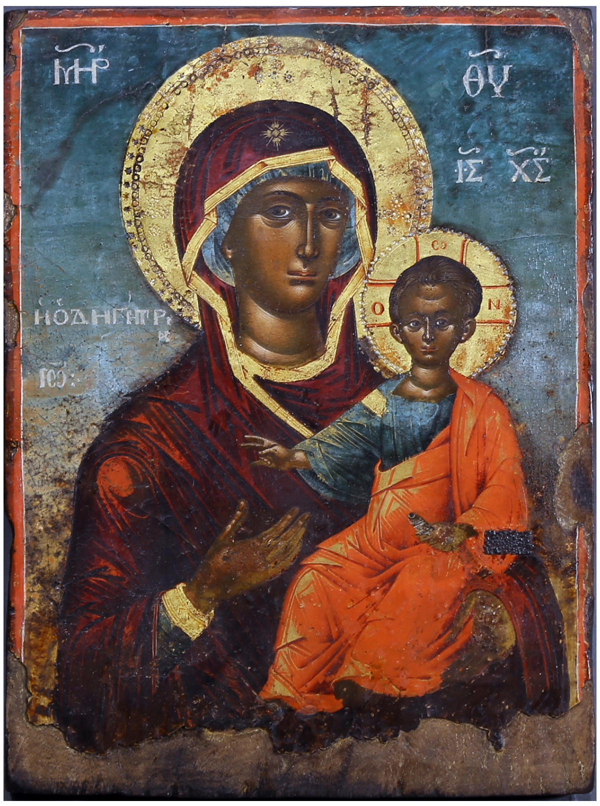
Painting	**Width, Height**	**Pixel size**	**Dwell time (ms/pixel)**	Beam diameter (μm)
1	202, 318	200x200	8	200
2	350, 228	500x500	25	580
3	382, 272	200x200	20	100
4	564, 428	500x500	50	580
5	364, 274	200x200	20	100

Table 5.1: Details of the available MA-XRF images.


(a) Painting 1

(b) Painting 2


(c) Painting 3


(d) Painting 4

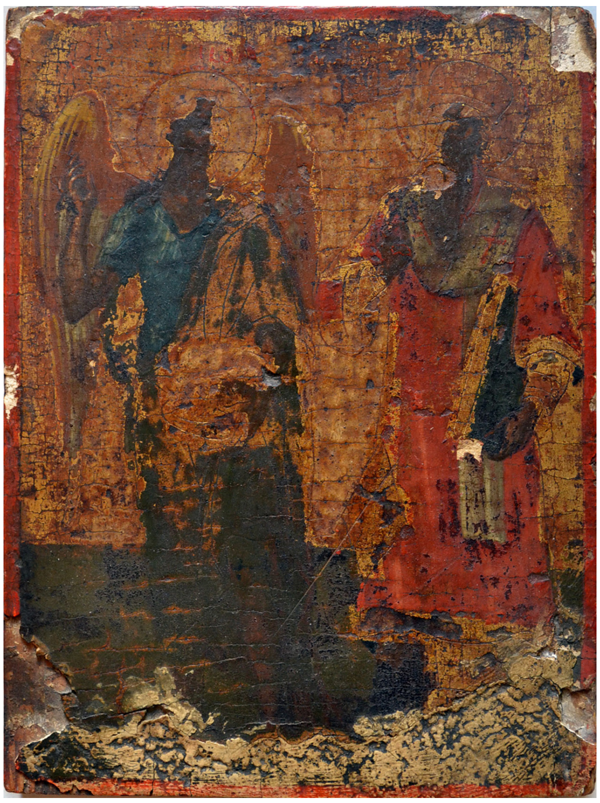

(e) Painting 5

Figure 5.1: The Dataset consists of 5 religious panel paintings.

Regarding the Ground-truth (GT) values that correspond to the count numbers for all K and L characteristic XRF transitions, a careful preparation was carried out by experts using the PyMca code (version 5.6.7) [43]. Twelve chemical elements were analyzed and selected for the present study, which include distinct transitions including the K transition of S, K, Ca, Cr, Mn, Fe, Cu, Zn, and Sr, as well as the L transition of Au, Hg, and Pb. Figure 5.2 illustrates the elemental maps for each characteristic transition of Painting 4, obtained through analysis and normalized to values between 0 and 1.

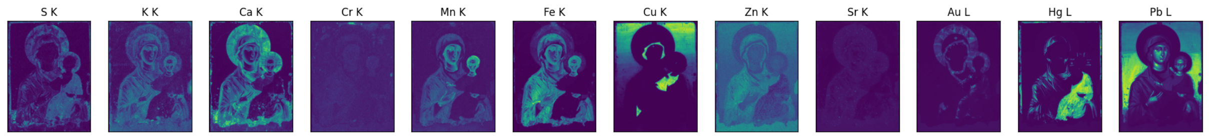


Figure 5.2: Elemental Distribution Maps of Painting 4.

5.2 Experimental Setup

Four distinct deep network architectures are investigated, described in Chapter 4: the Fully Connected Network (FCN), the 1D Convolutional Neural Network (1DCNN), as well as their extended counterparts, FCN+ and 1DCNN+, which incorporate the Physics-Based prior knowledge through the additional Prior Layer. Regarding optimization, the models were trained for 1000 epochs by employing the gradient-based optimization method Adam [44], a widely acknowledged technique within the domain of deep learning. The learning rate, a pivotal hyperparameter essential for the convergence and stability of the training process, is fine-tuned to 0.001.

To ensure precise training of neural networks for accurate predictions of element maps, this study relies on MA-XRF spectra. When working with the five religious panel paintings, each consisting of hundreds of thousands of pixels/spectra, it is imperative to use efficient sampling techniques. One significant reason for employing sampling is the spatial dependence of pixels on their neighboring ones. This inherent spatial correlation can lead to an imbalanced dataset, particularly affecting pixels located on edges and in areas with significant damage or deterioration. To address this issue and make the training process cost-effective and time-efficient, strided sampling with a stride step of 10 is employed. This technique systematically reduces the dataset's size while preserving essential information.

For the purpose of ensuring the accuracy of experimental inferences, a **leave-one-out** technique is implemented [45]. This involves evaluating the model's predictions on an entire, unaltered image, which is maintained as a separate dataset known as the **testset**. Importantly, no form of sampling is applied to this dataset. Regarding the remaining images in the dataset, from which pixels are extracted for training, a different approach is taken. This subset is divided into two distinct datasets: the **trainset** which encompasses 80% of the data, and the **valset**, which accounts for the remaining 20%. To ensure the selection of the best-performing model, a stringent evaluation criterion implemented that relies on validation scores. This validation score corresponds to the objective function discussed in Section 4.6. It is also utilized the early stopping technique, which monitors the validation score during training. The chosen model configuration will be the one that minimizes this score, facilitating optimal generalization on unobserved data. This approach ensures that the model will be fine-tuned to perform well not only on the training data but also on new, unseen data.

Finally, for all the experiments conducted, the PyTorch library [46] is utilized, harnessing the computational capabilities of a CUDA-enabled device for the efficient training and evaluation of deep neural network models.

5.3 Results and Analysis

During the study, various architectures of deep neural networks were tested, along with different initial training conditions and hyperparameter adjustments. The results of the experiments presented in this section are related to the best individual architecture of each model, followed by subsequent comparisons to identify the most efficient method and network in relation to the others.

5.3.1 Painting 4 Evaluation

The evaluation of the models performance on Painting 4 involved the use of all images for training, with Painting 4 kept separate in the testset. The evaluation process for Painting 4 commenced with the utilization of a fundamental and intuitive measure: the comparison of elemental distribution maps predicted by the proposed deep models with the corresponding ground truth maps. The predicted elemental maps for all target elements within Painting 4, which was utilized in testing, are presented in Figure 5.3. It should be noted that the models underwent effective training and careful evaluation on Painting 4, which was not part of the training data. This visual comparison offers valuable insights into the adaptability and generalization capabilities of each model across different contexts and datasets, thereby providing a comprehensive assessment of their overall performance and robustness.

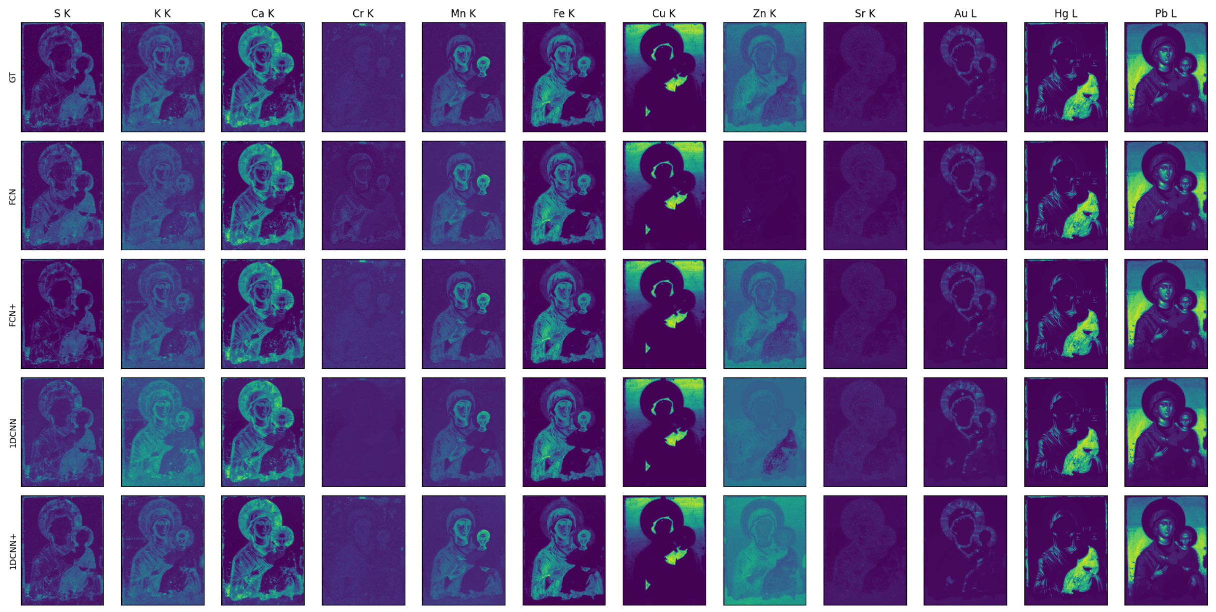


Figure 5.3: Visualization and comparison of predicted and ground-truth Elemental Distribution Maps for Painting 4.

Upon visual inspection, the predicted elemental maps appear to yield satisfactory predictions. However, upon closer examination, notable deviations become evident when comparing these predicted maps with their corresponding ground truth (GT) counterparts. Particularly, in Figure 5.3, significant disparities are noticeable, especially in the case of the K transition elements: potassium (K), sulfur (S), zinc (Zn), and chromium (Cr). These disparities are attributed to the extremely low elemental concentrations, which pose considerable challenges for models in effectively predicting the actual ones.

Nonetheless, it is essential to emphasize that a thorough assessment of model performance extends beyond visual analysis. A careful examination of the actual concentration values, beyond standardized visual representations, is imperative to draw definitive conclusions regarding model effectiveness.

To this end, Table 5.2 presents the evaluation results for Painting 4 using the Structural Similarity measure. Among the models subjected to evaluation, 1DCNN+ emerged as the top performer, achieving an impressive average SSIM score of 0.982. This high score signifies its exceptional capability to accurately capture the similarity

	FCN	FCN+	1DCNN	1DCNN+
Total	0.982	**0.998**	0.991	**0.998**
Avg	0.875	0.909	0.872	**0.982**
S K	0.872	0.398	0.963	**0.981**
K K	0.917	0.938	0.946	**0.968**
Ca K	0.989	0.991	0.991	**0.997**
Cr K	0.819	0.985	0.967	**0.995**
Mn K	0.989	0.989	0.883	**0.997**
Fe K	0.995	0.997	0.964	**0.999**
Cu K	0.89	0.979	0.966	**0.992**
Zn K	0.766	0.914	0.548	**0.971**
Sr K	0.901	0.968	0.948	**0.982**
Au L	0.743	0.896	0.682	**0.965**
Hg L	0.853	0.906	0.866	**0.975**
Pb L	0.761	0.951	0.740	**0.965**

Table 5.2: Structural Similarity evaluation of models for Painting 4.

between the predicted Elemental Distribution Maps and the Ground Truth. Conversely, FCN exhibited less favorable performance, with an average SSIM score of 0.875, indicating comparatively lower performance in terms of similarity. Notably, 1DCNN outperformed FCN, underscoring the superior effectiveness of convolutional layers in generating more precise representations. A comparison between models with and without prior knowledge revealed a conspicuous advantage in incorporating prior knowledge. For instance, FCN+ surpassed FCN, and 1DCNN+ outperformed 1DCNN in nearly all cases, underscoring the significant impact of prior knowledge in augmenting model performance.

Table 5.3 and Table 5.4 present the evaluation results for Painting 4 using the Pearson Correlation and Slope measures, respectively. These measures gauge the performance of different models in capturing elemental distribution maps as compared to the Ground Truth. In terms of Pearson Correlation, 1DCNN+ achieved the highest average score of 0.996, indicative of a strong linear correlation capture, while FCN exhibited a lower score of 0.924. The incorporation of prior knowledge consistently resulted in enhanced model performance, with FCN+ outperforming FCN and 1DCNN+ surpassing 1DCNN across various elements. Regarding Slope measure,

	FCN	**FCN+**	**1DCNN**	**1DCNN+**
Avg	0.924	0.984	0.936	**0.996**
S K	0.950	0.898	0.984	**0.999**
K K	0.945	0.973	0.986	**0.989**
Ca K	**0.999**	**0.999**	**0.999**	**0.999**
Cr K	0.718	0.986	0.956	**0.994**
Mn K	0.997	0.998	0.971	**0.999**
Fe K	**0.999**	**0.999**	**0.999**	**0.999**
Cu K	**0.999**	**0.999**	**0.999**	**0.999**
Zn K	0.497	0.961	0.376	**0.983**
Sr K	0.987	0.995	0.990	**0.996**
Au L	0.993	**0.998**	0.982	**0.998**
Hg L	**0.999**	**0.999**	**0.999**	**0.999**
Pb L	**0.999**	**0.999**	0.996	**0.999**

Table 5.3: Pearson Correlation evaluation of models for Painting 4.

1DCNN+ also outperformed other models with an average score of 1.009, demonstrating superior slope capture. FCN yielded a slightly higher score of 1.11. Once again, the inclusion of prior knowledge proved advantageous, with FCN+ outperforming FCN and 1DCNN+ surpassing 1DCNN in multiple elements.

Model	**0-1**	**1-2**	**2-3**	**> 3**
FCN	0.606	0.107	0.065	0.222
FCN+	0.901	0.086	0.010	0.001
1DCNN	0.802	0.129	0.044	0.023
1DCNN+	0.925	0.060	0.011	0.002

Table 5.5: Absolute z-score values distribution analysis for Painting 4.

	FCN	FCN+	1DCNN	1DCNN+
Avg	1.11	0.99	1.144	**1.009**
S K	0.950	**0.961**	1.068	0.927
K K	0.945	**1.029**	1.183	1.109
Ca K	**0.999**	0.992	1.008	1.014
Cr K	0.718	0.970	**0.989**	0.965
Mn K	**0.997**	1.008	0.948	1.020
Fe K	0.999	**1.000**	1.001	1.009
Cu K	**0.999**	0.991	1.005	1.011
Zn K	0.497	**0.980**	2.261	1.023
Sr K	0.987	0.989	**1.007**	**0.993**
Au L	0.993	0.974	0.908	**1.000**
Hg L	**0.999**	0.991	1.042	1.02
Pb L	**0.999**	0.991	1.307	1.017

Table 5.4: Slope evaluation of models for Painting 4.

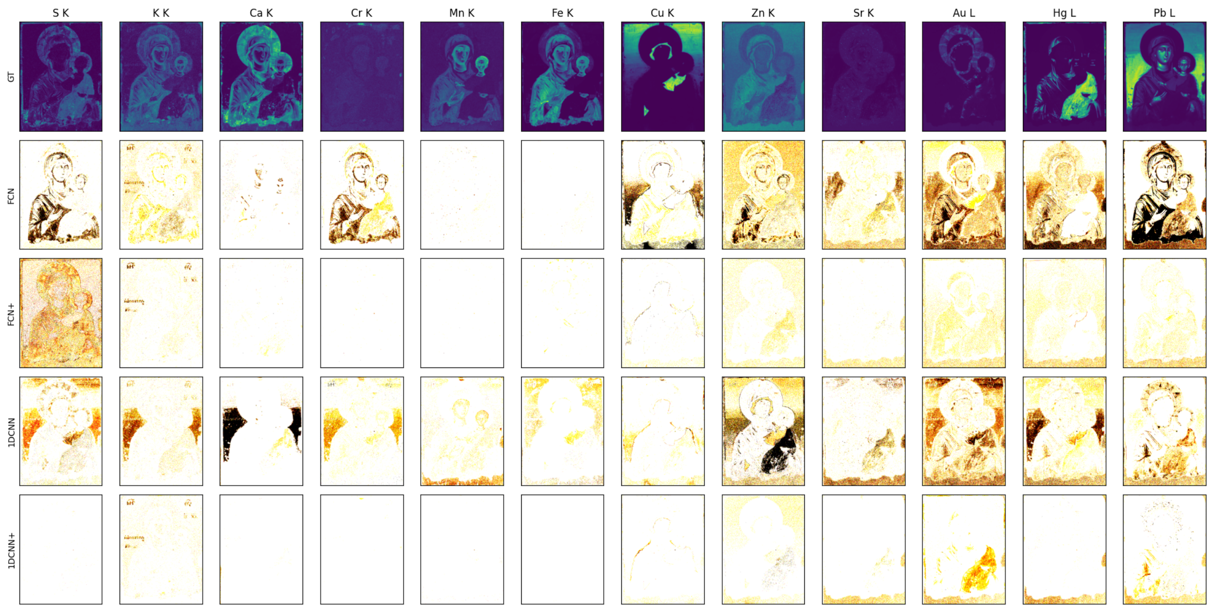


Figure 5.4: Visualization of absolute z-score values spatial distribution for Painting 4.

A comprehensive analysis of the absolute z-score values, as illustrated in Figure 5.4, revealed the superior performance of both the FCN+ and 1DCNN+ models. Figure's pixel colors transitioned from white to yellow, orange, red, and black, signifying increasing deviations from the actual values within the respective intervals: $(0, 1], (1, 2], (2, 3]$, and $(3, +\infty)$. The predictions of the 1DCNN+ model demonstrated strong agreement with the Ground Truth concentration values, evident in the error areas displaying minimal deviations, nearly appearing white.

For a more precise assessment of the performance of FCN+ and 1DCNN+ models, which had been visually assessed through absolute z-score values, Table 5 offers a concise and structured representation. This table quantified the percentage of predicted values falling within specific intervals, namely $(0, 1], (1, 2], (2, 3]$, and $(3, +\infty)$, providing a logical and easily interpretable format for evaluating model efficiency.

5.3.2 Painting 5 Evaluation

In parallel with the evaluation of Painting 4, the assessment of the model's performance on Painting 5 followed a similar protocol, utilizing all images for training within the trainset and reserving Painting 5 exclusively for the testset. Notably, in the evaluation of the models' performance on Painting 5, a consistent trend emerged, with the 1DCNN+ model consistently outperforming all other models in every aspect. The 1DCNN+ has achieved the highest SSIM scores for all target elements. Its remarkable capacity to accurately replicate the visual complexities, as demonstrated in Painting 5 (Table 5.6), is reflected in its average SSIM score of 0.994.

	FCN	FCN+	1DCNN	1DCNN+
Total	0.986	0.969	0.993	**0.999**
Avg	0.933	0.867	0.963	**0.994**
S K	0.989	0.404	0.977	**0.997**
K K	0.972	0.859	0.964	**0.976**
Ca K	0.996	0.989	0.992	**0.997**
Cr K	0.987	0.941	0.959	**0.993**
Mn K	0.978	0.914	0.978	**0.998**
Fe K	0.954	0.976	0.998	**0.999**
Cu K	0.975	0.940	0.975	**0.994**
Zn K	0.933	0.883	0.946	**0.997**
Sr K	0.818	0.842	0.977	**0.994**
Au L	0.812	0.853	0.902	**0.991**
Hg L	0.836	0.847	0.909	**0.993**
Pb L	0.946	0.954	0.982	**0.995**

Table 5.6: Structural Similarity evaluation of models for Painting 5.

However, it is noteworthy that FCN+ did not perform as effectively for Painting 5 compared to its performance for Painting 4. In this assessment, both the FCN and 1DCNN models appear to outshine FCN+ by achieving higher structural similarity with the ground truth concentration values. This deviation from the performance observed for Painting 4 suggests that Painting 5 contains subtler details or complexities that impose a more significant challenge to the models (Table 5.6).

Despite this, it is important to highlight that FCN+ still achieved remarkable SSIM scores, coming remarkably close to the performance of models without prior physics knowledge incorporation. This outcome is particularly noteworthy considering that FCN+ employs only 12 features derived from Prior Layer incorporation. This result underscores the model's capacity to capture the painting's visual characteristics effectively despite its relatively limited feature set, highlighting the potential of the proposed approach even in challenging scenarios.

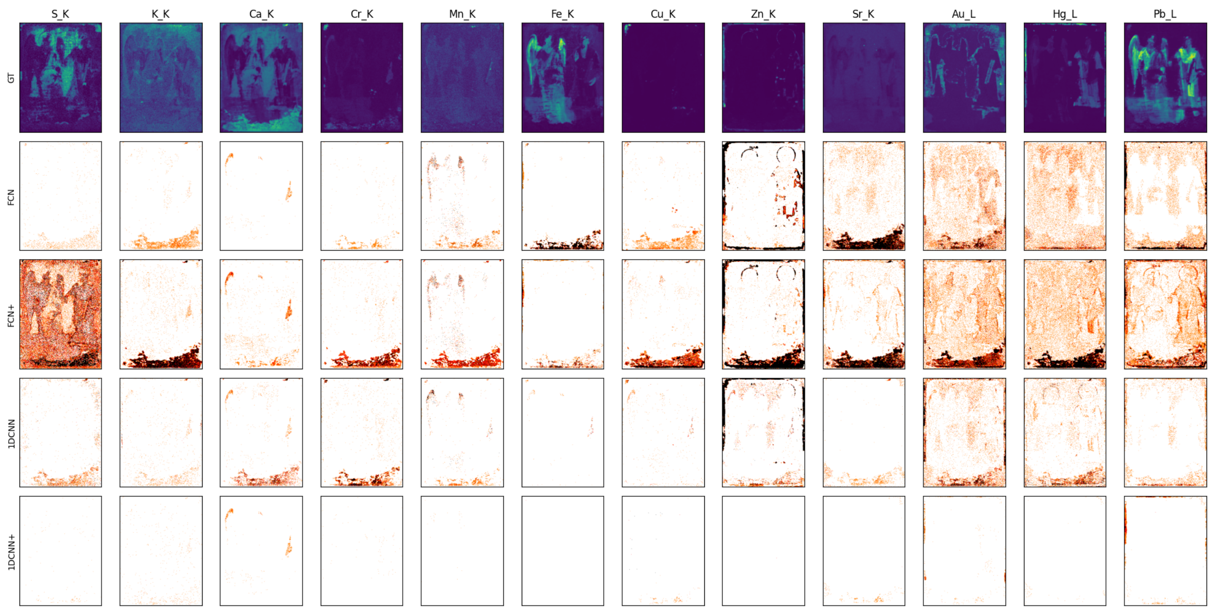


Figure 5.5: Visualization of absolute z-score values spatial distribution for Painting 5.

In complement to the Structural Similarity measure, the Pearson Correlation (Table 5.7) was also assessed as a complementary evaluation measure. Notably, the 1DCNN+ model exhibited exceptional performance with an average correlation coefficient of 0.998. This signifies a robust alignment between its predictions and the actual values, underscoring its capability to capture the intricate patterns within the painting's data. Across various elemental regions, such as Calcium K (Ca K), Iron K (Fe K), and Copper K (Cu K), 1DCNN+ consistently outperformed other models, demonstrating its effectiveness in discerning subtle nuances within the spectra. This finding echoes what was observed with Painting 5, but it's worth highlighting the model's versatility and reliability across various evaluation measures. This consistency underlines its strength in handling intricate artistic data, suggesting it could shine in a wide range of scenarios.

Table 5.8 offers an analysis of the distribution of absolute z-score values, further assessing the models' effectiveness. The 1DCNN+ model excels in this evaluation, exhibiting a distribution that signifies its ability to closely approximate the actual values for Painting 5. These results reinforce the notion that 1DCNN+ stands out as a powerful model for capturing the intricacies of the painting, both in terms of Pearson Correlation and z-score distribution, highlighting its potential for applications in MA-XRF analysis.

	FCN	FCN+	1DCNN	1DCNN+
Avg	0.976	0.923	0.988	**0.998**
S K	**0.999**	0.902	0.998	**0.999**
K K	0.977	0.828	0.977	**0.986**
Ca K	**0.999**	**0.999**	**0.999**	**0.999**
Cr K	0.995	0.957	0.971	**0.998**
Mn K	0.968	0.906	0.973	**0.998**
Fe K	**0.999**	**0.999**	**0.999**	**0.999**
Cu K	**0.999**	0.998	**0.999**	**0.999**
Zn K	0.961	0.685	0.951	**0.999**
Sr K	0.820	0.812	0.991	**0.998**
Au L	0.995	0.991	0.994	**0.999**
Hg L	**0.999**	0.998	**0.999**	**0.999**
Pb L	**0.999**	**0.999**	**0.999**	**0.999**

Table 5.7: Pearson Correlation evaluation of models for Painting 5.

Model	0-1	1-2	2-3	>3
FCN	0.838	0.111	0.016	0.035
FCN+	0.703	0.215	0.046	0.036
1DCNN	0.941	0.049	0.006	0.002
1DCNN+	0.978	0.015	0.004	0.001

Table 5.8: Absolute z-score values distribution analysis for Painting 5.

Figures 5.6 and 5.7 are presented indicatively to emerge the generalization capabilities offering the convolutional operations, displaying the training and validation loss curves over epochs for Painting 5. The consistently lower training loss in the 1DCNN+ model can be attributed to its specialized architecture for one-dimensional data, enabling it to capture complex patterns more effectively. The incorporation of prior physics-based knowledge further enhances its representations, facilitating quicker convergence and lower training loss compared to other models.

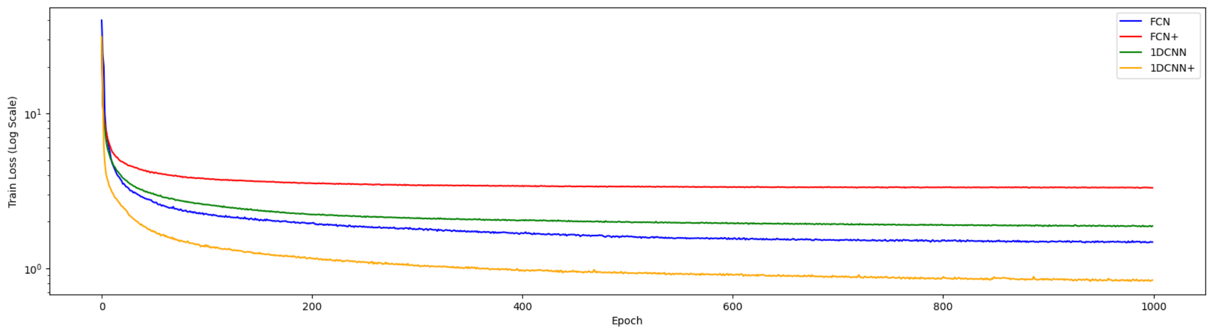


Figure 5.6: Training Loss of models for Painting 5.

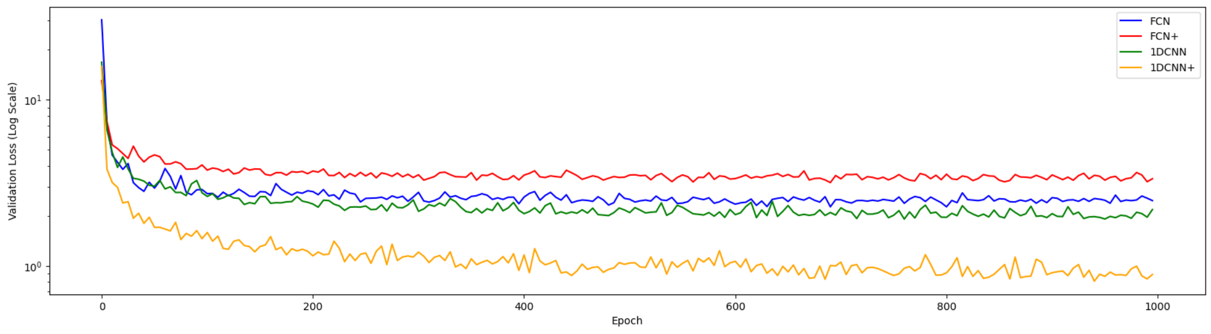


Figure 5.7: Validation Loss of models for Painting 5.

However, it's important to note that in the case of FCN and 1DCNN, the difference in training and validation loss trends can be attributed to their distinct generalization capabilities. FCN, despite achieving a lower training loss than 1DCNN, exhibits a propensity to overfit the training data due to its architecture. This leads to a higher validation loss as it struggles to generalize effectively to unseen data. In contrast, 1DCNN, although having a slightly higher training loss, showcases better generalization capabilities. Its architecture enables it to capture pertinent features without overfitting, resulting in a lower validation loss. This characteristic underscores the robustness of 1DCNN's learned representations, which is pivotal for achieving improved predictive performance on unseen or real-world data.

Chapter 6

Epilogue

6.1 Conclusion

In conclusion, the research conducted in this thesis has successfully addressed the problem of Elemental Maps Prediction in the context of Macro-XRF analysis. Two prominent deep architectures, the Fully Connected Neural Network and the Convolutional Neural Network, were utilized as multi-output regressors. Notably, this study introduced a novel approach by incorporating Physics-Based prior knowledge into these deep learning models through the inclusion of a predefined layer utilizing the spectral properties of the elements under investigation. This incorporation of prior knowledge increased the accuracy of the models considerably. The experimental outcomes underscore the potential of integrating deep learning techniques with domain-specific physics-based insights to advance the field of MA-XRF elemental mapping.

6.2 Future work

As the field of Elemental Map Prediction continues to evolve, there are several promising avenues for future research and development. In this section, some potential directions for future work are outlined that can further advance the understanding and application of elemental mapping:

- **Enriched dataset**: Expanding the dataset by incorporating more paintings and increasing the number of samples per artwork is essential. A larger and more diverse dataset would enable the models to learn a broader range of artistic styles, variations, and complexities, potentially leading to even more accurate and robust results.

- **Pre-training Models**: Pre-training models on related tasks or using transfer learning from other domains can be explored. Pre-training can help initialize the models with valuable knowledge and representations, making it easier for them to learn the elemental mapping task effectively. This approach has the potential to enhance the models' performance and speed up convergence.

- **2D Convolutional Networks**: Investigating the use of 2D convolutional networks is a promising avenue. The incorporation of a prior layer could efficiently reduce the number of channels, making the computational resource requirements less demanding. Additionally, 2D convolutional networks can capture spatial dependencies within the paintings, potentially improving the models ability to discern subtle patterns and details.

- **Efficient Sampling for Pixel-Based Training**: When training models with pixel data as independent features, there is a potential for improving efficiency through advanced pixel sampling techniques. Future research could explore methods such as edge detection algorithms to identify and prioritize informative pixels, as well as selecting pixels whose features deviate significantly from the mean.

- **Enhancing the Prior Layer with more pure spectra**: Future research could explore the incorporation of a broader set of pure spectra into the prior layer of models. Instead of relying solely on target spectra, enriching the prior layer with a diverse range of pure spectra from the periodic table is a promising research.

  Multiple Regressors: An alternative avenue for future investigation lies in the decomposition of the problem into training individual single-output regressors as opposed to a unified multi-output regressor.

- **Hyperparameters and Architectures**: Conducting a thorough exploration of hyperparameters and model architectures is crucial. Further fine-tuning of hyperparameters and experimentation with different model architectures could lead to further improvements in performance.

Bibliography

[1] M. Horf, R. Gebbers, S. Vogel, M. Ostermann, M.-F. Piepel, and H.-W. Olfs, "Determination of nutrients in liquid manures and biogas digestates by portable energy-dispersive x-ray fluorescence spectrometry," *Sensors*, vol. 21, no. 11, p. 3892, 2021.

[2] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[3] S. Kogou, L. Lee, G. Shahtahmassebi, and H. Liang, "A new approach to the interpretation of xrf spectral imaging data using neural networks," *X-Ray Spectrometry*, vol. 50, no. 4, pp. 310–319, 2021.

[4] J. S. Iwanczyk, E. Nygard, O. Meirav, J. Arenson, W. C. Barber, N. E. Hartsough, N. Malakhov, and J. C. Wessel, "Photon counting energy dispersive detector arrays for x-ray imaging," *IEEE transactions on nuclear science*, vol. 56, no. 3, pp. 535–542, 2009.

[5] P. Lechner, S. Eckbauer, R. Hartmann, S. Krisch, D. Hauff, R. Richter, H. Soltau, L. Strüder, C. Fiorini, E. Gatti *et al.*, "Silicon drift detectors for high resolution room temperature x-ray spectroscopy," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 377, no. 2-3, pp. 346–351, 1996.

[6] P. Brouwer, "Theory of xrf," *Almelo, Netherlands: PANalytical BV*, 2006.

[7] A. Gürol, "Measurements of the k x-ray intensity ratios by using energy-dispersive x-ray fluorescence spectrometry," *Applied Radiation and Isotopes*, vol. 66, no. 3, pp. 372–376, 2008.

[8] T. Schoonjans. (2017) Xmi-msim: Monte carlo simulation of energy-dispersive x-ray fluorescence spectrometers. [Online]. Available: https://github.com/tschoonj/xmimsimy

[9] N. Rohani, E. Pouyet, M. Walton, O. Cossairt, and A. K. Katsaggelos, "Nonlinear unmixing of hyperspectral datasets for the study of painted works of art," *Angewandte Chemie*, vol. 130, no. 34, pp. 11 076–11 080, 2018.

[10] H. Deborah, M. O. Ulfarsson, and J. Sigurdsson, "Fully constrained least squares linear spectral unmixing of the scream (verso, 1893)," in *2021 11th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS)*. IEEE, 2021, pp. 1–5.

[11] X. Cao, S. Jiang, J. R. Gunn, P. Bruza, and B. W. Pogue, "Single pixel hyperspectral cherenkov-excited fluorescence imaging with linac x-ray sheet scanning and spectral unmixing," *Optics Letters*, vol. 45, no. 22, pp. 6130–6133, 2020.

[12] R. Rajabi and H. Ghassemian, "Spectral unmixing of hyperspectral imagery using multilayer nmf," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 1, pp. 38–42, 2014.

[13] L. Li, H. Yan, W. Xu, D. Yu, A. Heroux, W.-K. Lee, S. I. Campbell, and Y. S. Chu, "Pyxrf: Python-based x-ray fluorescence analysis package," in *X-Ray Nanoimaging: Instruments and Methods III*, vol. 10389. SPIE, 2017, pp. 38–45.

[14] M. Vermeulen, A. McGeachy, B. Xu, H. Chopp, A. Katsaggelos, R. Meyers, M. Alfeld, and M. Walton, "Xrfast a new software package for processing of ma-xrf datasets using machine learning," *Journal of Analytical Atomic Spectrometry*, vol. 37, no. 10, pp. 2130–2143, 2022.

[15] A. Migliori, P. Bonanni, L. Carraresi, N. Grassi, and P. Mando, "A novel portable xrf spectrometer with range of detection extended to low-z elements," *X-Ray Spectrometry*, vol. 40, no. 2, pp. 107–112, 2011.

[16] A. Voulodimos, N. Doulamis, A. Doulamis, E. Protopapadakis *et al.*, "Deep learning for computer vision: A brief review," *Computational intelligence and neuroscience*, vol. 2018, 2018.

[17] J. Chai, H. Zeng, A. Li, and E. W. Ngai, "Deep learning in computer vision: A critical review of emerging techniques and application scenarios," *Machine Learning with Applications*, vol. 6, p. 100134, 2021.

[18] N. O'Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, L. Krpalkova, D. Riordan, and J. Walsh, "Deep learning vs. traditional computer vision," in *Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC), Volume 1 1*. Springer, 2020, pp. 128–144.

[19] U. Kamath, J. Liu, and J. Whitaker, *Deep learning for NLP and speech recognition*. Springer, 2019, vol. 84.

[20] M. M. Lopez and J. Kalita, "Deep learning applied to nlp," *arXiv preprint arXiv:1703.03091*, 2017.

[21] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[22] M. Hussain, J. J. Bird, and D. R. Faria, "A study on cnn transfer learning for image classification," in *Advances in Computational Intelligence Systems: Contributions Presented at the 18th UK Workshop on Computational Intelligence, September 5-7, 2018, Nottingham, UK*. Springer, 2019, pp. 191–202.

[23] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[24] X. J. Zhu, "Semi-supervised learning literature survey," 2005.

[25] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[26] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[27] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[28] Y. LeCun, Y. Bengio *et al.*, “Convolutional networks for images, speech, and time series,” *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[29] A. Mazzinghi, C. Ruberto, L. Castelli, C. Czelusniak, L. Giuntini, P. A. Mandò, and F. Taccetti, “Ma-xrf for the characterisation of the painting materials and technique of the entombment of christ by rogier van der weyden,” *Applied Sciences*, vol. 11, no. 13, p. 6151, 2021.

[30] G. Van der Snickt, S. Legrand, J. Caen, F. Vanmeert, M. Alfeld, and K. Janssens, “Chemical imaging of stained-glass windows by means of macro x-ray fluorescence (ma-xrf) scanning,” *Microchemical Journal*, vol. 124, pp. 615–622, 2016.

[31] P. Ricciardi, S. Legrand, G. Bertolotti, and K. Janssens, “Macro x-ray fluorescence (ma-xrf) scanning of illuminated manuscript fragments: potentialities and challenges,” *Microchemical Journal*, vol. 124, pp. 785–791, 2016.

[32] R. Castaing, “Electron probe microanalysis,” in *Advances in electronics and electron physics*. Elsevier, 1960, vol. 13, pp. 317–386.

[33] D. E. Newbury* and N. W. Ritchie, “Is scanning electron microscopy/energy dispersive x-ray spectrometry (sem/eds) quantitative?” *Scanning*, vol. 35, no. 3, pp. 141–168, 2013.

[34] B. J. Xu, Y. Wu, P. Hao, M. Vermeulen, A. McGeachy, K. Smith, K. Eremin, G. Rayner, G. Verri, F. Willomitzer *et al.*, “Can deep learning assist automatic identification of layered pigments from xrf data?” *Journal of Analytical Atomic Spectrometry*, vol. 37, no. 12, pp. 2672–2682, 2022.

[35] A. Hore and D. Ziou, “Image quality metrics: Psnr vs. ssim,” in *2010 20th international conference on pattern recognition*. IEEE, 2010, pp. 2366–2369.

[36] E. Kreyszig, K. Stroud, and G. Stephenson, “Advanced engineering mathematics,” *Integration*, vol. 9, no. 4, 2008.

[37] “Ceramics & composites laboratory (ccl),” http://www.materials.uoi.gr/ccl/.

[38] G. P. Mastrotheodoros, A. Asvestas, T. Gerodimos, and D. F. Anagnostopoulos, “Revealing the materials, painting techniques, and state of preservation of a

heavily altered early 19th century greek icon through ma-xrf," *Heritage*, vol. 6, no. 2, pp. 1903–1920, 2023.

[39] T. Gerodimos, A. Asvestas, G. P. Mastrotheodoros, G. Chantas, I. Liougos, A. Likas, and D. F. Anagnostopoulos, "Scanning x-ray fluorescence data analysis for the identification of byzantine icons' materials, techniques, and state of preservation: A case study," *Journal of Imaging*, vol. 8, no. 5, p. 147, 2022.

[40] S. Sotiropoulou and S. Daniilia, "Material aspects of icons. a review on physicochemical studies of greek icons," *Accounts of chemical research*, vol. 43, no. 6, pp. 877–887, 2010.

[41] G. P. Mastrotheodoros, K. Beltsios, Y. Bassiakos, and V. Papadopoulou, "On the metal-leaf decorations of post-byzantine greek icons," *Archaeometry*, vol. 60, no. 2, pp. 269–289, 2018.

[42] "Bruker m6 jetstream product page," https://www.bruker.com/en/products-and-solutions/elemental-analyzers/micro-xrf-spectrometers/m6-jetstream.html, accessed: September 15, 2023.

[43] V. A. Solé, E. Papillon, M. Cotte, P. Walter, and J. Susini, "A multiplatform code for the analysis of energy-dispersive x-ray fluorescence spectra," *Spectrochimica Acta Part B: Atomic Spectroscopy*, vol. 62, no. 1, pp. 63–68, 2007.

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

[45] T.-T. Wong, "Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation," *Pattern recognition*, vol. 48, no. 9, pp. 2839–2846, 2015.

[46] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019.

Short Biography

Ioannis Georvasilis, born in Ioannina, Greece in 1996, completed his undergraduate program in Computer Science and Engineering at the University of Ioannina in 2021. His diploma thesis, titled ”Image Text Removal with Generative Adversarial Networks,” represented a notable initiation in his academic path. In 2021, he pursued post-graduate studies in the same department, where his research focused on applying Deep Learning techniques to analyze complex data from Macro-XRF spectra. His contributions led to participation in Spectroscopy conferences and publications in scientific journals. Moreover, he has a keen and dedicated interest in Machine Learning, with a particular emphasis on Generative Learning.