UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Vanessa Brischi Olivatto Fioravanti

Autoencoder-based Dictionary Learning and Sparse Coding

Aprendizagem de Dicionários e Codificação Esparsa Baseada em Autoencoders

Campinas
2023

UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Vanessa Brischi Olivatto Fioravanti

Autoencoder-based Dictionary Learning and Sparse Coding
Aprendizagem de Dicionários e Codificação Esparsa Baseada em Autoencoders

Dissertation presented to the School of Electrical and Computer Engineering of the University of Campinas in partial fulfillment of the requirements for the degree of Doctor in Electrical Engineering, in the area of Telecommunications and Telematics.

Tese apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Doutora em Engenharia Elétrica, na Área de Telecomunicações e Telemática.

Orientador: Prof. Dr. Romis Ribeiro de Faissol Attux

Este trabalho corresponde à versão final da tese defendida pela aluna Vanessa Brischi Olivatto Fioravanti, e orientada pelo Prof. Dr. Romis Ribeiro de Faissol Attux

---

Campinas

2023

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

F511a

Fioravanti, Vanessa Brischi Olivatto, 1991-
Autoencoder-based dictionary learning and sparse coding / Vanessa Brischi Olivatto Fioravanti. – Campinas, SP : [s.n.], 2023.

Orientador: Romis Ribeiro de Faissol Attux.
Tese (doutorado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Esparsidade. 2. Processamento digital de sinais. 3. Aprendizado de máquina. 4. Interface cérebro-computador. I. Attux, Romis Ribeiro de Faissol, 1978-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações Complementares

Título em outro idioma: Aprendizagem de dicionários e codificação esparsa baseada em autoencoders
Palavras-chave em inglês:
Sparsity
Digital signal processing
Machine learning
Brain-computer inteface
Área de concentração: Telecomunicações e Telemática
Titulação: Doutora em Engenharia Elétrica
Banca examinadora:
Romis Ribeiro de Faissol Attux [Orientador]
Renato da Rocha Lopes
Levy Boccato
André Kazuo Takahata
Ricardo Suyama
Data de defesa: 23-05-2023
Programa de Pós-Graduação: Engenharia Elétrica

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-2331-3320
- Currículo Lattes do autor: http://lattes.cnpq.br/6218409312659286

COMISSÃO JULGADORA – TESE DE DOUTORADO

Candidato: Vanessa Brischi Olivatto Fioravanti RA: 096002
Data da Defesa: 23 de maio de 2023
Título da Dissertação: Autoencoder-based Dictionary Learning and Sparse Coding

Prof. Dr. Romis Ribeiro de Faissol Attux
Prof. Dr. Renato da Rocha Lopes
Prof. Dr. Levy Boccato
Prof. Dr. André Kazuo Takahata
Prof. Dr. Ricardo Suyama

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no SIGA (Sistema de Fluxo de Dissertação/Tese) e na Secretaria de Pós Graduação da Faculdade de Engenharia Elétrica e de Computação.

"Be water, my friend."

Bruce Lee

Abstract

This dissertation investigates the dictionary learning problem in sparse signal representation and their applications across real-world scenarios. The study begins by presenting state-of-the-art algorithms for sparse dictionary learning, which aim to minimise representation error and ensure sparsity through different variations of alternating minimisation methods. Expanding on these methods, a proposed sparse dictionary learning framework is introduced, employing an autoencoder design. This approach utilises the Kullback-Leibler divergence as the sparsity penalty term, replacing the traditional adoption of ℓ_1 and ℓ_0 norms in previous methods. The sparse representations are obtained from both fully-connected and convolutional encoder architectures.

The autoencoder-based framework is able to replace the classical approaches based on alternating minimisation. Rather than keeping an estimate of the dictionary fixed while estimating the sparse coefficients and subsequently updating the dictionary based on these coefficients, the autoencoder design effectively learns the dictionary and the sparse coefficients simultaneously. This is achieved by training the autoencoder in an end-to-end manner, where the input data is encoded into sparse codes, and then reconstructed to the original data source using the decoder part of the model, representing the learned dictionary. This leads to a more efficient, flexible, and streamlined process.

The proposed framework is explored in signal modelling and image compression, demonstrating its competitiveness and advantages over other signal transforms with fixed, predefined bases such as Wavelets and Fourier. The final segment of this study introduces a discriminative dictionary learning setting that combines a classification neural network model and the autoencoder-based framework. The joint optimisation of the sparse representation and the classifier allows for simultaneous learning and refinement of both components. This enables the models to mutually benefit from each other's training process, leading to improved performance and more effective representation of the data. The proposed discriminative framework is applied in the context of SSVEP signals from BCI systems.

Keywords: Sparsity; Digital Signal Processing; Machine Learning; Brain-Computer Interface.

Resumo

Esta tese investiga o problema de aprendizagem de dicionários para representação esparsa de sinais e suas aplicações em cenários do mundo real. O estudo começa apresentando algoritmos do estado-da-arte para aprendizado de dicionário esparsos, que visam minimizar o erro de representação e garantir a esparsidade por meio de variações de métodos de minimização alternada. Expandindo os métodos apresentados, introduz-se uma nova estrutura de formulação e resolução do problema, empregando um modelo de autoencoder. Essa abordagem utiliza a divergência de Kullback-Leibler como termo de penalização de esparsidade, substituindo as tradicionais normas ℓ_1 e ℓ_0 adotadas em métodos anteriores. As representações esparsas são obtidas a partir de arquiteturas de redes totalmente conectadas e convolucionais.

A estrutura baseada em autoencoder substitui a abordagem clássica baseada na minimização alternada. Em vez de manter uma estimativa do dicionário fixa enquanto estima os coeficientes esparsos e posteriormente atualiza o dicionário com base nesses coeficientes, o modelo de autoencoder aprende efetivamente o dicionário e os coeficientes esparsos de forma simultânea. Isso é obtido a partir do treinamento do modelo de autoencoder, em que os dados de entrada são representados a partir de códigos esparsos via encoder e, em seguida, reconstruídos a partir do decodificador do modelo, que representa o dicionário aprendido. Isso leva a um processo mais eficiente, flexível e otimizado.

A estrutura proposta é explorada na modelagem de sinais e compressão de imagens, demonstrando sua competitividade e vantagens em relação a outras transformações de sinal a partir de bases fixas e predefinidas, como Wavelets e Fourier. O último segmento deste estudo introduz um ambiente de aprendizado de dicionário discriminativo que combina modelos de redes neurais de classificação na estrutura baseada em autoencoder. A otimização conjunta da representação esparsa e do classificador permite o aprendizado e aprimoramento simultâneos de ambos os componentes. Isso permite que os modelos se beneficiem mutuamente do processo de treinamento, resultando em melhor desempenho e representação mais efetiva dos dados. A estrutura discriminativa proposta é aplicada no contexto de sinais SSVEP de sistemas BCI.

Keywords: Esparsidade; Processamento Digital de Sinais; Aprendizado de Máquina; Interface Cérebro-Computador.

List of Figures

List of Tables

Acronyms

AE Autoencoder

BCI Brain Computer Interface

COPAR Commonality and Particularity

CV Cross-validation

DCT Discrete Cosine Transform

DFT Discrete Fourier Transform

DL Dictionary Learning

DLSI Support Vector Machines

DWT Discrete Wavelet Transform

EEG Electroencephalogram

FDDL Fisher Discriminative Dictionary Learning

FFT Fast Fourier Transform

HCGSN HydroCel Geodesic Sensor Net

KL Kullback-Leibler

KNN K-Nearest Neighbor

LOSO Leave-One-Subject-Out

LRSDL Low-rank Shared Dictionary Learning

MAMEM Multimodal Authoring using your Eyes and Mind

ML Machine Learning

MLP Multi Layer Perceptron

MOD Method of Optimal Directions

MSE Mean-Squared Error

NN Neural Network

NNDDL Neural Network Dictionary Learning

OMP Orthogonal Matching Pursuit

SRC Sparse Representation Classifier

SSIM Structural Similarity Index

SSVEP Steady-State-Visual Evoked Potentials

SVD Singular Value Decomposition

SVM Support Vector Machines

Contents

Chapter 1

Introduction

Typical methods to represent real-valued signals are often built from a linear superposition of basis functions, which can efficiently encode high-dimensional data spaces. Common bases such as Fourier harmonically-related complex exponentials and wavelet bases can provide a suitable representation of some signals. However, these alternatives are limited, as they are not data-driven bases.

A more general method for signal representation is to use overcomplete bases or dictionaries, which are composed of a set of dictionary atoms that are not necessarily orthogonal. Ideally, this basis should be adapted to the data so that each redundant basis function captures as many features as possible, specifically for the signal class of interest. Dictionary atoms, which are essentially individual basis functions, can capture different aspects of the signal and offer a flexible representation. The set of redundant atoms can allow for several different representations of the same signal and a certain flexibility in the sparse representation. This approach can be used to develop more efficient signal representations for various applications.

Sparse dictionary learning is a computationally challenging optimisation problem that falls into the NP-Hard class. The standard method of solving this problem involves a two-step iterative procedure, which typically utilises con-

vex relaxation or a greedy-based approach. However, these methods are known to have limitations and may not always provide satisfactory results. In this study, we propose a novel approach of sparse dictionary learning based on sparse autoencoder models. Our approach overcomes the limitations of traditional techniques and provides a more flexible and efficient way of learning dictionaries. By utilising backpropagation to simultaneously update both the dictionary and the sparse code, our method can capture the essential features of the input data and optimise them for specific tasks, such as signal classification. Our proposed approach has shown promising results in experimental evaluations, demonstrating the effectiveness of this method in learning high-quality dictionaries for sparse representation.

An autoencoder is a neural network that typically includes an encoder function that maps the input data to a low dimensional latent space, and a decoder function that reconstructs the original data from the encoded representation. Although autoencoders can reduce the dimensionality of the input data in the encoder latent space, their primary objective is to minimise the loss function during training while preserving as much information as possible.

Autoencoders with specific cost functions can provide an alternative framework for dictionary learning and sparse coding problems. Unlike traditional methods that update the dictionary and its sparse code iteratively, autoencoders utilise the backpropagation algorithm during training to simultaneously update both the dictionary and its sparse code. The dictionary atoms can be obtained by extracting the weights of the single-layer decoder side of the trained autoencoder. The number of neurons in the latent space determines the number of atoms in the dictionary. This approach can improve classification performance and avoid issues with local minima, making it a promising solution for handling large datasets with significant signal spaces.

In this work, we propose a novel method for discriminative dictionary

learning based on sparse autoencoders. The method is initially analysed in terms of sparse representation error and also from the standpoint of image classification using the standard dataset CIFAR-10 (KRIZHEVSKY *et al.*, 2009)). A second contribution is the application of the proposed method to signal processing in brain-computer interfaces based on steady state visually evoked potentials (SSVEPs).

This dissertation is organised as follows:

- Chapter 2: In this chapter, we provide an overview of conventional dictionary learning and sparse coding methods, including the popular K-SVD and OMP algorithms. We discuss the fundamental principles of dictionary learning, which involves learning a dictionary that provides a sparse representation of the input data, and the subsequent use of sparse coding techniques to encode the input data into a compressed representation using the learned dictionary. We also explore the limitations of these conventional approaches and highlight the need for more innovative and flexible techniques, which we will address in subsequent chapters.

- Chapter 3: This chapter introduces k-sparse autoencoders and their properties, which can be designed to solve various problems by encoding data into a compressed representation using a learned dictionary. The sparsity constraint in the autoencoder objective function encourages the learned dictionary to be sparse, leading to a compact representation of the input data. The proposed methodology builds upon fundamental principles in the field of dictionary learning and sparse coding. The chapter proposes a novel approach that offers a flexible selection of non-linear activation functions on the encoder side, with no constraints tying up the encoder and decoder weights. The methodology adopts a threefold cost function, whose terms: (i) encourage sparsity in the latent space, (ii) penalise representation error and (iii) constrain the norms of the dictionary atoms. The proposed

methodology offers an alternative solution that can accommodate a wide range of purposes. In particular, it is suitable for large problem sizes where conventional algorithms may not be feasible.

- Chapter 4: In this chapter, we present our studies in the field of image compression using analytic and data-driven dictionaries. We provide an overview of the literature on image compression, highlighting the use of analytic transforms in image codecs. We then explore the use of data-driven dictionary learning for image compression and discuss various approaches in the field. Our study focuses on comparing the visual quality of compressed images obtained using the proposed dictionary learning framework with that of compressed images obtained using the Fourier and Wavelet transforms. We evaluate the performance of the methods in terms of the peak signal-to-noise ratio (PSNR) and the visual quality of the reconstructed images at different compression rates. We provide a comprehensive study of data-driven dictionary learning for image compression, which clarifies the potential of the proposed method as competitive alternative to analytic transforms for image codecs.

- Chapter 5: In this chapter, we proposed a Neural Network-based Discriminative Dictionary Learning and Sparse Coding (NNDDL) method that represents class-specific and shared features of a signal, and distinguishes their sparse representations accordingly. This method uses a sparse autoencoder to learn the features in the dictionary and the sparse codes, and a second model to classify the residual vector calculated from the autoencoder output and its input. The combined models encourage the discriminative power of the classifier model, implicitly encouraging a dictionary with class-specific features, and implicitly encouraging a dictionary with shared features containing common patterns. The proposed method is applied to the context of SSVEP-BCI signals. The results showed that the proposed

method achieved higher classification accuracy than Sparse Representation Classification (SRC), although it performed worse than other methods. It is our belief that there is significant room for improvement in aspects like hyperparameter tuning, and this is a promising subject for future research.

- Chapter 6: In this chapter we summarise and discuss the results of this dissertation, pointing point out further directions of research.

Chapter 2

Background on Dictionary Learning and Sparse Coding

Signal processing involves the rigorous examination and manipulation of extensive data sets. The task of extracting pertinent information becomes particularly demanding in certain domains, such as denoising and signal classification. To overcome these challenges, employing a sparse representation of signals has proven to be a highly effective solution.

By adopting this methodology, the processing speed and efficiency can be significantly enhanced, rendering it a widely favored option in numerous signal processing applications. A prevalent technique for achieving sparse signal representation involves decomposing signals into elementary waveforms derived from a collection of redundant elements known as a dictionary. A dictionary allows us to generate a robust data representation that captures essential information from complex signals in a streamlined and proficient manner.

There exist two primary methodologies for dictionary learning within the framework of sparse representation. The first one is the modelling approach, wherein dictionaries are constructed using non-adaptive functions such as DCT bases, Wavelets, and Curvelets. These functions serve as atoms to form the dictionaries. The second approach, known as the learning-based approach, involves

solving an optimisation problem to obtain dictionaries that enable more precise and sparse representation of the original signals.

Wavelet bases and local time-frequency dictionaries have paved the way for numerous novel signal transformation techniques. These methods adapt sparse representations to accommodate the unique properties exhibited by specific signals, expanding the realm of possibilities in signal processing.

The modelling approach to dictionary learning presents notable advantages, primarily its inherent simplicity and computational efficiency, enabling the generation of sparse representations from a predetermined and known dictionary. However, a significant limitation of this approach lies in the potential unsuitability of the resulting transforms under certain domains and for certain families of signals. Additionally, there are limited means of identifying such inadequacies in advance. Therefore the efficacy of these approaches is contingent upon the suitability of the chosen dictionary for data representation.

Conversely, the learning-based approach is custom-tailored to optimise sparsity and other desired properties. This approach offers the advantage of adaptability to the data while also fulfilling specific objectives of interest, such as the power of classifying represented signals. Nonetheless, a noteworthy drawback of this approach is the potential for a higher computational burden compared to the modelling approach utilising preexisting base functions to build dictionaries.

In the realm of real vector spaces, an orthogonal basis is a dictionary of minimum size characterised by its atoms being mutually orthogonal, ensuring their independence. Such a basis might be designed to concentrate the energy of the signals across a small subset of vectors. On the other hand, an overcomplete dictionary encompasses a larger collection of vectors compared to an orthogonal basis. The vectors in an overcomplete dictionary are not required to be orthogonal, providing greater flexibility and enabling richer and redundant data representations. These dictionaries are frequently employed in constructing

sparse representations. The analogy for overcomplete dictionaries can be drawn to a vast set of natural languages, allowing for the formation of concise and precise sentences, as first indicated by (MALLAT; ZHANG, 1993). However, the selection of vectors for redundant dictionaries is a complex undertaking that necessitates signal decomposition and optimisation algorithms.

Building on the understanding of signal representation, further insights were gained through the pioneering work of *Hubel and Wiesel* in 1962, who demonstrated that a significant proportion of cells within the visual cortex exhibit a preference for specific edges at particular angles. Subsequently, it has been established that these cells possess receptive fields that are spatially localised and oriented, and are capable of selective processing of the structural features of visual input across different spatial scales.

One approach towards understanding the response properties of visual neurons was introduced in *Olshausen and Field's* seminal work in 1996 (OLSHAUSEN; FIELD, 1996). The authors pursued the topic of sparse coding by investigating the relationship between visual neurons and the statistical structure of natural images, with the aim of developing efficient coding strategies. They proposed an algorithm to identify sparse linear codes for natural images, and designed a comprehensive set of localised, oriented, and bandpass receptive fields that closely resemble those found in the primary visual cortex. Furthermore, they posited that such neurons could model the structure of images, denoted by $I(x, y)$, as a linear superposition of basis functions ϕ_i and additive noise $\epsilon(x, y)$, i.e.,

$$I(x, y) = \sum_i s_i \phi_i(x, y) + \epsilon(x, y) \tag{2.1}$$

where (x, y) denotes the two-dimensional position within an image. This formulation provides a useful framework for understanding how visual neurons process and represent natural images. The coefficient values, denoted by s_i represent the activities of neurons in a cortical patch responsible for encoding the

relevant image region. The basis functions, denoted by ϕ_i, may be viewed as a set of feature vocabulary or dictionary atoms that need not be orthogonal to one another. Notably, the objective of efficient coding is to determine a set of basis functions ϕ_i that span the image space, while ensuring that the coefficient values are sparse and statistically independent across a given ensemble of natural images.

The concept of sparse coding has emerged as a critical principle for neural representations of sensory input, particularly in the visual system. This principle is evident in the visual cortex, where neurons encode information using only a small number of active units. The notion of statistical independence in coding results in a reduced redundancy of the code and reflects the biological fact that neurons are assumed to be independent of the activity of their neighboring cells.

One potential solution to address the challenges of sparseness and independence is based on Principal Component Analysis (PCA), which aims to discover a set of mutually orthogonal basis functions that capture the directions of maximum variance in the data and produce pairwise uncorrelated coefficients s_i. Nevertheless, the receptive fields generated by PCA are not spatially localised, which is a desirable feature. Additionally, PCA assumes that the underlying data follows a Gaussian distribution, which is not the case for many natural scenes that exhibit higher order forms of statistical structure. Independent Component Analysis (ICA) has the potential to overcome this limitation by accounting for non-Gaussian distributions (LEWICKI; SEJNOWSKI, 2000). Nonetheless, both techniques share two common limitations: (I) they cannot separate the signal structure from the noise in the input, and (II) they cannot allow for overcomplete codes, where the number of basis functions exceeds the input dimensions.

In (OLSHAUSEN; FIELD, 1997), the authors proposed a different ap-

proach to sparse modeling than previously established methods. Whereas classical techniques in signal processing used fixed dictionaries, the method introduced in their work consists of learning from training data. They demonstrated that dictionary learning could easily recognise underlying structures in natural image patches. In the past decades, their approach found several other applications in many different fields such as in image and audio processing (see, e.g., (ELAD; AHARON, 2006), (YANG *et al.*, 2010a), (YANG *et al.*, 2010b), and (RAMIREZ *et al.*, 2010)). Recent research has focused on applying dictionary learning in areas such as image classification (WANG *et al.*, 2022), anomaly detection (PILASTRE *et al.*, 2020), and medical imaging (MIAO *et al.*, 2020). There has also been a growing interest in combining dictionary learning with other machine learning methods such as deep neural networks (TANG *et al.*, 2020) for improved performance in tasks such as face recognition and hyperspectral image analysis.

In signal processing, the observations, or data vectors, are called signals, and data modeling is an important step for many processing tasks such as denoising, compression, and for solving inverse problems. The sparsity principle also plays an important role in these scenarios (see, e.g., (MALLAT; ZHANG, 1993), (DONOHO, 2006)). Signals are approximated by a sparse linear combination of prototypes called dictionary elements or atoms, resulting in simple and compact models.

In this chapter, we will further describe the following aspects of dictionary learning: in Section 2.1, we overview the problem illustrating further challenges involving dictionary learning and sparse coding solution. We introduce a model formulation, in which we attempt to obtain the dictionary D and coefficients X with the best trade-off regarding sparsity and fidelity to the observed data. We also exploit the non-convex formulation of the problem, the ℓ_0-penalty and ℓ_1 approximation. In Section 2.3, we describe how this optimisation problem has been recently handled through convex framework that iteratively solves the problem with respect to either the dictionary or the sparse code, updating one

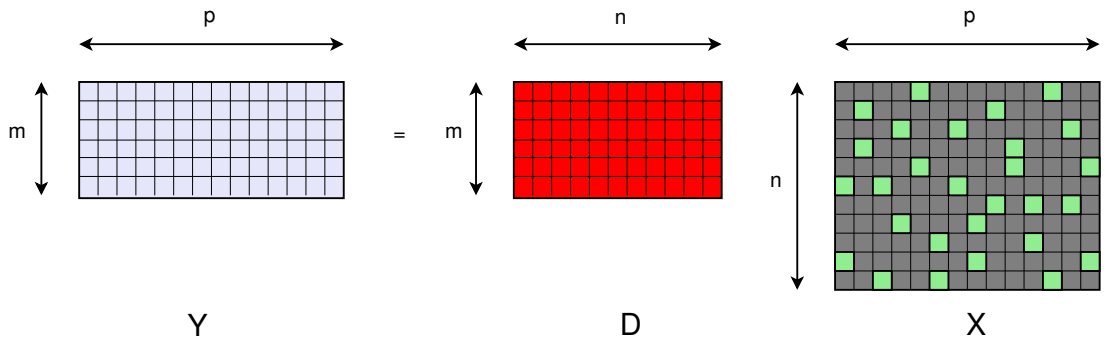


Figure 2.1 – Example of the sparse dictionary learning framework: the columns of matrix D represent the atoms of the dictionary. The columns of matrix Y contain the training samples, $\mathbf{y}_i$. These samples are used to optimise the dictionary and its corresponding sparse representations in the columns of X. In this particular example, each sample $\mathbf{y}_i$ is approximated as a linear combination of $k = 2$ dictionary atoms, based on the sparse coefficients highlighted in green in the columns of X.

and fixing the other variable. Finally, in Section 2.2, we present the results introduced in (LEWICKI; SEJNOWSKI, 2000), showing that overcomplete bases can yield a better approximation of the underlying statistical distribution of the data and can thus lead to greater coding efficiency.

2.1 Problem Overview

Consider a matrix D, each column being referred to as a dictionary atom. A dictionary is classified as *undercomplete* if the number of columns n is smaller than the number of rows m, or, conversely, *overcomplete* in the case that $n > m$. Typically, the framework for sparse dictionary learning assumes overcompleteness, allowing for more flexible dictionary structures and richer data representations.

Sparse coding aims to build succinct representations of input the data using a linear combination of only a few dictionary atoms. The dictionary is learned from the input data. Sparse coding has been widely used in applications such as image and audio processing and machine learning.

Given a set of input signals $Y \in \mathcal{R}^{m \times p}$, $Y = \{\mathbf{y_1}, \mathbf{y_2}, \cdots, \mathbf{y_p}\}$, sparse coding aims at finding a good and sparse approximation $\mathbf{x}_j \in \mathcal{R}^n$ of the signal

$\mathbf{y}_j$ using the linear combination of the dictionary atoms $D = \{\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_n\}$ $\mathbf{d}_j \in \mathcal{R}^m$, i.e., $\mathbf{y}_j = \sum_{k=1}^{n} \mathbf{d}_k x_j^k$ and most coefficients x_j^k are zero or very close to zero. Figure 2.1 helps to illustrate the problem formulation.

Sparse coding can be typically formulated as the following optimisation problem:

$$\min_{D,\{\mathbf{x}_i\}_1^p} \sum_{i=1}^{p} \|\mathbf{y}_j - D\mathbf{x}_j\|_2^2 + \lambda \|\mathbf{x}_j\|_0 \tag{2.2}$$

subject to $\|\mathbf{d}_j\| = 1, j = 1, 2, \cdots, n$. The constraint ensures that the dictionary atoms will not reach high values allowing for arbitrarily low (but nonzero) values. To measure the sparsity, the ℓ_0 penalty is defined as the number of nonzero elements of a vector, i.e., $\|\mathbf{x}\|_0 = \#\{j|x_j \neq 0\}$, where $\#\{\}$ indicates the cardinality of a set. The vector is called sparse if $\|\mathbf{x}\|_0 << n$, $\mathbf{x} \in \mathcal{R}^n$.

The optimisation problem presented in equation (2.2) is a non-convex problem due to the following facts:

- The presence of the sparsity term, promoted by the ℓ_0 penalty;
- The optimisation is performed jointly over both the dictionary D and the sparse codes in X, which interact with each other in the first term of equation (2.2).

Due to the bi-linearity in the first term of Equation (2.2), this problem can become a bi-convex optimisation problem with respect to each of the variables D and x_j, i.e., x_j is optimised when D is fixed and vice-versa.

The problem in Equation (2.2) is a challenging NP-hard problem, and only sub-optimal solutions can be found in polynomial time. Most existing algorithms either use greedy algorithms to iteratively select locally optimal solutions (e.g. Orthogonal Matching Pursuit (OMP) (MALLAT; ZHANG, 1993)) or replace the non-convex ℓ_0 penalty with its convex relaxation ℓ_1 norm (e.g. Basis Pursuit (CHEN *et al.*, 2001)). The dictionary for sparse approximation is usu-

ally learned from training samples in order to maximise the efficiency of the sparse approximation, i.e., improving the sparsity degree.

The classical approaches for dictionary learning and sparse coding (see theoretical foundations from e.g. (AHARON *et al.*, 2006), (MAIRAL *et al.*, 2010), (RUBINSTEIN *et al.*, 2008)) involves an alternating iteration between the following tasks:

1. The sparse coding: calculating the coefficients α_i and;
2. The dictionary update: fixing α_i and updating D

This iterative approach has been widely used in the literature and provides a principled framework for dictionary learning.

Despite the success of the aforementioned alternating iterative methods, none of them established the global convergence property for Equation (2.2). Nevertheless, the work from (BAO *et al.*, 2016) has recently proposed a multi-block alternating proximal method with global convergence property for solving a class of ℓ_0-penalty for non-convex problems arising from sparse coding based applications.

Several algorithms have been developed to solve the problem of sparse coding and dictionary learning. The following sections are dedicated to explaining these methods in detail.

2.2 Tackling the Sparse Coding Problem

The sparse coding problem can be discussed and grouped into different categories according to the chosen norm regularisation. The general goal of sparse representation is to recover a set of signals from the most sparse coefficients of the linear combination of dictionary atoms resulting also on the

minimum representation error between the input signals and the sparse approximation. Formally, if $\mathbf{y}$ is a column signal $\mathcal{R}^m$ and $D \in \mathcal{R}^{m \times n}$ is the dictionary, the sparsity assumption can be described as the following sparsity approximation

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{s.t.} \quad \|\mathbf{y} - D\mathbf{x}\|_2^2 \leqslant \epsilon \tag{2.3}$$

or

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\mathbf{y} - D\mathbf{x}\|_2^2 \quad \text{s.t.} \quad \|\mathbf{x}\|_0 \leqslant K \tag{2.4}$$

where $\mathbf{x}$ is the sparse representation of $\mathbf{y}$, ϵ is the error tolerance and the ℓ_o-penalty is defined as

$$\|\mathbf{x}\|_0 = \#\left\{i = 1,\ 2,\ ,\ \cdots,\ p | x^i \neq 0\right\} \tag{2.5}$$

The sparse coding problem can be formulated as the task of approximating multiple input signals $Y = [\mathbf{y}_1,\ \mathbf{y}_2,\ \cdots,\ \mathbf{y}_p]$ simultaneously. This involves finding optimal liner combinations of dictionary atoms to achieve good approximations. The problem can be defined in terms of minimising the error in representation ($\sum_{i=1}^{p} \|\mathbf{y}_i - D\mathbf{x}_i\|_2^2 = \|Y - DX\|_F^2$) while applying a penalty term through an ℓ_0 norm constraint to promote sparsity in the columns of the coefficient matrix X ($\|\mathbf{x}_i\|_0 \leqslant K, \forall i \in \{1,\ 2,\ \cdots,\ p\}$).

The structure of the sparse signal $\mathbf{x}$ is significantly influenced by the regularisation imposed in the formulation shown in Equation (2.3). Using the number of nonzero coefficients as a penalty term is problematic due to the non-differentiable and nonconvex nature of the ℓ_0 term. As a result, approximations to this term have been proposed to address the challenges posed by the ℓ_0 NP-hard problem. There are two main approaches for approximating the original sparsity penalty term:

1. using a convex relaxation or
2. using a greedy algorithm.

Due to it faster convergence, greedy approaches are computationally more convenient than the convex relaxation ones. Greedy methods choose locally optimal solutions at each stage. However, the reconstruction errors obtained from it are relatively larger than the ℓ_1-norm solutions. Convex relaxations, such as Basis Pursuit and Lasso algorithm, have better theoretical guarantees and recovery ability, but are more time consuming.

Notice that the optimisation task described in Equations (2.3) and (2.4) can be changed to

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_0 \tag{2.6}$$

so that the constraint becomes a penalty term weighted by the constant λ. There are reasonably good conditions wherein ℓ_0 and ℓ_1 regularisation are said to be equivalent (for a proper choice of λ, the problems are equivalent to each other). However, such equivalence is not trivial as considering that the original problem of ℓ_0 regularisation is NP-hard. It is also noteworthy to mention that the regularisation parameter λ actually controls the trade-off between overfitting and sparsity of the resulting target function in both cases, to ℓ_0 and to ℓ_1 regularisation.

The greedy strategy searches for the best local optimal solution in each iteration (TROPP, 2004). For the sparse representation method, the greedy strategy approximation chooses the K *most appropriate* atoms to approximate the input data vector $\mathbf{y}_i$. Two algorithms of this type will be presented in this section:

1. The Orthogonal Matching Pursuit (OMP), which was introduced for simple sparse approximation in (DAVIS *et al.*, 1997) and (PATI *et al.*, 1993). At each iteration, a greedy pursuit makes the best local improvement to the current approximations in hopes of obtaining a good overall solution.

2. The Batch-OMP algorithm, whose major advantage is that it has a simple and fast implementation originally described in (RUBINSTEIN *et al.*,

2008).

In addition, we will introduce the convex relaxation approach, which addresses the problem of sparse coding by formulating it as a convex optimisation problem. This formulation allows for efficient polynomial time solutions using standard mathematical programming software (CHEN *et al.*, 2001). The Basis Pursuit (BP) algorithm is a benchmark of this approach. It ransforms the original sparse approximation problem into a linear programming problem with an ℓ_1-penalty term (CHEN *et al.*, 2001).

2.2.1 The Orthogonal Matching Pursuit (OMP)

A Matching Pursuit (MP) is a family of greedy algorithms that iteratively refines the signal approximation instead of directly solving the optimal approximation problem (DAVIS *et al.*, 1997). The objective of Matching Pursuit algorithms is to approximate the solution of the sparsity-constrained problem, defined as follows:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\mathbf{y} - \mathsf{D}\mathbf{x}\|_2^2 \quad \text{s.t} \quad \|\mathbf{x}\|_0 \leqslant \mathsf{K} \tag{2.7}$$

or the error-constrained sparse coding problem, given by

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{s.t} \quad \|\mathbf{y} - \mathsf{D}\mathbf{x}\|_2^2 \leqslant \epsilon \tag{2.8}$$

For a signal $\mathbf{y}$ and a fixed dictionary D, the algorithm iteratively generates a sorted list of atom indices based on their *importance*. The algorithm also determines scalar coefficients that weight these atoms, resulting in locally-optimal solutions.

Pseudocode 2.1 Matching Pursuit:

Initialise Variables:

- Initialise Sparse Coefficients

$$\hat{\mathbf{x}} = \mathbf{0}$$

- Initialise Residual

$$\mathbf{r} = \mathbf{y}$$

while $(\|\hat{\mathbf{x}}\|_0 \leqslant K)$ **or** $\left(\|\mathbf{y} - D\hat{\mathbf{x}}\|_2^2 \leqslant \epsilon\right)$ **:**

- Select the dictionary atom $\mathbf{d}_i$ with maximum absolute inner product with the residual $\mathbf{r}$

$$\hat{i} = \underset{i=\{1,\ \cdots,\ n\}}{\arg\max}\ |\mathbf{d}_i^T \mathbf{r}|$$

- Update the residual, and the sparse coefficient at $\hat{i}$-th position of $\mathbf{x}$

$$\hat{\mathbf{x}}[\hat{i}] = \hat{\mathbf{x}}[\hat{i}] + \mathbf{d}_{\hat{i}}^T \mathbf{r}$$

$$\mathbf{r} = \mathbf{r} - (\mathbf{d}_{\hat{i}}^T \mathbf{r})\mathbf{d}_{\hat{i}}$$

An extension of the Matching Pursuit (MP) algorithm is the Orthogonal Matching Pursuit (OMP). In OMP, a notable distinction from MP lies in the coefficient update step. At each iteration, the coefficients in $\mathbf{x}$ are updated by performing the orthogonal projection of the signal onto the subspace spanned by the previously selected atoms, rather than solely onto the current atom. Incorporating this orthogonal projection enhances the algorithm's capability to achieve improved results when compared to the standard MP approach.

The OMP algorithm follows a greedy strategy where, at each step, the atom with the highest absolute inner product with the current residual is selected. Once this atom is chosen, the signal is orthogonally projected onto the span of the entire set of previously selected atoms. The residual is then recomputed,

and the process continues until convergence is achieved (RUBINSTEIN *et al.*, 2008).

The selection of the most relevant dictionary atom is based on its absolute inner product value with the residual vector $\mathbf{r}[k]$ at the k-th iteration of the algorithm. The residual vector plays a crucial role in extracting the most significant columns of the dictionary D. Let $\mathbf{d}_i$ denote the i-th column of D, and $\lambda[k]$ denote the column index of D with the highest absolute inner product with the residual at the k-th iteration. The index set $\Lambda_{[k]}$ stores all the indices of the most important atoms of D in terms of signal representation.

The residual vector $\mathbf{r}_{[k]}$ is initialised as $\mathbf{r}_{[k]} = \mathbf{y}$, and $\Lambda_{[k]} = \emptyset$. The index $\lambda_{[k]}$ is then calculated as

$$\lambda_{[k]} = \arg\max_i |\mathbf{d}_i^T \mathbf{r}| \tag{2.9}$$

Note that the columns of D must be normalised to unit ℓ_2-norm to make sure that the inner product between any two columns is within the range $[-1, +1]$ and hence the absolute value of the inner product between any two columns is bounded, i.e., $0 \leqslant |\langle \mathbf{d}_i, \mathbf{d}_j \rangle| \leqslant 1$.

At k-th iteration, the active index set is augmented to $\Lambda_{[k]} = \Lambda_{[k-1]} \cup \{\lambda_{[k]}\}$. The sparse codes $\hat{\mathbf{x}}[\Lambda_{[k]}]$, i.e., the nonzero coefficients at the index positions stored in $\Lambda_{[k]}$ are updated by solving the least squares problem with the sub-matrix $D_{\Lambda_{[k]}}$

$$\hat{\mathbf{x}}[\Lambda_{[k]}] = \arg\min_{\mathbf{x}} \left\| \mathbf{y} - D_{\Lambda_{[k]}} \mathbf{x} \right\|_2^2 \tag{2.10}$$

It is important to note that the computation of $\hat{\mathbf{x}}[\Lambda_{[k]}]$ can be computationally intensive. This is mainly due to the matrix inverse operation when calculating the pseudoinverse of $D_{\Lambda_{[k]}}$, as follows:

$$\hat{\mathbf{x}}[\Lambda_{[k]}] = (D_{\Lambda_{[k]}}^T D_{\Lambda_{[k]}})^{-1} D_{\Lambda_{[k]}}^T \mathbf{y} \tag{2.11}$$

Therefore, practical implementations usually employ a progressive Cholesky update process (BLUMENSATH; DAVIES, 2008) to reduce computational costs.

Finally, the residual is then updated to

$$\mathbf{r}_{[k]} = \mathbf{y} - \mathbf{D}_{\Lambda_{[k]}}(\mathbf{D}_{\Lambda_{[k]}}^{T}\mathbf{D}_{\Lambda_{[k]}})^{-1}\mathbf{D}_{\Lambda_{[k]}}^{T}\mathbf{y} \tag{2.12}$$

The main loop finishes either when $k = K$, due to the sparse-constrain, or when the error-constrain $\left\|\mathbf{y} - \mathbf{D}\mathbf{x}_{[k]}\right\|_2^2 \leqslant \epsilon$ is achieved. The OMP algorithm is summarised next.

Pseudocode 2.2 Orthogonal Matching Pursuit:

Initialise Variables:

- Initialise Sparse Coefficients

$$\hat{\mathbf{x}} = \mathbf{0}$$

- Initialise Residual

$$\mathbf{r} = \mathbf{y}$$

- Initialise the set of active indices

$$\Lambda_{[k]} = \emptyset$$

while $(\|\hat{\mathbf{x}}\|_0 \leqslant K)$ **or** $\left(\|\mathbf{y} - \mathbf{D}\hat{\mathbf{x}}\|_2^2 \leqslant \epsilon\right)$ **:**

- Select the atom with maximum absolute inner product with the residual

$$\lambda_{[k]} = \underset{i=\{1,\ \cdots,\ n\}}{\arg\max}\ |\mathbf{d}_i^{T}\mathbf{r}|$$

- Update the set of active indices

$$\Lambda_{[k]} = \Lambda_{[k-1]} \cup \{\lambda_{[k]}\}$$

- Update the residual and the sparse coefficients

$$\hat{\mathbf{x}}[\Lambda_{[k]}] = (\mathbf{D}_{\Lambda_{[k]}}^{T}\mathbf{D}_{\Lambda_{[k]}})^{-1}\mathbf{D}_{\Lambda_{[k]}}^{T}\mathbf{y}$$

$$\mathbf{r}_{[k]} = \mathbf{y} - \mathbf{D}_{\Lambda_{[k]}}(\mathbf{D}_{\Lambda_{[k]}}^{T}\mathbf{D}_{\Lambda_{[k]}})^{-1}\mathbf{D}_{\Lambda_{[k]}}^{T}\mathbf{y}$$

2.2.2 The Batch-Orthogonal Matching Pursuit

Batch-OMP is a variant of the Orthogonal Matching Pursuit (OMP) algorithm that has been specifically optimised for efficiently performing sparse coding on large sets of signals using the same dictionary. A key insight presented in Rubinstein et al. (RUBINSTEIN *et al.*, 2008) is that the atom selection step in each iteration does not require the explicit calculation of the residue, $\mathbf{r}[k] = \mathbf{y} - \langle \mathbf{y}, \mathbf{dj} \rangle \frac{\mathbf{dj}}{\langle \mathbf{dj}, \mathbf{dj} \rangle}$, or the sparse representation $\hat{\mathbf{x}}[k] = (D\Lambda[k]^T D\Lambda[k])^{-1} D^T_{\Lambda_{[k]}} \mathbf{y}$. Instead, only the term $D^T\mathbf{r}$ is required, which facilitates the selection of the most relevant atoms from the dictionary as $\lambda_{[k]} = \arg\max_j |\langle \mathbf{dj}, \mathbf{r}[k] \rangle|$.

It is noteworthy that the matrix $D^T_{\Lambda_{[k]}} D_{\Lambda_{[k]}}$ is always symmetric and positive definite. Exploiting this property, we can bypass the need for inverting large matrices thus reducing computational costs. By decomposing the term from Equation (2.11) into two triangular matrices L and L^T, we can employ the incremental Cholesky decomposition. This iterative process leverages the results from previous iterations and adds a single new row and column to L at each step. The decomposition can be expressed as follows:

$$L = \begin{pmatrix} L & 0 \\ \boldsymbol{w}^T & \sqrt{1 - \boldsymbol{w}^T \boldsymbol{w}} \end{pmatrix} \tag{2.13}$$

The fundamental concept underlying this algorithm is to substitute the explicit computation of $\mathbf{r}$ and its multiplication by D^T with a more efficient calculation of $D^T\mathbf{r}$. Letting $\alpha = D^T\mathbf{r}$, $\boldsymbol{\alpha}^0 = D^T\mathbf{y}$, and $G = D^TD$, we can express this as follows:

$$\begin{aligned} \boldsymbol{\alpha} &= D^T\mathbf{r} && (2.14) \\ &= D^T(\mathbf{y} - D_\Lambda (D_\Lambda)^+ \mathbf{y}) && (2.15) \\ &= \boldsymbol{\alpha}^0 - G_\Lambda (G_{\Lambda,\Lambda})^{-1} \boldsymbol{\alpha}^0_\Lambda && (2.16) \end{aligned}$$

Hence, given pre-computed values of $\boldsymbol{\alpha}^0$ and G, we can calculate $\boldsymbol{\alpha}$ in each iteration without the explicit computation of $\mathbf{r}$. The updated step is

modified by replacing the multiplication with D^T by the matrix G_Λ. The term $G^{-1}_{\Lambda,\Lambda}$ is evaluated using progressive Cholesky factorisation. This method can also be extended to the error-driven case by deriving efficient incremental formulas for the squared error $|\mathbf{r}^{[k]}|_2^2$. The initial step involves expressing the residual in terms of the sparse approximations at the k-th iteration and its previous iteration, denoted by $[k-1]$, i.e.,

$$\begin{aligned} \mathbf{r}^{[k]} &= \mathbf{y} - D\mathbf{x}^{[k]} && (2.17) \\ &= \mathbf{r}^{[k-1]} - D(\mathbf{x}^{[k-1]} - \mathbf{x}^{[k]}) && (2.18) \end{aligned}$$

The second step involves leveraging the fact that the orthogonalisation process in OMP guarantees the orthogonality between the residual and the current signal approximation at each iteration, as indicated below:

$$(\mathbf{r}^{[k]})^T D\mathbf{x}^{[k]} = 0 \tag{2.19}$$

By substituting the expression from Equation (2.19) into Equation (2.18), we obtain the following formula for updating the error

$$\left\|\mathbf{r}^{[k]}\right\|_2^2 = \left\|\mathbf{r}^{[k-1]}\right\|_2^2 - (\mathbf{x}^{[k]})^T G\mathbf{x}^{[k]} + (\mathbf{x}^{[k-1]})^T G\mathbf{x}^{[k-1]} \tag{2.20}$$

2.2.3 Basis Pursuit

Basis Pursuit (BP) is a method for finding signal representations in overcomplete dictionaries through convex optimisation. It aims to find a decomposition that minimises the ℓ_1 norm of the sparse coefficients. BP uses linear programming. Advancements in large-scale linear programming methods, specifically those associated with interior-point methods, can be applied to BP. These advancements have the potential to solve the BP optimisation problem nearly linearly for certain dictionaries.

The principle of BP is to seek a signal representation whose coefficients possess the smallest ℓ_1 norm. Mathematically, this problem is formulated

in (CHEN *et al.*, 2001) as follows:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\mathbf{y} - \mathsf{D}\mathbf{x}\|_2^2 - \lambda \|\mathbf{x}\|_1 \tag{2.21}$$

This forms the foundation for both LASSO and BP, which are similar techniques but were developed independently by different research communities. LASSO originated from the statistics community, whereas BP emerged from the signal processing domain. BP involves solving a convex quadratic optimisation problem.

The obtained solution $\hat{\mathbf{x}}^\lambda$ is dependent on the parameter λ and provides a decomposition of the signal into a signal component and a residual component as follows

$$\mathbf{y} = \mathsf{D}\hat{\mathbf{x}}^\lambda + \mathbf{r} \tag{2.22}$$

The size of the residual is controlled by λ. As $\lambda \to 0$, the residual goes to zero and the solution behaves exactly like BP applied to $\mathbf{y}$. As $\lambda \to \infty$, the residual gets large, $\mathbf{r}(\lambda) \to \mathbf{y}$ and $\mathsf{D}\hat{\mathbf{x}}^\lambda \to \mathbf{y}$.

Equation (2.21) always has a solution, although it may not be unique. It can be equivalently reformulated as a quadratic optimisation problem, which can be solved using the Quadratic programming (QP). The quadratic formulation is given as follows

$$\min_{\mathbf{u},\boldsymbol{v},\mathbf{p}} \quad \lambda \mathbf{1}^\mathsf{T} \begin{bmatrix} \mathbf{u} \\ \boldsymbol{v} \end{bmatrix} + \frac{1}{2}\mathbf{p}^\mathsf{T}\mathbf{p} \tag{2.23}$$

$$\text{s.t.} \qquad \begin{bmatrix} \mathsf{D} & -\mathsf{D} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \boldsymbol{v} \end{bmatrix} + \mathbf{p} = \mathbf{y}; \tag{2.24}$$

$$\mathbf{u}, \qquad \boldsymbol{v} \geqslant 0 \tag{2.25}$$

where $\mathbf{x} = \mathbf{u} - \boldsymbol{v}$. The QP problem can be effectively solved using various general solvers, including interior point methods. However, for exceptionally large problem instances, specialised methods that surpass the efficiency of interior point methods have been developed. One such approach is the alternating direction

method of multipliers (BOYD *et al.*, 2011) (ADMM), which has been proposed in several variants for solving large-scale ℓ_1 problems.

The Alternating Direction Method of Multipliers is a popular a optimisation technique used in many fields. In the context of dictionaries and sparse coding, the optimisation objective is divided into three subproblems: updating X by solving a sparse coding problem with a penalty term, updating D by solving a dictionary update problem, and updating the auxiliary variable by adjusting the Lagrange multiplier. These subproblems are solved iteratively until convergence is reached. The method leverages the alternating minimisation strategy to jointly optimise X and D, while the auxiliary variable and Lagrange multiplier ensure consistency between the variables.

2.3 Tackling the Dictionary Learning Problem

Traditional dictionary learning methods can be divided into three main categories:

- Probabilistic learning methods, including the Maximum Likelihood (ML) Dictionary Learning Method, the Maximum a Posteriori Probability (MAP) Method for Dictionary Learning, and the Method of Optimal Directions (MOD).
- Learning methods based on clustering, such as K-SVD.
- Methods for learning dictionaries with specific characteristics. These methods are typically guided by prior knowledge about the structure of the data or the intended application of the learned dictionary.

This section aims to provide a comprehensive overview of representative algorithms in the first two categories of dictionary learning methods listed above, highlighting their fundamental principles and key contributions.

2.3.1 Maximum Likelihood Dictionary Learning Method

The methods introduced in the works of Olshausen and Field (OLSHAUSEN; FIELD, 1996) and Lewicki and Sejnowski (LEWICKI; SEJNOWSKI, 2000) employ a probabilistic framework, assuming that the input data matrix $Y = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_p]$ is independently and identically distributed (i.i.d.). These methods propose a generative model that describes the underlying process of data generation, which can be expressed as follows

$$Y = DX + \epsilon \tag{2.26}$$

with ϵ being a Gaussian white residual vector with zero mean $\mu = 0$ and variance σ^2. Since the examples in each column of Y are i.i.d,

$$P(Y|D) = \prod_{i=1}^{p} P(\mathbf{y}_i|D) \tag{2.27}$$

The likelihood function requires integrating out the hidden (unobservable) zero mean, i.i.d. source vectors, $X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_p]$, i.e.,

$$P(\mathbf{y}_i|D) = \int P(\mathbf{y}_i, \mathbf{x}|D)\, dx = \int P(\mathbf{y}_i|\mathbf{x}, D)P(\mathbf{x})\, dx \tag{2.28}$$

From the Gaussian assumption made over the noise model,

$$P(\mathbf{y}_i|\mathbf{x}, D) = C \exp\left\{-\frac{1}{2\sigma^2}\|\mathbf{y}_i - DX\|^2\right\} \tag{2.29}$$

for C a constant value. Different authors suggested different distributions for $P(\mathbf{x})$. As a matter of fact, regardless the assumed distribution, the integration over $\mathbf{x}$ is not easy to evaluate in Equation (2.28), because it requires integrating over all possible states of $\mathbf{x}$, which is in general intractable (OLSHAUSEN; FIELD, 1997). Indeed, in (OLSHAUSEN; FIELD, 1996), the authors formulate the optimisation as follows

$$D = \arg\max_{D} \sum_{i=1}^{p} \max_{x_i} \{P(\mathbf{y}_i|\mathbf{x}_i, D)\} \tag{2.30}$$

$$D = \arg\min_{D} \sum_{i=1}^{p} \min_{x_i} \left\{\|\mathbf{y}_i - D\mathbf{x}_i\|_2^2 + \lambda \|\mathbf{x}_i\|_1\right\} \tag{2.31}$$

The price paid for this approximation is the possibility of obtaining trivial solutions for the sparse vectors $\mathbf{x}_i$, as the norm of the dictionary atoms $\mathbf{d}_k$ increases, causing the individual elements of $\mathbf{x}_i$ to decrease. To address this issue, a constraint on the ℓ_2-norm of each atom is imposed.

To solve Equation (2.31), a two-phase iterative method, consisting of an outer and inner phase, is proposed in (OLSHAUSEN; FIELD, 1997). In the inner phase, Equation (2.31) is minimised with respect to $\mathbf{x}_i$ for each input signal while keeping D fixed. This is achieved through a network implementation where each output unit represents the value of a single coefficient x_i^j. The output activities are then fed back through the dictionary atoms to reconstruct the signal $\mathbf{y}_i$. The reconstructed signal is subtracted from the input signal, and the resulting residual is propagated forward to update each output x_i^j.

The second step involves updating the dictionary using a gradient descent method, given by

$$D^{(n+1)} = D^{(n)} - \eta \sum_{i=1}^{p} \left(D^{(n)} \mathbf{x}_i - \mathbf{y}_i \right) \mathbf{x}_i^T, \tag{2.32}$$

where η represents the learning rate.

2.3.2 Maximum a Posteriori Probability Method for Dictionary Learning

In the Maximum a Posteriori (MAP) probability framework, rather than working with the likelihood function $P(Y|D)$, the posterior $P(D|Y)$ is considered. It incorporates statistical preferences on the learned dictionary. By incorporating prior information, the posterior distribution is given by $P(D|Y) \propto P(Y|D)P(D)$, where the likelihood expression from the Maximum Likelihood (ML) approach is combined with a prior distribution $P(D)$. For instance, when imposing the prior which constrains the dictionary to have a unit Frobenius norm $\|D\|_F = 1$, the update expression is obtained as follows:

$$D^{(n+1)} = D^{(n)} + \eta \left(EX^T + \mathrm{Tr}\left({D^{(n)}}^T EX^T \right) D^{(n)} \right) \tag{2.33}$$

where

$$E = \left\| Y - D^{(n)} X \right\|_F^2 \tag{2.34}$$

is the representation mean squared error and $\eta > 0$ is the learning rate. However, the previous Frobenius norm constraint $\|D\|_F = 1$ introduces a challenge where certain columns of D tend to approach zero. This issue arises because the algorithm penalises the dictionary columns associated with terms in X that have large magnitudes. If an atom $\mathbf{d}_k$ has a relatively small magnitude, the weight of its coefficient $x[i]$ can be large, resulting in higher penalties compared to those from columns with a larger norms. Consequently, this constraint can lead to underuse of certain atoms.

This observation motivates the exploration of an alternative and more restrictive form of the constraint, where each atom $\mathbf{d}_k$ is normalised to a constant value. Specifically, we enforce the condition $\|\mathbf{d}_k\|_2 = c$, where $c > 0 \in \mathbb{R}^k$. The adoption of this second prior leads to a modified update expression

$$\mathbf{d}_i^{(n+1)} = \mathbf{d}_i^n + \eta \left(I - \mathbf{d}_i^n \mathbf{d}_i^{(n+1)^T} \right) E \mathbf{x}_i^T \tag{2.35}$$

where $\mathbf{x}_i^T$ is the i-th column of the matrix X^T.

2.3.3 Method of Optimal Directions (MOD)

The Method of Optimal Directions (MOD) was among the pioneering methods proposed to address the problem of sparse dictionary learning. The fundamental principle of this method involves solving a minimisation problem with an ℓ_0-penalty constraint, formulated as follows:

$$\min_{D,X} \quad |Y - DX|_F^2 \text{ s.t.} \qquad\qquad |\mathbf{x}_i|_0 \leqslant T, \quad \forall i$$

The first term in the objective function penalises the representation error. The use of the Frobenius norm is equivalent to the ℓ_2-norm when considering the vector form, i.e., $\min_{D,\{\mathbf{x}_i\}_1^p} \sum_{i=1}^p \|\mathbf{y}_j - D\mathbf{x}_j\|_2^2$.

The MOD algorithm iteratively alternates between the sparse coding task, employing methods such as matching pursuit, and updating the dictionary by computing the analytical solution using the Moore-Penrose pseudoinverse X^{+} i.e.,

$$\mathsf{D} = \mathsf{Y}\mathsf{X}^{+} \tag{2.36}$$

Alternatively, the dictionary update can be expressed as

$$\mathsf{D} = \mathsf{Y}\mathsf{X}^{\mathsf{T}}(\mathsf{X}\mathsf{X}^{\mathsf{T}})^{-1} \tag{2.37}$$

Following the dictionary update, the atoms in D are normalised to satisfy the unit-norm constraints. Subsequently, a new sparse coding step is performed, and this iterative process continues until convergence, which is typically achieved when the residue becomes sufficiently small.

Although the MOD is known for its efficiency in handling low-dimensional input data X, requiring only a small number of iterations for convergence, it faces challenges when dealing with large datasets. Specifically, the computational complexity associated with the matrix inversion operation, required for computing the pseudoinverse, renders it impractical in many cases. Consequently, this limitation has driven the exploration of alternative dictionary learning methods.

2.3.4 K-SVD

The K-SVD is an algorithm introduced in the works of Aharon et al. (AHARON *et al.*, 2006) and Rubinstein et al. (RUBINSTEIN *et al.*, 2008). It employs Singular Value Decomposition (SVD) to iteratively update the atoms of the dictionary one by one, while ensuring that each element of the input data $\mathbf{y}_i$ can be encoded by a linear combination of at most T atoms. The algorithm consists of two main steps:

- In this step, a sparse representation matrix X is generated by leveraging approximate solving methods such as Orthogonal Matching Pursuit (OMP),

Basis Pursuit (BP), Focal Under-Determined System Solver (FOCUSS), among others. Given the current fixed dictionary D, the sparse coding phase computes the sparse coefficients for each input signal, resulting in the matrix X;

- The dictionary update phase focuses on updating the dictionary atoms based on the current sparse representations. The underlying principle of this step involves updating one column of D at a time while keeping all other columns in D fixed, except for d_k. The update procedure for the k-th atom entails optimising a target function specific to the atom being updated.

OMP is an iterative greedy algorithm that selects at each step the column, which is most correlated with the current residuals. This method finds matrix X for a given fixed dictionary D, and formulates the optimisation problem using either the sparsity constraint or the representation error constraint, i.e.,

$$\min_X \|Y - DX\|_F^2 \tag{2.38}$$

$$\text{s.t.} \|x_i\|_0 \leqslant T, \quad \forall i \in \{1, \ , \cdots, \ p\} \tag{2.39}$$

or equivalently,

$$\min_X \|x_i\|_0, \quad \forall i \in \{1, \ , \cdots, \ p\} \tag{2.40}$$

$$\text{s.t.} \|Y - DX\|_F^2 \leqslant \epsilon \tag{2.41}$$

Further details regarding the solving approach employed in OMP can be found in Section 2.2.1 of the corresponding reference.

Once the sparse coding stage is completed and the matrix X is computed, the K-SVD algorithm proceeds to the second stage, which involves updating the dictionary. In this stage, the dictionary atoms d_k are updated individually while keeping the remaining atoms fixed. The objective is to minimise the mean

squared error (MSE) of the signal representations. By considering the coefficients of d_k as x_T^k, corresponding to the k-th row of X, the penalty term can be reformulated as follows:

$$\|Y - DX\|_F^2 = \left\|Y - \sum_{j=1}^{k} d_j x_{T}^{j}\right\|_F^2 \tag{2.42}$$

$$= \left\|(Y - \sum_{j \neq k} d_j x_T^j) - d_k x_T^k\right\|_F^2 \tag{2.43}$$

$$= \left\|E_k - d_k x_T^k\right\|_F^2 \tag{2.44}$$

where E_k refers to the approximation error without the k-th dictionary atom. However, it is important to note that the aforementioned update does not explicitly enforce any sparsity constraint. To address this, the K-SVD algorithm considers only a subset of examples denoted as Y_k^R, which specifically utilise the dictionary atom d_k. Additionally, a subset of error columns E_k^R is selected based on the use of the correspondent atom. To further promote sparsity, the coefficients of the row vector x_T^k are filtered by discarding the entries that are equal to zero. This results in the short version of x_T^k denoted as x_R^k.

To minimise the error E_k^R with respect to d_k, the authors propose employing SVD decomposition, expressed as $E_k^R = U\Delta V^T$. Accordingly, the solution for the dictionary atom $\tilde{d}_k$ is derived from the first column of U, which corresponds to the eigenvector associated with the maximum eigenvalue $\Delta_{(1,1)}$. This eigenvalue represents the maximum variance of the error matrix. Similarly, the coefficient vector x_R^k is updated by multiplying $\Delta_{(1,1)}$ with the first column of V. A noteworthy aspect of this approach is that it ensures that the support of all representations x_R^k either remains the same or becomes smaller, potentially containing null terms (AHARON *et al.*, 2006).

The K-SVD algorithm shares similar limitations to MOD, as it demonstrates efficiency primarily for training signals with relatively low dimensionality and is susceptible to local minima. In order to address these challenges, a fas-

ter approach was proposed in (RUBINSTEIN *et al.*, 2008), which employs an approximate solution that guarantees a complexity reduction in the final objective function. An important advantage of this method lies in its ability to avoid explicit computation of the matrix E_R^k, which is a computationally and memory-intensive operation. Instead, the approximate formulation involves computing matrix-vector products related to E_R^k, resulting in significant time and memory savings. To further enhance the dictionary update step, the implementation of the Approximate K-SVD algorithm incorporates Batch-OMP (RUBINSTEIN *et al.*, 2008) as the sparse coding method.

2.3.5 LASSO Approach

Conventional approaches to dictionary learning are predicated on the availability of a typical and sufficiently large input data set $Y = [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_p]$ for training the algorithm. However, when the input data is presented as a stream $\mathbf{y}_i$, this assumption is clearly violated, and such scenarios fall within the realm of online dictionary learning. In this field, the focus is on the iterative updates to the dictionary as new data points $\mathbf{y}_i$ become available.

The algorithm introduced in (MAIRAL *et al.*, 2010) indicates a methodology for updating D_t with respect to the input data stream received at time instant t. It works as follows

1. For the new input data sample $\mathbf{y}_t$, find a sparse coding using LARS (Least-Angle-Regression):

$$\mathbf{x}_t = \underset{\mathbf{x}\in\mathcal{R}^n}{\arg\min} \left(\frac{1}{2} \|\mathbf{y}_t - \mathbf{D}_{t-1}\mathbf{x}\| + \lambda \|\mathbf{x}\|_1 \right) \tag{2.45}$$

   where λ is a regularisation parameter. Despite being well known that the ℓ_1-norm yields a sparse solution for $\mathbf{x}_t$, there is no analytic link between the value of λ and the corresponding effective sparsity in $\mathbf{x}_t$ (MAIRAL *et al.*, 2010). To prevent D from being arbitrarily large (which would lead to

arbitrarily small values of $\mathbf{x}_t$), it is common to constrain its atoms $\mathbf{d}_k$ to have ℓ_2-norm less than or equal to one.

2. Update dictionary using block-coordinate descent approach: it computes D_t from its previous version D_{t-1} so that

$$D_t = \arg\min_D \frac{1}{t}\sum_{i=1}^{t}\left(\frac{1}{2}\|\mathbf{y}_i - D\mathbf{x}_i\|_2^2 + \lambda\|\mathbf{x}_i\|_1\right) \tag{2.46}$$

$$= \arg\min_D \sum_{i=1}^{t}\left(\frac{1}{2}\mathbf{x}_i^T D^T D\mathbf{x}_i - \mathbf{y}_i^T D\mathbf{x}_i\right) \tag{2.47}$$

$$= \arg\min_D \frac{1}{2}\mathrm{Tr}\left(D^T D A_t\right) - \mathrm{Tr}\left(D^T B_t\right) \tag{2.48}$$

where the matrices A_t and B_t essentially carry all the information from the past coefficients in the vectors $[\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_t]$ and all the information from the previous input data vectors $[\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_t]$, i.e., $A_t = \sum_{i=1}^{t}\mathbf{x}_i\mathbf{x}_i^T = A_{t-1} + \mathbf{x}_t\mathbf{x}_t^T$ and $B_t = \sum_{i=1}^{t}\mathbf{y}_i\mathbf{x}_i^T = B_{t-1} + \mathbf{y}_t\mathbf{x}_t^T$. The online adjustment of D for each new stream $\mathbf{y}_i$ is made for each dictionary atom (*in blocks*) by solving Equation (2.48). Taking $A_t = [\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_n] \in \mathcal{R}^{n\times n}$ and $B_t = [\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n] \in \mathcal{R}^{m\times n}$ the atoms $\mathbf{d}_j$ are updated as follows:

$$\mathbf{u}_j = \frac{1}{A_{j,j}}(\mathbf{b}_j - D_{t-1}\mathbf{a}_j) + \mathbf{d}_j^{t-1} \tag{2.49}$$

$$\mathbf{d}_j = \frac{1}{\max(\|\mathbf{u}_j\|_2, 1)}\mathbf{u}_j \tag{2.50}$$

The solution to Equation (2.48) with respect to the j-th column of D, while holding the other columns fixed under the constraint $\mathbf{d}_j^T\mathbf{d}_j \leqslant 1$, can be derived using a sequential least squares approach. This solution is expressed by Equation (2.50).

Chapter 3

Using Autocoders for Dictionary Learning and Sparse Coding

An autoencoder can be considered a semi-supervised neural network model. The reason autoencoders can be considered semi-supervised is that while they do not rely on explicit class labels during training, they can effectively utilise partially labeled data by learning the underlying structure and patterns in the data, which can enhance the performance of downstream classification tasks.

Typically, autoencoders (AEs) are used to reconstruct their input using partial information available from a bottleneck layer, which generates a latent space. In the case of autoencoders, the compression is achieved by training the network to learn the optimal representation that minimises the reconstruction error between the input and output at the bottleneck layer. The existence of a bottleneck layer, with a smaller number of neuron units compared to the input size, enables the projection of data from a higher dimension to a lower dimension through nonlinear transformations while preserving the most relevant features according to the cost function's emphasis.

Autoencoders consist of two main stages: the encoder, which produces the representation space, and the decoder, which reconstructs the original input

by minimising the representation error. AEs find applications in various tasks, such as learning and selecting features from partially labeled data in classification problems, noise reduction (HWAIDI; CHEN, 2022), dimensionality reduction (KAMAL; BAE, 2022), and anomaly detection (ZHOU; PAFFENROTH, 2017).

In this chapter, we propose a gradient based neural training algorithm that can solve the dictionary learning and sparse coding problem using an autoencoder architecture. We provide theoretical and experimental evidences that these problems can be tackled by training autoencoders with specific characteristics. The primary objective of this section is to conduct a comprehensive review of pertinent literature and illustrate the proposed framework.

The proposed loss function incorporates three penalty terms: one to promote sparsity in the latent space, one to penalise representation error, and one to encourage the unity norm of the dictionary atoms. The autoencoder-based framework can accommodate a wide range of nonlinear activation functions, resulting in greater flexibility and improved performance compared to conventional dictionary learning approaches described in Chapter 2. Moreover, the sparse autoencoder proposed in this chapter can be a promising method for large dataset, where conventional dictionary learning and sparse coding algorithms may be challenging to apply.

In addition to the proposed autoencoder architecture we also incorporate the Kullback-Leibler (KL) divergence as a penalty term in the loss function used throughout the training stage. The proposed output layer employs a linear activation function (specifically, the identity function) with the weight matrix being associated to the dictionary, $W_2 = \mathsf{D}^\mathsf{T}$. Additionally, to keep consistency with the original problem formulation, no bias vector is added at this decoder side.

3.1 Sparse Dictionary Learning: A Fully-Connected Autoencoder Approach

An autoencoder is a neural network introduced in (RUMELHART *et al.*, 1986). It maps $\mathcal{R}^n \rightarrow \mathcal{R}^n$ and consists of an encoder and a decoder model. The encoder network typically consists of one or more layers that compress the input data into a latent space with a reduced dimension. The decoder network, on the other hand, usually consists of one or more layers for reconstructing the input back to the original dimensions of the input. The characteristics of the latent space can also be controlled by incorporating regularisation terms into the loss function applied for model training.

Autoencoders are designed to learn efficient representations of data from a reduced-dimensional representation called the latent space. Typically, autoencoders have a bottleneck layer in the latent space, which imposes a constraint on the amount of information that can flow through the model.

However, in the context of sparse dictionary learning, we propose a novel approach that deviates from the conventional autoencoder architecture. Our model architecture involves a higher-dimensional space in the latent space, allowing for a richer representation of the input data. To encourage sparsity in this expanded latent space (NG, 2011), we add a sparsity penalty term to the loss function of the proposed model.

In this section, we delve into the investigation of fully-connected sparse autoencoders for the task of image reconstruction. By leveraging the benefits of a higher-dimensional latent space and incorporating a sparsity penalty, we aim to enhance the model's ability to capture features and reconstruct images with improved fidelity while also exploring a new method to learn sparse codes and dictionaries.

The experimental analysis will focus on evaluating the performance of

the proposed sparse autoencoder model using benchmark datasets. By exploring this alternative approach to autoencoder design, we aim to contribute to the advancements in sparse dictionary learning using deep learning models.

The training criterion utilised in sparse autoencoders (GOODFELLOW *et al.*, 2016) encompasses two essential components: the penalty denoted as $\mathsf{L}_\epsilon()$ to account for reconstruction error and a sparsity penalty referred to as $\mathsf{P}_s()$ targeting the output of the latent space, i.e.,

$$\mathsf{J} = \mathsf{L}_\epsilon\left(\mathbf{y} - \mathsf{g}(\mathsf{f}(\mathbf{y}))\right) + \mathsf{P}_s\left(\mathsf{f}(\mathbf{y})\right) \tag{3.1}$$

where $\mathbf{h} = \mathsf{f}(\mathbf{y})$ is the latent space, $\mathsf{g}()$ is the decoder function and $\mathsf{f}()$ is the encoder function.

According to (MAKHZANI; FREY, 2013), the principle of the sparse autoencoders can be summarised as follows

- An autoencoder with linear activation function, where in hidden layers only the k highest activities are kept;
- sparse autoencoders can enforce exact or arbitrary sparsity levels in the hidden layers;
- sparse autoencoders are suitable for pre-training deep discriminative neural networks;
- sparse autoencoders may constitute a sparse coding method that is well-suited to large problem sizes.

To bridge an equivalence between the original dictionary learning problem and an AE training scheme, we propose the input data, denoted as Y, undergoes the sparse coding step through a multi-layer encoder stage. Without loss of generality, let us consider a single-layer encoder composed of a weight matrix, denoted as W_1, a bias vector $\mathbf{b}_1$, and a nonlinear activation function, denoted as $\mathsf{f}_1()$, governing the behavior of the hidden layer neurons.

In the loss function we must induce sparsity at the latent space, denoted as $\mathbf{x}$. To achieve this, we incorporate Kullback-Leibler divergence as a sparsity regularisation term in the network's loss function. Subsequently, these sparse representation is transformed back to their original representation using a linear activation function and the weight matrix W_2 at the decoder side of the model, which represents the dictionary's transpose, i.e., $W_2 = D^T$.

The proposed architecture employs the sigmoid function as the activation applied to the encoder model, which naturally constrains the latent space to the interval $\sigma \in [0, 1]$. The decode layer employs a linear identity activation function. The autoencoder structure is illustrated in Figures 3.1, 3.2 and 3.3. Formally, the hidden layer and the output layer can be defined as follows

$$\mathbf{x}_i = \sigma\left(W_1^T \mathbf{y}_i + \mathbf{b}_1\right) \tag{3.2}$$

$$\hat{\mathbf{y}} = W_2^T \mathbf{x}_i \tag{3.3}$$

$$\sigma(x) = \frac{1}{1 + \exp^{-x}} \tag{3.4}$$

The vector $\mathbf{y}_i \in \mathcal{R}^m$ is an input sample, $W_1 \in \mathcal{R}^{m \times n}$ is the linear transformation implemented at the first layer of the encoder, $\mathbf{x}_i \in \mathcal{R}^n$ is the latent space obtained after the nonlinear activation function, $\mathbf{b}_1 \in \mathcal{R}^n$ is the bias vector at the encoder side, $W_2 \in \mathcal{R}^{n \times m}$ is the linear transformation applied at the decoder side, and $\hat{\mathbf{y}}_i \in \mathcal{R}^m$ is the output of the autoencoder.

Our loss function is defined as follows

$$J = \frac{1}{2p} \sum_{i=1}^{p} \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|_2^2 + \frac{\alpha}{n} \sum_{i=1}^{n} \left\|1 - \mathbf{d}_i^T \mathbf{d}_i\right\|_1 + \frac{\beta}{n} \sum_{j=1}^{n} KL(\rho \| \hat{p}_j) \tag{3.5}$$

$$\hat{p}_j = \frac{1}{p} \sum_{i=1}^{p} h_j^{(L)}(\mathbf{x}_i) \tag{3.6}$$

$$KL(\rho \| \hat{p}_j) = \rho \log \frac{\rho}{\hat{p}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{p}_j} \tag{3.7}$$

where $\mathbf{d}_i$ is the i-th dictionary atom and $h_j^{(L)}$ is the j-th neuron of L-th layer.

Our objective is to minimise the loss function J with respect to the variables W_1, W_2, and b_1. It is important to note that the weight matrix on the decoder side, denoted as W_2, is directly related to the dictionary, which can be expressed as $W_2 = D^T$. Additionally, the vector $\mathbf{x}_i$ represents the sparse code of the i-th input sample and corresponds to the latent space of the autoencoder.

The first term in the loss function aims to penalise the representation error between the inputs and outputs of the autoencoder by utilising the ℓ_2-norm regularisation. The second term enforces the unit ℓ_2-norm constraint on the dictionary atoms. This constraint prevents situations where dictionary atoms have arbitrarily large norms while sparse codes falsely exhibit small values not related to sparsity. Finally, the third term encourages sparsity in the codes represented by $\mathbf{h}(x)$.

The penalty function imposes a condition such that $KL(\rho||\hat{p}_j) = 0$ when $\hat{p}_j = \rho$. On the other hand, the penalty increases monotonically as $\hat{p}_j$ diverges from ρ. The goal is to ensure that the average activation of the encoder's hidden neuron j closely approximates the desired value of ρ. Specifically, we aim to set ρ to be approximately equal to zero. Figure 3.4 illustrates the behavior of $KL(\rho I|\hat{p}_j)$ for a specific value $\rho = 0.2$.

The sparsity level achieved at the latent space is defined in terms of the Kullback-Leibler sparsity parameter ρ as follows

$$s_i = \frac{\#\left\{j \mid \left\|x_i^{(j)}\right\|_1 < \rho\right\}}{n} \tag{3.8}$$

and the density level is

$$d_i = 1 - \frac{\#\left\{j \mid \left\|x_i^{(j)}\right\|_1 < \rho\right\}}{n} \tag{3.9}$$

The sparsity parameter must be kept close to zero. In particular, we adopt $\rho = 0.01$ to encourage sparsity. Figures 3.5 and 3.6 illustrate the probability distribution function of the values in the latent space, $\mathbf{h}$, and the probability

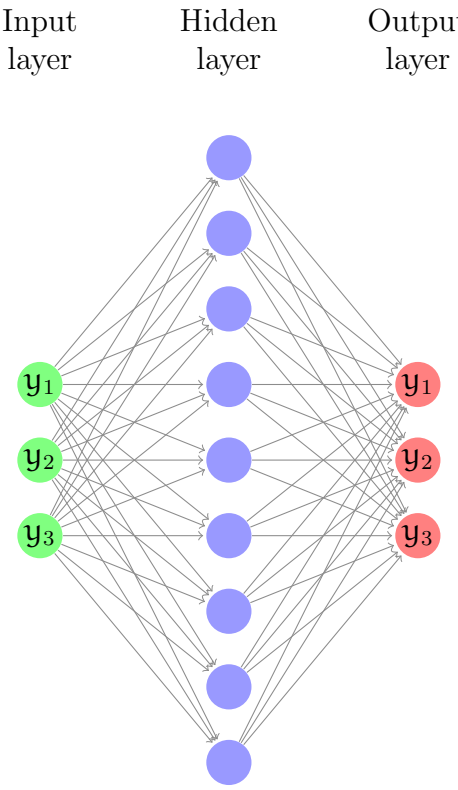


Figure 3.1 – The proposed autoencoders can be easily modified and extended to incorporate various architectural enhancements, regularisation techniques, and loss functions.

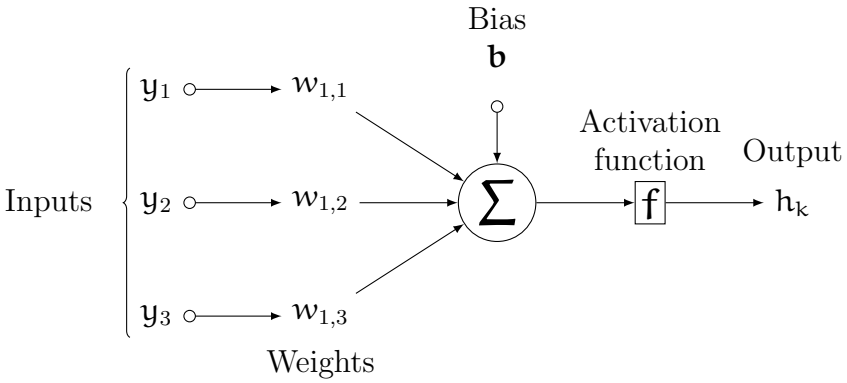


Figure 3.2 – Encoder layer configuration.

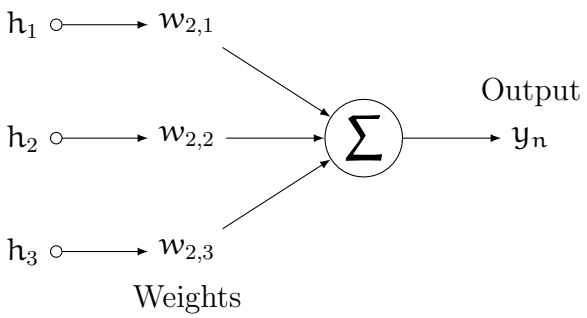


Figure 3.3 – Decoder layer configuration.

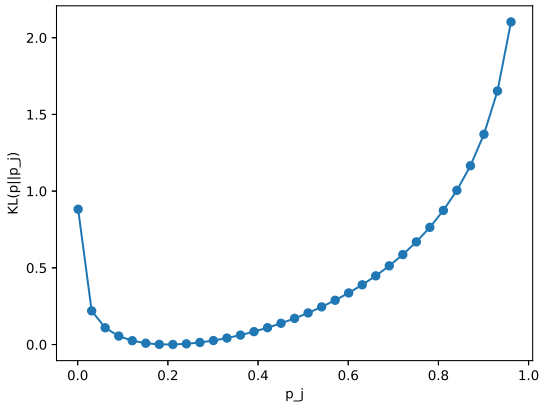


Figure 3.4 – Kullback-Leibler divergence achieves its minimum at $\hat{\rho}_j = \rho$ and blows up as $\hat{\rho}_j$ approaches 0 or 1. In this example $\rho = 0.2$.

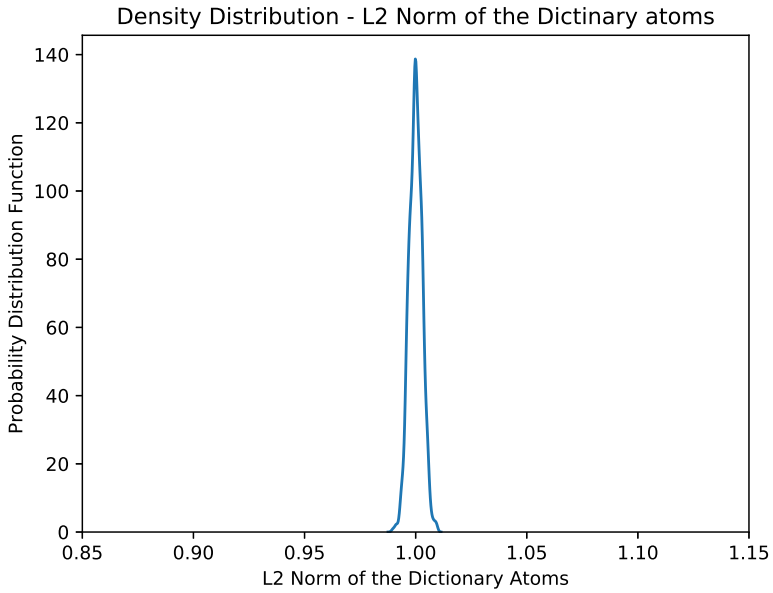


Figure 3.5 – The probability density function of the sparse codes $\mathbf{h}$.

distribution function of the ℓ_2-norm of $\mathbf{d}_i$, respectively. The values in $\mathbf{h}$ are concentrated around ρ, and $\mathbf{d}_i^\top \mathbf{d}_i$ values are also concentrated around 1, indicating that most dictionary atoms have a unit norm.

In this first example of the proposed autoencoder-based approach, we analyse the results obtained with the CIFAR-10 images. This dataset comprises a collection of $60,000$ color images, each with dimensions of 32×32 pixels, distributed among 10 distinct and mutually exclusive classes, with $6,000$ images

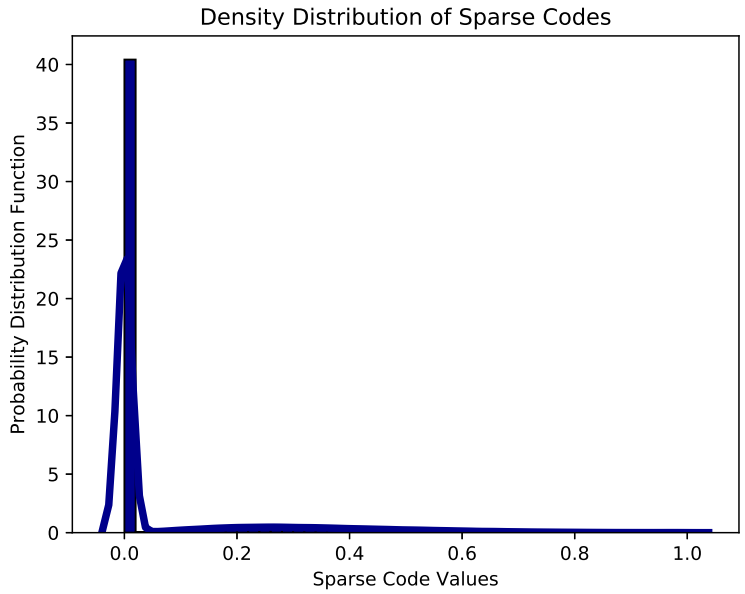


Figure 3.6 – The probability density function the ℓ_2-norm of the dictionary atoms $\mathbf{d}_i$.

per class. The dataset is divided into a training set, which consists of $50,000$ images, and a test set, which contains $10,000$ images. The dataset's classes are entirely exclusive, with no overlap between the automobile and truck categories. The automobile class encompasses sedans, sport utility vehicles, and similar vehicles, while the truck class comprises only large trucks, excluding pickup trucks.

Figure 3.7 illustrates a set of images used for testing the trained autoencoder as well as the results obtained by reconstruction using $\hat{\mathsf{Y}} = \mathsf{DX}$. The performed simulation is implemented in Python using TensorFlow (ABADI *et al.*, 2015) library, version 1.13. The experiments were performed in a notebook with Intel Core 8xI7-4720HQ Processor, 16GB RAM and a GPU GTX 970M.

To convert RGB images from CIFAR-10 to grayscale, thus decreasing the number of connections between the neurons of the autoencoder, we take the RGB values for each pixel and make as output a single value reflecting the brightness of that pixel. For that end we take the weighted average of each channel pixel $(\mathsf{R}_i, \mathsf{G}_i, \mathsf{B}_i)$ in the following way $\mathsf{p}_i = 0.3\mathsf{R}_i + 0.59\mathsf{G}_i + 0.11\mathsf{B}_i$.

(a) Original Images

(b) Sparse reconstruction

Figure 3.7 – The reconstruction of a selected subset of images from test dataset. The average density level is $\overline{\mathrm{d}} = 0.19$ indicating that approximately 19% of the 2500 atoms are linearly combined to reconstruct these images.

Converting a three-channel image into its grayscale representation leads to a reduction in the required model's size. The images are downgraded from three channels to one, reducing by three times the number of parameters to be trained. From the classification perspective, grayscale preprocessing is particularly effective in datasets where the semantic meaning of object color is minimal. However, it should be noted that this conversion is not considered beneficial for classifying nature images such as those found in CIFAR-10.

For instance, when considering the ship class, ships are typically expected to appear against blue or green backgrounds rather than red backgrounds, as water is predominantly blue or green. Therefore, in the context of classification, grayscale preprocessing is not the most effective technique for CIFAR-10 images. However, it is worth noting that the autoencoder-based framework can be successfully applied to RGB images by stacking the three color channels into a single input vector, denoted as $\mathbf{y}_{\mathfrak{i}} \in \mathcal{R}^{3\dot{m}}$.

The preprocessing stage is completed by normalising the pixel values, which involves rescaling the intensity range from $[0, 255]$ to $[0, 1]$. This normalisation is achieved by dividing each pixel value by 255.

To optimise all the parameters of the autoencoder model, we employ the Adam (KINGMA; BA, 2014) algorithm, derived from the concept of *adaptive moment estimation*. This algorithm computes individual adaptive learning rates for different parameters based on estimates of the first and second moments of the gradients. The Adam algorithm utilises the following configuration parameters:

- α: also referred to as the learning rate or step size. Larger values results in faster initial learning before the rate is updated. Smaller values results in slow learning right down during training;
- β_1: the exponential decay rate for the first moment estimates;

Table 3.1 – Detailed configuration used to train the autoencoder.

Attribute	Used Value
Activation Function	Sigmoid
Batch Size	400
Loss Function	J
Optimiser	Adam
Epochs	250
Learning Rate	0.001
Initialiser function $(W_1, W_2, \mathbf{b}_1)$	$\mathcal{N}(0, 0.3^2)$
ρ sparsity parameter from KL	0.01
$\mathfrak{m}$ - figure size	$1024(32 \times 32)$
$\mathfrak{p}$ - number of training samples	50000
$\mathfrak{n}$ - number of dictionary atoms	2500
α hyperparameter from J	250
β hyperparameter from J	300

- β_2: the exponential decay rate for the second-moment estimates;
- ϵ: it is a very small number to prevent any division by zero in the implementation.

In the experiments reported next, we use the default parameters suggested in (KINGMA; BA, 2014), i.e., $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. We remark that tuning the learning rate parameter α with a learning rate scheduler to adjust step size throughout the training epochs has not achieved good results and due to that, we kept it fixed. Table 3.1 summarises the attributes used for training autoencoder, and Figure 3.8 illustrates the loss function J throughout the training.

The primary aim of the initial phase of our simulation is to evaluate the autoencoder's potential to achieve significant sparsity levels while minimising representation errors. In this phase, we focus on the autoencoder's sparse representation capabilities, not focusing on its denoising capabilities widely explored in the literature (VINCENT *et al.*, 2008). Therefore, we refrain from introducing stochastic noise to the autoencoder inputs at this point. The results of this preliminary assessment are presented in Figure 3.9, which shows a subset

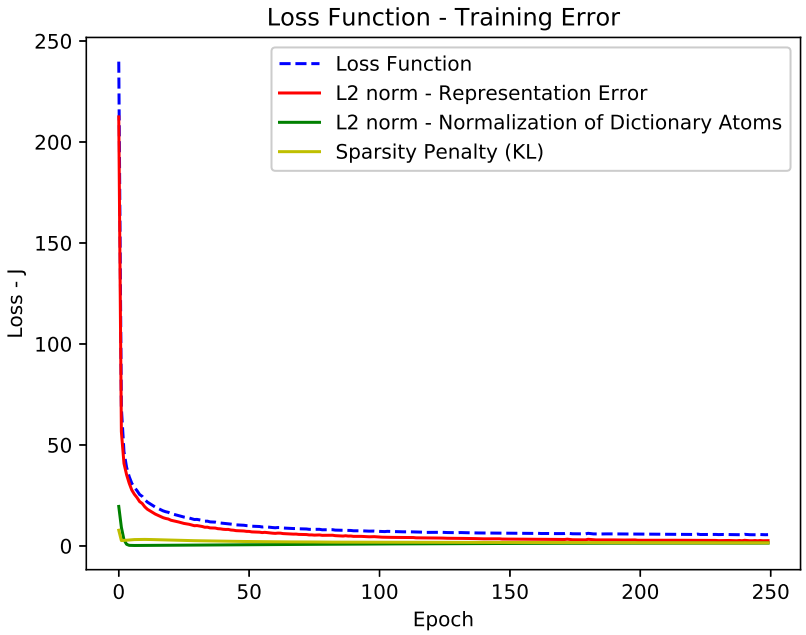


Figure 3.8 – The loss function J and the individual terms that account for sparsity penalty, unit ℓ_2-norm of the dictionary atoms, and the representation error throughout the training epochs. Results were compared for five different values of learning rate. Minimum loss value was achieved for learning rate $\alpha = 0.001$.

of the achieved results.

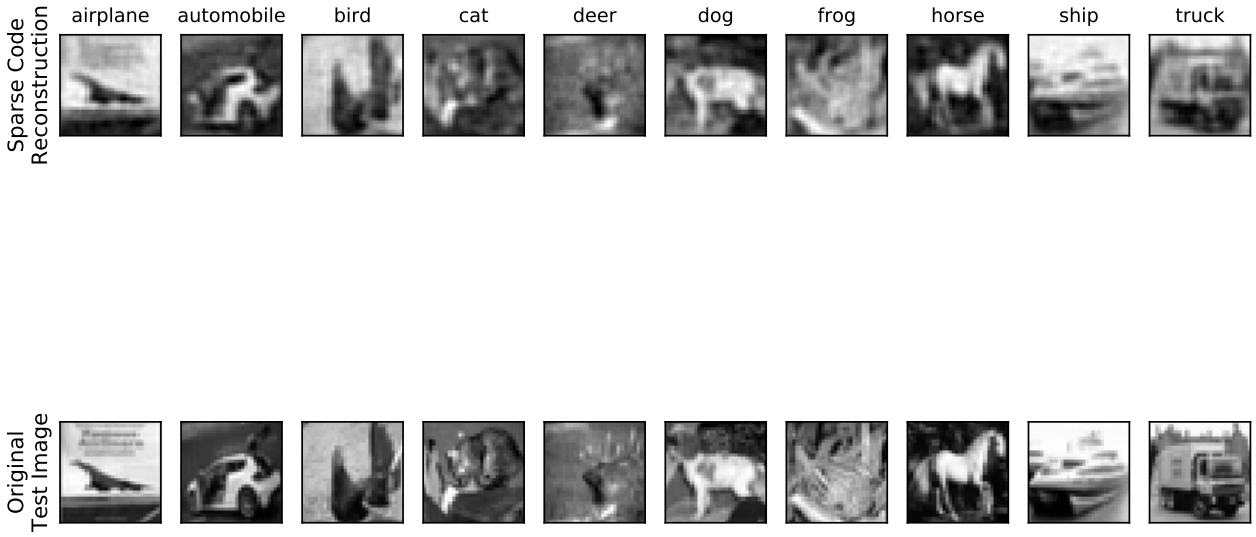


Figure 3.9 – Comparing the original set of testing images from CIFAR-10 dataset and the sparsely reconstructed images $\hat{\mathsf{Y}} = \mathsf{DX}$. The average density level is $\overline{\mathrm{d}} = 0.19$.

Figure 3.10 illustrates the most activated atoms of each class in CIFAR-10 dataset. These atoms are represented as images where the grayscale tones are associated to the element values of each atom. This representation allows us to visually inspect and interpret the atoms. The highly activated atoms represent the dominant features that the autoencoder has learned to extract among different samples used in the training. By examining these atoms, we cannot see highly discriminative characteristics that the autoencoder has identified for each class. In fact, this model was not trained to capture these discriminative features and therefore we cannot clearly identify distinguishable patterns among the activated atoms.

Interestingly, when examining the most frequent atom for the images in each class (except the ship class), we observe that they are the same. This suggests the presence of a shared common pattern that is present in the majority of samples across different classes. This finding highlights the presence of a recurring visual characteristic that is consistent across various objects, such as sedans, sport utility vehicles, trucks, and more, within the dataset.

Figure 3.11 depicts the relationship between sparsity, representation error, and a sparsity threshold parameter denoted by $\mathtt{t}$. The threshold parameter $\mathtt{t}$ is applied after to the latent space values. In particular, if $x_{i,j}$ is less than $\mathtt{t}$, then $x_{i,j}$ is rounded to zero.

In Figure 3.12, the highlighted region indicates a lower level of sparsity (higher density) with a relatively constant level of error achieved for the representation. This region sets the best trade-off between sparsity and accuracy and therefore it is useful in selecting the best value of $\mathtt{t}$ for the autoencoder.

The proposed autoencoder model, designed for sparse dictionary learning, can also serve as an effective tool for noise removal. To this end, during its training the denoising autoencoder aims to reconstruct the original input data from its noisy versions. Once the training process is completed, the trained

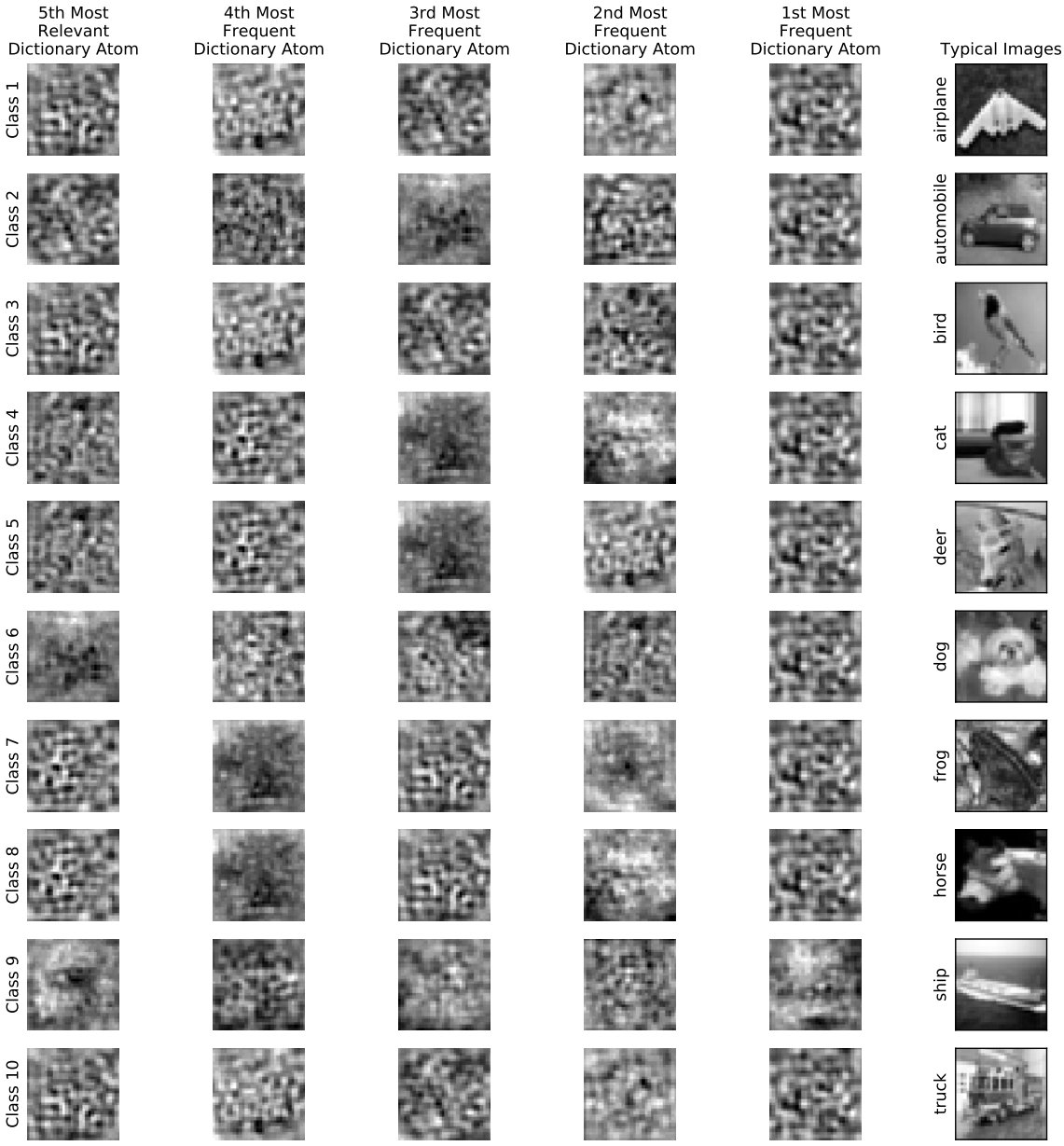


Figure 3.10 – Top 5 most activated dictionary atoms for each class of CIFAR-10 dataset.

denoising autoencoder demonstrates the ability to remove noise from unseen data.

To add Gaussian noise to the input images, we use the following steps:

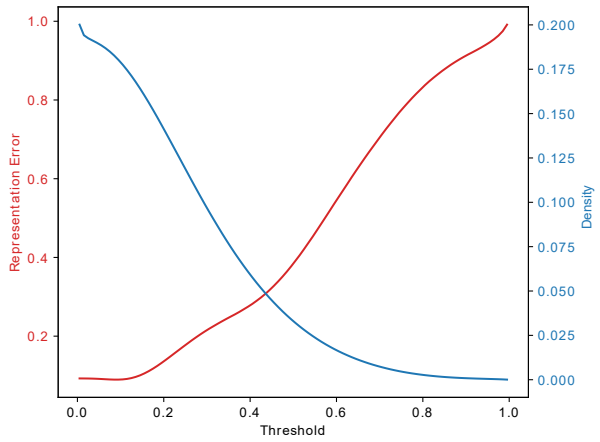


Figure 3.11 – The figure illustrates the relationship between thresholds, densities of sparse codes, and achieved representation error.

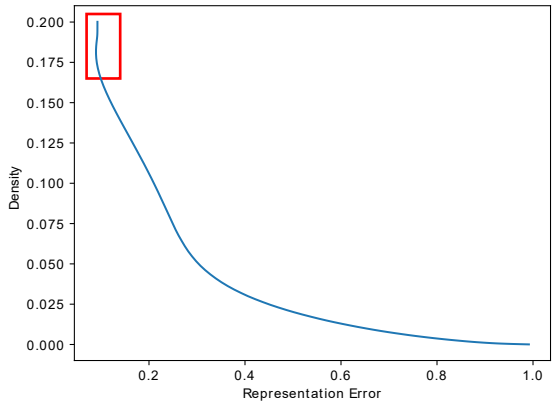


Figure 3.12 – The figure illustrates the relationship between thresholds, densities of sparse codes, and achieved representation error.

1. Normalise the pixel values of the input image to be between 0 and 1.
2. Generate a Gaussian noise matrix of the same shape as the input image. The values of the Gaussian noise matrix should have a mean of 0 and a standard deviation of a chosen level of noise.
3. Add the Gaussian noise matrix to the normalised input image.
4. Clip the pixel values of the noisy image to be between 0 and 1 to ensure that the image remains in the valid pixel range.

This approach adds a random noise signal to the input image, which can be useful for simulating real-world scenarios where images are often corrupted by noise. The amount of Gaussian noise added can be controlled by adjusting the standard deviation of the Gaussian noise matrix. Higher standard deviation results in noisier images. For this experiment, the noise matrix Q is added to a grayscale image $E \in \mathcal{R}^{\sqrt{m}\times\sqrt{m}},\ e_{i,j} \in [0,1]$ from CIFAR-10 as follows

$$W = \mathcal{N}(0^{\sqrt{m}\times\sqrt{m}},\ \mathrm{diag}(\sigma)^{\sqrt{m}\times\sqrt{m}}) \tag{3.10}$$

$$E_q = E + Q \tag{3.11}$$

$$E_c = \max(0, \min(1, E_q)) \tag{3.12}$$

The peak signal-to-noise ratio (PSNR) is a metric used to assess the quality of a reconstructed image in comparison to the original image. It quantifies the ratio of the peak signal power to the mean squared error (MSE) between the two images. A higher PSNR value indicates better image quality. PSNR is commonly expressed in decibels (dB).

The MSE is a metric commonly used to assess image quality. It calculates the average squared error between the reconstructed and original images. In contrast, the PSNR measures the peak error between the two images. A lower MSE value indicates a lower overall error, while a higher PSNR value indicates lower peak errors. The PSNR is computed by utilising the MSE, which is defined according to the following equation:

$$MSE = \frac{\sum_{i=1}^{m}\sum_{j=1}^{p}\left[y_{i,j} - \hat{y}_{i,j}\right]^2}{m \cdot p} \tag{3.13}$$

where m is the number of pixels from the input images y_i, and p is the total number of samples under evaluation. The PSNR is given by

$$PSNR = 10\log_{10}\left(\frac{R^2}{MSE}\right) \tag{3.14}$$

where R is the maximum fluctuation in the input image data type. Particularly, the input images used throughout this section have a single-precision floating-point data type and R is 1.

The classification accuracy obtained from the ResNet-56 architecture will be also utilised as a third criterion to evaluate the image quality achieved by the proposed sparse dictionary learning framework. ResNet, known as the Residual Network, has demonstrated outstanding performance in various computer vision tasks, including image classification. While classification is not the primary objective in assessing image reconstruction quality with sparse codes, it can serve as an auxiliary metric of success. ResNet allows us to compare the classification performance between the original image samples and the sparse reconstruction achieved from the autoencoder.

ResNet-56, a specific variant with 56 layers, was employed in this study. The obtained classification results from ResNet-56 will be compared against the benchmark accuracy achieved with the original image samples.

Table 3.2 presents the accuracy results obtained for Y_c and $\hat{Y}$. Here, Y represents the original image, Y_c represents the noisy version of it, and $\hat{Y}$ represents the sparse reconstruction of Y. The compression rate C is defined based on the density d, the number of atoms n, and the stacked figure size, $f_{size} = 32 \times 32 = 1024$. The compression rate is given as follows

$$C = \frac{d \times n}{f_{size}} \tag{3.15}$$

To improve the statistical significance to a certain extent, we performed 10 independent runs and calculated the average results. We systematically varied the values of σ and t to examine the impact of noise on classification accuracy. The outcomes of these experiments are summarised in Table 3.2. We explored different values of σ ranging from 0 to 0.15 and adjusted the threshold parameter t from $t = \rho = 0.01$ to $t = 0.42$, which corresponds to the elbow point of the density-representation error curve depicted in Figure 3.11. After training for 150 epochs, the ResNet-56 model achieved a reference accuracy of 91.13% on the validation dataset when $\sigma = 0$ (i.e., $Y_c = Y_w = Y$). However, the accuracy associated with Y_c is highly dependent on the noise level controlled by σ.

Table 3.2 – ResNet-56 v2 classification results. Red highlights indicate accuracies surpassing the reference value of noisy test images (Y_c). Blue highlight represents benchmark accuracy without additional noise corruption ($Y_c = Y,\ \sigma = 0$).

σ	PSNR [dB]	t	Density	C	Accuracy $\hat{Y}$	Accuracy Y_c
0	25.84	0.01	0.1939	0.4734	60.19%	91.13%
	26.04	0.12	0.1751	0.4275	60.90%	
	14.83	0.42	0.0579	0.1414	29.58%	
0.05	25.00	0.01	0.1907	0.4656	49.71%	43.43%
	25.14	0.12	0.1700	0.4150	50.38%	
	15.37	0.42	0.0544	0.1328	25.66%	
0.1	23.60	0.01	0.1857	0.4534	35.78%	23.45%
	23.61	0.12	0.1614	0.3940	36.21%	
	15.11	0.42	0.0485	0.1184	21.53%	
0.15	22.2947	0.01	0.1697	0.4143	24.18%	15.50%
	22.3071	0.12	0.1470	0.3589	23.98%	
	15.4884	0.42	0.049	0.1196	16.92%	

In contrast, although the accuracy of the reconstructed images $\hat{Y}$ at $\sigma = 0$ is lower than that of Y_c at $\sigma = 0$, $\hat{Y}$ exhibits higher accuracy compared to Y_c for $\sigma > 0$. This also indicates the potential of the sparse dictionary learning framework on achieving improved accuracy and enhanced robustness against Gaussian noise.

In our experiment, we also investigated the use of different patch sizes by dividing the images into smaller non-overlapping patches. Specifically, we considered patch sizes of 16×16, 8×8, and 4×4. The results are illustrated in Figure 3.14. Detailed results are presented in Table 3.3.

To this experiment, the compression rate, denoted as C, is determined by considering the patch size r, the density d, and the number of atoms n. The equation defining the compression rate is given by

$$C = \frac{dn}{r} \tag{3.16}$$

Additionally, the density threshold at which the compression rate equals 1 can be calculated by dividing the patch size by the number of dictionary atoms, which we set to 2500. In our analyses, if the density exceeds this threshold, it

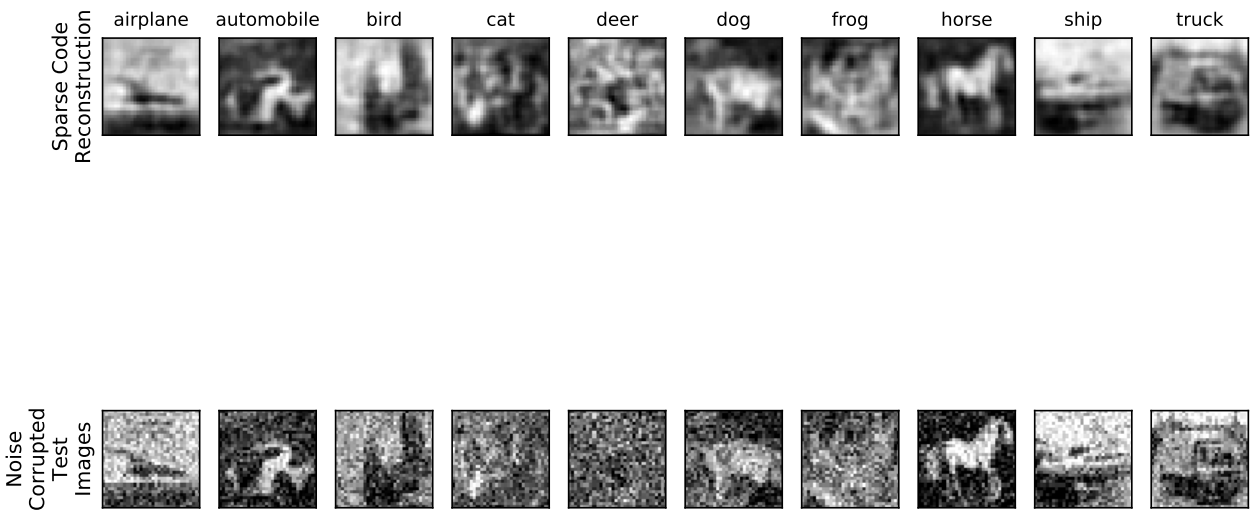


Figure 3.13 – Comparison between grayscale test images from CIFAR-10 dataset after white Gaussian noise with $\sigma = 0.15$, and the sparse reconstruction $\hat{\mathsf{Y}} = \mathsf{DX}$. The average density level is $\overline{\mathsf{d}} = 0.1746$.

implies that no compression is achieved.

Decreasing the patch size leads to an increase in the required sparsity for compression. This means that smaller patches require a higher level of sparsity in order to achieve efficient compression. Additionally, reducing the patch size results in lower accuracy under compression conditions. This indicates that smaller patches are more challenging to accurately reconstruct.

Moreover, the limitations of fully-connected networks in image reconstruction tasks are evident from the experiment, highlighting the need for alternative approaches such as convolutional networks. Fully-connected networks have drawbacks such as a large number of parameters, making them computationally expensive and challenging to train. They also lack the ability to capture spatial information, disregarding the important relationships between pixels necessary for accurate image reconstruction. Additionally, fully-connected networks are prone to overfitting, resulting in poor generalisation performance.

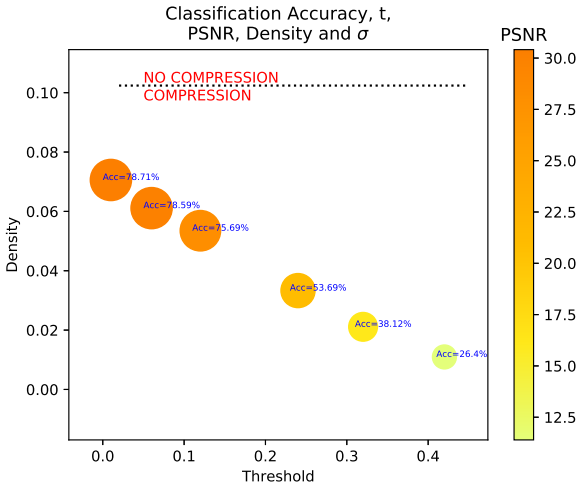
Table 3.3 – ResNet-56 v2 classification accuracy with diffrent image patch sizes. Red values exceed reference from noise-corrupted images (Y_c), while blue value is benchmark accuracy without additional noise ($Y_c = Y$, $\sigma = 0$).

r	**PSNR [dB]**	**t**	**Density**	C	**Accuracy $\hat{Y}$**	**Accuracy Y_c**
16	30.52	0.01	0.0711	0.6943	78.71%	91.13%
	27.22	0.06	0.0611	0.5967	78.59%	
	25.77	0.12	0.0536	0.5234	75.69%	
	20.49	0.24	0.0334	0.3262	53.69%	
	16.61	0.32	0.0209	0.2041	38.12%	
	11.10	0.42	0.0108	0.1055	26.40%	
8	36.78	0.01	0.1006	3.9297	87.97%	91.13%
	24.52	0.06	0.0326	1.2734	68.27%	
	22.34	0.12	0.0241	0.9414	60.74%	
	18.87	0.24	0.0179	0.6992	50.76%	
	15.71	0.32	0.0103	0.4023	35.95%	
	10.08	0.42	0.0037	0.1445	23.27%	
4	23.87	0.01	0.1013	15.8281	78.44%	91.13%
	15.12	0.06	0.0301	4.7344	50.19%	
	13.77	0.12	0.0206	3.2187	24.17%	
	12.94	0.24	0.0128	2.0000	21.94%	
	11.87	0.32	0.0088	1.3750	18.04%	
	9.17	0.42	0.0047	0.7344	14.30%	

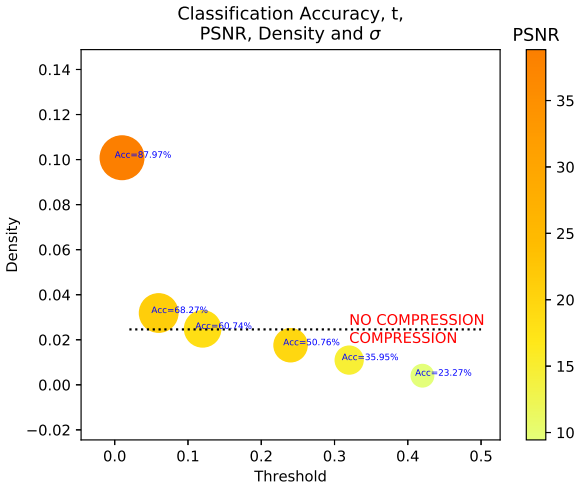
To address these limitations, we also investigate the convolutional neural networks (CNNs), which typically offer enhanced performance in image processing tasks, as they can capture spatial information using convolutional filters. Therefore, exploring convolutional autoencoders for sparse dictionary learning holds potential for improving results in the image reconstruction set. This prospect naturally motivates us to a deeper investigation in this set, which is conducted in the next session.

3.2 Convolutional Sparse Autoencoder for Dictionary Learning and Sparse Coding

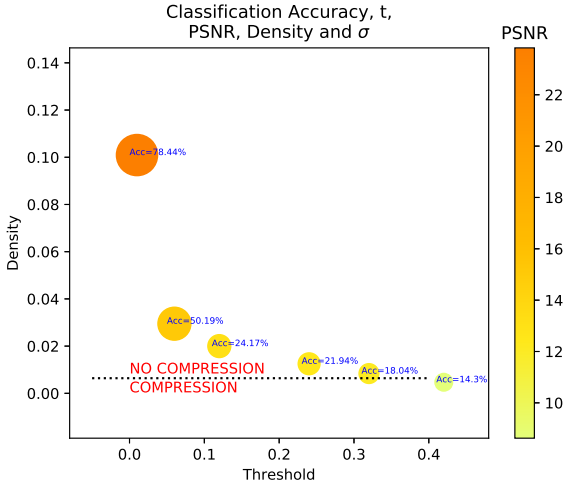
Convolutional Neural Networks (CNNs) are a special type of deep neural network that perform particularly well in computer vision problems such as image classification and object detection (LECUN *et al.*, 2015) (HE *et al.*, 2017).



(a) Patch Size = 16.



(b) Patch Size = 8.



(c) Patch Size = 4.

Figure 3.14 – Results obtained in terms of the classification accuracy, the threshold $\mathtt{t}$, PSNR and Density. The training was performed for no noise condition only, i.e., $\sigma = 0$.

In Section 3.1, it was demonstrated that the fully-connected encoder within the autoencoder-based sparse dictionary learning framework exhibits certain limitations. One of the drawbacks of a fully-connected encoder is the loss of spatial information in the encoding process, as it involves flattening and stacking the input data. These limitations motivate the exploration of the convolutional encoder architecture.

Convolutional layers are capable of preserving the spatial characteristics of input images. This enables the extraction of relevant information th-

rough local connectivity (GOODFELLOW *et al.*, 2016), offering the potential for enhanced performance in the sparsification task. Accordingly, we will thoroughly investigate the utilisation of a convolutional encoder while maintaining the integrity of the traditional sparse dictionary learning formulation. In order to achieve this objective, the decoder architecture will continue to feature a single fully-connected layer with no activation function. Consequently, the dictionary will be represented by the transpose of the decoder's weight matrix, i.e., $\mathsf{D} = W_2^{\mathsf{T}}$.

The convolutional encoder significantly reduces computational operations by leveraging convolution on neighboring pixel patches, capitalising on the meaningful information and salient features present in adjacent pixels (GOODFELLOW *et al.*, 2016). Additionally, pooling layers can be employed to downscale the image, as the spatially organised features throughout the network align with the image structure. In contrast, downsampling vectors in traditional 1D stacked inputs is not feasible due to the absence of spatial coherence between consecutive elements.

The encoder models employed in this section are illustrated in Figure 3.15. The depicted convolutional blocks encompass three sequential operations: 2D convolutions, batch normalisation, and ReLU activation function. To downsample the data passing through the various encoder schemes, we utilised strided convolutions with a stride value of $\mathsf{s} = 2$.

The selection of network architectures is driven by the characteristics of the data under consideration. For example, in Section 3.1, the training datasets consisted of different sizes of image patches: $50,000$ patches of size 32×32, $200,000$ patches of size 16×16, $800,000$ patches of size 8×8, and $3,200,000$ patches of size 4×4. These patches were stacked into vectors $\mathbf{y}_\mathsf{i} \in \mathbb{R}^{1024}$, $\mathbf{y}_\mathsf{i} \in \mathbb{R}^{256}$, $\mathbf{y}_\mathsf{i} \in \mathbb{R}^{64}$, and $\mathbf{y}_\mathsf{i} \in \mathbb{R}^{16}$, respectively. However, since the convolutional encoder performs convolutions and downsampling operations extracting smaller image

patches from the original images, and considering that the original image sizes are small, computational intensity is not a concern. Thus, in this experiment, the sample tensors $\mathbf{y}_i$ will always correspond to the i-th non-stacked full image, i.e., $\mathbf{y}_i \in \mathbb{R}^{32\times32}$.

In order to evaluate the sparse convolutional architecture, gray-scale images were utilised. The objective was to explore the trade-off between sparsity and the quality of representation achieved by the dictionary and sparse codes. Four different autoencoder models, as illustrated in Figure 3.15, were trained at varying levels of noise corruption: $\sigma = 0$, $\sigma = 0.05$, $\sigma = 0.1$, and $\sigma = 0.15$. To compare the image reconstruction performance of each model, classification accuracy was assessed using the same ResNet-56 configuration as described in Section 3.1. It is worth noting that ResNet-56 was trained using the original CIFAR-10 images without additive Gaussian noise ($\sigma = 0$). To convert the RGB CIFAR-10 images to grayscale, the pixel values p_i were calculated as the weighted average of each channel pixel (R_i, G_i, B_i) using the formula: $p_i = 0.3R_i + 0.59G_i + 0.11B_i$.

The CIFAR-10 dataset consists of images that are relatively small in comparison to modern photographs. Due to the extremely low resolution, it can be challenging to discern and differentiate the depicted content accurately. This limited resolution is likely one of the primary factors contributing to the modest performance achieved by state-of-the-art algorithms on this dataset. Moreover, there exists a considerable variation in viewpoints, poses, and object localisation within the images. To investigate potential convolutional encoder architectures from scratch, systematic tests were conducted during the model design phase. These tests encompassed exploring suitable numbers of layers, kernel sizes, activation functions, optimisers, learning rates, and other relevant training parameters. Notably, the following observations were made during this phase:

1. Excessive reduction of the image size along the encoder side deteriorates performance, regardless the adopted pooling method.

2. Strided 2D convolutions yield superior results compared to a combination of 2D convolutions followed by a Max Pooling layer.

3. The inclusion of batch normalisation layers significantly enhances performance.

4. Shallow convolutional encoders outperform deep convolutional encoders (i.e., with more than three convolutional layers).

The encoder architectures proposed in this section consist of one to four convolutional layers, each equipped with ReLU activation functions. To convert the output of the last convolutional layer into a 1D feature vector, we incorporate a flatten layer. This feature vector is then connected to a dense layer with a sigmoid activation function, resulting in the sparse latent space $\mathbf{x}_i$. By utilising the sigmoid function, denoted as $\sigma()$, the values of $\mathbf{x}_i$ naturally fall within the interval $[0, 1]$. The decoder layer employs an identity activation function, utilising the weight matrix W_D without a bias vector. The latent space and output layer are defined as follows

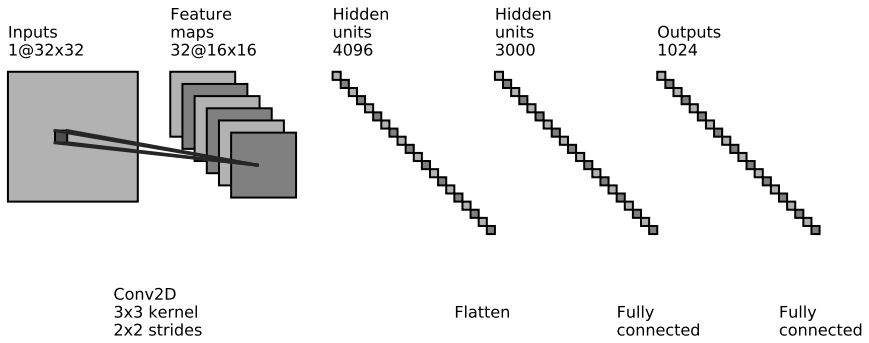
$$\mathbf{x}_i = \sigma\Big(W_e \cdot \mathrm{ReLU}\Big(W_L \circledast \cdots \Big(\mathrm{ReLU}(W_2 \circledast \mathrm{ReLU}(W_1 \circledast \mathbf{y}_i + \mathbf{b}_1) + \mathbf{b}_2)\Big) \cdots + \mathbf{b}_L\Big) + \mathbf{b}_e\Big) \tag{3.17}$$

$$\hat{\mathbf{y}} = W_D^T \mathbf{x}_i \tag{3.18}$$

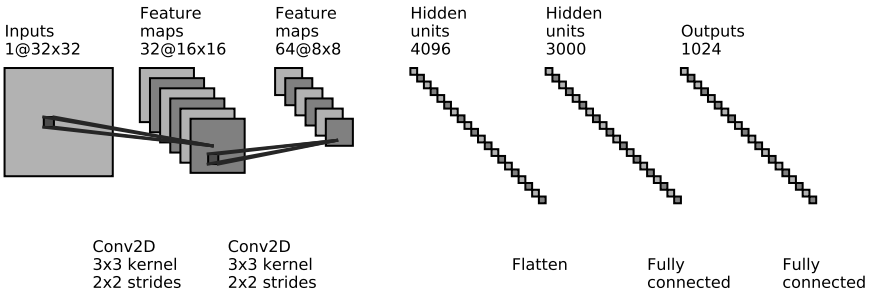
$$\sigma(x) = \frac{1}{1 + \exp^{-x}} \tag{3.19}$$

where $\circledast$ denotes the convolution operation.

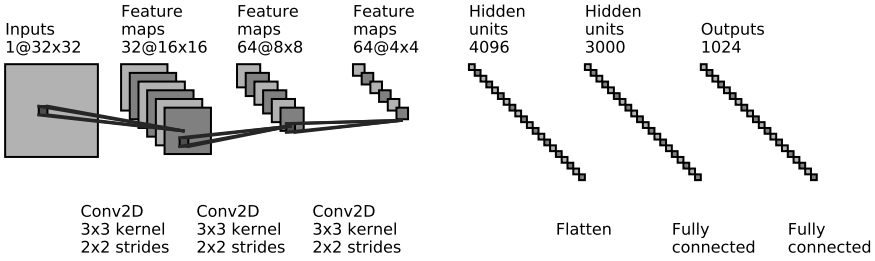
In this formulation, we have $\mathbf{y}_i \in \mathbb{R}^m$, where $m = 1024$, representing the fully stacked version of a single input image used for training the autoencoder. The weight tensors and bias vectors from the encoder's convolutional layers are denoted as $W_1, W_2, \ldots, W_L \in \mathbb{R}^{k \times k \times d \times f}$ and $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_L \in \mathbb{R}^f$, respectively, where k represents the kernel size, d denotes the depth size, and f represents



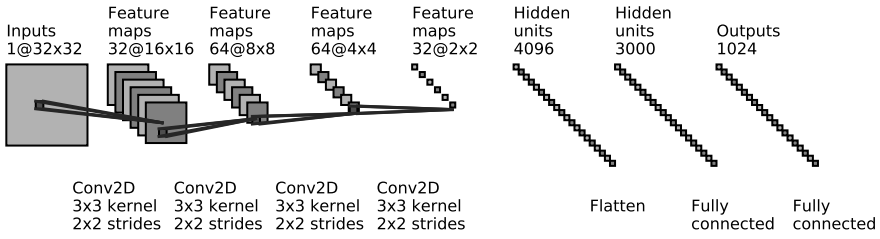
(a) Autoencoder model with a single convolutional layer at encoder side.



(b) Autoencoder model with two convolutional layers at encoder side.



(c) The autoencoder model achieved the best reconstruction performance by employing a three-layer convolutional encoder architecture.



(d) Autoencoder model with four convolutional layers at encoder side.

Figure 3.15 – Design of the autoencoders evaluated in the experiments.

the filter size. The fully-connected layer that connects $\mathbf{x}_i$ to the flattened version of the L-th convolutional layer is characterised by the weight matrix W_e and the bias vector $\mathbf{b}_e$. The decoder side employs the linear transformation given

by $W_D \in \mathbb{R}^{n \times m}$, and the reconstructed output of the autoencoder is represented by $\hat{\mathbf{y}}_i \in \mathbb{R}^m$. For simplicity, batch normalisation after each 2D convolution is not included in this formulation. However, this operation is included in the experiments discussed next.

The objective function that governs our optimisation problem is similar to that from Section 3.1. Recall that

$$J = \frac{1}{2p} \sum_{i=1}^{p} \|\mathbf{y}_i - \hat{\mathbf{y}}_i\|_2^2 + \frac{\alpha}{n} \sum_{i=1}^{n} \left\|1 - \mathbf{d}_i^T \mathbf{d}_i\right\|_1 + \frac{\beta}{n} \sum_{j=1}^{n} KL(\rho \| \hat{p}_j) \tag{3.20}$$

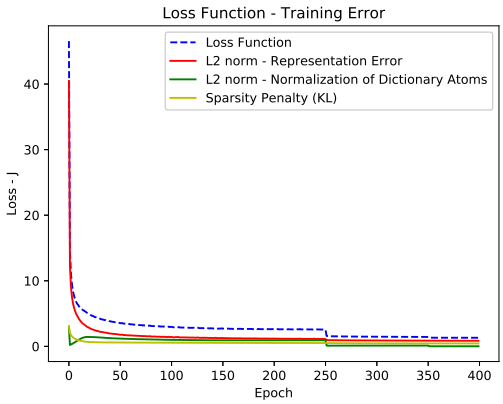
$$\hat{p}_j = \frac{1}{p} \sum_{i=1}^{p} h_j^{(L)}(\mathbf{x}_i) \tag{3.21}$$

$$KL(\rho \| \hat{p}_j) = \rho \log \frac{\rho}{\hat{p}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{p}_j} \tag{3.22}$$

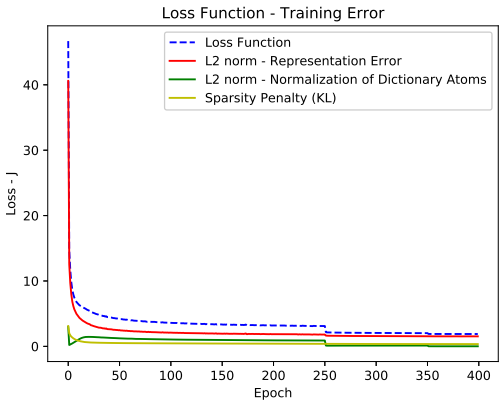
where $\mathbf{d}_i$ is the i-th dictionary atom and $h_j^{(L)}$ is the j-th neuron of L-th encoder layer. Our objective is to minimise J with respect to the weight matrices W_e and W_D, the convolutional filters $W_1, W_2, \cdots, W_L$ and bias vectors $\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_L, \mathbf{b}_e$. The first term imposes a penalty on the representation error between the inputs and outputs of the autoencoder, measured using the ℓ_2-norm. The second term promotes unit ℓ_2-norm for the dictionary atoms, preventing situations where atoms have arbitrarily large norms while inducing sparse codes to have small values. Lastly, the third term encourages sparsity at the latent space.

The autoencoder model, depicted in Figure 3.15(c), with three convolutional layers in the encoder achieved superior reconstruction performance. The losses achieved by this model at various noise levels are compared in Figure 3.16. Additionally, we illustrate a subset of the reconstructed images obtained from this model in Figures 3.17 and 3.18.

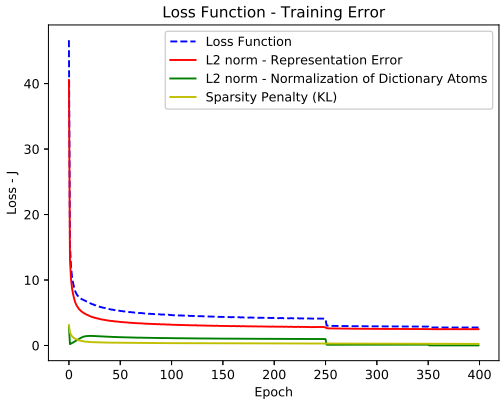
All training images were normalised such that the pixel values range from 0 to 1. The autoencoder was trained for 400 epochs using the Adam optimiser. The learning rate schedule was set to 0.001 for epochs $\leqslant 250$, 1×10^{-4} for



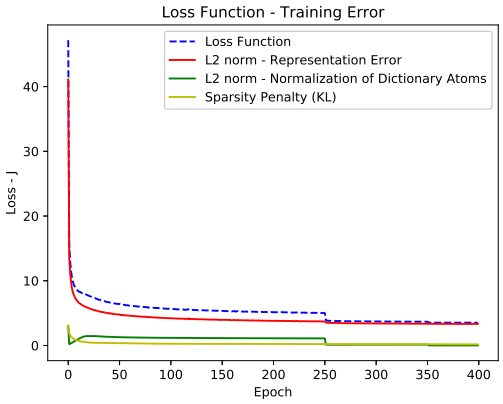
(a) Loss function for encoder model with three convolutional layers, $\sigma = 0$.



(b) Loss function for encoder model with three convolutional layers, $\sigma = 0.05$.



(c) Loss function for encoder model with three convolutional layers, $\sigma = 0.1$.

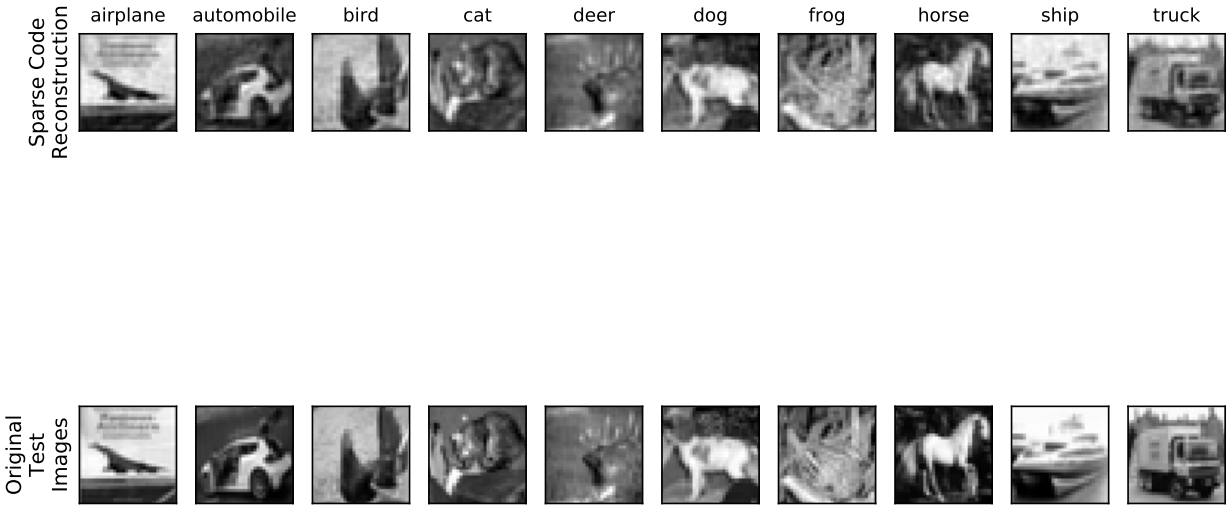


(d) Loss function for encoder model with three convolutional layers, $\sigma = 0.15$.

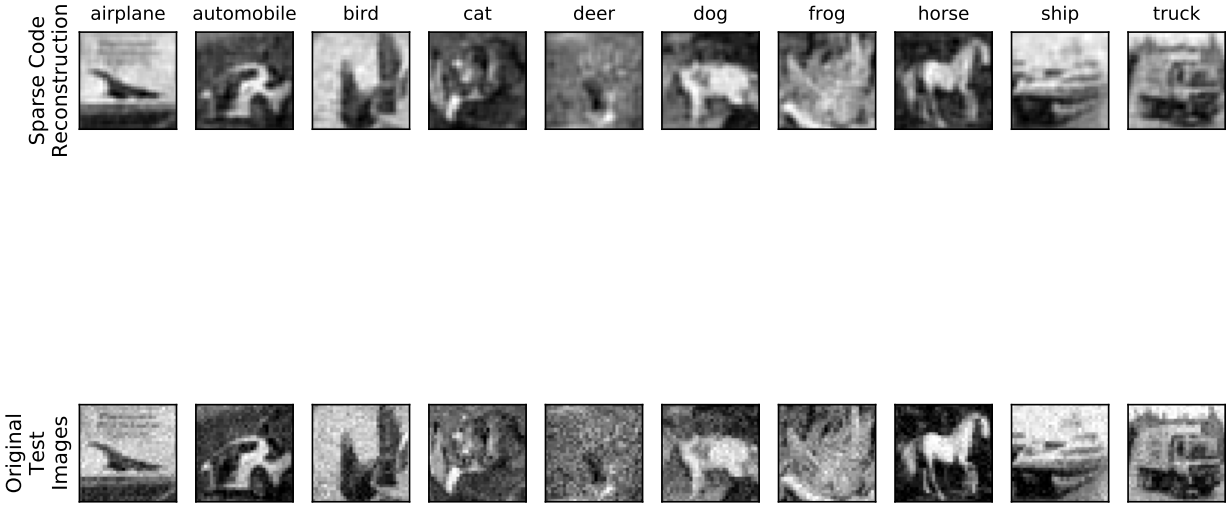
Figure 3.16 – Loss functions of the best encoder architecture.

epochs > 250 and $\leqslant 350$, and 1×10^{-5} for epochs > 350 and $\leqslant 400$. The results are presented in Figure 3.19, which demonstrates the influence of hyperparameters t, α, and β on both the quality of fit, measured by reconstruction accuracy, and density, measured by the number of nonzero elements. The compression density threshold ($C = 1$) is defined by Equation (3.15) with $d_{\text{limit}} = 1024/3000$. To account for the stochastic nature of the experiment, the average accuracy was computed over 10 independent realisations for a fixed σ, and the results are shown in Figure 3.19.

Notably, while the results obtained from the autoencoder in terms of

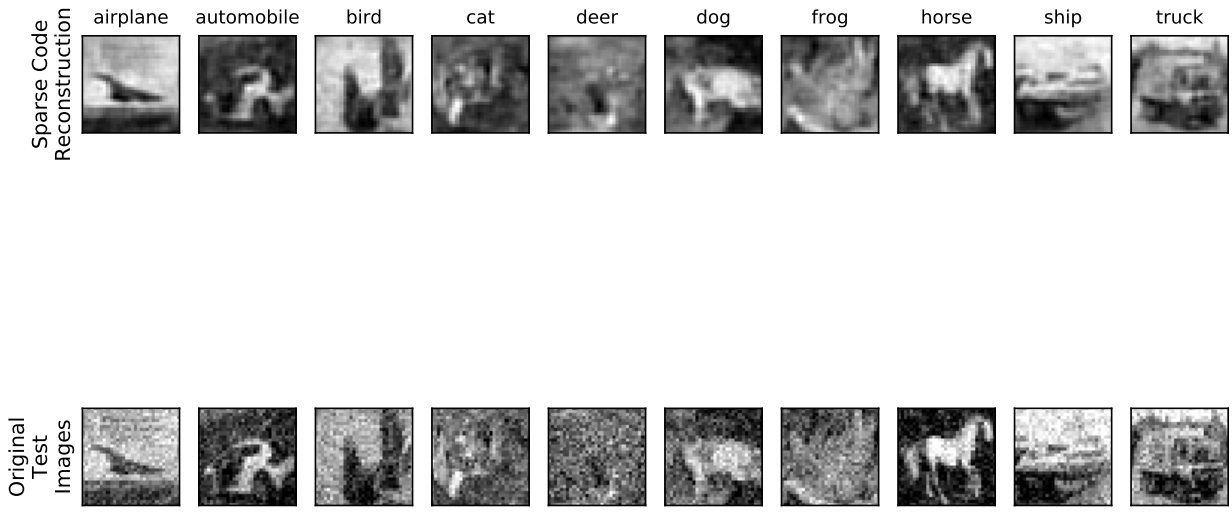


(a) Sparse reconstruction with $\sigma = 0$.

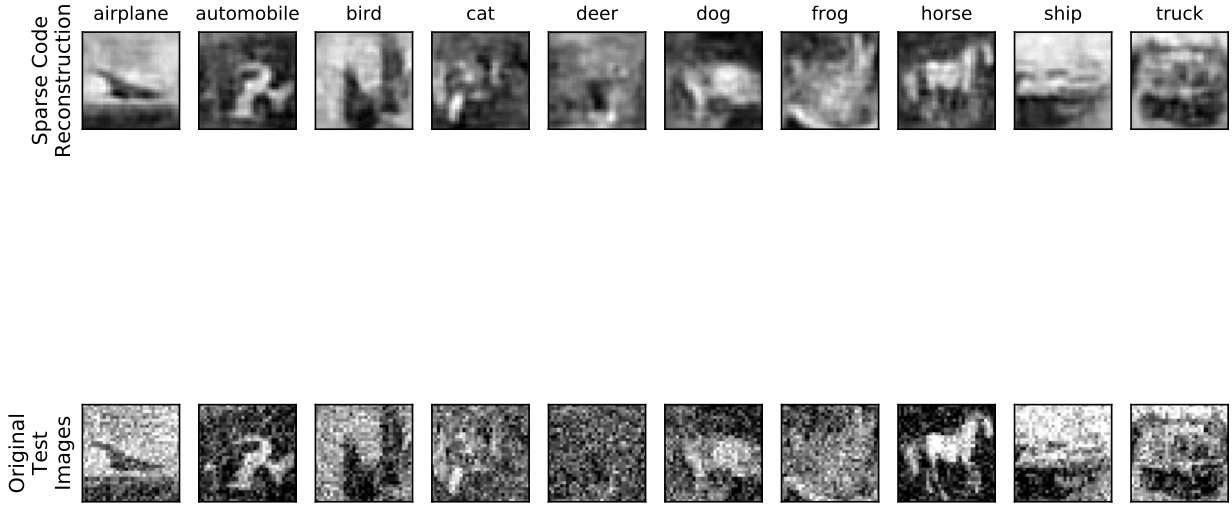


(b) Sparse reconstruction with $\sigma = 0.05$.

Figure 3.17 – Reconstruction achieved by utilising sparse codes and dictionary obtained from the best encoder model.



(a) Sparse reconstruction with $\sigma = 0.1$.



(b) Sparse reconstruction with $\sigma = 0.15$.

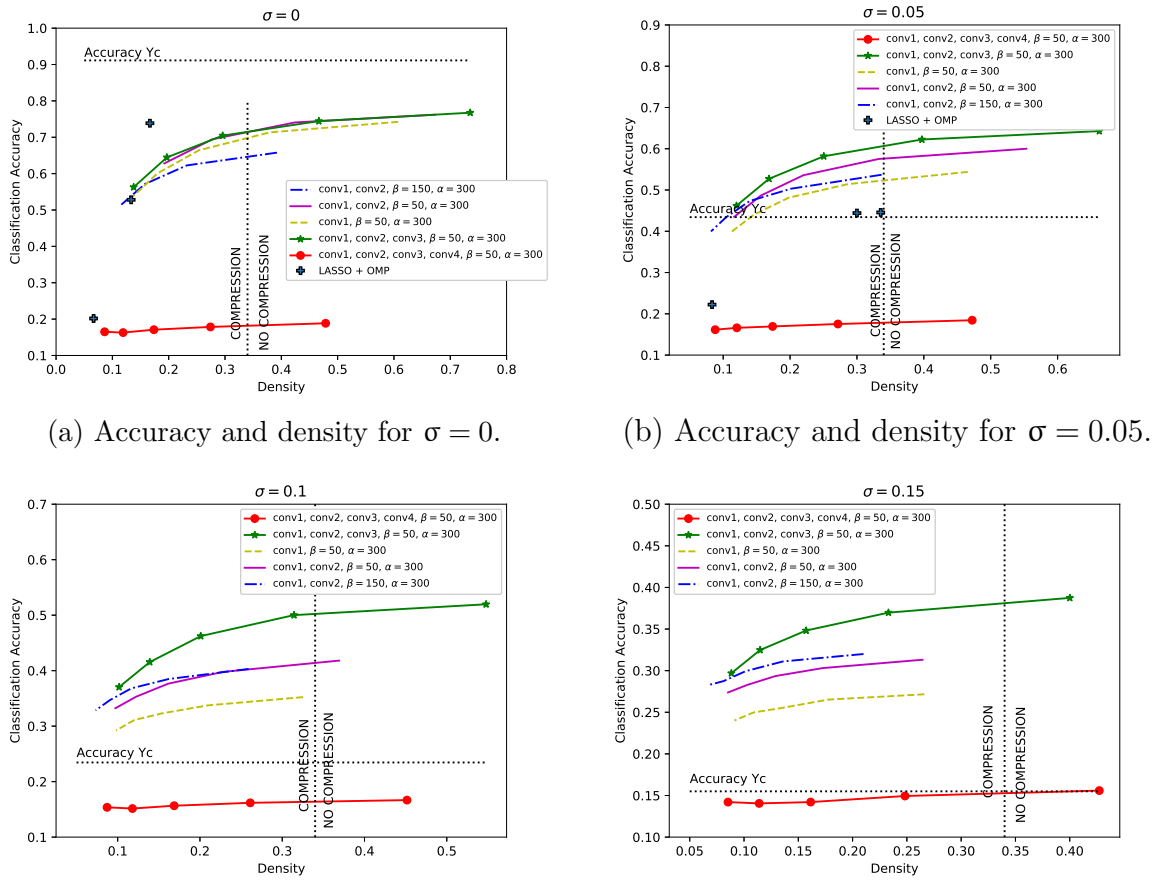
Figure 3.18 – Reconstruction achieved by utilising sparse codes and dictionary obtained from the best encoder model.

accuracy and density metrics have not yet surpassed the performance of the LASSO approach (MAIRAL *et al.*, 2009) described in Section 2.3.5, they have achieved competitive performance for $\sigma = 0$. However, when dealing with noisy conditions ($\sigma > 0$), the proposed approach outperformed the LASSO method, as demonstrated in Figure 3.19(b). Furthermore, training the autoencoder to address the problem of learning dictionaries and sparse codes proved to be highly scalable for large datasets like CIFAR-10.

In our experiment, we chose the online LASSO approach using the `MiniBatchDictionaryLearning` implementation from (PEDREGOSA *et al.*, 2011), which efficiently handles large datasets by processing subsets at a time. However, running simulations for high-density levels proved challenging due to time and memory constraints. Conversely, training convolutional and fully-connected autoencoders are not affected by sparse regularisation or the Kullback-Leibler threshold, assuming similar hardware and dataset size, making them advantageous in terms of execution time.

In a second experiment with Resnet-56, we varied the values of $\sigma \in [0,\ 0.05,\ 0.1,\ 0.15]$ and the threshold values $t \in [0.01,\ 0.02,\ 0.03,\ 0.04,\ 0.05]$. The accuracy of Y_c is highly dependent on the noise ratio controlled by σ. While the accuracy achieved for reconstructed images $\hat{Y}$ with $\sigma = 0$ is lower than that obtained with Y_c and $\sigma = 0$, it is worth noting that for $\sigma > 0$, reconstructed images $\hat{Y}$ can achieve higher classification accuracy than those from Y_c. This suggests that training the classification network using the sparse reconstruction $\hat{Y}$ could potentially lead to higher accuracy and reduced sensitivity to noise levels in the input images.

To evaluate the quality of image reconstruction, we utilised the peak signal-to-noise ratio (PSNR) and classification accuracy metrics. Notably, our approach involves an unsupervised cost function that does not explicitly incorporate labels to enforce discriminative power.



(a) Accuracy and density for $\sigma = 0$.

(b) Accuracy and density for $\sigma = 0.05$.

(c) Accuracy and density for $\sigma = 0.1$.

(d) Accuracy and density for $\sigma = 0.15$.

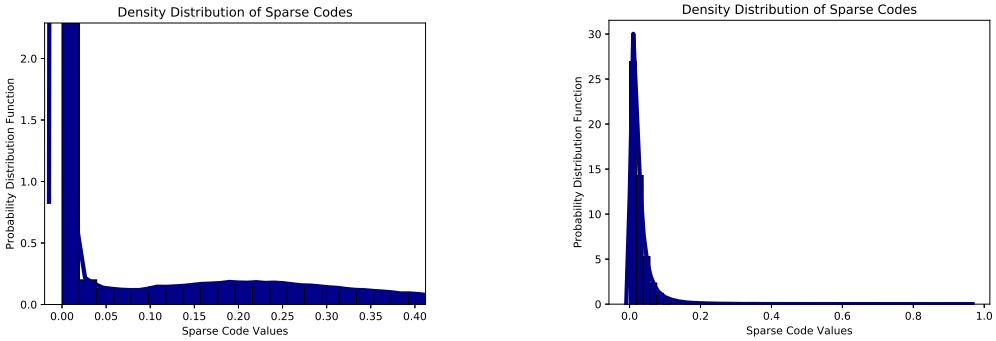
Figure 3.19 – Classification accuracy and density obtained from different encoder models.

In this study, we have examined the impact of replacing fully-connected layers with convolutional operations on the classification accuracy performance and compression rate of an autoencoder. The empirical results suggest that utilising convolutional operations to improve classification accuracy incurs a trade-off with regards to compression rate. Additionally, we have observed that deep encoder models perform less effectively than shallow models.

Our investigation has revealed that the distribution of sparse values arising from convolutional architecture displays a lower variance compared to fully-connected architecture. Figure 3.20 demonstrates that the values resulting from convolutional encoders are more prone to being in proximity to ρ, leading to a unimodal distribution. Conversely, values derived from fully-connected encoders manifest a bimodal distribution. The majority of values in one mode are

close to ρ, while the other mode has a considerable variance.

Our findings indicate that minor changes in the threshold can significantly affect the density and representation error specially in the convolutional model. This observation is supported by Figure 3.21, which underscores this relationship.



(a) Density Distribution of values of the sparse codes $\mathbf{x_i}$ for the fully-connected encoder model.

(b) Density Distribution of values of the sparse codes $\mathbf{x_i}$ for the convolutional encoder model.

Figure 3.20 – Probability Density Distribution of the sparse code values for convolutional encoder and fully-connected encoder.

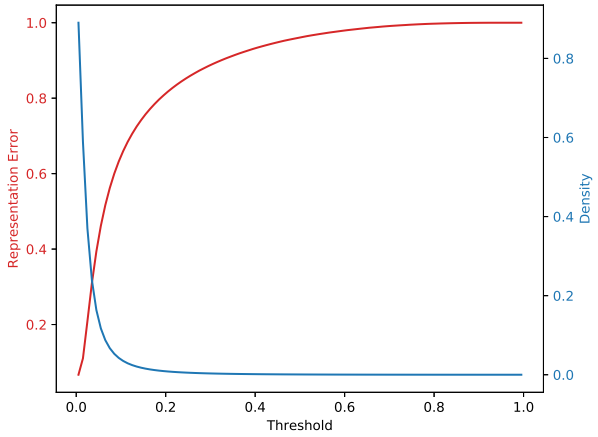


Figure 3.21 – The thresholds $\mathbf{t}$ imposed afterwards convolutional autoencoder training and their effects over density and the overall representation error $\|Y - DX\|_F^2$.

The study involving the convolutional sparse autoencoders investigated the trade-off between sparsity and the quality of representation achieved by the dictionary and the sparse codes. To this end we trained four different

autoencoder models at four different levels of additive Gaussian noise, and then compared their performance based on the image reconstruction provided by each model and using the classification accuracy obtained with the ResNet-56.

3.2.1 Sparse Dictionary Learning using U-Net architecture: an exploratory study

Exploring different convolutional architectural designs can also lead to improved performance to our end. One notable example of a promising architecture to be further explored is the U-Net models, which was initially introduced in a paper on biomedical image segmentation (RONNEBERGER *et al.*, 2015). This architecture consists of two paths: the encoder and decoder paths. The encoder path incorporates convolutional and max-pooling layers to capture the contextual information of the image. In contrast, the decoder path employs transposed convolutions to enable precise localisation and incorporates skip-like connections with the feature maps before the bottleneck layer. These connections facilitate the recovery of compressed data and enable concatenation with previous states. The U-Net architecture has demonstrated superior performance compared to previous methods in various segmentation tasks and possesses the ability to achieve good results even with a small number of training images.

In the following, we present an exploratory study incorporating the U-Net architecture and the sparse autoencoder. Our goal is to investigate an autoencoder that utilises a U-Net-type network in the encoder side while maintaining the decoder side as linear operations between a weight matrix and the sparse codes from the latent space. To reduce the training cost associated with the U-Net autoencoder architecture, we have downscaled the CIFAR-10 training dataset to 33% of its original samples. The results obtained from training the U-Net autoencoder are depicted in Figure 3.22, demonstrating a high average density level. However, this can be controlled by adjusting the threshold parameter $\mathtt{t}$, as depicted in Figure 3.23.

While the U-Net architecture has shown promising results in previous

(a) Original Images.

(b) Sparse Reconstruction.

Figure 3.22 – The sparse reconstruction achieved for a selected subset of images from CIFAR-10 test dataset.

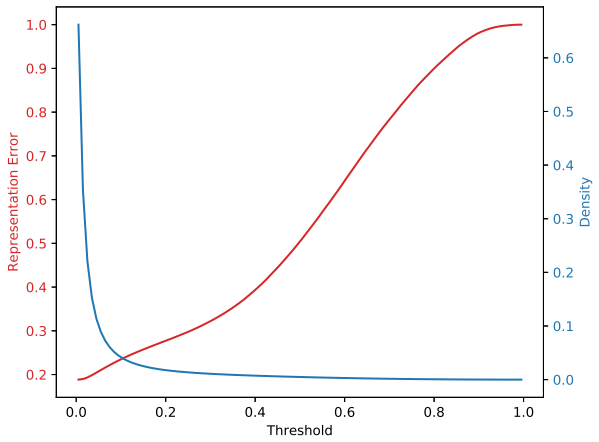


Figure 3.23 – The thresholds $\mathtt{t}$ imposed after U-Net autoencoder training and their effects on the average density levels and the overall representation error $|\mathbf{Y} - \mathbf{DX}|_{\mathrm{F}}^2$.

studies (RONNEBERGER *et al.*, 2015) (WANG *et al.*, 2021), its effectiveness in the context of sparse coding was not as successful in this exploratory study. We found that the U-net based encoder did not achieve similar levels of accuracy or sparsity as fully-connected and convolutional autoencoder architectures that have been proposed previously. The average density level and the representation error was higher than those achieved with previous autoencoder architectures.

When considering sparse coding, other proposed architectures such as fully-connected and convolutional autoencoders may be more suitable. Further research may be needed to investigate the limitations and potential benefits of using the U-Net architecture for sparse coding in more depth.

3.3 Related Work

In this Section, we briefly review studies on sparse dictionary learning approaches using autoencoders. In the study by Ayinde et al. (AYINDE; ZURADA, 2017), the output of the proposed model is given by

$$\hat{\mathbf{y}} = \mathsf{f}_{W,\mathbf{b}}(\mathbf{y}) \approx \mathbf{y} \tag{3.23}$$

where $\mathbf{y}$ is a normalised input vector, and $W = \{W_1, W_2\}$ and $\mathbf{b} = \{\mathbf{b}_1, \mathbf{b}_2\}$ represent the weight and biases of the network, respectively. The input data Y

are first encoded through W_1 into features $\mathbf{h}$. Subsequently, the features $\mathbf{h}$ are mapped back to the original input $\hat{\mathbf{y}}$ through W_2 using the following equation $\hat{\mathbf{y}} = \sigma(W_2\sigma(W_1\mathbf{y} + \mathbf{b}_1) + \mathbf{b}_2)$, where $\sigma()$ is the sigmoid function adopted for activation. To determine the optimal parameters W and $\mathbf{b}$, the average reconstruction error serves as the optimisation objective function, given by

$$J_{AE}(W, \mathbf{b}) = \frac{1}{p}\sum_{i=1}^{p} \|\sigma(W_2\sigma(W_1\mathbf{y}_i + \mathbf{b}_1) + \mathbf{b}_2)\|_2^2 \tag{3.24}$$

This term is also used to penalise representation error in the sparse dictionary learning problem. To enforce the autoencoder to learn a sparse representation, the output of $\mathbf{h}$ is bounded using the Kullback-Leibler (KL) divergence function. The average activation of a particular neuron j, given the input $\mathbf{y}_i$, is denoted by $h_j(x_i)$ and is calculated as

$$\hat{p}_j = \frac{1}{p}\sum_{i=1}^{p} h_j(\mathbf{y}_i) \tag{3.25}$$

The Kullback-Leibler divergence is a measure of the difference between two probability distributions, and is commonly used as a sparsity-inducing regulariser in machine learning. The idea is to use KL divergence to encourage the model to produce probability distributions that are closer to a sparse distribution than to a dense distribution. To this end we can use a target distribution, such as a uniform distribution or a Bernoulli distribution with a small probability of a nonzero value. Then we minimise the KL divergence between the model's output distribution and the target distribution. The KL divergence is employed to penalise the activation of the neuron units in the latent space of the autoencoder $\hat{\mathbf{p}} = \{\hat{p}_j\}|_{j=1}^{\nu}$, where ν is the total number of neurons at the hidden layer. The divergence is defined as follows

$$S(\rho||\hat{p}) = \sum_{j}^{\nu} \rho \log\frac{\rho}{\hat{p}_j} + (1-\rho)\log\frac{(1-\rho)}{(1-\hat{p}_j)} \tag{3.26}$$

In addition to that, a weight decay term $\mathcal{D}$ is also added to the cost function of the autoencoder in order to prevent overfitting. The cost function of

the sparse autoencoder proposed in (AYINDE; ZURADA, 2017) is given by

$$J_{SAE} = J_{AE}(W, \mathbf{b}) + \beta \mathcal{S}(\rho \| \hat{\rho}) + \mathcal{D}(W) \tag{3.27}$$

where β controls the sparsity penalty term. The main contributions of this paper are the introduction of the constrained autoencoder framework for dictionary learning, and the demonstration of its effectiveness for classification tasks involving MNIST (DENG, 2012) dataset.

In (TARIYAL *et al.*, 2016), the authors propose a deep dictionary learning approach in which, instead of learning a single dictionary, multiple levels of dictionaries are learned using deep autoencoders. Learning all the dictionaries simultaneously makes the problem highly nonconvex. Also, learning so many parameters (atoms of many dictionaries) can easily lead this approach to overfitting.

The authors propose a deep learning approach for dictionary learning that learns multiple levels of dictionaries using deep autoencoders. The approach learns the dictionaries in a greedy way to account for nonconvexity and overfitting. The learned representations are encouraged to be sparse using a constraint. The method is evaluated on classification and clustering tasks, and obtained promising results.

The work presented in (MAKHZANI; FREY, 2013) is the one in which we can find greater consistency with respect to the contributions brought in this dissertation. The article explains how sparse autoencoders can be handled in the context of sparse coding with incoherent matrices. This perspective helps us clarifying why the sparse autoencoders can achieve good classification results. As the presented sparse autoencoder achieves exact sparsity in the hidden representation, we can use the resulting representation to achieve state-of-the-art classification accuracies without using any other nonlinearity or regularisation.

A sparse autoencoder maps an input vector $\mathbf{y}_i$ onto a hidden representation $\mathbf{h}_i = f(W_1 \mathbf{y}_i + \mathbf{b})$ where f is the activation function, e.g., linear,

sigmoidal or ReLU, W_1 is the weight matrix and $\mathbf{b}_1$ is the bias vector of the encoder layer. The hidden representation is linearly mapped onto the output using $\hat{\mathbf{y}}_i = W_2\mathbf{h}_i + \mathbf{b}_2$. The parameters $\{W_1, W_2, \mathbf{b_1}, \mathbf{b}_2\}$ are expected to minimise the mean squared error of $\|\mathbf{y}_i - \hat{\mathbf{y}}_i\|_2^2$ over all training samples. Once the k largest activities are selected in the hidden layer, the function computed by the network is linear. So the only nonlinearity comes from the selection of the k largest activities in the hidden layer. This selection step acts as a regularisation that prevents the use of a large number of hidden units when reconstructing the input. In addition, the authors imposed a tied weight restriction so that $W_1 = W_2^T$.

Furthermore, instead of using only the k largest elements of $W_1\mathbf{y}_i + \mathbf{b}$ as the features, the authors have also observed slightly better performance whether using the $\alpha k, \alpha > 1$ largest hidden units with α selected using the set of validation data. The algorithm is summarised as follows.

Pseudocode 3.1 Sparse Autoencoders:

Training

1. Perform the feedforward phase and compute

$$\mathbf{h}_i = W_1 \mathbf{y}_i + \mathbf{b}_1$$

2. Find the k largest activations of $\mathbf{h}$ and set the rest to zero

$$\mathbf{h}_{i,\Gamma} = 0 \quad \text{where} \quad \Gamma = \sup_k (\mathbf{h}_i)$$

3. Compute the output and the error using the sparsified $\mathbf{h}$

$$\hat{\mathbf{y}}_i = W_2 \mathbf{h}_i + \mathbf{b}_2$$

$$\|\mathbf{y}_i - \hat{\mathbf{y}}_i\|_2^2$$

4. Backpropagate the error through the k largest activations defined by Γ and iterate.

Sparse Coding

1. Compute the features $\mathbf{h}_i = W_1 \mathbf{y}_i + \mathbf{b}_1$. Find its αk largest activations and set the rest to zero

$$\mathbf{h}_{i,\Gamma} = 0 \quad \text{where} \quad \Gamma = \sup_{\alpha k} (\mathbf{h}_i)$$

Dictionary Learning

$$D = W_2^T$$

As shown in Chapter 2, the conventional approaches to obtain the sparse codes usually rely on using the current dictionary D and a pursuit algorithm to solve Equation (2.4). Convex relaxation methods such as ℓ_1-norm minimisation or greedy methods such as OMP are used to find suited sparse codes. These sparse codes are then used to update the dictionary, using techniques such as the Method of Optimal Directions (MOD) or K-SVD. As these methods are computationally expensive - MOD requires inverting the data ma-

trix at each step and K-SVD needs to compute a SVD in order to update every column of the dictionary - the sparse autoencoder training ends up being a relatively more efficient and faster way to achieve suitable representations specially at large datasets.

3.3.1 Comparing Proposed Approach and Related Work

This session highlights the distinctions between the proposed approach and other existing methods. The main difference between Hu et al. (HU; TAN, 2018) and the proposal introduced in Section 3.1 lies in the architecture and the way the sparsity constraint is incorporated. In (HU; TAN, 2018), a nonlinear autoencoder is used with a fixed sparse penalty parameter. The dictionary and sparse codes are learned jointly using an iterative algorithm based on the Alternating Direction Method of Multipliers (ADMM). Our proposal uses a more flexible approach where the nonlinearity in the encoder can be chosen arbitrarily.

Moreover, in (HU; TAN, 2018), sparsity is enforced using a fixed penalty parameter, while in our proposal, sparsity is enforced using the Kullback-Leibler (KL) divergence between the empirical distribution of the hidden layer activations and a target distribution. This allows for more fine-grained control over the sparsity level and can be adjusted to different settings and datasets. Overall, both methods aim to learn a sparse representation of the input data using an autoencoder architecture but differ in the specifics of the architecture and the sparsity constraint.

In Ayinde et al (AYINDE; ZURADA, 2017), the cost function for the autoencoder is defined as the average reconstruction error, which penalises the difference between the input and output of the autoencoder using the squared Euclidean distance. To induce sparsity, the KL divergence function is used to bound the activation of the hidden layer neurons. The authors also incorporate regularisation terms to avoid overfitting and to control the norm of the dictionary atoms. They use the ℓ_1-norm of the difference between the identity matrix

and the outer product of the dictionary atom, which forces the norm of the atoms to unity.

Overall, the difference between the approach in (AYINDE; ZURADA, 2017) and ours is in the adopted regularisation techniques used to induce sparsity and impose certain properties on the learned dictionary. Our proposal combines ℓ_2-norm regularisation and KL divergence, while (AYINDE; ZURADA, 2017) uses KL divergence to bound the hidden representation $\mathbf{h}$ and the ℓ_1-norm to bound the dictionary atoms.

3.4 Final considerations

In this chapter, we explored the performance of different encoder architectures, including the fully-connected encoder, the convolutional encoder, and the U-Net architecture, from the perspective of image reconstruction quality and achieved levels of sparsity. Through our investigations, important conclusions were drawn.

Firstly, the fully-connected encoder demonstrated its effectiveness in solving the sparse dictionary learning problem, reconstructing the CIFAR-10 images employed to train the model, and achieving high reconstruction accuracy under sparse representations enforced in the latent space. Additionally, we showed that decreasing image patch size increases the sparsity required for compression and decreasing patch size lowers the accuracy obtained at the compression conditions. In this case, the accuracy is inversely proportional to the achieved reconstruction error.

Secondly, the convolutional encoder also exhibited reasonable performance by leveraging its local connectivity and weight sharing. However, the proposed fully-connected encoders outperformed the convolutional architectures achieving higher sparsity levels under lower reconstruction errors on CIFAR-10.

Lastly, the U-Net architecture, which incorporates skip connections to

establish direct connections between the encoder and decoder layers, showed promising potential in image reconstruction tasks. However, in our exploratory study, we found that its effectiveness in the context of sparse coding was not as successful as the other proposed architectures. The U-Net-based encoder did not achieve comparable levels of accuracy or sparsity as the fully-connected and convolutional encoders.

Based on these findings, the next chapter will focus on leveraging the strengths of the proposed autoencoder-based framework to investigate a novel sparse dictionary learning compression scheme. We will delve into the details of the proposed framework, its experimental setup, and the evaluation of its compression performance while comparing it to other important methods used in image compression.

Chapter 4

Image Reconstruction Using Sparse Autoencoder and Analytic Dictionaries

There are two types of image compression processes: lossy compression and lossless compression. In lossless compression, we are able to perfectly retrieve the details of the original image, while in lossy compression the image recovery is not perfect and therefore only represents an approximation. PNG is an example of file format that allows lossless image storage as it preserves all details of the original image. JPEG, on the other hand, is an example of file format that allows lossy image storage, as it is not able to perfectly retrieve the original image details. Image compression is a very important process to reduce memory size occupied while loading or storing images and videos.

The reason behind the adoption of sparsity-based modelling under compression tasks is the fast decay of the sparse coefficients over each dictionary atom. Early attempts for designing analytic dictionaries were based on building a set of atoms that lead to several well-known transforms such as the Fourier Transform and its discrete version, the Discrete Cosine Transform, Wavelet Transforms, Curvelets among others (RUBINSTEIN *et al.*, 2010).

A different approach to the sparse modelling consists of learning a dictionary from some training data samples since the customised dictionaries could

more efficiently capture the underlying structures of such image patches as well as generalise them to certain classes of signals. Finding appropriate dictionaries with good reconstruction power of as many signals as possible, and high sparseness and compactness ability was the main target of the work presented in Chapter 3.

In this chapter, we aim to investigate the limitations of traditional analytic dictionaries in effectively capturing the diverse patterns present in images. To this end, we explore Fourier and Wavelet transforms and compare their results with those obtained from dictionary learning and sparse coding methods based on autoencoders. Furthermore, we survey several research papers that focus on dictionary learning and its applications in image compression, highlighting the fingerprint for further investigations in this field to develop more effective compression methods. Finally, we present a dictionary-based image coding framework that employs a combination of dictionary learning, quantisation, and entropy coding to achieve efficient compression while preserving essential image features.

4.1 Fourier and Wavelet Transforms

Dictionary learning aims to construct a mapping that simplifies data representation into a new coordinate system, which is a common technique in various domains such as data analysis and digital signal processing. Data transformations and spectral decomposition are fundamental concepts in various scientific fields, including data analysis and digital signal processing. For instance, the singular value decomposition (SVD) and graph spectral decomposition are examples of data transformations used in data analysis. Similarly, spectral decomposition are commonly used in digital signal processing to analyse the frequency content of signals and to simplify their representation in the frequency domain.

The Fourier basis, introduced in 1822 (FOURIER, 1822) to study the theory of heat, describes a signal in terms of its frequency content, expressed as a combination of orthogonal waveforms. The Fourier transform gained popularity with the introduction of the Fast Fourier Transform (FFT) in 1965 (COOLEY; TUKEY, 1965), which enabled efficient numerical computation of the transform through a recursive approach and breaking down the DFT of composite sizes into smaller DFTs. Consequently, the FFT has become a fundamental tool in computational mathematics and engineering, facilitating real-time image and audio compression, communication networks, advanced signal processing techniques, and data analysis.

More complex problems and datasets have led to the development of tailored bases such as data-driven dictionaries. In addition, wavelets have also emerged as popular alternatives for advanced signal processing, denoising techniques, and compression efforts (BRUNTON; KUTZ, 2019). Unlike Fourier-based components (sine and cosine functions), wavelets are typically localised in both time and frequency domains, making them more effective in capturing local features of the signal. Fourier-based components have infinite support, meaning that they are defined for all $\mathtt{t} \in \mathcal{R}$ and are not zero within any finite interval contained in $\mathcal{R}$. As a result, the Fourier transform extends over the whole time domain and the Fourier coefficients represent an average for the entire time domain at each frequency considered by the transform.

The continuous wavelet transform is a method of decomposing signals that involves dynamically scaling and shifting the analysis kernel to better localise features in time and frequency. The *Heisenberg Uncertainty Principle* states that frequency spread and duration (temporal spread) cannot both be made arbitrarily small. Wavelet transforms are designed to overcome this limitation. A wavelet is a function that grows and decays in a limited and closed interval, being zero outside of it. This is the compact support property which is satisfied by the Daubechies wavelets (MALLAT, 1999). The wavelets can accurately identify

low frequencies and also provide improved time localisation at high frequencies. This is because wavelets have a limited and closed interval of support satisfying the Heisenberg Uncertainty Principle, unlike the sine and cosine Fourier basis functions that oscillate over the entire real domain.

While the FFT algorithm is generally preferred for computational efficiency, it is more instructive to begin with the fundamental formulation of the Discrete Fourier Transform (DFT). The DFT is mathematically defined as follows:

$$X[f] = \sum_{k=0}^{N-1} x[k] e^{-\frac{j2\pi fk}{N}} \tag{4.1}$$

and the inverse discrete Fourier transform (IDFT) is given by:

$$x[k] = \sum_{f=0}^{N-1} X[f] e^{\frac{j2\pi fk}{N}} \tag{4.2}$$

The Discrete Fourier Transform (DFT) is a linear operator represented by a matrix that maps the discrete-time sequence of data points in $x[k]$ onto the discrete frequency domain $X[f]$. The DFT can be computed by matrix multiplication, which can be expressed as

$$\begin{bmatrix} X[1] \\ X[2] \\ X[3] \\ \vdots \\ X[N] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w_N & w_N^2 & \cdots & w_N^{(N-1)} \\ 1 & w_N^2 & w_N^4 & \cdots & w_n^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & w_N^{(N-1)} & w_N^{2(N-1)} & \cdots & w_N^{(N-1)^2} \end{bmatrix} \begin{bmatrix} x[1] \\ x[2] \\ x[3] \\ \vdots \\ x[N] \end{bmatrix} \tag{4.3}$$

with integer multiples of a fundamental frequency $w_N = e^{-\frac{2\pi i}{N}}$, resulting in a complex-valued matrix. As a result, the output $X[f]$ has both magnitude and phase components, which provide useful physical interpretations. Figure 4.1 displays the real part of the DFT matrix for an input size of $N = 128$.

The Fast Fourier Transform (FFT) is a computationally efficient algorithm that exploits the fact that the Discrete Fourier Transform (DFT) can

be implemented much more effectively when the number of data points N is a power of 2. Specifically, when $N = 1024 = 2^{10}$, for example, the DFT matrix F^{1024} can be expressed in a simplified form as follows:

$$F^{1024} = \begin{bmatrix} I_{512} & -D_{512} \\ I_{512} & -D_{512} \end{bmatrix} \begin{bmatrix} F_{512} & 0 \\ F_{512} & 0 \end{bmatrix} \tag{4.4}$$

where I_{512} is the 512×512 identity matrix, and D_{512} is given by

$$D_{512} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & w & 0 & \cdots & 0 \\ 0 & 0 & w^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & w^{511} \end{bmatrix} \tag{4.5}$$

where w is the fundamental frequency $w = e^{\frac{2\pi i}{N}}$. The DFT can be thus calculated as

$$X[f] = F^{1024} \begin{bmatrix} x_{even} \\ x_{odd} \end{bmatrix} \tag{4.6}$$

where x_{even} are the even index elements of x, and x_{odd} are the odd index elements of x. If $N = 2^p$, the same process can be repeated, and F_{512} can be represented by F_{256}, which can then be represented by $F_{128} \rightarrow F_{64} \rightarrow F_{32} \cdots$. If $N \neq 2^p$, the vector can be padded with zeros until it reaches a power of 2. The FFT then involves an efficient recurrence of even and odd indices of subvectors of x, and the computation of several smaller 2×2 DFT computations (BRUNTON; KUTZ, 2019).

The two-dimensional FFT of a matrix $Y \in \mathcal{R}^{m \times p}$ is obtained by first applying the one-dimensional FFT to every row of the matrix, and then applying the one-dimensional FFT to every column of the intermediate matrix. This process of performing row-wise and column-wise Fourier transform is illustrated in Figure 4.2. It is important to mention that the order of taking the Fourier transform of rows and columns can be interchanged without affecting the final result.

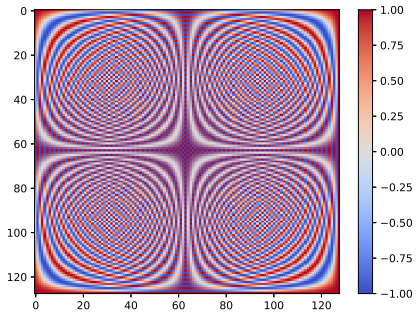


Figure 4.1 – Real part of DFT matrix for $N = 128$

Wavelet transforms extend the concepts of Fourier transform to more general orthogonal bases, and they partially overcome the uncertainty principle mentioned earlier by relying on a multi-resolution decomposition. This approach is particularly useful for decomposing complex signals arising from multi-scale processes, such as images and audio signals. The basic idea behind wavelet analysis is the mother wavelet function $\psi(t)$, which can be used to generate a family of scaled and translated versions of the function:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right) \tag{4.7}$$

The parameters a and b are responsible for scaling and translating the function $\psi(t)$, respectively. The scaling factor ensures that the energy of the wavelet is conserved across different scales. The family of wavelets obtained by varying a and b is known as the wavelet basis. By convolving a signal with each of the wavelets in the wavelet basis and computing the inner products, we obtain the wavelet coefficients that represent the signal at different scales and positions. The wavelet coefficients provide a multi-resolution representation of the signal, allowing us to capture both local and global features of the signal. Each segment in Figure 4.3 is obtained by scaling and shifting the mother wavelet function $\psi(t)$ by appropriate values of a and b. These values are chosen to capture different features of the signal at different scales and positions. If the family of scaled and translated mother wavelets functions are orthogonal, then the

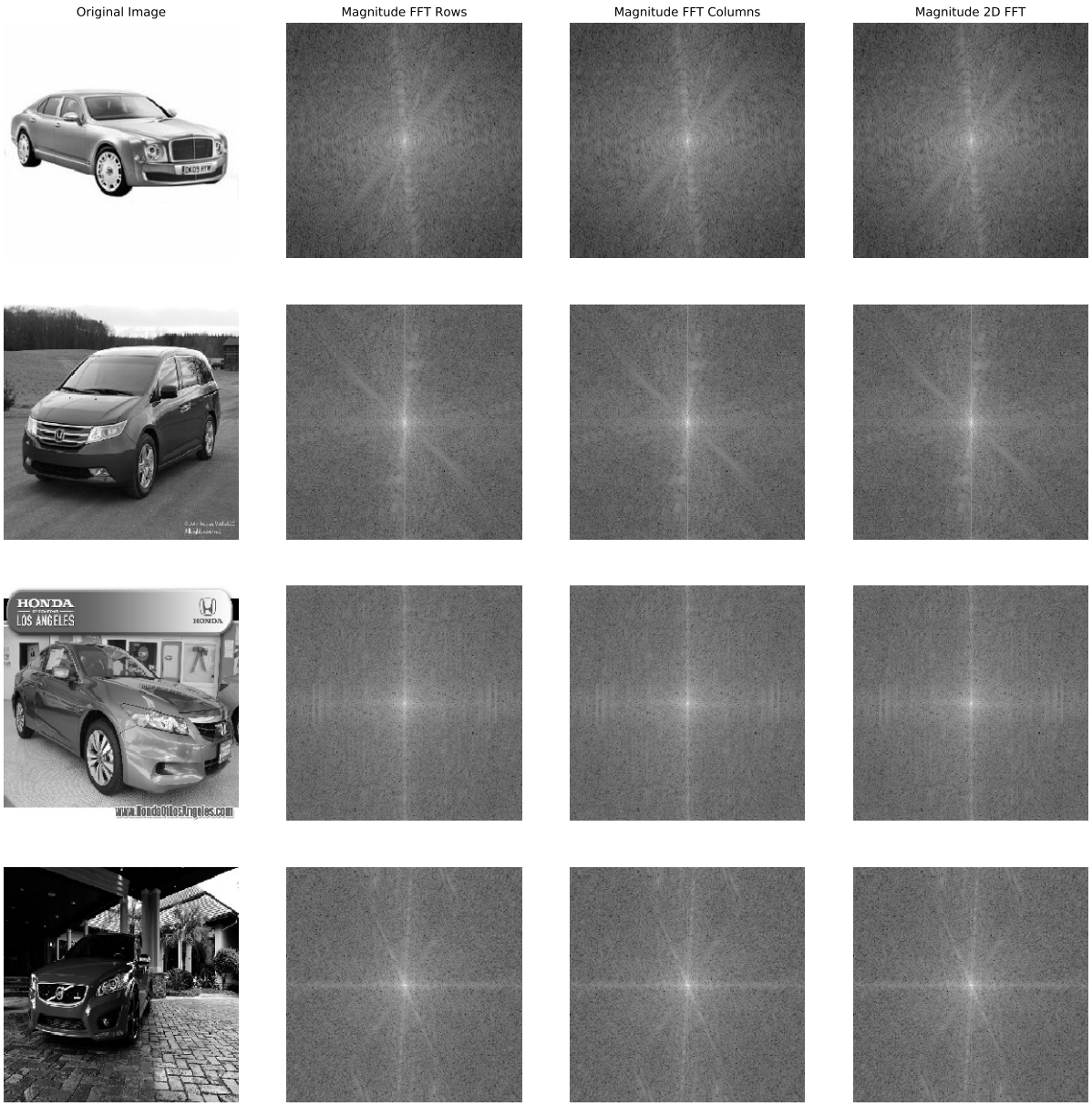


Figure 4.2 – Schematic of 2D FFT. At the second column the FFT is taken at each row. At the third column the FFT is taken at each column of the resulting transformed matrix.

resulting basis functions can be used for signal projection.

In the next section, we present our study on the Cars dataset (KRAUSE *et al.*, 2013) to build a data-driven dictionary for image coding. Our method involves a sparse autoencoder with residual connection in the encoder side for

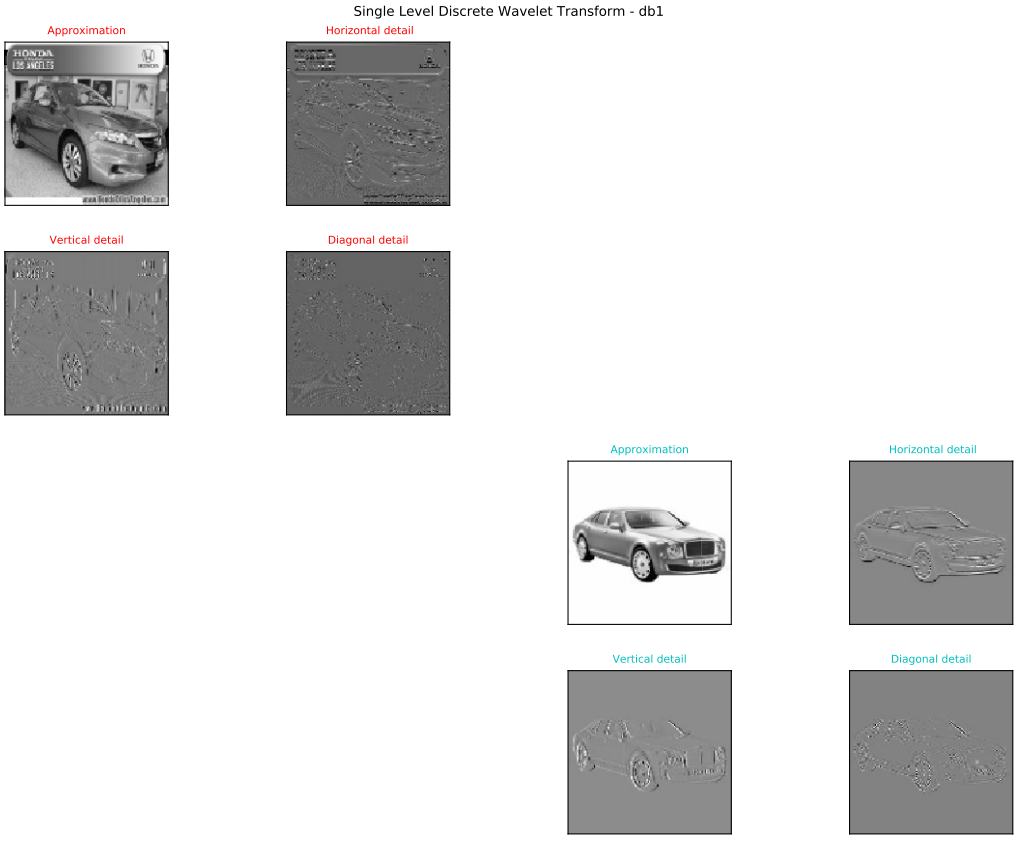


Figure 4.3 – Illustration of single level discrete wavelet transform.

capturing more complex features and patterns in images while also avoiding the problem of the vanishing gradient. We will compare the performance of our approach with that of the Fourier and Wavelet transforms, which are classical signal processing techniques commonly used for image compression.

4.2 Experiment and Results

In this section, we compare the visual quality of images obtained from three different image processing techniques: Fourier transform, Wavelet transform, and sparse representation from a data-driven dictionary. We evaluate the image quality using the Peak Signal-to-Noise Ratio (PSNR) metric. The purpose of this comparison is to demonstrate the effectiveness of the sparse representation approach in producing high-quality images and to assess its performance

Table 4.1 – Training Parameters.

Parameters	**Value**
Epochs	250
Batch Size	100
Train dataset	6108
Validation dataset	2036
Test dataset	8041
Dictionary atoms	12100
α	300
β	250
ρ (KL)	0.01
Patch Size	4 x 4
W x H	256 x 256

against the more established Fourier and Wavelet transform techniques.

We use the Cars dataset (KRAUSE *et al.*, 2013) containing 16,185 colour images. The data-driven dictionary is constructed to be a general-purpose dictionary applicable to the 196 classes available in the dataset. We train an autoencoder with 8144 grayscale-converted images from the training set to learn a sparse representation of the image data.

We obtained the dictionary and the sparse codes employed for the image coding framework using an approach similar to that introduced in Chapter 3. After converting the images to grayscale, we split them into patches to feed the encoder side and the output of the model. The cost function adopted on the model training is the same indicated in Equation (3.1). The regularisation parameters and the hyperparameters are indicated in Table 4.1.

We implemented a convolutional autoencoder with residual connections. Our intent was to convey the advantages of adding residual connection into the model to improve results. As discussed in Section 3.1, a residual neural network (ResNet) in known for skipping connections and creating shortcuts to jump over some layers while mitigating the problem of the vanishing gradient.

To convert color images to grayscale, each pixel is typically represented

by 8 bits, which correspond to the light intensity at that particular point. As a result, the brightness values in such images can range from 0 (representing a black pixel) to 255 (representing a white pixel).

For the Fourier and Wavelet transforms, we compressed images using percentile-based methods applied to the coefficients obtained from the respective transforms. We performed an 8-level decomposition using the Daubechies wavelets for the Wavelet transform, retaining only the largest $5\%, 3\%$, and 0.2% of the coefficients and setting the rest to zero. A similar approach was employed for the Fourier transform.

In our proposed dictionary learning and sparse coding framework, we adjusted the thresholds applied to the sparse codes to obtain compressed images with a similar number of coefficients as the other methods.

Figures 4.4, 4.5, and 4.6 demonstrate the effectiveness of the Fourier, Wavelet, and proposed dictionary learning frameworks in reconstructing images at different compression rates. The compression rates are determined by the chosen percentiles of coefficients used to reconstruct the images. Our proposed framework achieves superior visual quality and the highest PSNR value at the first two compression rates. However, at the highest compression rate, the visual quality of the DL framework is inferior to that of the other transforms.

Based on the results obtained, it can be concluded that the sparse autoencoder based framework for dictionary learning shows promising potential for achieving higher compression rates while preserving the visual quality of the image. This approach is a competitive alternative to traditional analytic transforms like the Wavelets, as it can effectively exploit the specific statistical properties of the signals of interest. In addition, the proposed DL framework also has the potential to surpass the performance of other transforms in terms of the quality of the reconstructed image. Therefore, it can be considered a promising option for developing image codec frameworks to achieve high compression rates

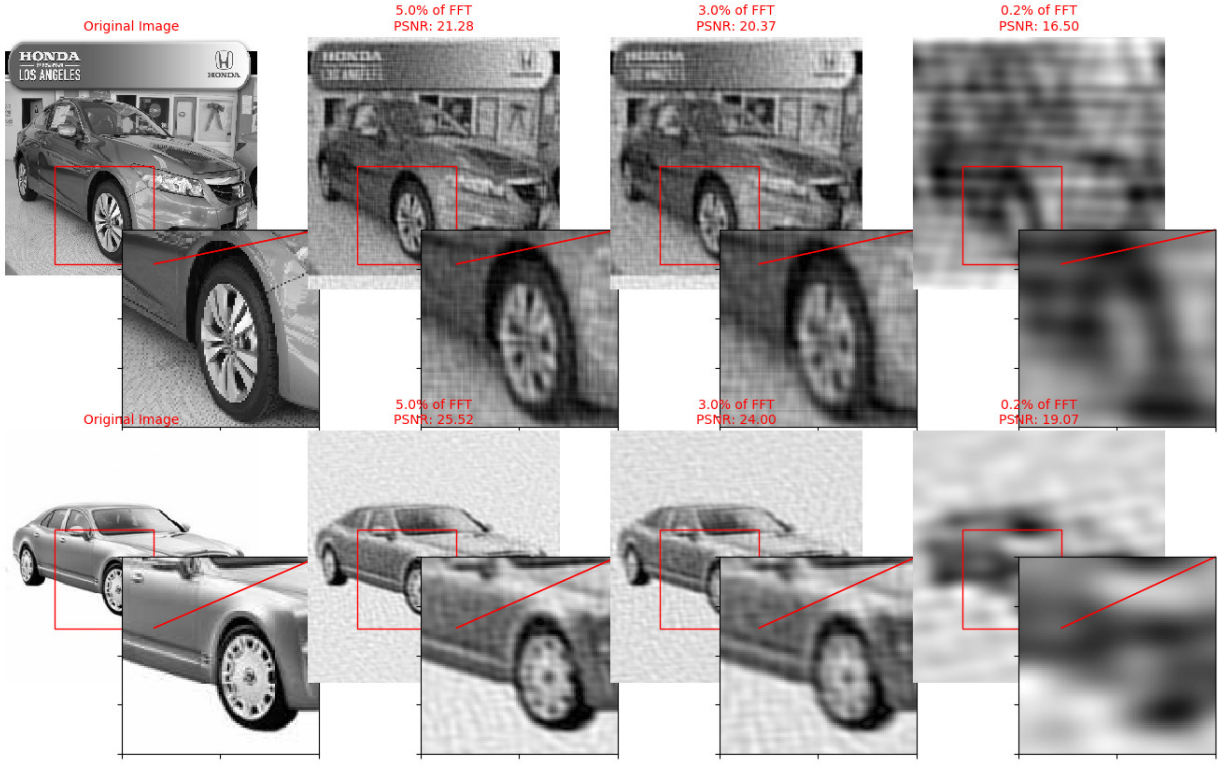


Figure 4.4 – Compressed images were obtained using percentiles of the coefficients. Fixed thresholds were set to keep 5%, 3% and 0.2% of the Fourier coefficients with the largest magnitudes.

while maintaining the quality of the image.

4.3 Final Considerations

Reducing the cost of storing or transmitting image signals with low degradation in quality is the key goal of any lossy image compression algorithm. These algorithms seek to remove redundancies in the data, capturing most of the image information. Typically, they encode the image into a transform domain with only a few significant coefficients. For example, the JPEG (WALLACE, 1992) and JPEG2000 standards (SKODRAS *et al.*, 2001) are both examples of image file formats using the sparse representation achieved from analytic dictionaries: the Discrete Cosine Transform (DCT) and Discrete Wavelet Transform

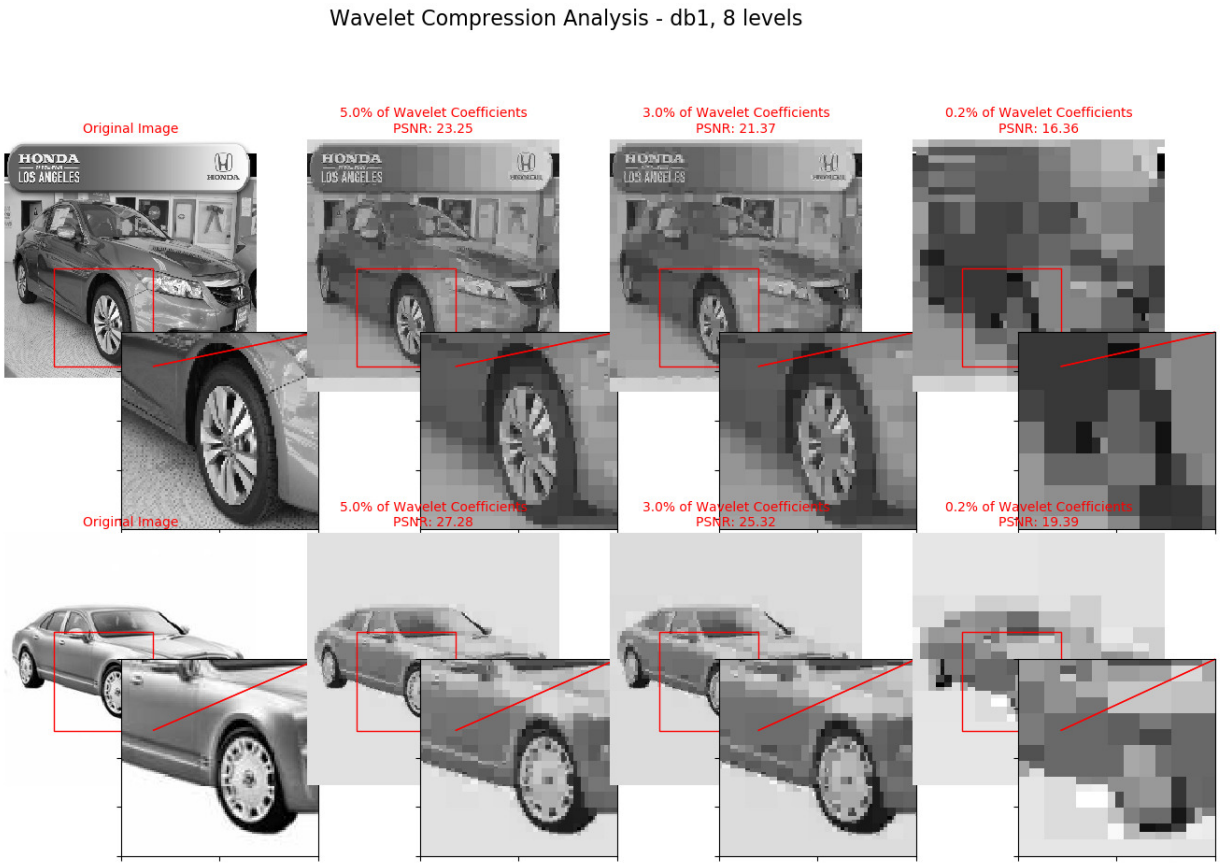


Figure 4.5 – Compressed images were obtained using percentiles of the coefficients. Fixed thresholds were set to keep 5%, 3% and 0.2% of the Wavelet coefficients with the largest magnitudes.

(DWT), respectively.

Dictionaries are essential tools for signal representation and analysis. Analytic dictionaries, which are based on smooth or piecewise-smooth functions like Fourier or Wavelet dictionaries, are commonly used due to their analytic formulation and optimal proofs and error rate bounds. However, these dictionaries may fall short in representing the complex patterns present in natural images that exhibit a variety of regularities, such as regular areas, edges, and textures. As a consequence, they may not be efficient in representing a wide range of regularities that are common in such images (AKBARI; TROCAN, 2019).

To address this limitation, a promising approach is to model sparse

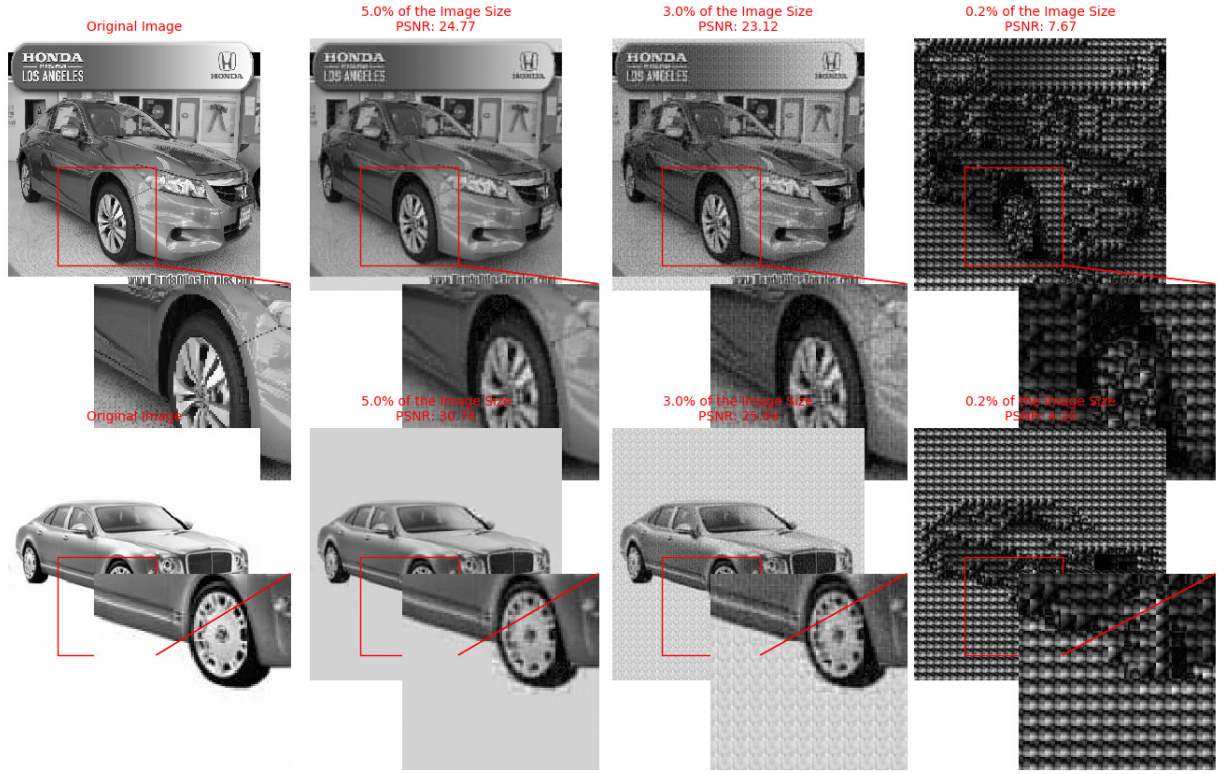


Figure 4.6 – Compressed images using Dictionary Learning and Sparse Coding Framework.

signals with a learned dictionary from the framework introduced earlier in this chapter and also detailed in Chapter 3. This approach has the potential to yield a compression framework that benefits from the sparse representation of images over a dictionary designed for a certain class of images. Future investigations can focus on exploring the potential of this approach in capturing the diverse regularities present in natural images and developing more effective and adaptable sparse signal models.

In the following subsections we will summarise a collection of image and video compression methods utilising dictionary learning and sparse coding techniques, and introduce our proposal of a new image compression framework using the elements of the sparse convolutional residual autoencoder described earlier in this chapter. We aim to explore the potential of this framework further in the future and develop more effective models for sparse signal representation in the context of image compression.

4.3.1 Dictionary Learning and its applications in the field of image and video compression

In (MAZAHERI *et al.*, 2013), the authors introduce a new method named *Tree K-SVD*. A set of dictionaries in a tree structure is trained. At each tree level, a dictionary is learned from the image residual achieved in the previous level, i.e., the residual is obtained from the difference between the original and recovered images using the trained dictionary at the previous level of the tree. Sparse coding is done by selecting the atoms from each dictionary at each tree level. Once a first atom is selected in the first level, a choice is made between staying at the same level, i.e., in the same dictionary to select another atom, or going to the next tree level. For that, the two most correlated atoms to the current residual vector are found, one at the same level and the other one at the next level. The atom minimising the energy of the residuals is kept in the representation, hence the sparsity per level is automatically adapted to decrease the distortion.

Authors in (SUN *et al.*, 2014b) introduce the multisample sparse representation online dictionary learning approach for image compression. Each image patch is encoded with a certain sparsity level. The sparse vector can be obtained through classical algorithms such as the basis pursuit algorithms and matching pursuit techniques, among others. As these conventional approaches consider constant sparsity levels for representing all the image patches with different patterns among each other, this may lead to a weakness in terms of image distortion and compression results.

In (ZHANG *et al.*, 2017), the dictionary is learned from a single image that shares the most common similar content with another image from the target set. The authors also suggest a dictionary reordering approach to improve the compression performance further. The reordering is made by adjusting the order of dictionary atoms according to their use frequency.

4.3.2 Dictionary Learning Image Codec

In this section, we present a dictionary-based image coding framework. Our proposed approach aims to leverage the benefits of dictionary learning in achieving high-quality image compression.

The image coding framework depicted in Figure 4.7 consists of two primary stages: a dictionary learning stage and a second stage for quantisation and entropy coding. To begin, the input image is partitioned into non-overlapping patches, and the mean values of these patches (DC components) and the AC components are encoded separately. The AC components are represented based on the dictionary and the sparse codes learned with the autoencoder framework. In contrast, the DC elements and sparse coefficients are subjected to entropy coding using the Huffman encoding algorithm to achieve efficient compression.

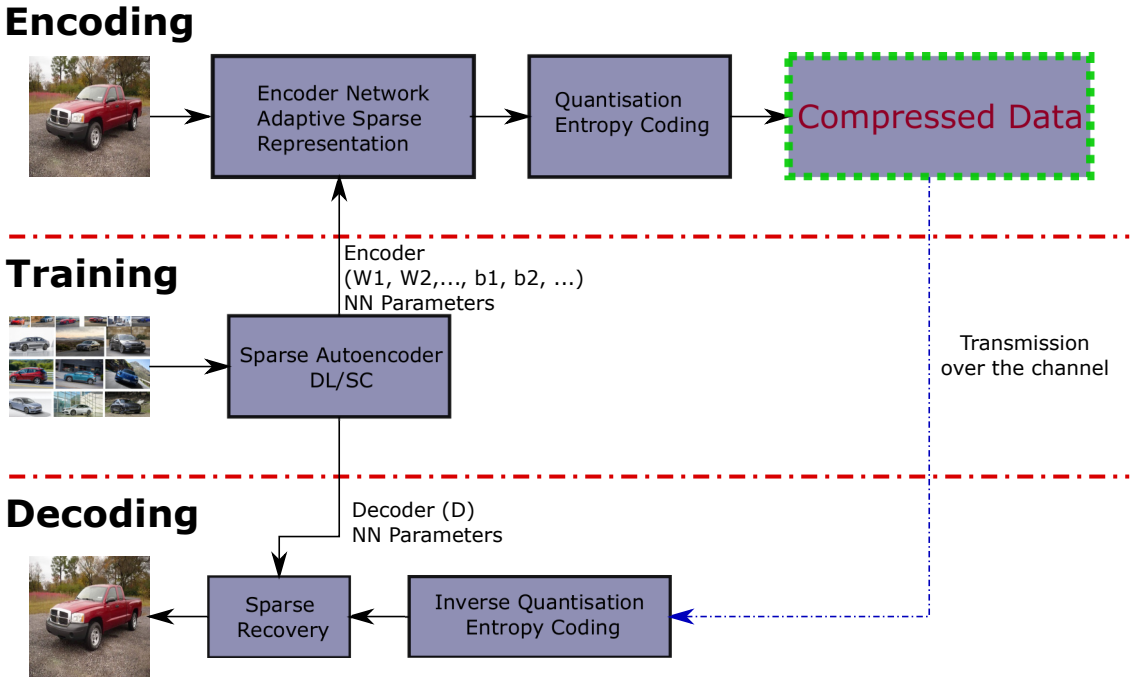


Figure 4.7 – Block diagram of the dictionary-based image coding framework.

In future investigations, we plan to compare the reconstruction quality and compression rates achieved by our proposed framework with those obtained from other commonly used methods, such as the Fourier transform, Wavelet

transform, and other dictionary-based approaches for image and video compression introduced in Section 4.3.1. By conducting such comparisons, we can gain deeper insights into the potential of our proposed approach and its effectiveness in addressing the challenges of image compression in a variety of contexts.

To provide a more detailed comparison that considers these factors in the future, we can gain a deeper understanding of the strengths and weaknesses of each method by

1. Comparing the output images or videos generated by each method and measuring their quality using additional objective metrics such as the Structural Similarity Index (SSIM).

2. Evaluating the performance of each method on different types of images or videos. For instance, some methods may work better on natural images while others may be more suitable for compressing medical images or surveillance footage.

3. Considering the computational complexity and memory requirements of each method. Some methods may require more computational resources or memory than others, which could impact their practical use in real-world application.

Chapter 5

Discriminative Dictionary Learning Algorithms for SSVEP-BCIs

Brain-Computer Interface (BCI) is a computer-based system that has the potential of allowing for a more direct connection between human and computers. Typically, BCI systems employ electrical signals generated by the brain to control specific devices according to the user's intention. However, converting Electroencephalogram (EEG) signals into commands that computers can understand is a complex problem. To do that, it is often necessary to jointly optimise many system parameters, from signal preprocessing to classification. In this chapter, we deal with EEG-based BCIs relying on Steady-State Visually Evoked Potentials (SSVEP), which are generally considered a particularly robust setting for EEG-based BCI applications (PFURTSCHELLER *et al.*, 2010).

An efficient SSVEP-based BCI system greatly depends on selecting and tuning suitable algorithms at several intermediate steps, such as signal filtering, artefact rejection, feature extraction, feature selection and classification. But choosing and tuning the best algorithms and parameters of the BCI configuration is a multi-objective parameter selection problem.

In (OIKONOMOU *et al.*, 2016), the authors examined several algorithms that are widely used in their respective domains as an effort to achieve

the state-of-the-art framework for SSVEP signals collected in the context of a research action named Multimodal Authoring using your Eyes and Mind (MAMEM) (BOSTANTJOPOULOU *et al.*, 2020).

The final decision step in the SSVEP BCI system is performed by a classification method. Support-Vector Machines (SVM) and Linear Discriminant Analysis (LDA) are the most popular classifiers among the SSVEP community and have been used in numerous works (CARVALHO *et al.*, 2015) (SINGLA; HASEENA, 2014). Furthermore, other methods, such as neural-networks (NN), have also been widely used in the literature (PAN *et al.*, 2023).

The state-of-the-art supervised discriminative dictionary learning methods encourage the sparse representation coefficients to have a small within-class dispersion, but a large between-class dispersion. As a result, the classification performance is not impressive on datasets with many classes and high dimensionality (SUN *et al.*, 2014a) (HUANG; ZHANG, 2010) (LIN *et al.*, 2018).

However, this criterion only partially addresses other essential characteristics of the signal of interest, limiting their classification performances. Therefore, one should add more elements to the learning criteria to mitigate this weakness, including class-shared, class-specific and disturbance components.

To overcome this drawback, in this chapter we propose a novel supervised discriminative dictionary learning and sparse coding method based on the combination of two neural network models, both connected in the same cost function. The first model is an autoencoder that learns how to sparsely encode the inputs from a linear combination of dictionary atoms to minimise the representation error. The second model is a classifier which uses the residual vector from the sparse stage to classify the input signals. The framework encourages distinguishable patterns at the residual vector calculated from sparse representations. The proposed approach spans into three main objectives including the sparseness of the encoding, the minimisation of the representation error achie-

ved from the learned dictionary, and the encouragement of discriminative power achieved from the sparse representation.

Furthermore, traditional methods for SSVEP signal classification may face challenges such as limited representation power or difficulties in handling complex and high-dimensional data. Therefore, sparse coding and discriminative dictionary learning may offer alternative approaches that can potentially address these limitations and provide more robust and accurate classification results.

Although different signal categories are associated with class-specific features, they often share some features. Unlike other state-of-the-art methods, the proposed approach learns a set of shared and class-specific features without explicit constraints to the dictionary structure or the sparse coefficients. The framework is characterised as a gradient method for solving discriminative dictionary learning and sparse coding problem through implicit differentiation and backpropagation.

In the context of BCI, the essence of this study is to sparsely encode the input samples from the EEG signals using a dictionary learned for representing these samples accurately while providing distinguishable residual patterns from their sparse representation. In our case study, experimental results are demonstrated using the MAMEM database 1, which consists of the 256-channel EEG signals of 11 subjects.

5.1 Introduction to EEG Signals

An EEG signal is an indirect measurement of currents generated by the activity of groups of neurons in the cerebral cortex. When the brain cells (neurons) are activated, the synaptic currents are produced and propagate through the dendrites. This current generates a magnetic field measurable by EMG machines and a secondary electrical field over the scalp measurable by EEG systems (SANEI; CHAMBERS, 2021). The EEG signals are often collected using

scalp electrodes on the surface of the head.

Although there are other ways of monitoring brain activity and functional monitoring, (e.g., through functional magnetic resonance imaging - fMRI- or magnetoencephalography - MEG), EEG remains the main functional brain scanning modality as it is cheap, portable, and widely available. It represents a viable way for diagnosis of many neurological disorders and other abnormalities in the human body. EEG signals may be used for investigation of various clinical conditions (ADELMAN *et al.*, 1987; TEPLAN *et al.*, 2002).

The 10-20 EEG setting system is a widely used method to describe the location of scalp electrodes in the context of an EEG experiment. It is recommended by The International Federation of Societies for Electroencephalography and Clinical Neurophysiology to a setting with 21 electrodes. This setting aims to ensure reproducibility of different experiments performed on different subjects, and it is based on the relationship between the locations of neighbour electrodes and the underlying area of cerebral cortex. The **10** and **20** refer to the distances between adjacent electrodes, which can be either 10% or 20% of the total front-back or right-left distance of the skull.

The correlation between the EEG signal and the underlying brain area depends on the accuracy of the electrodes placement. Each site has a letter to identify the lobe and a number or another letter to identify the hemisphere location. The notation is shown in Figure 5.1. The letters F, T, C, P, and O stand for Frontal, Temporal, Central, Parietal and Occipital, respectively. Although there is no *central lobe* in the brain, this notation is used to facilitate identification and keep it intuitive. The even numbers $(2, 4, 6, 8)$ refer to the right hemisphere while the odd numbers $(1, 3, 5, 7)$ refer to the left hemisphere. The letter z refers to an electrode placed on the midline. The smaller the number used on the identification of an electrode, the closer its position to the midline.

Upon different EEG recording setups where a larger number of elec-

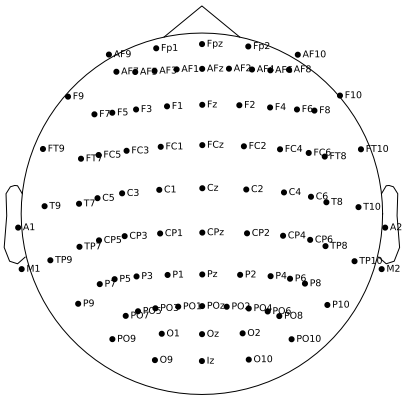


Figure 5.1 – A diagrammatic representation according to the international 10-20 electrode setting system ($94 + 3$ locations).

trodes is available, the configuration of the sensors follows the 10-20 system placing the additional electrodes equidistantly between the ones from conventional setup in Figure 5.1. An example of a setup with 256 electrodes is shown in 5.2.

Traditionally, many brain disorders are diagnosed by visual inspection of EEG signals. The clinical experts in the field are expected to be familiar with a typical manifestation of brain rhythms in the EEGs. However, there are various challenges in the interpretation of the signal which are mostly due to specificities in its nature. In healthy adults, the amplitudes and frequencies of such signals may change significantly from one human to human. The characteristics of the waves also change with age. There are five major brain waves distinguished by their different frequency ranges. These frequency bands from low to high frequencies are called alpha (α), theta (θ), beta (β), delta (δ) and gamma (γ), respectively.

Longer latency responses are related to higher cognitive functions such as *event-related potentials* (ERPs). It is a measure of brain response that is the direct result of a specific sensory, cognitive, or motor event, or any stereotyped electrophysiological response to a stimulus (SANEI; CHAMBERS, 2021). ERPs are useful diagnostic indicators in many applications to neurology as well as for

BCI applications. These signals are characterised by their spatial, temporal, and spectral locations, which are translated by their amplitudes, latencies, source locations, and frequency contents. However, extraction and classification of these signals usually require great expertise in the development of mathematical and signal processing algorithms.

The SSVEP signals are examples of ERPs. They are a natural response to visual stimulations at specific frequencies. When the retina is excited by a visual stimulus ranging from 3.5 to 75Hz, the brain generates an electrical activity at the same (or at multiples of the) frequency of the visual stimulus. These manifestations are used for understanding which stimulus the subject is looking at, in the case of stimuli with different flashing frequencies. The SSVEP is one of the five categories of the electrical activities used in BCI: the α and β rhythms, the P300 evoked potentials, the visual N100 and P200, and the SSVEP.

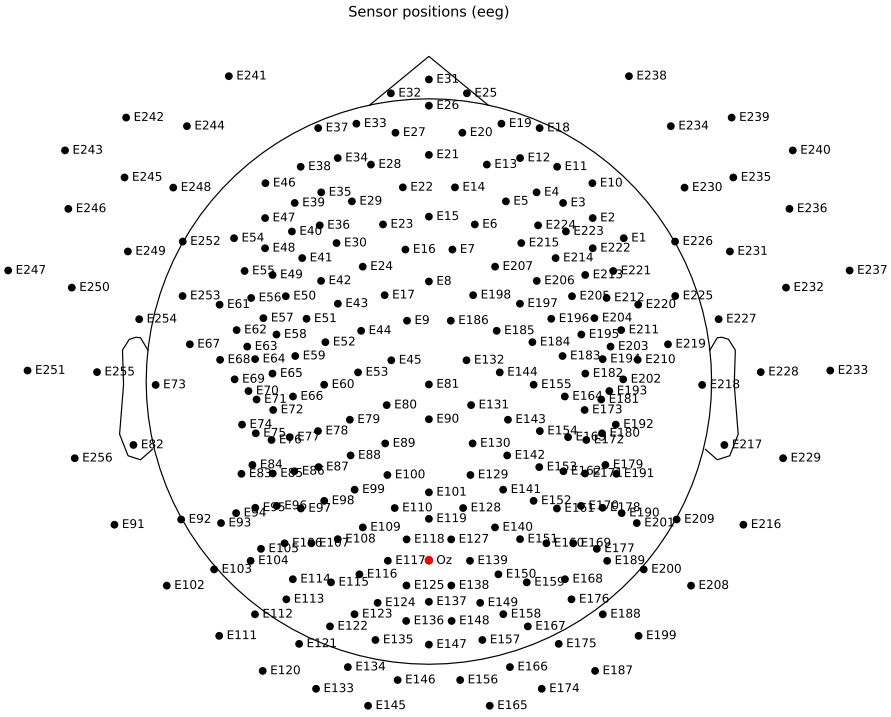


Figure 5.2 – The positions of 256 EEG electrodes used for data acquisition are marked by black dots. The place of the occipital channel Oz is highlighted in red.

5.2 Problem Formulation

To formulate the problem, we can assume the protocol for SSVEP acquisition is performed on N subjects, for which we can present N_v visual stimuli with coloured light flickering at either one of the S different frequencies $F = \{f^1, \cdots, f^S\}$ for a fixed time duration. This period is referred to as a trial $t_i, i = 1, \cdots, N_v$. The EEG signals captured at each trial t_i of each subject s_j are presented by $E(t_i, s_j), j = 1, \cdots, N$. The label that we want to predict is the frequency $f^k \in F$ presented in a certain trial. The prediction is based on the signals $E(t_i, s_j)$, captured during the corresponding time interval.

To this end, conventional BCI systems based on SSVEP usually start with the preprocessing step, where it applies filtering and artefact removal methods to the EEG signals. Afterwards, the signals are typically transformed into the frequency domain from which a set of features $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_{N_d}\}$ is extracted. The notation N_d represents the total number of trials for all subjects. Optional steps may also include feature selection or dimensionality reduction techniques that can be applied to these features.

After performing the aforementioned processing steps, we obtain a labeled dataset $D = \{\mathbf{x}_v, f_v\}, v = 1, \cdots, N_d$ and $f_v \in F$. The dataset is then employed in the learning and testing phases of the classification model chosen for the BCI system. In particular, the Leave-One-Subject-Out is employed here as a cross-validation scheme from which the dataset D is split into train T, and test set T^*, such that $\|D\| = \|T\| + \|T^*\|$. For further details on the Leave-One-Subject-Out, see Section 5.7.

Ultimately, given the labeled training set $T = \{x_t, f_t\}, t = 1, \cdots, N_t$ where $N_t = \|T\|, f_t \in F$, the main goal of the BCI system can be translated as learning a model to estimate a score that indicates whether the stimulus flickering at frequency $f^k \in F$ is the source of an EEG signal included in the test set $T^* = \{x_t^*, f_t^*\}, t^* = 1, \cdots, N_t^*$ where $N_t^* = \|T^*\|, f_t^* \in F$.

5.3 Preprocessing

Several methods have been applied in the preprocessing stage of the SSVEP-BCI systems. The main goal of this filtering is to attenuate the interference that is added to the EEG signal during the recording. These artefacts compromise the quality of the signal and affect the BCI performance. Usually, the first step of different preprocessing approaches involves a bandpass filter to keep the desired parts of the EEG signal. For filtering the signal spectrum, we can rely on either of two filtering categories: Finite Impulse Response (FIR) filters and Infinite Impulse Response (IIR) filters. FIR filters have an impulse response of finite duration, while IIR filters have an impulse response of infinite duration. In the literature, we can find several works (YIN *et al.*, 2020; AL-MUHAMMADI *et al.*, 2015) related to EEG filtering approaches with IIR and FIR filter banks. FIR filters are stable and usually present linear phase, i.e., all frequency components of the EEG signal would be shifted in time (usually delayed) by the same constant amount. On the other hand, IIR filters are not always stable and present nonlinear phase characteristics. However, the latter requires fewer coefficients than FIR filters, making it a more suitable choice specially when memory constraints are a critical aspect of the solution and when some phase distortion is tolerable.

Aside from classical time-domain filtering approaches, the Common Averaging Re-referencing (CAR) spatial filtering method is also employed in many works (CARVALHO *et al.*, 2015) to remove unwanted signal components such as those from eye blinks, eyes movement, and other muscle activities. The CAR method relies on a reference signal that should be optimally influenced by the same noise as that from the channel of interest. When subtracting this reference from the channel of interest, this noise is subtracted as well. For picking up consistent noise references, the channel(s) ultimately forming the new reference should hence be located close enough to the region of interest. Unfortunately,

there is no consensus on the *optimal reference* although two commonly employed referencing schemes are (i) the average mastoids (the mean mastoids reference signal has been modelled by the average of the TP9 and TP10 electrodes that are located in the proximity of the mastoids), and (ii) the average of all scalp channels, usually referred to as Common Average Reference (it is obtained from the average electrical activity measured across all scalp channels).

The AMUSE (TONG *et al.*, 1990) approach is a typical method applied to EEG signals in order to remove specific artefacts. It is a blind source separation algorithm that has some similarities with standard Principal Component Analysis (PCA). AMUSE consists of two separate steps based on second-order statistics and spatio-temporal decorrelation algorithms. Let us denote $Y = [\mathbf{y}_1, \cdots, \mathbf{y}_N] \in \mathcal{R}^{M \times N}$ as the signal matrix captured from all N sensor channels during a certain trial containing M samples in each sensor channel. The vector $\mathbf{y}(j) \in \mathcal{R}^N$ is the j-th row of the matrix Y, containing the observations from all sensor channels and $j = 1, \cdots, M$. The covariance matrix of the observations is calculated as $C_Y = E[YY^T] \in \mathcal{R}^{N \times N}$. AMUSE aims at decomposing the observation $\mathbf{y}_j$ into uncorrelated sources $\boldsymbol{z}(j) \in \mathcal{R}^N$, i.e., $\boldsymbol{z}(j) = W\mathbf{y}(j) \in \mathcal{R}^N$ and W is the unmixing unknown matrix. As a result of that, the observations can be then reproduced by linearly combining the uncorrelated sources, i.e., $\mathbf{y}(j) = A\boldsymbol{z}(j)$ and A is the unknown mixing matrix.

To obtain such uncorrelated sources, the first step of AMUSE is to perform the linear transformation known as whitening (ROMANO *et al.*, 2018) in order to transform the observations $\mathbf{y}(j)$, with known covariance matrix, into a set of new variables $\boldsymbol{z}(j)$, whose covariance is the identity matrix meaning they are uncorrelated and each has a unitary variance. Most commonly, the matrix $Q = C_Y^{-\frac{1}{2}}$ is adopted for the whitening transformation given by $\boldsymbol{z}(j) = Q\mathbf{y}(j)$.

The second step consists of applying Singular Value Decomposition (SVD) to a time-delayed covariance matrix of the whitened data $\boldsymbol{z}(j)$, i.e.,

$E[ZZ^T] = U\Lambda V$. The unmixing matrix is then estimated as $W = U^T Q$ and the uncorrelated sources are then estimated as $\hat{Z} = WY$. The AMUSE allows obtaining a rank of the uncorrelated components based on the singular values from SVD. It is well established (CHOI *et al.*, 2005) that typical sources of EEG noise such as eye blinks usually lie in the few first and last components generated from AMUSE. As a matter of fact, these components can be rejected before returning to the original signal space, such that eye blinks and other muscle activities are eliminated from the signal.

Another category of methods for artefact removal relies on the use of Independent Component Analysis (ICA). It assumes that each vector $\mathbf{y}(j)$ is a linear mixture of K unknown sources, i.e., $\mathbf{y}(j) = A\boldsymbol{z}(j)$, where the matrix of mixing coefficients A is unknown. The goal in ICA is to find the sources $\boldsymbol{z}_k(j), k = 1, \cdots, K$ and $j = 1, \cdots, N$, that are as independent as possible according to an information-theoretic cost function such as minima of Kullback-Leibler divergence or maximisation of cumulants (CHOI *et al.*, 2005). The motivation for such a criterion is that the independence of random variables is a more general concept than decorrelation (used by AMUSE). The random variables z_i and z_j are said to be statistically independent if knowledge of the values of z_i provides no information about the values of z_j. It should be noted that ICA can perform blind source separation as it estimates the true sources only if they are all statistically independent and there is a maximum of a single Gaussian source.

The application of the ICA algorithm to EEG data processing is usually performed in three steps: (i) the EEG data are decomposed into independent components; (ii) by visual inspection some of these components are excluded due to the artefacts, and finally (iii) the artefact-free EEG signals are obtained by mixing and projecting back onto the original channels those ICA components without artefacts.

Finally, the Common Spatial Patterns (CSP) method is commonly reported in the literature as a preprocessing technique applied to EEG signals. Given a classification problem consisting of two different classes, the classical CSP algorithm seeks an optimal projection direction (spatial filter) by maximising the variance of one class and simultaneously minimising the variance of the other class. Raw EEG signals are then mapped to this direction (YU *et al.*, 2019) and the classification problem becomes easier to solve. The projection is a spatial filtering process, which was initially designed for binary classification problems. However, it has also been extended to multi-class problems (DORNHEGE *et al.*, 2004).

For the experimental results presented in this chapter, the preprocessing module of the proposed SSVEP-based BCI system includes an IIR-Chebyshev I bandpass filter. To evaluate the effects of the frequency and magnitude specifications of the filter, an exhaustive search on various combinations of stopband and passband parameters was performed while monitoring the classification accuracy achieved from an arbitrary BCI configuration.

The algorithms and corresponding parameters of such arbitrary configuration result in specific levels of accuracy which serve as a comparison basis to decide whether the preprocessing algorithm and corresponding parameters introduce improvements to the BCI system. The feature extraction method and the classification algorithm of the arbitrary configuration are the Welch Power Spectrum Density estimation method and the Support Vector Machine, respectively. For feature extraction, the power spectrum of Welch's method is used with the frequency range applied to the entire spectrum, the size of the FFT is set to 512 samples, the frequency range is between 0 and 125Hz, the length of each segment is set to 156, and the overlap is 78 samples. The SVM classifier is configured with a linear kernel and the cost parameter C is set to 1, as reported in (OIKONOMOU *et al.*, 2016). In this case, a value of $\mathsf{C} = 1$ implies a balanced approach, where both margin maximisation and error minimisation

are considered important.

After conducting an exhaustive search of various types of bandpass filters, including the FIR Least Squares, the IIR Chebyshev I, the IIR Chebyshev II, and the IIR Butterworth, along with their respective frequency and magnitude specifications, it was determined that the IIR Chebyshev I filter outperforms the others and is therefore the chosen filtering method for integration into the optimal configuration proposed in this chapter. The specifications of the filter are as follows: Stopband Frequency 1 = 3Hz, Passband Frequency 1 = 5Hz, Passband Frequency 2 = 48Hz, Stopband Frequency 2 = 58Hz, Stopband Attenuation 1 = 50dB, Stopband Attenuation 2 = 50dB, and Passband Ripple 1 = 0.4dB.

Several experiments using AMUSE with the 256 EEG channels were also performed in the artefact removal step. After different combinations, we concluded that, although AMUSE resulted in an improvement of approximately 2% compared to the accuracy obtained from the BCI configuration with the IIR Chebyshev I filter only, it significantly increased the total execution time of the experiment. As our primary goal is to keep the preprocessing modules as simple as possible and focus on the decision step involving the signal classifier, the AMUSE was not incorporated into the adopted configuration.

5.4 Discriminative Dictionary Learning and Sparse Coding Methods

In this section, we introduce the state-of-the-art methods for discriminative dictionary learning and sparse coding. We describe some representative state-of-the-art methods such as the SRC (WRIGHT *et al.*, 2008), the DLSI (RAMIREZ *et al.*, 2010), the COPAR (KONG; WANG, 2012), the Fisher Discriminative Dictionary Learning (YANG *et al.*, 2011) and the Low-rank shared Dictionary Learning (VU; MONGA, 2016). In particular, we address their formulations, strengths, weaknesses and constraints to encourage certain featu-

res of the dictionaries.

In sparse representations, signals are expressed as a linear combination of the atoms taken from a dictionary. The sparse representation classifier (SRC) (WRIGHT *et al.*, 2008) was originally developed for face recognition. Later on, it was adapted to various signal and image classification problems. The central idea of the SRC is to represent a test sample as a linear combination of samples from the available training samples set. Sparsity arises because most of the nonzero coefficients in the linear combination correspond to the atoms that are similar to the test sample. The SRC assumes that each sample lies in its class subspace and that all class subspaces are non-overlapping.

Given C classes and a dictionary $D = [D_1, \cdots, D_C]$ with D_c corresponding to the training samples from the class c, $c = 1, ..., C$, a new sample $\mathbf{y}$ that belong to class c is expected to be represented as $\mathbf{y} \approx D_c\mathbf{x}$. Expressing $\mathbf{y}$ in terms of D results in $\mathbf{y} \approx D_1\mathbf{x}_1 + D_2\mathbf{x}_2 + \cdots + D_c\mathbf{x}_c + D_C\mathbf{x}_C$. However, note that most of the active elements of $\mathbf{x}$ should be located in $\mathbf{x}_c$ and thus, the coefficient vector x becomes sparse.

In matrix form, assuming $Y = [Y_1, \cdots, Y_c, \cdots, Y_C]$ is a set of samples where Y_c comprises those samples that belong to class c, the sparse matrix X would be, in the best fit, a block diagonal matrix as shown in Figure 5.3 (VU; MONGA, 2016). In general, learning a dictionary from specific training samples instead of using all of them as a dictionary enhances the performance achieved from SRC.

However, the assumption of non-overlapping subspaces between different classes in SRC is often unreal in most applications where many common features are shared. This problem has been partially addressed by recent efforts such as the DLSI and the COPAR methods.

In DLSI, the main objective is to learn a set of dictionaries that effectively represent each class. Unlike the standard SRC method where classes

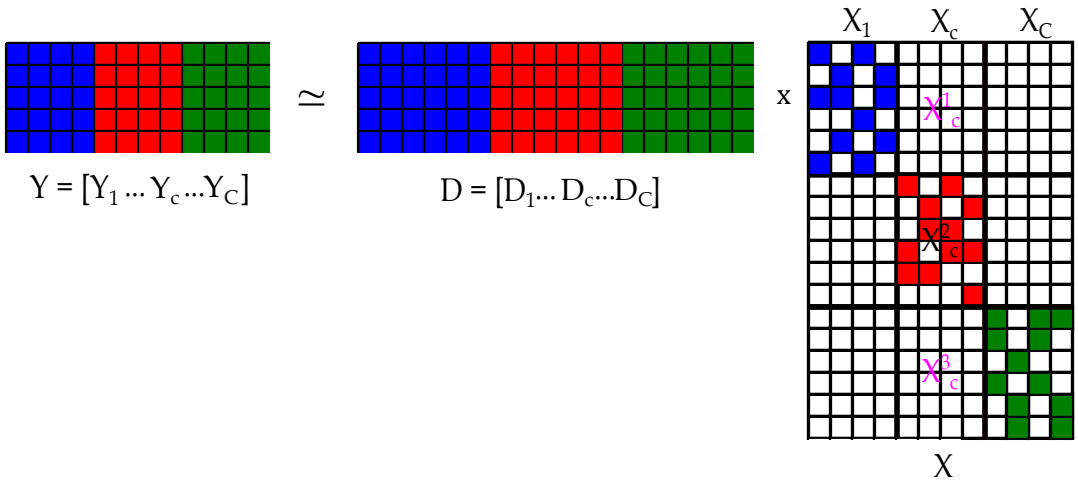


Figure 5.3 – Typical structure of the dictionary and the sparse matrix in SRC framework.

can have intersecting subspaces, DLSI promotes incoherence between the dictionaries of each class. This encourages independence between the bases of different classes by minimising coherence between cross-class bases. Although DLSI does not explicitly learn shared features, it aims to enhance discriminability by ensuring that the sub-dictionaries of each class capture distinct and non-overlapping features. Therefore, DLSI is a more refined approach compared to SRC, as it focuses on improving representation quality and encouraging structured incoherence among the learned dictionaries. The cost function $J(D, X)$ in DLSI is defined as:

$$J(D, X) = \sum_{c=1}^{C} \left(\|Y_c - D_c X^c\|_F^2 + \lambda \|X^c\|_1 + \frac{\eta}{2} \sum_{j=1,\ j\neq c}^{C} \left\| D_j^T D_c \right\|_F^2 \right) \tag{5.1}$$

Each class-specific dictionary D_c is updated by fixing others and solving the following equation

$$D_c = \arg\min_{D_c} \|Y_c - D_c X^c\|_F^2 + \eta \|A D_c\|_F^2 \tag{5.2}$$

with $A = [D_0, \cdots, D_{c_1}, D_{c+1}, \cdots, D_C]$. The solution for this problem updates

each column $\mathbf{d}_{c,j}$ of D_c one by one based on the following

$$\mathbf{u} = \left(\left\|\mathbf{x}_c^j\right\|_2^2 I + \eta A^T A\right)^{-1} \left(Y_c - \sum_{i \neq j} \mathbf{d}_{c,i}\mathbf{x}_c^i\right) \mathbf{x}_c^j \tag{5.3}$$

$$\mathbf{d}_{c,j} = \frac{\mathbf{u}}{\|\mathbf{u}\|_2^2} \tag{5.4}$$

To update sparse matrix X, in each iteration, DLSI solves C subproblems:

$$X_c = \arg\min_{X^c} \|Y_c - D_c X^c\|_F^2 + \lambda \|X^c\|_1 \tag{5.5}$$

On the other hand, the COPAR method can explicitly learn a shared sub-dictionary D_{C+1} with a common pattern pool (the commonality) and class-specific sub-dictionaries (the particularity) for classification. It is built under the empirical observation that images from different categories usually share some common patterns which are not helpful for classification but essential for representation. Note that the subspace spanned by the columns of the shared dictionary D_{C+1} must have a low rank. Otherwise, class-specific features would be also represented by the shared dictionary. In the worst case, the shared dictionary span would include all the features of each class and this would greatly harm the classification accuracy (VU; MONGA, 2017).

In DLSI, as the bases of the common patterns (shared features) may appear in several particularities (specific classes), the learned particularities may become redundant and less discriminative. Therefore, COPAR drives the common patterns to the commonality set and preserves the class-specific features in the particularity sets of the dictionary. To this end, it adds an incoherence term $\mathcal{Q}(D_i, D_j) = \left\|D_i^T D_j\right\|_F^2$ to its objective function. Although this penalty term has been used among the class-specific sub-dictionaries in Equation (5.1) of DLSI, COPAR method also deals with the incoherence of the commonality concerning the particularities.

To define the objective function of COPAR, let us denote the dictionary as $D = [D_1, \cdots, D_c, \cdots, D_{C+1}] \in \mathcal{R}^{M \times N}$, in which $N = \sum_{c=1}^{C+1} N_c$, $D_c \in$

$\mathcal{R}^{M\times N_c}$ stands for the particularity of the c-th class and $D_{C+1} \in \mathcal{R}^{M\times N_{C+1}}$ is the commonality. In addition, $Q_c = [\mathbf{q}_c^1, \cdots, \mathbf{q}_c^j, \cdots, \mathbf{q}_c^{N_c}] \in \mathcal{R}^{N\times N_c}$ is the selection operator in which the j-th column of Q_c is of the form (KONG; WANG, 2012)

$$\mathbf{q}_c^j = \left[\underbrace{0, \cdots, 0}_{\sum_{m=1}^{c-1} N_m}, \underbrace{0, \cdots, 0, 1, 0, \cdots, 0}_{N_c}, \underbrace{0, \cdots, 0}_{\sum_{m=c+1}^{C+1} N_m}\right]^T \tag{5.6}$$

The operators

$$Q_{/c} = [Q_1, \cdots, Q_{C_{c-1}}, Q_{c+1}, \cdots, Q_C, Q_{C+1}]$$

and

$$\tilde{Q}_{/c} = [Q_1, \cdots, Q_{C_{c-1}}, Q_{c+1}, \cdots, Q_C]$$

are both included in the objective function to force the coefficients, except that corresponding to the c-th particularity and those from commonality, to be zero. Finally, note that $Y = [Y_1, \cdots, Y_c, \cdots, Y_C] \in \mathcal{R}^{M\times P}$ is the dataset, wherein $Y_c \in \mathcal{R}^{M\times P_c}$, given $P = \sum_{c=1}^{C} P_c$, represents the data from the c-th class. To encourage sparsity the term $\phi(X_c) = \sum_{i=1}^{P_c} \left\|\mathbf{x}_c^i\right\|_1$ is used, for $X_c = [\mathbf{x}_c^1, \cdots, \mathbf{x}_c^i, \cdots, \mathbf{x}_c^{P_c}] \in \mathcal{R}^{N\times P_c}$. COPAR's notation framework can be visualised in Figure 5.4. The objective function of COPAR method is given by

$$\begin{aligned} J(D, X) = & \sum_{c=1}^{C} \left(\|Y_c - DX_c\|_F^2 + \left\|\tilde{Q}_{/c}^T X_c\right\|_F^2 + \left\|Y_c - D\tilde{Q}_c Q_c^T X_c\right\|_F^2 + \lambda\phi(X_c) \right) + \\ & \eta \sum_{c=1}^{C+1} \sum_{j=1, j\neq c}^{C+1} \mathcal{Q}(D_c, D_j) \end{aligned}$$

As a supervised Dictionary Learning method, the Fisher Discrimination Dictionary Learning (FDDL) method learns class-specific dictionaries for each class and makes them most discriminative through Fisher criteria (YANG *et al.*, 2011). Let $\mathbf{m}$, $\mathbf{m}_c$ be the mean vector of X, X_c, respectively. Let $M_c = [\mathbf{m}_c, \cdots, \mathbf{m}_c] \in \mathcal{R}^{N\times P_c}$, and $M = [\mathbf{m}, \cdots, \mathbf{m}]$, with number of columns, be

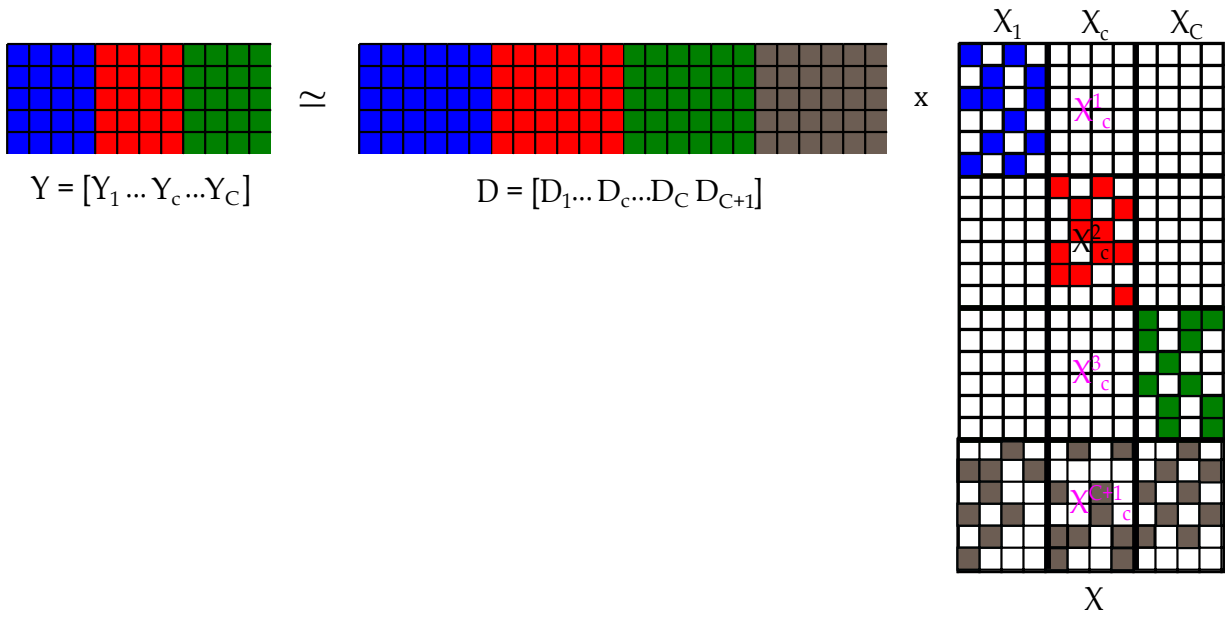


Figure 5.4 – Typical structure and notation adopted in the COPAR framework. Brown items indicate shared/common patterns whilst red, green and blue items from the dictionary and the sparse codes indicate class-specific patterns.

the mean matrices. In particular, the discriminative dictionary D and the sparse coefficient matrix X are learned based on minimising the following cost function

$$J(D, X) = \frac{1}{2} \sum_{c=1}^{C} r(Y_c, D, X_c) + \lambda_1 \|X\|_1 + \frac{\lambda_2}{2} f(X) \tag{5.7}$$

where

$$r(Y_c, D, X_c) = \|Y_c - DX_c\|_F^2 + \|Y_c - D_c X_c^c\|_F^2 + \sum_{i \neq c} \left\|D_i X_c^i\right\|_F^2 \tag{5.8}$$

and

$$\sum_{c=1}^{C} r(Y_c, D, X_c) \tag{5.9}$$

is the discriminative fidelity term. The dictionary D can be written as $D = [D_1, D_2, ..., D_C]$, where D_i is the sub-dictionary from the c-th class, and X_c^j is the coding coefficient of Y_c over the sub-dictionary D_i. The first term $\|Y_c - DX_c\|_F^2$ indicates that the dictionary D must represent Y_c accurately. The second term $\|Y_c - D_c X_c^c\|_F^2$ encourages the intra-class representation ability, that is, Y_c should be well represented by D_c but not by D_j, $j \neq c$. Furthermore, X_c^c should have

some significant coefficients such that the representation error is minimised, while $X_c^i,\ i \neq c$ should have nearly zero coefficients, such that the third term $\sum_{i \neq c} \left\| D_i X_c^i \right\|_F^2$ is small. This denotes the inter-class discriminative performance.

The term $\lambda_1 \|X\|_1$ encourages the sparsity of coefficients in X. Finally, the term $f(X) = \sum_{c=1}^{C} (\|X_c - M_c\|_F^2 - \|M_c - M\|_F^2)$ in Equation (5.7) is the Fisher-based discriminative regularisation. The first component,

$$\sum_{c=1}^{C} (\|X_c - M_c\|_F^2 - \|M_c - M\|_F^2) \tag{5.10}$$

calculates the Euclidean distances between the sparse codes X_c and their class means M_c. This component encourages the sparse codes of the same class to be close to their respective class means, promoting intra-class compactness. The second component, $\|M_c - M\|_F^2$, measures the distance between the class means M_c and the overall mean M of the data. It encourages the class means to be well separated, promoting inter-class separability.

The minimisation problem in Equation (5.7) is solved by alternatively optimising each X_c or D_c while fixing the other variables. Detailed optimisation procedures are presented in (YANG *et al.*, 2011). This method leads to an extremely slow convergence which is sometimes impractical for multi-class high-dimension problems. A prediction label is obtained based on

$$L(y) = \arg\min_i \left\| y - D_i \alpha^i \right\|_F^2 + \mu \left\| \alpha^i - m_i \right\|_2^2 \tag{5.11}$$

where α^i is the sparse coefficient vector associated with the i-th class. The first term is the reconstruction error of class i, the second term is the distance between the coefficient vector α^i and the mean vector m_i.

The Low-rank Shared Dictionary Learning (LRSDL) framework (VU; MONGA, 2016) is a generalised version of the FDDL with the additional capability of capturing shared features, resulting in better classification performance. With the additional shared dictionary D_{C+1}, it is expected that Y_c can be better

represented in the collaboration with the class-specific dictionary D_c, as illustrated in Figure 5.4. In practice, the discriminative fidelity term $r(Y_c, D, X_c)$ in Equation (5.7) can be extended to $\bar{r}(Y_c, \bar{D}, \bar{X}_c)$ defined as follows

$$\bar{r}(Y_c, \bar{D}, \bar{X}_c) = \left\|Y_c - \bar{D}\bar{X}_c\right\|_F^2 + \left\|Y_c - \bar{D}_c\bar{X}_c^c - \bar{D}_{C+1}\bar{X}_c^{C+1}\right\|_F^2 + \sum_{i\neq c}\left\|D_i X_c^i\right\|_F^2 \quad (5.12)$$

Similarly, the Fisher-based discriminative coefficient term $f(X)$ is extended to $\bar{f}(\bar{X})$ defined as follows

$$\bar{f}(\bar{X}) = f(X) + \left\|X^{C+1} - M^{C+1}\right\|_F^2 \quad (5.13)$$

where the term $\left\|X^{C+1} - M^{C+1}\right\|_F^2$ encourages the coefficients of X in all training samples represented via the shared dictionary, D_{C+1}, to be similar. Additionally, for the shared dictionary, LRSDL constrains the $rank(D_{C+1})$ to be small, using the nuclear norm $\|D_{C+1}\|_*$. The reason for that is to avoid adding class-specific features to the shared dictionary. The cost function $\bar{J}(\bar{D}, \bar{X})$ of the LRSDL is given by the following Equation

$$\bar{J}(\bar{D}, \bar{X}) = \frac{1}{2}\sum_{c=1}^{C}\bar{r}(Y_c, \bar{D}, \bar{X}_c) + \lambda\left\|\bar{X}\right\|_1 + \frac{\lambda_2}{2}\bar{f}(\bar{X}) + \eta\left\|D_{C+1}\right\|_* \quad (5.14)$$

Minimising $\bar{J}(\bar{D}, \bar{X})$ jointly finds the appropriate dictionaries and sparse codes.

Table 5.1 provides a comprehensive comparison of various aspects of the discriminative dictionary learning methods discussed in this section. It explores these methods based on the following factors:

- Intra-class Diversity: the atoms of a class-specific sub-dictionary should have low inner-product.
- Inter-class Separability: the class-specific sub-dictionaries should have low inner-product between their atoms.
- Shared and Class-Spec Sub-dictionaries Overlap: the atoms of the shared-features and class-specific sub-dictionaries should have low inner-product.

Table 5.1 – Comparing different aspects of the discriminative dictionary learning methods discussed in the section.

Method	Shared Sub-dict.	Class-Specific Sub-dict.	Intra-Class Diversity	Inter-Class Separability	Shared and Class-Spec. Sub-dict. Overlap
SRC	No	Yes	No	No	N/A
DLSI	No	Yes	No	Yes	N/A
COPAR	Yes	Yes	Yes	No	Low
FDDL	No	Yes	Yes	Yes	N/A
LRSDL	Yes	Yes	Yes	Yes	Low

5.5 Proposed Discriminative Dictionary Learning Framework

In this section, we introduce the proposed Discriminative Dictionary Learning Framework and how it can be applied to SSVEP-based BCI systems. We present a Neural Network-based Discriminative Dictionary Learning and Sparse Coding (NNDDL) framework for representing class-specific and shared features of the signal and distinguishing their sparse representations accordingly. We add no specific constraints to the dictionary structure. The rationale is to encourage dictionary and sparse codes to produce distinguishable patterns of the representation error that our classifier can learn.

Figure 5.5 illustrates the proposed scheme for EEG classification. After the learning process we obtain D and the classifier model function $\mathbf{f}(\mathbf{r}_i) = [\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3, \mathsf{C}_4, \mathsf{C}_5]$, where $\mathbf{r}_i$ is the signed residual vector of the i-th test signal sample $\mathbf{y}_i$, i.e., $\mathbf{r}_i = \mathbf{y}_i - \mathsf{D}\boldsymbol{\alpha}_i$ and the associated sparse vector is $\boldsymbol{\alpha}_i$. The values indicated as C_j are the estimated probabilities of $\mathbf{r}_i$ being the residual vector from the i-th sample belonging to the j-th class, $j = 1, \cdots, 5$. The class identity of the sample $\mathbf{y}_i$ is determined by $\hat{\mathsf{L}}(\mathbf{y}_i) = \arg\max(\mathbf{f}(\mathbf{r}_i))$.

The encoder side of the proposed NNDDL framework is composed of convolutional and 1D upsampling pooling layers with activation functions set to the ReLU. The decoder side linearly reverses the sparse latent variables to the data space without including biases and activation functions. The classifier input is fed with $\mathsf{Y}-\mathsf{DX}$, processing the signed residual between the original data

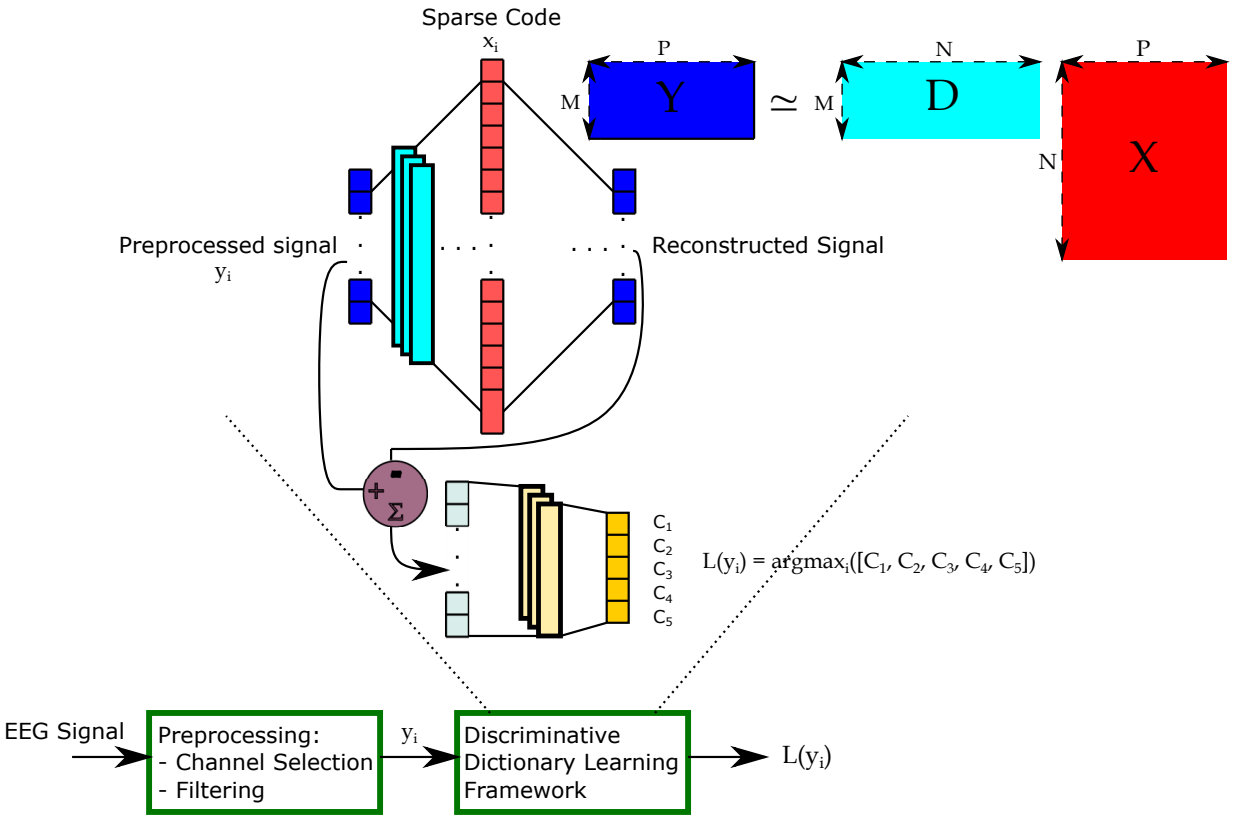


Figure 5.5 – Illustration of the proposed discriminative dictionary learning and sparse coding framework.

space Y and the reconstruction from its sparse representation DX. It comprises convolutional, 1D max-pooling and dense layers with activation functions set to ReLU. The activation function of the classification layer is the Softmax. The architecture of the proposed encoder is depicted in Figure 5.6. During the training phase, dropout layers with rates between 0.1 and 0.4 were added after the model convolutional and dense layers. It aims to regularise the weights and prevent overfitting.

Most dictionary learning methods alternate between a sparse coding stage and a dictionary update stage, each of which may consist of iterations over a set of inner update steps. For instance, the ADMM method is a simple yet powerful approach that decouples optimisation variables and optimises the augmented Lagrangian in a primal-dual scheme. Despite ADMM advantages, due to its closed-form update of all the parameters in the problem formulation,

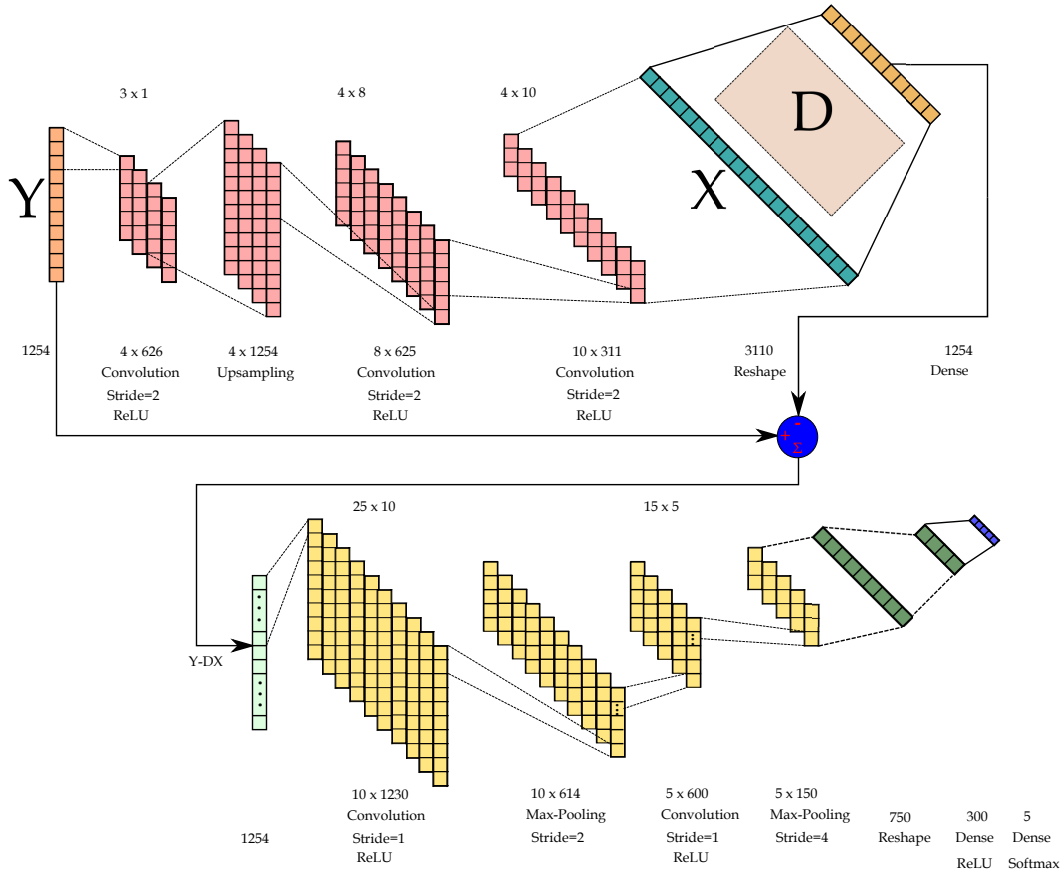


Figure 5.6 – Diagram illustrating the discriminative dictionary learning framework using neural networks, specifically designed for SSVEP signal classification.

it can be complicated to further extend it into more complex problems and larger datasets. The proposed framework learns a discriminative sparse representation using combined models. It uses an autoencoder to learn the features in the dictionary and the sparse codes, and a second model to classify the residual vector calculated from the autoencoder output and input. The combined models

1. encourage the discriminative power of the classifier model fed with the residuals from reconstruction,

2. implicitly encourage a dictionary with class-specific features to enhance the discriminative power of the classifier, and

3. implicitly encourage a dictionary with shared features containing common patterns that do not necessarily contribute to distinguishing signal classes

but rather minimising the representation error at the output of the auto-encoder network.

The cost function proposed to the network architecture illustrated in Figure 5.5 is given as follows

$$\begin{aligned} J(D,X) &= \alpha_1 \|Y - DX\|_F^2 + \alpha_2 \|X\|_{1,1} + \alpha_3 \mathcal{N}(D) + \\ &+ \alpha_4 H(L(\mathbf{y}_i), \hat{L}(\mathbf{y}_i)) + \alpha_5 \sum_{j=1}^{V} \theta_j^2 \end{aligned}$$

The proposed NNDDL method avoids explicit encouragement of pre-structured or pre-designed features in the dictionary, D, and sparse matrix, X. No prior structure is assumed or imposed on these matrices. The constraints applied to $J(D,X)$ do not aim to minimise the rank of sub-dictionaries with shared features or promote incoherence between class-specific sub-dictionaries.

The first term $\alpha_1 \|Y - DX\|_F^2$ is the penalty related to the Euclidean reconstruction error between the data samples Y and the decoded sparse features DX at the output of the same network. The second term encourages the sparsity at the latent space of the autoencoder using the mixed norm $\ell_{1,1}$, i.e., $\|X\|_{1,1} = \left(\sum_{i=1}^{N} \sum_{j=1}^{P} |x_{i,j}|^1\right)^{\frac{1}{1}}$. The third term is

$$\mathcal{N}(D) = \frac{1}{N} \sum_{i=1}^{N} |1 - \mathbf{d}_i^T \mathbf{d}_i|, \tag{5.15}$$

which enforces the unitary normalisation of each dictionary atom. The fourth term stands for the error of the classification model. It represents the cross-entropy function as follows

$$H(L(\mathbf{y}_i), \hat{L}(\mathbf{y}_i)) = -\frac{1}{P} \sum_{i=1}^{P} \left(L(\mathbf{y}_i) \log \hat{L}(\mathbf{y}_i) + (1 - L(\mathbf{y}_i)) \log(1 - \hat{L}(\mathbf{y}_i)) \right) \tag{5.16}$$

where the target vector $\mathrm{L}(\mathbf{y}_i)$ represents the probabilities for all the five classes with respect to the i-th training sample. In particular, it is a one-hot vector, meaning it has 1 on a single position and 0's everywhere else. Similarly, the prediction vector $\hat{\mathrm{L}}(\mathbf{y}_i)$ represents the predicted probabilities of all classes, summing up to 1.

Finally, the fifth term $\sum_{j=1}^{V} \theta_j^2$ is the ℓ_2 regularisation term that takes the sum of all the parameters of the neural network squared, except the ones from the decoder layer of the autoencoder network, which actually correspond to the already normalised dictionary atoms.

The following elements support the contributions of this chapter

1. The NNDDL requires gradient calculations to update the dictionary, the sparse code and the classifier model throughout the backpropagation algorithm. The state-of-the-art dictionary learning algorithms solve the same joint optimisation problem through variants of the ADMM method, which relies on decomposing the non-convex optimisation into two sub-problems that are solved separately. The ADMM coordinates solutions to these sub-problems to build the final solution back to the original problem. While efficient, many of these ADMM-based algorithms need a scalable infrastructure to solve the problem in parallel for many training samples. This work formulates the discriminative sparse coding and dictionary learning problem as a feedforward neural network training process. It is performed with an autoencoder and a classifier model connected to the sparse latent space. As a result, one can take advantage of the parallelism offered by GPUs to speed up learning and enhance classification results.

2. Decomposition-coordination methods, such as the ADMM, may not necessarily converge to stationary points of the optimisation problem in many situations (BOYD *et al.*, 2011). Since only non-increment property is ensured, even the convergence to stationary points cannot be guaranteed. In

these cases, the dictionary and the corresponding sparse codes may result in a relatively poor representation and low-accuracy classifier. Therefore, we propose an optimisation approach based on gradient descent for supervised learning to overcome this issue. As a result, the proposed framework is guaranteed to converge to stationary points in general, leading to high-quality sparse representations and higher accuracy classifiers.

3. The FDDL method has extremely slow convergence, which is sometimes impractical for multi-class high-dimension problems. The proposed method simultaneously optimises the dictionary and the spare codes in batch mode, resulting in a faster and accurate algorithm.

5.6 Dataset

The dataset includes EEG signals with 256 channels collected from 11 subjects executing an SSVEP-based experimental protocol. Five different frequencies presented in isolation have been used for the visual stimulation: 6.66Hz, 7.50Hz, 8.57Hz, 10.00Hz and 12.00Hz. The EGI 300 Geodesic EEG System (GES 300), using a 256-channel HydroCel Geodesic Sensor Net (HCGSN) and a sampling rate of 250Hz was used for collecting the signals. Further details regarding the acquisition setup can be found in (OIKONOMOU *et al.*, 2016). To relate the dense net array of EEG signals from HCGSN back to the classic EEG systems 10-20 or 10-10, you can refer to (LUU; FERREE, 2005), which describes the equivalence between the electrode positions for the 256-channels from HCGSN which are shown in Figure 5.2 and the classical 10-20 system.

The stimulus of the experiment was one violet box, presented on the center of a monitor with black background color. This box could flicker in one of the 5 aforementioned frequencies. The box flickering in a specific frequency was presented for 5 seconds, denoted hereafter as a trial, followed by 5 seconds without visual stimulation.

The protocol undertaken by each subject is repeated to identical sessions, as shown in Figure 5.7. Subjects S001, S003 and S008 participated in 3 sessions and S004 participated in 4 sessions. All the remaining subjects participated in 5 sessions. Each session initiates with 100 seconds of resting period, where the volunteer could look at the black screen of the monitor without being involved in any activity. It follows with another 100 seconds of adaptation period, which consisted in the presentation of the 5 selected frequencies randomly, plus an additional time for resting and get prepared for the trials. At this part of the protocol, the subject had the opportunity to familiarise with the visual stimulation.

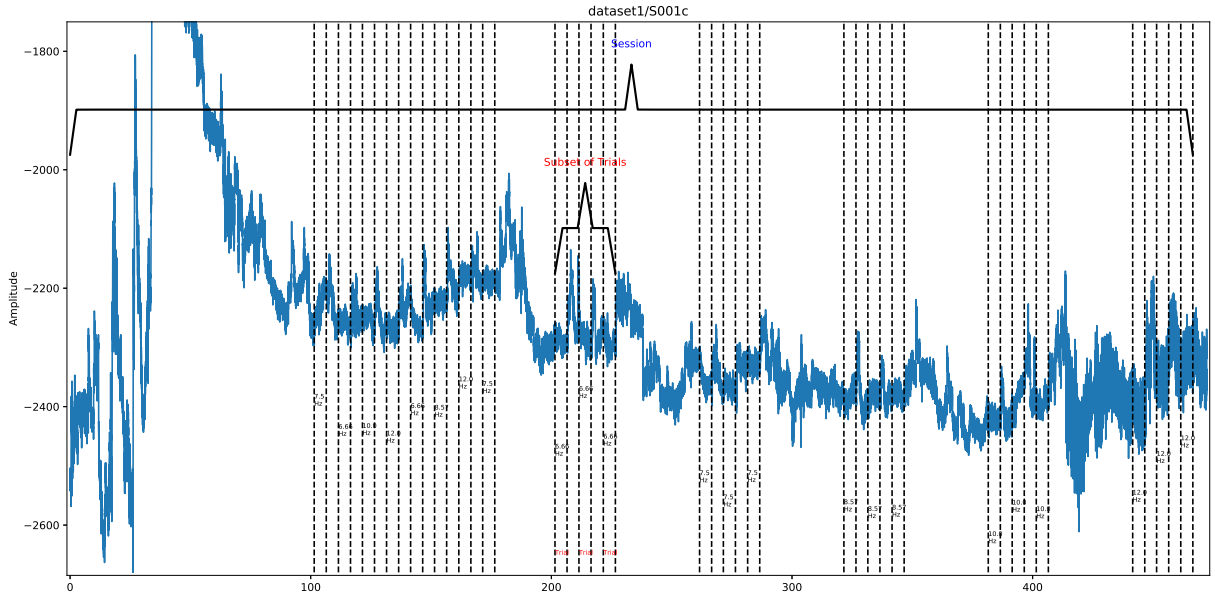


Figure 5.7 – Experimental setup for the third session (c) collected from subject S001: the session initiates with 100 seconds of resting and then follows to another 100 seconds of the adaptation period. After the adaptation period, the remaining trials initiate in t=200 s. Each subset of trials consists of presenting one of the five frequencies of interest three times, with a resting period of 5 s between each trial. The subsets are separated with a period of 30 s without visual stimulation.

In each session, the subset of trials consisted in presenting one of the 5 selected frequencies for 3 times. Each individual trial is separated by 5 seconds without visual stimulation. After each subset of trials, a 30 seconds break is added before the next subset of trials begins. In total, each session includes 23

trials, with 8 of them being part of the adaptation period. MAMEM dataset is downloadable without restrictions[1].

5.7 Evaluation Protocol

We defined an evaluation protocol for assessing the performance and comparing the results across the state-of-the-art discriminative dictionary learning algorithms introduced in Section 5.4, and the proposed method explained in Section 5.5. The systematic comparison focuses on evaluating these approaches regarding the SSVEP-based BCI system. We consider these methods should not foresee any subject-specific training samples before their evaluation using the same subject, which resulted in the adoption of the leave-one-subject-out evaluation protocol.

Cross-Validation (CV) techniques are usually employed to ensure a fair comparison between the different evaluated approaches. The CV relates to splitting the available dataset into two separated non-overlapping parts: the training set and the test set. We run a specific dictionary learning framework using only the samples from the training dataset. The performance of each framework is then evaluated with the samples from the test dataset. The variations of the CV framework are related to the way we choose to split the dataset and the metrics used to quantify the performance of each configuration.

For the experiments from now on discussed, we employ a CV approach where the splitting procedure is performed on the basis of subjects, and the performance is evaluated according to the accuracy and Matthew's Correlation Coefficient (MCC) achieved with the classifiers. This CV approach is known as the Leave-One-Subject-Out (LOSO). As the name indicates, the LOSO-CV leaves the data from one specific subject out of the training phase and uses them only in the test phase of the experiment. This splitting is often adopted

[1] <https://doi.org/10.6084/m9.figshare.5231053>

for experiments involving physiological data since it measures the ability to construct general-purpose systems.

An alternative CV approach could be performing the training procedure and the test phase with the EEG data slices from all subjects. To this end, we could use, for instance, a K-fold CV approach. In this case, different data slices from the same subject are included in the training and the test dataset. The expected variability of the results achieved by each subject would be lower when compared to the ones obtained from the LOSO approach. Lower variability happens because the knowledge of the test dataset leaks into the training dataset, i.e., the classifier uses data samples from the same subject during the training and test phases. Therefore, for evaluating a general, non-personalised BCI system, the LOSO-CV approach is more suitable for observing performance variability across different subjects and for avoiding data leakage.

5.8 Experimental Setup

In this section, we present an evaluation of the proposed discriminative dictionary learning algorithm and state-of-the-art methods described in Section 5.4. We aim to compare the existing techniques from the literature in terms of their classification performance within the context of SSVEP-based BCI systems. To support this analysis, we adopt two distinct default BCI configurations, where we fix specific algorithms at each module, including filtering, artefact removal, feature extraction, feature selection, and classification. With these default configurations, we can establish a performance baseline, which allows us to determine whether a particular algorithm introduces significant improvements.

It is worth noting that our objective is not to optimise all the different modules of the BCI system. Although performance depends on various stages, multiple algorithms can implement them, and we only focus on the classifier stage. In the first default configuration, we adopt the simplest choices at each

Table 5.2 – First default configuration of the EEG-based BCI application.

Channel
The electrode channels usually selected in SSVEP classification tasks are O1, O2 and Oz. All these electrodes are located at the visual cortex. The experiments reported hereinafter are performed using channel Oz only. This channel is collected from the midline of the occipital lobe, as shown in Figure 5.2.
Signal Filtering
The raw EEG signal is bandpass filtered between 5 and 48 Hz. An IIR-Chebyshev I filter is used with the following configuration: Stopband Frequency 1 is set to 3 Hz, Stopband Frequency 2 is set to 58 Hz, Passband Frequency 1 is set to 5 Hz, Passband Frequency 2 is set to 48 Hz, Stopband Attenuation is set to 50 dB and Passband ripple is set to 0.4 dB. Filter Response is shown in Figure 5.8
Artifact Rejection
No artifact rejection technique is added in this default configuration.
Feature Extraction
No feature extraction technique is added in this default configuration.
Feature Selection
No feature selection method is applied in this default configuration.
Classification
We apply the SVM classifier with a linear kernel and the cost parameter set to $\mathsf{C} = 1$. We used Scikit-Learn (PEDREGOSA *et al.*, 2011), which internally uses libsvm (CHANG; LIN, 2011) and liblinear (FAN *et al.*, 2008) libraries to handle all computations required to run SVM algorithm.

stage of the system, as outlined in Table 5.2, which shows the algorithms employed at each stage and their corresponding internal parameters. We utilise the SVM as the classifier method.

The magnitude and phase response of the IIR bandpass filter adopted at the first stage of this first experiment are shown in Figure 5.8. Table 5.3 presents the classification accuracy and Matthew's correlation coefficient achieved for each subject using such a configuration. The confusion matrices obtained in each experiment using LOSO-CV are shown in Figure 5.9.

In the second experiment of this section, we determine a second default configuration with further enhancements at each stage of the BCI system. To this end, we take similar approaches as those adopted in (OIKONOMOU *et al.*,

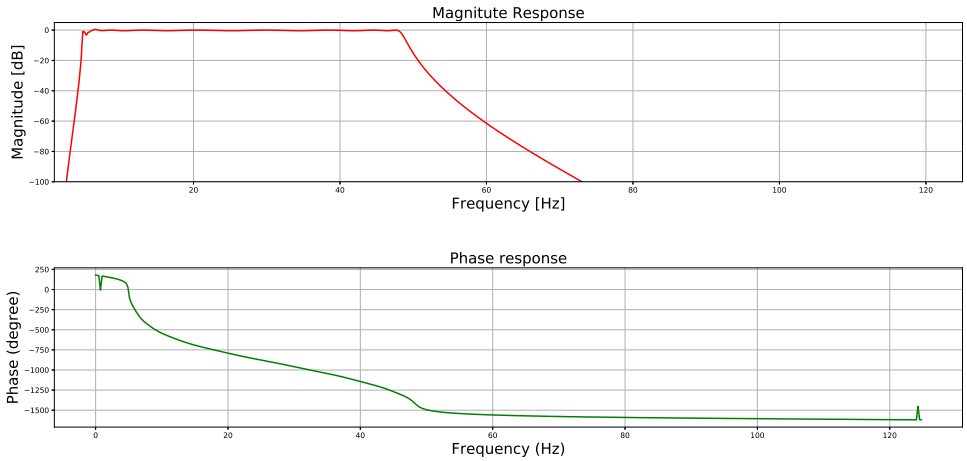


Figure 5.8 – Magnitude and phase response of the IIR-Chebyshev I filter applied as a bandpass filter to the raw EEG channels.

Table 5.3 – Performance achieved with the first default configuration.

Subject ID	Accuracy	Matthew's Coefficient	N. Trials
S001	1.000	1.000	69
S002	0.965	0.957	115
S003	0.420	0.282	69
S004	0.815	0.770	92
S005	0.356	0.192	115
S006	0.896	0.872	115
S007	0.765	0.707	115
S008	0.507	0.384	69
S009	0.991	0.989	115
S010	0.870	0.859	115
S011	0.991	0.989	115
Mean	**0.780**	**0.727**	

2016). With a more robust configuration, the authors in (OIKONOMOU *et al.*, 2016) categorised the subjects into three different classes

- **a) highly-accurate class**, where the subjects present the accuracy over 90%. In our experiment, the following subjects are included to this class: S001, S002, S009, and S011;
- **b) mid-accurate class**, where the subjects present the accuracy between

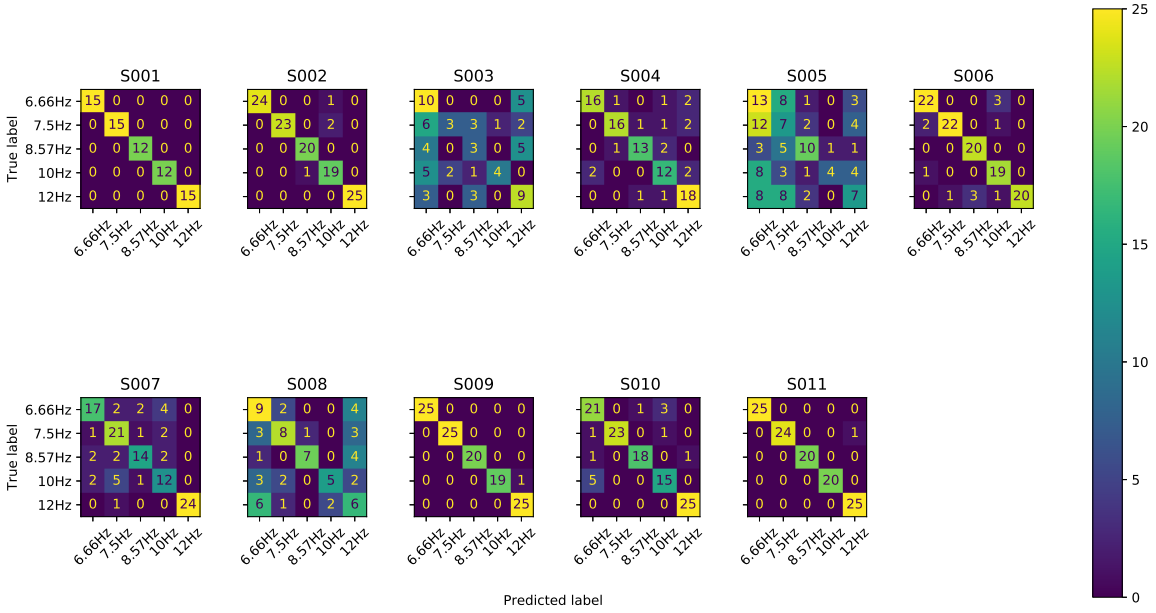


Figure 5.9 – Confusion matrices obtained with LOSO cross-validation method, using the first default configuration.

60% and 90%. In our experiment, the following subjects are included to this class: S004, S006 and S010, and

- **c) poorly-accurate class:**, where the subjects present the accuracy below 60%. In our experiment, the following subjects are included to this class: S003, S005, S007 and S008.

According to (OIKONOMOU *et al.*, 2016), all subjects in the highly-accurate and mid-accurate classes have either short or regular hair (with the only exception of S006), while the subjects in the poorly-accurate class appear to have thick hair (with the only exception of S007). Another interesting remark extracted from the same study indicates that the subject S007, from poorly-accurate class, was observed to excessively blink during the execution of the experiment. Finally, subject S005 is the only left-handed subject participating in the same data collection. Table 5.4 is indicating the algorithms adopted at

Table 5.4 – Second default configuration of the EEG-based BCI application.

Channel
The electrode channels usually selected in SSVEP classification tasks are O1, O2 and Oz. All these electrodes are located at the visual cortex. The experiments reported hereinafter are performed using channel Oz only. This channel is collected from the midline of the occipital lobe, as shown in Figure 5.2.
Signal Filtering
The raw EEG signal is bandpass filtered between 5 and 48 Hz. An IIR-Chebyshev I filter is used with the following configuration: Stopband Frequency 1 is set to 3 Hz, Stopband Frequency 2 is set to 58 Hz, Passband Frequency 1 is set to 5 Hz, Passband Frequency 2 is set to 48 Hz, Stopband Attenuation is set to 50 dB and Passband ripple is set to 0.4 dB. Filter Response is shown in Figure 5.8
Artifact Rejection
No artifact rejection technique is added in the default configuration.
Feature Extraction
We estimate the power spectrum using Welch's method using the following configuration: the number of FFT points is set to 512, the segment length is set to 156 and the overlap is set to 78.
Feature Selection
No feature selection method is applied in the default configuration.
Classification
We apply the SVM classifier with a linear kernel and the cost parameter set to $C = 1$. We used Scikit-Learn (PEDREGOSA *et al.*, 2011), which internally uses libsvm (CHANG; LIN, 2011) and liblinear (FAN *et al.*, 2008) libraries to handle all computations required to run SVM algorithm.

each stage and their corresponding internal parameters at our second default configuration.

Table 5.5 shows the performance achieved for each subject from the second default configuration. The confusion matrices obtained in each experiment with LOSO-CV are shown in Figure 5.10.

The results of the proposed NNDDL approach were obtained by replacing the classifier in the second default configuration while maintaining the same methods for the remaining stages. To evaluate its efficacy, we compared its performance against several state-of-the-art methods, including the SRC, DLSI, COPAR, FDDL, and LRSDL, by replacing the proposed method at the classifier

Table 5.5 – Performance achieved with the second default configuration.

Subject ID	Accuracy	Matthew's Coefficient	N. Trials
S001	1.000	1.000	69
S002	0.904	0.882	115
S003	0.319	0.155	69
S004	0.802	0.753	92
S005	0.243	0.048	115
S006	0.609	0.557	115
S007	0.522	0.416	115
S008	0.246	0.051	69
S009	0.991	0.989	115
S010	0.791	0.742	115
S011	0.983	0.978	115
Mean	**0.674**	**0.597**	

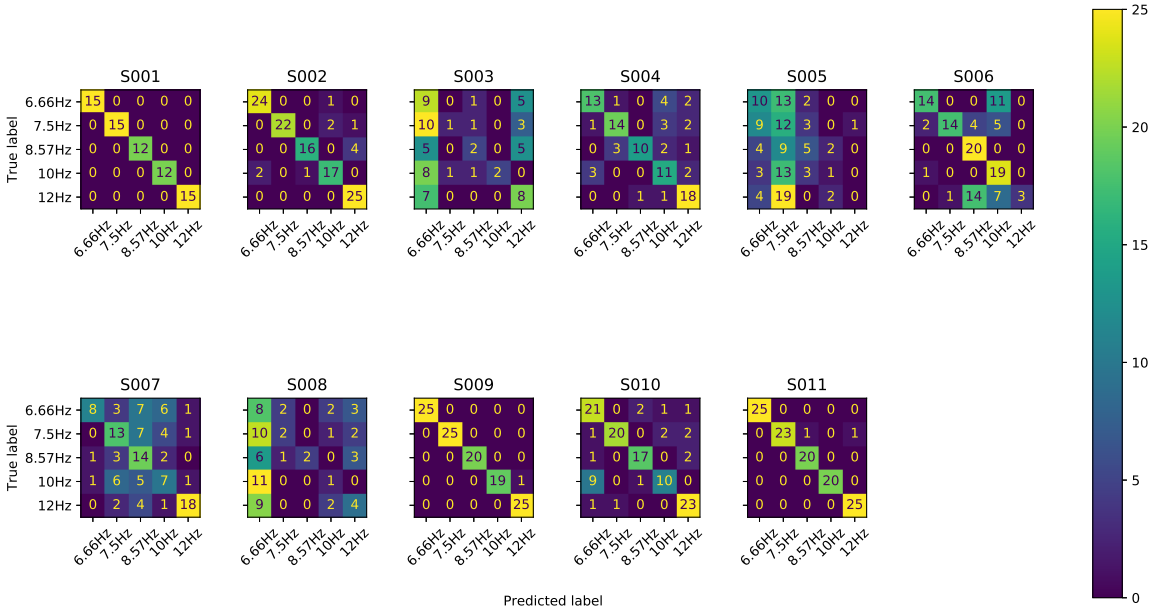


Figure 5.10 – Confusion matrices obtained with LOSO cross-validation method, using the second default configuration.

stage. The achieved classification accuracy with each method is summarised in Table 5.6. The experimental results indicate that the proposed NNDDL method outperformed the SRC method on the MAMEM dataset. However, it was found

to perform worse than other state-of-the-art techniques, such as DLSI, COPAR, FDDL, and LRSDL. Notably, our baseline SVM method achieved higher accuracy than the state-of-the-art discriminative dictionary learning methods analysed in this experiment. These observations suggest that while the proposed approach may be effective in some classification tasks, and even outperform some contender methods, it may not be the optimal choice for all scenarios. Further research is necessary to better understand the strengths and limitations of this method in comparison to other approaches.

Additionally, the proposed approach demonstrated superiority by outperforming all state-of-the-art methods when applied on subject 5, which is known to present the highest difficulty in accurately classifying SSVEP signals. The results highlighted in Table 5.7 clearly indicate the effectiveness of the proposed approach in tackling the challenges associated with subject 5's unique SSVEP patterns. These findings not only validate the strength of our method but also highlight its potential to significantly enhance SSVEP classification performance, particularly in challenging scenarios.

5.9 Analysis of the Proposed Method

In the Sparse Representation Classifier (SRC) method, signals are assigned to specific classes based on smaller residuals that indicate lower reconstruction error. Unlike the SRC method, the proposed Neural Network Dictionary Learning (NNDDL) method does not impose any structure on the matrix of atoms for each class. Therefore, both the dictionary matrix D and the sparse code matrix X exhibit unstructured activation patterns for each class. To illustrate this, Figure 5.11 shows the coefficient distributions of each class in the sparse code matrix X for the EEG signals. These patterns are visually challenging to distinguish when compared to the typical patterns obtained from the SRC method.

Table 5.6 – Performance achieved with the proposed NNDDL method and state-of-the-art methods of discriminative dictionary learning using the second default configuration.

	NNDDL		SRC		DLSI		COPAR		FDDL		LRSDL		
Subject	Acc.	M. Coef.	Acc.	M. Coef.	Acc.	M. Coef.	Acc.	M. Coef.	Acc.	M. Coef.	Acc.	M. Coef.	N. Trials
S001	0.826	0.788	0.812	0.769	0.928	0.911	0.957	0.947	0.957	0.947	0.957	0.947	69
S002	0.574	0.466	0.452	0.312	0.678	0.599	0.739	0.675	0.704	0.631	0.765	0.708	115
S003	0.348	0.180	0.217	0.003	0.377	0.220	0.391	0.243	0.362	0.202	0.304	0.130	69
S004	0.457	0.320	0.370	0.214	0.446	0.310	0.652	0.577	0.630	0.547	0.728	0.667	92
S005	0.296	0.118	0.183	-0.030	0.183	-0.026	0.209	0.005	0.226	0.024	0.287	0.116	115
S006	0.504	0.391	0.444	0.303	0.704	0.646	0.739	0.697	0.713	0.665	0.757	0.709	115
S007	0.435	0.293	0.409	0.262	0.609	0.517	0.670	0.590	0.644	0.556	0.644	0.557	115
S008	0.217	0.019	0.261	0.073	0.275	0.096	0.217	0.024	0.217	0.028	0.261	0.096	69
S009	0.835	0.796	0.661	0.576	0.930	0.913	0.991	0.989	0.983	0.978	1.000	1.000	115
S010	0.452	0.314	0.191	-0.018	0.383	0.227	0.487	0.360	0.461	0.329	0.530	0.415	115
S011	0.713	0.642	0.548	0.436	0.809	0.765	0.835	0.796	0.809	0.765	0.930	0.914	115
Mean	0.514	0.393	0.413	0.264	0.575	0.471	0.626	0.537	0.610	0.516	0.651	0.569	
Std	0.205	0.258	0.202	0.257	0.259	0.326	0.270	0.339	0.265	0.334	0.273	0.338	

Table 5.7 – Results achieved in Subject S005: the NNDDL method outperforms all state-of-the-art methods.

	NNDDL		SRC		DLSI		COPAR		FDDL		LRSDL	
	Acc.	M. Coef.	Acc.	M. Coef.	Acc.	M. Coef.	Acc.	M. Coef.	Acc.	M. Coef.	Acc.	M. Coef.
S005	0.296	0.118	0.183	-0.030	0.183	-0.026	0.209	0.005	0.226	0.024	0.287	0.116

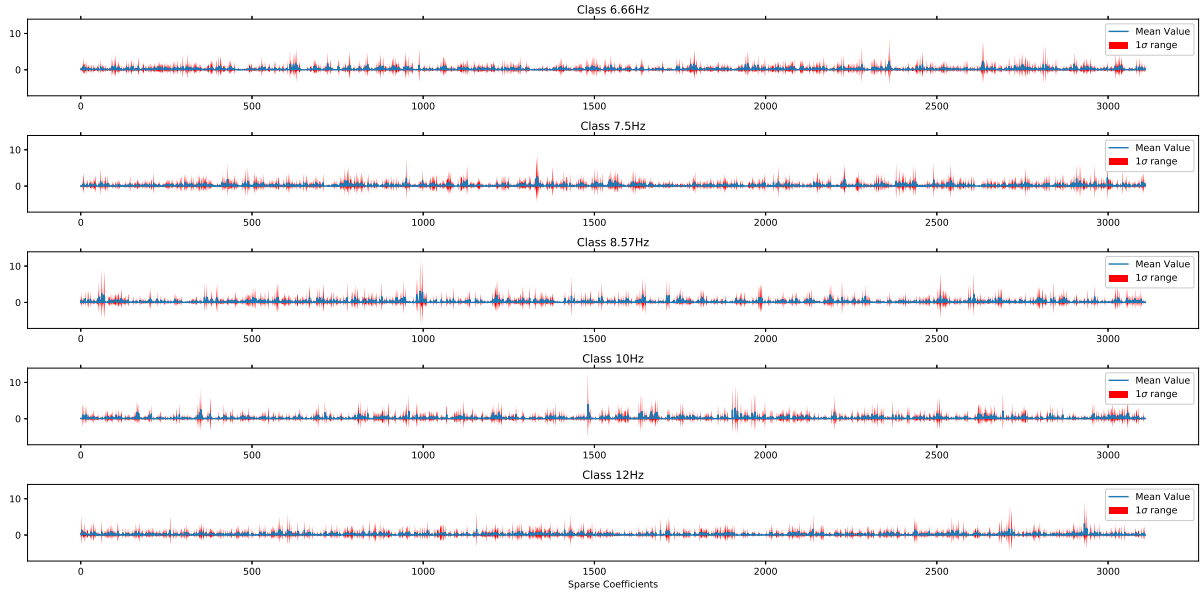


Figure 5.11 – The mean and standard deviation of coefficients in the sparse vectors of each class exhibit statistical patterns that are not easily distinguishable.

To gain a better understanding of the class-specific unstructured sub-dictionaries and the sub-dictionary with shared features, we used a modified version of the unsupervised clustering algorithm, k-means (also known as Lloyd's method). In this experiment, we applied the ℓ_1 norm as the similarity measurement between each sparse vector of $\mathbf{X}$. K-means is an iterative algorithm that assigns samples to clusters to minimise the sum of distances from each sample to its cluster centroid across all clusters. We set the number of clusters to $k = 5$ to match the number of classes, and initialised each cluster with the element-wise mean of the sparse vectors of each class. After a few iterations of k-means, we compared the class labels of each test sample to the clusters obtained from k-means, and the results are presented in Figure 5.12.

Figure 5.12 illustrates that clustering the sparse codes yields results

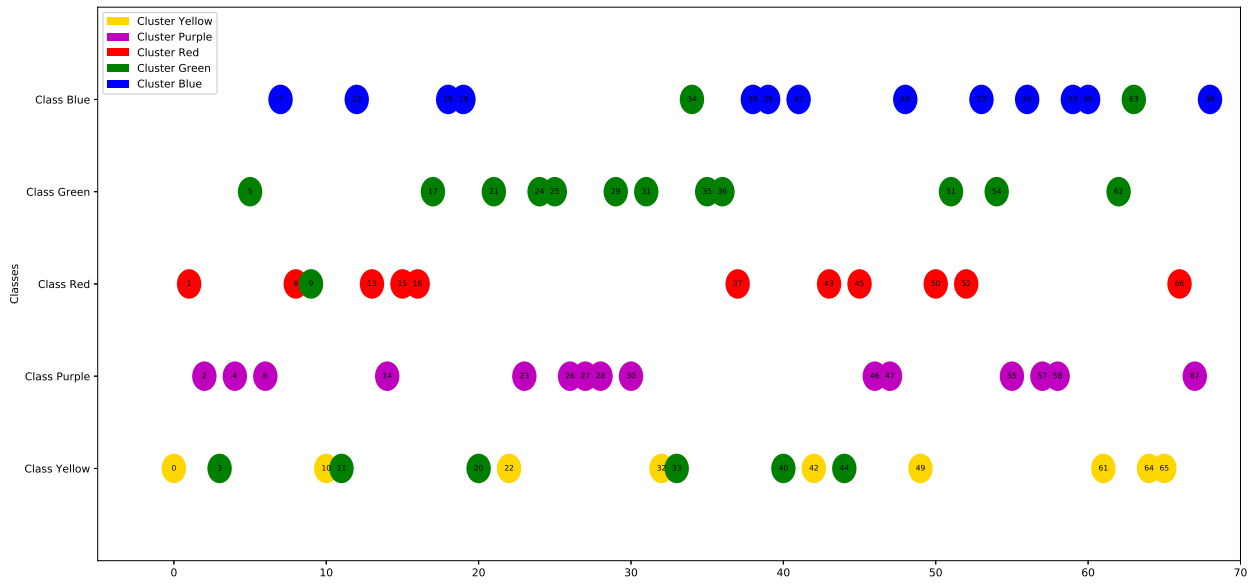


Figure 5.12 – Clustering of sparse vectors for Subject S001 using LOSO cross-validation. Each horizontal line represents the actual signal class, and the inner circles annotate the sample indices. The colors indicate the clusters obtained from adapted k-means. The horizontal axis represents the test sample indices.

that are mostly consistent with the expected labels. However, there exist sparse samples, namely those with $\mathbf{i} = 34, 9, 3, 11, 20, 33, 40, 44, 63$, where the class label and the cluster assignment do not match. This can be attributed to coefficients in the sparse vectors $\mathbf{x}_i$ that correspond to features $\mathbf{d}_j$ shared among multiple classes. These coefficients do not contribute to distinguishing samples, thus potentially leading to erroneous cluster assignments. Moreover, clustering methods rely solely on distance measures between sparse vectors and mean values of neighbouring samples, which can result in sub-optimal signal classification. Our experiment suggests that although we did not explicitly encourage incoherence between sub-dictionaries associated with different classes, the proposed model architecture intrinsically induces this behavior in the process of learning the dictionary, sparse coding, and classification.

In the second experiment focused on class-specific and shared features, we investigated the frequency of usage of each dictionary atom in reconstructing the test samples of each class. This analysis aimed to establish a corre-

Table 5.8 – Distribution of dictionary atoms utilised as class-specific features (in bold) and as shared features. The total number of dictionary atoms is 3110 and the sparsity threshold is tested for two borderline cases where: (I) the atom is considered activated if the coefficient $|\boldsymbol{x}_{i,j}| > 0$ and (II) the atom is considered activated if the coefficient $|\boldsymbol{x}_{i,j}| > 5$.

$C_1C_2C_3C_4C_5$	Threshold = 0 Number of Atoms	Threshold = 5 Number of Atoms
00000	13	2827
00001	20	58
00010	14	33
00011	29	1
00100	15	22
00101	34	1
00110	16	0
00111	64	0
01000	22	75
01001	38	3
01010	25	1
01011	88	0
01100	20	1
01101	75	0
01110	46	0
01111	180	0
10000	24	85
10001	43	1
10010	28	0
10011	59	0
10100	27	0
10101	65	0
10110	43	0
10111	210	0
11000	41	2
11001	114	0
11010	70	0
11011	290	0
11100	83	0
11101	270	0
11110	152	0
11111	892	0

lation between the distribution of class-specific and shared features and the sparsity threshold, which dictates the level of sparseness. Results were obtained from Subject S001 using a leave-one-subject-out cross-validation method and are presented in Table 5.8. Two extreme sparsity threshold values were examined: $\mathtt{t} = 0$ and $\mathtt{t} = 5$. In the former case, a dictionary atom was deemed active if its corresponding sparse coefficient was non-zero, whereas in the latter, activation required a coefficient magnitude greater than 5. Our analysis revealed that low sparsity thresholds produced sparse vectors that dispersed energy across a wide range of shared-feature atoms, with little concentration on class-specific atoms. Conversely, high thresholds resulted in sparse vectors that highly concentrated energy in the class-specific atoms. Notably, setting the sparsity threshold too high resulted in unused dictionary atoms ($\mathtt{C}_1\mathtt{C}_2\mathtt{C}_3\mathtt{C}_4\mathtt{C}_5 = 00000$), increasing residuals but possibly improving the classifier's performance. Nevertheless, this approach under utilises the dictionary's capacity to represent the signals of interest.

Chapter 6

Conclusions

In this dissertation, we proposed a method based on sparse coding and dictionary learning using sparse autoencoders. The method was tested in two different practical contexts: image compression and brain-computer interfaces (BCIs).

In Chapter 3, we started investigating sparsity in the pure reconstructive setting, which allowed us to understand the basics of sparse coding and dictionary learning. The primary objective of this chapter was to evaluate the ability to achieve optimal sparse levels while maintaining a low representation error. We observed that the convolutional autoencoder model outperformed the fully-connected and residual models in achieving higher sparsity levels with lower reconstruction errors. Furthermore, we tested the impact of a different regularisation technique based on Kullback-Leibler divergence. The results of these experiments indicate that the proposed framework of dictionary learning with convolutional autoencoder models and KL regularisation can be a viable approach for sparse representation and efficient data compression. However, further investigation is necessary to determine the effectiveness of this framework on more complex datasets and tasks.

The ADMM consensus is considered as one of the fastest methods implemented in parallel due to its separable structure. However, its usage on

large sets of images is computationally restricted by the dictionary update stage. On the other hand, the proposed autoencoder-based framework is a scalable approach whose number of training patterns is not a drawback.

In Chapter 4, the sparse autoencoder-based framework for dictionary learning has shown promising potential for achieving high compression rates while preserving the visual quality of the image. Compared to traditional analytic transforms like the Fourier and Wavelets, the proposed approach can effectively exploit the specific statistical properties of the signals of interest, and has the potential to surpass the performance of other transforms in terms of the quality of the reconstructed image. These results suggest that the proposed DL framework can be considered a competitive and promising option for developing image codec frameworks to achieve high compression rates while maintaining the quality of the image. Further research is needed to validate these findings on larger and more diverse datasets.

Our study shows that incorporating a convolutional autoencoder with residual connections into the data-driven dictionary approach can enhance the performance of image compression for the Cars dataset. This is demonstrated by comparing the visual quality and peak signal-to-noise ratio (PSNR) of our proposed framework with those of the Fourier and Wavelet transforms. We also adjusted the thresholds applied to the sparse codes to obtain compressed images with a similar number of coefficients as the other methods. The experimental results indicate that our proposed framework achieves the best visual quality and highest PSNR value at lower compression rates, but its visual quality is comparatively lower than the other transforms at the highest compression rate. Nevertheless, the proposed framework shows promise as an approach for data-driven dictionary learning in image compression, especially at lower compression rates.

In Chapter 5, the proposed discriminative dictionary learning method

has shown promising results for SSVEP classification. Compared to the SRC method, the proposed method achieved a better classification performance. Additionally, our method achieved better results than all the competitor methods on subject 5's, being an interesting alternative for particularly challenging scenarios. However, when compared to DLSI, COPAR, FDDL, and LRSDL, the proposed method performed worse, overall. It is our belief that with further studies and a finer hyperparameter tuning process, the performance achieved from NNDDL can be improved.

One of the key advantages of the proposed discriminative dictionary learning method is its ability to learn discriminative features that can improve the separability of different SSVEP signals. This is achieved by incorporating a discriminative term in the optimisation objective, which encourages the learned features to have high discriminative power for classification. Additionally, the proposed method can learn a dictionary that is tailored to the specific SSVEP signals, which can improve the accuracy of the classification.

Future Directions

The present work has opened up new paths for further research in the field of sparse coding and dictionary learning. Some possible directions for future research are:

- Investigating the performance of the proposed dictionary learning framework on larger and more complex datasets. This will help to validate the effectiveness of the framework for a wide range of applications.
- Performing a more thorough hyperparameter sensitivity analysis so as to form a clearer view of the potential of the proposed methodology.
- Exploring the use of different types of autoencoders and regularisation techniques for dictionary learning. This will help determine which models and

techniques are most effective for different types of data and applications.

- Investigating the effectiveness of the proposed image compression framework on different types of images and compare its performance with other state-of-the-art compression methods.
- Extending the proposed framework to video compression and evaluate its performance compared to existing video compression methods.
- Investigating the use of sparse autoencoders for dictionary learning and sparse coding techniques for other signal processing tasks, such as denoising, super-resolution, and image inpainting.
- Investigating the performance of the proposed discriminative dictionary learning method on other types of BCI paradigms, such as motor imagery and P300. This will help to determine whether the method can be applied to other types of EEG signals and to evaluate its applicability to diverse EEG signals.

Overall, the proposed frameworks and methods have the potential to significantly advance the state-of-the-art in sparse coding and dictionary learning, and further research is necessary to fully realise this potential.

References

ABADI, M.; AGARWAL, A.; BARHAM, P.; BREVDO, E.; CHEN, Z.; CITRO, C.; CORRADO, G. S.; DAVIS, A.; DEAN, J.; DEVIN, M.; GHEMAWAT, S.; GOODFELLOW, I.; HARP, A.; IRVING, G.; ISARD, M.; JIA, Y.; JOZEFOWICZ, R.; KAISER, L.; KUDLUR, M.; LEVENBERG, J.; MANé, D.; MONGA, R.; MOORE, S.; MURRAY, D.; OLAH, C.; SCHUSTER, M.; SHLENS, J.; STEINER, B.; SUTSKEVER, I.; TALWAR, K.; TUCKER, P.; VANHOUCKE, V.; VASUDEVAN, V.; VIéGAS, F.; VINYALS, O.; WARDEN, P.; WATTENBERG, M.; WICKE, M.; YU, Y.; ZHENG, X. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Software available from tensorflow.org. Disponível em: <https://www.tensorflow.org/>. Cited on page 58.

ADELMAN, G. *et al. Encyclopedia of neuroscience*. [S.l.]: Birkhäuser, 1987. Cited on page 112.

AHARON, M.; ELAD, M.; BRUCKSTEIN, A. *et al.* K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, IEEE INSTITUTE OF ELECTRICAL AND ELECTRONICS, v. 54, n. 11, p. 4311, 2006. Cited 3 times on pages 31, 45 e 47.

AKBARI, A.; TROCAN, M. On the application of dictionary learning to image compression. In: *Coding Theory*. [S.l.]: IntechOpen, 2019. Cited on page 104.

ALMUHAMMADI, W. S.; ABOALAYON, K. A.; FAEZIPOUR, M. Efficient obstructive sleep apnea classification based on eeg signals. In: IEEE. *2015 Long Island Systems, Applications and Technology*. [S.l.], 2015. p. 1–6. Cited on page 116.

AYINDE, B. O.; ZURADA, J. M. Discovery through constraints: Imposing constraints on autoencoders for data representation and dictionary learning. *IEEE Systems, Man, and Cybernetics Magazine*, IEEE, v. 3, n. 3, p. 13–24, 2017. Cited 4 times on pages 85, 87, 90 e 91.

BAO, C.; JI, H.; QUAN, Y.; SHEN, Z. Dictionary learning for sparse coding: Algorithms and convergence analysis. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 38, n. 7, p. 1356–1369, 2016. Cited on page 31.

BLUMENSATH, T.; DAVIES, M. E. Gradient pursuits. *IEEE Transactions on Signal Processing*, IEEE, v. 56, n. 6, p. 2370–2382, 2008. Cited on page 36.

BOSTANTJOPOULOU, S.; KATSAROU, Z.; DAGKLIS, I. Brain–computer interfaces in a home environment for patients with motor impairment - the mamem use case. *Signal Processing to Drive Human-Computer Interaction: EEG and eye-controlled interfaces*, Institution of Engineering and Technology, p. 33, 2020. Cited on page 110.

BOYD, S.; PARIKH, N.; CHU, E.; PELEATO, B.; ECKSTEIN, J. *et al.* Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, Now Publishers, Inc., v. 3, n. 1, p. 1–122, 2011. Cited 2 times on pages 41 e 132.

BRUNTON, S. L.; KUTZ, J. N. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. [S.l.]: Cambridge University Press, 2019. Cited 2 times on pages 95 e 97.

CARVALHO, S. N.; COSTA, T. B.; URIBE, L. F.; SORIANO, D. C.; YARED, G. F.; CORADINE, L. C.; ATTUX, R. Comparative analysis of strategies for feature extraction and classification in ssvep bcis. *Biomedical Signal Processing and Control*, Elsevier, v. 21, p. 34–42, 2015. Cited 2 times on pages 110 e 116.

CHANG, C.-C.; LIN, C.-J. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, Acm New York, NY, USA, v. 2, n. 3, p. 1–27, 2011. Cited 2 times on pages 137 e 140.

CHEN, S. S.; DONOHO, D. L.; SAUNDERS, M. A. Atomic decomposition by basis pursuit. *SIAM review*, SIAM, v. 43, n. 1, p. 129–159, 2001. Cited 3 times on pages 30, 34 e 40.

CHOI, S.; CICHOCKI, A.; PARK, H.-M.; LEE, S.-Y. Blind source separation and independent component analysis: A review. *Neural Information Processing-Letters and Reviews*, v. 6, n. 1, p. 1–57, 2005. Cited on page 118.

COOLEY, J. W.; TUKEY, J. W. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, JSTOR, v. 19, n. 90, p. 297–301, 1965. Cited on page 95.

DAVIS, G.; MALLAT, S.; AVELLANEDA, M. Adaptive greedy approximations. *Constructive approximation*, Springer, v. 13, n. 1, p. 57–98, 1997. Cited 2 times on pages 33 e 34.

DENG, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, IEEE, v. 29, n. 6, p. 141–142, 2012. Cited on page 87.

DONOHO, D. L. Compressed sensing. *IEEE Transactions on information theory*, IEEE, v. 52, n. 4, p. 1289–1306, 2006. Cited on page 28.

DORNHEGE, G.; BLANKERTZ, B.; CURIO, G.; MULLER, K.-R. Boosting bit rates in noninvasive eeg single-trial classifications by feature combination and multiclass paradigms. *IEEE transactions on biomedical engineering*, IEEE, v. 51, n. 6, p. 993–1002, 2004. Cited on page 119.

ELAD, M.; AHARON, M. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image processing*, IEEE, v. 15, n. 12, p. 3736–3745, 2006. Cited on page 28.

FAN, R.-E.; CHANG, K.-W.; HSIEH, C.-J.; WANG, X.-R.; LIN, C.-J. Liblinear: A library for large linear classification. *the Journal of machine Learning research*, JMLR. org, v. 9, p. 1871–1874, 2008. Cited 2 times on pages 137 e 140.

FOURIER, J. B. J. baron. *Théorie analytique de la chaleur*. [S.l.]: F. Didot, 1822. Cited on page 95.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep learning*. [S.l.]: MIT press, 2016. Cited 2 times on pages 53 e 72.

HE, K.; GKIOXARI, G.; DOLLÁR, P.; GIRSHICK, R. Mask r-cnn. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2017. p. 2961–2969. Cited on page 70.

HU, J.; TAN, Y.-P. Nonlinear dictionary learning with application to image classification. *Pattern Recognition*, Elsevier, v. 75, p. 282–291, 2018. Cited on page 90.

HUANG, J.; ZHANG, T. The benefit of group sparsity. *The Annals of Statistics*, Institute of Mathematical Statistics, v. 38, n. 4, p. 1978–2004, 2010. Cited on page 110.

HWAIDI, J. F.; CHEN, T. M. Classification of motor imagery eeg signals based on deep autoencoder and convolutional neural network approach. *IEEE access*, IEEE, v. 10, p. 48071–48081, 2022. Cited on page 51.

KAMAL, I. M.; BAE, H. Super-encoder with cooperative autoencoder networks. *Pattern Recognition*, Elsevier, v. 126, p. 108562, 2022. Cited on page 51.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. Cited 2 times on pages 60 e 61.

KONG, S.; WANG, D. A dictionary learning approach for classification: Separating the particularity and the commonality. In: SPRINGER. *European conference on computer vision*. [S.l.], 2012. p. 186–199. Cited 2 times on pages 120 e 124.

KRAUSE, J.; STARK, M.; DENG, J.; FEI-FEI, L. 3d object representations for fine-grained categorization. In: *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*. Sydney, Australia: [s.n.], 2013. Cited 2 times on pages 99 e 101.

KRIZHEVSKY, A.; HINTON, G. *et al.* Learning multiple layers of features from tiny images. Toronto, ON, Canada, 2009. Cited on page 21.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *nature*, Nature Publishing Group UK London, v. 521, n. 7553, p. 436–444, 2015. Cited on page 70.

LEWICKI, M. S.; SEJNOWSKI, T. J. Learning overcomplete representations. *Neural computation*, MIT Press, v. 12, n. 2, p. 337–365, 2000. Cited 3 times on pages 27, 29 e 42.

LIN, G.; YANG, M.; YANG, J.; SHEN, L.; XIE, W. Robust, discriminative and comprehensive dictionary learning for face recognition. *Pattern Recognition*, Elsevier, v. 81, p. 341–356, 2018. Cited on page 110.

LUU, P.; FERREE, T. Determination of the hydrocel geodesic sensor nets average electrode positions and their 10–10 international equivalents. *Inc, Technical Note*, p. 1–11, 2005. Cited on page 133.

MAIRAL, J.; BACH, F.; PONCE, J.; SAPIRO, G. Online dictionary learning for sparse coding. In: *Proceedings of the 26th annual international conference on machine learning*. [S.l.: s.n.], 2009. p. 689–696. Cited on page 80.

MAIRAL, J.; BACH, F.; PONCE, J.; SAPIRO, G. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, v. 11, n. Jan, p. 19–60, 2010. Cited 2 times on pages 31 e 48.

MAKHZANI, A.; FREY, B. K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013. Cited 2 times on pages 53 e 87.

MALLAT, S. *A wavelet tour of signal processing*. [S.l.]: Elsevier, 1999. Cited on page 95.

MALLAT, S.; ZHANG, Z. *Matching pursuit with time-frequency dictionaries*. [S.l.], 1993. Cited 3 times on pages 26, 28 e 30.

MAZAHERI, J. A.; GUILLEMOT, C.; LABIT, C. Learning a tree-structured dictionary for efficient image representation with adaptive sparse coding. In: IEEE. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. [S.l.], 2013. p. 1320–1324. Cited on page 106.

MIAO, J.; ZHOU, X.; HUANG, T.-Z. Local segmentation of images using an improved fuzzy c-means clustering algorithm based on self-adaptive dictionary learning. *Applied Soft Computing*, Elsevier, v. 91, p. 106200, 2020. Cited on page 28.

NG, A. *Lecture notes in CS294A: Sparse autoencoder*. [S.l.]: Stanford University, 2011. Cited on page 52.

OIKONOMOU, V. P.; LIAROS, G.; GEORGIADIS, K.; CHATZILARI, E.; ADAM, K.; NIKOLOPOULOS, S.; KOMPATSIARIS, I. Comparative evaluation of state-of-the-art algorithms for ssvep-based bcis. *arXiv preprint arXiv:1602.00904*, 2016. Cited 5 times on pages 109, 119, 133, 138 e 139.

OLSHAUSEN, B. A.; FIELD, D. J. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *NATURE*, v. 381, p. 607, 1996. Cited 2 times on pages 26 e 42.

OLSHAUSEN, B. A.; FIELD, D. J. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, Elsevier, v. 37, n. 23, p. 3311–3325, 1997. Cited 3 times on pages 27, 42 e 43.

PAN, Y.; CHEN, J.; ZHANG, Y. A survey of deep learning-based classification methods for steady-state visual evoked potentials. *Brain-Apparatus Communication: A Journal of Bacomics*, Taylor & Francis, n. just-accepted, p. 1–20, 2023. Cited on page 110.

PATI, Y. C.; REZAIIFAR, R.; KRISHNAPRASAD, P. S. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In: IEEE. *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*. [S.l.], 1993. p. 40–44. Cited on page 33.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Cited 3 times on pages 80, 137 e 140.

PFURTSCHELLER, G.; ALLISON, B. Z.; BAUERNFEIND, G.; BRUNNER, C.; ESCALANTE, T. S.; SCHERER, R.; ZANDER, T. O.; MUELLER-PUTZ, G.; NEUPER, C.; BIRBAUMER, N. The hybrid bci. *Frontiers in neuroscience*, Frontiers, p. 3, 2010. Cited on page 109.

PILASTRE, B.; BOUSSOUF, L.; D'ESCRIVAN, S.; TOURNERET, J.-Y. Anomaly detection in mixed telemetry data using a sparse representation and dictionary learning. *Signal Processing*, Elsevier, v. 168, p. 107320, 2020. Cited on page 28.

RAMIREZ, I.; SPRECHMANN, P.; SAPIRO, G. Classification and clustering via dictionary learning with structured incoherence and shared features. In: IEEE. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. [S.l.], 2010. p. 3501–3508. Cited 2 times on pages 28 e 120.

ROMANO, J. M. T.; ATTUX, R.; CAVALCANTE, C. C.; SUYAMA, R. *Unsupervised signal processing: channel equalization and source separation*. [S.l.]: CRC Press, 2018. Cited on page 117.

RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: Convolutional networks for biomedical image segmentation. In: SPRINGER. *International Conference on Medical image computing and computer-assisted intervention*. [S.l.], 2015. p. 234–241. Cited 2 times on pages 83 e 85.

RUBINSTEIN, R.; BRUCKSTEIN, A. M.; ELAD, M. Dictionaries for sparse representation modeling. *Proceedings of the IEEE*, IEEE, v. 98, n. 6, p. 1045–1057, 2010. Cited on page 93.

RUBINSTEIN, R.; ZIBULEVSKY, M.; ELAD, M. *Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit*. [S.l.], 2008. Cited 6 times on pages 31, 34, 36, 38, 45 e 48.

RUMELHART, D. E.; MCCLELLAND, J. L.; GROUP, P. R. *et al. Parallel distributed processing: Exploration in the microstructure of cognition, Vol. 1*. [S.l.]: Cambridge, MA: MIT Press, 1986. Cited on page 52.

SANEI, S.; CHAMBERS, J. A. *EEG signal processing and machine learning*. [S.l.]: John Wiley & Sons, 2021. Cited 2 times on pages 111 e 113.

SINGLA, R.; HASEENA, B. Comparison of ssvep signal classification techniques using svm and ann models for bci applications. *International Journal of Information and Electronics Engineering*, IACSIT Press, v. 4, n. 1, p. 6, 2014. Cited on page 110.

SKODRAS, A.; CHRISTOPOULOS, C.; EBRAHIMI, T. The jpeg 2000 still image compression standard. *IEEE Signal processing magazine*, IEEE, v. 18, n. 5, p. 36–58, 2001. Cited on page 103.

SUN, Y.; LIU, Q.; TANG, J.; TAO, D. Learning discriminative dictionary for group sparse representation. *IEEE Transactions on Image Processing*, IEEE, v. 23, n. 9, p. 3816–3828, 2014. Cited on page 110.

SUN, Y.; TAO, X.; LI, Y.; LU, J. Dictionary learning for image coding based on multisample sparse representation. *IEEE Transactions on Circuits and Systems for video Technology*, IEEE, v. 24, n. 11, p. 2004–2010, 2014. Cited on page 106.

TANG, H.; LIU, H.; XIAO, W.; SEBE, N. When dictionary learning meets deep learning: Deep dictionary learning and coding network for image recognition with limited data. *IEEE transactions on neural networks and learning systems*, IEEE, v. 32, n. 5, p. 2129–2141, 2020. Cited on page 28.

TARIYAL, S.; MAJUMDAR, A.; SINGH, R.; VATSA, M. Deep dictionary learning. *IEEE Access*, Ieee, v. 4, p. 10096–10109, 2016. Cited on page 87.

TEPLAN, M. *et al.* Fundamentals of eeg measurement. *Measurement science review*, v. 2, n. 2, p. 1–11, 2002. Cited on page 112.

TONG, L.; SOON, V.; HUANG, Y.; LIU, R. Amuse: a new blind identification algorithm. In: IEEE. *IEEE international symposium on circuits and systems*. [S.l.], 1990. p. 1784–1787. Cited on page 117.

TROPP, J. A. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information theory*, IEEE, v. 50, n. 10, p. 2231–2242, 2004. Cited on page 33.

VINCENT, P.; LAROCHELLE, H.; BENGIO, Y.; MANZAGOL, P.-A. Extracting and composing robust features with denoising autoencoders. In: *Proceedings of the 25th international conference on Machine learning*. [S.l.: s.n.], 2008. p. 1096–1103. Cited on page 61.

VU, T. H.; MONGA, V. Learning a low-rank shared dictionary for object classification. In: IEEE. *2016 IEEE International Conference on Image Processing (ICIP)*. [S.l.], 2016. p. 4428–4432. Cited 3 times on pages 120, 121 e 126.

VU, T. H.; MONGA, V. Fast low-rank shared dictionary learning for image classification. *IEEE Transactions on Image Processing*, IEEE, v. 26, n. 11, p. 5160–5175, 2017. Cited on page 123.

WALLACE, G. K. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, IEEE, v. 38, n. 1, p. xviii–xxxiv, 1992. Cited on page 103.

WANG, C.; DU, P.; WU, H.; LI, J.; ZHAO, C.; ZHU, H. A cucumber leaf disease severity classification method based on the fusion of deeplabv3+ and u-net. *Computers and Electronics in Agriculture*, Elsevier, v. 189, p. 106373, 2021. Cited on page 85.

WANG, W.; HAN, Y.; DENG, C.; LI, Z. Hyperspectral image classification via deep structure dictionary learning. *Remote Sensing*, MDPI, v. 14, n. 9, p. 2266, 2022. Cited on page 28.

WRIGHT, J.; YANG, A. Y.; GANESH, A.; SASTRY, S. S.; MA, Y. Robust face recognition via sparse representation. *IEEE transactions on pattern*

analysis and machine intelligence, IEEE, v. 31, n. 2, p. 210–227, 2008. Cited 2 times on pages 120 e 121.

YANG, J.; WRIGHT, J.; HUANG, T. S.; MA, Y. Image super-resolution via sparse representation. *IEEE transactions on image processing*, IEEE, v. 19, n. 11, p. 2861–2873, 2010. Cited on page 28.

YANG, J.; YU, K.; HUANG, T. Efficient highly over-complete sparse coding using a mixture model. In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2010. p. 113–126. Cited on page 28.

YANG, M.; ZHANG, L.; FENG, X.; ZHANG, D. Fisher discrimination dictionary learning for sparse representation. In: IEEE. *2011 International Conference on Computer Vision*. [S.l.], 2011. p. 543–550. Cited 3 times on pages 120, 124 e 126.

YIN, L.; ZHANG, C.; CUI, Z. Experimental research on real-time acquisition and monitoring of wearable eeg based on tgam module. *Computer Communications*, Elsevier, v. 151, p. 76–85, 2020. Cited on page 116.

YU, H.; LU, H.; WANG, S.; XIA, K.; JIANG, Y.; QIAN, P. A general common spatial patterns for eeg analysis with applications to vigilance detection. *IEEE Access*, IEEE, v. 7, p. 111102–111114, 2019. Cited on page 119.

ZHANG, X.; LIN, W.; ZHANG, Y.; WANG, S.; MA, S.; DUAN, L.; GAO, W. Rate-distortion optimized sparse coding with ordered dictionary for image set compression. *IEEE Transactions on Circuits and Systems for Video Technology*, IEEE, v. 28, n. 12, p. 3387–3397, 2017. Cited on page 106.

ZHOU, C.; PAFFENROTH, R. C. Anomaly detection with robust deep autoencoders. In: *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. [S.l.: s.n.], 2017. p. 665–674. Cited on page 51.