New Methods of Artificial Intelligence with Applications in the Analysis of Medical Data

by

Xiaochuan Li

(Under the Direction of Yuan Ke)

Abstract

Artificial intelligence (AI) has become an essential tool in medical data analysis. This dissertation presents new AI methods for medical data analysis, focusing on improving classification and prediction tasks. The dissertation is divided into three projects, and the first two are dealing with medical image classification problems. Unlike natural image datasets, the class labels in medical image datasets are usually severely imbalanced and with limited sample sizes subject to the availability of patients, and aggregating medical images from multiple sources can be challenging due to policy restrictions, privacy concerns, communication costs, and data heterogeneity caused by equipment differences and labeling discrepancies. Further, medical image classes are often highly overlapped. In the first project of this dissertation, we propose to address the imbalanced data and communication issues with the help of transfer learning and artificial samples created by generative models. Instead of requesting medical images from source data, our method only needs a parsimonious supplement of model parameters pre-trained on the source data. The second project introduces an augmented tensor factor model to address the challenge of overlapping issues in image classification. Our model combines samples from multiple classes and decomposes the tensor data into a pervasive component that shares the same pattern across classes and a specific component that varies from class to class. Finally, the third project studies statistical and machine learning models for COVID-19 mortality prediction, where the proposed SARIMAX model with exogenous hospital

information variables improves prediction accuracy and outperformed most of the models from the CDC. This dissertation presents new insights and solutions to important problems in medical data analysis, highlighting the potential of artificial intelligence in healthcare applications.

INDEX WORDS: Deep learning, Transfer learning, GAN, Tensor factor models, COVID-19, Medical imaging

New Methods of Artificial Intelligence with Applications in the Analysis of Medical Data

by

Xiaochuan Li

B.S., Hunan University, China, 2014

M.S., Florida State University, Tallahassee, 2017

A Dissertation Submitted to the Graduate Faculty of the

University of Georgia in Partial Fulfillment of the Requirements for the Degree

Doctor of Philosophy

Athens, Georgia

2023

©2023

Xiaochuan Li

All Rights Reserved

New Methods of Artificial Intelligence with Applications in the Analysis of Medical Data

by

Xiaochuan Li

Major Professor: Yuan Ke

Committee: Shuyang Bai
Justin Strait
Qian Xiao

Electronic Version Approved:

Ron Walcott
Dean of the Graduate School
The University of Georgia
May 2023

Acknowledgments

I am filled with gratitude for the support and assistance I received during the writing of this dissertation. My sincerest thanks go to all those who have supported me throughout my Ph.D. journey, particularly my advisor, Professor Yuan Ke, whose wisdom, encouragement, and unwavering support have been instrumental in shaping me as a researcher and helping me to complete this dissertation. I would also like to extend my heartfelt appreciation to my committee members, Professor Ray Bai, Professor Justin Strait, and Professor Qian Xiao, for their invaluable insights, critiques, and support, which have greatly enriched the quality of this work. I am deeply grateful to the faculty and staff at the University of Georgia, Department of Statistics, for their support and resources, which have been invaluable in furthering my research. Finally, I would like to acknowledge the love and support of my parents and my fiancé, whose unwavering belief in me and encouragement have been a constant source of strength and motivation throughout my Ph.D. journey. I am thankful to all those who have supported me and made this journey possible, and I could not have done it without your help.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Medical Data Digitalization

The advancement of information technology has brought about significant changes in various aspects of human life, particularly in the healthcare sector (Aceto et al., 2020; L. Wang and Alexander, 2015). One of the most notable outcomes is the exponential growth of data, driven by the improvement of storage capacity and the widespread adoption of digital technologies (Baig et al., 2019; Bhat, 2018). In recent years, medical data has seen a significant increase, mainly due to the growth of Electronic Health Records (EHRs), wearable devices, telemedicine, and clinical trials (Kindle et al., 2019; Mayo et al., 2017; M. Wu and Luo, 2019; P.-Y. Wu et al., 2016). Among these, EHRs, digital versions of traditional paper-based medical records, have played a crucial role in driving this growth by transforming the way healthcare providers store and access patient information, resulting in a marked increase in the volume of available medical data (Watson, 2016).

The digitization of medical data has significantly changed how healthcare providers and researchers approach patient care and medical research (Duggal et al., 2018). EHRs offer a comprehensive view of a patient's medical history, including test results, diagnoses, and treatment plans, allowing healthcare providers to make more informed treatment decisions, ultimately improving patient health outcomes (Burton et al., 2004). The benefits of digitalization extend to medical imaging as well, as digital medical

imaging allows for the efficient storage, analysis, and sharing of images. Researchers can now analyze large amounts of data, leading to new insights and understanding of various medical conditions, such as cancer, heart disease, and neurological disorders (Houssein et al., 2021; Noor et al., 2020; Ribeiro et al., 2022). Furthermore, digital medical data has allowed public health organizations to monitor and manage population health. For instance, digital contact tracing apps, which use Bluetooth technology, enable public health organizations to trace close contacts between individuals who have tested positive for COVID-19, providing crucial information for quickly identifying and isolating new cases, and slowing the spread of the virus. Healthcare providers also have the ability to quickly access and share information about patients who have tested positive for COVID-19 (Alsunaidi et al., 2021; Mbunge et al., 2021).

Medical data analysis has gained significant importance in recent years due to the exponential growth in the volume and complexity of medical data and the advancement in algorithms and analytics techniques (Raghupathi and Raghupathi, 2014; Y. Wang et al., 2018). The healthcare industry has adopted the use of Artificial Intelligence (AI) to analyze historical and current medical data, extract valuable insights, and develop new diagnostic tools and treatments (Shaheen, 2021; Yu et al., 2018). The sheer volume and complexity of medical data pose a major challenge for human experts to analyze and draw meaningful insights effectively. However, AI algorithms have demonstrated remarkable efficacy in processing and analyzing vast amounts of data accurately and at a much faster pace, thereby enabling quicker and more effective decision-making. The integration of AI in medical data analysis has the potential to revolutionize the healthcare industry (Davenport and Kalakota, 2019).

1.2 Artificial Intelligence

Artificial Intelligence (AI) is a rapidly developing field of technology that mimics human cognitive abilities through learning from large datasets. AI systems utilize algorithms to extract insights, make predictions, and self-correct based on feedback (Copeland, 2022). This technology has been widely adopted in a variety of fields, including social media personalized news feeds, voice-to-text conversion on smartphones, face recognition in cameras, grammar and spelling checks on computers, fraud detection in finance, and e-

commerce product searches (Choi et al., 2018; Dhieb et al., 2020; Gayed et al., 2022; Kozyreva et al., 2020; Pallathadka et al., 2021; Smith and Miller, 2022).

AI systems consist of three core elements: artificial intelligence, machine learning, and deep learning, as shown in Figure 1.1. Machine learning is a subset of AI that integrates computer science and statistics, enabling computers to learn by training models with data. The learning process of these machine learning algorithms can be broadly classified into four categories: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning (Rong et al., 2020). In supervised learning, the machine is trained on labeled data to infer a function mapping inputs to outputs. In unsupervised learning, the machine is trained on unlabeled data to predict outputs by identifying patterns in the input data. Semi-supervised learning combines both labeled and unlabeled data to build a prediction model. In reinforcement learning, the machine interacts with an environment to maximize rewards and minimize punishments, using feedback from its actions and experiences. The choice of learning category depends on the specific task and the nature of the data, and each category has its own strengths and limitations. This dissertation focuses on supervised learning, which can be further divided into regression and classification problems. Both regression and classification problems aim to construct a model that predicts the value of the output variable based on the input variables. The difference between the two is that the output variable is numerical in regression and categorical in classification.

Deep learning is a powerful subset of machine learning that draws inspiration from the information-processing patterns in the human brain (Litjens et al., 2017). The defining characteristic of deep learning that sets it apart from traditional machine learning is its ability to automatically learn hierarchical representations of data from massive training data without manual feature engineering (Janiesch et al., 2021). This makes deep learning particularly effective for high-dimensional and complex tasks, such as image recognition (Fujiyoshi et al., 2019; Islam et al., 2018; Y. Li, 2022), natural language processing (Le Glaz et al., 2021; Otter et al., 2020; L. Wu et al., 2023), and speech recognition (Mukhamadiyev et al., 2022; Nassif et al., 2019; Z. Zhang et al., 2018). There are several types of deep learning models, including artificial neural networks (ANNs), convolutional neural networks (CNNs), recurrent neural networks (RNNs),

and generative models. ANNs, also known as feedforward neural networks, are the simplest deep learning model and are used for simple classification problems (Saikia et al., 2020). CNNs are designed specifically for image and video recognition tasks (Chauhan et al., 2018; Hu et al., 2019). RNNs are used for sequential data processing, where the order of the data matters, such as speech or text (Ahmed et al., 2022; T. Kumar et al., 2022). Generative models are used to create new data similar to the training data and are useful for tasks such as image synthesis, text generation, and music composition (Louie et al., 2022; Sarkar et al., 2021; Shahriar, 2022).

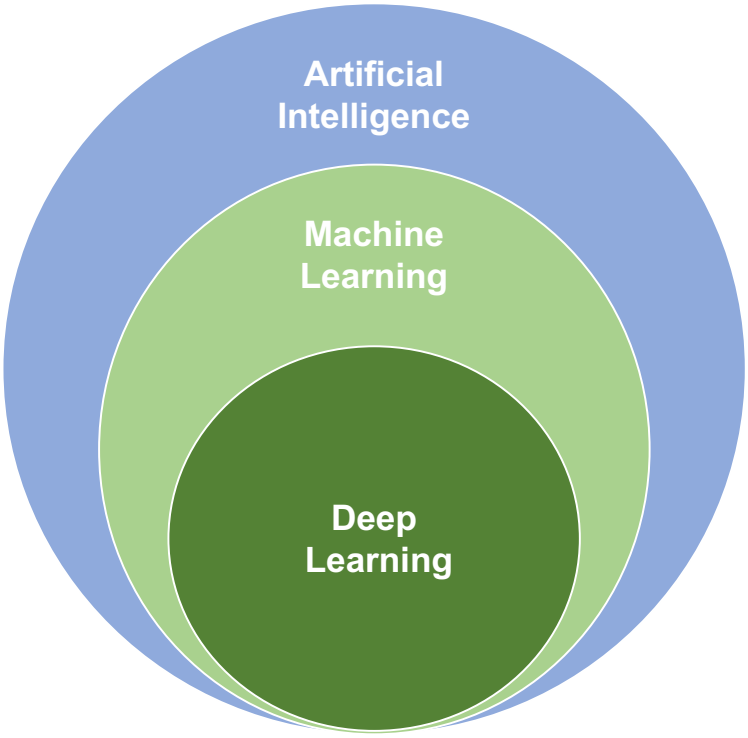


Figure 1.1: Artificial Intelligence, Machine Learning, Deep Learning.

1.3 Artificial Intelligence in Medical Data Analysis

The integration of AI in medical data analysis has been growing in recent years due to its efficiency in processing and analyzing large amounts of data. Although it is widely believed that AI will not completely replace healthcare professionals, studies have shown that AI can perform tasks such as cancer screenings

and disease diagnoses with the same accuracy or even better than human performance (Miller and Brown, 2018; Yu et al., 2018). The current state of the healthcare industry is facing challenges, including a shortage of skilled professionals, rising costs, and an aging population, putting a strain on healthcare professionals and increasing the risk of misdiagnosis (Drennan and Ross, 2019; Rother, 2017; Rowe et al., 2016). Advances in computer science and computing power have made it possible for AI to improve patient care through more accurate diagnoses, personalized treatment, preventive measures, patient monitoring, and cost reduction (Fujita, 2020; Jeddi and Bohr, 2020; Johnson et al., 2021).

AI has numerous applications in healthcare, one of which is its ability to analyze medical images such as Computed Tomography (CT) scans, Magnetic Resonance Imaging (MRI) scans, and X-rays to identify diseases and anomalies with accuracy (Çallı et al., 2021; Y. Chen et al., 2022; Gozes et al., 2020). Furthermore, AI can aid healthcare professionals in making faster and more accurate diagnoses by analyzing large amounts of patient data (Lauritsen et al., 2020; Lin et al., 2020). The automation of tasks like appointment scheduling, medical coding, and insurance claims processing through AI can provide healthcare professionals with more time to focus on delivering high-quality patient care (Ala and Chen, 2022; N. Kumar et al., 2019; Teng et al., 2020). Additionally, AI can predict disease patterns and epidemiology, aiding in pandemic control (A. Kumar et al., 2020; Vaishya et al., 2020).

AI can benefit not only healthcare professionals but also patients. The use of telemedicine and related applications, which collect data from wearable sensors and serve as inquiry systems, has become increasingly popular. These AI systems can be designed for speech recognition and oral or text communication (Fadhil, 2018; Pieczynski et al., 2021). They can provide treatment recommendations or direct the information to healthcare professionals.

However, it is essential to acknowledge that there are several limitations to the implementation of AI in healthcare, including data privacy, data integration, and ethical considerations, among others (Bartoletti, 2019; Dubovitskaya et al., 2019). Data privacy is critical in the healthcare industry as sensitive personal information, such as medical histories and genetic information, must be protected. The use of AI in healthcare requires large amounts of data to train algorithms, which can lead to privacy violations

if proper security measures are not in place. Additionally, integrating data from multiple sources, such as hospitals, clinics, and wearable devices, presents challenges and requires robust data management systems (De Giacomo et al., 2007; P.-Y. Wu et al., 2016). Furthermore, the accuracy and reliability of AI algorithms must be thoroughly evaluated to avoid errors in diagnosis or treatment, which can have severe consequences for patients. Ethical considerations, such as bias and transparency, must also be addressed to ensure the responsible and equitable use of AI (Holmes et al., 2021).

1.4 Organization of Dissertation

This dissertation presents three projects that focus on developing new methods of artificial intelligence and their applications in medical data analysis.

The first project, presented in Chapter 3, introduces a privacy preserving and communication efficient information enhancement procedure for medical image classification. The proposed approach tackles the challenges of integrating data and protecting privacy that arises in the application of AI in medical analysis. The lack of a consistent data management system can hinder the augmentation of medical image datasets from multiple sources, and direct access to medical images from external sources such as hospitals or healthcare institutions can pose privacy and communication risks. Additionally, the available training images may be limited and have imbalanced label issues, making it challenging to train deep CNN models. To overcome these limitations, the proposed method combines transfer learning and artificial sampling from generative models, resulting in a method that improves the performance of deep CNN-based classification models while ensuring the privacy of patients' data. This is demonstrated through an application for pediatric pneumonia detection using chest X-rays.

The second project, presented in Chapter 4, proposes a new tensor factor model for improving image classification. The model is designed to handle the challenge of overlapping in image classification, where only a small portion of the image contains information specific to a class and the rest of the image is dominated by noise. The proposed method decomposes the tensor data into two components: a pervasive component that is common across all classes, and a specific component that varies from class to class. The

pervasive component is modeled using a low-rank latent tensor factor and a few factor loading matrices, estimated through principal component analysis. The proposed tensor factor model has been shown to be effective through both synthetic experiments and an application to COVID-19 pneumonia diagnosis from chest X-ray images.

The third project, presented in Chapter 5, explores the use of machine learning methods for forecasting daily deaths caused by COVID-19 in the United States. The proposed models incorporate exogenous medical information-related variables, such as COVID-19 hospitalizations and the number vaccinated, to enhance their prediction performance. By combining information from lagged variables and exogenous medical variables, the proposed models have the potential to improve prediction accuracy and aid in monitoring the spread of the virus.

Chapter 2 provides a comprehensive review of the relevant research background and literature in the fields of artificial intelligence. It covers the key concepts, theories, and methods related to the three projects and provides a foundation for the research presented in the following chapters. Finally, Chapter 6 provides a conclusion and discussion of the key findings and contributions of the three projects presented in this dissertation.

Chapter 2

Research Background and Literature Review

In this chapter, we will provide a comprehensive overview of the core concepts of machine learning, starting with an introduction to two of the most important types of neural networks: artificial neural networks and convolutional neural networks. We will delve into the fundamental building blocks of these networks, including activation functions, forward and backward propagation, and regularization. This will help readers to understand how these neural networks work and their respective strengths and weaknesses. In the second part of this chapter, we will introduce the tensor factor model, a powerful tool for decomposing high-dimensional data. Specifically, we will discuss three tensor factorization techniques: principal component analysis (PCA), CP decomposition, and Tucker decomposition. We will explain how these techniques can be used to analyze complex datasets and extract meaningful information. Overall, this chapter lays a solid foundation for understanding the core concepts of neural networks and tensor factorization. This knowledge will be crucial for understanding the subsequent chapters of this dissertation.

2.1 Neural Networks

Neural Networks (NNs) are a subcategory of machine learning algorithms inspired by the structure and function of the human brain in processing data and creating patterns. They consist of multiple layers and are designed to capture and model complex patterns of large-scale data (Voulodimos et al., 2018). NNs have been proven to be a powerful learning method with widespread applications in many fields, such as natural language processing (Goldberg, 2016, 2017), speech recognition (Deng et al., 2013; Nassif et al., 2019), and image processing (Kasar et al., 2016; Naranjo-Torres et al., 2020). Several types of NNs have been created to address different problems, including Artificial Neural Networks, Convolutional Neural Networks, and Generative Adversarial Networks.

2.1.1 Artificial Neural Network

Artificial Neural Network (ANN) is a specific type of neural work that utilizes artificial intelligence methods. They are designed to use a layered structure to simulate the behavior of biological neurons in the human brain, a simple example shown in Figure 2.1. Such an architecture consists of three key components: (1) a single input layer, which receives the input data and passes it on to the next layer in the network. For example, if feed a grayscale image of size 8×8 into the network, the input layer should have 64 nodes, corresponding to the 64 pixels; (2) a single output layer, which produces the final prediction based on the processed input data. According to the task requirement, it can output numerical values (Specht et al., 1991), class labels (Dreiseitl and Ohno-Machado, 2002), or both (Xu et al., 2019); (3) one or more hidden layers, which process and transform the input data, allowing the network to learn complex relationships between the inputs and outputs.

In an ANN, each layer is composed of several neurons. These neurons are connected to each neuron in the neighboring layers and can receive inputs from multiple stimuli. Every neuron in an ANN has two key components that determine its functionality: a linear function and a non-linear activation function. The linear function is defined by two parameters, weights w_i and bias b. Weights are values that control

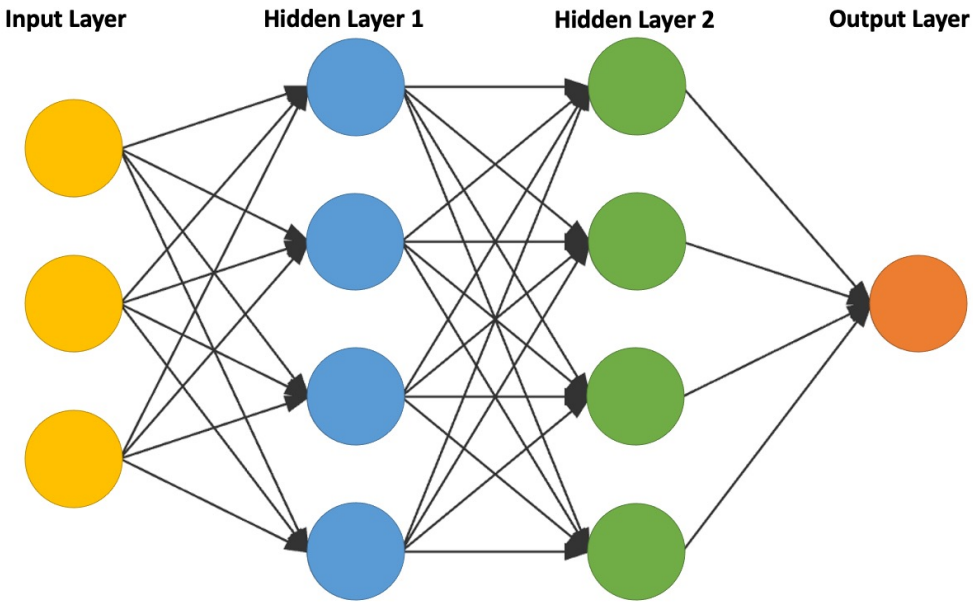


Figure 2.1: Artificial Neural Network architecture example with two hidden layers.

the strength of the connection between two neurons, while bias is a constant that is added to the weighted input before the activation function is applied. The activation function then non-linearly transforms the weighted and biased input to produce the output of the neuron. In other words, the output of a neuron can be formulated as follows:

$$f\left(x_i, w_i, b\right) = \varphi\left(\sum_{i=1}^{n}\left(w_i \times x_i\right) + b\right),$$

where x_i is the i^{th} input, w_i is the i^{th} weight of the neuron, b is the bias, φ is the activation function and n is the number of inputs. The central idea behind this is to extract linear combinations of the inputs as derived features and then model the target as a nonlinear function of these features (Hastie et al., 2009). Doing so can increase the expressiveness and flexibility of the model and enables the model to capture more complex relationships between the inputs and the target.

Activation Functions

The activation function is a critical factor in determining the behavior of neurons in an ANN. It determines the activation status of a neuron based on the inputs received. As per the findings of Ramachandran et al., 2017, the choice of activation function can significantly impact the training efficiency and convergence characteristics of the neural network. Hence, the activation function is regarded as a hyperparameter that can be adjusted to enhance the modeling accuracy.

The activation functions of interest for this dissertation (Figure 2.2) are as follows:

- *Sigmoid Function*

$$\sigma(x) = \frac{1}{1 + \exp(-x)},$$

  which maps real-valued numbers to values within the range of $[0, 1]$. This function is widely used in binary classification problems to estimate the likelihood of a specific event occurring. However, one of the major drawbacks of the sigmoid function is that it can cause the gradient to vanish. As the input values move away from zero, the gradient of the sigmoid function becomes increasingly small, making it difficult for the model to update its weights during training. This can slow down or hinder the convergence of the model.

- *Tanh Function*

$$\sigma(x) = \tanh(x),$$

  transforms any real number into a value within the range of $[-1, 1]$. This function has similarities with the Sigmoid activation function, but its output is centered around zero. The advantage of using the Tanh function over the Sigmoid function is that it prevents the issue of vanishing gradients, as the derivative of the Tanh function always remains between -1 and 1.

- *Rectified Linear Unit (ReLU)* (Nair and Hinton, 2010)

$$\sigma(x) = \max(x, 0),$$

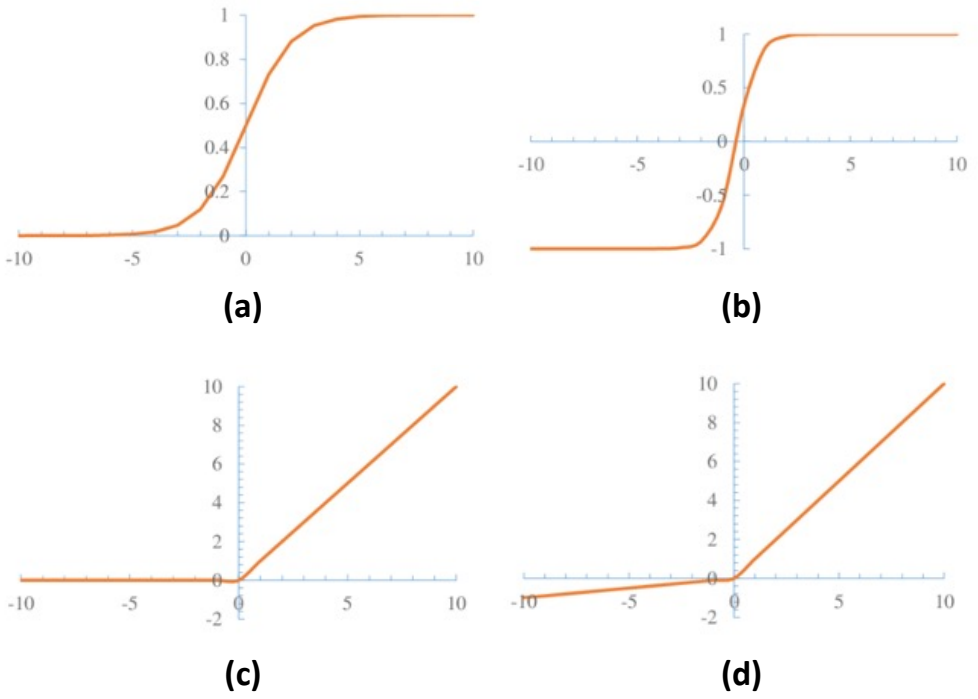


Figure 2.2: Non-linear activation functions: (a) Sigmoid; (b) Tanh (Hyperbolic Tangent); (c) ReLU (Rectified Linear Unit); (d) Leaky ReLU.

which is currently one of the most popular activations and is particularly popular for use in the hidden units of deep neural networks. This is due to its ability to reduce complexity and computational time. Additionally, the derivative of a sigmoid function at the tail ends of its domain becomes unreliable as it flattens. In contrast, the ReLU activation always provides a strong direction of change, making it a more effective choice than the sigmoid function.

- *Leaky ReLU* (Maas et al., 2013)

$$\sigma(x) = \begin{cases} x & x \geq 0 \\ \alpha x & x < 0 \end{cases},$$

which is an extended version of the ReLU activation function and addresses the issue of "dying ReLU". Unlike ReLU, which sets all negative inputs to zero, Leaky ReLU assigns a slight negative slope to negative inputs. This helps ensure that the activation function never becomes completely

inactive, even for negative inputs. The "dying ReLU" problem occurs when the activation function in ReLU gets stuck at zero, leading to dead neurons that no longer respond to inputs. By assigning a small negative slope to negative inputs, Leaky ReLU helps address this problem, making it a more robust activation function.

- *SoftMax Function*

$$\sigma\left(x_j\right) = \frac{e^{x_j}}{\sum_{i=1}^{k} e^{x_i}},$$

which is a common choice for the final layer of a neural network in multi-class classification problems. It transforms the outputs from the last layer of neurons into a probability distribution across all classes, allowing the model to make predictions about the likelihood of each class. By doing so, the softmax activation helps to provide a clear and interpretable representation of the model's confidence in each class.

Forward and Backward Propagation

The process of forward propagation in neural networks is a method for calculating the output of a network for a given input (Deng, Yu et al., 2014). It involves passing the input through each layer of neurons, determining the result of each neuron, and finally determining the final output. To evaluate the performance of the model, the final output is compared to the actual output using a loss function, which measures the difference between the predicted and actual output. The form of the loss function is dependent on the activation function chosen for the network. For example, in a binary classification problem with a sigmoid activation function in the last fully connected layer, the binary cross-entropy loss function would be used.

Neural networks update their weights through learning algorithms in order to minimize the loss function. Backpropagation (Hecht-Nielsen, 1992) is a widely used learning algorithm that computes the gradient of the loss function with respect to the network's weights using the chain rule. Backpropagation starts at the final output and moves backward through the network, updating the weights in each layer in order to minimize the loss. The optimization process is usually performed using a gradient-based

optimizer, such as Stochastic Gradient Descent (SGD), Root Mean Square Propagation (RMSprop), and Adaptive Moment Estimation (Adam) (Graves, 2013; Kingma and Ba, 2017; Ruder, 2016).

Regularization

Deep neural networks often suffer from overparameterization, where the number of parameters in the model is much larger than the size of the training dataset. This can result in overfitting, where the model becomes excessively complex and focuses on memorizing the training set, including any noise and outliers, rather than learning the general patterns. This leads to poor generalization performance, as the model is unable to make accurate predictions on new, unseen data. To address this challenge, various regularization techniques are commonly used during the training process to control the model's complexity and prevent overfitting.

- *Weight Regularization*: Implement a penalty term in the loss function to prevent the weights and biases of the network from becoming too far from zero. This is commonly achieved through L_2 regularization, which is the most widely used form of weight regularization.

- *Dropout*: This technique involves randomly disregarding or dropping out a predefined portion of neurons during each forward pass of the training process. This forces the network to learn multiple independent representations of the same data, reducing overfitting and improving generalization.

- *Batch Normalization*: This technique normalizes the activations of each layer in the network, reducing the internal covariate shift and making the optimization problem more manageable and easier to solve. This can help prevent overfitting by stabilizing the distribution of activations in each layer.

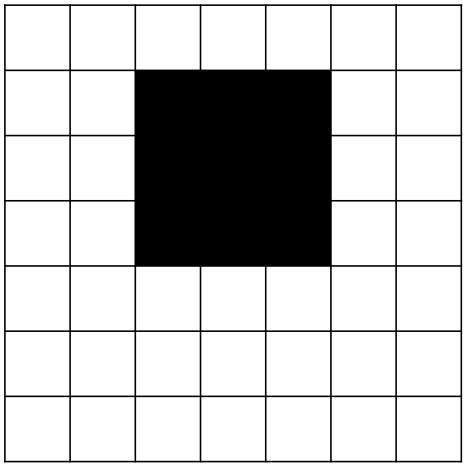
- *Early Stopping*: The technique involves monitoring the performance of the model on a validation set, which is a separate dataset used to evaluate its generalization capabilities. The training process should be stopped when the performance on the validation set stops improving. This helps prevent overfitting by avoiding the scenario where the model continues to learn the noise present in the training set.

- *Data Augmentation*: The size of the training set can be artificially increased by generating new, transformed versions of existing data. This helps the model learn a more robust representation of the data, reducing the risk of overfitting.

- *Transfer learning*: This technique utilizes the advantages of a pre-trained model by fine-tuning it for a new task. The pre-trained model serves as a solid foundation, as it has already acquired a general understanding of features from a substantial amount of data, thereby reducing the risk of overfitting.

2.1.2 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are a type of Neural Network architecture that is specifically designed for image and video data processing. Their popularity in the fields of Machine Learning and Computer Vision has been greatly influenced by the pioneering work of Krizhevsky et al., 2012b, which demonstrated exceptional performance in image classification on the ImageNet dataset (Russakovsky et al., 2015).



(a)

255	255	255	255	255	255	255
255	255	0	0	0	255	255
255	255	0	0	0	255	255
255	255	0	0	0	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255

(b)

Figure 2.3: Example of grid-like data processed by Convolutional Neural Network: (a) 7×7 single-channel grayscale image of a square figure; (b) 2-way tensor representation of the image with pixel brightness values.

The idea behind the inception of CNNs was to emulate the functionality of the visual cortex in cats, as described by LeCun et al., 1998, which is composed of layers of simple and complex cells that progressively extract more complex features from an image as it is processed (Hubel and Wiesel, 1968). As a result, the architecture of CNNs consists of convolutional layers, which perform simple operations, and pooling layers, which detect complex features. These networks are trained through a series of operations to learn hierarchical representations of the input data automatically, making them highly suitable for processing grid-like data with structural information, such as 1D audio signals, 2D images, and 3D videos (Aleshin-Guendel and Alvarez, 2017). In Figure 2.3, an instance of such grid-like data is presented. The illustration on the left displays a single-channel grayscale image, consisting of a square shape, with each cell in the grid representing a single pixel. The corresponding brightness values for each pixel are depicted in the number grid on the right, where each pixel's brightness is quantified using a numerical value that ranges from 0, representing no color, to 255, representing full saturation.

The fundamental components of a CNN consist of three types of layers, which are convolutional layers, pooling layers, and fully connected layers.

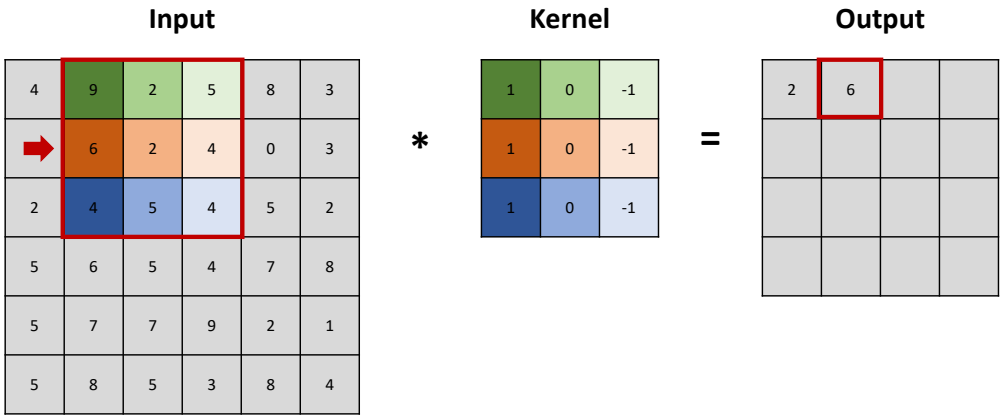


Figure 2.4: Convolution operation with a 3×3 kernel on a 6×6 input, resulting in a 4×4 output.

Convolutional Layer

Convolutional layers are essential building blocks of convolutional neural networks (CNNs), aimed at extracting salient features from input images. These layers employ filters, or kernels, designed to learn local

patterns, edges, shapes, and textures within the input image. Convolution operations involve sliding the kernel matrix over the input matrix, performing element-wise matrix multiplication at each location, and then summing up the results on a feature map. The resulting features capture the unique characteristics of the input image, enabling the network to make accurate predictions.

Figure 2.4 illustrates a basic convolutional operation wherein a 6×6 input matrix is convolved with a 3×3 kernel using a stride of 1, resulting in a 4×4 output feature map. The stride, which is a hyperparameter, determines the number of pixels a filter moves across the input matrix during a convolution operation. By adjusting the stride, the spatial dimensions of the output feature map can be controlled, reducing computational cost and regulating the overlap between adjacent receptive fields. By fine-tuning the size and number of kernels, various features can be extracted from input data, allowing the CNN to extract relevant features essential for its performance in different applications.

As depicted in Figure 2.4, convolutional operations lead to a reduction in the height and width of the output feature map relative to the input. This size reduction poses a challenge when applying multiple convolutional layers as the feature maps progressively decrease in size. The reduction in feature maps' size can make it difficult to extract features effectively from deep convolutional architectures. Additionally, the center of the kernel never overlaps with the outermost pixels of the input matrix. To address these challenges, some convolutional layers incorporate padding, which involves adding zero values to the edges of the input matrix. This method preserves the spatial dimensions of the input, retaining information at the edges to enhance feature extraction. For example, consider a 5×5 input image and a 3×3 filter. If a stride of 1 is applied and a fully padded convolution is used, 1 row and 1 column of zeros will be added to the input image, resulting in a 7×7 image. The filter is then convolved with the input image in a way that maintains the spatial dimensions of the input image in the output feature map.

In some cases, it may be necessary to expand the output size relative to the input. Transposed convolutions are a type of convolutional operation that can increase the dimensions of the input tensor by using a kernel with a larger receptive field and padding the output to maintain the desired dimensions. This technique has been successfully employed in deep learning architectures, such as Generative Adversarial

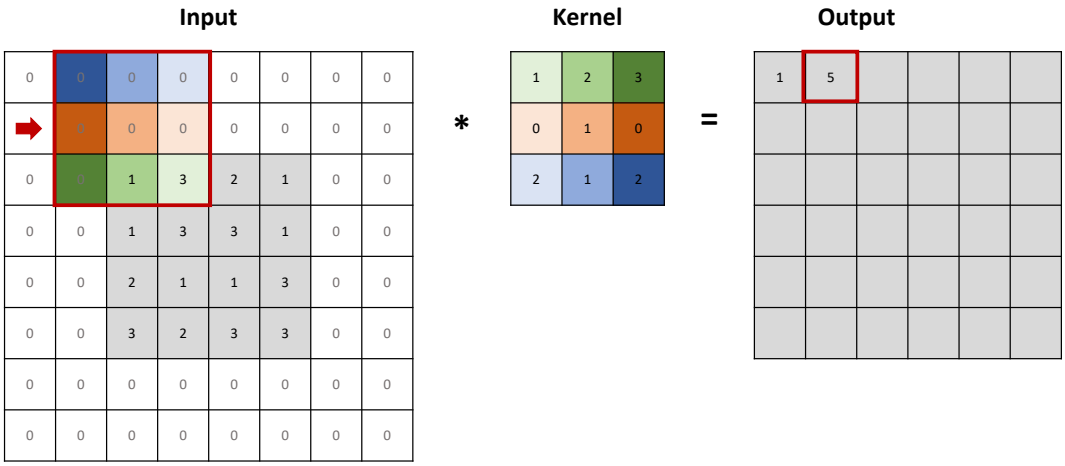


Figure 2.5: Transposed convolution operation with a 3×3 kernel on a 4×4 input, resulting in a 6×6 output.

Networks (GANs), to generate high-resolution images from low-resolution inputs (Goodfellow et al., 2014). Figure 2.5 presents an example of a transposed convolution operation.

Pooling Layer

The feature map output of a convolutional layer precisely records the position of input features, making the model sensitive to minor changes in input images. This property can lead to overfitting, and the resulting models may fail to generalize to new data. To address this issue, pooling layers are typically inserted after the nonlinearity layer. These layers reduce the spatial dimensions of the feature maps, leading to spatial invariance while preserving essential features. There are various pooling techniques, including max, average, and sum pooling, among others. Max pooling is the most common approach, which extracts the maximum activation value from each distinct region of the feature map. Pooling layers improve the computational efficiency of CNNs by reducing the number of parameters and the computational cost, while also controlling overfitting and improving the model's generalization capability. Figure 2.6 illustrates instances of two common pooling techniques: max pooling and average pooling.

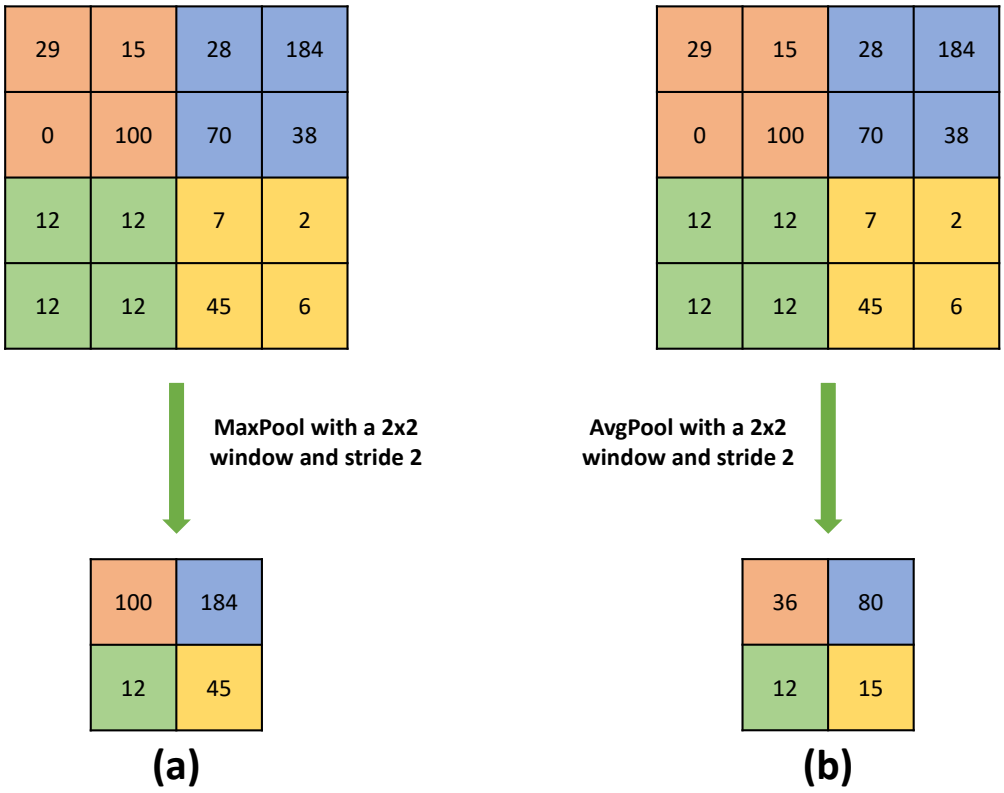


Figure 2.6: Example of a 2×2 (a) max pooling and (b) average pooling layer on a 4×4 feature map with stride 2.

Fully Connected Layer

The final stage of the CNN architecture, fully connected layers, is crucial for accurately predicting the class of an input image. These layers are also referred to as dense layers and are typically placed after several convolutional and pooling layers. They play a vital role in classifying the extracted features of the input image by transforming the output of previous layers into a probability vector that represents the likelihood of each potential class. The input to the fully connected layer is usually the output of the last pooling layer, and the number of neurons in this layer determines the number of features that the CNN can recognize and classify. However, the large number of parameters in fully connected layers can result in overfitting to the training data. To overcome this problem, dropout regularization is commonly utilized in fully connected layers, where a percentage of the neurons is randomly dropped during training.

2.2 Tensor Factor Models

The tensor factor model (TFM) is a statistical framework that has been developed to address the challenges of analyzing high-dimensional data that are represented as tensors. Tensors are a generalization of matrices that can contain any number of dimensions, allowing for a more flexible and expressive representation of complex relationships between variables. TFM is grounded on the premise that the observed tensor can be decomposed into a product of lower-dimensional tensors or matrices that are explicitly designed to capture the underlying structure of the data (Kolda and Bader, 2009). This approach provides a powerful tool for exploring hidden patterns and relationships that may be present in high-dimensional data, by utilizing a structured decomposition of the tensor into its constituent factors (Bacciu and Mandic, 2020). By reducing the dimensionality of the data, TFM facilitates a more interpretable representation of the data, allowing for more efficient and effective analysis of complex multidimensional data. In recent years, TFM has been applied to a diverse range of domains, including computer vision, medical imaging, natural language processing, and recommender systems (Bouchard et al., 2015; Hatvani et al., 2018; Panagakis et al., 2021; Symeonidis, 2016). A few popular types of TFM include principal component analysis (PCA), Tucker decomposition, and CP decomposition.

2.2.1 Tensor Definition and Preliminaries

Tensors are also known as multidimensional arrays or higher-order arrays, where the modes of a tensor correspond to the dimensions of a multidimensional array. Let D be a positive integer that specifies the number of modes for a tensor. A D-way tensor is an object that can be represented as a multidimensional array of elements in a field, such as $\mathbb{R}$. The extents of a tensor correspond to the sizes of each mode of the array. Let $p_1 \times p_2 \times \cdots \times p_D$ denote the extents associated with a D-way tensor. We denote tensors of a dimension of three or more by boldface Euler script letters, e.g., $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{p_1 \times p_2 \times \cdots \times p_D}$. Tensors of lower dimensions, such as matrices and vectors, can be considered as special cases of tensors. Matrices are two-way tensors and can be denoted by bold capital letters, e.g., $\mathbf{X} \in \mathbb{R}^{p_1 \times p_2}$. Similarly, vectors are

one-way tensors and can be denoted by bold lowercase letters, e.g., $\mathbf{x} \in \mathbb{R}^{p_1}$. Scalars are the simplest form of tensors and can be denoted by lowercase letters, e.g., $x \in \mathbb{R}$. Scalars are zero-way tensors.

One way to comprehend the structure of tensors is to view them as a collection of fibers or slices, where each fiber or slice corresponds to a particular dimension of the tensor. A fiber of a tensor is an array of numerical elements, each of which corresponds to a fixed value of all indices except one. In other words, a fiber is obtained by holding all indices of the tensor constant, except for one, and thus represents a one-dimensional subset of the tensor. On the other hand, a slice of a tensor is an array of numerical elements, each of which corresponds to a fixed value of all indices except two. Slices are obtained by holding all indices of the tensor constant, except for two, and thus represent a two-dimensional subset of the tensor. For example, consider a three-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{p_1 \times p_2 \times p_3}$. Each mode-1 fiber of $\boldsymbol{\mathcal{X}}$ is a one-dimensional array of p_1 numerical elements, corresponding to a fixed pair of values of the second and third indices. Each frontal slice of $\boldsymbol{\mathcal{X}}$ is a two-dimensional array of size $p_1 \times p_2$, corresponding to a fixed value of the third index. Graphical examples of fibers and slices for a 3-way tensor are given in Figure 2.7.

In addition to fibers and slices, two other important concepts related to tensors are matricization and vectorization.

Definition 1 (Matricization)**.** *The mode-d matricization (unfolding) of $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{p_1 \times p_2 \times \cdots \times p_D}$ is denoted as $\boldsymbol{\mathcal{X}}^{(d)} \in \mathbb{R}^{p_d \times \prod_{j \neq d} p_j}$ and arranges the mode-d fibers of $\boldsymbol{\mathcal{X}}$ into the columns of $\boldsymbol{\mathcal{X}}^{(d)}$. Formally, the $(i_1, i_2, \ldots, i_D)^{th}$ tensor element maps to matrix element $(i_d, j)^{th}$, where*

$$j = 1 + \sum_{m \neq d}^{D} (i_m - 1)\, J_m,\ J_m = \prod_{q \neq d}^{m-1} p_q.$$

For a three-way tensor, there are three mode-d matricizations, denoted $\boldsymbol{\mathcal{X}}^{(1)}$, $\boldsymbol{\mathcal{X}}^{(2)}$, and $\boldsymbol{\mathcal{X}}^{(3)}$, all of which are demonstrated in Figure 2.8.

Definition 2 (Vectorization)**.** *Let $\mathbf{X} \in \mathbb{R}^{m \times n}$ be an $m \times n$ matrix with $\mathbf{a}_i$ as the i-th column of $\mathbf{X}$. The $\mathrm{vec}(\cdot)$ operator turns $\mathbf{X}$ into a column vector as follows*

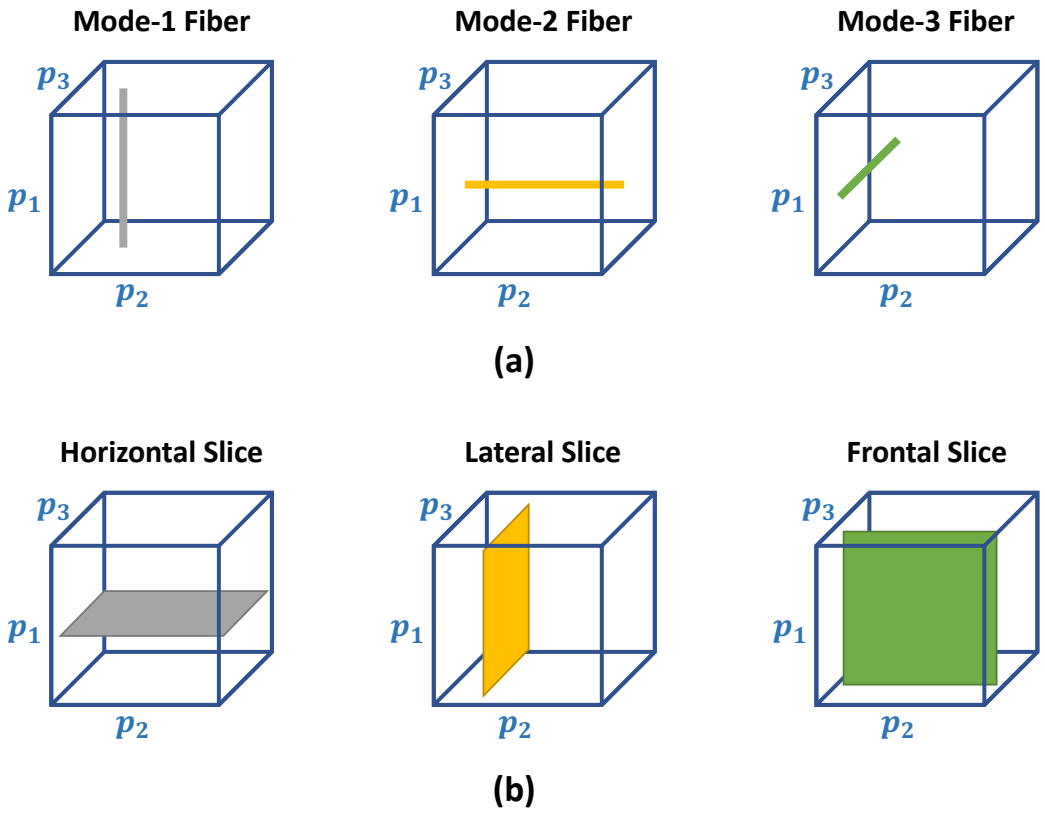


Figure 2.7: Example of fibers and slices for a three-way tensor. (a) tensor fibers: (left) mode-1 fibers; (middle) mode-2 fibers; (right) mode-3 fibers; (b) tensor slices: (right) horizontal slices; (middle) lateral slices; (right) frontal slices.

$$\text{vec}(\mathbf{X}) = \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_n \end{pmatrix}.$$

Similarly, we can define the vectorization of a D-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{p_1 \times p_2 \times \cdots \times p_D}$ by vectorizing the mode-1 matricization of $\boldsymbol{\mathcal{X}}$, i.e.,

$$\text{vec}(\boldsymbol{\mathcal{X}}) = \text{vec}(\boldsymbol{\mathcal{X}}^{(1)}).$$

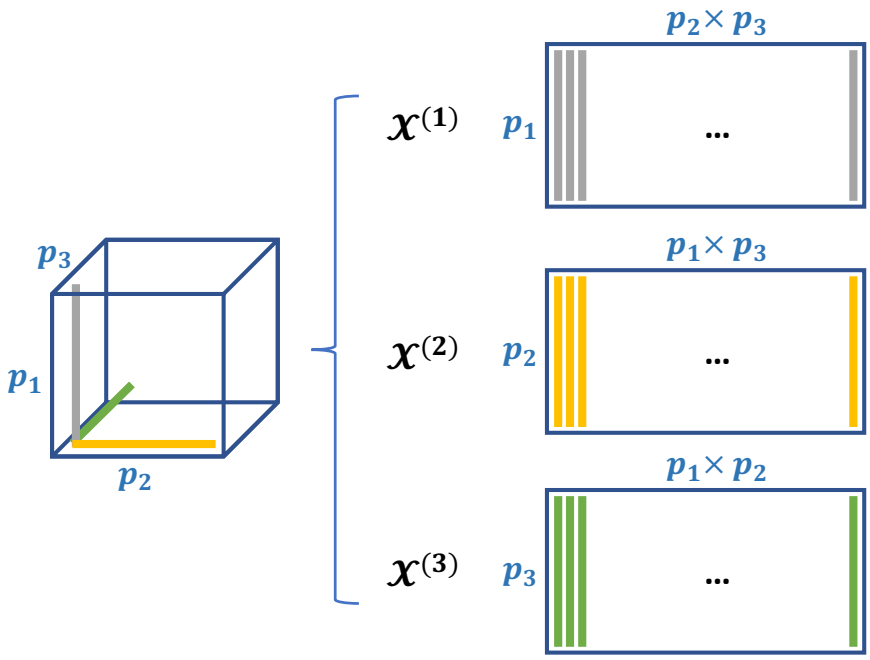


Figure 2.8: Mode-1, Mode-2, and Mode-3 matricizations for a three-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{p_1 \times p_2 \times p_3}$.

For example, the vectorization of a three-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{p_1 \times p_2 \times p_3}$ is obtained by stacking its fibers along the first mode, resulting in a vector of length $p_1 p_2 p_3$.

Some Tensor Operations

Definition 3. *An orthogonal matrix* $\mathbf{Q} \in \mathbb{R}^{n \times n}$ *has orthonormal columns which means that* $\mathbf{q}_i^T \mathbf{q}_j = 0$ *and* $\mathbf{q}_i^T \mathbf{q}_i = 1 \; \forall \; i, j \in 1, \ldots, n$. *Equivalently,*

(i) $\mathbf{Q}^T \mathbf{Q} = \mathbf{Q}\mathbf{Q}^T = \mathbf{I}_n$

(ii) $\|\mathbf{Q}\mathbf{x}\| = \|\mathbf{x}\|$ *for all* $\mathbf{x} \in \mathbb{R}^n$

(iii) $\mathbf{Q}^T = \mathbf{Q}^{-1}$

(iv) The columns of $\mathbf{Q}$ *are an orthonormal basis for* $\mathbb{R}^n$

Theorem 2.1 (The Spectral Theorem.)**.** *Every symmetric matrix has the form,*

$$\mathbf{S} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$$

Note that $\mathbf{Q}$ represents the orthogonal eigenvector matrix of $\mathbf{S}$, and $\mathbf{\Lambda}$ denotes the diagonal matrix of corresponding eigenvalues. Singular Value Decomposition (SVD) is considered a generalization of The Spectral Theorem, which provides a means of analyzing non-symmetric and non-square matrices.

Theorem 2.2. *For an $m \times n$ matrix $\mathbf{A}$ with rank of r, the Singular Value Decomposition (SVD) of $\mathbf{A}$ is,*

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T \quad \textit{s.t } \sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r$$

The dimensions of the matrices $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{V}$ are determined by the size of $\mathbf{A}$, and since $\mathbf{A}$ is not necessarily square, the columns of $\mathbf{U}$ are orthogonal in $\mathbb{R}^m$, and the columns of $\mathbf{V}$ are orthogonal in $\mathbb{R}^n$. Therefore, the vectors of $\mathbf{U}$ and $\mathbf{V}$ are commonly referred to as the "left singular vectors" and "right singular vectors", respectively. The matrix $\mathbf{\Sigma}$ is a diagonal matrix whose entries correspond to the singular values of $\mathbf{A}$.

Definition 4. *The inner product of two tensors of the same modes $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \in \mathbb{R}^{p_1 \times p_2 \times \cdots \times p_D}$ is the sum of the products of their elements, i.e.,*

$$\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \rangle = \sum_{i_1 \ldots i_D} x_{i_1 \ldots i_D} y_{i_1 \ldots i_D}.$$

This will define the Frobenius norm of tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{p_1 \times p_2 \times \cdots \times p_D}$ extended from the matrix case as follow:

$$\|\boldsymbol{\mathcal{X}}\|_F = \sqrt{\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{X}} \rangle} = \left(\sum_{i_1 \ldots i_D} x_{i_1 \ldots i_D}^2 \right)^{1/2}.$$

Definition 5. *The outer product of D vectors $\mathbf{a}^{(1)} \in \mathbb{R}^{p_1}, \ldots, \mathbf{a}^{(D)} \in \mathbb{R}^{p_D}$ is an D-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{p_1 \times p_2 \times \cdots \times p_D}$ whose elements are defined as*

$$x_{i_1 \ldots i_D} = a_{i_1}^{(1)} \ldots a_{i_D}^{(D)}, \quad 1 \leq i_j \leq p_j$$

A D-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{p_1 \times p_2 \times \cdots \times p_D}$ is of rank on if it can be strictly decomposed into the outer product of D vectors, i.e.,

$$\boldsymbol{\mathcal{X}} = \mathbf{a}^{(1)} \circ \ldots \circ \mathbf{a}^{(D)}.$$

Figure 2.9 illustrates the rank one three-way tensor $\boldsymbol{\mathcal{X}} = \mathbf{a} \circ \mathbf{b} \circ \mathbf{c}$.

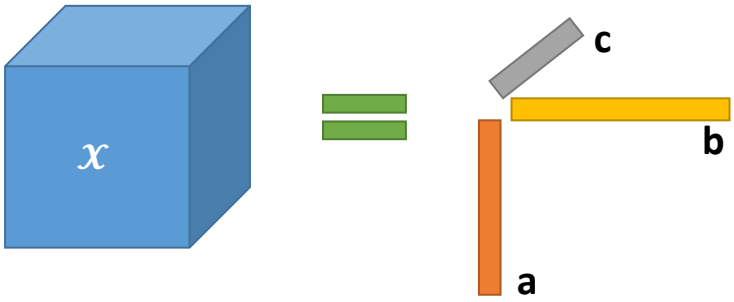


Figure 2.9: Rank 1 Three-way Tensor.

Definition 6. *For two matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{p \times q}$, the Kronecker product (denoted by $\otimes$) is given by*

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{pm \times qn}.$$

Definition 7. *The mode-d product between a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{p_1 \times \cdots \times p_D}$ and a matrix $\mathbf{A} \in \mathbb{R}^{r_d \times p_d}$ is denoted by $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{X}} \times_d \mathbf{A} \in \mathbb{R}^{p_1 \times \cdots \times p_{dn-1} \times r_d \times p_{d+1} \cdots \times p_D}$. Element-wise, this operation can be expressed as follows*

$$y_{i_1 \cdots i_{d-1} j_d i_{d+1} \cdots i_D} = \sum_{i_d} x_{i_1 \cdots i_D} a_{i_d j_d}.$$

This product definition is closely related to the mode-d matricization, given that:

$$\mathcal{Y} = \mathcal{X} \times_d \mathbf{A} \quad \Leftrightarrow \quad \mathcal{Y}^{(d)} = \mathbf{A} \cdot \mathcal{X}^{(d)}.$$

2.2.2 Principal Component Analysis

Principal Component Analysis (PCA) is a widely employed technique for dimensionality reduction in various fields, with the goal of identifying low-dimensional subspaces that capture the maximum amount of variation in the data (2017). PCA is a specific instance of TFMs, and it is closely related to SVD, with one essential step in PCA being to center the data.

Definition 8. *The Sample Covariance Matrix of* $\mathbf{A} \in \mathbb{R}^{n \times p}$ *is defined by*

$$\mathbf{S} = \frac{\mathbf{A}\mathbf{A}^\mathbf{T}}{n-1}.$$

Given a sample covariance matrix $\mathbf{S}$ of $\mathbf{A}$, we can apply SVD to $\mathbf{S}$, yielding $\mathbf{S} = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^\mathbf{T}$. Since $\mathbf{S}$ is symmetric, the columns of $\mathbf{U}$ are eigenvectors of $\mathbf{S}$, which are also the left singular vectors of $\mathbf{A}$. Subsequently, the columns of $\mathbf{U}$ become the principal components of $\mathbf{A}$ upon applying SVD to $\mathbf{A}$. Furthermore, the eigenvalues of $\mathbf{S}$ correspond to the squared singular values of $\mathbf{A}$, and the total variance of $\mathbf{A}$ is given by $\sum_{i=1}^{r} \sigma_i^2/(n-1)$. The first k singular vectors account for more variation in the data than any other set of k singular vectors, providing the primary motivation for dimensionality reduction.

Despite its widespread use, applying PCA to high-dimensional tensor data presents several limitations that can impede its effectiveness. One of the most significant drawbacks of PCA on tensor data is the increased computational complexity. While PCA was designed for matrix data that can be manipulated using standard linear algebraic methods, tensor data requires more advanced techniques, such as tensor algebra, to perform the necessary calculations. This increased complexity can render PCA impractical or

excessively time-consuming when dealing with high-dimensional tensor data. Furthermore, PCA cannot capture non-linear correlations between variables when applied to tensor data. PCA assumes that the relationships between variables are linear, which may not be the case in the context of tensor data. In many cases, the interrelations between variables in tensor data can be highly complex and non-linear, making it difficult for PCA to effectively capture these correlations (Zare et al., 2018). These limitations must be taken into account when selecting appropriate methods for analyzing high-dimensional tensor data.

2.2.3 CP Decomposition

Canonical Polyadic Decomposition (CPD), also known as PARAFAC, is a powerful technique to extract latent factors from high-dimensional tensors (Bro, 1997). The goal of CP decomposition is to represent the high-dimensional tensor as a sum of rank-one tensors, each corresponding to a latent factor. This representation allows us to uncover the underlying structure and patterns in the data.

Mathematically, under the CP format, a D-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{p_1 \times \cdots \times p_D}$ can be decomposed into a set of D matrices $\{\mathbf{U}_d\}_{d=1}^{D}$ sharing the same number columns as follows:

$$\boldsymbol{\mathcal{X}} \approx \hat{\boldsymbol{\mathcal{X}}} = \sum_{i=1}^{r} \mathbf{U}_1[:, i] \circ \cdots \circ \mathbf{U}_D[:, i],$$

where the tensor factor $\mathbf{U}_d$ is of size $p_d \times r$, with $1 \leq d \leq D$. $\mathbf{U}_d[:, i]$ denotes the i^{th} column of $\mathbf{U}_d$. The non-negative scalar value r is the CP-rank of $\boldsymbol{\mathcal{X}}$. $\hat{\boldsymbol{\mathcal{X}}}$ is the reconstruction of the original tensor. An illustration of an approximate CP-decomposition is given in Figure 2.10.

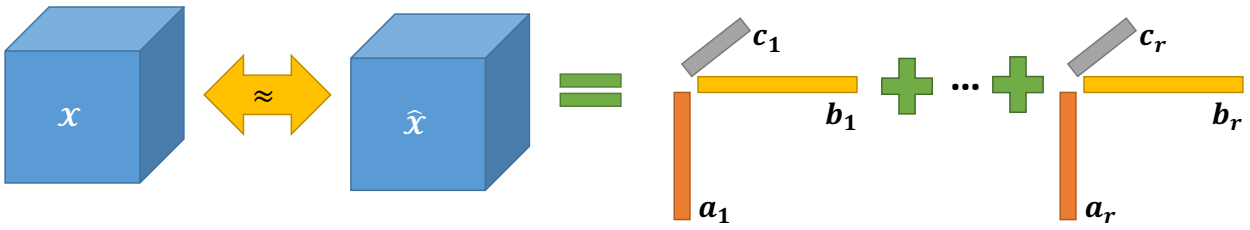


Figure 2.10: CP decomposition of a three-way tensor $\boldsymbol{\mathcal{X}}$ into r components.

The ultimate aim of CP decomposition is to construct a rank-r tensor that can further minimize the Frobenius norm of the difference between the original tensor and its approximation, i.e., $\min_{\hat{\boldsymbol{\mathcal{X}}}} \|\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}}\|_F$. This can be achieved using various algorithms, but the Alternating Least Squares (ALS) algorithm is the most common approach used in practice (Kolda and Bader, 2009). The ALS algorithm is an iterative optimization approach that optimizes a single factor matrix while keeping all other factor matrices fixed. This process is repeated for every factor matrix until a specified stopping criterion is reached.

The main limitation of CP decomposition is the computationally-intensive task of determining the true rank r of a tensor. This is because computing the rank of a tensor is a problem known to be NP-hard, and there is no trivial algorithm for solving it (Rabanser et al., 2017). Even if the CP-rank is known in advance, it is not guaranteed that the best rank-r approximation of a tensor exists (De Silva and Lim, 2008). As a result, many algorithms used in practice attempt to fit multiple ranks and select the best approximation.

2.2.4 Tucker Decomposition

The Tucker decomposition is a tensor factorization method that, similar to the CP decomposition, also aims to obtain the best approximation of a given tensor. However, in contrast to the CP decomposition, which expresses the approximation as a sum of r rank-one tensors, the Tucker decomposition decomposes the original tensor into a core tensor that has different scalings along each mode (Tucker, 1966). This method is analogous to the approach used in PCA and is often referred to as higher-order PCA. By doing so, the Tucker decomposition captures the essential features of the original tensor, which can be used for tasks such as subspace estimation, compression, or dimensionality reduction (Rabanser et al., 2017).

Mathematically, the Tucker decomposition constructs $\hat{\boldsymbol{\mathcal{X}}}$ to approximate $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{p_1 \times \cdots \times p_D}$ using a reduced core tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{r_1 \times \cdots \times r_D}$ and D factor matrices $\{\mathbf{U}_d \in \mathbb{R}^{p_d \times r_d}\}_{d=1}^{D}$:

$$\boldsymbol{\mathcal{X}} \approx \hat{\boldsymbol{\mathcal{X}}} = \boldsymbol{\mathcal{G}} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \ldots \times_D \mathbf{U}_D,$$

where $r_d \leq p_d$. The vector $\mathbf{r} = [r_1, r_2, \ldots, r_D]$ is called the multilinear rank of rank-$(r_1, r_2, \ldots, r_D)$ of $\boldsymbol{\mathcal{X}}$. Figure 2.11 depicts the model for the three-way tensor decomposition.

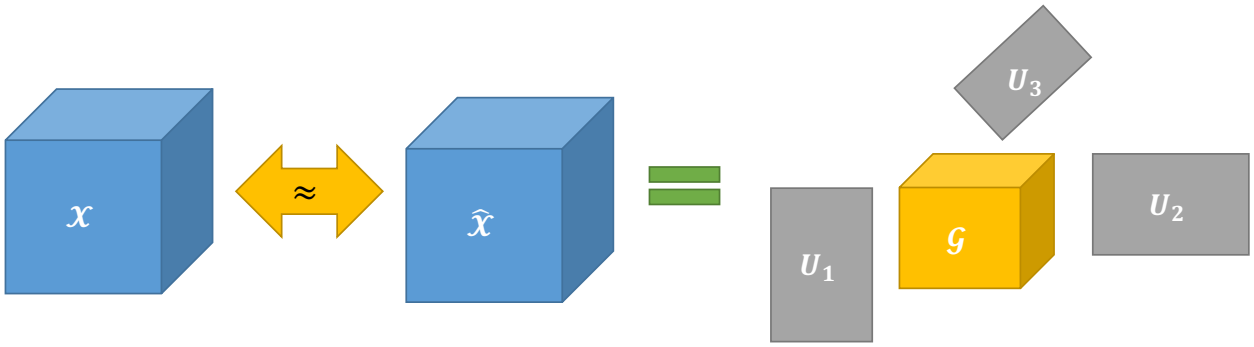


Figure 2.11: Tucker decomposition of a three-way tensor $\boldsymbol{\mathcal{X}}$ into a core tensor $\boldsymbol{\mathcal{G}}$ and factor matrices $\mathbf{U}_1$, $\mathbf{U}_2$, and $\mathbf{U}_3$.

It is noteworthy that the general Tucker decomposition is not inherently guaranteed to generate a unique solution. Hence, it is imperative to impose specific constraints on the factor matrices to ensure the uniqueness of the resulting solution. To achieve a unique solution, various approaches have been proposed, including the Higher Order Singular Value Decomposition (HOSVD) and the Higher Order Orthogonal Iteration (HOOI) (De Lathauwer et al., 2000). The HOSVD algorithm entails the computation of the singular value decomposition (SVD) of each mode matricization of the tensor, resulting in the formation of a set of orthogonal matrices for each mode. Subsequently, these orthogonal matrices are employed to establish the core tensor and factor matrices, which represent the original tensor in a compressed form. Algorithm 1 provides a detailed representation of the HOSVD algorithm.

Algorithm 1 Higher Order Singular Value Decomposition (HOSVD)

Input: A tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{p_1 \times \cdots \times p_D}$, and desired ranks $r_1, \ldots, r_D$.

1: **for** $d = 1$ to D **do**
2: Compute the mode-d matricization $\boldsymbol{\mathcal{X}}^{(d)}$ of $\boldsymbol{\mathcal{X}}$.
3: Compute the SVD of $\boldsymbol{\mathcal{X}}^{(d)}$, $\boldsymbol{\mathcal{X}}^{(d)} = \mathbf{U}_d \boldsymbol{\Sigma}_d \mathbf{V}_d^T$, where $\mathbf{U}_d$ is a matrix of orthogonal columns.
4: Set $\mathbf{U}_d = \mathbf{U}_d[:, 1 : r_d]$, where r_d is the desired rank of the dth mode.
5: **end for**
6: Compute the core tensor $\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{X}} \times_1 \mathbf{U}_1^T \times_2 \mathbf{U}_2^T \cdots \times_D \mathbf{U}_D^T$.

Output: Core tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{r_1 \times \cdots \times r_D}$, and factor matrices $\mathbf{U}_d \in \mathbb{R}^{p_d \times r_d}$, $d = 1, \ldots, D$.

In HOOI, the Tucker decomposition is initialized using the HOSVD of the original tensor. Following the construction of the initial approximation, HOOI utilizes an iterative algorithm to refine the factor matrices and core tensor, aiming to minimize the error between the original tensor and the Tucker approximation, i.e., $\min_{\hat{\boldsymbol{\mathcal{X}}}} \|\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}}\|_F$. In each iteration, one factor matrix is updated, while the other factor matrices and core tensor are held fixed. The update is conducted by performing a matrix factorization of a specific matrix that is dependent on the prevailing values of the other factor matrices and the core tensor. This process is repeated until the error converges to a minimum. Algorithm 2 offers a comprehensive description of the HOOI algorithm.

Algorithm 2 Higher-Order Orthogonal Iteration (HOOI)

Input: A tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{p_1 \times \cdots \times p_D}$, and desired ranks $r_1, \ldots, r_D$.

1: Initialize factor matrices $\mathbf{U}_d \in \mathbb{R}^{p_d \times R}$, $d = 1, \ldots, D$, using HOSVD.
2: **while** not converged **do**
3: **for** $d = 1$ to D **do**
4: Compute $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{X}} \times_1 \mathbf{U}_1^T \times_2 \ldots \times_{d-1} \mathbf{U}_{d-1}^T \times_{d+1} \mathbf{U}_{d+1}^T \ldots \times_D \mathbf{U}_D^T$.
5: Compute the mode-d matricization $\boldsymbol{\mathcal{Y}}^{(d)}$ of $\boldsymbol{\mathcal{Y}}$.
6: Compute the SVD of $\boldsymbol{\mathcal{Y}}^{(d)}$, $\boldsymbol{\mathcal{Y}}^{(d)} = \mathbf{U}_d \boldsymbol{\Sigma}_d \mathbf{V}_d^T$, where $\mathbf{U}_d$is a matrix of orthogonal columns.
7: Set $\mathbf{U}_d = \mathbf{U}_d[:, 1 : r_d]$, where r_d is the desired rank of the dth mode.
8: **end for**
9: **end while**
10: Compute the core tensor $\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{X}} \times_1 \mathbf{U}_1^T \times_2 \mathbf{U}_2^T \cdots \times_D \mathbf{U}_D^T$.

Output: Core tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{r_1 \times \cdots \times r_D}$, and factor matrices $\mathbf{U}_d \in \mathbb{R}^{p_d \times r_d}$, $d = 1, \ldots, D$.

Chapter 3

Artificial Intelligence Methods for Imbalanced Image Classification

In this chapter, we present a privacy preserving and communication efficient information enhancement procedure for medical image classification. The proposed method tackles the challenges of imbalanced labels, data aggregation, and privacy protection through the use of transfer learning and synthetic data generated by generative models. Rather than requiring access to the raw medical images, our approach only requires a limited number of pre-trained model parameters obtained from the source data. The target and source datasets used in this chapter are described in Section 3.2, along with a discussion of the problem setup. In Section 3.3, we introduce the transfer learning technique and the generative models employed in our procedure. Our results are presented and analyzed in Section 3.4, where we conduct experiments and compare the performance of several competing models. These results are based on the findings of (X. Li and Ke, 2022).

3.1 Introduction and Related Work

Since a deep Convolutional Neural Network (CNN) architecture won the ImageNet computer vision competition in 2012 (Krizhevsky et al., 2012b), deep CNN model gain popularity in almost all computer vision-related areas, such as image classification (Lu and Weng, 2007), feature extraction (Mao and Jain, 1995), face recognition (Lawrence et al., 1997), image segmentation (J. Long et al., 2015; Ronneberger et al., 2015), image understanding (Mahendran and Vedaldi, 2015; Maninis et al., 2016) and many more. Besides the aforementioned promising applications, computer-aided diagnosis for medical images with CNN-based classification algorithms have attracted intense research interests, see (Gao et al., 2019; Giger and Suzuki, 2008; Halalli and Makandar, 2018; Krizhevsky et al., 2012a; Stoitsis et al., 2006) and references therein.

The first medical image was created in 1895. A German professor of physics, Wilhelm Rontgen, took an X-ray image of his wife's hand. Since then, medical imaging has started to play a vital role in healthcare. Medical imaging refers to numerous techniques of imaging the interior of the human body to diagnose, monitor, or treat clinical conditions. Each of these techniques sends a particular form of energy, such as electromagnetic spectrum, radiofrequency (RF) waves, and γ rays, into the human body to see how the body reacts to the energy and returns a signal to a detector, typically outside the body. As medical imaging has the ability to reveal hidden structures covered by skin and bones, these advanced techniques have been wildly used in diagnosing clinical problems and dramatically reduce the need to perform exploratory surgery, which is time-consuming and has the risk of causing mortality. Medical imaging types include X-ray, Computed Tomography (CT), Magnetic Resonance Imaging (MRI), Ultrasound Imaging, Unclear Medicine, etc. Figure 3.1 displays several examples of medical images. The type of medical imaging used in our research is chest X-ray (CXR) images, typically used to diagnose a range of conditions, including fractures, pneumonia, heart failure, pneumothorax, and lung disease.

Unlike natural images, medical images have several salient features which may cause additional challenges in their analysis processes. First, medical images are in general expensive and time-consuming to

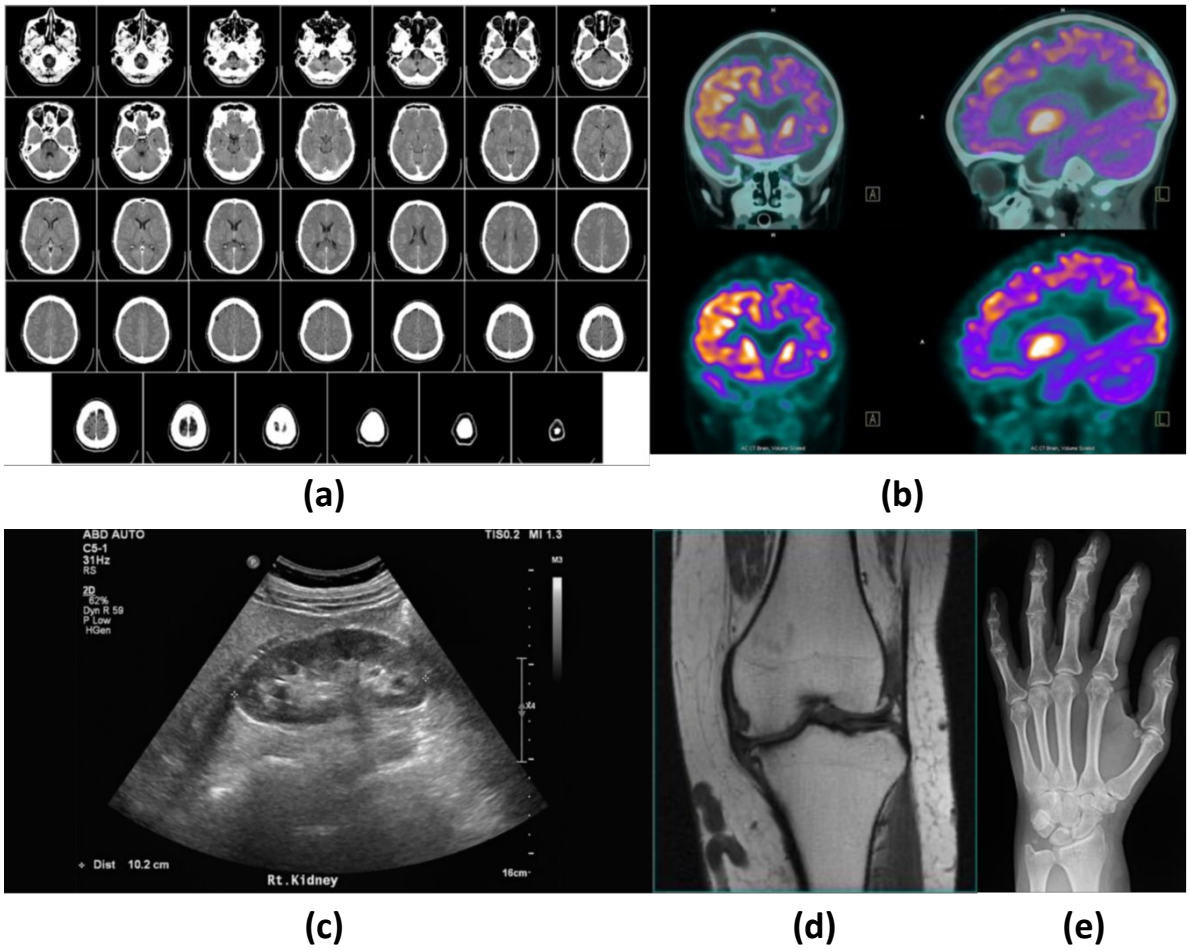


Figure 3.1: Medical imaging examples. (a) CT scan of a brain, from the base of the skull to the top; (b) Nuclear Medicine image of a brain; (c) Ultrasound of a kidney; (d) MRI of a knee; (e) X-ray of a wrist and hand.

collect due to equipment availability, materials cost, and human labor expense. Thus, the sample sizes of many medical image datasets are not as large as the datasets we usually have in other computer vision applications. Second, it is common to observe class imbalance in medical image datasets. For example, in the pediatric CXR image dataset we are going to study, the Normal class is much smaller than the Pneumonia class as healthy kids are less likely to visit doctors and take X-ray images. As a result of the first two salient features, training a deep CNN-based classification model with randomly initialized weights can be a luxury for many medical image datasets as the process requires a large and well-balanced sample

to output satisfactory results. Third, naive augmentation of medical image datasets from multiple sources may not always be a realistic option. On the one hand, the format, size, and quality of medical images vary across multiple sources due to differences in equipment models, operation norms, and labeling standards. On the other hand, hospitals and medical institutes are usually not passionate about publishing or sharing their medical image data due to policy restrictions, privacy concerns, and communication issues. Therefore, researchers of medical image classification usually have no or restricted access to additional medical image data sources.

In this chapter, we propose a privacy preserving and communication efficient information enhancement procedure for medical image classification. To reflect the salient features discussed above, we study the binary classification for a small and heavily imbalanced pediatric CXR image dataset (target dataset). We assume there exists another X-ray image dataset (source dataset) that we can borrow additional information from. However, we prohibit ourselves from directly accessing the images in the source dataset to respect the privacy preserving concerns. The first component of our procedure is transfer learning. Transfer learning aims to re-use the parameters pre-trained on a large source dataset as the initial weights for the model training on a small target dataset. It allows the model training on the target dataset to have a "warm start", and hence alleviates the data insufficient problem. Existing transfer learning studies for deep CNN models are mainly focused on natural image analysis, where the source model is usually trained on a vast natural image database, such as ImageNet (Russakovsky et al., 2015). Recently, transfer learning has been a widely investigated method to improve the performance of various medical image classification applications, such as X-ray (Lee et al., 2020), MRI (C. Zhang et al., 2019), skin lesion (Hosny et al., 2019), CT scan (Chowdhury et al., 2019), and more. However, transferring a deep CNN model pre-trained on a natural image dataset may not provide significant improvement for medical image classification problems, where pathology detection is based on local textures, like local white opaque patches in CXR images are signs of pneumonia and consolidation. In our research, we investigate the transfer learning between two medical image datasets whose domains and class labels can be different. The second component of our procedure is artificial sampling with generative models. Naive data augmentation techniques like

duplicating, rotating, flipping, zooming, and cropping of the existing images (Buda et al., 2018; Sharma et al., 2020; Stephen et al., 2019) are not suitable for medical image datasets as the generated sample are not interpretable. In recent years, generative networks, including AutoEncoder-based and Generative Adversarial Networks (GANs) based models, have shown great potential in capturing the underlying distribution of data and generating images mimicking the existing ones (Albahli et al., 2021; Qin et al., 2019). In our research, we explore several popular generative models and create artificial samples to address the class imbalance issue. Both components in our procedure do not require direct access to the medical images in the source dataset. Therefore, our procedure strictly preserves the privacy of the patient in the source dataset. Besides, our procedure is communication efficient as it only requires the transmission of model parameters rather than the raw medical images. The empirical advantage of our procedure is validated by extensive numerical experiments.

3.2 Data Description and Problem Setup

3.2.1 Target Dataset

The target dataset is a collection of frontal CXR images of pediatric patients under five years old, which are labeled by professionals in Guangzhou Women and Children's Medical Center, Guangzhou, China (Kermany et al., 2018). The dataset [1] consists of $5,910$ frontal CXR images. Among the X-ray images, $1,576$ are labeled as Normal, and $4,334$ are labeled as Pneumonia. The major research objective is to develop a computer-aided diagnosis method that can accurately and efficiently classify Normal and Pneumonia pediatric patients using their frontal CXR images.

In this study, we randomly partition the target dataset into three sub-samples for training, validation, and testing. The validation and testing sets are designed to be balanced to evaluate the classification performance, while the training set is intentionally left severely imbalanced. The partition sample sizes are summarize in Table 3.2.1.

[1] Avaliable at https://data.mendeley.com/datasets/rscbjbr9sj/2

Table 3.2.1: Target dataset splitting sample sizes.

	Training	Validation	Testing
Normal	800	334	442
Pneumonia	3500	390	444

3.2.2 Source Dataset

To alleviate the information insufficiency in the target dataset, we introduce a supplement CXR dataset [2] (X. Wang et al., 2017) released by the National Institutes of Health Clinical Center (NIHCC). The NIHCC dataset consists of $112,120$ CXR images collected from more than $30,000$ patients in all age groups, including many with one or more of 14 lung diseases, such as Infiltration, Atelectasis, Effusion, etc. In the dataset, $60,361$ images are labeled as No-Findings which means no lung disease can be diagnosed according to these images. Besides, $30,963$ images are labeled to have only one of the 14 lung diseases, while the others may be labeled by multiple lung diseases. Among the single disease classes, Infiltration, which indicates visible pulmonary infiltrates in the lung area, is the largest class with a class size of $9,547$ images.

In our research, to match the binary classification problem of interest in the target dataset, we choose a bi-class subset of the NIHCC dataset as our source dataset. To be specific, the source dataset includes all $9,547$ Infiltration images plus $9,547$ No Finding images randomly sampled from the NIHCC dataset.

3.2.3 Problem Setup

In this chapter, we design our experiments to mimic several typical challenges in medical image classification applications. First, the target dataset has a limited sample size which is insufficient to train a deep CNN model from scratch. Second, according to Table 3.2.1, the training sample in the target dataset is severely imbalanced as the ratio between Normal and Pneumonia is less than $1/4$. The label imbalance is a

[2] Avaliable at https://nihcc.app.box.com/v/ChestXray-NIHCC

salient feature for many X-ray datasets since healthy, asymptomatic, and mild patients may not necessarily take X-ray images to assist their diagnoses. As a result, the performance of deep CNN on the target dataset is not ideal. For example, the classification accuracy of a VGG-16 (Simonyan and Zisserman, 2015) model is less than 85%, where nearly one-third of Pneumonia images in the testing set are misclassified to the Normal class. Such a high false negative rate may result in serious negligence and delay in the treatment of pneumonia patients.

According to the data description, there exists a clear domain adaption issue caused by transferring the information from the source dataset to the target dataset. The images in the source dataset are taken over all age groups while the target dataset only contains images of pediatric patients under five years old. In addition, the medical meaning of the disease label in the source dataset (Infiltration) is also different from the one in the target (Pneumonia). In Figure 3.2, we list several images in each dataset, where the domain difference is visually recognizable. Therefore, we will show that directly augmenting the source dataset into the training process is not the best strategy to improve the classification accuracy in the target dataset.

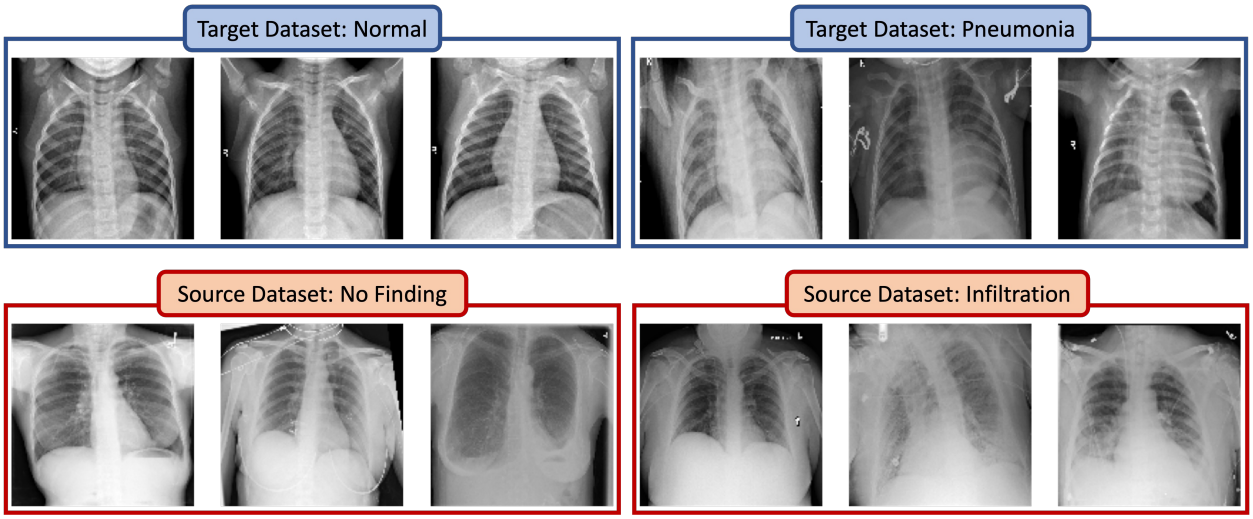


Figure 3.2: Sample observations from target and source datasets.

In addition, we consider the scenario that direct access to the images in the source dataset is prohibited, which is, unfortunately, a widely encountered obstacle in medical image studies. Hospitals and medical

institutes are usually reluctant to publish or share their medical image data due to policy restrictions and privacy concerns. Recent studies (Rocher et al., 2019; White et al., 2022) have shown that traditional anonymization methods, like removing or shuffling IDs, may not truly preserve the privacy of patients. To avoid the risk of privacy leakage and reduce the communication cost, we propose to let the owner of the source data pre-train a deep CNN model they trust, and only share the trained model parameters to the researchers of the target dataset. In such an information sharing scheme, the privacy of patients is fully preserved and the transmission of gigantic raw medical image datasets is avoided.

3.3 Methods

3.3.1 Transfer Learning

In this chapter, we first propose transfer learning as a viable solution to overcome the challenges discussed in Section 3.2.3. The intuition of transfer learning is to re-use a model pre-trained on a source dataset to a target dataset with a similar task. We can formulate a popular transfer learning scheme for image classification as follows. Let $\{\boldsymbol{X}_i^s, Y_i^s\}_{i=1}^m$ be a source image dataset, where $(\boldsymbol{X}_i^s, Y_i^s)$ is the ith image and its corresponding class label. We pre-train a deep neural network model on the source dataset. Denote $T(\cdot;\theta^s)$ the trained model parametrized by an augmented coefficients vector θ^s. Then we propose to transfer $T(\cdot;\theta^s)$ to a target dataset $\{\boldsymbol{X}_i^t, Y_i^t\}_{i=1}^n$, such that we predict Y_i^t by $T(\boldsymbol{X}_i^t;\theta^s)$. Several modifications can be made to the transferred model such as layer selection, feature map screen, or model update by treating θ^s as initial estimates. Despite many attempts have been made by statisticians (Cai and Wei, 2021; Maity et al., 2020; Tian and Feng, 2021), a unified mathematical explanation to the success of transfer learning is still lacking. Nevertheless, an interesting phenomenon that has been repeatedly discovered in many computer vision studies (Azizpour et al., 2015; M. Long et al., 2015; M. Long et al., 2016; Mou et al., 2016; Yosinski et al., 2014): the feature maps learned by the low-level layers of a CNN model are prone to represent some common visual notions like shapes, edges, geometric symmetry, effects of lighting, which are highly generalizable. On the other hand, feature maps on high-level layers are task-specific and hence

can be useful when the target task is similar to the source. We refer to Figure 3.3 for a toy example to visualize the feature maps in each layer of a CNN model. In general, the ability of lower-level layers is more general, and that of higher-level layers is more specific to the classification task. Therefore, a CNN model pre-trained on a large source dataset could help the learning for a similar target task with a small dataset through transfer learning.

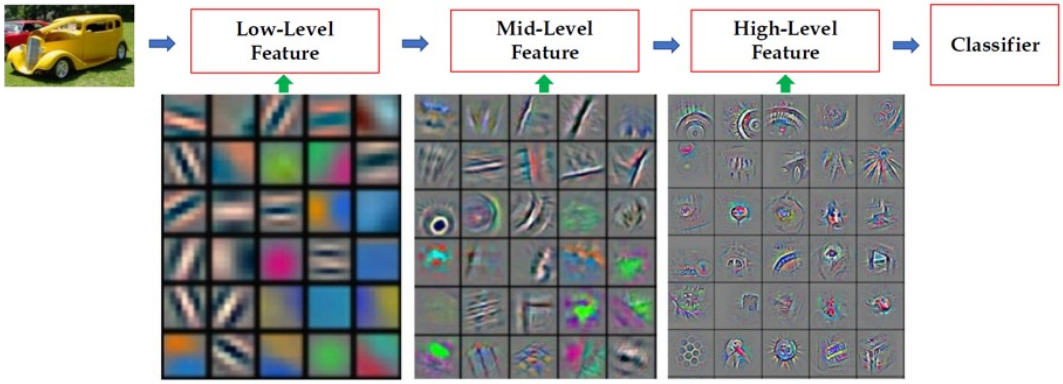


Figure 3.3: Visualization of feature maps in low, mid, and high level layers of a CNN model.

3.3.2 Generative Models and Artificial Sample

The core idea behind generative models is to capture the underlying distribution of the input images and generate high-quality artificial samples. To address the class imbalance issue, we propose to boost the minority class with artificial images generated from a distribution learned by generative models. In this project, we consider two representative types of generative models: Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) and AutoEncoders (Hinton and Salakhutdinov, 2006).

GAN

A GAN model simultaneously trains two neural networks: a generator, denoted G, and a discriminator, denoted D (or a critic, denoted C, as in some literature). Figure 3.4 demonstrates the structure of basic GANs. The two models are trained in an adversarial manner. The generator is an inverse neural network that takes a vector of random noise, denoted by z, which is sampled from the prior distribution (Gaussian

distributions are commonly adopted), denoted by p_z. Then the random noise vectors are up-sampled by the generator to artificial images, denoted $x_a = G(z)$, which belong to the same space as the real images, denoted x. Synthetic artificial examples try to mimic authentic data, denoted x, whose distribution is represented by p_{data}, and aim to fool the discriminator. In contrast, the inputs of the discriminator are a mixture of real images x and artificial images x_a. The discriminator, a feed-forward neural network, is then trained to distinguish between them, returning the probability of an input image being "true". So it gives us the following minimax optimization problem:

$$\min_G \max_D \{V(D,G)\} = \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))].$$

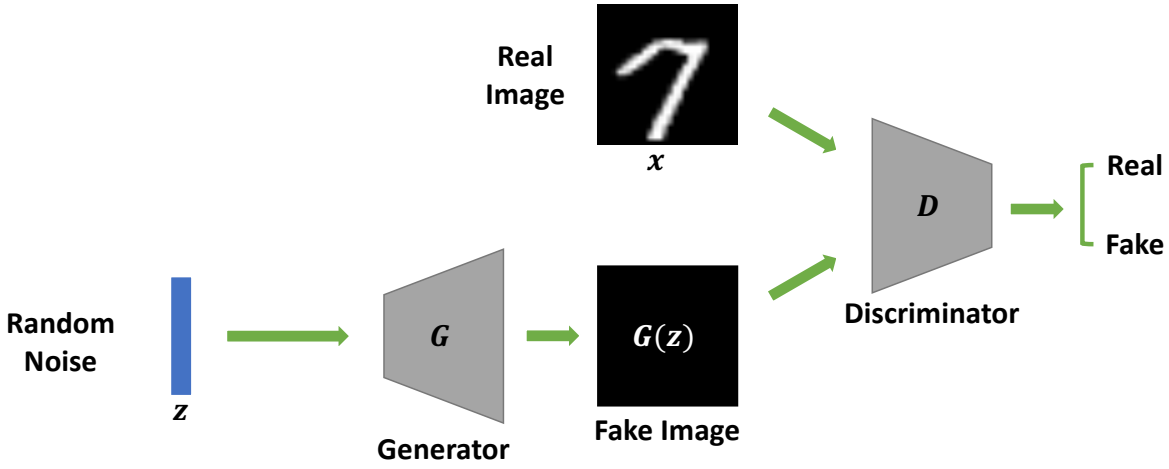


Figure 3.4: The general framework of GANs.

We train the discriminator to maximize $\log(D(x))$ the probability of assigning the correct label to both authentic and artificial images, and train the generator to minimize $\log(1 - D(G(z)))$ to deceive the discriminator into committing misclassifications. By competing with the discriminator, the generator is gradually optimized to generate artificial images that are increasingly indistinguishable from the authentic ones. In other words, proven by Arora et al., 2017, if the generator is successful, we would expect the empirical distribution of x_a, denoted by p_{fake}, to be identical to the population p_{data}. When reaching such an equilibrium, the discriminator cannot differentiate artificial examples from authentic ones. Eventually, it has to make random guesses and return probabilities of around 0.5.

The innovative idea of GAN has motivated a large number of studies. In our research, we adapt three widely used variations of GAN in our experiments, which are introduced as follows.

Deep Convolutional GAN

Compared to the original GANs that use multilayer perceptrons, Deep Convolutional GANs (DCGAN) utilize CNNs for the generator and discriminator networks and thus have the ability to generate complicated images (Radford et al., 2016). Specifically, the DCGAN architecture (1) replaces pooling layers with fractional-strided convolutions in the generator and strided convolution in the discriminator; (2) removes fully connected layers and directly connects the output to the convolutional layers; (3) applies batch normalization (Ioffe and Szegedy, 2015) to both the generator and discriminator (except the generator output layer or the discriminator input layer) to make the learning process efficient and stable; (4) utilizes ReLU (Nair and Hinton, 2010) activation in the generator (except output layers which use Tanh activation) and Leaky ReLU (Maas et al., 2013) in the discriminator to prevent gradients from vanishing.

Wasserstein GAN with Gradient Penalty

The optimization of GANs involves training two neural network models in the opposite direction and hence is well known to suffer from instability, mode collapse, and slow convergence (Arjovsky and Bottou, 2017). Wasserstein GAN (WGAN) addresses these optimization challenges by using the 1-Wasserstein distance (also known as the Earth-Mover distance) as the loss function. The 1-Wasserstein distance measures the optimal cost of transporting mass in converting the real data distribution p_{data} to the artificial data distribution p_{fake} and is defined as:

$$\mathrm{W}\left(p_{data}, p_{fake}\right) = \inf_{\gamma \sim \prod\left(p_{data}, p_{fake}\right)} \mathbb{E}_{(x,y)\sim\gamma}\|x - y\|.$$

In addition, WGAN applies weights clipping trick onto the discriminator to enforce the Lipschitz constraint. In detail, the weights of the discriminator are forced to be within a specific range controlled by a hyperparameter c. Compared to the discriminator in GAN, the one in WGAN does not use the sigmoid

function. Other than returning a probability, it outputs a scale score, representing the fidelity of the input images. Proven by Arjovsky and Bottou, 2017, the 1-Wasserstein distance outperforms Jensen-Shannon divergence and Kullback-Leibler divergence, as it provides a smoother gradient and thus helps stabilize the training of WGAN with reduced mode collapse.

However, weight clipping introduces model convergence problems when the hyperparameter c is not tuned correctly. In follow-up work, Gulrajani et al., 2017 improved the training of WGAN by replacing the clipping with a gradient penalty term to enforce the Lipschitz constraint. The new method, named WGAN_GP, admits loss functions as

$$\mathcal{L}_D^{WGAN\_GP} = -\mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log D(G(z))] + \lambda \mathbb{E}_{\hat{x}\sim p_{\hat{x}}}\left[(\|\nabla D_{\hat{x}}(\hat{x})\|_2 - 1)^2\right]$$
$$\mathcal{L}_G^{WGAN\_GP} = -\mathbb{E}_{z\sim p_z}[\log D(G(z))],$$

where $\hat{x} = \alpha x + (1-\alpha)G(z)$, and α is a random number uniformly distributed over the interval $[0, 1]$. Consequently, WGAN_GP penalizes the model if the gradient norm drifts away from 1. According to the experiments, it is not causing the problems observed in WGAN with weight clipping.

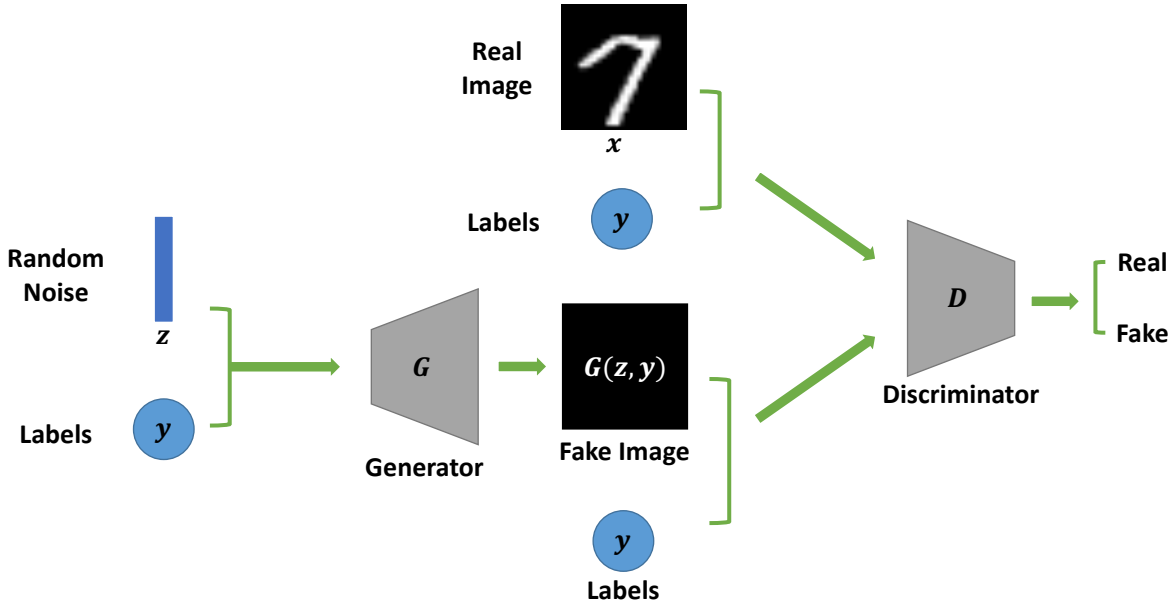


Figure 3.5: Conditional GAN (CGAN) model architecture.

Conditional GAN

Conditional GANs (CGAN) modify the training process of the classical GANs by imposing priors drawn from exogenous information (Mirza and Osindero, 2014). Such additional information denoted y can be categorical labels, textual contexts, other partial or low-resolution images, etc. Figure 3.5 shows the model architecture of CGANs. Different from the framework of GAN, the generator in CGAN takes random noise vectors z and labels y as inputs, denoted as (z, y), and synthesizes an artificial example $G(z, y) = x_a|y$. Afterward, the real examples with labels (x, y) and artificial examples with labels $(x_a|y, y)$ are fed into the discriminator, which outputs a single probability indicating whether the input example is a real, matching image-label pair. Generally, CGAN can be formulated as:

$$\mathcal{L}_D^{CGAN} = -\mathbb{E}_{(x,y)\sim p_{\text{data}}(x,y)}[\log D(x|y)] - \mathbb{E}_{z\sim p_z, y\sim p_{\text{data}}(y)}[\log(1 - D(G(z|y)))]$$
$$\mathcal{L}_G^{CGAN} = -\mathbb{E}_{z\sim p_z, y\sim p_{\text{data}}(y)}[\log D(G(z|y))].$$

While training a CGAN model, the generator and discriminator are conditioned on the exogenous information by concatenating it to the input layers. Figure 3.6 displays the procedures of combining the label with (a) input random noise vectors or (b) input random noise vectors. Specifically, when training the generator, labels and random noise vectors are embedded and reshaped into two three-way arrays whose first two dimensions are identical. Then, the concatenation of these two arrays serves as the input layer of the generative network. While for the discriminator, as the input images (real ones and artificial ones) are formulated as three-way arrays, we only embed and reshape the labels.

For GANs, although the domain of examples the model learned to emulate could be controlled by the selection of the training images, it's unable to specify any of the characteristics of the data samples the GAN would generate. For example, if we select images from multiple classes as training examples, we cannot control whether it would produce examples from a specific category. In contrast, CGAN allows the generator to synthesize images of a particular class after training the model over images from multiple classes.

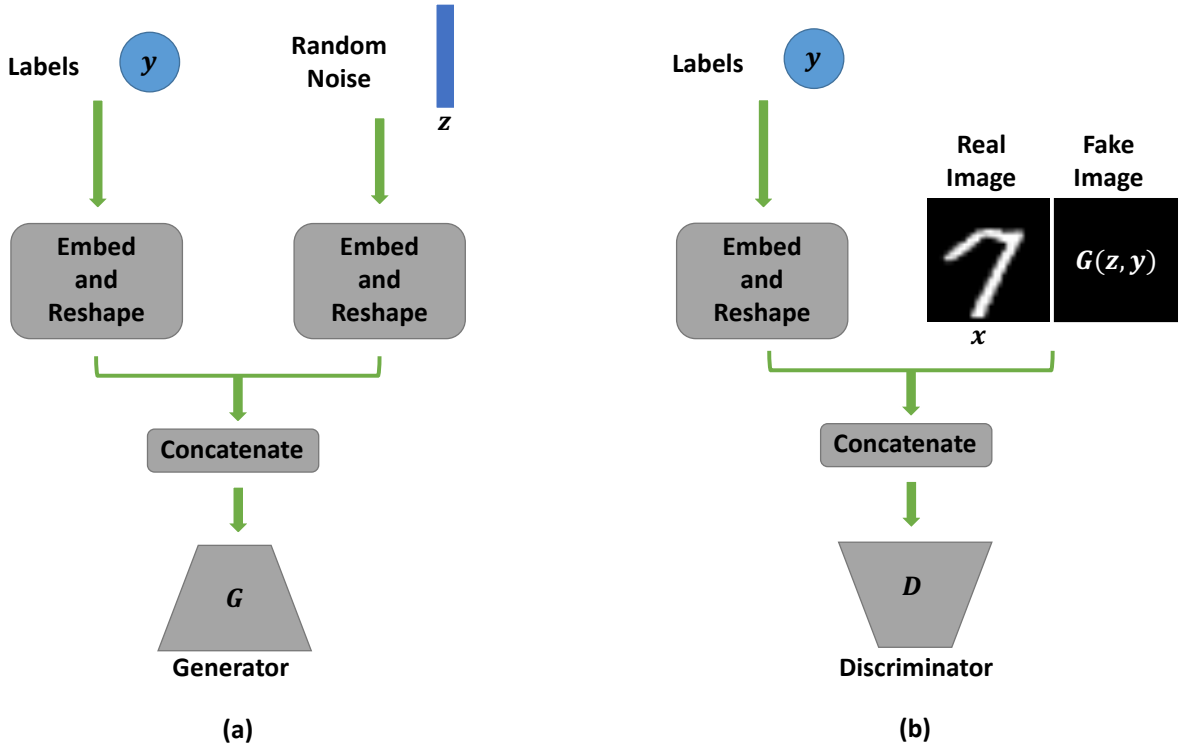


Figure 3.6: The procedure of combining labels in the CGAN model. (a) Combining labels y with random noise vectors z in the generator. (b) Combining labels y with real images x or fake images $G(z, y)$ in the discriminator.

AutoEncoder

An AutoEncoder is an unsupervised learning neural network that compresses the underlying distribution in a lower-dimensional latent space (Hinton and Salakhutdinov, 2006), whose structure is represented in Figure 3.7. It consists of two parts: the encoder, denoted by F, and the decoder, denoted by G. The encoder is structured by a sequence of layers that sequentially decrease the dimension of their outputs and aims to encode the input x image into a latent space $z = F(x)$ (sometimes called a bottleneck). In contrast, the decoder, structured with a stack of increasing dimension layers, reconstructs the compressed representation to an output image $\hat{x} = G(F(x))$ with the same dimension as the input.

The system is expected to reconstruct images the same as the corresponding input images, i.e., $\hat{x} \approx x$. There are various metrics to quantify the difference between two images. The one commonly used is the

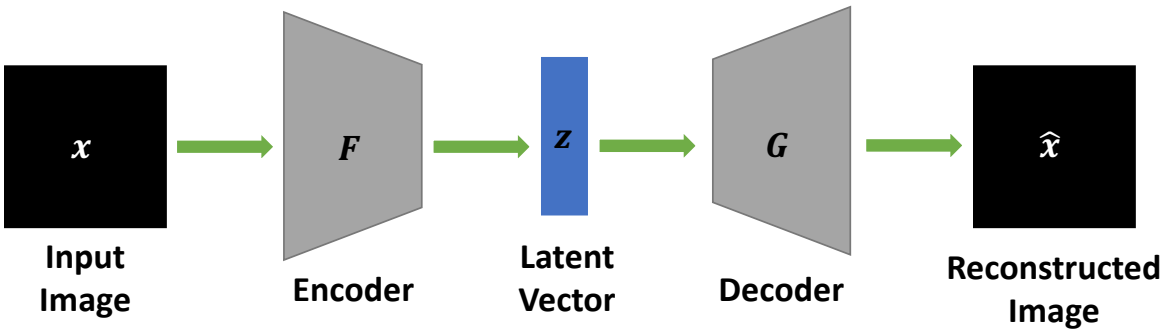


Figure 3.7: The framework of AutoEncoder.

reconstruction error, which measures the pixel-wise difference between x and $\hat{x}$, say Mean Squared Error (MSE), which is given as:

$$\mathcal{L}_{AE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2,$$

where N is the number of training examples, or the batch size used to calculate the loss. The naive AutoEncoder is tailored to reconstruct the input rather than generate a near identically distributed but independent artificial observation. However, there are many variations of AutoEncoder which can be used as generative models.

Variational Autoencoder

One of the popular AutoEncoder-based generative models is Variational Autoencoder (VAE), which enforces continuous latent spaces and allows random sampling (Kingma and Welling, 2014). Therefore, it can be a helpful technique for generative modeling. Unlike the classic AutoEncoder, VAE aims to compress an input image into a latent vector that follows a constrained multivariate latent distribution, such as multivariate normal. As shown in Figure 3.8, after the encode taking in the input image x, it outputs two latent variables: mean z_μ and variance z_σ, which are the parameters of the multivariate

normal distribution that the model learns during the training. The latent vector z is sampled from the learned distribution and is then passed to the decoder to reconstruct the image $\hat{x}$.

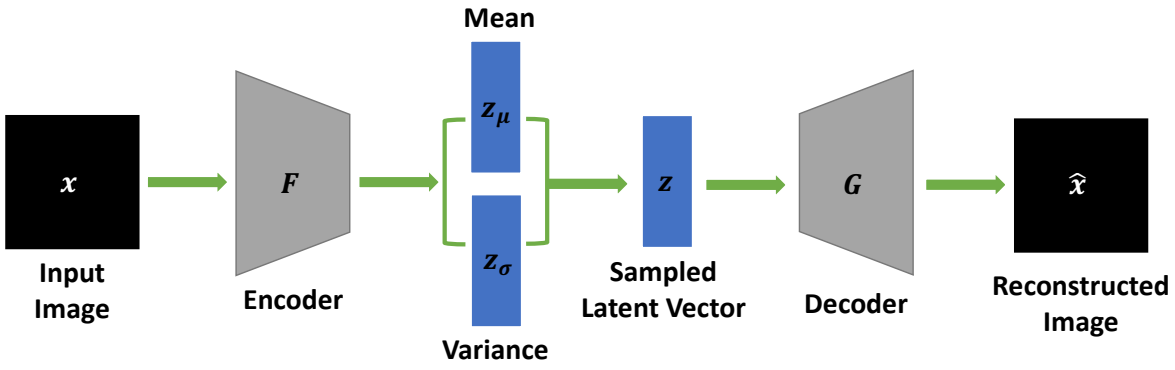


Figure 3.8: The framework of Variational AutoEncoder.

To ensure each datapoint has the encoding in the same Euclidean space region, in addition to the reconstruction loss (e.g. MSE), VAE also penalizes the KL-divergence between the real posterior $\mathcal{N}(\mu, \sigma)$ and the estimated one $\mathcal{N}(z_\mu, z_\sigma)$. The loss function is formulated as:

$$\mathcal{L}_{VAE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2 + KL[\mathcal{N}(z_\mu, z_\sigma) \,||\mathcal{N}(\mu, \sigma)].$$

Such an architecture design allows us to sample the multivariate latent encoding vector from the constrained distribution $\mathcal{N}(\mu, \sigma)$ and use a decoder to generate artificial images.

3.4 Experiments and Numerical Results

3.4.1 Classification Model

VGG16 is a CNN model proposed by Simonyan and Zisserman, 2014 and is proven to be a significant milestone in the field of computer vision. As shown in Figure 3.9 (a), it is a 16-layer network comprised of convolutional and fully connected layers, whose input layer takes in RGB images with 3 channels as input and output layer with Softmax activation returns features corresponding to the $1,000$ categories.

The 1,000 refers to the total number of possible classes in the ImageNet dataset on which VGG16 is pretrained. This dissertation borrows the idea of VGG16 and modifies the neural network structure to satisfy the specific classification requirement.

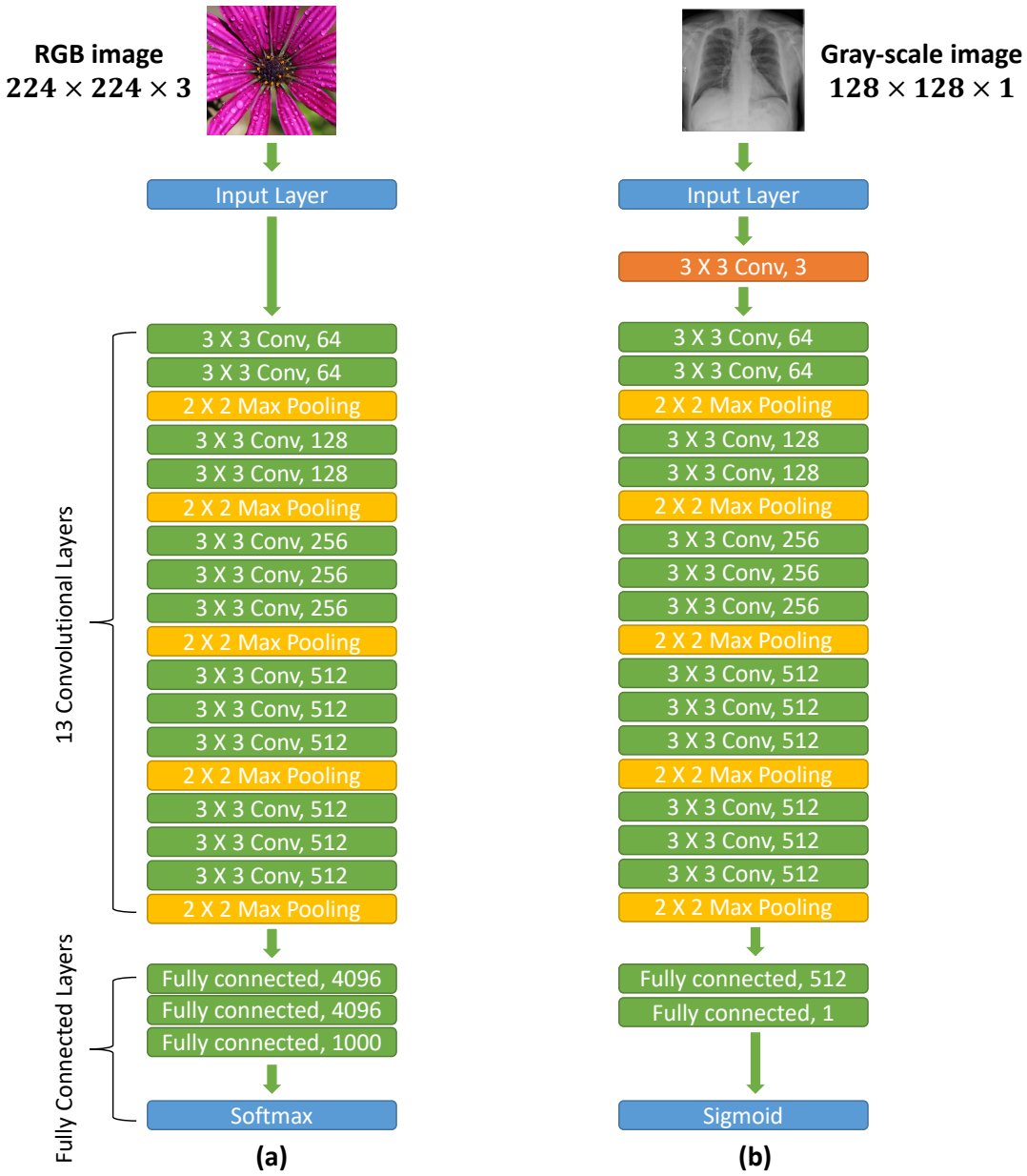


Figure 3.9: CNN model architectures of (a) VGG16 and (b) modified VGG16.

In our experiments, we unify X-ray images to a fixed size of 128×128 pixels and treat them as single-channel gray-scale images. The precise architecture of the modified VGG16 model for image classification visualized in Figure 3.9 (b) is as follows:

1. Compared to VGG16, an extra convolutional layer is added after the input layer to transform a single-channel gray-scale input into a 3-channel image compatible with the input of VGG16 architecture. It has 3 kernel filters, and the filter size is 3×3. As input gray-scale image passed into this convolutional layer, dimensions change to $128 \times 128 \times 3$.

2. The main body of the architecture uses the convolutional layers of VGG16. There is a total of five blocks. The first two blocks have two convolutional layers of filter size 3×3, stride one and padding one, while the last three blocks have three convolutional layers with the same filter size, stride, and padding. Each of these five blocks is followed by a max-pooling layer of size 2×2.

3. The VGG16's fully connected layers have been replaced to produce binary classification results. The final output is the probability of a specific CXR image having Pneumonia.

In the rest of this section, we call the above model the classification model for the simplicity of presentation. The classification models are trained with a batch size of 32 over 30 epochs using Adam optimizer (Kingma and Ba, 2017) with a learning rate of 0.00002, unless otherwise specified.

3.4.2 Training from Scratch on Imbalanced Dataset

First, we follow the sample partition scheme in Table 3.2.1, and train and test the classification model on the imbalanced target dataset from scratch. The weights of the classification model are randomly initialized. Notice that, the training sample ($4,300$ images) is clearly insufficient to empower the classification model in Figure 3.9 (b) who has nearly 19 million trainable parameters. Therefore, during the training over 30 epochs, the model tends to overfit the training set and has an unsatisfactory generalization performance. As a result, the prediction accuracy measured over the 886 testing images is 84.42%. We treat this performance as our baseline, as this baseline model does not receive any information enhancement.

The experiments were run on the Google colab Pro+ platform, which has 52 GB RAM and uses K80, T4 and P100 GPUs. It takes 389 seconds to fit the baseline model.

3.4.3 Information Enhancement by the Source Dataset

The source and target datasets are different in various aspects. The target dataset is collected and labeled by a hospital in China, which focused on pediatric patients aged under five. The source dataset is augmented by NIHCC from multiple sources, where most patients are adults. Besides, their class labels are of different meanings. To see this point, we use the baseline model trained in Section 3.4.2 to compute the probabilities of the images in the source dataset that belongs to the Pneumonia class. The histogram with smoothed density plots are presented in Figure 3.10. Recall that the baseline model is trained to separate Pneumonia labeled X-ray image from the Normal class, and the output probability represents the possibility of an input X-ray image is likely to be labeled as Pneumonia. According to Figure 3.10, both No Finding and Infiltration classes in the source dataset have histograms heavily skewed to 1, meaning that most of the images in the source data will be classified to the Pneumonia class, regardless they are labeled as No Finding or Infiltration. Although there is a clear domain difference, we show the source dataset can help improve the prediction of the target dataset by the following two approaches.

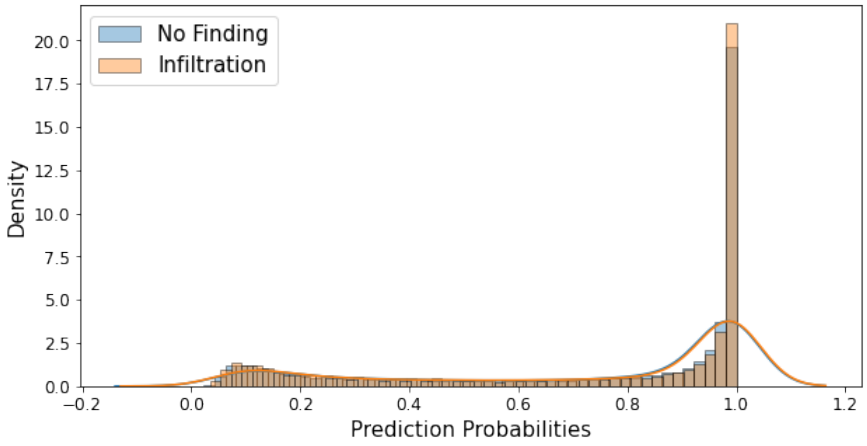


Figure 3.10: Histogram and smoothed density plot for Pneumonia probabilities of the images in the source dataset.

Augment No Finding Images to Normal Class

According to the label description in the source data, No Finding indicates that the X-ray image does not exhibit visible characteristics of lung disease. Therefore, as the target training set is imbalanced and has a gap of $2,700$ images, we randomly sample $2,700$ No Finding images from the source dataset and add them to the Normal class to balance the two classes. Then, a classification model following the architecture in Figure 3.9 (b) is trained in a similar manner as described in Section 3.4.2 over the augmented balanced target training dataset. The model performance is measured over the same 886 testing images. To account for the variability, we repeat the random sampling for seven replications. For each replication, we randomly sample $2,700$ No Finding images and add them to the Normal class in the training set On average, the new model improves the prediction accuracy from the baseline model to 90.60%.

Transfer Learning

The experiment in the above section approved that directly borrowing similar images from the source dataset can improve the prediction accuracy of the small and imbalanced target dataset. However, such a naive data augmentation approach may not always be available in practice due to the various challenges we have discussed in Section 3.2.3. In this experiment, we examine the performance of a transfer learning model where the target task has no direct access to the raw images in the source dataset. Instead, we pre-train the classification model on the source dataset as follows. The source dataset is randomly partitioned into two subsets: 90% for training and 10% for validation. A classification model with the same architecture as in Figure 3.9 (b) is trained on the source dataset to do a binary classification between No Finding and Infiltration. The training is over 100 epoch with a batch size of 64. The model with the highest prediction accuracy on the validation set is selected as the source model whose parameters are transferred to the target dataset. Then, we train the classification model on the target dataset and initialize the training with the source model parameters. After fine-tuning the weights over the imbalanced target training set following the same strategy in Section 3.4.2, the prediction accuracy on the target testing set is improved to 89.95%.

The prediction accuracy by transfer learning has an increment of 5.53% over the baseline model and is fairly close to the performance of directly adding No Finding images to the target dataset.

Compared to the naive augmentation approach, transfer learning achieves a similar prediction accuracy improvement without transmitting any images from the source dataset. Thus, patients' privacy has been fully preserved. Besides, it is well known that the process of sharing medical images between institutes can be complicated and time-consuming, and the file sizes of high-quality medical images are usually large. The transfer learning approach avoids the transmission of the raw images but only requires a transfer of model parameters which can greatly reduce the communication cost. In our experiment, the file size of the images in the source dataset is over 7.7GB. In contrast, the parameters of the pre-trained model can be stored and transferred by a single file of size around 200MB. To sum up, in our experiments, transfer learning is a privacy preserving and communication efficient approach to improve the classification accuracy in a small and imbalanced X-ray image dataset. The performance of transfer learning is comparable to the naive augmentation approach, which directly adds the images in the source dataset to the target task.

3.4.4 Information Enhancement by Artificial Sample

Next, we consider an alternative information enhancement strategy to address the label imbalance issue in the target dataset that does not require any access to the source dataset. Classical image augmentation techniques that randomly rotate, zoom, or flip the training images are not suitable for medical image analysis as the generated images are often hard to interpret and not independent of the training sample. Instead, we propose to supplement the under-sampled Normal class with artificial images created by generative models. In this experiment, we train DCGAN, WGAN_GP, and VAE over 800 Normal images in the training set. CGAN is trained over $4,300$ Normal and Pneumonia images in the training as it allows multi-class data.

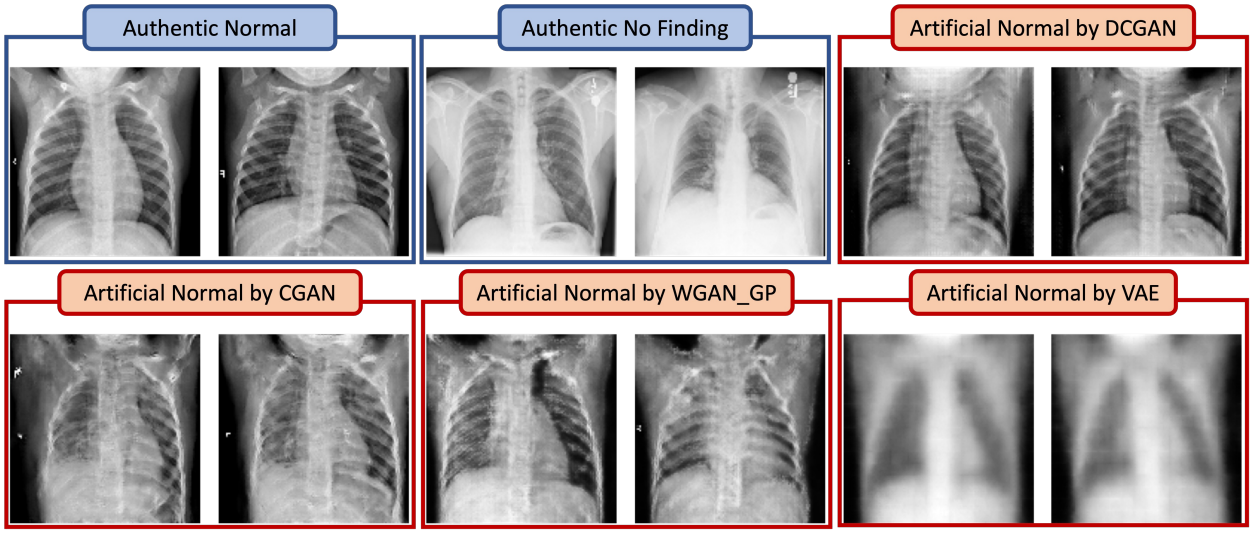


Figure 3.11: Authentic images from Normal and No Finding classes, and artificial images generated by different generative models.

DCGAN, WGAN_GP, and CGAN

The three GAN based generative models share similar generator and discriminator architectures. A six-layer CNN generator takes a random noise vector of dimension 100 as input (except for CGAN, which also uses image labels as input), and outputs a single-channel image of size 128×128 pixels. The discriminator (critic for WGAN_GP), a five-layer CNN, takes artificial images generated by the generator and authentic Normal images as inputs, aiming to distinguish artificial images from true ones. The models' weights are randomly initialized and are trained with a batch size of 64 using Adam optimizer. Finally, the generator trained at the 300th epoch is saved to generate artificial images.

VAE

The encoder uses the same architecture as the discriminator for GAN based models, except for the output layer, which outputs two vectors of size 100, representing the means and variances. The decoder has the same architecture as the generator for GAN based models. Also, the VAE model is trained with the same

batch size and optimizer. Finally, we save the decoder trained at the 300th epoch to generate artificial images.

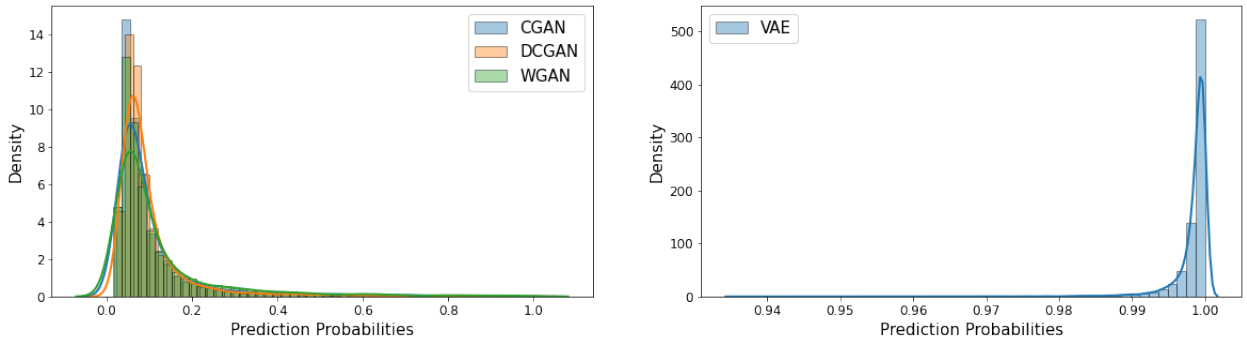


Figure 3.12: Prediction results of the baseline model on the sets of artificial images generated by different generative models.

Results

Figure 3.11 displays some artificial Normal images generated by the four generative models. Among the four methods, the artificial images generated by DCGAN are visually most clear and closest to the authentic Normal images. The images generated by VAE are blurred and not visually informative. We also evaluate the quality of artificial normal images by computing their prediction probabilities to be labeled as Pneumonia. Ideally, we expect a good artificial sample of the Normal class should have prediction probabilities concentrated around 0. Figure 3.12 visualizes these prediction probabilities for the artificial Normal samples generated by the four generative models. As we can see, three GAN based models work as expected as the empirical distributions of their artificial samples perform similarly to the true Normal images in the classification model. Further, the artificial sample generated by DCGAN has the smallest standard deviation. In contrast, the prediction probabilities of the artificial Normal images generated by VAE are concentrated around 1 which is not ideal.

According to the analysis above, we propose to use DCGAN to fill the sample size gap in two classes in the training set. In other words, we use DCGAN to generate 2, 700 artificial images for the Normal class, such that the training sample is equally contributed by two classes. Similarly, the training process has

been repeated seven times. Evaluating the models over the target testing set, we have the lowest prediction accuracy of 90.07%, and the highest one is 97.18%. The average prediction accuracy is 94.49%.

3.4.5 Privacy Preserving and Communication Efficient Information Enhancement Procedure

In previous experiments, we have separately demonstrated that transfer learning and artificial sample can improve CXR image classification, preserve patients' privacy, and reduce or avoid communication costs. In this experiment, we combine these two ideas to form our privacy preserving and communication efficient information enhancement procedure, which can be illustrated by the following 3 steps: (1) Augment the minority class with $2,700$ artificial images generated by DCGAN, so that the two classes are balanced. (2) Initialize the weights of the classification model by the parameters transferred from the model pre-trained on the source dataset. (3) Fine-tune all layers over the balanced target training set. We repeat this procedure 7 times to address the variability in random sampling. The prediction accuracy over the target testing set ranges from 93.34% to 97.63% with an average 95.95%, which improves the baseline model by 11.53%.

3.4.6 Comparison and Analysis of Experiment Results

Table 3.4.1 summarises and compares the prediction results of the five classification models over the 886 testing images. The five models, as introduced in Sections 3.4.2– 3.4.5, apply different transfer learning and data augmentation strategies.

The first row in Table 3.4.1 represents the baseline model which uses neither transfer learning nor data augmentation. The baseline model has a classification accuracy of 84.42%. Besides, baseline commits one false-positive (misclassify Normal to Pneumonia) and 137 false-negative (misclassify Pneumonia to Normal). The performance of the baseline model is not ideal as nearly one-third of Pneumonia pictures in the testing set are mistakenly diagnosed as Normal. Ignoring pneumonia patients as healthy people may delay their treatments and cause severe consequences in practice. The best model among the 5 utilizes

Table 3.4.1: Summary of 5 models. There are three data augmentation choices: no data augmentation (No); add No Finding to Normal (No Finding), and artificial sample (DCGAN). The Transfer Learning column indicates if we use transfer learning. AUC stands for the area covered under a receiver operating characteristic curve. FP and FN are average number of false positives and false negatives over 7 replications.

Data Augmentation	Transfer Learning	Accuracy	AUC	FP	FN
No	No	0.8442	0.9829	1	137
No Finding	No	0.9060	0.9859	1	82.29
No	Yes	0.8995	0.9857	4	85
DCGAN	No	0.9449	0.9927	3	45.86
DCGAN	Yes	**0.9595**	**0.9935**	4	31.86

both transfer learning and data augmentation with an artificial sample generated by DCGAN. It achieves a classification accuracy of 95.95% and reduces the average false-negative cases to 31.86.

Further, we plot Receiver Operating Characteristic (ROC) curves for the five models in Figure 3.13. The areas under the curve (AUC) values are reported in Table 3.4.1. As we can see, the ROC curve of the baseline model is at the bottom, and hence the baseline model has the smallest AUC value. On the other hand, the ROC curve of the model that utilizes both transfer learning and an artificial sample is at the top, and hence it has the largest AUC value. The model that only utilizes transfer learning and the model that did data augmented by adding No Finding images to the training set have a similar performance. The model that did data augmented with an artificial sample generated by DCGAN ranks second among the five.

3.5 Conclusion

In this chapter, we study computer-aided diagnosis for a small and imbalanced pediatric CXR image dataset. Training a deep CNN based classification model on this dataset from scratch does not work well as a lot of pneumonia kids can be misdiagnosed as normal. To improve the classification performance, we propose a privacy preserving and communication efficient procedure to enhance the information in model

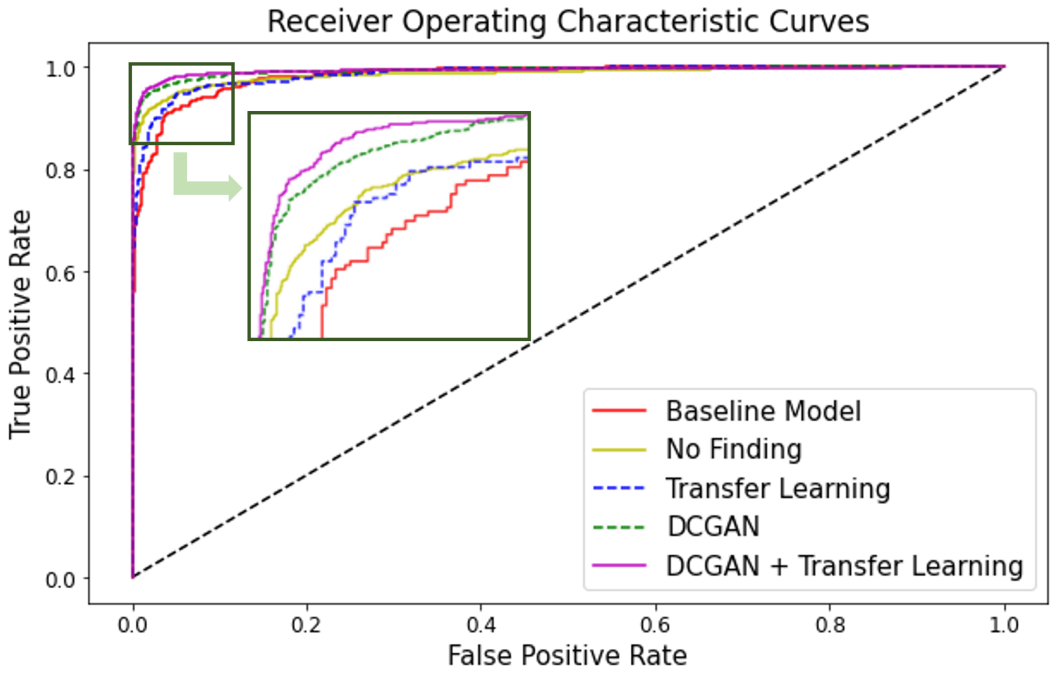


Figure 3.13: ROC curves of five classification models over the target testing images.

training on the target dataset. Additional information comes from a source dataset that contains X-ray images of patients over all age groups and labeled by 14 lung diseases. So, there is a clear domain adaption and target shift between the source and target datasets. Further, we prohibit ourselves from directly accessing the images in the source dataset to reflect the privacy concerns widely arising in medical data sharing. Our procedure has two key components: transfer learning and artificial sampling by generative models. The former one transfers the parameters of a deep CNN based model pre-trained on the source dataset to the target dataset as the initial weights. Transfer learning gives the training on the target dataset a warm start and improves the prediction accuracy by around 6%. The second component, artificial sampling, aims to supplement the minority class with artificial images generated from a distribution close to the minority population. The minority population is learned by several GAN based methods. The best generative model alone can increase another 6% to the prediction accuracy. Our procedure effectively combines these two components and can improve the prediction accuracy by more than 10%. Our

procedure strictly preserves the privacy of the patients as no medical images in the source dataset will be released to the target dataset user. Our procedure is also communication efficient as it only requires the transmission of model parameters instead of the raw medical images. We expect our procedure to have broad impacts on a wide range of medical image classification problems where the researchers may suffer from insufficient sample size, imbalanced classes, and restricted access to additional data sources.

Chapter 4

Augmented Tensor Factor Model for Overlapping Image Classification

This chapter presents a solution to the overlapping issue in image classification through the use of an augmented tensor factor model. The model combines samples from multiple classes and decomposes the tensor data into two components: a pervasive component that is common across classes and a specific component that includes the signals used to distinguish classes. The proposed model and its estimation procedure, as well as the theoretical results of the estimators, are presented in Section 4.2. The factor adjustment procedure for overlapping image classification is described in Section 4.3. The performance of the proposed image classification method is evaluated through synthetic examples in Section 4.4 and real COVID-19 chest X-ray images in Section 4.5.

4.1 Introduction and Related Work

With the advancement of modern data science, data structured as tensors, or multidimensional arrays, are becoming increasingly prevalent in diverse fields. For example, samples of images or video clips are

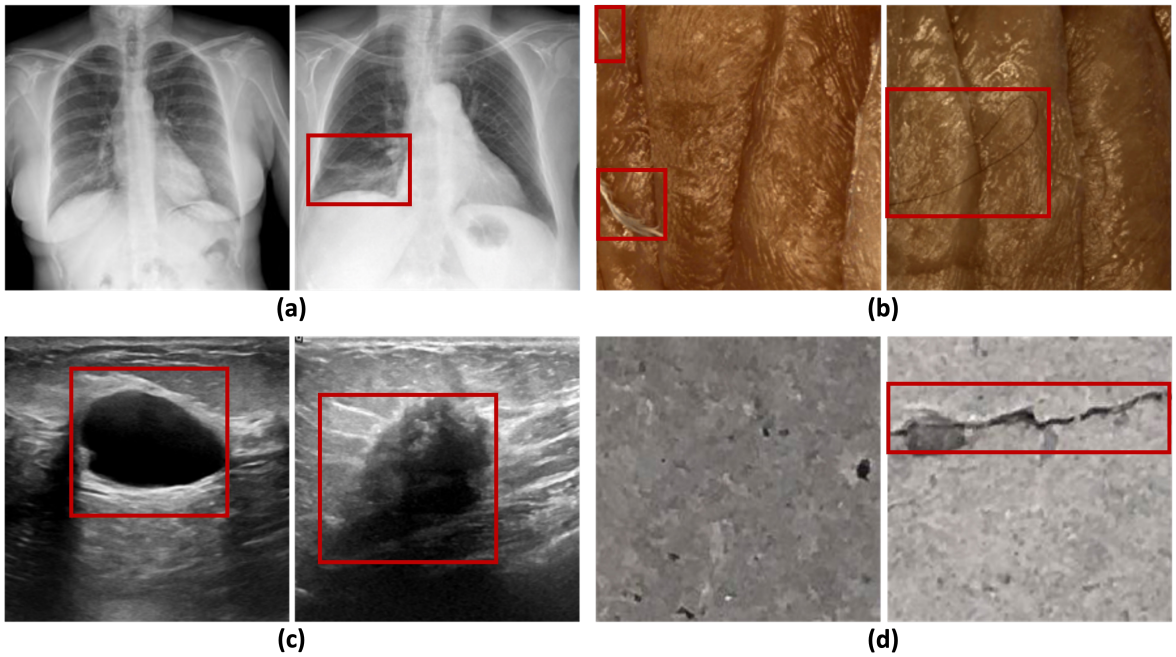


Figure 4.1: Examples of overlapping issues in image classification. The red boxes indicate the key areas used to distinguish positive and negative classes. (a) COVID-19 diagnosis via CXR: normal (left) vs. pneumonia (right); (b) Food contamination detection: feather (left) vs. hair (right); (c) Breast cancer detection via ultrasound: benign (left) vs. malignant (right); (d) Surface crack detection: no crack (left) vs. crack (right).

naturally three or more dimensional tensors. Electrical health records (EHRs), as another example, collect multiple features of many patients over a period of time and can be treated as tensor time series (C. Zhang et al., 2021). Other examples of tensor data can be found in sensor networks (J. He et al., 2016), magnetic resonance imaging (Hasan et al., 2011), spatial-temporal analysis (Ran et al., 2016), climate research (Q. Zhang et al., 2009), microbiome studies (Mor et al., 2022), and quantitative finance (Huang et al., 2018). As tensor data is typically high-dimensional and large-scale, learning its information through parsimonious yet flexible models is desirable. To that end, classical latent factor models (e.g. Bai, 2003; Bai and Li, 2012; Fan et al., 2008; Fan et al., 2011; Lam and Yao, 2012) have been revitalized by the statistics and machine learning communities to understand their computational, statistical, and theoretical properties for tensor data applications (e.g. E. Y. Chen and Fan, 2021; E. Y. Chen et al., 2019; R. Chen et al., 2022; Y. He et al., 2022; D. Wang et al., 2019; A. Zhang and Han, 2019).

Computer-aided diagnosis for medical images has attracted significant attention in the computer vision domain over the past decade. During the COVID-19 pandemic, deep neural network-based image classification methods have been widely used to detect pneumonia caused by the virus from chest X-ray (CXR) images (e.g. Ismael and Şengür, 2021; Jain et al., 2021; H. Li et al., 2022). Early detection of lung infection has been proven critical for COVID-19 patients with a high risk of developing severe symptoms, as timely treatment can reduce their mortality rates (Goyal et al., 2020; Sun et al., 2020). A well-observed challenge in medical image classification is that the images from positive and negative classes can be highly overlapping, in the sense that only a small area of the image contains the specific information to distinguish the two classes. In comparison, the rest of the image contains pervasive noise between the two classes, such as background, body shape, skeleton, and tissues. In Figure 4.1, we provide several examples of this overlapping phenomenon in image classification applications. As a result, even cutting-edge deep learning-based classifiers may suffer from limited correct classification rates since the weak specific signals are obscured by large pervasive noise.

In this chapter, we introduce a novel augmented tensor factor model to address the aforementioned challenge in image classification, which may provide insight into more general heterogeneous tensor data analysis problems. Our model combines samples from multiple classes and decomposes the tensor data into a pervasive component that shares the same pattern across classes and a specific component that varies from class to class. The pervasive component is modeled by the production of a low-rank latent tensor factor and a few factor loading matrices. We propose to matricize the augmented tensor factor model into a series of matrix variate factor models. Then, factor loading matrices and latent tensor factors are estimated using principal component analysis (PCA), which leverages the wisdom of classical latent factor analysis. The ranks of latent tensor factors are estimated using a modified eigen-ratio method. Additionally, we prove the theoretical properties of our estimators. Our estimators for factor loading matrices and latent tensor factors benefit from large sample sizes and high dimensionality, leading to fast convergence rates. The proposed rank estimation method can also consistently recover the ranks of latent tensor factors. Thanks to these desirable and novel theoretical results, we developed a factor adjustment

procedure for overlapping image classification. We use the training sample to estimate ranks and factor loading matrices; then, we regress the testing set on the estimated factor loading matrices. Finally, we apply the classifier to the regression residuals, which are consistent estimates of specific components. The factor adjustment procedure improves the signal-to-noise ratio by removing pervasive noise. The effectiveness of this procedure is demonstrated through synthetic experiments. We also show that our method improves the correct COVID-19 pneumonia diagnostic rate from CXR images by 10.5%.

4.2 Augmented Tensor Factor Model

4.2.1 Model Setup

Suppose we observe two D-way tensor valued random samples $\{\mathcal{X}_i^{(+)} \in \mathbb{R}^{p_1 \times \cdots \times p_D}\}_{i=1}^{n_1}$ and $\{\mathcal{X}_i^{(-)} \in \mathbb{R}^{p_1 \times \cdots \times p_D}\}_{i=1}^{n_2}$ from positive and negative classes, respectively. We assume the two samples share a common pervasive component $\mathcal{P}$ but different specific components $\mathcal{S}^{(+)}$ and $\mathcal{S}^{(-)}$. To be specific,

$$\mathcal{X}_i^{(+)} = \mathcal{P}_i + \mathcal{S}_i^{(+)}, \quad \text{for } i = 1, \ldots, n_1,$$
$$\mathcal{X}_i^{(-)} = \mathcal{P}_i + \mathcal{S}_i^{(-)}, \quad \text{for } i = 1, \ldots, n_2.$$

As the pervasive component does not contain useful information to distinguish the positive and negative classes, we aim to adjust it and implement the binary classification on the specific components instead of the original observations.

Let $\{\mathcal{X}_i \in \mathbb{R}^{p_1 \times \cdots \times p_D}\}_{i=1}^{n} = \{\mathcal{X}_i^{(+)}\}_{i=1}^{n_1} \cup \{\mathcal{X}_i^{(-)}\}_{i=1}^{n_2}$ be the augmented sample with $n = n_1 + n_2$, $\mathcal{X}_i = \mathcal{X}_i^{(+)}$ if $i \leq n_1$ and $\mathcal{X}_i = \mathcal{X}_{i-n_1}^{(-)}$ if $i > n_1$. Without loss of generality, we can decompose $\mathcal{X}_i$ into two parts as

$$\mathcal{X}_i = \mathcal{P}_i + \mathcal{S}_i, \quad \text{for } i = 1, \ldots, n, \tag{4.1}$$

where $\mathcal{P}_i$ is a pervasive component and $\mathcal{S}_i$ is the augmented specific component that satisfies $\mathcal{S}_i = \mathcal{S}_i^{(+)}$ if $i \leq n_1$ and $\mathcal{S}_i = \mathcal{S}_{i-n_1}^{(-)}$ if $i > n_1$. Further, we assume $\mathcal{P}_i$ admits a Tucker tensor decomposition (Tucker, 1966) as

$$\mathcal{P}_i = \mathcal{G}_i \times_1 \boldsymbol{U}_1 \times_2 \cdots \times_D \boldsymbol{U}_D, \quad \text{for } i = 1, \ldots, n, \tag{4.2}$$

where $\mathcal{G}_i \in \mathbb{R}^{r_1 \times \cdots \times r_D}$ is a latent tensor factor with $r_d \ll p_d$ for $d = 1 \ldots, D$, and $\boldsymbol{U}_d \in \mathbb{R}^{p_d \times r_d}$ is a factor loading matrix for the d-th mode of $\mathcal{X}_i$.

The tensor factor model (4.2) is flexible in the sense that both the latent tensor factor $\mathcal{G}_i$ and factor loading matrices $\{\boldsymbol{U}_1, \ldots, \boldsymbol{U}_D\}$ are unobservable. Thus, like classical factor models, model (4.2) has identifiability issues. To be specific, let $\{\boldsymbol{H}_d \in \mathbb{R}^{r_d \times r_d}\}_{d=1}^D$ be a set of orthogonal matrices. Then, model (4.2) also holds for the latent tensor factor $\mathcal{G}_i \times_1 \boldsymbol{H}_1^{-1} \times_2 \cdots \times_D \boldsymbol{H}_D^{-1}$ and factor loading matrices $\{\boldsymbol{U}_1\boldsymbol{H}_1, \ldots, \boldsymbol{U}_D\boldsymbol{H}_D\}$. To facilitate the discussions in estimation and theoretical analysis, we impose the identification conditions as $p_d^{-1}\boldsymbol{U}_d^{\mathrm{T}}\boldsymbol{U}_d = \boldsymbol{I}_{r_d}$ for $d = 1, \ldots, D$. Please note that the identification conditions will not create any problems for our estimation method as we are mainly interested in estimating the linear space spanned by the columns of $\boldsymbol{U}_d$, which is the same as the one spanned by $\boldsymbol{U}_d\boldsymbol{H}_d$. Additionally, the proposed augmented tensor factor model can easily be generalized to multiple samples. To keep our presentation focused, we will not discuss this further in this chapter.

4.2.2 Matricization and PCA Estimation

Motivated by the PCA estimation of the classical factor models, we introduce a PCA type tensor factor model estimation method based on tensor matricization. In this subsection, we assume the ranks $r_1, \ldots, r_D$ are known for the ease of presentation. The estimation of ranks will be discussed in Section 4.2.3. Recall that, we denote the mode-d matricization of $\mathcal{X}_i$ as $\mathcal{X}_i^{(d)} \in \mathbb{R}^{p_d \times \prod_{m \neq d} p_m}$. Then, we can rewrite (4.1) and (4.2) as D matrix variate factor models (e.g. E. Y. Chen and Fan, 2021; E. Y. Chen et al., 2019; D. Wang et al., 2019), i.e.

$$
\begin{cases}
\mathcal{X}_i^{(1)} & = \boldsymbol{U}_1 \mathcal{G}_i^{(1)} \boldsymbol{U}_{-1}^{\mathrm{T}} + \mathcal{S}_i^{(1)}, \\
& \vdots \\
\mathcal{X}_i^{(d)} & = \boldsymbol{U}_d \mathcal{G}_i^{(d)} \boldsymbol{U}_{-d}^{\mathrm{T}} + \mathcal{S}_i^{(d)}, \quad \text{for } i = 1, \ldots, , n, \\
& \vdots \\
\mathcal{X}_i^{(D)} & = \boldsymbol{U}_D \mathcal{G}_i^{(D)} \boldsymbol{U}_{-D}^{\mathrm{T}} + \mathcal{S}_i^{(D)},
\end{cases} \tag{4.3}
$$

where $\mathcal{G}_i^{(d)}$ and $\mathcal{S}_i^{(d)}$ are mode-d matricizations of $\mathcal{G}_i$ and $\mathcal{S}_i$, and $\boldsymbol{U}_{-d} = \otimes_{m \neq d} \boldsymbol{U}_m$.

To estimate the factor loading matrix $\boldsymbol{U}_d$, we first compute the model-d sample variance covariance matrix $\boldsymbol{\Sigma}_d \in \mathbb{R}^{p_d \times p_d}$ as

$$
\boldsymbol{\Sigma}_d = \frac{1}{n p_\pi} \sum_{i=1}^{n} \mathcal{X}_i^{(d)} \mathcal{X}_i^{(d)\mathrm{T}}, \quad \text{for } d = 1, \ldots, D,
$$

where $p_\pi = \prod_{d=1}^{D} p_d$ is the product of all D dimensions.

Next, we apply an eigen-decomposition to each $\boldsymbol{\Sigma}_d$. Let $\lambda_1^{(d)} \geq \cdots \geq \lambda_{p_d}^{(d)}$ be the eigenvalues of $\boldsymbol{\Sigma}_d$ sorted in the descending order and $\boldsymbol{v}_1^{(d)}, \ldots, \boldsymbol{v}_{p_d}^{(d)}$ be the corresponding eigenvectors. For a given rank r_d, we propose to estimate $\boldsymbol{U}_d$ by

$$
\widehat{\boldsymbol{U}}_d := \widehat{\boldsymbol{U}}_d(r_d) = \sqrt{p_d} \left(\boldsymbol{v}_1^{(d)}, \ldots, \boldsymbol{v}_{r_d}^{(d)} \right), \quad \text{for } d = 1, \ldots, D.
$$

By collecting all estimates of factor lading matrices and utilizing identification conditions, we estimate the tensor latent factors by

$$
\widehat{\mathcal{G}}_i = \frac{1}{p_\pi} \mathcal{X}_i \times_1 \widehat{\boldsymbol{U}}_1^{\mathrm{T}} \times_2 \cdots \times_D \widehat{\boldsymbol{U}}_D^{\mathrm{T}}, \quad \text{for } i = 1, \ldots, n.
$$

Then, naturally, we estimate the pervasive and specific components by

$$\widehat{\mathcal{P}}_i = \widehat{\mathcal{G}}_i \times_1 \widehat{\boldsymbol{U}}_1 \times_2 \cdots \times_D \widehat{\boldsymbol{U}}_D \tag{4.4}$$
$$\text{and} \quad \widehat{\mathcal{S}}_i = \mathcal{X}_i - \widehat{\mathcal{P}}_i, \quad \text{for } i = 1, \ldots, n.$$

4.2.3 Estimation of Ranks

Estimating the ranks of tensor factor models is a challenging problem due to the unsupervised nature of the task. The matricization decomposes the mode-D tensor factor model into D matrix variate factor models as in (4.3). This motivates us to borrow the wisdom from classical factor model inference literature (e.g. Ahn and Horenstein, 2013; Bai and Ng, 2002; Chamberlain and Rothschild, 1982; Chang et al., 2015; Lam and Yao, 2012; Stock and Watson, 2002).

For every mode d, we use the modified eigen-ratio method (Chang et al., 2015) to estimte the fixed but unknown rank r_d. Let $\lambda_k(\boldsymbol{\Sigma}_d)$ the k-th largest eigenvalue of the model-d sample variance-covariance matrix, $K_{\max}$ be a prescribed upper bound, and C be a small positive constant. We propose to estimate r_d by

$$\widehat{r}_d = \underset{1 \le k \le K_{\max}}{\operatorname{argmin}} \frac{\lambda_k(\boldsymbol{\Sigma}_d) + C}{\lambda_{k+1}(\boldsymbol{\Sigma}_d) + C}, \quad \text{for } d = 1, \ldots, D. \tag{4.5}$$

The threshold $K_{\max}$ controls the computational cost. It represents the belief of the largest possible value of r_d. To avoid random spikes when $\lambda_{k+1}(\boldsymbol{\Sigma}_d)$ is close to 0, a small positive constant C is used.

4.2.4 Theoretical Results

In this subsection, we investigate the theoretical properties of the proposed estimators. All assumptions, technical lemmas, and proofs can be found in the appendices.

Theorem 4.1 (Convergence of factor loading matrices)**.** *Suppose Assumptions 1 - 3 in Appendix A hold. We assume the ranks* $\{r_d\}_{d=1}^D$ *are fixed, but allow sample size* n *and dimensions* $\{p_d\}_{d=1}^D$ *to diverge. Then,*

we have

$$\|\widehat{\boldsymbol{U}}_d - \boldsymbol{U}_d \boldsymbol{H}_d\|_F^2 = O_p(\frac{1}{p_d} + \frac{p_d^2}{np_\pi}), \quad \textit{for } d = 1, \ldots, D,$$

where $\{\boldsymbol{H}_d \in \mathbb{R}^{r_d \times r_d}\}_{d=1}^D$ *is a set of orthogonal matrices.*

Theorem 4.1 shows our model-d factor loading matrix estimator can consistently estimate the truth up to an r_d by r_d orthogonal matrix. Further, detailed analysis in the proof of Theorem 4.1 shows that $\boldsymbol{H}_d = \frac{1}{np_d} \sum_{i=1}^n \mathcal{G}_i^{(d)} \mathcal{G}_i^{(d)\mathrm{T}} \boldsymbol{U}_d^{\mathrm{T}} \widehat{\boldsymbol{U}}_d \boldsymbol{\Lambda}_d^{-1}$ with $\boldsymbol{\Lambda}_d = (\lambda_1 \ \ldots, \ \lambda_{r_d})$ being a diagonal matrix.

Theorem 4.2 (Convergence of latent tensor factors)**.** *Suppose Assumptions 1 - 3 in Appendix A hold. We assume the ranks* $\{r_d\}_{d=1}^D$ *are fixed, but allow sample size* n *and dimensions* $\{p_d\}_{d=1}^D$ *to diverge. With the same* $\{\boldsymbol{H}_d\}_{d=1}^D$ *as in Theorem 4.1, we have, for* $i = 1, \ldots, n$*,*

$$\|\widehat{\mathcal{G}}_i - \mathcal{G}_i \times_1 \boldsymbol{H}_1^{-1} \times_2 \cdots \times_D \boldsymbol{H}_D^{-1}\|_F^2 = O_p\left(\frac{1}{n} + \sum_{d=1}^D \frac{1}{p_d}\right).$$

Theorem 4.2 provides an estimation error upper bound for the latent tensor factors. The convergence rate depends on both sample size, n, and dimension $\{p_1, \ldots, p_D\}$. This "blessing of dimensionality" phenomenon is in line with classical factor model analysis, as noted in references such as (Bai, 2003) and (Fan et al., 2008). Theorems 4.1 and 4.2 allow for consistent estimation of both the pervasive and specific components. When $\sum_{i=1}^D p_d^2 < p_\pi$, some simple algebra shows that, for $i = 1, \ldots, n$, we have

$$\left\|\widehat{\mathcal{P}}_i - \mathcal{P}_i\right\|_F^2 = O_p\left(\frac{1}{n} + \sum_{d=1}^D \frac{1}{p_d}\right)$$

$$\text{and} \quad \left\|\widehat{\mathcal{S}}_i - \mathcal{S}_i\right\|_F^2 = O_p\left(\frac{1}{n} + \sum_{d=1}^D \frac{1}{p_d}\right).$$

Theorem 4.3 (Convergence of ranks)**.** *Suppose Assumptions 1 - 3 in Appendix A hold. We assume the true ranks* $\{r_d\}_{d=1}^D$ *are fixed, but allow sample size* n *and dimensions* $\{p_d\}_{d=1}^D$ *to diverge. Then, we have*

$$P(\widehat{r}_d = r_d) \to 1, \quad \textit{for } d = 1, \ldots, D.$$

Theorem 4.3 demonstrates that the modified eigen-ratio method can correctly estimate all ranks with high probability. Theorem 4.3 offers a theoretical guarantee for the challenging rank estimation problem and is of independent interest.

4.3 Factor Adjustment for Overlapping Image Classification

In this section, we focus on applying the augmented tensor factor model proposed in Section 4.2 to address challenges in overlapping image classification. To illustrate our ideas, we consider a binary classification between two sets of gray-scale images. Let $\{\mathcal{X}_i^{(+)} \in \mathbb{R}^{p_1 \times p_2}\}_{i=1}^{n_1}$ and $\{\mathcal{X}_i^{(-)} \in \mathbb{R}^{p_1 \times p_2}\}_{i=1}^{n_2}$ be two observable sets of images with positive and negative class labels, respectively. Without loss of generality, we assume the images have been pre-processed to have a fixed size of p_1 by p_2 pixels. Denote $\{\mathcal{X}_i\}_{i=1}^n = \{\mathcal{X}_i^{(+)}\}_{i=1}^{n_1} \cup \{\mathcal{X}_i^{(-)}\}_{i=1}^{n_2}$ be the augmented sample with $n = n_1 + n_2$. Our goal is to train a classifier on this sample and predict the class label of images in a testing sample $\{\widetilde{\mathcal{X}}_i \in \mathbb{R}^{p_1 \times p_2}\}_{i=1}^m$.

To avoid the distraction of the pervasive component and improve the classification accuracy, we propose a tensor factor adjustment image classification procedure which can be introduced by the following two phases.

Training phase

1. We treat the training sample as a three-way tensor $\{\mathcal{X}_i\}_{i=1}^n$ and model it with the augmented tensor factor model

$$\mathcal{X}_i = \mathcal{G}_i \times_1 \boldsymbol{U}_1 \times_2 \boldsymbol{U}_2 + \mathcal{S}_i, \quad \text{for } i = 1, \ldots, n.$$

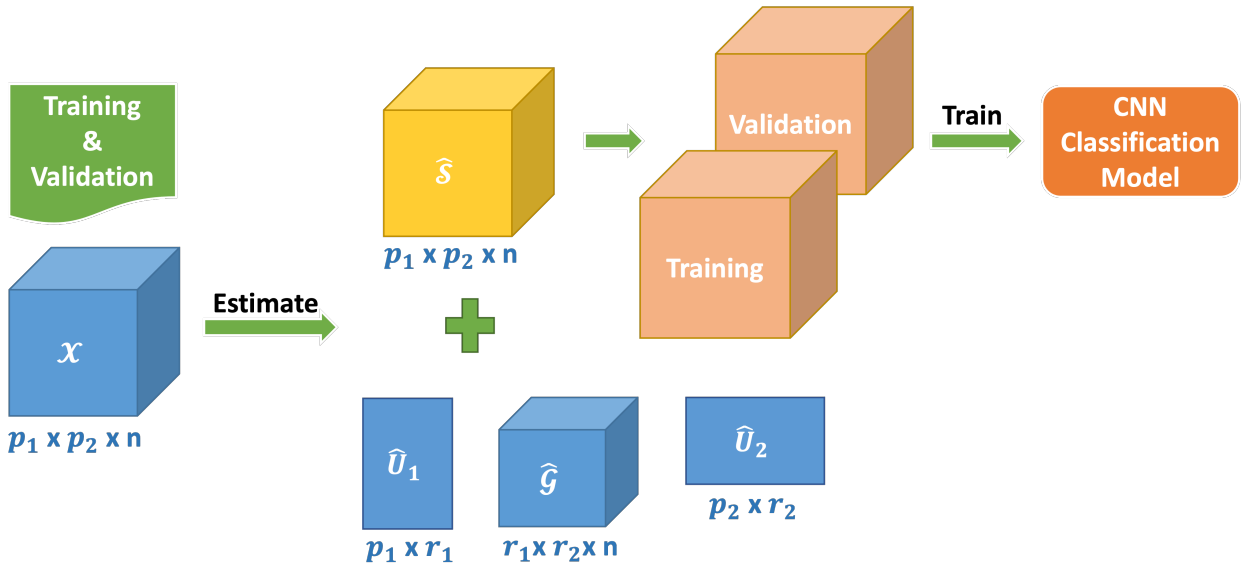


Figure 4.2: A flowchart for the training phase of the factor adjustment method.

2. Obtain the matricization of the above tensor factor model by the method introduced in Section 4.2.2.

3. Estimate the ranks by the modified eigen-ratio method introduced in Section 4.2.3.

4. Estimate factor loading matrices and latent tensor factors by the method introduced in Section 4.2.2. Then, by (4.4), we estimate the specific components as $\{\widehat{\mathcal{S}}_i\}_{i=1}^n$.

5. We train a classifier on $\{\widehat{\mathcal{S}}_i\}_{i=1}^n$ with the true class labels.

Testing phase

1. We regress the testing sample $\{\widetilde{\mathcal{X}}_i\}_{i=1}^m$ onto the estimated factor loading matrices from the training phase.

2. Estimate the specific components of the testing sample as the regression residuals, denoted as $\{\widetilde{\mathcal{S}}_i\}_{i=1}^m$.

3. We apply the classifier trained in the training process to classify $\{\widetilde{\mathcal{S}}_i\}_{i=1}^m$. The classification results will be used to predict the class labels of $\{\widetilde{\mathcal{X}}_i\}_{i=1}^m$.

The flowcharts for the training and testing phases are summarized in Figure 4.2 and Figure 4.3, respectively.

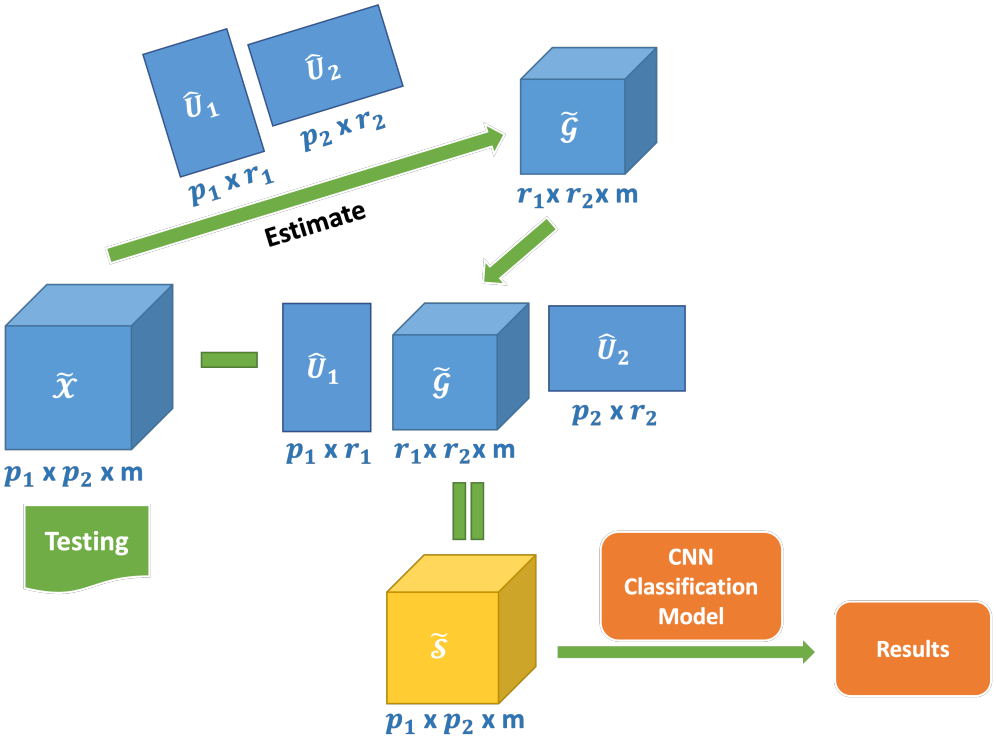


Figure 4.3: A flowchart for the testing phase of the factor adjustment method.

4.4 Synthetic Experiments

In this section, we carefully evaluate the performance of the proposed tensor factor adjustment image classification procedure using various synthetic experiments. In Section 4.4.1, we present the experiment settings and implementation details. The results of the experiments are reported and analyzed in Section 4.4.2.

4.4.1 Experiment Settings

For each experiment, we generate a positive sample $\{\mathcal{X}_i^{(+)}\}_{i=1}^{n_1}$ and a negative sample $\{\mathcal{X}_i^{(-)}\}_{i=1}^{n_2}$ from the following tensor factor model

$$\begin{cases} \mathcal{X}_i^{(+)} &= \beta \mathcal{G}_i \times_1 \boldsymbol{U}_1 \times_2 \boldsymbol{U}_2 + \mathcal{S}_i^{(+)} + \mathcal{W}_i, \\ \mathcal{X}_i^{(-)} &= \beta \mathcal{G}_i \times_1 \boldsymbol{U}_1 \times_2 \boldsymbol{U}_2 + \mathcal{S}_i^{(-)} + \mathcal{W}_i, \end{cases}$$

where β is a nonnegative scalar parameter to control the overlapping amount between the positive and negative samples, and $\mathcal{W}_i$ is a white noise term to add some randomness to the specific components.

The data-generating process for each component in the above model is summarized as follows.

1. The elements in latent tensor factors $\{\mathcal{G}_i\}_{i=1}^{n_1+n_2}$ are drawn from i.i.d. uniform distribution between $-a$ and a, where a is a positive parameter to partially control the signal to noise ratio.

2. The factor loading matrices $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$ are designed to be orthonormal matrices to satisfy the identification conditions.

3. The elements in the white noise term $\mathcal{W}_i$ are drawn from i.i.d. standard Gaussian distribution.

4. The specific components $\mathcal{S}_i^{(+)}$ and $\mathcal{S}_i^{(-)}$ are generated from some class specific image patterns. To be specific, we consider the following four pairs of image patterns: (a) none vs. random filled circle; (b) random filled square vs. random filled circles; (c) random filled oval vs. random filled circles, and (d) random handwritten 1 vs. random handwritten 7 drawn from the MNIST dataset. We illustrate these four settings in Figure 4.4.

Each synthetic example consists of a signal component, denoted as $\mathcal{S}_i + \mathcal{W}_i$, and a noise component, denoted as $\mathcal{G}_i \times_1 \boldsymbol{U}_1 \times_2 \boldsymbol{U}_2$. In order to ensure a fair comparison across all settings, it is important to control the signal-to-noise ratio at an approximate level in each setting. While the signal component can be easily held and set to a desired amount since its generation procedure is identical for each setting and

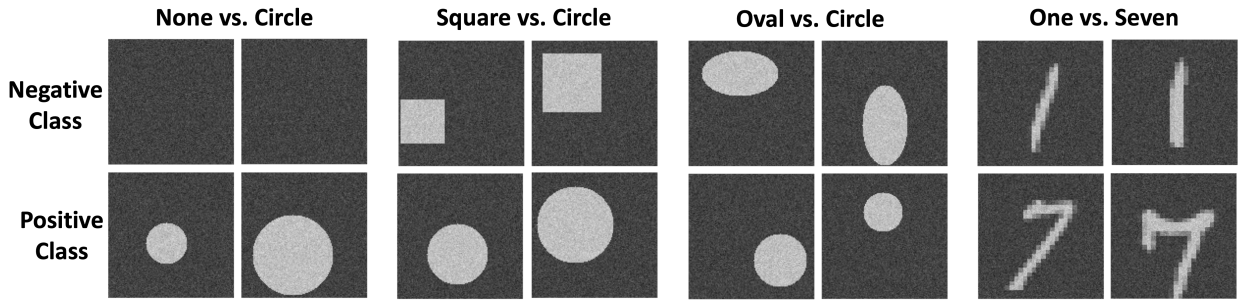


Figure 4.4: Synthetic specific component examples for each setting.

irrelevant to ranks, the noise amount is sensitive to the ranks of the pervasive component, which affects the signal-to-noise ratio of the generated images.

To achieve the fixed signal-to-noise ratio goal, we must control the noise amount with the signal amount unchanged. In our experiments, we measure the noise amount using the variance of the pervasive component elements, which can be denoted as $\text{Var}(\text{vec}(\boldsymbol{\mathcal{P}}))$ if we vectorize $\{\mathcal{P}_i\}_{i=1}^{n_1+n_2}$ into a column vector $\text{vec}(\boldsymbol{\mathcal{P}})$. We can approximate the variance as follows:

$$\text{Var}(\text{vec}(\boldsymbol{\mathcal{P}})) \approx \text{Var}(\text{vec}(\boldsymbol{\mathcal{G}})) \cdot \prod_{i=1}^{2} r_i \cdot \text{Var}(\text{vec}(\mathbf{U}_i)) = \frac{a^2 r_1 r_2}{3 p_1 p_2},$$

where $\text{Var}(\text{vec}(\boldsymbol{\mathcal{G}}))$ and $\text{Var}(\text{vec}(\mathbf{U}_i))$ are the variances of the factor loading matrix vectorizations and a, r_i, p_i are parameters of the model, with $i = 1, 2$.

When fixing the parameter a but increasing the ranks (r_1, r_2), the variance $\text{Var}(\text{vec}(\boldsymbol{\mathcal{P}}))$ increases, which leads to an increase in noise amount and a decrease in signal-to-noise ratio. Figure 4.5 (a) shows the variance change using density plots of pervasive component elements. To keep all settings at approximately the same signal-to-noise ratio, we adjust the parameter a as $a(r_1, r_2) = \sigma\sqrt{\frac{3p_1p_2}{r_1r_2}}$, where σ is the desired noise level. This ensures that the value of $\text{Var}(\text{vec}(\boldsymbol{\mathcal{P}}))$ remains unchanged at σ^2. Figure 4.5 (b) provides examples after the parameter adjustment, where all the distributions are centered around a similar range.

Further, we use the parameter β to control the overlapping ratio between the two classes. When we increase β, the pervasive components weigh more in both classes; hence, the true signals in the specific

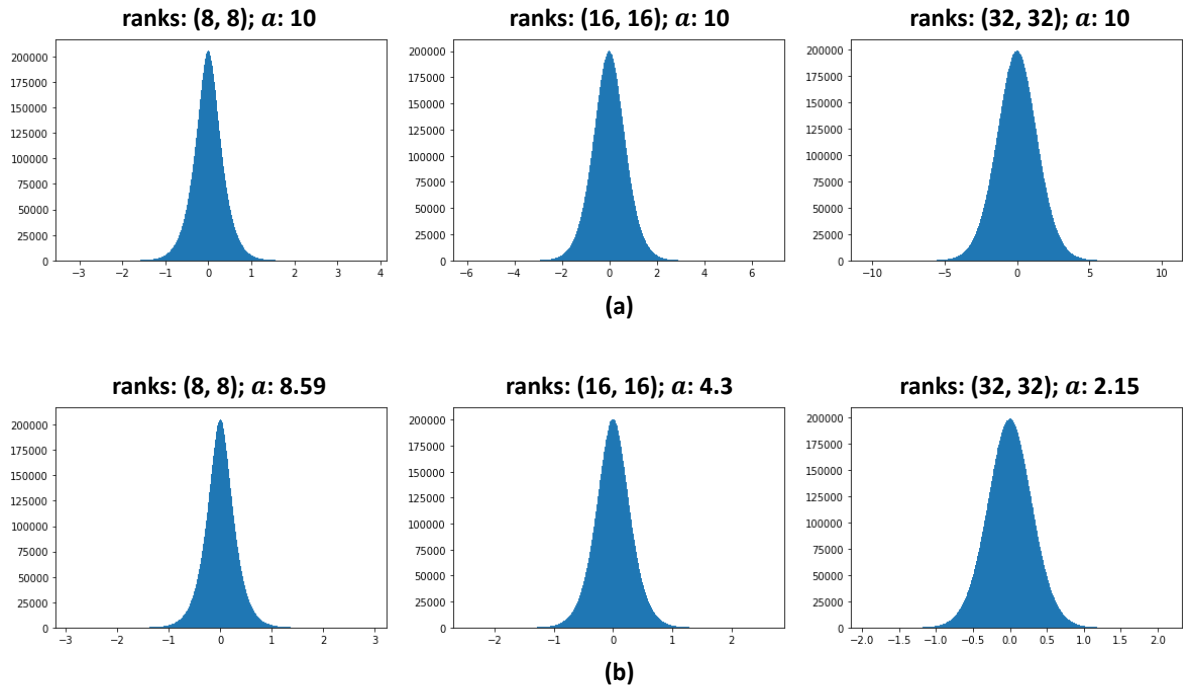


Figure 4.5: Example density plots of the simulated pervasive component elements (a) with and (b) without parameter a adjustment. (a) keep the value of parameter a fixed at 10, as ranks increase from $(8, 8)$ to $(32, 32)$, the distribution of the simulated pervasive component elements became more spread out. (b) adjust the parameter a according to the ranks. The pervasive component elements are controlled to be distributed within a similar range.

components are harder to recognize. Figure 4.6 provides a list of synthetic examples with β increased from 1 to 9. When the pervasive component dominates the data-generating process, as expected, the images in positive and negative classes are very hard to distinguish.

4.4.2 Implementation and Results

As outlined in Section 4.4.1, we generate synthetic images using the four signal settings demonstrated in Figure 4.4. For each setting, we set the image sizes to be $p_1 \times p_2 = 128 \times 128$ and the sample sizes $n_1 = n_2 = 2,400$. The ranks (r_1, r_2) are set as $(8, 8)$, $(16, 16)$ and $(32, 32)$ to cover low to moderate rank settings. The generated samples are then divided into training, validation, and testing sets as specified in Table 4.5.1. Throughout this chapter, we use a modified VGG16 model introduced in Section 3.4.1 as

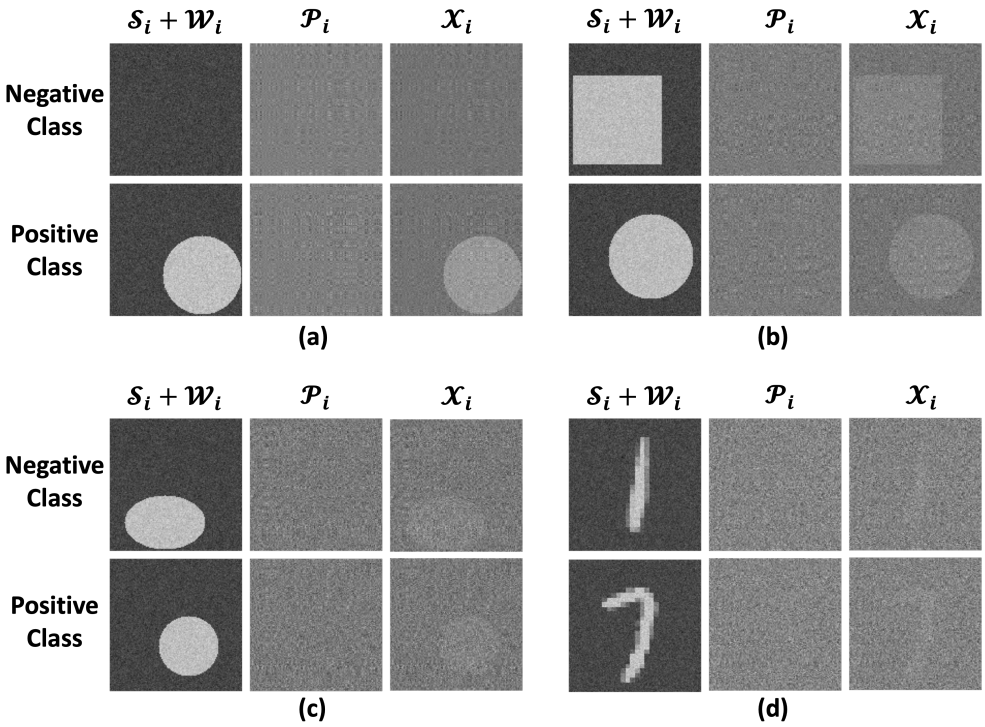


Figure 4.6: Visualization of synthetic examples. (a) none vs. circle with ranks $(8, 8)$ and $\beta = 1$; (b) square vs. circle with ranks $(16, 16)$ and $\beta = 3$; (c) oval vs. circle with ranks $(32, 32)$ and $\beta = 6$; (d) one vs. seven with ranks $(64, 64)$ and $\beta = 9$. For each panel, the three columns from left to right represent specific components (signal with white noise), pervasive components (overlapping noise), and observed images.

our binary image classifier. We use the training set to train our classifier and the validation set to tune hyper-parameters. The trained classifier will be used to predict the class labels in the test set. The prediction accuracy is measured by the correct classification rate.

For each signal setting, we consider and compare the following three scenarios.

Non-overlapping: We generated a pair of non-overlapping positive and negative samples $\{\mathcal{X}_i^{(+)}\}_{i=1}^{n_1}$ and $\{\mathcal{X}_i^{(-)}\}_{i=1}^{n_2}$ by setting the overlapping parameter $\beta = 0$. Then, it is equivalent to directly train and test the classifier on specific components plus white noises, i.e., $\{\mathcal{S}_i^{(+)} + \mathcal{W}_i\}_{i=1}^{n_1}$ and $\{\mathcal{S}_j^{(-)} + \mathcal{W}_j\}_{j=1}^{n_2}$.

Overlapping: We set the overlapping parameter β to be large enough such that the modified VGG16 classifier behaves as badly as random guesses, i.e., the correct classification rate on the testing set is about

0.5. In this scenario, we simulate the images that are highly overlapping between two classes, and most existing classifiers are not tailored to handle such cases.

Factor adjustment: We use the same highly overlapping data as generated in the overlapping scenario but apply the factor adjustment procedure as described in Section 4.3. Then, the classifier is trained and tested on the estimated specific components $\{\widehat{S}_i\}_{i=1}^{n_1+n_2}$ instead of the overlapping raw samples.

Table 4.4.1 presents the detailed experiment results. In all scenarios, the modified VGG16 classifier can perfectly predict the true class labels in the testing set for non-overlapping cases, but performs as poorly as random guesses for overlapping cases. This demonstrates that the presence of pervasive components can greatly hinder even simple image classification tasks, as the two classes are highly overlapping or correlated with each other. However, as expected, the proposed factor adjustment method effectively recovers the true signals dominated by the pervasive component and decorrelates the two overlapping classes. As a result, the factor adjustment method achieves near-perfect correct classification rates for all scenarios. In general, the factor adjustment method works as if we were classifying with the true signals.

Table 4.4.1: Correct classification rate on testing set for synthetic experiments.

	Dataset	Ranks (r_1, r_2)		
		(8, 8)	(16, 16)	(32, 32)
None vs. Circle	Non-overlapping	0.995		
	Overlapping	0.5	0.5	0.5
	Factor adjusted	0.995	0.995	0.9975
Square vs. Circle	Non-overlapping	1.0		
	Overlapping	0.5	0.5	0.5
	Factor adjusted	0.995	0.995	0.9975
Oval vs. Circle	Non-overlapping	1.0		
	Overlapping	0.5	0.5	0.5
	Factor adjusted	1.0	0.9925	0.975
One vs. Seven	Non-overlapping	0.9925		
	Overlapping	0.5	0.5	0.5
	Factor adjusted	0.9875	0.9875	0.9825

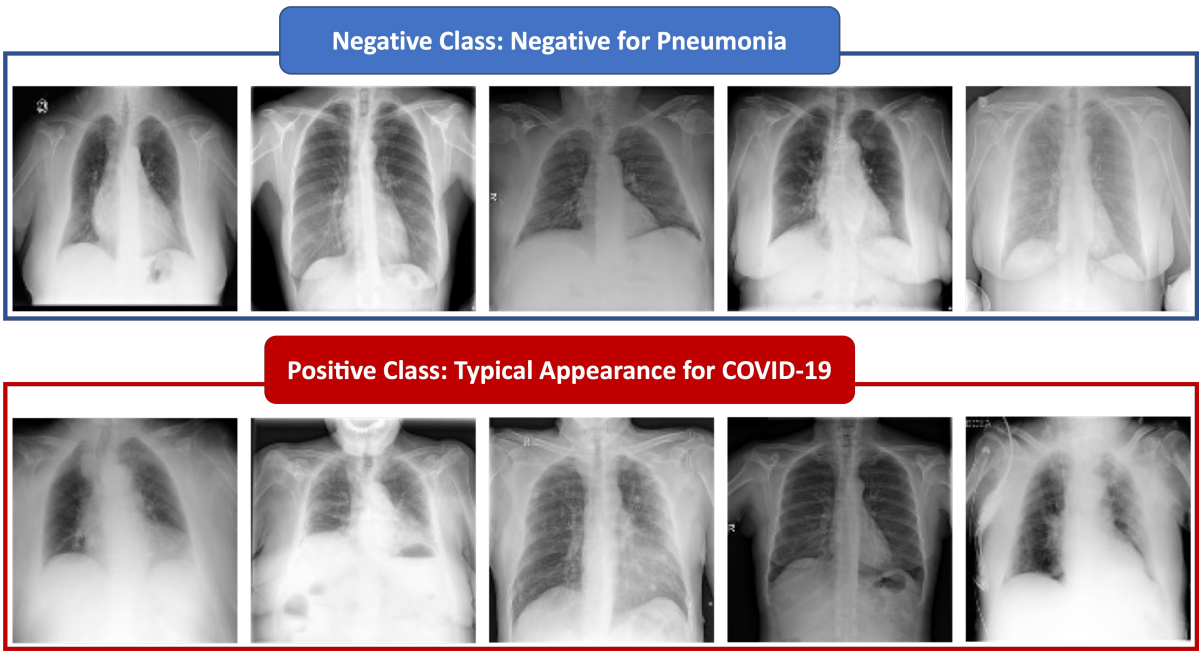


Figure 4.7: Sample images in the COVID-19 CXR dataset.

4.5 Frontal Chest X-ray Image Classification for COVID-19 Diagnostic

The repeated waves of COVID-19 have infected over 660 million people and resulted in over 6.5 million deaths worldwide since its emergence in early 2020. Recently, a new wave of infections hit China as it relaxed its "Zero-COVID" policy. While nucleic acid amplification tests and antigen tests are widely accepted for detecting positive cases, frontal chest X-ray (CXR) image analysis remains a prominent method for diagnosing lung infections and pneumonia, which can be critical indicators for potential severe symptoms of COVID-19 (Ismael and Şengür, 2021; Jain et al., 2021; H. Li et al., 2022). However, the typical appearance of a COVID-19 lung infection can be complex for non-experts to recognize, as the differences between negative and positive images are subtle and largely obscured by pervasive noises such as skeletons, organs, and shadows. In that sense, it is essential to develop an accurate and efficient classification method to adjust for pervasive noises and identify positive CXR images from negative ones.

In this section, we apply the proposed augmented tensor factor model and factor adjustment method to tackle this task.

Table 4.5.1: Synthetic and real data splitting sample sizes.

	Synthetic Images		COVID-19 CXR Images	
	Negative Class	Positive Class	Negative Class	Positive Class
Training	2,000	2,000	1,223	2,325
Validation	200	200	200	200
Testing	200	200	200	200

4.5.1 Data Description

The dataset we study is collected and provided by the Society for Imaging Informatics in Medicine (SIIM) [1]. The dataset combines high-quality CXR images from the existing public BIMCV (Vayá et al., 2020) and MIDRC-RICORD (Tsai et al., 2021) COVID-19 datasets and annotates these chest radiographs by 22 radiologists. Following the guidelines listed in (Lakhani et al., 2021), radiologists have annotated CXR images into four mutually exclusive categories, including "Negative for Pneumonia", "Typical Appearance for COVID-19", "Indeterminate Appearance for COVID-19" and "Atypical Appearance for COVID-19". Such a labeling strategy is based on the prior knowledge of radiographic manifestations of COVID-19 and can contribute to the consistency in radiology reporting (Litmanovich et al., 2020).

In this section, we consider a binary classification problem by focusing on images from the"Negative for Pneumonia" and "Typical Appearance for COVID-19" classes. To keep the presentation simple, we denote "Negative for Pneumonia" as the negative class and "Typical Appearance for COVID-19" as the positive class. The negative and positive classes contain $1,623$ and $2,725$ images, respectively. All images have been pre-processed to have a fixed size of 128×128 gray-scale pixels. Figure 4.7 lists several sample images from both positive and negative classes. From a non-professional perspective, the images from the two classes are highly overlapping as they are dominated by skeletons and organs with no valuable

[1] Avaliable at https://www.kaggle.com/c/siim-covid19-detection

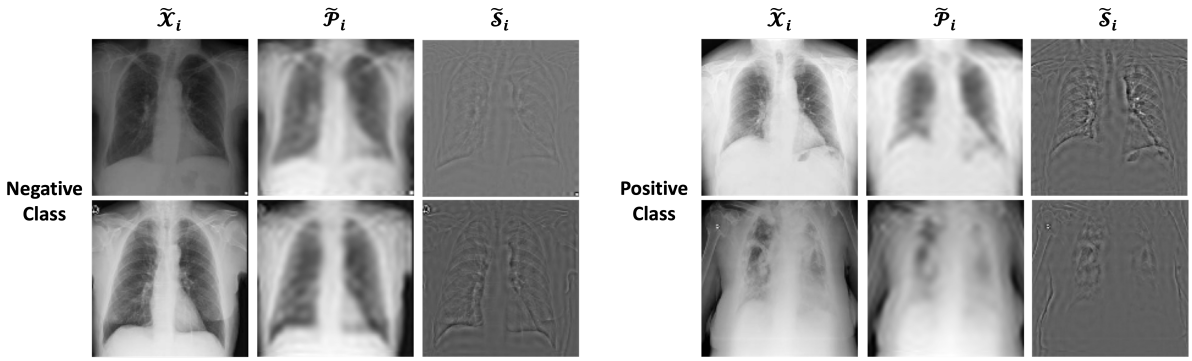


Figure 4.8: Examples of the raw images, the estimated pervasive components, and the estimated specific components from the test CXR images.

information to diagnose COVID-19. The useful information resides in the lung area, but the signals are weak and largely obscured by pervasive noises. In our experiment, we randomly divide both positive and negative classes into training, validation, and testing sets, as detailed in Table 4.5.1.

4.5.2 Analysis Results

We perform image classification on positive and negative classes of COVID-19 using a modified VGG16 classifier. The classifier is trained on the training set, and its hyperparameters are fine-tuned utilizing the validation set. Subsequently, the trained classifier is employed to predict class labels in the test set. Due to the imbalanced nature of the two classes, a threshold of 0.66 is utilized instead of the standard 0.5 to account for the sample size ratio in the training set. The resulting classification accuracy is 71%, which is considered as the baseline performance and presented in Table 4.5.2. The baseline performance results in 52 false-positive errors (misclassifying negative cases as positive) and 64 false-negative errors (misclassifying positive cases as negative).

Next, we apply the factor adjustment procedure as outlined in Section 4.3. We estimate and adjust the pervasive components during the training and prediction of class labels. The ranks are selected using the modified eigen-ratio method described in Section 4.2.3. Specifically, after setting the prescribed upper bound to one-fourth of the tensor size ($K_{max} = 32$), the eigen-ratio method chooses $(r_1, r_2) = (28, 30)$.

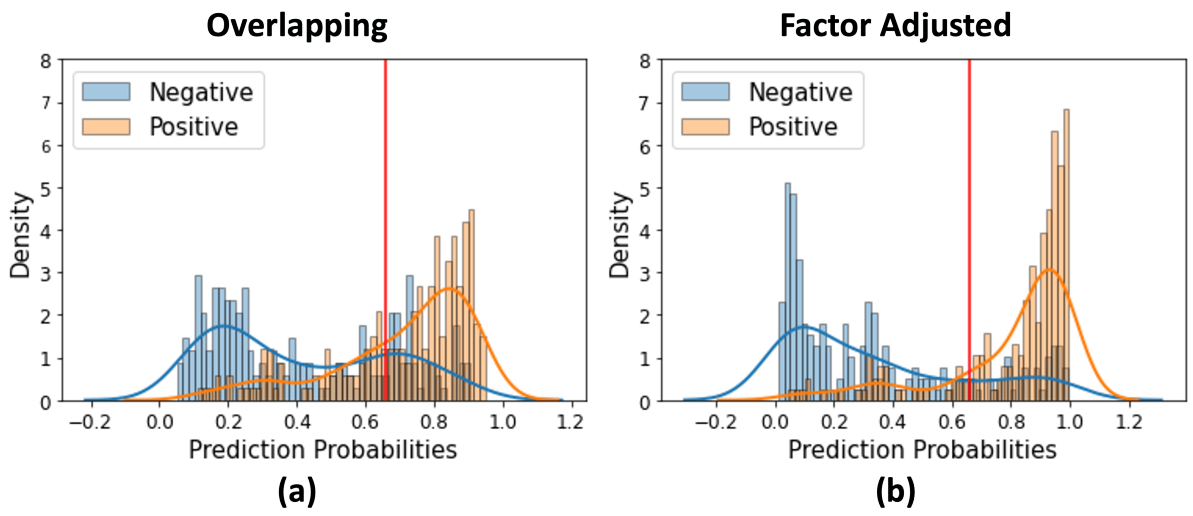


Figure 4.9: Histograms and density plots of classification probabilities for test images: (a) the modified VGG16 model trained on overlapping and (b) factor adjustment method. In both panels, the red vertical line represents the classification threshold.

Figure 4.8 illustrates several pairs of positive and negative CXR images in the testing set, along with their estimated pervasive and specific components. The pervasive components effectively identify the human skeleton regions, while the estimated specific components are more focused on the lung area in comparison to the raw images. Figure 4.9 compares the classification probabilities of test images predicted by the modified VGG16 model trained on raw overlapping samples (left panel) and the factor-adjusted samples (right panel). The raw samples have heavily overlapping positive and negative classes, making it challenging to establish a clear threshold. However, the factor-adjusted method reduces the overlap between the two classes and skews the negative class prediction distribution towards 0 and the positive class prediction distribution towards 1, indicating a more distinguishable estimation of specific components. The correct classification rate for the factor-adjusted method is 81.5%, which is a 10.5% improvement from the baseline. Both false-positive and false-negative values are also reduced.

Table 4.5.2: Summary of classification results on COVID-19 CXR testing images.

	K_{max}	Accuracy	AUC	FP	FN
Overlapping	–	0.71	0.8132	52	64
Factor Adjusted	16	0.7725	0.8574	58	33
	32	0.815	0.8673	39	35
	64	0.7775	0.8491	43	46
	128	0.755	0.8279	40	58

4.6 Conclusion and Future Work

This chapter is motivated by a COVID-19 chest X-ray image classification problem, where the positive and negative classes are largely overlapping as the images are dominated by common noises such as skeleton, organs, and shadow. The specific signals that can separate the two classes are weak and obscured by pervasive noises. We introduced a tensor factor model that decomposes the augmented samples into pervasive and specific components. The pervasive component is characterized by the production of latent tensor factors and factor loading matrices. We proposed a matricization plus PCA strategy to estimate this tensor factor model and investigated the theoretical properties of our estimators. Then, we developed a factor adjustment procedure to remove the pervasive component and apply the classifier directly on the specific components. The intuition is to adjust the noises present in the pervasive component and improve the signal-to-noise ratio in classification. The empirical performance of this procedure is well justified by various synthetic experiments. In the application of frontal chest X-ray image-based COVID-19 diagnosis, our method improves the baseline by 10.5% in terms of classification accuracy.

The choice of upper bound K_{max} in our proposed method can have an impact on the signal-to-noise ratio, as shown in Table 4.5.2. The classification accuracy initially increases and then decreases as K_{max} increases. When K_{max} is small, the model may not effectively remove enough noise from the specific signals. On the other hand, if K_{max} is too large, over-factorization of the data tensor may result in the pervasive component including not only noise but also signal. This can lead to partial removal of the

specific signals and poor performance on the testing images as the classifier has not learned enough from the training data. These findings suggest the need for further research in this area.

Chapter 5

Statistical and Machine Learning Models for COVID-19 Mortality Prediction

In this chapter, we conduct a comprehensive analysis of various statistical and machine learning models for forecasting daily deaths due to COVID-19 in the United States. The chapter is organized into four main sections, with the first section, Section 5.2, providing details on the data used in the study. Section 5.3 introduces the rolling-validation strategy and the machine learning models evaluated, including a discussion on incorporating eXogenous variables. The results of the experiments comparing the performance of the models are presented in Section 5.4. Finally, Section 5.5 compares the best model found in our experiments with 18 other models submitted to the United States Centers for Disease Control and Prevention (CDC) over a continuous 52-week period starting in December 2020 and ending in December 2021. These results are based on the findings of (Chaudhari et al., 2021; Toutiaee et al., 2021).

5.1 Introduction and Related Work

The coronavirus disease 2019 (COVID-19) began spreading at the start of 2020 and has since spread to every continent and country worldwide. On March 11^{th}, 2020, the World Health Organization (WHO) officially declared COVID-19 a pandemic. To date, the pandemic has resulted in over 758 million confirmed cases and nearly 7 million deaths globally. The United States has been one of the countries hardest hit by the pandemic, with over 102 million cases and over 1 million deaths ('WHO Coronavirus (COVID-19) Dashboard', 2021). COVID-19 has presented a rare opportunity for the study of pandemics due to the increased collection of data. Since the onset of the pandemic, various organizations have been gathering and compiling a wide range of healthcare data related to the pandemic, including the number of confirmed cases, deaths, hospitalizations, patients in Intensive Care Units (ICU), and individuals vaccinated.

The ability to accurately forecast the number of COVID-19 infections and deaths is crucial for healthcare workers and policymakers, as it enables them to effectively manage resources, control outbreaks, and take preventive measures to protect public health. The COVID-19 Forecast Hub [1] has collected hundreds of thousands of predictions from over 50 academic, industry, and independent research groups, who have used a variety of models, including SEIR-like models (Karlen, 2020; Srivastava et al., 2020), curve-fitting models (Nishimoto and Inoue, 2020), predictive models (COVID, Murray et al., 2020), and ensemble models (Abbott et al., 2020).

In this chapter, we focus on predictive modeling techniques, specifically machine learning and statistical models such as Seasonal Autoregressive Integrated Moving Average (SARIMA), SARIMA with eXogenous factors (SARIMAX), Vector Auto-Regressive (VAR), and Minimax Concave Penalty (MCP). These models have the potential to provide accurate short- and near-term forecasts due to their high parameterizability and data-dependence. On the other hand, other techniques that incorporate the underlying mechanics of the pandemic, such as SEIR models, may perform better for longer-term forecasting (Furtado, 2021).

[1] Avaliable at https://covid19forecasthub.org/eval-reports/

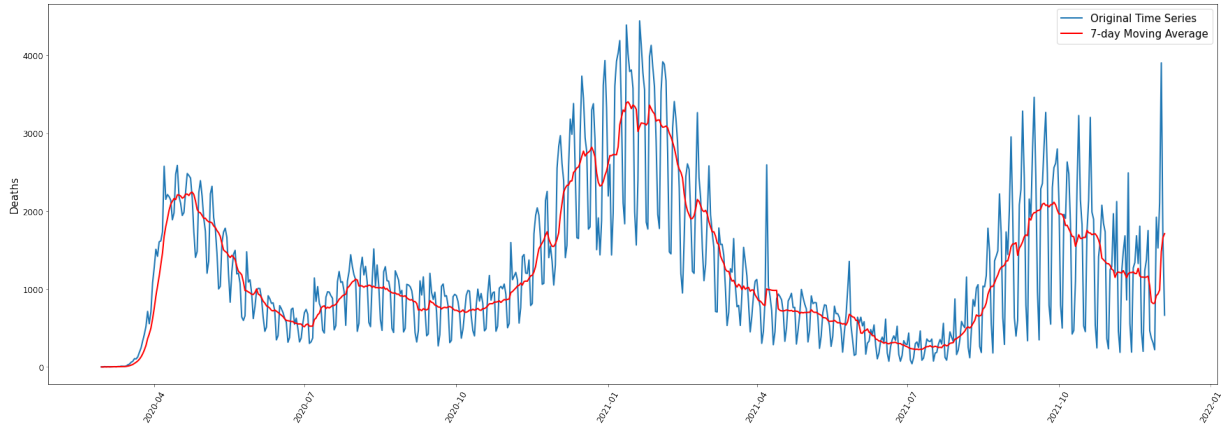


Figure 5.1: Daily Deaths in the US

We extend current research on pandemic modeling by examining statistical and machine learning methods for forecasting *daily deaths* caused by COVID-19 in the United States. We focus on daily deaths because of their critical importance and their impact on model interpretability. Although cumulative deaths have also been studied, models tend to perform similarly as they predict slowly changing large numbers, while differences become more pronounced when focusing on daily deaths. Figure 5.1 shows the daily deaths and weekly moving average in the US.

In addition, we incorporate exogenous medical information-related variables, such as "COVID-19 hospitalizations", "test positivity rate", and "number of individuals vaccinated", to improve the performance of our predictive models. We demonstrate that such exogenous medical variables combined with lagged variables within the predictive models can significantly enhance prediction accuracy and facilitate the monitoring of virus spread.

5.2 Data Description

Since the start of the pandemic, there have been numerous publicly available datasets for COVID-19 cases, deaths, hospitalizations, and other relevant information. However, these datasets each have their

own strengths and limitations. For instance, the COVID Tracking Project dataset [2] was used by many research groups, but it ceased collecting data on March 7^{th}, 2021, after starting on January 13^{th}, 2020. The official CDC COVID Data Tracker dataset [3] and the dataset provided by the Center for Systems Science and Engineering at Johns Hopkins University (CSSE-JHU) (Dong et al., 2020) have the most up-to-date time-series data on COVID-19 cases and deaths in the US, but they do not include information on hospitalizations, which is crucial for our study.

In our research, we utilize the COVID-19 dataset provided by Our World In Data (OWID) (Ritchie et al., 2020), which is currently the most comprehensive dataset on COVID-19 cases, deaths, hospitalizations, ICU patients, tests, vaccination, and other relevant information. The data is aggregated from multiple sources, with death and case data collected from the CSSE-JHU dataset. In addition to national-level daily deaths and cases, the dataset also includes several medical variables, such as "hospitalized patients", "ICU patients", "individuals vaccinated", and "people fully vaccinated".

The data we collected spans from January 22^{nd}, 2020 to December 4^{th}, 2021 (683 days), excluding the first 38 days due to missing values. We start from February 29^{th}, 2020, when the first COVID-19 death in the US was recorded. As a result, the national-level time-series analyzed in our research encompasses 645 days. We use the first 258 days as the initial training set and employ a two-week ahead rolling window forecast on the remaining 387 days as the test set.

5.3 Methods

5.3.1 Rolling Validation for Multiple Horizons

Classical multi-fold cross-validation approaches are not suitable for time-series data due to serial dependence. In this chapter, we adopt a rolling validation scheme. As stated in Section 5.2, we reserve 258 days in the time series as the training set and use the remaining 387 days for a rolling window forecast. The model is trained on the training set and used to make a 14-day ahead prediction. Then, the rolling window

[2] Avaliable at https://covidtracking.com/data/national
[3] Avaliable at https://covid.cdc.gov/covid-data-tracker

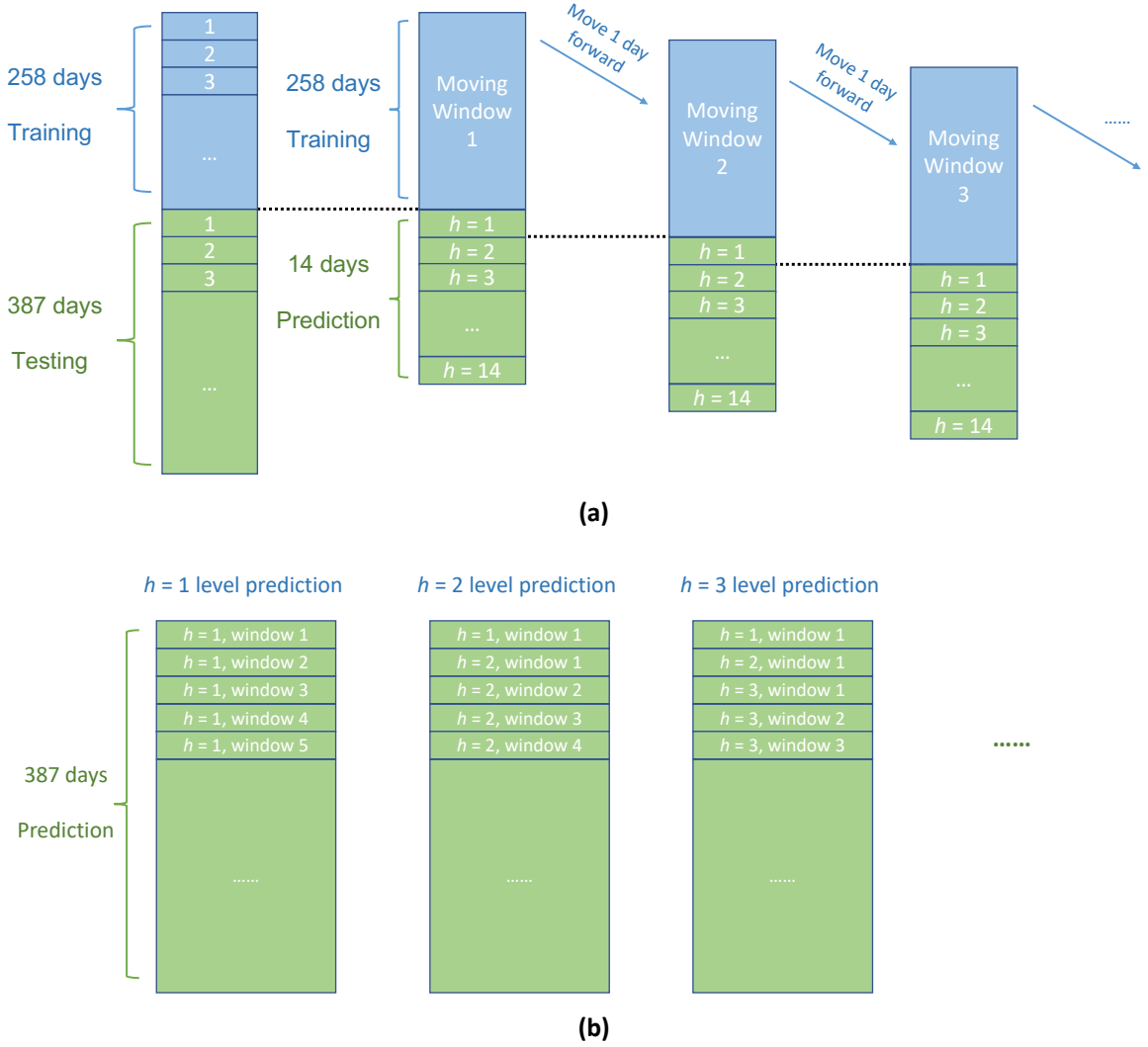


Figure 5.2: Graphical illustration of (a) rolling-validation scheme and (b) horizon level prediction.

is moved forward by one day, which involves appending the first value from the test set to the training set and removing the first value from the training set. This process is repeated until the rolling window reaches the end of the test set, and 14-day forecasts are obtained at each step. This procedure is depicted in Figure 5.2 (a).

In the end, predictions are generated for all 14 horizon levels, where the *horizon* (h) refers to the number of days into the future for which forecast values are generated. For instance, the prediction values

on horizon level $h = 1$ are obtained by aggregating the first-day forecasts of each rolling window. The accuracy of the predictions is measured using the symmetric mean absolute percentage error (sMAPE), with a smaller value indicating better performance. The procedure is illustrated in Figure 5.2 (b).

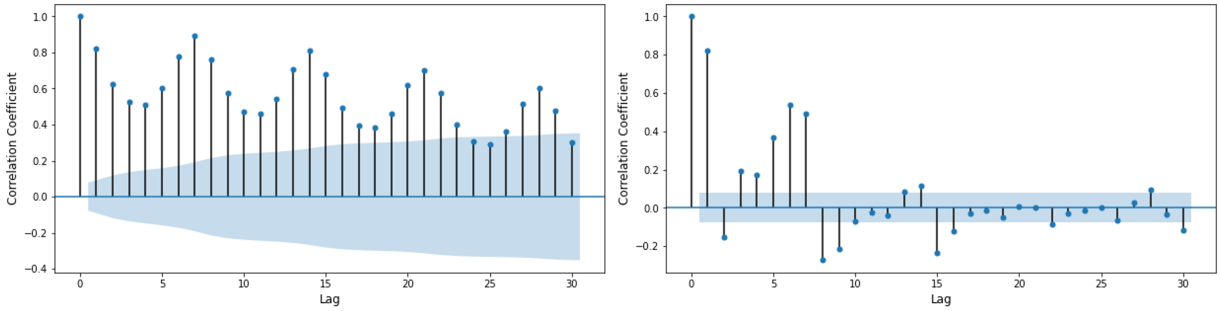


Figure 5.3: (a) PACF; (b) ACF plot of the daily deaths.

5.3.2 SARIMA and SARIMAX

The Partial Auto-Correlation Function (PACF) on the daily deaths data, as presented in Figure 5.3 (b), shows the first 15 lags are significant and there is a clear weekly pattern. Weekly seasonality of daily deaths can also be observed using the Auto-Correlation Function (ACF), as shown in Figure 5.3 (a). Then, we propose to fit the daily deaths by a Seasonal Auto-Regressive, Integrated, Moving Average (SARIMA) (Arunraj et al., 2016) model defined as

$$\varphi_p(B)\Phi_P\left(B^s\right)\nabla^u\nabla_s^v z_t = \theta_q(B)\Theta_Q\left(B^s\right)\varepsilon_t, \tag{5.1}$$

where z_t is a variable to forecast, i.e., the logarithm of *daily deaths*, $t = 1, 2, \ldots$, $\varphi_p(B)$ is a regular AR polynomial of order p, $\theta_q(B)$ is a regular MA polynomial of order q, $\Phi_P\left(B^s\right)$ is a seasonal AR polynomial of order P, and $\Theta_Q\left(B^s\right)$ is a seasonal MA polynomial of order Q. The differencing operator ∇^u and the seasonal differencing operator ∇_s^v eliminate the non-seasonal and seasonal non-stationarity, respectively.

The SARIMA with eXogenous factor (SARIMAX) model is an extension of the SARIMA model in (5.1), which can include exogenous variables, such as hospitalization, ICU occupancy rate, and vaccination

rate. The SARIMAX model can be defined as:

$$\varphi_p(B)\Phi_P\left(B^s\right)\nabla^u\nabla_s^v z_t = \theta_q(B)\Theta_Q\left(B^s\right)\varepsilon_t + \sum_{i=1}^{m}\beta_i x_t^i,$$

where $\{x_t^1, \ldots, x_t^m\}$ are the m exogenous variables defined at time t with coefficients $\{\beta_1, \ldots, \beta_m\}$. Further, we apply a log transformation to stabilize the changing variance.

5.3.3 Minimax Concave Penalty

We also consider a penalized linear regression approach to predict daily deaths by historical data and medical variables. We denote $z_t^* = \log z_t - \log z_{(t-7)}$ as the weekly log-return of *daily deaths* at time t and $\mathbf{x_t}$ are m explanatory medical variables at time t. We linearly regress z_t^* on $\{z_{t-k}^*, \ldots, z_{t-(2k-1)}^*\}$ and $\{\mathbf{x}_{t-k}, \ldots, \mathbf{x}_{t-(2k-1)}\}$, where $k = 14$, as we are conducting a 14-day ahead forecast. Since the medical variables are highly correlated, we first implement a sure independent screening (Fan and Lv, 2008) procedure to reduce the dimensionality of $\mathbf{x_t}$. As a result, only the top seven explanatory variables are included in the following partial penalized regression model:

$$Q(\boldsymbol{\beta} \mid \mathbf{X}, \mathbf{z}) = \frac{1}{2N}\|\mathbf{z}^* - \mathbf{X}^\dagger\boldsymbol{\beta}\|^2 + \sum_{j=1}^{k(m+1)} P_\gamma\left(\beta_j; \lambda\right),$$

where $\mathbf{z}^* = (z_1^*, \ldots, z_N^*)^{\mathrm{T}} \in \mathbb{R}^N$ is the vector of weekly log-return of *daily deaths* over a sample size N, $\mathbf{X}^\dagger \in \mathbb{R}^{N\times k(m+1)}$ is the augmented design matrix of all predictors (e.g. lags of z_t and explanatory variables), and $\boldsymbol{\beta} = (\beta_1, ..., \beta_{k(m+1)})^{\mathrm{T}}$ is a vector of unknown regression coefficients. $P_\gamma(\cdot; \lambda)$ is the Minimax Concave Penalty (MCP) (C.-H. Zhang, 2010) which satisfies

$$P_\gamma(\beta; \lambda) = \begin{cases} \lambda|x| - \frac{\beta^2}{2\gamma}, & \text{if } |\beta| \le \gamma\lambda, \\ \frac{1}{2}\gamma\lambda^2, & \text{if } |\beta| > \gamma\lambda. \end{cases}$$

Since $k(m+1)$ predictors in the regression model tend to overfit, the number of predictors has been reduced from $k(m+1)$ to $m^* = 7$. The $m^* = 7$ explanatory variables are the top m^*, which have larger marginal Pearson's correlation coefficients with the response variable. We also tried the L_1 penalty (LASSO) (Tibshirani, 1996) and smoothly clipped absolute deviation (SCAD) (Gu and Gu, 2013) as two alternative penalty functions. We find MCP performs the best among the three due to its smaller penalty away from zero.

5.3.4 Vector Auto-Regressive Model

Vector Auto-Regression (Zivot and Wang, 2006) is another popular approach to model and predict multivariate time series. For a d-dimensional response vector of interest, say:

$$\mathbf{z}_t = (z_t^{(1)}, z_t^{(2)}, \ldots, z_t^{(d)})^{\mathrm{T}},$$

a vector auto-regressive model of order q, i.e. VAR(q), is defined as

$$\mathbf{z}_t = \boldsymbol{\alpha} + \Phi^{(0)}\mathbf{z}_{t-q} + \Phi^{(1)}\mathbf{z}_{t-q+1} + \ldots + \Phi^{(q-1)}\mathbf{z}_{t-1} + \boldsymbol{\epsilon}_t,$$

where $\boldsymbol{\alpha} \in \mathbb{R}^d$ is an intercept vector, $\Phi^{(s)} \in \mathbb{R}^{d\times d}$ for $s = 0, \ldots, p-1$ are regression coefficient matrices, and $\boldsymbol{\epsilon}_t \in \mathbb{R}^d$ is an error vector.

Similar to the data pre-processing procedure described in Section 5.3.3, a weekly log-return has been applied to both the response vector and medical variables to remove the seasonality.

Table 5.3.1: Multi-Horizon (h) Rolling Forecasts for the United States (National Level Data): Competitive Models (sMAPE).

Horizon	SARIMA	**SARIMAX**	MCP	VAR
$h = 1$	19.42	18.49	20.50	20.26
$h = 2$	20.21	19.18	20.96	20.86
$h = 3$	20.82	19.33	21.56	21.34
$h = 4$	21.30	19.87	21.87	21.64
$h = 5$	21.46	19.71	21.93	21.64
$h = 6$	22.08	20.22	22.01	21.61
$h = 7$	22.44	20.40	22.06	21.50
$h = 8$	25.57	22.18	25.77	24.51
$h = 9$	26.07	22.49	25.84	24.67
$h = 10$	26.74	22.53	26.48	24.88
$h = 11$	27.81	22.96	26.77	25.06
$h = 12$	28.40	22.74	26.79	25.12
$h = 13$	29.46	23.28	27.04	25.19
$h = 14$	29.65	23.45	27.29	25.28
Average	24.39	**21.20**	24.06	23.11

5.4 Experiments and Numerical Results

5.4.1 Hyperparameters

SARIMA: In the SARIMA model, the hyperparameter, p, was tuned from 1 to 6, and the best-performing model was SARIMA $(2, 1, 4) \times (3, 1, 1, 7)$.

SARIMAX: For the SARIMAX model, medical variables such as "icu_patients", "hosp_patients", "people_vaccinated", etc. were selected among several hospitalization and vaccination-related variables. The hyperparameter, p, was also tuned following the same strategy as in the SARIMA model. The best-performing model was SARIMAX $(1, 1, 4) \times (3, 1, 1, 7)$ with medical variables "hosp_patients", "new_tests", "total_tests" and "people_fully_vaccinated".

MCP: In the MCP model, serial dependence among daily new cases in hospitals was considered. The new hospitalized count for each day was dependent on the previous 14 days of observations. The regularized parameters in MCP were selected through multi-fold cross-validation, resulting in 7 predictors being included in the MCP model.

VAR: Like the SARIMAX model, the VAR model contains multiple predictors, including "icu_patients", "hosp_patients", "positive_rate", and "new_deaths". The hyperparameters were selected using the Bayesian Information Criterion (BIC).

Table 5.4.1: Compare the forecast performance of the SARIMAX model with that of models generated by CDC groups.

Groups	1 Week Ahead Forecast	2 Weeks Ahead Forecast	3 Weeks Ahead Forecast	4 Weeks Ahead Forecast
Karlen	9.69(1)	10.81(1)	14.03(1)	20.87(3)
USC	12.33(2)	13.06(2)	16.38(2)	20.55(1)
BPagano	12.74(3)	14.97(3)	18.23(3)	20.62(2)
SARIMAX	**15.35(4)**	**19.22(6)**	**22.61(6)**	**25.57(5)**
Ensemble	**15.37(5)**	**18.04(5)**	**21.07(5)**	**26.43(6)**
CovidComplete	15.43(6)	17.33(4)	20.59(4)	26.58(7)
JHU-APL	16.32(7)	19.52(8)	24.39(8)	25.08(4)
GT-DeepCOVID	16.74(8)	20.77(10)	26.95(12)	34.29(14)
UMass-MB	16.78(9)	20.11(9)	26.61(10)	30.05(11)
JHU-CSSE	16.89(10)	19.37(7)	23.57(7)	27.11(8)
UCSD-NEU	17.51(11)	21.67(12)	26.80(11)	29.33(10)
MOBS	17.98(12)	21.29(11)	25.79(9)	29.30(9)
UM	18.03(13)	24.25(16)	29.88(16)	34.32(15)
LSHTM	19.03(14)	23.06(13)	27.85(13)	32.13(12)
MIT-ORC	19.20(15)	23.10(14)	29.61(14)	36.90(16)
ESG	19.97(16)	25.00(17)	31.65(17)	39.47(17)
Columbia	20.77(17)	24.20(15)	29.73(15)	33.45(13)
DDS	24.57(18)	44.77(18)	64.72(19)	72.20(19)
PSI	31.86(19)	44.81(19)	54.36(18)	61.16(18)

5.4.2 Results

In this section, we compare the performance of all the models discussed in Section 5.3. For each model, we make a 14-day ahead rolling window forecast as described in Section 5.3.1. The forecast performance is measured by the sMAPE score at each horizon $1 \leq h \leq 14$. The results of the national-level forecast are reported in Table 5.3.1.

The SARIMAX model performed the best on this dataset with the lowest sMAPE score. The VAR and MCP models were the second and third best methods in terms of the average sMAPE over all horizons. All models that incorporated eXogenous variables outperformed the SARIMA model.

5.5 Comparing the SARIMAX Model with Other Forecasting Models Submitted to the CDC

In this section, we compare the performance of the SARIMAX model with 18 other models that were submitted to the United States Centers for Disease Control and Prevention (CDC) during a 52-week period from December 2020 to December 2021. To ensure fairness of comparison, we used the observed weekly new deaths from CDC reports as the ground truth value.

In order to align our forecasting strategy with that of the CDC, we adjusted our approach by changing the moving window steps from one day per move to one week per move, prolonging the prediction length from 14 days to 28 days, and shifting the evaluation approach from horizon level to weekly level. At each week, the "4 weeks ahead" predictions were calculated from a model trained on historical information with a lag of 28 days, while the "1 week ahead" predictions were calculated from a model trained on historical information with a lag of 7 days. The model prediction performance was measured using the sMAPE values.

The comparison results, shown in Table 5.4.1, reveal that the SARIMAX model ranks high among the 18 models and performs close to the ensemble model. However, as observed in Figure 5.4, the SARIMAX

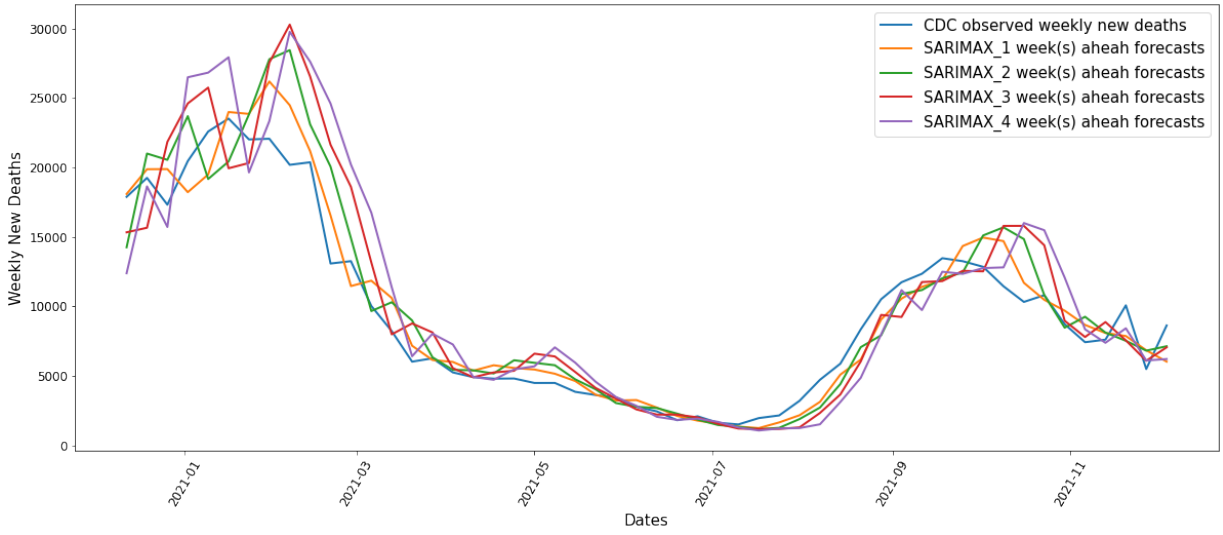


Figure 5.4: Comparison of SARIMAX model forecasts with observed weekly COVID-19 deaths in the United States.

model shows some areas for improvement. Specifically, it tends to overestimate the weekly deaths at truth value peaks, and this overestimation problem becomes more severe as the prediction length increases.

5.6 Conclusion and Future Work

This chapter aimed to evaluate and compare different statistical and machine learning models for forecasting daily deaths caused by COVID-19 in the United States. The study highlighted the importance of incorporating eXogenous variables in the forecasting models, specifically in the SARIMAX and MCP models. These models have the advantage of taking into account factors such as hospitalization rates, ICU occupancy, and vaccination data, which are not captured by the SARIMA model.

A comparison was made between our best model, the SARIMAX model, and 18 other models submitted to the United States Centers for Disease Control and Prevention (CDC) over a continuous 52-week period from December 2020 to December 2021. The SARIMAX model performed well, ranking high among the 18 models and performing similarly to the multi-model ensemble forecast.

However, a visualization of the SARIMAX model predictions at each of the four forecast levels (Figure 5.4) revealed areas for improvement. The model tended to overestimate weekly deaths at peaks in the truth values, with the overestimation becoming more pronounced as the prediction length increased. In future work, we aim to enhance the performance of our model.

Chapter 6

Conclusion

In this dissertation, we addressed three different problems in medical image classification and mortality prediction using deep learning, tensor factor modeling, and statistical and machine learning techniques.

We proposed a transfer learning-based approach to tackle the problem of small sample sizes and imbalanced medical image classification. Our approach is a privacy-preserving and communication-efficient procedure that can be applied to a variety of medical image classification tasks and has the potential to improve clinical diagnosis. Our study shows that the proposed method can improve the pneumonia classification accuracy on a small but heavily imbalanced mchest X-ray image dataset by 11.53% which performs even better than directly augmenting that source data into the training process.

We introduced an augmented tensor factor model to handle overlapping image classification. Our model decomposes tensor data into a pervasive component and a specific component. The pervasive component is characterized by the production of a low-rank latent tensor factor and factor loading matrices. We proposed a matricization plus principal component analysis strategy to estimate this tensor factor model and developed a factor adjustment procedure to remove pervasive noise and improve signal-to-noise ratio. We demonstrated the effectiveness of our method through synthetic experiments and COVID-19 pneumonia diagnosis from chest X-ray images.

In addition to the medical image classification, we focused on the problem of COVID-19 mortality prediction using statistical and machine learning models. We evaluated several models, such as SARIMA,

SARIMAX, VAR, and MCP. We showed that the SARIMAX model generated forecast values using eXogenous hospital information variables that could enrich the underlying model and improve prediction accuracy. Our best machine learning model performed better than most of the CDC models based on the four weeks ahead prediction.

Overall, our proposed methods have demonstrated promising results and have the potential to improve medical data analysis, diagnosis, and treatment. We believe that the proposed methods can be extended to various medical applications, leading to better patient care and outcomes.

Bibliography

Abbott, S., Hellewell, J., Thompson, R. N., Sherratt, K., Gibbs, H. P., Bosse, N. I., Munday, J. D., Meakin, S., Doughty, E. L., Chun, J. Y., et al. (2020). Estimating the time-varying reproduction number of sars-cov-2 using national and subnational case counts. *Wellcome Open Research*, *5*(112), 112.

Aceto, G., Persico, V., & Pescapé, A. (2020). Industry 4.0 and health: Internet of things, big data, and cloud computing for healthcare 4.0. *Journal of Industrial Information Integration*, *18*, 100129.

Ahmed, M., Chakraborty, P., & Choudhury, T. (2022). Bangla document categorization using deep rnn model with attention mechanism. *Cyber Intelligence and Information Retrieval: Proceedings of CIIR 2021*, 137–147.

Ahn, S. C., & Horenstein, A. R. (2013). Eigenvalue ratio test for the number of factors. *Econometrica*, *81*(3), 1203–1227.

Ala, A., & Chen, F. (2022). Appointment scheduling problem in complexity systems of the healthcare services: A comprehensive review. *Journal of Healthcare Engineering*, *2022*.

Albahli, S., Rauf, H. T., Arif, M., Nafis, M. T., & Algosaibi, A. (2021). Identification of thoracic diseases by exploiting deep neural networks. *neural networks*, *5*, 6.

Aleshin-Guendel, S., & Alvarez, S. (2017). Examining the structure of convolutional neural networks.

Alsunaidi, S. J., Almuhaideb, A. M., Ibrahim, N. M., Shaikh, F. S., Alqudaihi, K. S., Alhaidari, F. A., Khan, I. U., Aslam, N., & Alshahrani, M. S. (2021). Applications of big data analytics to control covid-19 pandemic. *Sensors*, *21*(7), 2282.

Arjovsky, M., & Bottou, L. (2017). Towards principled methods for training generative adversarial networks.

Arora, S., Ge, R., Liang, Y., Ma, T., & Zhang, Y. (2017). Generalization and equilibrium in generative adversarial nets (gans). *International Conference on Machine Learning*, 224–232.

Arunraj, N. S., Ahrens, D., & Fernandes, M. (2016). Application of sarimax model to forecast daily sales in food retail industry. *International Journal of Operations Research and Information Systems (IJORIS)*, *7*(2), 1–21.

Azizpour, H., Razavian, A. S., Sullivan, J., Maki, A., & Carlsson, S. (2015). Factors of transferability for a generic convnet representation. *IEEE transactions on pattern analysis and machine intelligence*, *38*(9), 1790–1802.

Bacciu, D., & Mandic, D. P. (2020). Tensor decompositions in deep learning. *arXiv preprint arXiv:2002.11835*.

Bai, J. (2003). Inferential theory for factor models of large dimensions. *Econometrica*, *71*(1), 135–171.

Bai, J., & Li, K. (2012). Statistical analysis of factor models of high dimension. *The Annals of Statistics*, *40*(1), 436–465.

Bai, J., & Ng, S. (2002). Determining the number of factors in approximate factor models. *Econometrica*, *70*(1), 191–221.

Baig, M. I., Shuib, L., & Yadegaridehkordi, E. (2019). Big data adoption: State of the art and research challenges. *Information Processing & Management*, *56*(6), 102095.

Bartoletti, I. (2019). Ai in healthcare: Ethical and privacy challenges. *Artificial Intelligence in Medicine: 17th Conference on Artificial Intelligence in Medicine, AIME 2019, Poznan, Poland, June 26–29, 2019, Proceedings 17*, 7–10.

Bhat, W. A. (2018). Bridging data-capacity gap in big data storage. *Future Generation Computer Systems*, *87*, 538–548.

Bouchard, G., Naradowsky, J., Riedel, S., Rocktäschel, T., & Vlachos, A. (2015). Matrix and tensor factorization methods for natural language processing. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing: Tutorial Abstracts*, 16–18.

Bro, R. (1997). Parafac. tutorial and applications. *Chemometrics and intelligent laboratory systems*, *38*(2), 149–171.

Buda, M., Maki, A., & Mazurowski, M. A. (2018). A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, *106*, 249–259.

Burton, L. C., Anderson, G. F., & Kues, I. W. (2004). Using electronic health records to help coordinate care. *The Milbank Quarterly*, *82*(3), 457–481.

Cai, T. T., & Wei, H. (2021). Transfer learning for nonparametric classification: Minimax rate and adaptive classifier. *The Annals of Statistics*, *49*(1), 100–128.

Çallı, E., Sogancioglu, E., van Ginneken, B., van Leeuwen, K. G., & Murphy, K. (2021). Deep learning for chest x-ray analysis: A survey. *Medical Image Analysis*, *72*, 102125.

Chamberlain, G., & Rothschild, M. (1982). Arbitrage, factor structure, and mean-variance analysis on large asset markets.

Chang, J., Guo, B., & Yao, Q. (2015). High dimensional stochastic regression with latent factors, endogeneity and nonlinearity. *Journal of Econometrics*, *189*(2), 297–312.

Chaudhari, Y., Javeri, I., Arpinar, I., Miller, J. A., Li, X., Li, B., Ke, Y., Toutiaee, M., & Lazar, N. (2021). Enhance covid-19 mortality prediction with human mobility trend and medical information. *2021 IEEE 23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, 1245–1252.

Chauhan, R., Ghanshala, K. K., & Joshi, R. (2018). Convolutional neural network (cnn) for image detection and recognition. *2018 first international conference on secure cyber computing and communication (ICSCCC)*, 278–282.

Chen, E. Y., & Fan, J. (2021). Statistical inference for high-dimensional matrix-variate factor models. *Journal of the American Statistical Association*, 1–18.

Chen, E. Y., Tsay, R. S., & Chen, R. (2019). Constrained factor models for high-dimensional matrix-variate time series. *Journal of the American Statistical Association*.

Chen, R., Yang, D., & Zhang, C.-H. (2022). Factor models for high-dimensional tensor time series. *Journal of the American Statistical Association*, *117*(537), 94–116.

Chen, Y., Schönlieb, C.-B., Lio, P., Leiner, T., Dragotti, P. L., Wang, G., Rueckert, D., Firmin, D., & Yang, G. (2022). Ai-based reconstruction for fast mri—a systematic review and meta-analysis. *Proceedings of the IEEE*, *110*(2), 224–245.

Choi, J., Gill, H., Ou, S., Song, Y., & Lee, J. (2018). Design of voice to text conversion and management program based on google cloud speech api. *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, 1452–1453.

Chowdhury, N. I., Smith, T. L., Chandra, R. K., & Turner, J. H. (2019). Automated classification of osteomeatal complex inflammation on computed tomography using convolutional neural networks. *International forum of allergy & rhinology*, *9*(1), 46–52.

Copeland, B. (2022). Artificial intelligence encyclopedia britannica; 2020. *Available from (Accessed May 26, 2020): https://www. britannica. com/technology/artificial-intelligence*.

COVID, I., Murray, C. J., et al. (2020). Forecasting covid-19 impact on hospital bed-days, icu-days, ventilator-days and deaths by us state in the next 4 months. *MedRxiv*.

Davenport, T., & Kalakota, R. (2019). The potential for artificial intelligence in healthcare. *Future healthcare journal*, *6*(2), 94.

De Giacomo, G., Lembo, D., Lenzerini, M., & Rosati, R. (2007). On reconciling data exchange, data integration, and peer data management. *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 133–142.

De Lathauwer, L., De Moor, B., & Vandewalle, J. (2000). A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, *21*(4), 1253–1278.

De Silva, V., & Lim, L.-H. (2008). Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM Journal on Matrix Analysis and Applications*, *30*(3), 1084–1127.

Deng, L., Hinton, G., & Kingsbury, B. (2013). New types of deep neural network learning for speech recognition and related applications: An overview. *2013 IEEE international conference on acoustics, speech and signal processing*, 8599–8603.

Deng, L., Yu, D., et al. (2014). Deep learning: Methods and applications. *Foundations and trends® in signal processing*, *7*(3–4), 197–387.

Dhieb, N., Ghazzai, H., Besbes, H., & Massoud, Y. (2020). A secure ai-driven architecture for automated insurance systems: Fraud detection and risk measurement. *IEEE Access*, *8*, 58546–58558.

Dong, E., Du, H., & Gardner, L. (2020). An interactive web-based dashboard to track covid-19 in real time. *Infectious Diseases*, *20*. https://doi.org/https://doi.org/10.1016/S1473-3099(20)30120-1

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

Drennan, V. M., & Ross, F. (2019). Global nurse shortages: The facts, the impact and action for change. *British medical bulletin*, *130*(1), 25–37.

Dubovitskaya, A., Novotny, P., Thiebes, S., Sunyaev, A., Schumacher, M., Xu, Z., & Wang, F. (2019). Intelligent health care data management using blockchain: Current limitation and future research agenda. *Heterogeneous Data Management, Polystores, and Analytics for Healthcare: VLDB 2019 Workshops, Poly and DMAH, Los Angeles, CA, USA, August 30, 2019, Revised Selected Papers 5*, 277–288.

Duggal, R., Brindle, I., & Bagenal, J. (2018). Digital healthcare: Regulating the revolution.

Fadhil, A. (2018). Beyond patient monitoring: Conversational agents role in telemedicine & healthcare support for home-living elderly individuals. *arXiv preprint arXiv:1803.06000*.

Fan, J., Fan, Y., & Lv, J. (2008). High dimensional covariance matrix estimation using a factor model. *Journal of Econometrics*, *147*(1), 186–197.

Fan, J., Liao, Y., & Mincheva, M. (2011). High dimensional covariance matrix estimation in approximate factor models. *Annals of statistics*, *39*(6), 3320.

Fan, J., & Lv, J. (2008). Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *70*(5), 849–911.

Fujita, H. (2020). Ai-based computer-aided diagnosis (ai-cad): The latest review to read first. *Radiological physics and technology*, *13*(1), 6–19.

Fujiyoshi, H., Hirakawa, T., & Yamashita, T. (2019). Deep learning-based image recognition for autonomous driving. *IATSS research*, *43*(4), 244–252.

Furtado, P. (2021). Epidemiology sir with regression, arima, and prophet in forecasting covid-19. *Engineering Proceedings*, *5*, 52.

Gao, J., Jiang, Q., Zhou, B., & Chen, D. (2019). Convolutional neural networks for computer-aided detection or diagnosis in medical image analysis: An overview. *Mathematical Biosciences and Engineering*, *16*(6), 6536–6561.

Gayed, J. M., Carlon, M. K. J., Oriola, A. M., & Cross, J. S. (2022). Exploring an ai-based writing assistant's impact on english language learners. *Computers and Education: Artificial Intelligence*, *3*, 100055.

Giger, M. L., & Suzuki, K. (2008). Computer-aided diagnosis. In *Biomedical information technology* (pp. 359–XXII). Elsevier.

Goldberg, Y. (2016). A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, *57*, 345–420.

Goldberg, Y. (2017). Neural network methods for natural language processing. *Synthesis lectures on human language technologies*, *10*(1), 1–309.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial networks.

Goyal, D. K., Mansab, F., Iqbal, A., & Bhatti, S. (2020). Early intervention likely improves mortality in covid-19 infection. *Clinical Medicine*, *20*(3), 248.

Gozes, O., Frid-Adar, M., Greenspan, H., Browning, P. D., Zhang, H., Ji, W., Bernheim, A., & Siegel, E. (2020). Rapid ai development cycle for the coronavirus (covid-19) pandemic: Initial results for

automated detection & patient monitoring using deep learning ct image analysis. *arXiv preprint arXiv:2003.05037*.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Gray, V. (2017). *Principal component analysis: Methods, applications, and technology*. Nova Science Publishers, Incorporated.

Gu, C., & Gu, C. (2013). *Smoothing spline anova models* (Vol. 297). Springer.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. (2017). Improved training of wasserstein gans.

Halalli, B., & Makandar, A. (2018). Computer aided diagnosis-medical image analysis techniques. *Breast Imaging*, *85*.

Hasan, K. M., Walimuni, I. S., Abid, H., & Hahn, K. R. (2011). A review of diffusion tensor magnetic resonance imaging computational methods and software tools. *Computers in biology and medicine*, *41*(12), 1062–1072.

Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (Vol. 2). Springer.

Hatvani, J., Basarab, A., Tourneret, J.-Y., Gyöngy, M., & Kouamé, D. (2018). A tensor factorization method for 3-d super resolution with application to dental ct. *IEEE transactions on medical imaging*, *38*(6), 1524–1531.

He, J., Sun, G., Zhang, Y., & Geng, T. (2016). Data recovery in heterogeneous wireless sensor networks based on low-rank tensors. *2016 IEEE symposium on computers and communication (ISCC)*, 616–620.

He, Y., Li, L., & Trapani, L. (2022). Statistical inference for large-dimensional tensor factor model by weighted/unweighted projection. *arXiv preprint arXiv:2206.09800*.

Hecht-Nielsen, R. (1992). Theory of the backpropagation neural network. In *Neural networks for perception* (pp. 65–93). Elsevier.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507.

Holmes, W., Porayska-Pomsta, K., Holstein, K., Sutherland, E., Baker, T., Shum, S. B., Santos, O. C., Rodrigo, M. T., Cukurova, M., Bittencourt, I. I., et al. (2021). Ethics of ai in education: Towards a community-wide framework. *International Journal of Artificial Intelligence in Education*, 1–23.

Hosny, K. M., Kassem, M. A., & Foaud, M. M. (2019). Classification of skin lesions using transfer learning and augmentation with alex-net. *PloS one*, *14*(5), e0217293.

Houssein, E. H., Emam, M. M., Ali, A. A., & Suganthan, P. N. (2021). Deep and machine learning techniques for medical imaging-based breast cancer: A comprehensive review. *Expert Systems with Applications*, *167*, 114161.

Hu, M., Wang, H., Wang, X., Yang, J., & Wang, R. (2019). Video facial emotion recognition based on local enhanced motion history image and cnn-ctslstm networks. *Journal of Visual Communication and Image Representation*, *59*, 176–185.

Huang, J., Zhang, Y., Zhang, J., & Zhang, X. (2018). A tensor-based sub-mode coordinate algorithm for stock prediction. *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, 716–721.

Hubel, D. H., & Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, *195*(1), 215–243.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift.

Islam, M. T., Siddique, B. N. K., Rahman, S., & Jabid, T. (2018). Image recognition with deep learning. *2018 International conference on intelligent informatics and biomedical sciences (ICIIBMS)*, *3*, 106–110.

Ismael, A. M., & Şengür, A. (2021). Deep learning approaches for covid-19 detection based on chest x-ray images. *Expert Systems with Applications*, *164*, 114054.

Jain, R., Gupta, M., Taneja, S., & Hemanth, D. J. (2021). Deep learning based detection and analysis of covid-19 on chest x-ray images. *Applied Intelligence*, *51*(3), 1690–1700.

Janiesch, C., Zschech, P., & Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, *31*(3), 685–695.

Jeddi, Z., & Bohr, A. (2020). Remote patient monitoring using artificial intelligence. In *Artificial intelligence in healthcare* (pp. 203–234). Elsevier.

Johnson, K. B., Wei, W.-Q., Weeraratne, D., Frisse, M. E., Misulis, K., Rhee, K., Zhao, J., & Snowdon, J. L. (2021). Precision medicine, ai, and the future of personalized health care. *Clinical and translational science*, *14*(1), 86–93.

Karlen, D. (2020). Characterizing the spread of covid-19. *arXiv preprint arXiv:2007.07156*.

Kasar, M. M., Bhattacharyya, D., & Kim, T. (2016). Face recognition using neural network: A review. *International Journal of Security and Its Applications*, *10*(3), 81–100.

Kermany, D. S., Zhang, K., & Goldbaum, M. H. (2018). Labeled optical coherence tomography (oct) and chest x-ray images for classification.

Kindle, R. D., Badawi, O., Celi, L. A., & Sturland, S. (2019). Intensive care unit telemedicine in the era of big data, artificial intelligence, and computer clinical decision support systems. *Critical care clinics*, *35*(3), 483–495.

Kingma, D. P., & Ba, J. (2017). Adam: A method for stochastic optimization.

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational bayes.

Kolda, T. G., & Bader, B. W. (2009). Tensor decompositions and applications. *SIAM review*, *51*(3), 455–500.

Kozyreva, A., Herzog, S., Lorenz-Spreen, P., Hertwig, R., & Lewandowsky, S. (2020). Artificial intelligence in online environments: Representative survey of public attitudes in germany.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012a). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012b). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 1097–1105.

Kumar, A., Gupta, P. K., & Srivastava, A. (2020). A review of modern technologies for tackling covid-19 pandemic. *Diabetes & Metabolic Syndrome: Clinical Research & Reviews*, *14*(4), 569–573.

Kumar, N., Srivastava, J. D., & Bisht, H. (2019). Artificial intelligence in insurance sector. *Journal of the Gujarat Research Society*, *21*(7), 79–91.

Kumar, T., Rajest, S. S., Villalba-Condori, K. O., Arias-Chavez, D., Rajesh, K., & Chakravarthi, M. K. (2022). An evaluation on speech recognition technology based on machine learning. *Webology*, *19*(1), 646–663.

Lakhani, P., Mongan, J., Singhal, C., Zhou, Q., Andriole, K. P., Auffermann, W. F., Prasanna, P., Pham, T., Peterson, M., Bergquist, P. J., et al. (2021). The 2021 siim-fisabio-rsna machine learning covid-19 challenge: Annotation and standard exam classification of covid-19 chest radiographs.

Lam, C., & Yao, Q. (2012). Factor modeling for high-dimensional time series: Inference for the number of factors. *The Annals of Statistics*, 694–726.

Lauritsen, S. M., Kristensen, M., Olsen, M. V., Larsen, M. S., Lauritsen, K. M., Jørgensen, M. J., Lange, J., & Thiesson, B. (2020). Explainable artificial intelligence model to predict acute critical illness from electronic health records. *Nature communications*, *11*(1), 3852.

Lawrence, S., Giles, C. L., Tsoi, A. C., & Back, A. D. (1997). Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, *8*(1), 98–113.

Le Glaz, A., Haralambous, Y., Kim-Dufor, D.-H., Lenca, P., Billot, R., Ryan, T. C., Marsh, J., Devylder, J., Walter, M., Berrouiguet, S., et al. (2021). Machine learning and natural language processing in mental health: Systematic review. *Journal of Medical Internet Research*, *23*(5), e15708.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Lee, K.-S., Jung, S.-K., Ryu, J.-J., Shin, S.-W., & Choi, J. (2020). Evaluation of transfer learning with deep convolutional neural networks for screening osteoporosis in dental panoramic radiographs. *Journal of clinical medicine*, *9*(2), 392.

Li, H., Zeng, N., Wu, P., & Clawson, K. (2022). Cov-net: A computer-aided diagnosis method for recognizing covid-19 from chest x-ray images via machine vision. *Expert Systems with Applications*, *207*, 118029.

Li, X., & Ke, Y. (2022). Privacy preserving and communication efficient information enhancement for imbalanced medical image classification. *Medical Image Understanding and Analysis: 26th Annual Conference, MIUA 2022, Cambridge, UK, July 27–29, 2022, Proceedings*, 663–679.

Li, Y. (2022). Research and application of deep learning in image recognition. *2022 IEEE 2nd International Conference on Power, Electronics and Computer Applications (ICPECA)*, 994–999.

Lin, W.-C., Chen, J. S., Chiang, M. F., & Hribar, M. R. (2020). Applications of artificial intelligence to electronic health record data in ophthalmology. *Translational vision science & technology*, *9*(2), 13–13.

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., Van Der Laak, J. A., Van Ginneken, B., & Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical image analysis*, *42*, 60–88.

Litmanovich, D. E., Chung, M., Kirkbride, R. R., Kicska, G., & Kanne, J. P. (2020). Review of chest radiograph findings of covid-19 pneumonia and suggested reporting language. *Journal of thoracic imaging*, *35*(6), 354–360.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3431–3440.

Long, M., Cao, Y., Wang, J., & Jordan, M. (2015). Learning transferable features with deep adaptation networks. *In Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML)*, *13*, 97–105.

Long, M., Zhu, H., Wang, J., & Jordan, M. I. (2016). Unsupervised domain adaptation with residual transfer networks. *Advances in Neural Information Processing Systems*, 136–144.

Louie, R., Engel, J., & Huang, C.-Z. A. (2022). Expressive communication: Evaluating developments in generative models and steering interfaces for music creation. *27th International Conference on Intelligent User Interfaces*, 405–417.

Lu, D., & Weng, Q. (2007). A survey of image classification methods and techniques for improving classification performance. *International journal of Remote sensing*, *28*(5), 823–870.

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*.

Mahendran, A., & Vedaldi, A. (2015). Understanding deep image representations by inverting them. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5188–5196.

Maity, S., Sun, Y., & Banerjee, M. (2020). Minimax optimal approaches to the label shift problem. *arXiv preprint arXiv:2003.10443*.

Maninis, K.-K., Pont-Tuset, J., Arbeláez, P., & Van Gool, L. (2016). Deep retinal image understanding. *International conference on medical image computing and computer-assisted intervention*, 140–148.

Mao, J., & Jain, A. K. (1995). Artificial neural networks for feature extraction and multivariate data projection. *IEEE transactions on neural networks*, *6*(2), 296–317.

Mayo, C. S., Matuszak, M. M., Schipper, M. J., Jolly, S., Hayman, J. A., & Ten Haken, R. K. (2017). Big data in designing clinical trials: Opportunities and challenges. *Frontiers in oncology*, *7*, 187.

Mbunge, E., Akinnuwesi, B., Fashoto, S. G., Metfula, A. S., & Mashwama, P. (2021). A critical review of emerging technologies for tackling covid-19 pandemic. *Human behavior and emerging technologies*, *3*(1), 25–39.

Miller, D. D., & Brown, E. W. (2018). Artificial intelligence in medical practice: The question to the answer? *The American journal of medicine*, *131*(2), 129–133.

Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets.

Mor, U., Cohen, Y., Valdés-Mas, R., Kviatcovsky, D., Elinav, E., & Avron, H. (2022). Dimensionality reduction of longitudinal'omics data using modern tensor factorizations. *PLOS Computational Biology*, *18*(7), e1010212.

Mou, L., Meng, Z., Yan, R., Li, G., Xu, Y., Zhang, L., & Jin, Z. (2016). How transferable are neural networks in nlp applications? *arXiv preprint arXiv:1603.06111*.

Mukhamadiyev, A., Khujayarov, I., Djuraev, O., & Cho, J. (2022). Automatic speech recognition method based on deep learning approaches for uzbek language. *Sensors*, *22*(10), 3683.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. *Proceedings of the 27th international conference on machine learning (ICML-10)*, 807–814.

Naranjo-Torres, J., Mora, M., Hernández-García, R., Barrientos, R. J., Fredes, C., & Valenzuela, A. (2020). A review of convolutional neural network applied to fruit image processing. *Applied Sciences*, *10*(10), 3443.

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

Nishimoto, Y., & Inoue, K. (2020). Curve-fitting approach for covid-19 data and its physical background. *medRxiv*.

Noor, M. B. T., Zenia, N. Z., Kaiser, M. S., Mamun, S. A., & Mahmud, M. (2020). Application of deep learning in detecting neurological disorders from magnetic resonance images: A survey on the detection of alzheimer's disease, parkinson's disease and schizophrenia. *Brain informatics*, *7*, 1–21.

Otter, D. W., Medina, J. R., & Kalita, J. K. (2020). A survey of the usages of deep learning for natural language processing. *IEEE transactions on neural networks and learning systems*, *32*(2), 604–624.

Pallathadka, H., Ramirez-Asis, E. H., Loli-Poma, T. P., Kaliyaperumal, K., Ventayen, R. J. M., & Naved, M. (2021). Applications of artificial intelligence in business management, e-commerce and finance. *Materials Today: Proceedings*.

Panagakis, Y., Kossaifi, J., Chrysos, G. G., Oldfield, J., Nicolaou, M. A., Anandkumar, A., & Zafeiriou, S. (2021). Tensor methods in computer vision and deep learning. *Proceedings of the IEEE*, *109*(5), 863–890.

Pieczynski, J., Kuklo, P., & Grzybowski, A. (2021). The role of telemedicine, in-home testing and artificial intelligence to alleviate an increasingly burdened healthcare system: Diabetic retinopathy. *Ophthalmology and therapy*, *10*(3), 445–464.

Qin, X., Bui, F. M., & Nguyen, H. H. (2019). Learning from an imbalanced and limited dataset and an application to medical imaging. *2019 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, 1–6.

Rabanser, S., Shchur, O., & Günnemann, S. (2017). Introduction to tensor decompositions and their applications in machine learning. *arXiv preprint arXiv:1711.10781*.

Radford, A., Metz, L., & Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks.

Raghupathi, W., & Raghupathi, V. (2014). Big data analytics in healthcare: Promise and potential. *Health information science and systems*, *2*, 1–10.

Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

Ran, B., Tan, H., Wu, Y., & Jin, P. J. (2016). Tensor based missing traffic data completion with spatial–temporal correlation. *Physica A: Statistical Mechanics and its Applications*, *446*, 54–63.

Ribeiro, J. M., Astudillo, P., de Backer, O., Budde, R., Nuis, R. J., Goudzwaard, J., Van Mieghem, N. M., Lumens, J., Mortier, P., Mattace-Raso, F., et al. (2022). Artificial intelligence and transcatheter interventions for structural heart disease: A glance at the (near) future. *Trends in cardiovascular medicine*, *32*(3), 153–159.

Ritchie, H., Mathieu, E., Rodés-Guirao, L., Appel, C., Giattino, C., Ortiz-Ospina, E., Joe Hasell, B. M., Beltekian, D., & Roser, M. (2020). Coronavirus pandemic (covid-19). *Our World in Data*. https://ourworldindata.org/coronavirus

Rocher, L., Hendrickx, J. M., & De Montjoye, Y.-A. (2019). Estimating the success of re-identifications in incomplete datasets using generative models. *Nature communications*, *10*(1), 1–9.

Rong, G., Mendez, A., Assi, E. B., Zhao, B., & Sawan, M. (2020). Artificial intelligence in healthcare: Review and prediction case studies. *Engineering*, *6*(3), 291–301.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical image computing and computer-assisted intervention*, 234–241.

Rother, J. (2017). Top of the administration's agenda: Stem the rising cost of healthcare. *Generations*, *40*(4), 30–37.

Rowe, J. W., Fulmer, T., & Fried, L. (2016). Preparing for better health and health care for an aging population. *Jama*, *316*(16), 1643–1644.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, *115*(3), 211–252. https://doi.org/10.1007/s11263-015-0816-y

Saikia, P., Baruah, R. D., Singh, S. K., & Chaudhuri, P. K. (2020). Artificial neural networks in the domain of reservoir characterization: A review from shallow to deep models. *Computers & Geosciences*, *135*, 104357.

Sarkar, K., Liu, L., Golyanik, V., & Theobalt, C. (2021). Humangan: A generative model of human images. *2021 International Conference on 3D Vision (3DV)*, 258–267.

Shaheen, M. Y. (2021). Applications of artificial intelligence (ai) in healthcare: A review. *ScienceOpen Preprints*.

Shahriar, S. (2022). Gan computers generate arts? a survey on visual arts, music, and literary text generation using generative adversarial network. *Displays*, 102237.

Sharma, H., Jain, J. S., Bansal, P., & Gupta, S. (2020). Feature extraction and classification of chest x-ray images using cnn to detect pneumonia. *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 227–231.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition.

Smith, M., & Miller, S. (2022). The ethical application of biometric facial recognition technology. *Ai & Society*, 1–9.

Specht, D. F., et al. (1991). A general regression neural network. *IEEE transactions on neural networks*, *2*(6), 568–576.

Srivastava, A., Xu, T., & Prasanna, V. K. (2020). Fast and accurate forecasting of covid-19 deaths using the sikjα model. *arXiv preprint arXiv:2007.05180*.

Stephen, O., Sain, M., Maduh, U. J., & Jeong, D.-U. (2019). An efficient deep learning approach to pneumonia classification in healthcare. *Journal of healthcare engineering*, *2019*.

Stock, J. H., & Watson, M. W. (2002). Forecasting using principal components from a large number of predictors. *Journal of the American statistical association*, *97*(460), 1167–1179.

Stoitsis, J., Valavanis, I., Mougiakakou, S. G., Golemati, S., Nikita, A., & Nikita, K. S. (2006). Computer aided diagnosis based on medical image processing and artificial intelligence methods. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *569*(2), 591–595.

Sun, Q., Qiu, H., Huang, M., & Yang, Y. (2020). Lower mortality of covid-19 by early recognition and intervention: Experience from jiangsu province. *Annals of intensive care*, *10*(1), 1–4.

Symeonidis, P. (2016). Matrix and tensor decomposition in recommender systems. *Proceedings of the 10th ACM conference on recommender systems*, 429–430.

Teng, F., Ma, Z., Chen, J., Xiao, M., & Huang, L. (2020). Automatic medical code assignment via deep learning approach for intelligent healthcare. *IEEE journal of biomedical and health informatics*, *24*(9), 2506–2515.

Tian, Y., & Feng, Y. (2021). Transfer learning under high-dimensional generalized linear models. *arXiv preprint arXiv:2105.14328*.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267–288.

Toutiaee, M., Li, X., Chaudhari, Y., Sivaraja, S., Venkataraj, A., Javeri, I., Ke, Y., Arpinar, I., Lazar, N., & Miller, J. (2021). Improving covid-19 forecasting using exogenous variables. *arXiv preprint arXiv:2107.10397*.

Tsai, E. B., Simpson, S., Lungren, M. P., Hershman, M., Roshkovan, L., Colak, E., Erickson, B. J., Shih, G., Stein, A., Kalpathy-Cramer, J., et al. (2021). The rsna international covid-19 open radiology database (ricord). *Radiology*, *299*(1), E204–E213.

Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, *31*(3), 279–311.

Vaishya, R., Javaid, M., Khan, I. H., & Haleem, A. (2020). Artificial intelligence (ai) applications for covid-19 pandemic. *Diabetes & Metabolic Syndrome: Clinical Research & Reviews*, *14*(4), 337–339.

Vayá, M. d. l. I., Saborit, J. M., Montell, J. A., Pertusa, A., Bustos, A., Cazorla, M., Galant, J., Barber, X., Orozco-Beltrán, D., Garcıa-Garcıa, F., et al. (2020). Bimcv covid-19+: A large annotated dataset of rx and ct images from covid-19 patients. *arXiv preprint arXiv:2006.01174*.

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

Wang, D., Liu, X., & Chen, R. (2019). Factor models for matrix-valued high-dimensional time series. *Journal of econometrics*, *208*(1), 231–248.

Wang, L., & Alexander, C. A. (2015). Big data in medical applications and health care. *American Medical Journal*, *6*(1), 1.

Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., & Summers, R. M. (2017). Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wang, Y., Kung, L., & Byrd, T. A. (2018). Big data analytics: Understanding its capabilities and potential benefits for healthcare organizations. *Technological forecasting and social change*, *126*, 3–13.

Watson, A. R. (2016). Impact of the digital age on transforming healthcare. *Healthcare Information Management Systems: Cases, Strategies, and Solutions*, 219–233.

White, T., Blok, E., & Calhoun, V. D. (2022). Data sharing and privacy issues in neuroimaging research: Opportunities, obstacles, challenges, and monsters under the bed. *Human Brain Mapping*, *43*(1), 278–291.

Who coronavirus (covid-19) dashboard. (2021). https://covid19.who.int/

Wu, L., Chen, Y., Shen, K., Guo, X., Gao, H., Li, S., Pei, J., Long, B., et al. (2023). Graph neural networks for natural language processing: A survey. *Foundations and Trends® in Machine Learning*, *16*(2), 119–328.

Wu, M., & Luo, J. (2019). Wearable technology applications in healthcare: A literature review. *Online J. Nurs. Inform*, *23*(3).

Wu, P.-Y., Cheng, C.-W., Kaddi, C. D., Venugopalan, J., Hoffman, R., & Wang, M. D. (2016). –omic and electronic health record big data analytics for precision medicine. *IEEE Transactions on Biomedical Engineering*, *64*(2), 263–273.

Xu, D., Shi, Y., Tsang, I. W., Ong, Y.-S., Gong, C., & Shen, X. (2019). Survey on multi-output learning. *IEEE transactions on neural networks and learning systems*, *31*(7), 2409–2429.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? *In Advances in neural information processing systems (NIPS)*, 3320–3328.

Yu, K.-H., Beam, A. L., & Kohane, I. S. (2018). Artificial intelligence in healthcare. *Nature biomedical engineering*, *2*(10), 719–731.

Zare, A., Ozdemir, A., Iwen, M. A., & Aviyente, S. (2018). Extension of pca to higher order data structures: An introduction to tensors, tensor decompositions, and tensor pca. *Proceedings of the IEEE*, *106*(8), 1341–1358.

Zhang, A., & Han, R. (2019). Optimal sparse singular value decomposition for high-dimensional high-order data. *Journal of the American Statistical Association*, *114*(528), 1708–1725.

Zhang, C., Fanaee-T, H., & Thoresen, M. (2021). Feature extraction from unequal length heterogeneous ehr time series via dynamic time warping and tensor decomposition. *Data Mining and Knowledge Discovery*, *35*(4), 1760–1784.

Zhang, C., Qiao, K., Wang, L., Tong, L., Hu, G., Zhang, R.-Y., & Yan, B. (2019). A visual encoding model based on deep neural networks and transfer learning for brain activity measured by functional magnetic resonance imaging. *Journal of neuroscience methods*, *325*, 108318.

Zhang, C.-H. (2010). Nearly unbiased variable selection under minimax concave penalty.

Zhang, Q., Berry, M. W., Lamb, B. T., & Samuel, T. (2009). A parallel nonnegative tensor factorization algorithm for mining global climate data. *International Conference on Computational Science*, 405–415.

Zhang, Z., Geiger, J., Pohjalainen, J., Mousa, A. E.-D., Jin, W., & Schuller, B. (2018). Deep learning for environmentally robust speech recognition: An overview of recent developments. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *9*(5), 1–28.

Zivot, E., & Wang, J. (2006). Vector autoregressive models for multivariate time series. *Modeling financial time series with S-PLUS®*, 385–429.

Appendix A

Proofs of the theoretical analysis

In this section, we provide assumptions, technical lemmas, and proofs of the theoretical results of Chapter 4.

A.1 Assumptions

Assumption 1. *Let c be a large enough positive absolute constant. For $i = 1, \ldots, n$, we make the following assumptions.*

(i) The observations in $\{\mathcal{X}_i\}_{i=1}^n$ are independent.

(ii) The latent tensor factors satisfy $\mathbb{E}(\mathcal{G}_i) = \mathbf{0}$ and $\mathbb{E}(\|\mathcal{G}_i\|_F^4) \leq c$.

(iii) The specific components satisfy $\mathbb{E}(\mathcal{S}_i) = \mathbf{0}$ and $\mathbb{E}(\|\mathcal{S}_i\|_F^8) \leq c$.

Assumption 2. *Let c be a large enough positive absolute constant. For $i = 1, \ldots, n$ and $d = 1, \ldots, D$, we assume $\{\mathcal{G}_i^{(d)}\}_{i=1}^n$ and $\{\mathcal{S}_i^{(d)}\}_{i=1}^n$ are two uncorrelated sequences and satisfy*

$$\max_{i,d} \left\| \mathbb{E}[\mathcal{G}_i^{(d)} \boldsymbol{U}_{-d}^{\mathrm{T}} \mathcal{S}_i^{(d)\mathrm{T}}] \right\|_F^2 \leq c p_\pi \text{ and } \max_{i,d} \left\| \mathbb{E}[\mathcal{S}_i^{(d)} \mathcal{S}_i^{(d)\mathrm{T}}] \right\|_F^2 \leq c p_\pi^2 / p_d.$$

Assumption 3. *For $d = 1, \ldots, D$, we assume*

(i) The mode-d latent factor $\mathcal{G}_i^{(d)}$ satisfies

$$\frac{1}{n}\sum_{i=1}^{n}\mathcal{G}_i^{(d)}\mathcal{G}_i^{(d)\mathrm{T}} \to^p \mathbf{\Omega}_d, \quad \text{as } n \to \infty,$$

where "$\to^p$" stands for converging in probability and $\mathbf{\Omega}_d$ is a positive definite matrix with all positive and distinct eigenvalues.

(ii) The factor loading matrices satisfy $p_d^{-1}\boldsymbol{U}_d\boldsymbol{U}_d^{\mathrm{T}} = \boldsymbol{I}_{r_d}$.

Assumption 1 (i) states that the observations are independent, which is typically satisfied in medical image classification applications. However, if necessary, this assumption can be relaxed to allow for strong mixing among observations. Our results should still hold without significant modification. Assumption 1 (ii) and (iii) are common center and moment conditions in most factor analysis literature. Assumption 2 states that for each mode, the dependence between latent factors and specific components is weak and the population covariance matrix of specific components is bounded. Both are mild and reasonable conditions that have been discussed in previous literature (e.g. Bai, 2003; E. Y. Chen and Fan, 2021; Fan et al., 2008). Assumption 3 reiterates the identification condition on metricized tensor factor model. These assumptions can be easily satisfied by properly choosing a series of orthogonal matrices to rotate the latent factors and factor loading matrices. This is a natural generalization of the identification conditions in classical factor models.

A.2 Technical Lemmas

In this section, we provides a two technical lemmas to facilitate the proofs of our main theoretical results.

Lemma 1. *Suppose the assumptions in Theorem 2.1 holds, we have*

$$\left\| \frac{1}{np_\pi} \sum_{i=1}^{n} \boldsymbol{U}_d \mathcal{G}_i^{(d)} \boldsymbol{U}_{-d}^{\mathrm{T}} \mathcal{S}_i^{(d)\mathrm{T}} \right\|_F^2 = O_p(\frac{p_d}{np_\pi}), \quad \textit{for } d = 1, \ldots, D.$$

Proof. Denote $\mathcal{G}_i^{(d)} \boldsymbol{U}_{-d}^{\mathrm{T}} \mathcal{S}_i^{(d)\mathrm{T}} := \mathcal{Z}_i^{(d)}$, we have

$$\left\| \frac{1}{np_\pi} \sum_{i=1}^{n} \boldsymbol{U}_d \mathcal{G}_i^{(d)} \boldsymbol{U}_{-d}^{\mathrm{T}} \mathcal{S}_i^{(d)\mathrm{T}} \right\|_F^2 \leq \frac{1}{n^2 p_\pi^2} \left\| \boldsymbol{U}_d \right\|_F^2 \left\| \sum_{i=1}^{n} \mathcal{Z}_i^{(d)} \right\|_F^2 = \frac{r_d p_d}{n^2 p_\pi^2} \left\| \sum_{i=1}^{n} \mathcal{Z}_i^{(d)} \right\|_F^2,$$

where the last equation uses the identification condition $p_d^{-1} \boldsymbol{U}_d^{\mathrm{T}} \boldsymbol{U}_d = \boldsymbol{I}_{r_d}$.

Next, we have

$$\begin{aligned} \left\| \sum_{i=1}^{n} \mathcal{Z}_i^{(d)} \right\|_F^2 &= \left\| \sum_{i=1}^{n} (\mathcal{Z}_i^{(d)} - \mathbb{E}\mathcal{Z}_i^{(d)} + \mathbb{E}\mathcal{Z}_i^{(d)}) \right\|_F^2 \\ &\leq \left\| \sum_{i=1}^{n} (\mathcal{Z}_i^{(d)} - \mathbb{E}\mathcal{Z}_i^{(d)}) \right\|_F^2 + \sum_{i=1}^{n} \left\| \mathbb{E}\mathcal{Z}_i^{(d)} \right\|_F^2. \end{aligned} \tag{A.1}$$

According to Assumption 1, we know $\{\mathcal{Z}_i^{(d)} - \mathbb{E}\mathcal{Z}_i^{(d)}\}_{i=1}^n$ is a sequence of independent random variables with zero mean and bounded variance. Then, according to the central limit theorem, we have

$$\left\| \sum_{i=1}^{n} (\mathcal{Z}_i^{(d)} - \mathbb{E}\mathcal{Z}_i^{(d)}) \right\|_F^2 = O_p(np_\pi).$$

Also, according to Assumption 2, we have

$$\max_i \left\| \mathbb{E}\mathcal{Z}_i^{(d)} \right\|_F^2 = O(p_\pi).$$

Therefore, we can complete the proof by showing

$$\left\| \frac{1}{np_\pi} \sum_{i=1}^{n} \boldsymbol{U}_d \mathcal{G}_i^{(d)} \boldsymbol{U}_{-d}^{\mathrm{T}} \mathcal{S}_i^{(d)\mathrm{T}} \right\|_F^2 = \frac{r_d p_d}{n^2 p^2} \left\{ O_p(np_\pi) + O(np_\pi) \right\} = O_p(\frac{p_d}{np_\pi}).$$

□

Lemma 2. *Suppose the assumptions in Theorem 2.1 holds, we have*

$$\left\| \frac{1}{np_\pi} \sum_{i=1}^{n} \mathcal{S}_i^{(d)} \mathcal{S}_i^{(d)\mathrm{T}} \right\|_F^2 = O_p(\frac{p_d}{np_\pi} + \frac{1}{p_d}), \quad \textit{for } d = 1, \ldots, D.$$

Proof. Similar to the decomposition in (A.1), we have

$$\begin{aligned} \left\| \sum_{i=1}^{n} \mathcal{S}_i^{(d)} \mathcal{S}_i^{(d)\mathrm{T}} \right\| &= \left\| \sum_{i=1}^{n} \left(\mathcal{S}_i^{(d)} \mathcal{S}_i^{(d)\mathrm{T}} - \mathbb{E}[\mathcal{S}_i^{(d)} \mathcal{S}_i^{(d)\mathrm{T}}] + \mathbb{E}[\mathcal{S}_i^{(d)} \mathcal{S}_i^{(d)\mathrm{T}}] \right) \right\|_F^2 \\ &\leq \left\| \sum_{i=1}^{n} \left(\mathcal{S}_i^{(d)} \mathcal{S}_i^{(d)\mathrm{T}} - \mathbb{E}[\mathcal{S}_i^{(d)} \mathcal{S}_i^{(d)\mathrm{T}}] \right) \right\|_F^2 + \sum_{i=1}^{n} \left\| \mathbb{E}[\mathcal{S}_i^{(d)} \mathcal{S}_i^{(d)\mathrm{T}}] \right\|_F^2 \end{aligned}$$

According to Assumption 1, we know $\{\mathcal{S}_i^{(d)} \mathcal{S}_i^{(d)\mathrm{T}} - \mathbb{E}[\mathcal{S}_i^{(d)} \mathcal{S}_i^{(d)\mathrm{T}}]\}_{i=1}^n$ is a sequence of independent random variables with zero mean and bounded variance. Then, according to the central limit theorem, we have

$$\left\| \sum_{i=1}^{n} \left(\mathcal{S}_i^{(d)} \mathcal{S}_i^{(d)\mathrm{T}} - \mathbb{E}[\mathcal{S}_i^{(d)} \mathcal{S}_i^{(d)\mathrm{T}}] \right) \right\|_F^2 = O_p(np_d p_\pi).$$

Also, according to Assumption 2, we have

$$\max_i \left\| \mathbb{E}[\mathcal{S}_i^{(d)} \mathcal{S}_i^{(d)\mathrm{T}}] \right\|_F^2 = O(\frac{p_\pi^2}{p_d}).$$

Thus, we can compelte the proof by showing

$$\left\| \sum_{i=1}^{n} \mathcal{S}_i^{(d)} \mathcal{S}_i^{(d)\mathrm{T}} \right\|_F^2 = O_p(np_d p_\pi + \frac{n^2 p_\pi^2}{p_d}).$$

□

A.3 Proof of Theorem 4.1

Proof. Let $d \in \{1, \ldots, D\}$ be any mode index. Recall that, the mode-d sample variance-covariance matrix is defined as

$$\begin{aligned}
\boldsymbol{\Sigma}_d =& \frac{1}{np_\pi} \sum_{i=1}^{n} \mathcal{X}_i^{(d)} \mathcal{X}_i^{(d)\mathrm{T}} \\
=& \frac{1}{np_\pi} \sum_{i=1}^{n} (\boldsymbol{U}_d \mathcal{G}_i^{(d)} \boldsymbol{U}_{-d}^{\mathrm{T}} + \mathcal{S}_i^{(d)})(\boldsymbol{U}_d \mathcal{G}_i^{(d)} \boldsymbol{U}_{-d}^{\mathrm{T}} + \mathcal{S}_i^{(d)})^{\mathrm{T}} \\
=& \frac{1}{np_d} \sum_{i=1}^{n} \boldsymbol{U}_d \mathcal{G}_i^{(d)} \mathcal{G}_i^{(d)\mathrm{T}} \boldsymbol{U}_d^{\mathrm{T}} + \frac{1}{np_\pi} \sum_{i=1}^{n} \boldsymbol{U}_d \mathcal{G}_i^{(d)} \boldsymbol{U}_{-d}^{\mathrm{T}} \mathcal{S}_i^{(d)\mathrm{T}} \\
& + \frac{1}{np_\pi} \sum_{i=1}^{n} \mathcal{S}_i^{(d)} \boldsymbol{U}_{-d} \mathcal{G}_i^{(d)\mathrm{T}} \boldsymbol{U}_d^{\mathrm{T}} + \frac{1}{np_\pi} \sum_{i=1}^{n} \mathcal{S}_i^{(d)} \mathcal{S}_i^{(d)\mathrm{T}} \\
:=& \Delta_1 + \Delta_2 + \Delta_3 + \Delta_4, \qquad \text{(A.2)}
\end{aligned}$$

where the first term in the second last equation uses the fact that $\boldsymbol{U}_{-d}$ is an orthogonal matrix and the identification condition $p_{-d}^{-1} \boldsymbol{U}_{-d}^{\mathrm{T}} \boldsymbol{U}_{-d} = \boldsymbol{I}$.

By applying an eigen-decomposition on $\boldsymbol{\Sigma}_d$, we denote $\{\lambda_1, \ldots, \lambda_{p_d}\}$ are the distinct eigenvalues of $\boldsymbol{\Sigma}_d$ sorted in the descending order and $\{\boldsymbol{v}_1^{(d)}, \ldots, \boldsymbol{v}_{p_d}^{(d)}\}$ are their corresponding eigenvectors. For a given rank r_d, the estimator of the model-d loading matrix $\boldsymbol{U}_d$ is defined as

$$\widehat{\boldsymbol{U}}_d := \widehat{\boldsymbol{U}}_d(r_d) = \sqrt{p_d} \left(\boldsymbol{v}_1^{(d)}, \ldots, \boldsymbol{v}_{r_d}^{(d)} \right).$$

Then, it is suffice to show that

$$\|\widehat{\boldsymbol{U}}_d - \boldsymbol{U}_d \boldsymbol{H}_d\|_F^2 = O_p(\frac{1}{p_d} + \frac{p_d^2}{np_\pi}),$$

where $\boldsymbol{H}_d = \frac{1}{np_d} \sum_{i=1}^{n} \mathcal{G}_i^{(d)} \mathcal{G}_i^{(d)\mathrm{T}} \boldsymbol{U}_d^{\mathrm{T}} \widehat{\boldsymbol{U}}_d \boldsymbol{\Lambda}_d^{-1}$ and $\boldsymbol{\Lambda}_d = (\lambda_1 \ \ldots, \ \lambda_{r_d})$.

Using the property of eigenvectors and the decomposition in (A.2), we can show that

$$\begin{aligned}\widehat{\boldsymbol{U}}_d - \boldsymbol{U}_d\boldsymbol{H}_d =& \boldsymbol{\Sigma}_d\widehat{\boldsymbol{U}}_d\boldsymbol{\Lambda}_d^{-1} - \frac{1}{np_d}\sum_{i=1}^{n}\boldsymbol{U}_d\mathcal{G}_i^{(d)}\mathcal{G}_i^{(d)\mathrm{T}}\boldsymbol{U}_d^{\mathrm{T}}\widehat{\boldsymbol{U}}_d\boldsymbol{\Lambda}_d^{-1} \\ =& \left(\Delta_1 - \frac{1}{np_d}\sum_{i=1}^{n}\boldsymbol{U}_d\mathcal{G}_i^{(d)}\mathcal{G}_i^{(d)\mathrm{T}}\boldsymbol{U}_d^{\mathrm{T}} + \Delta_2 + \Delta_3 + \Delta_4\right)\widehat{\boldsymbol{U}}_d\boldsymbol{\Lambda}_d^{-1} \\ =& \Delta_2\widehat{\boldsymbol{U}}_d\boldsymbol{\Lambda}_d^{-1} + \Delta_3\widehat{\boldsymbol{U}}_d\boldsymbol{\Lambda}_d^{-1} + \Delta_4\widehat{\boldsymbol{U}}_d\boldsymbol{\Lambda}_d^{-1}, \end{aligned} \tag{A.3}$$

where the last equation uses the fact that $\Delta_1 = \frac{1}{np_d}\sum_{i=1}^{n}\boldsymbol{U}_d\mathcal{G}_i^{(d)}\mathcal{G}_i^{(d)\mathrm{T}}\boldsymbol{U}_d^{\mathrm{T}}$.

Next, we aim to bound the Frobenius norms of the three terms in the last line of (A.3). To begin with, we have

$$\begin{aligned}\left\|\Delta_2\widehat{\boldsymbol{U}}_d\boldsymbol{\Lambda}_d^{-1}\right\|_F^2 =& \left\|\Delta_2\left(\widehat{\boldsymbol{U}}_d - \boldsymbol{U}_d\boldsymbol{H}_d + \boldsymbol{U}_d\boldsymbol{H}_d\right)\boldsymbol{\Lambda}_d^{-1}\right\|_F^2 \\ \leq& \|\Delta_2\|_F^2\left\|\boldsymbol{U}_d\right\|_F^2\left\|\boldsymbol{H}_d\right\|_F^2\left\|\boldsymbol{\Lambda}_d^{-1}\right\|_F^2 + \|\Delta_2\|_F^2\left\|\boldsymbol{\Lambda}_d^{-1}\right\|_F^2\left\|\widehat{\boldsymbol{U}}_d - \boldsymbol{U}_d\boldsymbol{H}_d\right\|_F^2 \\ =& O_p(\frac{p_d^2}{np_\pi}) + O_p(\frac{p_d}{np_\pi})\left\|\widehat{\boldsymbol{U}}_d - \boldsymbol{U}_d\boldsymbol{H}_d\right\|_F^2, \end{aligned} \tag{A.4}$$

where the last equation follows Lemma 1, the identification condition, and r_d is fixed.

Similarly, we can show that the second term satisfies

$$\begin{aligned}\left\|\Delta_3\widehat{\boldsymbol{U}}_d\boldsymbol{\Lambda}_d^{-1}\right\| =& \left\|\Delta_3\left(\widehat{\boldsymbol{U}}_d - \boldsymbol{U}_d\boldsymbol{H}_d + \boldsymbol{U}_d\boldsymbol{H}_d\right)\boldsymbol{\Lambda}_d^{-1}\right\|_F^2 \\ =& O_p(\frac{p_d^2}{np_\pi}) + O_p(\frac{p_d}{np_\pi})\left\|\widehat{\boldsymbol{U}}_d - \boldsymbol{U}_d\boldsymbol{H}_d\right\|_F^2. \end{aligned} \tag{A.5}$$

For the third term, we can utilize the results in Lemma 2 and show that

$$\begin{aligned}\left\|\Delta_4\widehat{\boldsymbol{U}}_d\boldsymbol{\Lambda}_d^{-1}\right\| =& \left\|\Delta_4\left(\widehat{\boldsymbol{U}}_d - \boldsymbol{U}_d\boldsymbol{H}_d + \boldsymbol{U}_d\boldsymbol{H}_d\right)\boldsymbol{\Lambda}_d^{-1}\right\|_F^2 \\ \leq& \|\Delta_4\|_F^2\left\|\boldsymbol{U}_d\right\|_F^2\left\|\boldsymbol{H}_d\right\|_F^2\left\|\boldsymbol{\Lambda}_d^{-1}\right\|_F^2 + \|\Delta_4\|_F^2\left\|\boldsymbol{\Lambda}_d^{-1}\right\|_F^2\left\|\widehat{\boldsymbol{U}}_d - \boldsymbol{U}_d\boldsymbol{H}_d\right\|_F^2 \\ =& O_p(\frac{p_d^2}{np_\pi} + 1) + O_p(\frac{p_d}{np_\pi} + \frac{1}{p_d})\left\|\widehat{\boldsymbol{U}}_d - \boldsymbol{U}_d\boldsymbol{H}_d\right\|_F^2. \end{aligned} \tag{A.6}$$

By plugging the results in (A.4) –(A.6) back to (A.3), we complete the proof of Theorem 2.1.

□

A.4 Proof of Theorem 4.2

Proof. We can rewrite $\widehat{\mathcal{G}}_i$ as

$$
\begin{aligned}
\widehat{\mathcal{G}}_i =& \frac{1}{p_\pi}\mathcal{X}_i \times_1 \widehat{\boldsymbol{U}}_1^{\mathrm{T}} \times_2 \cdots \times_D \widehat{\boldsymbol{U}}_D^{\mathrm{T}} \\
=& \frac{1}{p_\pi}\mathcal{G}_i \times_1 \widehat{\boldsymbol{U}}_1^{\mathrm{T}}\boldsymbol{U}_1 \times_2 \cdots \times_D \widehat{\boldsymbol{U}}_D^{\mathrm{T}}\boldsymbol{U}_D + \frac{1}{p_\pi}\mathcal{S}_i \times_1 \widehat{\boldsymbol{U}}_1^{\mathrm{T}} \times_2 \cdots \times_D \widehat{\boldsymbol{U}}_D^{\mathrm{T}} \\
=& \frac{1}{p_\pi}\mathcal{G}_i \times_1 \widehat{\boldsymbol{U}}_1^{\mathrm{T}}(\boldsymbol{U}_1 - \widehat{\boldsymbol{U}}_1\boldsymbol{H}_1^{-1} + \widehat{\boldsymbol{U}}_1\boldsymbol{H}_1^{-1}) \times_2 \cdots \times_D \widehat{\boldsymbol{U}}_D^{\mathrm{T}}(\boldsymbol{U}_D - \widehat{\boldsymbol{U}}_D\boldsymbol{H}_D^{-1} + \widehat{\boldsymbol{U}}_D\boldsymbol{H}_D^{-1}) \\
&+ \frac{1}{p_\pi}\mathcal{S}_i \times_1 (\widehat{\boldsymbol{U}}_1 - \boldsymbol{U}_1\boldsymbol{H}_1 + \boldsymbol{U}_1\boldsymbol{H}_1)^{\mathrm{T}} \times_2 \cdots \times_D (\widehat{\boldsymbol{U}}_D - \boldsymbol{U}_d\boldsymbol{H}_d + \boldsymbol{U}_d\boldsymbol{H}_d)^{\mathrm{T}} \\
=& \mathcal{G}_i \times_1 \boldsymbol{H}_1^{-1} \times_2 \cdots \times_D \boldsymbol{H}_D^{-1} + \underbrace{\frac{1}{p_\pi}\mathcal{S}_i \times_1 \boldsymbol{H}_1^{\mathrm{T}}\boldsymbol{U}_1^{\mathrm{T}} \times_2 \cdots \times_D \boldsymbol{H}_D^{\mathrm{T}}\boldsymbol{U}_D^{\mathrm{T}}}_{\Delta_5} \\
&+ \underbrace{\sum_{d=1}^{D}\frac{1}{p_d}\mathcal{G}_i \times_1 \boldsymbol{H}_1^{-1} \times_2 \cdots \times_d \widehat{\boldsymbol{U}}_d^{\mathrm{T}}(\boldsymbol{U}_d - \widehat{\boldsymbol{U}}_d\boldsymbol{H}_d^{-1}) \times_{d+1} \cdots \times_D \boldsymbol{H}_d^{-1}}_{\Delta_6} \\
&+ \underbrace{\frac{1}{p_\pi}\sum_{d=1}^{D}\mathcal{S}_i \times_1 \boldsymbol{H}_1^{\mathrm{T}}\boldsymbol{U}_1^{\mathrm{T}} \times_2 \cdots \times_d (\widehat{\boldsymbol{U}}_d - \boldsymbol{U}_d\boldsymbol{H}_d)^{\mathrm{T}} \times_{d+1} \cdots \times_D \boldsymbol{H}_D^{\mathrm{T}}\boldsymbol{U}_D^{\mathrm{T}}}_{\Delta_7} \\
&+ \Delta_8,
\end{aligned}
\tag{A.7}
$$

where Δ_8 is the reminder that contains all higher-order interaction terms.

Denote $\hat{\eta}_{j_d,k_d}$ is the (k_d, j_d)-th entry of $\boldsymbol{H}_d^{\mathrm{T}}\boldsymbol{U}_d^T$ and $S_{i,j_1\cdots j_D}$ is the $(j_1,\ldots,j_D)$-th entry of $\mathcal{S}_i$. By the definition of Model-d product, we can bound $\|\Delta_5\|_F^2$ as

$$\begin{aligned}\|\Delta_5\|_F^2 =&\frac{1}{p_\pi^2}\sum_{k_1=1}^{r_1}\cdots\sum_{k_D=1}^{r_D}\sum_{j_1=1}^{p_1}\cdots\sum_{j_D=1}^{p_D}\left(\eta_{k_1,j_1}\cdots\eta_{k_D,j_D}S_{i,j_1\cdots j_D}\right)^2\\ =&O_p(\frac{1}{p_\pi}),\end{aligned} \tag{A.8}$$

where the last equation uses the facts that r_d is fixed and $\hat{\eta}_{j_d,k_d}$ converges to a fixed number when $n, p_d \to \infty$.

Next, we can bound the $\|\Delta_7\|_F^2$ as

$$\begin{aligned}\|\Delta_7\|_F^2 \leq&\frac{1}{p_\pi^2}\sum_{d=1}^{D}\left\|\mathcal{S}_i\times_1\boldsymbol{H}_1^{-1}\times_2\cdots\times_d(\widehat{\boldsymbol{U}}_d-\boldsymbol{U}_d\boldsymbol{H}_d)^{\mathrm{T}}\times_{d+1}\cdots\times_D\boldsymbol{H}_d^{-1}\right\|_F^2\\ \leq&\frac{1}{p_\pi^2}\sum_{k_1=1}^{r_1}\cdots\sum_{k_{d-1}=1}^{r_{d-1}}\sum_{k_{d+1}=1}^{r_{d+1}}\cdots\sum_{k_D=1}^{r_D}\sum_{j_1=1}^{p_1}\cdots\sum_{j_{d-1}=1}^{p_{d-1}}\sum_{j_{d+1}=1}^{p_{d+1}}\cdots\sum_{j_D=1}^{p_D}\\ &\left(\eta_{k_1,j_1}\cdots\eta_{k_{d-1},j_{d-1}}\eta_{k_{d+1},j_{d+1}}\eta_{k_D,j_D}S_{i,j_1\cdots j_D}\right)^2 k_d p_d\left\|\widehat{\boldsymbol{U}}_d-\boldsymbol{U}_d\boldsymbol{H}_d\right\|_F^2\\ =&\frac{1}{p_\pi}\cdot O_p(\frac{1}{p_d}+\frac{p_d^2}{np_\pi}) = O_p(\frac{1}{p_\pi p_d}+\frac{p_d^2}{np_\pi^2}),\end{aligned} \tag{A.9}$$

where the second inequality uses the fact that $\left\|\widehat{\boldsymbol{U}}_d-\boldsymbol{U}_d\boldsymbol{H}_d\right\|_{\max} \leq \left\|\widehat{\boldsymbol{U}}_d-\boldsymbol{U}_d\boldsymbol{H}_d\right\|_F$, and the last equality follows Theorem 2.1.

Before we move on to to bound $\|\Delta_6\|_F^2$, we first show that

$$\begin{aligned}\left\|\frac{1}{p_d}\widehat{\boldsymbol{U}}_d^{\mathrm{T}}(\boldsymbol{U}_d-\widehat{\boldsymbol{U}}_d\boldsymbol{H}_d^{-1})\right\|_F^2 =&\frac{1}{p_d^2}\left\|\widehat{\boldsymbol{U}}_d^{\mathrm{T}}(\boldsymbol{U}_d-\widehat{\boldsymbol{U}}_d\boldsymbol{H}_d^{-1})\boldsymbol{H}_d\boldsymbol{H}_d^{\mathrm{T}}\right\|_F^2\\ =&\frac{1}{p_d^2}\left\|\widehat{\boldsymbol{U}}_d^{\mathrm{T}}(\boldsymbol{U}_d\boldsymbol{H}_d-\widehat{\boldsymbol{U}}_d)\boldsymbol{H}_d^{\mathrm{T}}\right\|_F^2\\ \leq&\frac{1}{p_d^2}\left\|\widehat{\boldsymbol{U}}_d^{\mathrm{T}}\right\|_F^2\left\|\widehat{\boldsymbol{U}}_d-\boldsymbol{U}_d\boldsymbol{H}_d\right\|_F^2\|\boldsymbol{H}_d\|_F^2\\ =&O_p(\frac{1}{p_d}+\frac{p_d}{np_\pi}).\end{aligned}$$

Therefore, similar to (A.9), we can bound $\|\Delta_6\|_F^2$ as

$$\|\Delta_6\|_F^2 \lesssim \sum_{d=1}^{D} O_p(\frac{1}{p_d} + \frac{p_d}{np_\pi}) = O_p(\frac{1}{n} + \sum_{d=1}^{D} \frac{1}{p_d}). \tag{A.10}$$

To sum up, the results in (A.7) – (A.10) yield

$$\begin{aligned}\left\|\widehat{\mathcal{G}}_i - \mathcal{G}_i \times_1 \boldsymbol{H}_1^{-1} \times_2 \cdots \times_D \boldsymbol{H}_D^{-1}\right\|_F^2 \leq& \|\Delta_5\|_F^2 + \|\Delta_6\|_F^2 + \|\Delta_7\|_F^2 + \|\Delta_8\|_F^2 \\ \leq& O_p\left(\frac{1}{n} + \sum_{d=1}^{D} \frac{1}{p_d}\right).\end{aligned}$$

□

A.5 Proof of Theorem 4.3

Proof. Recall that, (A.2) in the Proof of Theorem 2.1 decomposes the mode-d sample variance covariance matrix $\boldsymbol{\Sigma}_d$ into four terms and we have proved the following results

$$\|\Delta_2\|_F^2 = O_p(\frac{p_d}{np_\pi}),\ \|\Delta_3\|_F^2 = O_p(\frac{p_d}{np_\pi}),\ \text{and}\ \|\Delta_3\|_F^2 = O_p(\frac{p_d}{np_\pi} + \frac{1}{p_d}).$$

Further, it is easy to notice that the eigenvalues of Δ_1 satisfies

$$\lambda_k(\Delta_1) = \begin{cases} \lambda_k(\frac{1}{n}\sum_{i=1}^n \mathcal{G}_i^{(d)}\mathcal{G}_i^{(d)\mathrm{T}}), & 1 \leq k \leq r_d, \\ 0, & r_d + 1 \leq k \leq p_d, \end{cases}$$

where $\lambda_{r_d}(\frac{1}{n}\sum_{i=1}^n \mathcal{G}_i^{(d)}\mathcal{G}_i^{(d)\mathrm{T}}) \geq c > 0$ according to Assumption 3.

The results above together with Weyl's theorem lead to

$$\lambda_k(\mathbf{\Sigma}_d) = \begin{cases} \lambda_k(\frac{1}{n}\sum_{i=1}^n \mathcal{G}_i^{(d)}\mathcal{G}_i^{(d)\mathrm{T}}) + o_p(1), & 1 \le k \le r_d, \\ o_p(1), & r_d + 1 \le k \le p_d. \end{cases}$$

Therefore, with $K_{\max} > r_d$ and a reasonable choice of C, the modified eigen-ration method introduced in (4.5) can consistently estimate r_d by detecting the gap between $\lambda_{r_d}(\mathbf{\Sigma}_d)$ and $\lambda_{r_d+1}(\mathbf{\Sigma}_d)$ with a probability approaching 1.

□