UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

CONTRIBUTIONS TO PATTERN RECOGNITION WITH MASSIVE UNLABELED DATA

TESIS

QUE PARA OPTAR POR EL GRADO DE
DOCTOR EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

PRESENTA

EDGAR DE JESÚS EK CHACÓN

TUTOR

DR. ERIK MOLINO MINERO RE
INSTITUTO DE INVESTIGACIONES EN MATEMÁTICAS APLICADAS Y EN SISTEMAS, UNAM.

MÉRIDA, YUCATÁN, MÉXICO. NOVIEMBRE, 2024.

Dedication

I dedicate this dissertation to my family, whose unwavering encouragement, emotional support, and loving atmosphere propelled me forward. To the church IPRM "Monte de los Olivos" in Tahdziú, Yucatán, I express my gratitude for the spiritual support, implicitly teaching me to trust in God's faithfulness every Sunday. Their assurance that all things will be fine, despite moments of trouble, has been a guiding light. To God, the Creator and King of Heaven and Earth, I acknowledge with deep appreciation for providing me with these two beautiful platforms upon which I could firmly stand and successfully finish this project.

Acknowledgements

This journey would not have been possible without the diligent support of my advisor, Dr. Erik Molino, who provided unconditional assistance, offering faithful guidance and invaluable patience through countless pieces of advice. My heartfelt gratitude extends to my committee, Dr. Paul and Dr. Antonio, whose recommendations and motivations helped me gauge whether I was on the right path semester by semester. Special thanks to Hector Angeles, who led us during the application stage of this project. Recognition is also due to the technical support staff of IIMAS-UNAM (LUCAR and Mérida). Finally, I express my appreciation to CONAHCyT for their crucial financial support.

Abstract

This study aims to investigate and propose methodologies addressing the challenge of analyzing vast amounts of unlabeled data, where the scarcity of labeled data hinders the development of efficient models across various domains, such as natural language processing, healthcare, and reservoir characterization. The integration of semi-supervised methods with deep neural networks has been shown to efficiently utilize unlabeled data, thereby enhancing the effectiveness of supervised learning with limited annotated examples. This technique, known as deep semi-supervised learning (DSSL), has significantly contributed to advancements in domains such as image recognition, text categorization, and speech analysis. In this study, we apply the DSSL framework to introduce two novel methodologies: self-training layer-wise for classification tasks and a semi-supervised approach for petrophysical estimation. The former leverages extensive unlabeled data from two perspectives to enhance the robustness of classification models with limited annotated training data, while the latter improves acoustic impedance data estimation using substantial amounts of unlabeled pre-stack seismic data. Our self-training layer-wise method demonstrated superior performance when evaluated on the large Quickdraw and medium datasets. On the other hand, the DSSL outperformed its supervised counterpart in predicting acoustic impedance at well locations and beyond. Therefore, strategically employing large quantities of unlabeled data proves beneficial in developing more robust machine learning models.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

2D	Two Dimensional
3D	Three Dimensional
ABC	Artificial Bee Colony Algorithm
AI	Acoustic Impedance
Bs	Batch size
CDP	Common Depth Points
CIT	Conditional Independence Test
CNN	Convolutional Neural Network
DI	Dependence Index
DNA	Deoxyribonucleic Acid
DNN	Deep Neural Network
DSSL	Deep Semi-Supervised Learning
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
GRU	Gated Recurrent Units
kNN	k-Nearest Neighbors
LAS	Log ASCII Standard Format

LEGW Linear-epoch Gradual-warmup

Lr Learning rate

LSTM Long Short-Term Memory

MCI Momentary Conditional Independence

micro-CT Micro-computed tomography

MLP Multilayer Perceptron

MNIST MNIST Corrupted

MNIST The Modified National Institute of Standards and Technology database

MSE Mean square error

NLP Natural Language Processing

OOM Out-of-memory errors

PCA Principal Component Analysis

PCMCI PC algorithm and Momentary Conditional Independence

PC Pearson correlation

PC algorithm Probability Distribution in an Undirected Graph C

PSTM Pre-stack Time Migration

ReLUs Rectifier Linear Units

ResNet Residual Networks

RNN Recurrent Neural Network

SC-LSTM-I Skip-Connected LSTM Identity

SCiForest Unsupervised isolation forest with split-selection criterion algorithm

SE-ResNet Squeeze-and-Excitation block and ResNet

SEG-Y The Society of Exploration Geophysicists Y Format

SeisInvNet Seismic Inversion Network

SLMNIST The Sign Language MNIST

SOM Self-organizing Map

SVM Support Vector Machine

TCN Temporal Convolutional Network

TCRFR Transductive conditional random field regression

TVDSS True Vertical Depth Sub-Sea

We Warmup epochs

Chapter 1

Introduction

In social media, e-commerce, and scientific research, vast amounts of text, images, and videos are generated, constituting unstructured data lacking a predetermined model for analysis. Unlike structured data, which can be organized in tables for straightforward analysis, unstructured data presents a more complex challenge. Nevertheless, the analysis of unstructured data holds significant value due to its potential for discovering valuable information [3]. For instance, Netflix, the world's largest video-streaming service with approximately 247 million users globally [96], utilizes tagged videos and user viewing behavior to predict customer preferences and the potential success of new content on the platform [82]. Analyzing large datasets has contributed to a customer retention rate of around 93%, with the recommendation system playing a pivotal role by suggesting approximately 80% of consumed content [3, 108].

Since the start of the digital age in 2002 [48], two key companies emerged near this timeline—Google in 1998 and Facebook in 2004. The adept handling of digital data contributed to their founding. Google provided a range of services by processing substantial volumes of data, while Facebook facilitated users in expressing their thoughts, resulting in the generation of extensive data logs. Recent advancements in information technology, cloud computing, and the Internet of Things have further accelerated the growth of digital data across various domains [24]. For instance, with approximately 2.2 billion users globally, Facebook generated 4 petabytes of data daily by 2018 [75]. In commerce and business, the data volume of global companies doubles every 1.2 years, exemplified by Amazon and Walmart processing millions of transactions daily. In scientific research, particularly in astronomy, the Vera C. Rubin Observatory is anticipated to capture 30 terabytes of universe images in a single day [79].

Managing extensive volumes of unstructured data presents both opportunities and challenges. The challenges include novel storage systems that enable rapid retrieval and scalability

and the requirement for parallel and distributed computing to process this data efficiently within constrained time frames. Additionally, the scarcity of categorized or labeled data inherently poses a substantial obstacle to machine learning and data analysis [74].

This study addresses the challenge of scarce labeled data in extensive volumes of unstructured or unlabeled data, a prevalent issue across diverse domains. In the healthcare sector, various forms of unstructured data accumulate, including medical imaging data, internet data from search engines, and patient-reported data [19]. Specifically, unstructured and ungrammatical text medical reports, such as radiology reports, pose challenges in categorization, requiring expertise from health professionals, which is both costly and time-consuming [46]. Timely patient follow-up relies on accurate document classification. In oil exploration, advancements in digital technology have enhanced seismic data acquisition techniques, resulting in the collection of substantial volumes of seismic data [8]. Seismic data consists of seismic traces, essentially unstructured time series or seismic signals. Geoscientists leverage this data to understand the Earth's subsurface for purposes like hydrocarbon reservoir detection [122]. However, manual identification of subsurface areas in large seismic volumes becomes impractical due to limited well-log information and the complexity of the Earth's interior. Additionally, the high cost of drilling wells makes obtaining known information (or labels) challenging [99].

The deep learning approach employs neural network architectures consisting of multiple layers with specialized neuron units. The lower layers focus on capturing less abstract features, while the higher layers can learn more abstract features from the input data [74]. A pivotal aspect of this approach lies in the automatic extraction and representation of meaningful patterns from the input data. In contrast to traditional machine learning methods [77], deep neural networks (DNNs) excel in capturing complex representations from vast amounts of unlabeled data. The utilization of copious unlabeled data has demonstrated enhanced performance of DNNs in real-world applications, including reservoir characterization [81], image classification [61], and natural language processing (NLP) [46]. Conversely, reliance on limited supervised data can impede model performance and generalization.

1.1 Problem Statement

This study addresses the challenge posed by the limited availability of annotated data within large datasets. The insufficiency of such data can negatively affect model performance and generalization capabilities. We aim to introduce mathematical and computational methodologies for pattern recognition within this context. The primary facets of the research include:

- Feature analysis.
- Explore and examine unsupervised algorithms for harnessing extensive unlabeled datasets.
- The investigation of classification and regression algorithms.

This can be achieved through the utilization of deep neural networks in the context of unsupervised learning. We aim to contribute a computational analysis technique to address this task, with a specific case study focused on reservoir characterization. This domain involves the utilization of vast volumes of unknown seismic data alongside limited quantities of well-known well-log data.

1.2 Hypothesis

Semi-supervised training, leveraging optimal utilization of extensive unlabeled data combined with a small amount of labeled data, can significantly enhance overall model performance.

1.3 General Objective

To contribute a pattern recognition technique for analyzing partially labeled large multivariate datasets.

1.4 Specific Objectives

- To investigate semi-supervised methods, assessing their suitability for feature extraction on unlabeled data.
- To assess diverse datasets to determine the optimal choice for testing the methods employed in the study.
- To select suitable technologies for managing extensive datasets, including programming languages and libraries.
- To introduce a novel methodology capable of pattern recognition on large partially labeled datasets.
- To disseminate our research findings through publication in reputable scientific journals.

1.5 Justification

One of the machine learning objectives is the development of learning systems capable of addressing specific tasks within various application domains. Notable achievements have been made in image and speech recognition, natural language processing (NLP), and drug discovery [59, 56, 114]. The effectiveness of deep learning approaches in these tasks is particularly evident when large sets of high-quality annotated data are utilized for training. However, acquiring such annotated data often involves substantial expert knowledge and is characterized by high costs, labor-intensive efforts, and time-consuming processes. For instance, accurately transcribing 1 hour of speech in speech recognition may require 400 hours of human expert annotation. Similarly, in protein 3D structure prediction, the association of a deoxyribonucleic acid (DNA) sequence with its corresponding 3D protein structure (label) can take months [40]. In reservoir characterization, obtaining labeled data from wells involves drilling, which incurs significant monetary costs, and additional analyses of well log data are related to a big-time investment.

Encountering partially labeled massive datasets is a common scenario in real-world problems. Developing effective learning systems capable of performing well in the presence of limited labeled data holds substantial implications across various domains, leading to significant savings in effort, time, and costs.

The semi-supervised approach in machine learning utilizes abundant unlabeled data to enhance model performance, particularly in scenarios with limited labeled data for training. Deep neural networks can uncover underlying features when employed in unsupervised learning, contributing to improved model generalization.

1.6 Contributions

To address the challenge posed by limited labeled data, we employ deep semi-supervised learning (DSSL), a category of semi-supervised methods built upon deep neural network (DNN) architectures. This approach is designed to capitalize on unlabeled data, mitigating the impact of insufficient labeled data. Within the scope of this study, we present two primary contributions to address this issue.

- We introduce a novel methodology for semi-supervised learning in classification tasks, termed self-training layer-wise. This approach maximizes the utilization of extensive unlabeled data to enhance the robustness of machine learning models. To assess the efficacy of the proposed method, we devised an experimental framework encompassing various scenarios with limited labeled datasets, employing three benchmark datasets.

The neural network architecture utilized in this study is based on a recurrent neural network with Long Short-Term Memory (LSTM) units.

- We introduce a practical application to mitigate the challenge posed by the scarcity of annotated datasets in reservoir characterization. Our proposed methodology [35] focuses on estimating petrophysical properties from seismic data. By adopting a semi-supervised approach and incorporating limited well-log information, we enhance petrophysical estimation by effectively utilizing extensive unlabeled seismic data.

1.7 Thesis Structure

This work is structured as follows: Chapter 1 introduces the context of massive volumes of data, and discusses its benefits and challenges. It also outlines the hypothesis, general and specific objectives, and justification. Moreover, this introductory section provides a brief overview of our contributions. Chapter 2 elucidates the theoretical framework and fundamental concepts that support our methodologies.

Chapter 3 reviews the state-of-the-art, examining recent works related to image classification using sequence-based methods, semi-supervised techniques, and the utilization of deep neural networks in reservoir characterization. Chapter 4 focuses on the methodology, detailing our proposed approach for classification tasks using recurrent neural networks and its application in the context of massive datasets. Moreover, we describe the procedures for data augmentation, which significantly impact model performance. We also introduce a deep semi-supervised approach for seismic data analysis.

Chapter 5 presents the results of experiments on image classification, our method's performance on medium and large datasets, and the outcomes of seismic data analysis using a semi-supervised method. The ensuing discussion of these results is found in Chapter 6. Finally, Chapter 7 contains the conclusions derived from the study.

Chapter 2

Theoretical Framework for Advanced Deep Learning: Models, Learning Paradigms, and Optimization Strategies

2.1 Recurrent Neural Networks

Sequential data is characterized by the inherent significance of the order in which samples or bits of information are arranged. Notably, this phenomenon is evident in log data generated by sensors (and servers), producing what is commonly referred to as time-series data. This data is characterized by a collection of organized data points over time, representing the temporal progression of dynamic processes [77]. Sequential data of this nature is frequently encountered in domains such as genomic characterization, natural language processing, and weather forecasting. Additionally, a specific machine learning method known as Recurrent Neural Networks (RNNs) is well-suited for effectively managing this type of data [76, 84, 42].

The recurrent neural network processes sequential features at time step t by receiving input x into the state h, which recurrently feeds back on itself, producing an output o. This process is illustrated on the left-hand side of Fig. 2.1. On the right-hand side, the temporal processing is depicted in detail, showing the progression from time step $t-1$, to time step t, and then to time step $t+1$. Notably, the state h from the previous time step is incorporated into the current data processing.

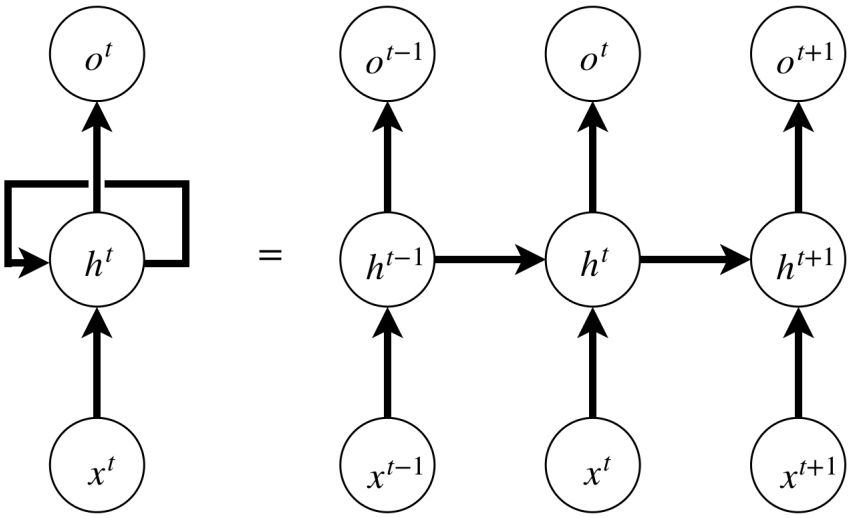


Fig. 2.1 Compact and unfolded representations of a recurrent neural network during data processing.

2.2 Recurrent Neural Network: Long Short-Term Memory

In addition to overcome the vanishing and exploding gradient problems [43], Long Short-Term Memory (LSTM) networks are capable of capturing long-range dependencies in sequential data over time. This ability makes LSTM a powerful model for applications involving sequential data [102].

Fig. 2.2 illustrates a unit of the LSTM architecture. The internal cell state memory c^t is responsible for capturing long-range dependencies over time. Additionally, the input gate i^t, output gate o^t, and forget gate f^t are crucial for reading and modifying this cell state.

To produce the output h^t, the cell state c^t is updated upon receiving the inputs x^t and h^{t-1} at time step t.

$$f^t = \sigma(W^{xf}x^t + W^{hf}h^{t-1} + W^{cf}c^{t-1} + b^f) \tag{2.1}$$

$$i^t = \sigma(W^{xi}x^t + W^{hi}h^{t-1} + W^{ci}c^{t-1} + b^i) \tag{2.2}$$

$$c^t = f^t c^{t-1} + i^t \tanh(W^{xc}x^t + W^{hc}h^{t-1}) \tag{2.3}$$

$$o^t = \sigma(W^{xo}x^t + W^{ho}h^{t-1} + W^{co}c^t + b^o) \tag{2.4}$$

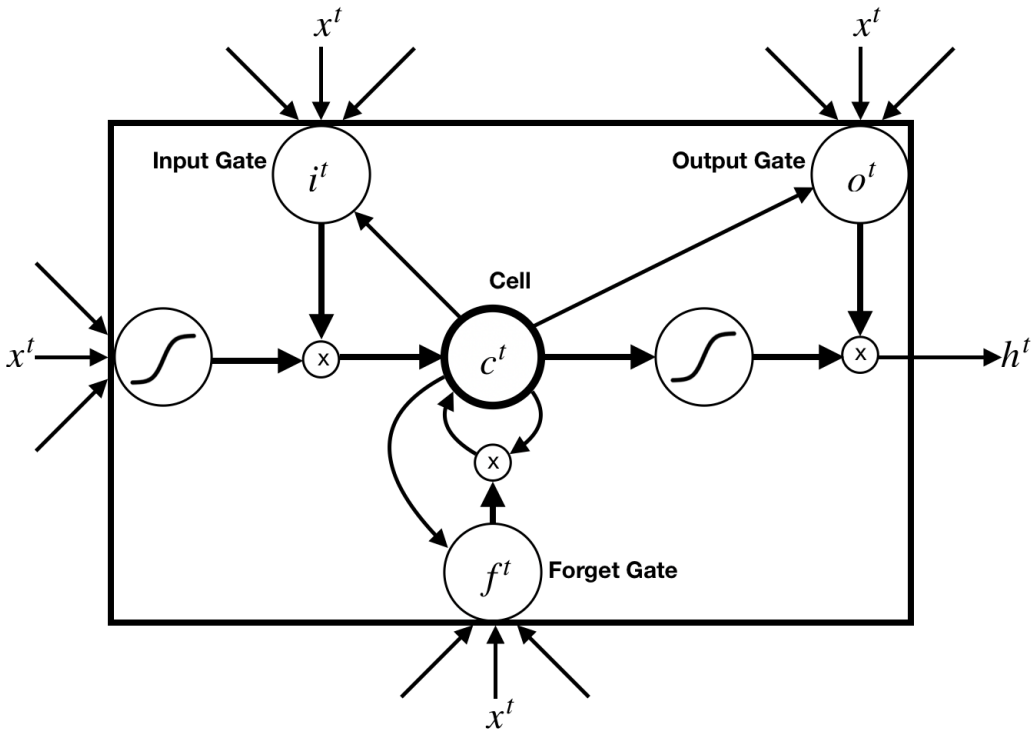


Fig. 2.2 LSTM unit.

$$h^t = o^t \tanh(c^t) \tag{2.5}$$

the LSTM architecture employs two types of weights: dense matrices and diagonal matrices ($W^{c\bullet}$). Additionally, the logistic sigmoid (σ) and hyperbolic tangent (tanh) functions serve as activation functions. The biases for the input, forget, output gate, and cell state are denoted by b^i, b^f, b^o, and b^c, respectively. Finally, the y output is computed as

$$y^t = \phi(W^{hy}h^t + b^y) \tag{2.6}$$

W and b represent the weights and biases, respectively. The outputs from the activation function ϕ can be generated for each time step or at the final time step [100], [43], [84].

2.3 Supervised Learning

The primary objective of supervised learning is the identification of a hypothesis function, denoted as h, which effectively establishes the mapping from input elements x to corresponding output y pairs contained within a designated training dataset of N examples, typically represented as (x_N, y_N). The y value is the label or meaning of the instance x. The latent

function $f(x)$ inherently generates the output values of y. The learning process involves discovering a hypothesis function h that closely approximates or models the underlying function $f(x)$.

A collection of previously unseen instances, often referred to as test examples, assesses the efficacy of the optimal hypothesis. The ability of the hypothesis to correctly predict the output values indicates its proficiency in generalization. These output values, represented as y, may exhibit a binary or categorical nature in the context of classification tasks or assume continuous, real-valued characteristics when pertaining to regression objectives [90].

2.4 Semi-supervised Learning

Semi-supervised learning involves leveraging limited l labeled data $\{(x_i, y_i)\}_{i=1}^{l}$ and a substantial pool of u unlabeled data $\{x_j\}_{j=1}^{u}$ to discover an optimal hypothesis function h. This function should exhibit robust generalization capabilities across labeled and unlabeled datasets, with y values taking on either discrete forms for classification or continuous values for regression tasks.

Semi-supervised learning offers significant practical utility in scenarios where the scarcity of labeled data constrains the efficacy of conventional supervised learning methods. The acquisition of labels y can present challenges, often necessitating human annotators, specialized equipment, or costly and time-consuming experiments. For example, instances x in speech recognition represent spoken utterances, while y denotes the associated transcriptions. Precise transcription, a labor-intensive task often performed by expert human annotators, can demand a substantial amount of time.

On the other hand, large amounts of unlabeled data can be easily acquired from radio transmissions. Exploiting these data reservoirs, semi-supervised learning can enhance the predictive accuracy for y when compared to the utilization of exclusively labeled data in supervised learning. Furthermore, it can contribute to annotating additional x instances, thereby mitigating the associated costs of labeling [40].

Overall, semi-supervised learning is a type of machine learning that utilizes both a small amount of labeled data and a large amount of unlabeled data. One specific technique within this framework is self-training, where the model iteratively labels the unlabeled data to enhance its performance.

2.5 Self-training

Self-training is an iterative optimization approach aimed at refining a hypothesis function denoted as h, originally derived through training on a restricted quantity of labeled data instances $\{(x_i, y_i)\}_{i=1}^{l}$. This iterative advancement is achieved by augmenting the training dataset, encompassing both the initially limited set of labeled data and pseudo-labeled data, the latter originating from unlabeled data sources $\{x_j\}_{j=1}^{u}$.

Algorithm 1 Self-training

Input: labeled data $\{(x_i, y_i)\}_{i=1}^{l}$, unlabeled data $\{x_j\}_{j=1}^{u}$.
Initially, let $L = \{(x_i, y_i)\}_{i=1}^{l}$ and $U = \{x_j\}_{j=1}^{u}$.
Repeat:
  Train h from L using supervised learning.
  Apply h to unlabeled instances in U.
  Remove a subset S from U; add $\{(x, h(x)) | x \in S\}$ to L.

As described by Algorithm 1, during the iterative loop stage, the model h undergoes supervised learning, followed by its application for label prediction on unlabeled data, thereby yielding a set of pseudo-labeled data. This augmented dataset is subsequently merged with the original labeled data. The model h then is subjected to retraining using the expanded training dataset, and the procedure repeats [40].

2.6 Autoencoders

An autoencoder creates a compressed internal representation of input data from unlabeled datasets using neural network techniques. The hidden layer h, encapsulates a code that represents the input and can subsequently be used for reconstruction in the output.

Building an autoencoder comprises two stages: the input unlabeled data is mapped to a lower-dimensional representation by a function $h = f(x)$, referred to as the encoder. Subsequently, another function $r = g(h)$, known as the decoder, reconstructs the input from this representation. A multi-layer neural network can be used for each stage.

Various architectural configurations can be selected based on the nature of the data under consideration. For image data, convolutional neural networks (CNNs) are often employed, while recurrent neural networks may be favored for sequential data. And feedforward neural networks are well-suited for less complex data.

Autoencoders find utility across multiple applications, including but not limited to anomaly detection, data denoising, and dimensionality reduction. The latter application

proves particularly beneficial for feature extraction, whereby the neural network autonomously identifies and represents the most informative dataset characteristics [42, 77].

2.7 Sequence Autoencoders

A sequence autoencoder represents a specialized form of autoencoder architecture that leverages recurrent neural networks for processing extended input sequences, directing them into a hidden layer denoted as h. Subsequently, it enables the reconstruction of the original input sequence. A Long Short-Term Memory (LSTM) autoencoder, a subtype of the sequence autoencoder, typically comprises a single hidden layer. This layer is responsible for encoding input sequences into a compact representation. It can be further employed in the decoding process, facilitated by an output layer, to reconstruct the initial input sequence [70, 113].

2.8 Semi-supervised Learning with Deep Neural Networks

In scenarios where large quantities of unlabeled data are available and labeled data are costly to obtain, semi-supervised learning methods are the optimal choice. These methods can leverage unlabeled data to learn essential patterns, thereby enabling the construction of a more effective supervised model that performs well with a limited number of labeled instances [23]. Despite the significant breakthroughs of deep neural networks (DNNs) when applied to large annotated datasets in fields such as speech recognition [110], image classification [56], and natural language processing (NLP) [119], these networks face challenges when learning from limited labeled data. This issue is particularly evident in real-world problems like reservoir characterization, where only a restricted number of boreholes provide known subsurface information.

While it is true that DNNs are significantly impacted by the limited availability of annotated datasets, these networks have the ability to extract valuable patterns from large quantities of unlabeled data. By leveraging this capability, they can effectively mitigate the challenges posed by the scarcity of labeled examples. Furthermore, the integration of deep neural networks with semi-supervised methods has led to the development of the deep semi-supervised learning (DSSL) field [117].

2.9 The Greedy Layer-wise Pre-training

Among semi-supervised methods for deep neural networks, greedy layer-wise pre-training is particularly notable when used for weight initialization with unlabeled data, as it brings the network closer to an optimal solution compared to random weight initialization in supervised learning [17]. This approach has demonstrated significant performance improvements, particularly when applied with Long Short-Term Memory (LSTM) networks on sequential data [113, 91].

To enable neural networks to leverage both unlabeled and labeled data, the greedy layer-wise technique follows these steps:

1. Construct the first LSTM layer as an autoencoder and train it using unlabeled data.
2. Add the next LSTM autoencoder layer, using the output of the previous layer as its input, and train it with the unlabeled dataset.
3. Repeat step 2 to build the desired number of layers for the deep neural network.
4. Attach a supervised output to the last LSTM layer based on the requirements of the machine learning task.
5. Fine-tune all parameters of the deep neural network using the limited labeled examples through a supervised learning approach.

In addition to constructing a specific internal representation for each layer from the input data, this initialization mitigates issues related to exploding and vanishing gradients in subsequent supervised learning, thereby facilitating faster convergence.

2.10 Conditional Independence Test

Conditional independence between two random variables, X and Y, given a third variable, Z, is established if and only if condition $P(X,Y|Z) = P(X|Z)P(Y|Z)$ is satisfied. This relationship is symbolically denoted as $X \perp\!\!\!\perp Y|Z$, while its opposite, indicating conditional dependence, is represented as $X \not\perp\!\!\!\perp Y|Z$.

The Conditional Independence Test (CIT) entails the assessment of the independence between variables X and Y, considering the presence of variable Z, or more broadly, considering the values of an additional set of variables [22]. This test represents a formidable challenge in the domain of causal discovery. In the context of time series analysis, the primary objective of

causal discovery is the statistically robust estimation of causal relationships among variables, including their temporal lags.

PC algorithm and Momentary Conditional Independence (PCMCI) is a causal discovery method that accommodates various conditional independence tests. The PC algorithm (Probability Distribution in an Undirected Graph C), which is a Markov discovery algorithm, combined with momentary conditional independence (MCI), forms the basis of the PCMCI method [89].

2.11 Entropy

Entropy, often referred to as Shannon Entropy, quantifies the degree of uncertainty or disorder in a system. For example, consider determining the sex of an unborn child. Initially, there is a 50% probability of correctly guessing the sex. However, as gestation progresses, prenatal ultrasound screening provides a more accurate determination due to the fetus's developing biological characteristics. This increase in accuracy, known as information gain, results from a reduction in uncertainty, or entropy, over time [25].

The uncertainty of the random variable X is quantified by Shannon Entropy $H(X)$, which is defined based on the probability mass function $p(x)$:

$$H(X) = -\sum_{x} p(x) log_2 p(x), \tag{2.7}$$

here, a base-2 logarithm is used, meaning the average uncertainty associated with X is measured in bits. Specifically, this represents the average number of bits required to fully describe the random variable X [27].

2.12 Linear-epoch Gradual-warmup

The use of large datasets for training deep neural networks is a fundamental aspect of deep learning. However, managing vast datasets during training presents challenges, particularly in terms of time consumption if memory is not utilized efficiently. Batch size plays a crucial role in reducing training time; however, as batch size increases to accelerate the process, neural network performance can degrade. The linear-epoch gradual warmup method offers an approach to optimize batch size, along with other parameters, ensuring efficient neural network training while minimizing time consumption.

The linear-epoch gradual warmup (LEGW) method can be employed for batch size adjustment in architectures such as Long Short-Term Memory (LSTM) and convolutional

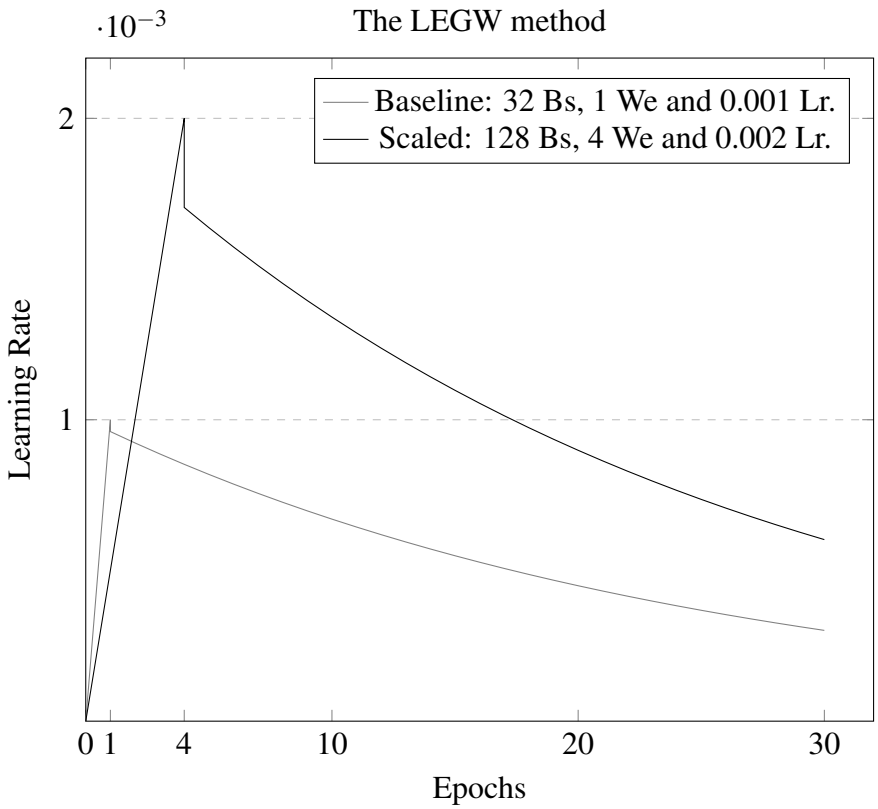


Fig. 2.3 Learning rate behavior and scaling through the LEGW method.

neural networks (CNNs). This approach requires tuning only baseline hyperparameters to calculate a scaled batch size for faster training. The key hyperparameters include batch size (Bs), warmup epochs (We), and learning rate (Lr), where Bs and We are scaled by a factor of k, and Lr is scaled by a factor of $\sqrt{k}$ [120].

The gradual warmup, or warmup epochs, involves a systematic increase in the Lr from an initial value of 0 to a predetermined fixed value (e.g., 0.001) throughout N epochs or m batches. As the baseline hyperparameters depicted in Fig. 2.3, when the We value equals 1, the learning rate achieves its designated fixed value within a single epoch. Subsequently, the Lr undergoes a decay process (e.g., exponential or multi-step) throughout the remaining epochs until a stopping criterion, such as early stopping, is met.

A similar behavior is observed when the baseline hyperparameters are scaled (Fig. 2.3) by a factor of 4, transitioning from 32 Bs, 1 We, and 0.001 Lr to 128, 4, and 0.002, respectively. Notably, this adjustment results in a larger Bs.

2.13 Multi-worker Strategy

The multi-worker strategy represents one of the diverse strategies offered within TensorFlow 2.x, with the specific aim of reducing model training time through the utilization of multiple graphics processing unit (GPUs) distributed across multiple machines, where each machine may host one or more GPUs [80].

In this particular strategy, each machine assumes the role of a *worker*, with one designated as the *chief worker* responsible for coordination. During the training process, all machines engage in synchronous training. This entails the aggregation of model parameters across all machines in a synchronized manner after each epoch step and over varying input data [2].

2.14 Data Augmentation

Using data augmentation techniques effectively enhances the generalization performance of deep neural networks, particularly in scenarios where the availability of labeled data is constrained [84, 78]. While architectural enhancements, as seen in models like ResNet and DenseNet, can contribute to improved generalization, data augmentation primarily addresses intrinsic limitations within the dataset itself [97].

Small datasets pose challenges related to limited diversity in examples and potential class imbalance, which may lead to overfitting in neural networks. Data augmentation addresses these issues by introducing a range of transformations to the limited dataset, generating new examples while maintaining the distribution characteristics of the original dataset. Consequently, this augmentation expands the effective size of the dataset [78].

Data augmentation methods are exclusively applied to the training data, leaving the test data unaltered. These techniques can be categorized into two primary types: oversampling and data-warping augmentations. Oversampling entails the generation of synthetic examples, often implemented through techniques like generative adversarial networks (GANs) [97]. This approach is frequently utilized to address issues related to class imbalance. Conversely, data warping augmentation involves applying conventional transformation techniques to the existing dataset, including flipping, rotation, translation, zooming in, and zooming out [84].

2.15 Seismic and Well Data

The primary aim of seismic data acquisition is to gain insight into the geological properties of the Earth's subsurface. This is achieved by initiating controlled seismic events, typically by generating artificial earthquakes on the Earth's surface. Subsequently, a collection of

time-series signals is recorded as a response to these events. Following a comprehensive signal processing procedure, these recorded signals are organized into a three-dimensional seismic cube, facilitating the interpretation of subsurface structures [87].

In contrast to post-stack seismic data, pre-stack time migration seismic cubes contain instances of common depth points (CDP), where a defined number of seismic traces can be identified. These signals are generated by capturing the energy from artificial seismic events using specialized sensors. In post-stack seismic cubes, a mathematical operation is applied to the seismic traces within each CDP to produce a single composite signal. As Fig. 2.4 shows, each CDP encompasses 40 traces arranged in cross-lines along an in-line. Multiple such in-lines collectively form the complete pre-stack seismic cube [122].

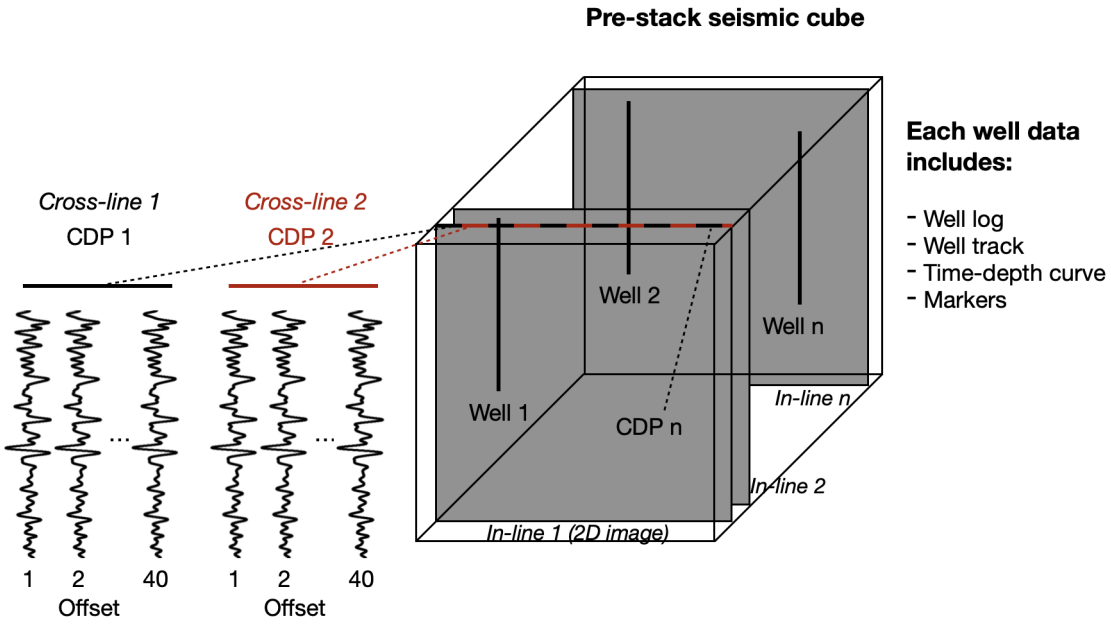


Fig. 2.4 Pre-stack seismic cube representation and well data.

Well data provides direct insights into subsurface characteristics, as it is acquired through specialized sensing tools deployed along the entire depth of a borehole. Typically, each well yields multiple logs obtained from various sensors, which are subsequently integrated into a comprehensive well log dataset. This dataset offers a detailed description of the geological and fluid properties in the vicinity of the borehole [68].

Numerous parameters characterizing subsurface attributes are present within well log data, including, but not limited to, gamma ray intensity and electrical resistance. However, our specific focus lies on two essential properties: bulk density and sonic velocity. Bulk density measures the overall porosity within the geological formation, while sonic velocity quantifies the propagation time of acoustic waves through subsurface materials.

Alongside well log data, supplementary well-related datasets include the following: well track data, which accounts for borehole deviation in X, Y, and Z coordinates denoting latitude, longitude, and depth, respectively; the time-depth curve, consistently featuring depth and two-way travel time logs, with the former typically referring to True Vertical Depth Sub-Sea (TVDSS) and the latter representing the time elapsed for seismic waves to travel from the source to the reflector and back to the receiver; and markers data, encompassing a collection of identifiable bedrock strata, often exhibiting distinct physical characteristics, which can be recognized and traced over substantial horizontal distances, such as geological eras like Pliocene, Eocene, and Miocene.

Chapter 3

Literature Review on Supervised, and Semi-supervised Learning Approaches for Sequential Image Classification and Reservoir Characterization

3.1 Sequential Image Classification

Various reports in the literature have employed recurrent neural networks (RNNs) and related architectures to analyze images represented as sequences. Table 3.1 provides an overview of accuracy scores for several neural networks, including Long Short-Term Memory (LSTM) networks, Gated Recurrent Units (GRU), Temporal Convolutional Networks (TCN), and Skip-Connected LSTM Identity (SC-LSTM-I). The TCN represents a type of convolutional network that adopts a temporal perspective akin to the RNNs. SC-LSTM-I, on the other hand, enhances information propagation across distant time steps within an LSTM unit. Notably, these architectures have demonstrated strong performance in grayscale image classification. A sequence length of 784 was used to evaluate TCN, GRU, and SC-LSTM-I models. Additionally, previous studies employed a sequence length of 28 [20], while in [94], a range of sequence lengths between 140 and 420 samples was utilized for various LSTM architectures.

The aforementioned works aimed to improve the accuracy of LSTM networks through various strategies. For example, the SC-LSTM-I methodology, which introduces additional connections between time steps, enhanced the neural network's performance, particularly when using a sequence length of 784, as demonstrated in the P-MNIST dataset [58]. In [20],

an alternative approach is documented, focusing on the impact of various architectures and hyperparameters on network performance. The networks under examination were trained on the MNIST dataset, with images represented as sequences of 28 pixels in length. Conversely, catastrophic forgetting was investigated by Schak in [94], particularly in the context of long sequences. This phenomenon arises when a deep LSTM network loses previously acquired knowledge due to retraining on new samples. Additionally, in [13], an alternative framework for addressing sequence modeling tasks is presented, introducing the TCN, which applies a convolutional network architecture with a temporal focus.

Table 3.1 Current advancements in LSTM utilization for datasets containing black-and-white images.

Method	Dataset	Accuracy
SC-LSTM-I [111]	P-MNIST	94.80%
LSTM [20]	MNIST	99.27%
LSTM [94]	Devanagari	99.40%
GRU [13]	Seq. MNIST	96.20%
TCN [13]	Seq. MNIST	99.00%

3.2 Semi-supervised Methods

Pre-training and self-training methods are significant techniques in semi-supervised learning, each offering distinctive advantages in utilizing unlabeled data. This study explores their applications within contemporary advancements in machine learning.

Pre-training methods offer a valuable means to enhance the effectiveness of the supervised learning phase within the context of a semi-supervised framework. Notably, techniques such as autoencoders [29] and sequence autoencoders [70] have proven highly effective in this context. These pre-training strategies involve the construction of neural network layers utilizing Rectifier Linear Units (ReLUs) or Long Short-Term Memory (LSTM) units. Their application has consistently yielded improved classification performance in various experimental settings encompassing speech, text, and image datasets.

A noteworthy application of pre-training can be observed in the DeepHeart model [14], a semi-supervised deep learning framework applied within the domain of medical science. DeepHeart incorporates a sequence autoencoder with an LSTM-based architecture during the pre-training phase, followed by fine-tuning through supervised learning. This model has demonstrated exceptional accuracy with diverse medical datasets, including conditions such as high cholesterol, high blood pressure, and sleep apnea. Comparative analyses have

underscored the superior performance of pre-trained models like DeepHeart when contrasted with their non-pre-trained counterparts.

A notable category of pre-training techniques involves applying greedy layer-wise unsupervised learning, which has demonstrated its capacity to optimize deep neural networks [17]. Notably, this approach has been effectively employed in LSTM stacked autoencoders, enhancing these deep networks. These optimized deep networks have found application in diverse domains, including multivariate time series forecasting [91], image recognition, and video recognition [113]. They consistently outperform networks trained with random initialization exclusively, underscoring the efficacy of the layer-wise pre-training methodology. Importantly, this layer-wise approach has also found successful application within convolutional neural networks (CNNs) [65, 103].

On the other hand, self-training represents an alternative approach for harnessing supplementary data, notably unlabeled data. The self-training method, specifically when founded upon the principles of noisy student training, has shown remarkable performance, surpassing traditional pre-training and fine-tuning methodologies. This efficacy is exemplified in a comprehensive study outlined in [123], where the COCO dataset's labeled data is systematically varied in size, and the ImageNet dataset is incorporated as an additional source of unlabeled data.

Another facet of self-training takes the form of naive semi-supervised deep learning [63], characterized by using an initially well-trained classifier with limited labeled data. This proficient classifier assigns pseudo-labels to unlabeled data, which are then employed for training a deep model. Subsequently, fine-tuning is carried out using authentic labeled data. This process is iteratively repeated, with the deep model predicting pseudo-labels for unlabeled data, until accuracy convergence is achieved. Experimental findings reveal superior accuracy when this approach is applied to datasets like MNIST, CIFAR-10, and IMDB, provided an effective classifier is employed in conjunction with the deep model.

3.3 Deep Neural Networks in Reservoir Characterization

The shortage of labeled data in practical applications substantially impedes the efficacy and generalization of supervised deep-learning approaches. This scarcity primarily arises from the need for expert knowledge in data annotation, which is not only time- and effort-intensive but also constrained by various other factors. This challenge is particularly evident in fields such as reservoir characterization for oil exploration [4, 81], healthcare [53, 105], and natural language processing [114, 39]. While labeled data is difficult and costly to acquire, unlabeled data is abundant and relatively easy to obtain. For example, in reservoir

characterization, large volumes of unstructured seismic data are stored as seismic cubes [9, 99], while in healthcare, unlabeled data often comes in the form of genomic information [7, 37]. Given this availability, it is prudent to leverage unlabeled data to augment supervised models. Unsupervised learning methods, in particular, have shown considerable potential in extracting valuable patterns from such data and improving model performance.

The intricate subsurface structures of the Earth compound the complexity of seismic data and well-log data. The association of seismic data with well-log data is often utilized to construct labeled datasets [92]. Consequently, effectively modeling such data necessitates the capacity to discern nonlinear relationships between input and output variables. Deep neural networks represent well-suited tools for addressing this complex task.

3.3.1 Supervised Learning Approach

Deep neural networks have found extensive applications in geophysics, particularly in interpreting seismic data, encompassing data inversion, data quality enhancement, lithofacies classification, and predicting petrophysical properties. In the context of data inversion, a convolutional neural network [71] and specialized architectures like the seismic inversion network (SeisInvNet) [60, 86] were deployed individually to derive accurate velocity models. Additionally, data quality improvement has been notably advanced through utilizing generative adversarial network (GANs), as demonstrated in the work of Azevedo [10], which pertains to subsurface model reconstruction. Furthermore, Kaur [51, 50] has harnessed GANs for tasks related to interpolating missing traces and mitigating ground-roll noise within seismic data. Meanwhile, Song's research [98] has effectively employed convolutional neural networks (CNNs) to eliminate random noise in seismic data, and Liu's work [66] has yielded high-resolution seismic images through the application of CNN-based methodologies.

A range of deep learning techniques have demonstrated noteworthy capabilities in lithofacies classification for seismic data analysis. Cunha [28], for instance, employed transfer learning techniques in conjunction with CNNs to successfully detect genuine seismic faults, even when dealing with limited datasets. Additionally, Islam's approach [49] integrated a combination of U-Net and SE-ResNet architectures to identify salt bodies within seismic attributes effectively. Also, Yang [115] framed seismic horizon tracking as an image classification task, leveraging the power of CNNs to surpass traditional methods. Furthermore, Dixit [31] introduced a methodology that harnesses a multilayer perceptron (MLP) to detect gas chimneys within multi-attribute seismic data, enabling insights into fluid behavior within the geological pathways. Meanwhile, He [47] employed a distinct MLP for lithology and fluid classification, primarily based on log data derived from a tight sandstone reservoir. Bedi's

research [16], in contrast, leveraged an Long Short-Term Memory (LSTM) architecture for porosity classification, focusing on seismic attributes.

Recent advancements in predicting petrophysical properties have witnessed the integration of deep learning techniques. Tembely's approach [104], for example, incorporated CNNs as a complementary tool alongside conventional methods to predict permeability from 3D micro-computed tomography (micro-CT) images. Similarly, Araya-Polo [6] achieved predictions of this rock property, albeit from 2D micro-CT images. On the other hand, Gu [44] introduced a hybrid data-driven model, combining continuous restricted Boltzmann machines, particle swarm optimization, and support vector regression to predict permeability, primarily relying on log data. Kim's research [54] drew upon well-log data and augmented core data to train a deep neural network comprising multiple hidden layers, with the primary objective of predicting mineralogy. This endeavor significantly contributes to an enhanced understanding of rock properties. Zhang [121] took a distinct approach, using elastic parameters (compressional wave velocity, shear wave velocity, and density) in conjunction with a gated recurrent neural network to predict multiple physical parameters, including porosity, water saturation, and shale content—a method of particular practicality in exploration contexts. Yang [116], meanwhile, employed both CNN and Bi-directional LSTM architectures in estimating porosity, specifically from pre-stack seismic gathers.

3.3.2 Semi-supervised Learning Approach

Predominantly, the investigations in Section 3.3.1 are focused on model training for classification or prediction tasks using labeled datasets comprised of well-log or annotated seismic data. The availability of well-log data is often limited due to the substantial costs associated with drilling operations. Regarding annotated seismic data, its utilization demands domain expertise and considerable time investment. Nonetheless, a wealth of unlabeled seismic data is available. This reservoir of unlabeled data holds the potential for feature extraction and performance enhancement in prediction and classification facilitated by deep semi-supervised learning (DSSL). Surprisingly, this approach remains underutilized in this domain [92].

The subsequent papers exemplify the utilization of DSSL. For instance, Pratama [81] introduced a methodology encompassing two stages. In the initial unsupervised stage, this approach employs kernel Principal Component Analysis (PCA) and the K-means clustering algorithm to generate labeled data from seismic attributes autonomously. Subsequently, this labeled data is utilized to train a convolutional neural network (CNN), specifically the U-Net architecture, for the automated detection of geological features. Asghar [9] employed a deep neural network (DNN) to predict seismic facies. In this approach, the DNN is initially trained using seismic inverted data as unlabeled data and well-log data

as labeled data. The training process initiates with supervised learning using the available labeled data. Subsequently, the DNN predicts pseudo-labeled examples from the adjacent unlabeled data near the well. These pseudo-labeled examples are then incorporated into the dataset for subsequent supervised learning iteration. This process is conducted iteratively to refine the model. Song's approach [99] involved using pre-stack seismic data for feature extraction and well-log data for labeling, thereby constructing a training dataset for the k-Nearest Neighbors (kNN) classifier. This kNN classifier was employed to predict pseudo-labeled data from unlabeled seismic information. The pseudo-labeled data was subsequently incorporated into the actual dataset. This augmented dataset was then used to pre-train a CNN, followed by the fine-tuning stage, yielding a robust CNN-based gas-bearing classifier. Liu [67] conducted training of a deep autoencoder that incorporated labeled and unlabeled data, this last encompassing well-data and seismic data. This approach aimed to extract latent and potentially hidden features within the data. At this stage, to build a model for facies identification, a small amount of annotated borehole data was used for neural network fine-tuning. In contrast, seismic data annotated with borehole information was employed for semi-supervised learning, with a particular focus on incremental learning [101]. Using a similarity method, a comparable dataset is generated. This similar dataset is subsequently employed in supervised learning, utilizing a U-Net model initialized with unlabeled seismic data. Pseudo-labels are predicted in this phase and incorporated into the reference dataset for iterative refinement. This method enhances seismic facies identification.

Marfurt [72] has advocated using pre-stack seismic data in light of advancements in multi-attribute analysis techniques. This data type can extract more comprehensive information about subsurface properties than stacked seismic data. Despite these advantages, pre-stack seismic data is underutilized in reservoir characterization within DSSL and even with conventional models in semi-supervised learning. These traditional models include the self-organizing map (SOM), artificial bee colony (ABC) algorithm [21], K-means algorithm and Laplacian support vector machine (SVM) [62, 32], least squares SVM [69], unsupervised isolation forest with split-selection criterion (SCiForest) algorithm [5], transductive conditional random field regression (TCRFR) [64] and other shallow models [83].

Chapter 4

Methodology: Supervised, and Semi-supervised Learning on Sequential Data, and a Case Study Analysis of Seismic Data

4.1 Long Short-Term Memory Classification: An Approach Addressing Variations in Sequence Length and Order

Recurrent neural networks are widely applied in fields such as genomics, time-series analysis, and natural language processing, where sequential data is predominant. Various engineering challenges can be reformulated as sequential data problems [95, 30]. For instance, in classification tasks, conventional image data, typically treated as two or more-dimensional arrays [41], can be restructured into sequences through vectorization, wherein images are transformed into pixel vectors with specific lengths and order. Recurrent Neural Networks (RNNs) have exhibited favorable performance when employed to classify grayscale images in this manner [111, 20, 94, 13].

These pixel vectors serve as the feature vectors employed for input to the neural network. In the following section, our focus lies on the reconfiguration of the feature vector structure, wherein we aim to investigate the impact of changing the order and length of sequences on the overall performance of RNNs, particularly when leveraging Long Short-Term Memory (LSTM) units.

By applying this procedure, the MNIST dataset images are treated as sequential data, where each image is represented as a sequence of pixel vectors. Although the MNIST

dataset is not inherently temporal, this method imposes an explicit pixel order throughout the sequence. This methodology can be further generalized to images that are naturally generated over time, such as seismic images (comprising time-series signals), dynamic hand gesture images [93], and various medical imaging modalities like magnetic resonance imaging (MRI) [73]. These types of images inherently exhibit long-range temporal dependencies, making them particularly well-suited for modeling using recurrent neural networks, which are designed to capture and learn temporal dependencies within the training data.

In our experiments, we utilize the LSTM network to evaluate the impact of varying sequence orders and lengths derived from datasets of black-and-white images. The neural network output can be generated at each time step: $t-1$, t, and $t+1$. The model architecture comprises three hidden layers, and training is conducted under supervised conditions.

Various configurations were tested to determine the optimal LSTM architecture and associated parameters for consistent image dataset classification. The most favorable performance results were attained by adopting the following parameters: a three-layer architecture with 512 LSTM units per layer, a learning rate of 0.001 featuring exponential decay, and a batch size of 256 images. An early stopping criterion was also implemented, terminating training when no further changes in the loss function occurred. The model was compiled using the Adam optimizer [55], the cross-entropy [77] loss function and the accuracy metric.

A 2D black-and-white image is represented by the matrix A, where each pixel $a_{i,j}$ is positioned at coordinates (i, j). This matrix, consisting of $M \times N$ pixels, is processed by the LSTM network row by row across the time steps $t-1$, t, ..., $t+M$. The input data is formally described by equation (4.2).

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M-1,0} & a_{M-1,1} & \cdots & a_{M-1,N-1} \end{bmatrix} \tag{4.1}$$

$$x^{t-1} = \begin{bmatrix} a_{0,0} \\ a_{0,1} \\ \vdots \\ a_{0,N-1} \end{bmatrix}, \; x^{t} = \begin{bmatrix} a_{1,0} \\ a_{1,1} \\ \vdots \\ a_{1,N-1} \end{bmatrix} \; ..., and \qquad x^{t+M} = \begin{bmatrix} a_{M-1,0} \\ a_{M-1,1} \\ \vdots \\ a_{M-1,N-1} \end{bmatrix} \tag{4.2}$$

The training examples consist of a set of black-and-white images. Fig. 4.1 explicitly demonstrates the process by which the LSTM network receives sequential representations of an image from the MNIST dataset.

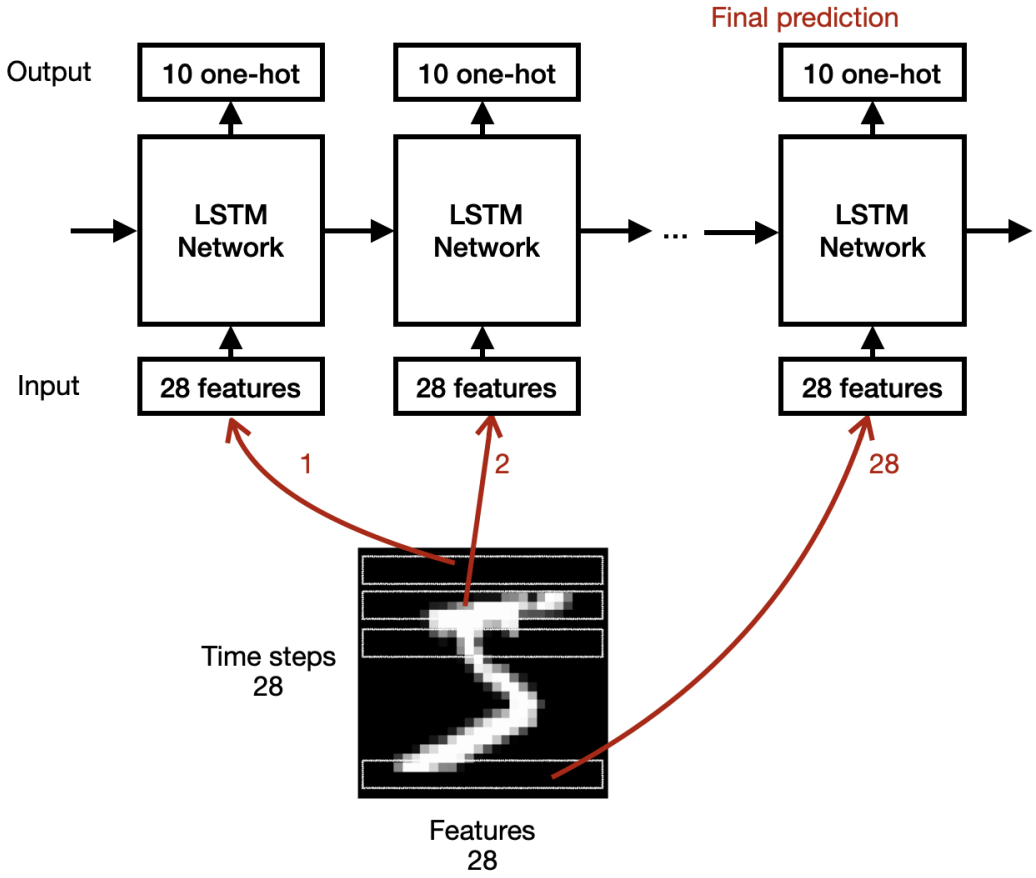


Fig. 4.1 The procedure for feeding an MNIST image into an LSTM neural network

Three methods were designed to alter the sequence order of images, generating different training datasets. In the horizontal order method (H), image rows were used as input sequences for the neural network, while in the vertical order method (V), image columns served as the input sequences. Additionally, the spiral order (S) method extracted sequences by starting from the center of the image and spiraling outward until the outermost pixels were reached.

In this study, the raw data consists of black-and-white images with dimensions $M \times N$, corresponding to 28×28 pixels. To modify the sequence length, the values of $M \times N$ were adjusted, while sequence order variations were achieved using the horizontal (H), vertical (V), and spiral (S) methods.

Experiment E is carried out with a specific combination of $M \times N$ image shape and a sort order H, V, or S as shown in Table 4.1. Each experiment yields an accuracy.

To modify sequence length, the values of $M \times N$ were set to: (2,392), (4,196), (7,112), (8,98), (14,56), (16,49), (28,28), (49,16), (56,14), (98,8), (112,7), and (196,4). These

Table 4.1 Multiple datasets were generated by applying specific $M \times N$ configurations and sequence order methods for experimental purposes.

Shape	Dataset1		
$M \times N$	E	E	E
Order Sorts	H	V	S

configurations, in combination with sequence order alteration methods, were applied to the Sign Language MNIST (SLMNIST), FashionMNIST, MNIST, MNIST-C, and notMNIST datasets, resulting in 180 experimental runs.

4.2 Conditional Independence Testing with Black-and-White Images

Among the multiple conditional independence tests that can be employed by the PCMCI (PC algorithm and Momentary Conditional Independence) causal discovery framework, the linear partial correlation (ParCorr) test is utilized for evaluating black-and-white images. Additionally, this methodology is implemented in Python for enhanced usability [88].

For conditional independence testing on the black-and-white image dataset, the PCMCI framework considers images in the form $M \times N$, where N represents the set of variables and M denotes the sample size. A subset of 500 images was selected from the training dataset, resulting in a total sample size of $500 \times M$.

The method employs default values for three free parameters, except the maximum time lag parameter, denoted as τ, specifically configured to a value of 2.

Following the application of PCMCI to analyze the dataset, it is possible to extract significant links representing dependence or independence relationships among variables or to construct a network graph illustrating the relationships among the variables. However, comparing variables between two datasets may be challenging to illustrate using the previously mentioned representations.

The PCMCI algorithm returns a pair of arrays that enable our analysis. The independence array predominantly consists of zero values, indicating a lack of independence between variables. In contrast, the dependence array, with values ranging from 0 to 1 and selected for analysis, provides crucial information about the dependencies between variables.

A variable measure in the dependence array is represented by $d_{i,j,k}$, which takes the form $(N, N, \tau + 1)$. Here, N_i and N_j denote the variables measured at time lag τ_k.

In the dependence array, to obtain a single measure of dependence for the entire dataset, a summation operation was applied to $d_{i,j,k}$,

$$\sum_{k=0}^{\tau} \sum_{j=0}^{N-1} \sum_{i=0}^{N-1} d_{i,j,k}, \tag{4.3}$$

this single measurement for a dataset consisting of images in the form $M \times N$ is referred to as the *dependence index* (DI).

Table 4.2 A dataset consisting of images in the form $M \times N$ has a corresponding dependence index (DI).

Shape	Dataset1		
$M \times N$	DI	DI	DI
Order Sorts	H	V	S

4.3 Techniques for Measuring Entropy in Image Datasets

Two methods were developed to measure entropy in black-and-white image datasets: entropy per image and entropy per image row.

To compute entropy per image, which is used to assess the entropy of a dataset, the probability distribution of the pixels in each image must first be determined. This distribution is then applied to the entropy formula (2.7). In this manner, the entropy of each individual image is calculated, followed by the computation of the average entropy across the entire dataset.

Similar to the previous method, calculating entropy per image row requires determining the probability distribution, though in this case, the distribution is derived from a single row of pixels in an image. This distribution is then applied to the entropy formula (2.7). To estimate the entropy for an entire image dataset, entropy per image row is computed for each row of all images, and the average value is obtained by considering the rows across the entire dataset [107].

4.4 Proposed Method: Self-training Layer-wise

Greedy layer-wise pre-training plays a pivotal role in regularizing neural network parameters. This regularization is the foundation for diverse advantages, including improved initialization of local minima. This, in turn, results in enhanced training optimization and subsequently improves the generalization of test examples for both regression and classification tasks [36, 113].

In contrast, self-training has proven more adaptable in constructing an effective model representation for the specific task at hand [26]. For instance, it has demonstrated its ability to effectively leverage unlabeled data in cases where pre-training methods may falter, mainly when slight differences exist between the labeled and unlabeled datasets. Moreover, self-training outperforms pre-training methods as the size of the labeled dataset increases or when augmentation techniques are applied to labeled data [123, 112]. In fact, the pseudo-labeling capability of self-training is regarded as a form of data augmentation.

Enhancing the robustness of machine learning models can be achieved through the effective utilization of unlabeled data [112]. The two methods above employ distinct strategies for handling unlabeled data. The pre-training method involves direct training on unlabeled data, while self-training necessitates the assignment of pseudo-labels to unlabeled data for its incorporation into the model. Each of these capacities holds the potential to contribute to the construction of a superior pre-trained model and an improved task-specific model, respectively. This study introduces a novel self-training layer-wise approach that integrates these capacities, harnessing the advantages of working with unlabeled data. Utilizing this innovative method can yield a more robust model for various machine-learning tasks.

Algorithm 2 Self-training layer-wise

Input: labeled data $\{(x_i, y_i)\}_{i=1}^{l}$, unlabeled data $\{x_j\}_{j=1}^{u}$.
Initially, let $L = \{(x_i, y_i)\}_{i=1}^{l}$ and $U = \{x_j\}_{j=1}^{u}$.
1. Using greedy layer-wise pre-training initialize the multi-layer neural network h with U using unsupervised learning:
   1. Train the first layer of h.
   2. Add the next layer of h, using the output of the previous layer as its input, and train it with U. The weights of all previous layers are fixed.
   3. Repeat step 2 to build the desired number of layers for h.
2. Add the new supervised layer to the output of h.
3. Repeat:
   1.Train h from L using supervised learning.
   2. Apply h to unlabeled instances in U.
   3. Remove a subset S from U; add $\{(x, h(x))|x \in S\}$ to L.

In the context of semi-supervised learning, the proposed method is designed for scenarios characterized by a scarcity of labeled data L and an abundance of unlabeled data U. The technique is outlined by Algorithm 2, which can be summarized by three steps. Firstly, the method leverages the extensive unlabeled dataset U to perform pre-training on a multi-layer neural network h in a greedy layer-wise manner (also see Section 2.9), utilizing unsupervised techniques on large datasets to uncover latent patterns. Secondly, following the initialization of model h's parameters, the output from h is directed into a new supervised layer designed

for the specific machine-learning task. Thirdly, this enhanced version of h, which includes the pre-trained model h and the new supervised layer, undergoes fine-tuning, a process that incorporates both the unlabeled data U and the limited labeled data L within a self-training framework. In addition to increasing the number of training examples and improving the model's overall generalization, the self-training approach can contribute to building an optimal model for the required machine learning task. Other operations such as the selection process of the subset S (or pseudo-labeled examples), and the termination criteria for the iterative self-training process are consistent with the self-training method described in Section 4.5.

Notably, this novel approach can utilize the same unlabeled data from various perspectives, improving model performance.

Fig. 4.2 schematically represents the data flow involving unlabeled, labeled, and pseudo-labeled data throughout the pre-training and fine-tuning processes of the self-training layer-wise algorithm. The illustration effectively delineates the two training phases. In both phases, unlabeled data is leveraged, with the initial phase focusing on unsupervised learning exclusively using unlabeled data. In the subsequent phase, the same unlabeled data is combined with the limited labeled examples. This latter phase significantly influences model performance, as it undergoes fine-tuning via self-training, as opposed to traditional supervised learning.

4.5 Semi-supervised Methods with Medium Size Datasets

The scarcity of labeled data poses a significant challenge across various domains, including healthcare, reservoir characterization, and text and image analysis. Developing an optimal machine learning model typically requires a substantial amount of labeled data, which is often costly and difficult to obtain. In contrast, large volumes of unstructured or unlabeled data are readily available in these fields. Semi-supervised methods, therefore, offer a promising approach for extracting valuable information from unlabeled data.

In addressing this issue, our objective is to introduce a semi-supervised methodology. To achieve this, we investigate three distinct approaches: supervised learning, semi-supervised layer-wise techniques, and self-training methods. Based on our exploration of these methods, we formulate a novel methodology (Section 4.4) designed to address the problem at hand effectively.

Our approach emphasizes the utilization of benchmark datasets for the specific purpose of method development and assessment, with particular emphasis on the methodology rather than the data's intricacies. Our perspective aligns with the semi-supervised paradigm, which

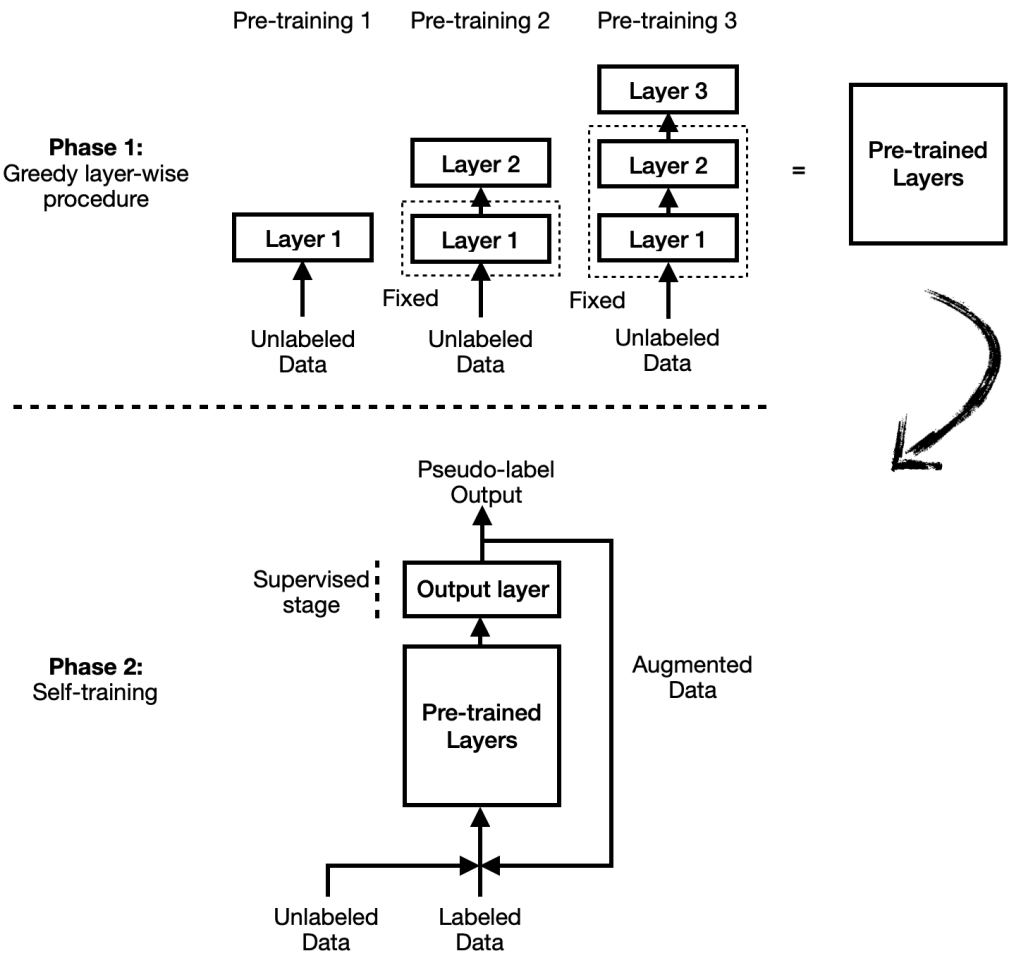


Fig. 4.2 Flow of data through particular neural network layers by applying the self-training layer-wise method.

mandates the presence of both labeled and unlabeled instances. To adhere to this perspective, a substantial portion of labels in the datasets was intentionally removed, leaving only a limited subset of examples with labels. In essence, within a given labeled dataset, we partition a small fraction as labeled instances while most are treated as unlabeled.

In our experimentation, we employ two well-established benchmark datasets, MNIST and FASHION, which are described in Section 5.1.1, to evaluate the performance of four distinct models. These models encompass the purely supervised approach, the semi-supervised layer-wise method, self-training, and our proposed self-training layer-wise technique described in Section 4.4. These models are implemented using recurrent network architectures, specifically Long Short-Term Memory (LSTM) networks.

In preparation for utilizing the four distinct methods, each dataset transforms from its original subsetting—comprising training and test sets—to a modified arrangement, namely

pre-training, training, and test sets. Training and test subsets are labeled and small, while the pre-training subset is unlabeled and large. This adjustment is delineated in Table 4.3. Note that the treatment of the Quickdraw bitmap dataset is detailed in Section 4.6.

We will now provide a detailed exposition of the four methods, focusing on the strategies employed when dealing with a limited number of labeled examples. It is also worth highlighting that each method utilized 32 batch size for training.

Table 4.3 The original subsets and the modified subsets across all datasets.

Dataset	Original subsets	Modified subsets
MNIST (70k full)	60k Training	50k Pre-training
		10k Training
	10k Test	10k Test
FASHION (70k full)	60k Training	50k Pre-training
		10k Training
	10k Test	10k Test
Quickdraw bitmap (725,000 full)	700k Training	583,310 Pre-training
		116,690 Training
	25k Test	25k Test

Supervised. After an extensive evaluation of multiple LSTM models within a supervised learning framework, the optimal hyperparameters are determined as follows: a three-layer architecture with 512 LSTM units and two dense layers as output (512 and 10 units, respectively), employing the Adam optimizer with a learning rate of 0.001 featuring exponential decay, utilizing a cross-entropy loss function, and assessing model performance based on accuracy as a metric. The neural network architecture is illustrated in Fig. 4.3. With these settings, the model achieves peak performance on the MNIST dataset, yielding an accuracy rate of 98.95% when operating on the original training and test subsets (refer to Table 4.3).

While the procedure above is applicable in scenarios with a sufficient quantity of labeled examples, our investigation extends to cases with a limited number of labeled instances. To address this, we significantly diminish the number of labeled examples delineated in Table 4.4. For instance, in the case of the MNIST dataset, the labeled examples for training are substantially reduced from the original 60k (constituting 100%) to 10k representing 16.67% of the total training instances. In various experiments, this number is further reduced to as low as 200 examples, corresponding to a mere 0.33% of the total. Notably, this minimal percentage of labeled examples represents the most challenging scenario among our experiments. A similar reduction approach is applied to the FASHION dataset.

Semi-supervised layer-wise. Semi-supervised learning represents an approach that leverages the utility of unlabeled data, offering various strategies for its incorporation. In our

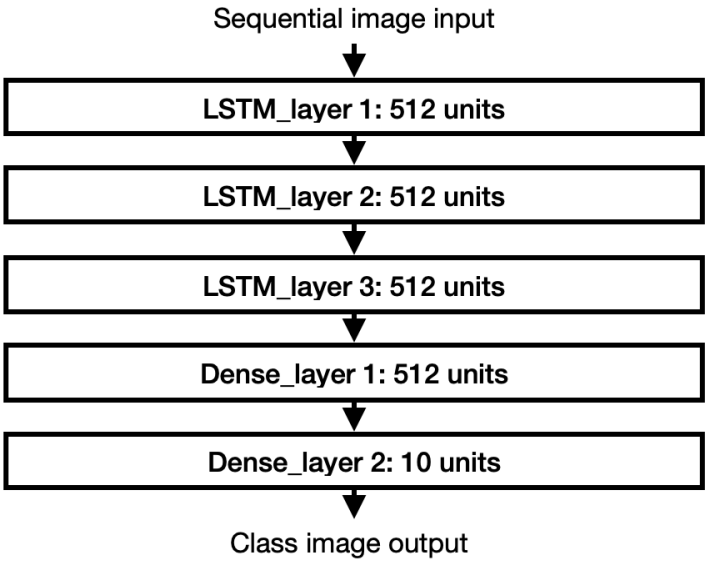


Fig. 4.3 Neural network architecture used for supervised learning and self-training.

investigation, we concentrate on pre-training methodologies and specifically evaluate two distinct approaches: a sequence autoencoder [70, 14] and a greedy layer-wise procedure [91, 113]. Notably, the latter outperforms the former in accordance with our experimental findings and those of [113]. Consequently, we adopt the greedy layer-wise procedure as the unsupervised pre-training component within the semi-supervised layer-wise methodology.

During the unsupervised learning phase, the model is configured with specific hyperparameters as shown in Fig. 4.4: three layers, each containing 512 LSTM units, a dense layer of 28 (related to the number of features) units as the output stage, and a learning rate of 0.001 featuring exponential decay. The model is compiled utilizing the Adam optimizer, employs a mean squared error loss function, and employs the loss value as a metric. Pre-training is carried out using the designated *pre-training subset* of the corresponding dataset, as detailed in Table 4.3.

The pre-training phase is an initialization process for the layers within the LSTM architecture. Subsequently, fine-tuning is performed using the identical hyperparameters employed in the supervised learning method except for the output stage, which is changed to two dense layers of 28 and 10 units. Fine-tuning is executed utilizing the limited number of labeled examples, as specified in Table 4.4, resulting in various experimental scenarios.

Self-training. The architectural configuration utilized in this method remains consistent with that employed in supervised learning (Fig. 4.3). Specifically, we employ an LSTM-based architecture with three layers (512 units each) and two dense layers as output, with the

Table 4.4 The number of training examples is drastically reduced to 16.67%, followed by a gradual decrease to 0.33% in various experiments for each dataset.

MNIST	FASHION	Quickdraw	%
10k	10k	116690	16.67
9k	9k	105000	15.00
8k	8k	93310	13.33
7k	7k	81690	11.67
6k	6k	70000	10.00
5k	5k	58310	8.33
4k	4k	46690	6.67
3k	3k	35000	5.00
2k	2k	23310	3.33
1k	1k	11690	1.67
0.9k	0.9k	10500	1.50
0.8k	0.8k	9310	1.33
0.7k	0.7k	8190	1.17
0.6k	0.6k	7000	1.00
0.5k	0.5k	5810	0.83
0.4k	0.4k	4690	0.67
0.3k	0.3k	3500	0.50
0.2k	0.2k	2310	0.33

hyperparameters maintained at identical settings. The training method is described by the Algorithm 1 in Section 2.5.

The quantity of pseudo-labeled examples to be incorporated is determined by $f(x)$, representing the number of examples to augment the set of x actual examples (or pseudo-labeled and actual examples) for subsequent training iterations.

$$f(x) = \begin{cases} 100 & \text{for x} \in [100, 1k) \\ 1k & \text{for x} \in [1k, 10k) \\ 10k & \text{for x} \in [10k, 100k) \end{cases} \tag{4.4}$$

The selection process for determining the best prediction, utilized as a pseudo-labeled example, relies on the neural network output. The neural network employs a multi-class (ten classes) output with the softmax function activation. This approach enables the assessment of class probability outputs for a specific input example, with the pseudo-label assigned to the example associated with the highest predicted class probability.

The iterative self-training process is terminated based on two distinct conditions: depletion of the unlabeled examples or class imbalance, wherein at least one class has fewer

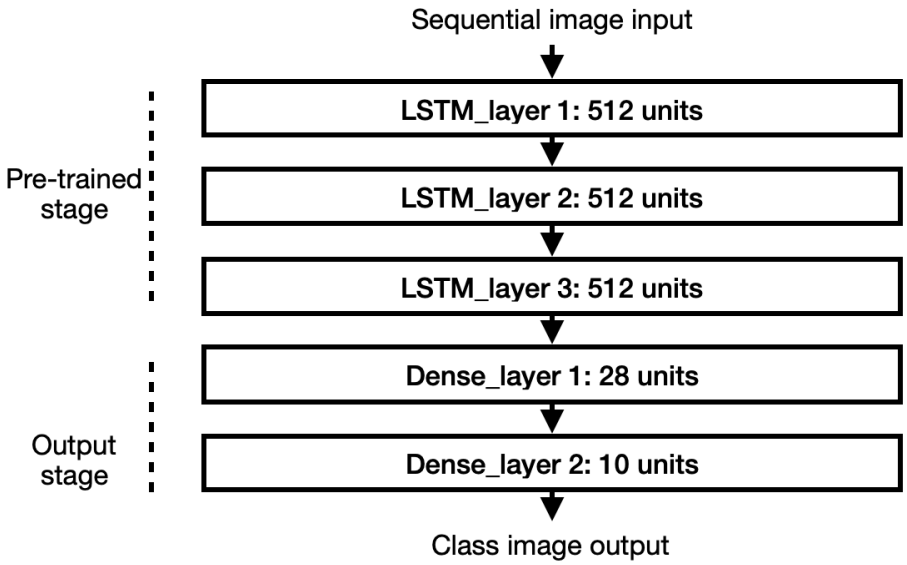


Fig. 4.4 Neural network architecture used for semi-supervised layer-wise and self-training layer-wise.

than the specified threshold of examples. For instance, if each class is expected to have a minimum of 100 examples, the iterative process ceases when one class fails to meet this requisite.

The self-training methodology is subjected to training using varying quantities of limited labeled examples, as specified in Table 4.4. The process involves employing the pre-training (or unlabeled) subset detailed in Table 4.3 to generate pseudo-labeled examples. Subsequently, model performance is assessed using the test subset for evaluation.

Self-training layer-wise. This method comprises two stages: pre-training and fine-tuning with self-training. The architectural configurations employed in each stage align with those used in the semi-supervised layer-wise approach and consistent hyperparameters (Fig. 4.4). Algorithm 2 and Fig. 4.2 provide a comprehensive outline of the learning process involving unlabeled and labeled datasets as specified in Table 4.3. Multiple experiments were conducted as indicated in Table 4.4. The selection process of pseudo-labeled examples, the quantity of these pseudo-labeled examples integrated with actual examples during self-training, and the termination criteria for the iterative self-training process all align with the self-training framework.

4.6 The Proposed Method Extended for Large Size Datasets

We previously introduced a framework for leveraging unlabeled data in scenarios with limited annotated datasets. The methods described in Section 4.5 were assessed using standard benchmark datasets like MNIST and Fashion. In this section, we expand our investigation to include the application of the previously developed machine learning approach to large datasets, with a specific focus on the Quickdraw dataset.

The primary goal of this methodology is to overcome challenges related to data pre-processing and training time when dealing with large datasets, such as the Quickdraw dataset. To achieve this goal, we have utilized the capabilities provided by the TensorFlow library.

For data pre-processing, TensorFlow offers a valuable tool: the *Dataset class* within its *data module*. This class is adept at handling substantial datasets efficiently, as it operates by streaming data rather than requiring the entire dataset to be loaded into memory. This approach effectively mitigates issues related to graphics processing unit (GPU) memory constraints. Additionally, we harnessed the capabilities of *TensorFlow Datasets* to streamline downloading and creating specific Dataset instances, such as the Quickdraw dataset [2].

We employed two tools to enhance the training process: TensorFlow's Multi-worker strategy and the linear-epoch gradual-warmup (Section 2.12) method. The former tool improves training speed by harnessing multiple GPUs across different servers, while the latter optimizes GPU memory using larger batch sizes.

To harness the computational power of multiple GPUs, working with larger models and/or larger batch sizes is typically necessary during the training process. While our Long Short-Term Memory (LSTM) model is not particularly large, we can leverage GPUs by significantly increasing the batch size. However, a larger batch size often leads to reduced model accuracy. The LEGW technique is employed for the LSTM architecture (and also applied to convolutional neural networks) to address this issue. For example, when provided with a set number of training examples and a baseline model configured with hyperparameters such as a batch size of 32, a learning rate of 0.001, and 1 warmup epoch, following the LEGW approach allows us to substantially increase the batch size (along with the hyperparameters) while maintaining the same level of performance and saving time when utilizing multiple GPUs.

Our research is conducted in a context where experiments involve varying quantities of labeled examples. The question arises: how can the LEGW method effectively manage this dynamic range of training examples? The answer to this question will be treated in the following subsections.

4.6.1 Linear-epoch Gradual-warmup and a Changing Number of Training Examples

Adapting the linear-epoch gradual-warmup (LEGW) method to manage a dynamic number of training examples is imperative for two primary reasons. Firstly, our experiments encompass a range of incremental numbers of a few labeled examples. Secondly, both self-training and our proposed method, self-training layer-wise, undergo training with varying training examples as pseudo-labeled data is incrementally introduced into the actual labeled examples during the training process.

Notably, our adaptation of the LEGW method started with supervised learning, followed by the integration of semi-supervised layer-wise, self-training, and self-training layer-wise approaches. Each of these methods posed unique challenges and necessitated distinct utilization strategies. Thus, we initiate our discussion with supervised learning.

4.6.2 Linear-epoch Gradual-warmup Method and Supervised Learning

Before adjusting learning rate (Lr), warmup epochs (We), and batch size (Bs), it is essential to fine-tune these hyperparameters in alignment with the number of training examples. However, our empirical observations indicate that consistently using a baseline configuration of 32 Bs, 0.001 Lr, and 1 We relative to the quantity of training examples yields optimal accuracy. The selection of these values is deliberate: the choice of a small batch size (e.g., 32) is motivated by the presence of a limited number of labeled examples (0.33% of the original training set, equivalent to 2,310 items in the Quickdraw dataset), while a 0.001 learning rate is a commonly adopted default value for initiating deep learning training. The determination to utilize 1 warmup epoch emerged from empirical experimentation, revealing its efficacy.

Subsequently, in the pursuit of employing a larger batch size, we executed hyperparameter scaling following the linear-epoch gradual-warmup (LEGW) principles, while retaining a constant warmup epoch of 1, as visually depicted in Fig. 4.5. The scaling process mirrors the approach detailed in Section 2.12.

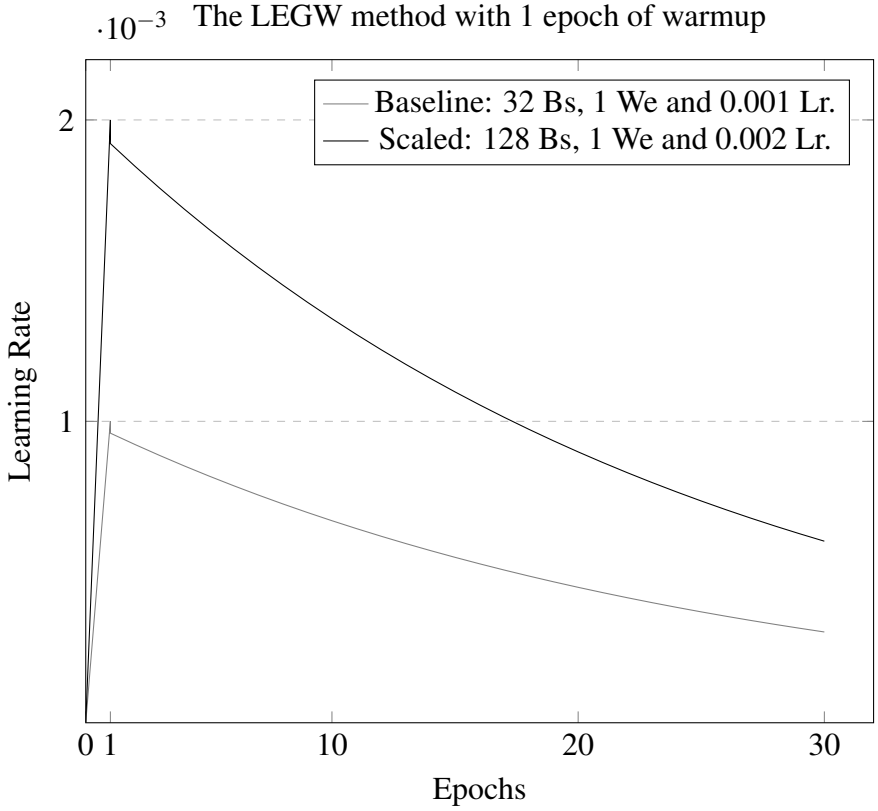


Fig. 4.5 The learning rate, both baseline and scaled, exhibits its behavior throughout training epochs. All hyperparameters are scaled except the warmup epochs.

Considering the progressive increase in the number of few labeled examples throughout our experiments, we must account for the possibility that a scaling factor of 4, as shown in Fig. 4.5, may become insufficient when dealing with a substantial amount of training data. We turn to the quantity of batches or steps within an epoch to determine the appropriate scaling factor. We have empirically established that a consistent 30 steps or batches yield the optimal accuracy. We can calculate a significantly larger scaling factor by utilizing this step count and the given quantity of training examples as parameters. This approach is applied when training examples exceed 5,000; otherwise, the baseline hyperparameters are retained. The procedure for computing the scaling *Factor* is detailed as follows.

$$BsFactor = \left\lfloor log_2(\frac{numberTrainingExamples}{steps}) \right\rfloor, \tag{4.5}$$

$$ScaledBs = 2^{BsFactor}, \tag{4.6}$$

$$Factor = \frac{ScaledBs}{32}, \tag{4.7}$$

Where $BsFactor$ is a rounded-down ($\lfloor\rfloor$) value, which is utilized to determine the scaled batch size ($ScaledBs$). This latter value is divided into 32 baseline batch size to get the $Factor$, by which the hyperparameters are scaled except the warmup epoch.

4.6.3 Linear-epoch Gradual-warmup Method and Semi-supervised Layer-wise Learning

This learning approach comprises two distinct phases: the pre-training layer-wise phase and the subsequent supervised fine-tuning phase. The linear-epoch gradual-warmup (LEGW) method is employed in different manners throughout each of these phases.

Given the substantial size of our dataset, utilizing a significant portion (83.33%) for the pre-training layer-wise phase, it is advisable to employ a large batch size for this unsupervised learning phase. Specifically, we use a batch size (Bs) of 2048, a warmup epoch (We) of 1, and a learning rate (Lr) of 0.001 as baseline values for training. A large batch size is justified by the ample pre-training dataset available. These hyperparameters remain unaltered to prevent potential out-of-memory (OOM) errors, and this batch size ensures a swift training process.

In contrast, when fine-tuning the pre-trained model with a defined number of few labeled examples through supervised learning, the following values serve as the baseline: 32 for Bs, 16 for We, and 0.001 for Lr. We must exceed 1 in this fine-tuning phase to facilitate effective model learning. Except for the Bs, these specified hyperparameters remain constant and are not subject to scaling or modification as part of the LEGW methodology.

The Bs is scaled using a calculated scaling $Factor$, as detailed in the preceding section.

4.6.4 Linear-epoch Gradual-warmup Method and Self-training

Self-training leverages the linear-epoch gradual-warmup (LEGW) method in a manner akin to supervised learning. In self-training, the neural network undergoes multiple training iterations, with the training dataset gradually expanding by including pseudo-labeled examples alongside the initially limited set of labeled examples. To work on this dynamic dataset size, the scaling method in Section 4.6.2 adaptively adjusts the batch size (Bs) and learning rate (Lr) while the warmup epoch (We) remains constant. This adaptive approach ensures efficient memory utilization as the dataset size increases.

Utilizing 30 steps with these hyperparameters results in an out-of-memory (OOM) error. For instance, starting with 7k few labeled examples, the combined count of labeled and pseudo-labeled instances can extend to approximately 280k, necessitating a batch size of 8,192, which exceeds memory capacity and leads to an OOM error. Nevertheless, employing 120 steps mitigates this issue, as even when training with a maximum of 700k instances (the most significant possible number of labeled and pseudo-labeled examples across various experiments with few labeled examples), the batch size remains manageable at 4,096. The steps parameter serves as a means to *manually tailor* the batch size according to available memory resources. In essence, larger step values correspond to smaller batch sizes.

4.6.5 Linear-epoch Gradual-warmup Method and Self-training Layer-wise

Our proposed method employs a scaling approach akin to the semi-supervised layer-wise method, encompassing two phases: pre-training layer-wise and self-training fine-tuning. In the pre-training phase, we apply the linear-epoch gradual-warmup (LEGW) approach consistent with previous descriptions, utilizing hyperparameters of 0.001 for learning rate (Lr), 1 for warmup epoch (We), and 2048 for batch size (Bs) for unsupervised pre-training. In the subsequent phase, which involves self-training fine-tuning, we continue to utilize the LEGW method, maintaining baseline values of 32 for Bs, 16 for We, and 0.001 for Lr, with these values remaining constant except for the Bs. To adjust the Bs, we employ the scaling $Factor$ computed using 120 steps to mitigate out-of-memory (OOM) errors during training, especially when working with increasing examples.

4.6.6 More Improvements for Self-training to Work with Large Datasets

In addition to the improvements in training time (multi-worker strategy) and the utilization of the linear-epoch gradual-warmup (LEGW) method, the self-training approach incorporates several enhancements, detailed as follows. Firstly, rather than pseudo-labeling the entire unlabeled dataset (comprising 583,310 examples for Quickdraw), we adopted a strategy of pseudo-labeling only a subset containing 50k examples. This selective pseudo-labeling approach significantly reduces the time required for label prediction during each iteration of the method. Secondly, from this subset of pseudo-labeled examples, a sub-sample of size $f(x)$ (equation 4.8) is extracted and subsequently integrated with the set of x actual examples (or pseudo-labeled and actual examples. The process of adding pseudo-labeled examples to actual examples is carried out following the principles outlined in equation 4.4, thus facilitating the effective management of extensive datasets ($20k$ for x $\in [100k, 1M)$).

$$f(x) = \begin{cases} 100 & \text{for x} \in [100, 1k) \\ 1k & \text{for x} \in [1k, 10k) \\ 10k & \text{for x} \in [10k, 100k) \\ 20k & \text{for x} \in [100k, 1M) \end{cases} \tag{4.8}$$

After selecting $f(x)$ pseudo-labeled examples from the 50k subset, this subset is consistently refilled with examples from the entire pool of pre-training data, ensuring continuous availability of 50k unlabeled examples for potential pseudo-labeling. Thirdly, the self-training process is terminated under specific conditions, including a substantial accuracy decline of more than 10% from the initial accuracy, depletion of the pre-training examples, or a pronounced imbalance in the pseudo-labeled examples. Importantly, these enhancements in the self-training methodology are similarly applied to the fine-tuning phase within the self-training layer-wise framework.

4.7 The Proposed Method and Augmented Datasets

In scenarios involving a limited number of labeled examples, data augmentation is a viable strategy for mitigating the challenges associated with neural network performance by augmenting the training dataset through various transformation techniques, also called data warping. This study employed a series of transformations on both medium and large datasets. Notably, the transformations included flipping, rotation, translation, and zooming in and zooming out. To illustrate, these transformations were applied to a representative class example, specifically class 7 within the MNIST dataset, as depicted in Fig. 4.6. The TensorFlow pre-processing module was employed to implement these transformations, each including a random component during application.

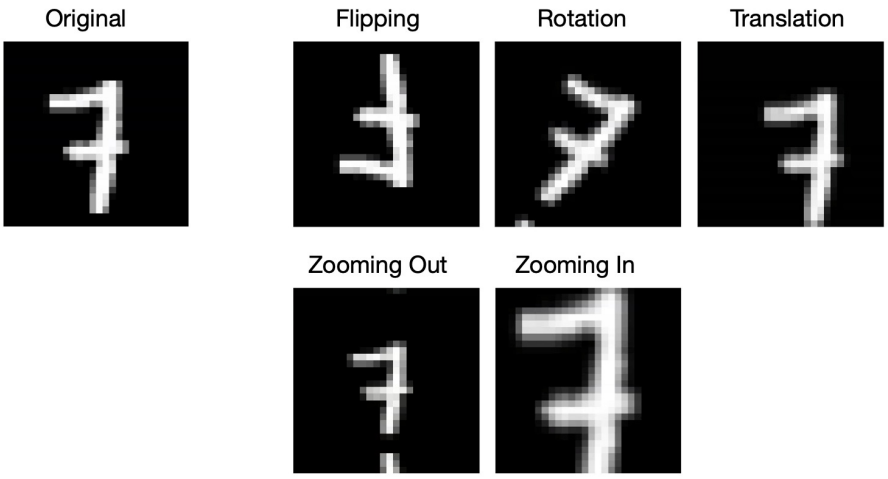


Fig. 4.6 Five transformations were applied to the original class 7 example to enhance diversity and increase the number of examples.

According to the methodology outlined in Fig. 4.6, for a dataset consisting of N examples, five transformations were applied to each example, resulting in a modified dataset of 5N examples. These modified examples were combined with the original dataset, resulting in an augmented dataset of 6N elements. This data augmentation process was applied to the datasets detailed in Table 4.3 (Section 4.5), specifically to the training subsets of the *Original subsets* column. In the case of the MNIST dataset, the 100% of augmented training examples were split into 83.33% for pre-training (300k examples) and 16.67% for training (60k examples), as presented in Table 4.5. The test subset remained unchanged. The same approach was employed for the FASHION and Quickdraw bitmap datasets.

In Section 4.5, various dataset sizes were employed to simulate scenarios with limited labeled examples for evaluating the proposed method and other relevant techniques. This section aims to assess these methods using the augmented versions of the datasets above. Specifically, we selected the 16.67% of the training subset from Table 4.5. We systematically reduced it to 0.33% to generate varying-sized datasets for experimental purposes, as Table 4.6 shows. Hence, this table represents the augmented counterpart of Table 4.4.

4.7.1 Neural Network Setup for Augmented Medium Dataset and Augmented Large Dataset

The neural network architecture and hyperparameter configuration for supervised, semi-supervised, self-training, and self-training layer-wise methods applied to the augmented medium datasets remain consistent with the descriptions outlined in Section 4.5 for medium

Table 4.5 The number and diversity of examples are augmented after applying the data warping to the datasets.

Dataset	Augmented subsets
MNIST (370k full)	300k Pre-training
	60k Training
	10k Test
FASHION (370k full)	300k Pre-training
	60k Training
	10k Test
Quickdraw bitmap (4,225,000 full)	3,499,860 Pre-training
	700,140 Training
	25k Test

Table 4.6 Experiments are structured according to the diverse dataset sizes available within the training subset.

AugMNIST	AugFASHION	AugQuickdraw	%
60012	60012	700140	16.67
54000	54000	630000	15.00
47988	47988	559860	13.33
42012	42012	490140	11.67
36000	36000	420000	10.00
29988	29988	349860	8.33
24012	24012	280140	6.67
18000	18000	210000	5.00
11988	11988	139860	3.33
6012	6012	70140	1.67
5400	5400	63000	1.50
4788	4788	55860	1.33
4212	4212	49140	1.17
3600	3600	42000	1.00
2988	2988	34860	0.83
2412	2412	28140	0.67
1800	1800	21000	0.50
1188	1188	13860	0.33

datasets. The key distinction lies in utilizing the *dataset class* from the TensorFlow data module to manage the increased data volume effectively.

For the augmented Quickdraw bitmap dataset, we adopt the approach discussed in Section 4.6 for large datasets to train neural networks using the four learning methods. The neural network architecture remains consistent with the design outlined for medium datasets in Section 4.5. However, the linear-epoch gradual-warmup technique, which assumes a crucial role when handling large augmented datasets, computes key hyperparameters, including batch size, learning rate, and warmup epochs. Furthermore, we leverage TensorFlow's multi-worker strategy to enhance training efficiency by harnessing the computational power of graphics processing units (GPUs) distributed across multiple servers.

4.8 Deep Semi-supervised Approach for Seismic Data Analysis

In this Section, we present a methodology designed to harness the potential of large unannotated seismic data, particularly in cases where the availability of labeled data obtained from borehole measurements is limited. Our approach estimates absolute acoustic impedance (AI) by integrating deep neural networks, specifically Long Short-Term Memory (LSTM) networks, with semi-supervised techniques, a method referred to as deep semi-supervised learning (DSSL). The DSSL model for AI prediction was trained using data extracted from pre-stack time migration (PSTM) seismic cubes. As Fig. 4.7 shows, the data preparation and pre-processing stages were critical for constructing the training dataset, as PSTM seismic cubes contain only raw, unprocessed data.

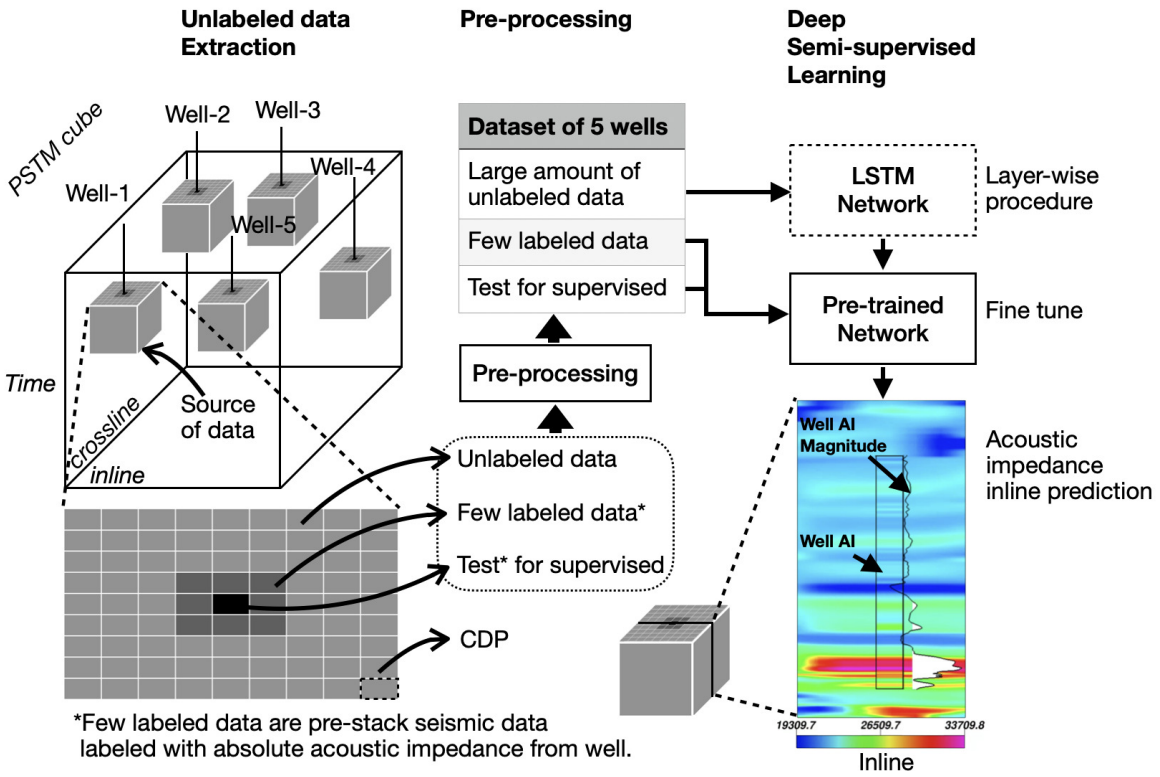


Fig. 4.7 Illustration of the process for extracting labeled and unlabeled data from the PSTM cube, dataset configuration, and the general input data fed into the neural network using the deep semi-supervised learning approach.

4.8.1 Seismic data preparation

The open-source software OpendTect was employed for the preparatory phase, integrating seismic and well data. This preparation predominantly entailed the well tie procedure, aligning well data with the corresponding pre-stack time migration (PSTM) seismic dataset.

The seismic data is structured as a volumetric cube within the Society of Exploration Geophysicists Y (SEG-Y) format, a standardized format for storing geophysical data. In parallel, the well-log data adopts the Log ASCII Standard (LAS) format, with each row corresponding to a specific depth value and associated with various data records. Additional well-related data, including the well track and time-depth transformation [106], are also available.

To facilitate the integration of these datasets into OpendTect, we initiated the importation of the SEG-Y file alongside the well-related data encompassing well-logs, well-tracks, and the time-depth transformation. After successfully integrating these datasets within the OpendTect platform, we embarked on a critical seismic interpretation task: the well-tie process.

In the process of well-tie, seismic data and well data are aligned, and beyond these data an initial wavelet is required. This wavelet undergoes convolution with a reflection coefficient, yielding a synthetic seismogram, subsequently matched with the seismic data. The determination of the reflection coefficient (R) involves mathematical operations on the acoustic impedance (Z) derived from bulk density ($RHOB$ or ρ) and wave velocity (V). The calculation of wave velocity is, in turn, based on sonic velocity measurements (DT) [15, 1].

$$V = \frac{10^6}{DT}, \tag{4.9}$$

$$Z = \rho V, \tag{4.10}$$

$$R = \frac{Z_2 - Z_1}{Z_2 + Z_1}, \tag{4.11}$$

where Z_2 is the acoustic impedance of the current line of data or formation, and Z_1 is the line above.

After the computation of these variables, the procedure advances to the squeezing and stretching phase. This step primarily involves optimizing the correlation between the synthetic seismogram and the seismic data. Throughout this iterative process, adjustments are made to the frequency content of the synthetic seismogram until it closely approximates that of the seismic data. After that, the acoustic impedance information, used as labels for seismic data near the borehole, was extracted in the time domain using OpendTect.

The process of aligning seismic data with well data was replicated for five drilled wells. For Well-1, the synchronization of these datasets is depicted in Fig. 4.8.

4.8.2 Training Data Pre-processing

For constructing input data for neural network training, the pre-stack time migration (PSTM) cube underwent a preprocessing stage. This process utilized the PSTM cube as the basis for feature extraction. Additionally, the PSTM cube was stored in the Society of Exploration Geophysicists Y (SEG-Y) file format, while the acoustic impedance data, used as the target, was stored in plain text format.

Fig. 4.9 shows the extraction of traces per common depth point (CDP) or gather, which consists of 40 traces. From these, 18 were selected as input features to optimize the neural network's performance. This selection was determined through experimentation with 6, 12, and 24 traces as input features.

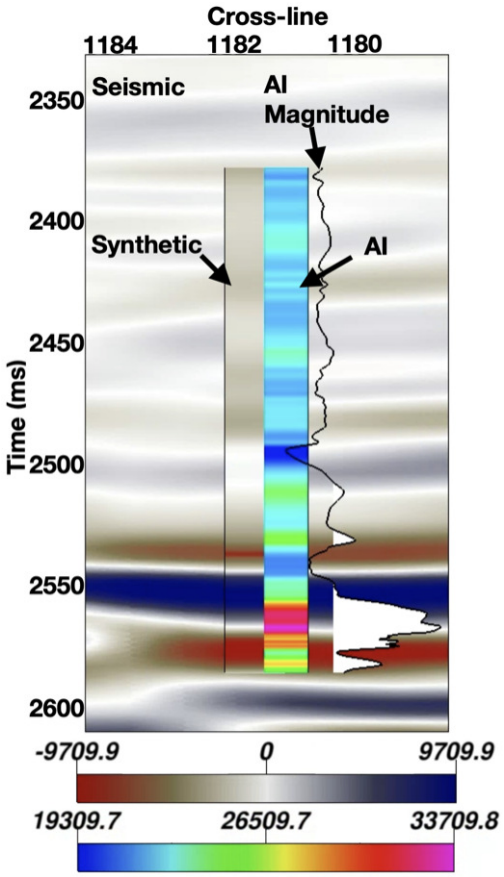


Fig. 4.8 Results of data preparation for Well-1, demonstrating proper alignment between seismic and well data after the squeezing and stretching procedure. Additionally, synthetic and AI data were derived from the well data.

The temporal range from which these traces were acquired remained consistent for both the acoustic impedance (AI) target and the input features. Following this, rescaling was applied to both the features and the target using the MaxAbsScaler function from the scikit-learn library, normalizing their values to the ranges [-1, 1] for the features and [0, 1] for the target.

A notable disparity in sample counts between the seismic traces and the AI data was observed, with the sample rate of the AI data exceeding that of the seismic traces. To achieve consistency in sample rates across both datasets, an interpolation technique was employed. Radial basis function interpolation, specifically employing the cubic function, was used to achieve this synchronization. This interpolation procedure was uniformly applied to all 18 traces within the CDP in Fig. 4.9.

Mere interpolation proved insufficient to facilitate effective learning within the neural network framework. Therefore, the down-sampling technique was strategically employed

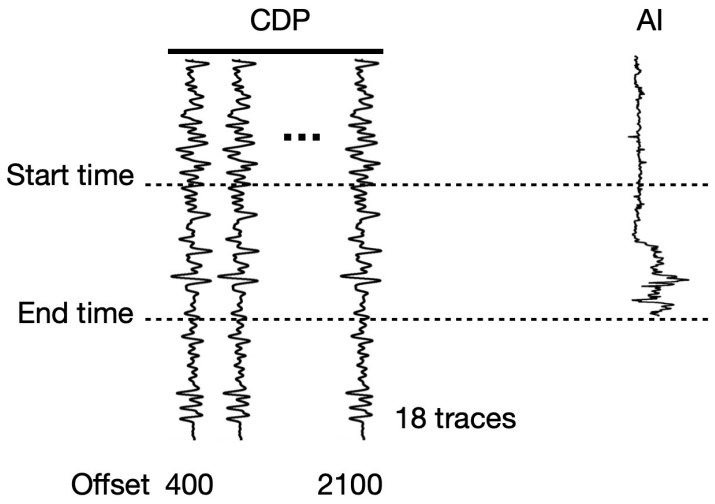


Fig. 4.9 A CDP instance consisting of the selected number of raw seismic traces, with the AI target represented as a single signal, along with the corresponding time range from which they are extracted.

to harmonize the sample rates between the seismic traces and the AI data. This procedure systematically removed a specified number of samples at regular intervals, effectively reducing the overall sample rate. In this case, the selected interval was determined through experimentation, ensuring that the data quality remained intact. After evaluating various configurations, it was determined that removing 8 samples resulted in the optimal sequence length. This approach enhanced the network's ability to discern and learn valuable patterns from the data.

Each dataset example for neural network input was meticulously constructed by extracting a window of features from the initial to the final time point. Fig. 4.10 visually illustrates this process, wherein an example is generated by windowing across 53 samples derived from the 18 traces. To establish the target for each example, a single sample was selected from the AI data, precisely positioned at the midpoint or the 27th position within the 53-sequence length. At this stage, the feature window advances by one sample to extract the next labeled example, continuing this process until it reaches the final time of the CDP. This sliding window technique is applied across various CDPs, generating the full set of labeled examples.

Following the aforementioned procedure, training and test examples were generated from CDPs, as illustrated by the cells in Fig. 4.11. Test examples were sourced from the borehole CDP, where the labels originated. These same labels were also used to annotate training examples generated from neighboring gathers. Sharing identical labels between the test and training examples proved advantageous, as it enabled the generation of additional training

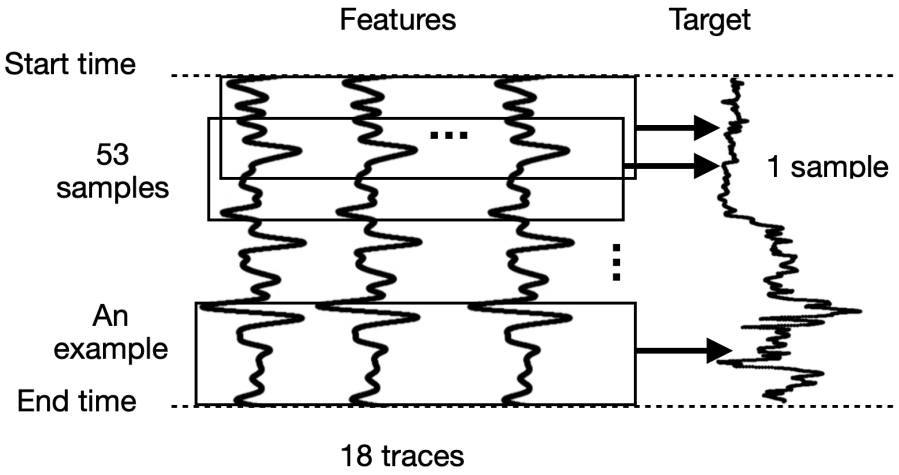


Fig. 4.10 Constructing labeled examples by selecting a window of features from the seismic traces and assigning the corresponding AI sample label.

data from nearby gathers. This approach was feasible because CDPs within the same vicinity exhibit similar patterns. Consequently, this strategy contributed to more effective supervised training by increasing the amount of labeled data available.

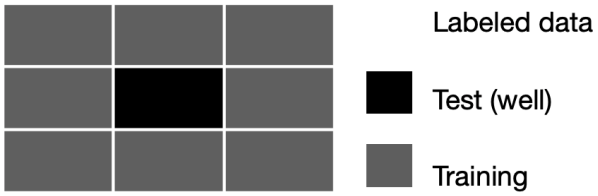


Fig. 4.11 Organization of CDPs from which labeled training and test data were extracted.

Acoustic impedance values are indicative of the types of materials found in the subsurface. For example, shale, water-bearing sands, and hydrocarbon-bearing sands exhibit acoustic impedance ranges of 24,500 to 27,500 (ft/s)(g/cc), 22,000 to 24,500 (ft/s)(g/cc), and 17,500 to 21,500 (ft/s)*(g/cc), respectively [33, 11]. While this classification is not applied in the current study, AI can serve as reference information for labeling seismic data, where AI values are used solely as sample data for labeling purposes.

The pre-processing steps for both semi-supervised and supervised learning share common operations, such as the number of windowed features, the interpolation technique used, and the number of traces extracted per CDP. However, well AI data was not utilized for examples intended for unsupervised training. Fig. 4.12 illustrates the number of CDPs used to generate unlabeled data. Of these CDPs, 65% of the data was used for training examples and 35% for test examples during unsupervised training. For the supervised learning stage, the labeled dataset was used.

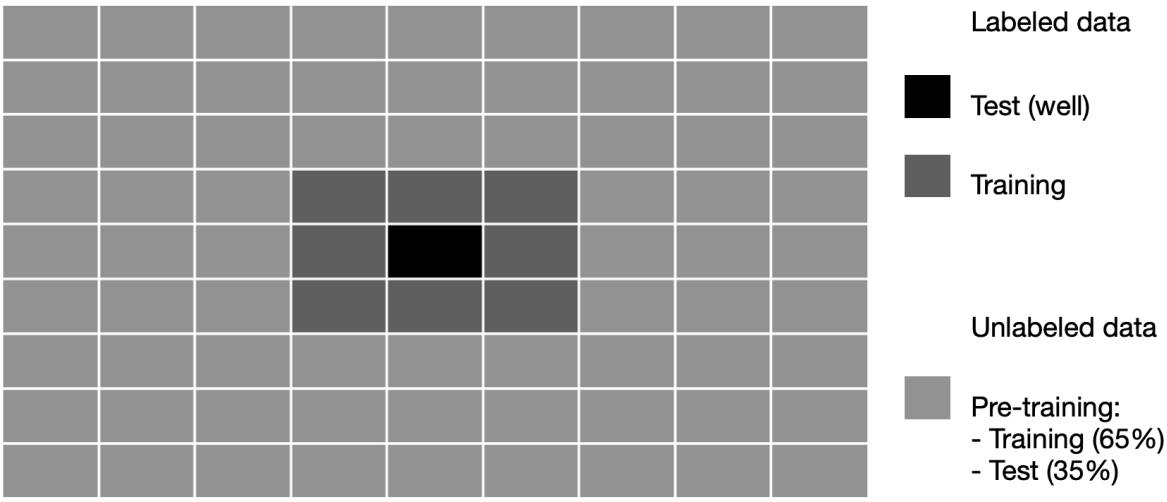


Fig. 4.12 Configuration of CDPs for unlabeled data extraction in relation to labeled data for use in semi-supervised learning.

Data from five boreholes were utilized to apply the previous methodology for constructing the training dataset, with examples generated based on the start and end times outlined in Table 4.7. Additionally, Table 4.8 provides an overview of the entire dataset organization, primarily designed for semi-supervised learning, including unlabeled data for unsupervised training, and labeled data for the supervised stage.

Table 4.7 For dataset construction, examples were generated from a specified temporal range within the PSTM cube.

Time (ms)	**Well-1**	**Well-2**	**Well-3**	**Well-4**	**Well-5**
start	2375	2408	2402	2582	2640
end	2584	2648	2691	2831	3013

Table 4.8 Comprehensive organization of the dataset primarily designed for semi-supervised learning.

Dataset	**Labeled**	**Unlabeled**
Pre-stack Seismic (164,100 Full)	10,800 training	98,280 training
	2,100 Test (well)	52,920 test

4.8.3 Configuration of the Long Short-Term Memory Neural Network

Neural Network Configuration for Supervised Learning

Fig. 4.13 depicts a recurrent neural network based on Long Short-Term Memory (LSTM) architecture for supervised learning, featuring the following configuration: the model consists of three neural network layers, each containing 1,024 units, with a dropout rate of 0.75 applied per layer. This LSTM network is followed by an output stage comprising two dense layers with 1,024 units and 1 unit, respectively. Notably, there was no prior knowledge regarding these hyperparameters; all were determined through experimentation with the constructed training dataset. The mean absolute error was utilized as the loss function, and the Adam optimization algorithm was employed for model compilation.

The linear-epoch gradual warmup method was employed to optimize hyperparameters, including batch size, warmup epochs, and learning rate, with baseline values set at 32, 1, and 0.001, respectively. Subsequently, a scaling factor of 2 was applied to these values, while the warmup epoch remained unscaled.

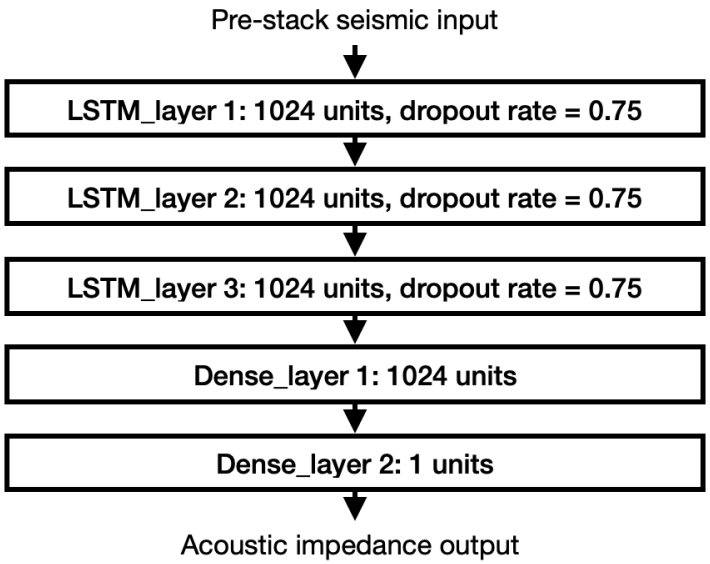


Fig. 4.13 Neural network architecture used for supervised learning.

Neural Network Configuration for Deep Semi-supervised Learning

For deep semi-supervised learning (DSSL), the architecture closely resembled the one previously described, with some notable distinctions. Similar to the configuration used in label-driven learning, the model architecture for unsupervised learning consisted of three LSTM neural network layers, each with 1,024 units and a dropout rate of 0.75. Additionally,

two dense layers with 1,024 and 18 units, respectively, were attached to the LSTM network as the output stage as shown in Fig. 4.14. To ensure optimal performance during the pre-training phase, validation loss was carefully monitored. The model was compiled using the mean squared error loss function and the Adam optimization algorithm. For the supervised model training stage, the same hyperparameters from the supervised learning setup were applied, and an additional layer with 1 unit was attached to the previously unsupervised architecture to accommodate supervised training.

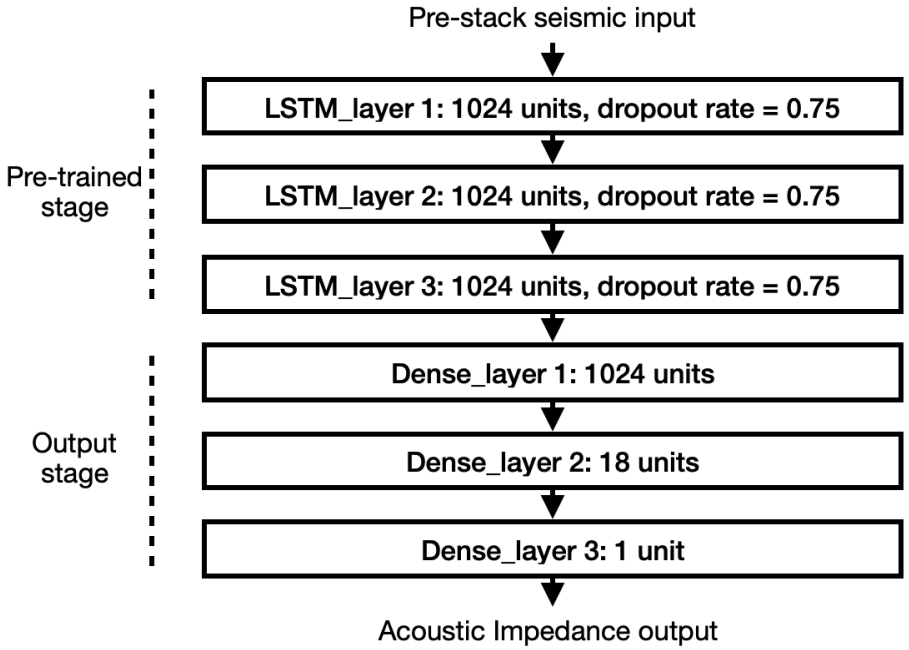


Fig. 4.14 Neural network architecture used for semi-supervised learning.

In the label-driven learning stage of the semi-supervised approach, hyperparameters were optimized using the linear-epoch gradual warmup method. In contrast, while the hyperparameters were not scaled during unsupervised model training, the linear-epoch gradual warmup significantly contributed to this stage, particularly with the warmup epoch parameter set to 1. This facilitated the use of baseline hyperparameters, including a large batch size of 2,048 and a learning rate of 0.001, enhancing model performance during training.

Training the neural network architecture with large volumes of unlabeled data required advanced techniques to reduce training time. To achieve this, the TensorFlow multi-worker strategy was employed to ensure optimal communication between two servers equipped with graphics processing units (GPUs). Additionally, GPU memory utilization was optimized using the linear-epoch gradual warmup method, allowing for efficient resource allocation and maximizing hardware performance during training.

4.8.4 Acoustic Impedance Estimation from Seismic Inlines

For the estimation of acoustic impedance (AI), the trained Long Short-Term Memory (LSTM) neural network was utilized to predict AI based on the input unlabeled data generated through the preprocessing techniques outlined in Section 4.8.2. Specifically, the model estimates acoustic impedance from pre-stack seismic data sourced from seven common depth points (CDPs) within a defined temporal range, with the central CDP corresponding to the well location. The acoustic impedance estimation was performed for five seismic inlines.

The neural networks employed for prediction were trained using supervised and deep semi-supervised approaches. For acoustic impedance prediction, the seismic data examples were systematically extracted on a CDP basis. Although the AI values were initially normalized for the training and prediction phases, the neural network outputs were subsequently rescaled to their original values utilizing the MaxAbsScaler function from the scikit-learn library.

Chapter 5

Experiments and Results on Benchmark Datasets and a Real-World Dataset

5.1 Experimental Setup

5.1.1 Datasets

This section introduces the datasets employed in this study. Firstly, MNIST, MNIST-C, notMNIST, FashionMNIST, and SLMNIST are utilized to assess LSTM performance under variations in sequence order. Secondly, MNIST and FashionMNIST datasets form the primary basis for developing our proposed method, while the Quickdraw bitmap dataset is used to evaluate the method's scalability for large datasets. Finally, Mexico's pre-stack seismic data is employed to address the challenge of limited supervised data within massive amounts of unlabeled data in real-world scenarios. Except for the seismic data, all datasets consist of grayscale images with dimensions of 28 x 28 pixels.

MNIST. The dataset consists of handwritten digits (0–9) and is divided into training and testing subsets, containing 60,000 and 10,000 images, respectively. Additionally, the dataset is categorized into 10 distinct classes, each corresponding to one digit.

MNIST-C. It is a modified version of the MNIST dataset, featuring alterations specifically related to glass blur corruption. The dataset consists of 60,000 images for training and 10,000 for testing and encompasses 10 classes, mirroring the structure of the MNIST dataset.

notMNIST. The dataset consists of images representing ten distinct types of letters from A to J in the English alphabet. Furthermore, it contains a total of 500,000 training examples

and 19,000 testing examples, thereby enhancing the original dataset's size and diversity. For this study, a subset of 60,000 images was randomly chosen for training, while 10,000 images were set aside for testing.

FashionMNIST. This dataset, referred to as FASHION, includes ten classes of images representing various articles from Zalando, including T-shirts and bags. It comprises 60,000, and 10,000 examples for training, and testing, respectively

SLMNIST (or the Sign Language MNIST). This dataset represents the letters of standard American Sign Language, excluding the letters J and Z. It contains 24 distinct letter types, with a total of 27,455 training examples and 7,172 testing examples.

Quickdraw bitmap. This dataset comprises 28x28 grayscale images featuring 345 drawing classes with 50 million examples. For our specific study, we selected 10 classes, with 700,000 examples for training and an additional 25,000 for testing.

Pre-stack seismic data from Mexico. The pre-stack time migration seismic cube, comprising 461 crosslines and 491 inlines, was acquired using specialized sensors over a 141.47 km² land area. The data spans from the surface to a subsurface depth corresponding to 1251 ms in the time domain. The spatial bin separation is 25 meters, while the seismic trace sampling rate was set at 4 ms. In addition to the seismic data, geophysical logs from five drilled wells are available. The raw data undergoes necessary steps of preparation and pre-processing for further analysis.

5.1.2 Long Short-Term Memory Classification under Changes in Sequences Order

The methodology outlined in Section 4.1 is implemented on the MNIST, MNIST-C, notMNIST, FashionMNIST, and SLMNIST datasets to evaluate the performance of Long Short-Term Memory (LSTM) models under variations in sequence order. These datasets underwent image shape transformation to introduce sequence length and order variations. The recurrent neural network architecture employed consisted of three layers, each comprising 512 units of LSTM. Additional analysis, including conditional independence tests and entropy analysis, was conducted on these modified datasets.

All datasets were standardized to achieve a zero mean and unit variance per image, resulting in values within the $[-1, 1]$ range. This scaling method was implemented to

facilitate the optimization of weights during the training process [77]. The constructed model utilizes TensorFlow (v2.2), the Keras API, and Python (v3.7), executed on an NVIDIA GeForce RTX 2080Ti GPU accelerator.

5.1.3 The Proposed Methodology for Medium and Large Datasets

The proposed method outlined in Section 4.4 to tackle challenges associated with a limited number of labeled examples is evaluated across three datasets, namely MNIST, FashionMNIST, and Quickdraw Bitmap. These datasets serve as the testing ground for three alternative methods, ensuring a comprehensive assessment of the proposed approach. The alternative methodologies under comparison include supervised learning, semi-supervised learning, and self-training, as detailed in Section 4.5.

The original dataset subsets underwent a series of transformations to have the pre-training, training, and test subsets (refer to Table 4.3). These subsets were utilized for the evaluation of machine learning methods. The training subset was systematically reduced to simulate scenarios with a limited number of labeled examples, resulting in various scenarios of annotated training datasets, as depicted in Table 4.4.

The four machine learning methods, including the proposed approach, are based on LSTM recurrent neural network architecture, employing similar hyperparameters to ensure equitable comparisons of outcomes, as detailed in Section 4.5.

In contrast to the evaluation methodologies applied to MNIST and FashionMNIST datasets, assessing methods using the Quickdraw Bitmap dataset necessitates a slight deviation due to its categorization as a large dataset. Managing extensive datasets requires enhanced capabilities for computational and memory resource management. To address this, we employed the linear-epoch gradual-warmup (LEGW) method to leverage larger batch sizes for optimal memory utilization. We implemented the multi-worker strategy to harness additional computational resources, specifically graphics processing unit (GPUs), across different servers, as detailed in Section 4.6. Furthermore, we integrated an augmentation data approach to enhance model performance and generalization, elucidated in Section 4.7.

5.1.4 Deep Semi-supervised Approach for Seismic Data Analysis

The pre-stack seismic dataset from Mexico, consisting of gathers with 40 traces each, was utilized as a real-world application of the proposed approach outlined in Section 4.8. Borehole data from five wells, as detailed in Section 5.1.1, were employed for constructing labeled examples.

The initial measurements obtained from the pre-stack time migration, in conjunction with borehole data, were utilized to construct the training dataset during the preparation and pre-processing phases outlined in Section 4.8. The resulting dataset consists of both annotated and unannotated subsets, as illustrated in Table 4.8.

In addition to the deep semi-supervised learning (DSSL) approach, which incorporates both a limited amount of annotated data and a substantial quantity of unannotated data, a supervised learning methodology was also employed for comparative purposes. Table 4.8 presents an overview of the dataset utilized for both approaches.

For the implementation of the DSSL model, the neural network hyperparameters specified in Section 4.8.3 were utilized, which also describes the model configuration for supervised learning.

5.2 Long Short-Term Memory Classification: An Approach Addressing Variations in Sequence Length and Order

We present the results of evaluating the Long Short-Term Memory (LSTM) architecture using datasets with feature vectors organized in various configurations.

Multiple LSTM models were assessed to identify the optimal architecture, which consists of three neural network layers, each containing 512 units. The evaluation utilized the SLMNIST dataset as a reference, with the sequence length determined by $M \times N$, as explained in Section 4.1. Various configurations were explored, combining units of 512, 256, 128, 64, and 32, with the number of layers ranging from 1 to 6, resulting in the evaluation of 30 distinct models. For these experiments, both the horizontal order and the $M \times N$ image shape were employed for model assessment. Consequently, the LSTM architecture with three layers of 512 units each achieved an optimal accuracy of 90.53

Having selected the optimal architecture, we conducted experiments using five datasets with varying pixel arrangements: shape and order. The primary outcomes of our evaluations are presented in Table 5.1, yielding the following observations:

- Undoubtedly, sequences with the shape (28,28) comprising 28 features demonstrated the highest number of maximum accuracies, precisely six experiments, evenly distributed for vertical and horizontal sequence order.
- Four instances achieved the highest accuracies with images formatted as (2,392), primarily utilizing the sequence order extracted through the spiral methodology.

- In general, the performances achieved using the vertical and horizontal sequence order approaches surpass those obtained with the spiral method.
- Overall, the highest performances are likely to occur for sequences with lengths exceeding (28,28), compared to those with shorter lengths.

Table 5.1 Neural network performance is evaluated based on sequence length and order for each dataset, with the highest performances highlighted.

Shape	MNIST			MNIST-C			notMNIST		
(2,392)	98.16	98.19	**98.16**	97.45	97.36	97.19	94.74	94.55	**94.52**
(4,196)	97.41	98.50	97.94	97.33	97.01	**97.40**	94.81	94.59	94.08
(7,112)	98.52	98.43	**98.16**	97.96	**97.89**	97.05	95.38	95.43	93.81
(8,98)	98.52	98.04	97.63	96.84	97.09	97.20	94.24	94.58	94.12
(14,56)	98.83	98.34	97.89	**98.14**	97.74	96.53	95.37	**95.86**	93.17
(16,49)	98.30	98.09	97.76	97.28	96.96	96.81	93.91	94.09	93.13
(28,28)	**98.95**	**98.71**	97.55	98.10	**97.89**	96.43	**95.54**	95.60	92.40
(49,16)	98.56	97.43	96.69	97.80	95.85	95.67	93.17	93.00	91.60
(56,14)	98.86	98.08	96.90	97.83	97.34	95.33	93.98	94.29	90.95
(98,8)	98.32	97.36	95.07	97.46	96.72	94.00	93.04	92.57	90.63
(112,7)	98.35	96.97	94.36	97.77	95.84	92.84	93.10	92.66	89.53
(196,4)	98.23	97.59	94.79	97.42	95.65	93.40	90.89	93.56	90.56
Order Sorts	H	V	S	H	V	S	H	V	S

Table 5.1 Continued: for the two last datasets.

Shape	FashionMNIST			SLMNIST		
(2,392)	90.02	**89.58**	89.72	79.94	79.38	**79.45**
(4,196)	**90.20**	89.43	**89.94**	82.57	82.65	76.05
(7,112)	89.26	88.56	88.80	85.57	87.55	71.42
(8,98)	89.35	89.27	88.69	77.24	71.53	71.57
(14,56)	90.09	88.43	88.38	84.66	89.46	72.64
(16,49)	88.75	88.06	87.63	65.96	70.30	74.58
(28,28)	89.65	88.88	87.71	**90.53**	**92.28**	68.38
(49,16)	88.57	87.20	87.05	85.30	83.81	63.23
(56,14)	89.36	88.71	87.09	80.83	85.46	61.89
(98,8)	88.25	87.07	86.23	88.39	78.08	66.47
(112,7)	88.85	88.51	85.92	73.58	84.68	52.30
(196,4)	88.87	88.06	84.16	73.93	77.76	57.82
Order Sorts	H	V	S	H	V	S

We analyzed the accuracies presented in Table 5.1 using boxplot charts, as shown in Fig. 5.1. Based on the medians and maximum values observed, the H-order method applied to MNIST and MNIST-C datasets outperforms the vertical (V) and spiral (S) approaches.

Furthermore, the spiral (S) method consistently yields the lowest performance. A similar pattern is observed in the FASHION dataset, though with overall lower performance levels.

The notMNIST dataset, as illustrated in Fig. 5.1, exhibits varying outcomes, similar to those observed in the SLMNIST dataset. Specifically, the vertical (V) method demonstrates a slightly higher median performance compared to the horizontal (H) approach. Despite a significant difference in overall accuracy—approximately 95% for one dataset and 85% for the other—both datasets follow a consistent performance trend. In contrast, the spiral (S) method yielded the minimal median performance across both datasets.

In contrast, as evident from Fig. 5.1, the H order's accuracies exhibit lower overall dispersion, whereas the S order displays greater dispersion. This suggests that working with the horizontal order yields increased reliability, characterized by superior median accuracies and reduced variability compared to the other orders.

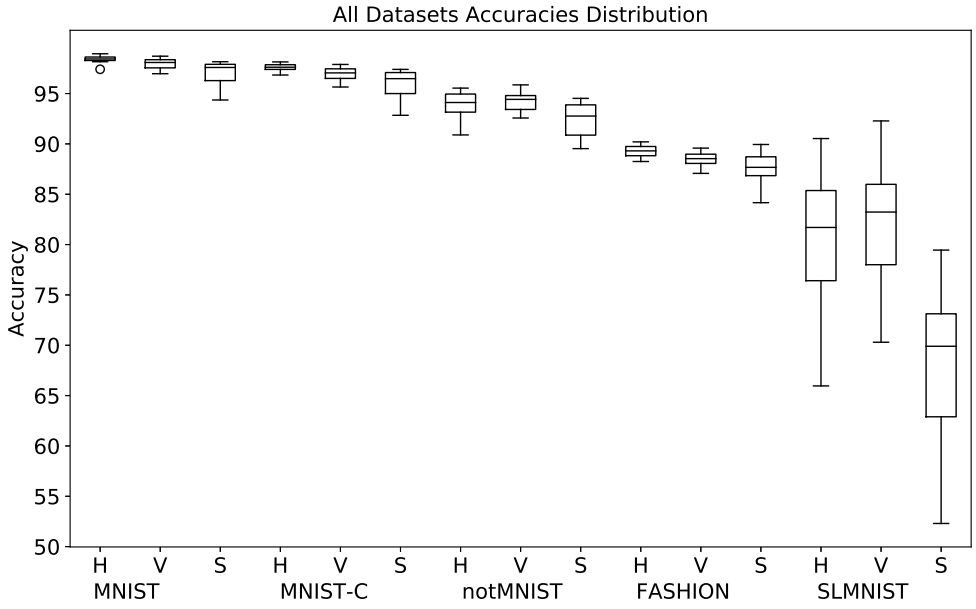


Fig. 5.1 Neural network performance for different types of order modifications.

Fig. 5.2 illustrates the mean epochs number required to obtain the performances reported in Table 5.1. First, the SLMNIST dataset required over 200 epochs for convergence. Second, both the horizontal and vertical methods required approximately 100 epochs for MNIST-C dataset, while the spiral approach converged in fewer than 25 epochs. Third, around 25 epochs were necessary for the FASHION, notMNIST, and MNIST datasets.

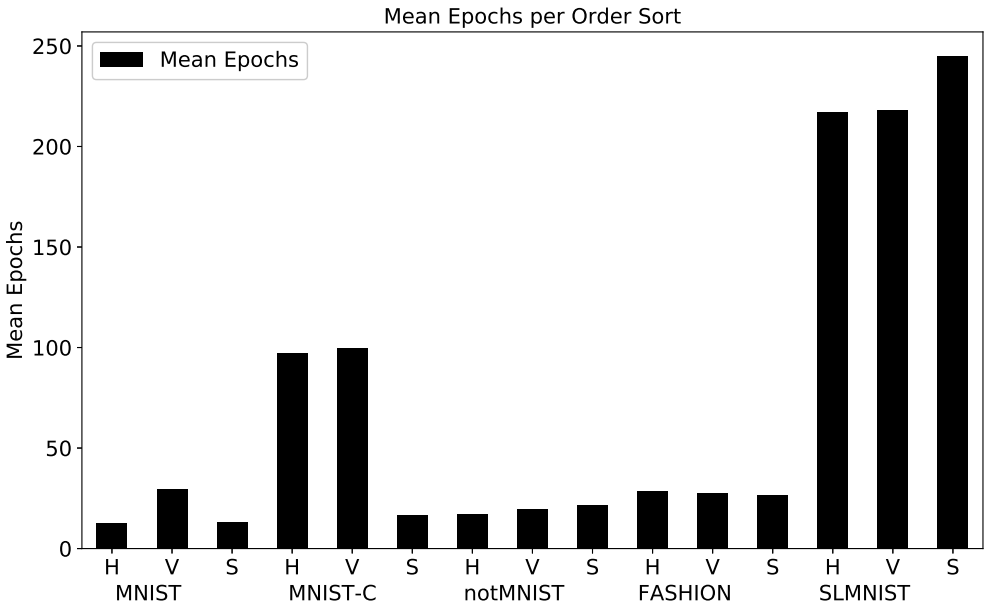


Fig. 5.2 The average number of epochs necessary to achieve optimal model performance across various types of order alterations.

An alternative perspective on epochs is based on the variance in sequence length, as presented in Table 5.1. After analyzing the performances across datasets for each row and considering the number of epochs, Fig. 5.3 was generated. In this figure, normalized mean values, obtained using the min-max scaling method, are depicted. The epochs are standardized within the range of 45 to 177, while the accuracies fall within the range of 88% to 92.5%. Both epochs and accuracies are presented per sequence length. Furthermore, the figure presents a comparative analysis of epochs and accuracy, indicating that images in the format (14,56) were learned more quickly, achieving a relatively high performance of 90%. Conversely, although images in the format (196,4) required the highest number of epochs, they exhibited the minimal performance.

Table 5.2 depicts image shapes exhibiting maximum or minimum training epochs when the neural network processes. In the context of the MNIST dataset, it is evident that image class 5 serves as an illustrative example. Specifically, when this class is represented with dimensions of (14,56), it effectively replicates its features, albeit at a reduced resolution, when organized in horizontal and vertical arrangements. Conversely, these features are less apparent when the class is arranged in a spiral pattern. However, if the image class is configured with dimensions of (196,4), the features become less visually and spatially distinct; this happens for the three methodologies, namely the spiral, vertical, and horizontal configurations.

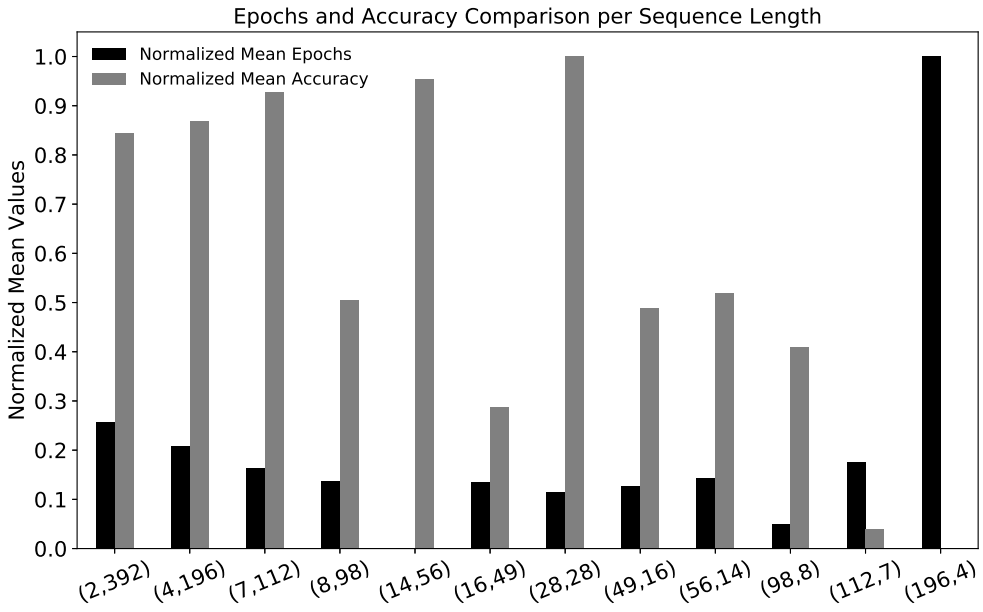


Fig. 5.3 The relationship between accuracy and the number of epochs in relation to sequence length for training the neural network.

Table 5.2 Sequence lengths of (14x56) and (196x4) were utilized, with the former requiring less time for training and the latter taking more time.

Shape	MNIST(for example)		
14×56			
196×4			
Order Sorts	H	V	S

5.3 Experimental Results for Conditional Independence Test

Table 5.3 shows the dependence index (DI) results, which yield several noteworthy observations. Firstly, a decrease in the number of variables N consistently leads to a decline in the dependence index across all datasets. However, it is essential to note that these variations in the DI do not correspond proportionally to changes in accuracy, as evidenced in Table

5.1. Secondly, the information conveyed by Table 5.3 is visually summarized in Fig. 5.4, which employs boxplot charts to depict the similarity of median lines among each dataset's H, V, and S arrangements. Conversely, when examining the accuracy chart in Fig. 5.1, the boxplots median lines are dissimilar among the horizontal (H), vertical (V), and spiral (S) approaches across all datasets, as they lie outside the neighboring boxplots. In the analysis of the accuracy and DI charts, there is a lack of consistency in the median lines, highlighting the absence of a direct correlation. Lastly, many boxplots in Fig. 5.4 exhibit outliers, particularly in response to significant variations in the dependence index. However, these outliers are less prominent in the accuracy data displayed in Fig. 5.1.

Table 5.3 Distribution of the dependence index as a function of sequence lengths and order for each dataset.

Shape	MNIST			MNIST-C			notMNIST		
(2,392)	2588.57	2284.74	9104.79	4763.58	4377.89	14221.11	23084.16	25730.64	29847.78
(4,196)	2441.18	2592.74	5010.51	3410.92	4687.81	6682.01	7193.16	8321.07	6925.10
(7,112)	1003.30	1192.69	1825.33	1079.35	1886.42	2378.47	2736.26	3009.62	2128.19
(8,98)	947.29	1009.38	1628.29	882.62	1304.18	2085.20	1954.37	2164.55	1554.87
(14,56)	290.88	307.38	534.44	214.47	436.37	686.80	800.04	695.20	515.75
(16,49)	296.72	282.90	443.09	195.84	342.96	597.21	523.20	488.31	443.21
(28,28)	68.16	66.59	147.50	44.88	117.78	186.04	218.75	151.69	195.28
(49,16)	57.48	58.61	70.20	50.54	59.87	81.78	74.65	66.09	92.13
(56,14)	43.51	49.25	57.91	40.97	49.47	65.73	55.96	63.75	74.41
(98,8)	23.34	24.52	26.90	25.38	30.43	29.08	26.56	31.24	32.67
(112,7)	19.09	20.53	21.99	20.91	25.07	23.16	22.55	27.61	25.79
(196,4)	8.40	8.55	8.98	7.96	9.31	8.61	8.83	9.71	9.53
Order	H	V	S	H	V	S	H	V	S

Table 5.3 Continued: for the two last datasets.

Shape	FashionMNIST			SLMNIST		
(2,392)	24446.24	23584.04	24448.73	44546.30	49307.73	40557.46
(4,196)	8820.22	10268.31	8063.70	11890.63	14382.26	9533.88
(7,112)	3244.85	3847.89	4320.24	3881.56	4652.22	2411.29
(8,98)	2333.17	2742.65	3203.95	2839.19	3312.38	1718.03
(14,56)	776.37	925.88	953.90	918.86	1042.19	583.16
(16,49)	552.54	709.64	748.42	662.50	751.61	484.34
(28,28)	199.34	224.79	250.89	202.58	229.18	261.73
(49,16)	84.97	100.68	121.51	76.29	90.66	114.91
(56,14)	74.49	96.26	93.12	71.98	83.92	90.93
(98,8)	32.41	33.39	35.47	29.86	33.63	35.22
(112,7)	24.61	28.22	27.31	26.10	29.33	26.98
(196,4)	9.10	9.59	9.54	9.36	10.28	9.62
Order	H	V	S	H	V	S

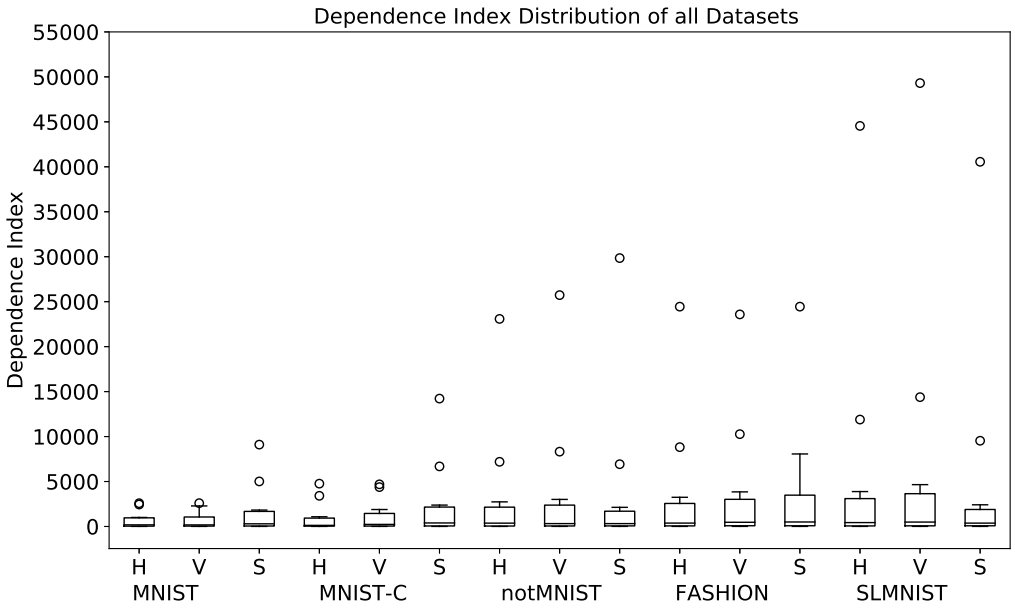


Fig. 5.4 The dependence index results for each dataset are presented in accordance with the changes in sequence order.

5.4 Entropy Results for Black-and-white Images

Table 5.4 presents the entropy values for individual dataset, calculated using the entropy-per-image approach applied to the training set examples.

Table 5.4 Average entropy of all training examples, computed on a per-image basis.

Dataset	Entropy
MNIST	1.60
MNIST-C	3.52
notMNIST	3.68
FASHION	4.12
SLMNIST	6.72

A correlation between Table 5.4 and Figure 5.1, which displays dataset accuracies, can be observed. For instance, MNIST exhibits lower entropy than SLMNIST; correspondingly, MNIST achieves higher accuracies than SLMNIST. This suggests that lower entropy is associated with improved accuracy.

The entropy per image row results for individual dataset are presented as boxplots in Fig. 5.5. A comparative analysis of accuracy and entropy, illustrated in Fig. 5.1 and Fig. 5.5, respectively, reveals the subsequent observations: (A) For the MNIST, MNIST-C, and

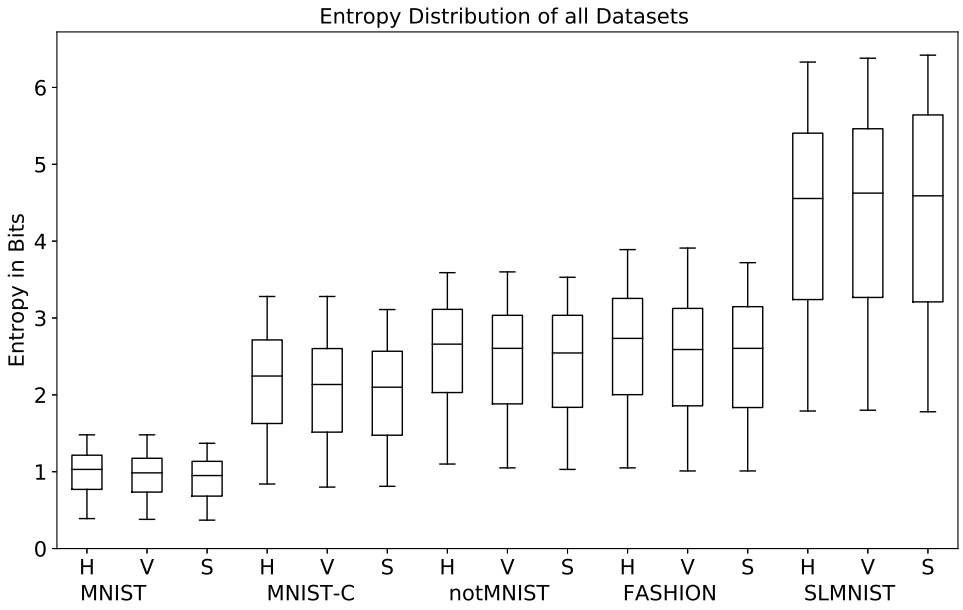
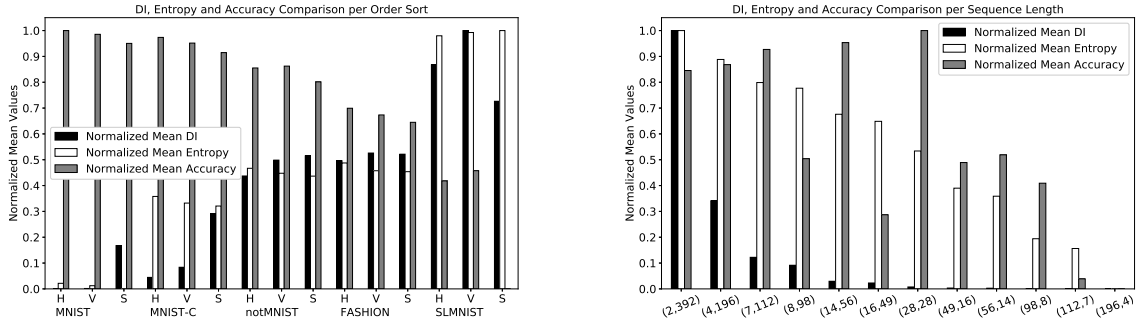


Fig. 5.5 The entropy results for each dataset are presented in relation to the changes in sequence order.

FASHION datasets, the H order exhibits less dispersion in the accuracy chart compared to the entropy chart. (B) In the case of the vertical (V) and spiral (S) methods, the entropy values remain relatively consistent, while their accuracy results differ significantly. This suggests that entropy and accuracy may not exhibit a straightforward correlation under different sequence orders. (C) There appears to be an inverse relationship between entropy and accuracy within each dataset; accuracy tends to decrease as entropy increases.

Finally, Fig. 5.6a and Fig. 5.6b present the analysis of accuracy, DI, and entropy. The min-max normalization approach was applied to rescale the data shown in the figures. Consequently, rescaled accuracy, DI, and entropy are depicted in Fig. 5.6a within the context of order sorting, while Fig. 5.6b illustrates the same analysis with respect to sequence length.

Several key observations can be made from Fig. 5.6a: (A) Entropy and DI increase across the datasets from MNIST to SLMNIST, while accuracy decreases. (B) For the vertical (V) and spiral (S) methods, changes in entropy and accuracy are generally proportional, exhibiting similar trends in most cases. However, this proportionality is not observed in the V sequence order for notMNIST and the S sequence order for SLMNIST, where a disproportion occurs. (C) Proportionality between DI and accuracy is evident in the V sequence order for notMNIST and SLMNIST, as well as in the S sequence order for SLMNIST. However, a lack of proportionality is noted in the V and S sequence orders across most datasets. (D)



(a) (b)

Fig. 5.6 The relationship among DI, entropy, and performance is presented as a function of changes in sequence order (a) and sequence length (b).

These observations indicate that DI consistently demonstrates an inverse proportionality to performance, except for the three exceptional cases. Additionally, in most datasets, the horizontal (H), vertical (V), and spiral (S) sequence orders exhibit similar entropy values, which tend to increase alongside accuracy.

Fig. 5.6b reveals notable trends. The DI and entropy maintain a proportional relationship for each sequence length. However, the performance trends follow an inverse pattern. Instead, the accuracy results are closely tied to specific sequence lengths. Notably, using the image format (28,28) as a basis for comparison, longer sequences on the left generally correspond to higher accuracies, though some exceptions are observed. In comparison, shorter sequences on the right side correspond to lower accuracies.

5.5 Semi-supervised Methods with Medium Size Datasets

In this section, we present the experimental outcomes obtained from the application of our proposed method and other techniques, which are described in Section 4.5, to two benchmark datasets, MNIST and FASHION, under various conditions specified in Table 4.4. Each experiment is conducted with a limited number of labeled examples, and this process is repeated ten times to ensure robustness. The results from these repetitions are analyzed, and the maximum accuracy achieved is reported.

Supervised. Our initial experimentation involves the application of the supervised learning method to the two standard datasets while varying the number of labeled examples to observe the method's behavior. Figure 5.7 illustrates the accuracy achieved through supervised learning on MNIST and FASHION datasets as the number of labeled training examples is systematically reduced, ranging from 16.67% (10k examples) to 0.33% (200 examples). As

expected, the accuracy of the supervised method decreases for both datasets as the number of training examples diminishes.

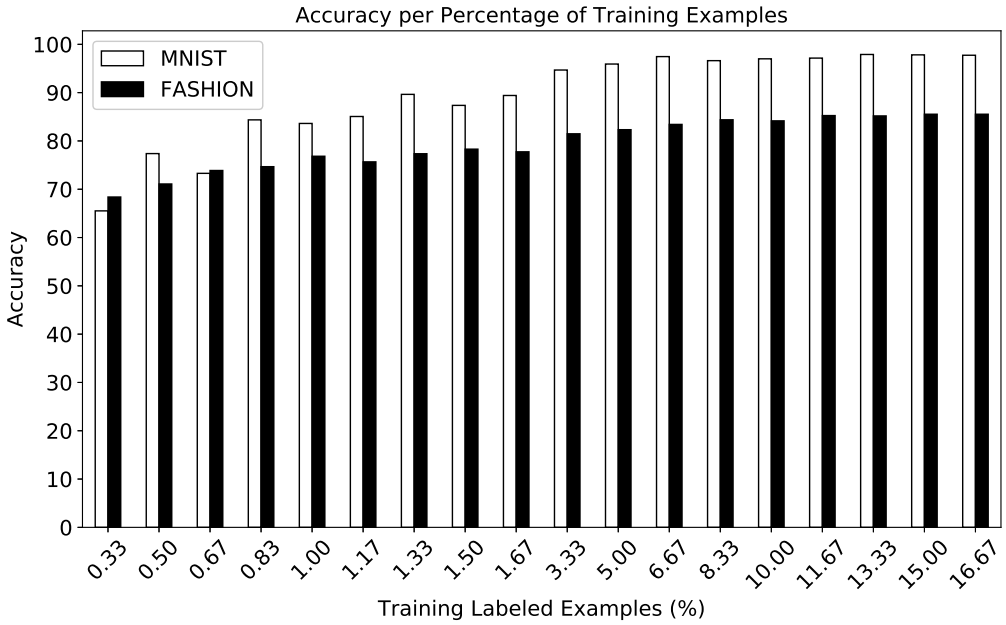


Fig. 5.7 The accuracies of supervised learning with the MNIST and FASHION datasets exhibit a gradual decline as the number of training examples decreases.

Supervised and semi-supervised layer-wise. One approach to address the decreasing accuracy as the number of examples diminishes is to employ the semi-supervised layer-wise method, which can mitigate this decline by leveraging unlabeled examples in the unsupervised phase. Fig. 5.8 illustrates the accuracy comparison between supervised learning and the semi-supervised layer-wise method using the MNIST dataset. The latter consistently outperforms the former, exhibiting significantly higher accuracy between 0.33% and 1.67% of training examples and maintaining a slight advantage from 3.33% to 16.67%. While the gradual reduction of labeled examples does affect accuracy, the overall trend indicates improvement with the semi-supervised layer-wise approach.

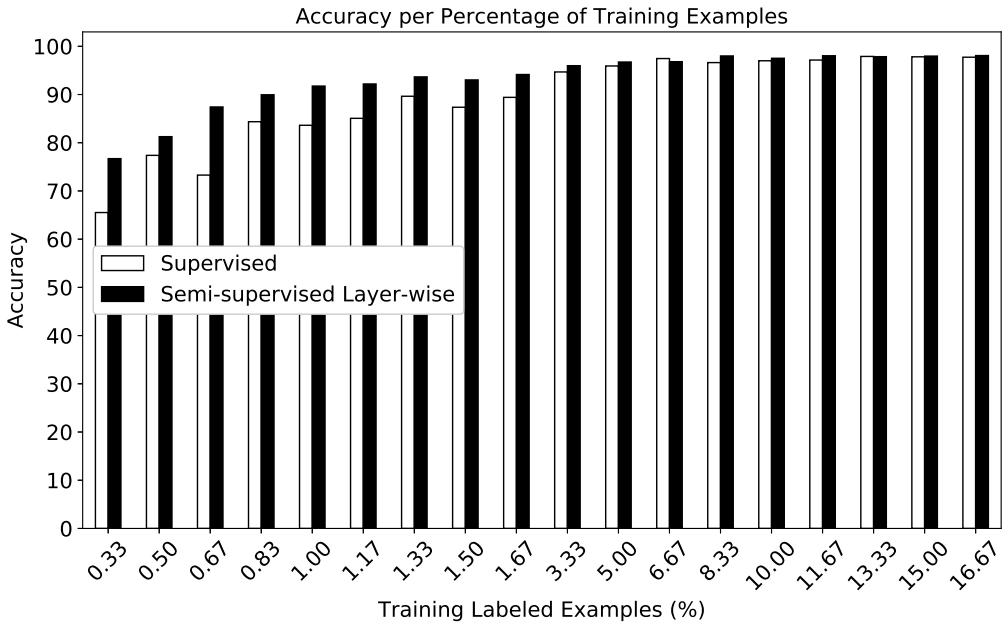


Fig. 5.8 The accuracies achieved by the semi-supervised layer-wise approach surpass those attained by the supervised method when applied to the MNIST dataset.

Supervised and self-training. We compared the outcomes achieved through supervised learning and self-training, the latter being an alternative approach to enhance accuracy in scenarios with limited labeled examples. As illustrated in Fig. 5.9, it is evident that the self-training method exhibits a substantial improvement over supervised learning, akin to the performance enhancement observed with the semi-supervised layer-wise approach. Both self-training and semi-supervised layer-wise methods fall within the domain of semi-supervised learning. This noteworthy improvement is particularly pronounced in the 0.33%–1.67% range of training examples and exhibits marginal yet consistent enhancements between 3.33% and 16.67%.

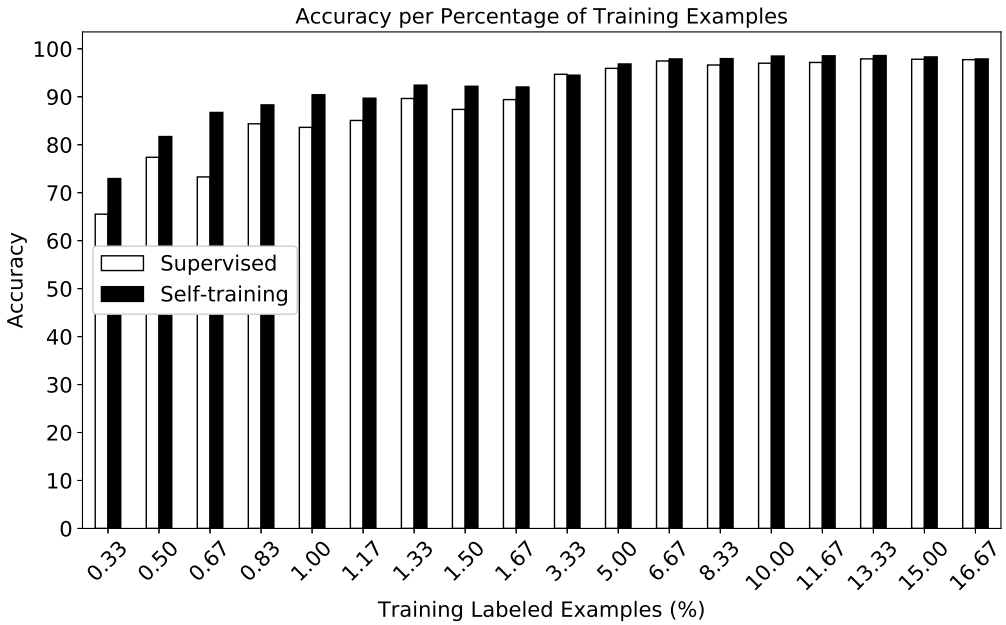


Fig. 5.9 The accuracies reached through self-training outperform those obtained with the supervised method on the MNIST dataset.

Self-training layer-wise. The figures above illustrate the utilization of unlabeled data by semi-supervised methods to enhance accuracy. With our self-training layer-wise method, we aim to investigate its potential to increase this metric further, considering its combination of the self-training and greedy layer-wise capacities, both of which leverage unlabeled examples.

Fig. 5.10 illustrates the performance reached when unlabeled data is employed by the self-training layer-wise method, leveraging it in a combined capacity of layer-wise procedure and self-training. Specifically, Fig. 5.10 demonstrates that our novel self-training layer-wise method consistently outperforms other methods in the 0.33%–3.33% range, except 1.67%, and slightly surpasses them between 5.00% and 16.67%, when evaluated with the MNIST dataset. Remarkably, our method achieves higher accuracies when the percentage of training examples is equal to or less than 3.33%.

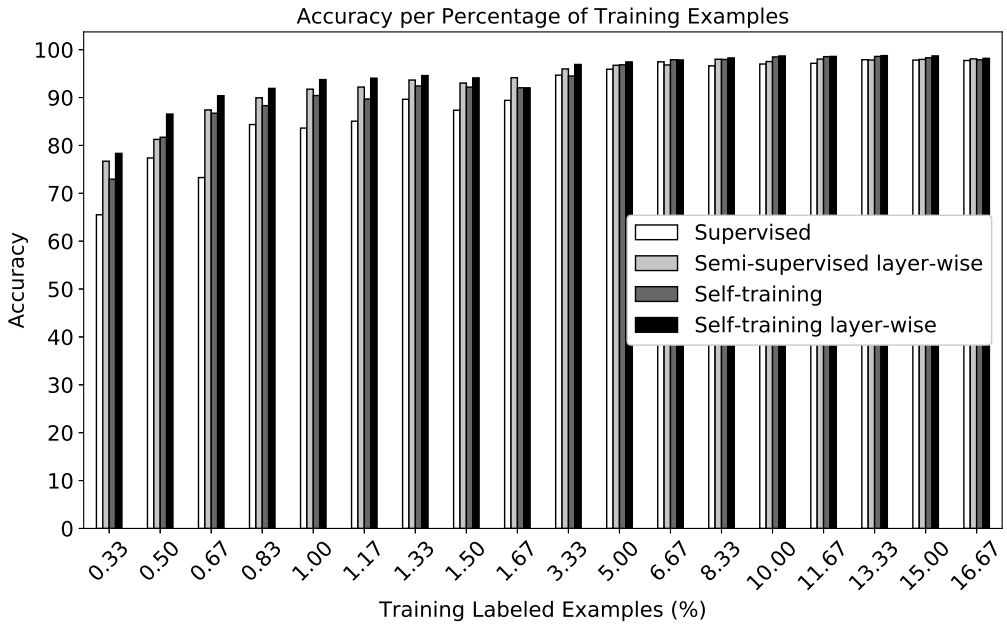


Fig. 5.10 The accuracies attained by the self-training layer-wise method outperform those of the other methods when applied to the MNIST dataset.

Our method and FASHION dataset. So far, the performance of the proposed method with the MNIST dataset exhibits significant superiority over the other three methods. Now, let's examine the performance when the FASHION dataset is utilized.

Examining the chart for the FASHION dataset, depicted in Fig. 5.11, we observe a slight improvement in the performance of the self-training layer-wise method compared to the other methods. This improvement is evident in the ranges of labeled examples percentages spanning from 0.67% to 0.83%, 1.17% to 1.50%, and 5.00% to 13.33%. While not surpassing the performance achieved with the MNIST dataset, our method still attains superior accuracies in scenarios where the percentage of labeled examples is below 3.33%.

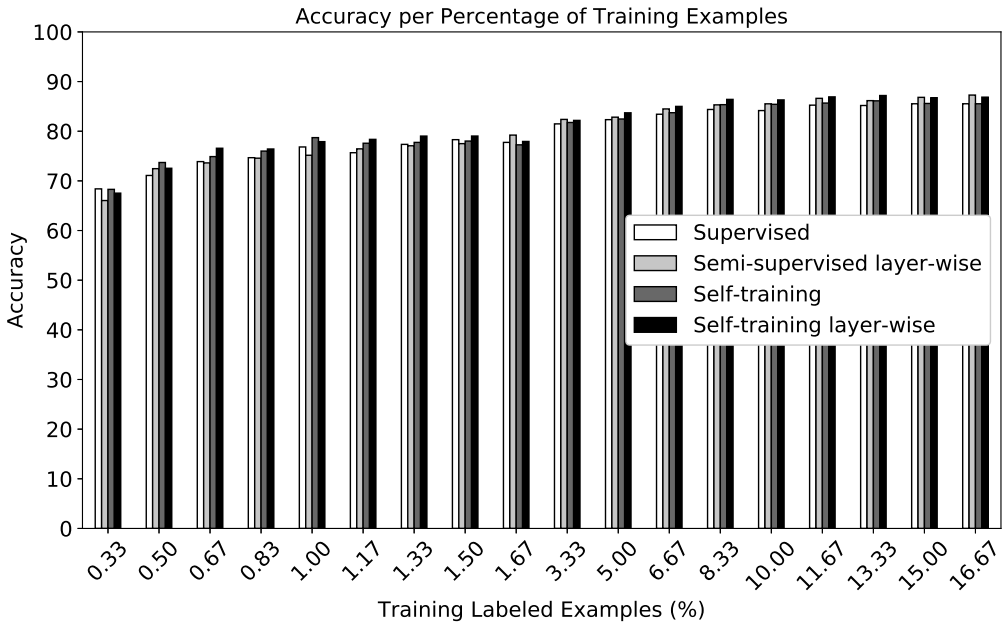


Fig. 5.11 Most of the time, the self-training layer-wise method achieves superior accuracies compared to the other methods when applied to the FASHION dataset.

5.6 The Proposed Method Extended for Large Size Datasets

Adhering to the approach designed for extensive datasets as elucidated in Section 4.6, we have obtained results for the Quickdraw dataset using the proposed method in conjunction with other methods. The maximum performance among ten experiments is considered. Yet, before delving into these results, we conduct an in-depth analysis of the behavior of the adapted linear-epoch gradual-warmup (LEGW) method within each learning approach.

As the experiments encompass various percentages of training examples, the LEGW method adjusts the batch size (Bs) for each specific method. Fig. 5.12 illustrates (b) the scaling of the batch size and (c) the learning rate (Lr) through the utilization of the scaling factor. The parameters determining this scaling factor are (a) the number of steps and the quantity of training examples. Notably, (d) the warmup epochs (We) remain unaffected by the scaling process, although its value changes depending on the learning method.

Fig. 5.12a displays the stability of the steps parameter for semi-supervised layer-wise, self-training, and self-training layer-wise methods. The former method consistently employs 30 steps and the two last 120 steps throughout the varying percentages of training examples or the number of examples. In contrast, the supervised learning method exhibits a more dynamic approach to the steps parameter. It utilizes 30 steps up to 8.33% of the dataset, then transitions to 70 steps at 10.00%, and then maintains 120 steps for the remaining percentages.

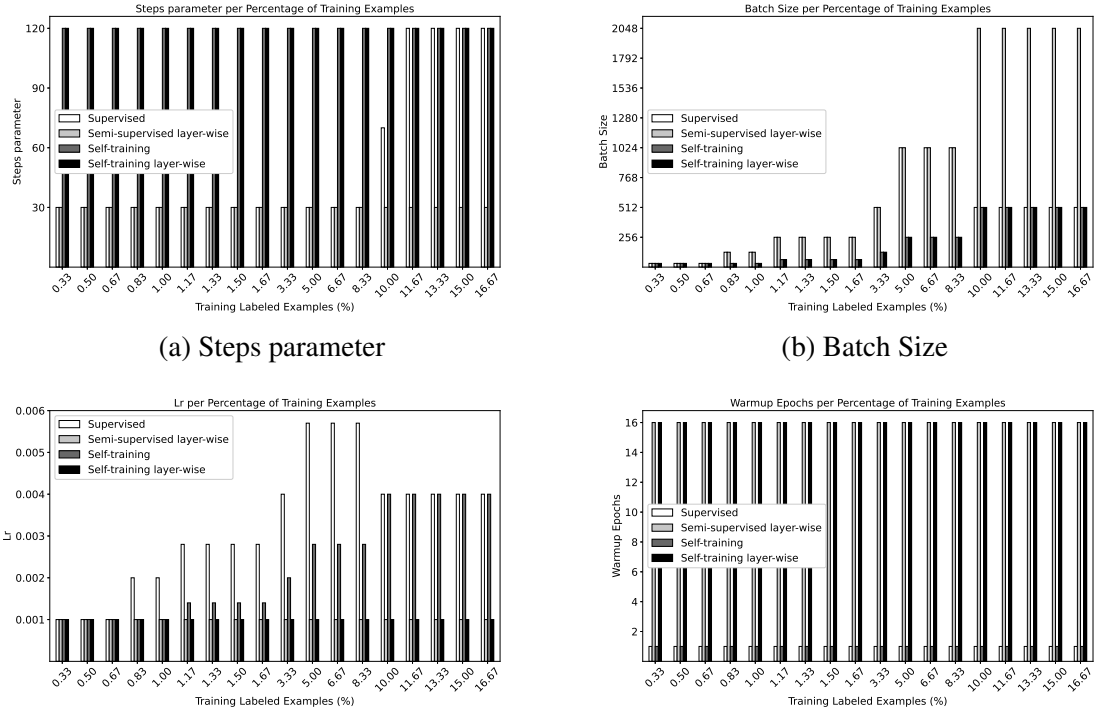
The increment in the steps parameter is *implemented manually* to decrease the batch size, as a larger batch size would hinder the learning process given the provided number of examples. Fig. 5.12b illustrates the reduction of batch size to 512 in the range of $10.00\% - 16.67\%$, where the steps parameter is increased (Fig. 5.12a), especially for supervised learning. Maintaining a consistent steps parameter offers the advantage of automated batch size computation, eliminating the need for manual adjustments.

As illustrated in Fig. 5.12b, the batch size scaling is closely tied to the steps parameter. In the 5.00%–8.33% range, both the supervised and semi-supervised layer-wise methods effectively operate with a fixed batch size of 1024. However, the latter method makes a notable shift by increasing its batch size to 2048 between the 10.00% and 16.67% training example percentages. On the other hand, the self-training and self-training layer-wise methods maintain a consistent batch size, which seems smaller compared to the semi-supervised layer-wise approach. Notably, in the case of the self-training layer-wise method, the initial batch size of 512 at the 10.00% training example percentage serves as the starting point in the first iteration of the learning process. In the subsequent iterations, this batch size progressively increases to 2048 due to the inclusion of pseudo-labeled examples, which is not illustrated by the figure.

Fig. 5.12c illustrates the utilization of the Lr by the various methods across the range of experiments. Like the batch size, the learning rate is determined based on the steps parameter using the scaling factor. Notably, this scaling factor is exclusively applied to the supervised and self-training methods. The supervised method employs a higher learning rate than the self-training method within the 0.83% to 8.33% range. However, both methods converge to the same learning rate in the 10.00% to 16.67% range. Conversely, the learning rate remains constant throughout the experiment variations for the semi-supervised layer-wise and self-training layer-wise methods, as the scaling factor does not affect it.

Finally, it is essential to note that the warmup epochs remain unaffected by the scaling factor. It has been determined through experimentation that employing a warmup epoch value of 1 consistently yields the highest accuracy for both the supervised and self-training methods. In contrast, the semi-supervised layer-wise and self-training layer-wise methods exhibit optimal results with 16 warmup epochs, as indicated in Fig. 5.12d.

Fig. 5.13 provides a performance analysis across an increasing number of labeled examples. Supervised learning consistently exhibits the lowest performance among these examples. A competition arises between semi-supervised layer-wise and self-training methods, with the former outperforming in the first half of a few labeled examples and the latter surpassing it in the the second half. Notably, our proposed self-training layer-wise method consistently outperforms all other methods.



(a) Steps parameter

(b) Batch Size

(c) Lr

(d) Warmup Epochs

Fig. 5.12 (a) The steps and the number of training examples are the parameters used to determine the appropriate scaling factor. Subsequently, this factor is employed to compute the proper (b) batch size and (c) Lr as required for each method across a few labeled examples of the Quickdraw dataset. Notably, (d) the warmup epochs remain consistent throughout.

Our proposed method consistently outperforms the other methods from 0.50% to 5.00%, corresponding to scenarios with the smallest percentages of available examples. In the 0.33%, the proposed method exhibits only a marginal lead over the semi-supervised layer-wise method. The technique maintains solid performance in the broader range of 6.67% to 16.67%. Overall, it achieves accuracy below 80% at 0.33% and reaches approximately 90% at 16.67%.

5.7 Number of Repetitions of Self-Training and Self-Training Layer-Wise for Medium and Large Datasets

The self-training and self-training layer-wise methods involve multiple training iterations, with each iteration being terminated based on predetermined stopping conditions for medium and large datasets. We conducted ten experiments for each of the few labeled examples,

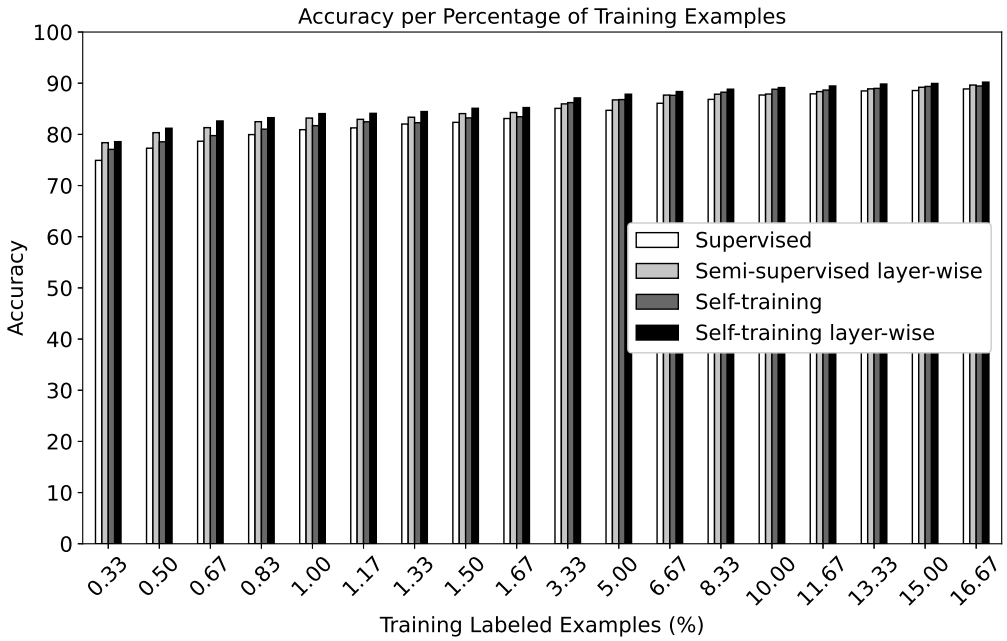


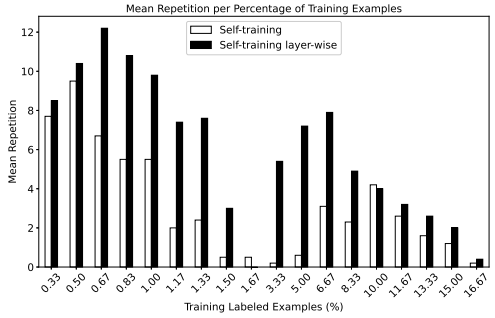
Fig. 5.13 Accuracy comparison of self-training layer-wise and other methods with Quickdraw dataset. Our method outperforms the other methods.

and each experiment was concluded based on these stopping conditions. In Fig. 5.14, we present the results regarding the mean number of iterations for each method using the MNIST, FASHION, and Quickdraw datasets. This number of iterations represents the point at which we achieve maximum accuracy, beyond which the training could continue but is eventually halted.

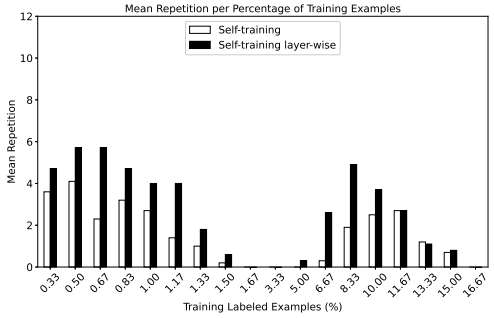
At first glance, in Fig. 5.14a and Fig. 5.14b, both self-training and self-training layer-wise exhibit a similar pattern in the number of training iterations. The first half of a few labeled examples involves more iterations, while the second half sees fewer repetitions. This reduction in iterations becomes noticeable after reaching 1.67% for MNIST and 3.33% for FASHION datasets. It's worth noting that self-training layer-wise often surpasses self-training in terms of the number of iterations, especially for the first half of the datasets, where the percentage of training examples is the lowest. The number of iterations in this initial phase is significant for self-training layer-wise, as it correlates with the accuracies it achieves, described in Section 5.5 for medium datasets.

In contrast, the pattern observed for the Quickdraw dataset differs, as shown in Fig. 5.14c. Despite the method's lower number of repetitions compared to self-training up to the 1.33% mark, it consistently achieves higher accuracy across varying numbers of training examples. This phenomenon may be attributed to the pre-training phase, which leverages a substantial portion of unlabeled examples, precisely 83.33% of the large dataset. Additionally, it is

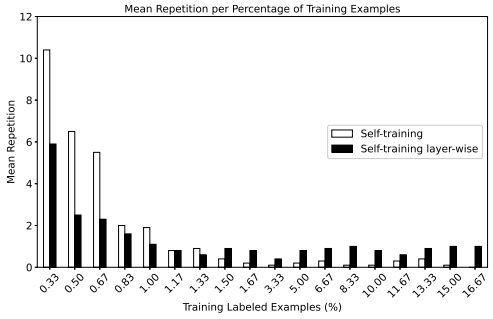
noticeable that the number of repetitions is higher when dealing with smaller percentages of examples, and as the percentage of examples increases, the number of repetitions decreases.



(a) MNIST



(b) FASHION



(c) Quickdraw

Fig. 5.14 Average training iteration count for self-training and self-training layer-wise methods across MNIST, FASHION, and Quickdraw datasets.

5.8 The Proposed Method and State-of-the-art Methods

The primary objective of this section is to review related works that have employed recurrent neural networks for analyzing sequential data within the framework of semi-supervised learning. An initial focus was placed on the MNIST dataset as a benchmark for evaluating our proposed method. Notably, the dataset was reinterpreted as sequential data, representing images as a series of pixels organized in feature vectors, rather than as conventional two-dimensional arrays or grids of pixels. This reinterpretation changes the processing approach, as sequential data inherently involves an ordered structure, which may correspond to a temporal sequence (e.g. time series) or, in other contexts, to a non-temporal sequence such as that found in genetic data.

Several studies have explored this alternative approach to analyzing the MNIST dataset using recurrent networks for supervised learning. For instance, Kaziha [52] conducted a

comparative analysis of the accuracy, optimal model size, and complexity (or computational cost) between Long Short-Term Memory (LSTM) networks and convolutional neural networks (CNNs) using the MNIST dataset. The findings indicated that while the LSTM network required greater complexity and larger model size, it achieved similar accuracy to the CNN, with results of 99.22% and 99.45%, respectively. In another study [34], we employed an LSTM network to assess the impact of variations in sequence order and length on the recurrent neural network performance using the MNIST dataset. This study yielded accuracies of 98.71% and 98.16% when the sequences were reorganized differently from the conventional 28x28 pixel format. Furthermore, several other works, as detailed in Section 3.1, have demonstrated improved LSTM performance on the MNIST dataset. These favorable outcomes suggest that LSTM networks are capable of learning the dependencies among pixel sequences in MNIST images. This sequential perspective on the MNIST dataset may enhance methods for analyzing other types of sequential data, such as genetic, speech, and seismic data.

Two lines of research are closely related to our project, although they differ in certain key aspects. The first line involves semi-supervised learning using CNNs with the MNIST dataset [57, 18, 63]. These studies differ from ours in that they assume independence among training examples, which is the fundamental characteristic when training with CNNs. The second line of research focuses on semi-supervised learning that accounts for long-range dependencies by employing sequential learning on datasets such as MNIST [113] and CIFAR-10 [70] with LSTM networks. However, these studies do not approach the datasets as partially labeled; they do not split the data into a small labeled subset and a larger unlabeled subset. Instead, all training examples are utilized for unsupervised learning and subsequent fine-tuning. A similar approach is applied to multivariate time series forecasting, with semi-supervised learning employing greedy layer-wise pre-training, where datasets such as the Capital Bike Sharing dataset and the PM2.5 Air Quality dataset in China were used.

The most closely related work to our project is the semi-supervised DeepHeart method, which is based on LSTM networks. This method predicts medical conditions such as diabetes, high cholesterol, and other conditions by pretraining on a substantial set of unlabeled sensor data using a sequence autoencoder approach, followed by fine-tuning on varying proportions of labeled data (5%, 10%, 20%, 50%, 70%, and 100%) to examine the effects of pretraining on unlabeled data.

To the best of our knowledge, no prior work has explored semi-supervised learning using a partially labeled sequential MNIST dataset with LSTM networks. Therefore, our approach represents a novel direction of investigation. For assessing the proposed method, three comparative approaches were implemented: supervised learning, semi-supervised learning

based on the greedy layer-wise approach, and a self-training method, as detailed in Section 4.5. The results of these comparisons are presented in Section 5.5, specifically in Fig. 5.10 for the MNIST dataset, and in Fig. 5.11 for the FASHION dataset. Additionally, the proposed method was tested on a larger dataset, the Quickdraw dataset, with the comparative results shown in Section 5.6, Fig. 5.13.

5.9 The Proposed Method and Augmented Datasets

Following the methodology outlined in Section 4.7, we augmented the MNIST, FASHION, and Quickdraw datasets, as presented in Table 4.5. These augmented datasets were then employed to conduct experiments involving a different dataset size with varying amounts of labeled data, as detailed in Table 4.6. Within the scope of these experiments, we evaluated the performance of our proposed self-training layer-wise method compared to other pertinent learning techniques using the augmented datasets. Each experiment was repeated ten times, and the highest achieved accuracy was reported. Initially, we compare the results of the self-training layer-wise method between the augmented datasets and their non-augmented counterparts. Subsequently, we provide a comparative analysis of the results obtained by all methods when using the augmented datasets exclusively.

Fig. 5.15 presents the accuracy outcomes of the proposed method trained on the MNIST and the Augmented MNIST datasets. Notably, the chart reveals that the technique achieved higher performance with the Augmented MNIST dataset, surpassing the results obtained with the standard MNIST dataset during the initial portion of training labeled examples. However, as the percentage of the training dataset increased, the accuracy results for both datasets converged, demonstrating similarity in performance during the latter stages of training.

Regarding the Augmented MNIST dataset, the method demonstrated an accuracy of approximately 95% with only 0.33% of labeled examples, and this accuracy increased to 97% with 1.67% of labeled examples. Subsequently, the method's performance exhibited modest improvements, reaching 98% between 3.33% and 16.67% labeled examples. Conversely, when applied to the standard MNIST dataset, the method achieved 78% accuracy with 0.33% labeled examples, then significantly improved to 95% at 1.33%, followed by a slight decrease to 92% at 1.67%. Nonetheless, within the range of 3.33% to 16.67% labeled examples, the method exhibited a slight increase in performance, reaching accuracy levels of 97% to 98%.

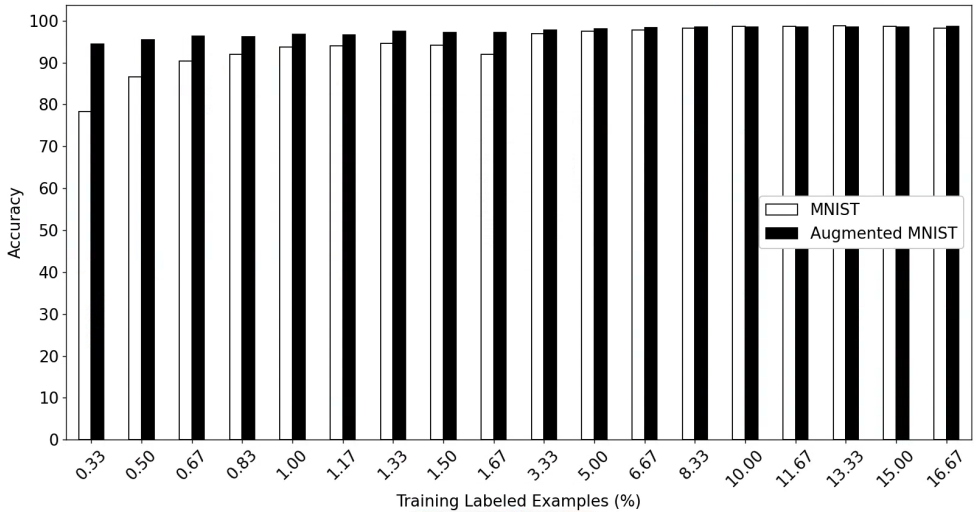


Fig. 5.15 Accuracy results of the self-training layer-wise method applied to both the MNIST and Augmented MNIST datasets.

The performance of the proposed method when applied to the FASHION and Augmented FASHION datasets is presented in Figure 5.16. The method consistently exhibited higher accuracy with the Augmented FASHION dataset than with the original one, irrespective of the percentage of labeled examples used for training. A detailed analysis of the Augmented FASHION dataset accuracy revealed that it started at 72% with 0.33% of the labeled dataset size and gradually increased to 81% at 1.50%. Although there was a slight decline to just below 81% at 1.67%, the accuracy continuously improved, reaching 90% when 16.67% of the dataset was labeled. In contrast, the accuracy results for the FASHION dataset began at slightly above 67% with 0.33% of the labeled dataset size, then rose to 79% at 1.50%. Despite a minor drop to 78% at 1.67%, it reached 87% with 16.67% of labeled examples.

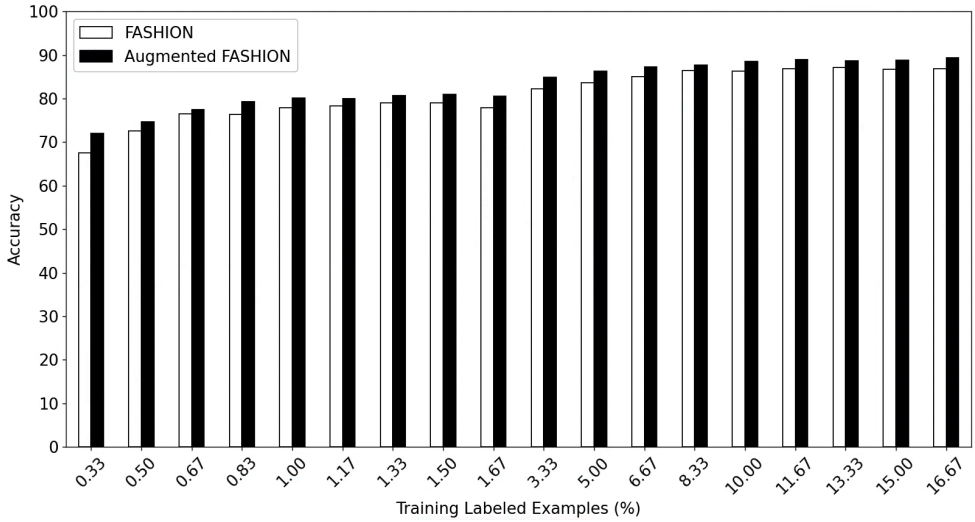


Fig. 5.16 The proposed method performance when the FASHION and Augmented FASHION datasets are employed.

The performance of the self-training layer-wise approach when applied to the Quickdraw and Augmented Quickdraw datasets is presented in Fig. 5.17. Notably, both datasets consistently increased accuracy across various dataset sizes. However, the method's performance with the Augmented Quickdraw dataset consistently outperformed that with the standard Quickdraw dataset. Specifically, the Augmented Quickdraw dataset achieved an accuracy of 85% with 0.33% of labeled examples, which further increased to 89% at 1.67%. Moreover, accuracy steadily improved from 90% to 92% within the range of 3.33% to 16.67%. In contrast, the accuracy results for the Quickdraw dataset began at just over 78% with 0.33% of labeled dataset size, improved to 85% at 1.67%, and further grew from 87% to 90% within the range of 3.33% to 16.67%.

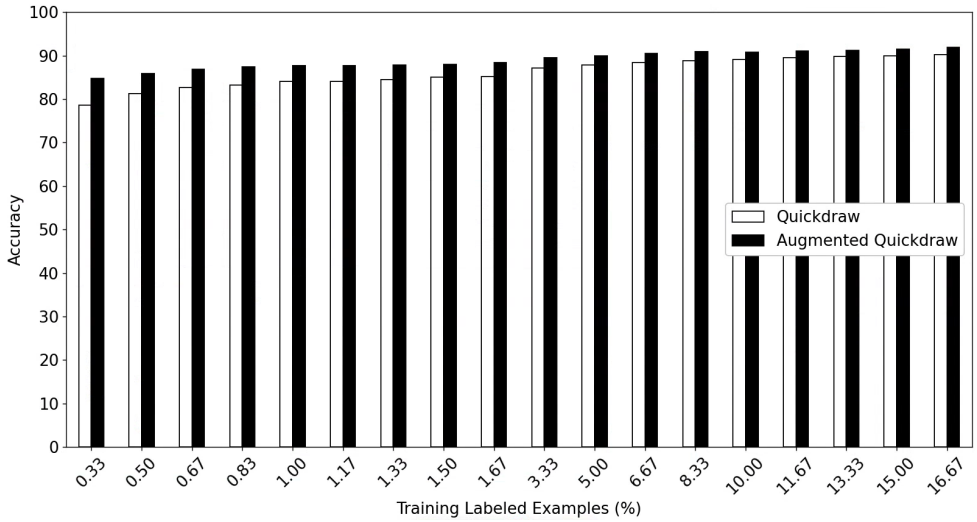


Fig. 5.17 Our method behavior throughout the different dataset sizes when the Quickdraw and Augmented Quickdraw are utilized.

We present a comparative analysis between the proposed self-training layer-wise method and other learning techniques when exclusively trained on the augmented dataset versions. Specifically, we compare self-training layer-wise, self-training, semi-supervised, and supervised methods across diverse scenarios involving varying labeled dataset sizes. Across all three augmented datasets, our method consistently outperforms the other techniques in situations characterized by limited labeled dataset sizes, particularly in the initial half of the few labeled examples. However, as the number of labeled examples increases or the percentage of labeled examples approaches the latter half, the proposed method's performance exhibits minimal superiority over the alternative methods, except for the Augmented FASHION dataset, where its superior performance is more pronounced.

Fig. 5.18 presents a comprehensive analysis of our method's performance and other techniques when trained on various labeled dataset sizes derived from the Augmented MNIST dataset. In the dataset size between 0.33% and 1.67%, our method consistently outperforms the alternatives, achieving accuracy rates of approximately 94% and 97%, respectively. However, as the labeled dataset size increases in the 3.33%–16.67% range, our method's accuracy results align closely with those of the alternative methods, typically hovering around 99%.

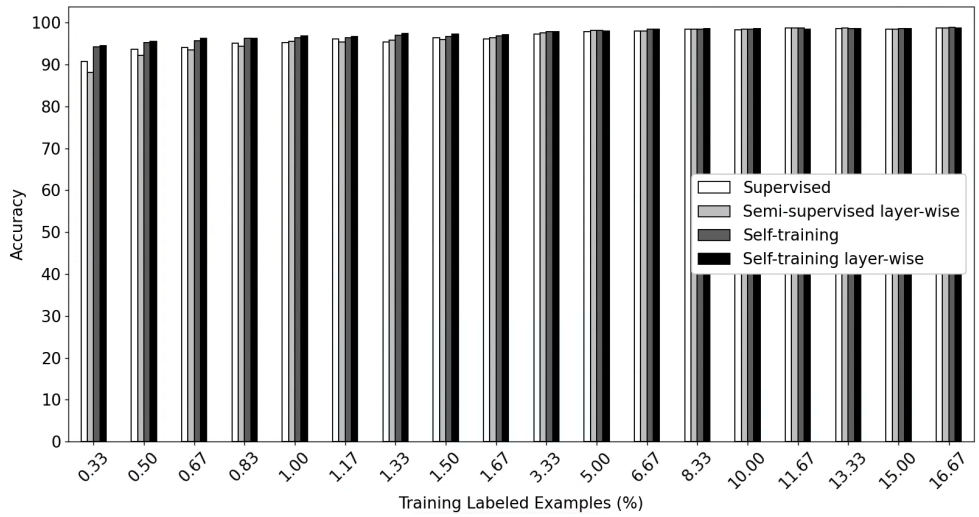


Fig. 5.18 Accuracy performance of the proposed method and alternative approaches when evaluated with Augmented MNIST.

The Augmented FASHION dataset compares our proposed method against alternative approaches, as depicted in Fig. 5.19. Our self-training layer-wise methodology surpasses the alternatives within the dataset size range of 0.33% to 1.50%, achieving accuracy rates that climb from 72% to 81%. Although the accuracy result does not exhibit superiority at 1.67%, it maintains its advantage and steadily increases from 85% to just under 90% within the dataset size range of 3.33% to 16.67%.

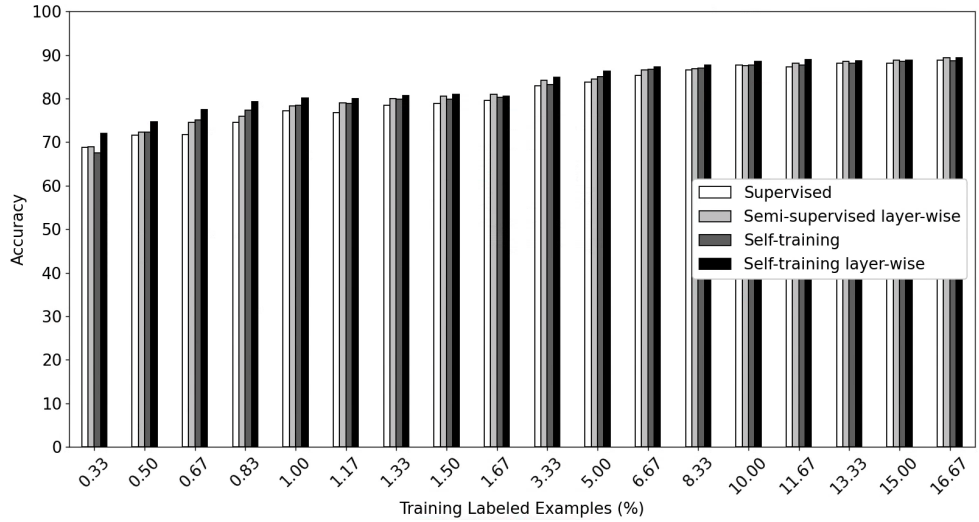


Fig. 5.19 Comparing the performance of self-training layer-wise with other methods when trained with Augmented FASHION.

Compared to alternative methods, the performance outcomes of the proposed method using the Augmented Quickdraw dataset are visually presented in Fig. 5.20. Within the dataset size range of 0.33% to 1.67%, our method exhibits superior performance, achieving accuracy rates of 85% and slightly exceeding 88%, respectively. Similarly, our method maintains a slight advantage over the alternatives, with accuracy results of approximately 90% and just above 92% in the dataset size range of 3.33% to 16.67% within various labeled training scenarios.

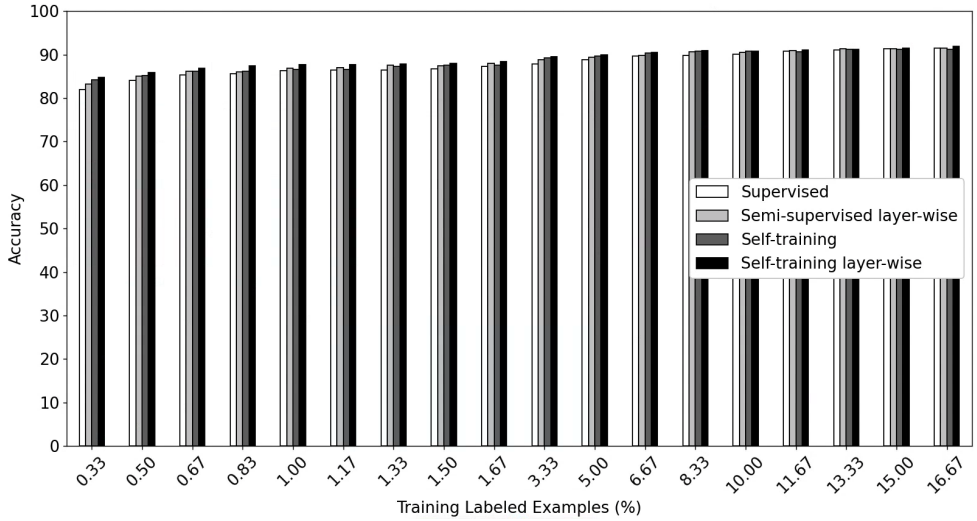


Fig. 5.20 A comparative analysis of our proposed method and other approaches when trained on the Augmented Quickdraw dataset.

5.10 Deep Semi-supervised Approach for Seismic Data Analysis

5.10.1 Training and Validation Curves for Neural Networks

Fig. 5.21 illustrates the faster convergence of the deep semi-supervised learning (DSSL) model, attributed to the weight initialization performed during the unsupervised training phase on the unannotated data [113]. In contrast, the supervised model exhibited instability in both the training and validation curves. Both approaches were trained for 225 epochs; however, the final loss values for DSSL and supervised learning were 0.0187 and 0.0247, respectively.

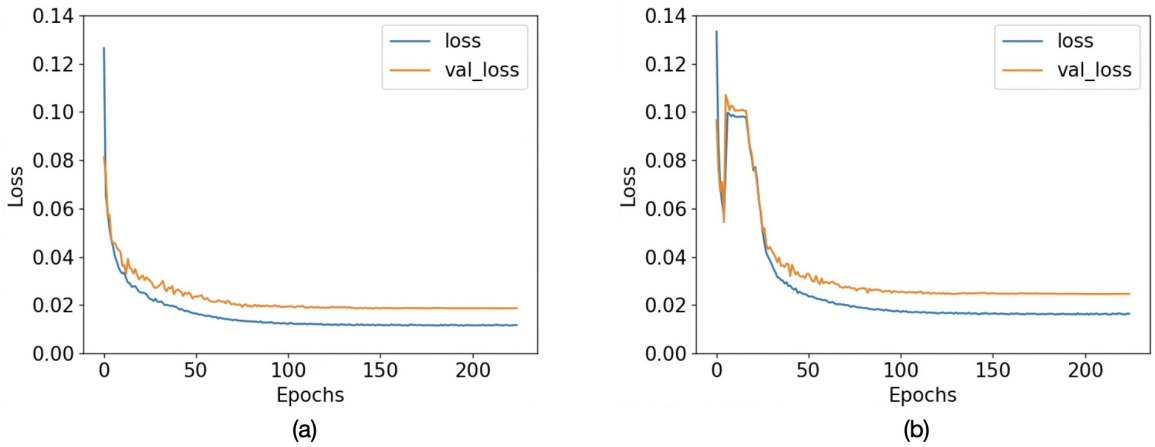


Fig. 5.21 The loss curves illustrate the benefit of weight initialization for (a) deep semi-supervised learning, leading to faster convergence compared to (b) its supervised counterpart.

5.10.2 Acoustic Impedance Estimation at the Borehole Site

The data from five boreholes were utilized to assess the predictive performance of the deep semi-supervised learning (DSSL) and supervised learning approaches. Fig. 5.22 illustrates that the absolute acoustic impedance (AI) predictions generated by the DSSL model more closely align with the actual well data compared to those produced by the supervised learning model.

The DSSL approach demonstrates a similar advantage over supervised learning in predicting absolute AI for both Well-1 and Well-3. Using Well-1 as an example, Fig. 5.22a illustrates that the DSSL prediction exhibits errors only between 2,500 ms and 2,550 ms. In contrast, the supervised method's prediction errors begin prior to 2,500 ms and extend beyond 2,550 ms, as highlighted in Fig. 5.22b.

For Wells 2, 4, and 5, while the DSSL approach outperforms the supervised method, a nuanced analysis is necessary to discern the subtle differences between the two approaches. For example, the DSSL's absolute AI prediction for Well-2 exhibits a closer alignment with the well data than that of the supervised method, particularly in the time interval between 2,500 ms and 2,600 ms.

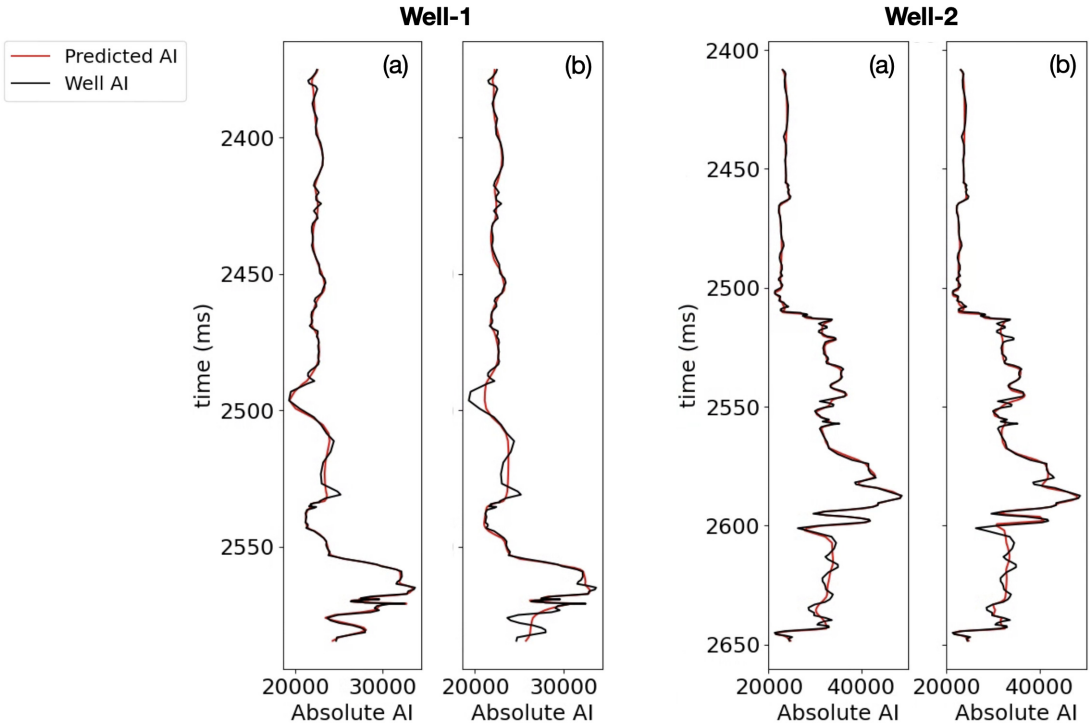


Fig. 5.22 A comparison of acoustic impedance values obtained from well-log data (Well AI) with those predicted using (a) DSSL and (b) supervised methods.

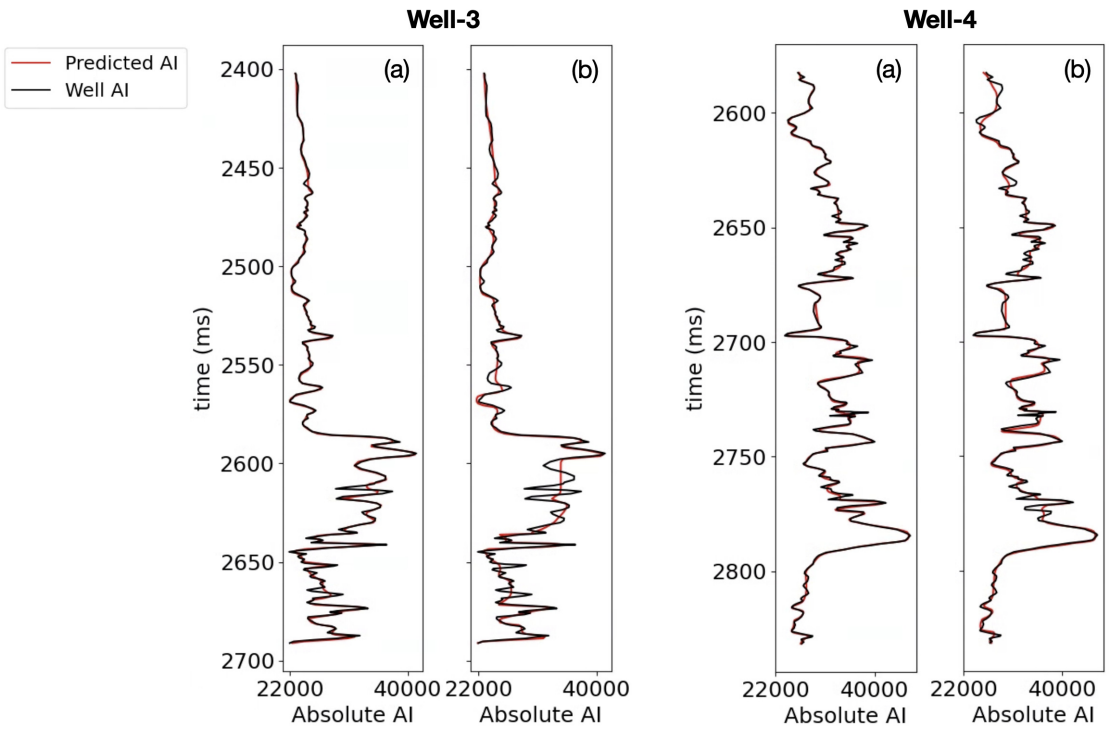


Fig. 5.22 (cont.) A comparison of acoustic impedance values obtained from well-log data (Well AI) with those predicted using (a) DSSL and (b) supervised methods.

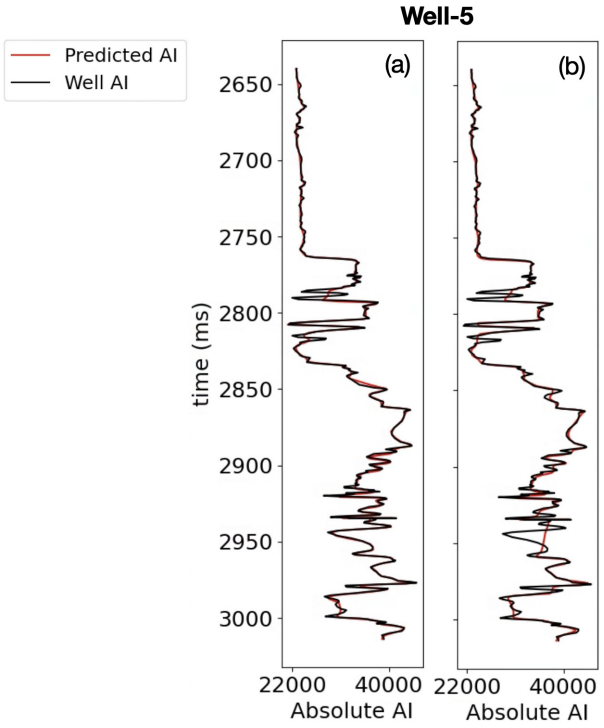


Fig. 5.22 (cont.) A comparison of acoustic impedance values obtained from well-log data (Well AI) with those predicted using (a) DSSL and (b) supervised methods.

5.10.3 Evaluating Acoustic Impedance Estimation at the Borehole Site

Visualization metrics, including crossplots, along with statistical tools such as Mean Square Error (MSE) and Pearson Correlation (PC), were employed to assess the quality of neural network predictions in relation to well-log data.

The acoustic impedance estimation results for the deep semi-supervised learning (DSSL) and supervised methods are visually represented in the crossplots shown in Fig. 5.23. These crossplots evaluate the correlation between the neural network outputs and the well data for each learning approach across the different wells. Generally, a proportional relationship exists between the predictions and the well acoustic impedance; however, the outputs from the DSSL exhibit reduced dispersion of data points, indicating a higher precision in the estimation.

Taking the supervised method predictions for Well-1 (Fig. 5.23) as an example, it is observed that the dispersion of points increases along the diagonal in certain regions, a phenomenon that occurs to a lesser extent with the DSSL method (Fig. 5.23a). Consequently, the pronounced dispersion observed in the supervised method (Fig. 5.23b) indicates less

accurate predictions of acoustic impedance compared to those generated by the DSSL approach.

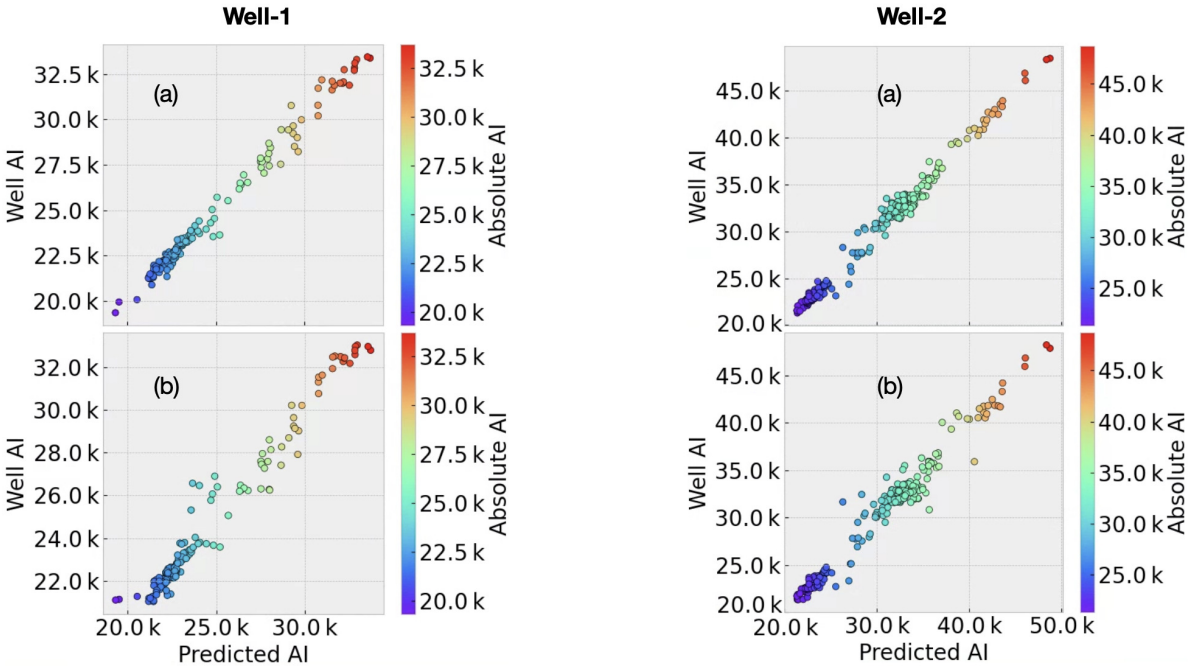


Fig. 5.23 Dispersion of AI data points for (a) DSSL and (b) the supervised approach. The reduced dispersion of points in DSSL indicates a stronger correlation between Well AI data and the model predictions.

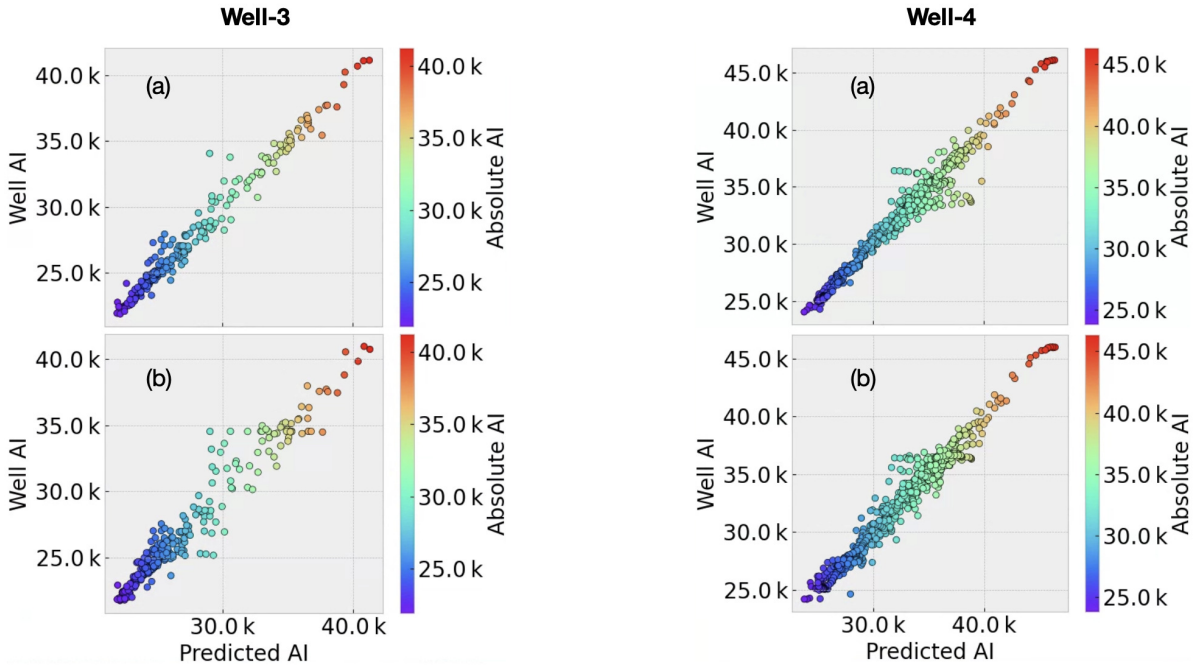


Fig. 5.23 (cont.) Dispersion of AI data points for (a) DSSL and (b) the supervised approach. The reduced dispersion of points in DSSL indicates a stronger correlation between Well AI data and the model predictions.

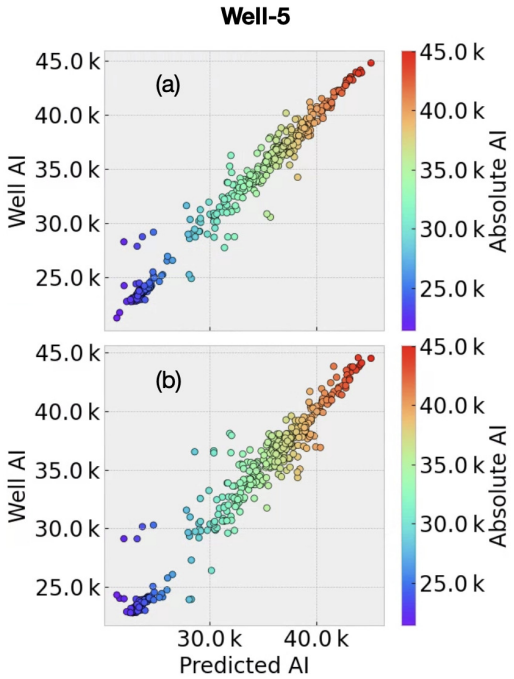


Fig. 5.23 (cont.) Dispersion of AI data points for (a) DSSL and (b) the supervised approach. The reduced dispersion of points in DSSL indicates a stronger correlation between Well AI data and the model predictions.

The Pearson correlation and mean square error analyses further substantiate the DSSL method as the more accurate model for AI estimation compared to the supervised approach. The PC indicates a stronger relationship between the AI predicted by the DSSL model and the well AI, while the MSE emphasizes that the estimation errors for DSSL are notably lower than those of the supervised model. These results are summarized in Table 5.5.

Table 5.5 Quantitative evaluations of deep semi-supervised and supervised predictions, using well AI as the reference. The p-value for the Pearson correlation analysis is less than 0.05.

	PC		MSE	
Well	DSSL	Supervised	DSSL	Supervised
1	0.9938	0.9837	1.15E-04	3.05E-04
2	0.9937	0.9851	2.05E-04	4.83E-04
3	0.9882	0.9721	2.46E-04	5.92E-04
4	0.9807	0.9800	3.33E-04	3.52E-04
5	0.9892	0.9791	4.46E-04	8.67E-04

5.10.4 Predictions Beyond Well Location

To further evaluate the capability of the deep semi-supervised learning (DSSL) method in predicting absolute acoustic impedance (AI), we extended the analysis to locations beyond the gathers where the boreholes are situated. This involved using seismic data from gathers at specified distances from the source well-log data, which continued to serve as a reference for the newly estimated AI values. For comparison, AI predictions were also generated using the supervised model alongside the DSSL method.

Fig. 5.24 presents the results for Well-1. The seismic data from the inline shown in Fig. 5.24a were used as input for both the trained models, DSSL and the supervised approach, generating the predicted absolute AI inlines depicted in Fig. 5.24b and Fig. 5.24c, respectively. While the well AI data in Fig. 5.24a primarily served as the target during the training phase for both methods, synthetic data played a crucial role in preparing the seismic inputs. It is important to note that the well AI remains consistent across all three figures, and is used in Fig. 5.24b and Fig. 5.24c for comparative analysis. Furthermore, the AI estimation using the DSSL method produced superior results for Well-1, Well-3, and Well-5. For instance, in the case of Well-1, the DSSL method demonstrated a more accurate estimation of absolute AI beyond the well location, as marked by the grey arrows in Fig. 5.24b, when compared to the predictions made by the supervised method, Fig. 5.24c.

In Fig. 5.25b and Fig. 5.25c, the inline AI estimations by the DSSL and supervised models appear to be equivalent, with no significant improvement of one over the other when evaluated for Well-2. Similarly, this equivalence is observed for Well-4, particularly near the well location. However, at greater distances from the well, the DSSL predictions (Fig. 5.27b) show substantial enhancement compared to the supervised results (Fig. 5.27c). Furthermore, the comparison of inline AI data with well AI data is restricted to the temporal range in which the latter is available.

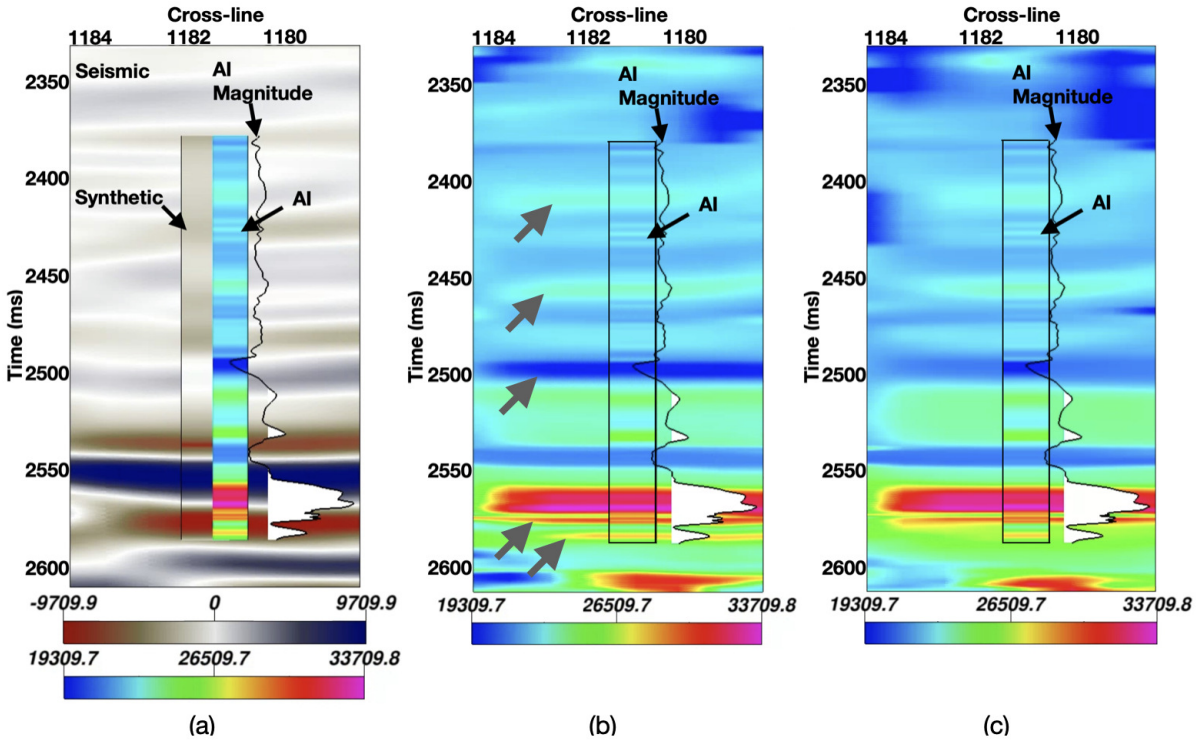


Fig. 5.24 For Well-1, (a) the seismic inline displayed is one of several slides utilized for neural network training employing (b) DSSL and (c) supervised approaches. The efficacy of DSSL in predicting acoustic impedance at long distances surpasses that of the supervised method.

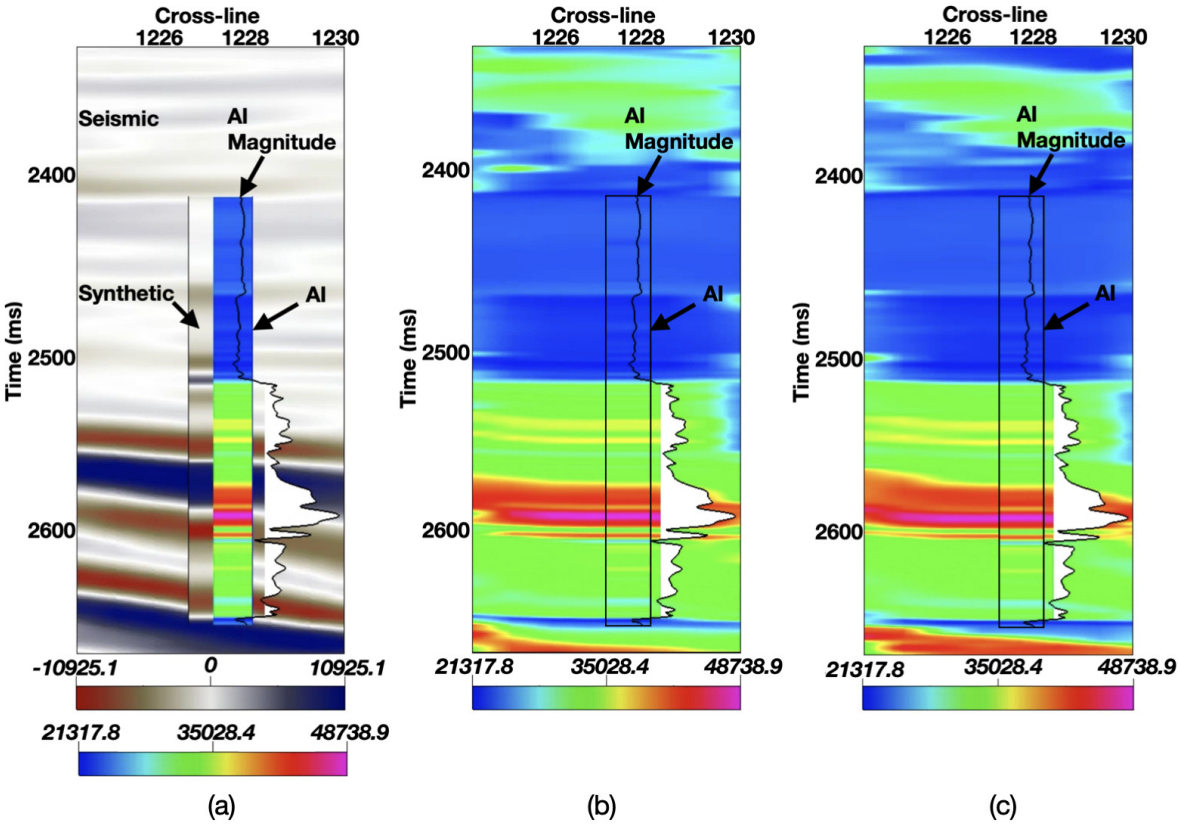


Fig. 5.25 (a) The seismic inline utilized for both (b) DSSL and (c) supervised methods for Well-2. In this context, there is no significant difference between the predictions produced by the two approaches.

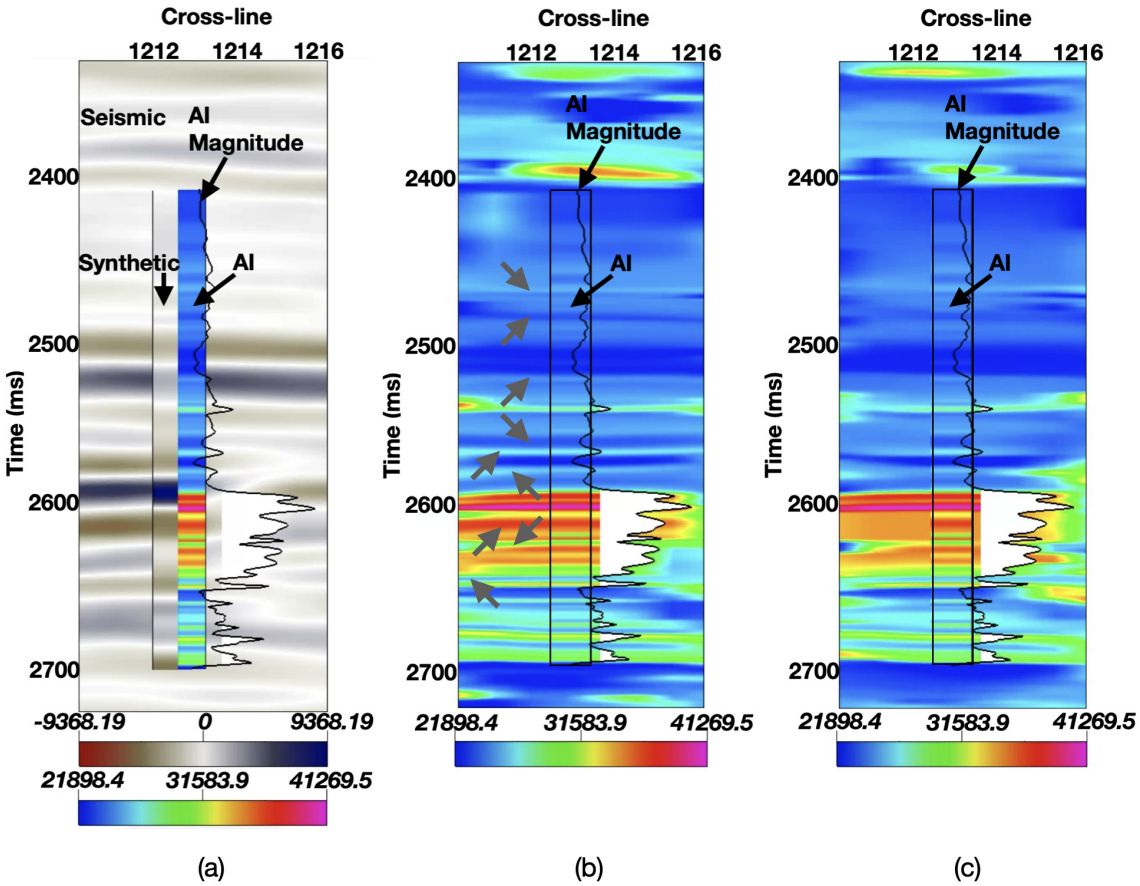


Fig. 5.26 For Well-3, (a) the seismic inline presented is one of several slides utilized for neural network training using (b) DSSL and (c) supervised approaches. The ability of DSSL to predict acoustic impedance at extended distances is superior to that of the supervised method.

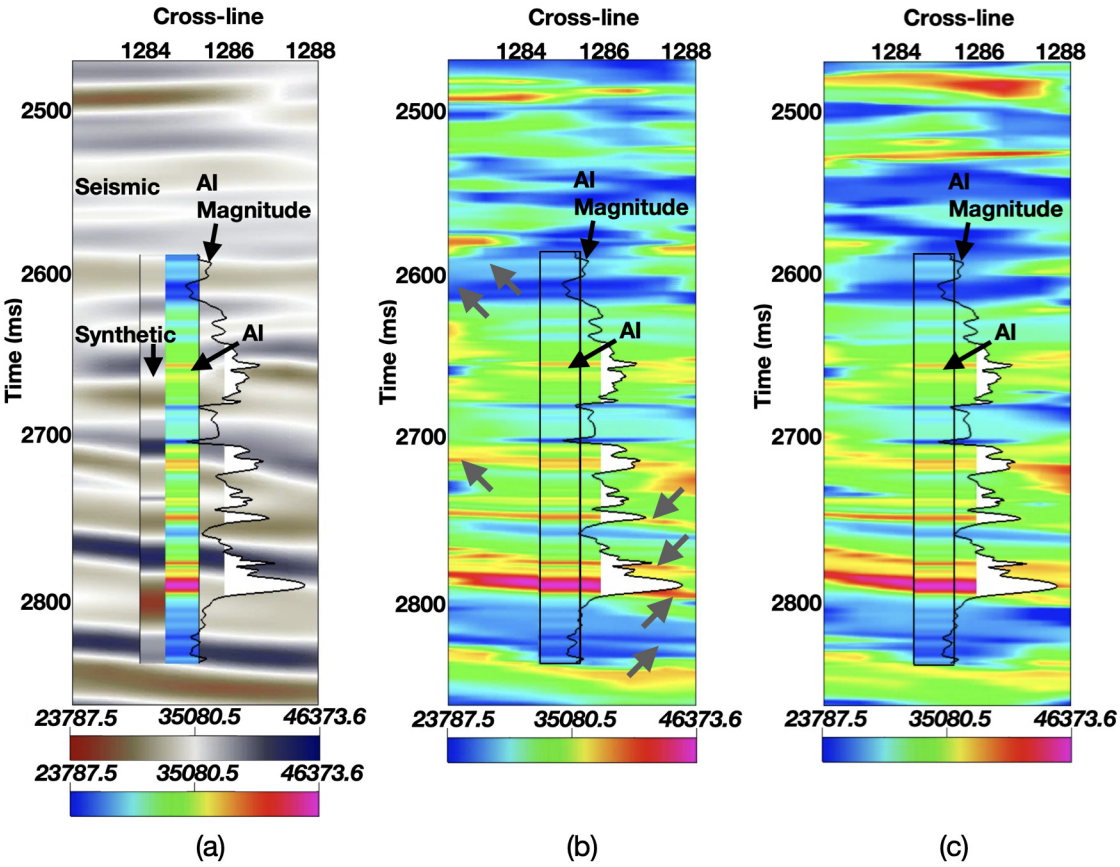


Fig. 5.27 For Well-4, (a) the displayed seismic inline represents one of several slides used for training the neural network with (b) deep semi-supervised learning (DSSL) and (c) supervised techniques. DSSL demonstrates a greater capability in predicting acoustic impedance over long distances compared to the supervised approach.

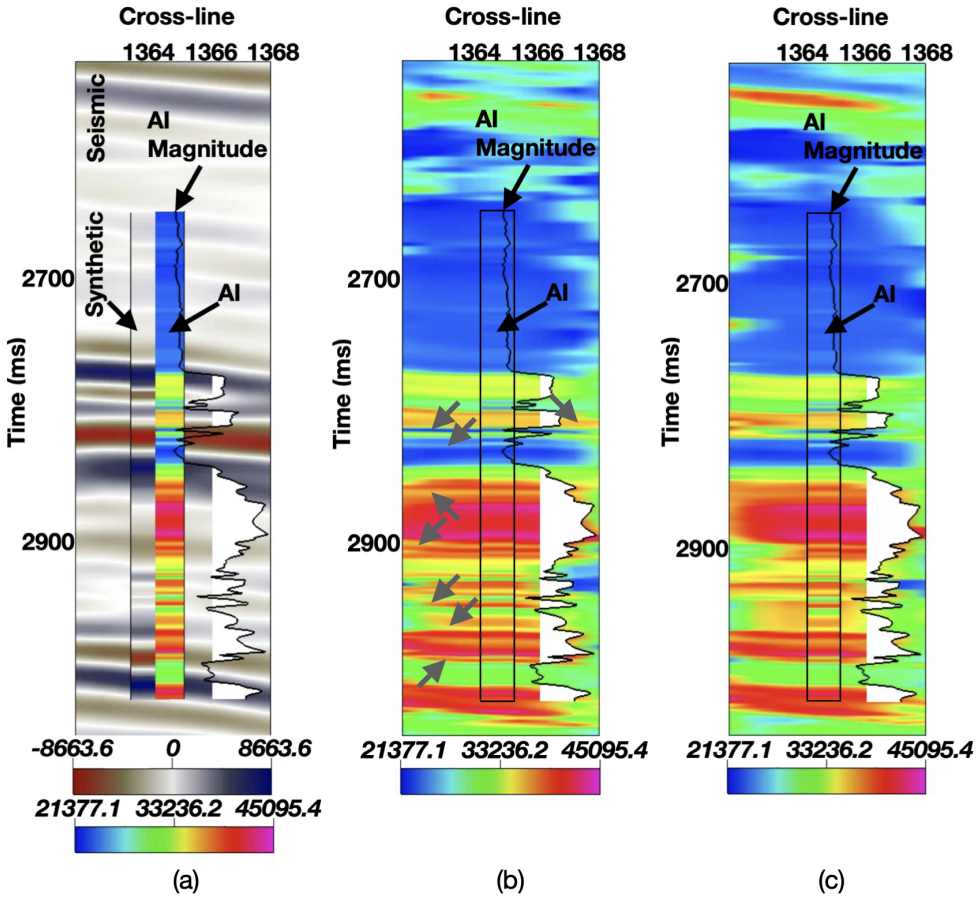


Fig. 5.28 For Well-5, (a) the displayed seismic inline and the corresponding AI target were utilized for training using both (b) deep semi-supervised learning (DSSL) and (c) supervised approaches. The capacity of DSSL to predict AI values at distances far from the well location is evident.

5.10.5 Analytical Acoustic Impedance

A new type of acoustic impedance, termed analytical acoustic impedance (AI), was generated using the open-source Pylops library [85]. The calculation of this AI required band-limited seismic data and a low-frequency model, the latter obtained from borehole measurements. Once the analytical AI was generated, it was compared to the well AI. The comparative results show that the analytical AI exhibits greater divergence from the well AI than the AI produced by the deep semi-supervised learning (DSSL) model. These results are illustrated in Fig. 5.29 for the five boreholes used.

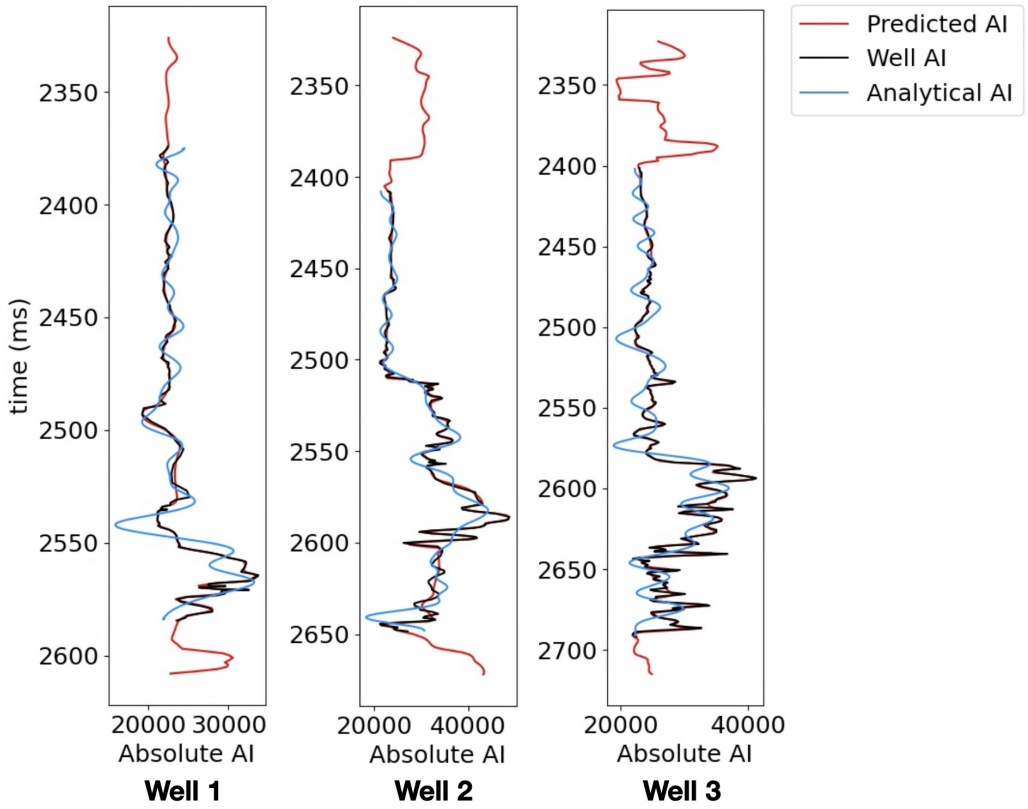


Fig. 5.29 The AI predictions generated by the neural network exhibit a closer alignment with the well log data (Well AI) compared to the analytical AI.

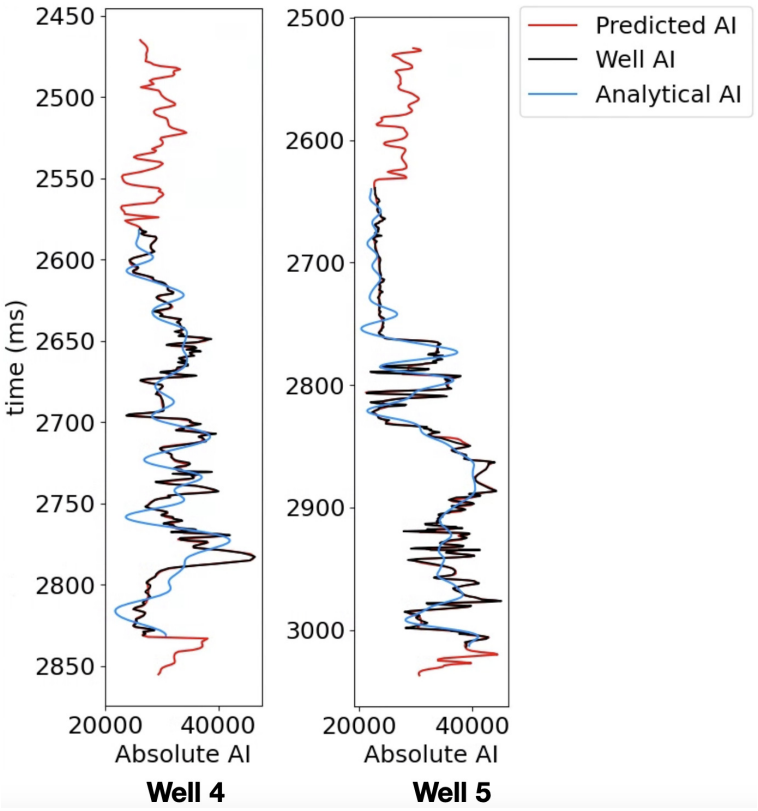


Fig. 5.29 (cont.) The AI predictions generated by the neural network exhibit a closer alignment with the well log data (Well AI) compared to the analytical AI.

Crossplots (Fig. 5.23), Pearson correlation, performance curves (Fig. 5.21), and mean square error (Table 5.5) collectively demonstrate that a model can be effectively developed using a large unannotated dataset with a small number of annotated samples. These metrics highlight the significant improvement in estimation capability achieved through this approach. Consequently, the unsupervised stage of the semi-supervised method successfully learned hidden features from the extensive unlabeled pre-stack seismic data. This capability is demonstrated when the DSSL method is employed to estimate acoustic impedance at distances further from the borehole site.

For certain inlines near the borehole site, such as Well-2 (Fig. 5.25), the predictions from the DSSL method do not show significant improvement over those from the supervised model. This suggests that, in this specific location, the use of additional unlabeled data may not have been necessary to capture the subsurface complexity.

Chapter 6

Discussion

6.1 Long Short-Term Memory Classification under Changes in Sequences Order

Recurrent Neural Networks (RNNs) are a prevalent choice for analyzing sequential data, encompassing various domains such as time-series data, natural language processing, genomics, and more. Moreover, RNNs can be effectively applied to engineering problems framed as sequential data tasks [95, 30]. For instance, in classification, images are conventionally treated as two-dimensional or higher-dimensional data [41], can be reformulated as sequential data by transforming them into sequences through processes like vectorization. This entails constructing pixel vectors of specific lengths and orders. RNNs have demonstrated promising outcomes for classifying grayscale images in this manner [111, 20, 94, 13].

The methodology outlined in Section 4.1 was developed to investigate the potential influence of sequence order and length on the overall performance of RNNs, with a specific emphasis on Long Short-Term Memory (LSTM) units. This question holds relevance for data setup procedures and has practical applications in areas such as video tagging and general image descriptions.

Our experimental results indicate that optimal accuracy is more likely to be achieved when working with grayscale images re-organized with a sequence length of 28 pixels. Nevertheless, the choice of sequence configuration, specifically 392 pixels, may yield better results for specific datasets. It is worth noting that the longer sequence length, as exemplified by 392 pixels, tends to correspond to improved performance.

On the other hand, it appears that the horizontal order proves to be the most consistent choice, as three out of five datasets exhibited improved outcomes when using the horizontal

order. Nevertheless, it is noteworthy that the vertical order also yielded optimal results for specific datasets.

Therefore, identifying the most suitable configuration involving order and sequence length for a given dataset can significantly enhance the neural network's learning process. For instance, in the cases of [13] and [94], where distinct sequence lengths (or image shapes) of 1 x 784 and 28 x 28 were employed, respectively, achieving an accuracy level of approximately 99% was possible.

Our investigation also revealed no correlation between the dependence index and the accuracy levels across the assessed datasets. Similarly, no significant relationship was observed between entropy and accuracy when modifications were made to the sequence orders and lengths. Nonetheless, it is noteworthy that lower entropy values were consistently associated with improved accuracy for each dataset (e.g., MNIST, MNIST-C, etc.).

6.2 Semi-supervised Methods with Medium Size Datasets

In Section 4.5, we evaluated four methods utilizing benchmark datasets, including MNIST and FASHION. The primary objective was to assess these methods across various dataset sizes and compare our proposed approach with alternative techniques. Experiments were designed to explore the potential enhancement in model performance when incorporating unlabeled data in scenarios with limited labeled data.

Applying supervised learning methods in scenarios with limited labeled examples significantly impacts model performance across various applications [105, 39, 81]. The scarcity of labeled examples, without incorporating pre-training methods or alternative techniques, inevitably reduces model accuracy. This phenomenon is evident in the performance of both the DeepHeart model [14] and the EfficientNet-B7 architecture [123].

We exemplified this phenomenon using the MNIST and FASHION datasets solely through supervised learning. Specifically, the model's accuracy markedly decreases when the labeled dataset percentage equals or falls below 1.67%. However, the model's performance improves by incorporating semi-supervised methods, namely, semi-supervised layer-wise and self-training. Harnessing unlabeled data through these methods significantly enhances model performance, particularly in severe labeled data scarcity scenarios, where the percentage is equal to or less than 1.67%.

Building upon our preceding observations, we introduce a novel methodology designed for optimal utilization of unlabeled data. This method underwent testing in scenarios with scarce labeled data, specifically on the MNIST and FASHION datasets. In the case of MNIST, our method demonstrated superior accuracy compared to alternative methods, mainly when

training examples were below 1.67%. However, the observed accuracy improvement of our method over others with the FASHION dataset seemed consistent across all dataset sizes, with less pronounced distinctions in cases of severe data scarcity, specifically when the percentage was less than 1.67%.

The variation in the behavior of our method in scenarios of limited labeled datasets for MNIST and FASHION datasets may be correlated with the entropy levels of these datasets. Specifically, the FASHION dataset exhibits higher entropy compared to MNIST. As detailed in Section 5.4, an inverse relationship exists between entropy and accuracy, where increased entropy tends to correspond with diminished accuracy.

On the other hand, beyond the critical scenarios or the 3.33% of training examples, the outcomes of alternative methods closely approximate those of the proposed method for the MNIST and FASHION datasets. This trend is attributed to the growing number of training examples. Thus, harnessing unlabeled data through semi-supervised methods proves most valuable in situations characterized by a severe shortage of labeled data.

6.3 The Proposed Method Extended for Large Size Datasets

Real-world datasets, including seismic data [81], medical records [105], and text corpora [118], often comprise substantial amounts of unlabeled data alongside a limited number of labeled examples. We employed the extensive benchmark Quickdraw Bitmap dataset to address the challenges associated with training neural networks on such datasets. A dataset is large or massive when it encompasses approximately one million examples.

The experimental framework for the Quickdraw Bitmap dataset is detailed in Section 4.5, involving the transformation of the original dataset into a partially labeled format. We then assessed our proposed method and alternative techniques using the established methodology outlined in Section 4.6, designed for handling massive-sized datasets. Throughout this procedure, we addressed technological challenges, including memory issues during data pre-processing and the slow neural network training process associated with managing large volumes of data.

In conjunction with utilizing the multi-worker strategy to accelerate the training process across multiple graphics processing units (GPUs), we implemented the linear-epoch gradual-warmup (LEGW) method to optimize memory utilization by employing larger batch sizes. Applying the LEGW method necessitated specific configurations for pre-training and fine-tuning across the four learning techniques under evaluation.

The primary objective of the LEGW method is to establish an optimal relationship between the learning rate and a specific large batch size during training. This process is

relatively straightforward when working with a particular learning method and a fixed number of training examples. However, challenges arise when dealing with diverse methods and varying numbers of training examples, making it difficult for the LEGW method to determine the optimal relation between the learning rate and batch size. To address this, we introduce the step parameter, which facilitates the adjustment of the scaling factor for a given learning method and the number of training examples. This parameter regulates the batch size based on memory availability and ensures optimal accuracy even in challenging scenarios.

Our method performs better when applied to a large dataset across various dataset sizes. Notably, alternative methods fail to exhibit superior behavior compared to our method at any percentage of the labeled dataset of Quickdraw. The proposed method's consistent performance can be explained by the extensive use of unlabeled data from the Quickdraw dataset during the pre-training stage, leading to enhanced generalization.

The favorable outcomes observed, where our method surpasses other techniques, show that the LEGW method exhibits notable behavior with various learning methods. It significantly addresses the problem articulated in Section 1.1, which outlines the challenge of analyzing massive partially labeled databases.

6.4 The Proposed Method and Augmented Datasets

In Section 5.5 and Section 5.6, our proposed method has been demonstrated to yield superior results compared to other learning techniques. Further enhancement in performance is achievable through the incorporation of augmentation techniques. Expanding on this, in Section 5.9, we leveraged augmented versions of MNIST, FASHION, and Quickdraw datasets, resulting in significant improvements in our method's accuracy. Notably, for the MNIST dataset, in the most challenging scenario with only 0.33% of training examples, the accuracy surged from approximately 78% using the standard dataset to around 95% when utilizing the augmented version. It is noteworthy that this heightened accuracy is achieved through the application of augmentation techniques to the initial 200 labeled examples. For FASHION and Quickdraw datasets, there was also an improvement in model performance at 0.33%, albeit to a lesser extent.

We observe a consistent enhancement in the outcomes of our method for FASHION and Quickdraw datasets when comparing between non-augmented and augmented versions. This improvement remains consistent across all dataset sizes. The favorable results can be attributed to the augmentation technique, which increases the number of examples for pre-training and fine-tuning. Implicitly, the effective utilization of unlabeled data in our proposed method contributes significantly to enhancing model generalization.

On the other hand, in the exclusive evaluation of the four methods using augmented datasets, it is observed that not only our method but also alternative techniques show improvements in accuracy results. Consequently, our method exhibits modest superiority over alternative methods in the case of Augmented MNIST and a slight advantage with the Augmented Quickdraw dataset. Intriguingly, our method distinctly outperforms the others with the Augmented FASHION dataset in the first half of the training example percentages.

Previously, it was mentioned that the standard FASHION dataset exhibits a notable level of entropy, posing challenges for our method to achieve substantial improvements in accuracy. Nevertheless, with the Augmented FASHION dataset, our method demonstrates a significant enhancement in performance compared to other methods. This suggests that, when entropy poses an obstacle to model performance, it can be alleviated by increasing training examples through data augmentation techniques.

6.5 Deep Semi-supervised Approach for Seismic Data Analysis

Supervised approaches have been widely applied to petrophysical estimations, such as porosity [116], permeability [104], and mineralogy [54]. In contrast, deep semi-supervised learning (DSSL) techniques, which leverage large amounts of unlabeled data, have been used for lithofacies classification [9, 67]. In the context of oil exploration, these petrophysical properties are critical for understanding the subsurface [12, 45]. In this study, the DSSL method is employed to develop a regression model for the estimation of absolute acoustic impedance, another key petrophysical property.

The sequence length and the number of features are critical parameters for the effective training of a Long Short-Term Memory (LSTM) network. In contrast to the common trend in the literature, which predominantly employs convolutional neural networks (CNNs), we utilize this recurrent neural network in a DSSL framework with seismic data. While sequence length significantly impacts both model performance and training time, selecting an appropriate number of features is also essential for optimal performance. Despite missing the first four traces for some common depth points (CDPs), these were included as they introduced a form of noise that improved the model's performance during training [109, 38].

While the DSSL model may produce acoustic impedance (AI) predictions with higher inaccuracies when input data originates from outside the region where the unlabeled data was sourced, its estimation accuracy improves as the input data is drawn from within the gathers used for the unlabeled data. The model's performance further improves as the data is

closer to the well location. Test data from the well site is crucial for ensuring the quality and reliability of the model's estimations.

Although small cubes were used for data extraction, a substantial amount of unlabeled data was generated from this limited portion of the seismic cube. The downsampling approach employed helped manage the large volume of generated examples. Future work should focus on developing a methodology for extracting relevant unlabeled examples, rather than generating data exhaustively, which can result in unmanageable datasets.

Chapter 7

Conclusion

This study aims to contribute an optimal methodology for analyzing massive datasets with partially known information. To achieve this, we conducted experiments in which a model architecture based on Long Short-Term Memory (LSTM) networks was evaluated using varying amounts of labeled training examples. We address this issue in the following manner.

- Firstly, a novel approach was devised and assessed using benchmark datasets, specifically the MNIST and FASHION datasets. Quantitative results indicate the superiority of our method over alternative techniques, particularly under conditions of severe shortage of labeled examples across the entire experimental design. Notably, our approach incorporates the utilization of unlabeled data during both pre-training and fine-tuning phases.

- Secondly, we extended the applicability of our proposed method to large-scale datasets, exemplified by the Quickdraw Bitmap dataset. Similarly, our method demonstrated clear superiority over alternative approaches, especially in scenarios involving a scarcity of labeled examples. Additional strategies, including the multi-worker strategy, the linear-epoch gradual-warmup (LEGW) method, and enhancements to our methodology, were implemented to address the challenges posed by massive datasets.

- Thirdly, the accuracy performance of our method is significantly enhanced through the application of augmentation techniques on the MNIST, FASHION, and Quickdraw datasets. Furthermore, our observations indicate that augmentation techniques can mitigate the adverse impact of high entropy on accuracy performance, particularly evident in the case of the FASHION dataset.

- Finally, the step parameter optimizes the utilization of a large batch size, taking into account the available memory resources. This becomes crucial in light of the fluctuating

number of training examples in the experimental setup and the inherent characteristics of the self-training framework.

Our framework aims to optimize the utilization of layer-wise pre-training and self-training approaches on extensive unlabeled datasets. The inherent nature of the problem necessitates the use of current technologies, such as deep neural networks, and consequently relies on substantial computational resources, including graphics processing units (GPUs). This approach holds the potential for addressing classification challenges in domains such as natural language processing, medicine, and reservoir characterization, where the availability of labeled examples is severely limited.

In classification tasks, neural networks can assess output predictions by assigning probabilities to the outputs. However, when dealing with regression problems, giving a score to output values becomes challenging. Providing a score is crucial for evaluating the prediction quality, particularly for incorporating examples into the self-training framework as pseudo-labeled instances. This limitation in our approach presents an opportunity for future research, offering the potential to develop methodologies that address the scoring challenge and thereby extend the applicability of our proposed method to regression problems.

7.1 Application

The scarcity of annotated data poses a significant challenge in various domains, particularly in reservoir characterization. Although increasing the quantity of labeled data requires considerable effort, it is essential for constructing an optimal model for subsurface characterization. To tackle this problem, we implement these procedures.

- Firstly, the DSSL approach was employed to leverage the substantial volume of unannotated data. Despite the limited size of the annotated dataset, the model's performance was enhanced.
- Secondly, the annotated dataset was expanded by labeling seismic data from gathers at the borehole site and from nearby gathers using the same well-log data. This expansion of labeled instances enriches the dataset and aids in training the model more effectively.
- Thirdly, we have introduced a technique for extracting unannotated examples from pre-stack seismic cube. This technique has been designed to facilitate the neural network's learning process, even when dealing with a small quantity of annotated examples.

In the context of limited absolute acoustic impedance data and substantial volumes of seismic data, the DSSL technique is introduced to derive acoustic impedance from these seismic datasets.

We believe that the developed approach holds significant potential in the domain of reservoir characterization, as the absolute acoustic impedance predicted by the DSSL method closely aligns with the well AI, particularly when compared to other acoustic impedance estimation techniques.

References

[1] (2002). Synthetic Seismic Profile Plot Description. Available online: https://www.kgs.ku.edu/software/SS/description.html (accessed on 20/07/2023).

[2] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mane, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viegas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.

[3] Acciarini, C., Cappa, F., Boccardelli, P., and Oriani, R. (2023). How can organizations leverage big data to innovate their business models? A systematic literature review. *Technovation*, 123(May 2022).

[4] Alfarhan, M., Deriche, M., and Maalej, A. (2022). Robust Concurrent Detection of Salt Domes and Faults in Seismic Surveys Using an Improved UNet Architecture. *IEEE Access*, 10:39424–39435.

[5] Ao, Y., Li, H., Zhu, L., and Yang, Z. (2019). A SCiForest based semi-supervised learning method for the seismic interpretation of channel sand-body. *J. Appl. Geophys.*, 167:51–62.

[6] Araya-Polo, M., Alpak, F. O., Hunter, S., Hofmann, R., and Saxena, N. (2020). Deep learning–driven permeability estimation from 2D images. *Comput. Geosci.*, 24(2):571–580.

[7] Archenaa, J. and Anita, E. A. M. (2016). Interactive big data management in healthcare using spark. In *3rd International Symposium on Big Data and Cloud Computing Challenges (ISBCC–16')*, pages 265–272. Springer.

[8] Arrowsmith, S. J., Trugman, D. T., MacCarthy, J., Bergen, K. J., Lumley, D., and Magnani, M. B. (2022). Big Data Seismology. *Reviews of Geophysics*, 60(2).

[9] Asghar, S., Choi, J., Yoon, D., and Byun, J. (2020). Spatial pseudo-labeling for semi-supervised facies classification. *J. Pet. Sci. Eng.*, 195(August):107834.

[10] Azevedo, L., Paneiro, G., Santos, A., and Soares, A. (2020). Generative adversarial network as a stochastic subsurface model reconstruction. *Comput. Geosci.*, 24(4):1673–1692.

[11] Babu, M. N., Ambati, V., and Nair, R. R. (2022). Lithofacies and fluid prediction of a sandstone reservoir using pre-stack inversion and non-parametric statistical classification: A case study. *J. Earth Syst. Sci.*, 131(1):55.

[12] Bagheri, H., Tanha, A. A., Doulati Ardejani, F., Heydari-Tajareh, M., and Larki, E. (2021). Geomechanical model and wellbore stability analysis utilizing acoustic impedance and reflection coefficient in a carbonate reservoir. *J. Pet. Expl. Prod. Tech.*, 11(11):3935–3961.

[13] Bai, S., Kolter, J. Z., and Koltun, V. (2018). (2018). An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling.

[14] Ballinger, B., Hsieh, J., Singh, A., Sohoni, N., Wang, J., Tison, G. H., Marcus, G. M., Sanchez, J. M., Maguire, C., Olgin, J. E., and Pletcher, M. J. (2018). Deepheart: Semi-supervised sequence learning for cardiovascular risk prediction. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 2079–2086.

[15] Barclay, F., Bruun, A., Rasmussen, K. B., Alfaro, J. C., Cooke, A., Cooke, D., Salter, D., Godfrey, R., Lowden, D., McHugo, S., Özdemir, H., Pickering, S., Pineda, F. G.,

Herwanger, J., Volterrani, S., Murineddu, A., Rasmussen, A., and Roberts, R. (2008). Seismic inversion: Reading between the lines. *Oilfield Review*, 20(1):42–63.

[16] Bedi, J. and Toshniwal, D. (2020). Features denoising-based learning for porosity classification. *Neural Computing and Applications*, 32(21):16519–16532.

[17] Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*. The MIT Press.

[18] Berkhahn, F., Keys, R., Ouertani, W., Shetty, N., Autoencoder, V., Detection, A., and Learning, R. (2019). A UGMENTING V ARIATIONAL A UTOENCODERS WITH S PARSE L ABELS : A UNIFIED FRAMEWORK FOR UNSUPERVISED ,. *arXiv preprint arXiv:1908.03015*.

[19] Bohr, A. and Memarzadeh, K. (2020). *Current healthcare, big data, and machine learning*. INC.

[20] Breuel, T. M. (2015). (2015). Benchmarking of LSTM Networks.

[21] Cai, H., Wu, Q., Ren, H., Li, H., and Qin, Q. (2019). Pre-stack texture-based semi-supervised seismic facies analysis using global optimization. *J. Seis. Expl.*, 28(6):513–532.

[22] Chalupka, K., Perona, P., and Eberhardt, F. (2018). (2018). Fast Conditional Independence Test for Vector Variables with Large Sample Sizes.

[23] Chapelle, O., Schölkopf, B., and Zien, A. (2006). *Semi-Supervised Learning*. The MIT Press, London, England.

[24] Chen, M., Mao, S., and Liu, Y. (2014). Big data: A survey. *Mobile Networks and Applications*, 19(2):171–209.

[25] Cielen, D., Meysman, A. D. B., and Ali, M. (2016). *Introducing Data Science. 1st edn.* Manning Publications Co., Shelter Island, US.

[26] Clark, K., Luong, M.-T., Manning, C. D., and Le, Q. (2019). Semi-Supervised Sequence Modeling with Cross-View Training. pages 1914–1925.

[27] Cover, T. M. and Thomas, J. A. (2006). *Elements of Information Theory. 2nd edn.* Wiley-Interscience, Hoboken, N.J.

[28] Cunha, A., Pochet, A., Lopes, H., and Gattass, M. (2020). Seismic fault detection in real data using transfer learning from a convolutional neural network pre-trained with synthetic seismic data. *Comput. Geosci.*, 135(July 2018):104344.

[29] Deng, J., Xu, X., Zhang, Z., Fruhholz, S., and Schuller, B. (2018). Semisupervised Autoencoders for Speech Emotion Recognition. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 26(1):31–43.

[30] Dietterich, T. G. (2002). Machine learning for sequential data: A review. In Caelli, T., Amin, A., Duin, R. P., de Ridder, D., and Kamel, M., editors, *Joint IAPR 9th International Workshop on Structural and Syntactic Pattern Recognition, SSPR 2002 and 4th International Workshop on Statistical Techniques in Pattern Recognition, SPR 2002*, volume 2396, pages 15–30, Windsor, ON, Canada. Springer Verlag.

[31] Dixit, A. and Mandal, A. (2020). Detection of gas chimney and its linkage with deep-seated reservoir in poseidon, NW shelf, Australia from 3D seismic data using multi-attribute analysis and artificial neural network approach. *J. Nat. Gas Sci. Eng.*, 83(August):103586.

[32] Dong, S., Zeng, L., Lyu, W., Xu, C., Liu, J., Mao, Z., Tian, H., and Sun, F. (2020). Fracture identification by semi-supervised learning using conventional logs in tight sandstones of Ordos Basin, China. *J. Nat. Gas Sci. Eng.*, 76(December 2019):103131.

[33] Ejovi, A. E. and John, A. O. (2019). *Integration of Seismic and Well Log Data Using Acoustic Impedance for Lithology and Hydrocarbon Evaluation of "Ovi" Field, Niger Delta*. PhD thesis, Federal University of Technology Akure.

[34] Ek-Chacón, E. and Molino-Minero-Re, E. (2020). LSTM Classification under Changes in Sequences Order. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12468 LNAI, pages 3–20.

[35] Ek-Chacón, E., Molino-Minero-Re, E., Méndez-Monroy, P. E., Neme, A., and Ángeles-Hernández, H. (2024). Semi-Supervised Training for (Pre-Stack) Seismic Data Analysis. *Applied Sciences (Switzerland)*, 14(10):1–23.

[36] Erhan, D., Courville, A., Bengio, Y., and Vincent, P. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 9(2007):201–208.

[37] Estiri, H., Abounia Omran, B., and Murphy, S. N. (2018). kluster: An Efficient Scalable Procedure for Approximating the Number of Clusters in Unsupervised Learning. *Big Data Research*, 13:38–51.

[38] Fields, T., Hsieh, G., and Chenou, J. (2019). Mitigating drift in time series data with noise augmentation. In *6th Annual Conference on Computational Science and Computational Intelligence, CSCI 2019*, pages 227–230. IEEE.

[39] Glaser, I., Sadegharmaki, S., Komboz, B., and Matthes, F. (2021). Data scarcity: Methods to improve the quality of text classification. In *10th International Conference on Pattern Recognition Applications and Methods*, pages 556–564. SciTePress.

[40] Goldberg, X. (2009). *Introduction to semi-supervised learning*, volume 6.

[41] Gonzales, R. G. and Woods, R. E. (2018). *Digital Image Processing. 4th edn.* Pearson, New York.

[42] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning. Final edn.* The MIT Press, Cambridge, Massachusetts, US.

[43] Graves, A. (2013). (2013). Generating Sequences With Recurrent Neural Networks.

[44] Gu, Y., Bao, Z., Song, X., Wei, M., Zang, D., Niu, B., and Lu, K. (2019). Permeability prediction for carbonate reservoir using a data-driven model comprising deep learning network, particle swarm optimization, and support vector regression: a case study of the LULA oilfield. *Arab. J. Geosci.*, 12(20):1–16.

[45] Guo, S. and Wang, H. (2019). Seismic absolute acoustic impedance inversion with L1 norm reflectivity constraint and combined first-and second-order total variation regularizations. *J. Geophys. Eng.*, 16(4):773–788.

[46] Hassanzadeh, H., Kholghi, M., Nguyen, A., and Chu, K. (2018). Clinical Document Classification Using Labeled and Unlabeled Data Across Hospitals. *AMIA ... Annual Symposium proceedings. AMIA Symposium*, 2018:545–554.

[47] He, M., Gu, H., and Wan, H. (2020). Log interpretation for lithology and fluid identification using deep neural network combined with MAHAKIL in a tight sandstone reservoir. *J. Pet. Sci. Eng.*, 194(May):107498.

[48] Hilbert, M. (2022). Information Quantity. *Encyclopedia of Big Data*, pages 568–571.

[49] Islam, M. S. u. (2020). Using deep learning based methods to classify salt bodies in seismic images. *J. Appl. Geophys.*, 178:104054.

[50] Kaur, H., Fomel, S., and Pham, N. (2020). Seismic ground-roll noise attenuation using deep learning. *Geophys. Prospect.*, 68(7):2064–2077.

[51] Kaur, H., Pham, N., and Fomel, S. (2021). Seismic data interpolation using deep learning with generative adversarial networks. *Geophys. Prospect.*, 69(2):307–326.

[52] Kaziha, O. and Bonny, T. (2019). A Comparison of Quantized Convolutional and LSTM Recurrent Neural Network Models Using MNIST. In *2019 International Conference on Electrical and Computing Technologies and Applications, ICECTA 2019*.

[53] Khan, P., Kumar, Y., and Kumar, S. (2022). CapsLSTM-Based Human Activity Recognition for Smart Healthcare With Scarce Labeled Data. *IEEE Trans. Comput. Soc. Syst.*

[54] Kim, D., Choi, J., Kim, D., and Byun, J. (2020). Predicting mineralogy by integrating core and well log data using a deep neural network. *J. Pet. Sci. Eng.*, 195(August):107838.

[55] Kingma, D. P. and Ba, J. L. (2015). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–15.

[56] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communic. ACM.*, 60(6):84–90.

[57] Kumar, T., Park, J., and Ali, M. S. (2021). Binary-Classifiers-Enabled Filters for Semi-Supervised Learning. *IEEE Access*, 9:167663–167673.

[58] Le, Q. V., Jaitly, N., and Hinton, G. E. (2015). A Simple Way to Initialize Recurrent Networks of Rectified Linear Units. *arXiv preprint arXiv:1504.00941*.

[59] Lecun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

[60] Li, S., Liu, B., Ren, Y., Chen, Y., Yang, S., Wang, Y., and Jiang, P. (2020a). Deep-Learning Inversion of Seismic Data. *IEEE Trans. Geosci. Remote Sens.*, 58(3):2135–2149.

[61] Li, Y., Xing, R., Jiao, L., Chen, Y., Chai, Y., Marturi, N., and Shang, R. (2019a). Semi-supervised PolSAR image classification based on self-training and superpixels. *Remote Sensing*, 11(16).

[62] Li, Z., Kang, Y., Feng, D., Wang, X. M., Lv, W., Chang, J., and Zheng, W. X. (2020b). Semi-supervised learning for lithology identification using Laplacian support vector machine. *J. Pet. Sci. Eng.*, 195(April):107510.

[63] Li, Z., Ko, B., and Choi, H.-J. (2019b). Naive semi-supervised deep learning using pseudo-label. *Peer-to-peer Net. Applic.*, 12(February 2018):1358–1368.

[64] Lima, L. A., Görnitz, N., Varella, L. E., Vellasco, M., Müller, K. R., and Nakajima, S. (2017). Porosity estimation by semi-supervised learning with sparsely available labeled samples. *Comput. Geosci.*, 106(September 2016):33–48.

[65] Lin, Z., Gu, Z., Li, Y., Yu, Z., and Li, Y. (2020). Layer-wise pre-training mechanism based on neural network for epilepsy detection. *12th International Conference on Advanced Computational Intelligence, ICACI 2020*, pages 224–227.

[66] Liu, W., Cheng, Q., Liu, L., Wang, Y., and Zhang, J. (2020). Accelerating high-resolution seismic imaging by using deep learning. *Appl. Sci. (Switz.)*, 10(7):2502.

[67] Liu, X., Li, B., Li, J., Chen, X., Li, Q., and Chen, Y. (2021). Semi-supervised deep autoencoder for seismic facies classification. *Geophys. Prospect.*, 69(6):1295–1315.

[68] Lopes, R. L. and Jorge, A. (2017). Mind the Gap: A Well Log Data Analysis. *arXiv*, (May).

[69] Luo, W. P., Li, H. Q., and Shi, N. (2016). Semi-supervised least squares support vector machine algorithm: application to offshore oil reservoir. *Appl. Geophys.*, 13(2):406–415.

[70] M. Dai, A. and V. Le, Q. (2015). Semi-supervised Sequence Learning. *Advances in neural information processing systems*, 34(4):227–230.

[71] Mao, B., Han, L. G., Feng, Q., and Yin, Y. C. (2019). Subsurface velocity inversion from deep learning-based data assimilation. *J. Appl. Geophys.*, 167:172–179.

[72] Marfurt, K. J. (2014). Seismic attributes and the road ahead. In *SEG Technical Program Expanded Abstracts 2014*, pages 4421–4426. Society of Exploration Geophysicists.

[73] Montalt-Tordera, J., Muthurangu, V., Hauptmann, A., and Steeden, J. A. (2021). Machine learning in Magnetic Resonance Imaging: Image reconstruction. *Physica Medica*, 83(February):79–87.

[74] Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., and Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *Journal of Big Data*, 2(1):1–21.

[75] Osman, M. (2023). Wild and Interesting Facebook Statistics and Facts. Available online: https://kinsta.com/blog/facebook-statistics/ (accessed on 01/12/2023).

[76] Ostmeyer, J. and Cowell, L. (2019). Machine learning on sequential data using a recurrent weighted average. *Neurocomputing*, 331:281–288.

[77] Patterson, J. and Gibson, A. (2017). *Deep Learning, a Practitioner's Approach. 1st edn.* O'Reilly Media, Inc., Sebastopol, California, US.

[78] Perez, L. and Wang, J. (2017). The Effectiveness of Data Augmentation in Image Classification using Deep Learning.

[79] Philip Chen, C. L. and Zhang, C. Y. (2014). Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Information Sciences*, 275:314–347.

[80] Poornachandra, S. (2020). *TensorFlow Jump Start*. Mumbai, India.

[81] Pratama, H. and Latiff, A. H. A. (2022). Automated Geological Features Detection in 3D Seismic Data Using Semi-Supervised Learning. *Appl. Sci. (Switz.)*, 12(13):6723.

[82] Quico, C. (2019). Television Reshaped By Big Data: Impacts and Implications for Netflix-Like Platforms in the Age of Dataism. *International Journal of Film and Media Arts*, 4(1):48–55.

[83] Radwan, A. E., Wood, D. A., and Radwan, A. A. (2022). Machine learning and data-driven prediction of pore pressure from geophysical logs: A case study for the Mangahewa gas field, New Zealand. *J. Rock Mech. Geotech. Eng.*, 14(6):1799–1809.

[84] Raschka, S. and Mirjalili, V. (2019). *Python Machine Learning. 3rd Edn.* Packt Publishing Ltd., Birmingham, UK.

[85] Ravasi, M. and Vasconcelos, I. (2020). PyLops—A linear-operator Python library for scalable algebra and optimization. *SoftwareX*, 11:100361.

[86] Ren, Y., Nie, L., Yang, S., Jiang, P., and Chen, Y. (2021). Building Complex Seismic Velocity Models for Deep Learning Inversion. *IEEE Access*, 9:63767–63778.

[87] Roy Chowdhury, K. (2011). Seismic Data Acquisition and Processing.

[88] Runge, J. (2019). (2019). TIGRAMITE – Causal discovery for time series datasets.

[89] Runge, J., Nowack, P., Kretschmer, M., Flaxman, S., and Sejdinovic, D. (2019). Detecting and quantifying causal associations in large nonlinear time series datasets. *Science Advances*, 5(11):eaau4996.

[90] Russell, S. and Norvig, P. (2010). *Artificial Intelligence A Modern Approach Third Edition*.

[91] Sagheer, A. and Kotb, M. (2019). Unsupervised Pre-training of a Deep LSTM-based Stacked Autoencoder for Multivariate Time Series Forecasting Problems. *Scient. Rep.*, 9(1):19038.

[92] Saikia, P., Baruah, R. D., Singh, S. K., and Chaudhuri, P. K. (2020). Artificial Neural Networks in the domain of reservoir characterization: A review from shallow to deep models. *Comput. Geosci.*, 135(March 2019):104357.

[93] Sarkar, A., Gepperth, A., Handmann, U., and Kopinski, T. (2017). Dynamic hand gesture recognition for mobile systems using deep LSTM. In *Intelligent Human Computer Interaction: 9th International Conference, IHCI 2017, Evry, France, December 11-13, 2017, Proceedings 9*, pages 19–31. Springer International Publishing.

[94] Schak, M. and Gepperth, A. (2019). A Study on Catastrophic Forgetting in Deep LSTM Networks. In Tetko, I., Kurková, V., Karpov, P., and Theis, F., editors, *28th International Conference on Artificial Neural Networks, ICANN 2019*, volume 11728 LNCS, pages 714–728, Munich, Germany. Springer Verlag.

[95] Schuster, M. (2000). Better generative models for sequential data problems: Bidirectional recurrent mixture density networks. In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13 - Proceedings of the 2000 Conference, NIPS 2000*, pages 589–594, Denver, CO, United states. Neural information processing system foundation.

[96] Shewale, R. (2023). 44 Netflix Statistics In 2023 (Users, Revenue & Insights). Available online: https://www.demandsage.com/netflix-subscribers/ (accessed on 01/12/2023).

[97] Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1).

[98] Song, H., Gao, Y., Chen, W., Xue, Y. j., Zhang, H., and Zhang, X. (2020). Seismic random noise suppression using deep convolutional autoencoder neural network. *J. Appl. Geophys.*, 178:104071.

[99] Song, Z., Li, S., He, S., Yuan, S., and Wang, S. (2022). Gas-Bearing Prediction of Tight Sandstone Reservoir Using Semi-Supervised Learning and Transfer Learning. *IEEE Geosci. Remote Sens. Lett.*, 19:1–5.

[100] Srivastava, N., Mansimov, E., and Salakhutdinov, R. (2015). Unsupervised learning of video representations using LSTMs. In Bach, F. and Blei, D., editors, *32nd International Conference on Machine Learning, ICML 2015*, volume 1, pages 843–852, Lile, France. International Machine Learning Society (IMLS).

[101] Su-Mei, H., Zhao-Hui, S., Meng-Ke, Z., San-Yi, Y., and Shang-Xu, W. (2022). Incremental semi-supervised learning for intelligent seismic facies identification. *Appl. Geophys.*, 19(1):41–52.

[102] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 27 - 28th Annual Conference on Neural Information Processing Systems 2014, NIPS 2014*, volume 4, pages 3104–3112, Montreal, QC, Canada. Neural information processing systems foundation.

[103] Tajbakhsh, N., Shin, J. Y., Gurudu, S. R., Hurst, R. T., Kendall, C. B., Gotway, M. B., and Liang, J. (2017). Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning? *arXiv*, 35(5):1299–1312.

[104] Tembely, M., AlSumaiti, A. M., and Alameri, W. (2020). A deep learning perspective on predicting permeability in porous media from network modeling to direct simulation. *Comput. Geosci.*, 24(4):1541–1556.

[105] Tiago, C., Gilbert, A., Beela, A. S., Aase, S. A., Snare, S. R., Šprem, J., and McLeod, K. (2022). A Data Augmentation Pipeline to Generate Synthetic Labeled Datasets of 3D Echocardiography Images Using a GAN. *IEEE Access*, 10(September):98803–98815.

[106] Van der Toorn, J., Martínez, G. C., Hanson, G., Tariq, H. H., Shalaby, H., van der Molen, M., and Shah, Z. A. (2021). Time-to-depth conversion. In *Applied Techniques to Integrated Oil and Gas Reservoir Characterization: A Problem-Solution Discussion with Geoscience Experts*, pages 213–230. Elsevier.

[107] van der Walt, S. and Schönberger, J. (2019). (2019). Scikit-Image: Image Processing in Python.

[108] van Es, K. (2023). Netflix & Big Data: The Strategic Ambivalence of an Entertainment Company. *Television and New Media*, 24(6):656–672.

[109] Wang, B., Li, L., Nakashima, Y., Kawasaki, R., Nagahara, H., and Yagi, Y. (2021). Noisy-LSTM: Improving Temporal Awareness for Video Semantic Segmentation. *IEEE Access*, 9:46810–46820.

[110] Wang, D. and Chen, J. (2018). Supervised speech separation based on deep learning: An overview. *IEEE/ACM Trans. Audio Speech Lang. Proc.*, 26(10):1702–1726.

[111] Wang, Y. and Tian, F. (2016). Recurrent residual learning for sequence classification. In Su, J., Duh, K., and Carreras, X., editors, *2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016*, pages 938–943, Austin, TX, United states. Association for Computational Linguistics (ACL).

[112] Xie, Q., Luong, M. T., Hovy, E., and Le, Q. V. (2020). Self-training with noisy student improves imagenet classification. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 10684–10695.

[113] Xu, K., Shen, X., Yao, T., Tian, X., and Mei, T. (2018). GREEDY LAYER-WISE TRAINING OF LONG SHORT TERM MEMORY NETWORKS. In *2018 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 1–6. IEEE.

[114] Yang, L., Huang, B., Guo, S., Lin, Y., and Zhao, T. (2023a). A Small-Sample Text Classification Model Based on Pseudo-Label Fusion Clustering Algorithm. *Appl. Sci. (Switz.)*, 13(8):4716.

[115] Yang, L. and Sun, S. Z. (2020). Seismic horizon tracking using a deep convolutional neural network. *J. Pet. Sci. Eng.*, 187(July 2019):106709.

[116] Yang, N., Li, G., Zhao, P., Zhang, J., and Zhao, D. (2023b). Porosity prediction from pre-stack seismic data via a data-driven approach. *J. Appl. Geophys.*, 211(February):104947.

[117] Yang, X., Song, Z., King, I., and Xu, Z. (2022). A Survey on Deep Semi-Supervised Learning. *IEEE Trans. Knowl. Data Eng.*, 35:8934–8954.

[118] Yang, Y. and Shafiq, M. O. (2018). Large Scale and Parallel Sentiment Analysis Based on Label Propagation in Twitter Data. *Proceedings - 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications and 12th IEEE International Conference on Big Data Science and Engineering, Trustcom/BigDataSE 2018*, pages 1791–1798.

[119] Yao, L., Mao, C., and Luo, Y. (2019). Graph Convolutional Networks for Text Detection. In *Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, page 19.

[120] You, Y., Hseu, J., Ying, C., Demmel, J., Keutzer, K., and Hsieh, C. J. (2019). Large-batch training for LSTM and beyond. In *International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16.

[121] Zhang, J., Liu, Z., Zhang, G., Yan, B., Ni, X., and Xie, T. (2022). Simultaneous prediction of multiple physical parameters using gated recurrent neural network: Porosity, water saturation, shale content. *Front. Earth Sci.*, 10(August):984589.

[122] Zhou, H.-W. (2014). *Practical seismic data analysis*. Cambridge University Press.

[123] Zoph, B., Ghiasi, G., Lin, T. Y., Cui, Y., Liu, H., Cubuk, E., and Le, Q. V. (2020). Rethinking pre-training and self-training. *Adv. Neu. Inf. Proc. Syst.*, 33:3833–3845.