

Characterization and Control of Surface Structures in Laser-based Processing using Machine/Deep Learning

Anastasis Litsas

Thesis submitted in partial fulfillment of the requirements for the
Masters' of Science degree in Computer Science and Engineering

University of Crete
School of Sciences and Engineering
Computer Science Department
Voutes University Campus, 700 13 Heraklion, Crete, Greece

Thesis Advisor: Dr *Yannis Pantazis*

This work has been performed at the University of Crete, School of Sciences and Engineering, Computer Science Department.

The work has been supported by the Foundation for Research and Technology - Hellas (FORTH), Institute of Computer Science (ICS).


UNIVERSITY OF CRETE
COMPUTER SCIENCE DEPARTMENT


**Characterization and Control of Surface Structures in Laser-based
Processing using Machine/Deep Learning**


Thesis submitted by
Anastasis Litsas
in partial fulfillment of the requirements for the
Masters' of Science degree in Computer Science

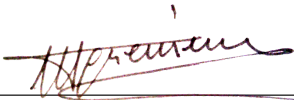
THESIS APPROVAL

Author: 
Anastasis Litsas

Committee approvals: 
Ioannis Tsamardinos
Professor, Committee Member


Nikolaos Komodakis
Associate Professor, Committee Member


Yannis Pantazis
Principal Researcher, Thesis Supervisor

Departmental approval: 
Polyvios Pratitakis
Associate Professor, Director of Graduate Studies

Heraklion, March 2024

Characterization and Control of Surface Structures in Laser-based Processing using Machine/Deep Learning

Abstract

The application of Convolutional Neural Networks (CNNs) for the characterization and control of surface structures in laser-based processing represents a groundbreaking integration of deep learning within the domain of materials science. This thesis presents the development of CNN models to analyze and extract complex spatial features from images of morphological patterns created by laser-material interactions. These patterns, critical for material properties, pose a challenge in accurately predicting laser parameters, a key factor for advancing laser processing technologies. In this work, we employ both Single Task Learning (STL) and Multi-Task Learning (MTL) techniques.

The first part of the thesis is dedicated to STL experiments, where predictions are made for each parameter separately. In the second part, which analyzes the MTL technique, predictions of some or all target parameters are assessed. Experiments were conducted utilizing both synthetic and real data. STL performed exceptionally well on real data, achieving near-perfect results on synthetic data. Although improvements were observed in some predictions in the case of MTL, its overall performance did not manage to surpass STL.

Χαρακτηρισμός και Έλεγχος Δομών Επιφάνειας σε Επεξεργασία με Λείζερ χρησιμοποιώντας Μηχανική/Βαθιά Μάθηση

Περίληψη

Η εφαρμογή των Συνελικτικών Νευρωνικών Δικτύων (CNNs) για τον χαρακτηρισμό και τον έλεγχο των επιφανειακών δομών που προήλθαν από επεξεργασία λέιζερ αποτελεί μια πρωτοποριακή ενσωμάτωση της βαθιάς μάθησης στον τομέα της επιστήμης των υλικών. Αυτή η διπλωματική εργασία παρουσιάζει την ανάπτυξη μοντέλων CNN για να αναλύσουν και να εξάγουν χαρακτηριστικά από εικόνες μορφολογικών μοτίβων που δημιουργήθηκαν από αλληλεπιδράσεις λέιζερ με υλικά. Αυτά τα μοτίβα, κρίσιμα για τις ιδιότητες των υλικών, αποτελούν μια πρόκληση στην ακριβή πρόβλεψη των παραμέτρων του λέιζερ, έναν καίριο παράγοντα για την προώθηση των τεχνολογιών επεξεργασίας με λέιζερ. Στην εργασία αυτή χρησιμοποιούμε τεχνικές "Single Task Learning" (STL) και "Multi-Task Learning" (MTL).

Το πρώτο μέρος της διπλωματικής εργασίας είναι αφιερωμένο σε πειράματα STL, όπου οι προβλέψεις γίνονται για κάθε παράμετρο ξεχωριστά. Στο δεύτερο μέρος, το οποίο αναλύει την τεχνική MTL, αξιολογούνται οι προβλέψεις ορισμένων ή όλων των παραμέτρων-στόχων. Τα πειράματα διεξήχθησαν χρησιμοποιώντας τόσο συνθετικά όσο και πραγματικά δεδομένα. Το STL είχε εξαιρετικά καλά αποτελέσματα σε πραγματικά δεδομένα, επιτυγχάνοντας σχεδόν άριστα αποτελέσματα στα συνθετικά. Αν και παρατηρήθηκαν βελτιώσεις σε ορισμένες προβλέψεις στην περίπτωση του MTL, η συνολική του απόδοση δεν κατάφερε να ξεπεράσει το STL.

Ευχαριστίες

Η ολοκλήρωση της διπλωματικής μου εργασίας δεν θα ήταν δυνατή χωρίς κάποιους συγκεκριμένους ανθρώπους γύρω μου, και σε αυτούς οφείλω ένα μεγάλο ευχαριστώ.

Πρώτα απ'όλα, ευχαριστώ τον επιβλέποντα μου, κ. Ιωάννη Πανταζή, για την πολύτιμη καθοδήγηση και υποστήριξη του όλου αυτόν τον καιρό. Η εμπειρία του και η επιμονή για την λεπτομέρεια συνέβαλαν σημαντικά στην ολοκλήρωση αυτής της εργασίας.

Επιπλέον ευχαριστώ θερμά τους φίλους μου και την κοπέλα μου για την συμπαράσταση, την υπομονή και την αισιοδοξία τους. Η βοήθεια τους, τόσο σε πρακτικό όσο και σε ψυχολογικό επίπεδο, ήταν απαραίτητη για να ολοκληρώσω αυτό το εγχείρημα.

Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου, την γιαγιά μου και την αδελφή μου, για την στήριξη και την πίστη τους σε εμένα.

στους γονείς μου

Contents

Table of Contents	iii
List of Tables	v
List of Figures	viii
1 Introduction	1
2 Laser Processing Background	5
2.1 Basic Principles of Ultrafast Laser Processing	5
2.2 Surface Structures: LIPSS and DLIP	5
2.3 Challenges in Laser Processing	6
2.4 Data Acquisition: Real and Synthetic Datasets	7
2.4.1 Real Dataset	7
2.4.2 Synthetic Dataset	9
3 Deep Learning Background	11
3.1 Introduction to Machine Learning and Deep Learning	11
3.1.1 Forward and Backpropagation	12
3.1.2 Activation Functions	13
3.1.2.1 Sigmoid Function	13
3.1.2.2 Tanh Function	13
3.1.2.3 ReLU Function	14
3.1.2.4 Leaky ReLU	14
3.1.2.5 Softmax Function	14
3.1.2.6 SELU and ELU	14
3.1.2.7 Parametric ReLU (PReLU)	15
3.1.3 Regularization	15
3.1.4 Optimization Techniques	17
3.2 Convolutional Neural Networks	18
3.2.1 Convolutional Neural Networks (CNNs)	18
3.2.1.1 Convolution Operation	18
3.2.1.2 Non-Linearity	19
3.2.1.3 Pooling/Subsampling	19

3.2.1.4	Fully Connected Layers	19
3.2.1.5	Stride and Padding	19
3.2.1.6	Architecture Overview	20
4	Classification &Regression	21
4.1	Overview of Single Task Learning	21
4.2	Model Selection	21
4.2.1	Convolutional Neural Networks Architecture	21
4.2.2	Training Details and Loss Functions	22
4.2.3	Rationale Behind Model Selection: Regression vs. Classification	23
4.2.4	Hyperparameter Optimization and Validation Strategy	23
4.2.5	Training Parameters for CNN Models	24
4.2.6	Final Model Selection	24
5	Multi Task Learning	27
5.1	Introduction to Multi Task Learning	27
5.1.1	Defining Multi Task Learning	27
5.1.2	Architecture for Multi Task Learning	28
5.1.2.1	Shared Base Architecture	28
5.1.2.2	Task-Specific Heads	29
5.1.2.3	Integration and Learning	29
5.1.2.4	Benefits of the Multi-Head Architecture	29
5.2	Multi-Task Learning as Multi-Objective Optimization	29
5.2.1	Definition of Pareto Optimality	30
5.2.2	Training Parameters in MTL	31
6	Results	33
6.1	Data Preparation	33
6.1.1	Data Augmentation	33
6.1.2	Addressing Data Imbalance with Stratified Cross-Validation	33
6.1.3	Comparative Analysis of Cross-Validation Approaches	36
6.2	Results of Single Task Learning Models	36
6.2.1	Baseline Model Performance	36
6.2.2	On the Synthetic Dataset	36
6.2.3	On the Real Dataset	38
6.2.3.1	Classification Task Performance	38
6.2.3.2	Regression Task Performance	38
6.2.3.3	Enhanced Performance through Stratified Cross-Validation	41
6.3	Results of Multi Task Learning Models	41
6.3.1	On the Synthetic Data	41
6.3.1.1	Two Tasks	41
6.3.1.2	Three Tasks	45
6.3.2	On the Real Data	46

6.3.2.1	Two Tasks	46
6.3.2.2	Three Tasks	55
6.3.2.3	Four Tasks	55
7	Conclusion and Future Work	57
	Bibliography	59

List of Tables

2.1	Range of Parameter Values in the Laser Dataset.	8
4.1	Training Parameters for CNN Models on Material Surfaces Dataset	24
4.2	Training Parameters for CNN Models on Energy Profiles Dataset .	25
6.1	Performance Metrics on Synthetic Dataset - Cross Validation . . .	38
6.2	Performance Metrics on Real Datasets - Cross Validation and Strat- ified Cross Validation	39
6.3	Best results for each configuration on Synthetic and Real data. With green color, we indicate the results that outperformed the single task model.	56

List of Figures

2.1	Side-by-side comparison of the processed surfaces (left) and their energy profiles (right) for different pattern types. The top row illustrates the vertical (V) pattern, the middle row the horizontal (H) pattern, and the bottom row the double (D) pattern.	10
3.1	Graphical representations of various activation functions	16
4.1	A graphical representation of the neural network architecture which consists of three convolutional layers followed by two fully-connected (FC) layers. The output layer is adjusted to the task at hand. . . .	22
5.1	A graphical representation of the multi-task neural network architecture, consisting of three convolutional layers followed by two fully connected (FC) layers. The output layer is divided into four distinct heads, with one dedicated to a classification task and the remaining three designed for regression tasks.	28
6.1	Data distribution across various tasks in standard 3-fold Cross-Validation.	34
6.2	Data distribution across various tasks in Stratified 3-fold Cross-Validation.	35
6.3	Confusion matrices for each of the three cross-validation splits on the synthetic dataset.	37
6.4	Average Confusion Matrix across the three stratified folds on real dataset.	39
6.5	A random selection of misclassified samples. The left plots depicts confusing surface structures, while the right plots depicts structures with notable defects.	40
6.6	Pareto fronts for the Pattern Type and Angle Task Configuration across three cross-validation splits on the synthetic dataset, with the top image representing fold 1, the middle image fold 2, and the bottom image fold 3.	42
6.7	Pareto fronts for the Pattern Type and Fluence Task Configuration across three cross-validation splits on the synthetic dataset, with the top image representing fold 1, the middle image fold 2, and the bottom image fold 3.	43

6.8	Pareto fronts for the Fluence and Angle Task Configuration across three cross-validation splits on the synthetic dataset, with the top image representing fold 1, the middle image fold 2, and the bottom image fold 3.	44
6.9	Pareto fronts for the simultaneous optimization of three tasks on the synthetic dataset: pattern type classification, fluence prediction, and angle prediction. The plots represent the trade-offs achieved between tasks across three folds, highlighting the convergence towards Pareto optimal solutions. The top image corresponds to fold 1, the middle to fold 2, and the bottom to fold 3.	47
6.10	Confusion matrices for the pattern type classification task within the three-task MTL framework on the synthetic dataset. Each matrix corresponds to one of the three folds, illustrating the classification distribution. Fold 1 (a) predominantly assigns the label V, fold 2 (b) leans towards label H, and fold 3 (c) shows a preference for label D.	48
6.11	Pareto fronts for the Pattern Type and Angle Task Configuration across three cross-validation splits on the real dataset, with the top image representing fold 1, the middle image fold 2, and the bottom image fold 3.	49
6.12	Pareto fronts for the Pattern Type and Fluence Task Configuration across three cross-validation splits on the real dataset, with the top image representing fold 1, the middle image fold 2, and the bottom image fold 3.	50
6.13	Pareto fronts for the Fluence and Angle Task Configuration across three cross-validation splits on the real dataset, with the top image representing fold 1, the middle image fold 2, and the bottom image fold 3.	51
6.14	Pareto fronts for the Number of Pulses and Angle Task Configuration across three cross-validation splits on the real dataset, with the top image representing fold 1, the middle image fold 2, and the bottom image fold 3.	52
6.15	Pareto fronts for the Number of Pulses and Fluence Task Configuration across three cross-validation splits on the real dataset, with the top image representing fold 1, the middle image fold 2, and the bottom image fold 3.	53
6.16	Pareto fronts for the Number of Pulses and Pattern Type Task Configuration across three cross-validation splits on the real dataset, with the top image representing fold 1, the middle image fold 2, and the bottom image fold 3.	54

Chapter 1

Introduction

Ultrafast laser processing has emerged as a transformative tool in materials science, enabling the creation of intricate surface structures on a variety of materials, including metals, ceramics, plastics, thin films, and glasses. Laser technology has a huge impact in manufacturing with applications including marking, cutting, engraving, welding, ablation and additive manufacturing [30, 10, 6]. The immense size of industrial and academic applications can be essentially attributed to the large number of configurable laser parameters such as fluence (or pulse energy), pulse length, polarization, wavelength and beam size. More complex irradiation schemes with multiple beams, temporally delayed pulses [12, 11], incident angles and varying number of pulses further increase the flexibility and thus the control of the laser fabrication process.

The ability to tailor the spatiotemporal beam profile in ultrafast laser processing holds great promise for achieving complex patterns with desirable biomimetic textures and properties [37]. Such capabilities open avenues for innovations in various sectors, especially in domains that rely on precision and material specificity.

However, the road to optimizing these surface structures is faced with challenges. The resulting surface morphologies are influenced by a multitude of factors, including the interplay between laser parameters and intrinsic material properties. Predicting and controlling these outcomes has proven to be a complex endeavor, one that traditional methods have struggled with. While mathematical modeling of laser-matter phenomena holds the promise of reducing extensive experimentation, accurately capturing these phenomena is a daunting task. It becomes especially challenging and computationally demanding when addressing non-linear behaviors at femtosecond temporal scales. Furthermore, uncertainties and errors inherent to the modeling process add layers of complexity to the optimization procedure.

In the contemporary era of data-driven solutions, machine learning, particularly Convolutional Neural Networks (CNNs), offers a promising approach to address the challenges of laser processing. CNNs' prowess in image recognition and analysis makes them a suitable candidate for predicting the post-fabrication surface morphologies based on given laser parameters [?, 22].

This thesis aims to delve deep into the potential of CNNs in revolutionizing ultrafast laser processing. Building upon previous research, we put forward five distinct models. The initial four models are evolved versions of those outlined in prior work, each offering a unique approach to tackling the predictive challenge posed by laser-induced patterns. These models are characterized by their innovative architectures, designed to capture the intricate spatial features of these patterns, and they each address specific aspects of the predictive task, demonstrating the strengths and limitations of varied CNN configurations.

In a significant advancement beyond these initial models, this thesis introduces a novel fifth model, employing a Multi-Task Learning (MTL) framework. This multi-task model is engineered to optimize the prediction of multiple laser parameters simultaneously, a step that not only capitalizes on the shared information across tasks but also aims to enhance the overall predictive performance of the system. By integrating and extending the foundational work established by the initial four models, this multi-task approach represents a comprehensive effort to harness the full potential of CNNs in the context of laser processing, setting a new precedent for the field.

As we explore the existing literature, it is evident that machine learning, and deep learning in particular, has made substantial inroads across a broad spectrum of scientific and engineering disciplines. This has led to the advent of data-driven and ML-assisted strategies in laser processing in recent years [27, 15]. The endeavor to predict the patterns of LIPSS-processed materials from laser parameters has been documented [38]. However, the literature on inferring laser parameters from images, especially those with surface structures produced through the Direct Laser Interference Patterning (DLIP) method, is remarkably sparse. To our knowledge, there exists no study that addresses the prediction of laser parameters from such images. Considering the quasi-periodic patterns evident in the recorded images, this scenario aligns closely with the field of texture analysis from a deep learning perspective. Although there exists a plethora of research employing CNNs for texture classification in a variety of contexts, including material processing [4, 25, 3, 8, 32, 14, 23], the peculiar challenges posed by the small size of our dataset render traditional texture classification methods less viable. This limitation highlights the necessity for novel analytical approaches in the examination and interpretation of laser-processed materials, thereby framing the thematic direction of this thesis.

Throughout this research, readers will be acquainted with the intricacies of each model, encompassing both the challenges encountered and the innovations achieved. This journey represents not merely a technical exploration but also exemplifies the transformative power of machine learning in revolutionizing the domain of materials science.

More specifically, the research delves into the following areas:

- **Laser Processing Background:** Insights into the distinct methods of laser processing, specifically Laser-Induced Periodic Surface Structures and Direct Laser Interference Patterning, will be provided. This section will elaborate

on the data acquisition methodologies and shed light on the nuances and challenges related to real and synthetic datasets.

- **Machine/Deep Learning Background:** A foundational exposition on the principles of machine and deep learning, emphasizing their pertinence to this research.
- **Single-Task Experiments:** A thorough discussion on the design, methodology, and implementation of single-task models.
- **Multi-Task Approach:** Comprehensive analysis of the multi-task model, underscoring its foundational design principles, merits, and inherent challenges.
- **Experiments and Results:** An exhaustive overview of the experimental designs, methodologies, and consequential discoveries.
- **Conclusion and Future Work:** Reflection on the entirety of the study, its broader implications, and potential avenues for subsequent exploration.

Chapter 2

Laser Processing Background

2.1 Basic Principles of Ultrafast Laser Processing

Ultrafast laser processing has revolutionized the field of materials science, offering an unparalleled avenue to interact with materials at atomic and molecular levels. This interaction, characterized by femtosecond to picosecond pulse durations, enables precise control over material removal and modification processes. The primary advantage of ultrafast laser processing is the minimal heat transfer to the surrounding regions, ensuring reduced thermal damage and allowing for high precision.

The efficacy of ultrafast laser processing is a result of its ability to induce non-linear absorption phenomena in materials. When materials are exposed to high-intensity ultrafast laser pulses, they exhibit multi-photon absorption and tunnel ionization, leading to rapid electron excitation. Subsequent electron-phonon interactions result in localized heating, facilitating precise material modifications. The swift nature of these processes ensures that the bulk material remains largely unaffected, preserving the material's intrinsic properties.

2.2 Surface Structures: LIPSS and DLIP

In the realm of ultrafast laser processing, two prominent techniques for surface structuring emerge: Laser-Induced Periodic Surface Structures (LIPSS) and Direct Laser Interference Patterning (DLIP).

LIPSS: LIPSS are periodic structures that form on material surfaces post-irradiation with ultrafast laser pulses. These structures are typically characterized by spatial periodicities close to or smaller than the laser wavelength. LIPSS formation is influenced by various factors, including laser parameters, material properties, and environmental conditions. There are two primary types of LIPSS: high spatial frequency LIPSS (HSFL) and low spatial frequency LIPSS (LSFL), each with its characteristic morphology and formation mechanism. This fabrication approach

produces patterns that resemble ripples, grooves or spikes in a self-organized manner [34, 5, 24].

DLIP: DLIP stands out for its exceptional capability to fabricate large-area periodic structures on material surfaces with high precision and versatility [2, 13]. This technique leverages the interference of multiple coherent laser beams to imprint a wide array of geometric patterns, ranging from simple linear gratings to intricate hierarchical structures. A key strength of DLIP lies in its adaptability; by fine-tuning the laser parameters and interference setup, it is possible to tailor the pattern geometries and spatial periodicities to specific application requirements.

For the scope of this thesis, the data and analyses in this work pertain solely to the DLIP method.

2.3 Challenges in Laser Processing

Laser processing, enhanced by machine learning’s analytical capabilities, presents significant challenges that this thesis addresses to optimize the fabrication of precise surface structures. The intricacies involved in the laser-material interaction are primarily governed by a complex parameter space, where settings such as fluence, the number of pulses, pattern type, and the angle of incidence interact in a highly non-linear manner, influencing the final surface morphology. This complexity is compounded by the unique responses of different materials to laser processing, necessitating a tailored approach to predict and achieve the desired outcomes accurately.

Compounding the difficulty is the scarcity of high-quality, labeled data which is crucial for the successful training and application of machine learning models. Data acquisition is often an expensive, time-consuming, and labor-intensive process, particularly when it requires capturing subtle differences in surface morphologies. Moreover, the non-linear phenomena that characterize laser-material interactions pose significant challenges to mathematical modeling, demanding advanced computational models that are capable of learning from data and generalizing to new, unseen examples.

These modeling efforts are further strained by computational constraints, as simulating laser processing with high fidelity demands substantial computational resources. Such simulations must navigate a high-dimensional parameter space with adequate temporal and spatial resolution to be effective.

This thesis stands at the confluence of physics, materials science, and engineering and aims to integrate knowledge from these domains to address the challenges outlined. In doing so, it contributes to the advancement of laser processing techniques, harnessing the power of machine learning to expand the possibilities of surface engineering.

2.4 Data Acquisition: Real and Synthetic Datasets

The precision of machine learning models in laser processing is contingent upon the availability of comprehensive and representative data. The challenges highlighted in Section 2.3 underscore the limitations of a sparse parameter space and the small size of available datasets. In response to these challenges, the acquisition of both real and synthetic datasets becomes paramount. The real dataset, meticulously curated from experimental procedures, offers authentic insights into the laser processing phenomena. However, to overcome the limitations posed by its size and parameter coverage, a synthetic dataset is constructed. This dataset serves to extend the parameter space and provides a controlled environment to evaluate the models thoroughly. Sections 2.4.1 and 2.4.2 detail the composition and generation of the real and synthetic datasets, respectively, showcasing their roles in addressing the data-related challenges of this study.

2.4.1 Real Dataset

One of the dataset used in this study, we also refer to this real dataset as material surfaces, provided by the ultra-fast laser micro/nano-processing group at IESL-FORTH, encompasses 208 high-resolution images (1280×960 pixels) showcasing a wide array of structured patterns on stainless steel surfaces. These images were produced through the application of varied laser parameters characteristic of the DLIP technique, and each image is annotated with the specific laser parameters that facilitated its creation.

DLIP is a sophisticated technique that harnesses the interference phenomenon of laser pulses to craft periodic nanostructures on material surfaces. The spatial intensity distribution resultant from this process can be modeled by two prominent patterns, derived from the interference of two and four laser beams, which in turn dictate the orientation of the surface patterns.

The spatial intensity distribution for the two-beam interference, denoted as $I^{(2)}$, is mathematically encapsulated by Equation (2.1), while the distribution for the four-beam interference, denoted as $I^{(4)}$, is articulated by Equation (2.2). Here, λ_L signifies the laser wavelength in nanometers, x and y represent the spatial coordinates, θ is the incident angle in degrees, R_0 stands for the beam width in micrometers, corresponding to the Gaussian beam's full width at half maximum, and F represents the fluence in millijoules.

The dataset has been curated with a diverse array of values for pattern type, incident angle, fluence, and the number of pulses. A summarization of these laser parameters is presented in Table I. The dataset comprises 105 images with the V pattern, 51 images with the H pattern, and 52 images with the D pattern, indicating a slight imbalance. Due to the intricate requirements in laser reconfiguration, the angle parameter, though continuous, is restricted to only three distinct values. Similarly, while the number of pulses is a continuous variable, the dataset covers a sufficient range, given the logarithmic influence of this parameter on the surface

Table 2.1: Range of Parameter Values in the Laser Dataset.

Parameter	Values
Pattern type	V, H, D
Angle (θ)	5.5, 8.5, 19
Fluence (F)	14 - 94
Number of Pulses (NP)	10, 20, 50, 100, 500
Laser wavelength (λ_L)	1026
Beam width (R_0)	150

structures. The fluence, another continuous variable, exhibits a mean value of 36.45 and a standard deviation of 15.13 across the dataset.

$$\begin{aligned}
I^{(2)} = F \times & \left[1 + \frac{1}{2} \cos \left(\frac{4\pi}{\lambda_L} x \sin(2\theta) \right) \right. \\
& \left. + 2 \cos \left(\frac{4\pi}{\lambda_L} x \sin(2\theta) \right) \cos \left(\frac{4\pi}{\lambda_L} x \sin(2\theta) \right) \right] \\
& \times e^{-4 \log(2) \left(\frac{x^2 + y^2}{R_0^2} \right)}
\end{aligned} \tag{2.1}$$

$$\begin{aligned}
I^{(4)} = F \times & \left[1 + \frac{1}{2} \cos \left(\frac{4\pi}{\lambda_L} x \sin(2\theta) \right) \right. \\
& + \frac{1}{2} \cos \left(\frac{4\pi}{\lambda_L} y \sin(2\theta) \right) \\
& \left. + 2 \cos \left(\frac{4\pi}{\lambda_L} y \sin(2\theta) \right) \cos \left(\frac{4\pi}{\lambda_L} x \sin(2\theta) \right) \right] \\
& \times e^{-4 \log(2) \left(\frac{x^2 + y^2}{R_0^2} \right)}
\end{aligned} \tag{2.2}$$

2.4.2 Synthetic Dataset

To address the constraints imposed by the limited size of the real dataset, a synthetic dataset was constructed to emulate the DLIP process parameters faithfully. This dataset was informed by the energy profiles derived from the fundamental laser generation mechanism, as expressed by Equations 2.1 and 2.2. By systematically altering the DLIP parameters—such as fluence (F), incident angle (θ), and pattern type, a synthetic dataset that spans the entire spectrum of potential DLIP configurations was synthesized.

This dataset was generated through uniform random sampling of the DLIP parameters. For pattern types, the selection was made from the vertical (V), horizontal (H), and double (D) configurations. The incident angle was uniformly sampled from the interval $[4, 20]$ degrees, and the laser fluence from the range $[10, 100]$. A total of 100,000 synthetic images were produced, providing a robust dataset that captures a balanced representation of the parameter space, crucial for the comprehensive evaluation of the models.

Figure 2.1 presents a side-by-side comparison of the actual laser-processed surfaces and their associated energy profiles for each pattern type. The sequence of rows delineates the distinct patterns: the top row for the vertical (V) pattern, the middle row for the horizontal (H) pattern, and the bottom row for the double (D) pattern.

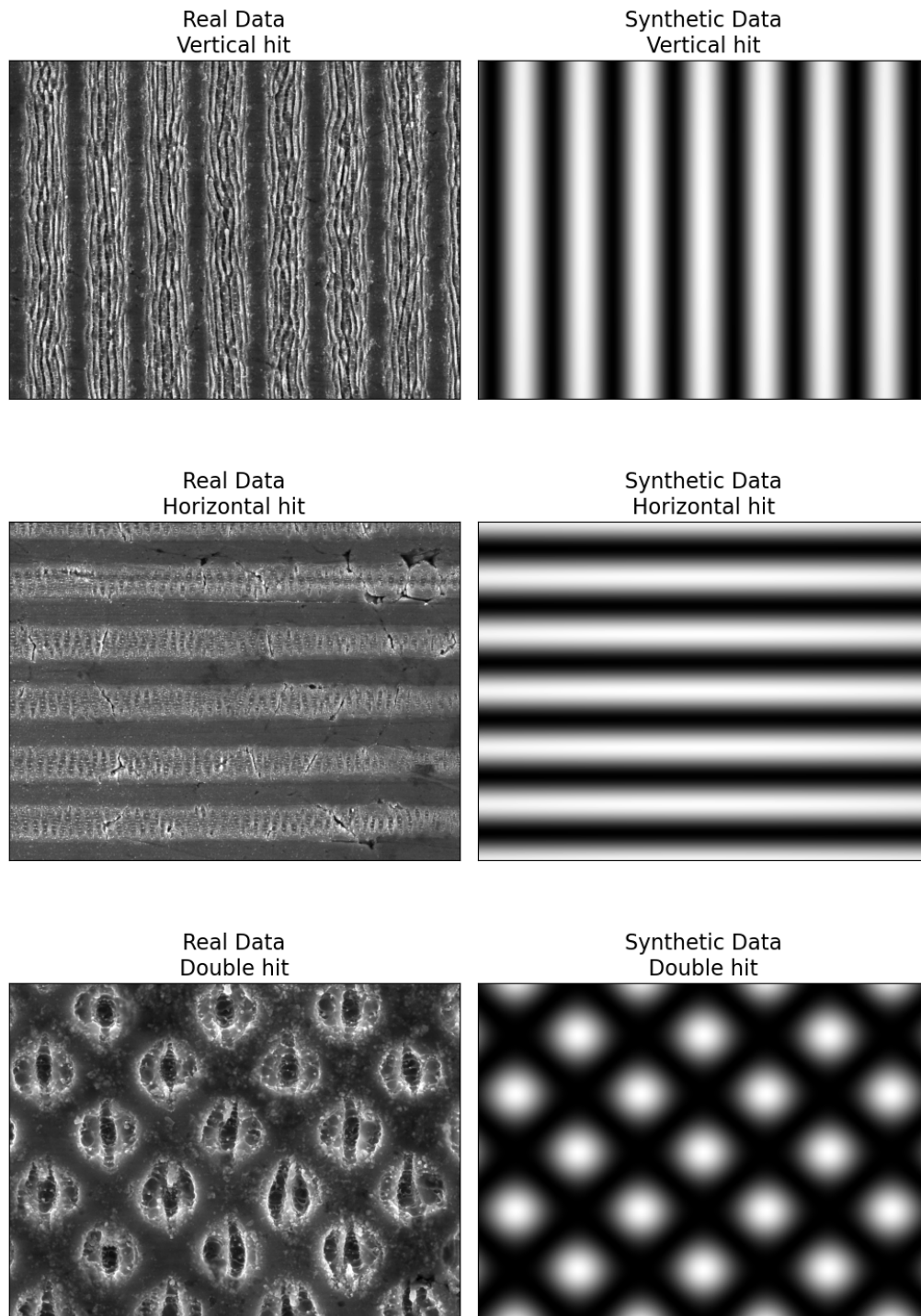


Figure 2.1: Side-by-side comparison of the processed surfaces (left) and their energy profiles (right) for different pattern types. The top row illustrates the vertical (V) pattern, the middle row the horizontal (H) pattern, and the bottom row the double (D) pattern.

Chapter 3

Deep Learning Background

3.1 Introduction to Machine Learning and Deep Learning

Deep learning has emerged as a powerful and transformative approach to artificial intelligence (AI), revolutionizing various fields, including computer vision, natural language processing and robotics [21]. Its ability to learn complex patterns and relationships from large amounts of data has led to significant breakthroughs in tasks such as image recognition, speech translation and self-driving cars.

At the heart of deep learning are artificial neural networks (ANNs), which are inspired by the structure and function of the human brain [26]. ANNs are composed of layers, which are collections of neurons or nodes. These layers are categorized into three types. The input layer is the first layer that receives the input signal. Hidden layers, are one or more layers where the actual processing is done through a system of weighted "neurons". These are termed "hidden" as they do not directly interact with the input or output. The final type is the output layer, the final layer that produces the output for given inputs. Each neuron in a layer receives an input signal, processes it, and passes an output signal to the subsequent layer. The strength of the connection between neurons is represented by weights, which are adjusted during training to minimize the difference between the network's prediction and the actual target values.

The depth of ANNs, referring to the number of hidden layers between the input and output layers, is what distinguishes deep learning from traditional machine learning approaches. Deep neural networks (DNNs) are capable of capturing complex patterns and relationships that may be difficult to detect with shallower networks. This increased depth allows DNNs to achieve higher levels of accuracy and performance on a wide range of tasks.

3.1.1 Forward and Backpropagation

Forward propagation in neural networks is the process through which input data is transformed into an output. This transformation occurs layer by layer, starting from the input layer, progressing through the hidden layers, and culminating at the output layer. At each layer, every neuron performs a specific calculation: it takes the inputs, applies weights to them, adds a bias, and finally applies an activation function. This process can be mathematically expressed as follows:

$$h_j^{(l)} = f \left(\sum_i w_{ij}^{(l)} x_i + b_j^{(l)} \right) \quad (3.1)$$

Here, $h_j^{(l)}$ is the output of neuron j in layer l , f denotes a non-linear activation function, $w_{ij}^{(l)}$ represents the weight from neuron i in layer $l-1$ to neuron j in layer l , x_i is the input from neuron i , and $b_j^{(l)}$ is the bias for neuron j .

The role of the activation function is crucial, as it introduces non-linearity into the network, enabling it to learn and model complex patterns. Different types of activation functions, such as sigmoid, ReLU, and tanh, have different mathematical properties and are suitable for different scenarios. Their detailed discussion will be presented in subsection 3.1.2.

The significance of forward propagation lies in its ability to map inputs to outputs in a way that captures the underlying relationships within the data. This capability forms the core of neural networks' function approximation ability. Hornik, Stinchcombe, and White's landmark paper [17] established that a feedforward network with just a single hidden layer, using a "squashing" type activation function, can approximate any Borel measurable function, validating the versatility and power of neural networks as universal approximators.

Backpropagation, on the other hand, is the learning algorithm used to train neural networks. It adjusts the weights and biases of the network in order to minimize the error between the network's predictions and the actual data. This process is iterative and occurs after forward propagation. The core of backpropagation is the computation of the gradient of the error function with respect to each weight in the network. This is achieved using the chain rule, a fundamental tool in calculus:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial h_j} \cdot \frac{\partial h_j}{\partial w_{ij}} \quad (3.2)$$

Here, E represents the error measure, which quantifies the difference between the predicted output and the actual output. The term $\frac{\partial E}{\partial h_j}$ represents the sensitivity of the error to the output of the neuron, and $\frac{\partial h_j}{\partial w_{ij}}$ denotes the sensitivity of the neuron's output to its weight. The process of backpropagation efficiently computes these gradients for all weights and biases in the network, a task that would be computationally infeasible with naive methods for networks of realistic size.

The effectiveness of backpropagation lies in its ability to propagate the error backwards from the output layer to the input layer, adjusting weights in a way that systematically reduces the error. This process not only updates the weights but also provides insight into which features are most important for making accurate predictions.

With the foundational understanding of forward and backpropagation mechanisms in place, it's pivotal to explore the strategies that optimize these processes. The upcoming sections will delve into activation functions, regularization techniques that prevent overfitting, examine various optimization algorithms that refine the performance of neural networks, and explore the diverse architectures that are at the forefront of deep learning advancements. These components are integral to harnessing the full potential of neural networks in complex tasks, including those in the domain of laser-based processing and machine/deep learning applications.

3.1.2 Activation Functions

Activation functions play a crucial role in neural networks by introducing non-linearity, allowing these models to capture complex relationships in data. Without activation functions, or with purely linear activation functions, neural networks would be unable to model anything beyond simple linear relationships, severely limiting their applicability.

The choice of activation function, such as sigmoid, tanh, ReLU, or others, greatly affects the network's learning dynamics and its capacity to handle non-linearities [1, 9, 28]. Below, we discuss some of the well-known activation functions:

3.1.2.1 Sigmoid Function

The sigmoid function, defined as

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (3.3)$$

, squashes its input to a range between 0 and 1. This characteristic makes it suitable for models where we need to predict probabilities. However, it suffers from the vanishing gradient problem, where gradients become increasingly small, impeding the weight update process during backpropagation.

3.1.2.2 Tanh Function

The hyperbolic tangent or tanh function, defined as

$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \quad (3.4)$$

, outputs values between -1 and 1. This makes it zero-centered, improving the efficiency of the learning process compared to the sigmoid function. However, it still suffers from the vanishing gradient problem in layers with high saturation.

3.1.2.3 ReLU Function

The Rectified Linear Unit (ReLU) function, defined as

$$\text{ReLU}(x) = \max(0, x) \quad (3.5)$$

, has gained popularity for its computational simplicity and efficiency in training. ReLU alleviates the vanishing gradient problem significantly, allowing models to learn faster and perform better. However, it can suffer from the "dying ReLU" problem, where neurons can become inactive and stop contributing to the learning process.

3.1.2.4 Leaky ReLU

Leaky ReLU is a variation of ReLU that addresses the dying ReLU problem. It is defined as

$$\text{LeakyReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases} \quad (3.6)$$

, where α is a small, positive parameter. This modification ensures that there is always a gradient, preventing neurons from dying out.

3.1.2.5 Softmax Function

Softmax is often used in the output layer of a neural network for multi-class classification tasks. It transforms the output into a probability distribution proportional to the exponentials of the input numbers. This function is particularly useful in scenarios where we need to classify inputs into multiple categories. The Softmax function for the i -th component of a vector z is given by:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \quad (3.7)$$

3.1.2.6 SELU and ELU

Scaled Exponential Linear Unit (SELU) [20] and Exponential Linear Unit (ELU) [7] are functions designed to improve the learning process by automatically normalizing the outputs. SELU and ELU functions have properties that push the mean and variance of the activations towards zero and one, respectively, which can lead to improved learning dynamics. The SELU function is defined as:

$$\text{SELU}(x) = \begin{cases} \lambda x & \text{if } x > 0 \\ \lambda \alpha (e^x - 1) & \text{if } x \leq 0 \end{cases} \quad (3.8)$$

, where λ and α are predefined constants. The ELU function is similar, with its formula being:

$$\text{ELU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha (e^x - 1) & \text{if } x \leq 0 \end{cases} \quad (3.9)$$

, where α is a hyperparameter.

3.1.2.7 Parametric ReLU (PReLU)

Parametric ReLU, or PReLU, extends the idea of ReLU by introducing a learnable parameter that adapts during training. This allows the function to learn the most appropriate form of non-linearity, offering a balance between the flexibility of ReLU and the problem of dead neurons. The PReLU function is defined as:

$$\text{PReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases} \quad (3.10)$$

In PReLU, α is a learnable parameter.

In summary, the choice of activation function is crucial as it influences the learning ability of the network and its capability to handle complex non-linear patterns. The specific selection depends on the nature of the task at hand and the particular characteristics of the data being modeled. In Figure 3.1, the distinct shapes and unique characteristics of the previously discussed activation functions are depicted, providing a clear visual representation of their transformative behaviors within neural network architectures.

3.1.3 Regularization

Regularization is a critical concept in machine learning that helps prevent overfitting, a scenario where a model learns the training data too well but performs poorly on unseen data. Here, several regularization techniques are outlined, elucidating their mathematical formulation and practical applications.

L_1 (Lasso) and L_2 (Ridge) regularization, equations 3.11 and 3.12 respectively, are two popular methods that impose a penalty on the magnitude of the model parameters during the training process by adding them to the loss function. L_1 regularization adds the absolute value of the coefficients as a penalty term to the loss function, inducing sparsity in the model parameters, meaning that it tends to drive some of the coefficients to zero. This can be useful for feature selection, as it effectively removes irrelevant or unimportant features from the model. On the other hand, L_2 regularization adds the square of the coefficients as a penalty term. That way, it encourages the model parameters to be small but non-zero. This can help to prevent overfitting, as it makes the model less sensitive to noise in the training data.

$$L_1(w) = \lambda \sum_j |w_j|, \lambda \in \mathbb{R}^+ \quad (3.11)$$

$$L_2(w) = \lambda \sum_j w_j^2, \lambda \in \mathbb{R}^+ \quad (3.12)$$

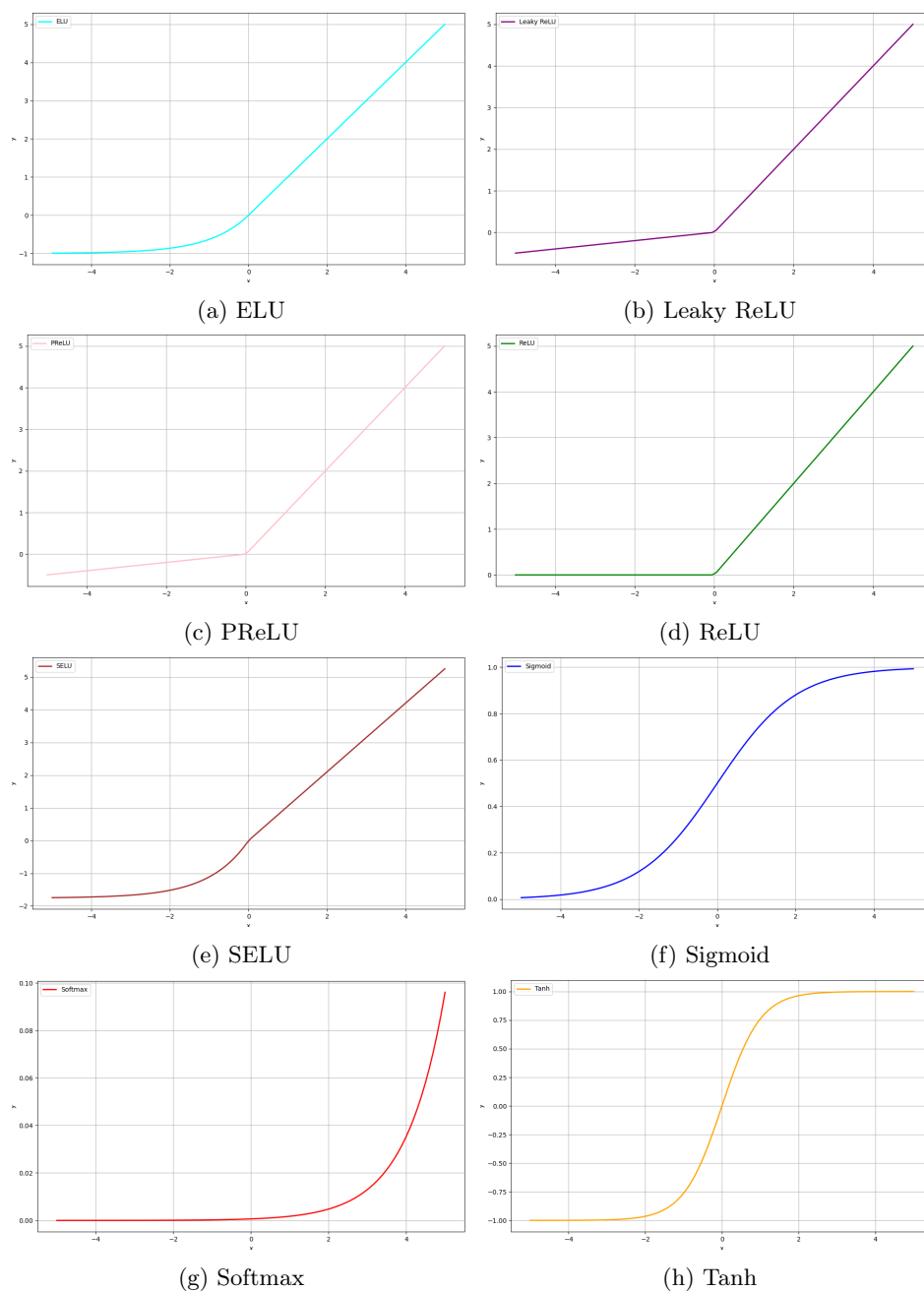


Figure 3.1: Graphical representations of various activation functions

In addition to L_1 and L_2 regularization, several other techniques can be employed to prevent overfitting in deep neural networks. Dropout, introduced by Srivastava et al. in 2014 [33], involves randomly removing a subset of neurons from the network during training. This forces the model to learn redundant features

and is controlled by the dropout rate, which represents the probability that a given neuron will be omitted during a training pass. Another technique, known as early stopping, involves halting the training process when performance on a validation set starts to deteriorate. This prevents overfitting by preventing the network from being trained for too many epochs. Finally, data augmentation is a technique that expands the training dataset by creating modified versions of the existing data. In image processing, this could involve scaling, cropping, rotating, or translating images.

By incorporating these regularization techniques, models are less likely to overfit and more likely to retain a high level of generalization when applied to new data. This balance is crucial for the development of robust machine learning systems.

3.1.4 Optimization Techniques

Optimization in the context of deep learning is the process of adjusting the parameters of neural networks to minimize the loss function. The optimization algorithm chosen for this task can significantly influence the training speed and the ultimate performance of the model.

Gradient descent is a fundamental iterative optimization algorithm used to find the minimum of a function [29]. The weights in the network are updated according to the rule:

$$w = w - \eta \cdot \nabla_w L(w), \quad (3.13)$$

where η is the learning rate, and $\nabla_w L(w)$ is the gradient of the loss function L with respect to the weights w .

In contrast to batch gradient descent, stochastic gradient descent (SGD) uses only a single sample or a mini-batch of samples to perform each update. This can lead to faster convergence, with updates described by:

$$w = w - \eta \cdot \nabla_w L(w; x^{(i)}, y^{(i)}), \quad (3.14)$$

where $x^{(i)}, y^{(i)}$ represent the input features and target output of a single training example or a mini-batch.

To improve upon the convergence properties of SGD, the concept of momentum is often used. Momentum accelerates the optimization in the right direction and dampens oscillations by combining the gradient of the current step with the gradient of the previous step, weighted by a factor γ :

$$v_t = \gamma v_{t-1} + \eta \cdot \nabla_w L(w), \quad (3.15)$$

$$w = w - v_t, \quad (3.16)$$

where v_t is the velocity at time t , and γ is the momentum coefficient.

Adaptive Moment Estimation (Adam) is another optimization method that computes individual adaptive learning rates for different parameters [19]. Adam combines the advantages of two other extensions of stochastic gradient descent,

AdaGrad and RMSProp, and calculates the exponential moving average of the gradients:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \cdot \nabla_w L(w), \quad (3.17)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) \cdot (\nabla_w L(w))^2, \quad (3.18)$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad (3.19)$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}, \quad (3.20)$$

$$w = w - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}}, \quad (3.21)$$

where m_t and v_t are estimates of the first and second moments of the gradients, respectively, β_1 and β_2 are the exponential decay rates for these moment estimates, and ϵ is a small constant that prevents division by zero.

These various optimization techniques are integral to the effective training of deep neural networks. Each technique has its own strengths and is suitable for different scenarios, depending on the nature of the problem and the dataset.

3.2 Convolutional Neural Networks

As our understanding of how deep learning works, and our need to tackle different tasks at hand, various architectures have been developed throughout the years. In the early years of deep learning, Multi-Layer Perceptron (MLP) was considered one of the foundational structures for pattern recognition. However, its fully connected nature makes it less suited for tasks with high-dimensional input data like images. Recognizing these limitations in Image processing, researchers introduced CNNs, which excel at capturing spatial hierarchies in image data.

3.2.1 Convolutional Neural Networks (CNNs)

CNNs are a class of deep neural networks, most commonly applied to analyzing visual imagery. They are known for their remarkable ability to automatically and adaptively learn spatial hierarchies of features from input images. CNNs are distinct from other neural networks due to their deep architecture and their method of learning, which involves understanding both the low-level and high-level features of images.

3.2.1.1 Convolution Operation

The core building block of a CNN is the convolutional layer, which applies a series of learnable filters to the input. Each filter in a convolution layer is small spatially (along width and height), but extends through the full depth of the input volume. For a given input image, the convolution operation involves sliding these filters

across the image spatially, computing dot products between the entries of the filter and the input at any position. This process can be mathematically represented as:

$$(F * I)(i, j) = \sum_m \sum_n F(m, n) \cdot I(i + m, j + n) \quad (3.22)$$

,where F is the filter (kernel), I is the input image, and $*$ denotes the convolution operation. This operation results in a feature map that represents the presence of specific features or patterns in the input image.

3.2.1.2 Non-Linearity

After each convolution operation, a non-linear layer (e.g., ReLU or any other activation functions we saw in 3.1.2) is applied. This is an element-wise operation and its purpose is to introduce non-linear properties to the system, enabling the network to learn more complex representations.

3.2.1.3 Pooling/Subsampling

Pooling layers follow the convolutional layers. The primary function of a pooling layer is to reduce the spatial dimensions of the input volume for the next convolutional layer. It reduces the number of parameters and computation in the network, and hence, controls overfitting [16]. The most common form of pooling is max pooling, which extracts sub-regions of the feature map (e.g., 2x2-pixel windows), keeping only the maximum value in each sub-region.

$$P_{\max}(S)(i, j) = \max_{m, n \in S} I(i + m, j + n) \quad (3.23)$$

,where P_{\max} represents the max pooling operation over a sub-region S of the input I .

3.2.1.4 Fully Connected Layers

Towards the end of the network, fully connected layers are used, where neurons have full connections to all activations in the previous layer. This part of the network is responsible for integrating the learned features into the final classification or regression output.

3.2.1.5 Stride and Padding

Stride and padding are two important concepts in CNNs that control how a filter convolves around the input volume.

Stride refers to the number of pixels by which we move the filter across the input matrix. A stride of 1 moves the filter one pixel at a time, while a larger stride results in fewer overlaps and thus a smaller output dimension. The stride is mathematically represented as:

$$O = \frac{W - F + 2P}{S} + 1 \quad (3.24)$$

where O is the output size, W is the input size, F is the filter size, P is the amount of padding, and S is the stride.

Padding involves adding an appropriate number of rows and columns (usually filled with zeros) to the input image. This allows the filter to cover the border areas and adjust the spatial size of the output volume. Zero padding ensures that the spatial dimensions are either preserved or controlled after applying the filter.

3.2.1.6 Architecture Overview

In a typical CNN architecture, several convolutional and pooling layers are stacked, followed by fully connected layers. The initial layers capture basic features like edges, while deeper layers in the network identify more complex features. By stacking these layers, CNNs can effectively learn a hierarchy of increasingly complex visual features.

The above-mentioned layers and operations constitute the basic components of a CNN. Their orchestration and precise configuration depend on the specific task at hand and the nature of the input data. CNNs have been found to be exceptionally effective in tasks involving image and video recognition, image classification, medical image analysis, and many more areas where harnessing spatial hierarchies is crucial.

Chapter 4

Classification & Regression

4.1 Overview of Single Task Learning

Single Task Learning (STL) in machine learning focuses on optimizing and deploying models dedicated to a specific task. This method contrast to Multi-Task Learning (MTL), where a single model is trained to handle multiple tasks simultaneously. STL has several advantages, including simplicity, interpretability, and better generalization performance. There is only one task to optimize, which makes the learning process more straightforward. This also makes it easier to interpret the model's weights and biases, which are directly related to the task at hand and can be useful for understanding the underlying patterns in the data.

4.2 Model Selection

This thesis is focused on utilizing CNNs for this purpose due to their proven effectiveness in spatial feature extraction in image-based tasks.

4.2.1 Convolutional Neural Networks Architecture

The architecture of CNNs plays a pivotal role in their ability to effectively model and predict laser parameters from surface images. Our CNN models consist of three convolutional layers, each followed by batch normalization, max pooling, and dropout operations to counter overfitting. Batch normalization and dropout are crucial in stabilizing and regularizing the network, preventing it from learning noisy patterns and overfitting to the training data.

The convolutional layers in our neural network architecture employ the ReLU activation function. ReLU is particularly effective for non-linear transformations and is known for mitigating the vanishing gradient problem, which is crucial for maintaining the flow of gradients during backpropagation in deep networks. This choice ensures that our model captures complex patterns efficiently while addressing common issues associated with deep learning architectures.

After the convolutional layers, the network architecture transitions to two fully connected (dense) layers. In these layers, the SELU activation function is employed. The decision to use SELU for the fully connected layers is grounded in its self-normalizing property. SELU activation function introduces a scale parameter and an exponential component, which helps in maintaining a mean of zero and a variance of one for the inputs. This normalization effect is particularly beneficial in deep neural networks, as it contributes to a more stable and faster learning process. It also reduces the need for additional batch normalization layers, making the network architecture more efficient.

The final layer of the architecture is tailored to the specific task at hand: for regression tasks, a one-dimensional output is utilized, while classification tasks are addressed with a three-dimensional softmax output.

Figure 4.1 presents a graphical representation of the single task models employed.

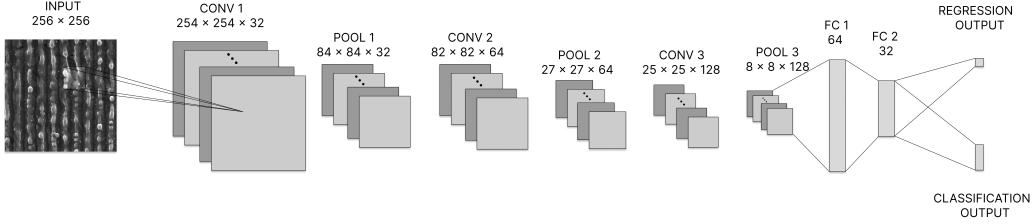


Figure 4.1: A graphical representation of the neural network architecture which consists of three convolutional layers followed by two fully-connected (FC) layers. The output layer is adjusted to the task at hand.

4.2.2 Training Details and Loss Functions

For training the models, different approaches were used for regression and classification tasks. Regression models were trained using the Mean Squared Error (MSE) loss function. This choice is fitting for regression tasks as MSE effectively captures the average squared difference between the estimated values and the actual value, providing a clear measure of model performance. The formula for MSE is given as:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (\theta_i^* - \hat{\theta}_i)^2 \quad (4.1)$$

where θ_i^* is the actual value of the laser parameter, $\hat{\theta}_i$ is the predicted value from the model, and n is the number of observations.

Additionally, the Relative Mean Squared Error (RMSE) was used as a performance metric for regression tasks. The RMSE is computed as a normalized error metric, providing a direct understanding of the relative magnitude of the error,

which is particularly useful when the range of true values is large. The formula for RMSE, in the context of predicting a laser parameter θ , is given as:

$$\text{RMSE}(\theta) = \sqrt{\frac{1}{n} \sum_{i=1}^n \left(\frac{\theta_i^* - \hat{\theta}_i}{\theta_i^*} \right)^2} \quad (4.2)$$

where θ_i^* is the true value of the laser parameter and $\hat{\theta}_i$ is the predicted value for the i -th test sample.

In contrast, classification models were trained using the standard categorical cross-entropy loss function, which is ideal for models that classify input data into various categories. The categorical cross-entropy loss is given by:

$$\text{Cross-Entropy} = - \sum_{c=1}^M y_{o,c} \log(p_{o,c}) \quad (4.3)$$

where M is the number of classes, y is a binary indicator of whether class label c is the correct classification for observation o , and p is the predicted probability observation o is of class c .

The training utilized the Adam optimizer, renowned for its efficiency and excellent performance in various deep learning tasks. This optimizer is particularly adept at handling sparse gradients and adapting the learning rate during training, which is beneficial for our complex dataset.

4.2.3 Rationale Behind Model Selection: Regression vs. Classification

The decision to utilize regression models over classification stems from a strategic consideration of the predictive tasks at hand - estimating laser parameters such as angle, number of pulses, and fluence. While these parameters exhibit a non-sparse distribution within our dataset, suggesting a potential for classification, we anticipate the emergence of new data points outside the current range. Utilizing regression models allows for a flexible and continuous prediction space, accommodating future values without necessitating model restructuring. This approach ensures our models remain adaptable and capable of integrating new data, reflecting a proactive strategy in handling the dynamic nature of laser processing applications.

4.2.4 Hyperparameter Optimization and Validation Strategy

Hyperparameter optimization was carried out through a systematic grid search, enabling the identification of the most effective regularization strategy for each model. Key parameters optimized included the learning rate, epochs, dropout rate, and batch size. An 70% – 30% split for training and testing was adopted, allowing for robust model evaluation and the opportunity to fine-tune the model based on

Table 4.1: Training Parameters for CNN Models on Material Surfaces Dataset

Parameter	Classification	Regression		
		Angle	Fluence	logNP
Learning Rate	0.00005	0.001	0.0005	0.001
Epochs	400	200	400	200
Dropout Rate	0.5	0.4	0.2	0.2
Batch Size	32	32	32	32
Optimizer	Adam	Adam	Adam	Adam
Activation Function	SELU	SELU	SELU	SELU
Total Parameters	619.7K	619.6K	619.6K	619.6K

its performance on the testing set. This approach ensured that the models were not only trained effectively but also tested thoroughly to assess their generalizability and performance on unseen data.

In the evaluation of our Single Task Learning models, both for classification and regression tasks, we adopted two distinct approaches to cross-validation: standard 3-fold cross-validation and stratified 3-fold cross-validation. This dual approach was driven by the nature of our dataset, particularly its imbalance, which necessitated a more nuanced method of validation to ensure robust and reliable model performance assessment.

4.2.5 Training Parameters for CNN Models

To provide a comprehensive understanding of the model training process, this subsection presents detailed training parameters for the CNN utilized in this study. Distinct tables 4.1, 4.2 are dedicated to outlining the parameters employed for the single-task learning approach across real and synthetic datasets. These parameters are pivotal in comprehending the models' behavior and the outcomes of the classification and regression tasks undertaken.

4.2.6 Final Model Selection

Considering these insights, the stratified 3-fold cross-validation was chosen as the final approach for our model evaluation. This method proved to be more effective in handling the imbalance in our dataset, providing a more accurate and consistent assessment of the model's performance. The choice of stratified cross-validation aligns with our commitment to rigor in model validation, ensuring that the models we develop are not only high-performing but also robust and reliable across various data scenarios. For a comprehensive understanding of our validation strategy and the rationale behind selecting the stratified 3-fold cross-validation as our final

Table 4.2: Training Parameters for CNN Models on Energy Profiles Dataset

Parameter	Classification	Regression	
		Angle	Fluence
Learning Rate	0.0005	0.0005	0.001
Epochs	200	200	200
Dropout Rate	0.2	0.2	0.2
Batch Size	32	32	32
Optimizer	Adam	Adam	Adam
Activation Function	SELU	SELU	SELU
Total Parameters	619.7K	619.6K	619.6K

approach, please refer to Section 6.1 in the Results chapter, where we delve deeper into these methodological choices and their implications.

Chapter 5

Multi Task Learning

5.1 Introduction to Multi Task Learning

Multi Task Learning (MTL) represents a paradigm shift in the world of machine learning, challenging the traditional approach of training separate models for each task. This chapter introduces the concept of MTL, its inherent advantages, and its relevance in the context of laser processing and material science.

5.1.1 Defining Multi Task Learning

MTL is a learning strategy where a single model is trained simultaneously on multiple related tasks. The core philosophy behind MTL is that tasks, especially those that are related or share common features, can be learned more effectively together than separately. This approach leverages the potential synergies between tasks, allowing the model to generalize better by learning shared representations.

In traditional machine learning, models are trained in isolation for each task. While this approach is straightforward, it often overlooks the interdependencies and shared structures that might exist between different tasks. MTL, on the other hand, capitalizes on these commonalities, leading to a more holistic learning process.

The theoretical foundation of MTL is deeply entrenched in the principle that concurrent learning of related tasks leads to a more efficient and representative model of the data. This concept is particularly salient in the realm of image processing, as exemplified in this thesis.

In laser processing, image data often encapsulate rich, multifaceted information where multiple attributes or phenomena might interplay. For instance, characteristics such as surface patterns, intensity variations, and material responses are intricately captured in the images. Traditional single-task models might treat each of these characteristics independently, potentially missing out on the nuanced correlations that exist among them. MTL, by learning these tasks simultaneously, seeks to build a more holistic model of the image data.

A significant advantage of MTL in the context of image data for laser processing is its ability to address situations where certain tasks, such as predicting the number

of pulses or fluence, have limited data. In these instances, MTL is particularly beneficial as it allows the model to leverage shared information from closely related tasks.

In essence, MTL harnesses the interconnected nature of these tasks, enhancing the model’s capability to effectively analyze and predict critical parameters from complex image data, thus offering a more comprehensive solution in laser material processing.

In summary, the application of MTL to image data in laser processing is not just a theoretical exercise but a practical approach that capitalizes on the inherent complexity and richness of such data. This thesis demonstrates how MTL can be effectively employed to develop models that are not only efficient but also robust and capable of capturing the intricate relationships present in complex image datasets.

5.1.2 Architecture for Multi Task Learning

The architecture of the MTL model, shown in figure 5.1, in this thesis is an extension of the structure discussed in the previous chapter, tailored to accommodate multiple tasks simultaneously. The key modification in this architecture lies in the integration of four distinct ‘heads,’ each corresponding to a different task, one for classification and three for regression.

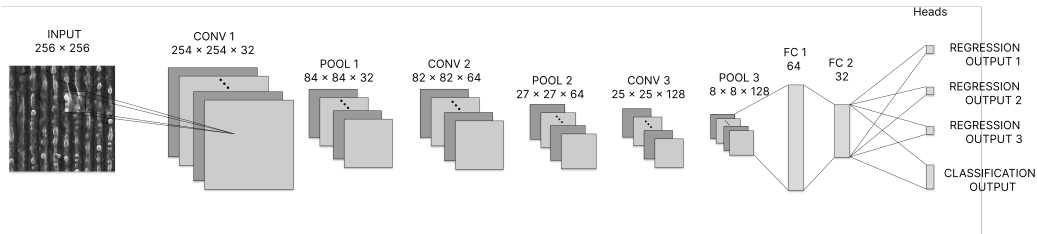


Figure 5.1: A graphical representation of the multi-task neural network architecture, consisting of three convolutional layers followed by two fully connected (FC) layers. The output layer is divided into four distinct heads, with one dedicated to a classification task and the remaining three designed for regression tasks.

5.1.2.1 Shared Base Architecture

The foundation of the MTL model remains similar to the single-task model, comprising convolutional layers that serve as shared feature extractors. This shared base is crucial for learning general representations that are applicable across all tasks. It consists of a series of convolutional layers, each followed by batch normalization [18], max pooling, and dropout operations. These layers capture and process the complex patterns present in the image data, forming a robust base for the subsequent task-specific layers.

5.1.2.2 Task-Specific Heads

The MTL model diverges into four distinct heads from the shared base, each meticulously crafted to address a specific task:

1. **Classification Head:** The classification head is specifically designed for categorizing input data into one of three predefined classes. This head concludes with a softmax layer that outputs a probability distribution over the classes, ensuring a clear and definitive classification.
2. **Regression Heads:** The three regression heads each target a continuous output variable: the number of pulses, fluence, and angle. These heads are tailored for real-valued predictions and culminate in a single neuron dedicated to generating direct regression outputs.

5.1.2.3 Integration and Learning

The integration of these heads with the shared base architecture is a crucial aspect of MTL. During training, the model learns to optimize the shared base to extract features that are beneficial for all tasks while simultaneously fine-tuning the task-specific heads. This integrated learning approach enables the model to balance the learning requirements of each task, leading to a comprehensive understanding of the image data.

5.1.2.4 Benefits of the Multi-Head Architecture

The multi-head architecture offers several benefits:

- **Efficient Learning:** By sharing the lower layers, the model efficiently learns representations that serve multiple tasks, reducing the redundancy of learning separate features for each task.
- **Specialized Task Handling:** Each head can specialize in its respective task, allowing for more focused and accurate predictions in both classification and regression tasks.
- **Flexibility:** This architecture provides the flexibility to add or modify heads as needed, making it adaptable to a range of tasks and datasets.

In conclusion, the architecture of the MTL model in this thesis is a sophisticated blend of shared and task-specific components, each playing a vital role in the model's ability to learn and predict a variety of tasks efficiently and accurately.

5.2 Multi-Task Learning as Multi-Objective Optimization

As we discussed above, MTL is an approach where multiple tasks are solved simultaneously, sharing common inductive biases. However, MTL inherently involves

multi-objective optimization since different tasks may conflict, requiring a trade-off. The common practice of minimizing a weighted linear combination of per-task losses is limited and often inadequate when tasks compete.

In their work Ozan and Vladlen [31], MTL is reformulated as a multi-objective optimization problem with the goal of finding Pareto optimal solutions. The key is to employ gradient-based optimization algorithms from multi-objective optimization literature. However, these algorithms face scalability issues with the dimensionality of gradients and the number of tasks in large-scale learning problems.

The core idea is to define a parametric hypothesis class for each task $f_t(x; \theta_{sh}, \theta_t) : X \rightarrow Y_t$, where θ_{sh} are parameters shared across tasks, and θ_t are task-specific. The task-specific loss functions $L_t(\cdot, \cdot) : Y_t \times Y_t \rightarrow \mathbb{R}_+$ lead to the following empirical risk minimization formulation:

$$\min_{\theta_{sh}, \theta_1, \dots, \theta_T} \sum_{t=1}^T c_t \hat{L}_t(\theta_{sh}, \theta_t) \quad (5.1)$$

where c_t are weights for each task and $\hat{L}_t(\theta_{sh}, \theta_t)$ is the empirical loss for task t .

Although this weighted summation formulation is intuitive, it often requires an extensive grid search over different scalings or the use of heuristic methods, which can be inefficient or suboptimal. Recognizing these limitations, an alternative formulation for MTL is proposed as a multi-objective optimization problem, employing a vector-valued loss L . This approach shifts from optimizing a single aggregate objective to optimizing a vector of losses, each representing a different task:

$$\min_{\theta_{sh}, \theta_1, \dots, \theta_T} L(\theta_{sh}, \theta_1, \dots, \theta_T) = \min_{\theta_{sh}, \theta_1, \dots, \theta_T} \left(\hat{L}_1(\theta_{sh}, \theta_1), \dots, \hat{L}_T(\theta_{sh}, \theta_T) \right) \quad (5.2)$$

This vector-valued loss formulation aligns with the multi-objective nature of MTL, addressing the complexity of simultaneously optimizing multiple, potentially conflicting, task-specific objectives.

In this thesis, we adopt and apply the principles outlined in Ozan and Vladlen's work to our MTL model. Specifically, we incorporate their method of setting an upper bound for the multi-objective loss, which can be optimized efficiently. Adhering to this strategy ensures that the optimization process yields Pareto optimal solutions under practical conditions. By applying these gradient-based multi-objective optimization techniques to deep networks in MTL, our model aims to surpass the performance of recent MTL frameworks and individual task-specific training approaches.

5.2.1 Definition of Pareto Optimality

In the context of multi-objective optimization, Pareto Optimality is a fundamental concept. A solution is considered Pareto optimal if there is no other solution that

improves some objectives without worsening at least one other objective. Formally, a solution \mathbf{x}^* is Pareto optimal if there does not exist another solution \mathbf{x} such that $f_i(\mathbf{x}) \leq f_i(\mathbf{x}^*)$ for all objectives i and $f_j(\mathbf{x}) < f_j(\mathbf{x}^*)$ for at least one objective j .

Mathematically, this can be expressed as:

$$\nexists \mathbf{x} : (\forall i, f_i(\mathbf{x}) \leq f_i(\mathbf{x}^*)) \wedge (\exists j, f_j(\mathbf{x}) < f_j(\mathbf{x}^*)) \quad (5.3)$$

The set of all Pareto optimal solutions forms what is known as the Pareto front. This front represents the trade-offs between different objectives, providing a spectrum of optimal solutions under different prioritization of objectives.

5.2.2 Training Parameters in MTL

The selection of training parameters plays a pivotal role in the optimization and effectiveness of MTL models. For our experiments, we maintained consistency in certain parameters across all tasks to ensure a stable foundation for comparison and analysis. Specifically, the batch size was set at 32, the optimizer of choice was Adam, and the activation function employed was SELU across all configurations. This uniformity in the foundational parameters facilitated a focused examination of the effects of variable parameters on model performance.

For the learning rate, a critical parameter influencing the convergence and learning efficiency of the model, we adopted a strategy of selecting the smallest learning rate from those identified as optimal in the STL configurations. For instance, given the learning rates for classification ($lr = 0.00005$), angle prediction ($lr = 0.01$), fluence prediction ($lr = 0.0005$), and logarithm of the number of pulses ($\log(np)$, $lr = 0.001$) in the STL setting, the MTL configurations would adopt the minimum of these values. This conservative approach was designed to mitigate potential overfitting risks and ensure stable convergence across the combined tasks.

Conversely, for the number of epochs, we selected the maximum value observed among the individual tasks in the STL setup. This decision was informed by the intention to allow the MTL model ample opportunity to learn from the diverse and complex data representations inherent in multiple simultaneous tasks.

The dropout rate, a regularization parameter used to prevent overfitting, was determined in a somewhat stochastic manner. For each MTL configuration, the dropout rate was randomly selected within an interval defined by the minimum and maximum dropout rates identified in the STL experiments. This strategy introduced an element of variability, ensuring that the models were robustly tested against overfitting while navigating the nuanced landscapes of multi-task optimization.

Through this meticulous parameter selection process, our MTL models were tailored to leverage the strengths of uniform foundational settings while accommodating the specific learning dynamics necessitated by the combination of tasks. This balanced approach aimed to maximize learning efficiency and model performance across the diverse objectives of our study.

Chapter 6

Results

In this chapter, we are going to provide more details regarding the data preparation steps and discuss the outcomes from applying the Single and Multi Task Learning models.

6.1 Data Preparation

6.1.1 Data Augmentation

Starting with data augmentation, this step played a crucial role in ensuring the quality and reliability of the models' outcomes.

The original real dataset consisted of 208 high-resolution images, each measuring 1280×960 pixels. To manage the dimensionality and expand our dataset for training purposes, we employed a random cropping technique. We segmented each image into 100 smaller segments, each with dimensions of 256×256 pixels. This approach not only helped in reducing the pixel-space dimensionality but also significantly amplified our dataset size, expanding from the original count of 208 images to an augmented collection of 20,800 images. Although these newly created samples are derivatives of the same original images, this augmentation strategy plays a vital role in reinforcing the shift invariance characteristic within the dataset.

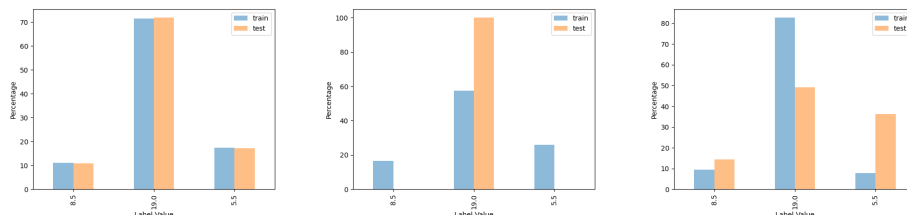
6.1.2 Addressing Data Imbalance with Stratified Cross-Validation

Data imbalance, particularly in classification tasks, poses a significant challenge in machine learning, often leading to skewed model performance. In our study, we initially employed a standard 3-fold cross-validation approach. However, this approach did not sufficiently address the class distribution skewness, especially in our real dataset for classification tasks.

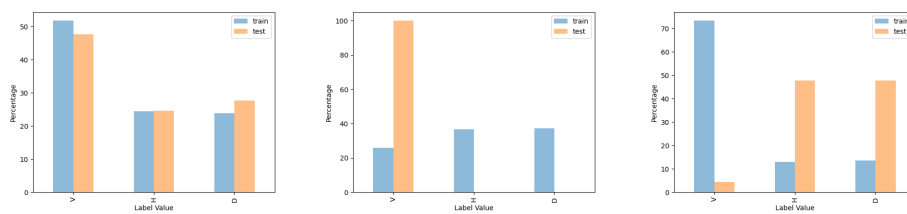
This realization led us to adopt stratified 3-fold cross-validation, an approach that ensures each fold proportionally represents all classes. Stratified cross-validation is particularly effective for datasets with uneven class distribution, as it provides a more accurate and representative performance evaluation [35, ?, 36].

Cross-Validation (CV) Distributions

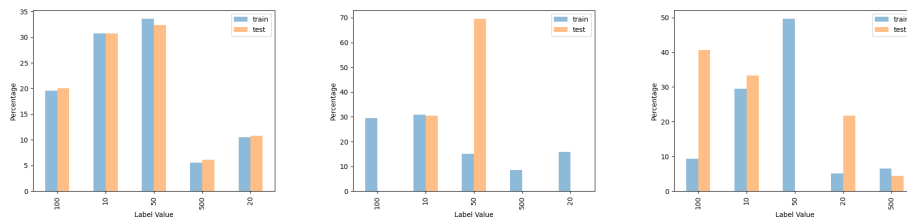
Angle Distribution



Pattern Type Distribution



Number of Pulses Distribution



Fluence Distribution

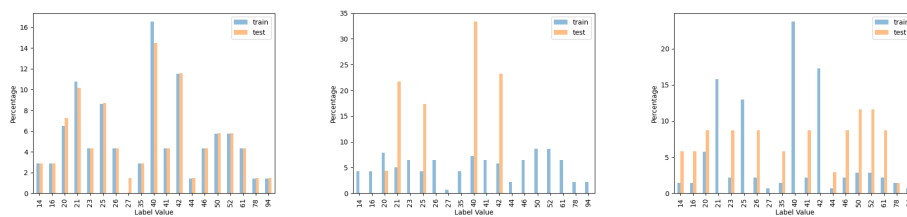
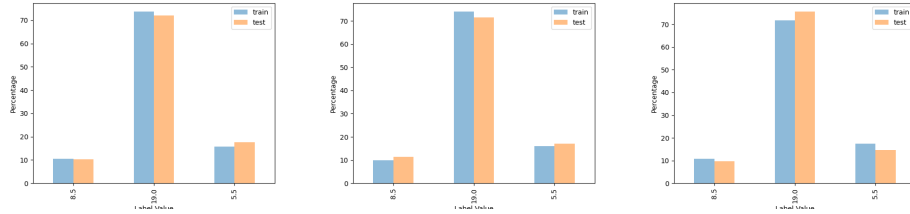


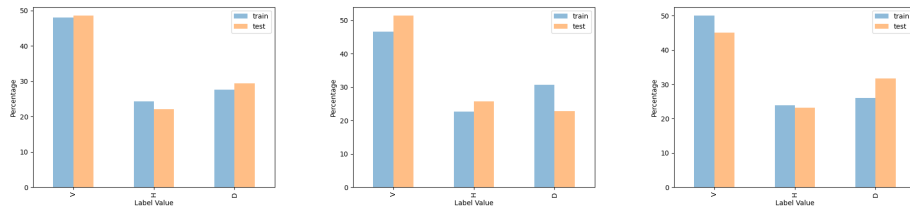
Figure 6.1: Data distribution across various tasks in standard 3-fold Cross-Validation.

Stratified Cross-Validation Distributions

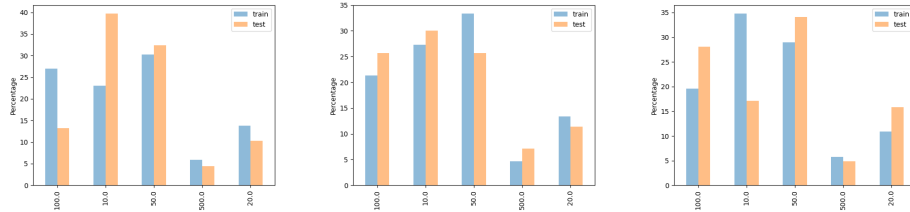
Angle Distribution



Pattern Type Distribution



Number of Pulses Distribution



Fluence Distribution

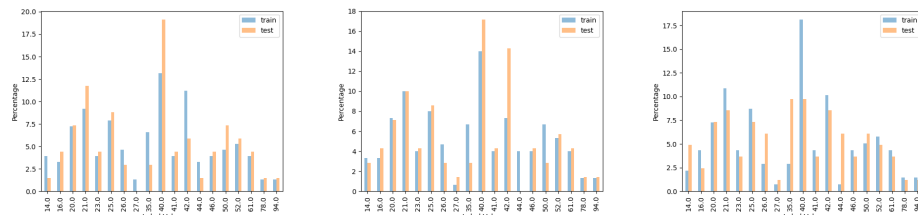


Figure 6.2: Data distribution across various tasks in Stratified 3-fold Cross-Validation.

6.1.3 Comparative Analysis of Cross-Validation Approaches

A comparative analysis of the two cross-validation approaches revealed notable differences in their performance metrics. With the standard 3-fold cross-validation, we observed higher variability in metrics across different folds. In contrast, the stratified 3-fold cross-validation showed a marked improvement in consistency and reliability. This approach, which accounted for the class distribution in the dataset, resulted in less variance in performance metrics across the folds. Notably, the classification accuracy improved significantly, and the MSE for regression tasks showed more stability. For the fluence prediction, for example, the variance of MSE using standard cross-validation was 5991, which reduced to 4472 with stratified cross-validation, marking an approximately 25.35% decrease in variance. This substantial decrease highlights the effectiveness of stratified cross-validation in providing more stable and reliable model evaluations, especially important in the presence of data imbalance. The comparative data distributions across various tasks in both cross-validation approaches can be seen in Figures 6.1 and 6.2, demonstrating the differences in consistency and reliability between the two methods.

6.2 Results of Single Task Learning Models

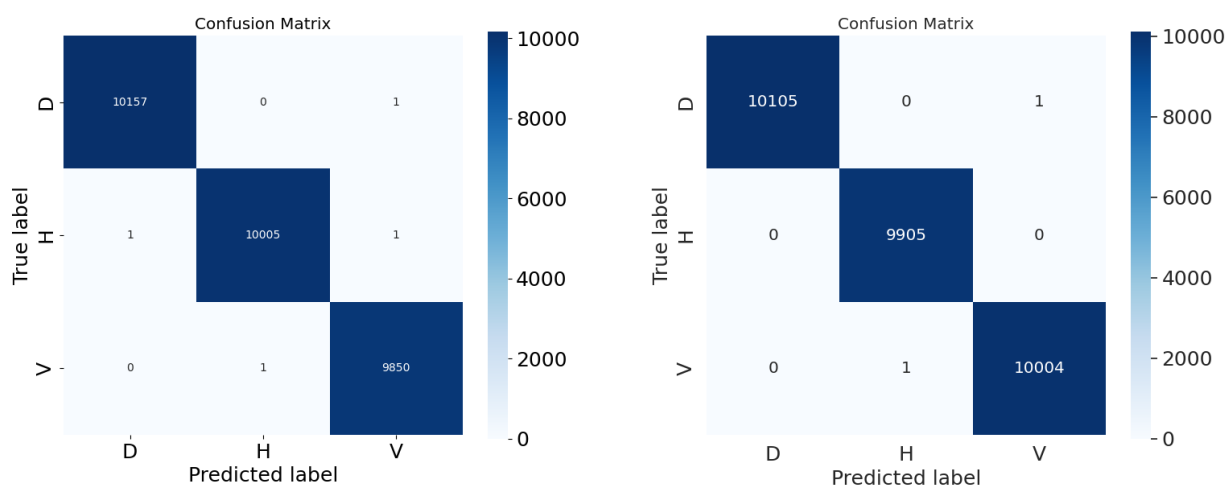
6.2.1 Baseline Model Performance

To establish a performance benchmark for our predictive models, we developed baseline models for both classification and regression tasks. The classification baseline, implemented through stratified cross-validation, yielded a mean accuracy of 33%, serving as an initial metric for comparison. For regression, the baseline approach entailed predicting the mean value of each task (angle, number of pulses and fluence) from the training set and evaluating the MSE against the actual values in the test set, following stratified cross-validation. The results indicated MSE values of 30.53 for angle, 226.21 for fluence, and 1.16 for the number of pulses.

These baseline outcomes provide a quantitative foundation from which the improvements introduced by our CNN models can be assessed, highlighting the advanced models' ability to surpass these initial performance metrics through more sophisticated feature extraction and learning mechanisms.

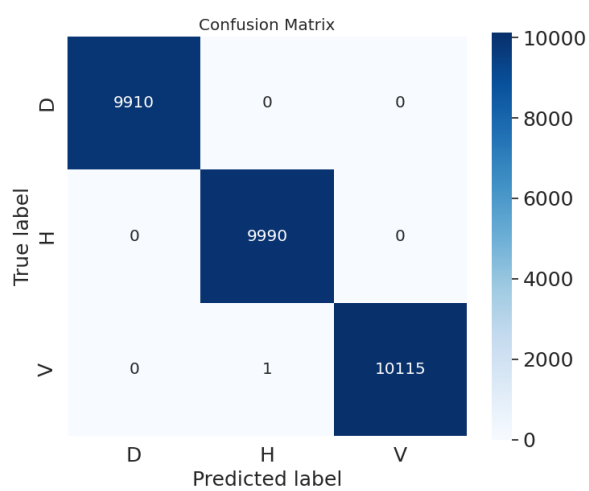
6.2.2 On the Synthetic Dataset

Our models demonstrated exemplary performance on the synthetic dataset, which was crucial in establishing the foundation for our real-world applications. The classification model achieved an accuracy of 99.9%, indicating its robust capability in pattern recognition from the noise-free and uniformly sampled synthetic dataset. The regression models exhibited impressive performance, yielding a MSE value of 0.008 for predicting the angle parameter. For the fluence parameter, the MSE was recorded at 0.89. These low error values indicate the models' proficiency in making precise estimations for these parameters.



(a) Split 1

(b) Split 2



(c) Split 3

Figure 6.3: Confusion matrices for each of the three cross-validation splits on the synthetic dataset.

For a more detailed view, refer to Table 6.1 for numerical data and Figure 6.3 for a visual representation of the classification model’s performance.

Table 6.1: Performance Metrics on Synthetic Dataset - Cross Validation

Task	Subtask	Accuracy (%)	MSE	RMSE
Classification	-	99.9	-	-
Regression	Angle	-	0.008	0.0001
Regression	Fluence	-	0.89	0.0005

6.2.3 On the Real Dataset

6.2.3.1 Classification Task Performance

In our experiments with the real dataset, the results varied significantly between different validation approaches due to the inherent data imbalances. When utilizing standard cross-validation methods, our model achieved an accuracy of 66.1%. This relatively lower performance can be attributed to the imbalanced nature of the dataset, which tends to affect the model’s ability to generalize effectively.

In contrast, the application of stratified cross-validation, which accounts for the distribution of classes in each fold, led to a marked improvement in model performance. In this setting, the model’s accuracy increased substantially to 94.9%. This improvement highlights the importance of considering data distribution in model validation, especially in scenarios with imbalanced datasets.

Additionally, to provide a more comprehensive overview of the model’s performance across different classes, in figure 6.4 we present the Average Confusion Matrix computed over the three stratified folds. The matrix offers insights into the model’s classification capabilities and the nature of errors across different classes. We also present four randomly selected samples of misclassified images in Figure 6.5. Material defects as well as hard to identify surface structures are the main reasons for misclassification errors.

6.2.3.2 Regression Task Performance

In the regression tasks, the results were as follows:

- For the Angle prediction task, the MSE was 16.54 with an RMSE of 0.25, indicating a good level of precision in the model’s predictions.
- The Fluence prediction task presented a higher challenge, reflected in an MSE of 174.98 and an RMSE of 0.013. This suggests a need for further refinement in this specific task.

Table 6.2: Performance Metrics on Real Datasets - Cross Validation and Stratified Cross Validation

Validation Type	Task	Subtask	Accuracy	MSE	RMSE	Std of Splits
Unstratified CV	Classification	-	66.1 (%)	-	-	0.09
	Regression	Angle	-	22.69	0.41	13.09
	Regression	Fluence	-	159.29	0.12	77.4
	Regression	logNP	-	0.30	0.019	0.019
Stratified CV	Classification	-	94.9 (%)	-	-	0.023
	Regression	Angle	-	16.54	0.25	0.82
	Regression	Fluence	-	170.38	0.11	66.87
	Regression	logNP	-	0.29	0.014	0.09

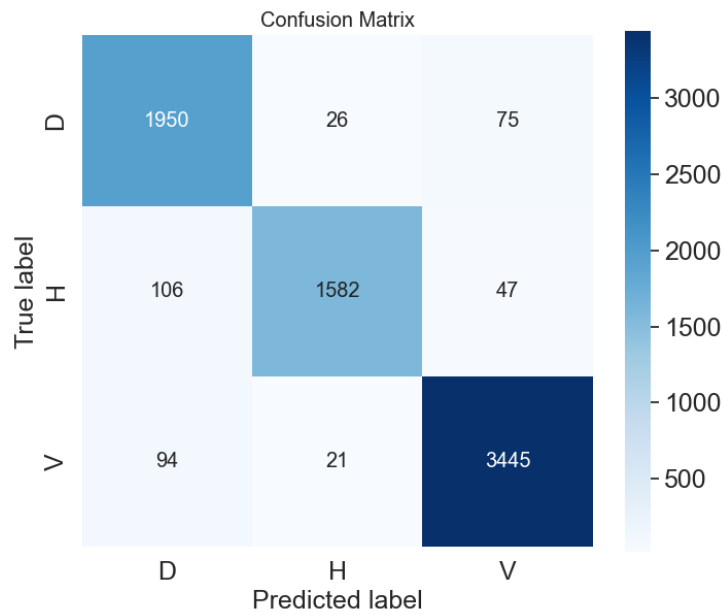


Figure 6.4: Average Confusion Matrix across the three stratified folds on real dataset.

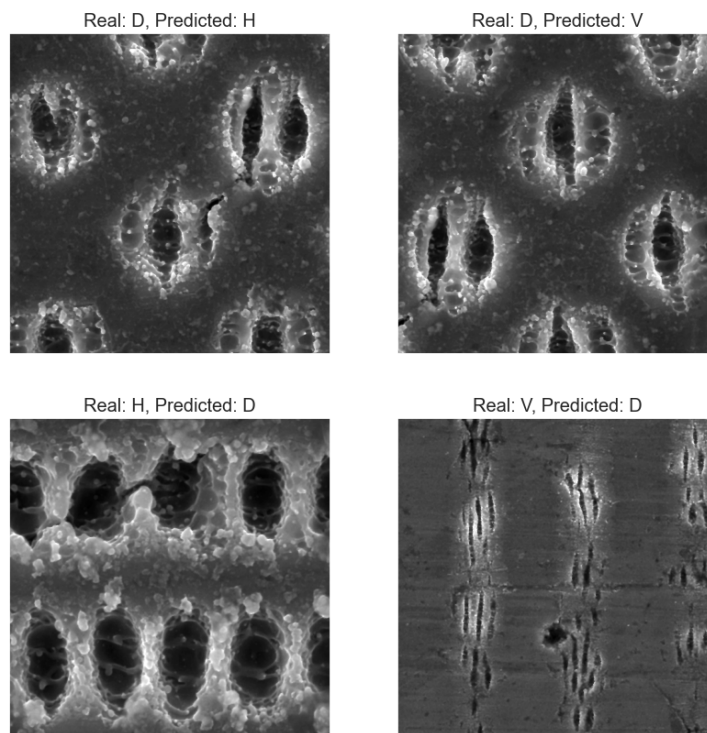


Figure 6.5: A random selection of misclassified samples. The left plots depicts confusing surface structures, while the right plots depicts structures with notable defects.

- The Number of Pulses prediction task showed promising results with an MSE of 0.29 and an RMSE of 0.014, highlighting the model’s efficacy in this aspect.

For a complete picture of the performance metrics, including MSE, RMSE, and accuracy for both simulated and actual datasets, please consult Tables 6.1, 6.2.

6.2.3.3 Enhanced Performance through Stratified Cross-Validation

Stratified cross-validation was particularly effective in reducing the variance of the model’s performance, especially in regression tasks. For instance, in the Angle prediction task, the stratified cross-validation resulted in an MSE of 16.54, a substantial improvement from the non-stratified cross-validation MSE of 22.69.

The Fluence prediction task showed an MSE of 174.98 in the stratified setting, compared to 159.29 in the standard setting. Though there was an increase in MSE, the reduction in variance was deemed more critical for consistent model performance.

The Number of Pulses prediction task showed an MSE of 0.29 in the stratified cross-validation, slightly better than the non-stratified MSE of 0.30, reinforcing the benefits of this approach in handling data imbalance.

6.3 Results of Multi Task Learning Models

This section delineates the outcomes derived from the MTL framework, as elaborated in Section 5.2. Initially, the analysis underscores the algorithm’s efficacy on synthetic data, followed by insights from real dataset applications. MTL posits an enhanced performance potential by simultaneously optimizing across multiple tasks. Our experimental framework methodically evaluated the MTL paradigm by optimizing across an increasing number of tasks—two and three tasks for the synthetic dataset, reflecting its scope, and extending up to four tasks for the real dataset, where a broader task spectrum is applicable. This incremental strategy aimed at discerning the impact of concurrent task optimization on the overall model performance. The synthetic data, with its controlled environment, offers a benchmark for comparing MTL against Single Task Learning, thereby establishing a baseline for expected performance enhancements when multiple tasks are optimized concurrently.

6.3.1 On the Synthetic Data

6.3.1.1 Two Tasks

As mentioned earlier, several experiments were conducted on the synthetic dataset for MTL. Starting with the optimization of two tasks simultaneously, we explored three different task configurations: pattern type and angle, pattern type and fluence, and fluence and angle. For each configuration, we adopted the cross-validation technique across three folds.

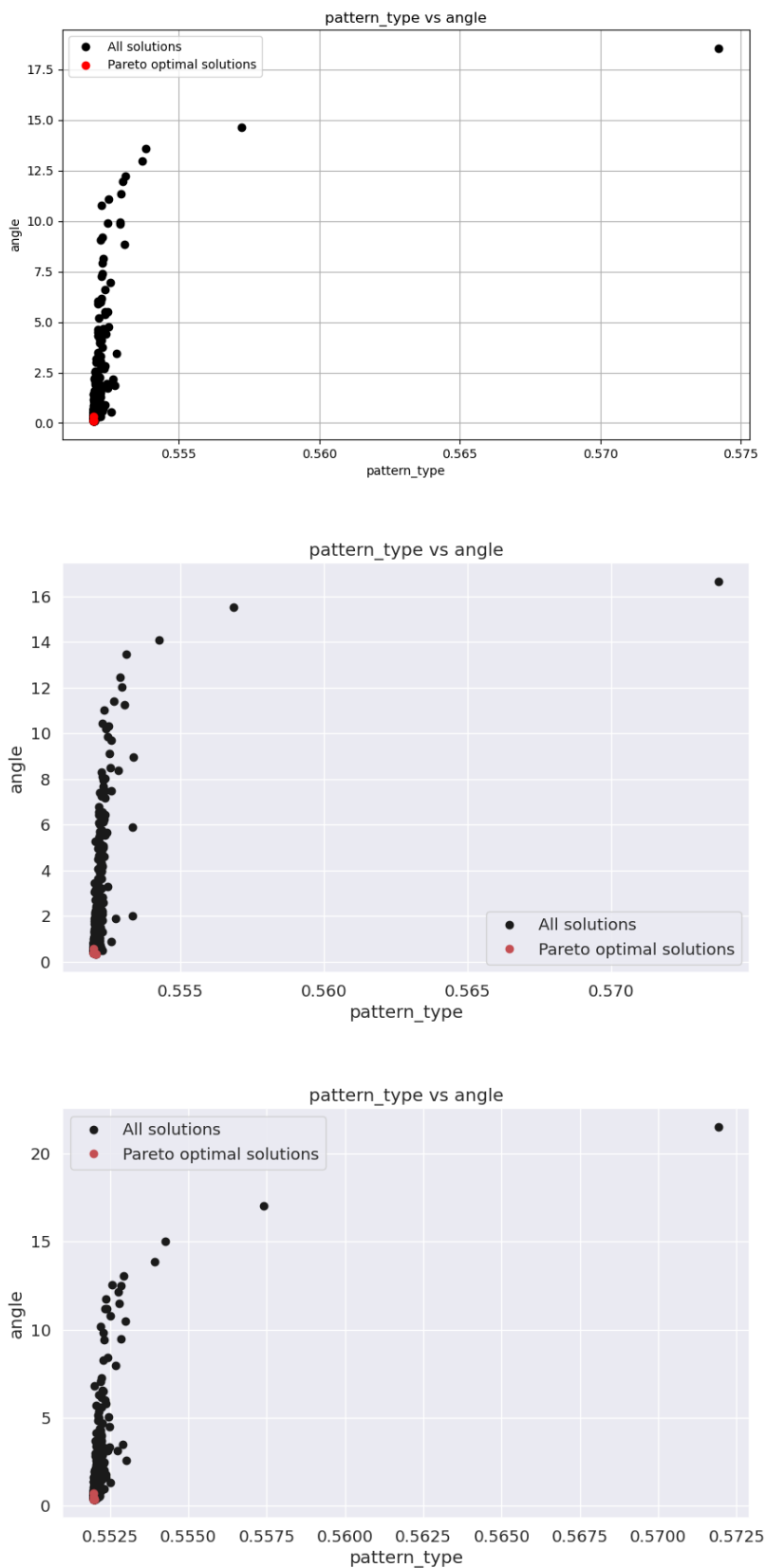


Figure 6.6: Pareto fronts for the Pattern Type and Angle Task Configuration across three cross-validation splits on the synthetic dataset, with the top image representing fold 1, the middle image fold 2, and the bottom image fold 3.

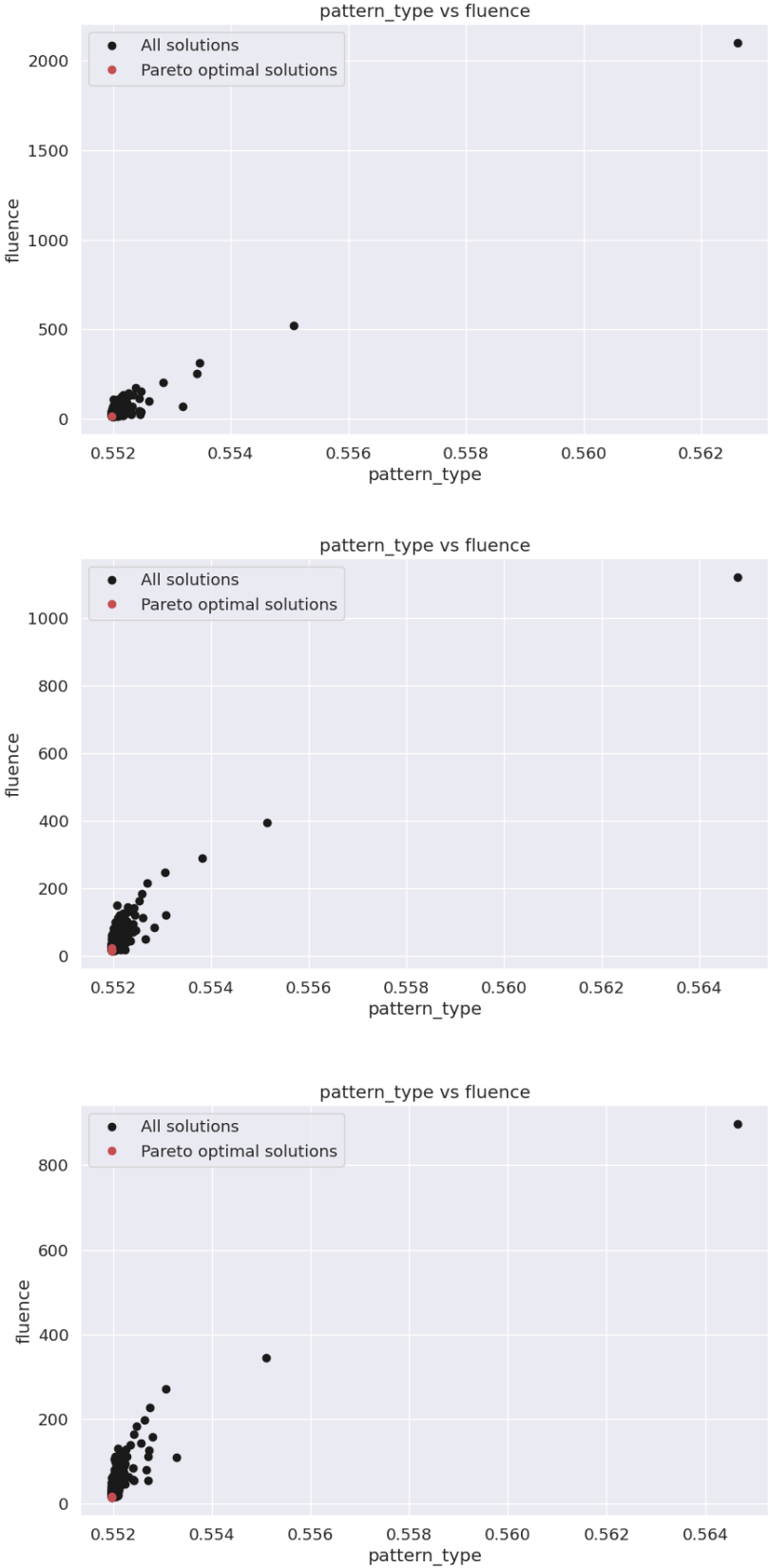


Figure 6.7: Pareto fronts for the Pattern Type and Fluence Task Configuration across three cross-validation splits on the synthetic dataset, with the top image representing fold 1, the middle image fold 2, and the bottom image fold 3.

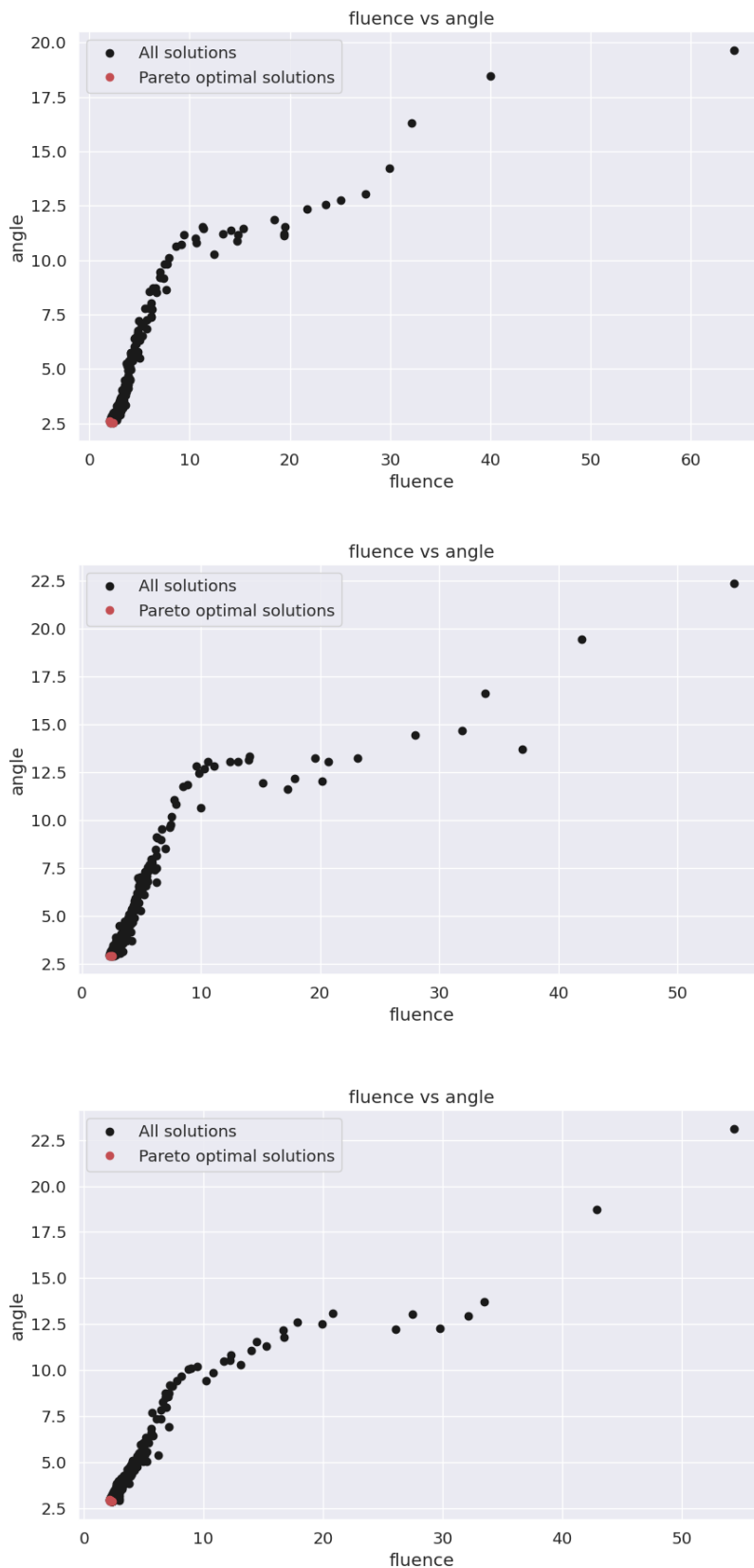


Figure 6.8: Pareto fronts for the Fluence and Angle Task Configuration across three cross-validation splits on the synthetic dataset, with the top image representing fold 1, the middle image fold 2, and the bottom image fold 3.

Let’s start with the first configuration with tasks pattern type and angle. The model achieved a mean accuracy of 99.9% for the classification task and a mean MSE loss of 0.38 (mean RMSE of 0.005). Continuing with the pattern type and fluence task configuration, we have again a 99.9% mean accuracy for the pattern type and mean MSE loss of 16.92 with RMSE value of 0.01. Lastly, for the fluence, angle optimization, we were able to obtain a much smaller MSE loss for the fluence, and that is 2.32 MSE with RMSE of 0.001. Regarding the angle task, the model had a mean MSE 2.8566 and RMSE of 0.033. These results, while not as low as those seen in single-task learning scenarios, demonstrate the model’s adeptness at balancing multiple tasks effectively. Figures 6.6, 6.7, and 6.8 presents the Pareto fronts derived from these experiments, illustrating the models’ convergence towards optimal solutions.

6.3.1.2 Three Tasks

The endeavor to harness the capabilities of MTL extended to the simultaneous optimization of three tasks within the synthetic dataset. This step served as a sanity check to verify the algorithm’s functionality. The configuration comprising the tasks of angle prediction, fluence prediction, and pattern type classification was subject to this examination.

For the angle prediction task, the model demonstrated a mean MSE of 0.008 and a mean RMSE of 0.0001, aligning closely with the performance benchmarks established in the single-task learning scenario. This outcome signifies a strong model proficiency in capturing the geometric intricacies represented in the synthetic images.

Contrastingly, the fluence prediction task manifested a mean MSE of 21.75 and a mean RMSE of 0.019, indicating a moderate learning curve. While this shows a deviation from the desired low-error threshold, it still represents a significant learning from the data.

The pattern type classification task, however, exhibited a mean accuracy of 34.5%, which, for a three-class problem, suggests performance near random chance. This indicates that while the angle task maintained a strong signal in the multi-task setting, the fluence and pattern type tasks might require further fine-tuning or data augmentation to reach a comparable level of accuracy.

Figure 6.9 illustrating the Pareto fronts for these experiments will shed light on the trade-offs and compromises inherent in optimizing multiple objectives. These visualizations serve as a testament to the model’s convergence towards solutions that balance the competing demands of each task.

The confusion matrices in figure 6.9 derived from the three-task optimization on the synthetic dataset reveal distinct labeling biases in each fold. Specifically, fold 1 shows a strong inclination towards assigning the label V, accounting for 67.6% of the predictions. Fold 2 predominantly assigns the label H, making up 43.14% of its classifications, while fold 3 heavily favors the label D, with 81.29% of images assigned this label. The pronounced biases indicate an imbalanced classification

performance across the folds, underscoring the necessity for strategic model refinements or the implementation of data balancing techniques to enhance accuracy and ensure a more uniform class distribution.

6.3.2 On the Real Data

6.3.2.1 Two Tasks

Exploring the MTL framework’s efficacy on the real dataset, we initially focused on combinations of two tasks.

The MTL framework demonstrated differential performance across various task pairings, underscoring the influence of concurrent optimization on model effectiveness:

- The *fluence* and *angle* task pairing exhibited a mean MSE of 187.38 for fluence (RMSE 0.13) and 26.78 for angle (RMSE 0.51), which, when compared to the single-task learning outcomes (Fluence MSE 170.38, RMSE 0.11; Angle MSE 16.54, RMSE 0.25), indicates a decrease in performance for both tasks under the MTL regime.
- Optimizing for *number of pulses* (log scale) and *angle* yielded an MSE of 0.34 for number of pulses (RMSE 0.017) and 27.78 for angle (RMSE 0.51). Here again, MTL did not surpass the single-task learning performance, especially notable for the angle prediction.
- The combination of *fluence* and *number of pulses* resulted in an MSE of 260.87 for fluence (RMSE 0.11) and 0.3 for number of pulses (RMSE 0.014). This outcome reveals a significant performance dip for fluence prediction in MTL compared to its single-task counterpart.
- Regarding the *pattern type* classification in conjunction with a regression task:
 - Pairing with number of pulses resulted in a classification accuracy of 55.59% and an MSE of 0.42 for number of pulses (RMSE 0.012), indicating a stark contrast to the single-task accuracy of 94.9%.
 - When coupled with angle prediction, the model achieved a notable accuracy of 96.01%, with an MSE of 20.5 for angle (RMSE 0.30), closely aligning with single-task results but not exceeding them.
 - The fusion with fluence prediction led to a classification accuracy of 96.3% and an MSE of 233.82 for fluence (RMSE 0.26), slightly improving the classification accuracy but at the cost of increased error in fluence prediction.

These insights suggest that while MTL holds the promise of enhanced learning by leveraging shared information across tasks, the realization of this potential is

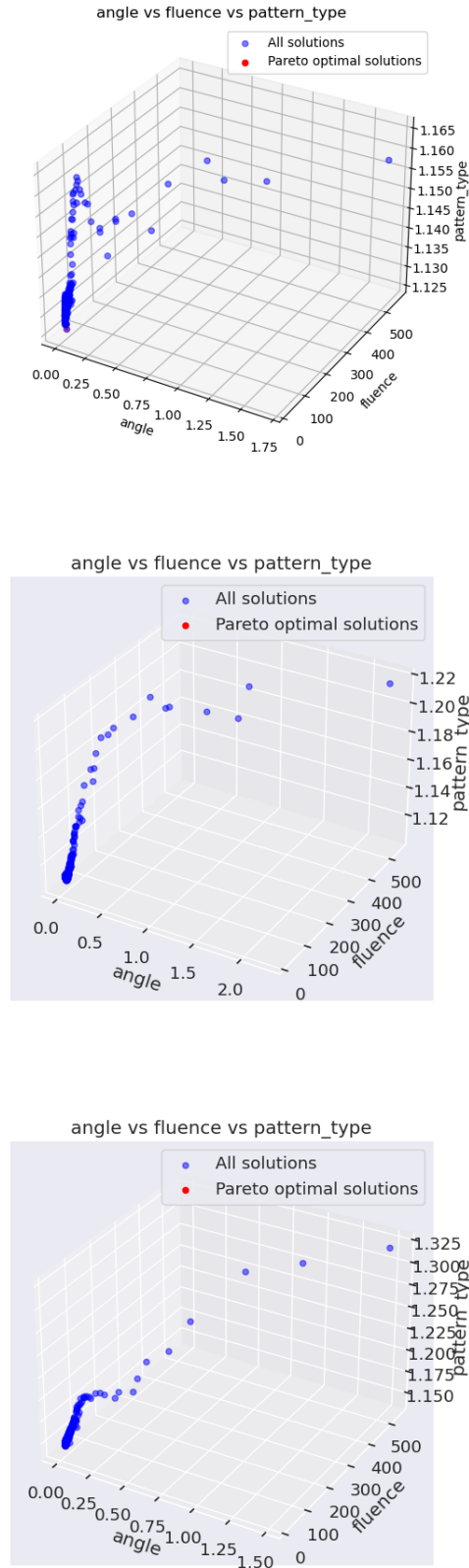
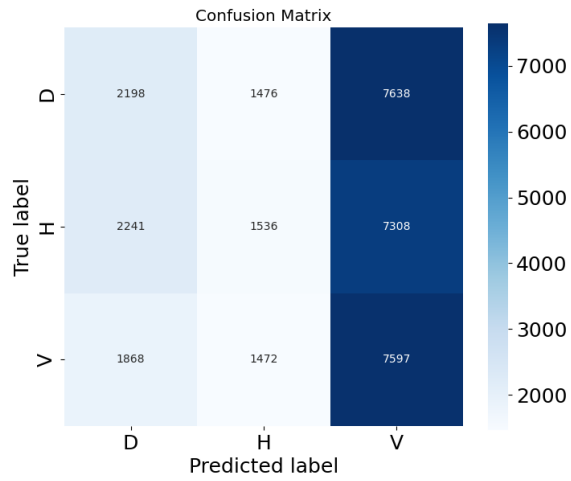
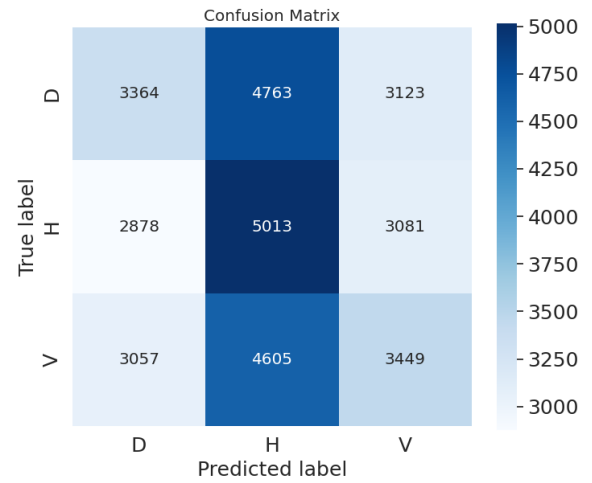


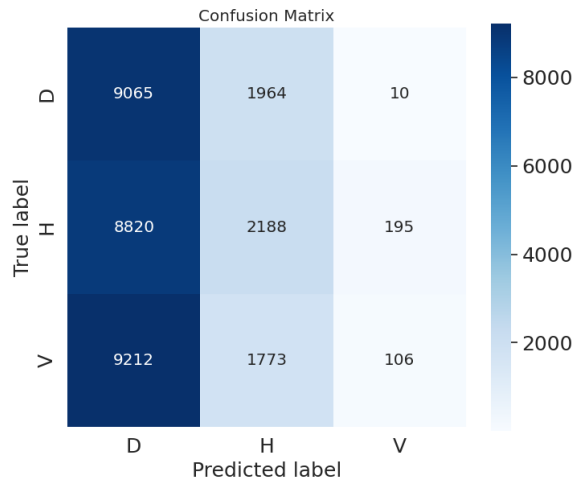
Figure 6.9: Pareto fronts for the simultaneous optimization of three tasks on the synthetic dataset: pattern type classification, fluence prediction, and angle prediction. The plots represent the trade-offs achieved between tasks across three folds, highlighting the convergence towards Pareto optimal solutions. The top image corresponds to fold 1, the middle to fold 2, and the bottom to fold 3.



(a) Split 1



(b) Split 2



(c) Split 3

Figure 6.10: Confusion matrices for the pattern type classification task within the three-task MTL framework on the synthetic dataset. Each matrix corresponds to one of the three folds, illustrating the classification distribution. Fold 1 (a) predominantly assigns the label V, fold 2 (b) leans towards label H, and fold 3 (c) shows a preference for label D.

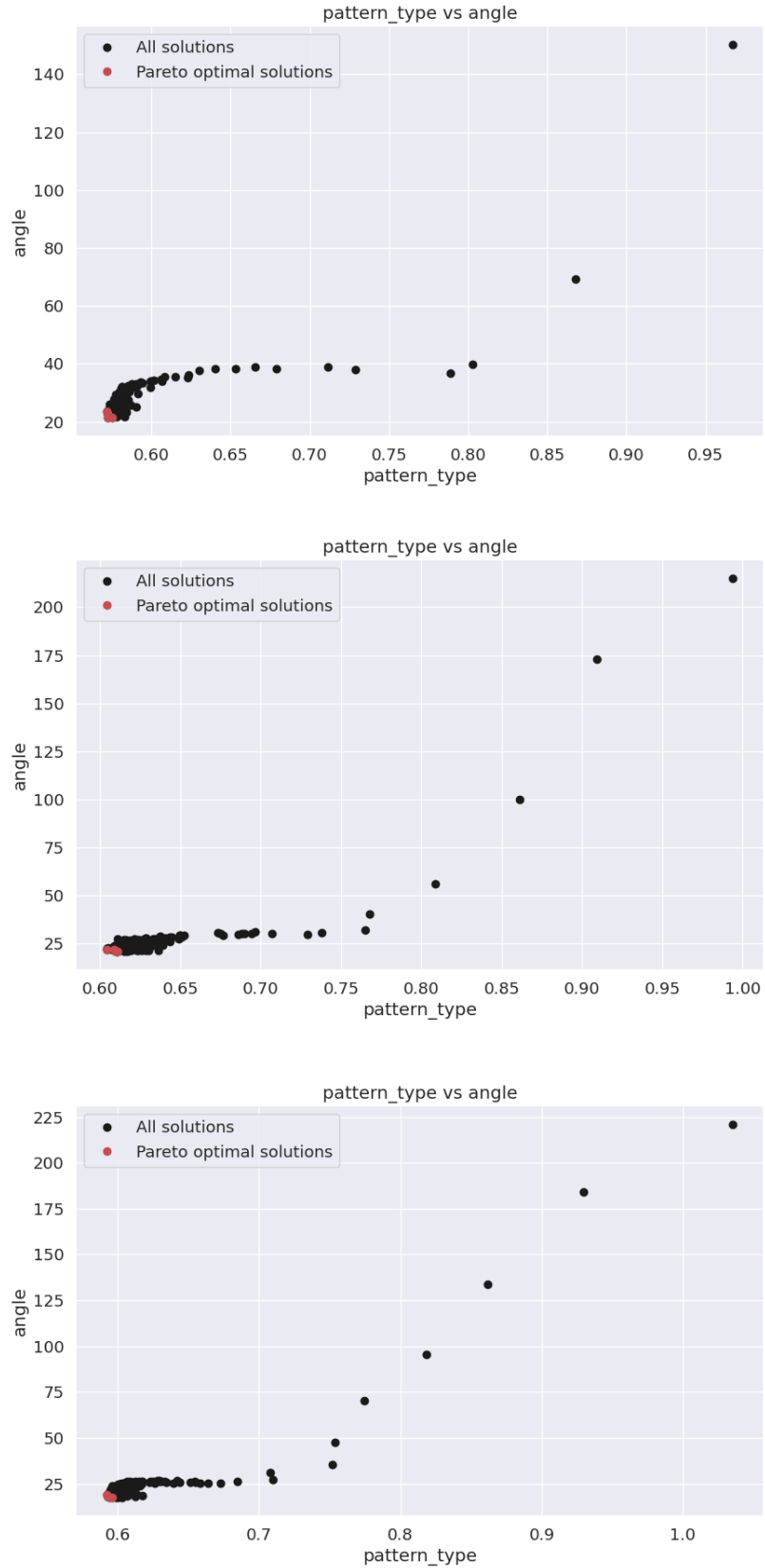


Figure 6.11: Pareto fronts for the Pattern Type and Angle Task Configuration across three cross-validation splits on the real dataset, with the top image representing fold 1, the middle image fold 2, and the bottom image fold 3.

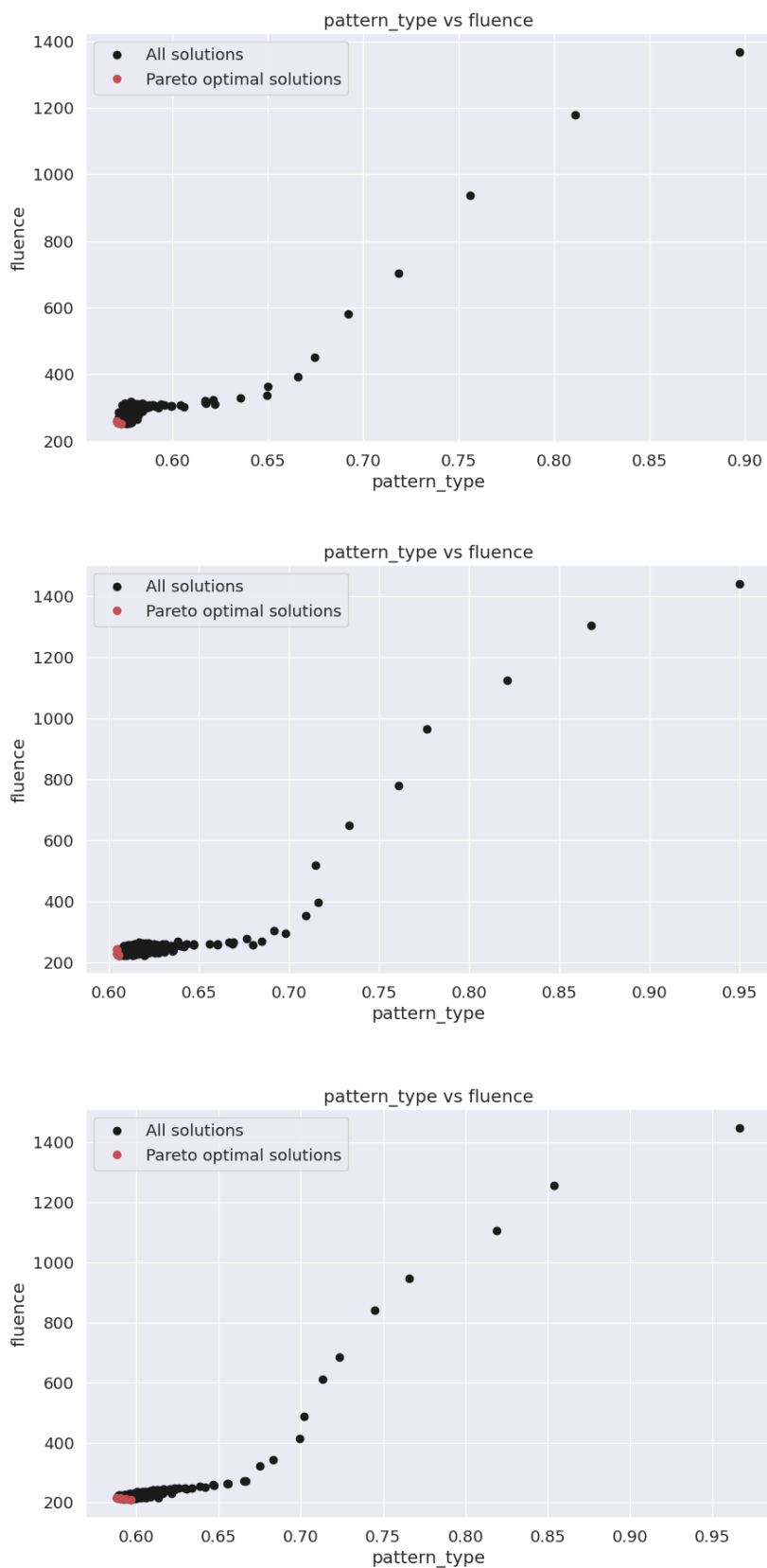


Figure 6.12: Pareto fronts for the Pattern Type and Fluence Task Configuration across three cross-validation splits on the real dataset, with the top image representing fold 1, the middle image fold 2, and the bottom image fold 3.

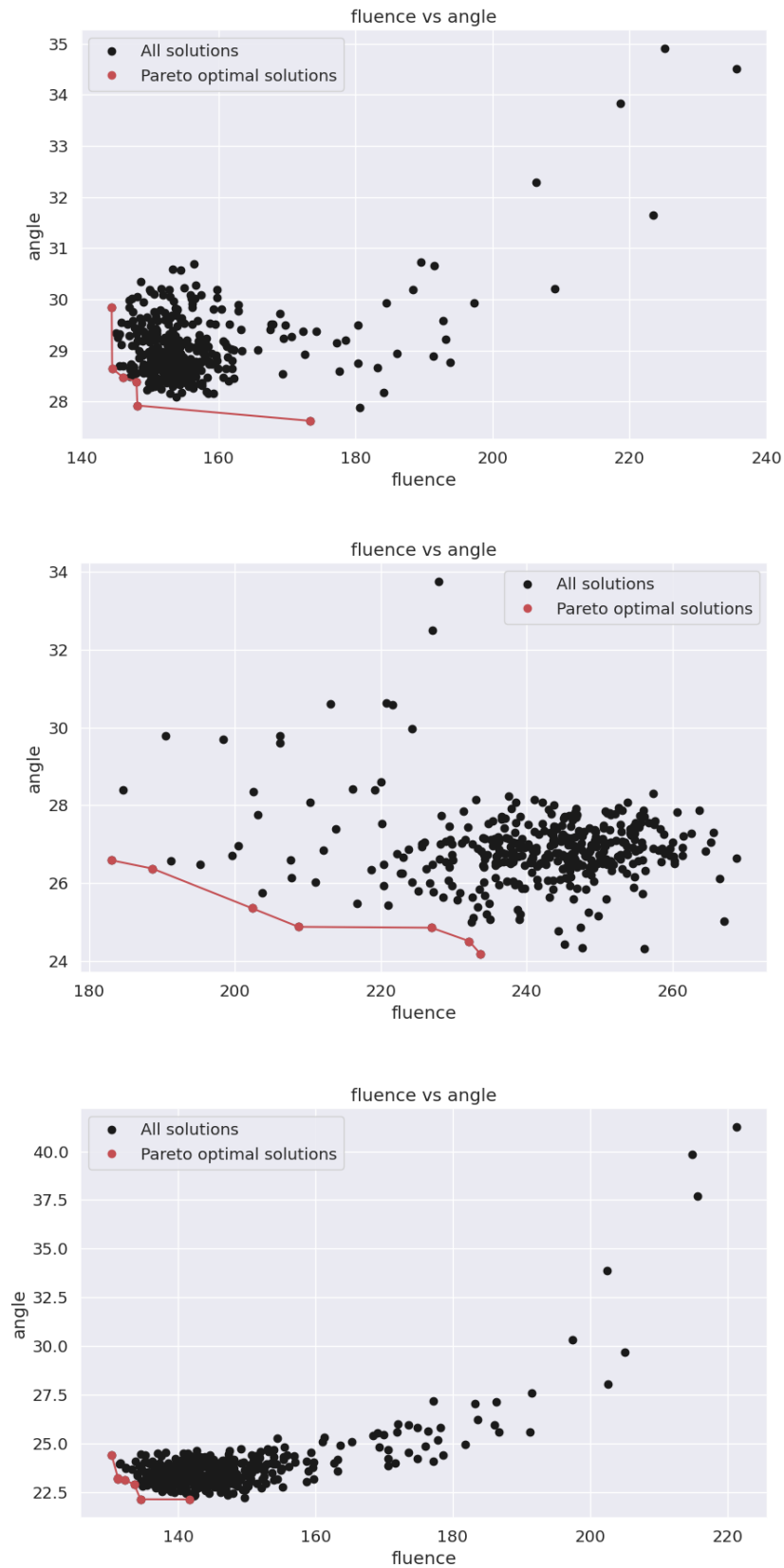


Figure 6.13: Pareto fronts for the Fluence and Angle Task Configuration across three cross-validation splits on the real dataset, with the top image representing fold 1, the middle image fold 2, and the bottom image fold 3.

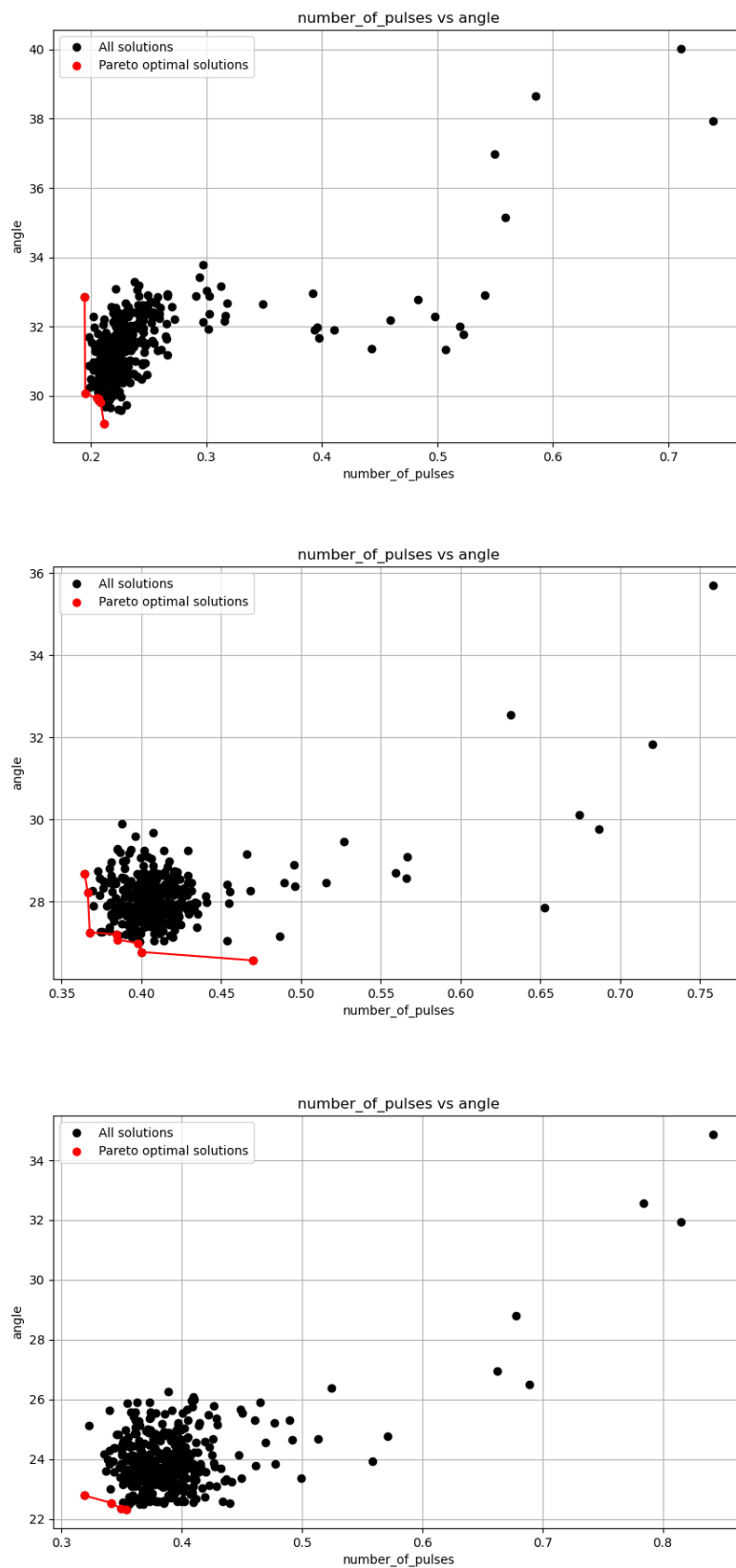


Figure 6.14: Pareto fronts for the Number of Pulses and Angle Task Configuration across three cross-validation splits on the real dataset, with the top image representing fold 1, the middle image fold 2, and the bottom image fold 3.

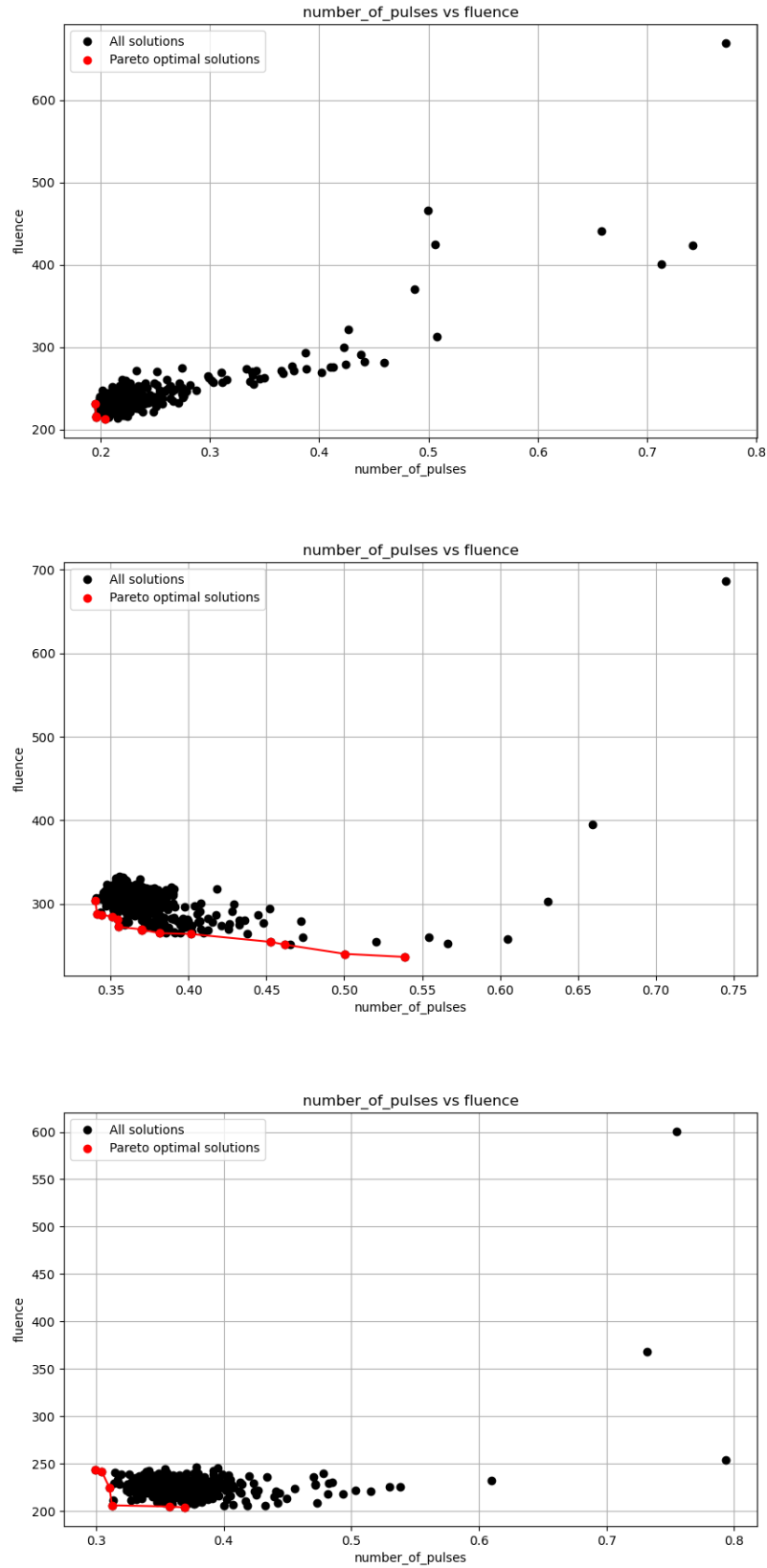


Figure 6.15: Pareto fronts for the Number of Pulses and Fluence Task Configuration across three cross-validation splits on the real dataset, with the top image representing fold 1, the middle image fold 2, and the bottom image fold 3.

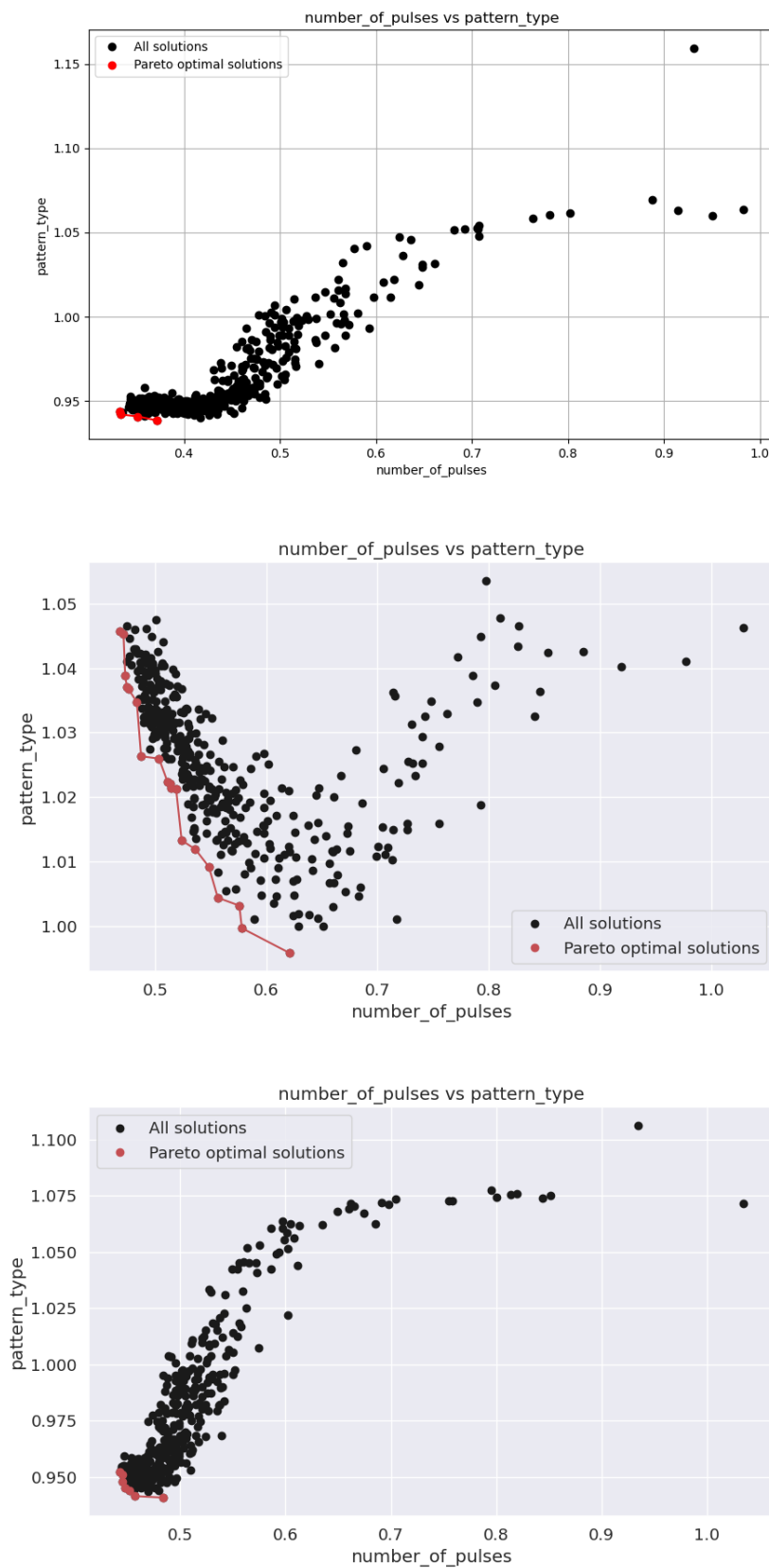


Figure 6.16: Pareto fronts for the Number of Pulses and Pattern Type Task Configuration across three cross-validation splits on the real dataset, with the top image representing fold 1, the middle image fold 2, and the bottom image fold 3.

contingent upon the specific task combinations and the inherent complexities of their interrelations. In our dataset, the MTL configurations did not uniformly surpass the benchmarks set by single-task learning, indicating the need for refined strategies or more nuanced model architectures to fully exploit the advantages of MTL. In figures 6.13, 6.14, 6.15, 6.16, 6.11, and 6.12 we present the pareto fronts of the configurations discussed above.

6.3.2.2 Three Tasks

Here, we continue with the results when we optimize three tasks.

The first configuration assessed the model’s capability in angle prediction, fluence prediction, and pattern type classification, yielding the following results: The mean angle MSE was 14.66 with a mean angle RMSE of 0.216. The mean fluence MSE reached 230.69 with a mean fluence RMSE of 0.293. The mean accuracy for pattern type classification was observed at 28.29%, indicating moderate performance in fluence prediction but suggesting challenges in accurately classifying pattern types. Notably, the model outperformed the single-task model in angle prediction, resulting in an 11% decrease in MSE value.

Results from the second configuration, which focused on the estimation of the number of pulses, angle prediction, and fluence prediction, indicated a relatively low error for the number of pulses estimation with a mean MSE of 0.353 and a mean RMSE of 0.017. The angle prediction task yielded a mean MSE of 25.65 and a mean RMSE of 0.46, while the fluence prediction showed significant challenges with a mean MSE of 1500.41 and a mean RMSE of 0.981.

In the third configuration, the model encountered similar challenges in angle prediction as seen in the second configuration but with slightly increased error metrics: a mean MSE of 27.18 and a mean RMSE of 0.539 for angle prediction. The mean MSE for the number of pulses was 0.49 with a mean RMSE of 0.03, and the mean accuracy for pattern type classification was 28.98%.

The fourth configuration exhibited a pattern type classification accuracy of 31.84%. The mean number of pulses MSE was 0.41 with a mean RMSE of 0.025. The fluence prediction tasks presented significant prediction errors with a mean MSE of 235.54 and a mean RMSE of 0.272.

These results collectively underscore the intricacies of multi-task learning, especially when optimizing for three diverse tasks simultaneously. The variance in performance across different configurations highlights the intricate balance required in model tuning and task weighting to achieve optimal results in all tasks. Future work may explore strategies to mitigate the observed discrepancies, potentially through advanced model architectures or task-specific adjustments.

6.3.2.3 Four Tasks

In the conclusive experimental setup involving four tasks simultaneously, the MTL framework was tasked with optimizing the number of pulses, angle prediction,

fluence prediction, and pattern type classification. This comprehensive approach aimed to further test the model’s adaptability and efficiency across a broader spectrum of tasks. The results from this configuration are as follows:

The mean MSE for the number of pulses was recorded at 0.365 with a corresponding RMSE of 0.02. Angle prediction yielded a mean MSE of 26.72 and an RMSE of 0.515, reflecting the model’s consistent performance in geometric estimations. For fluence prediction, the mean MSE escalated to 1519.75, accompanied by an RMSE of 1.0006, indicating a significant challenge in accurately predicting fluence within this complex multi-task context. Notably, the mean accuracy for pattern type classification was observed at 42.43%.

This configuration underscores the intricate balance and the inherent trade-offs involved in optimizing multiple tasks simultaneously within the MTL framework. The increased complexity of handling four tasks is evident in the diverse performance metrics, particularly highlighting the model’s struggle with fluence prediction while achieving a notable MSE in the number of pulses prediction. Future enhancements may focus on addressing these challenges through refined model architectures or task-specific optimization strategies. Refer to Table 6.3 for a comprehensive comparison of the best results achieved across different configurations for both synthetic and real data sets.

Table 6.3: Best results for each configuration on Synthetic and Real data. With green color, we indicate the results that outperformed the single task model.

Data Type	Tasks	Number of Pulses	Angle	Fluence	Pattern Type
Synthetic	Two Tasks	-	0.38	2.32	99.9(%)
	Three Tasks	-	0.008	21.75	34.5(%)
Real	Two Tasks	0.34	20.5	187.38	96.3(%)
	Three Tasks	0.353	14.66	230.69	31.84(%)
	Four Tasks	0.365	26.72	1519.75	42.42(%)

Chapter 7

Conclusion and Future Work

This thesis represents a pioneering effort in the application of deep learning techniques to predict laser parameters from images capturing morphological patterns induced by laser processing. Through meticulous experimentation and analysis, we have demonstrated that CNNs, both in Single-task learning and Multi-task learning configurations, are highly effective in extracting and learning from the complex spatial features present in laser-induced patterns on material surfaces.

Our work has established a foundational framework for utilizing deep learning in the analysis and control of laser processing outcomes. The performance of our models on synthetic datasets was near-perfect, underscoring the capability of the proposed deep learning architectures to accurately learn and predict laser parameters. This success highlights the potential of deep learning as a transformative tool for advancing laser processing technologies.

However, the transition from synthetic to real datasets revealed challenges inherent in dealing with natural data variations, including uncertainties, irregularities, and limited sample sizes. Despite these challenges, the models exhibited significant predictive capability, marking a substantial step forward in applying machine learning to direct laser interference patterning and related laser processing techniques.

Our future research agenda is ambitious and multifaceted. We plan to expand our data collection efforts to address the limitations imposed by the size and variability of real datasets. By acquiring a broader range of images that capture a wider variety of laser-induced surface morphologies, we aim to enhance the robustness and accuracy of our predictive models.

One notable limitation in our work is the potential of exploring transfer learning strategies to enhance predictive performance in laser-induced surface structuring. Foundation models, which are pretrained on extensive datasets, hold the promise of leveraging universal representations that could be particularly beneficial for our domain. However, the direct application and effectiveness of these models in our

context have yet to be fully explored. Future efforts should be directed towards assessing the viability of transfer learning strategies in improving our model's adaptability to new, unseen data. This entails a systematic examination of pretrained models to ascertain their compatibility and performance concerning our specific predictive tasks.

Given the promising results obtained from initial Multi-Task Learning (MTL) experiments, further investigation into more sophisticated MTL architectures and training methodologies is warranted. This includes exploring dynamic weighting of task losses and investigating novel network architectures that can more effectively share knowledge between tasks.

Moving beyond discriminative tasks, we aim to develop generative deep learning models capable of synthesizing images of material surfaces based on specified laser parameters. This generative approach could revolutionize the design and optimization of laser processing experiments, enabling predictive modeling of surface morphologies.

Bibliography

- [1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.
- [2] Sabri Alamri, Fotis Fraggelakis, Tim Kunze, Benjamin Krupop, Girolamo Mincuzzi, Rainer Kling, and Andrés Fabián Lasagni. On the interplay of dlip and lipss upon ultra-short laser pulse irradiation. *Materials*, 12(7):1018, 2019.
- [3] Laleh Armi and Shervan Fekri-Ershad. Texture image analysis and texture classification methods-a review. *arXiv preprint arXiv:1904.06554*, 2019.
- [4] Manish H Bharati, J Jay Liu, and John F MacGregor. Image texture analysis: methods and comparisons. *Chemometrics and intelligent laboratory systems*, 72(1):57–71, 2004.
- [5] Jörn Bonse, Sandra Höhm, Sabrina V Kirner, Arkadi Rosenfeld, and Jörg Krüger. Laser-induced periodic surface structures—a scientific evergreen. *IEEE Journal of selected topics in quantum electronics*, 23(3), 2016.
- [6] Subhash Chandra Singh Chunlei Guo. Handbook of laser technology and applications lasers applications: Materials processing and spectroscopy. 2021.
- [7] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.
- [8] Ujjawal Dixit, Apoorva Mishra, Anupam Shukla, and Ritu Tiwari. Texture classification using convolutional neural network optimized with whale optimization algorithm. *SN Applied Sciences*, 1:1–11, 2019.
- [9] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 2022.
- [10] W. W. Duley. Laser processing and analysis of materials. 2012.
- [11] Fotis Fraggelakis, Girolamo Mincuzzi, John Lopez, Inka Manek-Hönninger, and Rainer Kling. Controlling 2d laser nano structuring over large area with double femtosecond pulses. *Applied Surface Science*, 470:677–686, 2019.

- [12] Fotis Fraggelakis, George D Tsibidis, and Emmanuel Stratakis. Tailoring sub-micrometer periodic surface structures via ultrashort pulsed direct laser interference patterning. *Physical Review B*, 103(5):054105, 2021.
- [13] Fotis Fraggelakis, George D Tsibidis, Emmanuel Stratakis, et al. Ultrashort pulsed laser induced complex surface structures generated by tailoring the melt hydrodynamics. *Opto-Electron. Adv*, 5:210052, 2022.
- [14] Shin Fujieda, Kohei Takayama, and Toshiya Hachisuka. Wavelet convolutional neural networks for texture classification. *arXiv preprint arXiv:1707.07394*, 2017.
- [15] Goëry Genty, Lauri Salmela, John M Dudley, Daniel Brunner, Alexey Kokhanovskiy, Sergei Kobtsev, and Sergei K Turitsyn. Machine learning and applications in ultrafast photonics. *Nature Photonics*, 15(2):91–101, 2021.
- [16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.
- [17] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.
- [18] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.
- [19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [20] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.
- [21] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [22] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [23] Xiu Liu and Chris Aldrich. Deep learning approaches to image texture analysis in material processing. *Metals*, 12(2):355, 2022.
- [24] Nikolaos Livakas, Evangelos Skoulas, and Emmanuel Stratakis. Omnidirectional iridescence via cylindrically-polarized femtosecond laser processing. *Opto-Electronic Advances*, 3(5):190035–1, 2020.

- [25] Andrzej Materka, Michal Strzelecki, et al. Texture analysis methods—a review. *Technical university of lodz, institute of electronics, COST B11 report, Brussels*, 10(1.97):4968, 1998.
- [26] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 12 1943.
- [27] Benjamin Mills and James A Grant-Jacob. Lasers that learn: The interface of laser machining and machine learning. *IET Optoelectronics*, 15(5):207–224, 2021.
- [28] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.
- [29] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.
- [30] Peter Schaaf. Laser processing of materials: fundamentals, applications and developments. 2010.
- [31] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018.
- [32] Philomina Simon and V Uma. Deep learning based feature extraction for texture classification. *Procedia Computer Science*, 171:1680–1687, 2020.
- [33] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.
- [34] E Stratakis, Jörn Bonse, J Heitz, Jan Siegel, GD Tsibidis, E Skoulas, A Papadopoulos, A Mimidis, A-C Joel, P Comanns, et al. Laser engineering of biomimetic surfaces. *Materials Science and Engineering: R: Reports*, 141:100562, 2020.
- [35] Szilvia Szeghalmy and Attila Fazekas. A comparative study of the use of stratified cross-validation and distribution-balanced stratified cross-validation in imbalanced learning. *Sensors*, 23(4):2333, 2023.
- [36] Ioannis Tsamardinos, Elissavet Greasidou, and Giorgos Borboudakis. Bootstrapping the out-of-sample predictions for efficient and accurate cross-validation. *Machine learning*, 107:1895–1922, 2018.
- [37] George D Tsibidis and Emmanuel Stratakis. Ultrafast laser biomimetic micro-/nanostructuring. *Ultrafast Laser Nanostructuring: The Pursuit of Extreme Scales*, pages 921–949, 2023.

- [38] Maria-Christina Velli, George D Tsibidis, Alexandros Mimidis, Evangelos Skoulas, Yannis Pantazis, and Emmanuel Stratakis. Predictive modeling approaches in laser-based material processing. *Journal of Applied Physics*, 128(18):183102, 2020.