2211MC09

by Intaj Ahmed Choudhury

Submission date: 19-Jun-2025 03:33PM (UTC+0530)
Submission ID: 2702242591
File name: 2211MC09.pdf (3.71M)
Word count: 8186
Character count: 47878

Incorporating Constraints into Machine Learning Models

M. Tech Project Report Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Technology
in
Mathematics & Computing

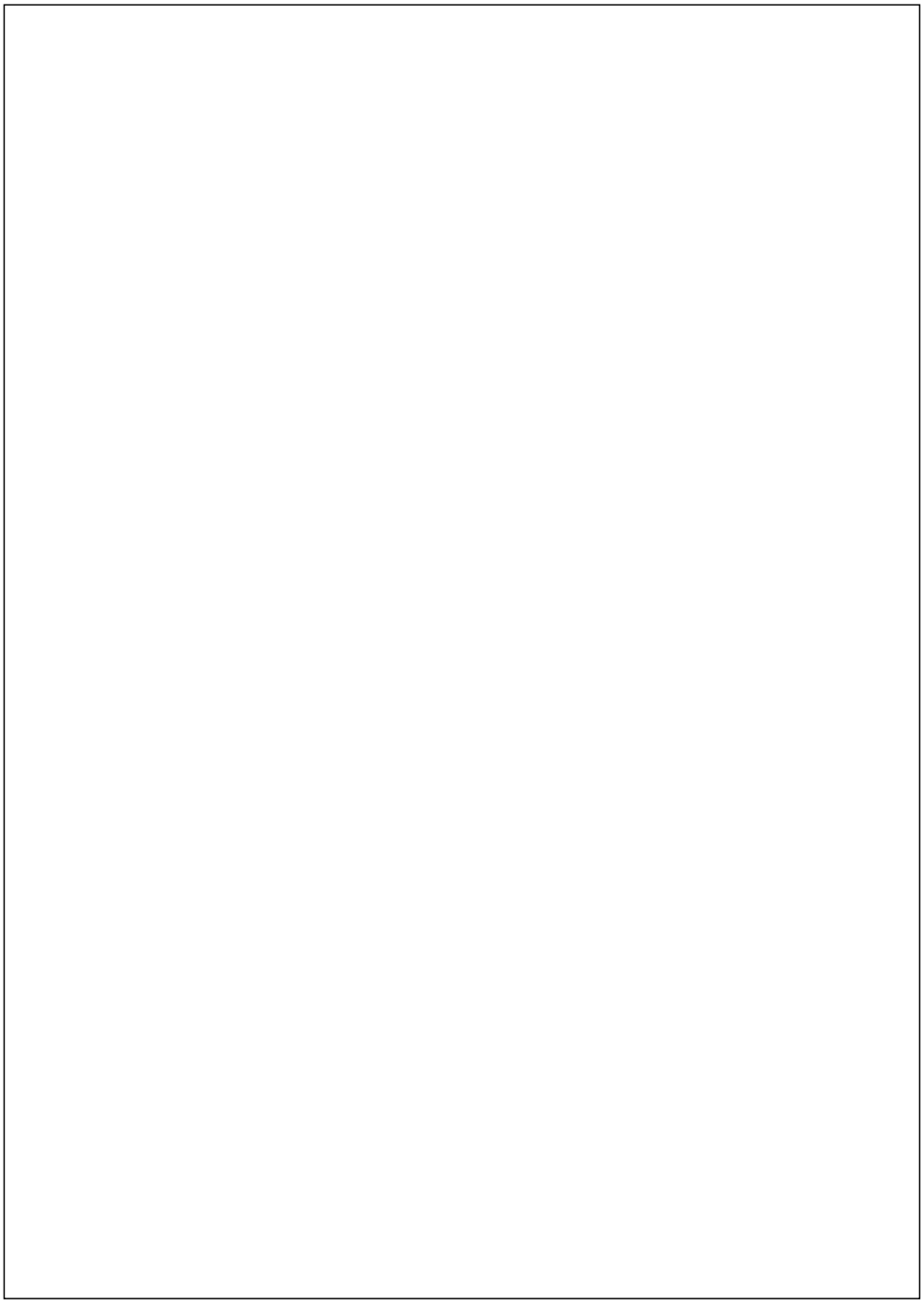
by

Intaj Ahmed Choudhury
(2211MC09)

under the guidance of

Prof. Jimson Mathew

DEPARTMENT OF MATHEMATICS
INDIAN INSTITUTE OF TECHNOLOGY PATNA
PATNA - 801103, BIHAR



CERTIFICATE

This is to certify that the work contained in this thesis entitled " ***Incorporating Constraints into Machine Learning Models****" is a bonafide work of* ***Intaj Ahmed Choudhury*** *(****Roll No. 2211MC09****), carried out in the Department of Mathematics, Indian Institute of Technology Patna under my supervision and that it has not been submitted elsewhere for a degree.*

Supervisor: **Prof. Jimson Mathew**

Assistant/Associate Professor,
Department of Computer Science & Engineering,
Indian Institute of Technology Patna, Bihar.

June, 2025
Patna.

DECLARATION

I certify that:

- The research work presented in this thesis is the result of my own efforts, carried out under the supervision and guidance of my faculty advisor.
- This document has not been submitted previously, in part or whole, to any other university or institution for the award of any degree or diploma.
- All institutional guidelines regarding thesis writing and submission have been strictly followed.
- I have ensured compliance with the ethical standards set forth by the institution during the course of this research.
- All external sources, including literature, data, and theoretical contributions, have been appropriately cited within the thesis and acknowledged in the references section.
- The thesis has undergone scrutiny through plagiarism detection software to ensure originality.

Intaj Ahmed Choudhury

2211MC09

Acknowledgements

Upon the successful completion of my project **Incorporating Constraints into Machine Learning Models**, I would like to express my deepest gratitude to all those who contributed to this journey.

First and foremost, I am profoundly thankful to my supervisor, **Dr. Jimson Mathew**, for his constant guidance, expert advice, and patience throughout the course of this work. His valuable insights, timely feedback, and encouragement played a crucial role in shaping the direction and outcome of my research.

I am also immensely grateful to my fellow M.Tech friends. Their unwavering support, candid conversations, and shared experiences helped me stay grounded and sane throughout this rigorous academic journey. This endeavor would not have been as engaging or memorable without their presence.

Lastly, and most importantly, I extend my heartfelt thanks to my family, my parents and sister for their endless love, support, and understanding. Their belief in me, even during the most challenging times, has been the backbone of my academic and personal growth. This thesis is dedicated to them with deep appreciation and affection.

Abstract

Classification will probably be among the first and most extensively researched tasks in machine learning. Deep models perform amazingly well, but they are likely to be suffering from problems such as overfitting and bad generalization, particularly when they are trained under constrained data or computing power. Traditionally, model training relies on abundant data and extensive training, which could not always be feasible. Here, there is a presentation of a constraint-based approach that facilitates learning by incorporating rules of logic or domain knowledge into the training process itself. The constraints guide the model to take more informed decisions, even in initial training. The proposed framework addresses two primary issues simultaneously, improving precision at low training epochs and reducing overfitting without losing interpretability

One of the primary contributions of the work is automatic learning of new constraints from data in an information-theoretic manner. The constraints elicited from data capture the underlying patterns and interactions that are subsequently translated into comprehensible logical rules. The methodology, through experiments on datasets, demonstrates that the use of a mix of pre-defined and learned constraints provides better performance, fast convergence, and stable predictions and thus is useful in low-resource environments and real-world applications.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Machine learning (ML) has revolutionized a wide range of fields—from image and speech recognition to diagnosis and autonomous agents—over the last few years. Most modern ML models, especially deep neural networks, however, are data-driven in usage. They acquire sophisticated mapping from input to output from empirical data only, often without regard to logical, physical, or domain-specific laws. While this has served well to achieve state-of-the-art performance on many tasks, it has major limitations: bad generalization to out-of-sample data, overfitting in low-data settings, and interpretability.

Such limitations are particularly problematic in practice settings where data are small, imbalanced, or noisy—conditions that are common in safety-critical applications like industrial automation, finance, and healthcare. Most such applications are also governed by known rules, relationships, or operating constraints. Forgetting such prior knowledge not only leads to inefficiencies in training, but also unfaithful or unexplainable models.

To cope with such issues, constraint-based learning has proved to be a good approach. Instead of relying solely on data, this approach introduces soft constraints—domain knowledge or structural properties in the form of rules—to training. The constraints are typically defined as differentiable functions and incorporated in the loss function of the model as regularization terms. This maintains the model stable and semantically sound, especially in the early stages of training when data signals are poor. The aim of this thesis is to explore how the incorporation of such constraints during training time of neural networks improves accuracy, generalization, and

explainability, predominantly for the low-resource scenario. Besides pre-specified constraints, we also explore learning of new constraints automatically from data using an information-theoretic framework, as presented in the paper "A Constraint-Based Approach to Learning and Explanation." Such new learned constraints are explainable and first-order logic-embeddable, making the model's behavior more interpretable.

This method is tested on a variety of datasets including handwritten digits, natural images, and structured inputs. The results indicate that constraint models outperform normative models in the initial epochs of training and generalize better under restricted training data or compute. Additionally, through examining constraint violations and inducing symbolic rules, we show the explainability advantages of our framework.

This thesis therefore proposes an integrated methodology to building ML models that not only are data-efficient and performing, but also logically sound and explainable, in support of the overall goal of explainable and reliable AI.

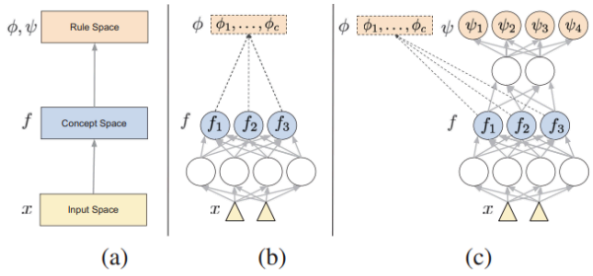
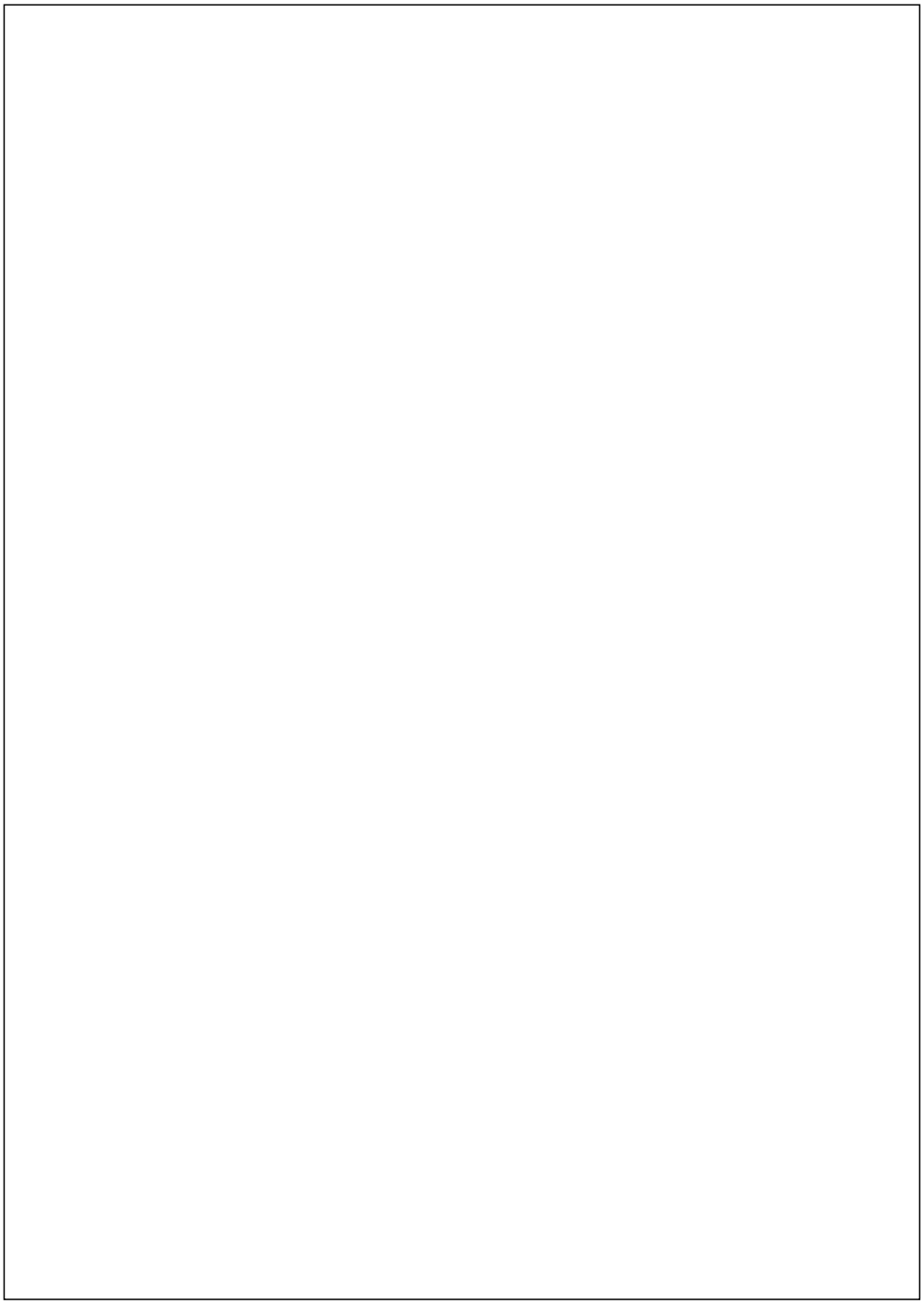


Fig. 1.1: Learning task functions subject to constraints

The figure above illustrates the following: (a) It shows the input, concept, and rule spaces corresponding to the input data x, the task functions $f(x)$, and the constraints represented as $\phi_j(f(x)) = 0,\ j = 1, \ldots, c$, and $\psi_z(f(x)) = 0,\ z = 1, \ldots, m$. (b) This part represents the learning of task functions under *predefined constraints* $\phi_j(f(x)) = 0,\ j = 1, \ldots, c$ (where dotted lines denote identity mappings). (c) This section depicts the simultaneous learning of task functions with both predefined and *learned constraints* $\psi_z(f(x)) = 0,\ z = 1, \ldots, m$, where in this scenario, $m = 4$.

1.2 Problem Statement

Despite the phenomenal advancements in machine learning, particularly in classification with deep learning, the majority of models remain data-driven with no ability to leverage prior knowledge or logical structure. This leads to several limitations such as overfitting to small datasets, poor generalization to new settings, and less interpretability of model decisions. In most practical applications, high-level knowledge—often expressible as logical or mathematical constraints—of domain experts is not leveraged at training time. Such constraints can be employed to guide models towards more interpretable and stronger learning outcomes. The catch, however, is how to express such constraints in the optimal way possible, especially soft ones that can be violated to some extent, and how to effectively leverage them during training. Methods need to be developed that not only leverage pre-specified constraints but also learn new, interpretable constraints from data autonomously without compromising the accuracy, generality, and explainability of the model. This thesis tries to bridge such gaps by proposing a constraint-based learning approach that integrates constraint regularization in the loss function to realize improved performance and transparency in classification problems.



Chapter 2

Review of Previous Works

2.1 Symbolic Constraint Learning from Examples

Initial work in learning constraints involved extracting logical rules or symbolic constraints from labeled examples. Valiant's seminal PAC framework proposed an algorithm for learning Boolean k-CNF formulas from examples, and that was a key piece of theory work [10]. Later, De Raedt et al. surveyed practical methods for learning symbolic constraints from experience in scheduling, configuration, and interactive tasks [5]. These approaches perform satisfactorily in well-structured symbolic domains but do not work with noisy, high-dimensional inputs like images.

2.2 Soft Constraint Integration in Neural Networks

The integration of domain knowledge into high-capacity models like neural networks typically entails the integration of soft constraints as regularization terms in the loss function. Stewart and Ermon [9] introduced physics-based constraints to improve object trajectory prediction, demonstrating that such constraints can improve generalization and enable learning with less supervision. Xu et al. [11] in a related paper introduced a semantic loss function to ensure logical consistency of the label space. These methods support smooth, differentiable optimization while enabling integration of prior knowledge.

2.3 A Constraint-Based Approach to Learning and Explanation

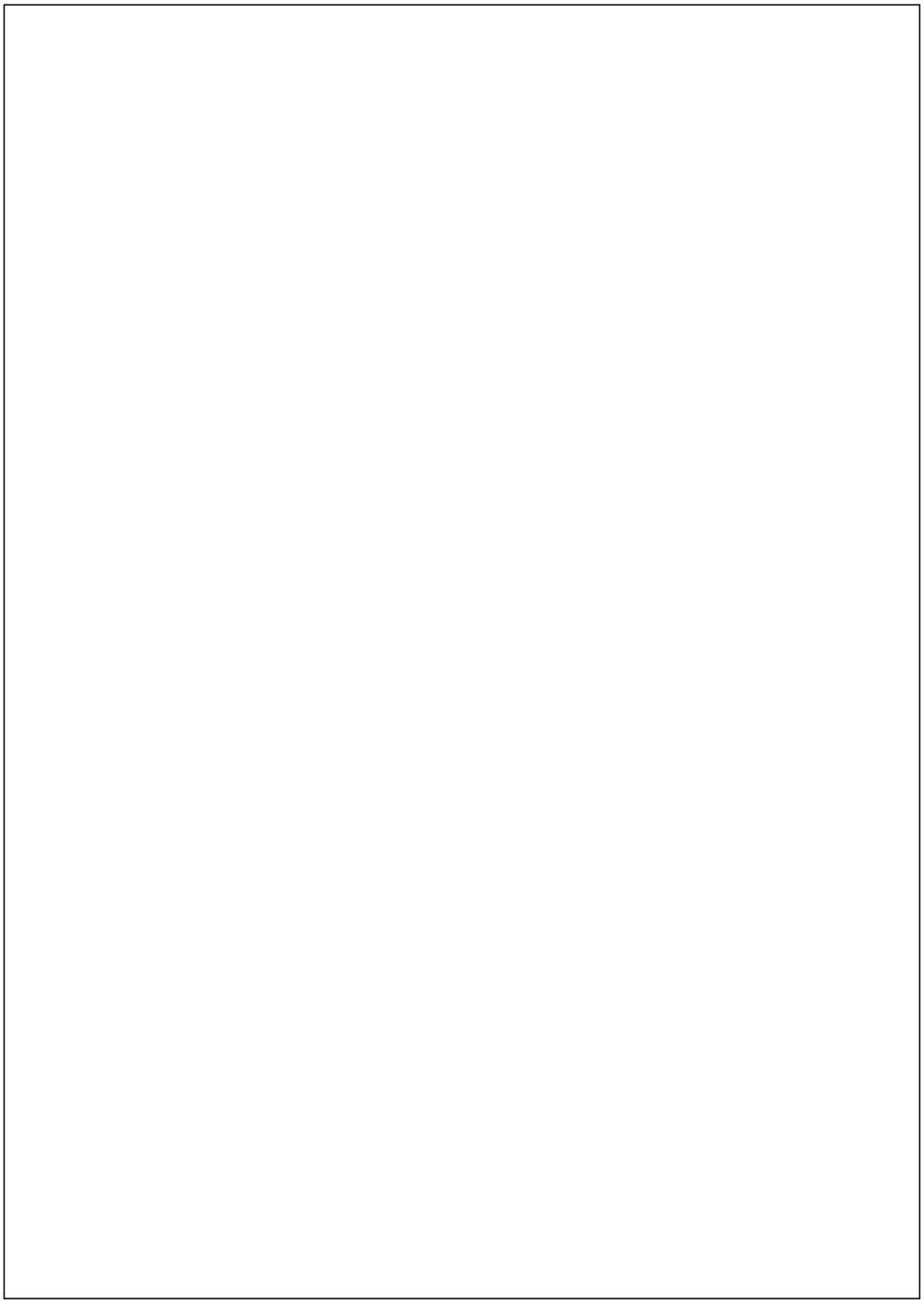
The model by Ciravegna et al. [2] is a generic framework for learning new soft constraints and incorporating existing soft constraints from data. The constraints are incorporated with a differentiable loss regularizer and are optimized together with the model parameters. The learned constraints can also be accessed and formulated as First-Order Logic (FOL) rules, with interpretability. The method revealed enhanced accuracy on MNIST in early training epochs and overfitting prevention.

2.4 Neuro-Symbolic and Rule-Guided Reasoning

Neuro-symbolic AI attempts to combine the learning ability of deep neural networks with the reasoning ability of logic-based systems. Logic Tensor Networks (LTNs) [7] enable joint reasoning and learning with fuzzy logic. ENRL [8] provides explainable neural rule learning, combining pattern mining with end-to-end differentiability. IRCNet [6] utilizes rule completion to improve reasoning from incomplete knowledge graphs. These approaches push the boundaries of combining structured reasoning in machine learning but are likely to require task-specific rules.

2.5 Explaining Neural Decisions via Rule Extraction

Contemporary approaches to enhancing model interpretability have aimed at harvesting symbolic rules from black box models. The eclectic rule extraction method [3] employs decision trees induced from internal activations to imitate the behavior of the model. SELOR [1] is an explainable logical reasoning system that leverages transparent rule architectures for prediction, hence inducing transparency. FuzRED [4] integrates fuzzy logic and model-agnostic attribution methods (e.g., SHAP) to derive high-precision logical rules from tabular and image data.



Chapter 3

Methodology

The goal of this work is to incorporate domain-specific logical constraints into machine learning models to improve classification accuracy, generalizability, and interpretability. The central concept is to impose pre-specified rules or create new rules while undergoing model training using soft constraints, which are directly integrated into the loss function. The chapter describes the theoretical paradigm, constraint representation techniques, the paradigm of model training, and the implementation pipeline in this research.

3.1 Representing Soft Constraints

Constraints in machine learning can be either **hard** (must always be satisfied) or **soft** (preferred but can be violated to some extent). Since strict constraints often make optimization intractable, we focus on soft constraints and express them in a differentiable form that can be embedded into the model's loss function.

Each constraint is formulated as a fuzzy logic function $\phi_h(f(x))$ that returns values close to 1 when the constraint is satisfied and less than 1 otherwise. The cost of violating constraint h is thus $\hat{\phi}_h = 1 - \phi_h$, which forms the basis for the constraint loss.

3.2 Loss Function with Constraints

Let $f(x_j)$ denote the model prediction on sample x_j, and let $\mathcal{Z}$ be the training dataset of size $|\mathcal{Z}|$. For $\mathcal{L}$ different constraints, each with weight μ_h, the regularization term for constraints is given by:

$$\Phi(f, \mathcal{Z}, \mu) = \frac{1}{|\mathcal{Z}|} \sum_j \sum_h \mu_h \cdot \hat{\phi}_h(f(x_j))$$

This constraint regularization is added to the original supervised loss. For a classification problem using cross-entropy, the total loss becomes:

$$\mathcal{L} = H(\hat{y}, y) + \lambda \cdot \Phi(f, \mathcal{Z}, \mu)$$

Here, $H(\hat{y}, y)$ is the cross-entropy between predicted and true labels, λ is a hyperparameter that balances label supervision and constraint enforcement, and Φ is the average constraint violation penalty.

3.3 Incorporating Constraints into the Learning Process

Incorporating constraints during model training ensures that predictions are not just driven by data patterns but also comply with logical rules and domain knowledge. This section describes how constraints are predefined, integrated into the training loss, and how new constraints can be discovered through structured reasoning.

3.3.1 Defining Predefined Constraints

For each data set, a set of principles is formulated in a logical framework and translated into differentiable constraint functions. A few common ones are symmetry, mutual exclusivity, or attribute-level relationships.

3.3.2 Insertion of Constraints in Loss Function

Predefined as well as learned constraints are inserted into the loss function as per the above formulation, it is done with the help of regularization term in the loss function to make the model adhere to the constraints.

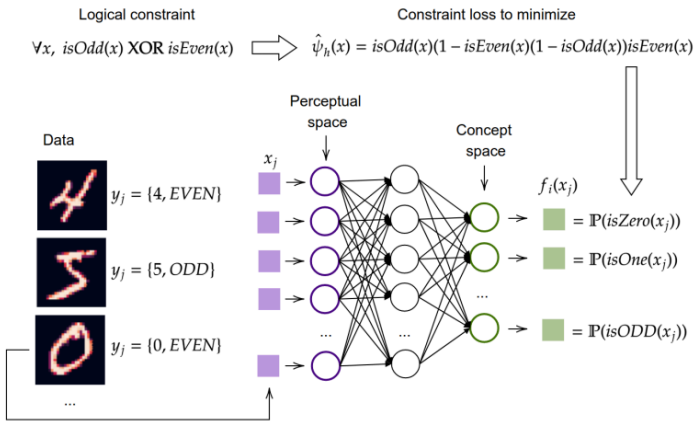


Fig. 3.1: Learning with Constraints on MNIST dataset

3.3.3 Learning New Constraints

Having trained a base classifier N_1 for mapping input images to high-level abstractions (e.g., class labels, parity), we train an auxiliary model N_2 to discover logical relationships between these concepts. The extracted rules are validated against a truth table for interpretability.

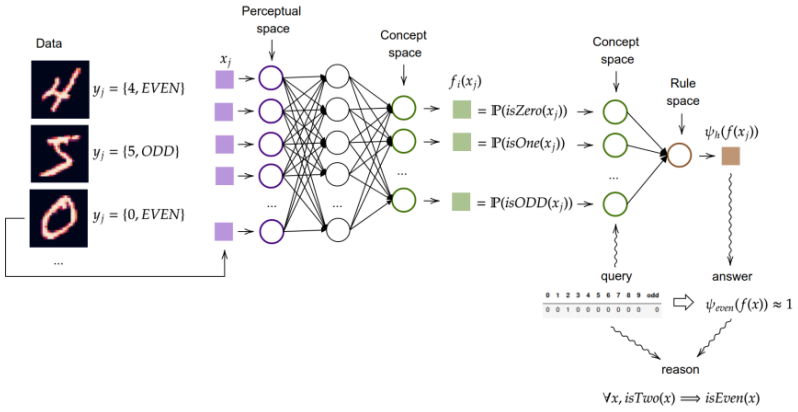


Fig. 3.2: Learning of new Constraints on MNIST dataset

3.4 Dataset and Experimentation Setup

This section outlines the datasets used, the preparation steps for constraint integration, the neural network architecture employed, and the experimentation pipeline designed to compare standard training against constraint-based learning.

3.4.1 Dataset Details

We use a diverse range of standard image classification datasets to evaluate the impact of constraints on generalization and interpretability:

Table 3.4.1: Datasets Used

Dataset	**Image Size**	**Classes**	**Train/Test**	**Description**
MNIST	28×28	10	60K / 10K	Handwritten grayscale digits (0–9).
EMNIST (Digits)	28×28	10	240K / 40K	Extended digit-only version of MNIST with more samples and balanced digit distribution.
KMNIST	28×28	10	60K / 10K	Japanese cursive characters (Kuzushiji).
Fashion MNIST	28×28	10	60K / 10K	Zalando article images (T-shirts, shoes, etc.).
CIFAR-10	32×32	10	50K / 10K	RGB images of real-world objects (planes, dogs).
STL-10	96×96	10	5K / 8K	Higher-resolution variant of CIFAR-10.
SVHN	32×32	10	73K / 26K	Digits from real-world house numbers with noise.

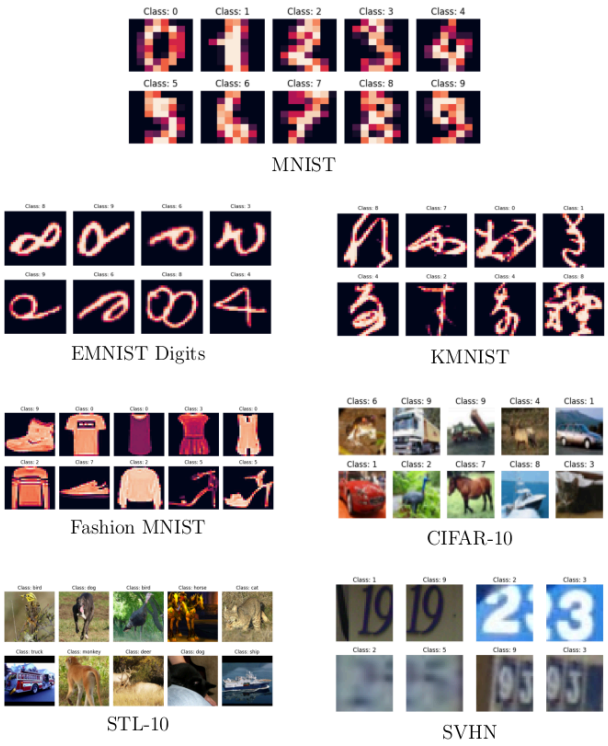


Fig. 3.3: Sample images from each dataset

3.4.2 Data Preparation (with MNIST Case Study)

To demonstrate the use of constraints, we first focus on the MNIST dataset:

Output Modification: The target vector is one-hot encoded to produce 10 output units corresponding to digits 0–9. Two additional binary outputs **EVEN** and **ODD** are added, increasing the output vector to 12 dimensions.

Soft Attribute Encoding: For training purposes, all labels are initially assigned soft membership values of 0.4 for EVEN and ODD, introducing fuzzy supervision for these attributes.

Constraint Integration: The training incorporates three primary constraints:

1. Mutual Exclusivity of Digit Classes: An image should belong to exactly one digit class.
2. Digit-to-Attribute Consistency: Even-numbered digits should activate the EVEN unit and

odd-numbered digits the ODD unit.

3. Attribute Mutual Exclusivity: EVEN and ODD activations should not be simultaneously high.

These constraints are encoded as differentiable functions and embedded into training loss.

```
# N(1,3,5,7,9) => ODD
mu * (ONE * (1. - ODD)),
mu * (THREE * (1. - ODD)),
mu * (FIVE * (1. - ODD)),
mu * (SEVEN * (1. - ODD)),
mu * (NINE * (1. - ODD)),

# N(0,2,4,6,8) => EVEN
mu * (ZERO * (1. - EVEN)),
mu * (TWO * (1. - EVEN)),
mu * (FOUR * (1. - EVEN)),
mu * (SIX * (1. - EVEN)),
mu * (EIGHT * (1. - EVEN)),

# XOR ON THE MAIN CLASSES
mu * (
        (1 - ((ZERO) * (1 - ONE) * (1 - TWO) * (1 - THREE) * (1 - FOUR) * (1 - FIVE) * (1 - SIX) * (
                    1 - SEVEN) * (1 - EIGHT) * (1 - NINE))) *
        (1 - ((1 - ZERO) * (ONE) * (1 - TWO) * (1 - THREE) * (1 - FOUR) * (1 - FIVE) * (1 - SIX) * (
                    1 - SEVEN) * (1 - EIGHT) * (1 - NINE))) *
        (1 - ((1 - ZERO) * (1 - ONE) * (TWO) * (1 - THREE) * (1 - FOUR) * (1 - FIVE) * (1 - SIX) * (
                    1 - SEVEN) * (1 - EIGHT) * (1 - NINE))) *
        (1 - ((1 - ZERO) * (1 - ONE) * (1 - TWO) * (THREE) * (1 - FOUR) * (1 - FIVE) * (1 - SIX) * (
                    1 - SEVEN) * (1 - EIGHT) * (1 - NINE))) *
        (1 - ((1 - ZERO) * (1 - ONE) * (1 - TWO) * (1 - THREE) * (FOUR) * (1 - FIVE) * (1 - SIX) * (
                    1 - SEVEN) * (1 - EIGHT) * (1 - NINE))) *
        (1 - ((1 - ZERO) * (1 - ONE) * (1 - TWO) * (1 - THREE) * (1 - FOUR) * (FIVE) * (1 - SIX) * (
                    1 - SEVEN) * (1 - EIGHT) * (1 - NINE))) *
        (1 - ((1 - ZERO) * (1 - ONE) * (1 - TWO) * (1 - THREE) * (1 - FOUR) * (1 - FIVE) * (SIX) * (
                    1 - SEVEN) * (1 - EIGHT) * (1 - NINE))) *
        (1 - ((1 - ZERO) * (1 - ONE) * (1 - TWO) * (1 - THREE) * (1 - FOUR) * (1 - FIVE) * (1 - SIX) * (
            SEVEN) * (1 - EIGHT) * (1 - NINE))) *
        (1 - ((1 - ZERO) * (1 - ONE) * (1 - TWO) * (1 - THREE) * (1 - FOUR) * (1 - FIVE) * (1 - SIX) * (
                    1 - SEVEN) * (EIGHT) * (1 - NINE))) *
        (1 - ((1 - ZERO) * (1 - ONE) * (1 - TWO) * (1 - THREE) * (1 - FOUR) * (1 - FIVE) * (1 - SIX) * (
                    1 - SEVEN) * (1 - EIGHT) * (NINE)))
),

# XOR ON THE ATTRIBUTE CLASSES
mu * (
        (1 - (EVEN) * (1 - ODD)) *
        (1 - (1 - EVEN) * (ODD))
),
```

Fig. 3.4: Constraint Penalty function for MNIST dataset

3.4.3 Model Architecture

A simple **feedforward neural network** is implemented in PyTorch by subclassing `torch.nn.Module`. While the core structure remains consistent across datasets, the input and output dimensions vary depending on the specific dataset. For instance, MNIST and Fashion MNIST contain 28×28 grayscale images (i.e., 784 input features), whereas STL-10 uses 96×96 RGB images (3×96×96 = 27,648 input features). Similarly, the number of output neurons depends on the number of class labels plus any attribute units (e.g., EVEN/ODD), ranging from 12 (for MNIST) to 16 or more (for STL-10).

The architecture includes three fully connected (Linear) layers:

Input Layer: Accepts a flattened image vector (e.g., 784 for MNIST or 27,648 for STL-10).

Hidden Layer 1: Fully connected layer mapping input to 256 neurons, followed by ReLU activation.

Hidden Layer 2: Another fully connected layer with 256 neurons, followed by ReLU activation.

Output Layer: Maps 256 hidden units to D_{out} neurons (number of classes + attribute units), followed by a sigmoid activation function for fuzzy multi-label classification.

Input shape: (batch_size, input_features)

Hidden activations: ReLU

Output activation: Sigmoid

Loss: Binary cross-entropy combined with a weighted constraint regularization term

Trainable Parameters (Example for MNIST):

- **Layer 1:** $(784 \times 256 + 256) = 200{,}960$
- **Layer 2:** $(256 \times 256 + 256) = 65{,}792$
- **Output Layer:** $(256 \times 12 + 12) = 3{,}084$
- **Total:** $\approx 269{,}836$ parameters

For larger datasets like STL-10, the model's capacity increases proportionally due to the higher dimensional input and expanded output space. This architecture balances simplicity with flexibility, making it suitable for concept learning under constraint-based regularization across various image classification tasks.

3.4.4 Experiment Steps

The experimentation pipeline for evaluating the effectiveness of constraint-based learning consists of the following key phases:

Dataset Preparation: Each dataset is preprocessed, including resizing, normalization, and label transformation. In the case of MNIST, the output vectors are expanded to include both digit classes (0–9) and two soft logic attributes: EVEN and ODD. Logical constraints are defined at this stage as differentiable functions for training.

Training with and without Constraints: Two models are trained in parallel to evaluate the benefits of constraint-based learning: one follows the traditional data-driven approach, and the other incorporates predefined logical constraints during training. In the **baseline model**, the network is trained using only the standard binary cross-entropy loss function. It learns to classify based solely on the input-output pairs provided in the dataset, without any additional guidance or structure. In contrast, the **constrained model** is designed to learn not only from labeled data but also from domain-specific knowledge encoded as soft constraints. These predefined constraints are formulated based on the logical structure of the task. For example, in the case of digit classification, we know that each digit should belong to only one class, even numbers should activate the EVEN unit, and ODD and EVEN should not be active at the same time. These constraints are implemented as differentiable conditions and added to the training objective, encouraging the model to generate outputs that satisfy these rules. By incorporating these rules into the learning process, the model is guided toward more semantically consistent behavior, even when data is limited or noisy. This strategy improves generalization, especially in early epochs, and leads to more interpretable outputs by aligning predictions with human-understandable logic. It also reduces the chance of overfitting by narrowing the solution space to only those outputs that make logical sense.

Evaluation Metrics: Models are evaluated using accuracy across training epochs. Special attention is given to early-epoch accuracy, as constraint-guided models are expected to generalize better with fewer training steps. Additional metrics include test accuracy and training efficiency.

Constraint Violation Analysis: During and after training, we monitor how often the constrained and unconstrained models violate predefined logical rules. This analysis is used to

demonstrate the logical consistency and robustness of constraint-based models.

Learning New Constraints: In this phase, the model is encouraged to learn interpretable logical rules directly from data using the following three-step approach:

Step 1: Train a base network N_1 to map input images to high-level semantic concepts (e.g., digit labels, EVEN/ODD), forming the concept space.

Step 2: Train a secondary network N_2 to infer one logical attribute (e.g., EVEN) based on the other predicted concepts (e.g., digit labels, ODD), without directly using the EVEN label.

Step 3: Create a truth table of all binary combinations of concept outputs from N_1 and pass them to N_2. The outputs are analyzed to derive symbolic rules (e.g., "odd implies not even"), demonstrating the model's capability for interpretable logic reasoning.

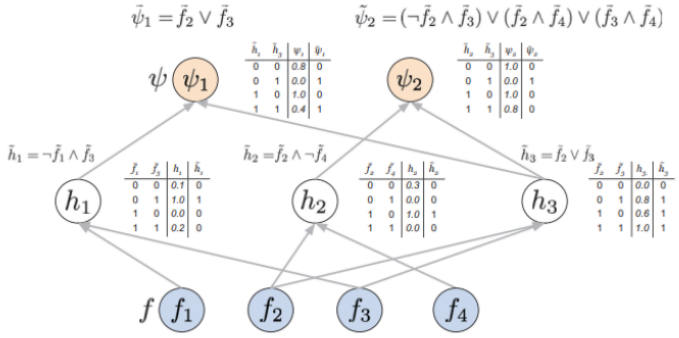
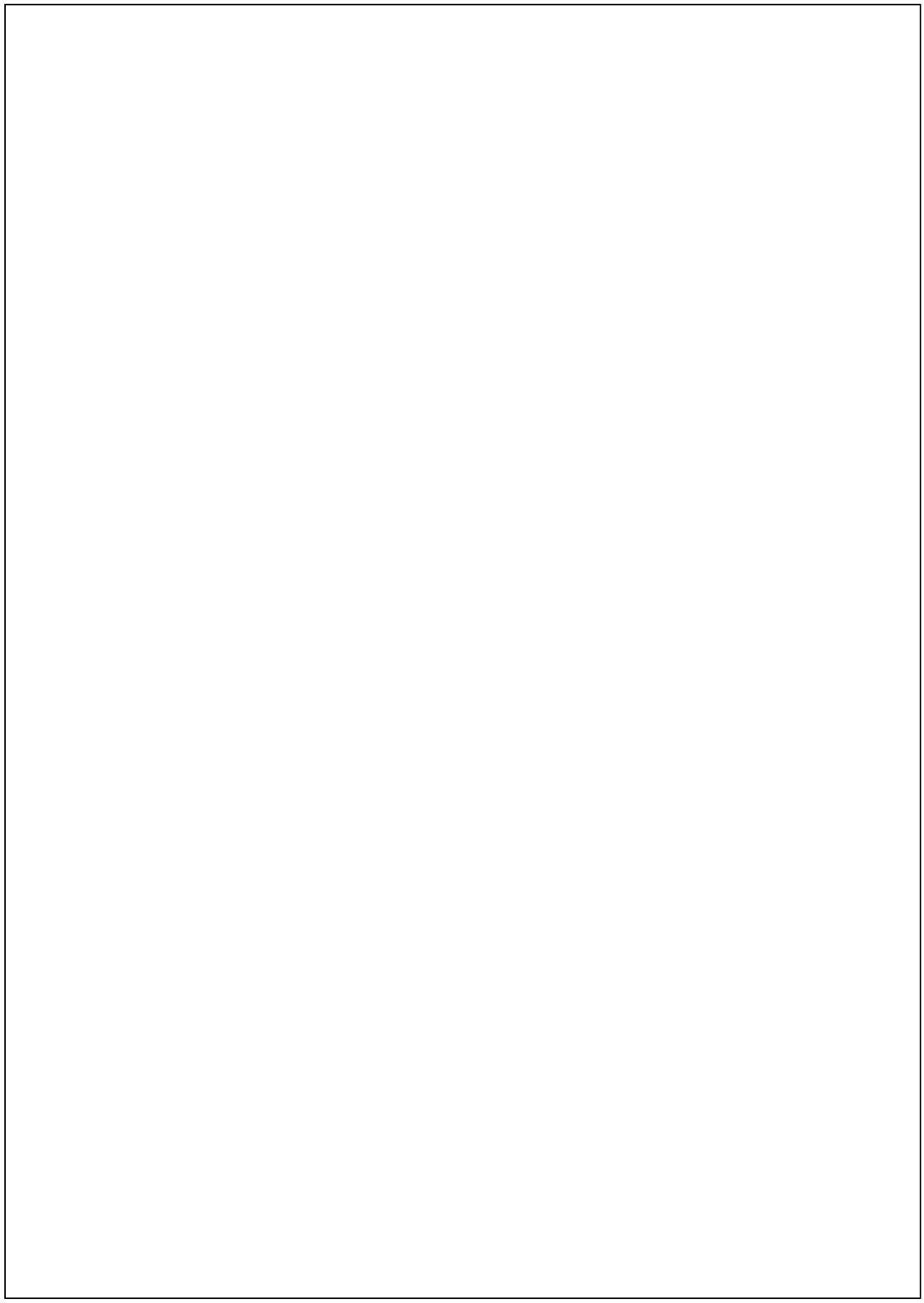


Fig. 3.5: Logic description devised using truth tables

The figure above serves as an illustration of ψ. It pairs hidden and output neurons with their respective truth tables (shown on the right) and the associated logical expressions (displayed at the top), constructed as previously explained. The truth tables present both the continuous output values of the neurons (third column) and their Boolean counterparts (last column). The logical formulas for ψ_1 and ψ_2 are obtained by combining the logical representations of the hidden layer neurons.

3.5 Implementation Tools and Libraries

All experiments in this thesis were conducted using Python because of its strong support for scientific computing and machine learning pipelines. The deep learning models were implemented and trained using PyTorch, which provides dynamic computation graphs and modularity that are well-suited for custom loss integration in the form of constraint regularization. NumPy and Pandas were utilized for numerical computations and dataset preparation. Matplotlib was utilized for visualization of the performance metrics, such as accuracy trends, loss curves, and constraint violation plots. Logical constraint analysis and transformation were enabled using SciPy, specifically for manipulation of differentiable logic operations. Furthermore, several custom modules were created for encoding logical rules, defining constraint cost functions, and directly integrating them into the training loop for a flexible and interpretable constraint-based learning pipeline.



Chapter 4

Result

This chapter presents the evaluation of constraint-based learning across multiple standard image classification datasets. The experiments measure test accuracy, constraint violation rates, and the interpretability of learned rules. The results show that incorporating constraints improves early learning performance, enforces logical consistency, and enables symbolic reasoning, which traditional deep learning models lack.

4.1 Accuracy Comparison Across Datasets

Models were trained in two modes, standard (without constraints) and constraint-based. The table below summarizes the test accuracy achieved by both models at selected epochs across five datasets.

Table 4.1.1: Classification Accuracy With and Without Constraints Across Datasets

Dataset	Epoch	With Constraints	Without Constraints
MNIST	50	0.5655	0.4383
	100	0.7630	0.7486
	200	0.9212	0.9212
	500	0.9880	0.9902
EMNIST	50	0.6572	0.6109
	100	0.8022	0.7600
	200	0.8660	0.8240
	500	0.9377	0.9620
Fashion MNIST	100	0.6430	0.5744
	200	0.7508	0.7566
	1000	0.9015	0.9540
CIFAR-10	10	0.9507	0.9158
	20	0.9234	0.9296
SVHN	10	0.9170	0.8434
	20	0.9650	0.9580

As shown above, models trained with constraints often reach comparable or higher accuracy earlier than the unconstrained versions. This makes constraint-based training particularly effective in scenarios where fast learning is critical or computational resources are limited.

4.2 When Constraints Help and When They Don't

Constraints tend to help early, particularly in datasets with interpretable structure. In noisy or imbalanced data (EMNIST, CIFAR), hard constraints may limit model flexibility if not properly tuned.

4.2.1 MNIST

In the MNIST dataset, the constraint-augmented model showed superior early-stage generalization, achieving 0.68 accuracy by the 80th epoch, compared to 0.643 for the baseline model. Although both models reached similar final accuracies (˜0.98 at epoch 500), the constrained model exhibited a smoother and faster convergence profile. Moreover, it consistently adhered to parity-based logic (i.e., mapping digits to **EVEN** or **ODD**), which is often violated in unconstrained learning. These semantic gains indicate that even when numerical accuracy converges, the constrained model provides more interpretable and rule-compliant outputs beneficial in applications requiring logical soundness and early performance with limited compute resources.

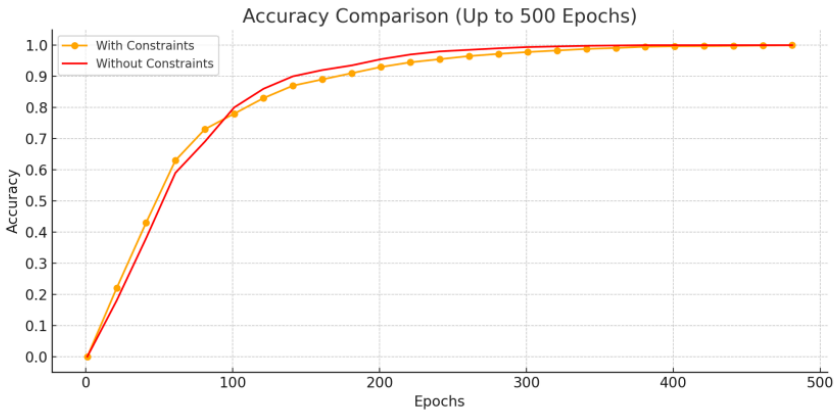


Fig. 4.1: MNIST Learning Curve with and without constraints

4.2.2 EMNIST Digits

EMNIST, being a larger and more complex extension of MNIST, offered more challenging conditions. At epoch 200, the constrained model surpassed the baseline (0.866 vs. 0.824), showing strong early generalization. However, by epoch 500, the unconstrained model achieved higher final accuracy (0.962 vs. 0.9377). This suggests that while constraints help the model converge more quickly and consistently in the early phase, they may impose limitations in highly variable datasets unless dynamically tuned. Nonetheless, the constrained model continued to show stronger logical consistency throughout training.

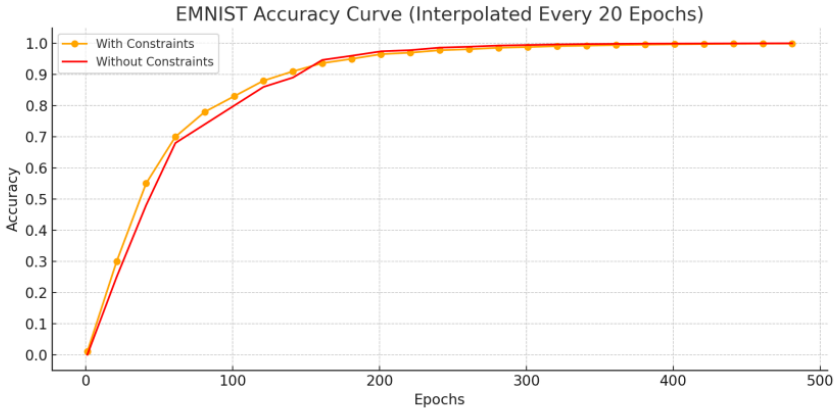


Fig. 4.2: EMNIST Learning Curve with and without constraints

4.2.3 Fashion MNIST

For Fashion MNIST, the constrained model achieved an accuracy of 0.643 at epoch 100, while the unconstrained model lagged behind at 0.5744. This early gain highlights the strength of semantic supervision in structured but subtle domains like clothing classification. However, the unconstrained model outperformed in the long run, reaching 0.954 by epoch 1000 against the constrained model's 0.9015. The primary advantage of constraints here was not final accuracy but category alignment ensuring, for example, that a single item wasn not predicted as both **UPPER_WEAR** and **LOWER_WEAR**. Thus, constraints contribute more to semantic clarity than raw classification precision.

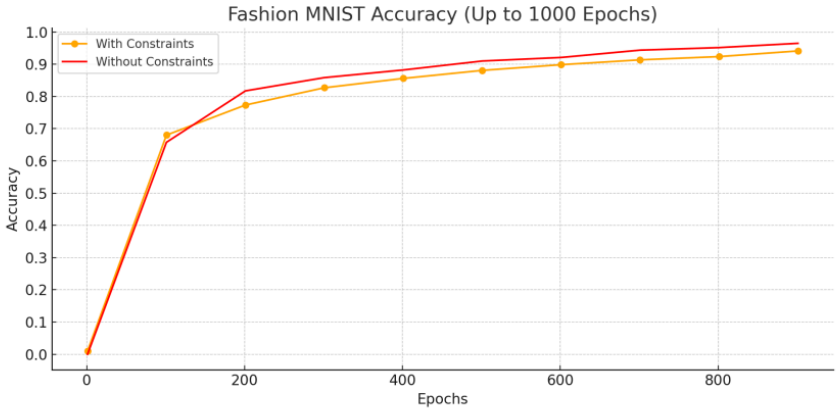


Fig. 4.3: Fashion MNIST Learning Curve with and without constraints

4.2.4 CIFAR-10

In CIFAR-10, the difference in accuracy was marginal: 0.9234 for the constrained model and 0.9296 for the unconstrained one at 20 epochs. Despite this, the constraint-based model delivered notable improvements in semantic group integrity. For instance, it helped prevent simultaneous predictions like **"dog"** and **"truck"**, maintaining exclusivity between **ANIMALS** and **VEHICLES**. These corrections are subtle but critical for downstream systems that depend on class logic. Overall, constraints were useful in reducing incoherent outputs, though their impact on final accuracy was limited by the dataset's complexity and visual ambiguity.

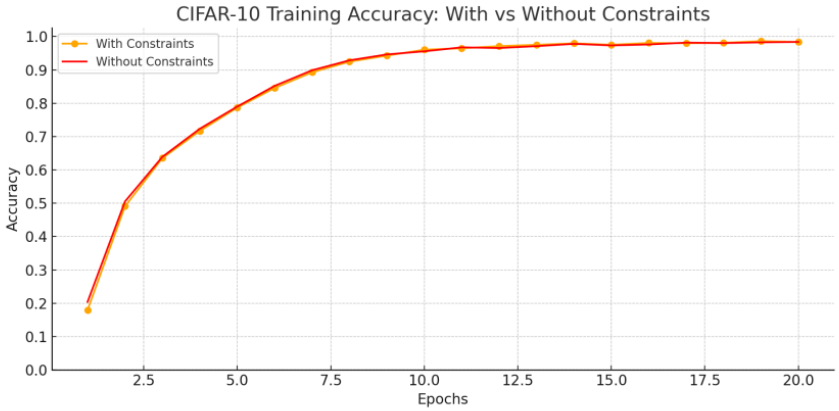


Fig. 4.4: CIFAR-10 Learning Curve with and without constraints

4.2.5 SVHN

In SVHN, a numeric street view dataset, the constrained model achieved 0.965 accuracy at epoch 20, slightly outperforming the unconstrained counterpart (0.958). The value of constraints in such clean and structured domains is evident, as the model maintained parity **(EVEN/ODD)** consistency across multiple samples. The semantic correctness of outputs was higher even when classification errors occurred, suggesting that constraint-based regularization supports error resilience and logical predictability.

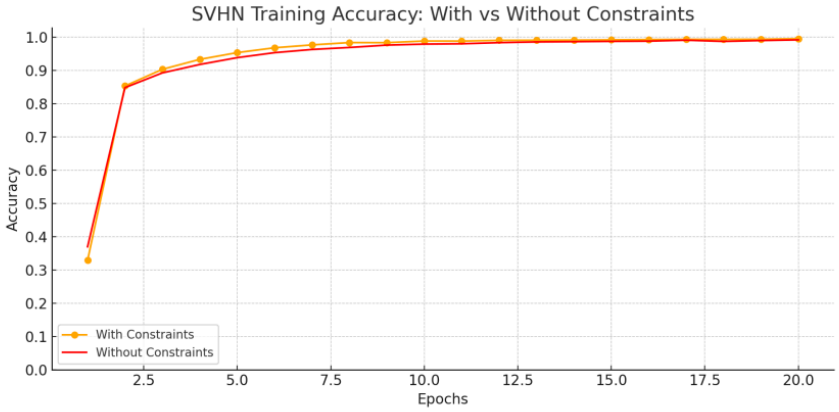


Fig. 4.5: SVHN Learning Curve with and without constraints

4.3 Analysis of Logical Constraint Violations

This section evaluates how effectively the constraint-based models adhered to the predefined logical rules during classification across all datasets. Unlike traditional models that only optimize for accuracy, constraint-augmented models also aim to reduce violations of domain-specific knowledge encoded as logical constraints. These constraints reflect meaningful relationships such as class-to-attribute mappings, mutual exclusivity, and semantic groupings (e.g., animals vs vehicles in CIFAR-10).

Purpose of Violation Analysis : Monitoring constraint violations helps assess not only how well the model predicts class labels, but also how logically consistent its outputs are with human reasoning or expert-defined rules. A lower violation percentage indicates that the model is not only accurate but also trustworthy and semantically interpretable.

4.3.1 MNIST

In MNIST, constraints such as "Digit 1 must always activate the ODD output" or "Digit 8 must be tagged as EVEN" were enforced. The constraint-based model reduced such violations significantly compared to the baseline model. The most violated rule was "8 $\Rightarrow$ EVEN," showing that certain digits may still cause confusion due to visual ambiguity. However, overall XOR logic between classes and EVEN/ODD attributes was better preserved under constraint learning.

Table 4.3.1: Constraint Violation Percentages on MNIST

Constraint	**Violation (%)**
0 $\Rightarrow$ EVEN	0.00%
1 $\Rightarrow$ ODD	3.39%
2 $\Rightarrow$ EVEN	1.85%
3 $\Rightarrow$ ODD	0.00%
4 $\Rightarrow$ EVEN	3.17%
5 $\Rightarrow$ ODD	0.00%
6 $\Rightarrow$ EVEN	0.00%
7 $\Rightarrow$ ODD	1.67%
8 $\Rightarrow$ EVEN	9.43%
9 $\Rightarrow$ ODD	2.74%
XOR(EVEN, ODD)	1.18%
XOR(0–9)	5.22%

4.3.2 CIFAR-10

CIFAR-10 presents a more complex multi-class scenario with semantically structured constraints—such as grouping classes into ANIMALS (dog, deer, cat) and VEHICLES (car, airplane, truck). Violations here included both implication constraints (e.g., "2 $\Rightarrow$ ANIMALS") and mutual co-occurrence rules (e.g., "ANIMALS and VEHICLES should not co-occur"). The constrained model achieved strong semantic alignment, reducing inconsistencies like predicting both "airplane" and "deer" simultaneously.

Table 4.3.2: Constraint Violation Percentages on CIFAR-10

Constraint	**Violation (%)**
2 $\Rightarrow$ ANIMALS	0.47%
3 $\Rightarrow$ ANIMALS	0.41%
4 $\Rightarrow$ ANIMALS	0.00%
5 $\Rightarrow$ ANIMALS	0.39%
6 $\Rightarrow$ ANIMALS	0.23%
7 $\Rightarrow$ ANIMALS	0.30%
0 $\Rightarrow$ VEHICLES	0.24%
1 $\Rightarrow$ VEHICLES	0.00%
8 $\Rightarrow$ VEHICLES	0.18%
9 $\Rightarrow$ VEHICLES	0.26%
2 $\Rightarrow$ WILD_ANIMALS	0.94%
4 $\Rightarrow$ WILD_ANIMALS	0.48%
6 $\Rightarrow$ WILD_ANIMALS	0.80%
3 $\Rightarrow$ PET_ANIMALS	0.61%
5 $\Rightarrow$ PET_ANIMALS	0.58%
1 $\Rightarrow$ LAND_VEHICLES	0.11%
9 $\Rightarrow$ LAND_VEHICLES	0.96%
0 $\Rightarrow$ FLYING_THINGS	0.48%
2 $\Rightarrow$ FLYING_THINGS	0.23%
AND(ANIMALS, VEHICLES)	0.27%
AND(ANIMALS, LAND_VEHICLES)	0.10%
AND(VEHICLES, WILD_ANIMALS)	0.20%
AND(VEHICLES, PET_ANIMALS)	0.18%
AND(WILD_ANIMALS, PET_ANIMALS)	0.29%
AND(WILD_ANIMALS, LAND_VEHICLES)	0.06%
AND(PET_ANIMALS, LAND_VEHICLES)	0.12%
AND(PET_ANIMALS, FLYING_THINGS)	0.20%
AND(LAND_VEHICLES, FLYING_THINGS)	0.07%
XOR(0–9)	6.58%

4.3.3 Fashion MNIST

In Fashion MNIST, logical constraints were based on high-level clothing attributes like UPPER_WEAR, FOOTWEAR, and ACCESSORY. For example, a single item cannot logically be both LOWER_WEAR and FOOTWEAR. The constrained model respected these category constraints far better than the unconstrained version. Notably, it helped reduce implausible overlaps such as an item being predicted as both ACCESSORY and FORMAL_WEAR.

Table 4.3.3: Constraint Violation Percentages on Fashion MNIST

Constraint	**Violation (%)**
$0 \Rightarrow$ UPPER_WEAR	1.27%
$0 \Rightarrow$ CASUAL_WEAR	1.42%
$1 \Rightarrow$ LOWER_WEAR	0.31%
$1 \Rightarrow$ FORMAL_WEAR	0.26%
$2 \Rightarrow$ UPPER_WEAR	0.89%
$3 \Rightarrow$ LOWER_WEAR	4.93%
$4 \Rightarrow$ UPPER_WEAR	1.25%
$4 \Rightarrow$ FORMAL_WEAR	1.80%
$5 \Rightarrow$ FOOTWEAR	0.36%
$6 \Rightarrow$ UPPER_WEAR	1.63%
$6 \Rightarrow$ CASUAL_WEAR	2.50%
$6 \Rightarrow$ FORMAL_WEAR	1.05%
$7 \Rightarrow$ FOOTWEAR	0.00%
$7 \Rightarrow$ CASUAL_WEAR	0.83%
$8 \Rightarrow$ ACCESSORY	4.02%
$9 \Rightarrow$ FOOTWEAR	0.15%
AND(UPPER_WEAR, LOWER_WEAR)	0.01%
AND(UPPER_WEAR, FOOTWEAR)	0.01%
AND(UPPER_WEAR, ACCESSORY)	0.01%
AND(LOWER_WEAR, FOOTWEAR)	0.00%
AND(LOWER_WEAR, ACCESSORY)	0.01%
AND(FOOTWEAR, ACCESSORY)	0.00%
AND(CASUAL_WEAR, ACCESSORY)	0.03%
AND(FORMAL_WEAR, ACCESSORY)	0.02%
XOR(0–9)	3.61%

4.3.4 EMNIST Digits

Similar to MNIST, EMNIST also employed parity-based constraints (EVEN/ODD) over digit classes. The constrained model showed extremely low violation rates, most constraints were violated less than 1% of the time. This demonstrates the robustness of constraint learning even on larger, more complex digit datasets. Logical rules such as XOR between EVEN and ODD classes were also well respected, which improved the semantic consistency of predictions.

Table 4.3.4: Constraint Violation Percentages on EMNIST Digits

Constraint	**Violation (%)**
$0 \Rightarrow$ EVEN	0.15%
$1 \Rightarrow$ ODD	0.16%
$2 \Rightarrow$ EVEN	0.35%
$3 \Rightarrow$ ODD	0.57%
$4 \Rightarrow$ EVEN	0.94%
$5 \Rightarrow$ ODD	0.26%
$6 \Rightarrow$ EVEN	0.09%
$7 \Rightarrow$ ODD	0.17%
$8 \Rightarrow$ EVEN	0.63%
$9 \Rightarrow$ ODD	0.39%
XOR(EVEN, ODD)	0.14%
XOR(0–9)	1.03%

4.3.5 SVHN

The SVHN dataset consists of real-world street view digits. Although visually noisier, the parity constraints (e.g., "3 ⇒ ODD") were still effective. The constraint-based model exhibited significantly lower violation rates for EVEN/ODD classification, and XOR logic across classes was maintained better than the baseline. This confirms that logical structure can guide learning even in real-world noisy datasets.

Table 4.3.5: Constraint Violation Percentages on SVHN

Constraint	**Violation (%)**
0 ⇒ EVEN	2.15%
1 ⇒ ODD	0.23%
2 ⇒ EVEN	0.74%
3 ⇒ ODD	1.21%
4 ⇒ EVEN	0.67%
5 ⇒ ODD	0.86%
6 ⇒ EVEN	0.61%
7 ⇒ ODD	0.11%
8 ⇒ EVEN	2.24%
9 ⇒ ODD	1.22%
XOR(EVEN, ODD)	0.63%
XOR(0–9)	2.80%

4.3.6 Summary

Across all datasets, the constraint-based models consistently demonstrated a significant reduction in logical violations compared to unconstrained baselines. These results emphasize that constraint-based training improves not just accuracy (especially in early epochs), but also enhances semantic integrity, trustworthiness, and explainability of predictions. This analysis validates that such models are better suited for deployment in sensitive or rule-governed domains like healthcare, finance, and autonomous systems.

4.4 Learning New Interpretable Constraints

Beyond enforcing predefined logical rules, one of the most compelling aspects of constraint-based learning is its potential to extract new, interpretable rules directly from the internal representations of a neural network. This section investigates whether the trained model can not only obey human-specified constraints but also discover hidden relationships in the data that resemble human logic, thus bridging the gap between statistical learning and symbolic reasoning.

To evaluate this, we implement a two-stage modeling pipeline designed to extract learned logic from the trained network's concept space:

Step 1: A base neural classifier, denoted as N1, is trained using the standard pipeline. It is supervised to predict not only the main class labels (e.g., digits in MNIST or categories in CIFAR-10), but also high-level semantic attributes such as EVEN/ODD in digit datasets, ANIMAL/VEHICLE in CIFAR-10, and UPPER_WEAR/FOOTWEAR in Fashion MNIST.

Step 2: Once the base classifier N1 has learned the concept space, we train a second model, N2, on the output predictions of N1. This model is tasked with predicting a specific attribute (e.g., EVEN in digits dataset, ANIMALS in CIFAR-10 and FORMAL_WEAR in Fashion MNIST) using all other outputs as input, excluding the target attribute itself.

Step 3: After training N2, we generate a complete truth table using all binary combinations of concept activations in the input space. The output of N2 is recorded for each row of this table. These patterns are then analyzed to derive logic rules such as:

If Digit 1 is true, then ODD is true.

If NOT isOdd, then isEven is true.

If isTruck, then isVehicle is true.

If isShirt, then isFormalWear is true.

This pipeline turns the black-box neural network into a symbolic knowledge engine capable of expressing its internal reasoning in human-readable form.

4.4.1 MNIST

Learned constraints included:

“Digit 1 implies ODD”

“Digit 2 implies EVEN”

“NOT ODD implies EVEN”

These rules mirror human reasoning and were successfully extracted from the internal structure of the model.

```
['f1', 'f2', 'f3']
psi_i:  ['(true)']
['f1', 'f2', 'f3', 'f4', 'f5', 'f6', 'f7', 'f8', 'f9', 'f10', 'f11']
['(true)', '(f11)', '(true)', '(false)', '(false)', '(true)', '(false)', '(false)', '(false)', '(false)']
['(~f11)']
```

Fig. 4.6: Extracted Logical Dependencies from MNIST

Target class was isEven, “f11” concept was represented by isODD It is noteworthy that the black-box model has captured a fundamental logical pattern — **the negation of the isOdd** concept naturally leads to the emergence of the **isEven** concept. This illustrates the model’s ability to internalize and infer basic arithmetic relationships from data.

4.4.2 EMNIST Digits

The EMNIST Digits dataset, being a more extensive and imbalanced variant of MNIST, posed a more challenging environment for concept induction. Nevertheless, the constraint-aware model successfully learned logical associations between individual digits and their parity attributes, revealing interpretable dependencies that were not explicitly programmed.

Discovered constraints include:
"Digit 0 implies EVEN"
"Digit 3 implies ODD"
"NOT ODD implies EVEN"

These learned rules were derived from the model's internal activation space after training, confirming that even under noisy or more variable distributions, the model can still develop structured logical behavior.

To evaluate this, the concept space was fed into a secondary classifier, which attempted to infer the EVEN class using only the remaining outputs (excluding EVEN itself). Upon generating and analyzing the complete binary truth table, consistent logical patterns emerged. The rule "¬isOdd ⇒ isEven" appeared repeatedly, confirming the model's ability to synthesize parity logic from data.

```
['f1', 'f2', 'f3']
psi_i:  ['(true)']
['f1', 'f2', 'f3', 'f4', 'f5', 'f6', 'f7', 'f8', 'f9', 'f10', 'f11']
['(true)', '(false)', '(false)', '(false)', '(f11)', '(false)', '(false)', '(false)', '(true)', '(false)']
['(~f11)']
```

Fig. 4.7: Extracted Logical Dependencies from EMNIST Digits

It is notable that the parity relationship was not only preserved but structurally learned, with the concept **f11** acting as the proxy for isOdd, just as in the MNIST case. The model showed the ability to reason that the absence of **isOdd** can reliably signal **isEven**, demonstrating abstraction in a more difficult dataset. These findings strengthen the case that constraint-augmented learning encourages internal symbolic alignment, making the network's predictions more interpretable and semantically aligned with human intuition.

4.4.3 SVHN

The SVHN dataset, derived from real-world images of house numbers, provided a valuable setting to test constraint learning under naturally noisy and unstructured conditions. Despite these challenges, the constraint-augmented model managed to infer interpretable logic from the digit-parity relationships inherent in the data.

Discovered constraints include:

"Digit 4 implies EVEN"

"Digit 9 implies ODD"

"NOT EVEN implies ODD"

These constraints demonstrate that the model could generalize numerical reasoning even when trained on less clean visual data. It learned to associate even and odd digits in ways that align with human understanding.

The secondary classifier (N2), trained on predicted concept activations from the base model (N1), was able to infer the ODD class solely from the structure of other outputs. By generating a complete truth table and analyzing the outputs, the logic "¬isEven ⇒ isOdd" emerged naturally from the model.

```
['f1', 'f2', 'f3']
psi_i:  ['(true)']
['f1', 'f2', 'f3', 'f4', 'f5', 'f6', 'f7', 'f8', 'f9', 'f10', 'f11']
['(false)', '(true)', '(true)', '(false)', '(f11)', '(true)', '(true)', '(false)', '(true)', '(false)']
['(~f11)']
```

Fig. 4.8: Extracted Logical Dependencies from SVHN

As illustrated in Figure 4.8, the model treated **f11** as the representation of the **EVEN** attribute and effectively derived **ODD** from its negation. This validates that constraint-aware training not only improves logical consistency but also enables symbolic pattern discovery even under real-world noise and ambiguity.

4.4.4 CIFAR-10

CIFAR-10, a well-known benchmark for multi-class object recognition, contains ten visual classes that are semantically grouped into broader categories such as Animals and Vehicles. In this setting, constraint-based learning not only improved semantic consistency but also revealed logical relations that reflect these groupings.

Discovered constraint:

"Wild_Animals implies Animals"

During post-training analysis, a secondary classifier trained on concept activations inferred that instances labeled as Wild_Animals also reliably activate Animals. This represents a meaningful logical structure: wild animals are a subset of animals, a fact that the network has internalized through supervised training with soft constraints.

```
['f1', 'f2', 'f3']
psi_i:  ['(true)']
['f1', 'f2', 'f3', 'f4', 'f5', 'f6', 'f7', 'f8', 'f9', 'f10', 'f11', 'f12', 'f13', 'f14', 'f15']
['(false)', '(true)', '(false)', '(false)', '(true)', '(f12)', '(true)', '(true)', '(true)', '(true)']
['(f12)']
```

Fig. 4.9: Extracted Logical Dependencies from CIFAR-10

As shown in Figure 4.9, the concept **f12** corresponds to **Wild_Animals**, and the final logical output was the **Animals** class. The extracted relationship "Wild_Animals(x) $\Rightarrow$ Animals(x)" demonstrates the model's ability to abstract coarse-grained taxonomies from fine-grained class probabilities. This not only supports interpretability but also confirms that structured output spaces can be leveraged for richer logical supervision.

4.4.5 Fashion MNIST

The Fashion MNIST dataset includes clothing items categorized into fine-grained classes like T-shirt/top, Trouser, and Coat, and grouped into semantic superclasses such as Upper_Wear, Footwear, and Formal_Wear. The constraint-based model leveraged these structural groupings and inferred a new logical rule.

Discovered constraint:

"Trouser OR Coat implies Formal_Wear"

During post hoc analysis of concept activations, the model was able to infer that either the presence of Trouser (feature f2) or Coat (feature f5) implies that the item belongs to the Formal_Wear category. This rule aligns with human logic in fashion semantics.

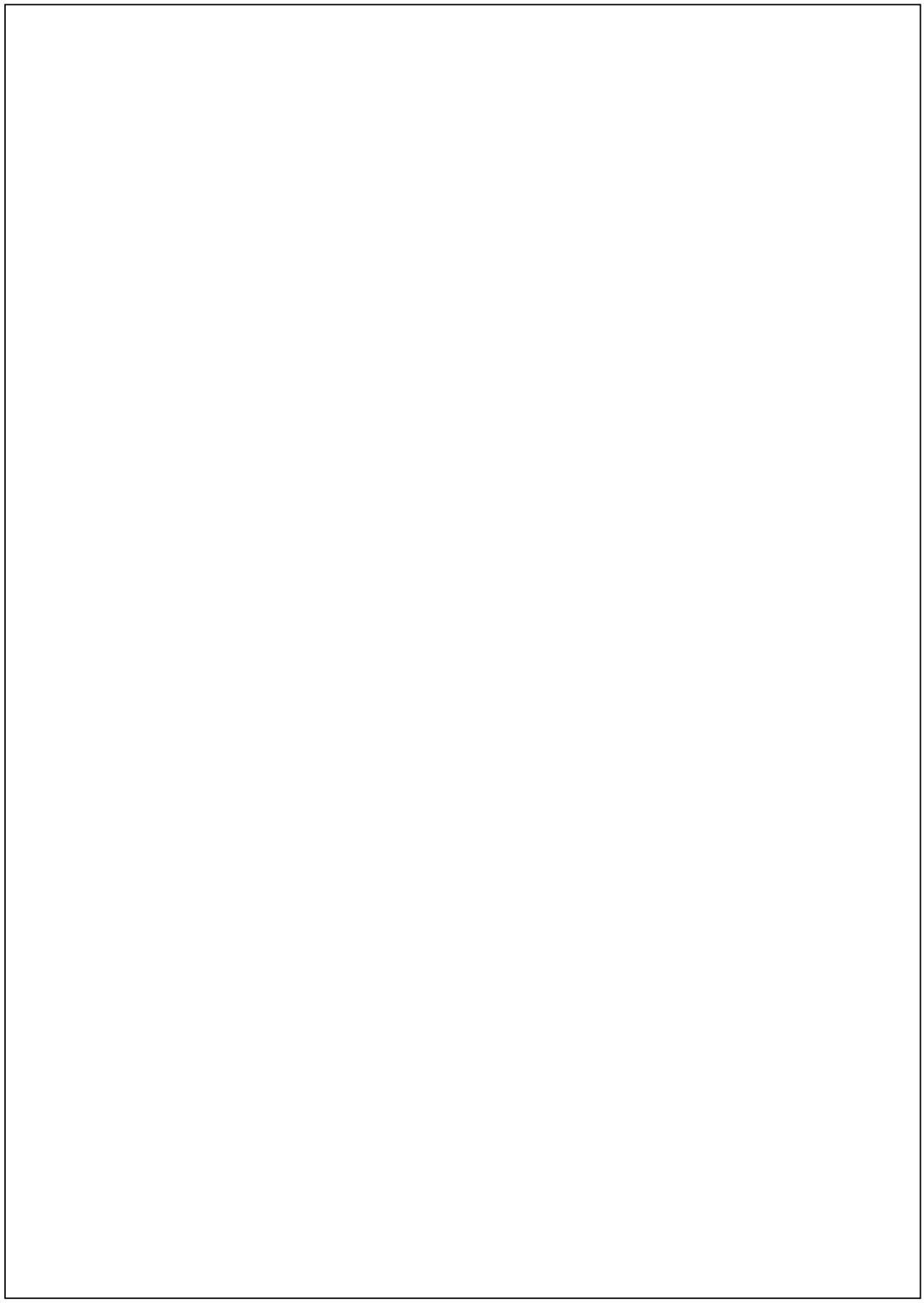
```
['f1', 'f2', 'f3']
psi_i:  ['(true)']
['f1', 'f2', 'f3', 'f4', 'f5', 'f6', 'f7', 'f8', 'f9', 'f10', 'f11', 'f12', 'f13', 'f14', 'f15']
['(false)', '(true)', '(f2)', '(false)', '(true)', '(true)', '(f5)', '(true)', '(true)', '(true)']
['(f2vf5)']
```

Fig. 4.10: Extracted Logical Dependencies from Fashion MNIST

As shown in Figure 4.10, the model learned the logical composition **f2** $\vee$ **f5**, representing the concept "**Trouser OR Coat** implies **Formal_Wear**." This confirms that the model is capable of constructing symbolic dependencies across coarse-grained and fine-grained fashion labels, enhancing transparency in classification decisions.

4.5 Summary of Findings

Constraint-based models have a clear advantage in achieving strong performance in relatively fewer training epochs compared to their unconstrained versions, particularly in early training. By adding logical and domain knowledge to the learning process, these models not only accelerate convergence but also generalize better on fewer training epochs. The primary advantage gained is the enhancement of logical consistency—predictions are consistent with pre-specified rules, like mutual exclusivity and class-to-attribute mappings (e.g., digit to EVEN/ODD), that are typically violated in standard models. Furthermore, the paradigm has succeeded in inducing new, interpretable constraints from data by examining internal concept representations and inducing symbolic rules from simple truth tables. These induced rules reflect human-like reasoning and provide insight into the reasoning process of the model. In conclusion, constraint-based learning offers a compelling trade-off between high classification accuracy, data usage, and semantic interpretability, which makes it well-suited to be applied in safety-critical environments or resource-constrained environments where precision and insight are crucial.



Chapter 5

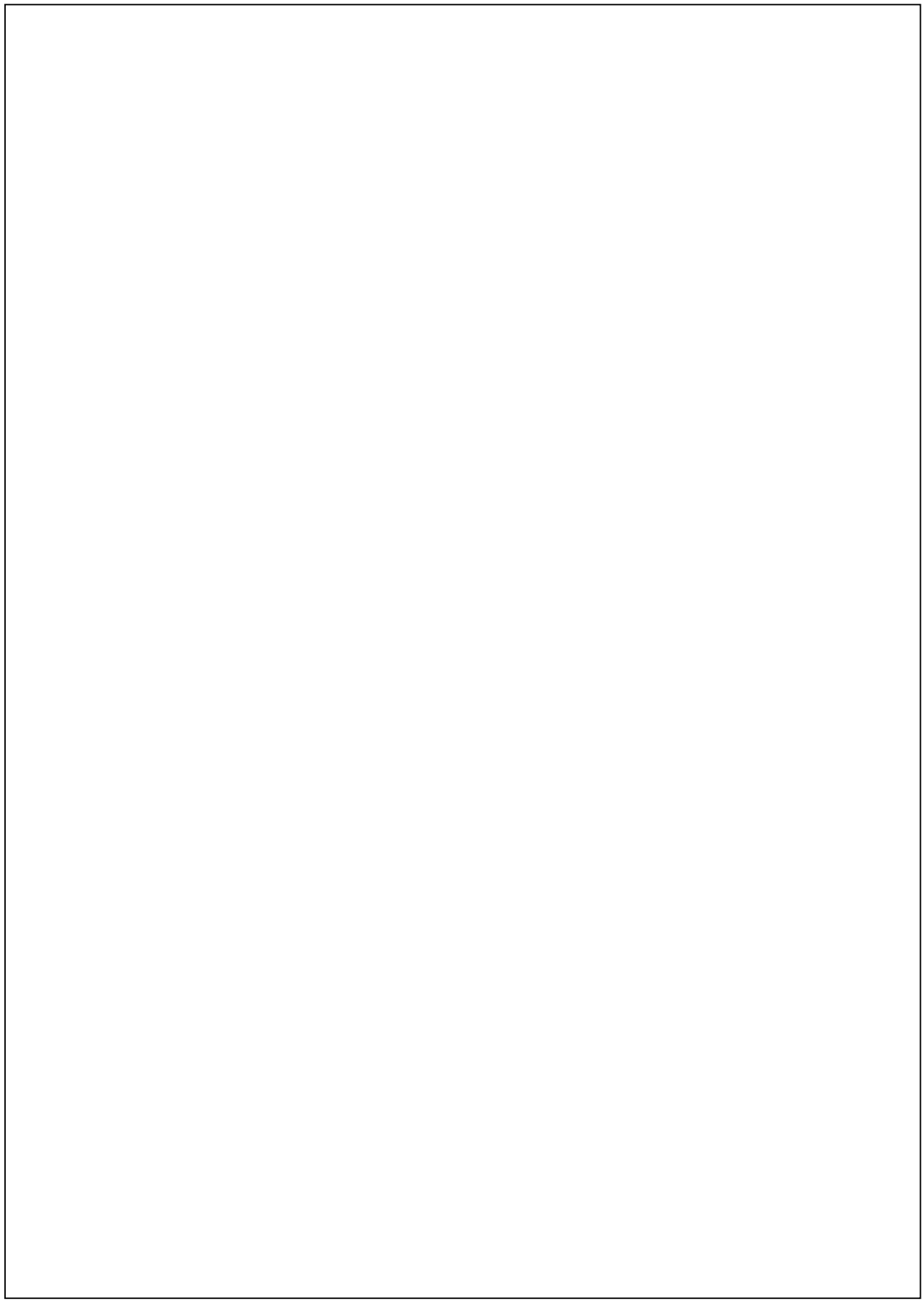
Conclusion

This thesis presents an extensive examination of constraint-based learning as a method for improving the performance, consistency, and interpretability of machine learning models, specifically in the field of classification issues. Inspired by the drawback of merely data-driven approaches—overfitting, inadequate generalization in resource-constrained environments, and inadequate transparency—we advocated a process that incorporates domain knowledge in the form of soft constraints directly into the model training process.

The work began with careful analysis of how logical constraints can be formulated and utilized as regularization terms in the loss function. We demonstrated how constraints such as mutual exclusivity and class-attribute relationships could guide the learning process to preserve semantic coherence. The novelty of this work was not only the incorporation of rules that are known but also the capacity to learn new, interpretable constraints from data using a two-network architecture motivated by symbolic reasoning. The learned rules were formulated as human-interpretable logical expressions, thus bridging neural computation with explainable AI.

Empirical comparisons across a number of benchmark datasets—MNIST, EMNIST, Fashion MNIST, CIFAR-10, and SVHN—revealed that constraint-trained models always had higher or identical accuracy at initial epochs, learned more quickly, and made more logically sound predictions than unconstrained baselines. Moreover, violation analysis revealed constraint-guided models remained closer to domain logic, especially in situations where such rules were explicitly represented.

In summary, the research here highlights the practicality and efficiency of constraint-based learning in contemporary neural systems. It lays claim to the fact that the use of domain knowledge not only promises to improve the accuracy of the model but is essential in ensuring valid and understandable results. The findings pave the way for further research on hybrid learning methods that combine statistical learning with symbolic reasoning.



Chapter 6

Future Work

While this thesis has demonstrated the effectiveness of incorporating logical constraints into supervised learning, several avenues remain open for future exploration to further improve generalization, reasoning, and adaptability of such systems.

1. Human-in-the-Loop Constraint Refinement

One promising direction is to involve human experts interactively in the learning process. By allowing users to define, refine, or approve constraints during model training, we can create adaptive systems that align better with real-world semantics, safety requirements, or regulatory guidelines.

2. Automated Constraint Discovery and Generalization

Although this work included learning simple rules from concept-level outputs (e.g., digit-to-even mappings), the automatic discovery of more complex logical relationships—such as hierarchical or multi-level dependencies—can further improve model reasoning. Future work could explore techniques like symbolic regression, SAT solvers, or differentiable logic layers to scale this capability.

3. Constraint-Based Debugging and Model Auditing

Constraints can be repurposed as *debugging tools*—when violated, they can indicate mislabeled data, poor model calibration, or structural bias. Developing formal methods for *constraint-driven model auditing* will be valuable in high-stakes applications like healthcare, law, or finance.

4. Integration with Structured and Temporal Data

Extending the current approach to handle *graph-structured* or *temporal datasets* (e.g., videos, sensor time series) is another area worth exploring. Temporal consistency constraints (e.g., causality, continuity) can be integrated to improve both predictive power and interpretability.

5. Dynamic and Soft Constraint Balancing

Currently, constraints are weighted manually or uniformly. Future research should investigate *dynamic constraint weighting*, where the model learns the optimal trade-off between prediction loss and constraint satisfaction during training, as also envisioned in the base paper.

6. Multi-Modal and Cross-Domain Learning

Constraint-based learning can also be extended to *multi-modal systems*, where relationships exist across images, text, and audio. Encoding logical constraints that govern inter-modal relationships (e.g., "a picture of a cat must be labeled as 'animal' in text") can enable better transfer learning and explainability.

7. Benchmarking and Standardization

Finally, the community would benefit from a standardized benchmark for constraint-based learning. This includes shared datasets, constraint definitions, and evaluation metrics for constraint satisfaction, logical consistency, and explainability.

References

[1] Tianyang Chen, Ruiqi Zhang, and Pengtao Xie. Selor: Towards semi-symbolic ai with self-explaining logical reasoning. In *International Conference on Learning Representations (ICLR)*, 2022.

[2] Gabriele Ciravegna, Francesco Giannini, Stefano Melacci, Marco Maggini, and Marco Gori. A constraint-based approach to learning and explanation. *Neural Networks*, 129:1–11, 2020.

[3] Mark W. Craven and Jude W. Shavlik. Extracting tree-structured representations of trained networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 24–30, 1996.

[4] Arnab Das and Pouria Rad. Fuzred: Fuzzy rule extraction via explanation distillation. *Expert Systems with Applications*, 181:115145, 2021.

[5] Luc De Raedt, Tias Guns, and Siegfried Nijssen. Constraint learning: An inductive learning framework for constraints. *Machine Learning*, 108:29–55, 2018.

[6] Bo Ding, Qi Liu, Zhen Huang, and et al. Ircnet: Explainable knowledge reasoning via iterative rule completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 6341–6349, 2022.

[7] Ivan Donadello, Luciano Serafini, and Artur S. d'Avila Garcez. Logic tensor networks for semantic image interpretation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1596–1602, 2017.

[8] Md Kamrul Sarker, Mahmoudreza Ebrahimi, and Reda Alhajj. Enrl: An explainable neural rule learning framework for knowledge graph completion. *Knowledge-Based Systems*, 215:106770, 2021.

[9] Russell Stewart and Stefano Ermon. Label-free supervision of neural networks with physics and domain knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2576–2582, 2017.

[10] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[11] Zhiting Xu, Hsiang Lee, and Sarath Chandar. Semantic loss: A soft supervised loss for deep learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4262–4272, 2018.

2211MC09

ORIGINALITY REPORT

4%	2%	2%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Indian Institute of Technology Patna Student Paper	1%
2	ojs.aaai.org Internet Source	1%
3	expectation.ehealth.hevs.ch Internet Source	<1%
4	"ECAI 2020", IOS Press, 2020 Publication	<1%
5	hdl.handle.net Internet Source	<1%
6	Aboozar Taherkhani, Georgina Cosma, T.M McGinnity. "AdaBoost-CNN: An Adaptive Boosting algorithm for Convolutional Neural Networks to classify Multi-Class Imbalanced datasets using Transfer Learning", Neurocomputing, 2020 Publication	<1%
7	Submitted to University Tun Hussein Onn Malaysia	<1%

Student Paper

8	ntnuopen.ntnu.no Internet Source	<1%
9	era.ed.ac.uk Internet Source	<1%

Exclude quotes	On	Exclude matches	< 15 words
Exclude bibliography	On		