INCORPORATING PIECEWISE LINEAR FUNCTIONS WITH CONSTANT REGIONS IN BACKPROPAGATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ADNAN HARUN DOĞAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

JANUARY 2025

Approval of the thesis:

INCORPORATING PIECEWISE LINEAR FUNCTIONS WITH CONSTANT REGIONS IN BACKPROPAGATION

submitted by **ADNAN HARUN DOĞAN** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Naci Emre Altun
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Prof. Dr. Sinan Kalkan
Supervisor, **Computer Engineering, METU** _____

Assoc. Prof. Dr. Emre Akbaş
Co-supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Gökberk Cinbiş
Computer Engineering, METU _____

Prof. Dr. Sinan Kalkan
Computer Engineering, METU _____

Assoc. Prof. Dr. Hacer Yalım Keleş
Computer Engineering, Hacettepe University _____

Date:10.01.2025

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Adnan Harun Doğan

Signature :

ABSTRACT

INCORPORATING PIECEWISE LINEAR FUNCTIONS WITH CONSTANT REGIONS IN BACKPROPAGATION

Doğan, Adnan Harun
M.S., Department of Computer Engineering
Supervisor: Prof. Dr. Sinan Kalkan
Co-Supervisor: Assoc. Prof. Dr. Emre Akbaş

January 2025, 85 pages

Solving many fundamental problems, such as travelling salesman (TS), shortest path (SP), and graph matching (GM), requires the use of piecewise linear functions with constant regions (PFC). Although using such functions in deep neural networks (DNN) is promising, integrating a PFC into a DNN poses a significant challenge for gradient-based iterative optimizations. That is, traditional backpropagation methods struggle when encountering PFCs in DNN pipelines, leading to zero or undefined gradients that stall training. Although various heuristics-based gradient approximations exist, these approaches often remain task-specific and theoretically ungrounded.

This thesis addresses the need for a unified and theoretically sound methodology for gradient approximation through PFC layers. First, it provides a comprehensive review and comparative analysis of existing techniques, highlighting that, despite their diversity, most methods share a common underlying principle. Building on these insights, the thesis introduces the Generalized Update (GU) as a unifying framework capable of representing known approximations as special cases and inspiring the development of new variants. The thesis also integrates Optimal Transport (OT) theory

to replace non-differentiable label assignment in models like DEtection TRansformer (DETR), demonstrating how OT-based solutions can enhance tasks involving discrete decision-making.

Overall, the thesis empirically validates the Generalized Update method's effectiveness across multiple domains, including object detection, combinatorial optimization, and quantization. By closing the theoretical gap, offering a unified perspective, and validating the proposed approach in practice, this work provides a robust foundation for incorporating PFCs into DNN pipelines, ultimately broadening the scope and applicability of gradient-based optimization methods.

Keywords: Gradients of Piecewise Linear Functions with Constant Regions, Combinatorial Optimization, Optimal Transport, Error-driven Update, Identity Update, Bipartite Matching in Object Detection

ÖZ

SABİT BÖLGELİ PARÇALI DOĞRUSAL FONSİYONLARIN GERİ YAYILIM ALGORİTMASINA DAHİL EDİLMESİ

Doğan, Adnan Harun
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Sinan Kalkan
Ortak Tez Yöneticisi: Doç. Dr. Emre Akbaş

Ocak 2025, 85 sayfa

Birçok temel problemi çözmek —örneğin Gezgin Satıcı (TS), En Kısa Yol (SP) ve Çizge İzleme (GM)— için parçalı doğrusal sabit aralıklı fonksiyonların (PFC) kullanılması gerekir. Bu fonksiyonları derin sinir ağlarında (DNN) kullanmak umut vaat etse de, bu entegrasyon gradyan tabanlı yinelemeli optimizasyon için önemli bir zorluk oluşturur. Öyle ki, derin sinir ağlarında de kullanılan geleneksel geri yayılım[1] yöntemleri, bu fonksiyonlarla karşılaştıklarında, sıfır veya tanımsız türevin eğitimi durdurmasından dolayı, çalışmaz ve sorun yaşarlar. Çeşitli sezgisel gradyan benzetimleri geliştirilmiş olsa da, bu yaklaşımlar genellikle probleme özgü ve/veya kuramsal temelden yoksun kalmaktadır.

Bu tez, PFC katmanları üzerinden gradyan benzetimi için birleşik ve kuramsal açıdan sağlam bir metodoloji ihtiyacını ele almaktadır. İlk olarak, mevcut tekniklerin kapsamlı bir incelemesi ve karşılaştırmalı analizi sunularak, yöntemlerin çeşitliliğine rağmen ortak bir temel ilkeye dayandığı vurgulanmıştır. Bu analizden hareketle, bili-

[1] çev.: backpropagation

nen benzetimleri özel durumlar olarak temsil edebilen ve yeni varyantların geliştirilmesine ilham veren Genelleştirilmiş Güncelleme (Generalized Update - GU) yöntemi tanıtılmıştır. Ayrıca, DEtection TRansformer (DETR) gibi modellerde türevi alınamayan etiket atamasını değiştirmek amacıyla Optimal Taşıma (OT) kuramı entegre edilerek, OT tabanlı çözümlerin ayrık karar süreçlerini iyileştirebileceği gösterilmiştir.

Genel olarak, tez, Genelleştirilmiş Güncelleme yönteminin nesne tespiti, kombinatoryel optimizasyon ve kuantizasyon gibi alanlarda etkinliğini deneysel olarak doğrulamaktadır. Bu yaklaşım, bu fonksiyonların derin sinir ağlarına entegrasyonu için sağlam bir temel sunarak, gradyan tabanlı optimizasyon yöntemlerinin kapsamını ve uygulanabilirliğini genişletmektedir.

Anahtar Kelimeler: Parçalı Doğrusal Sabit Aralıklı Fonksiyonların Türevi, Kombinatöryel Optimizasyon, Optimal Taşıma, Hata-tabanlı Güncelleme, Özdeşlik Güncelleme, Nesne Tespitinde Müşterek Eşleşme

To those who embrace life's brevity and the enduring pursuit of craftsmanship, advancing both ourselves and the world.

ACKNOWLEDGMENTS

I express my deepest gratitude to my advisors, Professor Dr. Sinan Kalkan and Associate Professor Dr. Emre Akbas, for their invaluable guidance, continuous support, and insightful feedback throughout this research. Their expertise and encouragement have shaped this thesis and greatly enriched my academic journey.

I sincerely thank the DENGE Research Group and the METU ImageLab[2] communities for fostering a collaborative and intellectually stimulating environment. I am especially grateful to Dr. Kemal Oksuz for his indispensable assistance during critical junctures, to Feyza Yavuz for her unwavering support and camaraderie throughout the research process, and to Doğukan Araslı for his dedication and valuable contributions during our collaborative efforts.

I extend special thanks to METU/ROMER[3] and TÜBİTAK/TRUBA[4] for their generous provision of the computing infrastructure that made this work possible.

I owe my deepest gratitude to my family for their boundless love and encouragement, whose unwavering belief in me has been a wellspring of strength during the most challenging phases of this journey.

I am profoundly thankful to my friends for their encouragement, enriching conversations, and invaluable perspectives that have broadened my intellectual and personal horizons.

Lastly, I also extend special gratitude to my housemates, including our beloved cat, whose warmth, camaraderie, and support created a nurturing and harmonious home environment, making this journey more balanced and fulfilling.

[2] METU/CENG Image Processing and Pattern Recognition Laboratory
[3] Robotics and Artificial Intelligence Technologies Application and Research Center at METU
[4] TÜBİTAK ULAKBİM High Performance and Grid Computing Center

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS

ADMM	Alternating Direction Method of Multipliers
DETR	Detection Transformer
DINO	Detection Transformer with Improved Denoising
DNN	Deep Neural Networks
GM	Graph Matching
GU	Generalized Update
KKT	Karush-Kuhn-Tucker
KL	Kullback-Leibler
KR	Kantorovich-Rubinstein
LP	Linear Programming
MIP	Mixed Integer Programming
MK	Monge-Kantorovich
PFC	Piecewise Linear Function with Constant Regions
PPO	Proximal Policy Optimization
QP	Quadratic Programming
SPO	“Smart-Predict”, then Optimize
STE	Straight-Through Estimator
TRPO	Trust Region Policy Optimization
TS	Traveling Salesman

CHAPTER 1

INTRODUCTION

1.1 Motivation

This section discusses the importance of piecewise linear functions with constant regions (PFC), the challenges associated with their use in deep neural networks, and why we need gradient approximations.

1.1.1 Piecewise Linear Functions with Constant Regions in Deep Learning

Modern deep learning methods have achieved remarkable success across various domains, including computer vision [8, 9, 10], natural language processing [11, 12, 13], and beyond [14, 15]. The ability to train large, complex models end-to-end using gradient-based optimization is a key contributor to this success. Researchers increasingly utilize *piecewise linear functions with constant regions* (PFC) in their deep neural network (DNN) architectures. These functions often appear as step functions [16], threshold operations [17, 18], combinatorial modules [1, 19], rank-based transformations [20, 21, 22, 23], or discrete decision-making layers [24, 25]. However, standard gradient-based training assumes that all neural network components have an informative, i.e., non-zero and tractable, gradient.

PFCs are fundamental to operating several early and modern machine learning models. Among them, the thresholding function (Eq. 1.1) is one of the simplest and most

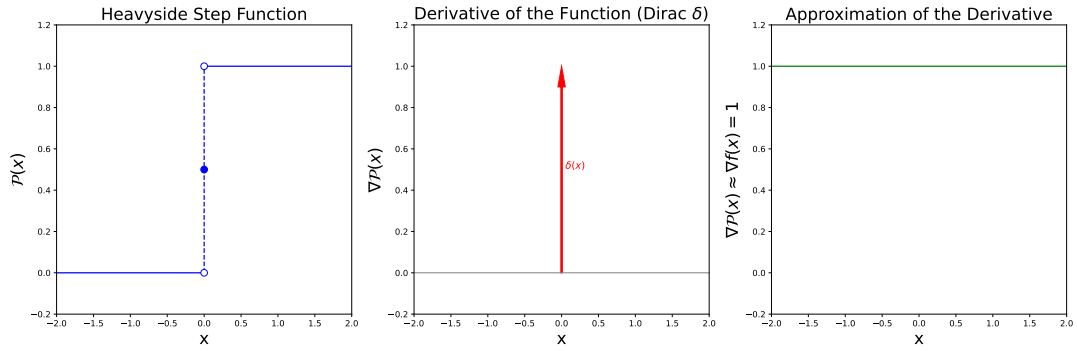

Figure 1.1: In the left figure, the Heaviside step function $\mathcal{P}(\mathbf{x})$ is shown with a discontinuity at $\mathbf{x} = 0$ indicated by open circles and a single filled circle. The middle figure draws the derivative of the Heaviside step function, represented by a Dirac delta $\delta(\mathbf{x})$. Lastly, the right figure displays the derivative of a surrogate function $f(\mathbf{x}) = \mathbf{x}$, which is simply the constant 1; this differentiable surrogate is introduced because its derivative can effectively replace the Dirac delta in practical applications.

widely used examples of PFCs. It is defined as:

$$\mathcal{P}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \geq 0, \\ 0 & \text{if } \mathbf{x} < 0. \end{cases} \tag{1.1}$$

Figure 1.1 demonstrates a typical discontinuity in PFCs and exemplifies their utility in transforming continuous inputs into discrete outputs for decision-making tasks.

PFCs enable discrete decision-making, efficient computations, and structured predictions within deep learning architectures. They model discrete phenomena such as thresholding and binary decisions, as seen in combinatorial layers, which support tasks like structured prediction and enforcing logical constraints [16, 26]. PFCs also enhance computational efficiency; for instance, binarized weights reduce memory usage and improve inference speed, particularly beneficial for resource-constrained applications [17, 18]. Furthermore, they allow for structured and interpretable outputs, enabling models to solve ranking problems, combinatorial optimizations, and other tasks requiring logical decision-making [19, 27, 28].

A pioneering example is the Perceptron model [16], introduced by Frank Rosenblatt in 1958. As one of the earliest neural networks, the Perceptron leverages the (Heaviside) step function as its activation function to make binary classification decisions.

This model is pivotal in understanding the role and challenges associated with PFCs in neural network architectures [16, 29].

Example 1.1 (Perceptron Model) *The Perceptron model is a foundational example of early neural networks [16]. It learns a linear decision boundary based on the input data and the associated weights and biases. The perception model performs binary classification by applying the Heaviside step function (Eq. 1.1) as an activation function. The output is computed as:*

$$\hat{\mathbf{y}} = \mathcal{P}(\theta^\top \boldsymbol{x} + \beta), \tag{1.2}$$

where the input data $\boldsymbol{x}$ and the associated weights θ and biases β.

As seen in Figure 1.1, the gradient of the Heaviside step function is zero almost everywhere. Therefore, standard gradient-based optimization cannot be applied directly to update the model parameters. This ineffective gradient forces the use of alternative strategies, such as the perceptron update rule (Eq. 3.12), which adjusts the weights based on the error between the predicted output and the desired output without relying on gradient computations. Such an approach allows the model to learn a linear decision boundary through iterative updates despite the inability to propagate gradients through the step function.

The *Average Precision* (AP) Loss [30] is a more recent example. The AP Loss optimizes the average precision metric directly by reformulating it as a ranking problem. In the forward computation, we calculate pairwise score differences for each pair of logits i and j, expressed as $x_{ij} = \mathbf{z}_j - \mathbf{z}_i$, where $\mathbf{z}_i$ and $\mathbf{z}_j$ represent the predicted scores for the respective logits. The goal is to ensure that positive logits are ranked higher than negatives regarding their confidence scores.

Example 1.2 (Average Precision Loss) *The AP Loss captures ranking violations by transforming labels into a pairwise representation. To measure ranking consistency, a Heaviside step function $\mathcal{P}(x_{ij})$ or its smoothed variant is applied to the score differences to calculate the primary terms:*

$$L_{ij} = \frac{\mathcal{P}(x_{ij})}{1 + \sum_{k \neq i} \mathcal{P}(x_{ik})}. \tag{1.3}$$

These terms prioritize logits with higher confidence scores while penalizing incorrect rankings. The overall AP Loss is defined as:

$$\mathcal{L}_{AP} = \frac{1}{|P|} \sum_{i \in P} \sum_{j \in N} L_{ij}, \tag{1.4}$$

where P and N represent the set of positive and negative logits, respectively, the term L_{ij} ensures that positive logits are weighted appropriately based on their ranks relative to negatives.

In the backward computation, the AP Loss employs an error-driven update scheme inspired by perceptron learning. Pairwise updates for the score differences are calculated as $\Delta x_{ij} = -L_{ij}$, which captures the degree of violation for each pair. These updates are then aggregated into gradients for individual logits:

$$\frac{\partial \mathcal{L}_{\text{AP}}}{\partial \mathbf{z}} = \frac{1}{|P|} \left(\sum_j \Delta x_{ij} - \sum_j \Delta x_{ji} \right). \tag{1.5}$$

This gradient accumulation ensures that the ranking adjustments are distributed across all pairs involving the given logit. Focusing on pairwise rankings directly aligns the optimization process with the evaluation metric, making it particularly effective for imbalanced datasets and detection tasks.

1.1.2 Monotonicity in Vector-Valued Functions

Even though both the Perceptron model (Example 1.1) and the AP Loss (Example 1.2) utilize different update rules to solve various problems, they demonstrate a form of *monotonicity* in their underlying PFCs. In the Perceptron, the Heaviside step function is non-decreasing: As its input increases, the function value never decreases. Similarly, the pairwise ranking mechanism in the AP Loss ensures that higher-scoring (i.e., larger) logits, i.e. $\mathbf{z}$, lead to more favourable terms in the loss, reflecting the same principle of preserving an input order.

Formally, a function $f : \mathbb{R} \to \mathbb{R}$ is said to be *monotonic* if an increase in its input does not decrease its output. In other words, if $x \leq y$, then $f(x) \leq f(y)$. This property simplifies analysis and proofs within optimization and learning theory because it provides predictable function behaviour as inputs vary [16, 26].

However, real-world problems, like multi-class decision-making or multi-dimensional structured prediction, require an extension of the monotonicity to the vector-valued functions, i.e., $\mathcal{P} : \mathbb{R}^n \to \mathbb{R}^n = \mathcal{P}_1 \times \cdots \times \mathcal{P}_n$. Therefore, a PFC, defined as $\mathcal{P} : \mathbb{R}^n \to \mathbb{R}^n$, is monotonically increasing if $x \leq y$, then $\mathcal{P}_i(x) \leq \mathcal{P}_i(y)$ for every $i = 1, 2, \ldots, n$ and $x, y \in \mathbb{R}$. Lastly, vice versa, for the monotone decreasing case.

1.2 Challenges of Using PFCs in Gradient-Based Optimization

Incorporating PFCs, such as the Heaviside step function in the Perceptron model and the AP Loss, into neural networks is foundational for enabling discrete decision-making, efficient computations, and structured predictions. However, once these PFCs are introduced into neural networks, their zero or undefined gradients render standard gradient-based optimization inapplicable. Consequently, it becomes necessary to propose alternative update rules that accommodate the discontinuities while preserving the task's inherent structure [15, 31, 32, 33, 34].

1.2.1 Task-Specific Solutions for PFC Gradients

Although techniques like the Straight-Through Estimator (STE) [31] partially address these issues by approximating gradients through non-differentiable components, they often lack formal guarantees and can limit model design flexibility [35, 36]. Consequently, discrete gating, combinatorial layers, and quantization steps remain significant bottlenecks to fully end-to-end training. Despite these limitations, PFCs are essential for tasks requiring discrete decisions, thresholding, and structured outputs, often found in combinatorial layers [16, 26].

Researchers have developed specialized methods to accommodate or bypass discontinuities introduced by PFCs. For instance, ranking and sorting operations rely on structured losses implemented via stochastic optimization [37], differentiable sorting [38], and top-k selection [39]. In structured prediction, approaches like CRFs [40], multiscale models [41], and constraint-based learning [1] accommodate interdependent outputs.

Similarly, combinatorial optimization and decision-focused learning embed discrete reasoning into neural architectures [24, 42, 43, 44, 45], with reinforcement learning [35, 46] and object detection [40, 47] serving as prominent applications. Furthermore, general-purpose frameworks for differentiable layers [28, 48, 19] enable robust regression [49, 50], probabilistic modeling [51, 52], and metric learning [32, 53], underscoring the adaptability of these techniques across diverse domains.

1.2.2 Research Gaps in Gradient Approximation Methods

Current research lacks a unified framework for gradient approximation. As a result, researchers and practitioners rely on isolated techniques that do not offer systematic guidance. A comprehensive approach is needed to address both theoretical and practical aspects of gradient approximation. Such a framework would provide robust solutions for integrating decision-making components, reduce the need for trial-and-error in model design, and allow seamless adaptation across different tasks. Establishing this foundation would unify existing methods and drive progress in learning and optimization pipelines.

1.3 Problem Definition and Scope of the Thesis

The central challenge of this thesis is to develop a robust and general-purpose framework for approximating gradients through PFC layers. Rather than treating existing gradient approximations as isolated solutions, this work aims to unify them under one theoretical perspective. This unified approach is expected to enable effective backpropagation through PFC layers, thereby expanding the range of architectures and tasks that can be handled using gradient-based optimization.

More specifically, this thesis consolidates existing gradient approximation methods into a single framework, termed the Generalized Update (GU) method. Additionally, the thesis incorporates insights from Optimal Transport (OT) theory [54, 55, 56, 57] and applies them to models such as the DEtection TRansformer (DETR) [5] to refine assignments in object detection tasks. This combined approach addresses the theoretical need for a unifying perspective and the practical requirement for methods that

handle discrete decision-making layers in continuous optimization pipelines. Empirical results across applications, including object detection, combinatorial optimization, and quantization tasks, are presented to validate the proposed methods.

1.4 Contributions

This thesis makes several key contributions to the understanding and practical handling of PFC layers in deep neural networks:

1. **Comprehensive Literature Review and Comparative Analysis**: A systematic review of existing gradient approximation methods is provided, highlighting their commonalities and differences. This analysis clarifies the underlying patterns in a fragmented research area.

2. **Introduction of the Generalized Update Method**: The thesis introduces a unified framework, the Generalized Update (GU). This framework generalizes existing approaches and recovers known approximations as special cases while offering a clear basis for developing new techniques.

3. **Theoretical Analysis and Connections to Established Principles**: The thesis connects the Generalized Update (GU) method to classical algorithms like the Perceptron and statistical techniques such as Maximum Likelihood Estimation (MLE). These connections deepen the theoretical understanding of gradient approximation.

4. **Empirical Validation in Diverse Applications**: The proposed framework is validated through experiments in object detection, combinatorial optimization, and quantization tasks. Results show that the unified approach improves performance and provides better guidance for model design in real-world applications.

1.5 Outline of the Thesis

Chapter 2 provides background on the role of PFC layers in deep learning and reviews existing gradient approximation methods. It also introduces the core concepts of Optimal Transport, explaining the need for a unified approach.

Chapter 3 presents the Generalized Update methodology in detail, including theoretical derivations, algorithmic implementations, and integration with standard deep learning frameworks. This chapter also features experiments that demonstrate the method's effectiveness.

Chapter 4 describes the experimental evaluation of the proposed methods. It covers three key applications: the Warcraft Shortest Path problem, a Conditional Random Field (CRF) task for semantic segmentation, and an object detection experiment with DETR. These experiments illustrate the strengths and limitations of the unified framework in various settings.

Chapter 5 summarizes the key findings, discusses the limitations, suggests directions for future research, and explains the broader impact of unifying gradient approximation methods in deep learning.

CHAPTER 2

RELATED WORK AND BACKGROUND

Deep learning has demonstrated exceptional success in image classification [58, 59], object detection [60, 61, 62, 63], segmentation [64, 65] tasks. Moreover, it has revolutionized natural language processing —for example, improving machine translation [11], sentiment analysis [66], and language modelling [13]— and has driven advancements in robotics, where techniques such as reinforcement learning enable autonomous navigation [67] and robotic manipulation [68]. This progress is primarily attributed to the end-to-end training framework enabled by backpropagation, which requires differentiability for propagating gradients through deep neural networks (DNNs).

As DNNs are applied to complex problems more and more, challenges arise when *piecewise linear functions with constant regions* (PFC), such as the Heaviside step function [17, 69], branch-and-bound algorithms [70], and SAT solvers [71], are incorporated into the training process. These discrete elements obstruct the direct use of gradient-based optimization. Various heuristic and principled methods have been proposed to approximate or circumvent gradient computations to address this. While techniques like the Straight-Through Estimator (STE) [69] have achieved notable empirical success, they often lack theoretical rigour and generality, resulting in diverse but fragmented solutions.

In parallel, the field has witnessed a growing interest in Optimal Transport (OT) theory [54, 55, 56, 57] as a tool for measuring and aligning probability distributions. Initially developed in mathematics and economics, OT has numerous applications in machine learning and computer vision. Its ability to handle structured, global assignments between sets has made it a candidate for addressing discrete matching steps

in complex pipelines, as in object detection using transformers [72], or differentiable ranking and sorting [73, 74].

2.1 Gradient-Based Optimization via PFCs

The integration of mathematical optimization into machine learning has enabled the development of methods for structured prediction, decision-focused tasks, ranking, and reinforcement learning. This section systematically explores methodologies, emphasizing their shared principles, innovative contributions, and application of gradient-based optimization techniques in a unified notation.

2.1.1 Differentiable Optimization Layers

Differentiable optimization layers embed constrained optimization problems as trainable components within neural networks, enabling structured predictions and decision-making. These layers integrate mathematical optimization into the neural network pipeline, allowing gradients to flow through optimization solvers and facilitating end-to-end learning.

Quadratic Optimization Amos and Kolter [48] introduced quadratic programming (QP) layers to solve optimization problems of the form:

$$\begin{aligned} \underset{\mathbf{s}}{\text{minimize}} \quad & \frac{1}{2}\mathbf{s}^\top Q(\mathbf{z})\mathbf{s} + c(\mathbf{z})^\top \mathbf{s}, \\ \text{subject to} \quad & A(\mathbf{z})\mathbf{s} = b(\mathbf{z}), \\ & G(\mathbf{z})\mathbf{s} \leq h(\mathbf{z}) \end{aligned} \tag{2.1}$$

where $Q(\mathbf{z})$ is positive semi-definite, and $Q(\mathbf{z}), c(\mathbf{z})$ are components of the objective function, $A(\mathbf{z}), b(\mathbf{z})$ are linear, and $G(\mathbf{z}), h(\mathbf{z})$ are affine. The solution $\mathbf{s}$ is computed by solving the problem using the Karush-Kuhn-Tucker (KKT) conditions. Gradients are derived by differentiating through the KKT system, which involves solving a large

linear system:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial Q} &= \frac{1}{2}\left(\frac{\partial \mathcal{L}}{\partial \mathbf{s}}\mathcal{P}\left(\mathbf{z}\right)_{\mathbf{s}}^{\top} + \mathcal{P}\left(\mathbf{z}\right)_{\mathbf{s}}\frac{\partial \mathcal{L}}{\partial \mathbf{s}}^{\top}\right) & \frac{\partial \mathcal{L}}{\partial q} &= -\frac{\partial \mathcal{L}}{\partial \mathbf{s}} \\
\frac{\partial \mathcal{L}}{\partial A} &= \frac{\partial \mathcal{L}}{\partial \nu}\mathcal{P}\left(\mathbf{z}\right)_{\mathbf{s}}^{\top} + \mathcal{P}\left(\mathbf{z}\right)_{\nu}\frac{\partial \mathcal{L}}{\partial \mathbf{s}}^{\top} & \frac{\partial \mathcal{L}}{\partial b} &= -\frac{\partial \mathcal{L}}{\partial \nu} \\
\frac{\partial \mathcal{L}}{\partial G} &= \operatorname{diag}(\mathcal{P}\left(\mathbf{z}\right)_{\lambda})\frac{\partial \mathcal{L}}{\partial \lambda}\mathcal{P}\left(\mathbf{z}\right)_{\mathbf{s}}^{\top} + \mathcal{P}\left(\mathbf{z}\right)_{\lambda}\frac{\partial \mathcal{L}}{\partial \mathbf{s}}^{\top} & \frac{\partial \mathcal{L}}{\partial h} &= \operatorname{diag}(\mathcal{P}\left(\mathbf{z}\right)_{\lambda})\frac{\partial \mathcal{L}}{\partial \lambda},
\end{aligned} \tag{2.2}$$

where $\frac{\partial \mathcal{L}}{\partial \nu}$ and $\frac{\partial \mathcal{L}}{\partial \lambda}$ represent the differentials (or Jacobians) of the loss $\mathcal{L}$ with respect to the dual equality multipliers ν, and the dual inequality multipliers λ, respectively. The terms $\mathcal{P}\left(\mathbf{z}\right)_{\mathbf{s}}$, $\mathcal{P}\left(\mathbf{z}\right)_{\nu}$, and $\mathcal{P}\left(\mathbf{z}\right)_{\lambda}$ are values computed during the forward pass by solving the quadratic optimization problem (2.1) to represent the optimal solutions for the primal and dual variables, ensuring that the KKT conditions are satisfied. This direct approach provides exact gradients and ensures numerical accuracy. However, it requires explicit inversion or factorization of the KKT matrix, which can be computationally expensive and memory-intensive for large-scale QP problems. OptNet's reliance on KKT-based solvers limits its scalability but makes it suitable for smaller optimization tasks requiring high precision.

Butler et al. [75] extended this work by introducing an Alternating Direction Method of Multipliers (ADMM)-based approach for QP layers. While solving the same optimization problem, the ADMM method decomposes the problem into simpler subproblems that are solved iteratively, avoiding direct manipulation of the KKT matrix. Gradients are computed by differentiating through the ADMM iterations rather than the KKT conditions, trading off exactness for computational efficiency. This iterative framework scales better to large and complex QP problems, particularly those encountered in large-scale machine learning and structured optimization tasks.

SATNet [71] approximates Boolean satisfiability by relaxing the problem into a quadratic programming (QP) form, as defined in OptNet [48]. The problem is solved using the same quadratic programming formulation and constraints as in OptNet. This relaxation enables smooth and differentiable optimization for inherently discrete and logical tasks.

Canonicalized Optimization Agrawal et al. [76] propose a differentiable framework for solving cone programming problems. The forward pass solves the opti-

mization problem formulated as:

$$\mathbf{s} = \underset{s}{\text{minimize}} \ c(\mathbf{z})^T \mathbf{s}, \quad \text{subject to } A(\mathbf{z})\mathbf{s} + w = b(\mathbf{z}), \ w \in \mathcal{K}, \tag{2.3}$$

where $\mathcal{K}$ is a convex cone, such as the nonnegative orthant[1], second-order cone, or positive semidefinite cone. The formulation does not include explicit inequality constraints but instead represents all constraints through cone membership.

The authors employ the Homogeneous Self-Dual Embedding (HSDE) to reformulate the problem into a residual map:

$$\Phi(\mathbf{s}, \lambda, \tau, \kappa) = H\mathbf{s} - r, \tag{2.4}$$

where H is a structured matrix, and λ, τ, κ are dual and scaling variables. Solving this system yields the primal and dual optimal solutions.

For the backward pass, gradients are computed by differentiating the HSDE residual map using the *implicit function theorem*. The Jacobian of the residual system is used to compute:

$$\frac{\partial \Phi}{\partial \mathbf{s}} \frac{\partial \mathbf{s}}{\partial \mathbf{z}} = -\frac{\partial \Phi}{\partial \mathbf{z}}. \tag{2.5}$$

The cone's structured nature and dual enables scalable differentiation, making this approach suitable for large-scale optimization tasks in machine learning and robust control.

Agrawal et al. [28] extend differentiable optimization layers to general convex problems under disciplined convex programming. The optimization problem is defined as:

$$\mathbf{s} = \underset{\mathbf{s}}{\text{minimize}} \ f_0(\mathbf{s}; \mathbf{z}), \quad \text{subject to } A(\mathbf{z})\mathbf{s} = b(\mathbf{z}), \quad f_i(\mathbf{s}; \mathbf{z}) \leq 0, \ i = 1, \ldots, m. \tag{2.6}$$

Here, $\mathbf{z}$ encapsulates the input parameters while solving the problem in an affine-solver-affine pipeline facilitates efficient computation through convex solvers.

The backward pass uses the Lagrangian $\mathbf{L}$, which incorporates equality and inequality constraints:

$$\mathbf{L}(\mathbf{s}, \lambda, \nu; \mathbf{z}) = f_0(\mathbf{s}; \mathbf{z}) + \lambda^T (A(\mathbf{z})\mathbf{s} - b(\mathbf{z})) + \nu^T f(\mathbf{s}; \mathbf{z}), \tag{2.7}$$

[1] A n-dimensional generalization of the terms quadrant (in two-dimensional Cartesian space) and octant (in three-dimensional space).

where λ and ν are dual variables for equality and inequality constraints, respectively. The Karush-Kuhn-Tucker (KKT) conditions are applied:

$$\nabla_{\mathbf{s}}\mathbf{L}(\mathbf{s}, \lambda, \nu; \mathbf{z}) = 0, \quad A(\mathbf{z})\mathbf{s} = b(\mathbf{z}), \quad f_i(\mathbf{s}; \mathbf{z}) \leq 0, \quad \nu_i \geq 0, \quad \nu_i f_i(\mathbf{s}; \mathbf{z}) = 0. \tag{2.8}$$

Integer Programming Integer programming (IP) layers enable neural networks to address structured decision-making tasks that involve discrete constraints. These methods extend the quadratic programming (QP) framework introduced by OptNet [48] to handle integer variables, leveraging relaxation and projection techniques to maintain differentiability while solving or learning from discrete optimization problems.

MIPaaL [24] approaches integer programming by first relaxing the integer constraints to continuous variables, solving the relaxed problem to find an optimal solution, and then projecting this solution back onto the feasible integer domain. This process is equivalent to finding a QP problem within the convex hull of the integer constraints that yields the same optimal solution as the original mixed-integer program. Gradients are computed by differentiating the Karush-Kuhn-Tucker (KKT) conditions of the relaxed problem, ensuring smooth backpropagation. MIPaaL applies a cutting-plane approach during the projection step to refine the solution iteratively, making it suitable for tasks like resource allocation and combinatorial reasoning.

CombOptNet [1] focuses on fitting the integer programming problem to the data rather than directly solving it. As shown in Figure 2.1 CombOptNet learns the cost terms and constraints of the integer program from data, enabling the model to adaptively encode the structure of the optimization problem without necessarily solving it. The forward pass involves solving a relaxed version of the integer program:

$$\mathbf{s} = \underset{\mathbf{s}\in\mathbb{Z}^n}{\text{minimize}} \; c(\mathbf{z})^\top \mathbf{s}, \quad \text{subject to } A(\mathbf{z})\mathbf{s} \leq b(\mathbf{z}). \tag{2.9}$$

Gradients are computed by relaxing the problem to a continuous domain and differentiating through the KKT conditions. After training, CombOptNet can be evaluated on the discrete problem, but its primary goal is learning the structure of the optimization problem rather than strictly solving it.

Gao et al. [77] introduce a framework for integrating combinatorial losses into neural network training by leveraging the concept of subdifferentials and generalized gradi-

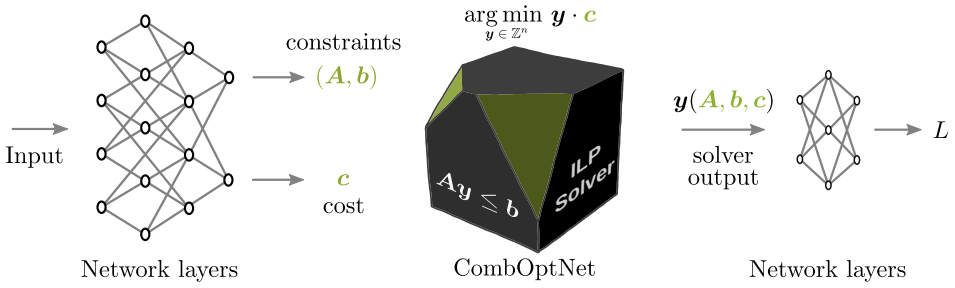

Figure 2.1: A depiction of the CombOptNet framework, integrating integer linear programming (ILP) solvers into neural networks. The network learns constraints (A, b) and cost vector c from input data, which are fed into the ILP solver to compute the discrete solution $y(A, b, c)$. The solver's output is integrated into the network layers for end-to-end optimization, enabling combinatorial reasoning within deep learning pipelines. [Figure taken from: [1]]

ents within linear programming (LP) formulations. The forward pass involves solving a combinatorial optimization problem, conceptually represented as a linear program:

$$s = \underset{\mathbf{s}}{\text{minimize}}\, c(\mathbf{z})^\top \mathbf{s}, \quad \text{subject to } A(\mathbf{z})\mathbf{s} = b(\mathbf{z}),\ \mathbf{s} \geq 0, \tag{2.10}$$

where c, A, and b are parameters derived from the neural network's output, and $\mathbf{s}$ is the optimal objective value. A black-box combinatorial solver computes $\mathbf{s}$ alongside primal and dual solutions.

For the backward pass, the gradients of the combinatorial loss are computed using subdifferentials, enabling efficient propagation through the non-differentiable optimization process. The subgradients of $z^\star$ with respect to c, A, and b are:

$$\frac{\partial z^\star}{\partial c} = u^\star, \quad \frac{\partial z^\star}{\partial b} = v^\star, \quad \frac{\partial z^\star}{\partial A} = -v^\star u^{\star\top}. \tag{2.11}$$

These subgradients are derived directly from the LP's primal and dual solutions, where $v^\star$ lies in the Clarke subdifferential[2] of $z^\star$. The total gradient of the loss function L with respect to the neural network parameters β is computed as:

$$\frac{\partial L}{\partial \beta} = \frac{\partial L}{\partial z^\star}\left(u^\star \frac{\partial c}{\partial \beta} - v^\star u^{\star\top}\frac{\partial A}{\partial \beta} + v^\star \frac{\partial b}{\partial \beta}\right). \tag{2.12}$$

[2] The subdifferential of a convex function $f : \mathbb{R}^n \to \mathbb{R}$ at a point x is the set denoted by $\partial f(x) = \{g \in \mathbb{R}^n \mid f(y) \geq f(x) + g^\top(y - x), \quad \forall y \in \mathbb{R}^n\}$. The subdifferential generalizes the gradient to non-differentiable points, capturing all possible slopes of supporting hyperplanes at x. That is $\partial f(x) = \text{conv}\left\{\lim_{x_k \to x} \nabla f(x_k)\right\}$.

2.1.2 Gradient Approximation Techniques

Gradient approximation techniques are pivotal in integrating PFC components—such as combinatorial solvers, binary decision modules, or ranking functions—into end-to-end trainable learning frameworks. These methods enable gradient-based optimization by approximating or bypassing gradients for piecewise linear operations.

Straight-Through Estimator (STE) [31] is a simple yet effective method to approximate gradients for discrete or non-differentiable functions. During the backward pass, STE approximates the gradient as:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}} \approx \frac{\partial \mathcal{L}}{\partial \mathbf{s}} \cdot \mathbf{1}_{\{0<\}}, \tag{2.13}$$

where $\mathbf{1}_{\{0<\}}$ represents the forward evaluation of the indicator function. This method is widely applied in Binarized Neural Networks [17], where weights are constrained to $\{-1, 1\}$. STE bypasses the need for continuous relaxations by approximating gradients with the identity function, making it computationally efficient for large-scale applications.

Finite Difference Gradients In tasks involving black box solvers, exact gradients may not be accessible. Pogancic et al. [19] approximates gradients using finite differences:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}} \approx \frac{\mathcal{L}(\mathbf{z} + \delta) - \mathcal{L}(\mathbf{z})}{\delta}, \tag{2.14}$$

where δ is a hyperparameter that determines the scale over which finite differences are computed. The greater the δ, the smoother the gradient approximation becomes; however, it strays from the local gradient and produces a gradient that deviates from the actual value. Conversely, the lower the δ, the more locally informative and sharper the gradient, capturing the local behaviour more accurately but behaving more unstable due to the sharpness.

Perturbation-Based Gradients Berthet et al. [44] introduces stochasticity to smooth, non-differentiable objectives. The problem is modified by adding a noise term:

$$\mathbf{s} = \underset{\mathbf{s}}{\text{minimize}} \;\; f(\mathbf{s}, \mathbf{z}) + \epsilon \cdot r(\mathbf{s}, \zeta), \tag{2.15}$$

where $r(\mathbf{s}, \zeta)$ is a noise function parameterized by ζ. The added perturbation allows gradients to flow through the optimization process, making the solver differentiable.

This method is particularly useful for blackbox optimization and structured prediction tasks.

Surrogate Loss Functions Surrogate losses provide a differentiable approximation of the original non-differentiable objective for optimisation-based modelling tasks. Domke et al. [78] propose designing surrogate loss functions that align gradient computations with downstream task objectives. By replacing the non-differentiable objective with a smooth approximation, surrogate losses enable backpropagation through complex decision-making pipelines.

Implicit Differentiation for Bi-Level Optimization

In bi-level optimization problems, the decision variable $\mathbf{s}$ is determined as the solution to a nested optimization problem:

$$\underset{\mathbf{s}}{\text{minimize}} \; g(\mathbf{s}; \mathbf{z}), \tag{2.16}$$

where $\mathbf{z}$ represents the input or parameter of the objective function g, and $\mathbf{s}$ is the decision variable.

Gould et al. [36] utilize the implicit function theorem to compute gradients of the solution $\mathbf{s}$ with respect to $\mathbf{z}$. The gradient is given by:

$$\frac{\partial \mathbf{s}}{\partial \mathbf{z}} = -\left(\frac{\partial^2 g}{\partial \mathbf{s}^2}\right)^{-1} \frac{\partial^2 g}{\partial \mathbf{z} \partial \mathbf{s}}, \tag{2.17}$$

where $\frac{\partial^2 g}{\partial \mathbf{s}^2}$ is the Hessian of g with respect to $\mathbf{s}$, $\frac{\partial^2 g}{\partial \mathbf{z} \partial \mathbf{s}}$ is the Jacobian of the gradient of g with respect to $\mathbf{z}$. This method ensures exact gradient computation by solving a linear system involving the Hessian $\frac{\partial^2 g}{\partial \mathbf{s}^2}$. It is particularly effective for applications in structured prediction and decision-focused learning, where the quality of the decision depends on accurately differentiating through the nested optimization problem.

2.1.3 Ranking-Based Methods

Ranking-based methods address the inherent non-differentiability of ranking and sorting objectives, enabling their integration into neural networks for object detection, retrieval, and structured prediction tasks. These methods focus on approximating

ranking and sorting operations with smooth, differentiable formulations or surrogate loss functions.

Differentiable Ranking Using Optimal Transport Cuturi et al. [27] formulates ranking as an optimal transport problem, where the goal is to find a doubly stochastic matrix P that minimizes a cost:

$$\mathbf{s} = \underset{\mathbf{s}\in U(a,b)}{\text{minimize}} \langle \mathbf{s}, C(\mathbf{z}) \rangle + \lambda H(\mathbf{s}), \tag{2.18}$$

where $U(a, b)$ is the set of doubly stochastic matrices ensuring row and column sums match a and b, $C(\mathbf{z})$ is a cost matrix encoding ranking preferences, $H(\mathbf{s})$ is an entropy regularization term promoting smoothness. This approach leverages Sinkhorn distances to ensure computational tractability, making ranking objectives differentiable and scalable for large datasets.

Permutohedron-Based Sorting Blondel et al. [38] approximates sorting operations by projecting onto the permutohedron[3], the convex hull of all permutations of a vector y:

$$\underset{x\in P_n}{\text{minimize}} \|x - y\|^2, \tag{2.19}$$

where P_n represents the permutohedron for n-dimensional vectors, y is the target ranking.

This method achieves $O(n \log n)$ complexity, making it efficient for large-scale ranking tasks. The permutohedron-based approach smooths the ranking operation, enabling gradients to propagate through sorting layers while maintaining computational efficiency.

Rank & Sort Loss The Rank & Sort (RS) Loss [21] introduces a ranking-based loss function tailored for visual detection tasks such as object detection and instance segmentation. The RS Loss incorporates *ranking positives above negatives* and *sorting positives among themselves* based on their localization qualities (e.g., IoU values).

The RS Loss combines a *ranking objective* to rank positives higher than negatives and a *sorting objective* to sort positives in descending order of their continuous IoU

[3] The permutohedron P_n can be constructed by acting the symmetric group S_n on the permutohedron by permutation of coordinates. In other words, for a n-dimensional vector v, the permutohedron P_v is the convex hull of all vectors obtained by permuting the coordinates of v, i.e. $P_v = \text{conv}\{(\pi(v_1), \ldots, \pi(v_n)) | \pi \in S_n\}$, where π ranges over all permutations in the permutation set S_n.

labels. This results in:

$$\mathcal{L}_{\text{RS}} = \frac{1}{|P|} \sum_{i \in P} \left[\underbrace{\frac{\text{NFP}(i)}{\text{rank}(i)}}_{\text{Ranking Error}} + \underbrace{\sum_{j \in P} \frac{H(x_{ij})(1 - y_j)}{\text{rank}^+(i)}}_{\text{Sorting Error}} \right], \tag{2.20}$$

where $H(x_{ij})$ is a smoothed unit step function, $\text{NFP}(i)$ is the number of false positives ranked higher than i, $\text{rank}(i)$ is the rank of i relative to all examples, $\text{rank}^+(i)$is the rank of i among positives, y_j: IoU of example j. The sorting error penalizes positives whose logits are higher than i but have lower IoU labels. Using the reformulated rule, the gradient of RS Loss w.r.t. logits s_i is expressed, for negatives ($i \in N$) as:

$$\frac{\partial \mathcal{L}_{\text{RS}}}{\partial s_i} = \frac{1}{|P|} \sum_{j \in P} \underbrace{\frac{\text{NFP}(j)}{\text{rank}(j)} \cdot p_R(i|j)}_{\text{Ranking Update}}, \tag{2.21}$$

where $p_R(i|j)$ distributes the ranking error over negatives. For positives ($i \in P$), the gradient includes both ranking and sorting updates:

$$\frac{\partial \mathcal{L}_{\text{RS}}}{\partial s_i} = \frac{1}{|P|} \left[(\ell^*_{\text{RS}}(i) - \ell_{\text{RS}}(i)) + \sum_{j \in P} \underbrace{(\ell_{\text{S}}(j) - \ell^*_{\text{S}}(j)) \cdot p_S(i|j)}_{\text{Sorting Update}} \right], \tag{2.22}$$

where $\ell^*_{\text{RS}}(i)$ and $\ell^*_{\text{S}}(j)$ are the target errors, $p_S(i|j)$ is the probability mass function for sorting. This formulation ensures that RS Loss optimizes ranking-based objectives while addressing the non-differentiability of ranking and sorting operations. It eliminates the need for auxiliary heads or sampling heuristics, making it robust to imbalanced datasets.

Blackbox Differentiation for Rank-Based Metrics Rolinek et al. [53] enables rank-based metrics such as Average Precision (AP) in end-to-end training. By approximating gradients for such metrics, it bridges the gap between evaluation criteria and training objectives, allowing neural networks to optimize directly for ranking performance.

Distributional and Top-k Ranking Losses Other ranking methods include various ranking loss functions such as DRLoss [47] and Average Top-k Loss [79]. In more detail, Qian et al. [47] optimizes distributional ranking objectives for object detection, focusing on balancing classification and localization quality. Additionally, Fan et

al. [79] Emphasizes the top-k elements during training, providing better alignment with ranking-based performance metrics.

MetricOpt MetricOpt [23] focuses on optimizing non-differentiable blackbox evaluation metrics (e.g., F1-score) in reinforcement learning. The forward pass involves computing the desired metric over a batch of predicted and actual outputs:

$$M(\hat{y}, y) = \text{Metric}(\hat{y}, y), \tag{2.23}$$

where $\hat{y}$ are predictions and y are ground truths.

Gradients with respect to the policy are approximated using finite differences or surrogate gradient methods. For example, using a differentiable surrogate $\tilde{M}(\hat{y}, y)$, the gradient is:

$$\nabla_\theta L(\theta) = \nabla_\theta \mathbb{E}[\tilde{M}(\hat{y}, y)], \tag{2.24}$$

where $\tilde{M}$ approximates the blackbox metric M.

2.1.4 Decision-Focused Learning

Smart Predict-then-Optimize (SPO) The Smart Predict-then-Optimize (SPO) framework [42] is a decision-focused learning approach that directly integrates predictive modelling with downstream optimization tasks. Instead of minimizing prediction error alone, SPO focuses on optimizing decision quality, ensuring the predictive model is trained concerning its impact on the optimization problem.

The framework predicts cost parameters $\mathbf{z} = \mathrm{F}_\theta(\mathbf{x})$, where $\mathbf{x}$ represents input features, and θ denotes the model parameters. These predicted costs are used in the following optimization problem:

$$\mathcal{P}(\mathbf{x}; \theta) = \underset{\mathbf{s} \in \mathcal{S}}{\text{minimize}}\ \mathbf{s}^\top \mathrm{F}_\theta(\mathbf{x}), \tag{2.25}$$

where $\mathcal{P}(\mathbf{x}; \theta)$ is the decision (i.e. predicted solution) based on the prediction $\mathbf{z} = \mathrm{F}_\theta(\mathbf{x})$, and $\mathcal{S}$ is the feasible region defined by problem constraints.

SPO introduces the SPO loss function, which measures the regret of decisions made using the predicted costs. The regret compares the cost of the predicted solution $\mathcal{P}(\mathbf{z})$

against the true optimal solution $\mathcal{P}(\mathbf{z}^\star)$:

$$\begin{aligned}\mathcal{L}_{\text{SPO}} &= \mathcal{P}(\mathbf{z})^\top \mathbf{z}^* - \xi(\mathbf{z}^*) = \left(\mathcal{P}(\mathbf{z}) - \mathcal{P}(\mathbf{z}^\star)\right)^\top \mathbf{z}^\star \\ &\leq \max_{\mathbf{s}\in\mathcal{P}(\mathbf{z})} \mathbf{s}^\top \mathbf{z}^* - \xi(\mathbf{z}^*) = \mathcal{L}_{\text{SPO+}}.\end{aligned} \tag{2.26}$$

where $\xi(\mathbf{z}) := \min_{\mathbf{s}\in\mathcal{S}} \mathbf{s}^\top \mathbf{z} = \mathcal{P}(\mathbf{z})^\top \mathbf{z}$ is the cost of the optimal solution. This loss penalizes the predictive model for solutions that lead to suboptimal decisions.

A challenge in using the SPO loss directly is the non-differentiability of the optimization operator with respect to c. To address this, the SPO+ framework introduces a convex surrogate loss that approximates the SPO loss while remaining computationally efficient and differentiable. The surrogate loss is defined as:

$$\begin{aligned}\mathcal{L}_{\text{SPO+}} &= \max_{s\in\mathcal{P}(\mathbf{z})} \left\{ \mathbf{s}^\top \mathbf{z}^* - \alpha\, \mathbf{s}^\top \mathbf{z} \right\} \;+\; \alpha\, \xi(\mathbf{z}) - \xi(\mathbf{z}^*) \\ &\leq \inf_{\alpha\in\mathbb{R}_+} \left\{ \max_{s\in\mathcal{P}(\mathbf{z})} \left(\mathbf{s}^\top \mathbf{z}^* - \alpha\, \mathbf{s}^\top \mathbf{z}\right) \;+\; \alpha\, \xi(\mathbf{z}) \right\} - \xi(\mathbf{z}^*). \\ \mathcal{L}_{\text{SPO+}} &= \lim_{\alpha\to\infty} \left\{ \max_{s\in\mathcal{P}(\mathbf{z})} \left(\mathbf{s}^\top \mathbf{z}^* - \alpha\, \mathbf{s}^\top \mathbf{z}\right) \;+\; \alpha\, \xi(\mathbf{z}) \right\} - \xi(\mathbf{z}^*).\end{aligned} \tag{2.27}$$

where the second term ensures differentiability by using the optimization problem's dual structure. This surrogate enables the end-to-end training of the predictive model with gradients propagated through the optimization step.

$$\Delta\mathbf{z} = -\alpha \left(\mathcal{P}(\alpha\mathbf{z} - \mathbf{z}^*) - \mathcal{P}(\mathbf{z}^*)\right). \tag{2.28}$$

The SPO framework has proven effective in domains where predictive errors can have significant downstream impacts, such as supply chain optimization, portfolio management, and energy resource allocation.

Melding the Data-Decisions Pipeline This approach embeds a combinatorial solver into the forward pass. For a problem:

$$x^\star = \arg\min_{x\in\mathcal{C}} f(x;\theta), \tag{2.29}$$

the optimization problem is solved for a given input θ.

Gradients are computed by differentiating through the optimization problem. Using the implicit function theorem, the gradient with respect to θ is:

$$\nabla_\theta x^\star = -\nabla_x^2 f(x^\star;\theta)^{-1} \nabla_\theta \nabla_x f(x^\star;\theta). \tag{2.30}$$

This approach requires solving a linear system involving the Hessian of the objective, making it computationally intensive for large-scale problems. However, it enables precise gradients for combinatorial tasks, ensuring that the predicted parameters lead to optimal decisions.

2.1.5 Reinforcement Learning

Reinforcement learning (RL) methods optimize sequential decision-making tasks by leveraging gradient-based approaches to improve policies. These methods differ in their techniques for forward computation of policy evaluations and backward optimization of gradient updates. Below, we merge and analyze key RL methods mathematically, focusing on their formulations for forward and backward computations.

Proximal Policy Optimization (PPO) PPO [35] stabilizes policy updates by clipping the objective function to prevent overly large updates that can destabilize training. The PPO loss function is given by:

$$L^{\text{PPO}}(\theta) = \mathbb{E}\left[\min\left(r(\theta)A, \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)A\right)\right], \tag{2.31}$$

where: - $r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}$ is the probability ratio of the new policy π_θ to the old policy $\pi_{\theta_{\text{old}}}$, - $A(s,a)$ is the advantage function, which estimates the relative value of an action a in state s, - ϵ is a clipping parameter that bounds the ratio $r(\theta)$.

Gradients are computed using stochastic gradient descent (SGD) to update policy parameters θ:

$$\nabla_\theta L^{\text{PPO}}(\theta) = \nabla_\theta \mathbb{E}\left[\min\left(r(\theta)A, \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)A\right)\right]. \tag{2.32}$$

This clipping mechanism ensures that the policy does not deviate significantly from the old policy, stabilizing training.

Trust Region Policy Optimization (TRPO) TRPO [80] constrains policy updates within a trust region defined by the Kullback-Leibler (KL) divergence. The optimization problem is:

$$\max_\theta \ \mathbb{E}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}A(s,a)\right], \quad \text{subject to } \text{KL}(\pi_\theta||\pi_{\theta_{\text{old}}}) \leq \delta, \tag{2.33}$$

where: - $\pi_\theta(a|s)$ is the policy probability of taking action a in state s, - $\mathrm{KL}(\pi_\theta||\pi_{\theta_{\text{old}}})$ measures the divergence between the new and old policies, - δ is the maximum allowed divergence, controlling update stability.

The optimization uses conjugate gradient methods to compute the policy update direction, solving:

$$\theta' = \theta + \alpha \nabla_\theta L(\theta), \tag{2.34}$$

where α is a step size chosen to satisfy the KL constraint. The conjugate gradient ensures computational efficiency while maintaining the trust region.

REINFORCE The REINFORCE algorithm [81] proposes a stochastic gradient-based approach for policy optimization. The policy gradient is expressed as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) R(\tau) \right], \tag{2.35}$$

where: - $\tau = (s_1, a_1, \ldots, s_T, a_T)$ is a trajectory sampled from the policy, - $R(\tau)$ is the cumulative reward of trajectory τ.

Gradients are computed using the policy gradient theorem:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) R(\tau_i) \right], \tag{2.36}$$

where N is the number of sampled trajectories. This method is simple but often suffers from high variance, making variance-reduction techniques crucial for stable training.

2.1.6 Probabilistic and Structured Modeling

Probabilistic approaches integrate uncertainty and discrete latent variables into learning frameworks. Discrete Variational Autoencoders [51] address the challenge of gradient computation for discrete variables through reparameterization techniques. Implicit MLE [52] extends this by enabling backpropagation through exponential family distributions using implicit differentiation methods. Stochastic Neurons [31] further advance probabilistic modeling by introducing gradient propagation techniques for stochastic units, facilitating conditional computation within neural networks.

Structured prediction methods focus on capturing dependencies among interdependent outputs, often leveraging graphical models. Conditional Random Fields (CRFs) are reformulated as differentiable recurrent layers in Conditional Random Fields as Recurrent Neural Networks [40], providing end-to-end trainability. Gaussian CRFs [82] enable efficient closed-form inference, particularly useful in low-level vision applications. Multiscale Conditional Random Fields [41] extend this framework to model fine-grained multiscale interactions, improving image segmentation performance.

2.2 Object Detection

Object detection [83, 84, 85, 86] is a critical task in computer vision that involves object classification and localization. The goal is to identify objects within an image and predict their spatial locations as bounding boxes. Each detected object is assigned a class label from a predefined set of categories. Object detection applications span various domains, including autonomous driving [67], medical imaging [87, 88] and video surveillance [89].

Object detection methods have evolved from traditional computer vision techniques relying on handcrafted features to modern, deep learning-based approaches. Early methods, such as sliding window detectors with feature descriptors like Histogram of Oriented Gradients (HOG) [90], struggled with efficiency and scalability [91]. The introduction of convolutional neural networks (CNNs) revolutionized the field, enabling models to learn hierarchical features directly from data and significantly improve accuracy [15]. However, CNN-based methods have limitations in capturing global context due to their localized receptive fields. This challenge has been partially addressed by incorporating transformer architectures, which leverage self-attention mechanisms to model global dependencies effectively [10].

2.2.1 Notation

Let's consider an input image I to formalise the object detection task. The objective is to predict a set of N objects, where two components characterize each object:

Table 2.1: Comparison of Differentiation and Optimization Methods: This table summarizes various methods used in optimization and machine learning tasks, including their mathematical formulations, input-output behavior, and primary use cases. These methods span applications such as combinatorial optimization, ranking, resource allocation, and reinforcement learning.

Method	Formulation	Input-Output	Use Cases
Straight-Through Estimator (STE)	$\frac{\partial \hat{y}}{\partial x} \approx 1$	Increasing	Binary activations, binarized networks
Finite-Difference	$\frac{\mathcal{L}(x^\star+\delta)-\mathcal{L}(x^\star)}{\delta}$	Nonlinear	Blackbox solvers, combinatorial optimization
Implicit Differentiation	$-\nabla_\theta \nabla_x^2 f(x^\star,\theta)^{-1}\nabla_x \mathcal{L}$	Increasing or decreasing	Quadratic programming, convex optimization layers
Relaxation-Based Learning	$\min_x \ \tilde{f}(x)+\lambda H(x)$	Increasing	Ranking, sorting, combinatorial tasks
Decision-Focused Learning	$\min_y \ g(y,h(x;\theta))$	Task-dependent	Resource allocation, planning, combinatorial tasks
Direct Optimization	$\min_\theta \mathcal{L}_{\text{Metric}}(y, f(x;\theta))$	Increasing	Object detection, ranking-based learning
Policy Optimization	$\mathbb{E}[\min(r(\theta)A, \text{clip}(\cdot))]$	Increasing	Policy optimization, reinforcement learning

- A class label $c_i \in \mathcal{C}$, where $\mathcal{C} = \{c_1, c_2, \ldots, c_K, c_{\text{bg}}\}$ represents the set of K predefined object classes, and c_{bg} denotes the background class.
- A bounding box $\mathbf{b}_i \in \mathbb{R}^4$, parameterized as (x_i, y_i, w_i, h_i), where (x_i, y_i) are the coordinates of the top-left corner, and (w_i, h_i) specify the width and height of the bounding box.

The distinction between anchor-free and anchor-based models arises in how these bounding boxes are predicted:

Anchor-Free Models predict bounding boxes directly from image features or key-points $k^{(i)}$, where $\mathbf{k} = \{k^{(1)}, k^{(2)}, \ldots, k^{(M)}\}$, without relying on predefined anchors [83]. These models determine bounding boxes based on key features, such as object centers or corners, simplifying the detection pipeline.

Anchor-Based Models rely on a set of predefined reference boxes, called anchors, distributed across the image. Each anchor $a^{(j)}$, where $\mathbf{a} = \{a^{(1)}, a^{(2)}, \ldots, a^{(N)}\}$, is refined through bounding box regression to align with the ground truth [83]. During training, a matching process assigns anchors to ground-truth objects based on metrics such as Intersection over Union (IoU).

The training objective for both approaches involves minimizing a combined loss function:

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \lambda \mathcal{L}_{\text{loc}}, \tag{2.37}$$

where $\mathcal{L}_{\text{cls}}$ denotes the classification loss, $\mathcal{L}_{\text{loc}}$ is the localization loss, and λ balances the contributions of the two terms.

2.2.2 Loss Functions for Object Detectors

Loss functions are critical in object detection as they guide the optimization of both classification and localization tasks. A well-designed loss function ensures accurate class predictions and precise bounding box regressions while addressing challenges like class imbalance and misaligned predictions. This section explores the most widely used loss functions, categorized into score-based losses, localization losses, and advanced formulations like IoU-based and ranking-based losses [83, 84, 85, 86].

2.2.2.1 Score-Based Loss Functions

Score-based losses focus on optimizing the classification component of object detection. These losses aim to ensure that each predicted bounding box is assigned the correct class label with high confidence.

Cross-Entropy Loss Cross-entropy loss is a widely used loss function for classification tasks, including object detection. It measures the dissimilarity between the predicted probability distribution and the true class labels. For a binary classification scenario, Cross-Entropy Loss is defined as:

$$\mathcal{L}_{\text{cls}} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(p_i) + (1-y_i)\log(1-p_i)\right], \tag{2.38}$$

where y_i is the ground-truth label, and p_i is the predicted probability for the positive class. For multi-class classification, the loss is extended by summing over all classes.

While effective, Cross-Entropy Loss often struggles with class imbalance, particularly in dense prediction tasks where background regions dominate.

Focal Loss To address class imbalance, Focal Loss [92] modifies Cross-Entropy Loss by down-weighting well-classified examples and focusing on hard-to-classified ones. The loss is defined as:

$$\mathcal{L}_{\text{focal}} = -\alpha(1-p_t)^{\gamma}\log(p_t), \tag{2.39}$$

where p_t is the predicted probability for the true class, α is a weighting factor for class imbalance, and γ controls the scaling factor. Higher values of γ focus more on hard examples. Focal Loss has been instrumental in improving performance in models like RetinaNet.

2.2.2.2 Localization Loss Functions

Localization losses measure the accuracy of predicted bounding boxes by comparing them with ground-truth boxes. These losses aim to minimize discrepancies in position, size, and shape.

Smooth L1 Loss Smooth L1 Loss, introduced in Fast R-CNN [4], is a robust alternative to L1 Loss. It is less sensitive to outliers due to its piecewise quadratic formulation:

$$\mathcal{L}_{\text{loc}} = \begin{cases} 0.5(x-y)^2 & \text{if } |x-y| < 1, \\ |x-y| - 0.5 & \text{otherwise,} \end{cases} \tag{2.40}$$

where x is the predicted bounding box coordinate, and y is the ground-truth coordinate. This loss strikes a balance between L1 Loss and L2 Loss, making it suitable for bounding box regression.

IoU-Based Losses Intersection over Union (IoU) is a standard metric for evaluating bounding box overlap, and IoU-based losses extend this concept to regression tasks. Basic IoU Loss is defined as:

$$\mathcal{L}_{\text{IoU}} = 1 - \text{IoU}(\mathbf{b}, \hat{\mathbf{b}}), \tag{2.41}$$

where $\mathbf{b}$ and $\hat{\mathbf{b}}$ are the ground-truth and predicted bounding boxes, respectively.

Extensions of IoU Loss, such as Generalized IoU (GIoU) [93], Distance IoU (DIoU), and Complete IoU (CIoU) [94], address limitations like non-overlapping boxes and incorporate factors such as box distance and aspect ratio consistency. For example, GIoU Loss improves alignment for non-overlapping boxes:

$$\mathcal{L}_{\text{GIoU}} = 1 - \text{IoU} + \frac{\text{Area of Union - Area of Smallest Enclosing Box}}{\text{Area of Smallest Enclosing Box}}. \tag{2.42}$$

2.2.2.3 Ranking-Based Loss Functions

Ranking-based losses focus on optimizing the relative ordering of predictions, ensuring that positive predictions are ranked higher than negatives. These losses align closely with metrics like Average Precision (AP), commonly used to evaluate object detection models.

Average Precision (AP) Loss AP Loss [95] directly optimizes the Average Precision metric by considering the ranking of positive and negative samples. The loss uses a pairwise comparison approach to adjust the confidence scores of predictions, improving the model's performance on densely populated datasets.

aLRP Loss The Localization-Recall-Precision (LRP) metric extends the AP concept by integrating localization quality into ranking. The aLRP Loss [96] optimizes this metric, prioritizing accurate classification and precise bounding box predictions. It is particularly effective in scenarios where localization quality is critical.

Rank & Sort Loss Rank & Sort Loss [21] extends ranking-based optimization by considering the continuous IoU values among positive predictions. This loss improves alignment between predicted and ground-truth boxes by focusing on both classification scores and IoU-based ranking.

2.2.3 Datasets for Object Detection

Datasets are pivotal in advancing object detection research, providing the necessary benchmarks for training, validation, and algorithm comparison. A good dataset includes diverse and challenging scenarios and provides comprehensive annotations for object localization and classification. This section explores some of object detection research's most widely used datasets.

COCO: Common Objects in Context The COCO (Common Objects in Context) dataset [97] is one of the most prominent benchmarks for object detection, instance segmentation, and image captioning. It features over 330,000 images, with 80 object categories, including animals, vehicles, and household items. COCO's annotations include object bounding boxes, segmentation masks, and key points for human pose estimation.

One of COCO's defining features is its emphasis on context. Images are selected to include complex scenes with multiple objects, occlusions, and interactions. For example, an image might depict people interacting with objects in crowded environments, creating challenges for object detectors to identify small, overlapping, or partially visible objects.

COCO evaluation metrics, such as mean Average Precision (mAP) at IoU thresholds from 0.5 to 0.95 (mAP@[0.5:0.95]), have become standard in the field. These metrics provide a comprehensive assessment of both classification and localization performance.

Pascal VOC The Pascal Visual Object Classes (VOC) dataset [98] was one of the first large-scale benchmarks for object detection. Pascal VOC provides annotated images across various scenes for 20 object categories, ranging from animals to vehicles. The dataset consists of two main splits: Pascal VOC 2007 and Pascal VOC 2012.

VOC annotations include bounding boxes, class labels, and segmentation masks, making them suitable for multi-task learning. Its relatively small size compared to COCO allows for rapid experimentation and model validation. VOC's evaluation metric, Average Precision (AP) at IoU = 0.5, remains widely used despite the dataset being overshadowed by newer benchmarks.

Cityscapes The Cityscapes dataset [99] focuses on semantic understanding of urban street scenes, making it particularly relevant for applications in autonomous driving and smart city planning. Cityscapes includes 5,000 finely annotated images and 20,000 coarsely annotated images collected from 50 cities. The dataset features annotations for 30 classes grouped into pedestrians, vehicles, and road infrastructure categories.

Cityscapes provides pixel-level annotations for semantic segmentation and instance segmentation, along with bounding box annotations. This makes it a comprehensive resource for evaluating object detection in real-world urban environments, where challenges like occlusion, motion blur, and varying lighting conditions are common.

LVIS: Large Vocabulary Instance Segmentation

The LVIS (Large Vocabulary Instance Segmentation) dataset [100] is designed to address the long-tail distribution problem in object detection. Unlike COCO and VOC, which focus on relatively balanced distributions of object categories, LVIS features a vocabulary of over 1,000 object categories. These categories include frequent and rare objects, highlighting the challenges of detecting objects with limited examples in the training data.

LVIS provides detailed instance segmentation masks for over 2 million object instances across 164,000 images. Its evaluation metrics include mAP, with a focus on measuring performance across frequent, common, and rare object categories. LVIS has become an essential benchmark for studying the scalability and robustness of

object detection models.

Open Images The Open Images dataset [101] is one of the most prominent object detection datasets, featuring over 9 million images annotated with 16 million bounding boxes across 600 object categories. Open Images includes diverse scenes, with annotations for bounding boxes, segmentation masks, and object relationships.

The dataset is designed for large-scale learning and includes various training, validation, and testing splits. Open Images is particularly valuable for evaluating object detectors at scale, as it covers a broader range of object classes and contexts than COCO and VOC.

Specialized Datasets

In addition to general-purpose datasets, several specialized datasets cater to specific domains, addressing unique challenges in object detection. The KITTI dataset [102], designed for autonomous driving, provides images captured from moving vehicles with annotations for cars, pedestrians, and cyclists. It is widely used for evaluating object detection models in road scenes. Objects365 [103], on the other hand, is a large-scale dataset featuring 365 categories and 10 million bounding boxes, making it suitable for training models that need to handle a diverse range of object types across various scenarios. Meanwhile, the DOTA dataset [104] focuses on object detection in aerial imagery, offering annotations for objects such as vehicles, ships, and buildings with oriented bounding boxes, making it a valuable resource for remote sensing and aerial surveillance applications. These specialized datasets enable the development of domain-specific object detection models tailored to meet the demands of distinct real-world applications.

2.2.4 Models for Object Detections

Object detectors are algorithms designed to identify and localize objects within an image, often represented by bounding boxes and associated class labels. Over the years, these detectors have been categorized into two primary types: anchor-based and anchor-free methods. Both categories have made significant contributions to object detection, addressing challenges such as accuracy, speed, and scalability. This

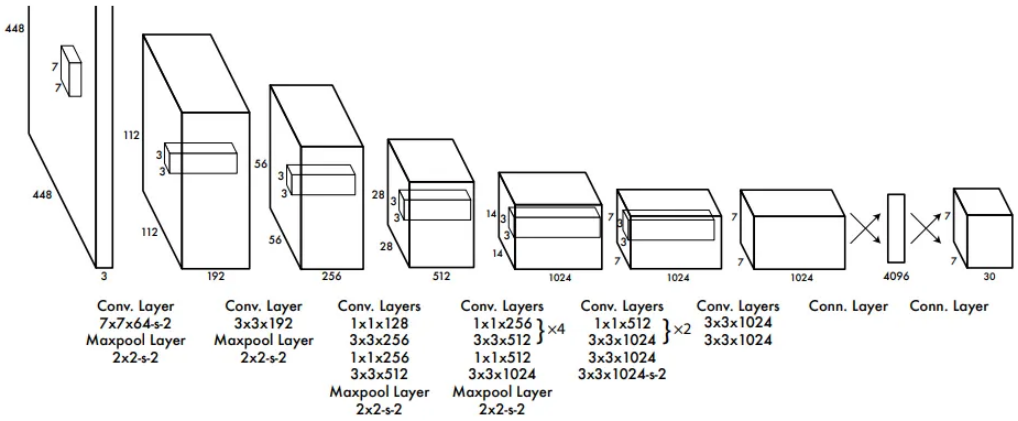

Figure 2.2: Architecture of YOLOv1: The input image is divided into a grid, and each grid cell predicts bounding boxes and class probabilities in a single forward pass. This unified structure enables real-time object detection. [Figure taken from [2]]

section explores both approaches' evolution and key features, as well as recent developments leveraging transformer architectures.

2.2.4.1 Anchor-Based Object Detectors

Anchor-based object detectors rely on a predefined set of reference boxes, or anchors, distributed across an image at different scales and aspect ratios. These anchors are initial candidates for bounding box predictions and are refined during training to align with ground-truth objects. The anchor-based approach has been a cornerstone of object detection, enabling precise localization and robust performance on benchmark datasets such as COCO [97] and Pascal VOC [98, 7].

One-Stage Detectors One-stage detectors simplify the object detection pipeline by eliminating the proposal generation step. These models directly predict bounding boxes and class labels from the input image, achieving real-time performance.

YOLO (You Only Look Once) [2] is one of the most prominent one-stage detectors. Figure 2.2 shows that YOLO divides the input image into a grid and predicts bounding boxes and class probabilities for each cell. Its single-pass architecture makes it highly efficient, but early versions struggled with small object detection. Subsequent

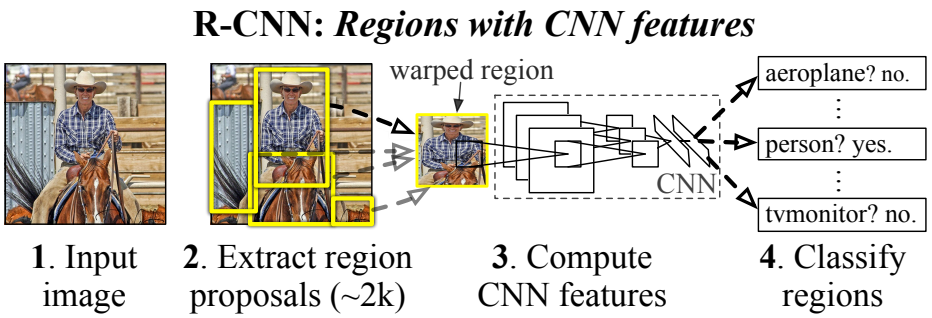

Figure 2.3: Overview of the R-CNN workflow: Region proposals are generated using selective search and processed independently through a CNN for feature extraction. Class labels and bounding box coordinates are predicted using SVMs and regression models [Figure taken from [3]].

versions, such as YOLOv3 [105] and YOLOv4 [106], introduced multi-scale feature maps and improved feature extraction techniques, significantly enhancing accuracy without compromising speed.

Similarly, SSD (Single Shot MultiBox Detector) [107] employs multi-scale feature maps and predefined anchors to handle objects of varying sizes. SSD balances speed and accuracy, making it a popular choice for real-time applications. Advances in one-stage detectors have focused on improving assignment strategies, feature representation, and post-processing techniques to bridge the performance gap with two-stage models [108].

Two-Stage Detectors Two-stage detectors pioneered using region proposals to achieve high accuracy in object detection tasks. These models generate a set of candidate object regions in the first stage, which are then refined in the second stage for classification and localization.

The R-CNN family represents a significant milestone in the development of two-stage detectors. The original R-CNN [3] introduced, depicted in 2.3, the concept of using selective search to generate approximately 2,000 region proposals, each of which was processed by a CNN to extract features. These features were subsequently classified using a support vector machine (SVM). While R-CNN demonstrated the potential of deep learning in object detection, its computational inefficiency, arising

from redundant feature extraction, limited its scalability.

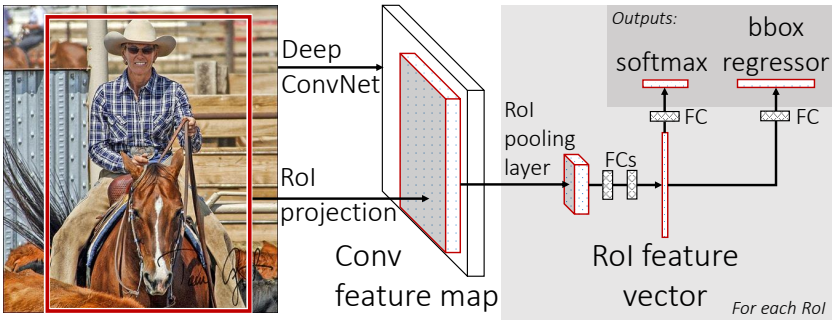

Figure 2.4: Fast R-CNN architecture: The entire image is passed through a CNN to produce a feature map, and region proposals are projected onto this map for ROI pooling. This process significantly reduces redundancy compared to R-CNN. [Figure taken from [4]]

Fast R-CNN [4] addressed these inefficiencies by introducing a shared convolutional feature map for the entire image. Figure 2.4 shows that the region proposals were projected onto this feature map, and region-of-interest (ROI) pooling was used to extract fixed-size feature representations for each proposal. This approach significantly reduced inference time while maintaining high detection accuracy. However, it still relied on external region proposal generation.

To overcome this dependency, Faster R-CNN [109] integrated a Region Proposal Network (RPN) into the detection pipeline, enabling end-to-end training. The RPN generates proposals directly from the shared feature map, streamlining the detection process. Faster R-CNN remains a benchmark for two-stage object detectors, achieving state-of-the-art performance across multiple datasets.

Extensions of Faster R-CNN, such as Mask R-CNN [110] and Cascade R-CNN [111], introduced additional capabilities and refinements. Mask R-CNN adds an instance segmentation branch, enabling pixel-level object segmentation, while Cascade R-CNN employs a multi-stage refinement process to improve detection quality at progressively higher IoU thresholds. These models have further advanced the accuracy and versatility of two-stage detectors.

2.2.4.2 Anchor-Free Object Detectors

Anchor-free object detectors eliminate the need for predefined anchors, predicting bounding boxes directly from key points or dense regions in the image. This approach simplifies the detection pipeline and reduces hyperparameter tuning while maintaining competitive performance.

Keypoint-based detectors represent a significant innovation in anchor-free object detection. CornerNet [112] formulates object detection to predict bounding box corners. By introducing a novel corner pooling mechanism, CornerNet enhances the localization of corners and achieves robust performance. Building on this concept, CenterNet [113] detects object centres along with their dimensions and offsets, streamlining the detection process and improving inference speed.

Another anchor-free approach, FCOS (Fully Convolutional One-Stage Object Detection) [114], treats object detection as a dense regression problem. FCOS predicts the distances from each feature map point to the edges of bounding boxes and introduces a centerness score to prioritize predictions near the object's centre. This method eliminates the need for anchor boxes and significantly reduces computational complexity while achieving state-of-the-art results on benchmark datasets.

2.2.4.3 Transformer-Based Object Detectors

The advent of transformer architectures has introduced a new paradigm in object detection. Transformers leverage self-attention mechanisms to model global dependencies, addressing limitations in CNN-based methods, such as localized receptive fields.

Detection Transformer (DETR) [5] was the first model to integrate transformers into the object detection pipeline, which can be seen in Figure 2.5. By formulating object detection as a set prediction problem, DETR eliminates the need for components like anchors, region proposals, and NMS. Its encoder-decoder architecture processes image features extracted by a CNN backbone while object queries predict bounding boxes and class labels. DETR's bipartite matching algorithm ensures a unique assign-

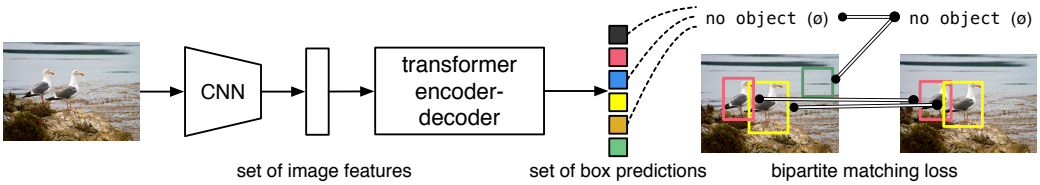

Figure 2.5: DETR pipeline: The input image is processed by a CNN backbone to produce feature maps, which are passed to a transformer encoder-decoder. Object queries interact with the encoded features to predict bounding boxes and class labels, with matching handled by the Hungarian algorithm. [Figure taken from [5]]

ment between predictions and ground truth, simplifying training and post-processing.

Despite its innovation, DETR suffers from slow convergence, often requiring hundreds of epochs to achieve optimal performance. Variants such as Deformable DETR [115], Conditional DETR [116], and Sparse DETR [117] have addressed these challenges through modifications like sparse attention, conditional queries, and learnable sparsity. DINO [118] has further improved the efficiency and scalability of this approach. These models have significantly improved the efficiency and scalability of transformer-based object detection.

2.3 Optimal Transport

The purpose of this section is to concisely introduce Optimal Transport (OT) [119] in $\mathbb{R}^d$. Let $\mathbb{P}(\mathbb{R}^d)$ denote the space of probability measures on $\mathbb{R}^d$. For a function $T : \mathbb{R}^d \to \mathbb{R}^d$, its push-forward operator $T_\sharp$ is defined by

$$(T_\sharp P)(A) = P\big(T^{-1}(A)\big) \quad \text{for any set } A \subset \mathbb{R}^d, \tag{2.43}$$

Let Lip_1 be the set of 1-Lipschitz functions, and let $\mathrm{CVX}(P)$ be the set of convex functions that have finite moments with respect to some distribution $P \in \mathbb{P}(\mathbb{R}^d)$. For a function $f : \mathbb{R}^d \to \mathbb{R}$, its *convex conjugate* is

$$f^*(\mathbf{x}_2) = \sup_{\mathbf{x}_1 \in \mathbb{R}^d} \Big(\langle \mathbf{x}_1, \mathbf{x}_2 \rangle - f(\mathbf{x}_1) \Big). \tag{2.44}$$

2.3.1 Monge and Kantorovich Formulations

Consider two distributions $P, Q \in \mathbb{P}(\mathbb{R}^d)$. The *Monge* approach [54] seeks a transport map T pushing P forward to Q ($T_\sharp P = Q$) and minimizes

$$\inf_{T_\sharp P = Q} \mathcal{L}_M(T) = \mathbb{E}_{\mathbf{x}\sim P}\big[c(\mathbf{x}, T(\mathbf{x}))\big]. \tag{2.45}$$

Here $c : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$ is the "ground cost." However, it is challenging to handle the constraint $T_\sharp P = Q$. The *Kantorovich* version [55] instead replaces the map T with a transport plan γ, a measure over $\mathbb{R}^d \times \mathbb{R}^d$, and solves

$$\inf_{\gamma \in \Gamma(P,Q)} \mathcal{L}_K(\gamma) = \mathbb{E}_{(\mathbf{x}_1,\mathbf{x}_2)\sim\gamma}\big[c(\mathbf{x}_1, \mathbf{x}_2)\big], \tag{2.46}$$

subject to the condition that γ has marginals P and Q. This version, known as Monge-Kantorovich (MK), is more tractable because $\Gamma(P, Q)$ and $\mathcal{L}_K(\gamma)$ are linear in γ.

By linear-program arguments [120], the MK problem has a dual in terms of Kantorovich potentials $\varphi, \psi : \mathbb{R}^d \to \mathbb{R}$. This yields

$$\sup_{(\varphi,\psi)\in\Phi_c} \mathcal{L}_K^*(\varphi, \psi) = \mathbb{E}_{\mathbf{x}\sim P}\big[\varphi(\mathbf{x})\big] + \mathbb{E}_{\mathbf{x}\sim Q}\big[\psi(\mathbf{x})\big], \tag{2.47}$$

with Φ_c being those pairs (φ, ψ) satisfying $\varphi(\mathbf{x}_1) + \psi(\mathbf{x}_2) \le c(\mathbf{x}_1, \mathbf{x}_2)$. In the special case $c(\mathbf{x}_1, \mathbf{x}_2) = \|\mathbf{x}_1 - \mathbf{x}_2\|_2^2$, Brenier's theorem [121] links Equations (2.45) and (2.46) by asserting that the optimal map is the gradient $\nabla\varphi$.

2.3.2 OT with Specific Costs

When $c(\mathbf{x}_1, \mathbf{x}_2) = d(\mathbf{x}_1, \mathbf{x}_2)^p$ for a metric d and $p \ge 1$, the resulting cost is widely studied. A noteworthy instance is $p = 1$, yielding the Kantorovich–Rubinstein (KR) formulation [57]:

$$\sup_{\varphi\in \mathrm{Lip}_1} \mathcal{L}_{KR}^*(\varphi) = \mathbb{E}_{\mathbf{x}\sim P}\big[\varphi(\mathbf{x})\big] - \mathbb{E}_{\mathbf{x}\sim Q}\big[\varphi(\mathbf{x})\big], \tag{2.48}$$

When $p > 1$, a time-dependent *dynamic OT*perspective [120] interprets mass being moved continuously from P to Q. One obtains an interpolation $\rho(t, \mathbf{x})$ with $\rho(0, \cdot) = P,\ \rho(1, \cdot) = Q$, and a velocity field $\mathbf{v}$. In that language, Equation (2.46) is replaced by

$$\mathcal{L}_B(\rho, \mathbf{v}) = \int_0^1 \int_{\mathbb{R}^d} \|\mathbf{v}(t, \mathbf{x})\|_2^p \ \rho_t(\mathbf{x})\ d\mathbf{x}\, dt, \tag{2.49}$$

under the continuity equation

$$\frac{\partial \rho_t}{\partial t} + \nabla \cdot (\rho_t \, \mathbf{v}) = 0. \tag{2.50}$$

2.3.3 OT-Induced Distances

A central property of OT is how it induces a *distance* between probability measures P and Q:

$$\mathcal{T}_c(P, Q) = \inf_{\gamma \in \Gamma(P,Q)} \mathbb{E}_{(\mathbf{x}_1, \mathbf{x}_2) \sim \gamma} \big[c(\mathbf{x}_1, \mathbf{x}_2) \big]. \tag{2.51}$$

If c arises from a metric on $\mathcal{X}$, then $\mathcal{T}_c$ is itself a metric [120]. The *Wasserstein* distances, for $p \geq 1$, are

$$W_p(P, Q) = \Big(\mathcal{T}_{d^p}(P, Q) \Big)^{\frac{1}{p}}. \tag{2.52}$$

These distances see broad use in machine learning. OT-based interpolation between distributions can be expressed via *Wasserstein geodesics* [122] and OT-based averaging is via *Wasserstein barycenters* [123, 124].

For instance, a geodesic from P to Q can be realized by $\mathbf{P}_t = \pi_{t,\sharp}\gamma$, where $\pi_t(\mathbf{x}_1, \mathbf{x}_2) = (1-t)\mathbf{x}_1 + t\mathbf{x}_2$. Similarly, for a collection of measures $\mathcal{P} = \{P_i\}_{i=1}^N$ with weights $\alpha \in \Delta_N$, the Wasserstein barycenter solves

$$\mathcal{B}\big(\alpha; \mathcal{P}\big) = \operatorname*{argmin}_{Q} \sum_{i=1}^{N} \alpha_i W_p\big(P_i, Q\big)^p. \tag{2.53}$$

2.4 Computational Optimal Transport

Research on computational aspects of OT has become quite active in machine learning. For example, Levy et al. [125] and Peyre et al. [126] provided key foundations in their resources, and widely used softwares such as POT [127] and OTT [128] are provided for researchers' use. In this overview, we concentrate on two common ways to discretize OT problems: (i) discretizing the ambient domain itself, and (ii) approximating distributions via samples. Concretely, let $\mathbf{x}_i^{(P)}$ be drawn from P with probability $p_i > 0$. Then the empirical approximation $\hat{P}$ of P takes the form

$$\hat{P}(\mathbf{x}) = \sum_{i=1}^{n} p_i \, \delta \left(\mathbf{x} - \mathbf{x}_i^{(P)} \right), \tag{2.54}$$

Summation of the weights implies $\sum_i p_i = 1$, or $\mathbf{p} \in \Delta_n$. Choosing to discretize the domain amounts to dividing it into bins (with each $\mathbf{x}_i^{(P)}$ as a bin center) and treating p_i as the proportion of mass in each bin. Alternatively, one can draw $\mathbf{x}_i^{(P)}$ as i.i.d. samples from P; in that scenario, all p_i are $1/n$, so the sample locations become the parameters.

Assume two sets of points $\{\mathbf{x}_i^{(P)}\}_{i=1}^n$ and $\{\mathbf{x}_j^{(Q)}\}_{j=1}^m$, coming from distributions P and Q with probabilities $\mathbf{p}$ and $\mathbf{q}$. The Monge formulation in the discrete realm seeks a transport map T minimizing

$$\sum_{i=1}^{n} c\left(\mathbf{x}_i^{(P)},\, T(\mathbf{x}_i^{(P)})\right), \tag{2.55}$$

subject to $\sum_{i\in\mathcal{I}} p_i = q_j$ for $\mathcal{I} = \left\{i : \mathbf{x}_j^{(Q)} = T(\mathbf{x}_i^{(P)})\right\}$. This approach is non-linear with respect to T, and no solution exists if $m > n$.

The Monge–Kantorovich (MK) alternative uses an OT plan $\gamma \in \mathbb{R}^{n\times m}$, with $\gamma_{i,j}$ representing mass moved from the i-th sample of P to the j-th sample of Q. The plan γ must minimize

$$\hat{\gamma} = T \arg\min_{\gamma\in\Gamma(\mathbf{p},\mathbf{q})} \sum_{i=1}^{n}\sum_{j=1}^{m} \gamma_{i,j}\, c\left(\mathbf{x}_i^{(P)}, \mathbf{x}_j^{(Q)}\right), \tag{2.56}$$

under constraints $\sum_i \gamma_{i,j} = q_j$ and $\sum_j \gamma_{i,j} = p_i$, i.e. $\gamma \in \Gamma(\mathbf{p}, \mathbf{q})$. This is a classic linear program, solvable by the simplex method [129] in $\mathrm{O}(n^3 \log n)$ time. A faster option arises from adding an entropic penalty, as introduced by Cuturi [130], leading to

$$\hat{\gamma}_\epsilon = \arg\min_{\gamma\in\Gamma(\mathbf{p},\mathbf{q})} \left\{ \sum_{i,j} \gamma_{i,j}\, c(\mathbf{x}_i^{(P)}, \mathbf{x}_j^{(Q)}) + \epsilon\, H(\gamma) \right\}. \tag{2.57}$$

This is more computationally efficient and can be solved using the Sinkhorn algorithm. An additional advantage is that the solution $\hat{\gamma}_\epsilon$ factorizes as

$$\hat{\gamma}_\epsilon = \mathrm{diag}(\mathbf{f})\, \exp\!\left(-\tfrac{\mathbf{C}}{\epsilon}\right)\, \mathrm{diag}(\mathbf{g}), \tag{2.58}$$

where $\mathbf{f}$ and $\mathbf{g}$ act as Kantorovich potentials (as in Equation (5)).

When using finite samples, one obtains an empirical estimate of $\mathcal{T}_c$ or W_p, namely $\mathcal{T}_c(\hat{P}, \hat{Q}) = \mathcal{L}_K(\hat{\gamma})$. Also, for γ_ϵ^*, we get $\mathcal{T}_{c,\epsilon}(\hat{P}, \hat{Q}) = \mathcal{L}_K(\gamma_\epsilon^*)$. Genevay et al. [131] proposes a *Sinkhorn divergence*

$$S_{p,\epsilon}(\hat{P}, \hat{Q}) = W_{p,\epsilon}(\hat{P}, \hat{Q}) - \frac{W_{p,\epsilon}(\hat{P}, \hat{P}) + W_{p,\epsilon}(\hat{Q}, \hat{Q})}{2}. \tag{2.59}$$

This smoothly interpolates between the Maximum Mean Discrepancy [132] and the Wasserstein distance. In practical terms, entropic OT offers two computational benefits over exact OT: it is GPU-friendly and can be computed in $\mathrm{O}(L\,n^2)$ steps after L Sinkhorn iterations. Furthermore, $S_{c,\epsilon}$ is a differentiable approximation of W_p [133] and has improved sample complexity [134].

2.4.1 Algorithmic Foundations of Optimal Transport

This section examines foundational algorithms and their theoretical underpinnings, focusing on their design principles and efficiency in solving OT problems. By leveraging the structure of the OT problem—specifically, its connections to combinatorial optimization and linear programming—these algorithms achieve robust performance across diverse domains.

2.4.1.1 Kantorovich Formulations and Algorithmic Foundations

The OT problem, formalized by Kantorovich, considers two probability distributions $\mathbf{a}$ and $\mathbf{b}$ defined on finite spaces and seeks a transport plan $\mathbf{P}$ that minimizes the cost of moving mass from one distribution to the other. This optimization problem is expressed in its primal form:

$$L_{\mathbf{C}}(\mathbf{a},\mathbf{b}) = \min_{\mathbf{P}\in\mathbf{U}(\mathbf{a},\mathbf{b})} \sum_{i=1}^{n}\sum_{j=1}^{m} \mathbf{C}_{ij}\mathbf{P}_{ij}, \tag{2.60}$$

where $\mathbf{C}$ is a cost matrix, and $\mathbf{U}(\mathbf{a},\mathbf{b})$ represents the feasible transport plans satisfying marginal constraints:

$$\mathbf{P}\mathbb{1}_m = \mathbf{a}, \quad \mathbf{P}^\top\mathbb{1}_n = \mathbf{b}, \quad \mathbf{P} \geq 0. \tag{2.61}$$

The dual formulation, derived using linear programming duality, introduces potential vectors $\mathbf{f}$ and $\mathbf{g}$ associated with the supply and demand distributions:

$$L_{\mathbf{C}}(\mathbf{a},\mathbf{b}) = \max_{\mathbf{f},\mathbf{g}} \left\{ \sum_{i=1}^{n} \mathbf{f}_i\mathbf{a}_i + \sum_{j=1}^{m} \mathbf{g}_j\mathbf{b}_j : \mathbf{f}_i + \mathbf{g}_j \leq \mathbf{C}_{ij}\ \forall i,j \right\}. \tag{2.62}$$

2.4.1.2 Algorithmic Approaches to Optimal Transport

Linear Programming Solvers Linear programming (LP) provides a direct approach to solving OT problems, treating the transport cost as a linear objective function with constraints encoded in a sparse constraint matrix. This perspective enables the application of general LP solvers, such as the simplex method and interior-point methods.

By vectorizing $\mathbf{P}$ into $\mathbf{p} \in \mathbb{R}^{n \cdot m}$ and encoding constraints in a matrix $\mathbf{A}$, the primal problem becomes:

$$\min_{\mathbf{p} \geq 0, \mathbf{A}\mathbf{p} = \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}} \mathbf{c}^\top \mathbf{p}. \tag{2.63}$$

Since feasible transport plans often contain only $n + m - 1$ nonzero entries (extremal solutions of the transportation polytope), LP solvers can exploit sparsity to reduce computational complexity.

LP solvers are versatile but can be computationally intensive for large-scale problems due to the $\mathcal{O}(n^3)$ complexity of standard simplex implementations [26].

Simplex Algorithm The network simplex algorithm is a specialized variant of the simplex method designed for flow problems, including OT. It operates on the bipartite graph representing the transport problem, where nodes correspond to supply and demand locations, and edges represent transport routes with associated costs.

The algorithm starts with a feasible transport plan, often generated using heuristics like the North-West Corner Rule. At each step, the algorithm identifies cycles in the graph and adjusts flow along these cycles to reduce transport cost, ensuring feasibility at each iteration. The network simplex achieves significantly faster convergence than standard LP solvers by maintaining and updating only a subset of variables (those corresponding to active edges in the graph).

The network simplex is particularly effective for medium-sized problems but may face scalability issues due to degeneracy in large-scale settings .

Auction Algorithm The auction algorithm [135], introduced by Bertsekas, is inspired by market dynamics. It interprets the OT problem as a resource allocation task

in which suppliers "bid" for demand locations.

The algorithm iteratively updates dual variables (prices) to ensure complementary slackness while incrementally constructing a primal solution. Each supply node bids for the demand node offering the highest profit (reduced cost). The bid value reflects the difference between the current price and the second-best alternative, ensuring rapid convergence. The auction algorithm is highly parallelizable and well-suited for assignment problems (a special case of OT with unit supply and demand).

For general OT problems, adaptations of the auction algorithm incorporate scaling techniques to handle continuous supply distributions efficiently [135].

Sinkhorn Algorithm The Sinkhorn algorithm introduces entropy regularization to the OT problem, yielding a strictly convex objective:

$$L_{\mathbf{C},\lambda}(\mathbf{a},\mathbf{b}) = \min_{\mathbf{P}\in\mathbf{U}(\mathbf{a},\mathbf{b})} \sum_{i,j} \mathbf{C}_{ij}\mathbf{P}_{ij} + \lambda \sum_{i,j} \mathbf{P}_{ij}\log \mathbf{P}_{ij}. \tag{2.64}$$

The algorithm alternates between normalizing the rows and columns of $\mathbf{P}$ to satisfy marginal constraints:

$$\mathbf{P} \leftarrow \mathrm{diag}(\mathbf{u})\mathbf{K}\mathrm{diag}(\mathbf{v}), \tag{2.65}$$

where $\mathbf{K} = \exp(-\mathbf{C}/\lambda)$, and $\mathbf{u}, \mathbf{v}$ are scaling factors updated iteratively. Sinkhorn's method exhibits linear convergence under mild conditions, with a complexity of $\mathcal{O}(n^2/\lambda^2)$ per iteration [20].

The Sinkhorn algorithm is a cornerstone of modern OT, particularly in machine learning, due to its scalability and compatibility with GPU acceleration.

Multiscale and Parallel Methods Multiscale methods decompose the problem into hierarchies of smaller subproblems to tackle high-dimensional OT problems. At each scale, coarse approximations guide finer-scale solutions, enabling efficient convergence [136]. Parallel algorithms leverage GPU and distributed architectures to accelerate computation, making OT feasible for real-time applications.

Algorithm 1 Sinkhorn Algorithm for Entropic Regularized Optimal Transport

Require: Cost matrix $C \in \mathbb{R}^{n \times m}$, Source histogram $\mathbf{a} \in \mathbb{R}^n$, Target histogram $\mathbf{b} \in \mathbb{R}^m$, Regularization parameter λ (default $\lambda = 0.1$), Maximum number of iterations maxiters.

Ensure: Optimal transport plan $\mathbf{S} \in \mathbb{R}^{n \times m}$.

$\mathbf{K} \leftarrow \exp\left(-\frac{\mathbf{C}}{\lambda}\right)$ ▷ Compute the kernel matrix

$u \leftarrow \mathbb{1}_n$ ▷ Initialize scaling factor u

$v \leftarrow \mathbb{1}_m$ ▷ Initialize scaling factor v

$\mathbf{S} \leftarrow \text{diag}(u) \cdot K \cdot \text{diag}(v)$ ▷ Initialize transport plan

for $k = 1$ to maxiters **do**

  $u \leftarrow a \odot (K \cdot v)^{-1}$ ▷ Update scaling factor u (Eq. 2.69)

  $v \leftarrow b \odot \left(K^\top \cdot u\right)^{-1}$ ▷ Update scaling factor v (Eq. 2.69)

  $\mathbf{S} \leftarrow \text{diag}(u) \cdot \mathbf{K} \cdot \text{diag}(v)$ ▷ Update transport plan (Eq. 2.70)

end for

return S ▷ Return the optimal transport plan

2.4.2 Entropic Regularization in Optimal Transport

Entropic regularization has emerged as a pivotal approach in the numerical approximation of optimal transport problems, particularly the Kantorovich formulation. By introducing an entropic penalty, the regularization modifies the transport problem to achieve smoother, more computationally tractable solutions while preserving essential structural properties. This chapter delves into this methodology's mathematical underpinnings, computational techniques, and practical implications.

2.4.2.1 Entropic Regularization Framework

The discrete entropy function for a coupling matrix $\mathbf{P}$ is defined as:

$$\mathbf{H}(\mathbf{P}) = -\sum_{i,j} \mathbf{P}_{i,j} \left(\log(\mathbf{P}_{i,j}) - 1\right), \tag{2.66}$$

where $\mathbf{H}(\mathbf{P}) = -\infty$ if any entry of $\mathbf{P}$ is zero or negative. By integrating this entropy term into the classical transport problem, we define the regularized transport problem:

$$\mathrm{L}^{\varepsilon}_{\mathbf{C}}(\mathbf{a}, \mathbf{b}) = \min_{\mathbf{P} \in \mathbf{U}(\mathbf{a}, \mathbf{b})} \langle \mathbf{P}, \mathbf{C} \rangle - \varepsilon \mathbf{H}(\mathbf{P}), \tag{2.67}$$

where $\varepsilon > 0$ is the regularization parameter, and $\mathbf{U}(\mathbf{a}, \mathbf{b})$ represents the set of couplings with fixed marginals $\mathbf{a}$ and $\mathbf{b}$. Including entropy ensures that the solution is unique, smooth, and computationally stable, even in high-dimensional settings [20].

Computational Benefits The regularized formulation exhibits several computational advantages: 1. Simplified Optimization: The problem becomes ε-strongly convex, allowing the application of alternating minimization algorithms. 2. Efficient Matrix Operations: Iterative updates involve only matrix-vector products, facilitating parallelization on GPUs. 3. Memory Efficiency: Sparse representations reduce memory requirements, particularly for kernel evaluations. 4. Improved Differentiability: The regularised solution's smoothness enables automatic differentiation in optimization tasks.

Convergence Properties As $\varepsilon \to 0$, the regularized solution converges to the solution with maximum entropy among all optimal solutions of the original transport problem:

$$\mathbf{P}_\varepsilon \to \operatorname{argmin}\{-\mathbf{H}(\mathbf{P}) : \mathbf{P} \in \mathbf{U}(\mathbf{a}, \mathbf{b}), \langle \mathbf{P}, \mathbf{C} \rangle = \mathrm{L}_{\mathbf{C}}(\mathbf{a}, \mathbf{b})\}. \tag{2.68}$$

In contrast, as $\varepsilon \to \infty$, the solution converges to the outer product of the marginals $\mathbf{a}$ and $\mathbf{b}$, representing an independent coupling.

Sinkhorn's Algorithm Sinkhorn's algorithm is a key computational tool for solving the entropic regularized problem. This algorithm iteratively adjusts scaling factors $\mathbf{u}$ and $\mathbf{v}$:

$$\mathbf{u}^{(\ell+1)} = \frac{\mathbf{a}}{\mathbf{K}\mathbf{v}^{(\ell)}}, \quad \mathbf{v}^{(\ell+1)} = \frac{\mathbf{b}}{\mathbf{K}^T\mathbf{u}^{(\ell+1)}}. \tag{2.69}$$

where $\mathbf{K}_{i,j} = \exp(-\mathbf{C}_{i,j}/\varepsilon)$. The coupling satisfies the marginal constraints:

$$\mathbf{P}_{i,j} = \mathbf{u}_i \mathbf{K}_{i,j} \mathbf{v}_j, \tag{2.70}$$

Sinkhorn's algorithm exhibits linear convergence under suitable conditions and scales efficiently to large datasets [126].

2.5 Summary

This chapter reviewed a broad range of literature and key concepts related to gradient approximation methods. We presented various techniques for handling differentially

intractable or inefficient functions, which behave piecewise linearly with constant regions, discussing their applications and inherent limitations. We also introduced the Optimal Transport (OT) to create differentiable solutions for naturally discrete tasks, as illustrated by transformer-based object detectors.

Furthermore, we argued for a unified framework combining theoretical rigour and practical implementation. Such a framework would streamline the development and selection of gradient approximation methods. Additionally, we outlined measure-theoretic and classical machine learning principles that can guide the construction of this comprehensive approach.

These discussions lay the foundation for the subsequent chapters. The following sections will demonstrate how the Generalized Update method integrates these insights into a coherent, theoretically sound, and practically efficient approach for gradient approximation and OT integration in deep learning.

CHAPTER 3

GENERALIZED UPDATE

This chapter introduces Generalized Update (GU) as a generalization of the gradient-approximations methods reviewed in Section 2.1.

3.1 Piecewise-Linear Functions with Constant Regions (PFCs)

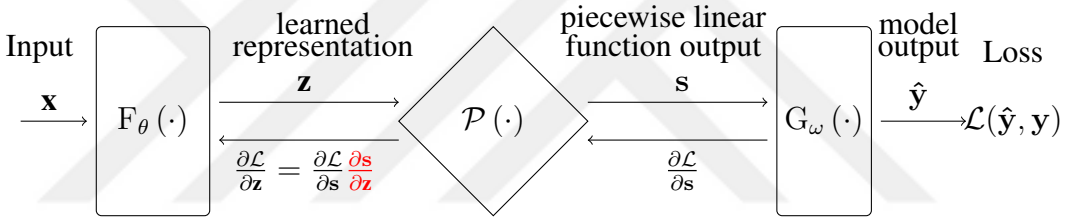

Figure 3.1: *Notation and Problem Formulation.* We are interested in incorporating PFCs into the deep neural network (DNN) pipelines. Considering this problem from a more general perspective is crucial as $\mathcal{P}(\cdot)$ can represent essential operations such as step function, a combinatorial solver or a performance measure. As such operations do not provide useful gradients $\frac{\partial \mathbf{s}}{\partial \mathbf{z}}$ for training, we review the existing methods to approximate $\frac{\partial \mathbf{s}}{\partial \mathbf{z}}$ and unify them in our *Generalized Update* method based on our theoretical and empirical analyses. As expected, our method provides approximate gradients for all operations, as mentioned earlier and provides notable gains in some cases.

As we have discussed in Section 2.1, incorporating piecewise linear functions with constant regions (PFCs) into neural networks extends the capability of deep learning models to address tasks involving discrete decision-making, combinatorial optimization, and other structured prediction problems. As shown in Figure 3.1, the PFC $\mathcal{P}(\cdot)$

represents discrete decision-making or structured prediction operations such as step functions, combinatorial solvers, or performance metrics, all of which have inherently either inefficient or intractable gradients. This section formalizes the problem, defines the key challenges, and identifies the fundamental issue to address.

Notation Given an input $\mathbf{x}$, a neural network $\mathrm{F}_\theta(\mathbf{x})$ generates an intermediate representation $\mathbf{z}$. This representation is passed through a PFC $\mathcal{P}(\cdot)$, producing the output $\mathbf{s}$, which is further processed by subsequent network layers $\mathrm{G}_\omega(\cdot)$ to yield the final output $\hat{\mathbf{y}}$. Mathematically, the pipeline can be expressed as:

$$\hat{\mathbf{y}} = \mathrm{G}_\omega(\mathbf{s}), \quad \mathbf{s} = \mathcal{P}(\mathbf{z}), \quad \mathbf{z} = \mathrm{F}_\theta(\mathbf{x}). \tag{3.1}$$

Computation of PFC In the forward pass, the model computes $\hat{\mathbf{y}}$ given $\mathbf{x}$ by sequentially applying $\mathrm{F}_\theta(\cdot)$, $\mathcal{P}(\cdot)$, and $\mathrm{G}_\omega(\cdot)$. This computation is straightforward because all functions in the pipeline are predefined and deterministic. For example:

$$\hat{\mathbf{y}} = \mathrm{G}_\omega(\mathcal{P}(\mathrm{F}_\theta(\mathbf{x}))). \tag{3.2}$$

While this enables efficient computation of $\hat{\mathbf{y}}$, the challenge arises in the backward pass, where gradients must be computed to update the model parameters.

Optimization of PFC In the backward pass, gradient-based optimization relies on the chain rule to compute parameter updates. Considering the pipeline:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})}{\partial \hat{\mathbf{y}}} \cdot \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{s}} \cdot \frac{\partial \mathbf{s}}{\partial \mathbf{z}} \cdot \frac{\partial \mathbf{z}}{\partial \theta}, \tag{3.3}$$

As shown in Figure 1.1, the gradient of a PFC, i.e., $\frac{\partial \mathbf{s}}{\partial \mathbf{z}}$, is zero or undefined because $\mathcal{P}(\cdot)$ is constant almost everywhere. This disrupts the flow of gradient information back to the learnable parameters θ in $\mathrm{F}_\theta(\cdot)$, rendering standard backpropagation inapplicable:

$$\frac{\partial \mathbf{s}}{\partial \mathbf{z}} = 0 \quad \text{(constant regions)}. \tag{3.4}$$

The network cannot learn effective representations without meaningful gradients, and the model training stagnates.

Fundamental Problem The core challenge of incorporating PFCs into neural networks is to define a substitute for the missing or zero gradients. Formally, the task is to find an update rule $\Delta\mathbf{z}$ to replace the ineffective gradient $\frac{\partial \mathbf{s}}{\partial \mathbf{z}}$ so the model can still

learn end-to-end. The following sections discuss various methods proposed in the literature to address this challenge, focusing on techniques for approximating gradients and enabling gradient-based optimization for piecewise linear functions with constant regions.

3.2 Limitations of Existing Gradient Approximation Methods

As discussed in Section 2.1, numerous methods have been proposed to address the challenge of backpropagating through PFC layers. Techniques such as the Straight-Through Estimator (STE) [69], Rank-Identity Update [21], blackbox differentiation via local perturbations [6], and reward maximization with REINFORCE [137] have made significant contributions. However, several key limitations remain.

Many methods are tailored to specific tasks (e.g., binary neural networks [138, 139]) or particular performance measures (e.g., ranking-based losses [21]). This restricts their applicability to a narrow set of non-differentiable functions. Surrogate-based or interpolation methods [22, 42, 140] may involve repeated function evaluations, leading to heavy computational demands. Similarly, optimization-as-a-layer approaches [42, 1] often rely on complex solvers or iterative schemes. Many gradient approximations lack a solid theoretical grounding, making it difficult to predict their behaviour, ensure stability, or guarantee convergence.

3.3 Introducing Generalized Update Method

We propose the *Generalized Update* method to address the limitations of the existing gradient approximation methods introduced in Section 3.2. Our approach provides a unifying framework that subsumes multiple existing gradient approximation strategies under a single theoretical umbrella, ensuring generality, simplicity, and improved computational properties.

3.3.1 Contributions of the Generalized Update Method

The Generalized Update method offers:

- **Generalization of Existing Methods**: We show that prominent methods, including STE and Rank-Identity Update, and methods linked to combinatorial solvers, can be seen as exceptional cases of the Generalized Update.
- **Theoretical Connections**: By linking our update rule to classical learning principles such as the Perceptron algorithm [141] and Maximum Likelihood Estimation (MLE) [142, 143], we provide a firm theoretical foundation.
- **Efficiency and Practicality**: The Generalized Update simplifies the backward pass through PFCs, reducing complexity and often lowering computational overhead.

3.3.2 Assumptions and Formal Definition

Following the problem formulation and notations defined in Section 3.1, given an input $\mathbf{x}$, we want to learn a model $\mathrm{F}_\theta(\mathbf{x})$ that produces a useful representation $\mathbf{z}$. This $\mathbf{z}$ is then passed through a PFC $\mathcal{P}(\cdot)$, resulting in an output $\mathbf{s}$, which is further processed in the network.

Assumptions Typically, $\mathrm{F}_\theta(\cdot)$ and $\mathrm{G}_\omega(\cdot)$ are neural networks. For $\mathcal{P}(\cdot)$, we impose two assumptions on PFCs:

First, the domain and range of $\mathcal{P}(\cdot)$ have the same dimensionality: $\dim(\mathcal{Z}) = \dim(\mathcal{S}) = n$.

Second, $\mathcal{P}(\cdot)$ is a vector-valued monotonic[1] (i.e. either increasing or decreasing) function.

Additionally, we assume that there exists a loss function $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$ such that $\frac{\partial \mathcal{L}}{\partial \mathbf{z}} \approx \hat{\mathbf{y}} - \mathbf{y}$.

[1] $\mathcal{P}(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$ is an increasing vector-valued function if $\mathbf{x}_i \leq \mathbf{y}_i$ whenever $\mathcal{P}(\mathbf{x})_i \leq \mathcal{P}(\mathbf{y})_i$ for every $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. Similarly, $\mathcal{P}(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$ is a decreasing vector-valued function if $\mathbf{x}_i \geq \mathbf{y}_i$ whenever $\mathcal{P}(\mathbf{x})_i \leq \mathcal{P}(\mathbf{y})_i$ for every $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. Monotonicity indicates it is either an increasing or decreasing function.

These assumptions, while constraining, still encompass a diverse set of operations. For instance, a step function or a combinatorial solver, such as Dijkstra's shortest path algorithm, satisfies our assumptions on $\mathcal{P}(\cdot)$. After obtaining $\mathbf{s}$, additional operations may be denoted by $\mathrm{G}_{\omega}(\cdot)$ in Figure 3.1. $\mathrm{G}_{\omega}(\cdot)$ can consist of learnable parameters (e.g., additional neural network layers) or operations to manipulate $\mathbf{s}$ for computing the loss function $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$, where $\hat{\mathbf{y}}$ and $\mathbf{y}$ denote the prediction and the ground truth, respectively.

Generalized Update The Generalized Update suggests:

$$\Delta \mathbf{z} = \alpha \left(\mathcal{P}(\mathbf{z}) - \mathcal{P}(\mathbf{z})^{*} \right), \tag{3.5}$$

where α is $+1$, when $\mathcal{P}(\cdot)$ is monotone increasing, and α is -1, when $\mathcal{P}(\cdot)$ is monotone decreasing.

In a more general form:

$$\Delta \mathbf{z} = \begin{cases} \mathcal{P}(\mathbf{z}) - \mathcal{P}(\mathbf{z})^{*}, & \text{if } \mathcal{P}(\cdot) \text{ is monotone increasing,} \\ \mathcal{P}(\mathbf{z})^{*} - \mathcal{P}(\mathbf{z}), & \text{if } \mathcal{P}(\cdot) \text{ is monotone decreasing.} \end{cases} \tag{3.6}$$

Here, $\mathcal{P}(\mathbf{z})^{*}$ represents the target value associated with $\mathcal{P}(\mathbf{z})$.

3.3.3 Motivations for the Generalized Update Method

Example 3.1 (Stock Price Risk Prediction) *Consider a stock price prediction scenario involving N stocks. Let $\boldsymbol{x} \in \mathbb{R}^{n}$ be a histogram of observable random variables that capture news articles, stock price reports, and Twitter data. A neural network F_{θ} processes $\boldsymbol{x}$ to produce a risk vector $\boldsymbol{z} = \mathrm{F}_{\theta}(\boldsymbol{x})$, which represents the cost or risk associated with investing in each of the N stocks.*

The goal is to train the network so that $\boldsymbol{z}$ accurately reflects the true risk associated with each stock, guiding the investment decision $\mathbf{s} = \mathcal{P}(\boldsymbol{z})^{*}$. The predicted investment decision is denoted by $\hat{\mathbf{s}} = \mathcal{P}(\boldsymbol{z})$.

To derive $\hat{\mathbf{s}}$, we formulate a linear programming problem. Specifically, we define a

cost function $\phi(\boldsymbol{z}, \hat{\mathbf{s}}) = \boldsymbol{z}^\top \hat{\mathbf{s}}$ *and solve the following optimization problem:*

$$\begin{aligned} \mathcal{P}(\boldsymbol{z}) = \underset{\mathbf{s}\in\mathbb{R}^N}{\text{minimize}} \quad & \boldsymbol{z}^\top \mathbf{s}, \\ \textit{subject to} \quad & \sum_{i=1}^{N} \mathbf{s}_i = 1, \\ & \mathbf{s}_i \geq 0 \quad \textit{for all } i. \end{aligned} \tag{3.7}$$

The solution $\hat{\mathbf{s}}$ *represents the investment decision that minimizes the overall risk cost.*

Since the decision-making process involves solving a linear programming (or the Heaviside step function used in some formulations), standard gradient-based optimization cannot be applied directly.

However, it is intuitive to consider a suboptimal case, i.e., $\hat{\mathbf{s}} \neq \mathbf{s}$ *since* $\hat{\mathbf{s}}_i \neq \mathbf{s}_i$ *for some* $0 \leq i < N$*. Without loss of generality, let's assume* $\hat{\mathbf{s}}_i \leq \mathbf{s}_i$ *for some* $0 \leq i < N$*. This indicates that the investment decision* $\hat{\mathbf{s}}_i$ *for the* i*th stock should be more favourable; in other words, the predicted risk* $\boldsymbol{z}_i$ *for this stock is higher than desired, and the model* F_θ *needs to adjust its output to lower this risk cost* $\boldsymbol{z}$*.*

Since the decision component —derived from solving the linear programming problem— is deterministic, whereas the neural network that produces the risk vector is probabilistic and updated iteratively based on its hyperparameters, the gradient for F_θ *must update* θ *so that the predicted risk* $\boldsymbol{z}_i = \mathrm{F}_\theta(\boldsymbol{x})_i$ *decreases for the* i*-th stock.*

Therefore, assuming $\hat{\mathbf{s}}_i \leq \mathbf{s}_i$ *for* $0 \leq i < N$ *provides information about the gradient's sign or direction, namely* $\frac{\partial \mathcal{L}}{\partial \boldsymbol{z}_i} < 0$*. In other words, a slight increase in the current cost value* $\boldsymbol{z}_i$ *leads to a decrease in the loss function* $\mathcal{L}$*, which measures the discrepancy between the current predicted decision* $\mathcal{P}(\mathrm{F}_\theta(\boldsymbol{x}))$ *and the target decision* $\mathcal{P}(\mathrm{F}_\theta(\boldsymbol{x}))^*$*.*

Conversely, if we assume $\hat{\mathbf{s}}_i \geq \mathbf{s}_i$ *for some* $0 \leq i < N$*, this indicates that the investment decision for the* i*th stock is too favourable —that is, the predicted risk* $\boldsymbol{z}_i$ *is lower than desired. In this case, the gradient for the neural network* F_θ *should update* θ *such that* $\boldsymbol{z}_i$ *increases to better reflect the true risk. This condition implies that* $\frac{\partial \mathcal{L}}{\partial \boldsymbol{z}_i} > 0$*; in other words, a slight increase in* $\boldsymbol{z}_i$ *would decrease the loss function* $\mathcal{L}$*.*

The Generalized Update, as described in Equation 3.6, is the minimal equation that satisfies the gradient directions discussed in Example 3.1, which is not coincidental

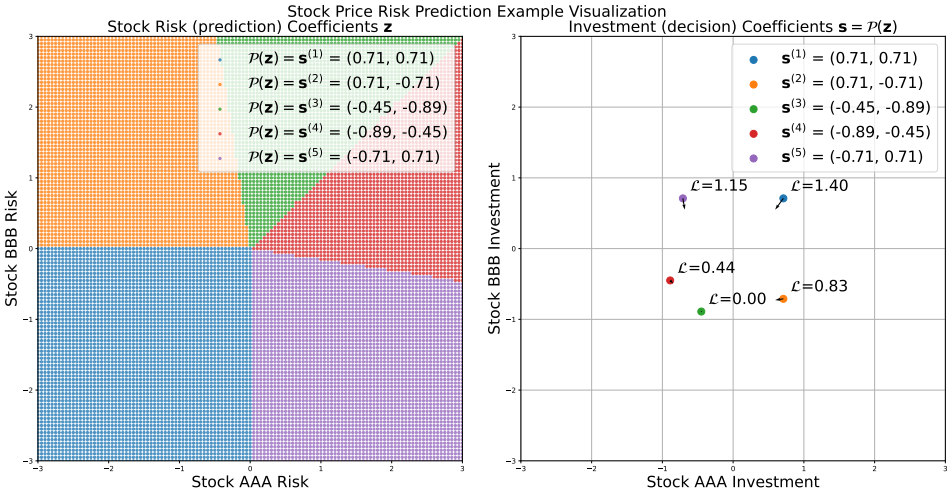

Figure 3.2: The left panel displays the prediction coefficients $\mathbf{z}$, where the price costs for each stock are visualized using the colour of the strategy that minimizes the risk. The right panel shows the decision coefficients, corresponding loss values and gradient update directions.

and arises because the decision-making layer $\mathcal{P}(\cdot)$ is a monotone-decreasing vector-valued function. In the Generalized Update, the gradient directions are consistently maintained and parametrized by α, which determines whether the update increases or decreases the predicted cost. This design ensures that the update rule preserves the essential directional information provided by the monotonicity of $\mathcal{P}(\cdot)$.

Example 3.2 (Illustrative Example of the Generalized Update Method) *Let's consider Example 3.1 from a geometric perspective. For visualization, we simplify the scenario by setting $N = 2$, meaning we have two stocks, AAA and BBB, and we restrict the strategy space to five discrete options:*

$$\mathcal{S} = \left\{ \left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2} \right), \left(\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2} \right), \left(-\frac{1}{2}, -1 \right), \left(-1, -\frac{1}{2} \right), \left(-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2} \right) \right\}. \tag{3.8}$$

Each strategy is assigned a unique colour: blue, orange, green, red, and purple, respectively. Figure 3.2 illustrates the geometric interpretation of the risk minimization process and provides additional insight into how the model updates its predictions based on the cost minimization principle.

Moreover, Figure 3.2 reveals that, for any given price cost vector $\mathbf{z}$, the predicted

investment $\hat{\mathbf{s}} \in \mathcal{S}$ is computed by $\mathcal{P}(\boldsymbol{z})$ such that the inner product with the risk vector $\boldsymbol{z}$ is maximized. This selection process follows directly from the linear programming formulation, where the goal is to minimize the overall cost $\phi(\boldsymbol{z}, \hat{\mathbf{s}})$ by selecting the strategy that best aligns with the current risk profile. As defined, the inner product $\mathbf{z}^\top \mathbf{s} = \|\mathbf{z}\|\|\mathbf{s}\| \cos(\theta)$ quantifies the alignment between $\boldsymbol{z}$ and the strategy vector $\mathbf{s}$; a smaller angle θ corresponds to a higher inner product.

Without loss of generality, for this hypothetical example, we selected the green strategy, i.e., $\mathbf{s}^{(3)} = (-\frac{1}{2}, -1)$, as the true investment strategy under the given scenario $\boldsymbol{x}$. Figure 3.2 also shows that applying the Generalized Update rule (Equation 3.6) helps F_θ adjust its hyperparameters θ so that the predicted risk vector $\boldsymbol{z}$ converges towards the optimal solution $\mathbf{s}^{(2)}$. This demonstrates that the update mechanism effectively reduces the discrepancy between the predicted and true strategies even with a minimal approximation of the gradient direction.

3.4 Relation to Existing Gradient-Approximation Methods

To understand the motivations behind the Generalized Update method, it is critical to examine foundational methodologies like the Straight-Through Estimator [31] (STE), the Perceptron algorithm [16], and Maximum Likelihood Estimation (MLE) [144]. These methods have shaped the development of modern techniques for handling piecewise linear components, discrete decision-making, and probabilistic learning.

Perceptron The Perceptron algorithm [16] is one of the earliest examples of gradient-based learning. Recalling from Eq. (1.1) and Eq. (1.2), it models binary decision-making through a simple linear classifier. During training, weights are updated using the rule:

$$\theta \leftarrow \theta + \eta(\mathbf{y} - \hat{\mathbf{y}})\mathbf{x}, \tag{3.9}$$

where η is the learning rate, $(\mathbf{y} - \hat{\mathbf{y}})$ represents the error, θ represents the weights, and $\mathbf{x}$ is the input.

Corollary 3.1 (Perceptron Algorithm) *The derivation for a single-layer perceptron classifier with the thresholding function simplifies to the perceptron update rule (3.9).*

Proof sketch. Since the Heaviside step function (1.1) is an increasing function, i.e. $\alpha = 1$, our assumption on the loss function $\frac{\partial \mathcal{L}}{\partial \mathbf{s}} = \hat{\mathbf{y}} - \mathbf{y}$ indicates that,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}} = \alpha(\hat{\mathbf{y}} - \mathbf{y}) \quad \text{where} \quad \alpha = 1. \tag{3.10}$$

Therefore, the gradient of the loss function with respect to the neural network's weight θ is:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \theta} = (\hat{\mathbf{y}} - \mathbf{y})\mathbf{x}. \tag{3.11}$$

Finally, using an iterative optimization approach like gradient descent uses the derivative as below

$$\theta_{t+1} = \theta_t - \eta_t \frac{\partial \mathcal{L}}{\partial \theta} = \theta_t - \eta_t(\hat{\mathbf{y}} - \mathbf{y})\mathbf{x}. \tag{3.12}$$

Here is exactly the perceptron update rule.

Straight-Through Estimator (STE) The Straight-Through Estimator [31] is pivotal in Binarized Neural Networks [17], where weights are constrained to binary values ($\{-1, 1\}$) for memory efficiency and computational speed. Similarly, XNOR-Net [18] extends this idea to binary convolutional neural networks, using STE to backpropagate gradients through discrete activations and weights. These innovations enabled efficient implementations of deep networks on resource-constrained devices.

While STE is effective for tasks requiring discrete decision-making, it introduces challenges such as gradient mismatch between the forward and backward passes. Despite these limitations, it remains a foundational tool for handling non-differentiability in modern neural networks.

STE provides a straightforward approach to handling non-differentiable functions, such as thresholding (3.13), in neural networks:

$$\mathcal{P}\left(\mathbf{x}\right)_{\text{STE}} = \begin{cases} 1 & \text{if } \mathbf{x} \geq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{3.13}$$

As it can be seen, the derivative will be zero or undefined during back-propagation, and therefore, the gradient wont propagate anything informative, since the preceding layers' weights won't get updated. STE sets the incoming gradients to a threshold function equal to its outgoing gradients, disregarding the derivative of the threshold function itself. That is:

$$\Delta \mathbf{z}_{\text{STE}} = \mathbf{1}. \tag{3.14}$$

The Universite Update rule also corresponds to the same gradient approximation since the thresholding function (3.13), is an increasing function.

Maximum Likelihood Estimation (MLE) Although the Generalized Update method directly tackles the issue of missing or zero gradients in PFCs, its underlying principle of adjusting model parameters based on discrepancies between predictions and targets aligns closely with the essence of MLE. In MLE, model parameters are chosen to maximize the likelihood of observed data (or equivalently, to minimize the negative log-likelihood), which often reduces to updating parameters in proportion to the difference between predicted probabilities and ground-truth outcomes. This core idea of "error-based" parameter updates in MLE parallels the way Generalized Update derives its approximate gradient direction from discrepancies between $\mathcal{P}(\mathbf{z})$ and its target $\mathcal{P}(\mathbf{z})^*$. Consequently, while Generalized Update is designed to handle non-differentiable functions by enforcing monotonic consistency, it naturally resonates with MLE's principle of aligning predictions with observations through gradient-based adjustments.

Maximum Likelihood Estimation [144] statistical method to find the values of parameters θ in a probability distribution that make the observed data $\mathbf{z}$ most likely. Given data $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N$ and a model $\mathrm{F}_\theta(\mathbf{x})$, MLE optimizes:

$$\theta^\star = \arg\max_\theta \prod_{i=1}^{N} \mathrm{F}_\theta(\mathbf{x}), \tag{3.15}$$

which, equivalently, minimizes the negative log-likelihood:

$$\mathcal{L}(\theta) = -\sum_{i=1}^{N} \log \mathrm{F}_\theta(\mathbf{x}). \tag{3.16}$$

In general, there is no analytical solution of this maximization problem and a solution must be found numerically [144].

MLE plays a crucial role in modern methods, such as Implicit MLE [52], which extends MLE to discrete exponential family distributions using implicit differentiation:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial}{\partial \theta} \log Z(\theta) - \mathbb{E}_{\mathrm{F}_\theta(\mathbf{x})}\left[\frac{\partial f(x;\theta)}{\partial \theta}\right], \tag{3.17}$$

where $Z(\theta)$ is the normalization constant.

Discrete Variational Autoencoders (VAEs) [51] also incorporate MLE principles by introducing reparameterization techniques for discrete latent variables. This allows gradient-based learning of probabilistic models with structured outputs.

MLE's focus on parameter estimation underpins probabilistic approaches in machine learning, from classical models to modern generative frameworks.

Corollary 3.2 (Logistic Regression) *The Derivation for Classifier with Sigmoid Activation.*

In the logistic regression model, the output variable $\mathbf{y}_i$ is a Bernoulli random variable, and

$$\Pr(\mathbf{y}_i = 1 \mid \mathbf{x}, \theta) = \mathcal{P}_{\text{logistic}}(\mathbf{x}_i\theta) = \hat{\mathbf{y}}_i, \tag{3.18}$$

where

$$\mathcal{P}_{\text{logistic}}(t) = \frac{1}{1 + \exp(-t)}, \tag{3.19}$$

is the (sigmoid) logistic function, $\mathbf{x}_i$ is a $1 \times K$ vector of inputs, and θ is a $K \times 1$ vector of coefficients. The vector of coefficients θ is the parameter to be estimated by maximum likelihood.

Assuming that the estimation is carried out with an identical and independent distributed sample comprising N data points $(\mathbf{y}_i, \mathbf{x}_i)$ for $i = 1, \ldots, N$.

The likelihood of an observation $(\mathbf{y}_i, \mathbf{x}_i)$ can be written as

$$L(\theta; \mathbf{y}_i, \mathbf{x}_i) = [\mathcal{P}_{\text{logistic}}(\mathbf{x}_i\theta)]^{\mathbf{y}_i} \left[1 - \mathcal{P}_{\text{logistic}}(\mathbf{x}_i\theta)\right]^{1-\mathbf{y}_i} = \hat{\mathbf{y}}_i^{\mathbf{y}_i}(1 - \hat{\mathbf{y}}_i)^{1-\mathbf{y}_i}. \tag{3.20}$$

Let $\mathbf{y}$ be the $N \times 1$ vector of all outputs, and $\mathbf{x}$ be the $N \times K$ matrix of all inputs.

Since the observations are identical and independent distributed, the likelihood of the entire sample is equal to the product of the likelihoods of the single observations:

$$L(\theta; \mathbf{y}, \mathbf{x}) = \prod_{i=1}^{N} \hat{\mathbf{y}}_i^{\mathbf{y}_i}(1 - \hat{\mathbf{y}}_i)^{1-\mathbf{y}_i}. \tag{3.21}$$

The negative log-likelihood of the logistic model is

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) := -\ln L(\theta; \mathbf{y}, \mathbf{x}) = -\sum_{i=1}^{N} \Big(\mathbf{y}_i \ln \hat{\mathbf{y}}_i + (1 - \mathbf{y}_i) \ln(1 - \hat{\mathbf{y}}_i)\Big). \tag{3.22}$$

Let $\mathbf{z} = \mathbf{x}_i\theta$ to follow our conventional notation and recall $\hat{\mathbf{y}} = \mathcal{P}_{\text{logistic}}(\mathbf{z})$. Then, the gradient of negative log-likelihood with respect to $\mathbf{z}$ follows:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}} = -\sum_{i-1}^{N} \left(\frac{\mathbf{y}}{\hat{\mathbf{y}}} \frac{\partial \mathcal{P}_{\text{logistic}}}{\partial \mathbf{z}} - \frac{1-\mathbf{y}}{1-\hat{\mathbf{y}}} \frac{\partial \mathcal{P}_{\text{logistic}}}{\partial \mathbf{z}} \right). \tag{3.23}$$

since the gradient of the (sigmoid) logistic function is $\frac{\partial \mathcal{P}_{\text{logistic}}}{\partial \mathbf{z}} = \hat{\mathbf{y}}(1-\hat{\mathbf{y}})$, Equation (3.23) simplifies,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}} = -\sum_{i-1}^{N} \mathbf{y}(1-\hat{\mathbf{y}}) + (1-\mathbf{y})\hat{\mathbf{y}}. \tag{3.24}$$

Therefore, we can deduce that $\frac{\partial \mathcal{L}}{\partial \mathbf{s}} = \hat{\mathbf{y}} - \mathbf{y}$.

The foundational methods—STE, Perceptron, and MLE—are often integrated into more complex architectures to address challenges in non-differentiability, structured prediction, and probabilistic modeling. For example, Learning with Differentiable Perturbed Optimizers [44] combines MLE principles with perturbation-based gradient approximations, while Binarized Neural Networks [17] extend STE for efficient learning on discrete weight spaces.

By embedding these methodologies into differentiable layers, modern approaches enable end-to-end training across a wide range of applications, including object detection, reinforcement learning, and decision-focused tasks.

CHAPTER 4

EXPERIMENTS

This chapter evaluates the Generalized Update framework on three challenging tasks: the Warcraft Shortest Path problem, semantic segmentation using Conditional Random Fields (CRF), and object detection with DETR. In the first experiment, optimal paths are inferred from terrain maps by combining CNN-based cost predictions with Dijkstra's algorithm. The second experiment refines segmentation outputs by leveraging global contextual information. The third experiment examines the integration of DETR with a differentiable assignment method based on the Sinkhorn-Knopp algorithm for 2D object detection. Our results demonstrate that Generalized Update outperforms established baselines, highlighting its robustness and scalability across diverse optimization challenges.

4.1 Warcraft Shortest Path Problem

The Warcraft Shortest Path problem is a combinatorial optimization challenge that involves finding the minimum-cost path on a terrain map. The dataset used for this problem consists of 10,000 randomly generated maps from the Warcraft II tileset [145]. Each map represents a $k \times k$ grid, where $k \in \{12, 18, 24, 30\}$. Every vertex on the grid corresponds to a specific terrain type, each associated with a fixed traversal cost. However, these costs are not explicitly provided to the learning model; they must be inferred based on the terrain image.

Problem The task requires determining the shortest path (minimum total cost) from the top-left corner to the bottom-right corner of the grid. The true shortest path for each map is encoded as an indicator matrix, which serves as the ground truth label

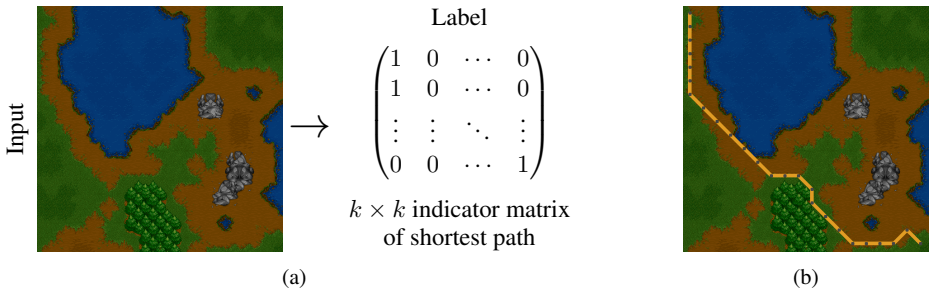

Figure 4.1: A sample terrain map from the Warcraft II tileset overlaid with the optimal shortest path from the top-left to the bottom-right corner. Each cell represents a terrain type with an inferred traversal cost, illustrating the challenge of determining the minimum-cost path in the Warcraft Shortest Path problem. [Figure taken from: [6]]

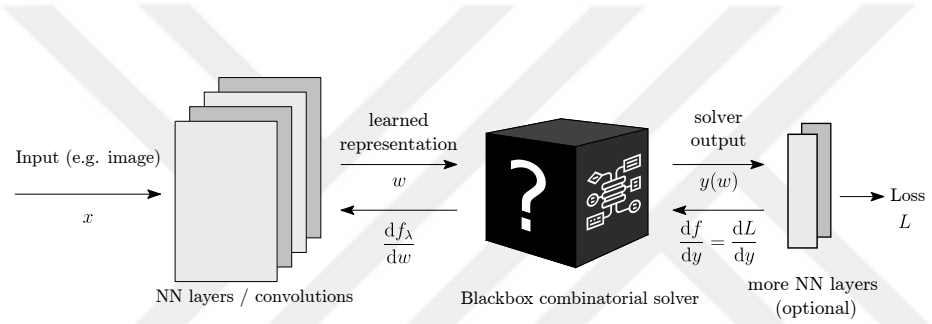

Figure 4.2: Blackbox Backprop [6] Network architecture for the Warcraft Shortest Path problem. The diagram illustrates how a CNN predicts vertex costs from terrain maps, which are then used by Dijkstra's algorithm to compute the optimal path. [Figure taken from: [6]]

for training. This problem serves as a benchmark for testing models' ability to solve complex combinatorial problems using machine learning techniques.

Network To tackle this problem, a convolutional neural network (CNN) is employed. The CNN processes the terrain map images $\mathbf{x}$ and predicts a $k \times k$ grid of vertex costs $\mathbf{z}$, representing the inferred traversal costs for each terrain type. These predicted costs are then input into Dijkstra's shortest path algorithm $\mathcal{P}_{\text{SP}}$:

$$\mathcal{P}_{\text{SP}}(\mathbf{z}) = \underset{\mathbf{s} \in \mathcal{S}_{\text{paths}}}{\text{minimize}}\, \mathbf{s}^\top \mathbf{z}, \tag{4.1}$$

which computes the shortest path across the map.

Objective The training objective is defined using the ground truth and predicted shortest paths. This loss function effectively measures the deviation of the model's predictions from the ground truth, guiding the optimization process during training:

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{i=0}^{k} \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|_1, \tag{4.2}$$

where $\hat{\mathbf{y}} = \mathcal{P}_{\text{SP}}(\mathbf{z})$. The performance of the Generalized Update method and other benchmarks is evaluated across various grid resolutions ($k \in \{12, 18, 24, 30\}$) to assess its robustness and scalability.

Table 4.1: Accuracy results for solving the Warcraft Shortest Path problem across different grid resolutions using various methods.

	Resolution			
Method	12×12	18×18	24×24	30×30
ResNet18	46.1%	0.8%	0.0%	0.0%
STE [Dijkstra] [69]	13.2%	0.2%	0.0%	0.0%
BB [Dijkstra] [6]	95.4%	94.8%	94.4%	93.6%
CombOpt [Dijkstra] [146]	95.3%	94.3%	93.5%	93.4%
UnivUpd [Dijkstra]	**96.2%**	**95.2%**	**94.5%**	**95.1%**

Results The Generalized Update approach outperforms existing methods in accuracy and generalization, as demonstrated by its high performance even on larger grid resolutions. This validates its effectiveness as a scalable and robust solution to combinatorial optimization problems, such as the Warcraft Shortest Path. The consistent performance across resolutions highlights the potential of Generalized Update in applications requiring precision and efficiency in decision-making tasks.

4.2 Semantic Segmentation Problem

The Conditional Random Fields (CRF) experiment evaluates the effectiveness of the Generalized Update framework. It uses the widely recognized PASCAL VOC 2012 segmentation dataset [7, 98], which comprises diverse images annotated with pixel-level labels for 20 object classes (i.e. categories). Figure 4.3 shows object-level and class-level segmentation on an example image in PASCAL VOC. This dataset serves

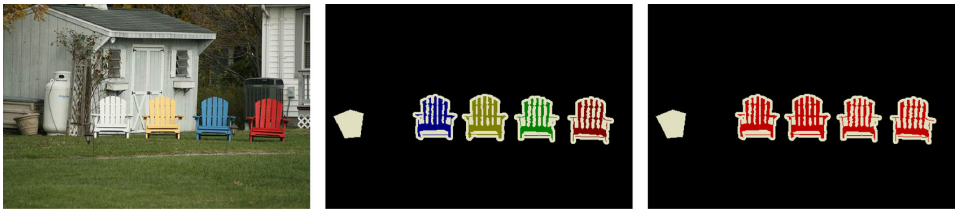

Figure 4.3: Example segmentation from the PASCAL VOC 2012 dataset. The left figure shows the original image from the PASCAL VOC 2012 segmentation dataset. The middle figure depicts the object-level segmentation where the chairs are segmented into distinct regions with colours such as blue, red, green, and yellow, while the background is marked in black. The right panel shows the class-level segmentation, where all chairs are uniformly labelled in red, and the background is depicted in black. [Figure taken from: [7]]

as a benchmark for assessing segmentation accuracy and the ability to incorporate context into predictions.

Problem The task for the Conditional Random Fields (CRF) experiment involves refining semantic segmentation outputs to achieve more precise pixel-level object categorization. Given an input image, the goal is to assign each pixel to one of several predefined object classes while maintaining consistency in spatial and contextual relationships. The CRF acts as a postprocessing step to smooth the initial segmentation predictions by considering the global image context and enforcing coherence in labelling. This task serves as a benchmark for evaluating the ability of models to integrate local predictions with global dependencies, thereby testing their effectiveness in improving segmentation accuracy through advanced optimization frameworks.

Network In this experiment, the CRF is applied as a postprocessing step to refine the initial segmentation outputs generated by a deep neural network. Due to its strong feature extraction capabilities, the baseline model is a ResNet50 [59], a widely adopted backbone for segmentation tasks. The Generalized Update framework is integrated into the CRF pipeline to improve the model's ability to capture spatial relationships and contextual dependencies in the image.

Objective The segmentation process begins with a deep neural network $F_\theta(X)$,

where $X \in \mathbb{R}^{H\times W\times 3}$ is the input image and $F_\theta(X) = \mathbf{z}$ provides initial pixel-wise label predictions, with $\mathbf{z} \in \mathbb{R}^{H\times W\times K}$ (where K is the number of classes). These predictions serve as the unary potentials for the subsequent CRF refinement. The CRF defines an energy function that integrates the network outputs with pairwise interactions between pixels:

$$E(\mathbf{s} \mid \mathbf{z}) = \sum_{i\in\mathcal{V}} \psi_u(\mathbf{s}_i, \mathbf{z}_i) + \sum_{(i,j)\in\mathcal{E}} \psi_p(\mathbf{s}_i, \mathbf{s}_j), \tag{4.3}$$

where $\mathcal{V}$ is the set of all pixels (with $\mathbf{s} \in \mathbb{R}^{|\mathcal{V}|}$) and $\mathcal{E}$ denotes the set of neighboring pixel pairs. The unary potential $\psi_u(\mathbf{s}_i, \mathbf{z}_i)$ leverages the network prediction for pixel i, while the pairwise potential $\psi_p(\mathbf{s}_i, \mathbf{s}_j)$ enforces spatial and contextual consistency between adjacent pixels. The final segmentation is obtained by solving the following minimization problem:

$$\mathcal{P}(\mathbf{z}) = \underset{\mathbf{s}\in\mathbb{R}^{|\mathcal{V}|}}{\text{minimize}} \quad E(\mathbf{s} \mid \mathbf{z}). \tag{4.4}$$

The refined segmentation $\mathbf{s}^*$ is then evaluated using the Mean Intersection-over-Union (Mean IU) metric, which quantifies the overlap between the refined segmentation and the ground truth labels.

Table 4.2: Mean IU accuracy results on segmentation with Conditional Random Field.

Method	CRF Postprocess
ResNet50 [59]	77.70%
CRFasRNN [147]	74.40%
UnivUpd [CRF]-Estim.	70.37%
UnivUpd [CRF]	**78.35**%

Evaluation The performance results are summarized in Table 4.2. The Generalized Update-enhanced CRF achieves a Mean IU of 78.35%, outperforming the baseline ResNet50 (77.70%) and CRFasRNN (74.40%) models. These results demonstrate that the Generalized Update framework effectively improves the segmentation quality by leveraging its capacity.

Notably, the Generalized Update framework surpasses traditional CRF postprocessing techniques and provides a more consistent and robust approach to segmentation

refinement. This improvement highlights Generalized Update's adaptability to various tasks and datasets, establishing its value as a versatile tool for semantic segmentation tasks.

4.3 COCO Object Detection Problem with DETR

This experiment explores the integration of the Sinkhorn-Knopp algorithm (Alg. 1) into the assignment step of DETR [5] for the 2D object detection problem. By replacing the traditional, non-differentiable, Hungarian matching method with a differentiable optimal transport-based approach, we aim to enhance assignment stability and overall detection performance on complex scenes from the COCO dataset [148].

Problem The task for object detection involves assigning a set of predicted anchors $\mathcal{A} = \{a^{(i)}\}_{i=1}^{N} = \{(\hat{c}_i, \hat{\mathbf{b}}_i)\}_{i=1}^{N}$ to the corresponding ground truth objects $\mathcal{G} = \{g^{(j)}\}_{j=1}^{M} = \{(c_j, \mathbf{b}_j)\}_{j=1}^{M}$. As introduced in Section 2.2.1, each predicted anchor $a^{(i)}$ comprises a class label $\hat{c}_i \in \mathcal{C} = \{c_1, c_2, \ldots, c_K, c_{\text{bg}}\}$, where c_{bg} denotes the background class, and a bounding box $\hat{\mathbf{b}}_i \in \mathbb{R}^4$, parameterized as (x_i, y_i, w_i, h_i). Therefore, The fundamental problem is to perform a one-to-one matching between the set of predicted anchors and a subset of ground truths.

Traditional DETR [5] employs Hungarian matching algorithm [149] to compute the one-to-one assignment matrix $\mathbf{S}$, that minimizes the pairwise assignment cost matrix $\mathbf{C} \in \mathbb{R}^{N \times M}$, but this approach is differentially intractable and can stall end-to-end training. In this experiment, we replace Hungarian matching with the Sinkhorn-Knopp algorithm (Alg. 1), an optimal transport-based method designed to yield a differentiable and robust assignment, ultimately improving detection performance on challenging scenes from the COCO dataset [148].

Network A transformer-based object detector workflow can be broken into three main stages: feature extraction, transformer encoding and decoding, and object prediction.

During feature extraction stage, the input image $I \in \mathbb{R}^{H \times W \times 3}$ is processed by a CNN backbone, such as ResNet, to extract feature maps $\mathbf{F} \in \mathbb{R}^{H_f \times W_f \times D}$, where H_f and

W_f are the height and width of the feature map, and D is the feature depth. These feature maps are flattened into a sequence of vectors $\mathbf{X} \in \mathbb{R}^{(H_f \cdot W_f) \times D}$, which serve as input to the transformer. Positional encodings $\mathbf{P} \in \mathbb{R}^{(H_f \cdot W_f) \times D}$ are added to $\mathbf{X}$ to retain spatial information:

$$\mathbf{X}_{\text{input}} = \mathbf{F} + \mathbf{P}. \tag{4.5}$$

In the transformer encoding and decoding stage, the transformer encoder processes $\mathbf{X}_{\text{input}}$ using multi-head self-attention and feed-forward layers to capture global context across the image. The output of the encoder is a set of encoded features $\mathbf{Z}_e \in \mathbb{R}^{(H_f \cdot W_f) \times D}$, where:

$$\mathbf{Z}_e = \text{softmax}\left(\frac{\mathbf{Q}_e \cdot \mathbf{K}_e^\top}{\sqrt{D}}\right) \cdot \mathbf{V}_e, \tag{4.6}$$

and $\mathbf{Q}_e, \mathbf{K}_e, \mathbf{V}_e$ represent the query, key, and value matrices derived from the input sequence.

The decoder takes as input a fixed set of N learnable object queries $\mathbf{Q}_d \in \mathbb{R}^{N \times D}$, which interact with the encoded features through cross-attention:

$$\mathbf{Z}_d = \text{softmax}\left(\frac{\mathbf{Q}_d \cdot \mathbf{K}_e^\top}{\sqrt{D}}\right) \cdot \mathbf{V}_e. \tag{4.7}$$

The decoder refines these queries over multiple layers, each predicting intermediate object representations.

Finally, in the object prediction stage, the refined queries $\mathbf{Z}_d$ are mapped to object predictions using two parallel feed-forward networks (FFNs): A classification head predicts each query's class label c_i. A regression head predicts the bounding box $\mathbf{b}_i$, parameterized as (x_i, y_i, w_i, h_i).

Objective During training, the following objective function is minimized:

$$\mathcal{L}\left(\left\{a^{(i)}\right\}_{i=0}^{N}, \left\{g^{(\sigma(i))}\right\}_{i=0}^{N}\right) = \sum_{i=1}^{N} \mathcal{L}_{\text{cls}}\left(c_i, \hat{c}_{\sigma(i)}\right) + \lambda \mathcal{L}_{\text{loc}}\left(\mathbf{b}_i, \hat{\mathbf{b}}_{\sigma(i)}\right), \tag{4.8}$$

where $\mathcal{L}_{\text{cls}}$ denotes the classification loss, $\mathcal{L}_{\text{loc}}$ represents the localization loss, and λ is a balancing hyperparameter. Here, σ is a permutation function that defines the optimal assignment between the N predicted anchors and the ground truth objects. In other words, $\sigma(i)$ indicates the index of the ground truth object that is matched to the ith prediction. This optimal assignment is typically determined by a matching algorithm

(e.g., Hungarian matching or an optimal transport-based approach), ensuring a one-to-one correspondence between predictions and ground truth, which is crucial for correctly computing the loss during training.

In DETR [5], the matching between a predicted anchor $a^{(i)}$ and a ground truth object $g^{(j)}$ is quantified by a combined loss $\mathcal{L}$ defined as a weighted sum of three components:

$$\mathcal{L}_{\text{DETR}} = \lambda_{\text{cls}}\,\mathcal{L}_{\text{cls}} + \lambda_{\text{bbox}}\,\mathcal{L}_{\text{bbox}} + \lambda_{\text{iou}}\,\mathcal{L}_{\text{iou}}. \tag{4.9}$$

The classification loss encourages correct class prediction by penalizing low probability for the target class ℓ_j. It is defined as

$$\mathcal{L}_{\text{cls}}(a^{(i)}, g^{(j)}) = -\log\left(\mathbf{p}^{(i)}_{\ell_j}\right). \tag{4.10}$$

The bounding box loss measures the discrepancy between the predicted bounding box $\mathbf{b}^{(i)}$ and the ground truth bounding box $\mathbf{b}^{(j)}$ using the L1 norm, thereby promoting precise localization:

$$\mathcal{L}_{\text{bbox}}(a^{(i)}, g^{(j)}) = \|\mathbf{b}^{(i)} - \mathbf{b}^{(j)}\|_1. \tag{4.11}$$

The IoU loss is designed to enhance the spatial overlap between the predicted and ground truth boxes and is formulated as follows:

$$\mathcal{L}_{\text{iou}}(a^{(i)}, g^{(j)}) = 1 - \text{IoU}(\mathbf{b}^{(i)}, \mathbf{b}^{(j)}). \tag{4.12}$$

Together, these unified loss components form the overall loss $\mathcal{L}$, guiding the optimal matching process during training by addressing classification and localization in a coherent framework. The overall goal is to enhance detection accuracy and convergence stability, as measured by the mean Average Precision (mAP) on the COCO dataset.

Results Figure 4.4 reveals a significant performance gap between the traditional DETR [5] using Hungarian matching [149] and the Sinkhorn-Knopp [130] assignment approach. The baseline DETR attains nearly 25 mAP within 24 epochs, demonstrating rapid convergence and effective matching. In contrast, our Sinkhorn-Knopp-based approach remains stalled at approximately 13-14 mAP even after 36 epochs, maintaining this suboptimal performance for over 10. This discrepancy suggests that the optimal transport-based assignment while offering differentiability, may suffer from reduced versatility and a coarse approximation of the matching process. These

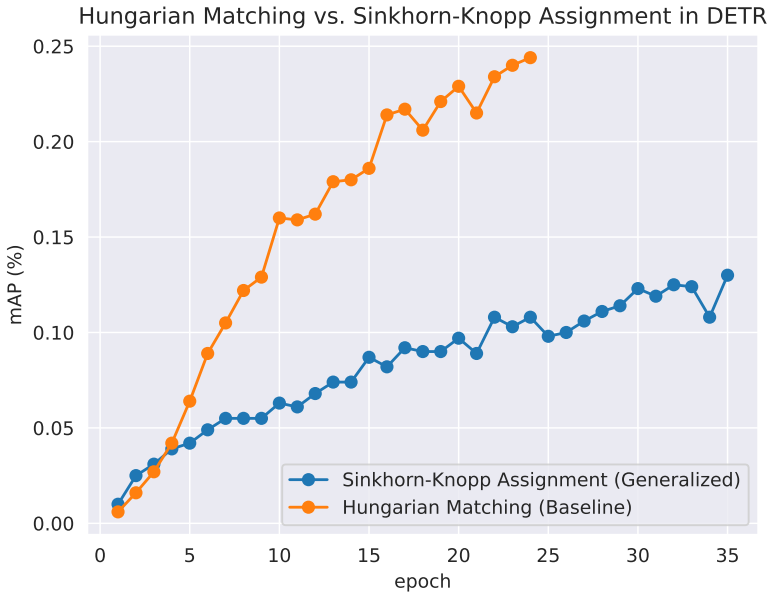

Figure 4.4: The comparison of the mean Average Precision (mAP) over training epochs for DETR using two different assignment strategies: the traditional Hungarian matching and the Sinkhorn-Knopp algorithm.

limitations likely contribute to the slower convergence and lower detection accuracy observed. Future work should focus on refining the Sinkhorn-Knopp integration—through improved parameter tuning or enhanced approximation techniques—to better harness its theoretical benefits in end-to-end object detection scenarios.

4.4 Summary

Performance Analysis The experimental results show that the Generalized Update framework works effectively across different tasks. In the Warcraft Shortest Path problem, it consistently achieved higher accuracy across all grid resolutions. In semantic segmentation, the Generalized Update-enhanced CRF reached the highest Mean IU accuracy on the PASCAL VOC 2012 dataset. However, in the DETR experiment, while the Hungarian matching baseline model reached nearly 25 mAP within 24 epochs, the Sinkhorn-Knopp approach remained around 13-14 mAP even after 36 epochs. This indicates that the Sinkhorn-Knopp method may not accurately approxi-

mate the matching process in this context.

Scalability and Robustness The framework shows good scalability and robustness in some tasks. Its performance stayed stable in the shortest path problem even when the grid size increased. Similarly, the framework effectively integrated contextual information in semantic segmentation to improve object boundaries. On the other hand, the DETR experiment reveals that the Sinkhorn-Knopp approach may have limitations in handling complex object detection scenarios, as evidenced by its lower and stagnant mAP values.

Generalization Across Problems A major advantage of the Generalized Update framework is its potential to generalize across various tasks. While our experiments focused on the shortest path problem, semantic segmentation, and 2D object detection with DETR, the framework's underlying principles can be applied to other areas, such as portfolio optimization, scheduling, and other combinatorial problems. Its ability to integrate with different models makes it a promising tool for many applications that require efficient and accurate solutions.

CHAPTER 5

CONCLUSION

In this thesis, we focused on integrating discrete decision-making processes, such as the PFC layer, with gradient-based iterative optimization techniques in deep learning. This integration is essential for developing models capable of handling complex tasks that require structured predictions and flexible learning.

5.1 Summary of Contributions and Results

The Generalized Update framework tackles the challenge of incorporating PFC layer components —including methods such as Dijkstra's algorithm and Conditional Random Fields (CRFs)— into neural networks. By approximating gradients for piecewise linear functions with constant regions (PFCs), our approach enables end-to-end training while preserving the behaviour of combinatorial solvers.

As seen in Table 4.1, our method achieved state-of-the-art performance in the Warcraft Shortest Path problem across various grid resolutions, outperforming ResNet, STE, and CombOpt. For example, our approach reached 96.2% accuracy on 12×12 grids and maintained 95.1% accuracy on 30×30 grids, demonstrating good scalability. In semantic segmentation, integrating CRFs via Generalized Updates improved the Mean IU on the PASCAL VOC 2012 dataset to 78.35%, surpassing the performance of both ResNet50 (77.70%) and CRFasRNN (74.40%) baselines (Table 4.2).

However, our experiments with DETR revealed a different outcome. When replacing Hungarian matching with the Sinkhorn-Knopp algorithm for object assignment, the detection performance did not improve. As plotted in Figure 4.4, the baseline DETR

achieved nearly 25 mAP at 24 epochs, while the Sinkhorn-Knopp method stalled at around 13–14 mAP even after 36 epochs. This suggests that the differentiable assignment provided by the Sinkhorn-Knopp algorithm may offer a coarser approximation and lacks the versatility needed for optimal performance in this context.

5.2 Limitations and Future Work

Despite these advancements, our proposed methods have limitations. The Sinkhorn-Knopp algorithm introduces additional computational overhead due to its iterative matrix operations, which can slow down training in high-dimensional scenarios. Additionally, the Generalized Update framework requires careful tuning of hyperparameters to ensure stable gradient approximations, particularly in environments with noisy or sparse data. Our evaluations have so far focused primarily on computer vision tasks; broader validation in areas such as natural language processing or reinforcement learning is needed.

Future work should concentrate on optimizing the computational efficiency of these methods-potentially through sparsity-aware implementations of the Sinkhorn-Knopp algorithm or adaptive regularization schedules. Extending the Generalized Update framework to additional domains, such as graph-based tasks or sequence modeling, would further demonstrate its generalizability. Addressing these challenges will advance the integration of discrete and continuous learning paradigms and help develop more adaptable and efficient models for real-world applications.

REFERENCES

[1] A. Paulus, M. Rolinek, V. Musil, B. Amos, and G. Martius, "Comboptnet: Fit the right np-hard problem by learning integer programming constraints," in *Proceedings of the 38th International Conference on Machine Learning* (M. Meila and T. Zhang, eds.), vol. 139 of *Proceedings of Machine Learning Research*, pp. 8443–8453, PMLR, 18–24 Jul 2021.

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 580–587, 2014.

[4] R. Girshick, "Fast r-cnn," in *International Conference on Computer Vision (ICCV)*, pp. 1440–1448, 2015.

[5] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European Conference on Computer Vision (ECCV)*, pp. 213–229, Springer, 2020.

[6] M. V. Pogančić, A. Paulus, V. Musil, G. Martius, and M. Rolinek, "Differentiation of blackbox combinatorial solvers," in *International Conference on Learning Representations (ICLR)*, 2020.

[7] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, pp. 98–136, Jan. 2015.

[8] L. Chen, T. Yang, X. Zhang, W. Zhang, and J. Sun, "Psvit: Better vision transformer via token pooling and attention sharing," in *Proceedings of the*

IEEE/CVF International Conference on Computer Vision, pp. 10449–10458, IEEE, 2021.

[9] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," 2021.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017.

[11] F. Stahlberg, "Neural machine translation: A review," *Journal of Artificial Intelligence Research*, vol. 69, pp. 343–418, 2020.

[12] G. Zhang, F. Li, X. Zhang, W. Liu, Z. Li, and J. Sun, "Semantic-aligned fusion transformer for one-shot object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1–10, 2022.

[13] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," *arXiv preprint arXiv:1602.02410*, 2016.

[14] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: concepts, cnn architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, mar 2021.

[15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[16] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[17] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in Neural Information Processing Systems* (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), vol. 29, Curran Associates, Inc., 2016.

[18] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision (ECCV)*, pp. 525–542, Springer, 2016.

[19] M. V. Pogančić, A. Paulus, V. Musil, G. Martius, and M. Rolinek, "Differentiation of blackbox combinatorial solvers," in *International Conference on Learning Representations*, 2020.

[20] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport," *Advances in Neural Information Processing Systems*, vol. 26, pp. 2292–2300, 2013.

[21] K. Oksuz, B. C. Cam, E. Akbas, and S. Kalkan, "Rank & sort loss for object detection and instance segmentation," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2989–2998, 2021.

[22] M. Rolinek, V. Musil, A. Paulus, M. Vlastelica, C. Michaelis, and G. Martius, "Optimizing rank-based metrics with blackbox differentiation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[23] C. Huang, S. Zhai, P. Guo, and J. M. Susskind, "Metricopt: Learning to optimize black-box evaluation metrics," *CoRR*, vol. abs/2104.10631, 2021.

[24] A. Ferber, B. Wilder, B. Dilkina, and M. Tambe, "Mipaal: Mixed integer program as a layer," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, p. 1504–1511, apr 2020.

[25] P. Donti, B. Amos, and J. Z. Kolter, "Task-based end-to-end model learning in stochastic optimization," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[26] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization*. Athena Scientific, 1997.

[27] M. Cuturi, O. Teboul, and J.-P. Vert, "Differentiable ranking and sorting using optimal transport," in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[28] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and Z. Kolter, "Differentiable convex optimization layers," in *Advances in Neural Information Processing Systems*, 2019.

[29] F. Rosenblatt, "The perceptron, a perceiving and recognizing automaton : $projectpara$, cornell aeronautical laboratory report," 1957.

[30] K. Chen, J. Li, W. Lin, J. See, J. Wang, L. Duan, Z. Chen, C. He, and J. Zou, "Towards accurate one-stage object detection with ap-loss," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5119–5127, 2019.

[31] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint*, 2013.

[32] Y. Song, A. G. Schwing, R. S. Zemel, and R. Urtasun, "Training deep neural networks via direct loss minimization," in *Proceedings of The 33rd International Conference on Machine Learning* (M. F. Balcan and K. Q. Weinberger, eds.), vol. 48 of *Proceedings of Machine Learning Research*, (New York, New York, USA), pp. 2169–2177, PMLR, 20–22 Jun 2016.

[33] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[34] B. Hanin, "Neural networks and the curse of dimensionality," *arXiv preprint*, 2018.

[35] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[36] S. Gould, B. Fernando, A. Cherian, P. Anderson, R. S. Cruz, and E. Guo, "On differentiating parameterized argmin and argmax problems with application to bi-level optimization," *ArXiv*, vol. abs/1607.05447, 2016.

[37] A. Grover, E. Wang, A. Zweig, and S. Ermon, "Stochastic optimization of sorting networks via continuous relaxations," in *International Conference on Learning Representations*, 2019.

[38] M. Blondel, O. Teboul, Q. Berthet, and J. Djolonga, "Fast differentiable sorting and ranking," in *Proceedings of the 37th International Conference on Machine Learning* (H. D. III and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 950–959, PMLR, 13–18 Jul 2020.

[39] F. Petersen, H. Kuehne, C. Borgelt, and O. Deussen, "Differentiable top-k classification learning," in *Proceedings of the 39th International Conference on Machine Learning*, pp. 17656–17668, PMLR, 2022.

[40] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr, "Conditional random fields as recurrent neural networks," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1529–1537, 2015.

[41] X. He, R. Zemel, and M. Carreira-Perpinan, "Multiscale conditional random fields for image labeling," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 2, pp. II–II, 2004.

[42] A. N. Elmachtoub and P. Grigas, "Smart "predict, then optimize"," *Management Science*, vol. 68, p. 9–26, jan 2022.

[43] J. Mandi, E. Demirovi?, P. J. Stuckey, and T. Guns, "Smart predict-and-optimize for hard combinatorial optimization problems," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, p. 1603–1610, apr 2020.

[44] Q. Berthet, M. Blondel, O. Teboul, M. Cuturi, J.-P. Vert, and F. Bach, "Learning with differentiable perturbed optimizers," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), vol. 33, pp. 9508–9519, Curran Associates, Inc., 2020.

[45] B. Wilder, B. Dilkina, and M. Tambe, "Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, p. 1658–1665, jul 2019.

[46] A. S. Pinto, A. Kolesnikov, Y. Shi, L. Beyer, and X. Zhai, "Tuning computer vision models with task rewards," 2023.

[47] Q. Qian, L. Chen, H. Li, and R. Jin, "Dr loss: Improving object detection by distributional ranking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[48] B. Amos and J. Z. Kolter, "OptNet: Differentiable optimization as a layer in neural networks," in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 136–145, PMLR, 06–11 Aug 2017.

[49] P. J. Rousseeuw, "Least median of squares regression," *Journal of the American Statistical Association*, vol. 79, pp. 871–880, 1984.

[50] C. Rudin, "Learning to rank with nonsmooth cost functions," in *Advances in Neural Information Processing Systems*, pp. 1233–1240, 2008.

[51] J. T. Rolfe, "Discrete variational autoencoders," in *International Conference on Learning Representations*, 2017.

[52] M. Niepert, P. Minervini, and L. Franceschi, "Implicit mle: Backpropagating through discrete exponential family distributions," in *Advances in Neural Information Processing Systems* (M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, eds.), vol. 34, pp. 14567–14579, Curran Associates, Inc., 2021.

[53] M. Rolínek, V. Musil, A. Paulus, M. V. P., C. Michaelis, and G. Martius, "Optimizing rank-based metrics with blackbox differentiation," *CoRR*, vol. abs/1912.03500, 2019.

[54] G. Monge, "Mémoire sur la théorie des déblais et des remblais," *Histoire de l'Académie Royale des Sciences de Paris, avec les Mémoires de Mathématique et de Physique pour la même année*, pp. 666–704, 1781.

[55] L. Kantorovich, "On the translocation of masses," *C.R. (Doklady) Acad. Sci. URSS (N.S.)*, vol. 37, pp. 199–201, 1942.

[56] C. Villani, *Topics in Optimal Transportation*. American Mathematical Society, 2003.

[57] C. Villani, *Optimal Transport: Old and New*. Berlin, Germany: Springer, 2008.

[58] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

[59] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[60] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[61] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 39, no. 6, pp. 1137–1149, 2017.

[62] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[63] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," in *The European Conference on Computer Vision (ECCV)*, 2016.

[64] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.

[65] X. Chen, R. Girshick, K. He, and P. Dollár, "Tensormask: A foundation for dense object segmentation," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[66] W. Medhat, A. Hassan, and H. Korashy, "Sentiment analysis algorithms and applications: A survey," *Ain Shams engineering journal*, vol. 5, no. 4, pp. 1093–1113, 2014.

[67] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Optimal and autonomous control using reinforcement learning: A survey," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 6, pp. 2042–2062, 2017.

[68] H. Nguyen and H. La, "Review of deep reinforcement learning for robot manipulation," in *2019 Third IEEE international conference on robotic computing (IRC)*, pp. 590–595, IEEE, 2019.

[69] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013.

[70] A. H. Land and A. G. Doig, "An automatic method of solving discrete programming problems," *Econometrica: Journal of the Econometric Society*, pp. 497–520, 1960.

[71] P.-W. Wang, P. Donti, B. Wilder, and Z. Kolter, "SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 6545–6554, PMLR, 09–15 Jun 2019.

[72] M. Imfeld, J. Graldi, M. Giordano, T. Hofmann, S. Anagnostidis, and S. P. Singh, "Transformer fusion with optimal transport," 2024.

[73] M. Blondel, O. Teboul, Q. Berthet, and J. Djolonga, "Fast differentiable sorting and ranking," in *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 950–959, 2020.

[74] M. Cuturi, O. Teboul, and J.-P. Vert, "Differentiable ranks and sorting using optimal transport," in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1–11, 2019.

[75] A. Butler and R. Kwon, "Efficient differentiable quadratic programming layers: an admm approach," 2021.

[76] A. Agrawal, S. Barratt, S. Boyd, E. Busseti, and W. M. Moursi, "Differentiating through a cone program," 2020.

[77] X. Gao, H. Zhang, A. Panahi, and T. Arodz, "Differentiable combinatorial losses through generalized gradients of linear programs," 2021.

[78] J. Domke, "Generic methods for optimization-based modeling," in *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics* (N. D. Lawrence and M. Girolami, eds.), vol. 22 of *Proceedings of Machine Learning Research*, (La Palma, Canary Islands), pp. 318–326, PMLR, 21–23 Apr 2012.

[79] Y. Fan, S. Lyu, Y. Ying, and B.-G. Hu, "Learning with average top-k loss," 2017.

[80] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), vol. 37 of *Proceedings of Machine Learning Research*, (Lille, France), pp. 1889–1897, PMLR, 07–09 Jul 2015.

[81] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, p. 229–256, may 1992.

[82] M. F. Tappen, C. Liu, E. H. Adelson, and W. T. Freeman, "Learning gaussian conditional random fields for low-level vision," *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2007.

[83] X. Wu, D. Sahoo, and S. C. Hoi, "Recent advances in deep learning for object detection," *Neurocomputing*, vol. 396, pp. 39–64, 2020.

[84] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proceedings of the IEEE*, vol. 111, no. 3, pp. 257–276, 2023.

[85] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[86] R. Padilla, S. L. Netto, and E. A. Da Silva, "A survey on performance metrics for object-detection algorithms," in *2020 international conference on systems, signals and image processing (IWSSIP)*, pp. 237–242, IEEE, 2020.

[87] D. Shen, G. Wu, and H.-I. Suk, "Deep learning in medical image analysis," *Annual review of biomedical engineering*, vol. 19, no. 1, pp. 221–248, 2017.

[88] K. Suzuki, "Overview of deep learning in medical imaging," *Radiological physics and technology*, vol. 10, no. 3, pp. 257–273, 2017.

[89] G. Sreenu and S. Durai, "Intelligent video surveillance: a review through deep learning techniques for crowd analysis," *Journal of Big Data*, vol. 6, no. 1, pp. 1–27, 2019.

[90] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893 vol. 1, 2005.

[91] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[92] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, p. 318–327, 2020.

[93] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 658–666, 2019.

[94] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-iou loss: Faster and better learning for bounding box regression," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, p. 12993–13000, Apr. 2020.

[95] K. Chen, W. Lin, J. Li, J. See, and J. Wang, "Ap-loss for accurate one-stage object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020.

[96] K. Oksuz, B. C. Cam, E. Akbas, and S. Kalkan, "Localization recall precision (lrp): A new performance metric for object detection," in *Proceedings of the European Conference on Computer Vision (ECCV)*, p. 68–86, 2021.

[97] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision (ECCV)*, pp. 740–755, 2014.

[98] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[99] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3213–3223, 2016.

[100] A. Gupta, P. Dollar, R. Girshick, K. He, B. H. Dollár, and C. L. Zitnick, "Lvis: A dataset for large vocabulary instance segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5356–5364, 2019.

[101] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, *et al.*, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *International Journal of Computer Vision*, vol. 128, no. 7, pp. 1956–1981, 2020.

[102] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3354–3361, 2012.

[103] S. Shao, Z. Li, T. Zhang, C. Peng, G. Yu, X. Zhang, J. Li, and J. Sun, "Objects365: A large-scale, high-quality dataset for object detection," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8429–8438, 2019.

[104] G.-S. Xia, X. Bai, J. Ding, Z. Zhu, S. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang, "Dota: A large-scale dataset for object detection in aerial images," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 42, no. 10, pp. 2452–2469, 2018.

[105] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[106] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[107] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European Conference on Computer Vision (ECCV)*, pp. 21–37, 2016.

[108] S. Zhang, C. Chi, Y. Yao, Z. Lei, and S. Z. Li, "Atss: Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection," 2020.

[109] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 91–99, 2015.

[110] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-cnn," in *International Conference on Computer Vision (ICCV)*, pp. 2961–2969, 2017.

[111] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6154–6162, 2018.

[112] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *European Conference on Computer Vision (ECCV)*, pp. 734–750, 2018.

[113] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 6568–6577, 2019.

[114] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," in *International Conference on Computer Vision (ICCV)*, pp. 9627–9636, 2019.

[115] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," in *International Conference on Learning Representations (ICLR)*, 2021.

[116] D. Meng, X. Chen, Z. Fan, G. Zeng, H. Li, Y. Yuan, L. Sun, J. Wang, L. Zhang, and H. Hu, "Conditional detr for fast training convergence," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10469–10478, IEEE, 2021.

[117] B. Roh, J. Shin, W. Shin, and S. Kim, "Sparse detr: Efficient end-to-end object detection with learnable sparsity," in *ICLR*, 2022.

[118] H. Zhang, F. Li, S. Liu, L. Zhang, H. Su, J. Zhu, L. M. Ni, and H.-Y. Shum, "DINO: DETR with improved denoising anchor boxes for end-to-end object detection," in *International Conference on Learning Representations*, 2022.

[119] E. F. Montesuma, F. N. Mboula, and A. Souloumiac, "Recent advances in optimal transport for machine learning," 2024.

[120] F. Santambrogio, *Optimal Transport for Applied Mathematicians: Calculus of Variations, PDEs, and Modeling*. Progress in Nonlinear Differential Equations and Their Applications, Springer International Publishing, 2015.

[121] Y. Brenier, "Polar factorization and monotone rearrangement of vector-valued functions," *Communications on Pure and Applied Mathematics*, vol. 44, pp. 375–417, 1991.

[122] S. Liu, S. Ma, Y. Chen, H. Zha, and H. Zhou, "Learning high dimensional wasserstein geodesics," 2021.

[123] M. Cuturi and A. Doucet, "Fast computation of wasserstein barycenters," 2014.

[124] J. Solomon, F. de Goes, G. Peyré, M. Cuturi, A. Butscher, A. Nguyen, T. Du, and L. Guibas, "Convolutional wasserstein distances: efficient optimal transportation on geometric domains," *ACM Trans. Graph.*, vol. 34, July 2015.

[125] B. Levy and E. Schwindt, "Notions of optimal transport theory and how to implement them on a computer," 2017.

[126] G. Peyré and M. Cuturi, "Computational optimal transport," 2020.

[127] R. Flamary, N. Courty, A. Gramfort, M. Z. Alaya, A. Boisbunon, S. Chambon, L. Chapel, A. Corenflos, K. Fatras, N. Fournier, L. Gautheron, N. T. Gayraud,

H. Janati, A. Rakotomamonjy, I. Redko, A. Rolet, A. Schutz, V. Seguy, D. J. Sutherland, R. Tavenard, A. Tong, and T. Vayer, "Pot: Python optimal transport," *Journal of Machine Learning Research*, vol. 22, no. 78, pp. 1–8, 2021.

[128] M. Cuturi, L. Meng-Papaxanthos, Y. Tian, C. Bunne, G. Davis, and O. Teboul, "Optimal transport tools (ott): A jax toolbox for all things wasserstein," *arXiv preprint arXiv:2201.12324*, 2022.

[129] G. B. Dantzig, A. Orden, and P. Wolfe, *Notes on Linear Programming: Part 1. The Generalized Simplex Method for Minimizing a Linear Form under Linear Inequality Restraints*. RAND Corporation, 1954.

[130] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport," in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2292–2300, 2013.

[131] A. Genevay, G. Peyre, and M. Cuturi, "Learning generative models with sinkhorn divergences," in *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics* (A. Storkey and F. Perez-Cruz, eds.), vol. 84 of *Proceedings of Machine Learning Research*, pp. 1608–1617, PMLR, 09–11 Apr 2018.

[132] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Scholkopf, and A. Smola, "A kernel approach to comparing distributions," in *AAAI Conference on Artificial Intelligence*, 2007.

[133] G. Luise, A. Rudi, M. Pontil, and C. Ciliberto, "Differential properties of sinkhorn approximation for learning with wasserstein distance," 2018.

[134] A. Genevay, L. Chizat, F. Bach, M. Cuturi, and G. Peyré, "Sample complexity of sinkhorn divergences," in *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics* (K. Chaudhuri and M. Sugiyama, eds.), vol. 89 of *Proceedings of Machine Learning Research*, pp. 1574–1583, PMLR, 16–18 Apr 2019.

[135] D. P. Bertsekas, "The auction algorithm: A distributed relaxation method for the assignment problem," *Annals of Operations Research*, vol. 14, p. 105–123, Dec. 1988.

[136] B. Schmitzer, "A sparse multiscale algorithm for dense optimal transport," *Journal of Mathematical Imaging and Vision*, vol. 56, p. 238–259, Apr. 2016.

[137] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3–4, p. 229–256, 1992.

[138] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in Neural Information Processing Systems* (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), vol. 29, Curran Associates, Inc., 2016.

[139] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1," 2016.

[140] M. Rolínek, P. Swoboda, D. Zietlow, A. Paulus, V. Musil, and G. Martius, "Deep graph matching via blackbox differentiation of combinatorial solvers," in *Computer Vision – ECCV 2020* (A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, eds.), (Cham), pp. 407–424, Springer International Publishing, 2020.

[141] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, pp. 65–386, 1958.

[142] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. De Jesús, *Neural Network Design*. Martin Hagan, 2nd ed., 2014.

[143] B. Widrow and M. Lehr, "30 years of adaptive neural networks: perceptron, madaline, and backpropagation," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1415–1442, 1990.

[144] J. Aldrich, "R. a. fisher and the making of maximum likelihood 1912-1922," *Statistical Science*, vol. 12, no. 3, pp. 162–176, 1997.

[145] F. Guyomarch, "War2edit." https://github.com/war2/war2edit, 2017.

[146] S. S. Sahoo, A. Paulus, M. Vlastelica, V. Musil, V. Kuleshov, and G. Martius, "Backpropagation through combinatorial algorithms: Identity with projection

works," in *The Eleventh International Conference on Learning Representations*, 2023.

[147] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr, "Conditional random fields as recurrent neural networks," *CoRR*, vol. abs/1502.03240, 2015.

[148] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *The European Conference on Computer Vision (ECCV)*, 2014.

[149] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, p. 83–97, Mar. 1955.

APPENDICES