# LOSS FUNCTIONS AND METRICS IN DEEP LEARNING

Published in Springer Artificial Intelligence Review as:
A comprehensive survey of loss functions and metrics in deep learning
https://doi.org/10.1007/s10462-025-11198-7
PLEASE CITE THE PUBLISHED VERSION

Juan R. Terven CICATA-Qro Instituto Politecnico Nacional Mexico jrtervens@ipn.mx Oiana M. Cordova-Esparza
Facultad de Informática
Universidad Autónoma de Querétaro
Mexico
diana.cordova@uaq.mx

O Alfonso Ramirez-Pedraza Visión Robótica
Centro de Investigaciones en Óptica A.C.
Mexico
pedro.ramirez@cio.mx

© Edgar A. Chavez-Urbiola CICATA-Qro Instituto Politecnico Nacional Mexico eachavezu@ipn.mx • Julio A. Romero-Gonzalez
Facultad de Informática
Universidad Autónoma de Querétaro
Mexico
julio.romero@uaq.mx

April 15, 2025

#### ABSTRACT

This paper presents a comprehensive review of loss functions and performance metrics in deep learning, highlighting key developments and practical insights across diverse application areas. We begin by outlining fundamental considerations in classic tasks such as regression and classification, then extend our analysis to specialized domains like computer vision and natural language processing including retrieval-augmented generation. In each setting, we systematically examine how different loss functions and evaluation metrics can be paired to address task-specific challenges such as class imbalance, outliers, and sequence-level optimization. Key contributions of this work include: (1) a unified framework for understanding how losses and metrics align with different learning objectives, (2) an in-depth discussion of multi-loss setups that balance competing goals, and (3) new insights into specialized metrics used to evaluate modern applications like retrieval-augmented generation, where faithfulness and context relevance are pivotal. Along the way, we highlight best practices for selecting or combining losses and metrics based on empirical behaviors and domain constraints. Finally, we identify open problems and promising directions, including the automation of loss-function search and the development of robust, interpretable evaluation measures for increasingly complex deep learning tasks. Our review aims to equip researchers and practitioners with clearer guidance in designing effective training pipelines and reliable model assessments for a wide spectrum of real-world applications.

**Keywords** Deep learning · loss functions · performance metrics · computer vision · natural language processing · retrieval augmented generation

### **Contents**

1	Intro	oduction	6
	1.1	Loss Functions vs. Performance Metrics	6
	1.2	Properties of Loss Functions	7

	1.3	Implen	nentation Considerations and Software Libraries	7
2	Reg	ression		11
	2.1	Regres	sion Loss Functions	11
		2.1.1	Mean Squared Error (MSE)	11
		2.1.2	Mean Absolute Error (MAE)	13
		2.1.3	Huber Loss	13
		2.1.4	Log-Cosh Loss	14
		2.1.5	Quantile Loss	14
		2.1.6	Poisson Loss	15
	2.2	Regres	sion Performance Metrics	16
		2.2.1	Root Mean Squared Error (RMSE)	16
		2.2.2	Mean Absolute Percentage Error (MAPE)	17
		2.2.3	Symmetric Mean Absolute Percentage Error (SMAPE)	18
		2.2.4	Coefficient of Determination $R^2$	19
		2.2.5	Adjusted $R^2$	20
3	Clas	sificatio	on	20
	3.1	Classif	ication Loss Functions	20
		3.1.1	Binary Cross-Entropy Loss (BCE)	21
		3.1.2	Categorical Cross-Entropy Loss (CCE)	22
		3.1.3	Sparse Categorical Cross-Entropy Loss	22
		3.1.4	Weighted Cross-Entropy (WCE)	23
		3.1.5	Cross-Entropy Loss with Label Smoothing	24
		3.1.6	Negative Log-Likelihood	24
		3.1.7	Polyloss	25
		3.1.8	Hinge Loss	25
	3.2	Classif	ication Performance Metrics	26
		3.2.1	Confusion Matrix	26
		3.2.2	Confusion Matrix in Multi-Class Classification	27
		3.2.3	Accuracy	29
		3.2.4	Precision, PPV, or TDR	30
		3.2.5	Recall, Sensitivity, or True Positive Rate (TPR)	31
		3.2.6	F1-Score	31
		3.2.7	Specificity or True Negative Rate (TNR)	32
		3.2.8	False Positive Rate (FPR)	32
		3.2.9	Negative Predictive Value (NPV)	33
		3.2.10	False Discovery Rate (FDR)	34
			Precision-Recall Curve	34
			Precision-Recall Curve in Multi-Class Classification	35

		3.2.13	Area Under the Receiver Operating Characteristic Curve (AUC-ROC)	36
		3.2.14	Multi-Class Receiver Operating Characteristic Curve (AUC-ROC)	36
4	Ima	ge Class	sification	38
	4.1	_	Classification Loss Functions	38
	4.2	·	Classification Metrics	39
5	Obj	ect Dete	ection	39
	5.1	Object	Detection Loss Functions	39
		5.1.1	Smooth L1 Loss	40
		5.1.2	Balanced L1 Loss	41
		5.1.3	Intersection Over Union (IoU) Loss	42
		5.1.4	GIoU Loss	42
		5.1.5	DIoU and CIoU Losses	43
		5.1.6	Focal Loss	44
		5.1.7	YOLO Loss	45
		5.1.8	Wing Loss	45
	5.2	Object	Detection Metrics	46
		5.2.1	Average Precision (AP)	47
		5.2.2	Average Recall (AR)	47
6	Ima	ge Segn	nentation	48
	6.1	Segme	entation Loss Functions	49
		6.1.1	Cross-Entropy Loss for Segmentation	49
		6.1.2	Dice Loss	50
		6.1.3	Intersection Over Union (IoU) Loss for Segmentation	51
		6.1.4	Tversky Loss	52
		6.1.5	Lovász Loss	52
		6.1.6	Focal Loss for Segmentation	53
	6.2	Segme	entation Metrics	53
		6.2.1	Pixel Accuracy	54
		6.2.2	Boundary F1 Score (BF)	54
		6.2.3	Masked Average Precision (Mask AP)	55
		6.2.4	Panoptic Quality (PQ)	56
7	Face	e Recogn	nition	57
	7.1	Face R	Recognition Loss Functions and Metrics	57
		7.1.1	Softmax Loss	57
		7.1.2	A-Softmax (SphereFace) Loss	58
		7.1.3	Center Loss	59
		714	CosFace: Large-Margin Cosine Loss	60

		7.1.5	ArcFace: Additive Angular Margin Loss	60
		7.1.6	Triplet Loss	61
		7.1.7	Contrastive Loss	61
		7.1.8	Circle Loss	62
		7.1.9	Barlow Twins Loss	63
		7.1.10	SimSiam Loss	63
8	Mon	ocular l	Depth Estimation (MDE)	64
	8.1	Depth 1	Estimation Loss Functions	64
		8.1.1	Point-wise Error	66
		8.1.2	Scale Invariant Error	66
		8.1.3	Structural Similarity Index Measure (SSIM)	67
		8.1.4	Photometric Loss	67
		8.1.5	Disparity Smoothness Loss	68
		8.1.6	Appearance Matching Loss	68
		8.1.7	Left-Right Consistency Loss	69
		8.1.8	BerHu Loss (Reverse Huber)	69
		8.1.9	Edge Loss	70
		8.1.10	Minimum Reprojection Loss	71
		8.1.11	Scale-and-shift-invariant Loss (SSI)	71
	8.2	Depth 1	Estimation Metrics	73
		8.2.1	Mean Absolute Relative Error (AbsRel)	73
		8.2.2	Root Mean Squared Error (RMSE)	73
		8.2.3	Logarithmic RMSE (RMSE(log))	74
		8.2.4	Threshold Accuracy $(\delta)$	74
		8.2.5	Mean Log10 Error	75
		8.2.6	Percentage of Pixels with High Error	75
		8.2.7	Weighted Human Disagreement Rate (WHDR)	75
		8.2.8	Scale-Invariant Error (Metric Form)	76
9	Imaș	ge Gene	ration	76
	9.1	Image	Generation Loss Functions	77
		9.1.1	Reconstruction Loss	77
		9.1.2	Kullback–Leibler Divergence Loss	79
		9.1.3	Perceptual Loss	80
		9.1.4	Adversarial Loss	81
		9.1.5	Wasserstein Loss	82
		9.1.6	Negative Log-Likelihood in Normalizing Flows	83
		9.1.7	Contrastive Divergence	83
	9.2	Image	Generation Metrics	85

		9.2.1	Peak Signal-to-Noise Ratio (PSNR)	85
		9.2.2	Structural Similarity Index (SSIM)	86
		9.2.3	Inception Score (IS)	86
		9.2.4	Fréchet Inception Distance (FID)	88
10	Natu	ıral Lar	nguage Processing (NLP)	88
	10.1	Loss F	unctions Used in NLP	89
		10.1.1	Cross-Entropy Loss (Token-Level)	89
		10.1.2	Hinge Loss	91
		10.1.3	Cosine Similarity Loss	92
		10.1.4	Marginal Ranking Loss	93
	10.2	Losses	for Sequence Generation	94
		10.2.1	Connectionist Temporal Classification (CTC) Loss	94
		10.2.2	Minimum Risk Training (MRT)	95
		10.2.3	REINFORCE Algorithm	96
	10.3	Perform	nance Metrics Used in NLP	97
		10.3.1	Accuracy	97
		10.3.2	Precision, Recall, and F1 Score	99
		10.3.3	AUC-ROC	00
		10.3.4	BLEU Score	01
		10.3.5	METEOR	02
		10.3.6	ROUGE Score	03
		10.3.7	Perplexity	04
		10.3.8	Exact Match (EM)	04
		10.3.9	Word Error Rate (WER)	05
		10.3.10	Character Error Rate (CER)	06
11	Retr	ieval-Aı	ugmented Generation (RAG)	107
	11.1	Loss F	unctions Used in RAG	07
	11.2	Perform	nance Metrics Used in RAG	08
		11.2.1	Answer Semantic Similarity	08
		11.2.2	Answer Correctness	10
		11.2.3	Answer Relevance	10
		11.2.4	Context Precision	11
		11.2.5	Context Recall	11
		11.2.6	Faithfulness	12
		11.2.7	Summarization Score	12
		11.2.8	Context Entities Recall	12
		11.2.9	Aspect Critique	13
				114

15	Acknowledgments	119
14	Conclusion	118
	13.3 Trends and Future Directions	118
	13.2 Addressing Challenges	
	13.1 Challenges in Deep Learning	117
13	Challenges and Trends	116
12	Combining Multiple Loss Functions	115
	11.2.12 Answer Relevance	115
	11.2.11 Answer faithfulness	

#### 1 Introduction

Learning has become the dominant technology for solving problems involving unstructured data, such as images [1, 2, 3, 4, 5, 6, 7, 8], video, audio [9, 10, 11, 12, 13], and text [14, 15, 16, 17, 18]. One of the critical components of deep learning is the selection of the loss function and performance metrics used for training and evaluating models. Loss functions measure how effectively a model can approximate the desired output, while performance metrics assess the model's ability to make accurate predictions on unseen data. Choosing the appropriate loss function and performance metric is essential for success in deep learning tasks. However, with many options to choose from, it can be challenging for practitioners to determine the most suitable method for their specific task.

In this paper, we present a detailed overview of the most commonly utilized loss functions and performance metrics in the field of deep learning. We analyze the strengths and weaknesses of each method and provide illustrative examples of its application in various deep learning tasks.

First, we delve into the prevalent regression and classification loss functions, such as mean squared error, cross-entropy, and hinge loss, delineating their respective advantages, limitations, and typical use cases. Subsequently, we explore standard tasks in computer vision, such as image classification, object detection, image segmentation, face recognition, and image generation. Finally, we conclude our review by outlining the prevalent loss functions and metrics employed in natural language processing and retrieval augmented generation (RAG).

This paper is structured as follows. Section 1.1 outlines the difference between loss functions and performance metrics. Sections 2 and 3 provide an overview of the most widely used losses and metrics in regression and classification. Sections 4 to 9 delve into the prevalent computer vision tasks, detailing the associated loss functions and metrics. Section 10 concentrates on the use of losses and metrics in Natural Language Processing (NLP). Section 11 is dedicated to Retrieval Augmented Generation (RAG) losses and metrics. Section 12 discusses the common practice of combining multiple metrics, Section 13 discusses challenges and trends, and Section 14 rounds out the paper with a concluding statement.

## 1.1 Loss Functions vs. Performance Metrics

Loss functions and performance metrics are two distinct tools for evaluating the performance of a deep learning model and serve different purposes.

During training, a loss function measures the difference between the predicted and expected outputs of the model with the objective of optimizing the model parameters by minimizing this difference.

In contrast, a performance metric is used to evaluate the model after training. It helps to determine how well the model can generalize to new data and make accurate predictions. Performance metrics also aid in comparing different models or configurations to identify the best performing one.

The following list details the key differences between loss functions and performance metrics.

1. During the training of a deep learning model, loss functions are used to optimize the model's parameters, whereas performance metrics are used to evaluate the model's performance after training.

- 2. The choice of loss function typically depends on the model's architecture and the specific task at hand. In contrast, performance metrics are less dependent on the model's architecture and can be used to compare different models or configurations of a single model.
- 3. The ultimate goal of training a deep learning model is to minimize the loss function, while evaluating a model aims to maximize the performance metric, with the exception of error performance metrics such as Mean Squared Error.
- 4. Loss functions can be challenging to interpret as their values are often arbitrary and depend on the specific task and data. In contrast, performance metrics are often more interpretable and can be used across different tasks.

#### 1.2 Properties of Loss Functions

Loss functions have a series of properties that need to be considered when selected for a specific task:

- 1. **Convexity**: A loss function is convex if any local minimum is also the global minimum. Convex loss functions are desirable because they can be easily optimized using gradient-based optimization methods.
- Differentiability: A loss function is differentiable if its derivative with respect to the model parameters exists and is continuous. Differentiability is essential because it allows the use of gradient-based optimization methods.
- 3. **Robustness**: Loss functions should be able to handle outliers and not be affected by a small number of extreme values.
- 4. Smoothness: A loss function should have a continuous gradient and no sharp transitions or spikes.
- 5. **Sparsity**: A sparsity-promoting loss function should encourage the model to produce sparse output. This is useful when working with high-dimensional data and when the number of important features is small.
- 6. **Monotonicity**: A loss function is monotonic if its value decreases as the predicted output approaches the true output. Monotonicity ensures that the optimization process is moving toward the correct solution.

The tables below provide a summary of the loss functions and performance metrics that are discussed in this work. Table 1 outlines the functions and metrics employed in general tasks such as regression, binary classification, and multiclass classification. Table 2 provides an overview of the loss functions and metrics related to computer vision, and Table 3 summarizes the work on natural language processing.

### 1.3 Implementation Considerations and Software Libraries

Practical deep learning development involves decisions about software frameworks and tools for implementing, training, and evaluating models. Popular frameworks such as PyTorch [19], TensorFlow/Keras [20, 21], and MATLAB [22] provide core functionalities like computational graphs, automatic differentiation, and pre-implemented losses (e.g., MSE, cross-entropy) alongside standard metrics such as accuracy or precision-recall. These frameworks also facilitate customization, enabling users to define novel loss functions and evaluation protocols.

**PyTorch** Developed by Facebook's AI Research, PyTorch offers a dynamic computational graph that supports flexible model-building in Python. Its *torch.nn* module provides common loss functions, including nn.MSELoss for regression and nn.CrossEntropyLoss for multi-class classification. For metrics, PyTorch-based libraries (e.g., torchmetrics) or the framework's built-in tensor operations allow quick computation of accuracy, F1 scores, and even more complex measures like AUC-ROC. This design encourages straightforward experimentation when creating or modifying custom losses and metrics.

**TensorFlow and Keras** TensorFlow, from Google, and its high-level Keras API are pivotal in deep learning. Modules like tf.keras.losses and tf.keras.metrics contain canonical loss functions such as MeanSquaredError or BinaryCrossentropy, along with evaluation metrics like Accuracy or MeanIoU (for segmentation). Keras simplifies customization by allowing users to define new losses in Python, returning a scalar loss from y\_true and y\_pred. Similarly, extended metrics can be created by subclassing tf.keras.metrics.Metric, enabling tailored evaluation workflows for specialized tasks.

MATLAB MATLAB's Deep Learning Toolbox provides built-in functions such as crossentropy and mse for loss calculations, plus object-oriented templates to code custom layers and losses. MATLAB's emphasis on data analysis, visualization, and vectorized operations can streamline metric computation. Though less common in open-source research, MATLAB remains well regarded in many industrial and academic settings for debugging, simulation, and rapid prototyping.

Deep Learning Task	Loss Functions	Performance Metrics	
Regression	Mean Squared Error (MSE) (2.1.1) Mean Absolute Error (MAE) (2.1.2) Huber loss (2.1.3) Log-Cosh (2.1.4) Quantile loss (2.1.5) Poisson loss (2.1.6)	Mean Squared Error (MSE)(2.1.1) Mean Absolute Error (MAE)(2.1.2) Root Mean Squared Error (RMSE)(2.2.1) Mean Absolute Percentage Error (MAPE) (2.2.2) Symmetric MAPE (SMAPE) (2.2.3) $R^2$ (2.2.4) Adjusted $R^2$ (2.2.5)	
Binary Classification	Binary Cross-Entropy (BCE) (3.1.1) Weighted Cross-Entropy (3.1.4) Hinge loss (3.1.8) Focal loss (5.1.6)	Accuracy (3.2.3) Precision (3.2.4) Recall or True Positive Rate (TPR) (3.2.5) F1-Score (3.2.6) AUC-ROC (3.2.13) Precision/Recall Curve (3.2.11)	
Multi-Class Classification	Categorical Cross-Entropy (CCE) (3.1.2) Weighted Cross-Entropy (3.1.4) Sparse Categorical Cross-Entropy (SCCE)(3.1.3) CCE w/label smoothing (3.1.5) Focal loss (5.1.6) PolyLoss (3.1.7) Hinge loss (3.1.8)	Accuracy (3.2.3) Precision (3.2.4) Recall or True Positive Rate (TPR) (3.2.5) F1-Score (3.2.6) Precision/Recall Curve (3.2.11)	

Table 1: Loss functions and performance metrics for general tasks.

scikit-learn Integration While PyTorch and TensorFlow focus primarily on neural network construction, the Pythonbased scikit-learn [23] library offers an extensive suite of evaluation metrics. Functions like accuracy\_score, precision\_score, recall\_score, and f1\_score can be readily applied to predictions from deep networks, bridging deep learning with standardized machine-learning ecosystems. This synergy ensures that model assessments align with established criteria, thereby permitting direct comparisons with more classical methods.

**Example: Custom Losses and Metrics** Researchers often create specialized losses to incorporate domain knowledge. Below is a PyTorch example that combines MSE and L1 penalties:

```
import torch
import torch.nn as nn
class CustomLoss(nn.Module):
    def forward(self, predictions, targets):
        mse_loss = torch.mean((predictions - targets)**2)
        11_penalty = torch.mean(torch.abs(predictions))
        return mse_loss + 0.01 * 11_penalty
The equivalent TensorFlow/Keras version:
import tensorflow as tf
def custom_loss(y_true, y_pred):
    mse_loss = tf.reduce_mean(tf.square(y_pred - y_true))
    11_penalty = tf.reduce_mean(tf.abs(y_pred))
    return mse_loss + 0.01 * 11_penalty
```

For evaluation, a trained model's predictions can easily be passed to scikit-learn metrics:

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score

# Suppose y\_true, y\_pred are NumPy arrays from a trained model.

Table 2: Loss functions and performance metrics used in Computer Vision.

Computer Vision Task	Loss Functions	Performance Metrics
Object Detection	Smooth L1 (5.1.1) Balanced L1 (5.1.2) Intersection over Union loss (5.1.3) GIoU (5.1.4) DIoU and CIoU (5.1.5) Focal loss (5.1.6) YOLO loss (5.1.7) Wing loss (5.1.8)	Average Precision (5.2.1) Average Recall (5.2.2)
Semantic Segmentation	Categorical Cross-Entropy (6.1.1) Intersection over Union loss (IoU) (5.1.3) Dice Loss (6.1.2) Tversky loss (6.1.4) Lovasz loss (6.1.5) Focal loss (6.1.6)	Intersection over Union (IoU) (5.1.3) Pixel Accuracy (6.2.1) Average Precision (AP) (5.2.1) Boundary F1 Score (6.2.2) Average Recall (AR) (5.2.2)
Instance Segmentation	Categorical Cross-Entropy (CCE) (6.1.1) Intersection over Union loss (IoU) (5.1.3) Smooth L1 (5.2.1) Average Recall (AR) (5.2.2)	Masked Average Precision (6.2.3)
Panoptic Segmentation	Categorical Cross-Entropy (CCE) (6.1.1) Dice Loss (6.1.2)	Panoptic Quality (PQ) (6.2.4)
Face Recognition	A-Softmax (7.1.2) Center loss (7.1.3) CosFace (7.1.4) ArcFace (7.1.5) Triplet loss (7.1.6) Contrastive loss (7.1.7) Circle loss (7.1.8) Barlow Twins (7.1.9) SimSiam (7.1.10)	Accuracy (3.2.3) Precision (3.2.4) Recall (3.2.5) F1-Score (3.2.6)
Monocular Depth Estimation	Point-wise Error (8.1.1) Scale Invariant Error (8.1.2) Structural Similarity Index (8.1.3) Photometric loss (8.1.4) Disparity Smoothness (8.1.5) Appearance Matching (8.1.6) Left-Right Consistency (8.1.7) BerHu loss (8.1.8) Edge loss (8.1.9) Minimum Reprojection (8.1.10) Scale-and-shift-invariant (SSI)	Mean Absolure Relative Error (8.2.1) Root Mean Squared Error (8.2.2) Logarithmic RMSE (8.2.3) Threshold Accuracy (8.2.4) Mean Log10 Error (8.2.5) % of Pixels with High Error (8.2.6) Weighted Human Disagrement Rate (8.2.7) Scale-Invariant Error (8.2.8)
Image Generation	Reconstruction loss (9.1.1) KL Divergence (9.1.2) Perceptual loss (9.1.3) Adversarial Loss (9.1.4) Wasserstein Loss (9.1.5) Negative Log-Likelihood (9.1.6) Contrastive Divergence (9.1.7)	Peak Signal to Noise Ratio (PSNR) (9.2.1) Structural Similarity Index (SSIM) (9.2.2) Inception Score (IS) (9.2.3), Fréchet Inception Distance (FID) (9.2.4)

Table 3: Loss functions and performance metrics used in Natural Language Processing.

NLP Task	Loss Functions	Performance Metrics
Text Classification	Token-level Cross-Entropy (10.1.1) Hinge loss (10.1.2)	Accuracy (10.3.1) Precision/Recall/F1 (10.3.2) AUC-ROC (10.3.3)
Language Modeling	Token-level Cross-Entropy (10.1.1)	Perplexity (10.3.7) BLEU (10.3.4) ROUGE (10.3.6)
Machine Translation	Token-level Cross-Entropy (10.1.1) MRT (10.2.2) REINFORCE (10.2.3)	BLEU (10.3.4) ROUGE (10.3.6) Perplexity (10.3.7) Exact match (10.3.8)
Name Entity Recognition	Token-level Cross-Entropy (10.1.1)	Accuracy (10.3.1) Precision/Recall/F1 (10.3.2) Exact match (10.3.8)
Part-of-Speech Tagging	Token-level Cross-Entropy (10.1.1)	Accuracy (10.3.1) Precision/Recall/F1 (10.3.2)
Sentiment Analysis	Token-level Cross-Entropy (10.1.1) Hinge loss (10.1.2) Cosine Similarity (10.1.3) Adj. Circle (7.1.8)	Precision/Recall/F1 (10.3.2) AUC-ROC (10.3.3)
Text Summarization	Token-level Cross-Entropy (10.1.1) MRT (10.2.2) REINFORCE (10.2.3)	BLEU (10.3.4) ROUGE (10.3.6) Exact match (10.3.8)
Question Answering	Token-level Cross-Entropy (10.1.1) Hinge loss (10.1.2) Cosine Similarity (10.1.3)	Precision/Recall/F1 (10.3.2) Exact match (10.3.8)
Language Detection	Token-level Cross-Entropy (10.1.1) Hinge loss (10.1.2)	Accuracy (10.3.1) Precision/Recall/F1 (10.3.2)
Retrieval Augmented Generation (RAG)	CE (10.1.1) Contrastive loss (7.1.7) Marginal Ranking (10.1.4) NLL (3.1.6) KL Divergence (9.1.2)	Answer Semantic Similarity (11.2.1) Answer correctness (11.2.2) Answer Relevance (11.2.3) Context Precision (11.2.4) Context Recall (11.2.5) Faithfulness (11.2.6) Summarization score (11.2.7) Context Entitities Recall (11.2.8) Aspect Critique (11.2.9) Context Relevance (11.2.10) Answer Faithfulness (11.2.11) Answer Relevance (11.2.12)
Speech Recognition	CTC (10.2.1)	Word Error Rate (WER) (10.3.9) Character Error Rate (CER) (10.3.10) Perplexity (10.3.7)

```
acc = accuracy_score(y_true, y_pred)
prec = precision_score(y_true, y_pred, average='macro')
rec = recall_score(y_true, y_pred, average='macro')
```

These examples illustrate the convenience of mixing deep learning frameworks with machine-learning libraries, allowing for both flexible modeling and standardized performance assessment.

Overall, modern deep learning software enables seamless definition and customization of loss functions and metrics. This flexibility aids experimental rigor and reproducibility, letting practitioners fine-tune objectives, combine multiple losses for multi-task learning, and leverage validated metrics to benchmark performance across different methodologies.

In the following sections, we dive into two essential tasks in machine learning: regression and classification. We examine the loss functions and performance metrics used and offer practical guidance on when to use each, as well as real-world applications.

# 2 Regression

Regression is a supervised learning problem in machine learning that aims to predict a continuous output value based on one or more input features. Regression is used in various domains, including finance, healthcare, social sciences, sports, and engineering. Some practical applications include house price prediction [24], energy consumption forecasting [25], healthcare and disease prediction [26], stock price forecasting [27], and customer lifetime value prediction [28].

In the following subsections, we review the most common lost functions and performance metrics used for regression.

### 2.1 Regression Loss Functions

In this section, we explore various regression loss functions that play a crucial role in evaluating and optimizing predictive models. A regression loss function  $\mathcal L$  is typically defined over a dataset  $\mathcal D=\{(x_i,y_i)\}_{i=1}^n$  of n samples, where  $x_i\in\mathbb R^d$  denotes the d-dimensional input feature vector and  $y_i\in\mathbb R$  is the corresponding continuous target value. Let  $\hat y_i=f_\theta(x_i)$  be the prediction of a model  $f_\theta$ , parameterized by  $\theta$ . The fundamental objective in regression is to find the parameter vector  $\theta$  that minimizes the overall loss function

$$\min_{\theta} \mathcal{L}(\{(x_i, y_i)\}_{i=1}^n, \theta). \tag{1}$$

Table 4 summarizes common regression loss functions, highlighting their uses, data traits, benefits, and drawbacks. It aids in choosing a suitable loss function by considering data nature, outlier robustness, and the balance between interpretability and optimization smoothness.

The following subsections describe each of these loss functions in more detail.

#### 2.1.1 Mean Squared Error (MSE)

Mean Squared Error (MSE), also referred to as the L2 loss, is one of the most widely used regression loss functions [29]. It is defined as the average of the squared differences between the predicted values and the true values:

$$MSE(\theta) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i - f_{\theta}(x_i))^2.$$
 (2)

# **Key Properties:**

- Non-negativity: Since squared errors are always non-negative, MSE returns a value ≥ 0, with 0 indicating a perfect fit.
- **Sensitivity to outliers:** By squaring the errors, large errors have a disproportionately higher impact on the loss, making MSE sensitive to outliers.
- **Differentiability:** MSE is smooth and differentiable with respect to both  $\hat{y}_i$  and  $\theta$ . The gradient of MSE with respect to a prediction  $\hat{y}_i$  is

$$\frac{\partial}{\partial \hat{y}_i} \left( y_i - \hat{y}_i \right)^2 = -2 \left( y_i - \hat{y}_i \right). \tag{3}$$

Table 4: Guideline	es for selecting a	regression loss	function	based on	usage,	data c	characteristics,	advantages,	and
limitations.	_	_			_				

Function	Usage	<b>Data Characteristics</b>	Advantages	Limitations
MSE	Standard tasks (e.g., linear regression) If you want to penal- ize large errors more severely	Data with few or no heavy outliers Errors approximately Gaussian	Differentiable every- where (smooth) Convex in parameters Commonly used, well- understood	Highly sensitive to outliers Scale-dependent; not always comparable across problems Can prioritize reducing large errors over smaller ones
MAE	When robustness to outliers is desired L1-regularized meth- ods, median-based tasks	Skewed error distribu- tions Presence of moderate outliers	Less sensitive to outliers than MSE Convex in parameters Interpretable as median error	Non-differentiable at zero error Scale-dependent
Huber	When a smooth, ro- bust loss is needed Want a tunable transi- tion between MSE and MAE	Data with occasional outliers Need partial robust- ness but still smooth optimization	Combines advantages of MSE (for small er- rors) and MAE (for large errors) Less sensitive to out- liers than MSE Fully differentiable	Requires choice of threshold $\delta$ Not scale-invariant
Log-Cosh	Similar use-cases to Huber where smooth- ness is preferred If you desire a simpler, always differentiable robust alternative	Presence of outliers but not extremely large ones	Smooth and differentiable everywhere Robust to moderate outliers	More sensitive to small errors than Huber Less intuitive thresh- olding than Huber
Quantile	When predicting intervals or specific percentiles Tasks requiring asymmetric error penalties	Heteroscedastic or skewed data Need to estimate conditional quantiles (e.g., risk, demand)	Captures the distribution of possible outcomes Robust to outliers when $q \neq 0.5$ Generalization of MAE $(q = 0.5)$	Must choose relevant quantile(s) Non-differentiable at zero error Interpretation can be less straightforward than MSE/MAE
Poisson	Count data prediction Data assumed to fol- low (or roughly fol- low) a Poisson process	Target is non-negative, discrete counts Often low-to- moderate integer values	Aligned with count- data likelihood Ensures non-negative predictions (with ex- ponential link)	Requires careful handling of zero/low counts Assumes Poisson-like distribution

This property facilitates gradient-based optimization methods such as Stochastic Gradient Descent (SGD).

- Convexity (in predictions): MSE is convex in terms of the predictions  $\hat{y}_i$ . For linear models  $\hat{y}_i = \theta^\top x_i$ , MSE is also convex in  $\theta$ . However, in deep neural networks with non-linear activation functions, the overall error surface can become non-convex.
- Scale-dependence: The magnitude of MSE depends on the scale of the target variable  $y_i$ , making cross-comparison of models on different scales less straightforward. Consequently, measures like Root Mean Squared Error (RMSE) or Mean Squared Percentage Error (MSPE) are often preferred for interpretability.

**Optimization:** Minimizing MSE can be interpreted as minimizing the squared  $\ell_2$ -norm of the residuals:

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} (y_i - f_{\theta}(x_i))^2. \tag{4}$$

In a simple linear regression scenario where  $f_{\theta}(x_i) = \theta^{\top} x_i$ , an analytic closed-form solution exists using the Normal Equation [29]. However, for more complex models such as neural networks, numerical methods like gradient descent are commonly employed.

**Practical Considerations:** MSE is easy to implement and interpret, but its emphasis on larger errors may yield suboptimal performance in the presence of outliers. In such situations, more robust alternatives like Mean Absolute Error (MAE) or Huber Loss may be preferable.

#### 2.1.2 Mean Absolute Error (MAE)

Mean Absolute Error (MAE), also referred to as the *L1 loss*, is another common choice for regression tasks [30]. It measures the average of the absolute differences between the predicted values and the true values:

$$MAE(\theta) = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| = \frac{1}{n} \sum_{i=1}^{n} |y_i - f_{\theta}(x_i)|.$$
 (5)

#### **Key Properties:**

- Non-negativity: MAE is always  $\geq 0$ . A value of 0 indicates a perfect fit.
- **Robustness to outliers:** By using the absolute value rather than squaring the error, MAE penalizes large errors linearly, making it less sensitive to outliers than MSE.
- Non-differentiability at 0: The absolute value function |z| is not differentiable at z=0. Consequently, the loss function is not differentiable when  $y_i = \hat{y}_i$ . However, subgradient methods [31, 32, 33, 34] can be employed in gradient-based optimization to address this issue. Formally, the subgradient w.r.t.  $\hat{y}_i$  is:

$$\partial_{\hat{y}_i} |y_i - \hat{y}_i| = \begin{cases} -1, & \text{if } \hat{y}_i > y_i, \\ +1, & \text{if } \hat{y}_i < y_i, \\ [-1, +1], & \text{if } \hat{y}_i = y_i. \end{cases}$$
(6)

• Convexity (in predictions): MAE is convex in  $\hat{y}_i$ , ensuring a single global minimum. As with MSE, in deep neural networks with non-linearities, the overall loss surface becomes non-convex in the model parameters.

**Optimization:** Minimizing MAE can be understood as minimizing the  $\ell_1$ -norm of the residuals:

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} |y_i - f_{\theta}(x_i)|. \tag{7}$$

Unlike the MSE case, there is no simple closed-form solution for linear models. Algorithms such as coordinate descent or iterative reweighted least squares (IRLS) can be used [31], or gradient-based methods can approximate a solution by using subgradients.

**Practical Considerations:** MAE is more robust to outliers compared to MSE but lacks the smoothness that MSE provides, potentially making optimization slower in certain scenarios. As with MSE, the absolute scale of  $y_i$  affects MAE's value, necessitating the use of scale-invariant metrics such as the Mean Absolute Percentage Error (MAPE) or Normalized Mean Absolute Error (NMAE) for inter-problem comparisons.

**Summary of MSE vs. MAE:** Both MSE and MAE are key loss functions for regression tasks. MSE emphasizes large errors more heavily and is differentiable everywhere, whereas MAE is more robust to outliers but non-differentiable at zero error. The choice between the two often depends on the application's tolerance to outliers, interpretability needs, and computational considerations during optimization.

#### 2.1.3 Huber Loss

The *Huber loss*, introduced by Huber in [35], is a piecewise-defined function that combines the advantageous properties of both the Mean Squared Error (MSE) and Mean Absolute Error (MAE). Let y be the true value,  $\hat{y}$  be the predicted value, and  $\delta$  be a user-specified threshold. The Huber loss for a single sample is defined by

$$L_{\text{Huber}}(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2, & \text{if } |y - \hat{y}| \le \delta, \\ \delta(|y - \hat{y}| - \frac{1}{2}\delta), & \text{otherwise.} \end{cases}$$
(8)

When the prediction error  $|y - \hat{y}|$  is smaller than  $\delta$ , the function behaves like a quadratic (similar to MSE). For large errors, it transitions into a linear penalty (similar to MAE). This property provides robustness to outliers compared to standard MSE while maintaining smoothness for smaller residuals.

### **Properties:**

- **Robustness to Outliers:** For errors larger than  $\delta$ , the linear penalty avoids the rapid growth that occurs in MSE. Thus, outliers have a smaller effect on parameter updates.
- Smoothness and Differentiability: Except for a point of non-differentiability at  $|y \hat{y}| = \delta$ , the function remains differentiable elsewhere, enabling gradient-based optimization. In practice, subgradient or piecewise derivative implementations address the corner case.
- **Piecewise Gradient:** The gradient of  $L_{\text{Huber}}$  with respect to  $\hat{y}$  is:

$$\frac{\partial L_{\text{Huber}}}{\partial \hat{y}} = \begin{cases}
-(y - \hat{y}), & \text{if } |y - \hat{y}| \leq \delta, \\
-\delta \operatorname{sign}(y - \hat{y}), & \text{if } |y - \hat{y}| > \delta.
\end{cases}$$
(9)

Here, sign(z) is +1 if z > 0 and -1 otherwise.

 Choice of δ: The threshold δ can be chosen empirically by cross-validation. A smaller δ behaves more like MAE, whereas a larger δ behaves closer to MSE.

**Practical Considerations:** Huber loss is commonly employed in robust regression settings (e.g., linear regression and time series forecasting) where outliers and noise may be present. Its piecewise-defined structure makes it suitable for scenarios in which the transition from quadratic to linear penalization can be controlled. Nonetheless, the introduction of  $\delta$  adds a hyperparameter that requires tuning.

## 2.1.4 Log-Cosh Loss

The Log-Cosh loss [36] uses the hyperbolic cosine to measure errors. For n samples, it is given by

$$L_{\text{LogCosh}} = \frac{1}{n} \sum_{i=1}^{n} \log[\cosh(y_i - \hat{y}_i)], \tag{10}$$

where  $y_i$  and  $\hat{y}_i$  respectively denote the true and predicted values for the *i*-th sample.

### **Properties:**

• Smoothness and Differentiability: The function  $\log(\cosh(z))$  is smooth and differentiable for all  $z \in \mathbb{R}$ . The derivative of  $\log(\cosh(z))$  with respect to z is  $\tanh(z)$ . Thus, the gradient of a single-sample loss  $\log[\cosh(y-\hat{y})]$  with respect to  $\hat{y}$  is:

$$\frac{\partial}{\partial \hat{y}} \log[\cosh(y - \hat{y})] = -\tanh(y - \hat{y}). \tag{11}$$

- Less Sensitivity to Outliers: Unlike MSE, the cosh function grows exponentially only for large  $|y \hat{y}|$  but the  $\log(\cdot)$  moderates this growth. Consequently, extreme errors do not dominate as heavily as in the squared error case, providing a degree of robustness.
- Emphasis on Small Errors: For moderate errors,  $\log(\cosh(z)) \approx \frac{z^2}{2}$  (since  $\cosh(z) \approx 1 + \frac{z^2}{2}$  for small z). Thus, the loss remains smooth and lightly penalizes small deviations. However, it can be more sensitive to smaller deviations compared to Huber loss because it continuously grows with  $|y \hat{y}|$ , though less sharply than MSE.

**Practical Considerations:** Log-Cosh loss is particularly useful when one desires a fully smooth and continuous loss landscape without the piecewise nature of Huber. It offers a balance of robustness and differentiability, making it attractive in complex models (e.g., deep networks) where smooth gradient feedback is advantageous.

### 2.1.5 Quantile Loss

Also known as *quantile regression loss*, this function is often used to produce predictive intervals or asymmetric error penalties [37]. For a quantile level 0 < q < 1, let  $\hat{y}$  be the predicted value and y be the true value. The quantile loss for a single prediction is given by:

$$L_{\text{quantile}}(y, \hat{y}) = q \max(y - \hat{y}, 0) + (1 - q) \max(\hat{y} - y, 0).$$
(12)

Intuitively, if the model underestimates  $(\hat{y} < y)$ , the error term  $(y - \hat{y})$  is weighted by q. Conversely, if the model overestimates  $(\hat{y} > y)$ , then the error term  $(\hat{y} - y)$  is weighted by (1 - q). Hence, quantile loss generalizes Mean Absolute Error (MAE), since for q = 0.5, the two coincide.

### **Properties:**

- Asymmetric Penalties: By choosing different values of q, one can penalize overestimation or underestimation more heavily. For example, q > 0.5 penalizes underestimates more strongly, which can be useful in safety-critical applications where under-forecasting is risky.
- Non-differentiability at Residual 0: Similar to MAE, the function  $\max(\cdot, 0)$  is not differentiable at 0. One typically uses subgradient or piecewise derivatives to handle optimization. Specifically, the subgradient of  $\max(\hat{y} y, 0)$  w.r.t.  $\hat{y}$  is 1 if  $\hat{y} > y$ , and 0 if  $\hat{y} < y$ . Analogous logic applies to  $\max(y \hat{y}, 0)$ .
- Relation to Median and Other Quantiles: When q=0.5, the solution to minimizing  $\sum L_{\text{quantile}}$  yields the conditional median of y. More generally, different q values yield the conditional q-th quantile of the response variable, facilitating interval predictions.

**Practical Use Cases:** Quantile regression is particularly helpful when a single-point estimate is insufficient for decision making. By estimating upper and lower quantiles, it provides *interval predictions* or confidence bounds for a variety of applications:

- Financial Risk Management: Estimating Value-at-Risk (VaR) and Conditional Value-at-Risk (CVaR) for extreme loss scenarios [38].
- Supply Chain and Inventory Management: Predicting demand ranges to balance stockouts and overstock situations [39].
- Energy Production: Forecasting power output ranges for grid stability [40].
- Economic Forecasting: Generating uncertainty bounds for economic indicators [41].
- Weather Forecasting: Producing uncertainty intervals for temperature or rainfall [42, 43].
- Real Estate Pricing: Offering predictive price intervals rather than a single estimate [44].
- **Healthcare:** Modeling a range of patient outcomes [45].

Thus, quantile loss provides a flexible approach to addressing asymmetric costs and quantifying predictive uncertainty.

#### 2.1.6 Poisson Loss

Poisson loss is utilized when the target variable  $y_i$  represents count data assumed to follow a Poisson distribution. It is derived from the negative log-likelihood of the Poisson distribution:

$$P(y_i \mid \lambda_i) = \frac{\lambda_i^{y_i} e^{-\lambda_i}}{y_i!}, \quad y_i \in \{0, 1, 2, \dots\},$$
(13)

where  $\lambda_i > 0$  is the Poisson rate parameter (i.e., the expected count). The corresponding negative log-likelihood (omitting constant terms that do not depend on  $\lambda_i$ ) is:

$$\ell_i(\lambda_i) = \lambda_i - y_i \log(\lambda_i). \tag{14}$$

In practice, for a dataset of n samples, the *Poisson loss* is often written as

$$L_{\text{Poisson}} = \frac{1}{n} \sum_{i=1}^{n} \left[ \hat{y}_i - y_i \log(\hat{y}_i) \right], \tag{15}$$

where  $\hat{y}_i$  is the model's predicted (non-negative) rate for the *i*-th observation.

**Non-negativity and Link Functions:** Because count data cannot be negative,  $\hat{y}_i$  must be constrained to  $\hat{y}_i > 0$ . In generalized linear models or neural networks, one often uses an exponential link function:

$$\hat{y}_i = \exp(\mathbf{w}^\top \mathbf{x}_i + b), \tag{16}$$

so that  $\hat{y}_i$  is guaranteed to be positive for all  $\mathbf{x}_i$ . Substituting (16) into (15) yields

$$L_{\text{Poisson}} = \frac{1}{n} \sum_{i=1}^{n} \left[ \exp(\mathbf{w}^{\top} \mathbf{x}_{i} + b) - y_{i} \log(\exp(\mathbf{w}^{\top} \mathbf{x}_{i} + b)) \right]$$
$$= \frac{1}{n} \sum_{i=1}^{n} \left[ \exp(\mathbf{w}^{\top} \mathbf{x}_{i} + b) - y_{i} (\mathbf{w}^{\top} \mathbf{x}_{i} + b) \right], \tag{17}$$

where the simplification  $\log(\exp(\cdot)) = \cdot$  has been applied.

**Applications:** The Poisson distribution is natural for modeling the number of times an event occurs in a fixed interval. Example use-cases include:

- Traffic Modeling: Predicting vehicle counts through a toll booth under varying conditions [46].
- Healthcare and Epidemiology: Forecasting disease cases by region to inform resource allocation [47].
- Insurance: Modeling insurance claim frequencies for rate-making [48].
- Customer Service: Predicting incoming call volumes to allocate call-center staff [49].
- Internet Usage: Modeling the number of website visits or ad-clicks in a time interval [50].
- Manufacturing: Estimating defects or failures in a production line for quality control [51].
- Crime Analysis: Assessing occurrences of crimes by area to guide resource deployment [52].

By directly modeling count-based observations under a Poisson assumption, Poisson loss allows both interpretability (expected count rates) and theoretical alignment with the discrete nature of the response variable.

### 2.2 Regression Performance Metrics

Selecting the right performance metrics is key to accurately evaluating regression models. Table 5 provides guidelines for choosing among five commonly used metrics based on data characteristics, robustness requirements, and task objectives. These metrics each offer distinct advantages and limitations, helping researchers account for factors such as error sensitivity, interpretability, and data scale. In the subsequent sections, we delve into the details of these metrics, omitting Mean Squared Error (MSE) and Mean Absolute Error (MAE) because they have been previously discussed as loss functions.

## 2.2.1 Root Mean Squared Error (RMSE)

Root Mean Squared Error (RMSE) is defined as the square root of the Mean Squared Error (MSE). For n data points  $\{(y_i, \hat{y}_i)\}_{i=1}^n$ , where  $y_i$  is the true value and  $\hat{y}_i$  is the predicted value, RMSE is given by

RMSE = 
$$\sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$
. (18)

### **Properties and Interpretation:**

- Unit Consistency: Since RMSE is the square root of the average squared error, it has the same unit of measurement as the target y. This property makes RMSE highly interpretable in many practical scenarios (e.g., predicting prices in currency units, demand in units sold, etc.).
- Geometric Perspective: RMSE can be viewed as the  $\ell_2$ -norm distance between the vector of true values  $\mathbf{y} = [y_1, y_2, \dots, y_n]^{\mathsf{T}}$  and the vector of predicted values  $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n]^{\mathsf{T}}$ . Specifically,

$$RMSE(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{\sqrt{n}} \|\mathbf{y} - \hat{\mathbf{y}}\|_{2}.$$
 (19)

Metric	Usage	Data Characteristics	Advantages	Limitations	
RMSE	Measures average error magnitude	Continuous data with moderate outliers	Same units as target Penalizes large errors	Sensitive to outliers Does not distinguish error types	
MAPE	Forecasting where relative error is critical	,		Undefined if true value = 0 Very sensitive to outliers	
SMAPE	Time-series with balanced penalty on over/under-prediction	Varied scales Cases with zero or near-zero values	Symmetric treatment of over/under error Useful in cost- sensitive scenarios	Undefined if both true and predicted = 0 Still sensitive to out- liers	
$R^2$	Explains proportion of variance in target	Data with primarily linear relationships	Intuitive interpretation as variance explained Allows model compar- ison	Can be misleading for non-linear data Increases with more predictors	
Adjusted $R^2$	Model comparison in multi-predictor scenarios	Larger feature sets Similar contexts as $\mathbb{R}^2$	Penalizes irrelevant predictors More reliable for model selection	Same assumptions as $\mathbb{R}^2$ Negative if worse than mean-only model	

Table 5: Guidelines for selecting a regression metric based on usage, data characteristics, advantages and limitations.

This represents the average "Euclidean distance" between predictions and actuals, emphasizing larger residuals (errors).

- Sensitivity to Outliers: Squaring the errors disproportionately penalizes large errors. Hence, RMSE is sensitive to outliers; a few large residuals can inflate its value significantly.
- Scale Dependence: RMSE is directly influenced by the scale of y. Larger or more variable targets can naturally produce higher RMSE values. This can hamper cross-comparison between datasets with different ranges.
- Normalization and Comparison: To mitigate scale effects and facilitate model comparisons across different datasets or target scales, one may normalize the RMSE by the standard deviation  $\sigma_y$  of the true values:

Normalized RMSE = 
$$\frac{\text{RMSE}}{\sigma_y}$$
. (20)

An RMSE comparable to or below  $\sigma_y$  often indicates a model that predicts on par with—or better than—simply predicting the mean.

• Relationship to Standard Deviation: When the model predictions  $\hat{y}_i$  approximate the true values  $y_i$  well, the distribution of residuals  $(y_i - \hat{y}_i)$  tends to have smaller variance, yielding a lower RMSE. Comparing RMSE to  $\sigma_y$  can reveal whether a model is capturing most of the variability in the data.

**Use Cases and Limitations:** RMSE is widely employed in fields such as financial forecasting, demand prediction, and various regression problems, due to its direct penalization of large errors. However:

- Outlier Sensitivity: Excessively large errors can overshadow overall performance.
- Lack of Error-Type Distinction: RMSE combines both systematic (bias) and random (variance) errors into a single measure. One cannot directly discern error structure from RMSE alone.
- Scale Effects: Care must be taken when comparing RMSE across datasets with very different units or ranges of y.

### 2.2.2 Mean Absolute Percentage Error (MAPE)

Mean Absolute Percentage Error (MAPE) gauges the average relative error of predictions, expressed as a percentage. For n samples,

MAPE = 
$$\frac{1}{n} \sum_{i=1}^{n} \left( \frac{|y_i - \hat{y}_i|}{y_i} \times 100 \right),$$
 (21)

where  $y_i$  denotes the true value and  $\hat{y}_i$  the predicted value.

### **Properties:**

- Scale Independence: Because each term in MAPE is a ratio  $|y_i \hat{y}_i|/y_i$ , it is invariant to scaling of  $y_i$ . This property makes MAPE suitable for comparing performance across different problems or datasets that involve significantly different ranges of values.
- Interpretability: Being expressed in percentage terms, MAPE has an intuitive interpretation in many business, finance, or retail contexts: for instance, "the model is off by 12% on average."
- Sensitivity to Zero or Near-Zero Values: If any  $y_i$  is zero (or extremely small), the term  $|y_i \hat{y}_i|/y_i$  can become undefined or extremely large. A common workaround is to add a small constant  $\epsilon$  to  $y_i$  in the denominator when necessary:

$$MAPE_{\epsilon} = \frac{1}{n} \sum_{i=1}^{n} \left( \frac{|y_i - \hat{y}_i|}{\max\{y_i, \epsilon\}} \times 100 \right). \tag{22}$$

- Outlier Sensitivity: While MAPE is a relative measure, a single large relative error can still inflate the metric, especially if  $y_i$  is small. Hence, outliers can disproportionately affect MAPE—albeit in terms of *relative* deviation rather than absolute magnitude.
- Non-Symmetric Error Treatment: Over- and under-predictions of the same magnitude do not result in symmetrical impact on MAPE, given the percentage basis. For example, predicting 50 instead of 100 is a 50% error, but predicting 150 instead of 100 is also a 50% error in raw terms—however, the absolute effect on MAPE can differ if there is variation in denominators across samples.

#### **Use Cases and Limitations:**

- Industries With Relative Error Focus: Finance, retail, and forecasting tasks where percentage deviations matter more than raw error magnitudes often favor MAPE.
- Comparative Model Assessments: MAPE is well-suited for comparing models trained on data with different scales, thanks to its dimensionless property.
- Handling of Zero or Near-Zero Targets: Special caution is required when any  $y_i \approx 0$ . Alternative metrics like the Symmetric MAPE (sMAPE), or an  $\epsilon$ -adjusted MAPE, might be used to mitigate infinite or excessively large percentage errors.

In summary, MAPE provides an intuitive percentage-based evaluation of model performance but can pose challenges in scenarios with zero or small true values and can be disproportionately influenced by large percentage errors.

### 2.2.3 Symmetric Mean Absolute Percentage Error (SMAPE)

Symmetric Mean Absolute Percentage Error (SMAPE) is a variant of the Mean Absolute Percentage Error (MAPE) often used to evaluate the accuracy of forecasts in time series settings [53]. For n samples  $\{(y_i, \hat{y}_i)\}_{i=1}^n$ , SMAPE is defined as

SMAPE = 
$$\frac{2}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{|y_i| + |\hat{y}_i|} \times 100$$
. (23)

# **Properties:**

- Symmetry for Over-/Under-Estimation: Unlike MAPE, SMAPE employs the denominator  $|y_i| + |\hat{y}_i|$ , which balances the penalty between over-prediction and under-prediction. Thus, overestimations and underestimations of the same magnitude produce identical error contributions.
- Scale Independence: By normalizing each absolute error by the sum of absolute values, SMAPE partially mitigates issues associated with the scale of  $y_i$ . Nevertheless, if both  $|y_i|$  and  $|\hat{y}_i|$  are very small, SMAPE terms can still be large or undefined in cases where both are zero.

- Handling of Zeros: If both  $y_i$  and  $\hat{y}_i$  are zero, the denominator  $|y_i| + |\hat{y}_i| = 0$ . A common workaround is to omit (or treat specially) these terms or add a small  $\epsilon$  in the denominator.
- Outlier Sensitivity: While SMAPE is more balanced than MAPE in treating over- and under-estimation, large relative errors (e.g., due to very small denominators) can still exert an outsized effect on the overall metric.

**Practical Implications:** SMAPE's symmetry property is particularly valuable in applications where over- and under-predictions have equally costly (albeit different) consequences:

- Inventory Management: Over-predicted demand can lead to excess stock and wasted resources, whereas under-predicted demand causes stockouts and lost sales [54].
- Energy Demand Forecasting: Over-prediction wastes resources via unneeded power generation; underprediction can cause blackouts or emergency generation [55].
- Financial Markets: Overestimating prices risks financial losses via unwise investments, while underestimation can miss profitable opportunities [56].
- Sales Forecasting: Overestimates lead to unused inventory or labor; underestimates cause missed revenues and poor customer service [57].
- Transportation and Logistics: Overestimates waste resources on underfilled vehicles or routes; underestimates lead to congestion or unmet demand [58].

Overall, SMAPE's symmetry makes it a fitting choice where the miscost of over-prediction is comparably important to that of under-prediction, though care must be taken with zeros and outliers.

# **2.2.4** Coefficient of Determination $R^2$

The Coefficient of Determination, denoted  $R^2$ , measures how well a model explains the total variation in the target variable y. Let  $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$  be the mean of the observed y-values. Then,

$$R^{2} = 1 - \frac{\sum_{i=1}^{n} (y_{i} - \hat{y}_{i})^{2}}{\sum_{i=1}^{n} (y_{i} - \overline{y})^{2}}.$$
 (24)

## **Interpretation:**

• Explained Variance: The denominator  $\sum (y_i - \overline{y})^2$  is the total variance of y. The numerator  $\sum (y_i - \hat{y}_i)^2$  is the residual sum of squares (RSS), representing the unexplained portion of variability by the model. Thus,

$$R^2 = 1 - \frac{\text{RSS}}{\text{Total Variance}}.$$
 (25)

An  $R^2$  close to 1 indicates a model that explains most of the variation in the data, whereas  $R^2 \approx 0$  indicates the model is barely better than the mean predictor. Notably,  $R^2$  can be negative if the model is worse than simply predicting  $\overline{y}$ .

- Model Comparison: Higher  $R^2$  values generally indicate better explanatory power. However,  $R^2$  must be interpreted in the context of the problem. For instance, a low  $R^2$  might still be acceptable if the data naturally exhibits high variance and partial predictability.
- Linear vs. Nonlinear Models:  $R^2$  was originally introduced for linear regression but is often used for more complex models. Its interpretation can become less straightforward in highly nonlinear settings. Still, it remains a commonly reported metric for uniform comparisons across model types.

#### **Limitations:**

- Overfitting in Multiple Regression:  $R^2$  typically increases (or remains the same) when additional predictors are introduced, even if they have little explanatory power. Adjusted  $R^2$  (Section 2.2.5) partially addresses this.
- Inability to Detect Bias: An  $R^2$  value does not disclose if the model systematically overestimates or underestimates the target variable.
- Sensitivity to Outliers: A few extreme points can shift the overall variance structure, thus affecting  $R^2$ . Careful examination of residual plots is advisable.
- Potential Misleading Use in Small Samples: In small datasets,  $R^2$  can be highly unstable or misleading.

# 2.2.5 Adjusted $R^2$

When multiple predictors are included in a regression model,  $Adjusted R^2$  offers a way to account for the fact that standard  $R^2$  naturally increases (or remains constant) as variables are added. Adjusted  $R^2$  is computed as

Adjusted 
$$R^2 = 1 - \left(\frac{1 - R^2}{1}\right) \cdot \frac{n - 1}{n - k - 1},$$
 (26)

where n is the number of observations, k is the number of predictors, and  $R^2$  is as in (24).

### **Key Rationale:**

- Penalty for Additional Predictors: The factor  $\frac{n-1}{n-k-1}$  imposes a penalty for each added predictor, ensuring that the adjusted  $R^2$  will only increase if the newly added predictor sufficiently reduces the residual sum of squares.
- Model Comparison: Adjusted  $R^2$  facilitates fair comparison of nested models (e.g., same dataset, differing number of predictors). A large jump in  $R^2$  but only a small (or negative) change in adjusted  $R^2$  suggests that the added predictors may not be truly informative.

#### **Benefits and Limitations:**

- 1. **Overfitting Mitigation:** By penalizing model complexity, adjusted  $R^2$  reduces the risk of overfitting that is sometimes masked by a raw  $R^2$ .
- 2. Interpretation Parallels: Like  $R^2$ , adjusted  $R^2$  also ranges up to 1, indicating how much variation is explained after penalizing for predictor count. Negative values imply the model is worse than predicting  $\overline{y}$ .
- 3. **Still Limited in Nonlinear or Complex Models:** Adjusted  $R^2$  remains most naturally interpreted in the context of linear regression or generalized linear modeling. It may not fully capture complex relationships (e.g., deep neural networks), though it is still sometimes reported as a reference.
- 4. Sample Size Considerations: For small n, the penalty factor can be large, and minor changes in  $\sum (y_i \hat{y}_i)^2$  can trigger large swings in adjusted  $R^2$ .

**Practical Example:** Consider a regression for housing prices based on factors like square footage, number of bedrooms, location, etc. Adding an irrelevant predictor (e.g., day of the week the listing was posted) might trivially increase  $R^2$ , but adjusted  $R^2$  would likely decrease unless the new variable genuinely improves explanatory power. Thus, adjusted  $R^2$  provides a more robust way to evaluate whether a predictor is meaningfully contributing to the model.

## 3 Classification

Classification is a supervised machine learning task in which a model maps input features  $\mathbf{x}_i \in \mathbb{R}^d$  to discrete labels  $y_i \in \mathcal{Y}$ . The goal is to learn a decision function

$$f_{\theta}: \mathbb{R}^d \to \mathcal{Y},$$
 (27)

parameterized by  $\theta$ , that assigns each input  $\mathbf{x}_i$  to a specific category in  $\mathcal{Y}$ . In typical settings:

- Binary Classification:  $\mathcal{Y} = \{0, 1\}.$
- Multi-Class Classification:  $\mathcal{Y} = \{1, 2, ..., C\}$  for C classes.
- Multi-Label Classification:  $\mathcal{Y} \subseteq \{0,1\}^C$ , allowing multiple classes to be assigned to the same instance.

Classification algorithms span a wide range of methods, such as Decision Trees, Naïve Bayes, k-Nearest Neighbors, Support Vector Machines, Random Forests, Gradient Boosting, and Neural Networks. Regardless of the specific algorithm, an appropriate choice of *loss function* is critical for guiding the training procedure. The following subsections present the most common classification loss functions and discuss their properties and typical use-cases.

#### 3.1 Classification Loss Functions

Several loss functions can be used for classification tasks, depending on the nature of the data, the number of classes, and the model architecture. Table 6 summarizes key classification loss functions, detailing their usage scenarios, data traits, advantages, and limitations. Below, we provide a more detailed discussion of each function.

Table 6: Guidelines for selecting a classification loss function based on usage, data characteristics, advantages, and limitations.

Function	Usage	Data Character- istics	Advantages	Limitations
BCE	Binary classifica- tion Common in logis- tic regression	Labels in {0,1} Sensitive to imbalance	Probabilistic inter- pretation Strong penalty for confident mistakes	Overconfidence if not regularized Often needs weighting
CCE	Multi-class classi- fication Softmax-based outputs	One-hot labels Class imbalance is possible	Differentiable Strong penalty for misclassifications	Requires one-hot (extra memory) Sensitive to class imbalance
Sparse CCE	Large-class prob- lems Integer-labeled data	No one-hot encoding Useful with large vocabularies	Memory efficient Same probabilistic basis as CCE	Only for integer labels Imbalance re- mains an issue
Weighted Cross- Entropy	Binary or multi- class tasks Class imbalance or cost sensitivity	Labels in {0,1} or one-hot Per-class or per- label weights	Mitigates imbalance by up-weighting minority classes Easily integrated into CE framework	Requires careful weight tuning Large weights can destabilize train- ing
Label Smooth- ing	Large-scale multi- class Reduces overcon- fidence	One-hot or near one-hot Uses small $\epsilon$	Improves general- ization Penalizes extreme confidence	Needs $\epsilon$ tuning May slow convergence
NLL	Equivalent to CE with one-hot General classification	Typically one-hot labels Probability match- ing	Interpretable as - log(true prob.) Common in likelihood frameworks	Suffers from imbalance Overconfidence = large gradient
PolyLoss	General frame- work Extensions of CE, focal loss	Binary or multi- class Flexible via poly- nomial terms	Tunable sensitiv- ity Additional hyper- parameter(s)	Requires careful coefficient choice Increases com- plexity
Hinge	Margin-based classification SVM-style learn- ing	Binary labels {-1, 1} Decision bound- ary approach	Encourages margin maximization Zero loss beyond margin	Not probabilistic Less suited for probabilistic out- puts

### 3.1.1 Binary Cross-Entropy Loss (BCE)

Binary Cross-Entropy (BCE), sometimes referred to as Log Loss, is a standard loss function for binary classification [59]. Consider a dataset  $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$  where each label  $y_i \in \{0, 1\}$ . Let  $\hat{p}_i = f_{\theta}(\mathbf{x}_i)$  be the model's predicted probability of  $y_i = 1$ . The BCE loss for a single instance is given by

$$L(y_i, \hat{p}_i) = -\left[y_i \log(\hat{p}_i) + (1 - y_i) \log(1 - \hat{p}_i)\right].$$
 (28)

In practice, the average loss over the dataset is minimized:

$$\mathcal{L}_{BCE}(\theta) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, \hat{p}_i).$$
(29)

# **Properties:**

- Information-Theoretic Origin: Cross-entropy measures the distance between two Bernoulli distributions: the true distribution P(y) and the predicted distribution Q(y). Minimizing it encourages  $\hat{p}_i$  to match the empirical probabilities of the labels.
- **Differentiability:** BCE is differentiable w.r.t.  $\hat{p}_i \in (0,1)$ . When  $\hat{p}_i = \sigma(z_i)$  (the logistic sigmoid), gradient-based methods such as Stochastic Gradient Descent (SGD) or its variants (Adam, RMSProp, etc.) naturally follow.
- Punishing Confident Mistakes: If  $\hat{p}_i$  is close to 1 but  $y_i = 0$ , or vice versa, the logarithmic term  $\log(\hat{p}_i)$  or  $\log(1 \hat{p}_i)$  becomes large in magnitude, heavily penalizing over-confident errors.
- Class Imbalance: In datasets where one class significantly outnumbers the other, BCE can lead to biased models. A *Weighted BCE* variant alleviates this:

$$L_{\text{weighted}}(y_i, \hat{p}_i) = -\left[w_1 y_i \log(\hat{p}_i) + w_0 (1 - y_i) \log(1 - \hat{p}_i)\right], \tag{30}$$

where  $w_1$  and  $w_0$  are class-specific weights, often taken inversely proportional to class frequencies. Weighted Cross-Entropy is discussed with more details in Section 3.1.4.

#### 3.1.2 Categorical Cross-Entropy Loss (CCE)

Categorical Cross-Entropy (CCE), also known as Multi-Class Log Loss, extends BCE to handle  $\mathbf{y}_i \in \{0,1\}^C$  in a one-hot representation across C classes. Let  $\hat{\mathbf{p}}_i = [\hat{p}_{i,1}, \dots, \hat{p}_{i,C}]$  be the predicted probability distribution for sample i. Then, the per-sample loss is

$$L(\mathbf{y}_i, \hat{\mathbf{p}}_i) = -\sum_{i=1}^C y_{i,j} \log(\hat{p}_{i,j}), \tag{31}$$

where  $y_{i,j} \in \{0,1\}$  and  $\sum_{j} y_{i,j} = 1$ . Averaging over n samples yields the dataset-level loss:

$$\mathcal{L}_{CCE}(\theta) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{C} y_{i,j} \log(\hat{p}_{i,j}).$$
 (32)

### **Key Points:**

• Softmax Output: In neural networks, the model typically outputs logits  $\mathbf{z}_i \in \mathbb{R}^C$ . A softmax function

$$\hat{p}_{i,j} = \frac{\exp(z_{i,j})}{\sum_{k=1}^{C} \exp(z_{i,k})}$$
(33)

ensures  $\sum_{i} \hat{p}_{i,j} = 1$ .

- Log-Likelihood Interpretation: Minimizing  $\mathcal{L}_{CCE}$  is equivalent to maximizing the log-likelihood of the correct class
- **Penalty for Confident Misclassifications:** As in BCE, CCE heavily penalizes predictions that assign large probability to an incorrect class.
- One-Hot Encoding Requirements: Standard CCE assumes one-hot (or probability-like) target vectors. For integer-encoded labels, *Sparse CCE* (Section 3.1.3) is often more efficient.

# 3.1.3 Sparse Categorical Cross-Entropy Loss

Sparse Categorical Cross-Entropy (Sparse CCE) addresses scenarios where class labels are given as integers  $y_i \in \{1, 2, \dots, C\}$  rather than one-hot vectors. Let  $\hat{p}_{i,j}$  be the predicted probability of class j for sample i. The individual sample loss is:

$$L_{\text{sparse}}(y_i, \hat{\mathbf{p}}_i) = -\log(\hat{p}_{i, y_i}), \tag{34}$$

 $L_{\text{sparse}}\big(y_i,\hat{\mathbf{p}}_i\big) = -\log\big(\hat{p}_{i,\,y_i}\big),$  where  $\hat{p}_{i,\,y_i}$  is the probability assigned to the correct class  $y_i$ . The overall loss is:

$$\mathcal{L}_{\text{SparseCCE}}(\theta) = -\frac{1}{n} \sum_{i=1}^{n} \log(\hat{p}_{i, y_{i}}). \tag{35}$$

#### **Usage**

- Efficiency: In classification tasks with a large number of classes, one-hot encoding can be memory-intensive. Sparse CCE indexes the correct probability directly, avoiding the need for full one-hot vectors.
- Similarity to Standard CCE: From an optimization standpoint, Sparse CCE is equivalent to CCE but uses integer labels to look up  $\hat{p}_{i...y_i}$ .
- **Gradient Computation:** As with other cross-entropy losses, it remains fully differentiable, allowing gradient-based training in neural networks.

#### 3.1.4 Weighted Cross-Entropy (WCE)

Weighted Cross-Entropy is a variant of the standard cross-entropy loss designed to address class imbalance by assigning greater importance to minority classes [60, 61]. This approach helps the model focus more on infrequent classes, where misclassification is often costlier or more critical.

**Binary Weighted Cross-Entropy:** In binary classification with labels  $y_i \in \{0,1\}$ , let  $\hat{p}_i \in [0,1]$  be the model's predicted probability of  $y_i = 1$ . Denote  $\alpha_1$  and  $\alpha_0$  as class-specific weights for the positive and negative classes, respectively. The *per-instance* weighted cross-entropy loss is:

$$L_{\text{WCE}}(y_i, \hat{p}_i) = -\left[\alpha_1 y_i \log(\hat{p}_i) + \alpha_0 (1 - y_i) \log(1 - \hat{p}_i)\right]. \tag{36}$$

The overall loss is obtained by averaging over n samples:

$$\mathcal{L}_{\text{WCE}} = \frac{1}{n} \sum_{i=1}^{n} L_{\text{WCE}}(y_i, \hat{p}_i). \tag{37}$$

**Multi-Class Weighted Cross-Entropy:** For multi-class problems with C classes, let  $\hat{\mathbf{p}}_i = [\hat{p}_{i,1}, \dots, \hat{p}_{i,C}]^{\top}$  be the predicted probability distribution over classes for sample i, and let  $\mathbf{y}_i$  be the one-hot encoded true label. Denote  $\alpha_c$  as the weight for class c. The *per-instance* weighted cross-entropy loss then generalizes to:

$$L_{\text{WCE}}(\mathbf{y}_i, \hat{\mathbf{p}}_i) = -\sum_{c=1}^{C} \alpha_c \, y_{i,c} \, \log(\hat{p}_{i,c}). \tag{38}$$

Again, the total loss is the mean over all samples:

$$\mathcal{L}_{\text{WCE}} = \frac{1}{n} \sum_{i=1}^{n} L_{\text{WCE}}(\mathbf{y}_i, \hat{\mathbf{p}}_i). \tag{39}$$

#### **Choosing Class Weights:**

- Inverse Frequency: A common heuristic is to set  $\alpha_c \propto 1/f_c$ , where  $f_c$  is the (relative) frequency of class c. Less frequent classes thus receive higher weights.
- **Proportional to Cost:** In cost-sensitive contexts, the weights may reflect the application's error cost for each class. For instance, in medical imaging, an under-diagnosed disease class might receive a large weight.
- Normalization: Often, weights  $\{\alpha_c\}$  are normalized so that  $\sum_c \alpha_c = C$ . This helps stabilize gradients if some  $\alpha_c$  is very large.

## **Interpretation and Practical Use:**

- Class Imbalance Mitigation: Weighted cross-entropy helps ensure that minority classes have sufficient impact on the gradient during training, reducing model bias toward majority classes.
- Extended to Multi-Label Scenarios: In multi-label classification, each label can be assigned a weight based on its frequency or importance. Then, a binary weighted cross-entropy term is computed per label and summed.
- Comparison with Other Losses: While WCE often provides a straightforward fix for imbalance, more advanced approaches like Focal Loss (Section 5.1.6) or Polyloss (Section 3.1.7) may also be used, especially if the number of easy vs. hard examples significantly differs.

#### **Limitations:**

- **Hyperparameter Tuning:** Determining a suitable weighting scheme can require extensive experimentation; overly large weights might harm overall performance or cause unstable training.
- **Distribution Shifts:** If the class distribution changes during deployment (e.g., a new data source), fixed weights may become suboptimal.

Weighted Cross-Entropy thus extends the standard cross-entropy by introducing per-class or per-label weighting factors, allowing models to place greater emphasis on classes of higher importance or lower frequency. This often yields more balanced decision boundaries when class distributions are skewed, aligning better with real-world objectives.

### 3.1.5 Cross-Entropy Loss with Label Smoothing

Label smoothing is a regularization strategy that softens the target distributions for multi-class classification [62, 63]. Instead of training the model to output probability 1 for the ground truth class and 0 for others, the targets are replaced by a smoothed distribution, reducing the model's overconfidence.

**Formulation:** Let  $\epsilon \in [0, 1]$  be the smoothing factor, C the number of classes, and  $\hat{\mathbf{p}}_i$  the predicted probability vector for sample i. In a one-hot label scenario, the original target  $\mathbf{y}_i$  has  $y_{i,j} = 1$  for the true class j, and 0 otherwise. Label smoothing modifies  $\mathbf{y}_i$  to  $\tilde{\mathbf{y}}_i$ :

$$\tilde{y}_{i,j} = (1 - \epsilon) y_{i,j} + \frac{\epsilon}{C}. \tag{40}$$

Then the usual cross-entropy is computed with  $\tilde{\mathbf{y}}_i$ :

$$L_{\text{labelSmooth}} = -\sum_{j=1}^{C} \tilde{y}_{i,j} \log(\hat{p}_{i,j})$$

$$= -\sum_{j=1}^{C} \left[ (1 - \epsilon) y_{i,j} + \frac{\epsilon}{C} \right] \log(\hat{p}_{i,j}). \tag{41}$$

#### Advantages:

- Overconfidence Mitigation: By disallowing the model to assign a probability of exactly 1 to any single class, label smoothing prevents overly confident predictions.
- **Better Generalization:** Empirical evidence suggests label smoothing helps models avoid overfitting on noisy data or unrepresentative training samples.

Typical values for  $\epsilon$  range from 0.05 to 0.2. Tuning  $\epsilon$  can be done via validation performance.

The model learns to distribute some probability mass to non-ground-truth classes, effectively acting as a regularizer. Large  $\epsilon$  might excessively blur the distinction between classes, whereas very small  $\epsilon$  may not significantly affect overconfidence.

## 3.1.6 Negative Log-Likelihood

In many classification tasks, the ground-truth distribution **p** for a given sample is represented by a one-hot encoded vector, meaning exactly one class has probability 1 and all others 0. Let **q** be the model's predicted distribution over classes. The *cross-entropy* between **p** and **q** for a single instance is given by

$$H(\mathbf{p}, \mathbf{q}) = -\sum_{i=1}^{C} p_i \log(q_i), \tag{42}$$

where C is the total number of classes,  $p_i \in \{0,1\}$  and  $\sum_{i=1}^{C} p_i = 1$ . Since  $\mathbf{p}$  is one-hot, exactly one  $p_i$  equals 1, and all others are 0. Hence, (42) simplifies to

$$H(\mathbf{p}, \mathbf{q}) = -\log(q_{y_t}),\tag{43}$$

where  $y_t$  denotes the true class index. This formulation is equivalent to the Negative Log-Likelihood (NLL), often written as

$$NLL(\mathbf{x}, y_t) = -\log(p(y_t \mid \mathbf{x})). \tag{44}$$

Thus, minimizing cross-entropy for a one-hot true distribution is equivalent to minimizing the negative log-likelihood. Both metrics incentivize the model to place high probability on the correct class  $y_t$ . In modern deep learning frameworks, the softmax activation is typically used to produce  $\mathbf{q}$ , ensuring  $\sum_i q_i = 1$  and  $q_i \in (0,1)$ . By descending the gradient of NLL, the model learns to adjust its parameters so that  $q_{y_t} \to 1$  for the correct class label.

### 3.1.7 Polyloss

*PolyLoss* [64] proposes a generalized loss function family that encompasses widely used losses like Cross-Entropy (CE) and Focal Loss as special cases. The key idea is to represent a loss function as a linear combination of polynomial expressions of the predicted probabilities, thus offering a flexible framework for adapting loss behavior based on task requirements.

**General Formulation:** Let  $p_t$  be the predicted probability of the true class t for a single instance. PolyLoss posits that one may write the loss function L as

$$L = \sum_{n=0}^{N} \alpha_n \left( 1 - p_t \right)^n, \tag{45}$$

where  $\alpha_n$  are polynomial coefficients controlling how heavily each  $(1-p_t)^n$  term contributes. Large values of n cause the loss to become more sensitive to small deviations in  $p_t$ ; thus, the  $\alpha_n$  sequence provides a knob to balance or emphasize certain error behaviors.

**Poly-1 Loss Variant:** A prominent example within this family is the *Poly-1 loss*, which adds a single linear term on top of the standard Cross-Entropy:

$$L_{\text{Poly}-1} = \text{CE}(\mathbf{y}, \hat{\mathbf{p}}) + \epsilon (1 - p_t), \tag{46}$$

where  $CE(\cdot)$  is the usual cross-entropy, and  $\epsilon$  is a hyperparameter. Note that setting  $\epsilon=0$  reduces Poly-1 to the standard cross-entropy loss. When  $\epsilon>0$ , the term  $\epsilon\,(1-p_t)$  penalizes confidently wrong predictions more heavily and can mitigate overfitting—particularly in imbalanced classification scenarios or tasks requiring high precision.

# **Use Cases and Advantages:**

- Imbalanced Datasets: By fine-tuning polynomial coefficients  $\alpha_n$ , PolyLoss can heighten penalties on hard examples or minority classes, analogous to focal loss but with more flexible control.
- Task-Specific Adaptation: Custom polynomial expansions may help the model learn from domain-specific misclassifications (e.g., medical diagnoses where false negatives are costlier than false positives).
- Minimal Hyperparameter Tuning: Poly-1 introduces only one additional parameter  $\epsilon$ . Compared to more complex losses (e.g., focal loss with multiple parameters), PolyLoss can simplify hyperparameter search.

By providing a Taylor-series-like expansion for the loss, PolyLoss helps unify and extend existing classification loss functions within a single polynomial framework.

#### 3.1.8 Hinge Loss

*Hinge Loss* is frequently employed in *maximum-margin* classification tasks, particularly in Support Vector Machines (SVMs) [65]. For a binary classification task with labels  $y \in \{-1, +1\}$ , the hinge loss for a single instance is

$$L(y, f(\mathbf{x})) = \max(0, 1 - y f(\mathbf{x})), \tag{47}$$

where  $f(\mathbf{x})$  is the raw (unthresholded) model output or decision function (often the signed distance from the decision boundary). The *margin* for the sample  $\mathbf{x}$  is given by the product  $y f(\mathbf{x})$ . A correct classification with a sufficient margin yields a positive product above 1, making the loss zero.

#### **Characteristics:**

- Margin Maximization: Hinge loss enforces not only that the prediction is correct  $(sign(f(\mathbf{x})) = y)$  but also that the margin  $|f(\mathbf{x})|$  is at least 1. Predictions on the correct side but with margin less than 1 still incur a positive loss.
- Linear Penalty: When  $y f(\mathbf{x}) < 1$ , the hinge loss increases linearly with the distance  $1 y f(\mathbf{x})$ . This *linear* penalty differs from cross-entropy's *logarithmic* penalty, often resulting in sparser gradients.
- Extension to Multi-Class: Though primarily associated with binary SVMs, hinge loss can be extended to multi-class settings (e.g., structured SVMs) [66], where sophisticated loss functions depend on inter-class margins.

#### **Variants:**

Squared Hinge Loss:

$$L_{\text{sq}}(y, f(\mathbf{x})) = \left(\max(0, 1 - y f(\mathbf{x}))\right)^{2},\tag{48}$$

penalizes margin violations more severely and can lead to smoother gradient behavior near the margin boundary.

Overall, hinge loss is valued for promoting large margins, which often correlates with better generalization in classification tasks that fit well into a maximum-margin framework.

#### 3.2 Classification Performance Metrics

This section covers metrics for evaluating classification models. Table 7 summarizes these metrics, with details in subsequent subsections. Table 8 offers guidelines for selecting the right metric considering usage, data traits, pros, and cons, providing support for choosing the best metric for varied data and performance needs.

## 3.2.1 Confusion Matrix

The *confusion matrix* is a fundamental tool for visualizing the performance of a binary classifier. It tallies the number of times each combination of predicted and actual labels occurs. For binary classification (with possible true labels {Positive, Negative}), the confusion matrix is a  $2 \times 2$  table, as shown in Table 9.

#### **Definitions:**

- True Positive (TP): The model correctly classifies a positive instance as positive.
- True Negative (TN): The model correctly classifies a negative instance as negative.
- False Positive (FP): Also known as *Type I error*, the model incorrectly classifies a negative instance as positive (a "false alarm").
- False Negative (FN): Also known as *Type II error*, the model incorrectly classifies a positive instance as negative (a "miss").

Many performance metrics (e.g., Accuracy, Precision, Recall, F1-score) directly derive from (TP, TN, FP, FN). Observing the confusion matrix highlights where the classifier errs most frequently, enabling targeted model improvements.

### **Derived Metrics:**

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN},$$
(49)

Precision = 
$$\frac{TP}{TP + FP}$$
, Recall =  $\frac{TP}{TP + FN}$ , (50)

$$F_{1}\text{-score} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$
 (51)

Common Name	Other Names	Abbr	Definitions	Interpretations
True Positive	Hit	TP	True Sample la- beled true	Correctly labeled True Sample
True Negative	Rejection	TN	False Sample la- beled false	Correctly labeled False sample
False Positive	False alarm,	FP	False sample la-	Incorrectly labeled False sample
El M	Type I Error	TN I	beled True	T (1.1.1.177)
False Negative	Miss, Type II Error	FN	True sample labeled false	Incorrectly label True sample
Recall	True Positive Rate	TPR	TP/(TP+FN)	% of True samples correctly labeled
Specificity	True Negative	SPC,	TN/(TN+FP)	% of False samples correctly la-
- · ·	Rate	TNR		beled
Precision	Positive Predic-	PPV	TP/(TP+FP)	% of samples labeled True that
Negative Pre-	tive Value	NPV	TN/(TN+FN)	really are True % of samples labeled False that
dictive Value		TVI V	110(1101110)	really are False
False Negative		FNR	FN/(TP+FN)=1-	% of True samples incorrectly
Rate			TPR	labeled
False Positive	Fall-out	FPR	FP/(FP+FN)=1-	% of False samples incorrectly
Rate False Discovery		FDR	SPC FP/(TP+FP)=1-	labeled % of samples labeled True that
Rate		TDK	PPV	are really False
True Discovery		TDR	FN/(TN+FN)=1-	% of samples labeled False that
Rate			NPV	are really True
Accuracy		ACC	$\frac{(TP+TN)}{(TP+TN+FP+FN)}$	Percent of samples correctly la- beled
F1 Score		F1	$\frac{(2*TP)}{((2*TP)+FP+FN)}$	Approaches 1 as errors decline

Table 7: Metrics used in classification task.

These metrics express different aspects of classification performance. For instance, *Precision* measures how often predicted positives are correct, while *Recall* indicates the fraction of positive instances the model successfully identifies.

#### 3.2.2 Confusion Matrix in Multi-Class Classification

For classification tasks involving N classes, the confusion matrix generalizes to an N  $\times$  N table. Each row i corresponds to the true class i, while each column j corresponds to the model's predicted class j. Let M[i,j] denote the number of instances whose ground-truth label is class i but which the model predicts as class j. Thus, each entry of the matrix satisfies:

$$\sum_{j=1}^{N} M[i, j] = \text{(total number of class } i \text{ instances)}, \tag{52}$$

$$\sum_{i=1}^{N} M[i,j] = \text{(total number of instances predicted as class } j\text{)}. \tag{53}$$

The diagonal entries M[i,i] represent correct classifications of class i. Off-diagonal entries M[i,j] ( $i \neq j$ ) reveal misclassifications from class i to class j.

#### **Interpretation in Multi-Class Problems:**

- ullet Diagonal Entries: High values on the main diagonal (i.e., M[i,i]) imply strong class-specific accuracy.
- Row Summations: For row i,  $\sum_{j=1}^{N} M[i,j]$  is the total number of class-i samples. Comparing M[i,i] with this row sum highlights the fraction of class-i samples misclassified.

Table 8: Guidelines for selecting a classification metric based on usage, data characteristics, advantages, and limitations.

Metric	Usage	Data Characteristics	Advantages	Limitations	
Confusion Matrix	Summarizes model outputs (TP, FP, FN, TN) Assists in error local- ization	Binary/multi-class Larger matrices for many classes	Straightforward error analysis Pinpoints specific mis- classifications	Not a single- performance measure Harder to compare multiple models	
Accuracy	Overall fraction of correct predictions	Mostly reliable if classes are balanced	Easy to interpret and compute	Misleading under class imbalance Masks type of errors	
Precision	Reliability of positive predictions	Works in binary/multi- class Affected by imbal- anced positives	Reduces false positives Critical where false alarms are costly	Ignores missed positives Must be combined with recall	
Recall	Proportion of actual positives detected	Binary/multi-class Sensitive to class im- balance	Minimizes missed positives Key in diagnos- tics/fraud	Can increase false pos- itives Requires balance with precision	
F1-Score	Harmonic mean of precision and recall	Helpful in skewed data Binary/multi-class	Balances false posi- tives/negatives More insightful than accuracy alone	Weighs errors equally Less useful if cost of errors differs	
Specificity	Proportion of actual negatives correctly identified	Common in binary scenarios Medical or screening tests	Reduces false alarms Complements recall	Neglects how positives are handled Less intuitive for balanced data	
False Positive Rate (FPR)	Fraction of negatives incorrectly labeled positive	Part of ROC analysis Extensible to multi- class (OvA, OvO)	Shows Type I error Useful with TPR (ROC curves)	Does not capture false negatives Needs TPR to com- plete the picture	
Negative Predictive Value (NPV)	Accuracy of predicted negatives	Effective in medical tests Affected by prevalence of negatives	Ensures few missed positives Valuable in screening	Highly sensitive to class ratio Less common as a primary metric	
False Discovery Rate (FDR)	Fraction of positives that are false	Relevant when cost of false positives is high			
Precision- Recall Curve	Visualizes trade-off between precision, recall	Suited for imbalanced data Can do one-vs-all for multi-class	data across thresholds across		
AUC- ROC	Combines TPR vs. FPR across thresholds	Standard in binary classification Extended via OvA or OvO	Threshold- independent Easy model compari- son	May be optimistic with skewed data Lacks cost/context of errors	

• Column Summations: For column j,  $\sum_{i=1}^{N} M[i,j]$  shows how many instances the model labeled as class j. When large row sums align with high misclassifications in a particular column, it suggests confusion between certain class pairs or subsets.

**Example Visualization:** Figure 1 illustrates a confusion matrix heatmap for three classes (A, B, C). Each cell's color intensity is proportional to M[i,j]. High diagonal intensities imply strong performance, whereas off-diagonal cells reveal specific misclassification patterns. For instance:

•  $A \rightarrow A$ : 95 correct predictions for class A,  $A \rightarrow B$ : 2 misclassifications,  $A \rightarrow C$ : 3 misclassifications.

Table 9: Confusion Matrix for a Binary Classification Task.

	<b>Predicted Positive</b>	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Table 10: Example of a Confusion Matrix for a Three-Class Classification Task. Diagonal entries M[i,i] indicate correct predictions for class i. Off-diagonal entries illustrate misclassifications.

	Pred. A	Pred. B	Pred. C
Actual: A Actual: B Actual: C	$M[1,1] \ M[2,1] \ M[3,1]$	$M[1,2] \ M[2,2] \ M[3,2]$	$M[1,3] \\ M[2,3] \\ M[3,3]$

- $B \rightarrow B$ : 90 correct predictions,  $B \rightarrow A$ : 4 errors,  $B \rightarrow C$ : 6 errors.
- $C \rightarrow C$ : 85 correct identifications,  $C \rightarrow A$ : 7 errors,  $C \rightarrow B$ : 8 errors.

Observing that Class C is more frequently misclassified as Class B than A can guide adjustments in feature engineering or model capacity to reduce that confusion.

### 3.2.3 Accuracy

Accuracy is one of the most widely used metrics for classification tasks and is defined as the ratio of correctly classified instances to the total number of instances [67]. Formally,

$$Accuracy = \frac{Correctly Classified Samples}{Total Number of Samples}.$$
 (54)

When expressed in terms of the confusion matrix for binary classification, where the elements (TP, TN, FP, FN) denote true positives, true negatives, false positives, and false negatives respectively, Accuracy becomes

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}.$$
 (55)

**Imbalance Consideration:** Although Accuracy is intuitive, it can be misleading in cases of severe class imbalance. For example, a classifier that predicts *negative* for all instances in a dataset where 99% of samples truly are negative would achieve 99% accuracy but fail to detect the minority positive class entirely.

**Multi-Class Accuracy:** For a classification problem with C classes, Accuracy generalizes by summing the number of correct predictions across all classes and dividing by the total number of samples:

$$Accuracy_{\text{multi}} = \frac{\sum_{i=1}^{C} \text{Correct}_i}{\text{Total Samples}},$$
(56)

where  $Correct_i$  is the number of correctly predicted samples for class i. Although straightforward, multi-class Accuracy still suffers from the same imbalance issues if certain classes are underrepresented.

**Balanced Accuracy:** To mitigate imbalance effects, *Balanced Accuracy* computes the mean of Recall (also known as Sensitivity) across classes:

Balanced Accuracy = 
$$\frac{1}{C} \sum_{i=1}^{C} \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i},$$
 (57)

where  $TP_i$  and  $FN_i$  respectively denote true positives and false negatives for class i. Balanced Accuracy ensures that each class's performance contributes equally, making it particularly useful in imbalanced multi-class problems.

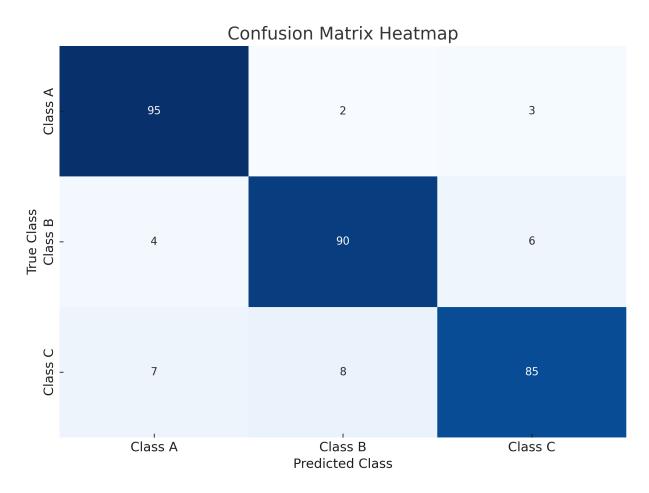


Figure 1: Confusion Matrix Heatmap for a Three-Class Classification Task. Diagonal entries show correct predictions, off-diagonal entries represent misclassifications. Color intensity correlates with the number of samples. This format helps identify model strengths (e.g., class A has few misclassifications overall) and weaknesses (e.g., class C is misclassified as B relatively frequently).

## 3.2.4 Precision, PPV, or TDR

*Precision*, also referred to as *Positive Predictive Value (PPV)* or *True Discovery Rate (TDR)*, quantifies the fraction of predicted positives that are genuinely positive [68]. In binary classification:

$$Precision = \frac{TP}{TP + FP},$$
(58)

where TP is the number of true positives and FP is the number of false positives. High Precision indicates that among all samples predicted as positive, a large proportion are correct—an essential property in domains where false alarms are costly (e.g., medical screenings, fraud detection).

**Precision vs. Recall Trade-off:** Maximizing Precision alone can be misleading if many actual positives are missed. Often, Precision is analyzed together with Recall (Section 3.2.5) to balance the cost of false positives against the cost of false negatives.

**Precision in Multi-Class Classification:** In multi-class settings with N classes, Precision is computed on a per-class basis and then averaged. Two common averaging schemes are:

• Macro-average Precision:

$$Macro-Precision = \frac{1}{N} \sum_{i=1}^{N} \frac{TP_i}{TP_i + FP_i},$$
 (59)

where  $TP_i$  and  $FP_i$  refer to class i. Macro-average treats all classes equally, regardless of class frequency.

• Micro-average Precision:

$$Micro-Precision = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N} (TP_i + FP_i)}.$$
 (60)

Micro-average aggregates global TP and FP counts across all classes, thus weighting the classes by their support (i.e., frequency).

Macro-average Precision is often preferred if one wants to measure performance equally across all classes, whereas Micro-average is influenced more by classes with higher sample counts.

### 3.2.5 Recall, Sensitivity, or True Positive Rate (TPR)

*Recall*, also known as *Sensitivity* or *True Positive Rate (TPR)*, measures the proportion of actual positives correctly identified by the model [68]. In binary classification:

$$Recall = \frac{TP}{TP + FN},\tag{61}$$

where TP denotes true positives and FN denotes false negatives. A high Recall means the model correctly identifies most of the positive cases, which is critical in scenarios where missing a positive instance (e.g., a disease diagnosis) is costly.

**Recall in Multi-Class Classification:** Similar to Precision, Recall must be evaluated per class and then averaged to gain an overall measure in multi-class settings:

• Macro-average Recall:

$$Macro-Recall = \frac{1}{N} \sum_{i=1}^{N} \frac{TP_i}{TP_i + FN_i}.$$
 (62)

• Micro-average Recall:

$$Micro-Recall = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N} (TP_i + FN_i)}.$$
 (63)

*Macro-average* gives equal weight to each class, while *Micro-average* aggregates the totals across classes. If class imbalance is severe, the choice of averaging method significantly affects the final reported Recall.

**Precision–Recall Trade-off:** Achieving high Recall often entails predicting more positives, which may inflate FP and thus reduce Precision. Conversely, focusing on high Precision can reduce Recall by being more conservative about labeling samples as positive. The *F1-score* (or F-measure) is a common single metric that seeks to balance the two:

$$F_1 = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$
 (64)

In practice, the choice to emphasize Precision or Recall (or a specific balance of both) depends on the cost of false positives vs. false negatives in the application domain.

#### 3.2.6 F1-Score

The *F1-score* is a statistical measure that combines *Precision* and *Recall* into a single indicator of a classification model's overall performance [68]. Specifically, the F1-score is the *harmonic mean* of Precision (Prec) and Recall (Rec), given by

$$F_1 = 2 \cdot \frac{\text{Prec} \times \text{Rec}}{\text{Prec} + \text{Rec}}.$$
 (65)

Because the harmonic mean heavily penalizes extreme disparities between Precision and Recall, the F1-score ensures neither metric is ignored. A higher F1-score indicates a more balanced classifier, capable of both correctly labeling the majority of positive instances (high Recall) and making relatively few false-positive predictions (high Precision).

#### **Use Cases and Limitations:**

- **Imbalanced Datasets:** F1 is especially meaningful when positive cases are rare (e.g., rare disease detection, fraud detection) and where a high Accuracy can be misleading.
- Lack of Cost Differentiation: F1 does not differentiate between FP (false positives) and FN (false negatives) beyond their impact on Precision and Recall. If the cost of false positives greatly outweighs that of false negatives (or vice versa), a more specialized metric may be necessary.

*F-Beta Score:* An extension to F1 is the  $F_{\beta}$ -score, which provides a mechanism for weighting Precision and Recall differently. It is defined as

$$F_{\beta} = (1 + \beta^2) \cdot \frac{\text{Prec} \times \text{Rec}}{\beta^2 (\text{Prec}) + \text{Rec}},$$
(66)

where  $\beta \in \mathbb{R}^+$ . When  $\beta > 1$ , Recall receives more emphasis; when  $\beta < 1$ , Precision is emphasized. This flexibility allows the metric to be tuned to application-specific costs of false positives and false negatives, making  $F_{\beta}$  a powerful tool in domains with asymmetrical error concerns.

## 3.2.7 Specificity or True Negative Rate (TNR)

Specificity, sometimes called the *True Negative Rate (TNR)*, measures the proportion of actual negatives correctly identified as negative [69]. In binary classification, it is computed as:

Specificity = 
$$\frac{TN}{TN + FP}$$
, (67)

where TN (true negatives) is the count of correctly identified negative instances, and FP (false positives) is the count of negative instances incorrectly labeled as positive.

### **Interpretation and Relevance:**

- Medical Diagnostics: High Specificity is vital when the cost of a false positive is high (e.g., unnecessary anxiety, invasive follow-up tests).
- **Financial Fraud Detection:** Overzealous blocking of legitimate transactions (FP) can harm customers, making Specificity important.
- **Complementary Metric to Recall:** Whereas Recall (or Sensitivity) concentrates on capturing actual positives, Specificity focuses on *rejecting* actual negatives correctly. In many cases, both metrics are examined together to capture a classifier's trade-off between false negatives and false positives.

### **Example Trade-offs:**

- A test with high Recall but low Specificity may flag most positives but incur many false alarms (FP).
- A test with high Specificity but low Recall will likely miss numerous actual positives, but it will rarely mislabel negatives as positives.

When combined with Recall, Specificity paints a more complete picture of a classifier's performance, aiding in balancing error types in high-stakes domains.

#### 3.2.8 False Positive Rate (FPR)

False Positive Rate (FPR) measures the proportion of actual negatives that are incorrectly predicted as positive. Also known as the *Type I Error rate*, it is an important complement to *Specificity* (also called the True Negative Rate, TNR). Formally, from the confusion matrix elements TN (true negatives) and FP (false positives), FPR is defined as

$$FPR = \frac{FP}{TN + FP}.$$
 (68)

The relationship between FPR and Specificity is given by

$$FPR = 1 - Specificity,$$
 (69)

since

Specificity = 
$$\frac{TN}{TN + FP}$$
. (70)

**Threshold Dependence:** FPR is heavily influenced by the decision threshold used to classify instances as positive vs. negative. In probabilistic models:

- Lower Threshold: More instances are labeled as positive, potentially increasing FP, hence increasing FPR.
- **Higher Threshold:** Fewer instances are labeled as positive, likely reducing FP and hence FPR.

**Practical Importance:** FPR is critical in applications where the cost of a false positive can be high:

- Financial Fraud Detection: A high FPR means legitimate transactions are flagged as fraudulent, harming customer satisfaction and possibly incurring fees.
- Medical Tests: A high FPR implies many healthy individuals are flagged as diseased, leading to unnecessary, invasive follow-up tests and anxiety.

**Usage in ROC Analysis:** FPR is commonly plotted on the *x*-axis of a *Receiver Operating Characteristic* (ROC) curve against the *True Positive Rate* (TPR) on the *y*-axis. Varying the classification threshold traces out the ROC curve. An ideal model aims for high TPR and low FPR. *Area Under the ROC Curve* (AUC) then provides a single, threshold-independent measure of how well the model balances TPR and FPR (see Section 3.2.13).

### 3.2.9 Negative Predictive Value (NPV)

Negative Predictive Value (NPV) measures the proportion of negative predictions that are correct. In other words, among all instances labeled negative by the model, NPV indicates the fraction that truly are negative. From the confusion matrix elements TN (true negatives) and FN (false negatives), NPV is defined as

$$NPV = \frac{TN}{TN + FN}. (71)$$

# **Interpretation and Relevance:**

- **High-Impact Misses (False Negatives):** NPV is central in contexts where missing a positive case (FN) is particularly detrimental (e.g., undiagnosed disease). A high NPV means the model rarely overlooks actual positives when it predicts negative.
- **Prevalence Sensitivity:** Similar to Positive Predictive Value (PPV), NPV is sensitive to the prevalence of the target condition. If negatives are abundant, a moderate TN plus a small FN can yield a high NPV.
- Comparison with PPV: PPV measures the correctness of positives, whereas NPV measures the correctness of negatives. Together, they provide a holistic view of a model's predictive reliability for both classes.

### **Use Cases:**

- **Medical Diagnostics (Rare Diseases):** In screening for a rare disease, FN can be fatal if a patient goes untreated. Hence, high NPV (few missed positives when labeled negative) is crucial.
- Quality Control: In manufacturing, incorrectly labeling defective items as "non-defective" (FN) can allow faulty products to pass inspection, incurring recalls or safety issues.

**Prevalence Effects:** Because NPV depends on the ratio of actual negatives, changes in the underlying data distribution (e.g., disease prevalence in a population) can affect NPV significantly. In designing classification systems or screening tools for low-prevalence conditions, ensuring a sufficiently high NPV often necessitates strategies to reduce FN, such as lowering the threshold for a positive test or employing further confirmatory tests.

#### 3.2.10 False Discovery Rate (FDR)

False Discovery Rate (FDR) quantifies the proportion of predicted positives that are actually negative. It is defined in terms of FP (false positives) and TP (true positives) as:

$$FDR = \frac{FP}{TP + FP}.$$
 (72)

Thus, FDR = 1 - Precision, highlighting its relationship to Precision (PPV).

### **Interpretation:**

- **Positive Class Reliability:** A low FDR indicates that among all instances predicted positive, only a small fraction are false positives. This is essential in domains that cannot afford erroneous positive alerts.
- Threshold Dependence: Like other metrics (e.g., FPR), FDR may shift as one adjusts the classification threshold. A more lenient threshold often inflates FP, driving FDR higher.

#### **Domain-Specific Applications:**

- Scientific Research / Multiple Hypothesis Testing: FDR control is critical to limit the number of false discoveries—untrue hypotheses declared significant. Techniques like the Benjamini–Hochberg procedure specifically target a controlled FDR in statistical inference.
- Medical Diagnostics & Fraud Detection: Minimizing FDR helps ensure that most of the flagged positives (TP + FP) are actually correct positives (TP). Excessive false discoveries can lead to wasted resources, unnecessary treatments, or alarm fatigue.

**FDR vs. FPR:** While FPR is the proportion of false positives among all actual negatives, FDR is the proportion of false positives among predicted positives. The choice between focusing on FDR or FPR depends on whether one cares more about:

- FPR: The effect of mislabeling actual negatives.
- FDR: The trustworthiness of declared positives.

In essence, FDR is invaluable for maintaining credibility in positive predictions. Where false alarms (false positives) are costly or resource-intensive, controlling FDR is a priority.

### 3.2.11 Precision-Recall Curve

The *Precision-Recall (PR) curve* visualizes the trade-off between *Precision* and *Recall* at varying decision thresholds for a binary classifier [68]. Recall that:

$$Precision = \frac{TP}{TP + FP}, \qquad Recall = \frac{TP}{TP + FN}, \tag{73}$$

where TP, FP, FN denote true positives, false positives, and false negatives, respectively.

### **Computing the PR Curve:**

- 1. Have a binary classifier that outputs a probability or score s(x) for the positive class.
- 2. For thresholds  $\tau$  in  $\{0, 0.01, 0.02, \dots, 1.0\}$ , predict "positive" if  $s(x) \ge \tau$ , otherwise "negative."
- 3. At each threshold, compute  $Precision(\tau)$  and  $Recall(\tau)$ .
- 4. Plot Recall $(\tau)$  on the x-axis vs. Precision $(\tau)$  on the y-axis.

A PR curve is thus a parametric plot of (Recall, Precision) pairs as  $\tau$  varies.

### **Interpretation of the PR Curve:**

- **Top-Right Corner** (1, 1): Corresponds to a hypothetical "perfect" classifier with no false positives and no false negatives. Approaching this corner indicates jointly high Precision and Recall across many thresholds.
- Area Under the PR Curve (AUC-PR): Summarizes the model's ability to maintain high Precision and high Recall simultaneously. Unlike the ROC curve, the baseline for AUC-PR depends on the fraction of positive instances in the data (class prevalence).
- Shape of the Curve: A steep initial slope often implies the model can achieve significantly higher Recall before Precision drops sharply. A more gradual slope suggests the model must sacrifice Precision in order to capture more positives (higher Recall).
- Model Comparison: Models whose PR curves lie consistently above another's generally exhibit better performance at a wide range of thresholds.

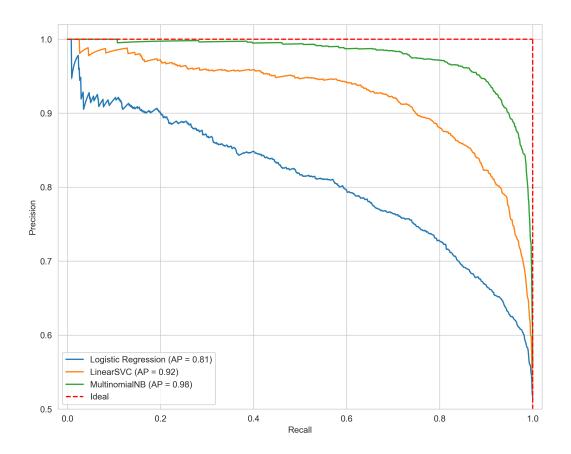


Figure 2: Precision-Recall curves for three models trained on the same dataset. The dashed line indicates ideal performance. Each curve's Average Precision (AP) provides a numerical summary akin to AUC-PR.

#### 3.2.12 Precision-Recall Curve in Multi-Class Classification

Although PR curves are traditionally formulated for binary classification, they can be extended to multi-class problems using a *One-vs-All (OvA)* approach for each class, treating that class as "positive" and merging all other classes as "negative."

### **Computing Multi-Class PR Curves:**

- 1. For each class  $c \in \{1, 2, ..., C\}$ , obtain probability scores  $s_c(x)$  indicating the model's confidence that x belongs to class c.
- 2. For each class c, treat it as the positive class and compute  $\operatorname{Precision}_c(\tau)$  and  $\operatorname{Recall}_c(\tau)$  across thresholds  $\tau \in [0, 1]$ .
- 3. Plot a PR curve for each class with  $\operatorname{Recall}_c(\tau)$  on the x-axis and  $\operatorname{Precision}_c(\tau)$  on the y-axis.

Each curve shows how well the model distinguishes class c from all other classes combined.

#### **Interpretation and Use of Multi-Class PR Curves:**

- **Per-Class Analysis**: A curve approaching (1, 1) suggests strong class-specific discrimination.
- AUC-PR per Class: One may compute the area under each class's PR curve, providing a single-value measure of how well the model isolates that class from the rest.
- Macro vs. Micro Averages: Macro-average treats each class equally, averaging Precision<sub>c</sub> and Recall<sub>c</sub> across classes. Micro-average aggregates across all classes, thus weighting classes by their frequencies.

**Advantages and Limitations:** Multi-class PR curves offer detailed, class-by-class insights—especially vital in imbalanced datasets or where certain classes are more critical. However, visualizing many classes can become cumbersome. Summary statistics (e.g., macro/micro averaged AUC-PR) or advanced visualization methods may be necessary when C is large.

### 3.2.13 Area Under the Receiver Operating Characteristic Curve (AUC-ROC)

The Receiver Operating Characteristic (ROC) curve plots the True Positive Rate (TPR) vs. the False Positive Rate (FPR) at various thresholds. Recall that:

$$TPR = \frac{TP}{TP + FN}, \qquad FPR = \frac{FP}{TN + FP}.$$
 (74)

# **Definition and Computation:**

- 1. For each threshold  $\tau$  in [0,1], classify an instance as positive if its score  $s(x) \geq \tau$ , otherwise negative.
- 2. Compute  $TPR(\tau)$  and  $FPR(\tau)$ .
- 3. Plot FPR on the x-axis vs. TPR on the y-axis. The resulting curve is the ROC curve.

### **Interpretation of the ROC Curve:**

- **Diagonal Line**: A purely random classifier. Points above the diagonal indicate better-than-random performance; points below indicate worse than random.
- Area Under the Curve (AUC): Provides a single-value summary of performance. An AUC of 1.0 indicates perfect separability; 0.5 indicates random guessing.
- **Top-Left Corner**: Ideally, a model aims for high TPR (capturing most positives) and low FPR (minimizing false positives).

**Limitations in Imbalanced Datasets:** When the positive class is rare, the ROC curve can be overly optimistic, since FPR can remain low simply because TN is large. In such cases, the Precision-Recall curve might be more informative, highlighting performance on the minority class more sensitively.

### 3.2.14 Multi-Class Receiver Operating Characteristic Curve (AUC-ROC)

Though ROC curves and AUC values were originally devised for binary classification, they can extend to multi-class tasks via multiple strategies.

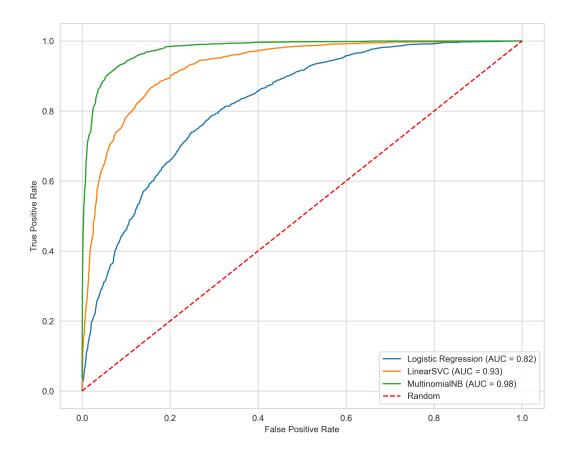


Figure 3: ROC curves for three models on the same dataset. The dashed line indicates random performance. Each model's AUC (area under the curve) is noted in the legend.

# One-vs-All (OvA) Approach:

- 1. For each class c, treat it as the positive class, merging all other classes as negative.
- 2. Construct a binary ROC curve for class c. Compute its AUC, AUC<sub>c</sub>.
- 3. Repeat for each class, yielding C ROC curves and C AUC scores.

One can then take the *macro-average* or *weighted-average* of these per-class AUC scores to obtain a single measure of multi-class performance.

One-vs-One (OvO) Approach: In problems with C classes, an OvO approach involves constructing a binary ROC curve for each pair of classes  $\{i,j\}$ , treating class i as positive and j as negative (and vice versa). This yields  $\frac{C(C-1)}{2}$  curves. A mean AUC across all pairs can then summarize the classifier's multi-class capability, although this can become computationally expensive for large C.

## Macro and Micro-Averaging in Multi-Class ROC:

- Macro-Averaged AUC: Calculates AUC for each OvA (or OvO) ROC curve and then averages them. This treats each class (or class pair) equally, regardless of frequency.
- Micro-Averaged AUC: Aggregates the confusion matrix counts over all classes into a single TPR and FPR curve, effectively weighting classes by their prevalence.

**Visualization and Interpretation:** Visualizing multi-class ROC curves can be done by overlaying each OvA curve on a single plot or employing a grid of subplots. The final choice depends on the number of classes and the intended level of detail. Metrics such as macro- and micro-averaged AUCs facilitate simpler quantitative comparisons between models.

In the preceding sections, we have covered a wide variety of loss functions and metrics for both regression and classification scenarios. Next, we delve into more specialized applications in computer vision, such as image classification (Section 4), object detection (Section 5), image segmentation (Section 6), face recognition (Section 7), monocular depth estimation (Section 8 and image generation (Section 9). These tasks introduce additional challenges and specialized metrics tailored to spatial, structural, and interpretability considerations.

# 4 Image Classification

Image classification involves categorizing an image as a whole into a specific label [70].

Image classification was the first task where deep learning demonstrated significant success, marking a critical moment in the advancement of artificial intelligence. This breakthrough occurred primarily with the development and application of Convolutional Neural Networks (CNNs). In 2012, the AlexNet architecture [71], presented by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, achieved state-of-the-art performance substantially better than the previous best results. This achievement established the potential of deep learning models to outperform traditional machine learning methods that relied heavily on hand-engineered features.

AlexNet's success led to a surge in research and development in deep learning. This resulted in significant improvements in deep learning algorithms and their applications in various domains. Image classification played a key role in this revolution, enabling advances that expanded the scope of deep learning into numerous other applications. This breakthrough has laid a strong foundation for further innovations in the field of AI.

Image classification is used in a diverse range of sectors, demonstrating its broad applicability. In medical imaging, it is used to improve diagnostic accuracy by automatically detecting and classifying abnormalities such as tumors, fractures, and pathological lesions in various types of scans, including X-rays, MRIs, and CT images [72, 73, 74, 75]. In agriculture, it helps to monitor crop health, predict yields, and detect plant diseases through images captured by drones or satellites [76, 77, 78, 79]. The retail industry uses this technology for automatic product categorization, inventory management, and online shopping enhanced with visual search capabilities [80, 81, 82, 83]. Security and surveillance systems use image classification to analyze footage for unauthorized activities, manage crowds, and identify individuals or objects of interest [84, 85]. In environmental monitoring, it is instrumental in tracking deforestation, monitoring changes in the natural environment, and assessing water quality using satellite imagery [85, 86, 87, 88, 89]. In addition, in the manufacturing sector, image classification automates quality control processes by identifying defects and verifying assembly, ensuring that products meet stringent standards with minimal human intervention [90, 91, 92, 93, 94].

#### 4.1 Image Classification Loss Functions

The core loss functions used for image classification are the same as those described in Section 3.1, such as Binary Cross-Entropy, Categorical Cross-Entropy, Weighted Cross-Entropy, Focal Loss, and Hinge Loss (see Table 6). Despite sharing a common foundation with standard classification tasks, image classification often presents additional considerations:

- Large-Scale Datasets: Datasets like ImageNet [95], with millions of labeled images across hundreds or thousands of classes, frequently employ the same cross-entropy-based losses. However, they may require more sophisticated regularization techniques (e.g., label smoothing [62] or focal loss [60]) to handle a large number of classes and to mitigate overfitting.
- Class Imbalance in Visual Data: Many image datasets, especially in medical imaging (e.g., rare pathology
  detection) or specialized industrial inspection, can exhibit severe class imbalance. Weighted Cross-Entropy
  or Focal Loss becomes essential for emphasizing minority classes, ensuring that rare but crucial categories
  receive sufficient attention during training.
- **Architectural Considerations:** Deep convolutional neural networks (CNNs) [71] or Vision Transformers [96] typically serve as the backbone for image classification. Although these architectures affect the feature extraction process, the fundamental loss computation remains unchanged.
- Label Smoothing in Vision Tasks: Label Smoothing (Section 3.1.5) is particularly common in large-scale vision classification. By replacing one-hot targets with a softly distributed label vector, models avoid becoming overconfident and can better generalize.

• Multi-Label Image Classification: In some applications, a single image can contain multiple labels (e.g., an image with both "cat" and "outdoor"). Binary cross-entropy losses are often used in multi-label classification setups, with each label treated as an independent positive/negative prediction.

Overall, while the mathematical definitions of these loss functions remain consistent across various domains, image classification tasks can require fine-tuning hyperparameters or incorporating additional techniques (like data augmentation or specialized regularizers) to optimize performance in large-scale or highly imbalanced image data.

## 4.2 Image Classification Metrics

Likewise, the performance metrics for image classification are the same as those described in Section 3.2 (see Table 7). Nonetheless, several image-specific factors and additional measures can be pertinent:

- **Top-k Accuracy:** A frequently reported metric in large-scale image classification (e.g., ImageNet benchmarks) is *Top-5 Accuracy*, where a prediction is considered correct if the true class is among the top-5 highest-probability predictions. This metric better reflects performance in settings with many similar classes.
- **Per-Class Accuracy and Balanced Metrics:** In highly imbalanced datasets, per-class accuracy or Balanced Accuracy (§3.2.3) can offer a more equitable view of model performance across rare and frequent categories. This is crucial in medical imaging, where detecting a rare disease can be more critical than accurately labeling a common condition.
- Precision-Recall Curves for Rare Classes: For multi-class or multi-label vision tasks where a particular class is rare yet high-impact (e.g., detection of a rare defect in industrial inspection), analyzing Precision-Recall curves (§3.2.11 and §3.2.12) can give deeper insight into model performance, especially when focusing on minority classes.
- Confusion Matrices for Visual Inspection: In multi-class image classification, confusion matrices (§3.2.1 and §3.2.2) are often visualized as heatmaps, allowing quick identification of classes that the model confuses (e.g., visually similar species of animals).
- Robustness and Interpretability: Beyond the standard metrics, recent research in image classification sometimes involves robustness evaluations (e.g., under adversarial attacks [97] or noise [98]) and interpretability metrics (e.g., saliency map consistency [99]). While these are not strictly performance metrics in the conventional sense, they reflect additional real-world considerations.
- Multi-Label Precision, Recall, and F1: In tasks where images can have multiple labels simultaneously (e.g., a single image could be "dog," "outdoor," and "playful"), metrics like Micro-/Macro-Precision, Recall, and F1 (§3.2.4, §3.2.5, and §3.2.6) need to be computed across all relevant labels. This ensures a complete view of performance when each image can belong to multiple categories.

While the core classification metrics (Accuracy, Precision, Recall, F1-score, ROC-AUC, Precision-Recall AUC, etc.) remain fundamentally the same, image classification scenarios often require additional approaches (e.g., top-k accuracy, multi-label evaluation) and visual methods (e.g., confusion matrices as heatmaps) to handle the unique challenges posed by large-scale or imbalanced image data.

## 5 Object Detection

Object detection in deep learning is a computer vision technique that involves localizing and recognizing objects in images or videos. It is common in various applications such as autonomous driving [100, 101, 102, 103], surveillance [104, 105, 106], human-computer interaction [107, 108, 109, 110], and robotics [111, 112, 113, 114]. Object detection involves identifying the presence of an object, determining its location in an image, and recognizing the object's class.

## 5.1 Object Detection Loss Functions

Object detection requires both localization—the accurate regression of bounding box coordinates around objects of interest—and classification—the correct assignment of class labels. Consequently, models typically employ composite loss functions comprising a classification component (e.g., cross-entropy-based) and a regression component (e.g., L1/L2 variants). By jointly penalizing misclassifications and bounding-box inaccuracies, these loss functions guide the training process toward accurate and robust detection across a variety of data distributions.

Table 11 provides an overview of commonly used loss functions in object detection, highlighting their applications, advantages, and limitations across different tasks and datasets.

Formally, for an image I containing N ground-truth objects, a typical detection model outputs:

$$\left\{ \left(\hat{c}_{i}, \hat{\mathbf{b}}_{i}\right) \right\}_{i=1}^{M},\tag{75}$$

where  $\hat{c}_i$  denotes predicted class probabilities (or logits) for the *i*-th detected region,  $\hat{\mathbf{b}}_i \in \mathbb{R}^4$  are the predicted bounding-box coordinates (e.g.,  $\{x_{\min}, y_{\min}, x_{\max}, y_{\max}\}$  or center/width/height parameterization), and M is the number of proposals or anchors. The overall loss  $\mathcal{L}$  is often a sum:

$$\mathcal{L} = \mathcal{L}_{\text{class}}(\{c_i\}, \{\hat{c}_i\}) + \lambda \mathcal{L}_{\text{reg}}(\{\mathbf{b}_i\}, \{\hat{\mathbf{b}}_i\}), \tag{76}$$

where  $\lambda$  is a weighting factor, and  $\{c_i, \mathbf{b}_i\}$  denote ground-truth class and bounding-box parameters. Below, we focus on the main bounding-box regression losses  $\mathcal{L}_{reg}$ .

Table 11: Guidelines for selecting an object detection loss function based on usage, data characteristics, advantages, and limitations.

Function	Usage	Data Characteristics	Advantages	Limitations
Smooth L1 Loss	Bounding box regression (e.g., RPN)	Handles moderate outliers; requires threshold $\beta$	Robust to outliers; less sensitive than MSE	Requires $\beta$ tuning; can degrade to L1 or near-0 loss
Balanced L1 Loss	Enhanced L1 for varying error magnitudes	Helpful for large or small bounding-box errors	Dynamic penalty; improves convergence, outlier handling	Extra hyperparameters; slightly higher complexity
IoU Loss	Directly optimizes overlap in object detection	Suitable when bounding-box overlap is primary metric	Aligns with common evaluation metric; intuitive	Zero gradient if no overlap; limited partial-overlap sensi- tivity
GIoU Loss	Extension of IoU for non/partially overlap- ping boxes	Useful when frequent non-overlaps occur	Overcomes IoU's zero-gradient issue; better feedback	Does not handle centroid distance or aspect ratio
DIoU, CIoU	Improved bounding- box regression (cen- troid, aspect ratio)	Applicable to precise alignment needs	DIoU handles spatial misalignment; CIoU adds aspect-ratio term	More parameters ( $\alpha$ , etc.); higher tuning complexity
Focal Loss	Addresses severe class imbalance	Effective for rare- class detection in large datasets	Down-weights easy negatives; focuses on hard examples	Extra parameters $(\gamma, \alpha)$ ; can be tricky to tune
YOLO Loss	Composite loss for YOLO detectors (lo- calization + classifica- tion + objectness)	Designed for grid- based real-time detection	Single-pass optimization; efficient in practice	Requires balancing components; may struggle with small objects
Wing Loss	Robust regression for facial landmarks and similar tasks	Suitable for data with heavy outliers	Less sensitive to outliers; emphasizes medium errors	Requires careful tuning of $\omega$ and $\epsilon$ ; instability risk

#### 5.1.1 Smooth L1 Loss

Smooth L1 (sometimes called L1-Smooth or Huber-like) is a robust loss function commonly used in bounding-box regression. Originally introduced in the Fast R-CNN framework [115], Smooth L1 mitigates outliers more effectively than Mean Squared Error (MSE), while retaining differentiability at zero. For a single coordinate, it is defined as:

$$L_{\text{smoothL1}}(y, \hat{y}) = \begin{cases} \frac{1}{2} (y - \hat{y})^2, & \text{if } |y - \hat{y}| < \beta, \\ |y - \hat{y}| - \frac{1}{2} \beta, & \text{otherwise,} \end{cases}$$

$$(77)$$

where y is the true bounding-box coordinate (e.g., an offset or width/height),  $\hat{y}$  is the predicted value, and  $\beta > 0$  is a threshold that controls the transition between the L2-like (quadratic) region and the L1-like (linear) region.

**Relation to Huber Loss:** Smooth L1 is closely related to Huber Loss (Section 2.1.3). In fact, one can consider Huber Loss scaled by  $1/\beta$ , with subtle differences in slope and transitions [116]:

- As  $\beta \to 0$ , Smooth L1  $\to$  absolute L1 loss.
- As  $\beta \to +\infty$ , Smooth L1  $\to$  constant zero in the wide region, while Huber transitions to MSE behavior.
- **Slope Differences:** In Smooth L1, the slope in the L1 region is always 1, whereas in Huber Loss, the slope in the linear region is *β*.

**Usage:** Smooth L1 is employed extensively in two-stage object detectors (e.g., Fast R-CNN, Faster R-CNN [117]). The Region Proposal Network (RPN) uses Smooth L1 for refining anchor boxes into region proposals, and the second stage uses it again for final bounding-box adjustments. Empirically, it offers stable gradients and is less sensitive to large bounding-box errors than MSE.

#### **Practical Considerations:**

- Anchor/Proposal Mappings: Bounding-box regression often predicts offsets relative to anchor or proposal boxes, making the typical (x, y, w, h) transformation more robust when combined with Smooth L1.
- **Hyperparameter**  $\beta$ : Typical values range from 0.5 to 1.0. Tuning  $\beta$  can yield small but noticeable improvements in detection accuracy for particular datasets.
- **Gradient Behavior:** The piecewise structure ensures that minor deviations are penalized quadratically (promoting stable convergence), whereas large deviations shift to a linear penalty, reducing the effect of outliers.

## 5.1.2 Balanced L1 Loss

Balanced L1 [118] is an enhancement over standard L1 for bounding-box regression, designed to improve robustness when object sizes vary widely or when large vs. small errors must be balanced differently. It adaptively modulates the penalty based on the magnitude of the error  $x = y - \hat{y}$ , applying a logarithmic form for large errors and a quadratic-like form for small errors. The loss is defined as:

$$L_{\text{balanced}}(x) = \begin{cases} \beta \ln\left(\frac{|x|}{\beta} + 1\right), & \text{if } |x| \ge \beta, \\ \frac{x^2 + \beta^2}{2\beta}, & \text{otherwise,} \end{cases}$$
(78)

where x is the bounding-box error and  $\beta$  is a threshold controlling the transition between these regions.

#### **Properties:**

- Logarithmic Penalty for Large Errors: When  $|x| \gg \beta$ , the penalty grows more slowly (logarithmically), mitigating gradient explosions that can occur with pure L1 in the presence of outliers or misaligned anchors.
- Quadratic Penalty for Small Errors: Near  $|x| < \beta$ , the squared term  $\frac{x^2}{2\beta}$  fosters fine-grained accuracy, akin to a smoothed L1 approach.
- Stabilizing Training: By balancing the response to large vs. small errors, the Balanced L1 Loss can yield more stable gradient updates and faster convergence.

# **Benefits for Object Detection:**

- Varying Object Scales: Datasets like MS COCO or Pascal VOC contain objects of vastly different sizes. Balanced L1 helps ensure large errors (e.g., on very big bounding boxes) do not overly dominate training while still providing precise adjustment for smaller bounding boxes.
- Outlier Resistance: Sudden large bounding-box offsets, often arising in early training stages or challenging examples, are handled gracefully via the log function, preventing gradient spikes.
- Integration with Classification Loss: As with Smooth L1, Balanced L1 is commonly added to a classification loss (e.g., Focal Loss or cross-entropy) for the overall detection loss (76).

**Hyperparameter Tuning:**  $\beta$  generally lies in a range (0.5 to 2), and in some implementations, additional coefficients modulate the log or quadratic segments. Empirical tuning or cross-validation can improve performance on specific datasets (indoor vs. outdoor images, small objects vs. large objects, etc.).

When to Use Balanced L1: While standard Smooth L1 suffices for many detection tasks, Balanced L1 can be advantageous when:

- The dataset exhibits a broad distribution of object scales or unusual bounding-box aspect ratios.
- One observes slow convergence or instability due to extreme bounding-box outliers.
- Precisely localizing small objects while also handling large ones is critical.

Balanced L1 loss offers a flexible, adaptive mechanism for bounding-box regression, addressing limitations of purely linear or purely quadratic losses in heterogeneous detection scenarios.

## 5.1.3 Intersection Over Union (IoU) Loss

Intersection over Union (IoU) measures the overlap between two bounding boxes, typically denoted as the predicted box  $B_p$  and the ground-truth box  $B_t$ . Mathematically, the IoU is defined as:

$$IoU = \frac{Area(B_p \cap B_t)}{Area(B_p \cup B_t)},$$
(79)

where  $Area(\cdot)$  calculates the 2D area of a bounding box, and  $B_p \cap B_t$  denotes their intersection region. The corresponding IoULoss is then:

$$L_{\text{IoU}}(B_p, B_t) = 1 - \text{IoU}(B_p, B_t).$$
 (80)

**Usage in Object Detection:** One-stage detectors (e.g., SSD [119] and YOLO [120]) commonly include IoU Loss as a localization term in a multi-task objective alongside a classification loss. Minimizing 1 - IoU encourages a high overlap between  $B_p$  and  $B_t$ , thus improving bounding-box alignment. Refer to Figure 4.

#### **Limitations:**

- Zero Gradient for Non-Overlapping Boxes: If  $B_p \cap B_t = \emptyset$ , the intersection area is zero, yielding IoU = 0. This offers no gradient signal to adjust  $B_p$ , making optimization difficult when boxes do not overlap initially.
- Sensitivity to Partial Overlaps: IoU may change slowly when boxes partially overlap but have varying aspect ratios or positions, limiting its ability to distinguish subtly different bounding-box configurations.

These issues motivate generalizations such as GIoU, DIoU, and CIoU, which enhance gradient signals for bounding-box regression under non-overlapping or partially overlapping conditions.

# 5.1.4 GIoU Loss

*Generalized IoU (GIoU) Loss* [121] extends IoU Loss by accounting for cases where bounding boxes do not overlap or only partially overlap. It incorporates the area outside both bounding boxes but within their smallest enclosing box. Formally:

$$L_{\text{GIoU}}(B_p, B_t) = 1 - \text{IoU}(B_p, B_t) + \frac{\text{Area}(\mathcal{C} \setminus (B_p \cup B_t))}{\text{Area}(\mathcal{C})}, \tag{81}$$

where C is the smallest bounding box that encloses both  $B_p$  and  $B_t$ . Compared to the standard IoU loss, GIoU offers:

- Non-Overlap Sensitivity: By including the region  $C \setminus (B_p \cup B_t)$  in the loss, GIoU penalizes disjoint boxes and guides them to move closer, even if they do not initially intersect.
- More Stable Gradients: GIoU provides a continuous gradient signal through all stages of bounding-box refinement, improving convergence.

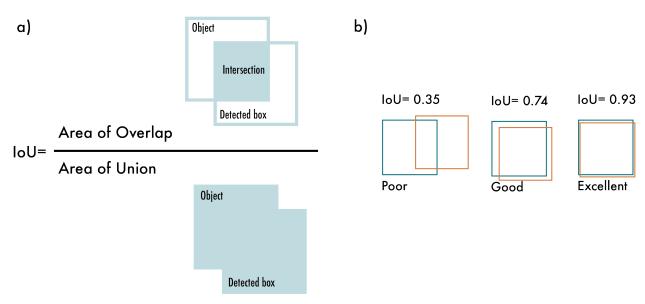


Figure 4: Intersection Over Union (IoU). (a) The IoU is computed as the ratio of the intersection area of two boxes over their union area. (b) Examples of three different IoU values for varied box positions.

### **Limitations of GIoU:**

- 1. **Spatial Misalignment:** GIoU does not directly address centroid distance between boxes, which can still result in suboptimal positioning even if GIoU is high.
- 2. **Aspect Ratio Differences:** GIoU does not incorporate width-height ratio mismatches, potentially leading to inaccurate regressions when object shapes vary significantly.

### 5.1.5 DIoU and CIoU Losses

Distance-IoU (DIoU) and Complete-IoU (CIoU) [122] are further refinements of IoU-based losses, introduced to handle spatial misalignment and aspect ratio differences more explicitly.

#### **DIoU Loss:**

$$L_{\text{DIoU}}(B_p, B_t) = 1 - \text{IoU}(B_p, B_t) + \frac{d^2(B_p, B_t)}{c^2}, \tag{82}$$

where  $d(B_p, B_t)$  is the Euclidean distance between the centroids of the two boxes, and c is the diagonal length of their smallest enclosing box. Hence, DIoU penalizes large centroid offsets even if the boxes overlap reasonably well.

#### **CIoU Loss:**

$$L_{\text{CIoU}}(B_p, B_t) = L_{\text{DIoU}}(B_p, B_t) + \alpha \cdot v, \tag{83}$$

where v measures the consistency in aspect ratios:

$$v = \frac{4}{\pi^2} \left( \arctan\left(\frac{w_t}{h_t}\right) - \arctan\left(\frac{w_p}{h_p}\right) \right)^2, \tag{84}$$

and  $\alpha$  is a weighting factor defined as:

$$\alpha = \frac{v}{(1 - \text{IoU}) + v}.\tag{85}$$

Here,  $(w_t, h_t)$  and  $(w_p, h_p)$  are the ground-truth and predicted bounding-box widths and heights, respectively. By combining DIoU with an aspect-ratio alignment term, CIoU yields more comprehensive bounding-box alignment.

### Advantages of DIoU and CIoU:

- Centroid Alignment: DIoU addresses scenarios where boxes overlap in area but remain spatially misaligned, improving positioning accuracy.
- **Aspect Ratio Correction:** CIoU adjusts for mismatches in width-height ratio, reducing the chance that two boxes have significant overlap but very different shapes.
- Stable, Continuous Gradients: Both losses provide a gradient signal even when boxes do not fully overlap, aiding the early stages of training.

## **Practical Considerations:**

- Computational Complexity: Calculating DIoU and CIoU involves centroid distance and aspect ratio terms but adds minimal overhead compared to simpler IoU computations.
- **Hyperparameters:** The weighting factor  $\alpha$  in CIoU is dynamically computed (85); thus, no extra hyperparameter is usually needed beyond standard bounding-box regression settings.
- **Integration in Detection Pipelines:** Both DIoU and CIoU replace the standard bounding-box loss term in multi-task objectives. They can be seamlessly plugged into frameworks such as YOLO or Faster R-CNN to enhance localization quality.

By addressing both centroid offset and aspect-ratio mismatch, DIoU and CIoU further refine IoU-based losses, leading to improved localization accuracy and more robust bounding-box regression in modern object detection pipelines.

#### 5.1.6 Focal Loss

Focal Loss, introduced by Lin et al. [60], is an enhancement of the standard cross-entropy loss aimed at tackling severe class imbalance. In many detection tasks, background (negative) samples vastly outnumber foreground (positive) samples, causing traditional loss functions to bias the model toward the majority class. Focal Loss modifies cross-entropy by down-weighting well-classified examples, thus shifting focus onto "hard," misclassified samples.

**Definition:** Let  $p \in [0,1]$  be the predicted probability for the ground-truth class  $y \in \{0,1\}$ . Define

$$p_t = \begin{cases} p, & \text{if } y = 1, \\ 1 - p, & \text{if } y = 0. \end{cases}$$
 (86)

The Focal Loss is then:

$$FL(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log(p_t), \tag{87}$$

where  $\alpha_t \in [0, 1]$  is a class-dependent weighting factor (often set inversely proportional to class frequency), and  $\gamma \geq 0$  is the *focusing parameter* that reduces the relative loss for well-classified examples.

## **Interpretation:**

- Focus on Hard Examples: When  $p_t$  is large (the example is predicted correctly with high confidence),  $(1-p_t)^{\gamma}\approx 0$ , thus diminishing the contribution of that "easy" example to the loss. Conversely, misclassified or uncertain examples ( $p_t$  small) incur larger losses, guiding the model to improve minority or hard-to-classify instances.
- Imbalanced Data: By setting  $\alpha_t$  based on class frequencies, Focal Loss helps the model pay sufficient attention to underrepresented classes.
- Typical Hyperparameters: Common values for  $\gamma$  range from 1 to 5. Larger  $\gamma$  emphasizes misclassified examples more strongly, but if too large, it may destabilize training.

**Applications:** Focal Loss has been extensively adopted in:

- One-stage Detectors: RetinaNet [60] uses Focal Loss to mitigate class imbalance in dense object detection.
- Medical Imaging: Enhances detection of rare pathologies by strongly penalizing misclassifications of minority disease classes.
- **Segmentation and Pose Estimation:** Similar imbalance scenarios (background vs. foreground pixels, etc.) can benefit from the focusing property.

#### 5.1.7 YOLO Loss

*YOLO Loss* [120] is a composite loss function specifically crafted for the YOLO family of one-stage object detectors. YOLO processes an image in a single forward pass, dividing the input into grid cells, each predicting bounding-box coordinates, an "objectness" score, and class probabilities.

The standard YOLO loss combines three terms:

#### 1. Localization Loss:

$$L_{\text{loc}} = \sum_{\text{pred hoxes}} \lambda_{\text{coord}} \left[ (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (w_i - \hat{w}_i)^2 + (h_i - \hat{h}_i)^2 \right], \tag{88}$$

where  $(x_i, y_i, w_i, h_i)$  and  $(\hat{x}_i, \hat{y}_i, \hat{w}_i, \hat{h}_i)$  are ground-truth and predicted box parameters, respectively, and  $\lambda_{\text{coord}}$  controls the relative importance of localization accuracy.

#### 2. Objectness (Confidence) Loss:

$$L_{\text{obj}} = \sum_{\text{pred boxes}} \left( C_i - \hat{C}_i \right)^2, \tag{89}$$

where  $C_i \in \{0, 1\}$  indicates whether an object is present, and  $\hat{C}_i$  is the predicted confidence score (sometimes interpreted as the IoU of the predicted box with any ground truth).

#### 3. Classification Loss:

$$L_{\text{class}} = \sum_{\text{pred boxes}} -\sum_{c=1}^{C} p_i(c) \log(\hat{p}_i(c)), \tag{90}$$

where  $p_i(c)$  is the one-hot vector of the true class and  $\hat{p}_i(c)$  is the predicted probability for class c. Typically, standard cross-entropy is used here.

#### **YOLO Loss:**

$$\mathcal{L}_{\text{YOLO}} = L_{\text{loc}} + L_{\text{obj}} + L_{\text{class}}.$$
 (91)

Adjusting the weighting factors in each term (e.g.,  $\lambda_{\rm coord}$ ) balances the trade-off among bounding-box precision, object confidence, and classification accuracy.

YOLO's grid-based approach relies heavily on these combined losses to perform localization and classification simultaneously. Early YOLO versions used sum-of-squares for bounding-box terms, which can be sensitive to object scale. Subsequent YOLO variants introduced modifications like IoU-based or GIoU-based losses for more robust box regression.

## 5.1.8 Wing Loss

Wing Loss [123] was proposed to address the limitations of pure  $L_1$  or  $L_2$  (MSE) losses in tasks sensitive to both small and large localization errors. In bounding-box regression (and related tasks such as facial landmark localization), mild errors should be penalized differently from outliers.

**Definition:** Let x be the prediction error (e.g., the difference between predicted and ground-truth coordinates), and let  $\omega$  and  $\epsilon$  be hyperparameters. Wing Loss is piecewise-defined:

WingLoss(x) = 
$$\begin{cases} \omega \ln\left(1 + \frac{|x|}{\epsilon}\right), & \text{if } |x| < \omega, \\ |x| - C, & \text{otherwise,} \end{cases}$$
(92)

where

$$C = \omega - \omega \ln\left(1 + \frac{\omega}{\epsilon}\right). \tag{93}$$

The constant C ensures continuity of the function at  $|x| = \omega$ .

## **Properties:**

- Logarithmic Region: For errors  $|x| < \omega$ , the penalty grows logarithmically with |x|, increasing sensitivity to small and moderate errors compared to L1's linear slope.
- Linear Region: For larger errors ( $|x| \ge \omega$ ), the loss reverts to a linear form |x| C, preventing extreme outliers from dominating gradient updates.
- Hyperparameters  $\omega$  and  $\epsilon$ :  $\omega$  sets the threshold where the function transitions from the logarithmic regime to the linear regime.  $\epsilon$  controls the steepness of the log curve. Large  $\epsilon$  increases sensitivity to moderate errors, but may require careful training to avoid instability.

#### Usage:

- Facial Landmarks: Wing Loss was originally proposed for face alignment tasks, ensuring small errors are penalized more precisely without overly punishing large outliers.
- Potential for Bounding-Box Regression: The same principle can be extended to object bounding-box localization, especially in tasks where small positional deviations can significantly impact performance (e.g., medical imaging).

Similar to Smooth L1 (Section 5.1.1), Wing Loss dampens large errors but handles small errors differently via a logarithmic term. Users may prefer Wing Loss when the distribution of regression errors is such that capturing "near-accurate" predictions is crucial, and absolute outliers must not dominate training.

## 5.2 Object Detection Metrics

In object detection, a model must correctly localize objects (bounding box regression) and classify them (object category). During evaluation, we typically rely on True Positives (TP), False Positives (FP), False Negatives (FN), and (in some definitions) True Negatives (TN) to quantify how well the detector performs. However, unlike in image-level classification, the notion of positive vs. negative predictions depends on spatial overlap between boxes, usually measured by the *Intersection over Union* (IoU) (see Figure 4 and Section 5.1.3).

**True Positive (TP):** A predicted bounding box is deemed a true positive if:

$$IoU(B_p, B_t) \geq \tau, \tag{94}$$

where  $B_p$  is the predicted box,  $B_t$  is the ground-truth box, and  $\tau \in [0, 1]$  is an IoU threshold. Common thresholds (0.25, 0.5, 0.75) reflect different stringency levels. For instance,  $\tau = 0.5$  is typical in many benchmarks, whereas higher thresholds (e.g., 0.75) demand more precise localization.

**False Positive (FP):** A predicted box is labeled as a false positive if it does not meet the IoU threshold for *any* ground-truth box (i.e., no match), or if it is deemed redundant after a match has already been assigned to another prediction. High FP counts degrade *Precision*.

**False Negative (FN):** A ground-truth box not matched by any prediction above threshold  $\tau$  is a false negative, indicating the model missed a true object in the image. High FN counts degrade *Recall*.

**True Negative (TN):** In many detection settings, TN is less frequently reported. Conceptually, TN indicates that the model correctly concludes no object is present in a negative region. However, since potential bounding-box search space is huge, TN is not always explicitly counted.

### **Common IoU Thresholds in Object Detection**

- IoU ≥ 0.5: Widely used in benchmarks like Pascal VOC [124], establishing a balance between moderate localization precision and practical acceptance.
- IoU ≥ 0.75: Used in stricter scenarios (e.g., autonomous driving) where precise localization is critical and false positives (poorly localized boxes) can have severe consequences.
- IoU ≥ 0.25: Occasionally used in tasks requiring high recall (e.g., medical imaging), ensuring that even loosely fitting bounding boxes can count as detections to avoid missing important findings.

Typical object detection metrics include:

- Average Precision (AP) and Mean Average Precision (mAP)
- Intersection over Union (IoU) (Section 5.1.3)
- Precision-Recall Curve (Section 3.2.11)

# 5.2.1 Average Precision (AP)

In multi-class object detection, a detector must correctly localize and classify objects across multiple categories. *Average Precision (AP)* quantifies per-category performance and then averages over categories (often termed *mean Average Precision* or mAP). By doing so, each class's performance is individually assessed, preventing categories with abundant instances from overshadowing others.

AP calculation typically incorporates an IoU threshold  $\tau$ , such that a predicted box is considered a correct detection if IoU  $\geq \tau$ . Datasets like COCO [125] examine multiple IoU thresholds ( $\tau = 0.5, 0.55, \ldots, 0.95$ ) to measure performance from lenient to strict localization criteria.

# **Pascal VOC AP Computation**

The Pascal VOC dataset [124] includes 20 object categories. The classical procedure for computing AP in VOC is:

- 1. Compute IoU: For each detection, compute IoU with ground-truth objects. A detection is matched to the ground-truth box with the highest IoU above  $\tau$ .
- 2. Precision and Recall Curve: For each category, sort predictions by confidence score (descending). Moving through these detections from highest to lowest confidence defines a sequence of precision-recall pairs  $(\text{Rec}(\theta), \text{Prec}(\theta))$  as the decision threshold  $\theta$  changes.
- 3. 11-Point Interpolation: VOC uses recall levels  $\{0, 0.1, 0.2, \dots, 1.0\}$ . For each recall level r, the interpolated precision is the maximum precision for any recall  $\geq r$ . This yields a piecewise constant precision-recall curve.
- 4. *Area Under Curve:* The AP is the area under this interpolated curve, computed as a sum over the discrete recall bins:

$$AP = \sum_{r \in \{0, 0.1, \dots, 1.0\}} \frac{\text{Prec}_{\text{interp}}(r)}{11}.$$
 (95)

## **Microsoft COCO AP Computation**

The COCO dataset [125] spans 80 categories and employs a more extensive AP definition:

- 1. *Intersection over Union*: As before, compute the IoU between each detection and ground truth. Match each detection to at most one ground truth if  $IoU > \tau$ .
- 2. *Precision-Recall Pairs:* For a wide range of model confidence thresholds, measure the proportion of TPs among all predicted positives (Precision) and the fraction of GT objects detected (Recall).
- 3. 101-Point Interpolation: COCO samples 101 recall levels  $\{0, 0.01, 0.02, \dots, 1.0\}$ . For each recall level r, compute the maximum precision for recall  $\geq r$ .
- 4. Compute AP for Each IoU: Unlike VOC, COCO averages over multiple IoU thresholds: 0.50, 0.55, ..., 0.95.
- 5. *Mean AP over Categories:* Average the per-category AP to obtain mAP. Optionally, separate calculations by object size (small, medium, large).

Common COCO metrics include  $AP_{50}$  (AP at IoU = 0.5) and  $AP_{50:95}$  (average AP across  $IoU \in \{0.50, \dots, 0.95\}$ ), summarizing performance across varying localization strictness. Table 12 lists all common COCO evaluation metrics.

## 5.2.2 Average Recall (AR)

Average Recall (AR) summarizes how well the model detects all objects over multiple IoU thresholds and possible per-image detection limits. It complements AP by focusing on the *recall* dimension—i.e., the proportion of ground-truth objects found—averaged across different scenarios.

#### **General Steps to Compute AR:**

- 1. **Compute IoU:** For each detection, compute IoU with each ground-truth box in the same image.
- 2. **Match Detections and Ground Truths:** A ground truth is *matched* if at least one detection has  $IoU \ge \tau$ . Each ground truth can only be matched once.
- 3. Measure Recall: For a given IoU threshold, the recall is  $\frac{TP}{GT}$ , where GT is the number of ground-truth boxes.
- 4. Average Over IoU Thresholds: Vary  $\tau$  in a set (e.g.,  $\{0.5, 0.55, \dots, 0.95\}$ ) and average the resulting recall values.
- 5. **Average Over Max Detections:** Models often limit the maximum number of detections (e.g., 100) per image. Repeat the recall calculation with different detection limits (e.g., 1, 10, 100) and average.
- 6. Aggregate Over the Dataset: Finally, average across all images in the dataset to obtain a single AR value.

**AR in COCO:** COCO commonly reports AR at different detection cutoffs (AR@1, AR@10, AR@100) and object sizes (small, medium, large). This helps assess the trade-off between detection quantity and detection quality, revealing how the detector performs when restricted to fewer or more bounding-box predictions.

In general, these object detection metrics: AP (including variations like mAP, AP<sub>50</sub>, AP<sub>50:95</sub>) and AR, together with IoU-based definitions of TP, FP, and FN, provide a robust framework to evaluate the precision and robustness of detection models in tasks ranging from generic object localization to specialized domains (e.g. autonomous driving or medical imaging).

Table 12: This table summarizes key metrics from the COCO dataset for evaluating object detection models, including Average Precision (AP) and Average Recall (AR) across various IoU thresholds and by object size. AP at different IoU thresholds measures model accuracy with varying overlap levels, while AP and AR across scales evaluate performance on different object sizes.

Average Precision (AP)	
$\begin{array}{c} \text{AP} \\ AP^{IoU=.50} \end{array}$	% AP at IoU=.50:.05:.95 (primary challenge metric) % AP at IoU=.50 (PASCAL VOC metric)
$AP^{IoU=.75}$	% AP at IoU=.75 (strict metric)
AP Across Scales:	
$AP^{small}$	$\%$ AP for small objects: area $< 32^2$
$AP^{medium}$	% AP for medium objects: $32^2 < \text{area} < 96^2$
$AP^{large}$	% AP for large objects: area $> 96^2$
Average Recall (AR):	
$AR^{max=1}$	% AR given 1 detection per image
$AR^{max=10}$	% AR given 10 detections per image
$AR^{IoU=100}$	% AR given 100 detection per image
AR Across Scales:	
$AR^{small}$	$\%$ AR for small objects: area $< 32^2$
$AR^{medium}$	% AR for medium objects: $32^2 < \text{area} < 96^2$
$AR^{large}$	$\%$ AR for large objects: area $> 96^2$

# 6 Image Segmentation

*Image segmentation* is the process of assigning a label or category to each pixel in an image, effectively partitioning the scene into meaningful regions. Unlike image-level classification or object detection, segmentation operates at the pixel level, requiring a detailed understanding of both local features and global context. Deep segmentation networks typically learn to predict a pixel's class label by considering its surrounding region or the entire image context.

Common segmentation approaches are usually grouped into:

- Semantic Segmentation: Focuses on partitioning the image into "stuff" categories (e.g., road, sky, grass), where all instances of the same class share one label [126, 127, 128, 129, 130, 131, 132, 133].
- Instance Segmentation: Aims to detect and segment each occurrence of a "thing" class (e.g., people, cars) individually [134, 135, 136, 137, 138]. Different instances of the same class receive distinct masks.
- **Panoptic Segmentation:** Unifies semantic and instance segmentation [139, 140, 141, 142, 143], assigning each pixel either a "stuff" label or a unique "thing" identifier.

Segmentation has broad applications, including but not limited to:

- Scene Understanding: In robotics or autonomous driving, detailed environmental parsing enhances navigation and decision-making [144, 145, 146, 147, 148].
- *Medical Imaging:* Accurate pixel-level delineation of anatomy or pathology is critical for diagnosis and treatment planning [149, 150, 151, 152, 153, 154, 155].
- Robotic Perception: Pixel-wise classification facilitates grasping and manipulation by identifying object boundaries [156, 157, 158, 159, 160].
- Autonomous Vehicles: Road-lane detection, pedestrian segmentation, and obstacle identification demand precise pixel-level predictions [161, 162, 163, 164, 165, 166].
- *Video Surveillance:* Segmentation aids in crowd analysis, anomaly detection, and activity recognition [167, 168, 169, 170, 171].
- Augmented Reality: Seamlessly overlays virtual elements by segmenting foreground objects or the user's environment [172, 173, 174, 175, 176].

#### **6.1 Segmentation Loss Functions**

Numerous loss functions have been devised to handle the unique challenges of segmentation, including dealing with heavy class imbalance and enforcing coherent shapes in the output masks. Table 13 summarizes the most common segmentation loss functions, along with their typical usage scenarios, data properties for which they work best, and practical advantages/limitations. Although many segmentation tasks can be approached using standard classification losses (e.g., pixel-wise cross-entropy), specialized losses often incorporate region-based considerations or adapt to heavily imbalanced pixel distributions.

The sections that follow present deeper explanations for each loss function and how they apply to different segmentation contexts. Depending on the nature of the application—medical imaging, large-scale street-scene parsing, or instance-level segmentation tasks—practitioners might favor certain losses for their robustness to imbalance or their direct optimization of region overlap metrics.

## 6.1.1 Cross-Entropy Loss for Segmentation

Cross-Entropy (CE) loss is a widely used measure in segmentation tasks to quantify the divergence between the model's predicted class probabilities and the true (one-hot or binary) labels on a per-pixel basis. In multi-class segmentation, one assumes each pixel i belongs to exactly one of C classes. Let  $p_{i,c} \in [0,1]$  be the predicted probability that pixel i is class c, and let  $y_{i,c} \in \{0,1\}$  be a one-hot indicator denoting the actual class label for pixel i. Then the CE loss is

$$L_{\text{CE}} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \log(p_{i,c}), \tag{96}$$

where N is the total number of pixels in the image. Minimizing  $L_{\rm CE}$  encourages  $p_{i,c}$  to approach 1 for the correct class c, and to remain near 0 for all other classes.

**Binary Segmentation Case:** For *binary* foreground/background segmentation, the cross-entropy loss simplifies to the *Binary Cross-Entropy* (BCE) form:

$$L_{\text{BCE}} = -\frac{1}{N} \sum_{i=1}^{N} \left[ y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \right], \tag{97}$$

Function	Usage	<b>Data Characteristics</b>	Advantages	Limitations
Cross- Entropy	Multi-class or binary segmentation Pixel-wise probability comparison	Balanced or large- scale datasets Works in general segmentation tasks	Straightforward, sta- ble Well-established	Sensitive to imbalance May require weight- ing or complementary losses
Dice Loss	Overlap-based seg- mentation Common in medical imaging	Highly imbalanced data Different class sizes	Directly optimizes overlap Robust to imbalance	Instability if overlap is very low May need combina- tion with CE
IoU (Jac- card) Loss	Maximizing Intersection over Union Used in semantic segmentation	Data where precision and recall both matter Multiple object classes	Correlates directly with IoU metric Penalizes FPs/FNs	Implementation complexity May need surrogate for better gradients
Tversky Loss	Extension of Dice Adjusts penalties for FPs/FNs	Severe class imbal- ance Medical/rare-class data	Hyperparameters $(\alpha, \beta)$ for fine control Focus on crucial errors	Requires careful tun- ing Risk of bias toward certain errors
Lovász Loss	Direct IoU optimiza- tion Smooth approxima- tion	Imbalanced data IoU as primary metric	Aligns well with final IoU Differentiable surrogate	More complex implementation Fewer available examples
Focal Loss	Class-imbalanced seg- mentation	Highly skewed data Medical, aerial im-	Down-weights easy cases	Hyperparameter sensitivity

Table 13: Guidelines for selecting image segmentation loss functions based on usage, data characteristics, advantages, and limitations.

where  $y_i \in \{0,1\}$  indicates whether pixel i belongs to the object of interest (1) or the background (0), and  $p_i$  is the predicted probability of pixel i being foreground. As with multi-class CE, minimizing BCE aligns predicted probabilities with the true binary labels at each pixel.

#### **Discussion and Advantages:**

Emphasizes hard ex- agery

amples

• **Pixel-Wise Classification**: CE-based losses treat each pixel classification independently, making them straightforward and efficient for large-scale segmentation tasks.

 $\alpha, \gamma$  fine-tuning

Over-focus on hard

samples

- **Probabilistic Interpretation**: By modeling segmentation as per-pixel Bernoulli or multinomial distributions, one can leverage well-established gradient-based optimization and network architectures.
- Limitations with Imbalance: In cases where the foreground is much smaller than the background (e.g., medical imaging), standard CE may underweight crucial classes. Solutions include weighting schemes, Focal Loss (§5.1.6), or region-based losses (e.g., Dice Loss).

#### 6.1.2 Dice Loss

Dice Loss, derived from the Dice Similarity Coefficient (DSC) or Sørensen–Dice index [177], offers an overlap-based metric for segmentation. In tasks where the foreground class is relatively rare—such as in lesion detection or other small-object scenarios—Dice-based losses can outperform pixel-wise classification losses by directly optimizing the spatial overlap between predicted and ground-truth regions.

**Definition:** Consider  $\hat{y}_i$  as the set of pixels in the predicted segmentation mask and  $y_i$  as the set of pixels in the ground-truth mask. The DSC measures the overlap:

$$DSC(\hat{y}_i, y_i) = \frac{2|\hat{y}_i \cap y_i|}{|\hat{y}_i| + |y_i|}.$$
(98)

The Dice Loss is then formulated as

$$L_{\text{Dice}} = 1 - \text{DSC}(\hat{y}_i, y_i), \tag{99}$$

which yields values from 0 (no overlap) to 1 (perfect overlap) in the coefficient domain, and from 1 to 0 in the loss domain.

**Pixel-Level Implementation:** In practice,  $\hat{y}_i$  and  $y_i$  can be represented as continuous maps (e.g., predicted probabilities and binary ground-truth labels). Summations or integrals over these maps approximate the set cardinalities and intersections. This approach is particularly effective for imbalanced datasets because it amplifies small-object region overlap and de-emphasizes large background areas.

#### **Advantages and Considerations:**

- **Imbalance Robustness**: Dice Loss intrinsically focuses on matching the pixels in the smaller target region, improving sensitivity to rare classes or fine structures.
- **Direct Overlap Optimization**: Unlike pixel-wise classification losses, Dice Loss directly penalizes inadequate region overlap, often leading to better boundary adherence and shape consistency.
- **Hyperparameter-Free**: The classic Dice formulation does not introduce extra hyperparameters, though variants like Tversky Loss (§6.1.4) do provide weighting mechanisms for FN vs. FP.

In general, dice loss is a natural choice in segmentation tasks, particularly in medical imaging, where overlapping predicted and ground-truth masks are more relevant than merely classifying each pixel independently.

# 6.1.3 Intersection Over Union (IoU) Loss for Segmentation

Intersection over Union (IoU), also called the Jaccard Index, measures the degree of overlap between the predicted segmentation mask  $\hat{y}_i$  and the ground-truth mask  $y_i$  at the pixel level. Formally, the IoU for each pixel i reflects whether that pixel belongs to both masks (intersection) or to at least one of the two masks (union). Let  $|y_i \cap \hat{y}_i|$  and  $|y_i \cup \hat{y}_i|$  respectively denote the pixel-level intersection and union; the IoU-based loss is often defined as:

$$L_{\text{IoU}} = 1 - \frac{1}{N} \sum_{i=1}^{N} \frac{|y_i \cap \hat{y}_i|}{|y_i \cup \hat{y}_i|},$$
(100)

where N is the number of pixels (or potentially superpixels/regions) in an image,  $y_i$  indicates the ground-truth label of pixel i, and  $\hat{y}_i$  is the predicted label. In practice, these sets are often implemented as continuous probability maps (for  $\hat{y}_i$ ) and binary indicators (for  $y_i$ ).

#### **Interpretation:**

- **IoU Score**: Ranges from 0 (no overlap) to 1 (perfect overlap).
- Loss: L<sub>IoU</sub> = 0 when the predicted mask exactly matches the ground truth, and approaches 1 when there is minimal or no overlap.

## Advantages:

- Balance of Precision and Recall: IoU intrinsically captures how many true positives (correctly overlapped pixels) exist relative to the sum of all positives (both predicted and actual).
- **Direct Alignment with Evaluation**: IoU is a popular performance metric in many segmentation benchmarks [178, 179, 126, 180]; using it as a loss can align training with the final evaluation measure.

## **Challenges:**

- **Gradient Near Zero Overlap**: Similar to other region-based losses, if the model's prediction initially fails to overlap with the ground truth, the IoU gradient may provide weaker feedback than pixel-wise losses (cf. Section 6.1.1).
- **Multiple Classes or Large Images**: Extending IoU to multi-class setups requires summing or averaging per-class IoU, which can be computationally more involved if done naively for large images.

In many modern segmentation frameworks, IoU loss can be combined with other pixel-wise losses or with Dice-based metrics (§6.1.2) to balance local pixel accuracy and global region-overlap optimization.

## 6.1.4 Tversky Loss

Tversky Loss, introduced by Salehi et al. [181], generalizes Dice-based losses to better address class imbalances and varying error penalties in segmentation tasks. It introduces two hyperparameters,  $\alpha$  and  $\beta$ , controlling how false positives and false negatives are weighted. Consider the sets  $\hat{y}_i$  (predicted mask) and  $y_i$  (ground-truth mask):

$$\operatorname{Tversky}(\hat{y}_i, y_i) = 1 - \frac{|\hat{y}_i \cap y_i|}{|\hat{y}_i \cap y_i| + \alpha |\hat{y}_i \setminus y_i| + \beta |y_i \setminus \hat{y}_i|}.$$
 (101)

Here,  $|\hat{y}_i \cap y_i|$  is the intersection (true positives),  $|\hat{y}_i \setminus y_i|$  is the set difference (false positives), and  $|y_i \setminus \hat{y}_i|$  denotes missed pixels (false negatives). By adjusting  $\alpha, \beta \in [0, 1]$ , practitioners emphasize or de-emphasize certain types of errors.

## **Key Motivations:**

- Class Imbalance: If one class (e.g., lesions in medical images) is significantly smaller, Tversky Loss can be tuned ( $\alpha < \beta$  or vice versa) to penalize false negatives more heavily.
- Enhanced Customization: Different domains may require prioritizing false positives vs. false negatives. Tversky Loss provides a direct mechanism for these domain-specific trade-offs.

## **Typical Settings:**

- $\alpha = \beta = 0.5$ : Recovers Dice Loss (Section 6.1.2).
- $\alpha + \beta = 1$ : Preserves a balancing pattern; e.g.,  $\alpha = 0.7, \beta = 0.3$  puts more emphasis on penalizing missed positives.

#### **Use Cases:**

- **Medical Imaging**: Pathologies can be tiny and critical to detect. Weighting false negatives more heavily can reduce missed diagnoses.
- Multi-Class Extensions: Tversky Loss can be applied per class and then summed or averaged. This approach helps if multiple classes are imbalanced in different ways.

In general, Tversky Loss provides a customizable tool for balancing segmentation errors, especially in high-stakes or highly imbalanced tasks. By choosing appropriate  $\alpha$  and  $\beta$ , one can tailor the loss configuration to a desired error profile.

#### 6.1.5 Lovász Loss

Lovász Loss, proposed by Berman et al. [182], is designed to directly optimize Intersection over Union (IoU), offering a smooth, differentiable surrogate to the Jaccard index. Rather than aggregating pixel-wise errors, the loss treats the IoU as a submodular set function and applies a convex Lovász extension that approximates the discrete IoU in a manner suitable for gradient-based optimization.

**Definition:** Given predicted segmentation probabilities  $\{p_i\}_{i=1}^N$  and binary ground truth labels  $\{y_i\}_{i=1}^N$  (0 or 1), Lovász Loss reorders pixels by *confidence errors* and sums a piecewise linear function derived from the Lovász extension. Conceptually:

$$L_{\text{Lovasz}} = \sum_{i \in \text{errors}} \phi(m_i), \tag{102}$$

where  $m_i$  is the margin-based error term for pixel i (e.g.,  $m_i = 1 - p_i$  if  $y_i = 1$  or  $m_i = p_i$  if  $y_i = 0$ ), and  $\phi(\cdot)$  is the Lovász extension that approximates the IoU gap.

#### Advantages:

• **Direct IoU Optimization:** Since IoU is a set-based metric, standard pixel-wise losses (e.g., cross-entropy) may not fully align with maximizing IoU. Lovász Loss bridges this gap by operating on IoU-related errors directly.

- Imbalance Handling: By focusing on misclassified pixels (i.e., those with high margin errors), Lovász Loss naturally emphasizes classes or regions that are under-segmented.
- Smooth Differentiability: The Lovász extension yields a continuous, piecewise linear approximation to the discrete Jaccard index, enabling backpropagation without discontinuities.

**Usage:** Lovász Loss is valuable in tasks where IoU is the primary performance metric (e.g., large-scale semantic segmentation benchmarks). It often outperforms purely pixel-level losses in cases of class imbalance or when small object segmentation is crucial, thanks to its alignment with overlap-based evaluation criteria.

## 6.1.6 Focal Loss for Segmentation

Focal Loss, introduced by Lin et al. [60] to handle class imbalance in object detection, has also been adopted for semantic segmentation—particularly when one or more foreground classes are rare or difficult to detect.

**Definition:** For a binary segmentation scenario, let  $p \in [0, 1]$  denote the predicted probability that a pixel belongs to the foreground class. Denote the ground-truth label by  $y \in \{0, 1\}$ . Then the Focal Loss is

$$FL(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log(p_t), \tag{103}$$

where

$$p_t = \begin{cases} p, & \text{if } y = 1, \\ 1 - p, & \text{if } y = 0, \end{cases}$$
 (104)

 $\alpha_t \in [0,1]$  is a class weighting factor (often used to weight foreground vs. background), and  $\gamma \geq 0$  is the focusing parameter. In multi-class segmentation, Focal Loss typically applies to each class channel separately and sums or averages them.

#### Mechanism:

- Down-Weight Easy Examples: When a pixel is correctly classified  $(p_t \approx 1)$ ,  $(1 p_t)^{\gamma} \approx 0$ . This diminishes the contribution of "easy" pixels to the loss, redirecting the model's attention to harder or misclassified pixels.
- Adjustable Class Importance: The factor  $\alpha_t$  can emphasize minority classes, ensuring that performance on these classes is not overshadowed by more frequent classes or background pixels.

## **Applications:**

- **Highly Imbalanced Segmentation:** Focal Loss shines in contexts where large portions of the image belong to background or dominant classes, while small, important regions (e.g., tumors in medical imaging) risk neglect.
- Focus on Ambiguous Boundaries: By heavily penalizing misclassifications, Focal Loss encourages the network to refine boundaries and subtle regions that standard cross-entropy might overlook.

## **Hyperparameter Tuning:**

- $\alpha$ : Balances importance between positive (foreground) and negative (background).
- γ: Governs how much the loss "focuses" on hard examples. Typical values range from 1 to 5, though dataset-specific tuning is often required.

In general, Focal Loss for segmentation provides a strong alternative to plain cross-entropy in scenarios demanding fine-grained attention to rare or challenging classes, leading to improved coverage of small or ambiguous structures.

## **6.2** Segmentation Metrics

Various metrics exist to assess the performance of segmentation models, taking into account both per-pixel classification accuracy and the consistency of predicted object boundaries or instances. Table 14 presents a concise overview of commonly used segmentation metrics, highlighting their typical usage scenarios, data characteristics, primary advantages, and key limitations.

Metric	Usage	Data Characteristics	Advantages	Limitations
Pixel accuracy	Semantic segmentation tasks where each pixel belongs to a class Works if class frequencies are not too skewed	Images labeled at pixel level Suitable when classes are relatively balanced	Very simple Fast to compute Intuitive interpretation	Dominated by majority classes Ignores subtle region errors No sense of boundary quality
Boundary F1	Scenarios emphasiz- ing precise object contours Evaluates boundary alignment	Images where boundary delin- eation is crucial Uses a distance threshold for matches	Focuses on boundary correctness Captures alignment of object edges	Interior segmenta- tion ignored Choice of distance threshold affects scores Can be less robust to noise near edges
Mask AP	Instance segmentation Requires accurate pixel-level masks for each object	Images with multi- ple instances Each instance has a ground-truth mask Typically evaluated at various IoU thresh- olds	Captures both lo- calization and mask quality Standard benchmark in instance seg (e.g., COCO)	Sensitive to IoU thresholds Complex matching logic for multiple predictions Calibration of con- fidence scores is critical
Panoptic	Panoptic segmenta-	Images with pixel-	Single metric for	Requires matching

Table 14: Guidelines for selecting a metric for segmentation based on usage, data characteristics, advantages, and limitations.

## 6.2.1 Pixel Accuracy

Quality

Pixel Accuracy is a simple ratio describing how many pixels are classified correctly. For an image of size N pixels and C classes, let  $\hat{y}_i \in \{1, \dots, C\}$  be the model's predicted label for pixel i, and  $y_i$  the ground-truth label. Then,

level labels for both

Combines semantic

+ instance segments

Pixel Accuracy = 
$$\frac{1}{N} \sum_{i=1}^{N} \mathbb{I}[\hat{y}_i = y_i],$$
 (105)

instance detection

region

object

and area coverage

Balances

accuracy + recognition

instances + evaluat-

Overlaps can compli-

Not as common outside panoptic tasks

ing stuff classes

cate matching

where  $\mathbb{I}[\cdot]$  is an indicator function equal to 1 if the expression is true, and 0 otherwise.

Unifies stuff + things

## **Interpretation and Limitations:**

- Balanced vs. Imbalanced Classes: Pixel accuracy may be dominated by large or frequently occurring classes.
   In tasks with severe foreground-background imbalance, high pixel accuracy might conceal poor performance on smaller classes.
- **Simplicity:** Despite its limitations, pixel accuracy remains a fast and intuitive measure for many semantic segmentation scenarios, especially when classes appear in roughly equal proportions.
- Complementary Metrics: Metrics like IoU (§6.1.3) or Dice are often reported alongside pixel accuracy for a more robust assessment of segmentation quality, especially in unbalanced contexts.

# 6.2.2 Boundary F1 Score (BF)

The *Boundary F1 Score (BF)* [183], sometimes denoted BF, evaluates segmentation quality by focusing on *boundary delineation*. Instead of measuring per-pixel overlaps, it checks how accurately predicted boundaries align with ground-truth boundaries.

The steps to compute the BF score are the following:

- 1. **Match Predicted and Ground-Truth Boundaries:** For each ground-truth boundary, find the closest predicted boundary within a distance threshold  $\delta$ . Similarly, each predicted boundary is matched to any ground-truth boundary within the same threshold.
- 2. Compute Precision (P):

$$P = \frac{TP}{TP + FP},\tag{106}$$

where TP (true positives) counts predicted boundary points that are matched to a ground-truth boundary, and FP (false positives) counts predicted boundary points that have no ground-truth match.

3. Compute Recall (R):

$$R = \frac{TP}{TP + FN},\tag{107}$$

where FN (false negatives) is the number of ground-truth boundary points not matched by any prediction.

4. Boundary F1 Score:

$$BF = \frac{2 \cdot P \cdot R}{P + R}.$$
 (108)

#### **Interpretation:**

- **Distance Threshold**  $\delta$ : Defines how close a predicted boundary must be to the ground-truth boundary to count as a match. Smaller  $\delta$  demands higher precision of boundary alignment; larger  $\delta$  is more lenient.
- Focused on Boundary Quality: Suited for tasks where accurate object outlines are paramount, e.g., medical image analysis of organs or lesions with critical boundary definitions.

Although BF Score emphasizes boundary precision, it may not reflect how well large interior regions are segmented. Thus, it is commonly used alongside region-based metrics (IoU, Dice) for a complete segmentation evaluation.

## 6.2.3 Masked Average Precision (Mask AP)

Masked Average Precision (Mask AP) is the standard evaluation metric for instance segmentation, extending the concept of bounding-box AP to pixel-level segmentation masks. Instead of checking whether a predicted bounding box overlaps sufficiently with a ground-truth box, Mask AP requires the predicted segmentation mask to achieve a certain Intersection over Union (IoU) threshold with the corresponding ground-truth mask.

**Definition:** Let  $\mathcal{G}$  be the set of ground-truth masks, and  $\mathcal{P}$  be the set of predicted masks for a given dataset of images. Each predicted mask  $\mathbf{p}_i \in \mathcal{P}$  is accompanied by a confidence score  $s_i$ . For each ground-truth mask  $\mathbf{g}_j \in \mathcal{G}$  associated with a category  $c_j$ , define the mask IoU as:

$$IoU(\mathbf{p}_i, \mathbf{g}_j) = \frac{\mathbf{p}_i \cap \mathbf{g}_j}{\mathbf{p}_i \cup \mathbf{g}_j}.$$
(109)

A predicted mask  $p_i$  is considered a *true positive* if (1) its predicted category matches the category of a ground-truth mask, (2) it has the highest IoU with that ground-truth mask compared to other predictions of the same category, and (3) the IoU surpasses a given threshold  $\tau$  (e.g., 0.50, 0.75). Otherwise, it is counted as a *false positive*.

After sorting predictions by decreasing confidence score  $s_i$ , define:

$$Precision(k) = \frac{\text{# of true positives among top-}k \text{ predictions}}{k},$$
(110)

$$Recall(k) = \frac{\text{# of true positives among top-}k \text{ predictions}}{\text{total # of ground-truth masks in that category}}.$$
 (111)

By varying the decision threshold on  $s_i$  (from high to low), a precision–recall curve is traced.

Average Precision (AP) at a Fixed IoU Threshold: For each category (or across categories), the AP at IoU threshold  $\tau$  is the area under the precision–recall curve. Formally, if one samples recall at discrete points  $\{r_j\}$ , the average precision can be approximated as:

$$AP_{\tau} = \int_{0}^{1} \operatorname{Precision}(r) d(r), \tag{112}$$

where  $\operatorname{Precision}(r)$  is interpolated at each recall level r. In practice, the MS-COCO dataset uses 101 recall points (from 0 to 1 in increments of 0.01) and averages the maximum precision observed at or above each recall.

**Mean Mask AP Over Multiple IoU Thresholds:** To account for varying segmentation qualities, one typically computes AP at multiple IoU thresholds  $\{\tau_k\}$ , such as  $\tau \in \{0.50, 0.55, \dots, 0.95\}$ . The final *mean Mask AP* is the average of AP<sub> $\tau$ </sub> across these thresholds, thereby providing a more comprehensive evaluation:

$$mAP_{mask} = \frac{1}{|\{\tau_k\}|} \sum_{\tau_k} AP_{\tau_k}.$$
(113)

#### Usage:

- **Instance Segmentation Benchmarks:** Datasets like MS-COCO, Cityscapes, or LVIS employ *mask IoU* and average precision across multiple IoU thresholds. This helps gauge the model's ability to accurately localize instances and produce precise boundaries.
- Category-level or Overall AP: One can compute Mask AP separately for each object category or average across categories to yield an overall performance measure. Typically reported are AP<sub>50</sub> (IoU=0.50), AP<sub>75</sub> (IoU=0.75), and the AP over  $\tau \in [0.50:0.05:0.95]$ .
- **Object Size Variations:** Additional breakdowns—AP<sub>S</sub>, AP<sub>M</sub>, AP<sub>L</sub>—reflect performance on small, medium, and large objects, respectively, illustrating how the model scales across object sizes.

#### **Limitations:**

- **Strict IoU Thresholds:** AP relies on discrete IoU thresholds. Small changes in mask boundaries can affect the IoU, potentially flipping a prediction between true and false positive if near the threshold.
- Confidence Calibration: Precise rank ordering of predicted masks by confidence is crucial. Poor calibration may degrade the precision—recall curve and thus the final AP.
- Multiple Instances and Overlapping: In crowded scenes, partial overlap or occlusion among objects can complicate matching predictions to ground truth, affecting the measured IoU and scoring.

Masked Average Precision (Mask AP) generalizes object detection's bounding-box AP to pixel-level segmentation tasks. By computing how well predicted masks match ground-truth masks across various IoU thresholds and confidence levels, **Mask AP** serves as the principal benchmark for instance segmentation performance. This measure ensures both correct object localization and accurate mask delineation.

# 6.2.4 Panoptic Quality (PQ)

Panoptic Quality (PQ) [139] is a unified metric for panoptic segmentation, which merges semantic segmentation ("stuff" classes) and instance segmentation ("thing" classes). PQ assesses both the accuracy of instance segmentation and the correct recognition of classes and object identities.

**Definition:** Let TP be the set of matched (predicted, ground-truth) segments with an Intersection-over-Union IoU(p,g) above a threshold (commonly 0.5). Let FP be the set of unmatched predicted segments, and FN be the set of ground-truth segments not matched by any prediction. Then,

$$PQ = \underbrace{\frac{\sum_{(p,g) \in \text{TP}} \text{IoU}(p,g)}{|\text{TP}|}}_{\text{Segmentation Quality (SQ)}} \times \underbrace{\frac{|\text{TP}|}{|\text{TP}| + \frac{1}{2}|\text{FP}| + \frac{1}{2}|\text{FN}|}}_{\text{Recognition Quality (RQ)}}.$$
(114)

## **Components:**

- Segmentation Quality (SQ): Averages IoU over all correctly matched segments (TP), measuring how well each object's shape overlaps with its ground truth.
- **Recognition Quality (RQ):** Reflects detection performance. By counting TP over TP  $+\frac{1}{2}$  (FP + FN), RQ penalizes both spurious predictions and missed instances.

## **Interpretation:**

- Range and Meaning:  $0 \le PQ \le 1$ . Higher PQ implies better per-instance segmentation (SQ high) and accurate instance recognition (RQ high).
- Comprehensive vs. Other Metrics: PQ is more holistic than mIoU or average precision alone, as it requires both correct pixel-level segmentation and correct instance labeling.
- **Application to "Stuff" vs. "Things":** Panoptic segmentation divides classes into "stuff" (e.g., roads, sky) and "things" (e.g., cars, people). Many benchmarks provide PQ metrics separately or in aggregate for each category group.

PQ is considered a robust choice in real-world scenarios with multiple overlapping objects and background classes. It balances region accuracy and instance identification, making it invaluable for tasks such as autonomous driving and complex scene understanding.

# 7 Face Recognition

Face recognition is the task of matching an individual's face in an image or video to a corresponding identity in a database of faces. Deep learning models, typically Convolutional Neural Networks (CNNs) or Transformers [184], are trained on large datasets of face images to extract discriminative features. These features enable accurate matching between input faces and stored identities. Face recognition finds applications in areas such as *security* [185, 186], *social media* [187], and *biometric identification systems* [188, 189].

## 7.1 Face Recognition Loss Functions and Metrics

Loss functions for face recognition often focus on preserving fine-grained relational structure among face embeddings (i.e., feature vectors). Broadly, they can be categorized as:

- Classification-based losses: Softmax, A-Softmax, Center Loss, Large-Margin Cosine Loss, and Additive Angular Margin Loss.
- Representation-based losses: Triplet Loss, Contrastive Loss, Circle Loss, and Barlow Twins Loss.

Common evaluation metrics for face recognition mirror general classification metrics—accuracy, precision, recall, F1-score, ROC curves—but often specialized protocols measure *verification* (is this the same person or not?) and *identification* (which person is this?). Table 15 summarizes frequently used face-recognition loss functions, including their typical data properties, advantages, and limitations. Below, we discuss the details of each.

## 7.1.1 Softmax Loss

Softmax Loss (cross-entropy) is a canonical classification objective. In face recognition, the network typically outputs a logit vector  $\mathbf{f}(x)$  for each identity class, which is mapped to probabilities via the softmax function. For a sample (x,y) (face image x of true identity y), let  $\mathbf{W} \in \mathbb{R}^{d \times n}$  be the learnable weights (one column per class) and  $\mathbf{b} \in \mathbb{R}^n$  the bias. Then each logit is

$$f_j(x) = \mathbf{w}_j^{\top} \mathbf{x} + b_j, \tag{115}$$

where  $\mathbf{x} \in \mathbb{R}^d$  is the deep feature vector extracted by the network. The probability of class j is

$$P(y = j \mid x; \mathbf{W}) = \frac{\exp(f_j(x))}{\sum_{k=1}^{n} \exp(f_k(x))}.$$
 (116)

The cross-entropy loss for a batch of N samples is

Table 15: Guidelines for selecting a suitable loss function for face recognition based on usage, data characteristics, advantages, and limitations.

Function	Usage	<b>Data Characteristics</b>	Advantages	Limitations
Softmax	Multi-class classifica- tion Common for image recognition	Large labeled datasets Clear class boundaries	Simple and widely used Stable training	Lacks explicit margin Limited discrimina- tive power
A-	Face recognition	Intraclass similarity	Improves class separa-	Sensitive to margin hy-
Softmax	Tasks needing angular margin	Need for angular sepa- rability	bility Enforces angular constraints	perparams Can be unstable if not tuned
Center	Combined with soft-	Data with well-	Tightens within-class	Requires balancing
Loss	max	defined classes	features	two losses
	Encourages intra-class clustering	Useful for identification tasks	Improves clustering	Needs careful center updates
CosFace	Large-margin face recognition	High intraclass variability	Direct cosine margin Usually stable training	Requires margin tun- ing
	Cosine-based separation	Need robust bound- aries		May not generalize to all tasks
ArcFace	Face/ID recognition Angular margin in fea-	Subtle interclass dif- ferences	Additive margin improves separation	Sensitive to margin and scale
	ture space	High-precision scenarios	Strong geometric intuition	Higher compute cost
Triplet	Verification and re-	Needs anchor-positive-	Flexible for unseen	Triplet mining is com-
Loss	trieval Enforces relative distances	negative Good for similarity tasks	classes Learns a metric space	plex Slower convergence
Contrastive	Siamese networks	Requires simi-	Straightforward met-	Single margin for all
loss	Pairwise similar-	lar/dissimilar labels	ric learning	negatives
	ity/dissimilarity	Potential imbalanced pairs	Extensible to new classes	Sensitive to pair sampling
Circle	Tasks with imbalanced	Need pairwise similar-	Optimizes both pos	Tuning margins can
Loss	data	ity scores	and neg pairs	be tricky
	Simultaneous pos/neg optimization	Complex data distribu- tions	Clear similarity-space target	Sensitive to hyper- params
Barlow	Self-supervised	Large unlabeled data	Removes redundancy	Requires large batch
Twins	Learns decorrelated	High-dimensional em-	No explicit negatives	size
1 11110	representations	beddings	needed	Sensitive to augmenta-
SimSiam	Self-supervised	Large unlabeled data	Simple approach	Sensitive to hyper-
	No negative pairs	Relies on augmenta- tions	Stop-gradient avoids collapse	params May learn low-level features

$$\mathcal{L}_{\text{softmax}} = -\frac{1}{N} \sum_{i=1}^{N} \log(P(y = y_i \mid x_i; \mathbf{W})). \tag{117}$$

**Limitations for Face Recognition:** Although softmax can learn to classify multiple identities, it does not explicitly enforce large margins between different classes or compress intra-class variability. This may hinder the network's ability to discriminate among visually similar faces. Consequently, more sophisticated margin-based losses or representation-learning objectives are often employed to enhance separation in the embedding space.

## 7.1.2 A-Softmax (SphereFace) Loss

A-Softmax [190], also referred to as SphereFace Loss, improves upon standard softmax by introducing an angular margin, thus enhancing the embedding discriminativeness. The key idea is to normalize both the weights and feature vectors so that classification depends on the angle between  $\mathbf{x}$  and each class vector  $\mathbf{w}_i$ .

# **Derivation:**

- 1. Weight Normalization: Let  $\|\mathbf{w}_j\| = 1$  for each class j. In practice, each column of  $\mathbf{W}$  is normalized, ensuring  $\|\mathbf{w}_j\| = 1$ .
- 2. **Feature Normalization:** Consider the feature vector  $\mathbf{x}$  with norm  $\|\mathbf{x}\|$ . The logit for class j effectively becomes  $\|\mathbf{x}\| \|\mathbf{w}_j\| \cos(\theta_j) = \|\mathbf{x}\| \cos(\theta_j)$ , where  $\theta_j$  is the angle between  $\mathbf{x}$  and  $\mathbf{w}_j$ .
- 3. **Angular Margin** m: A-Softmax modifies the logit for the true class y by replacing  $\cos(\theta_y)$  with  $\cos(m\theta_y)$ . This imposes a stricter requirement for correct classification, forcing  $\theta_y$  to be smaller (or  $\cos(\theta_y)$  bigger) by a factor of m. Formally,

$$\cos(m\,\theta_y) = \cos(m\arccos(\cos(\theta_y))). \tag{118}$$

4. Scaled Logits: The final logits become

$$\hat{y}_y = \|\mathbf{x}\| \cos(m\,\theta_y), \quad \hat{y}_j = \|\mathbf{x}\| \cos(\theta_j), \quad j \neq y. \tag{119}$$

The cross-entropy loss uses these adjusted logits:

$$\mathcal{L}_{\text{A-softmax}} = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{\exp(\|\mathbf{x}_i\| \cos(m \,\theta_{y_i}))}{\exp(\|\mathbf{x}_i\| \cos(m \,\theta_{y_i})) + \sum_{j \neq y_i} \exp(\|\mathbf{x}_i\| \cos(\theta_j))}. \tag{120}$$

## **Benefits for Face Recognition:**

- Margin-Based Separation: By enforcing  $\cos(m\theta)$  for the true class, the network encourages tighter intra-class clustering and larger inter-class separation in the angular space.
- **Better Discriminability:** Compared to plain softmax, A-Softmax reduces misclassification among visually similar faces, thus improving verification and identification metrics.

#### **Practical Considerations:**

- Margin m Tuning: Larger m leads to stronger separation but can be harder to optimize if m is too large.
- Scaling Factor: Some implementations multiply the norms by an additional scaling factor s to stabilize training.

A-Softmax generally exemplifies how incorporating an *angular margin* contributes to more resilient face embeddings. This method is favored in numerous face recognition systems that require enhanced inter-class distinction and tightly-knit intra-class clusters.

#### 7.1.3 Center Loss

Center Loss [191] is designed to reduce intra-class variation in feature representations by learning a class center for each identity and penalizing the distance between each sample's feature vector and its corresponding class center. When combined with a primary classification loss (e.g., softmax), this approach yields embeddings that are not only correctly classified but also tightly clustered by identity.

**Formulation:** Let  $\{\mathbf{x}_i\}_{i=1}^n$  be the feature vectors extracted by the network for a mini-batch of n samples, and let  $y_i \in \{1, \dots, K\}$  denote the class (identity) label of sample i. For each class k, we maintain a center vector  $\mathbf{c}_k \in \mathbb{R}^d$ . The *Center Loss* is

$$L_{\text{center}} = \frac{1}{2} \sum_{i=1}^{n} \|\mathbf{x}_{i} - \mathbf{c}_{y_{i}}\|_{2}^{2}.$$
 (121)

During backpropagation,  $\{c_k\}$  are also learnable parameters. To avoid abrupt fluctuations, the class centers are updated via an exponential moving average. For a sample  $(x_i, y_i)$ , the update rule is

$$\mathbf{c}_{y_i}^{(t+1)} = \mathbf{c}_{y_i}^{(t)} + \alpha \left( \mathbf{x}_i - \mathbf{c}_{y_i}^{(t)} \right), \tag{122}$$

where  $\alpha \in (0,1)$  is a small learning rate for the center vectors, ensuring smoother shifts.

Center Loss is typically added to the primary classification loss (e.g., softmax cross-entropy):

$$L = L_{\text{softmax}} + \lambda L_{\text{center}}, \tag{123}$$

where  $\lambda > 0$  balances the classification term and the clustering term. As a result, the network learns to position each sample close to its class center, improving intra-class compactness and discriminative power—critical qualities in face recognition tasks, where multiple faces must be distinguished with high accuracy.

#### 7.1.4 CosFace: Large-Margin Cosine Loss

CosFace [192], also known as the Large-Margin Cosine Loss, enhances face recognition by introducing a margin in cosine space rather than in the angular domain (as in SphereFace). The key idea is to directly subtract a margin m from the cosine similarity of the correct class, thereby enforcing a stricter separation between classes.

Let  $\mathbf{x}_i \in \mathbb{R}^d$  be the feature for a sample i, and let  $\mathbf{w}_j \in \mathbb{R}^d$  be the weight vector representing class j. CosFace normalizes both  $\mathbf{x}_i$  and  $\mathbf{w}_j$  so that  $\|\mathbf{x}_i\| = 1$  and  $\|\mathbf{w}_j\| = 1$ . The cosine of the angle  $\theta_j$  between  $\mathbf{x}_i$  and  $\mathbf{w}_j$  is:

$$\cos \theta_i = \mathbf{w}_i^{\top} \mathbf{x}_i. \tag{124}$$

If  $y_i$  is the correct class for sample i, the logit for this class is adjusted to  $\cos \theta_{y_i} - m$ . All other classes keep  $\cos \theta_j$  unchanged. Let s > 0 be a scale factor. The *CosFace Loss* for sample  $\mathbf{x}_i$  is:

$$L_i = -\log \frac{\exp(s(\cos \theta_{y_i} - m))}{\exp(s(\cos \theta_{y_i} - m)) + \sum_{j \neq y_i} \exp(s \cos \theta_j)}.$$
 (125)

# **Interpretation:**

- Margin in Cosine Space: By subtracting m from the correct class's cosine similarity, CosFace forces a larger gap between different classes in the normalized feature space.
- Stable Optimization: Unlike SphereFace, which manipulates angles using  $\cos(m\theta)$ , CosFace's direct subtraction of m is simpler and avoids potential non-monotonic behaviors.
- Normalization: Both  $x_i$  and  $w_j$  are normalized, preventing the network from trivially increasing norms to minimize the loss. Hence, the model truly learns to discriminate via angular separation.

CosFace often outperforms standard softmax or SphereFace in challenging face benchmarks by providing a direct margin in the cosine domain, leading to better generalization and discriminability.

## 7.1.5 ArcFace: Additive Angular Margin Loss

ArcFace [193], also called Additive Angular Margin Loss, refines margin-based approaches by placing a margin additively in the angle itself. This ensures a more geometrically interpretable boundary on the hypersphere.

**Angular Margin:** After normalizing  $\mathbf{x}_i$  and  $\mathbf{w}_{y_i}$ , let  $\theta_{y_i}$  be the angle between  $\mathbf{x}_i$  and the correct class weight  $\mathbf{w}_{y_i}$ . ArcFace modifies  $\theta_{y_i}$  to  $\theta_{y_i} + m$ , where m > 0 is a constant margin. Consequently,  $\cos(\theta_{y_i} + m)$  is used in place of  $\cos(\theta_{y_i})$ . With scale factor s, the logit for the correct class becomes:

$$s\cos(\theta_{u_i} + m). \tag{126}$$

**Loss Definition:** For a sample  $(\mathbf{x}_i, y_i)$ , the *ArcFace Loss* is:

$$L_i = -\log \frac{\exp(s\cos(\theta_{y_i} + m))}{\exp(s\cos(\theta_{y_i} + m)) + \sum_{j \neq y_i} \exp(s\cos(\theta_j))}.$$
(127)

#### **Geometric Interpretation:**

• Additive Angle Margin: By shifting the angle by m, ArcFace enforces that the model must produce a higher cosine for the correct class to compensate for this extra margin. This effectively increases inter-class separations on the hypersphere.

- **Stable and Interpretable:** The additive nature in the angle domain yields a monotonic boundary, making it simpler to optimize than purely multiplicative angle margins.
- State-of-the-Art Accuracy: ArcFace typically outperforms earlier margin-based losses (SphereFace, CosFace) on tasks like LFW or MegaFace, providing a robust face embedding with high intra-class compactness and inter-class separability [194].

In general, ArcFace advances the margin-based paradigm by adding a direct angular offset, leading to improved face verification and identification performance under challenging conditions of high identity similarity or limited training data.

## 7.1.6 Triplet Loss

Triplet Loss [195, 196] is a classical metric-learning approach in face recognition. It aims to ensure that images of the same person (positive) lie closer in the learned embedding space to a reference image (anchor) than do images of different persons (negative). By enforcing a margin separation, the model learns discriminative features, reducing intra-class distances while increasing inter-class distances.

**Definition:** A triplet consists of an anchor A, a positive P (same identity as A), and a negative N (different identity). Let  $\mathbf{f}_A, \mathbf{f}_P, \mathbf{f}_N \in \mathbb{R}^d$  be their respective embeddings. Triplet Loss imposes:

$$L_{\text{triplet}} = \max \left\{ 0, \|\mathbf{f}_A - \mathbf{f}_P\|_2^2 - \|\mathbf{f}_A - \mathbf{f}_N\|_2^2 + \alpha \right\},$$
 (128)

where  $\|\cdot\|_2$  denotes the Euclidean (L2) distance and  $\alpha>0$  is the margin parameter. Minimizing  $L_{\text{triplet}}$  enforces:

$$\|\mathbf{f}_A - \mathbf{f}_P\|_2^2 + \alpha < \|\mathbf{f}_A - \mathbf{f}_N\|_2^2,$$
 (129)

so that the anchor–positive distance is smaller than the anchor–negative distance by at least  $\alpha$ .

## **Interpretation and Key Points:**

- Margin α: A sufficiently large margin fosters clearer separation but may slow training if set too high. Tuning
  α is critical for balanced learning.
- Sampling Strategy: Effectiveness depends heavily on how triplets are chosen. Hard or semi-hard negative mining can accelerate convergence by providing informative examples [197, 198].
- **Embedding Consistency:** Because the loss compares distances among triplets, the network learns a *metric space* where same-identity samples cluster tightly, and different-identity samples are pushed apart.

Triplet Loss has proven successful in applications like *FaceNet* [196], where a single embedding space suffices for tasks like face verification or recognition without needing explicit class-based training for each new identity.

#### 7.1.7 Contrastive Loss

Contrastive Loss [199] encourages pairs of images deemed similar to lie close together in the embedding space and pairs deemed dissimilar to lie further apart. This approach is often implemented via a Siamese network architecture, which processes pairs  $(x_i^a, x_i^p)$  or  $(x_i^a, x_i^n)$  to learn meaningful relative distances.

**Definition:** Given a labeled pair  $(x_i^a, x_i^p)$  indicating *similarity* (same identity) or  $(x_i^a, x_i^n)$  indicating *dissimilarity* (different identities), let the network's embeddings be  $f(x_i^a)$  and  $f(x_i^p)$  or  $f(x_i^n)$ . A binary label  $y_i \in \{0, 1\}$  signifies whether the pair is similar  $(y_i = 1)$  or dissimilar  $(y_i = 0)$ . The *Contrastive Loss* is often written as:

$$L = \frac{1}{N} \sum_{i=1}^{N} \left[ y_i \| f(x_i^a) - f(x_i^p) \|_2^2 + (1 - y_i) \max \left( 0, \ m - \| f(x_i^a) - f(x_i^n) \|_2^2 \right) \right], \tag{130}$$

where m>0 is a margin hyperparameter, N is the number of pairs in a batch, and  $\|\cdot\|_2$  denotes the L2 distance.

## **Terms Explanation:**

- Similar Pairs  $(y_i = 1)$ : The term  $||f(x_i^a) f(x_i^p)||_2^2$  is minimized, bringing matched samples closer.
- Dissimilar Pairs  $(y_i = 0)$ : The loss encourages  $||f(x_i^a) f(x_i^n)||_2^2 \ge m$ . If the distance is already beyond m, the contribution is 0, i.e., no further penalty.

# **Key Properties:**

- Margin m: Controls how separate dissimilar pairs must be. Too small a margin may under-separate classes, while too large may impede convergence.
- **Metric Generalization:** Since the model learns a shared embedding space, *new* classes (identities) can be recognized without retraining a classifier—only the distance function is used.
- **Possible Limitation:** All negative pairs are treated equally, assuming a uniform notion of dissimilarity. This might be simplistic if some negative pairs are "less dissimilar" than others [194, 197, 198].

Contrastive Loss underlies many face verification systems, especially those requiring flexible addition of new individuals. By pushing same-identity pairs together and different-identity pairs beyond a margin, the learned feature space naturally supports verifying whether two faces belong to the same person.

#### 7.1.8 Circle Loss

Circle Loss [200] aims to enhance the discriminative power of learned embeddings by simultaneously maximizing positive-pair similarity and minimizing negative-pair similarity, all while maintaining a circle-like decision boundary in the similarity space. This approach handles challenges like imbalance and complex distributions, which are prevalent in tasks such as face recognition.

**Definition:** Let  $\{s_{pos_i}\}$  and  $\{s_{neg_j}\}$  represent similarity scores for positive and negative pairs, respectively. Circle Loss introduces two margins  $O_{pos} < O_{neg}$ , plus slack variables  $\alpha_{pos_i}$ ,  $\alpha_{neg_j}$  to account for whether each pair already meets its margin requirement. A scaling factor  $\gamma > 0$  amplifies the effects of violating these margins. Formally, each batch contributes:

$$\alpha_{pos_{i}} = \max(O_{pos} - s_{pos_{i}}, 0),$$

$$\alpha_{neg_{j}} = \max(s_{neg_{j}} - O_{neg}, 0),$$

$$\operatorname{sum}_{pos} = \sum_{i} \exp\left[-\gamma \alpha_{pos_{i}} (s_{pos_{i}} - O_{pos})\right],$$

$$\operatorname{sum}_{neg} = \sum_{j} \exp\left[\gamma \alpha_{neg_{j}} (s_{neg_{j}} - O_{neg})\right],$$

$$L_{\text{Circle}} = \ln(1 + \operatorname{sum}_{pos} \times \operatorname{sum}_{neg}).$$
(131)

## **Interpretation:**

- Margins  $(O_{pos}, O_{neg})$ : Encourages positive similarities  $s_{pos_i}$  to exceed  $O_{pos}$  and negative similarities  $s_{neg_j}$  to remain below  $O_{neg}$ . Typically  $O_{pos} < O_{neg}$ .
- Slack Variables  $\alpha_{pos_i}, \alpha_{neg_j}$ : If a pair already satisfies its margin, its slack variable is 0, contributing minimally to the exponential terms.
- Scaling  $\gamma$ : Magnifies the penalty for pairs failing their margin constraints. Larger  $\gamma$  imposes harsher penalties but may make training more sensitive.

## **Practical Considerations:**

- Margin Tuning: Choosing  $O_{pos}$  and  $O_{neg}$  (or a gap m with  $O_{neg} = O_{pos} + m$ ) is crucial. Inadequate margins may yield insufficient separation; overly large margins risk training instability.
- Similarity Space Convergence: Circle Loss directs optimization toward a "circle-like" boundary in the  $(s_{neg}, s_{pos})$  space, focusing the network on achieving consistent separation for both positive and negative pairs.

Circle Loss often excels in scenarios with complex inter-class relationships or skewed distributions, driving robust separation in the similarity space without requiring explicit angular or cosine-based margins.

#### 7.1.9 Barlow Twins Loss

Barlow Twins [201] is a self-supervised approach that learns invariant and decorrelated feature representations. By processing two different augmentations of the same input, it aligns them in embedding space while reducing redundancy across feature dimensions. Though not specific to face recognition, its emphasis on robust, non-redundant features can benefit large-scale facial representation learning.

**Notation:** Consider embeddings  $\mathbf{z}_a, \mathbf{z}_b \in \mathbb{R}^D$  produced by the network for two augmentations of the same image. Each embedding is normalized across a batch of size N, yielding  $\mathbf{z}_{a_{\text{norm}}}$  and  $\mathbf{z}_{b_{\text{norm}}}$  with zero mean and unit variance over the batch dimension.

**Cross-Correlation Computation:** Construct the  $D \times D$  cross-correlation matrix

$$\mathbf{C} = \frac{1}{N} \, \mathbf{z}_{a_{\text{norm}}}^{\top} \, \mathbf{z}_{b_{\text{norm}}}. \tag{132}$$

**Difference and Scaling:** Subtract the identity matrix I and square the result. Then optionally scale off-diagonal entries by a factor  $\lambda$ :

$$\mathbf{C}_{\text{diff}} = (\mathbf{C} - \mathbf{I})^2,\tag{133}$$

$$\mathbf{C}_{\text{diff}_{ij}} = \begin{cases} \mathbf{C}_{\text{diff}_{ij}}, & i = j, \\ \lambda \, \mathbf{C}_{\text{diff}_{ij}}, & i \neq j. \end{cases}$$
(134)

**Final Barlow Twins Loss:** Sum up all elements of the adjusted matrix:

$$L = \sum_{i=1}^{D} \sum_{j=1}^{D} \mathbf{C}_{\text{diff}_{ij}}.$$
(135)

Goals:

- High Diagonal Correlation: (C-I) drives diagonal elements of C to be 1, forcing  $z_a$  and  $z_b$  to match for each dimension.
- Low Off-Diagonal Correlation: Off-diagonal entries are penalized to be near 0, reducing redundancy across different dimensions of the learned representation.

Barlow Twins does not rely on negative examples, thus reducing data expansion overhead. However, it typically requires large embedding dimensionality and carefully chosen augmentations to perform effectively. In face recognition contexts, it can produce high-level features that remain robust under various transformations (occlusion, lighting changes), though additional fine-tuning or specialized augmentation may be needed.

### 7.1.10 SimSiam Loss

SimSiam [202] is another self-supervised method that learns invariant representations by comparing two augmented views of the same image. Although not specifically tailored for face recognition, it can be applied for unsupervised facial feature learning, leveraging data augmentations to achieve robust embeddings.

**Architecture and Loss:** Given two augmentations  $x_1$  and  $x_2$  of the same image, the network produces features  $\mathbf{f}_1$ ,  $\mathbf{f}_2$  and predictions  $\mathbf{p}_1$ ,  $\mathbf{p}_2$ . The *SimSiam Loss* is defined via negative cosine similarity between  $\mathbf{p}_i$  on one view and  $\mathbf{f}_j$  on the other:

$$L = -\frac{1}{2} \left[ \frac{\mathbf{p}_1^{\mathsf{T}} \mathbf{f}_2}{\|\mathbf{p}_1\| \|\mathbf{f}_2\|} + \frac{\mathbf{p}_2^{\mathsf{T}} \mathbf{f}_1}{\|\mathbf{p}_2\| \|\mathbf{f}_1\|} \right]. \tag{136}$$

The *stop-gradient* operation is applied to  $\mathbf{f}_2$  in the first term and  $\mathbf{f}_1$  in the second, preventing trivial collapse (identical outputs for all inputs).

#### Advantages:

- 1. **No Negative Pairs:** Unlike contrastive methods requiring pairs of dissimilar samples, SimSiam relies purely on positive pairs (two augmentations of the same image), simplifying data structures.
- 2. **Stop-Gradient Mechanism:** Crucial for avoiding degenerate solutions where all embeddings converge to a single vector.
- 3. **Simple Architecture:** SimSiam uses a straightforward predictor head and a symmetric design, making it less complex than some other self-supervised methods.

#### **Drawbacks:**

- 1. **Hyperparameter Tuning:** Performance can be sensitive to learning rate, weight decay, or batch size, requiring careful tuning.
- 2. **Dependence on Augmentations:** Like other self-supervised methods, the choice and strength of image transformations significantly impact the learned features.
- 3. **Potentially Non-semantic Features:** There is no absolute guarantee the model captures class-level semantics. The learned features might be robust to certain perturbations but not necessarily contain identity-relevant details.

In face recognition, SimSiam can serve as a pretraining approach to obtain general facial embeddings, later refined with supervised objectives (e.g., ArcFace or CosFace) for improved identity separation. Its reliance on strong augmentations and careful parameter settings underscores the importance of domain knowledge in designing unsupervised facial representation workflows.

# **8** Monocular Depth Estimation (MDE)

Monocular depth estimation in computer vision infers depth from a single 2D image, unlike stereo methods using multiple images. It has key applications in areas like autonomous driving and robotics, where multi-camera setups are often impractical. The main challenge is inferring 3D depth from a 2D projection, using cues like object size and scene context. Traditional methods with hand-crafted features had limited generalization. However, deep learning, especially CNNs and attention mechanisms, has dramatically improved depth estimation, leveraging large datasets for better accuracy. Supervised, self-supervised, and unsupervised methods using stereo pairs or video sequences are popular. Despite progress, challenges remain, such as handling varied lighting and occlusions and ensuring scalability for real-world scenarios. Current research aims to improve model efficiency, reduce computational needs, and integrate geometric and physics-based constraints for better predictions.

The following sections explain the loss functions and metrics utilized for depth estimation.

#### 8.1 Depth Estimation Loss Functions

The evolution of loss functions in monocular depth estimation (MDE) reflects a shift from simple global measures, like scale-invariant or point-wise errors, to composite losses that balance global accuracy and fine-grained detail. Modern approaches integrate structural and semantic priors through edge-aware and smoothness losses, while photometric consistency losses have become standard in self-supervised methods. Task-specific enhancements, such as scale-consistent and boundary-aware losses, address challenges like scale ambiguity and boundary delineation. The integration of adversarial, perceptual, and multi-domain training losses enables robustness to noisy data and improves generalization across diverse datasets. In general, MDE loss functions have advanced to prioritize depth accuracy, structural consistency, and computational efficiency.

Table 16 provides a detailed overview of commonly used loss functions in depth estimation, emphasizing their uses, characteristics of the data, advantages, and disadvantages.

Table 16: Guidelines for selecting a depth estimation loss function based on usage, data characteristics, advantages, and limitations.

Function	Usage	Data Characteristics	Advantages	Limitations
Point- wise MAE	Minimizes absolute difference Useful for moderate depth ranges	Moderate variance Less prone to out- liers	Easy to interpret Robust to large er- rors	Under-penalizes big residuals Slower convergence sometimes
Point- wise MSE	Minimizes squared difference Common in regres- sion	Works when outliers are rare Prefers smaller vari- ance	Penalizes large errors heavily Well-known, simple	Overemphasizes big errors Can oversmooth pre- dictions
Scale Invariant	Emphasizes relative depths Common for monoc- ular	Uncertain or inconsistent scale Single-view depth	Robust to global scale shifts Focuses on relative structure	May lose absolute scale Requires careful pa- rameter tuning
SSIM	Preserves structural details Perceptual measure	Depth maps with strong spatial cues Local contrast vital	Encourages tex- ture/edge fidelity Improves visual quality	Less focus on pixel- level accuracy Can be sensitive to noise
Photometric	Self-supervised from image recon. Minimizes appearance diffs	Multi-view data No explicit depth la- bels	Uses geometry consistency Good for unsupervised setups	Assumes consistent illumination/visibility Struggles with occlusions/dynamics
Disparity Smooth- ness	Regularizes depth in flat regions Weighted by image gradients	Clear edges help Texture-less areas benefit	Reduces random noise Preserves important contours	Can oversmooth actual edges Needs balance with edge terms
Appearance Match- ing	Reconstructs one view from another Uses L1 + SSIM	Stereo/multi-view data Lacks direct abso- lute supervision	Balances structure and pixel accuracy Effective in stereo setups	Depends on photo- metric constancy Sensitive to big viewpoint or light changes
L-R Consistency	Matches left/right disparities Penalizes inconsistent maps	Stereo pairs with known calibration Synchronized cap- tures	Improves stereo alignment Reduces mis- matched disparity	Needs two views Not for single-view data
BerHu	Combines $L_1$ and $L_2$ Handles wide error range	Large dynamic depth range Adaptive threshold	Balances large/small residuals Robust to big devia- tions	Threshold tuning is critical Varies across datasets
Edge Loss	Aligns depth edges with image edges Focuses on bound- aries	Useful for preserv- ing sharp contours Edge-based gradient check	Sharp transitions are maintained Reduces boundary blur	Sensitive to image noise Requires good edge detection
Min Reproj.	Picks min photomet- ric error across views Good for occlusion handling	Multi-view with dy- namic objects Avoids invalid pro- jections	Improves self- supervised stability Reduces occlusion artifacts	Can oversmooth in multi-valid areas May need extra masking
SSI	Scale/shift align- ment for diverse data Robust cross-dataset	Varied scales across datasets Ambiguous absolute depth	Better generalization Less scale- dependent	Extra overhead for scale/shift Sensitive to severe outliers

#### 8.1.1 Point-wise Error

*Point-wise errors* measure per-pixel discrepancies between the predicted depth map  $\hat{D}$  and the ground truth D. Let  $\{\hat{d}_i\}_{i=1}^N$  and  $\{d_i\}_{i=1}^N$  be the predicted and ground-truth depths for N valid pixels. Two common forms are *Mean Absolute Error (MAE)* and *Mean Squared Error (MSE)*.

Mean Absolute Error (MAE): MAE penalizes the average absolute deviation between prediction and ground truth:

$$\mathcal{L}_{\text{MAE}} = \frac{1}{N} \sum_{i=1}^{N} |\hat{d}_i - d_i|.$$
 (137)

MAE treats all residuals equally, making it robust to moderate outliers since large errors are weighted linearly.

**Mean Squared Error (MSE):** MSE penalizes large errors more heavily due to squaring:

$$\mathcal{L}_{MSE} = \frac{1}{N} \sum_{i=1}^{N} (\hat{d}_i - d_i)^2.$$
 (138)

While MSE is sensitive to outliers, it can drive the network to correct major depth misestimates quickly in early training stages.

**Usage and Limitations:** Point-wise errors are straightforward and have been employed in [203] and subsequent MDE works to minimize overall pixel-level depth residuals. However, they do not capture structural relationships or relative depth cues among neighboring pixels. Consequently, point-wise errors alone may yield suboptimal boundary delineation or fail in texture-less regions. To mitigate these shortcomings, many modern MDE frameworks combine point-wise errors with structural metrics (e.g., SSIM §8.1.3) or smoothness constraints (e.g., edge-aware regularization).

#### 8.1.2 Scale Invariant Error

*Scale-invariant error* addresses the inherent scale ambiguity in monocular depth estimation. Proposed by Eigen *et al.* [203], it shifts emphasis from absolute depth values to *relative* depth consistency.

**Definition:** Let  $\{\hat{d}_i\}_{i=1}^N$  and  $\{d_i\}_{i=1}^N$  be predicted and ground-truth depths for N pixels. The scale-invariant loss uses logarithms of these depths:

$$\mathcal{L}_{\text{scale-inv}} = \frac{1}{2} \sum_{i=1}^{N} (\log \hat{d}_i - \log d_i)^2 - \frac{\lambda}{N^2} \left( \sum_{i=1}^{N} (\log \hat{d}_i - \log d_i) \right)^2.$$
 (139)

The first term is an  $\ell_2$  measure on log-transformed depths, prioritizing multiplicative (relative) errors over additive. The second term counteracts mean bias in the log space, aligning the overall scale of  $\hat{D}$  to that of D. The hyperparameter  $\lambda$  controls how strongly the model penalizes that global log offset; commonly,  $\lambda=1$  reproduces the original scale-invariant formulation [203].

#### **Interpretation and Advantages:**

- Relative Depth Focus: By operating in log space, the model can match the *shape* of depth surfaces even if the absolute scale is off.
- Scale Ambiguity Mitigation: This error metric tolerates global scaling errors, crucial in monocular setups lacking absolute distance references.
- Combination with Other Losses: Often supplemented by smoothness (to encourage local regularity) or photometric alignment (in self-supervised settings), ensuring that relative depth structure remains consistent across the image.

Scale-invariant error is valuable in tasks where only relative depth matters. Nonetheless, if absolute metric scale is required (e.g., robotics tasks with real-world distances), additional constraints or multi-view geometry may be necessary to resolve scale fully.

## 8.1.3 Structural Similarity Index Measure (SSIM)

Structural Similarity Index Measure (SSIM) [204, 205] compares two images by factoring in luminance, contrast, and local structure. In Monocular Depth Estimation (MDE), SSIM is utilized to preserve perceptual fidelity in predicted depth maps by encouraging local structural consistency with respect to the ground truth.

**Formulation:** Let  $\hat{D}$  and D be the predicted and ground-truth depth maps, respectively, and denote  $\mu_{\hat{D}}$ ,  $\mu_D$  as local means,  $\sigma_{\hat{D}}$ ,  $\sigma_D$  as local standard deviations, and  $\sigma_{\hat{D}D}$  as the cross-covariance of  $\hat{D}$  and D over a small spatial window. The SSIM index for that window is:

$$SSIM(\hat{D}, D) = \frac{\left(2\mu_{\hat{D}}\mu_D + C_1\right)\left(2\,\sigma_{\hat{D}D} + C_2\right)}{\left(\mu_{\hat{D}}^2 + \mu_D^2 + C_1\right)\left(\sigma_{\hat{D}}^2 + \sigma_D^2 + C_2\right)},\tag{140}$$

where  $C_1$  and  $C_2$  are small constants to avoid division by zero. SSIM ranges from -1 to 1, with 1 indicating perfect structural alignment.

To use SSIM in a loss objective, we convert similarity to dissimilarity:

$$\mathcal{L}_{\text{SSIM}} = 1 - \text{SSIM}(\hat{D}, D). \tag{141}$$

Since SSIM is computed in local windows, a typical practice is to average over all windows:

$$\mathcal{L}_{\text{SSIM}} = \frac{1}{N} \sum_{i=1}^{N} \left( 1 - \text{SSIM}(\hat{D}_i, D_i) \right), \tag{142}$$

where N is the total number of local patches. This formulation encourages  $\hat{D}$  to exhibit structurally consistent edges and gradients relative to D. SSIM loss is often combined with numeric errors (e.g.,  $\ell_1$  or MSE) or other structural priors (e.g., smoothness) to balance overall depth accuracy with localized structural fidelity.

## 8.1.4 Photometric Loss

Photometric Loss [206] is commonly used in self-supervised monocular depth estimation. It leverages photometric consistency between different viewpoints rather than requiring ground-truth depth. The idea is to warp a source image  $I_t$  into a reference view  $I_r$  using the predicted depth  $\hat{D}$  and the camera pose T. Then the difference between  $I_r$  and the warped image  $I_s$  (synthesized from  $I_t$ ) forms the loss.

**Warping Procedure:** Given a depth map  $\hat{D}$  and relative pose T, a pixel in the reference view is projected into 3D using  $\hat{D}$ , then reprojected into the source frame to sample a color from  $I_t$ . Denote the warped image as:

$$I_s = \mathcal{W}(I_t, \hat{D}, T). \tag{143}$$

**Photometric Consistency Loss:** A combined measure of absolute intensity difference and structural similarity [206]:

$$\mathcal{L}_{\text{photometric}} = \frac{1}{N} \sum_{i=1}^{N} \left[ \alpha \frac{1 - \text{SSIM}(I_r^i, I_s^i)}{2} + (1 - \alpha) |I_r^i - I_s^i| \right], \tag{144}$$

where N indexes valid pixels,  $\alpha \in [0,1]$  weights the SSIM term relative to the absolute difference term, and  $I_r^i, I_s^i$  are pixel intensities in the reference and warped images respectively.

## **Usage and Limitations:**

• **Self-Supervised Depth:** Photometric loss is crucial when ground-truth depth is unavailable. The network infers  $\hat{D}$  purely from multi-view photometric consistency.

- Occlusions and Dynamic Scenes: Real-world scenarios with movement or reflectance changes violate the brightness constancy assumption. Solutions include masking invalid regions, minimum reprojection losses, and auto-masking of dynamic objects.
- Combination with Smoothness/Edge-Aware: Due to the possibility of local minima or incorrect warps, photometric loss is typically combined with smoothness (§8.1.5) and, optionally, geometric constraints for more robust depth estimates.

### 8.1.5 Disparity Smoothness Loss

Disparity Smoothness Loss [207] acts as a spatial regularizer, penalizing abrupt or noisy depth variations except where image edges imply real depth discontinuities. By weighting predicted depth gradients with image gradients, it promotes piecewise-smooth depth except near intensity edges.

**Formulation:** Let  $\hat{D}_{i,j}$  be the predicted depth (or disparity) at pixel (i,j), and let  $I_{i,j}$  be the corresponding intensity in a reference image. The smoothness loss is typically:

$$\mathcal{L}_{\text{smooth}} = \sum_{i,j} \left[ \left| \partial_x \hat{D}_{i,j} \right| e^{-\left| \partial_x I_{i,j} \right|} + \left| \partial_y \hat{D}_{i,j} \right| e^{-\left| \partial_y I_{i,j} \right|} \right], \tag{145}$$

where  $\partial_x$  and  $\partial_y$  denote horizontal and vertical gradients. Multiplying by  $\exp(-|\partial_x I_{i,j}|)$  or  $\exp(-|\partial_y I_{i,j}|)$  reduces the penalty near image edges.

#### **Interpretation and Extensions:**

- Spatial Coherence: In flat or texture-less regions, depth should remain relatively smooth. This term suppresses spurious gradient noise.
- Edge Preservation: Penalizing depth gradients less near strong intensity edges prevents over-smoothing objects' boundaries.
- Log-Scaled and Multi-Scale Variants: Some works apply smoothing to  $\log(\hat{D})$  to handle wide depth ranges uniformly; multi-scale smoothing addresses coherence across multiple image resolutions.

**Implementation in MDE Pipelines:** Smoothness loss is often combined with (i) *photometric reconstruction* (§8.1.4), (ii) *point-wise errors* (§8.1.1), or (iii) *scale-invariant terms* (§8.1.2). This synergy helps produce depth maps that are not only globally accurate but also consistent at object boundaries, reflecting the real scene structure.

# 8.1.6 Appearance Matching Loss

Appearance Matching Loss [207] enforces that a warped or reconstructed image should closely resemble its corresponding original view. By relying on photometric consistency, it obviates the need for ground-truth depth, making it particularly suited for self-supervised or semi-supervised depth estimation approaches.

**Definition:** Let I be the original (reference) image and  $\tilde{I}$  be the reconstructed (warped) image, derived from predicted disparity/depth. The *Appearance Matching Loss* is typically defined as a weighted sum of an  $\ell_1$  intensity difference and a Structural Similarity (SSIM) term:

$$\mathcal{L}_{\rm ap} = \frac{1}{N} \sum_{i,j} \left[ \alpha \frac{1 - \text{SSIM}(I_{ij}, \tilde{I}_{ij})}{2} + (1 - \alpha) |I_{ij} - \tilde{I}_{ij}| \right], \tag{146}$$

where N denotes the total number of pixels, SSIM is computed over a local window (often  $3 \times 3$ ), and  $\alpha \in [0, 1]$  balances the SSIM term against the absolute difference. A common choice is  $\alpha = 0.85$ .

#### **Interpretation and Usage:**

• Photometric Consistency: Encourages the warped image  $\tilde{I}$  to align with the reference I, leveraging stereo or temporal viewpoint shifts to infer geometry.

- SSIM Emphasis vs. Pixel-level Accuracy: The SSIM portion preserves local structure (edges, textures), while the  $\ell_1$  term maintains per-pixel intensity fidelity.
- **Self-Supervised Depth:** The network learns depth by minimizing reconstruction error, sidestepping the need for ground-truth disparities.

## **Practical Variations:**

- SSIM Computation: Often simplified to a  $3 \times 3$  box filter for computational efficiency.
- Loss Combination: Typically paired with smoothness losses (§8.1.5) or left-right consistency (§8.1.7) to ensure globally coherent depth.

## 8.1.7 Left-Right Consistency Loss

Left-Right Consistency Loss [207] is commonly employed in stereo-based or self-supervised depth estimation, reinforcing consistency between left-view and right-view disparity (or depth) maps without requiring ground-truth labels. It leverages two views,  $I^l$  and  $I^r$ , and their learned disparity fields  $\mathbf{d}^l$  and  $\mathbf{d}^r$ .

**Disparity Warping:** The network warps  $\mathbf{d}^r$  into the left coordinate frame via  $\mathbf{d}^l$ . For pixel (i, j), the location  $(j + d_{ij}^l)$  in the right disparity map is used. The *Left-Right Consistency Loss* penalizes divergence between these:

$$\mathcal{L}_{lr} = \frac{1}{N} \sum_{i,j} \left| d_{i,j}^l - d_{i,(j+d_{i,j}^l)}^r \right|, \tag{147}$$

where N is the number of valid pixels. The argument  $j + d_{i,j}^l$  indicates the horizontal shift due to the left-view disparity.

In many depth estimation tasks, especially when ground-truth data is scarce or expensive to obtain, stereo pairs serve as a form of weak supervision. By coupling  $\mathcal{L}_{lr}$  with a photometric reconstruction loss (see §8.1.6), the network learns to produce consistent disparities on both views and minimize appearance differences when one view is warped into the other. This setup removes the need for labeled depth maps, making the training process *unsupervised* (or self-supervised). Specifically, disparities are learned by enforcing internal geometric and photometric consistency, eliminating dependence on actual depth annotations. Enforcing left-right consistency complements the photometric loss, improving overall depth quality by penalizing mismatched disparities across views. Furthermore, large stereo datasets can be collected without manual labeling, making this approach appealing for real-world scenarios where labeled depth is limited.

**Total Pipeline Loss:** As introduced by [207],  $\mathcal{L}_{lr}$  is typically combined with appearance matching (§8.1.6) and smoothness (§8.1.5) terms:

$$\mathcal{L} = \alpha_{\rm ap} \, \mathcal{L}_{\rm ap} + \alpha_{\rm ds} \, \mathcal{L}_{\rm ds} + \alpha_{\rm lr} \, \mathcal{L}_{\rm lr},\tag{148}$$

where  $\alpha_{\rm ap}, \alpha_{\rm ds}, \alpha_{\rm lr}$  are hyperparameters controlling the relative importance of each term.

### **Advantages and Considerations:**

- Stereo Regularization: Ensures that left-view and right-view disparities remain consistent, reducing artifacts or mismatches.
- Training with Single-View Inputs: Although only one view might feed the main encoder, the other disparity map is still predicted, ensuring synergy between the two. This is especially useful when extending stereo-based methods to monocular sequences in a self-supervised manner.
- Occlusion Handling: Real stereo pairs have occluded regions visible only in one view. Usually, masks or confidence measures are introduced so that L<sub>lr</sub> excludes invalid pixels.
- **Generalizability:** By removing the need for explicit labels, left-right consistency methods can be adapted to various domains where collecting ground-truth depth is infeasible.

## 8.1.8 BerHu Loss (Reverse Huber)

BerHu Loss [208, 209] is a "Reverse Huber" formulation that switches between  $\ell_1$  and  $\ell_2$  norms based on a threshold  $\delta$ . It aims to robustly handle both small and large depth errors across wide depth ranges.

**Definition:** For a predicted depth map  $\{\hat{d}_i\}$  and ground-truth  $\{d_i\}$ , BerHu defines:

$$\mathcal{L}_{\text{BerHu}}(\hat{d}_i, d_i) = \begin{cases} |\hat{d}_i - d_i|, & \text{if } |\hat{d}_i - d_i| \le \delta, \\ \frac{(\hat{d}_i - d_i)^2 + \delta^2}{2\delta}, & \text{otherwise}, \end{cases}$$
(149)

where  $\delta$  is a threshold controlling where the loss transitions from  $\ell_1$  to  $\ell_2$ . If  $|\hat{d}_i - d_i| \leq \delta$ , the penalty is linear; if it exceeds  $\delta$ , the penalty grows quadratically.

**Adaptive Threshold:** One strategy sets  $\delta$  as a fraction of the maximum error in a batch  $\{\max_i |\hat{d}_i - d_i|\}$ , ensuring the threshold scales with current training dynamics. This can improve stability across varying depth ranges.

## **Usage and Benefits:**

- Small Errors vs. Large Errors: l
  1 handles finer deviations well, while the l
  2 region strongly penalizes large
  outliers.
- Wide Depth Ranges: Particularly effective where the disparity spans multiple orders of magnitude (e.g., indoor vs. outdoor scenes).

#### **Limitations:**

- Threshold Tuning: Incorrect  $\delta$  can hamper performance or slow convergence, motivating adaptive or datadriven choices.
- Structural Awareness: Like point-wise errors, BerHu does not inherently encode geometric or structural
  priors. Many MDE pipelines combine BerHu with edge-aware or scale-invariant terms to capture robust scene
  structure.

Thus, BerHu Loss offers a middle ground that linearly penalizes moderate errors but escalates penalties for large errors, making it suitable for real-world depth estimation tasks with varying depth scales.

#### 8.1.9 Edge Loss

Edge Loss [210] aims to preserve sharp transitions in the predicted depth map near object boundaries or edges. While smoothness terms (§8.1.5) encourage local coherence, they can inadvertently blur true discontinuities. Edge Loss mitigates this by aligning depth gradients with corresponding image gradients, ensuring that depth discontinuities match actual intensity edges in the reference image.

**Formulation:** Let  $\hat{D}$  be the predicted depth map and I be a reference intensity image. Denote  $\nabla_x$  and  $\nabla_y$  as gradient operators in the horizontal and vertical directions. The Edge Loss typically penalizes mismatches between the depth gradients  $\nabla_x \hat{D}$ ,  $\nabla_y \hat{D}$  and the image gradients  $\nabla_x I$ ,  $\nabla_y I$ . A common form is:

$$\mathcal{L}_{\text{edge}} = \sum_{i,j} \left( \left| \nabla_x \hat{d}_{i,j} - \nabla_x I_{i,j} \right| + \left| \nabla_y \hat{d}_{i,j} - \nabla_y I_{i,j} \right| \right), \tag{150}$$

where  $\hat{d}_{i,j}$  is the predicted depth at pixel (i,j), and  $I_{i,j}$  is the intensity in the reference image. By enforcing  $\nabla_x \hat{D} \approx \nabla_x I$  and  $\nabla_y \hat{D} \approx \nabla_y I$ , the network learns to place depth discontinuities only where genuine intensity edges exist.

## **Usage and Practical Aspects:**

- Edge-Aware Regularization: Unlike pure smoothness terms, Edge Loss prioritizes preserving structural detail, preventing over-smoothing around object boundaries.
- **Sensitivity to Noise**: Since it directly uses image gradients, noise or artifacts in *I* may produce spurious edges. Real implementations often incorporate edge-aware weighting or denoising to reduce false gradients.
- Multi-Scale or Combined Approach: Some approaches apply Edge Loss across multiple scales or combine it with point-wise or photometric objectives (e.g., §8.1.4) to refine both global and boundary-level accuracy.

Edge Loss thus establishes a balance between enforcing smooth depth in homogeneous regions and preserving crisp depth transitions at real scene edges, leading to more perceptually convincing and structurally consistent depth predictions in MDE.

## 8.1.10 Minimum Reprojection Loss

Minimum Reprojection Loss [211] is a self-supervised strategy to handle occlusions and dynamic objects in multi-view (or stereo) data. Instead of averaging photometric errors across all source views, it selects the minimal reprojection error per pixel, thus ignoring view-specific occlusions or erroneous projections.

**Motivation:** When warping multiple source images  $\{I_t^k\}$  into the reference view  $I_r$  using a predicted depth  $\hat{D}$  and known camera transformations, some pixel reprojections may be invalid due to occlusion or dynamic scene elements. By taking the minimum photometric error among all source images, the network automatically emphasizes valid, well-aligned viewpoints while disregarding noisy reprojected pixels.

**Definition:** For each pixel i, we compute a photometric error  $\mathcal{E}_{\text{photo}}$  w.r.t. each warped image  $\hat{I}_t^k$ . Commonly,  $\mathcal{E}_{\text{photo}}$  is a blend of SSIM and  $\ell_1$ -intensity differences [206], e.g.,

$$\mathcal{E}_{\text{photo}}\left(I_r^i, \hat{I}_t^{k,i}\right) = \alpha \frac{1 - \text{SSIM}\left(I_r^i, \hat{I}_t^{k,i}\right)}{2} + (1 - \alpha) \left|I_r^i - \hat{I}_t^{k,i}\right|,\tag{151}$$

where k indexes source images,  $\alpha \in [0, 1]$  balances the SSIM and absolute difference terms, and i indexes pixels. The *Minimum Reprojection Loss* is then:

$$\mathcal{L}_{\text{min-reproj}} = \frac{1}{N} \sum_{i=1}^{N} \min_{k} \mathcal{E}_{\text{photo}} \left( I_r^i, \hat{I}_t^{k,i} \right), \tag{152}$$

where N is the total number of valid pixels. Each pixel *chooses* the source view k yielding the least reprojection error.

## **Advantages and Caveats:**

- Occlusion Handling: If a pixel is occluded in one source view, it might be visible in another. Taking the minimum reduces penalizing erroneous warping of occluded regions.
- Dynamic Scenes: Minimizes artifacts from moving objects or inconsistent image patches across views.
- **Potential Oversmoothing**: In certain regions with multiple valid reprojections, the model can over-rely on the minimum, ignoring fine details. Automasking or advanced heuristics can mitigate this phenomenon [211].

Minimum Reprojection Loss is typically paired with smoothness (§8.1.5), edge-aware constraints (§8.1.9), or geometric consistency to build a robust self-supervised depth estimation framework. This synergy ensures both local coherence and the ability to disregard invalid or occluded pixels in multi-view training data, resulting in sharper, more accurate depth maps.

#### 8.1.11 Scale-and-shift-invariant Loss (SSI)

The Scale-and-Shift-Invariant Loss (SSI Loss) [212] was designed to address the challenges associated with depth (or disparity) prediction tasks, particularly where ground-truth depth values may vary across datasets due to differences in scale and shift. By making the loss function invariant to scale and shift ambiguities, it enables robust training and facilitates better generalization across diverse datasets with varying depth or disparity annotations.

Let  $\hat{d} \in \mathbb{R}^M$  denote the *predicted* disparities (or depths) and  $d \in \mathbb{R}^M$  represent the *ground-truth* disparities, where M is the number of valid pixels. The SSI Loss is defined as in (153):

$$L_{ssi}(\hat{d}, d) = \frac{1}{2M} \sum_{i=1}^{M} r(\hat{d}_i^* - d_i^*), \tag{153}$$

where  $\hat{d}^*$  and  $d^*$  are the *scale- and shift-aligned* predicted and ground-truth disparities, respectively, and  $r(\cdot)$  is a chosen loss function (e.g., mean squared error, absolute error, or trimmed absolute error).

**Scale-and-shift alignment.** The alignment of predicted and ground-truth disparities involves solving the least-squares problem:

$$s, t = \arg\min_{s, t} \sum_{i=1}^{M} \left( s \, \hat{d}_i + t - d_i \right)^2, \tag{154}$$

where s and t are the scale and shift parameters. The resulting aligned predictions and ground truths are given by

$$\hat{d}^* = s\,\hat{d} + t, \quad d^* = d. \tag{155}$$

Alternatively, robust estimators for  $\hat{d}$  may be used for scale and shift::

$$t(\hat{d}) = \text{median}(\hat{d}), \quad s(\hat{d}) = \frac{1}{M} \sum_{i=1}^{M} |\hat{d}_i - t(\hat{d})|.$$
 (156)

Advantages and trade-offs. This loss offers several benefits, including robustness to large variations across datasets, improved generalization, and numerical stability. By explicitly handling scale and shift ambiguities,  $L_{\rm ssi}$  can integrate heterogeneous data sources, each possibly having different disparity ranges. Numerical stability is also enhanced by working in disparity space (rather than log-space), avoiding issues that can arise from taking the logarithm of small or zero depth values.

A potential downside is the computational overhead of estimating  $\{s,t\}$  for each training sample, especially if robust estimation is used. While robust estimators mitigate the influence of outliers, extreme ground-truth errors can still negatively affect training. Additionally, the SSI Loss's effectiveness hinges upon accurate alignment; if scale/shift estimates are poor, performance may degrade.

Implementation details. In practice, the SSI Loss is implemented via the following steps:

- 1. Compute the optimal s and t (using least-squares (154) or robust approaches (156)).
- 2. Align the predicted and ground-truth disparities  $\hat{d}^*$ ,  $d^*$  using s and t as in (155).
- 3. Apply a robust loss function  $r(\cdot)$  (e.g., trimmed absolute error) to the aligned values:

$$L_{ssi}(\hat{d}, d) = \frac{1}{2M} \sum_{i=1}^{M} r(\hat{d}_i^* - d_i^*). \tag{157}$$

4. Optionally include regularization terms such as a gradient-matching loss (158):

$$L_{\text{reg}} = \frac{1}{M} \sum_{k=1}^{K} \sum_{i=1}^{M} |\nabla R_i^k|,$$
 (158)

where  $R = \hat{d}^* - d^*$  is the residual, and K is the number of scales (for multi-scale gradient consistency).

**Combining with gradient matching.** In [212], the authors add a gradient-matching regularization term to sharpen discontinuities and refine local details. The total loss function is given in (159):

$$L = L_{\rm ssi} + \alpha L_{\rm reg}, \tag{159}$$

where  $L_{\rm ssi}$  is the scale-and-shift-invariant term described above, and  $L_{\rm reg}$  is a multi-scale gradient-matching loss. Specifically, (160) details  $L_{\rm reg}$ :

$$L_{\text{reg}} = \frac{1}{M} \sum_{i=1}^{K} \sum_{i=1}^{M} \left( \left| \nabla_x R_i^k \right| + \left| \nabla_y R_i^k \right| \right), \tag{160}$$

where again  $R = \hat{d}^* - d^*$ , and  $\nabla_x$  and  $\nabla_y$  denote horizontal and vertical derivatives, respectively. By tuning  $\alpha$ , one can adjust the strength of the gradient-based penalty relative to the global alignment term, offering flexibility across different data domains or tasks.

In general, combining  $L_{ssi}$  with gradient matching has proven advantageous, yielding:

- ullet Complementary strengths:  $L_{
  m ssi}$  ensures global alignment while  $L_{
  m reg}$  sharpens object boundaries and local details
- Improved cross-dataset generalization: The combined loss often demonstrates superior zero-shot performance on unseen datasets.
- Controllability: By adjusting the hyperparameter  $\alpha$ , practitioners can balance global alignment against high-frequency refinement depending on task requirements.

## **8.2** Depth Estimation Metrics

Assessing the performance of depth estimation models requires the use of several metrics that reflect diverse dimensions of accuracy and quality. These metrics assess absolute and relative error rates, pixel-wise accuracy, and the alignment of predicted depth with the actual data. Listed below are the most commonly employed metrics for depth estimation; although some have been previously introduced, our focus here will be on their application in depth estimation.

The choice of metrics depends on the specific application and dataset. For example:

- AbsRel, RMSE, and  $\delta$  are commonly used for absolute depth datasets (e.g., KITTI [213], NYU [214, 215]).
- WHDR is specifically suited for datasets with ordinal depth annotations (e.g., DIW [216]).
- Scale-Invariant Error and RMSE(log) are ideal for tasks with varying scales or relative depth datasets.

Every single metric provides unique information, and combining multiple metrics leads to a comprehensive evaluation of the effectiveness of depth estimation.

Table 17 provides a summary of commonly used metrics in depth estimation, highlighting their uses, characteristics of the data, advantages, and disadvantages.

#### 8.2.1 Mean Absolute Relative Error (AbsRel)

Mean Absolute Relative Error (AbsRel) weights errors by the ground-truth depth, highlighting inaccuracies at smaller depths more strongly [203]. Let  $\{\hat{d}_i\}$  and  $\{d_i\}$  be predicted and true depths for M valid pixels:

AbsRel = 
$$\frac{1}{M} \sum_{i=1}^{M} \frac{|\hat{d}_i - d_i|}{d_i}$$
. (161)

## **Interpretation:**

- **Relative Emphasis:** Large errors at shallow depths can heavily impact the metric, reflecting the significance of small-scale geometry in many applications.
- Limitations: If any  $d_i \approx 0$  or the dataset spans wide depth ranges, AbsRel can be unstable or less intuitive, motivating complementary metrics like RMSE.

## 8.2.2 Root Mean Squared Error (RMSE)

*RMSE* [203] quantifies the overall deviation between predicted  $\hat{d}_i$  and ground-truth  $d_i$ , placing stronger penalization on large residuals:

RMSE = 
$$\sqrt{\frac{1}{M} \sum_{i=1}^{M} (\hat{d}_i - d_i)^2}$$
. (162)

## **Usage:**

- Global Error Sensitivity: RMSE magnifies large outliers, beneficial in catching significant depth mispredictions but can overshadow small, systematic errors.
- **Interpretation:** RMSE is dimensionally in units of depth, making it straightforward but sometimes less robust when scale ambiguities exist.

Table 17: Guidelines for selecting depth estimation metrics based on usage, data characteristics, advantages and limitations.

Function	Usage	Data Characteristics	Advantages	Limitations
AbsRel	Evaluates average relative deviation between predicted and ground truth	Emphasizes errors in smaller depths, useful for moderate ranges	Highlights relative dis- crepancies good for tasks needing fine-grained accuracy	Can undervalue errors at large depths sensitive to scale vari- ations
RMSE	Measures overall devi- ation, penalizes large errors heavily	Suitable for datasets without extremely high depth ranges	Common and intu- itive, highlights outlier errors	Large errors can dominate, not scale-invariant
RMSE(log)	Focuses on proportional differences in log space	Effective across wide depth scales, mitigates scale mis- matches	Balances errors for both small and large depths	Sensitive near zero- depth values, can be less intuitive
Threshold Accuracy $\delta$	Fraction of pixels within a multiplicative error bound	Useful for quick checks in diverse depth ranges	Simple to interpret, highlights near- accurate predictions	Discontinuous metric, choice of threshold is arbitrary
Mean Log10	Assesses log-scale deviations, robust for varied depth distributions	Suitable for indoor- outdoor mixes, handles large-scale di- versity	Scale invariant, balances small and large depth errors	Amplifies near-zero errors, can be harder to interpret than RMSE
Percentage of High Error	Identifies proportion of pixels exceeding a chosen relative error	Useful in safety- critical settings, focuses on large deviations	Emphasizes worst- case performance, reveals catastrophic failures	Threshold selection is subjective, ignores moderate errors
WHDR	Evaluates ordinal consistency with human-labeled pairs	Applicable to sparse or relative data, no absolute ground truth	Handles subjective depth ordering, good for datasets with pairwise labels	Cannot assess absolute depth, depends on annotation quality
Scale- Invariant	Focuses on structural accuracy regardless of global scale	Helpful when true scale is unknown or for cross-dataset tasks	Robust to scale mis- match, leverages log-space corrections	Less useful when exact scale is crucial, can mask global errors

# 8.2.3 Logarithmic RMSE (RMSE(log))

RMSE(log) [203] measures errors in log space, emphasizing relative over absolute deviations. Formally,

$$RMSE(log) = \sqrt{\frac{1}{M} \sum_{i=1}^{M} \left[ log(\hat{d}_i) - log(d_i) \right]^2}.$$
(163)

## Advantages:

- Large Depth Range Handling: Suitable for datasets mixing near-field and far-field distances (e.g., indoor vs. outdoor).
- Limitations: Depth values near zero cause numerical issues, and the log transform can complicate direct interpretability compared to linear measures.

## 8.2.4 Threshold Accuracy ( $\delta$ )

Threshold Accuracy [203] measures the fraction of pixels whose predicted depth falls within a factor of the ground truth:

$$\delta = \max\left(\frac{\hat{d}_i}{d_i}, \frac{d_i}{\hat{d}_i}\right). \tag{164}$$

One reports the percentage of pixels satisfying  $\delta < \delta_{\rm threshold}$ . Common thresholds are  $\delta_1 = 1.25$ ,  $\delta_2 = 1.25^2$ ,  $\delta_3 = 1.25^3$ .

#### **Interpretation:**

- Relative Criterion: Captures how many pixels are relatively accurate within a multiplicative bound.
- Multiple Thresholds: δ<sub>1</sub>, δ<sub>2</sub>, δ<sub>3</sub> reflect progressively looser accuracy constraints, giving a multi-level view of performance.

## 8.2.5 Mean Log10 Error

Mean Log10 Error [203] measures the absolute difference in base-10 logarithmic space:

Mean Log10 = 
$$\frac{1}{M} \sum_{i=1}^{M} \left| \log_{10}(\hat{d}_i) - \log_{10}(d_i) \right|$$
. (165)

## **Advantages and Limitations:**

- Logarithmic Emphasis: Similar to RMSE(log), it focuses on relative discrepancies, beneficial in multi-scale or cross-dataset applications.
- Interpretation Challenges: Depth near zero or extremely large can skew the metric, and base-10 logs may be less intuitive than linear errors for some tasks.

#### 8.2.6 Percentage of Pixels with High Error

This metric [213] evaluates the fraction of pixels whose relative error exceeds a threshold  $\tau$ . Formally,

Percent Error = 
$$\frac{1}{M} \sum_{i=1}^{M} \mathbb{M} \left( \frac{|\hat{d}_i - d_i|}{d_i} > \tau \right), \tag{166}$$

where  $\mathbb{F}(\cdot)$  is the indicator function. Threshold  $\tau$  might be 0.1, 0.2, or 0.5, depending on desired strictness.

## **Interpretation:**

- Focus on Worst-Case Pixels: Identifies how often the model severely mispredicts depth, which can be critical for safety-related or robotics tasks.
- Threshold Choice: Lower  $\tau$  yields a stricter standard, highlighting small permissible errors, whereas higher  $\tau$  is more lenient but less informative about mild deviations.

## 8.2.7 Weighted Human Disagreement Rate (WHDR)

Weighted Human Disagreement Rate (WHDR) [216] evaluates how well predicted depths match human-labeled ordinal relations (closer, same, or farther) in pairs of image pixels. It is especially suitable when absolute ground-truth depths are unavailable, and only pairwise judgments exist:

WHDR = 
$$\frac{\sum_{(i,j)} w_{ij} \mathcal{L}\left(l_{ij} \left(\hat{d}_i - \hat{d}_j\right) < 0\right)}{\sum_{(i,j)} w_{ij}},$$
(167)

where  $l_{ij} \in \{-1, 0, 1\}$  is the ordinal relationship for pair (i, j) (e.g.,  $l_{ij} = -1$  if i is nearer than j), and  $w_{ij}$  weights that pair's importance.

#### **Usage:**

- Ordinal-Only Datasets: WHDR is crucial in large-scale data lacking metric depths but providing relative ranks or "closer/farther" judgments.
- Limitations: Metric scale is not assessed, and it depends on the quality of human annotations. Also, ties or equal distances can complicate interpretation.

#### 8.2.8 Scale-Invariant Error (Metric Form)

Scale-Invariant Error [203] can also serve as a metric, complementing or replacing standard  $\ell_2$  measures by focusing on relative depth consistency. For predicted  $\{\hat{d}_i\}$  and true  $\{d_i\}$ , the scale-invariant metric is:

Scale-Invariant Error = 
$$\frac{1}{M} \sum_{i=1}^{M} \left[ \log(\hat{d}_i) - \log(d_i) + \frac{1}{M} \sum_{i=1}^{M} \left( \log(d_i) - \log(\hat{d}_i) \right) \right]^2. \tag{168}$$

## **Interpretation:**

- **Relative Depth Emphasis:** Penalizes shape mismatches rather than absolute scale. Especially apt for cross-dataset or relative depth tasks.
- **Drawback:** Omits large global scale errors if uniform scaling is off, so some applications needing exact metric consistency prefer absolute-based metrics.

Conclusion on MDE Metrics: Each metric captures different aspects: AbsRel or RMSE stress absolute residuals, RMSE(log) and MeanLog10 focus on multiplicative or log-space errors,  $\delta$  thresholds reveal how many pixels lie within a given factor, WHDR handles purely ordinal data, and Scale-InvariantError emphasizes relative shape. In practice, researchers often report multiple metrics to provide a comprehensive assessment of the depth estimation performance of a model.

# 9 Image Generation

Image generation in deep learning involves using artificial neural networks to generate new images. This task has progressed significantly with models such as Variational Autoencoders (VAEs) [217, 218, 219, 220], Generative Adversarial Networks (GANs) [221, 222, 223, 224, 225], Normalized Flow models (NFs) [226, 227, 228, 229, 230], Energy-Based Models (EBMs) [231, 232, 233, 234, 235, 236], and Diffusion Models [237, 238, 239, 240, 241]. These models can generate high-quality images that can be used in various applications such as image super-resolution [242, 243, 244, 245], denoising [246, 247, 248], inpainting [249, 250, 251, 252], and style transfer [253, 254, 255, 256].

Variational Autoencoders (VAEs) are generative models that use deep learning techniques to create new data and learn latent representations of the input data. They consist of an encoder and a decoder. The encoder compresses input data into a lower-dimensional latent space, while the decoder reconstructs the original data from points in the latent space. The process is trained to minimize the difference between original and reconstructed data and ensure the latent space approximates a standard Gaussian distribution. VAEs can generate new data by feeding the decoder points sampled from the latent space.

Generate Adversarial Networks (GANs) involve two neural networks, a Generator, and a Discriminator, playing a game against each other. The Generator tries to create data similar to the training data, while the Discriminator tries to distinguish between the real and generated data. Through this process, both networks improve: the Generator learns to produce increasingly realistic data, while the Discriminator becomes better at distinguishing between real and artificial data. This adversarial process continues until an equilibrium is reached, at which point the Generator is producing realistic data and the Discriminator is, at best, randomly guessing whether the data is real or generated. This equilibrium is conceptually referred to as a Nash equilibrium in game theory [257].

*Normalizing Flows* use invertible transformations to generate diverse outputs and provide an exact and tractable likelihood for a given sample, enabling efficient sampling and density estimation. However, they can be computationally intensive to train.

*Energy-Based Models (EBMs)* learn a scalar energy function to distinguish real data points from unlikely ones. A neural network often parameterizes this function and learns from the data. Sampling in EBMs is typically done via Markov

Chain Monte Carlo (MCMC) methods. EBMs can represent a wide variety of data distributions but can be challenging to train due to the intractability of the partition function and the computational expense of MCMC sampling.

*Diffusion models* use a random process to transform simple data distribution, like Gaussian noise, into the desired target distribution. This is controlled by a trained neural network, allowing the generation of high-quality data, like images, through a smooth transition from noise to the desired data.

The following sections will review the common lost functions and performance metrics used for image generation.

### 9.1 Image Generation Loss Functions

The loss function in a VAE consists of the reconstruction loss and the Kullback-Leibler Divergence Loss (KL). The reconstruction loss measures how well the decoded data matches the original input data. The KL divergence measures how much the learned distribution in the latent space deviates from a target distribution, usually a standard normal distribution. KL-divergence is used as a regularization term to ensure that the distributions produced by the encoder remain close to a unit Gaussian, penalizing the model if the learned distributions depart from it.

The most common loss function used in GANs is the adversarial loss, which is the sum of the cross-entropy loss between the generator's predictions and the real or fake labels. Later, WGAN [258] applied the Wasserstein distance as an alternative to training GANs to improve stability and avoid mode collapse that occurs when the generator network stops learning the underlying data distribution and begins to produce a limited variety of outputs, rather than a diverse range of outputs that accurately represent the true data distribution.

Normalizing Flows are typically trained using maximum likelihood estimation. Given a dataset, the aim is to maximize the log-likelihood of the data under the model by minimizing the negative log-likelihood of the data.

During training, Energy-based models (EBMs) minimize a loss function that encourages the energy function to assign lower energy values to data points from the training data and higher energy values to other points. Different types of EBMs use different loss functions, such as Contrastive Divergence (CD)[259], Maximum Likelihood Estimation (MLE)[260], and Noise-Contrastive Estimation (NCE) [261].

Diffusion models use a denoising loss function based on the Mean Absolute Error (MAE) or the Mean Squared Error (MSE) between the original and reconstructed data.

Table 18 offers an in-depth examination of popular loss functions in image generation, highlighting their applications, data attributes, benefits, and drawbacks.

In the following sections, we describe in detail each of these losses.

## 9.1.1 Reconstruction Loss

Reconstruction loss measures how faithfully a model reproduces the input image (or data) in tasks such as Variational Autoencoders (VAEs), image restoration, and generative image synthesis. Its primary objective is to penalize discrepancies between the original input  $\mathbf{x}$  and the reconstructed output  $\hat{\mathbf{x}}$ , driving the model to learn representations that preserve key image features.

**Mean Squared Error (MSE):** For continuous-valued image data, a common form of reconstruction loss is *Mean Squared Error* (MSE). Let  $\{\mathbf{x}_i\}_{i=1}^N$  be the original images and  $\{\hat{\mathbf{x}}_i\}_{i=1}^N$  be their reconstructions. The MSE loss is:

$$\mathcal{L}_{\text{recon}} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2^2,$$
(169)

where  $\|\cdot\|_2$  denotes the Euclidean norm. MSE directly measures pixel-level squared differences, making it suitable for continuous image intensities. However, it may over-penalize small local shifts (e.g., slight misalignments in edges or textures).

**Binary Cross-Entropy (BCE):** For binary or probabilistic image outputs, *Binary Cross-Entropy* (BCE) is often preferred. Let  $\mathbf{y}_i \in \{0,1\}^D$  be a binary ground-truth vector and  $\hat{\mathbf{y}}_i \in [0,1]^D$  the model's predicted probabilities. Then the BCE loss is:

Table 18: Guidelines for selecting a loss function for image generation, based on usage, data characteristics, advantages and limitations.

Function	Usage	Data Characteristics	Advantages	Limitations
MSE	Autoencoders Continuous image re- construction Image restoration	Real-valued intensi- ties Typically unimodal	Stable training Easy to implement	Can produce overly smooth outputs Sensitive to outliers
BCE	Binary image generation Reconstruction tasks with binary data	Binary or probabilistic pixel outputs Pixel-wise classifica- tion	Direct probability interpretation Common in classification-based setups	May saturate gradients Slow convergence if predictions are poor
KL Divergence	Variational autoen- coders Distribution matching in latent space	Known or approximated target distribution Typically continuous latent variables	Aligns predicted and target distributions Popular in generative modeling	Highly sensitive to zero probabilities Can be numerically unstable
Perceptual	Super-resolution, style transfer, advanced image gener- ation tasks	Real-valued or color images Requires pretrained feature extractor (e.g., VGG)	Preserves semantic details Less blurring than pixel-based losses Aligned with human perception	Depends on pretrained models Domain mismatch can degrade features Ignores exact pixel- level fidelity
Adversarial Loss	GAN-based image generation Domain adaptation for realism	Unlabeled data Real vs. synthetic dis- tributions	Generates high- fidelity images Encourages realistic outputs	Prone to mode collapse Training can be unstable
Wasserstein	WGANs Stable image genera- tion	Continuous or discrete data Flexible for various domains	More stable training Less mode collapse than standard GANs	Requires Lipschitz constraints Involves extra gradi- ent penalty steps
Negative Log- likelihood (NFs)	Normalizing flows Exact likelihood- based generation	Invertible transforma- tions Tractable Jacobian de- terminant	Provides exact density estimates Flexible architectures	High computational cost Memory-intensive for large images
Contrastive Diver- gence	Energy-based image models Restricted Boltzmann Machines	High-dimensional image data Continuous or discrete inputs	Approximate training of intractable likeli- hood Generally more effi- cient than full MCMC	Biased gradient esti- mates May need multiple sampling steps

$$\mathcal{L}_{BCE} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{D} \left[ y_{i,j} \log(\hat{y}_{i,j}) + (1 - y_{i,j}) \log(1 - \hat{y}_{i,j}) \right], \tag{170}$$

where D is the dimensionality (e.g., number of pixels). BCE measures the divergence between the true binary distribution and the predicted Bernoulli outputs, making it a natural choice for tasks involving binarized images or probabilistic pixel maps.

## **Usage in Image Generation:**

- 1. **Variational Autoencoders (VAEs):** Reconstruction loss is combined with a Kullback–Leibler (KL) divergence term to ensure the latent code follows a chosen prior distribution. Minimizing MSE or BCE encourages faithful image reconstructions while the KL term regularizes the latent space.
- 2. **Image Restoration:** In denoising, super-resolution, and inpainting, reconstruction loss enforces that the model output remains close to the pristine target image, recovering lost detail or removing artifacts.

- 3. **Generative Models:** Models like  $\beta$ -VAEs or certain GAN variations use reconstruction losses to keep outputs close to real data distributions. This includes reconstructing inputs or partially corrupted images.
- 4. **Anomaly Detection:** Reconstruction loss can highlight anomalous patterns if a model trained on "normal" images produces large residuals for out-of-distribution samples.

#### **Limitations and Extensions:**

- Overly Smooth Outputs: MSE can induce blurring or loss of sharp details. Perceptual losses or adversarial terms (§9.1.4) may be added to enhance realism.
- Limited Structural Awareness: Pixel-wise metrics do not explicitly account for high-level image features or semantics. SSIM-based terms (§8.1.3) or perceptual losses (§9.1.3) can improve structural fidelity.
- Adaptive Weighting: Some tasks require balancing reconstruction fidelity with constraints on latent space size, style consistency, or domain adaptation. Hence,  $\lambda_{\text{recon}}$  may be tuned in multi-objective loss setups.

In general, reconstruction loss remains an essential element of image generation and restoration tasks, providing a direct measure of fidelity between the original and the reconstructed images.

## 9.1.2 Kullback-Leibler Divergence Loss

Kullback-Leibler (KL) Divergence [262] measures how one probability distribution q diverges from a target distribution p. In a supervised setting,  $p(\mathbf{x})$  often represents the true or "target" probabilities over classes, while  $q(\mathbf{x})$  is the model's predicted distribution. Let  $x_1, x_2, \ldots, x_n$  be the outcomes or class labels. The KL divergence is defined as:

$$KL(p \parallel q) = \sum_{i=1}^{n} p(x_i) \log \left(\frac{p(x_i)}{q(x_i)}\right). \tag{171}$$

#### **Interpretation:**

- **Distribution Matching:** KL divergence quantifies how many extra bits are needed to encode samples from p using q. Minimizing KL enforces that  $q(x_i) \approx p(x_i)$  for all i.
- Asymmetric Measure: KL(p||q) is not the same as KL(q||p). The choice of which distribution is in the numerator vs. denominator can greatly affect optimization behavior, especially near zero probabilities.

## **Usage in Generative Models:**

- Variational Autoencoders (VAEs): VAEs minimize  $KL(q_{\phi}(\mathbf{z}|\mathbf{x}) \parallel p_{\theta}(\mathbf{z}|\mathbf{x}))$  or  $KL(q_{\phi}(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}))$  in the ELBO objective, guiding the learned latent distribution  $q_{\phi}(\mathbf{z}|\mathbf{x})$  to match a prior  $p(\mathbf{z})$ .
- Reinforcement Learning: KL divergence is used to keep an agent's updated policy close to a reference or previous policy (e.g., TRPO, PPO). Minimizing  $KL(\pi_{new} || \pi_{old})$  or vice versa ensures stable policy updates.

#### Sensitivity to Zero Probabilities:

- Numerical Instability: When  $p(x_i) = 0$  but  $q(x_i) > 0$  (or vice versa), the KL term can be undefined or infinite. This demands care in initial probability assignments or smoothing.
- **Practical Remedy:** A small constant  $\varepsilon$  (e.g.,  $10^{-7}$ ) is often added to both p and q to avoid taking the log of zero.

## **Comparison with Other Divergences:**

- Forward vs. Reverse KL: Minimizing  $\mathrm{KL}(p\|q)$  differs from  $\mathrm{KL}(q\|p)$ . For instance,  $\mathrm{KL}(p\|q)$  is more sensitive if  $p(x_i)$  is significantly non-zero but  $q(x_i) \approx 0$ .
- Alternative Metrics: In some scenarios, Jensen–Shannon divergence (symmetric version) or Wasserstein distance (§9.1.5) can be more stable or yield different optimization trajectories, especially in GAN training.

By matching the model's predicted distribution  $q(x_i)$  to the true (or target) distribution  $p(x_i)$ , KL divergence loss fosters distribution-level correctness, which is especially critical in generative modeling contexts (e.g., VAEs) and policy optimization in reinforcement learning. However, one must handle zero-probability events carefully to avoid numerical instability or poor gradient signals.

## 9.1.3 Perceptual Loss

Perceptual Loss, sometimes referred to as feature reconstruction loss or VGG-based loss, measures the discrepancy between high-level feature representations of the original and generated images rather than their raw pixel-level differences [263, 264, 265]. By comparing deep network activations (e.g., from a pretrained classification network), Perceptual Loss aims to ensure that the reconstructed image  $\hat{\mathbf{x}}$  not only matches  $\mathbf{x}$  in low-level details but also captures semantically important features and textures.

**Motivation:** Pixel-wise metrics such as MSE (§9.1.1) often produce blurred or less realistic outputs, as they overemphasize exact local correspondences. Perceptual Loss mitigates this issue by forcing the generated image to have similar *feature activations* in a deep CNN's intermediate layers, leading to sharper and more visually coherent results.

**Formulation:** Let  $\phi(\cdot)$  be a pretrained network (often a classification CNN such as VGG-16 [265]), and  $\phi_{\ell}(\mathbf{x}) \in \mathbb{R}^{C \times H \times W}$  be its  $\ell$ -th layer activations when input  $\mathbf{x}$  is forwarded. Then the *Perceptual Loss* between an original image  $\mathbf{x}$  and a reconstructed image  $\hat{\mathbf{x}}$  is computed as the  $\ell_2$ -distance between their feature activations:

$$\mathcal{L}_{perc}(\mathbf{x}, \hat{\mathbf{x}}) = \sum_{\ell \in \mathcal{L}} \frac{1}{C_{\ell} H_{\ell} W_{\ell}} \| \phi_{\ell}(\mathbf{x}) - \phi_{\ell}(\hat{\mathbf{x}}) \|_{2}^{2},$$
(172)

where  $\mathcal{L}$  is the set of chosen layers, and  $C_{\ell}$ ,  $H_{\ell}$ ,  $W_{\ell}$  are the channel, height, and width dimensions of layer  $\ell$ . The normalization factor  $\frac{1}{C_{\ell}H_{\ell}W_{\ell}}$  ensures consistent scale across layers.

## **Usage in Image Generation:**

- Super-Resolution and Deblurring: Perceptual Loss helps recover sharp edges and fine structures that pixel-wise errors tend to oversmooth [266].
- **Style Transfer**: By matching feature activations of style and content images, models achieve visually pleasing stylization [253, 263].
- GANs and VAEs: Combining Perceptual Loss with adversarial or reconstruction objectives yields images with globally and semantically coherent details [267, 268].

## **Advantages and Limitations:**

- **Preserving High-Level Semantics**: Focuses on how the generated image appears to a deep network, correlating better with human perception of sharpness and structure.
- **Decreased Local Fidelity**: It does not guarantee pixel-perfect alignment and might overlook small color mismatches if they do not significantly alter high-level features.
- Dependence on Pretrained Networks: Typically,  $\phi$  is a CNN trained on large image datasets (e.g., ImageNet), introducing a domain bias. If the target domain differs from the CNN's training distribution, perceived "feature relevance" might be suboptimal.

## **Extensions:**

- Layer Selection: Perceptual Loss can combine activations from multiple layers to capture both coarse and fine-level attributes.
- Style Loss (Gram Matrices): In style transfer or artistic generation, one may also penalize discrepancies in correlation statistics (Gram matrices) of layer features, capturing "texture" alignment [253].
- Multiscale Variants: Computing Perceptual Loss at different spatial resolutions can enhance stability and capture multi-level semantics.

Perceptual Loss thus balances high-level structural resemblance with a more "human-like" notion of image quality, making it a staple in advanced image generation, super-resolution, and style transfer pipelines. While it lacks pixel-accuracy guarantees, its synergy with low-level reconstruction terms or adversarial objectives often yields outputs that are both visually appealing and domain-faithful.

#### 9.1.4 Adversarial Loss

Adversarial Loss is foundational to Generative Adversarial Networks (GANs), introduced by Goodfellow  $et\,al.\,$  [221]. It arises from a minimax game between two networks: a generator G (which maps noise vectors to synthesized data) and a discriminator D (which classifies inputs as real or generated). By training G to produce samples indistinguishable from real data, and simultaneously training D to distinguish real from generated, this adversarial setup propels the generator's outputs toward realism.

**Basic Formulation:** Assume  $\mathbf{z} \sim p_z(\mathbf{z})$  is noise drawn from a latent distribution, and  $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$  is a real-data sample from the target distribution. The *Adversarial Loss* (in its non-saturated version) can be stated as:

$$\mathcal{L}_{\text{adv}}(G, D) = -\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[ \log D(\mathbf{x}) \right] - \mathbb{E}_{\mathbf{z} \sim p_z} \left[ \log (1 - D(G(\mathbf{z}))) \right].$$
(173)

Here,

- $D(\mathbf{x})$  outputs the probability that  $\mathbf{x}$  is a real sample.
- $G(\mathbf{z})$  generates a synthetic sample from noise  $\mathbf{z}$ .

GAN training is often viewed as a two-player minimax game. The discriminator D maximizes:

$$\max_{D} \Big\{ \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z} [\log (1 - D(G(\mathbf{z})))] \Big\}.$$
 (174)

Meanwhile, the generator G minimizes:

$$\min_{G} \mathbb{E}_{\mathbf{z} \sim p_z} \left[ \log \left( 1 - D(G(\mathbf{z})) \right) \right]. \tag{175}$$

Alternatively, one may train the generator by  $\max_G \mathbb{E}_{\mathbf{z} \sim p_z}[\log D(G(\mathbf{z}))]$  to help mitigate gradient saturation issues when D is strong.

The training procedure alternates between:

- 1. **Discriminator Update:** Holding G fixed, optimize D to distinguish real x vs. fake  $G(\mathbf{z})$ .
- 2. **Generator Update:** Holding D fixed, optimize G to produce samples that fool D.

Over time, G improves at creating realistic samples, while D refines its discrimination boundary. Under ideal conditions, this converges to a *Nash equilibrium* where G's outputs are indistinguishable from real data.

## **Advantages and Challenges:**

- **High-Fidelity Generation:** GANs can learn complex, high-dimensional data distributions, yielding photorealistic images in tasks like image synthesis, inpainting, or super-resolution.
- Mode Collapse: The generator may collapse to a limited subset of modes, producing less diverse outputs. Techniques like Wasserstein distance (§9.1.5) or multi-objective strategies mitigate this.
- Training Instability: Adversarial objectives can be sensitive to hyperparameters and require careful balancing of G and D capacities. Convergence is not guaranteed but can be stabilized via gradient penalties or spectral normalization.

#### **Extensions:**

- Wasserstein GANs (WGANs): Replace KL/JS divergences with Earth Mover's distance for more stable training (§9.1.5).
- Conditional GANs: Incorporate class labels or other conditions into G and D, enabling class-specific image synthesis or domain adaptation.
- CycleGAN, StyleGAN, etc.: Specialized architectures leverage adversarial loss for unpaired image-to-image translation, style generation, and more.

Adversarial Loss underpins GAN-based approaches by framing generation as a competition between generating realistic samples and detecting fakes. Despite challenges in training stability, it remains one of the most powerful tools in image generation, pushing the state of the art in visual realism and diversity.

## 9.1.5 Wasserstein Loss

Wasserstein Loss, introduced by Arjovsky et al. [258], provides an alternative objective to standard adversarial loss in Generative Adversarial Networks (GANs). Rather than directly matching distributions through JS or KL divergences, Wasserstein Loss employs the Earth Mover's Distance (EMD), offering smoother, more stable gradients when learning to transform a generator's distribution  $p_q$  toward the real data distribution  $p_d$ .

**Definition:** Let  $p_d(\mathbf{x})$  be the distribution of real data and  $p_g(\mathbf{x})$  be the distribution of generated data. The *Wasserstein-1* distance (W) between these two distributions can be expressed as an optimal transport problem:

$$W(p_d, p_g) = \inf_{\gamma \in \Gamma(p_d, p_g)} \int_{\mathbf{x}, \mathbf{y}} ||\mathbf{x} - \mathbf{y}|| \, d\gamma(\mathbf{x}, \mathbf{y}), \tag{176}$$

where

- $\Gamma(p_d, p_g)$  is the set of all joint distributions  $\gamma(\mathbf{x}, \mathbf{y})$  whose marginals are  $p_d$  and  $p_g$ .
- ||·|| typically denotes the Euclidean distance.
- $\gamma(\mathbf{x}, \mathbf{y})$  indicates how "mass" is transported from  $\mathbf{y}$  in distribution  $p_q$  to  $\mathbf{x}$  in distribution  $p_d$ .

The integral represents the minimum "cost" to morph one distribution into the other, where cost is the distance  $\|\mathbf{x} - \mathbf{y}\|$  weighted by  $\gamma(\mathbf{x}, \mathbf{y})$ .

In practice, one implements  $W(p_d,p_g)$  in GAN training by defining a critic (akin to a discriminator) that approximates the Wasserstein distance through its Lipschitz-constrained outputs [258]. Formally, for a function  $f_{\theta}$  constrained to be 1-Lipschitz,

$$W(p_d, p_g) \approx \max_{\theta, \|f_{\theta}\| < 1} \left( \mathbb{E}_{\mathbf{x} \sim p_d}[f_{\theta}(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim p_z}[f_{\theta}(G(\mathbf{z}))] \right). \tag{177}$$

Hence the Wasserstein Loss for the generator and critic can be written similarly to:

$$\mathcal{L}_{\text{critic}} = \mathbb{E}_{\mathbf{x} \sim p_d} [f_{\theta}(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim p_z} [f_{\theta}(G(\mathbf{z}))], \tag{178}$$

$$\mathcal{L}_{\text{gen}} = -\mathbb{E}_{\mathbf{z} \sim p_z} [f_{\theta}(G(\mathbf{z}))]. \tag{179}$$

# **Advantages Over Classic GAN Losses:**

- Smooth Gradient Signals: Wasserstein distance yields non-saturating gradients even when distributions have disjoint supports, reducing training instability.
- Mitigates Mode Collapse: The generator is penalized proportionally to how far  $p_g$  is from covering  $p_d$ , rather than simply whether samples look real or fake.
- **Stable Convergence:** Empirically, WGAN training can exhibit reduced mode collapse and more robust progression, though hyperparameters and Lipschitz constraints remain critical.

**Lipschitz Constraint and WGAN-GP:** To ensure  $||f_{\theta}|| \leq 1$ , Arjovsky *et al.* proposed weight clipping, but Gulrajani *et al.* [269] introduced a *gradient penalty* (GP) approach, often referred to as WGAN-GP. This enforces the 1-Lipschitz condition by penalizing deviations from  $||\nabla f_{\theta}|| = 1$ , improving training stability and removing the discrete clamp on weights.

## **Limitations:**

- Lipschitz Enforcement Overhead: Gradient penalties or clipping can add complexity and slow training.
- **Hyperparameter Sensitivity:** WGAN still requires careful tuning (penalty coefficients, learning rates) to fully reap its benefits of stable training.

## 9.1.6 Negative Log-Likelihood in Normalizing Flows

Normalizing Flows form a class of generative models where a bijective and differentiable transformation  $f_{\theta}$  maps data  $\mathbf{x}$  from a complex distribution  $p_{\theta}(\mathbf{x})$  to a simpler, tractable base distribution  $p_z(\mathbf{z})$  (often an isotropic Gaussian). Let  $\mathbf{z} = f_{\theta}(\mathbf{x})$  denote the forward transformation, implying  $\mathbf{x} = f_{\theta}^{-1}(\mathbf{z})$ . Because  $f_{\theta}$  is invertible, the model can compute an *exact* likelihood for data  $\mathbf{x}$  via the change of variables formula.

**Log-likelihood** Computation: For a single data point x, the log-likelihood under the flow model  $p_{\theta}(\mathbf{x})$  is given by

$$\log p_{\theta}(\mathbf{x}) = \log p_z(f_{\theta}(\mathbf{x})) + \log \left| \det \left( \frac{\partial f_{\theta}(\mathbf{x})}{\partial \mathbf{x}} \right) \right|, \tag{180}$$

where

- $\mathbf{z} = f_{\theta}(\mathbf{x}) \in \mathbb{R}^d$  is the latent representation, distributed according to a simple prior  $p_z(\mathbf{z})$  (e.g.,  $\mathcal{N}(\mathbf{0}, \mathbf{I})$ ).
- $\det\left(\frac{\partial f_{\theta}(\mathbf{x})}{\partial \mathbf{x}}\right)$  is the Jacobian determinant of  $f_{\theta}$  evaluated at  $\mathbf{x}$ . This term accounts for the volume change induced by the mapping.

**Training via Negative Log-likelihood:** To learn the parameters  $\theta$  of the flow, one maximizes the likelihood of a dataset  $\{\mathbf{x}_i\}_{i=1}^N$  or equivalently minimizes its negative log-likelihood. Thus, the loss function is:

$$\mathcal{L}(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \log p_{\theta}(\mathbf{x}_{i})$$

$$= -\frac{1}{N} \sum_{i=1}^{N} \left[ \log p_{z} \left( f_{\theta}(\mathbf{x}_{i}) \right) + \log \left| \det \frac{\partial f_{\theta}(\mathbf{x}_{i})}{\partial \mathbf{x}_{i}} \right| \right].$$
(181)

Minimizing  $\mathcal{L}(\theta)$  via stochastic gradient methods encourages  $f_{\theta}$  to transform real samples  $\mathbf{x}_{i}$  into plausible latent codes  $\mathbf{z}$  that align with the base distribution, while accurately capturing the data distribution's structure.

A key challenge is computing and differentiating through the Jacobian determinant  $\det(\partial f_{\theta}/\partial \mathbf{x})$ . To keep this tractable, Normalizing Flow architectures, such as RealNVP [226] and Glow [227], design invertible layers (e.g., *affine coupling*, *invertible*  $1 \times 1$  *convolutions*) that yield either a block triangular Jacobian or otherwise simplify the determinant calculation. Thus, the log-determinant term remains efficiently computable and backprop-friendly.

## **Advantages and Limitations:**

- Exact Likelihood: Unlike typical GANs (§9.1.5) or VAEs that approximate likelihood, Normalizing Flows give an exact  $\log p_{\theta}(\mathbf{x})$ .
- Flexible Distributions: Coupling-based transformations allow for complex multimodal distributions if stacked sufficiently.
- Computational Cost: The necessity of computing and differentiating  $\log |\det(\cdot)|$  can be memory-intensive. Flow-based models are often larger and slower to train than latent-variable methods with approximate posteriors (e.g., VAEs).

Hence, *Negative Log-Likelihood* in Normalizing Flows integrates invertible transformations with tractable determinant structures, offering exact density estimation and generative sampling. Despite heavier computational demands, this approach is highly appealing for tasks demanding precise likelihood evaluations.

## 9.1.7 Contrastive Divergence

Contrastive Divergence (CD) [270] is a method for approximately maximizing the log-likelihood of an Energy-Based Model (EBM), such as a Restricted Boltzmann Machine (RBM), when direct computation of the gradient is intractable. Rather than sampling from the model distribution until equilibrium, CD relies on short Markov Chain Monte Carlo (MCMC) runs, starting from real data samples, to estimate the negative phase gradient.

**Energy-Based Models and Log-likelihood:** Let  $\{x_i\}_{i=1}^N$  be a dataset of N points. An EBM parameterized by  $\theta$  defines an energy function  $E(x;\theta)$ , so that the model's density is

$$p(x;\theta) = \frac{\exp(-E(x;\theta))}{Z(\theta)},$$
(182)

where

$$Z(\theta) = \int \exp(-E(x;\theta)) dx$$
 (183)

is the partition function. The log-likelihood of the data is

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} \log p(x_i; \theta). \tag{184}$$

Maximizing  $L(\theta)$  is equivalent to minimizing  $-L(\theta)$ . However, computing  $\nabla_{\theta}L(\theta)$  exactly involves an expectation under  $p(x;\theta)$ , which is typically intractable.

## **Positive and Negative Phases:**

• Positive Phase: Computed from the data distribution directly. For data samples  $\{x_i\}$ , the gradient term is:

$$\frac{1}{N} \sum_{i=1}^{N} \frac{\partial E(x_i; \theta)}{\partial \theta}.$$
 (185)

• Negative Phase: Involves an expectation under the model's own distribution,

$$-\left\langle \frac{\partial E(x;\theta)}{\partial \theta} \right\rangle_{p(x;\theta)}.$$
 (186)

Since exact sampling from  $p(x;\theta)$  is expensive, Contrastive Divergence approximates this expectation via a short MCMC chain (e.g., k steps of Gibbs sampling) starting from the data point  $x_i$ . Let  $x_i'$  be the sample drawn after these k steps. Then the negative phase gradient is estimated by:

$$-\left\langle \frac{\partial E(x';\theta)}{\partial \theta} \right\rangle_{\rm CD},\tag{187}$$

where CD indicates the distribution induced by a short chain from data points.

**Parameter Update (CD-**k): Putting the two phases together, the update rule for  $\theta$  after observing  $\{x_i\}$  is:

$$\Delta \theta = \eta \left[ \frac{1}{N} \sum_{i=1}^{N} \frac{\partial E(x_i; \theta)}{\partial \theta} - \left\langle \frac{\partial E(x'; \theta)}{\partial \theta} \right\rangle_{\text{CD}} \right], \tag{188}$$

where  $\eta$  is the learning rate, and  $x'_i$  are samples obtained from a short chain (e.g., k steps of Gibbs sampling) starting at the real data  $x_i$ .

## **Advantages and Limitations:**

- Computational Efficiency: CD-k requires fewer MCMC steps than full equilibrium sampling, enabling faster training than exact gradient methods.
- **Bias:** Because the negative phase is only partially explored with a short chain, the gradient is a biased approximation of the true log-likelihood gradient. Increasing k can reduce this bias but raises computational cost.
- **Scope of Use:** Especially prominent in RBMs, DBMs, or other EBMs with local or pairwise interactions. More advanced approaches include *Persistent Contrastive Divergence* (PCD) [271], where the chain is not reinitialized from data each update, and *Mean-Field* methods [272] for approximate inference.

Contrastive Divergence trades off exactness for computational feasibility when training energy-based models, often yielding good empirical performance despite the inherent approximation in the negative phase. This makes it a practical cornerstone for learning parametric EBMs in high-dimensional image-generation tasks.

### 9.2 Image Generation Metrics

Evaluating the quality of generated images requires metrics that capture both low-level fidelity (e.g., pixel-wise accuracy) and high-level perceptual or distributional realism. Researchers have developed a range of quantitative measures, each highlighting different aspects of the generative model's output. This section introduces four commonly used metrics:

- Peak Signal-to-Noise Ratio (PSNR): Assesses image fidelity by comparing signal power to noise power on a
  pixel-wise basis.
- Structural Similarity Index (SSIM): Evaluates perceptual similarity by modeling luminance, contrast, and structural components.
- Inception Score (IS) [273]: Estimates both the diversity of generated samples and the alignment of each sample with a class-conditional label distribution.
- Fréchet Inception Distance (FID) [274]: Measures the distance between real and generated image distributions by comparing feature-space statistics.

In the following subsections, we present each metric in detail, discussing its mathematical formulation, practical application, and associated strengths and weaknesses for various image generation tasks.

## 9.2.1 Peak Signal-to-Noise Ratio (PSNR)

Peak Signal-to-Noise Ratio (PSNR) is a traditional metric that quantifies the fidelity between an original image  $\mathbf{x}$  and a reconstructed (or generated) image  $\hat{\mathbf{x}}$ . Widely employed in image/video compression and restoration, PSNR measures pixel-wise accuracy in a logarithmic decibel (dB) scale.

Let the Mean Squared Error (MSE) between x and  $\hat{x}$  be

$$MSE = \frac{1}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} (x_{i,j} - \hat{x}_{i,j})^{2},$$
(189)

where H and W are the image height and width (assuming a single channel for simplicity). If  $MAX_I$  denotes the maximum possible pixel intensity (e.g., 255 in 8-bit grayscale), then PSNR is given by

$$PSNR = 10 \log_{10} \left( \frac{(MAX_I)^2}{MSE} \right). \tag{190}$$

# Interpretation:

- **Higher is Better:** Since PSNR measures the ratio of signal power (i.e., squared maximum intensity) to error power, a larger PSNR (in dB) indicates fewer pixel-wise deviations, implying stronger fidelity.
- **Reference-based:** PSNR requires a reference "ground-truth" image, making it suited for tasks like super-resolution, denoising, or inpainting where the objective is to match a high-quality target.

## **Strengths and Limitations:**

- Advantages:
  - Simplicity: PSNR is straightforward to compute and interpret.
  - Historical Precedence: Widely used in compression/restoration literature, providing easy comparisons across studies.

#### • Limitations:

- *Poor Perceptual Correlation*: PSNR may not reflect human visual quality well; two images can have similar PSNR yet differ significantly in perceived detail.
- *Unsuitable for Generative Tasks*: For generative models aiming to create novel images (rather than exact reconstructions), PSNR's pixel-wise emphasis fails to capture realism or diversity.

**Usage:** While PSNR remains a standard for evaluating tasks with a known ground truth (e.g., super-resolution vs. a high-resolution reference, denoising vs. an uncorrupted image), it is less informative in adversarial or large-scale generative scenarios where the model produces new, unconstrained samples. In those cases, metrics like Inception Score (IS) or Fréchet Inception Distance (FID) (§9.2.4) often provide more perceptually meaningful assessments [275].

## 9.2.2 Structural Similarity Index (SSIM)

Structural Similarity Index (SSIM) [204] is a perceptual metric for comparing two images x and y. It serves as an improvement over purely pixel-wise measures (e.g., MSE or PSNR) by capturing local patterns of luminance, contrast, and structural details, thus correlating better with human visual perception.

**Motivation and Components:** SSIM decomposes similarity into three core components:

- 1. **Luminance** The mean intensity  $(\mu_x, \mu_y)$  of each image region.
- 2. **Contrast** The standard deviation  $(\sigma_{\mathbf{x}}, \sigma_{\mathbf{y}})$  or intensity variation.
- 3. **Structure** The normalized covariance  $(\sigma_{xy})$  capturing how pixel intensities co-vary between x and y.

By combining these in a single index, SSIM yields a measure ranging from -1 to 1, where 1 indicates perfect similarity, 0 indicates no correlation, and -1 indicates complete dissimilarity.

**Definition:** The SSIM for two image patches x and y of equal size is defined as

$$SSIM(\mathbf{x}, \mathbf{y}) = \frac{\left(2\,\mu_{\mathbf{x}}\,\mu_{\mathbf{y}} + C_1\right)\left(2\,\sigma_{\mathbf{x}\mathbf{y}} + C_2\right)}{\left(\mu_{\mathbf{x}}^2 + \mu_{\mathbf{y}}^2 + C_1\right)\left(\sigma_{\mathbf{x}}^2 + \sigma_{\mathbf{y}}^2 + C_2\right)},\tag{191}$$

where:

- $\mu_{\mathbf{x}}, \mu_{\mathbf{y}}$  are mean intensities (luminance) of  $\mathbf{x}$  and  $\mathbf{y}$ .
- $\sigma_{\mathbf{x}}, \sigma_{\mathbf{y}}$  are standard deviations (contrast) of  $\mathbf{x}$  and  $\mathbf{y}$ .
- $\sigma_{xy}$  is the covariance capturing structural similarity.
- $C_1, C_2$  are small stabilizing constants, typically set to  $(K_1L)^2$  and  $(K_2L)^2$  where L is the dynamic range (e.g., 255 for 8-bit grayscale) and  $K_1, K_2 \ll 1$ .

## **Interpretation:**

- **Robust to Brightness/Contrast Shifts:** Because SSIM separately compares luminance and contrast, small changes in overall brightness or global intensity do not overly penalize the metric.
- Local Evaluation: SSIM is often computed over local windows, then averaged across the image, providing a spatially aware measure of perceptual differences.

## **Relevance to Image Generation:**

- Objective Quality Assessment: In tasks like super-resolution, denoising, or inpainting, SSIM compares the
  generated image to a ground-truth reference, reflecting perceptual fidelity more accurately than MSE or PSNR
  alone.
- Limits for Generative Tasks: SSIM requires a reference target. In purely generative applications (e.g., unconditional GANs) where no exact ground truth exists, SSIM cannot measure diversity or distribution coverage.

While SSIM is valuable for assessing local perceptual quality, it does not account for global distributional properties or sample diversity. Hence, it is often used alongside *Inception Score* or *Fréchet Inception Distance* (§9.2.4) for a more holistic evaluation of generative models [275].

In general, SSIM provides a more perceptually aligned alternative to pixel-based error metrics, allowing image generation tasks that rely on fidelity to a reference image to benefit from a measure that better aligns with the human visual system.

## 9.2.3 Inception Score (IS)

*Inception Score (IS)* [273] provides a single numeric measure of both the *quality* (or recognizability) and the diversity of images generated by a model. The metric leverages a pretrained Inception-v3 classifier, originally trained on ImageNet, to evaluate how confidently generated samples can be classified and whether the overall set of generated images spans multiple classes.

**Definition:** Let  $p_g(\mathbf{x})$  be the distribution of generated images  $\mathbf{x}$ . When fed through the Inception model, each image  $\mathbf{x}$  yields a conditional label distribution  $p(y \mid \mathbf{x})$ , where y indexes ImageNet classes. Define the marginal distribution of classes as

$$p(y) = \int p(y \mid \mathbf{x}) p_g(\mathbf{x}) d\mathbf{x} \quad \text{(in practice, approximated by sampling)}. \tag{192}$$

The Inception Score is computed as

$$IS = \exp\left(\mathbb{E}_{\mathbf{x} \sim p_g} \left[ KL(p(y \mid \mathbf{x}) \mid p(y)) \right] \right), \tag{193}$$

where  $\mathrm{KL}\big(p(y\mid \mathbf{x})\|p(y)\big)$  is the Kullback–Leibler divergence of the conditional distribution  $p(y\mid \mathbf{x})$  from the marginal p(y).

## **Interpretation:**

- Quality (Confidence): The term  $p(y \mid \mathbf{x})$  is expected to have low entropy (i.e., be peaked around a single class), implying that  $\mathbf{x}$  is recognizable as a valid object.
- Diversity: The marginal p(y) should have high entropy—i.e., images are distributed among many classes. A high KL divergence indicates that x strongly activates a distinct class label relative to the overall distribution, reflecting class diversity.
- Exponential Form: Taking exp of the average KL divergence yields a positive scalar, commonly larger than 1. Higher IS values suggest better generative performance.

## **Practical Computation:**

- 1. Sampling Generated Images: Draw a sufficiently large set  $\{x_i\}$  from your generative model.
- 2. **Inception Predictions:** For each  $\mathbf{x}_i$ , compute  $p(y \mid \mathbf{x}_i)$  using Inception-v3's softmax output.
- 3. Approximate p(y): Estimate the marginal p(y) by averaging conditional predictions:

$$p(y) \approx \frac{1}{N} \sum_{i=1}^{N} p(y \mid \mathbf{x}_i). \tag{194}$$

4. Compute IS: Evaluate  $\mathbb{E}_{\mathbf{x} \sim p_q}[\mathrm{KL}(p(y \mid \mathbf{x}) || p(y))]$  and exponentiate the result.

### **Limitations:**

- 1. Dependence on Inception-v3: The metric heavily relies on ImageNet-trained features, making it less reliable in domains or label distributions far from ImageNet classes.
- 2. *Mismatch with Human Perception:* A high IS might not align with subjective visual realism; images can be "classifiable" yet lack photorealism or contain subtle artifacts.
- 3. *Potential Blindness to Mode Collapse:* If a few modes already suffice for diversity among recognized classes, the model may still yield repeated patterns within each class that IS fails to penalize.
- 4. *Insensitivity to Class Balance:* IS does not penalize generating certain classes more often than others if coverage of classes is still broad.
- 5. Sample Efficiency: A large number of generated samples is needed for a stable estimate, and the metric can exhibit high variance for smaller sets.

**Usage Context:** Despite these limitations, *Inception Score* remains a popular and straightforward heuristic for evaluating image quality and diversity in generative tasks—particularly when real-world labels are in line with ImageNet concepts. However, for thorough assessment, practitioners often complement IS with measures like *Fréchet Inception Distance* (§9.2.4) or user studies to better capture perceptual fidelity and distribution coverage.

## 9.2.4 Fréchet Inception Distance (FID)

Fréchet Inception Distance (FID) [274] evaluates the realism and diversity of generated images by measuring how closely their feature distribution matches that of real images. Unlike the Inception Score (§9.2.3), which focuses on per-image class distributions, FID compares the global statistics of high-level feature activations between real and generated datasets.

**Feature Extraction:** Let  $\mathbf{x} \in X$  be samples from the real dataset and  $\hat{\mathbf{x}} \in \hat{X}$  be samples from the model's generated dataset. Both sets are fed through a fixed Inception network, and feature activations (from a chosen layer, often a pool or hidden representation) are extracted:

$$\mathbf{x}' = \operatorname{Inception}(\mathbf{x}), \quad \hat{\mathbf{x}}' = \operatorname{Inception}(\hat{\mathbf{x}}).$$
 (195)

Denote  $\{x'\}$  and  $\{\hat{x}'\}$  as sets of feature vectors for real and generated data, respectively.

**Modeling with Multivariate Gaussians:** Assume that these feature activations follow a multivariate Gaussian distribution in  $\mathbb{R}^d$ . Let

$$\mu_x, \Sigma_x$$
 and  $\mu_{\hat{x}}, \Sigma_{\hat{x}}$  (196)

be the mean and covariance of real and generated activations. The FID then defines the *Fréchet distance* between these Gaussians as:

$$FID(\mathbf{x}', \hat{\mathbf{x}}') = \|\mu_x - \mu_{\hat{x}}\|^2 + Tr(\Sigma_x + \Sigma_{\hat{x}} - 2(\Sigma_x \Sigma_{\hat{x}})^{1/2}), \tag{197}$$

where:

- $\|\mu_x \mu_{\hat{x}}\|$  is the Euclidean distance of means, capturing shifts in the distribution's center.
- $Tr(\cdot)$  denotes the trace operator (sum of diagonal elements).
- $(\Sigma_x \Sigma_{\hat{x}})^{1/2}$  is the matrix square root of the product of the two covariance matrices, well-defined since  $\Sigma_x$  and  $\Sigma_{\hat{x}}$  are positive semi-definite.

## Advantages:

- **Realism and Diversity:** By comparing mean and covariance of feature embeddings, FID evaluates both the average "location" (quality) and the spread (diversity) of generated samples.
- Less Sensitive to Noise than IS: FID can capture subtle distribution shifts even if images appear recognizable to Inception but differ in style or texture.

#### **Limitations:**

- 1. **Domain Dependence on Inception:** Similar to Inception Score, the features are extracted from an ImageNettrained network, which may be suboptimal for domains far removed from natural images.
- 2. **Gaussian Assumption:** FID implicitly treats feature activations as Gaussian, which might not hold perfectly. Deviations from normality can impact the metric's accuracy.
- 3. **Requires Large Sample Sets:** Reliable estimation of means and covariances often demands a sufficiently large dataset for both real and generated samples.

**Usage Context:** Despite these limitations, *FID* has become a standard metric for evaluating generative models in vision tasks. By simultaneously penalizing discrepancies in feature-space location and spread, it provides a more robust assessment of how faithfully a model reproduces the underlying data distribution than single-sample or classification-based metrics.

# 10 Natural Language Processing (NLP)

Natural Language Processing (NLP) is a subfield of artificial intelligence that focuses on the interaction between computers and human languages. It involves the development of algorithms and models that allow computers to

understand, interpret, and generate human language in a way that is both meaningful and useful. NLP encompasses a wide range of tasks, from basic text processing and translation [184] to complex language generation [276].

The main goal of NLP is to bridge the gap between human communication and machine understanding, allowing more natural and efficient interactions with technology. This includes enabling machines to comprehend the subtleties of human language, such as syntax, semantics, context, and sentiment.

Tha main tasks in NLP are the following:

- 1. Text Classification: Involves categorizing text into predefined classes or categories. Examples include sentiment analysis [277], spam detection [278], and topic classification [279].
- 2. Language Modeling: Predicts the probability of a sequence of words. It is a foundational task in NLP, crucial for applications like text generation and autocomplete [276].
- 3. Machine Translation: Converts text from one language to another. Popular applications include translating documents, websites, and real-time speech translation [184].
- 4. Named Entity Recognition (NER): Identifies and classifies proper names, such as people, organizations, and locations, within text [280].
- 5. Part-of-Speech Tagging: Involves labeling each word in a sentence with its corresponding part of speech, such as noun, verb, adjective, etc [281].
- 6. Text Summarization: Automatically generates a concise summary of a larger body of text. It can be extractive (selecting key sentences) or abstractive (generating new sentences) [282].
- 7. Question Answering: Involves building systems that can automatically answer questions posed in natural language [283].

These tasks need specialized techniques, including different loss functions and performance metrics for model optimization and evaluation. NLP evolves rapidly due to machine learning advances and large dataset availability. The next sections will discuss key loss functions and metrics, explaining their applications and relevance.

## 10.1 Loss Functions Used in NLP

Loss functions are central to training robust and accurate Natural Language Processing (NLP) models. From classifying text and predicting next-word tokens to generating coherent sentences in machine translation or summarization, carefully chosen loss functions guide each model's parameters toward meaningful language representations and better performance on downstream tasks.

While general-purpose losses such as *Cross-Entropy* or *Hinge Loss* appear across multiple domains (including computer vision), they also feature prominently in NLP settings because many language tasks, such as text classification and sequence labeling, boil down to discriminative objectives. Moreover, certain loss formulations specifically address the intricacies of language modeling and sequence generation, acknowledging the sequential, probabilistic nature of text.

In this section, we provide an overview of commonly used loss functions in NLP, each one exhibits different strengths and trade-offs, from straightforward implementations for single-label classification to more elaborate designs that handle sequential dependencies, large vocabularies, or exposure bias in generated text. By selecting the most appropriate loss function, NLP practitioners can balance training efficiency, model interpretability, and the desired linguistic or functional outcome.

Table 19 summarizes these widely used losses for NLP tasks, detailing their typical application scenarios, data assumptions, advantages, and potential pitfalls. Understanding when and how to leverage each loss function is critical for achieving state-of-the-art results in modern NLP pipelines.

## 10.1.1 Cross-Entropy Loss (Token-Level)

Cross-Entropy (CE) loss is fundamental in NLP tasks involving sequence prediction, as it quantifies the divergence between the model's predicted probability distribution and the true distribution for each token. In practice, the ground-truth token is encoded as a one-hot vector, and the model's output is typically a softmax distribution over the vocabulary.

**Definition:** Let  $\mathbf{y} = (y_1, y_2, \dots, y_T)$  be the target token sequence of length T, and let  $\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2, \dots, \hat{y}_T)$  be the model's predicted tokens. During training, the model actually produces a probability distribution  $\mathbf{p}_t = p(\cdot \mid \mathbf{y}_{< t}) \in \mathbb{R}^V$  at each timestep t over a vocabulary of size V. The token-level cross-entropy loss is then:

Table 19: Guidelines for selecting a loss function in NLP based on usage, data characteristics, advantages and limitations.

Function	Usage	Data Characteristics	Advantages	Limitations
Cross- Entropy Loss	Language modeling Machine translation Summarization Token-level classifica- tion	Typically discrete to- kens Large vocabularies One-hot or sparse la- bels	Direct optimization of token accuracy Well-established and easy to implement	May not capture global sequence quality Can lead to exposure bias
Hinge Loss	Binary text classifica- tion Sentiment analysis Margin-based se- quence labeling	Mostly binary labels Structured or sparse features	Promotes large margin separation Robust to outliers	Primarily for binary tasks Needs adaptation for multi-class Less common in mod- ern NLP
Cosine Similarity Loss	Semantic similarity Paraphrase identifica- tion Information retrieval	Dense embeddings Continuous vector spaces	Aligns well with embedding-based tasks Captures semantic closeness	Zero similarity can be ambiguous Not ideal for sparse in- puts
Marginal Ranking Loss	Ranking documents Question-answer retrieval Recommendation systems	Paired relevance data Positive vs. negative samples	Improves ordering of relevant vs. irrelevant Flexible margin-based design	Pairwise comparisons can be expensive Fine-grained ranking errors may go unpenal- ized
CTC Loss	Speech recognition Sequence alignment- free tasks OCR-like problems	Input longer than output Unaligned sequence data Blank symbol handling	Learns alignment automatically Handles variable- length outputs	More complex to implement Can struggle with large vocabularies
Minimum Risk Training	Machine translation Summarization Dialog systems	Full-sequence metrics BLEU or ROUGE- based evaluation	Directly optimizes fi- nal metric Better global coher- ence	Computationally expensive sampling Requires careful tuning of scaling factors
REINFORC	ENon-differentiable metric optimization Generative tasks (MT, summarization)	Sequence-level re- wards Discrete token actions	Flexible for custom re- wards Improves performance on metrics like BLEU	High variance in gradient estimates Slow convergence without variance reduction

$$\mathcal{L}_{CE} = -\frac{1}{T} \sum_{t=1}^{T} \log p(y_t \mid y_{< t}),$$
 (198)

where  $p(y_t \mid y_{< t})$  denotes the predicted probability (from the model's softmax layer at time t) of the correct token  $y_t$ . Formally, if  $\mathbf{p}_t$  is the predicted distribution over all possible tokens at step t, then  $p(y_t \mid y_{< t}) = \mathbf{p}_t[y_t]$ .

Minimizing  $\mathcal{L}_{\mathrm{CE}}$  pushes the model to assign higher probability mass to the correct tokens at each timestep, effectively learning the conditional distributions needed for sequence generation or token-level classification.

## **Usage in NLP Tasks:**

- 1. **Language Modeling [284]**: Predicting the next word given previous context. Cross-entropy penalizes the model when it assigns low probability to the correct next word, thus learning coherent text generation.
- 2. **Machine Translation** [184]: Mapping a source-language sentence into a target-language sentence. CE is computed at each target token to align the generated translation with reference translations, guiding the model to produce contextually accurate sequences.

- 3. **Summarization** [285]: Generating a condensed summary of a longer text. The cross-entropy loss encourages the system to reproduce relevant tokens from reference summaries, ensuring fidelity and coverage of essential information.
- 4. **Token Labeling Tasks**: In part-of-speech tagging [286] or named entity recognition [287], each token's label is predicted. CE at each timestep penalizes errors in assigning the correct label, effectively treating each token as a classification instance.

## **Strengths and Limitations:**

#### • Strengths:

- Direct Token-Level Supervision: The model receives a clear gradient signal for each token misclassification, supporting stable training in sequence tasks.
- Broad Applicability: Widely used across classification-based NLP tasks, from language modeling to sequence labeling.

## • Limitations:

- Local Focus: Cross-entropy penalizes token mismatches independently, ignoring potential correlation or reordering effects in the final output (e.g., important for machine translation beyond lexical matches).
- *Discrepancy With End Metrics*: For tasks evaluated by BLEU, ROUGE, or other sequence-level metrics, token-level CE may not capture global sequence properties or semantic coherence.
- Exposure Bias: Models trained with "teacher forcing" see ground-truth history at each step, causing potential discrepancies at inference if previous tokens are mispredicted.

Token-level *Cross-Entropy Loss* remains the primary objective in many supervised and sequence-to-sequence frameworks, reliably guiding the model to align its per-token distributions with the ground truth. However, advanced or specialized tasks might incorporate additional objectives (e.g., sequence-level RL or minimum risk training) to address holistic output fidelity and potential mismatch with final evaluation metrics.

## 10.1.2 Hinge Loss

Hinge Loss is a margin-based objective often employed in binary classification tasks, where labels are encoded as  $\{-1, +1\}$ . Although hinge loss is more traditionally linked to Support Vector Machines (SVMs) in computer vision or general classification, it also features in various NLP contexts such as sentiment analysis, spam detection, or structured prediction under *Structured SVMs* [288, 66, 289, 290].

**Definition:** For an input  $\mathbf{x} \in \mathcal{X}$ , a model (e.g., linear or kernel-based) produces a real-valued score  $f(\mathbf{x})$ . The true label  $y \in \{-1, +1\}$ . The *Hinge Loss* is defined by:

$$\mathcal{L}_{\text{hinge}}(\mathbf{x}, y) = \max(0, 1 - y f(\mathbf{x})). \tag{199}$$

This implies:

- If y f(x) ≥ 1, the loss is 0, indicating that x is not only classified correctly but also lies outside or on the margin boundary (i.e., with sufficient confidence).
- If  $y f(\mathbf{x}) < 1$ , the model incurs a penalty proportional to  $(1 y f(\mathbf{x}))$ , pushing parameters to adjust until  $\mathbf{x}$  crosses the margin or is better separated.

#### **Usage in NLP:**

- 1. **Binary Text Classification:** Tasks like *spam detection* or *sentiment analysis* often reduce to a  $\{-1, +1\}$ -label scenario. A linear or kernel-based SVM with hinge loss defines a max-margin boundary separating positive from negative classes.
- 2. **Structured SVMs for Sequence Labeling:** In part-of-speech tagging [286] or named entity recognition [287], hinge loss can be adapted to *structured prediction* (structured hinge loss). The model's output is an entire label sequence, and the margin constraints incorporate sequence dependencies or feature-based interactions [288, 66].

- 3. **Sparse Linear Models in NLP:** L1- or L2-regularized hinge loss is employed to promote sparse solutions or certain generalization behaviors [291]. This can be beneficial when the input feature space is large (e.g., high-dimensional lexical features).
- 4. Adaptive Losses and Adversarial Frameworks: Recent methods adapt hinge-based objectives for balancing precision and recall (e.g., in document-level relation extraction [292]) or for adversarially aligning model predictions with performance metrics in structured tasks [290].

## Advantages:

- Max-Margin Principle: By ensuring  $\max(0, 1 y f(\mathbf{x}))$ , the hinge loss explicitly drives the decision boundary away from training samples, encouraging robust classification.
- Suitability for Sparse and Structured Models: Hinge loss, paired with regularization (L1/L2), can yield sparser parameter vectors in large-scale NLP tasks with vast lexical or n-gram features.

#### **Limitations:**

- **Binary Nature:** Extending hinge loss to multi-class problems requires one-vs.-all, one-vs.-one, or structured multi-class formulations, which can complicate implementation.
- No Probabilistic Interpretation: Hinge loss produces a margin-based decision function rather than a probability estimate, unlike cross-entropy which naturally yields class probabilities via softmax.
- **Potential Large Margin Tuning:** Selecting the margin constraints or regularizer weights can be non-trivial, impacting the stability and performance of classification or sequence labeling tasks.

## 10.1.3 Cosine Similarity Loss

Cosine Similarity Loss [293] leverages the cosine of the angle between two vector embeddings as a measure of semantic closeness. Widely used in NLP tasks involving sentence, phrase, or word embeddings, it aims to maximize the cosine similarity for pairs deemed "similar" (e.g., paraphrases) and minimize it for pairs deemed "dissimilar" (e.g., unrelated texts).

**Cosine Similarity:** Given two non-zero vectors  $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$ , the *cosine similarity* is:

$$\cos_{\sin}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|},$$
(200)

where  $\mathbf{u} \cdot \mathbf{v}$  denotes the dot product, and  $\|\mathbf{u}\|$  is the Euclidean norm of  $\mathbf{u}$ . The value of  $\cos_{\sin}$  ranges from -1 to 1; 1 indicates parallel (identical direction) vectors, 0 indicates orthogonality, and -1 indicates diametrically opposed vectors

To embed this measure into a *loss function*, one design is:

$$\mathcal{L}_{\cos}(\mathbf{u}, \mathbf{v}) = 1 - \cos_{\sin}(\mathbf{u}, \mathbf{v}), \tag{201}$$

which is minimized when  $\mathbf{u}$  and  $\mathbf{v}$  are maximally aligned ( $\cos_{\min} = 1$ ). For pairs that should be dissimilar, practitioners often define a complementary margin-based penalty to push their similarity below a certain threshold, for instance:

$$\mathcal{L}_{\text{neg}}(\mathbf{u}, \mathbf{v}) = \max(0, \cos_{\sin}(\mathbf{u}, \mathbf{v}) - \text{margin}), \tag{202}$$

so that vectors representing dissimilar text remain sufficiently far apart in the embedding space.

## **NLP Applications:**

- 1. **Semantic Textual Similarity (STS):** Models produce embeddings for sentence pairs  $\{u, v\}$ . Cosine similarity loss encourages semantically similar sentences to lie close (high cosine sim) and dissimilar ones to remain distant.
- 2. **Paraphrase Identification:** Sentences conveying the same meaning but different wording benefit from a training objective that drives embeddings of paraphrases to have near-cosine-sim = 1.

- 3. **Information Retrieval & Ranking:** Documents and queries are embedded. Relevant (query, doc) pairs are forced to have higher similarity than irrelevant ones, improving retrieval performance.
- 4. **Text Clustering:** By grouping embeddings of similar texts together and separating unrelated texts, the loss fosters coherent clusters in embedding space.

#### **Limitations:**

- **Zero Similarity Ambiguity:** A similarity of 0 suggests orthogonality, which might arise from actual dissimilarity or from insufficient representational overlap (e.g., no shared features).
- Dense Vector Requirement: Typically used with continuous embeddings (word2vec, BERT, etc.). For sparse or one-hot vectors, additional normalization or transformations may be needed.
- Margin Definition for Negative Pairs: A naive loss like  $1 \cos \le \min$  only accounts for pushing similar vectors together. Handling negative (dissimilar) pairs might require margin-based or multi-pair frameworks.

Cosine Similarity Loss directly aligns with the geometric intuition of semantic closeness, making it intuitive for tasks like STS, paraphrase detection, or retrieval/ranking problems in NLP. However, care is needed to handle negative examples or define appropriate margins, especially in large-scale or multi-class settings.

### 10.1.4 Marginal Ranking Loss

Marginal Ranking Loss (also called RankNet loss or pairwise ranking loss) [294, 295] is a margin-based objective commonly used in tasks where the model must rank items—such as text passages, documents, or sentences—by relevance. By comparing a *positive* (relevant) item to a *negative* (irrelevant) item, the model is encouraged to assign higher scores to relevant items, improving its ability to order results effectively.

**Definition:** Let  $\mathbf{q} \in \mathcal{Q}$  represent a query (or context),  $\mathbf{p}$  a positive/relevant item, and  $\mathbf{n}$  a negative/irrelevant item. Suppose the model defines a scoring function  $f(\mathbf{q}, \mathbf{z})$ , producing a scalar indicating how well item  $\mathbf{z}$  matches or is relevant to  $\mathbf{q}$ . The *Marginal Ranking Loss* is then:

$$\mathcal{L}(\mathbf{q}, \mathbf{p}, \mathbf{n}) = \max(0, \operatorname{margin} - f(\mathbf{q}, \mathbf{p}) + f(\mathbf{q}, \mathbf{n})), \tag{203}$$

where margin > 0 is a hyperparameter that enforces a separation between the score of positive pairs and negative pairs. Intuitively, the loss remains 0 if  $f(\mathbf{q}, \mathbf{p})$  already exceeds  $f(\mathbf{q}, \mathbf{n})$  by margin or more, indicating sufficient discrimination between relevant and irrelevant items. Otherwise, the model incurs a penalty proportional to the shortfall.

## **Applications in NLP:**

- 1. **Information Retrieval:** Given a user query  $\mathbf{q}$ , documents or passages are ranked by  $f(\mathbf{q}, \mathbf{d})$ . Marginal Ranking Loss compels the model to assign higher scores to relevant  $(\mathbf{p})$  than to non-relevant  $(\mathbf{n})$  documents, improving retrieval quality.
- 2. **Question-Answering (QA):** For QA systems, each candidate answer can be scored relative to the question. The model thus learns to rank correct/precise answers **p** above incorrect/less relevant answers **n**. This leads to higher-ranking correct answers at inference time.
- 3. **Recommendation & Personalization:** In scenarios where items (e.g., articles or products) are recommended based on user preference **q**, the ranking loss encourages relevant items to appear above irrelevant ones, tailoring suggestions to user tastes.
- 4. **Semantic Similarity / Paraphrase Detection:** The model ranks paraphrased or near-duplicate sentences **p** above dissimilar ones **n** for a given reference. This fosters more consistent semantic grouping or paraphrase identification.

## **Key Considerations:**

- Margin Tuning: The choice of margin is critical. If set too large, many pairs may incur non-zero loss, making optimization difficult; if too small, the model might not robustly separate relevant from irrelevant items.
- Computational Cost: For large-scale ranking tasks, enumerating all (p, n) pairs can be expensive. Practitioners often use negative sampling or partial approximations to scale training.

- **Fine-Grained Relevance:** When items differ only subtly in relevance, a simple margin-based approach may insufficiently penalize smaller ranking errors. Additional constraints or label-based weighting might be needed for nuanced tasks.
- Pairwise Nature: Marginal Ranking Loss compares pairs of items at a time, ignoring potential broader context
  or synergy among multiple items. More advanced methods (e.g., listwise ranking) can capture global ordering
  constraints.

Marginal Ranking Loss provides a direct mechanism for training models to rank relevant text segments higher than irrelevant ones. By imposing a margin-based separation in scores, it is effective in retrieval, QA, recommendation, and paraphrase detection tasks. However, handling subtle relevance differences, controlling computational complexity, and capturing global ordering constraints remain challenges in applying this pairwise ranking approach at scale.

## 10.2 Losses for Sequence Generation

Sequence generation tasks in NLP involve predicting a sequence of words or tokens given an input, such as translating a sentence from one language to another, summarizing a document, or generating a response in a conversation. The challenges associated with sequence generation include maintaining fluency and coherence and ensuring that the sequence generated accurately reflects the intended meaning or information. This section explores the loss functions commonly used to optimize models for these tasks.

## 10.2.1 Connectionist Temporal Classification (CTC) Loss

Connectionist Temporal Classification (CTC) [296] is designed for sequence-to-sequence learning tasks where the alignment between input and output sequences is unobserved or uncertain. Commonly used in automatic speech recognition and handwriting recognition, CTC aligns a long input sequence  $\mathbf{x}$  of length T to a shorter target sequence  $\mathbf{y}$  of length U, without the need for manually specified frame-to-label correspondences.

CTC introduces a special "blank" symbol  $(\emptyset)$  to handle variable-length alignments and repeated tokens. For each time step  $t=1,\ldots,T$ , the model outputs a probability distribution over an extended label set  $\mathcal{L}=\{\text{vocabulary tokens}\}\cup\{\emptyset\}$ . By *collapsing* consecutive identical tokens and removing blanks, one obtains a final predicted sequence  $\hat{\mathbf{y}}$ . The probability  $P(\mathbf{y} \mid \mathbf{x})$  is then computed by summing over the probabilities of all possible alignments  $\pi$  that collapse to  $\mathbf{y}$ .

Formally, let Align(y) be the set of all valid alignment paths of length T that collapse to the sequence y. Each alignment path  $\pi = (\pi_1, \dots, \pi_T)$  assigns a label  $\pi_t \in \mathcal{L}$  at each time t. Then the total probability of y is:

$$P(\mathbf{y} \mid \mathbf{x}) = \sum_{\pi \in \text{Align}(\mathbf{y})} P(\pi \mid \mathbf{x}), \tag{204}$$

where

$$P(\pi \mid \mathbf{x}) = \prod_{t=1}^{T} P(\pi_t \mid \mathbf{x}, t), \tag{205}$$

and  $P(\pi_t \mid \mathbf{x}, t)$  is the predicted probability of label  $\pi_t$  (including blank) at time t. The alignment path  $\pi$  is thus assigned a probability equal to the product of the model's label probabilities at each frame.

**Definition:** The CTC Loss is defined as the negative log-probability of the target sequence y:

$$\mathcal{L}_{\text{CTC}} = -\log P(\mathbf{y} \mid \mathbf{x}). \tag{206}$$

Minimizing this objective encourages the model to increase the sum of probabilities of all valid alignment paths that produce  $\mathbf{y}$ . Since  $P(\mathbf{y} \mid \mathbf{x})$  involves a summation over exponentially many alignments, efficient dynamic programming algorithms (e.g., the forward-backward algorithm specialized for CTC) are used to compute  $P(\mathbf{y} \mid \mathbf{x})$  and its gradient in polynomial time.

## Use Cases in NLP:

• Speech and Character-level Recognition: CTC often appears in speech-to-text systems, mapping audio frames to characters or phonemes without explicit frame-to-label alignments.

- Handwriting Recognition: By treating the image of a handwritten text line as the input sequence, CTC can learn to align and generate text transcriptions.
- Any Sequence Task Without Alignment Supervision: If an NLP task has unaligned input-output pairs and a monotonic alignment assumption (i.e., tokens in y must appear in chronological order in x), CTC can be adapted.

## **Limitations:**

- **Strict Monotonic Alignment Assumption:** CTC presumes that the output sequence cannot reorder input steps, which might be restrictive if the alignment is not strictly left-to-right.
- Blank Symbol Overhead: The blank token handles variable lengths but adds complexity to the output space, requiring careful training and hyperparameter tuning.
- Complexity in Multi-sequence or Non-monotonic Tasks: In general, tasks that violate the linear alignment property (e.g., reordering in machine translation) may not align well with CTC's core assumptions.

In general, *CTC Loss* provides a framework for learning sequence mappings without manual alignments, effectively summing over all plausible ways to match a shorter target sequence to a longer input. Dynamic programming algorithms enable tractable training, making CTC a powerful choice for tasks like speech recognition and character-level text generation under monotonic alignment constraints.

## 10.2.2 Minimum Risk Training (MRT)

Minimum Risk Training (MRT) [297] is a sequence-level optimization strategy that directly aligns model training with the final evaluation metric (e.g., BLEU [298], ROUGE [299]), rather than relying on token-level losses like cross-entropy. By treating metric-based discrepancies as "risk," MRT minimizes the expected risk under the model's predicted distribution, thereby better capturing global sequence quality.

Many NLP tasks—such as machine translation or text summarization—are evaluated by sequence-level metrics (e.g., BLEU, ROUGE) that aggregate the coherence, coverage, or overlap of the entire output. However, standard token-level losses, like cross-entropy, do not necessarily correlate with these metrics. MRT addresses this by replacing the local objective with a global risk function tied to a chosen evaluation metric.

**Definition:** Let  $\mathbf{x}$  be an input, and  $\mathbf{y}$  a target reference sequence from a dataset of size N. The model produces a distribution  $p_{\theta}(\hat{\mathbf{y}} \mid \mathbf{x})$  over possible output sequences  $\hat{\mathbf{y}}$ . Define a loss function (or "risk")  $\Delta(\hat{\mathbf{y}}, \mathbf{y})$  that measures the discrepancy between a predicted sequence  $\hat{\mathbf{y}}$  and the ground truth  $\mathbf{y}$  according to a specific metric. MRT then seeks to minimize the expected risk:

$$\mathcal{L}_{MRT} = \mathbb{E}_{\hat{\mathbf{y}} \sim p_{\theta}(\hat{\mathbf{y}}|\mathbf{x})} [\Delta(\hat{\mathbf{y}}, \mathbf{y})]. \tag{207}$$

However, summing over all possible sequences  $\hat{\mathbf{y}} \in \mathcal{Y}$  is generally intractable. A common approach samples a subset  $\mathcal{Y}_s \subseteq \mathcal{Y}$  from  $p_{\theta}(\hat{\mathbf{y}} \mid \mathbf{x})$ . The MRT objective becomes:

$$\mathcal{L}_{\text{MRT}} \approx \sum_{\hat{\mathbf{y}} \in \mathcal{Y}_{s}} \frac{\exp(\alpha \log p_{\theta}(\hat{\mathbf{y}} \mid \mathbf{x}))}{\sum_{\hat{\mathbf{y}}' \in \mathcal{Y}_{s}} \exp(\alpha \log p_{\theta}(\hat{\mathbf{y}}' \mid \mathbf{x}))} \Delta(\hat{\mathbf{y}}, \mathbf{y}), \tag{208}$$

where  $\alpha > 0$  scales or sharpens the distribution, and  $\Delta(\hat{y}, y)$  is the sequence-level discrepancy (e.g., 1 - BLEU). Minimizing  $\mathcal{L}_{\text{MRT}}$  pushes the model to assign higher probability to sequences with lower risk (i.e., better metric scores).

### **Applications:**

- Machine Translation: Directly optimize for BLEU or related translation metrics by defining  $\Delta(\hat{\mathbf{y}}, \mathbf{y}) = 1 \text{BLEU}(\hat{\mathbf{y}}, \mathbf{y})$ . The model is incentivized to produce outputs scoring higher in BLEU, thus often correlating better with human judgments.
- **Text Summarization**: Let  $\Delta(\hat{\mathbf{y}}, \mathbf{y}) = 1 \text{ROUGE}(\hat{\mathbf{y}}, \mathbf{y})$ . Training under MRT encourages the system to maximize ROUGE overlap, leading to more coverage of essential information from the source text.
- Dialog Systems or Generative QA: One can define a dialog metric  $\Delta$  capturing relevance and informativeness. MRT ensures the final outputs better align with conversation-level quality measures.

## Advantages:

- Closer Alignment with Evaluation Metrics: By training to minimize risk under a specific metric, MRT can yield improvements over token-level cross-entropy, especially for tasks demanding global coherence.
- Sampling Complexity: Approximating the expectation requires sampling from the model distribution, which can be computationally expensive for large vocabularies or long sequences.
- Choice of Scale and Loss Function: The scaling factor  $\alpha$  and the definition of  $\Delta(\hat{y}, y)$  significantly influence stability and convergence. Too large an  $\alpha$  can cause overly peaked distributions, hampering exploration.

Overall, *Minimum Risk Training* provides a powerful mechanism for sequence learning tasks where final performance is measured by specialized metrics (e.g., BLEU or ROUGE). By optimizing a direct alignment between training objectives and evaluation criteria, MRT can substantially boost the model's end-task performance. However, it requires careful sampling strategies, hyperparameter tuning, and metric design to achieve stable, efficient training.

## 10.2.3 REINFORCE Algorithm

REINFORCE [300] is a fundamental policy gradient method in reinforcement learning (RL) that has been adopted in sequence generation tasks within NLP to optimize non-differentiable metrics (e.g., BLEU [298], ROUGE [299]). Instead of backpropagating through each token with a differentiable loss like cross-entropy, REINFORCE treats each generated sequence as an outcome of a stochastic policy, using reward signals from final outputs to guide parameter updates.

Let a model parameterized by  $\theta$  define a stochastic policy  $\pi_{\theta}$  over sequences  $\tau = (x_1, x_2, \dots, x_T)$ . We interpret generating token  $x_t$  at step t as choosing an *action*, and the entire generated sequence  $\tau$  then yields a scalar reward  $R(\tau)$  that reflects performance under some non-differentiable metric (e.g., BLEU for machine translation). The RL objective is to *maximize* the expected reward:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta}} [R(\tau)]. \tag{209}$$

By the *policy gradient theorem* [301], the gradient of  $J(\theta)$  w.r.t.  $\theta$  can be approximated by sampling trajectories  $\tau$  from  $\pi_{\theta}$  and computing:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[ R(\tau) \nabla_{\theta} \log \pi_{\theta}(\tau) \right], \tag{210}$$

where  $\log \pi_{\theta}(\tau)$  is the log-probability of the sequence  $\tau$ . This gradient term indicates that sequences  $\tau$  with higher rewards  $R(\tau)$  should have their log-probability increased, while lower-reward sequences should be deprioritized.

**Algorithmic Update (REINFORCE Rule):** Given a sample  $\tau$  from the current policy, we update parameters  $\theta$  via:

$$\Delta \theta = \eta R(\tau) \nabla_{\theta} \log \pi_{\theta}(\tau), \tag{211}$$

where  $\eta$  is a learning rate. Over multiple samples, the expectation in Equation (210) is approximated.

## **Application to NLP Sequence Generation:**

- Machine Translation: The model generates a translation  $\tau$  for an input sentence. A metric like BLEU is used as the reward  $R(\tau)$ . By maximizing expected BLEU, the model aligns with how translations are ultimately scored.
- **Summarization:** Summaries are generated, then ROUGE or another measure of coverage is the reward. REINFORCE updates the model to produce more informative or coherent summaries.
- **Dialog Systems:** In conversation, custom rewards based on user satisfaction or engagement can be used. Each generated utterance leads to a final reward, guiding dialogue policy to produce relevant, context-aware responses.

## **Challenges:**

- 1. *High Variance in Gradient Estimates:* Random sampling of sequences can lead to noisy gradient updates, slowing convergence. Baseline functions or variance-reduction methods (e.g., subtracting the average reward) help stabilize training.
- 2. *Credit Assignment:* The reward is often only observed after generating an entire sequence, making it difficult to assign credit or blame to individual tokens. Techniques like reward shaping or partial intermediate rewards can alleviate this.
- Sample Efficiency: Large action spaces (vocabularies) and long sequences require many samples to accurately
  estimate expected rewards, potentially making REINFORCE computationally expensive in complex NLP
  tasks.

*REINFORCE* provides a principled approach for training NLP models on non-differentiable objectives by treating generation as a reinforcement learning problem. It can directly optimize final evaluation metrics, bridging the gap between local (token-level) training signals and global sequence-level quality. However, practitioners must address issues of variance, sample inefficiency, and delayed rewards to achieve stable, effective results in large-scale NLP applications.

#### 10.3 Performance Metrics Used in NLP

Performance metrics provide quantifiable measures to assess how well a model performs a given task. In NLP, these metrics vary widely depending on the nature of the task and the type of output the model produces. For tasks involving classification, metrics like Accuracy, Precision, Recall, F1 Score, and AUC-ROC are crucial for determining the model's ability to correctly classify text into categories. While these metrics are commonly used in general classification problems, their application in NLP involves specific considerations, such as handling imbalanced datasets and the multi-label nature of some tasks.

Metrics like BLEU [298] and ROUGE [299] are essential for sequence generation and text comparison tasks. These metrics evaluate the quality of generated text, such as translations or summaries, by comparing it to reference texts. They provide information on the fluency, adequacy, and informativeness of the generated content.

Other metrics, such as Perplexity [302], are specific to language modeling and measure how well a model predicts a sequence of words. Exact Match is often used in tasks like question answering, where the precision of the model's output is critical.

In this section, we will explore these performance metrics in detail, focusing on their application in NLP tasks. We will discuss their definitions, how they are calculated, and the specific challenges and considerations when applying them to NLP models. This analysis will help in understanding the strengths and limitations of each metric and guide the selection of appropriate metrics for evaluating different NLP applications.

Table 20 provides an overview of commonly employed metrics in NLP. It covers recommendations for use, data characteristics, advantages, and limitations.

## 10.3.1 Accuracy

Accuracy is a standard metric for classification tasks, evaluating the proportion of inputs classified correctly out of the total. Formally, for a dataset of size N with gold-standard labels  $\{y_i\}_{i=1}^N$  and model predictions  $\{\hat{y}_i\}_{i=1}^N$ , the accuracy metric is:

Accuracy = 
$$\frac{1}{N} \sum_{i=1}^{N} \mathbb{I}[\hat{y}_i = y_i],$$
 (212)

where  $\mathbb{I}[\cdot]$  is the indicator function, returning 1 if the condition is true (the prediction matches the gold label) and 0 otherwise.

### **Usage in NLP:**

1. **Text Classification:** Tasks such as *sentiment analysis*, *spam detection*, and *topic classification* involve mapping an input text to one of several discrete labels. Accuracy measures the fraction of texts for which the model assigns the correct label. For sentiment analysis, for instance, accuracy reflects how frequently the system correctly labels input as positive, negative, or neutral [277].

Table 20: Guidelines for selecting a metric in NLP based on usage, data characteristics, advantages, and limitations.

Metric	Usage	Data Characteristics	Advantages	Limitations
Accuracy	Classification tasks General correctness measure	Suitable for balanced data Discrete labels	Easy to interpret Widely recognized	Misleading with imbalanced classes Ignores nuances of errors
Precision, Recall, F1 Score	Tasks requiring fo- cus on false posi- tives/negatives Useful in NER, spam detection	Often imbalanced datasets Varied cost of errors	Provide deeper insight than accuracy F1 balances precision and recall	Hard to capture over- all performance with all three May ignore true nega- tives in some contexts
AUC- ROC	Binary classification Probabilistic outputs	Suited for large datasets Typically two-class problems	Threshold-invariant Summarizes TPR- FPR trade-off	May be less intuitive in multi-class Can be over-optimistic with skewed data
BLEU Score	Machine translation Some text generation tasks	Requires reference translations Text overlap-based	Widely adopted Measures n-gram pre- cision	Ignores synonyms Overly reliant on reference variety
METEOR	Machine translation Summarization Text generation	Reference-based Token-level alignments Uses lexical resources (synonyms, stemming)	Considers precision and recall Rewards synonyms, morphological vari- ants Often better corre- lation with human judgments	Relies on external dictionaries Still mostly token- based Single reference can penalize valid paraphrases
ROUGE Score	Summarization Content coverage tasks	Requires reference summaries Focus on text overlap	Recall-oriented Captures coverage of key content	Surface-level overlap Limited semantic in- sight
Perplexity	Language modeling Text generation	Large, diverse corpora Unconstrained vocab- ulary	Measures predictive fit Standard for evaluat- ing LM fluency	Depends on test corpus Does not reflect semantic correctness
Exact Match	QA, NLI Tasks needing strict output	Typically short answers Matching exact reference	Clear indication of full correctness Binary measure	No partial credit Fails with acceptable paraphrases
WER	Speech recognition Token-level output comparison	Sequence of words Reference transcrip- tions	Direct measure of lexical errors Intuitive fraction of incorrect words	Ignores semantic closeness Penalizes all word errors equally Highly domainsensitive
CER	Speech recognition Character-based or script-based tasks	Character sequences Useful for languages lacking clear word boundaries	Finer-grained than WER Handles scripts with unclear token segmen- tation	No semantic insight All character differ- ences treated equally Can over-penalize small edits

- 2. **Sequence Labeling:** In *part-of-speech tagging*, each token in a sentence receives a syntactic tag. Accuracy then represents the percentage of tokens correctly tagged. Similarly, in *named entity recognition (NER)*, accuracy can measure the proportion of tokens correctly identified as person, location, organization, etc. [286, 287].
- 3. **Language Detection:** For models that classify the language of a given text (e.g., English, Spanish, French), accuracy quantifies the fraction of texts for which the model identifies the correct language.

#### **Advantages and Limitations:**

## · Advantages:

- Simplicity and Interpretability: Accuracy is intuitive, requiring only a count of correct vs. incorrect predictions.
- Broad Applicability: Suited for any classification or labeling scenario with discrete labels.

#### • Limitations:

- Imbalanced Data: In highly skewed class distributions, accuracy can be misleading if a model trivially predicts the majority class.
- Ignores Fine-Grained Quality: Accuracy only indicates correct vs. incorrect classification, not how confident or semantically close an incorrect prediction might be.
- *Context Sensitivity*: Accuracy fails to capture contextual or structured relationships in predictions (e.g., dependencies among tokens in sequence labeling).

Despite its simplicity, *accuracy* remains a fundamental metric for validating classification performance in many NLP tasks. Nevertheless, practitioners often complement it with other metrics (e.g., precision, recall, F1-score, or more specialized measures) to address class imbalance, structured dependencies, and nuanced performance considerations.

# 10.3.2 Precision, Recall, and F1 Score

In earlier sections (§3.2.4, §3.2.5, §3.2.6), we introduced the formal definitions of *precision*, *recall*, and the *F1 score*, which combine both. In *Natural Language Processing (NLP)*, these metrics are indispensable for tasks where correct identification or extraction of linguistic units is essential—particularly under conditions of data imbalance or different costs associated with false positives and false negatives.

**Precision in NLP:** Precision measures how many of the predicted positive (or relevant) instances truly belong to the positive class. It is often critical when the cost of a false positive is high. For instance:

- Named Entity Recognition (NER): Precision indicates the fraction of model-identified entities that are actually correct (e.g., valid person, location, or organization). High precision ensures fewer spurious entity labels, which is essential in information extraction pipelines [287].
- Text Classification (e.g., Spam Detection): Among all emails labeled "spam," precision reflects how many are truly spam. High precision is vital to avoid misclassifying important messages as spam.

**Recall in NLP:** Recall measures the proportion of true positive instances that the model successfully identifies. It is typically crucial when missing a positive instance is costly. Examples include:

- **NER with High-Stakes Data:** Failing to detect critical entities (e.g., legal references, medical conditions) can have severe consequences. High recall helps ensure fewer missed entities.
- **Sentiment Analysis:** In some scenarios, detecting *all* instances of a particular sentiment (e.g., all positive reviews) might be more important than mislabeling a few neutral items as positive [303].

**F1 Score in NLP:** The *F1 score* is the harmonic mean of precision and recall. It succinctly balances both metrics, which is valuable when class distributions are imbalanced or both false positives and false negatives incur significant costs:

$$F1 = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)}.$$
 (213)

In NLP tasks:

- **Text Classification and NER:** F1 helps ensure that the model is not only precise but also captures the majority of positive cases. For instance, in legal or medical document classification, both incorrect inclusions and omissions carry high risk.
- Information Retrieval / QA Systems: Combining high precision (relevant documents or answers) and high recall (comprehensive retrieval) is often crucial. The F1 score offers a single measure balancing both.

#### **Considerations:**

- **Data Imbalance:** Precision and recall are especially important when the positive class is rare (e.g., spam vs. non-spam). Accuracy alone can be misleading if the negative class dominates.
- **Semantic Nuances:** In language tasks, near-miss cases (e.g., partial entity matches or closely related sentiments) can complicate strict definitions of precision and recall. Domain-specific guidelines sometimes refine how "true positives" are determined.
- Task Suitability: While these metrics are widely applicable to classification or sequence labeling, they may be less indicative of broader sequence coherence. Complementary metrics (e.g., BLEU, ROUGE) might be required for generative tasks.

In general, *precision*, *recall*, and *F1 score* remain core metrics in NLP for classification- and labeling-based tasks. Their ability to tease apart the impact of false positives vs. false negatives—then combine these insights into a single balanced score—makes them fundamental to evaluating model performance, especially under imbalanced class distributions or cost-sensitive environments.

## 10.3.3 AUC-ROC

In Section 3.2.13, we introduced *Area Under the Receiver Operating Characteristic* (AUC-ROC) as a measure of a classifier's ability to distinguish between positive and negative classes across various decision thresholds. In *Natural Language Processing (NLP)*, **AUC-ROC** is especially valuable in binary classification tasks where the model outputs class probabilities (e.g., spam vs. non-spam). By plotting the *true positive rate* (TPR) against the *false positive rate* (FPR) at different threshold settings, the area under this ROC curve summarizes the model's performance into a single scalar, effectively capturing how well the model discriminates between two classes overall.

# **Usage in Binary NLP Tasks:**

- 1. **Spam Detection** [304]: The model assigns a probability to each email/message indicating the likelihood of it being spam. The AUC-ROC measures how effectively the model ranks actual spam messages above legitimate ones across all possible thresholds. High AUC implies that, regardless of the specific threshold, the spam/non-spam ranking is robust.
- 2. **Binary Sentiment Analysis** [303]: When classifying text into positive or negative sentiment, AUC-ROC indicates the trade-off between correctly classifying positive reviews (TPR) and misclassifying negative reviews as positive (FPR). A large area under the curve indicates a strong ability to separate positive from negative sentiments at multiple classification cutoffs.
- 3. **Topic vs. Non-Topic Classification** [305, 306, 283]: For a scenario where the model classifies whether a document pertains to a particular topic (positive) or not (negative), AUC-ROC evaluates how well the model ranks truly relevant documents above irrelevant ones. This is central in content filtering or recommendation pipelines, where false positives can degrade user experience.

#### **Advantages and Considerations for NLP:**

- Threshold-Independent Assessment: AUC-ROC evaluates classifier performance across all threshold settings, offering a more holistic assessment than accuracy at a fixed cutoff—especially relevant if one needs to calibrate the decision boundary post-training.
- Robustness to Class Imbalance: By examining TPR vs. FPR, AUC-ROC captures how the model handles skewed class distributions, often found in real-world text classification (e.g., spam detection).
- **Interpretation Cautions:** An AUC near 1.0 signifies robust discrimination. However, in highly imbalanced settings or with cost-sensitive decisions, practitioners may also consult metrics like precision-recall AUC (especially if the positive class is rare).

AUC-ROC is a strong metric in NLP's binary classification tasks that require probabilistic outputs (spam detection, sentiment polarity, topic vs. non-topic classification). Integrating performance across a continuous spectrum of decision thresholds offers a more comprehensive view than single-threshold metrics like accuracy, particularly under class imbalance or variable cost scenarios.

#### 10.3.4 BLEU Score

BLEU (Bilingual Evaluation Understudy) [298] is a widely adopted metric for evaluating machine-generated text, particularly in machine translation. It estimates how closely a candidate translation (or generated text) aligns with one or more human reference translations. BLEU focuses on matching overlapping *n-grams* between candidate and references, capturing lexical and local ordering similarities.

**Steps:** Let the candidate output be c and let  $\{r^{(1)}, r^{(2)}, \dots\}$  be a set of reference strings (translations). BLEU aggregates the following steps:

1. **N-gram Precision Computation:** For n-grams of lengths n = 1, 2, ..., N (often up to 4), compute the "clipped" precision  $p_n$ . That is, count how many n-grams in  $\mathbf{c}$  also appear in any reference, limiting each  $\mathbf{n}$ -gram's count to the maximum times it appears in any single reference. Formally:

$$p_n = \frac{\sum_{\text{n-gram} \in \mathbf{c}} \min[\text{count}_{\mathbf{c}}(\text{n-gram}), \max_k \text{count}_{\mathbf{r}^{(k)}}(\text{n-gram})]}{\sum_{\text{n-gram} \in \mathbf{c}} \text{count}_{\mathbf{c}}(\text{n-gram})}.$$
 (214)

- 2. Geometric Mean of N-gram Precisions: Compute an average of  $\{\log p_1, \log p_2, \dots, \log p_N\}$ . Typically, each  $p_n$  is equally weighted by  $\frac{1}{N}$ .
- 3. **Brevity Penalty (BP):** To penalize excessively short candidates, a factor BP is applied if  $\mathbf{c}$  is shorter than the reference set. Let  $c = |\mathbf{c}|$  (length of the candidate) and r be the effective reference length (e.g., choosing the closest or shortest reference). The brevity penalty is:

$$BP = \begin{cases} 1, & \text{if } c > r, \\ \exp\left(1 - \frac{r}{c}\right), & \text{if } c \le r. \end{cases}$$
 (215)

4. **Final BLEU Score:** Combine the geometric mean of n-gram precisions with BP:

BLEU = BP × exp
$$\left(\sum_{n=1}^{N} w_n \log p_n\right)$$
, (216)

where  $w_n$  are weights for each n-gram precision (often  $w_n = \frac{1}{N}$ ).

#### **Applications in NLP:**

- Machine Translation [184]: BLEU is a de facto standard metric, comparing candidate translations to human references. A higher BLEU indicates better lexical overlap. Although it does not measure semantic adequacy or fluency perfectly, BLEU remains a valuable baseline measure.
- **Text Summarization [285]:** BLEU can be used to compare generated summaries against reference summaries. However, because summarization may allow paraphrasing or different sentence orderings, BLEU's precision-based approach might underestimate the model's actual summarization quality.
- Text Generation [307, 308]: In open-ended generation tasks, BLEU can help gauge how closely a generated
  response aligns with a known or expected sequence, though it may not fully capture creative or semantically
  correct variations.

#### **Limitations:**

- Lexical Overlap Bias: BLEU focuses on n-gram matches, ignoring semantic equivalences that use synonyms or rephrasings. Thus, perfectly valid translations may score lower if they differ lexically from references.
- **Reference Dependence:** More reference translations can capture diverse valid outputs. A single reference might penalize legitimate variations, lowering BLEU scores arbitrarily.
- **Precision Emphasis & Brevity Penalty:** The brevity penalty deters overly short outputs, but the overall precision-based design can undervalue correct expansions or rephrasings absent in references.

Although *BLEU* remains a dominant automatic metric for machine translation and related tasks, practitioners often supplement it with complementary metrics (e.g., METEOR §10.3.5, ROUGE §10.3.6) or human evaluations to obtain a fuller view of translation fluency and adequacy.

## **10.3.5 METEOR**

METEOR (Metric for Evaluation of Translation with Explicit ORdering) [309] is another automatic metric commonly used for evaluating machine translation outputs and other text generation tasks. Unlike BLEU ( $\S10.3.4$ ), which relies on n-gram precision and brevity penalties, METEOR attempts to capture both precision and recall at the unigram level while also considering synonyms, stem matches, and phrase reordering to some extent. Consequently, it often correlates better with human judgments, particularly on more nuanced outputs.

## **Key Concepts:**

- 1. **Token Matching:** METEOR begins by aligning unigrams between a candidate (c) and reference (r) translation. Alignments consider exact token matches, stem or lemma matches, synonyms (using external lexical resources like WordNet), and optional paraphrase matches.
- 2. **Precision and Recall:** Let m be the number of matched unigrams after alignments. Then, define:

$$P = \frac{m}{|\mathbf{c}|}$$
 and  $R = \frac{m}{|\mathbf{r}|}$ , (217)

where  $|\mathbf{c}|$  and  $|\mathbf{r}|$  are the lengths (in tokens) of the candidate and reference, respectively.

3. **Fragmentation Penalty:** METEOR adds a "fragmentation" or "chunking" penalty, which accounts for how well the matched tokens align in contiguous chunks. More scattered matches (i.e., many small segments) incur a higher penalty than fewer, larger segments.

**Scoring Formula:** A common METEOR scoring variant uses the harmonic mean  $F_{\alpha}$  of P and R, combined with a fragmentation penalty Pen. For example:

$$METEOR = (1 - Pen) \times \frac{(1 + \alpha^2) P R}{\alpha^2 P + R},$$
(218)

where  $Pen \in [0, 1]$  is computed from the number of matched segments vs. total matches, and  $\alpha$  is a parameter (often set around 0.9) controlling the relative weight of recall vs. precision.

## **Usage in NLP Tasks:**

- Machine Translation: METEOR aims to address some BLEU limitations by providing partial credit for synonyms or morphological variants. This can yield higher correlation with human judgments, especially in languages with rich morphology [309].
- **Text Summarization:** Like BLEU, METEOR can be employed to measure overlap between candidate summaries and reference summaries, taking into account stem/synonym matches to reward semantically equivalent expressions.
- Any Generation Task with References: For dialogues or paraphrase generation, METEOR offers a more flexible matching scheme than simple n-gram overlap, though it still may not fully capture discourse coherence or complex rephrasings.

## **Limitations:**

- Dependence on Lexical Resources: METEOR's ability to handle synonyms or morphological variants relies on external dictionaries (e.g., WordNet), which may be domain-dependent or incomplete for certain languages.
- **Unigram Focus:** While chunk-based penalties help somewhat, METEOR mainly operates at the token level. It may still miss longer-range syntactic or semantic structures.
- **Reference Dependence:** Like BLEU, the metric is sensitive to the reference set's coverage. A single reference might penalize valid alternate translations not reflected in the reference text.

METEOR serves as an improvement over simpler n-gram matching metrics by incorporating a more flexible matching strategy and balancing precision, recall, and fragmentation considerations. While it does not capture global coherence or deep semantics fully, METEOR often correlates better with human evaluations than BLEU, making it an important complementary metric for machine translation, summarization, and other generation tasks.

#### 10.3.6 ROUGE Score

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [299] is a family of metrics used extensively for automatic summarization and, to a lesser extent, machine translation and other text generation tasks. ROUGE mainly focuses on recall over various text overlaps (e.g., n-grams, sequences), indicating how much of the content in the reference text is recovered by the candidate text.

#### **Core Variants:**

• **ROUGE-N**: Measures the recall of *n*-gram overlaps between candidate (c) and reference (r). For ROUGE-1 (unigrams) or ROUGE-2 (bigrams), define

$$ROUGE-N = \frac{\sum_{\text{gram}_n \in \mathbf{r}} \min \left( \text{count}_{\mathbf{c}}(\text{gram}_n), \text{ count}_{\mathbf{r}}(\text{gram}_n) \right)}{\sum_{\text{gram}_n \in \mathbf{r}} \text{count}_{\mathbf{r}}(\text{gram}_n)},$$
(219)

where  $count_c$  and  $count_r$  are counts of n-grams in candidate and reference, respectively. Only overlapping n-grams that appear in both texts are credited, reflecting content recall.

• **ROUGE-L**: Uses the *Longest Common Subsequence* (LCS) to capture the maximum sequence overlap, potentially skipping tokens in between. If  $LCS(\mathbf{r}, \mathbf{c})$  is the length of the longest common subsequence, then:

$$ROUGE-L = \frac{LCS(\mathbf{r}, \mathbf{c})}{Length(\mathbf{r})}.$$
 (220)

This measure can detect re-ordered or spaced-out matches, better reflecting sentence-level structural overlap.

- ROUGE-W: A weighted extension of ROUGE-L emphasizing consecutive matches more heavily. Longer contiguous overlaps in word sequences yield higher scores.
- **ROUGE-S**: Also called *skip-bigram* ROUGE, it counts pairs of words in the reference (in order) that appear in the candidate, allowing gaps. This addresses non-consecutive word matches.

## **Usage in NLP Tasks:**

- 1. **Text Summarization:** ROUGE is the *de facto* standard for measuring how much important content from the source text is retained in the candidate summary. High recall of key phrases or word sequences is crucial [299].
- 2. **Machine Translation** [310]: Although BLEU (§10.3.4) is more common, ROUGE can provide additional recall-based insight, especially if capturing coverage of reference tokens is essential.
- 3. **General Text Generation** [311]: ROUGE can help evaluate how comprehensive or on-topic the generated text is compared to a gold-standard reference, such as in question-answering or dialogue systems.

As an example, *ROUGE-N* for *n*-gram-based recall is:

$$ROUGE-N = \frac{\sum_{gram_n \in \mathbf{r}} \min \left( count_{\mathbf{c}}(gram_n), count_{\mathbf{r}}(gram_n) \right)}{\sum_{gram_n \in \mathbf{r}} count_{\mathbf{r}}(gram_n)},$$
(221)

where the numerator sums clipped counts of n-grams that appear in both candidate and reference, reflecting how much of the reference n-gram space is covered by the candidate.

#### **Considerations and Limitations:**

- Surface-Level Overlaps: Like BLEU, ROUGE's n-gram comparisons may undervalue semantically correct paraphrases or synonyms that differ lexically.
- **Dependency on Reference Diversity:** Multiple, varied references can improve the metric's fairness by accommodating valid expression alternatives. A single reference might penalize correct solutions not reflected in its wording.
- Emphasis on Recall: ROUGE is intentionally recall-oriented, which can over-reward longer candidates that contain extraneous material. Combining with precision-oriented measures or brevity controls might be necessary.

ROUGE has become a mainstay for automatic summarization evaluation and is also applicable to other generation tasks needing content-coverage assessments. Although it provides a systematic approach to measuring n-gram and subsequence recall, users should be aware of its limited semantic sensitivity and reliance on reference coverage.

## 10.3.7 Perplexity

*Perplexity* [302] is a standard metric for evaluating the predictive power of language models, reflecting how "surprised" a model is by observed data. Formally, it measures the geometric mean of the inverse probabilities that a model assigns to each token in a sequence. Lower perplexity indicates greater confidence and accuracy in the model's token predictions, suggesting it captures language structure more effectively.

**Definition:** Let  $\mathbf{w} = (w_1, w_2, \dots, w_T)$  be a sequence of T words. A language model defines a conditional probability  $p(w_t \mid w_{1:t-1})$  for each word  $w_t$ . The *perplexity* of this model on the sequence  $\mathbf{w}$  is:

Perplexity(
$$\mathbf{w}$$
) = exp $\left(-\frac{1}{T}\sum_{t=1}^{T}\log p(w_t \mid w_{1:t-1})\right)$ . (222)

Equivalently, perplexity can be viewed as  $\exp(H)$ , where H is the cross-entropy of the model distribution relative to the empirical distribution of the words in  $\mathbf{w}$ .

## **Usage in NLP:**

- 1. **Language Modeling:** Perplexity is often the principal metric for *n-gram models* [312, 313], *RNN-based* [314], or *transformer-based* language models [315, 276]. A lower perplexity on a test set indicates that the model better predicts held-out text and thus better captures the syntactic/semantic structure of the language.
- 2. **Text Generation:** In chatbots and other generative systems, perplexity reflects how fluently or coherently the model can produce text. A model with lower perplexity generally generates fewer implausible token sequences and exhibits more context consistency.
- 3. **Embedded LM in Other Tasks:** Perplexity is also used in *speech recognition* or *machine translation* systems to evaluate the quality of the underlying language model. Better perplexity often correlates with improved final task performance, though it is not always strictly deterministic.

## **Advantages and Limitations:**

#### • Advantages:

- Direct Measure of Predictive Quality: Because perplexity is derived from probabilities assigned to the
  observed tokens, it succinctly captures how well the model "fits" the data.
- Broad Applicability: Perplexity is easy to compute for any language model that provides next-token probabilities.

#### • Limitations:

- Corpus/Domain Sensitivity: Perplexity scores can vary significantly across corpora with different vocabularies or styles. Comparisons are meaningful only if done on the same test set or similar domains.
- *Vocabulary Size Effects*: Increasing the vocabulary might inflate perplexity, as the model spreads probability mass over more tokens. Normalization or consistent vocabulary usage is essential for fair evaluation.
- Calibration Assumption: Perplexity presupposes well-calibrated probabilities. Overconfident or poorly
  calibrated models may yield deceptively low (or high) perplexities that do not translate to better generative
  performance.

*Perplexity* remains the cornerstone metric for language modeling, measuring the model's uncertainty about token sequences. Low perplexity typically indicates strong predictive capacity and alignment with linguistic regularities. Nevertheless, for certain tasks requiring sequence-level metrics (e.g., BLEU or ROUGE), perplexity should be complemented by evaluations more focused on semantic or global coherence aspects of text.

#### 10.3.8 Exact Match (EM)

Exact Match (EM) [316] is a binary evaluation metric particularly useful when the *entire* predicted output must match the reference output verbatim. It is widely adopted in tasks where minor deviations (such as rewording, word order, or formatting differences) invalidate the answer or reduce its utility.

**Definition:** Let there be N examples, each with a ground-truth output  $y_i$  and a predicted output  $\hat{y}_i$ . The EM metric assigns 1 if  $\hat{y}_i$  is *exactly* equal to  $y_i$  (token-for-token, character-for-character) and 0 otherwise:

Exact Match = 
$$\frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(y_i = \hat{y}_i), \qquad (223)$$

where  $\mathbb{I}(\cdot)$  is an indicator function returning 1 if the condition is true and 0 otherwise.

#### **Usage in NLP:**

- 1. Extractive Question Answering (QA) [316]: In tasks where the answer must be an exact substring (span) of a reference context, EM measures how frequently the predicted span perfectly matches the gold span. Even slight deviations—such as an omitted word—yield no credit.
- 2. **Natural Language Inference (NLI)**: When the model outputs one of a discrete set of labels (e.g., "entailment," "contradiction," "neutral"), EM checks if the predicted label precisely matches the correct label, making it a straightforward accuracy measure in a 3-class setting.
- 3. **Strictly Constrained Generation**: In specialized machine translation or summarization tasks (e.g., legal text or heavily format-dependent outputs), EM ensures the model's output reproduces an expected sequence exactly, leaving no room for paraphrase. Similarly, code generation or formula generation can rely on EM to verify syntactic correctness.
- 4. **Label Prediction with Immutable Formats**: If a classification label or keyword-based annotation must match a reference's exact format (e.g., a specific category name, diagnostic code), EM confirms total match.

#### **Limitations:**

- **Absolute Rigidity**: EM provides a score of 0 unless the prediction is an exact surface-form match, which can be overly strict in tasks where synonyms or slight re-phrasings are acceptable.
- **No Partial Credit**: The metric does not differentiate between predictions that are slightly off and those that are completely incorrect. Thus, it fails to capture near-correct attempts.
- Semantic Insensitivity: EM checks for literal equivalence and cannot assess deeper meaning. Tasks emphasizing semantic correctness (rather than exact wording) may find EM too unforgiving.

*Exact Match* is a reliable, straightforward metric in high-stakes or format-specific scenarios, such as extractive QA or code generation, where any discrepancy from the reference invalidates the model output. However, due to its binary nature and strict string matching, EM often requires supplementation with other metrics or human evaluation for tasks permitting equivalent, paraphrased, or partially correct responses.

#### 10.3.9 Word Error Rate (WER)

Word Error Rate (WER) is a standard metric for evaluating speech recognition or any system that produces a tokenized output (e.g., words) against a reference transcription. It quantifies the total number of substitutions, deletions, and insertions needed to transform the predicted word sequence into the reference sequence.

**Definition:** Let the reference be  $\mathbf{r} = (r_1, r_2, \dots, r_M)$  of length M, and the hypothesis (model output) be  $\mathbf{h} = (h_1, h_2, \dots, h_N)$  of length N. WER is computed as:

$$WER = \frac{S + D + I}{M},$$
(224)

where

- S is the number of substitutions (words in h that differ from reference words at matching positions),
- D is the number of deletions (words in r that have no corresponding word in h),
- *I* is the number of insertions (extra words in h not found in r).

Finding S, D, and I can be done via the Levenshtein (edit) distance algorithm, aligning words between the hypothesis and reference.

### **Usage in NLP:**

- Speech Recognition: The de facto metric for comparing automatically recognized transcripts to their ground-truth references. A lower WER indicates fewer word-level errors.
- Spoken Language Translation or Dictation Systems: Evaluating if the transcribed text in a target language or domain closely matches a reference script or user's intended words.

#### Advantages:

- Easy Interpretation: WER directly corresponds to a fraction of incorrect words, making it intuitive for end-users.
- Complete Overlap Check: Captures the count of exact lexical mismatches (substitutions, insertions, deletions).

#### **Limitations:**

- No Semantic Equivalence: Fails to give partial credit if a semantically correct but lexically different word is used.
- Same Penalty for All Errors: Substitutions vs. insertions/deletions are all included. This may not reflect user perceptions (some errors might be more severe).
- *Vocabulary and Domain Specificity*: Uncommon words or domain-specific terms can inflate WER if the model struggles with them.

WER is widely used to benchmark ASR (Automatic Speech Recognition) systems, but can be complemented with other measures like CER (Character Error Rate) or domain-specific metrics to capture subtleties of language use and user tolerance for certain error types.

# 10.3.10 Character Error Rate (CER)

*Character Error Rate (CER)* parallels WER but measures errors at the *character* (or sometimes subword) level rather than words. This can be especially relevant for languages with agglutinative morphology or for tasks involving scripts where word segmentation is less clear.

**Definition:** Given a reference string  $\mathbf{r}=(r_1,r_2,\ldots,r_P)$  of length P in characters and a hypothesis string  $\mathbf{h}=(h_1,h_2,\ldots,h_Q)$  of length Q, the CER is:

$$CER = \frac{S + D + I}{P},$$
(225)

where

- S is the number of substituted characters.
- D is the number of deleted characters,
- *I* is the number of inserted characters.

This again corresponds to the edit distance in characters, normalized by the reference length P.

## **Usage in NLP:**

- Speech Recognition for Character-Based Languages: When the system outputs characters directly (e.g., in Chinese or Japanese), CER can be more precise than WER because word boundaries might be ambiguous.
- ASR in Low-Resource Scripts or Morphologies: For languages where consistent word segmentation is not well-defined, character-level evaluation can be more stable.
- Handwriting Recognition: CER is also commonly used if the system outputs textual characters from an image-based input.

#### Advantages:

- More Granular Than WER: Reflects partial correctness in words if only some characters are off, offering finer error insight.
- Language Agnostic: Avoids complexities of word segmentation, especially valuable in languages with no whitespace or morphological complexity.

#### **Limitations:**

- *No Semantic Interpretation*: As with WER, CER penalizes any character difference equally, ignoring synonyms or small morphological variants with the same meaning.
- Not Always Reflecting Human-Readable Errors: A few character changes can drastically alter a word's meaning, but CER does not weigh severity of those changes differently.

CER is essential in scripts or domains where word-based segmentation is artificial or prone to errors. Together with WER, it provides complementary perspectives on the accuracy of textual outputs at different granularities.

# 11 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) [317] combines retrieval-based and generation-based NLP models to enrich tasks requiring substantial knowledge. By merging large language models (LLMs) with external knowledge sources, RAG enhances the accuracy, factuality, and relevance of generated content.

## **RAG Components:**

- 1. **Retrieval:** A dedicated neural retriever and dense index query external databases or knowledge bases. This step fetches relevant passages or records, allowing the system to incorporate up-to-date information. The retrieval mechanism is crucial for tasks like open-domain QA or domain-specific applications, where static model knowledge may be insufficient [317].
- 2. **Generation:** A pre-trained language model synthesizes coherent, context-aware text. It leverages both its internal learned representations and the retrieved external data to produce responses that are contextually aligned and factually supported.
- 3. **Augmentation:** The retrieved content is combined or appended to the model's input representation, enriching the model's generative process. This reduces hallucinations and outdated knowledge, since the LLM can refer to the retrieved material for accurate details and domain-specific terminologies.

RAG excels in tasks demanding broad or specialized knowledge—such as answering open-domain questions, drafting legal documents, or specialized advisory systems (e.g., TaxTajweez [318, 319]). By integrating external databases, RAG improves the precision and reliability of model outputs, crucial for applications where correctness is paramount. In addition, RAG can dynamically adapt to evolving knowledge, seamlessly incorporating new data over time [320].

#### 11.1 Loss Functions Used in RAG

Although *Retrieval-Augmented Generation (RAG)* systems can achieve strong performance without explicit fine-tuning—often by simply prompting a pre-trained large language model (LLM) to condition on retrieved information—certain applications demand deeper domain adaptation or more reliable retrieval. In such cases, fine-tuning allows the system to better integrate external knowledge and produce outputs aligned with specific domains, tasks, or user requirements. Below, we discuss common loss functions used *if* RAG requires fine-tuning, but many practical scenarios rely on carefully constructed prompts alone.

- 1. **Cross-Entropy Loss:** Typical for sequence-to-sequence training, improving the generation module's ability to produce correct target tokens. During fine-tuning, the model adjusts its vocabulary-level probabilitieto handle domain-specific terms betterms and reduce off-topic generation. Refer to §10.1.1.
- 2. **Contrastive Loss:** Applied when refining the retrieval module to differentiate relevant documents from irrelevant ones. By embedding queries and documents in a shared space, contrastive loss ensures higher similarity for relevant query-document pairs than for negative pairs. Refer to §7.1.7

- 3. **Marginal Ranking Loss:** Another retrieval-oriented objective that enforces a margin by which relevant documents must outrank irrelevant ones. This is valuable in large corpora scenarios, where fine-tuned ranking can significantly improve retrieved passages. Refer to §10.1.4.
- 4. **Negative Log-Likelihood (NLL):** Often used interchangeably with cross-entropy in sequence-to-sequence contexts. Fine-tuning on NLL encourages the model to assign higher probability mass to correct token sequences, beneficial for tasks requiring precise or domain-aware generation.
- 5. **KL Divergence Loss:** Employed in teacher-student or multi-component RAG settings to align probability distributions among sub-models. For instance, a teacher model's output can guide a student model to produce similar distributions for relevant passages or token probabilities. Refer to §9.1.2.

Although many RAG systems work effectively via *prompting only*, the option of fine-tuning remains critical for specialized domains or high-stakes use cases. By applying these losses—targeting either retrieval or generation—the fine-tuned RAG system learns to retrieve highly pertinent information and produce domain-specific text with enhanced accuracy and relevance.

#### 11.2 Performance Metrics Used in RAG

Evaluating Retrieval-Augmented Generation (RAG) systems poses unique challenges due to their *hybrid* structure—integrating both a retrieval mechanism (to fetch relevant external context) and a generation model (to produce final responses). This dual component design requires evaluating:

- Retrieval Quality: How accurately and precisely the system fetches relevant information from external knowledge sources.
- 2. **Generation Quality:** How the model uses the retrieved content to formulate coherent, contextually grounded, and factually correct outputs.

Conventional metrics (e.g., BLEU, ROUGE) often assume ground-truth references exist, but in many real-world RAG scenarios, explicit references or labeled answers may be unavailable [321, 322]. Evaluators thus turn to frameworks like RAGAS [323] and ARES [324], which propose reference-free or partially reference-based metrics to handle diverse RAG use cases (e.g., question answering, knowledge-grounded dialogue).

#### **Evaluation Considerations:**

- **Absence of Strict Reference Answers:** RAG tasks often address open-domain or knowledge-intensive queries for which no official ground truth is provided. Metrics must allow partial or reference-free evaluation and handle issues like model hallucination [321].
- Faithfulness to Retrieved Context: Ensuring that generated responses align with retrieved passages (i.e., do not hallucinate beyond the evidence) demands specialized metrics that cross-check model statements against the retrieved text [323].
- Relevance and Focus in Retrieval: The system should extract passages that are on-topic and concise, avoiding extraneous or off-topic data. Evaluating retrieval precision is essential to ensure the final generation is not contaminated by irrelevant or noisy content.

Table 21 summarizes the metrics employed to assess RAG systems, highlighting their benefits and drawbacks. In the following, we delve into the metrics provided by RAGAS and ARES.

#### 11.2.1 Answer Semantic Similarity

Answer Semantic Similarity measures how closely the model's generated answer aligns semantically with a reference or "ground-truth" answer. RAGAS [323] computes this score via a cross-encoder or embedding-based approach, typically using a cosine similarity:

SemanticSimilarity = 
$$\cos(\mathbf{E}_q, \mathbf{E}_t)$$
, (226)

where

•  $\mathbf{E}_q$  is the embedding (vector) of the generated answer,

Table 21: Guidelines for selecting a performance metric in RAG-based systems, based on usage, data characteristics, advantages, and limitations.

Function	Usage	Data Characteristics	Advantages	Limitations
answer semantic similarity	Checks semantic closeness to reference	Embedding or cross- encoder needs ground-truth	Captures meaning not just words	No guarantee of fac- tual accuracy depends on embed- ding quality
answer correct- ness	Combines token F1 and semantic similarity	Requires reference answer merges lexical + semantic checks	Penalizes missing tokens rewards accurate rephrasing	Sensitive to small token differences weighting factor needed
answer relevance	Rates topical alignment with the question	Embedding-based "reverse queries" does not test correctness	Detects off-topic or redundant text	Relevant yet possibly incorrect relies on LLM to gen- erate reverse queries
context precision	Emphasizes high-rank relevance	Binary relevant/irrelevant labeling top K chunks	Ensures key evidence is top-ranked	Ignores recall no coverage guarantee
context re- call	Checks if retrieval covers correct answer	Matches ground-truth statements ratio of "supported" content	Measures completeness of retrieval	No direct link to generation partial coverage untracked
faithfulness	Ensures generated text is grounded in context	Splits output into claims verifies each	Reduces hallucination crucial for factual tasks	Needs reliable claim- checking does not measure rele- vance
summarizati score	oiBalances coverage vs. brevity	QA on key points + length ratio	Encourages concise, accurate summaries	Relies on question generation synonyms might lower QA score
context entities recall	Ensures key named entities are preserved	Requires entity detection in both context and output	Good for fact-focused tasks	Fails on synonyms depends on NER accu- racy
aspect cri- tique	Binary checks for custom aspects (e.g., harmlessness)	Repeated LLM prompts for yes/no verdict	Fast content modera- tion flexible aspects	Subject to LLM bias yes/no may be simplistic
context relevance (ARES)	Classifies passage- query pairs for alignment	Contrastive training with positives/negatives	Improves retrieval targeting uses confidence intervals	Discrete notion of rel- evance depends on negative sample quality
answer faith- fulness (ARES)	Tests if answer is faithful to retrieved passages	Synthetic faith- ful/unfaithful sets contrastive approach	Reduces hallucination fosters context-based correctness	Needs good example generation subtle contradictions may slip
answer relevance (ARES)	Rates if response addresses the query	Pos/neg samples another LM judge	Ensures query alignment complements correctness	May miss factual errors irrelevant details can be true

- $\mathbf{E}_t$  is the embedding of the target or reference answer,
- $\cos(\cdot)$  is the cosine similarity function, yielding values in [-1,1].

In practice, the score is often scaled or clipped to lie in [0,1]. Higher values indicate closer semantic overlap, even if the surface wording differs. This metric helps reveal whether the model captures the essential meaning of an answer, rather than just lexical matches.

#### 11.2.2 Answer Correctness

Answer Correctness assesses both the factual accuracy and semantic coherence of the response, combining a token-level F1 measure with the previously described semantic similarity. Formally, RAGAS [323, 325] defines:

AnswerCorrectness = 
$$\alpha \times F1 + (1 - \alpha) \times SS$$
, (227)

where:

- SS is the semantic similarity from Equation (226),
- F1 is a token-level F1 score, capturing overlap in terms of lexical correctness,
- $\alpha \in [0,1]$  is a weighting factor controlling the relative importance of F1 vs. semantic similarity.

The token-level F1 score F1 is computed as:

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}},$$
(228)

where  $\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$  and  $\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}},$  with:

- TP (true positives): tokens in the generated answer matching those in the ground truth,
- FP (false positives): extra or incorrect tokens in the generated answer absent in the ground truth,
- FN (false negatives): tokens in the ground truth that the model's answer omitted.

By combining a token-level overlap check (F1) with an embedding-based similarity check (SS), the *Answer Correctness* metric aims to penalize missing or incorrect tokens while rewarding semantically accurate rephrasings. A higher score signals a more comprehensive and faithful response to the question or query.

#### 11.2.3 Answer Relevance

Answer Relevance [326] quantifies how well the generated answer addresses the posed question in terms of topicality and completeness, independently of factual accuracy. A relevant answer should not be redundant or off-topic, and it should align with the question's intent.

**Definition:** Let Q be the original question, and let A be the model's generated answer. Using a language model, we generate N "reverse-engineered" queries  $\{Q_{g_1}, Q_{g_2}, \ldots, Q_{g_N}\}$  by prompting the model to produce plausible questions that could lead to A. Each  $Q_{g_i}$  is then embedded into a vector space (e.g., via a sentence encoder) to obtain  $E_{g_i}$ . The original question Q is similarly embedded as  $E_o$ . The *Answer Relevance* is computed by averaging the cosine similarities:

AnswerRelevance = 
$$\frac{1}{N} \sum_{i=1}^{N} \cos(\mathbf{E}_{g_i}, \mathbf{E}_o),$$
 (229)

where  $\cos(\cdot, \cdot)$  is the cosine similarity function. A higher value indicates that, on average, the "reverse-engineered" queries resemble the original question in embedding space, suggesting the generated answer remains close to the question's topic.

## **Interpretation:**

- *Completeness vs. Redundancy:* Answers failing to address the question or containing extraneous text typically yield lower cosine similarity to the original query.
- No Accuracy Guarantee: Since this metric does not incorporate factual correctness, an answer can be "relevant" yet factually incorrect.

Thus, Answer Relevance captures how directly the system's output responds to the question's core intent, leaving exact correctness to other metrics (§11.2.2, etc.).

#### 11.2.4 Context Precision

Context Precision [323, 327] evaluates how well a system ranks relevant context items at the top among all retrieved passages or chunks. The rationale is that truly pertinent evidence should appear in the highest ranks, maximizing the precision in early positions.

**Score Formulation:** Consider a set of K retrieved context chunks  $\{c_1, \ldots, c_K\}$  sorted by relevance score. Let  $v_k$  be 1 if  $c_k$  is relevant to the question and 0 otherwise. Define Precision@k as:

$$Precision@k = \frac{TP@k}{TP@k + FP@k},$$
(230)

where TP@k is the count of true positives in the top k ranks (i.e., relevant chunks among the first k), and FP@k is the count of false positives. The *Context Precision* at top K chunks, CP@K, is then:

$$CP@K = \frac{\sum_{k=1}^{K} (Precision@k \times v_k)}{Total \text{ relevant items in top } K}.$$
 (231)

A higher CP@K indicates that relevant chunks consistently appear at the top. Unlike simple precision, CP@K weights each rank k by whether the chunk at that rank is truly relevant.

## **Interpretation:**

- *Emphasis on Early Ranks:* Critical in RAG systems where only top *K* chunks feed into the generation step. Poorly ranked relevant chunks effectively vanish if not included in the final context.
- No Recall Component: A system can have high context precision but fail to retrieve all relevant chunks if those items are missing from the top K set altogether.

Context Precision thus provides a lens on how well the retrieval module prioritizes relevant evidence in the top positions crucial for the subsequent generation.

#### 11.2.5 Context Recall

*Context Recall* [323, 328] measures the extent to which the retrieved context supports the correct (ground truth) answer. Intuitively, all statements within the reference answer should be attributable to at least some portion of the retrieved text. The higher the ratio of "attributable" statements, the better the context coverage. Mathematically:

$$Context Recall = \frac{|GT_{sentences attributed}|}{|GT_{total sentences}|},$$
(232)

where  $GT_{\rm sentences\ attributed}$  are the ground-truth answer's sentences that can be linked to (or found in) the retrieved context, and  $GT_{\rm total\ sentences}$  is the total number of sentences in the ground-truth answer. A higher Context Recall means the system retrieved enough relevant passages so that each statement in the correct answer is well-supported by the retrieved material.

### **Interpretation:**

- Completeness of Retrieval: A perfect score (1.0) indicates that every statement from the ground truth has corresponding evidence in the retrieved text.
- *Ignoring Generation Quality:* Context Recall only evaluates retrieval completeness, not whether the final generated answer is accurate. A system may retrieve relevant data yet generate an incorrect response (or vice versa).

### 11.2.6 Faithfulness

Faithfulness [323, 329] evaluates whether all statements in the generated response follow logically from the given context. A higher faithfulness score implies that the model does not fabricate or "hallucinate" details absent in the provided context. Each statement in the generated output is checked against the retrieval context for verifiability.

Score Definition: Let the generated response contain  $C_t$  total claims (statements). Out of these,  $C_s$  are directly supported by the retrieved context. Then:

Faithfulness Score = 
$$\frac{C_s}{C_t}$$
, (233)

where Faithfulness Score  $\in [0, 1]$ . A score of 1.0 indicates every statement in the response is context-grounded.

## **Interpretation:**

- Focus on Source-Consistency: If a statement is unsupported or contradicts the context, it lowers  $C_s$ . This metric is crucial for applications requiring factual accuracy tied to explicit references (e.g., legal or medical).
- *Domain Constraints:* In specialized fields, verifying faithfulness may require deeper semantic checks or external knowledge that goes beyond raw text matching.

#### 11.2.7 Summarization Score

Summarization Score [323, 330] determines how effectively a generated summary captures the essential information from the original text while encouraging brevity. It operates by generating questions from key phrases in the context and then verifying if the summary answers these questions correctly.

## **Two Key Components:**

1. **QA Score:** Let the system produce N questions based on important points in the context. For each question, the summary either answers it correctly (1) or incorrectly (0). Then,

$$QA Score = \frac{Correctly Answered Questions}{Total Questions}.$$
 (234)

2. **Conciseness Score:** To prevent trivial success by copying large swaths of text, a conciseness factor penalizes excessively long summaries:

$$Conciseness Score = \frac{Length(Summary)}{Length(Context)}.$$
 (235)

Summaries that replicate the entire context might answer all questions but yield a high ratio in Equation (235), which indicates poor conciseness.

## **Final Summarization Score:**

$$Sum Score = \frac{QA Score + Conciseness Score}{2}.$$
 (236)

## **Interpretation:**

- Balancing Completeness and Brevity: This metric rewards precise, minimal summaries that still answer key questions from the context.
- *Limitations:* Summaries may be semantically correct but use synonyms that do not directly answer enumerated questions, or the question-generation module might produce suboptimal queries. Nonetheless, this approach ensures some measure of coverage *and* conciseness.

### 11.2.8 Context Entities Recall

Context Entities Recall [331] quantifies how effectively a system retrieves or maintains the entities from the original context. It is particularly relevant in fact-intensive tasks (e.g., tourism info, historical QA), where omitting certain named entities (cities, dates, persons) can degrade the overall usefulness of the generated response.

**Definition:** Let Entities<sub>context</sub> be the set of named entities (or key domain-specific entities) identified in the reference context, and let Entities<sub>generated</sub> be the entities identified in the system's generated text. The metric:

$$CER = \frac{|Entities_{context} \cap Entities_{generated}|}{|Entities_{context}|},$$
(237)

yields a ratio in [0,1]. A higher value means more contextually significant entities from the original text reappear in the generated output.

### **Interpretation and Use:**

- *Entity-Focused Scenarios:* Ideal for verifying that essential names, places, or terms from the context remain present in the final generation.
- Factual Adequacy: While not a direct measure of correctness, a low CER might hint that crucial references were lost or omitted.

#### **Limitations:**

- Entity Identification Errors: Dependence on NER or entity extraction might produce inaccuracies if either the reference or generation's entity recognizer is imperfect.
- *No Semantic Variation:* If the system uses synonyms or paraphrases for certain terms (instead of direct mentions), such entities might not match exactly.

# 11.2.9 Aspect Critique

Aspect Critique is designed to evaluate submissions (model outputs) against a set of predefined or custom aspects (e.g., correctness, harmfulness) [323]. Each aspect yields a binary decision indicating whether the submission meets (Yes) or violates (No) the criterion.

## **Procedure:**

- 1. **Aspect Definition:** Users specify the aspect or dimension of interest (e.g., "harmlessness," "correctness," or any custom criterion).
- 2. **LLM-based Verification:** The system issues multiple prompts (often 2 or 3) to a Large Language Model (LLM), asking if the submission meets the aspect requirement. For instance, a *harmfulness* aspect might ask: "Does the submission cause or risk harm to individuals, groups, or society?"
- 3. **Majority Verdict:** If the LLM returns "No" (harmless) in 2 out of 3 calls, the final aspect critique is *No Harm*. If it returns "Yes" in the majority, the verdict is "Harmful." A *strictness* parameter (2 to 4) can regulate the level of confidence or self-consistency the model must display to finalize the verdict.

# **Interpretation:**

- Binary Feedback: The output is simply Yes or No for each aspect, offering a direct measure of compliance.
- *Modular Aspects:* Ragas Critiques or custom scripts can add or remove aspects (such as bias, factual correctness, etc.), tailoring the critique to domain requirements.

## **Limitations:**

- Reliance on LLM Consistency: Because the method re-prompts an LLM to judge the submission, it is subject to the model's own biases and variability.
- Subjective or Complex Aspects: Certain aspects (like "correctness" in specialized domains) may require deeper validation than a short textual query can offer.

## **Use Cases:**

• Content Moderation: The "harmfulness" aspect can identify harmful, offensive, or risky content automatically.

• Regulatory or Ethical Checklists: In finance or healthcare domains, aspects such as "compliance with policy" can be verified via aspect critiques before final deployment.

ARES aims to improve the precision and accuracy of evaluating RAG systems by using a combination of lightweight language model (LM) judges and prediction-powered inference (PPI) techniques. The framework generates its own synthetic data and uses them to fine-tune models that can assess the quality of individual RAG components. This approach allows ARES to offer automated, efficient, and domain-adaptive evaluation methods, which require minimal human annotations. ARES can be found at https://github.com/stanford-futuredata/ARES.

The three metrics provided by ARES are explained below.

#### 11.2.10 Context Relevance

Context Relevance assesses whether the information retrieved by the system is pertinent to the given query. In the ARES framework, context relevance is determined by fine-tuning a lightweight language model to classify query-passage pairs as either relevant or irrelevant. The process involves generating synthetic datasets of question-answer pairs from the corpus passages, which include both positive examples (relevant passages) and negative examples (irrelevant passages). The model is trained using a contrastive learning objective, which helps it distinguish between relevant and irrelevant contexts. The effectiveness of the context relevance metric is enhanced by using prediction-powered inference (PPI) to provide statistical confidence intervals, ensuring the accuracy of the evaluation [324].

The process to compute Context Relevance is the following:

- 1. Synthetic Dataset Generation: The process begins with the generation of synthetic datasets. A large language model (LLM) is used to create synthetic question-answer pairs from the corpus passages. This involves generating both positive examples (where the passage is relevant to the query) and negative examples (where the passage is irrelevant).
- 2. Fine-Tuning LLM Judges: A lightweight language model, specifically DeBERTa-v3-Large [332], is fine-tuned to serve as a judge for context relevance. This model is trained using a contrastive learning objective to classify query-passage pairs as relevant or irrelevant. The training leverages the synthetic dataset, which includes both positive and negative examples.
- 3. Negative Example Generation: To enhance the training process, two strategies are employed to generate negative examples. Weak Negatives are randomly sampled passages that are unrelated to the synthetic query. Strong Negatives are passages from the same document as the relevant passage or similar passages retrieved using BM25, but they do not directly answer the query.
- 4. Evaluation and Scoring: The fine-tuned LLM judge evaluates the context relevance of query-passage-answer triples from various RAG systems. The model assigns scores based on its classification of each triple as relevant or irrelevant.
- 5. Prediction-Powered Inference (PPI): To improve the accuracy of the evaluation and provide statistical confidence intervals, ARES uses PPI. This involves using a small set of human-annotated datapoints to adjust the predictions made by the LLM judge. PPI helps in generating confidence intervals for the context relevance scores, ensuring that the evaluations are robust and reliable.

#### 11.2.11 Answer faithfulness

Answer faithfulness measures whether the generated response is grounded in the retrieved context, without introducing hallucinated or extrapolated information. This metric is useful to ensure that the answers provided by the RAG system are accurate and reliable. In ARES, answer faithfulness is evaluated by fine-tuning a separate LM judge to classify query-passage-answer triples as either faithful or unfaithful. Training involves creating synthetic datasets with both faithful answers (correctly grounded in the retrieved passages) and unfaithful answers (containing hallucinated or contradictory information). The model learns to identify discrepancies between the generated answer and the retrieved context. As with context relevance, PPI is used to enhance the accuracy of the faithfulness evaluation by leveraging a small set of human-annotated datapoints to generate confidence intervals.

The process to compute Answer faithfulness follows the same strategy as Context Relevance. Through the following steps, ARES measures answer faithfulness, ensuring that the generated answers are accurately grounded in the retrieved context.

1. Synthetic Dataset Creation: The process begins with generating synthetic datasets that include both faithful and unfaithful examples. This involves using a large language model (LLM) to generate question-answer pairs

- based on passages from a corpus. The synthetic dataset includes positive examples where the answer is faithful to the retrieved passage and negative examples where the answer is not faithful.
- 2. Fine-Tuning LLM Judges: A lightweight language model, specifically DeBERTa-v3-Large [332], is fine-tuned to serve as a judge for answer faithfulness. The model is trained to classify query-passage-answer triples as either faithful or unfaithful using a contrastive learning objective. This training helps the model learn to identify when a generated answer is grounded in the retrieved passage without introducing hallucinated or contradictory information.
- 3. Negative Example Generation: To enhance the training process, two strategies are used to generate negative examples. Weak Negatives are randomly sampled answers from other passages that are not related to the given query-passage pair and Strong Negatives are contradictory answers generated by prompting the LLM with few-shot examples to produce answers that conflict with the information in the passage.
- 4. Evaluation and Scoring: The fine-tuned LLM judge evaluates the answer faithfulness of query-passage-answer triples produced by various RAG systems. The model assigns scores based on its classification of each triple as faithful or unfaithful.
- 5. Prediction-Powered Inference (PPI): Same as with Context Relevance. This step is used to improve the accuracy of the evaluation and provide statistical confidence intervals, ARES uses PPI. This involves using a small set of human-annotated datapoints to adjust the predictions made by the LLM judge.

#### 11.2.12 Answer Relevance

Answer relevance assesses whether the generated response is relevant to the original query, considering both the retrieved context and the query itself. This metric ensures that the answer not only aligns with the retrieved information but also addresses the query effectively. In the ARES framework, answer relevance is evaluated by fine-tuning another LM judge to classify query-passage-answer triples as either relevant or irrelevant. The training process involves generating synthetic datasets with examples of both relevant and irrelevant answers, allowing the model to learn the nuances of relevance in the context of the query and the retrieved passage. PPI is again employed to provide confidence intervals, improving the reliability of the relevance evaluation by combining the predictions from the LM judge with a small set of human annotations.

The process to measure answer relevance is similar to the previous ARES metrics. Through these steps, ARES ensures that the generated answers not only align with the retrieved information but also effectively address the original query.

- 1. Synthetic Dataset Creation: Similar to the other metrics, the process begins with generating synthetic datasets. A large language model (LLM) is used to create synthetic question-answer pairs from the corpus passages. This dataset includes both positive examples (where the answer is relevant to the query and retrieved passage) and negative examples (where the answer is irrelevant).
- 2. Fine-Tuning LLM Judges: A lightweight language model, such as DeBERTa-v3-Large [332], is fine-tuned to act as a judge for answer relevance. The model is trained to classify query-passage-answer triples as either relevant or irrelevant using a contrastive learning objective. This training helps the model learn to identify when a generated answer appropriately addresses the query using information from the retrieved passage.
- 3. Negative Example Generation: To enhance the training process, two types of negative examples are generated. Weak Negatives are randomly sampled answers from other passages that are not related to the given query-passage pair. Strong Negatives are answers generated by prompting the LLM to produce responses that do not address the query or are contradictory to the retrieved passage.
- Evaluation and Scoring: The fine-tuned LLM judge evaluates the answer relevance of query-passage-answer
  triples produced by various RAG systems. The model assigns scores based on its classification of each triple
  as relevant or irrelevant.
- 5. Prediction-Powered Inference (PPI): To improve the accuracy of the evaluation and provide statistical confidence intervals, ARES uses PPI using a small set of human-annotated datapoints to adjust the predictions made by the LLM judge. PPI helps in generating confidence intervals for the answer relevance scores, ensuring that the evaluations are robust and reliable.

## 12 Combining Multiple Loss Functions

A practical strategy in deep learning is the use of *multi-loss setups*, wherein two or more loss functions are combined—often in a weighted sum—to capture different facets of a complex task. This approach allows a model to optimize multiple objectives simultaneously, often improving final performance [333, 334]. Below are several exemplary cases:

- Localization vs. Classification in Object Detection. In object detection architectures such as YOLO [120] or Faster R-CNN [117], a regression loss (e.g., Smooth  $\ell_1$  [115]) is used for bounding-box coordinates, while a classification loss (e.g., Cross-Entropy) handles object labels. Training on both ensures accurate box localization *and* robust classification, thereby balancing spatial accuracy with semantic correctness.
- Generative Adversarial Networks (GANs). GANs typically rely on an *adversarial* loss (standard [222] or Wasserstein [258]) to push generated samples to be indistinguishable from real data. However, to control additional properties (e.g., content consistency, style, or sharper details), auxiliary terms such as pixel-level or perceptual losses [253, 244] may be introduced. This can help avoid mode collapse and improve the visual fidelity of generated images, as each component of the total loss addresses distinct aspects of image realism.
- Language Modeling with Additional Constraints. In summarization or machine translation, a standard token-level Cross-Entropy loss may be paired with a sequence-level loss (e.g., Minimum Risk Training [297] or REINFORCE [300]) that directly optimizes global quality metrics such as BLEU [298] or ROUGE [299]. Balancing token-by-token accuracy with entire-sequence objectives often yields more coherent and context-aware outputs, reducing exposure bias [335].
- Perceptual + Reconstruction Loss in Image-to-Image Tasks. In tasks like super-resolution or style transfer, a
  reconstruction loss (e.g., pixel-level MSE) may be combined with a perceptual loss computed via intermediate
  features of a pretrained CNN [263]. This ensures both low-level fidelity (sharp edges, stable colors) and
  high-level feature preservation (texture consistency, semantic alignment), ultimately leading to more realistic
  results.

A common method to combine losses is to form a weighted sum:

$$L_{\text{total}} = \lambda_1 L_1 + \lambda_2 L_2 + \dots + \lambda_k L_k, \tag{238}$$

where  $\lambda_i$  are nonnegative weighting factors for each objective  $L_i$ . The assignment of these weights remains an important challenge. On the one hand, i) if an auxiliary loss is overemphasized, it can overshadow the primary objective. On the other hand, ii) if underemphasized, it offers little benefit [334, 336]. Researchers often tune these hyperparameters empirically or employ heuristic methods such as gradient normalization (GradNorm) [336] or uncertainty weighting [334] to manage them automatically.

**Synergistic Effects of Multi-Loss Learning.** In many cases, combining carefully selected loss terms can produce synergy by tackling complementary aspects of a task. For example, in image segmentation, cross-entropy might be integrated with a loss based on Dice or IoU [337, 178] to handle class imbalance and emphasize overlapping regions. Similarly, in retrieval-augmented generation (RAG) [317], pairing a contrastive or ranking loss for the retrieval component with a sequence modeling loss for text generation enables the system to more effectively leverage external knowledge and produce high-fidelity responses.

Challenges and Future Directions. While multi-loss sets often improve performance, they can also introduce training instability if the losses have conflicting gradients [338]. In addition, too many objectives can lead to loss competition, where improvements in one objective degrade another. To address these issues, recent works study adaptive reweighting strategies (e.g., dynamic uncertainty weighting [334], GradNorm [336]) or multi-objective optimization techniques [339], allowing the model to balance objectives in a principled way. Looking forward, automated methods to discover or tune weights for multi-loss training are a promising avenue [340, 341].

Multi-loss frameworks are now ubiquitous across a variety of deep learning applications. As tasks become more complex and multifaceted, designing or automating optimal loss combinations will likely remain a key area of research. Incorporating task-specific constraints, domain knowledge, and trade-offs among different objectives will be essential to push the boundaries of performance and reliability in deep learning systems.

## 13 Challenges and Trends

In the past decade, deep learning has significantly advanced fields like computer vision [71, 342, 60], natural language processing [283, 276, 315], reinforcement learning [343], and generative modeling [222, 217]. These advancements pose various challenges for researchers and practitioners. This subsection outlines these challenges, how current loss functions and metrics address them, and the trends influencing deep learning's future.

## 13.1 Challenges in Deep Learning

## **Data Limitations and Imbalances**

Large amounts of high-quality labeled data are necessary for training deep learning models. In many practical scenarios, data are expensive to annotate or the classes are highly imbalanced. Traditional loss functions such as cross-entropy (§3.1.1) or MSE (§2.1.1) can struggle in these scenarios, because they treat every instance equally, potentially causing overfitting on the majority classes [60]. Methods like Weighted Binary Cross-Entropy (§3.1.4) and Focal Loss (§5.1.6) specifically address these issues by assigning higher weight to harder or minority samples, improving performance on imbalanced datasets.

## Sensitivity to Outliers and Noise

Standard losses such as (MSE) are known to be sensitive to outliers or noisy data, disproportionately penalizing large errors. This could lead to compromised performance on tasks with heavy-tailed error distributions, such as time series forecasting [25] or in real-time sensor data analyses. Robust alternatives, such as Huber Loss (§2.1.3) or Smooth L1 (§5.1.1), can mitigate these issues by transitioning between the squared and absolute error regimes based on a threshold.

## **Exposure Bias and Sequence-Level Learning**

In natural language generation tasks, models trained with next-token prediction suffer from exposure bias since, at inference time, they generate tokens sequentially but were trained to predict one token at a time using the ground-truth prefix [335]. This mismatch can lead to the accumulation of errors and degradation of sequence-level performance. Methods like Minimum Risk Training (§10.2.2) and REINFORCE (§10.2.3) address this issue by optimizing metrics such as BLEU (§10.3.4) or ROUGE (§10.3.6) directly, bridging the gap between training and inference conditions.

### **Complex Evaluation Requirements**

As tasks become more sophisticated (e.g. multi-object detection [120, 117], panoptic segmentation [139], retrieval-augmented generation [317]), a single metric or loss measure is insufficient. Models that are state-of-the-art in one metric might fail in others. This complexity demands task-specific or multi-faceted metrics that capture different quality dimensions (e.g., faithfulness (§11.2.6) vs. relevance (§11.2.3), bounding-box precision (§5.1.3) vs. recall §3.2.5).

# 13.2 Addressing Challenges

#### Addressing Class Imbalance and Data Scarcity

Loss functions like *Focal Loss* (§5.1.6) and *Weighted Cross-Entropy* (§3.1.4) emphasize minority classes by assigning higher loss to misclassified examples or underrepresented classes. Similarly, in retrieval-augmented generation, contrastive (§7.1.7) or ranking-based losses (§10.1.4) guide the system to focus on genuinely relevant documents from vast corpora.

## **Robustness to Outliers**

When dealing with outliers in applications like healthcare [149], robust losses such as Huber (§2.1.3), Smooth L1 (§5.1.1), or Log-Cosh (§2.1.4) are essential. These losses are less sensitive to extreme values, helping maintain stability and reducing the tendency to overfit.

## **Sequence-Level Training and Global Metrics**

Optimizing only at the token level often fails to capture global sequence properties. Metrics like BLEU (§10.3.4), ROUGE (§10.3.6), and specialized ones like METEOR (§10.3.5) can be integrated via Minimum Risk Training (§10.2.2) or Reinforcement Learning [300, 335], driving improvements in fluency and semantic alignment.

### Multi-Task and Multi-Loss Objectives

Modern deep learning tasks increasingly require balancing multiple objectives [334]. For instance, in object detection (bounding-box regression + classification), models combine smooth L1 or IoU-based losses for localization [121] with cross-entropy losses for classification. Meanwhile, generative models can integrate *adversarial* losses [222] alongside reconstruction or perceptual losses [253, 244] to achieve realistic and content-preserving outputs.

## 13.3 Trends and Future Directions

#### **Automated Loss Function Search**

Researchers have begun exploring methods to automatically discover or design loss functions ("Loss Function Search"), aiming to reduce the reliance on trial and error. Using gradient-based meta-learning or evolutionary algorithms, these approaches can search a space of differentiable loss operators [344, 345] to find the most effective for a given dataset or task.

## Task-Adaptive and Data-Adaptive Metrics

Generic metrics such as MSE or Cross-Entropy can be suboptimal when they do not align with real-world goals. Emerging methods adapt losses and evaluation metrics to specific objectives: for example, survival analysis in healthcare requires specialized metrics for censored data [346], while retrieval-augmented generation employs metrics like *Answer Faithfulness* (§11.2.6) or *Context Relevance* (§11.2.10) for real-world utility and factual correctness.

## **Continual and Online Learning Scenarios**

In many dynamic settings (e.g., streaming data [171], or rapidly changing domains), metrics and losses must adapt to evolving data distributions [347]. Online surrogates of cross-entropy or adversarial objectives have been proposed, while incremental evaluation protocols address shifts in data distribution without complete retraining.

### **Robustness and Safety in Deployment**

As deep models are deployed in safety-critical areas like medical imaging [149], autonomous driving [100], or legal advisory [318], new evaluation metrics (fairness, interpretability, reliability) and novel losses that penalize harmful behaviors are gaining prominence [348, 349]. Tailored metrics like *Answer Correctness* or *Answer Faithfulness* [323, 324] are already emerging in retrieval-augmented systems.

#### **Enhanced Interpretability and Explainability**

Future loss functions and metrics may incorporate interpretability constraints, ensuring that model predictions can be more transparent [350]. This is particularly relevant in retrieval-augmented generation [317], where the chain-of-thought or evidence path can be traced back to external documents.

### Large Language Models and Prompt-Engineering Paradigms

As large-scale language models [276, 315] become prevalent, new metrics are needed to evaluate context usage, reasoning chains, or prompt understanding. We foresee research into specialized loss functions that penalize factual inconsistencies or hallucinations, a rising concern in modern LLM-based applications [323, 324].

Deep learning loss functions and metrics are evolving to address challenges like imbalanced data, outliers, sequence-level training, and multi-objective optimization. Trends focus on robust, context-aware, application-specific objectives, highlighting the field's diversity and complexity. Innovations in task-adaptive losses and advanced evaluation frameworks offer reliable, interpretable, domain-specific solutions for future applications.

## 14 Conclusion

Loss functions and performance metrics lie at the core of effective deep learning. As this review demonstrates, no single approach applies universally across tasks: regression, classification, computer vision (for instance, object detection, face recognition, depth estimation, image generation), and NLP all require careful tailoring of the loss function and evaluation metric. The varying data characteristics, particularly in scenarios with class imbalance or noisy labels, emphasize the critical role of selecting or designing losses that align with the problem at hand.

Beyond conventional choices such as cross-entropy or MSE, specialized losses (e.g., focal loss, smooth L1, adversarial objectives) and advanced evaluation metrics (e.g., AP/AR, IoU-based measures, BLEU, ROUGE) have emerged to address domain-specific challenges. Moreover, multi-loss setups, in which two or more objectives are combined, often produce richer learning signals as long as weighting factors are tuned carefully or adapted to balance competing goals.

Looking ahead, several directions can strengthen our use of loss functions and metrics in deep learning. One key area is the automation of loss and metric selection through search algorithms or meta-learning techniques, thereby

reducing human trial and error. Another involves designing robust and interpretable objectives that remain stable in the presence of noise or domain shifts while offering more transparent optimization signals. There is also growing interest in task-adaptive metrics that align more closely with practical objectives, such as fairness, explainability, or retrieval faithfulness, rather than relying exclusively on generic measures. Finally, hybrid approaches will need to be extended for emerging tasks such as retrieval-augmented generation, where both retrieval accuracy and generative fidelity are crucial.

By advancing these avenues, researchers and practitioners will obtain stronger, more reliable deep learning models. In particular, better comprehension of losses and metrics will help overcome challenges associated with complex tasks, ultimately pushing modern AI systems to remain high-performing and robust in a rapidly evolving field.

# 15 Acknowledgments

We thank the National Council for Science and Technology (CONACYT) for its support through the National Research System (SNI).

# Declaration of generative AI and AI-assisted technologies in the writing process

We acknowledge the use of three AI tools: Grammarly Assistant to improve the grammar, clarity, and overall readability of the manuscript, Perplexity for finding relevant academic works, and GPT-40 to help with the wording and proofreading of the manuscript.

## References

- [1] J. Smith, "Deep learning for image recognition," *Journal of Artificial Intelligence*, vol. 15, no. 2, pp. 78–95, 2018.
- [2] E. Jones, "Object detection using neural networks," Computer Vision Review, vol. 8, no. 4, pp. 112–128, 2020.
- [3] M. Gonzalez, "Image segmentation using neural networks," *Pattern Recognition*, vol. 22, no. 1, pp. 36–52, 2017.
- [4] L. Wang, "Facial recognition using neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 6, pp. 1100–1112, 2019.
- [5] W. Chen, "Image generation using adversarial neural networks," ACM Transactions on Graphics, 2021.
- [6] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2414–2423, 2016.
- [7] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017.
- [8] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *International Conference on Learning Representations (ICLR)*, 2018.
- [9] L. Chen, "Automatic speech recognition with neural networks," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 6, pp. 1200–1212, 2019.
- [10] W. Liu, "Speech emotion recognition with neural networks," *IEEE Transactions on Affective Computing*, vol. 11, no. 3, pp. 500–512, 2020.
- [11] M. Kim, "Multilingual speech recognition with neural networks," *Speech Communication*, vol. 101, pp. 50–63, 2018.
- [12] P. Zhang, "Robust speech recognition with neural networks," *Computer Speech and Language*, vol. 65, pp. 101–120, 2021.
- [13] H. Wu, "End-to-end speech recognition with neural networks," *IEEE Signal Processing Letters*, vol. 24, no. 11, pp. 1631–1635, 2017.
- [14] J. Smith, "Sentiment analysis using natural language processing," *Journal of Artificial Intelligence*, vol. 15, no. 2, pp. 78–95, 2018.
- [15] E. Jones, "Named entity recognition using natural language processing," *Computational Linguistics*, vol. 46, no. 4, pp. 112–128, 2020.
- [16] M. Gonzalez, "Text summarization using natural language processing," *ACM Transactions on Information Systems*, vol. 35, no. 2, pp. 36–52, 2017.

- [17] L. Wang, "Part-of-speech tagging using natural language processing," *Journal of Machine Learning Research*, vol. 20, no. 7, pp. 1100–1112, 2019.
- [18] W. Chen, "Question answering using natural language processing," *IEEE Transactions on Knowledge and Data Engineering*, 2021.
- [19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 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.
- [20] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., "{TensorFlow}: a system for {Large-Scale} machine learning," in 12th USENIX symposium on operating systems design and implementation (OSDI 16), pp. 265–283, 2016.
- [21] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.
- [22] The MathWorks Inc., "Matlab," 2024.
- [23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.
- [24] D. Harrison Jr and D. L. Rubinfeld, "Hedonic housing prices and the demand for clean air," *Journal of environmental economics and management*, vol. 5, no. 1, pp. 81–102, 1978.
- [25] H.-x. Zhao and F. Magoulès, "A review on the prediction of building energy consumption," *Renewable and Sustainable Energy Reviews*, vol. 16, no. 6, pp. 3586–3592, 2012.
- [26] F. E. Harrell Jr, K. L. Lee, R. M. Califf, D. B. Pryor, and R. A. Rosati, "Regression modelling strategies for improved prognostic prediction," *Statistics in medicine*, vol. 3, no. 2, pp. 143–152, 1984.
- [27] S. Shen, H. Jiang, and T. Zhang, "Stock market forecasting using machine learning algorithms," *Department of Electrical Engineering, Stanford University, Stanford, CA*, pp. 1–5, 2012.
- [28] E. C. Malthouse and R. C. Blattberg, "Can we predict customer lifetime value?," *Journal of interactive marketing*, vol. 19, no. 1, pp. 2–16, 2005.
- [29] E. L. Lehmann and G. Casella, *Theory of point estimation*. Springer Science & Business Media, 2006.
- [30] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance," *Climate research*, vol. 30, no. 1, pp. 79–82, 2005.
- [31] A. Auslender and M. Teboulle, "Interior gradient and epsilon-subgradient descent methods for constrained convex minimization," *Mathematics of Operations Research*, vol. 29, no. 1, pp. 1–26, 2004.
- [32] Y. Bai, Q. Jiang, and J. Sun, "Subgradient descent learns orthogonal dictionaries," *arXiv preprint* arXiv:1810.10702, 2018.
- [33] P. Bianchi, W. Hachem, and S. Schechtman, "Stochastic subgradient descent escapes active strict saddles on weakly convex functions," *arXiv preprint arXiv:2108.02072*, 2021.
- [34] L. Xiao, "Dual averaging method for regularized stochastic learning and online optimization," *Advances in Neural Information Processing Systems*, vol. 22, 2009.
- [35] P. J. Huber, "Robust estimation of a location parameter," *Breakthroughs in statistics: Methodology and distribution*, pp. 492–518, 1992.
- [36] R. A. Saleh, A. Saleh, et al., "Statistical properties of the log-cosh loss function used in machine learning," arXiv preprint arXiv:2208.04564, 2022.
- [37] R. Koenker and K. F. Hallock, "Quantile regression," *Journal of economic perspectives*, vol. 15, no. 4, pp. 143–156, 2001.
- [38] M.-Y. Chen and J.-E. Chen, "Application of quantile regression to estimation of value at risk," *Review of Financial Risk Management*, vol. 1, no. 2, p. 15, 2002.
- [39] J. Bruzda, "Multistep quantile forecasts for supply chain and logistics operations: bootstrapping, the garch model and quantile regression based approaches," *Central European Journal of Operations Research*, vol. 28, no. 1, pp. 309–336, 2020.

- [40] J. B. Bremnes, "Probabilistic wind power forecasts using local quantile regression," *Wind Energy: An International Journal for Progress and Applications in Wind Power Conversion Technology*, vol. 7, no. 1, pp. 47–54, 2004.
- [41] W. P. Gaglianone and L. R. Lima, "Constructing density forecasts from quantile regressions," *Journal of Money, Credit and Banking*, vol. 44, no. 8, pp. 1589–1607, 2012.
- [42] L. Massidda and M. Marrocu, "Quantile regression post-processing of weather forecast for short-term solar power probabilistic forecasting," *Energies*, vol. 11, no. 7, p. 1763, 2018.
- [43] A. Zarnani, S. Karimi, and P. Musilek, "Quantile regression and clustering models of prediction intervals for weather forecasts: A comparative study," *Forecasting*, vol. 1, no. 1, pp. 169–188, 2019.
- [44] J. Zietz, E. N. Zietz, and G. S. Sirmans, "Determinants of house prices: a quantile regression approach," *The Journal of Real Estate Finance and Economics*, vol. 37, pp. 317–333, 2008.
- [45] R. Winkelmann, "Reforming health care: Evidence from quantile regressions for counts," *Journal of Health Economics*, vol. 25, no. 1, pp. 131–145, 2006.
- [46] H. Yang, K. Ozbay, and B. Bartin, "Effects of open road tolling on safety performance of freeway mainline toll plazas," *Transportation research record*, vol. 2324, no. 1, pp. 101–109, 2012.
- [47] J. Viel, "Poisson regression in epidemiology," *Revue D'epidemiologie et de Sante Publique*, vol. 42, no. 1, pp. 79–87, 1994.
- [48] Y. Mouatassim and E. H. Ezzahid, "Poisson regression and zero-inflated poisson regression: application to private health insurance data," *European actuarial journal*, vol. 2, no. 2, pp. 187–204, 2012.
- [49] H. Shen and J. Z. Huang, "Forecasting time series of inhomogeneous poisson processes with application to call center workforce management," *The Annals of Applied Statistics*, vol. 2, no. 2, pp. 601–623, 2008.
- [50] P. Avila Clemenshia and M. Vijaya, "Click through rate prediction for display advertisement," *International Journal of Computer Applications*, vol. 975, p. 8887, 2016.
- [51] D. Lambert, "Zero-inflated poisson regression, with an application to defects in manufacturing," *Technometrics*, vol. 34, no. 1, pp. 1–14, 1992.
- [52] D. W. Osgood, "Poisson-based regression analysis of aggregate crime rates," *Journal of quantitative criminology*, vol. 16, pp. 21–43, 2000.
- [53] P. Goodwin and R. Lawton, "On the asymmetry of the symmetric mape," *International journal of forecasting*, vol. 15, no. 4, pp. 405–408, 1999.
- [54] X. Tian, H. Wang, and E. Erjiang, "Forecasting intermittent demand for inventory management by retailers: A new approach," *Journal of Retailing and Consumer Services*, vol. 62, p. 102662, 2021.
- [55] D. Ramos, P. Faria, Z. Vale, and R. Correia, "Short time electricity consumption forecast in an industry facility," *IEEE Transactions on Industry Applications*, vol. 58, no. 1, pp. 123–130, 2021.
- [56] S. Kumar Chandar, "Grey wolf optimization-elman neural network model for stock price prediction," *Soft Computing*, vol. 25, pp. 649–658, 2021.
- [57] M. Lawrence, M. O'Connor, and B. Edmundson, "A field study of sales forecasting accuracy and processes," *European Journal of Operational Research*, vol. 122, no. 1, pp. 151–160, 2000.
- [58] K.-F. Chu, A. Y. Lam, and V. O. Li, "Deep multi-scale convolutional lstm network for travel demand and origin-destination predictions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 8, pp. 3219–3232, 2019.
- [59] R. Goodman, J. Miller, and P. Smyth, "Objective functions for neural network classifier design," in *Proceedings*. 1991 IEEE International Symposium on Information Theory, pp. 87–87, 1991.
- [60] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.
- [61] G. King and L. Zeng, "Logistic regression in rare events data," *Political analysis*, vol. 9, no. 2, pp. 137–163, 2001.
- [62] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.
- [63] R. Müller, S. Kornblith, and G. E. Hinton, "When does label smoothing help?," *Advances in neural information processing systems*, vol. 32, 2019.

- [64] Z. Leng, M. Tan, C. Liu, E. D. Cubuk, X. Shi, S. Cheng, and D. Anguelov, "Polyloss: A polynomial expansion perspective of classification loss functions," *arXiv* preprint arXiv:2204.12511, 2022.
- [65] L. Rosasco, E. De Vito, A. Caponnetto, M. Piana, and A. Verri, "Are loss functions all the same?," *Neural computation*, vol. 16, no. 5, pp. 1063–1076, 2004.
- [66] I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun, and Y. Singer, "Large margin methods for structured and interdependent output variables.," *Journal of machine learning research*, vol. 6, no. 9, 2005.
- [67] J. R. Taylor, An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements. University Science Books, 2 sub ed., 1996.
- [68] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," arXiv preprint arXiv:2010.16061, 2020.
- [69] R. Parikh, A. Mathai, S. Parikh, G. C. Sekhar, and R. Thomas, "Understanding and using sensitivity, specificity and predictive values," *Indian journal of ophthalmology*, vol. 56, no. 1, p. 45, 2008.
- [70] P. with Code, "Image classification." https://paperswithcode.com/task/image-classification. Accessed: 2024-04-25.
- [71] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.
- [72] N. Varshney, N. Shalini, M. Sharma, V. K. Yadav, D. V. Saravanan, and N. Kumar, "Resnet transfer learning for enhanced medical image classification in healthcare," 2023 International Conference on Artificial Intelligence for Innovations in Healthcare Industries (ICAIIHI), vol. 1, pp. 1–7, 2023.
- [73] V. S. A. Kagolanu, L. Thimmareddy, K. L. Kanala, and B. Sirisha, "Multi-class medical image classification based on feature ensembling using deepnets," 2022 9th International Conference on Computing for Sustainable Global Development (INDIACom), pp. 540–544, 2022.
- [74] L. Saleh and L. Zhang, "Medical image classification using transfer learning and network pruning algorithms," 2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 1932–1938, 2023.
- [75] G. Harika, K. Keerthi, D. H. Kommineni, and K. Soumya, "Classification of cervical cancer using resnet-50," 2023 Global Conference on Information Technologies and Communications (GCITC), pp. 1–8, 2023.
- [76] S. A. Shah, G. M. Lakho, H. A. Keerio, M. N. Sattar, G. Hussain, M. Mehdi, R. B. Vistro, E. A. Mahmoud, and H. O. Elansary, "Application of drone surveillance for advance agriculture monitoring by android application using convolution neural network," *Agronomy*, 2023.
- [77] Y. Yuan, "Computer vision and deep learning for precise agriculture: A case study of lemon leaf image classification," *Journal of Physics: Conference Series*, vol. 2547, 2023.
- [78] A. D. Nidhis, C. N. V. Pardhu, K. C. Reddy, and K. Deepa, "Cluster based paddy leaf disease detection, classification and diagnosis in crop health monitoring unit," *Computer Aided Intervention and Diagnostics in Clinical and Medical Images*, 2019.
- [79] A. Tendolkar, A. Choraria, M. M. M. Pai, S. Girisha, G. Dsouza, and K. S. Adithya, "Modified crop health monitoring and pesticide spraying system using ndvi and semantic segmentation: An agrocopter based approach," 2021 IEEE International Conference on Autonomous Systems (ICAS), pp. 1–5, 2021.
- [80] J. Peng, C. Xiao, and Y. Li, "Rp2k: A large-scale retail product dataset for fine-grained image classification," 2021.
- [81] M. I. H. Shihab, N. Tasnim, H. Zunair, L. K. Rupty, and N. Mohammed, "Vista: Vision transformer enhanced by u-net and image colorfulness frame filtration for automatic retail checkout," 2022.
- [82] A. Boriya, S. S. Malla, R. Manjunath, V. Velicheti, and M. Eirinaki, "Viser: A visual search engine for e-retail," 2019 First International Conference on Transdisciplinary AI (TransAI), pp. 76–83, 2019.
- [83] M. Alghamdi, H. A. Mengash, M. Aljebreen, M. Maray, A. A. Darem, and A. S. Salama, "Empowering retail through advanced consumer product recognition using aquila optimization algorithm with deep learning," *IEEE Access*, vol. 12, pp. 71055–71065, 2024.
- [84] T. M and J. Singh, "Unusual crowd activity detection in video using cnn, lstm and opency," *International Journal for Research in Applied Science and Engineering Technology*, 2023.
- [85] R. Ayad and F. Q. Al-Khalidi, "Convolutional neural network (cnn) model to mobile remote surveillance system for home security," *International Journal of Computing and Digital Systems*, 2023.

- [86] S. Ramadasan, K. Vijayakumar, S. Prabha, and E. Karthickeien, "Forest region extraction and evaluation from satellite images using cnn segmentation," 2024 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI), pp. 1–5, 2024.
- [87] M. Ahmed, R. Mumtaz, Z. Anwar, A. Shaukat, O. Arif, and F. Shafait, "A multi–step approach for optically active and inactive water quality parameter estimation using deep learning and remote sensing," *Water*, 2022.
- [88] F. Farahnakian, L. Zelioli, M. Middleton, I. Seppä, T. P. Pitkänen, and J. Heikkonen, "Cnn-based boreal peatland fertility classification from sentinel-1 and sentinel-2 imagery," 2023 IEEE International Symposium on Robotic and Sensors Environments (ROSE), pp. 1–7, 2023.
- [89] M. Ilteralp, S. Arıman, and E. Aptoula, "A deep multitask semisupervised learning approach for chlorophyll-a retrieval from remote sensing images," *Remote. Sens.*, vol. 14, p. 18, 2021.
- [90] A. Abdullah, M. Jawahar, N. Manogaran, G. Subbiah, K. Seeranagan, B. Balusamy, and A. C. Saravanan, "Leather image quality classification and defect detection system using mask region-based convolution neural network model," *International Journal of Advanced Computer Science and Applications*, 2024.
- [91] A.-M. A. Mamun, M. R. Hossain, and M. M. Sharmin, "Detection and classification of metal surface defects using lite convolutional neural network (lcnn)," *Material Science & Engineering International Journal*, 2024.
- [92] K. A. Pranoto, W. Caesarendra, I. Petra, G. M. Królczyk, M. D. Surindra, and P. W. Yoyo, "Burrs and sharp edge detection of metal workpiece using cnn image classification method for intelligent manufacturing application," 2023 IEEE 21st International Conference on Industrial Informatics (INDIN), pp. 1–7, 2023.
- [93] Y. Yang, L. Pan, J. Ma, R. Yang, Y. Zhu, Y. Yang, and L. Zhang, "A high-performance deep learning algorithm for the automated optical inspection of laser welding," *Applied Sciences*, 2020.
- [94] S. A. Brown, B. A. Weyori, A. F. Adekoya, P. K. Kudjo, S. Mensah, and S. Abedu, "Deeplabb: A deep learning framework for blocking bugs," in 2021 International Conference on Cyber Security and Internet of Things (ICSIoT), pp. 22–25, IEEE, 2021.
- [95] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in 2009 IEEE conference on computer vision and pattern recognition, pp. 248–255, Ieee, 2009.
- [96] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.
- [97] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," arXiv preprint arXiv:1412.6572, 2014.
- [98] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in 2017 ieee symposium on security and privacy (sp), pp. 39–57, Ieee, 2017.
- [99] R. Daroya, A. Sun, and S. Maji, "Cose: A consistency-sensitivity metric for saliency on image classification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 149–158, 2023.
- [100] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and M. Zieba, "End to end learning for self-driving cars," arXiv preprint arXiv:1604.07316, 2016.
- [101] F. Codevilla, M. Müller, A. López, and V. Koltun, "End-to-end driving via conditional imitation learning," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–9, 2018.
- [102] A. Sadeghian, V. Kosaraju, P. Sadeghian, N. Hirose, H. Rezatofighi, and S. Savarese, "Sophie: An attentive gnn for predicting driving paths from sensor data," *IEEE International Conference on Computer Vision (ICCV)*, pp. 9267–9276, 2019.
- [103] Z. Chen, Y. Liu, X. Li, X. Yang, and Q. Zhang, "Reinforcement learning-based motion planning for autonomous driving in dynamic environments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 2, pp. 736–747, 2020.
- [104] R. Chalapathy and A. Menon, "Anomaly detection using deep one-class classification," *arXiv preprint* arXiv:1802.06360, 2017.
- [105] Y. Li, M. Zhang, Z. Chen, and Q. Yang, "Human activity recognition using recurrent neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 766–779, 2018.
- [106] C. Zhang, H. Zhang, L. Zhang, and Q. Li, "Crowd analysis using deep learning: A survey," *IEEE Access*, vol. 4, pp. 212–228, 2016.
- [107] E. Y. Kim, E. Park, and J. H. Kim, "Emotion recognition based on physiological signals for human-computer interaction," *IEEE Transactions on Affective Computing*, vol. 9, no. 4, pp. 457–470, 2018.

- [108] F. Lotte, L. Bougrain, A. Cichocki, M. Clerc, M. Congedo, A. Rakotomamonjy, and F. Yger, "Deep learning-based electroencephalography analysis: A comprehensive review," *Journal of Neural Engineering*, vol. 15, no. 1, p. 011001, 2018.
- [109] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in Neural Information Processing Systems (NIPS)*, pp. 3104–3112, 2014.
- [110] S. Li and W. Deng, "Deep facial expression recognition: A survey," *IEEE Transactions on Affective Computing*, vol. 9, no. 3, pp. 321–341, 2017.
- [111] S. Levine, P. Pastor, A. Krizhevsky, and J. Ibarz, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.
- [112] T. Zhang, Z. McCarthy, Y. Yang, E. Schmerling, and C. J. Tomlin, "Deep reinforcement learning for robot motion planning in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3713–3720, 2019.
- [113] M. Gualtieri and A. Singh, "Pick-and-place using deep reinforcement learning," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8, 2018.
- [114] A. Hussein, H. Rad, and C. R. Smith, "Imitation learning for robot manipulation using deep neural networks," *arXiv preprint arXiv:1703.08612*, 2017.
- [115] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.
- [116] Pytorch, "torch.nn.smoothl1loss." https://pytorch.org/docs/stable/generated/torch.nn. SmoothL1Loss.html. Accessed: 2024-05-01.
- [117] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.
- [118] J. Pang, K. Chen, J. Shi, H. Feng, W. Ouyang, and D. Lin, "Libra r-cnn: Towards balanced learning for object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 821–830, 2019.
- [119] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pp. 21–37, Springer, 2016.
- [120] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.
- [121] 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*, pp. 658–666, 2019.
- [122] 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, pp. 12993–13000, Apr. 2020.
- [123] Z.-H. Feng, J. Kittler, M. Awais, P. Huber, and X.-J. Wu, "Wing loss for robust facial landmark localisation with convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2235–2245, 2018.
- [124] M. Everingham, 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.
- [125] 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*, pp. 740–755, Springer, 2014.
- [126] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings* of the IEEE conference on computer vision and pattern recognition, pp. 3431–3440, 2015.
- [127] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.
- [128] L.-C. Chen, Y. Zhu, G. Papandreou, and et al., "Encoder-decoder with atrous separable convolution for semantic image segmentation," *European Conference on Computer Vision (ECCV)*, pp. 801–818, 2018.

- [129] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 40, no. 4, pp. 834–848, 2017.
- [130] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pp. 234–241, 2015.
- [131] O. Oktay, J. Schlemper, L. L. Folgoc, and et al., "Attention u-net: Learning where to look for the pancreas," *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pp. 138–146, 2018.
- [132] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *European Conference on Computer Vision (ECCV)*, pp. 346–361, 2014.
- [133] S. Zheng, S. Jayasumana, B. Romera-Paredes, and et al., "Conditional random fields as recurrent neural networks," *International Conference on Computer Vision (ICCV)*, pp. 1529–1537, 2015.
- [134] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.
- [135] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8759–8768, 2018.
- [136] A. Kirillov, K. He, R. Girshick, and P. Dollár, "Panoptic feature pyramid networks," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6399–6408, 2019.
- [137] A. Kirillov, Y. Wu, K. He, and R. Girshick, "Pointrend: Image segmentation as rendering," *European Conference on Computer Vision (ECCV)*, pp. 405–421, 2020.
- [138] X. Wang, T. Kong, C. Shen, X. You, and L. Li, "Solo: Segmenting objects by locations," *IEEE International Conference on Computer Vision (ICCV)*, pp. 6319–6328, 2019.
- [139] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9404–9413, 2019.
- [140] Y. Xiong, H. Zhu, D. Lin, and et al., "Upsnet: A unified panoptic segmentation network," *European Conference on Computer Vision (ECCV)*, pp. 3–19, 2018.
- [141] L. Liu, X. Wang, C. Shen, X. You, and L. Li, "Panoptic feature pyramid networks," *IEEE International Conference on Computer Vision (ICCV)*, pp. 2704–2713, 2017.
- [142] B. Cheng, M. D. Zhang, Z. Zhang, and et al., "Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation," *European Conference on Computer Vision (ECCV)*, pp. 53–69, 2020.
- [143] P. Wang, Y. Chen, I. Stanton, and et al., "Panoptic segmentation transformer," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4077–4086, 2021.
- [144] L.-J. Li, R. Socher, and L. Fei-Fei, "Towards total scene understanding: Classification, annotation and segmentation in an automatic framework," in 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 2036–2043, IEEE, 2009.
- [145] B.-S. Hua, Q.-H. Pham, D. T. Nguyen, and et al., "Scenenet: Understanding real world indoor scenes with synthetic data," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3419–3428, 2016.
- [146] B. Zhou, H. Zhao, X. Puig, and et al., "Scene parsing through ade20k dataset," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 633–641, 2017.
- [147] S. Song, S. Lichtenberg, and J. Xiao, "Sunrgb-d: A rgb-d scene understanding benchmark suite," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 567–576, 2015.
- [148] F. Yu, Y. Zhang, S. Song, and et al., "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprint arXiv:1506.03365*, 2015.
- [149] D. L. Pham, C. Xu, and J. L. Prince, "Current methods in medical image segmentation," *Annual review of biomedical engineering*, vol. 2, no. 1, pp. 315–337, 2000.
- [150] K. Yan, X. Wang, L. Lu, and et al., "Deeplesion: Automated mining of large-scale lesion annotations and universal lesion detection with deep learning," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8262–8271, 2018.
- [151] Y. Liu, K. Gadepalli, M. Norouzi, and et al., "Densely connected convolutional networks for medical image segmentation," *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pp. 232–239, 2017.

- [152] M. Havaei, A. Davy, D. Warde-Farley, and et al., "Brain tumor segmentation with deep neural networks," *Medical Image Analysis*, vol. 35, pp. 18–31, 2017.
- [153] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," *3D Vision (3DV)*, pp. 565–571, 2016.
- [154] K. Kamnitsas, C. Ledig, V. F. Newcombe, and et al., "Efficient multi-scale 3d cnn with fully connected crf for accurate brain lesion segmentation," *Medical Image Analysis*, vol. 36, pp. 61–78, 2017.
- [155] H. Chen, X. Qi, Q. Dou, and et al., "Med3d: Transfer learning for 3d medical image analysis," *IEEE Transactions on Medical Imaging*, vol. 38, no. 12, pp. 2804–2813, 2019.
- [156] B. Pan, J. Sun, H. Y. T. Leung, A. Andonian, and B. Zhou, "Cross-view semantic segmentation for sensing surroundings," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4867–4873, 2020.
- [157] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 652–660, 2017.
- [158] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," *European Conference on Computer Vision (ECCV)*, pp. 492–505, 2014.
- [159] S. Jégou, M. Drozdzal, D. Vazquez, and et al., "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1175–1183, 2017.
- [160] F. N. Iandola, S. Han, M. W. Moskewicz, and et al., "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 685–694, 2016.
- [161] Q. Ha, K. Watanabe, T. Karasawa, Y. Ushiku, and T. Harada, "Mfnet: Towards real-time semantic segmentation for autonomous vehicles with multi-spectral scenes," in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5108–5115, IEEE, 2017.
- [162] V. Mnih, K. Kavukcuoglu, D. Silver, and et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [163] X. Chen, A. Liu, and M. Liu, "Deep sensor fusion for autonomous vehicles: A comprehensive review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 1, pp. 258–268, 2020.
- [164] Y. Gal and Z. Ghahramani, "Uncertainty in deep learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 39, no. 11, pp. 1892–1908, 2016.
- [165] A. Alahi, V. Goel, V. Ramanathan, and et al., "Social lstm: Human trajectory prediction in crowded spaces," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 961–971, 2016.
- [166] H. Chen, Q. Li, Y. Zhao, and et al., "A survey of deep learning techniques for autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 2, pp. 829–846, 2021.
- [167] S. Ammar, T. Bouwmans, N. Zaghden, and M. Neji, "Deep detector classifier (deepdc) for moving objects segmentation and classification in video surveillance," *IET Image processing*, vol. 14, no. 8, pp. 1490–1501, 2020.
- [168] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6525, 2017.
- [169] S. Yeung, O. Russakovsky, and G. Mori, "End-to-end learning of action detection from frame glimpses in videos," Conference on Neural Information Processing Systems (NeurIPS), pp. 2678–2686, 2016.
- [170] D. Tran, L. Bourdev, and R. Fergus, "Learning spatiotemporal features with 3d convolutional networks," *IEEE International Conference on Computer Vision (ICCV)*, pp. 4489–4497, 2015.
- [171] M. Hasan, J. Choi, and J. Neumann, "Learning deep representations for anomaly detection in video surveillance," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1889–1898, 2016.
- [172] T.-y. Ko and S.-h. Lee, "Novel method of semantic segmentation applicable to augmented reality," *Sensors*, vol. 20, no. 6, p. 1737, 2020.
- [173] Q. Cao, L. Liu, L. Zhou, and et al., "Real-time object recognition for augmented reality applications," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4933–4942, 2018.
- [174] Y. Kim and S. Kim, "Augmented reality visual tracking using convolutional neural networks," *IEEE Access*, vol. 6, pp. 3079–3087, 2018.

- [175] B. Tekin, I. Katircioglu, and M. Salzmann, "Real-time hand pose estimation for augmented reality applications using convolutional neural networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 5, pp. 1781–1790, 2018.
- [176] J. Bessesen and A. Kovashka, "Augmented reality with generative neural networks for synthetic object placement," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 11, pp. 2984–2992, 2018.
- [177] T. A. Sorensen, "A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on danish commons," *Biol. Skar.*, vol. 5, pp. 1–34, 1948.
- [178] M. A. Rahman and Y. Wang, "Optimizing intersection-over-union in deep neural networks for image segmentation," in *International symposium on visual computing*, pp. 234–244, Springer, 2016.
- [179] F. van Beers, A. Lindström, E. Okafor, and M. A. Wiering, "Deep neural networks with intersection over union loss for binary image segmentation.," in *ICPRAM*, pp. 438–445, 2019.
- [180] P. Luc, C. Couprie, S. Chintala, and J. Verbeek, "Semantic segmentation using adversarial networks," *arXiv* preprint arXiv:1611.08408, 2016.
- [181] S. S. M. Salehi, D. Erdogmus, and A. Gholipour, "Tversky loss function for image segmentation using 3d fully convolutional deep networks," in *Machine Learning in Medical Imaging: 8th International Workshop, MLMI 2017, Held in Conjunction with MICCAI 2017, Quebec City, QC, Canada, September 10, 2017, Proceedings 8*, pp. 379–387, Springer, 2017.
- [182] M. Berman, A. R. Triki, and M. B. Blaschko, "The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4413–4421, 2018.
- [183] G. Csurka, D. Larlus, and F. Perronnin, "What is a good evaluation measure for semantic segmentation?," in *British Machine Vision Conference*, pp. 10–5244, 2013.
- [184] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
- [185] M. Owayjan, A. Dergham, G. Haber, N. Fakih, A. Hamoush, and E. Abdo, "Face recognition security system," in *New trends in networking, computing, E-learning, systems sciences, and engineering*, pp. 343–348, Springer, 2015.
- [186] I. M. Sayem and M. S. Chowdhury, "Integrating face recognition security system with the internet of things," in 2018 International Conference on Machine Learning and Data Engineering (iCMLDE), pp. 14–18, IEEE, 2018.
- [187] P. Indrawan, S. Budiyatno, N. M. Ridho, and R. F. Sari, "Face recognition for social media with mobile cloud computing," *International Journal on Cloud Computing: Services and Architecture*, vol. 3, no. 1, pp. 23–35, 2013.
- [188] K. I. Chang, K. W. Bowyer, and P. J. Flynn, "Multimodal 2d and 3d biometrics for face recognition," in 2003 IEEE International SOI Conference. Proceedings (Cat. No. 03CH37443), pp. 187–194, IEEE, 2003.
- [189] A. W. Senior and R. M. Bolle, "Face recognition and its application," *Biometric Solutions: For Authentication in an E-World*, pp. 83–97, 2002.
- [190] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 212–220, 2017.
- [191] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VII 14*, pp. 499–515, Springer, 2016.
- [192] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, "Cosface: Large margin cosine loss for deep face recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5265–5274, 2018.
- [193] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4690–4699, 2019.
- [194] C. Manwatkar, "How to choose a loss function for face recognition," 2023. Accessed: 2023-06-28.
- [195] M. Schultz and T. Joachims, "Learning a distance metric from relative comparisons," *Advances in neural information processing systems*, vol. 16, 2003.

- [196] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.
- [197] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krahenbuhl, "Sampling matters in deep embedding learning," in *Proceedings of the IEEE international conference on computer vision*, pp. 2840–2848, 2017.
- [198] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *Similarity-Based Pattern Recognition:* Third International Workshop, SIMBAD 2015, Copenhagen, Denmark, October 12-14, 2015. Proceedings 3, pp. 84–92, Springer, 2015.
- [199] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 1, pp. 539–546, IEEE, 2005.
- [200] Y. Sun, C. Cheng, Y. Zhang, C. Zhang, L. Zheng, Z. Wang, and Y. Wei, "Circle loss: A unified perspective of pair similarity optimization," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6398–6407, 2020.
- [201] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, "Barlow twins: Self-supervised learning via redundancy reduction," in *International Conference on Machine Learning*, pp. 12310–12320, PMLR, 2021.
- [202] X. Chen and K. He, "Exploring simple siamese representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15750–15758, 2021.
- [203] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," *Advances in neural information processing systems*, vol. 27, 2014.
- [204] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.
- [205] Z. Wang, A. C. Bovik, and H. R. Sheikh, "Structural similarity based image quality assessment," in *Digital Video image quality and perceptual coding*, pp. 225–242, CRC Press, 2017.
- [206] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1851–1858, 2017.
- [207] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [208] L. Zwald and S. Lambert-Lacroix, "The berhu penalty and the grouped effect," arXiv preprint arXiv:1207.6868, 2012.
- [209] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in 2016 Fourth international conference on 3D vision (3DV), pp. 239–248, IEEE, 2016.
- [210] S. Paul, B. Jhamb, D. Mishra, and M. S. Kumar, "Edge loss functions for deep-learning depth-map," *Machine Learning with Applications*, vol. 7, p. 100218, 2022.
- [211] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth estimation," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3828–3838, 2019.
- [212] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 3, pp. 1623–1637, 2020.
- [213] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in 2012 IEEE conference on computer vision and pattern recognition, pp. 3354–3361, IEEE, 2012.
- [214] N. Silberman and R. Fergus, "Indoor scene segmentation using a structured light sensor," in 2011 IEEE international conference on computer vision workshops (ICCV workshops), pp. 601–608, IEEE, 2011.
- [215] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V 12*, pp. 746–760, Springer, 2012.
- [216] W. Chen, Z. Fu, D. Yang, and J. Deng, "Single-image depth perception in the wild," *Advances in neural information processing systems*, vol. 29, 2016.
- [217] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," arXiv preprint arXiv:1312.6114, 2013.
- [218] S. R. Bowman, L. Vilnis, O. Vinyals, and et al., "Generating sentences from a continuous space," *CoRR*, vol. abs/1511.06349, 2016.

- [219] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, no. 1, 2015.
- [220] S. Zhao, J. Song, and S. Ermon, "Infovae: Balancing learning and inference in variational autoencoders," *CoRR*, vol. abs/1706.02262, 2017.
- [221] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.
- [222] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems* (Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, eds.), vol. 27, pp. 2672–2680, Curran Associates, Inc., 2014.
- [223] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," 2017.
- [224] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5967–5976, 2017.
- [225] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "Midinet: A convolutional generative adversarial network for symbolic-domain music generation," 2017.
- [226] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," arXiv preprint arXiv:1605.08803, 2016.
- [227] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," *Advances in neural information processing systems*, vol. 31, 2018.
- [228] W. Grathwohl, R. T. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, "Ffjord: Free-form continuous dynamics for scalable reversible generative models," *arXiv preprint arXiv:1810.01367*, 2018.
- [229] D. J. Rezende and S. Mohamed, "Variational inference with normalizing flows," 2016.
- [230] D. J. Rezende and S. Mohamed, "Variational inference with normalizing flows," *Journal of Machine Learning Research (JMLR)*, vol. 18, no. 1, pp. 201–244, 2017.
- [231] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang, "A tutorial on energy-based learning," *Predicting structured data*, vol. 1, no. 0, 2006.
- [232] Y. Du and I. Mordatch, "Implicit generation and modeling with energy based models," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [233] T. Tieleman, "Training restricted boltzmann machines using approximations to the likelihood gradient," in *Neural Computation*, pp. 1064–1071, 01 2008.
- [234] T. Tieleman, "Training restricted boltzmann machines using approximations to the likelihood gradient," in *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, (New York, NY, USA), p. 1064–1071, Association for Computing Machinery, 2008.
- [235] D. Belanger and A. McCallum, "Structured prediction energy networks," in *International Conference on Machine Learning (ICML)*, p. 983–992, 2016.
- [236] R. E. Tillman, T. Balch, and M. Veloso, "Privacy-preserving energy-based generative models for marginal distribution protection," *Transactions on Machine Learning Research*, 2023.
- [237] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International Conference on Machine Learning*, pp. 2256–2265, PMLR, 2015.
- [238] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.
- [239] W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duvenaud, M. Norouzi, and K. Swersky, "Your classifier is secretly an energy based model and you should treat it like one," 2020.
- [240] J. R. Hershey, J. L. Roux, and F. Weninger, "Deep unfolding: Model-based inspiration of novel deep architectures," 2014.
- [241] Y. Song, Y. Zhang, and S. Ermon, "Deep variational diffusion models," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, 2021.
- [242] S. C. Park, M. K. Park, and M. G. Kang, "Super-resolution image reconstruction: a technical overview," *IEEE signal processing magazine*, vol. 20, no. 3, pp. 21–36, 2003.
- [243] C. Dong, C. C. Loy, K. He, and et al., "Learning a deep convolutional network for image super-resolution," in *European Conference on Computer Vision (ECCV)*, pp. 184–199, 2014.

- [244] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network.," *CoRR*, vol. abs/1609.04802, 2016.
- [245] Y. Zhang, K. Li, K. Li, and et al., "Image super-resolution using very deep residual channel attention networks," in *European Conference on Computer Vision (ECCV)*, p. 294–310, 2018.
- [246] C. Tian, L. Fei, W. Zheng, Y. Xu, W. Zuo, and C.-W. Lin, "Deep learning on image denoising: An overview," *Neural Networks*, vol. 131, pp. 251–275, 2020.
- [247] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, p. 341–349, 2012.
- [248] J. Lehtinen, "Noise2noise: Learning image restoration without clean data," *arXiv preprint arXiv:1803.04189*, 2018.
- [249] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 417–424, 2000.
- [250] J. Yu, Z. L. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Free-form image inpainting with gated convolution," 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 4470–4479, 2018.
- [251] D. Pathak, P. Krahenbuhl, J. Donahue, and et al., "Context encoders: Feature learning by inpainting," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2536–2544, 2016.
- [252] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image inpainting for irregular holes using partial convolutions," 2018.
- [253] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv preprint arXiv:1508.06576*, 2015.
- [254] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," 2016. cite arxiv:1603.08155.
- [255] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1510–1519, 2017.
- [256] F. Yang, H. Yang, J. Fu, H. Lu, and B. Guo, "Learning texture transformer network for image super-resolution," 2020.
- [257] J. Nash Jr, "Non-cooperative games," in Essays on Game Theory, pp. 22–33, Edward Elgar Publishing, 1996.
- [258] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*, pp. 214–223, PMLR, 2017.
- [259] M. A. Carreira-Perpinan and G. Hinton, "On contrastive divergence learning," in *International workshop on artificial intelligence and statistics*, pp. 33–40, PMLR, 2005.
- [260] I. J. Myung, "Tutorial on maximum likelihood estimation," *Journal of mathematical Psychology*, vol. 47, no. 1, pp. 90–100, 2003.
- [261] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 297–304, JMLR Workshop and Conference Proceedings, 2010.
- [262] I. Csiszar, "*I*-Divergence Geometry of Probability Distributions and Minimization Problems," *The Annals of Probability*, vol. 3, no. 1, pp. 146 158, 1975.
- [263] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European Conference on Computer Vision*, 2016.
- [264] J. Bruna, P. Sprechmann, and Y. LeCun, "Super-resolution with deep convolutional sufficient statistics," *arXiv* preprint arXiv:1511.05666, 2015.
- [265] K. Simonyan, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.
- [266] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al., "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4681–4690, 2017.
- [267] A. Dosovitskiy and T. Brox, "Generating images with perceptual similarity metrics based on deep networks," *Advances in neural information processing systems*, vol. 29, 2016.
- [268] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

- [269] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," *Advances in neural information processing systems*, vol. 30, 2017.
- [270] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural computation*, vol. 14, no. 8, pp. 1771–1800, 2002.
- [271] G. Desjardins, A. Courville, Y. Bengio, P. Vincent, and O. Delalleau, "Tempered markov chain monte carlo for training of restricted boltzmann machines," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 145–152, JMLR Workshop and Conference Proceedings, 2010.
- [272] M. Opper and D. Saad, Advanced mean field methods: Theory and practice. MIT press, 2001.
- [273] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *Advances in neural information processing systems*, vol. 29, 2016.
- [274] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.
- [275] D. Mukherkjee, P. Saha, D. Kaplun, A. Sinitca, and R. Sarkar, "Brain tumor image generation using an aggregation of gan models with style transfer," *Scientific Reports*, vol. 12, no. 1, p. 9141, 2022.
- [276] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.
- [277] B. Pang, L. Lee, et al., "Opinion mining and sentiment analysis," Foundations and Trends® in information retrieval, vol. 2, no. 1–2, pp. 1–135, 2008.
- [278] V. Metsis, I. Androutsopoulos, and G. Paliouras, "Spam filtering with naive bayes-which naive bayes?," in *CEAS*, vol. 17, pp. 28–69, Mountain View, CA, 2006.
- [279] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.
- [280] H. L. Chieu and H. T. Ng, "Named entity recognition with a maximum entropy approach," in *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pp. 160–163, 2003.
- [281] K. W. Church, "A stochastic parts program and noun phrase parser for unrestricted text," in *International Conference on Acoustics, Speech, and Signal Processing*, pp. 695–698, IEEE, 1989.
- [282] T. Shi, Y. Keneshloo, N. Ramakrishnan, and C. K. Reddy, "Neural abstractive text summarization with sequence-to-sequence models," *ACM Transactions on Data Science*, vol. 2, no. 1, pp. 1–37, 2021.
- [283] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
- [284] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, *et al.*, "Improving language understanding by generative pre-training," 2018.
- [285] J.-M. Torres-Moreno, Automatic text summarization. John Wiley & Sons, 2014.
- [286] H. Schmid, "Part-of-speech tagging with neural networks," arXiv preprint cmp-lg/9410018, 1994.
- [287] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates, "Unsupervised named-entity extraction from the web: An experimental study," *Artificial intelligence*, vol. 165, no. 1, pp. 91–134, 2005.
- [288] B. Taskar, C. Guestrin, and D. Koller, "Max-margin markov networks," *Advances in neural information processing systems*, vol. 16, 2003.
- [289] A. F. Martins, M. A. Figueiredo, and N. A. Smith, "Structured sparsity in natural language processing: models, algorithms and applications," in *Tutorial Abstracts at the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2012.
- [290] H. Wang, Adversarial Prediction Framework for Information Retrieval and Natural Language Processing Metrics. PhD thesis, University of Illinois at Chicago, 2017.
- [291] R. Moore and J. DeNero, "L1 and l2 regularization for multiclass hinge loss models," in *Symposium on machine learning in speech and language processing*, 2011.
- [292] J. Wang, X. Le, X. Peng, and C. Chen, "Adaptive hinge balance loss for document-level relation extraction," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3872–3878, 2023.

- [293] J. T. Hoe, K. W. Ng, T. Zhang, C. S. Chan, Y.-Z. Song, and T. Xiang, "One loss for all: Deep hashing with a single cosine similarity based learning objective," *Advances in Neural Information Processing Systems*, vol. 34, pp. 24286–24298, 2021.
- [294] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in neural information processing systems*, vol. 26, 2013.
- [295] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *International conference on machine learning*, pp. 2071–2080, PMLR, 2016.
- [296] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, pp. 369–376, 2006.
- [297] S. Shen, Y. Cheng, Z. He, W. He, H. Wu, M. Sun, and Y. Liu, "Minimum risk training for neural machine translation," *arXiv preprint arXiv:1512.02433*, 2015.
- [298] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318, 2002.
- [299] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out*, pp. 74–81, 2004.
- [300] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, pp. 229–256, 1992.
- [301] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.
- [302] F. Jelinek, R. L. Mercer, L. R. Bahl, and J. K. Baker, "Perplexity—a measure of the difficulty of speech recognition tasks," *The Journal of the Acoustical Society of America*, vol. 62, no. S1, pp. S63–S63, 1977.
- [303] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? sentiment classification using machine learning techniques," *arXiv preprint cs/0205070*, 2002.
- [304] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, "A bayesian approach to filtering junk e-mail," in *Learning for Text Categorization: Papers from the 1998 workshop*, vol. 62, pp. 98–105, Citeseer, 1998.
- [305] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *European conference on machine learning*, pp. 137–142, Springer, 1998.
- [306] D. D. Lewis, "Naive (bayes) at forty: The independence assumption in information retrieval," in *European conference on machine learning*, pp. 4–15, Springer, 1998.
- [307] Y. Heryanto and A. Triayudi, "Evaluating text quality of gpt engine davinci-003 and gpt engine davinci generation using bleu score," *SAGA: Journal of Technology and Information System*, vol. 1, no. 4, pp. 121–129, 2023.
- [308] M. S. Amin, A. Mazzei, L. Anselma, *et al.*, "Towards data augmentation for drs-to-text generation," in *CEUR WORKSHOP PROCEEDINGS*, vol. 3287, pp. 141–152, CEUR-WS, 2022.
- [309] S. Banerjee and A. Lavie, "Meteor: An automatic metric for mt evaluation with improved correlation with human judgments," in *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pp. 65–72, 2005.
- [310] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv* preprint arXiv:1910.13461, 2019.
- [311] Y. Wang, J. Jiang, M. Zhang, C. Li, Y. Liang, Q. Mei, and M. Bendersky, "Automated evaluation of personalized text generation using large language models," *arXiv preprint arXiv:2310.11593*, 2023.
- [312] H. Masataki, Y. Sagisaka, K. Hisaki, and T. Kawahara, "Task adaptation using map estimation in n-gram language modeling," in 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 2, pp. 783–786, IEEE, 1997.
- [313] H. Li, D. Cai, J. Xu, and T. Watanabe, "n-gram is back: Residual learning of neural text generation with n-gram language model," arXiv preprint arXiv:2210.14431, 2022.
- [314] M. Sundermeyer, H. Ney, and R. Schlüter, "From feedforward to recurrent 1stm neural networks for language modeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 517–529, 2015.
- [315] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

- [316] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.
- [317] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.
- [318] A. Chouhan and M. Gertz, "Lexdrafter: Terminology drafting for legislative documents using retrieval augmented generation," *arXiv preprint arXiv:2403.16295*, 2024.
- [319] M. A. Habib, S. Amin, M. Oqba, S. Jaipal, M. J. Khan, and A. Samad, "Taxtajweez: A large language model-based chatbot for income tax information in pakistan using retrieval augmented generation (rag)," in *The International FLAIRS Conference Proceedings*, vol. 37, 2024.
- [320] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, and H. Wang, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, 2023.
- [321] A. Salemi and H. Zamani, "Evaluating retrieval quality in retrieval-augmented generation," in *Proceedings* of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 2395–2400, 2024.
- [322] Z. Rackauckas, A. Câmara, and J. Zavrel, "Evaluating rag-fusion with ragelo: an automated elo-based framework," *arXiv preprint arXiv:2406.14783*, 2024.
- [323] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, "Ragas: Automated evaluation of retrieval augmented generation," *arXiv preprint arXiv:2309.15217*, 2023.
- [324] J. Saad-Falcon, O. Khattab, C. Potts, and M. Zaharia, "Ares: An automated evaluation framework for retrieval-augmented generation systems," *arXiv preprint arXiv:2311.09476*, 2023.
- [325] Ragas.io, "Answer correctness." https://docs.ragas.io/en/latest/concepts/metrics/answer\_correctness.html. Accessed: 2024-05-28.
- [326] Ragas.io, "Answer relevance." https://docs.ragas.io/en/latest/concepts/metrics/answer\_relevance.html. Accessed: 2024-05-28.
- [327] Ragas.io, "Context precision." https://docs.ragas.io/en/latest/concepts/metrics/context\_precision.html. Accessed: 2024-05-28.
- [328] Ragas.io, "Context recall." https://docs.ragas.io/en/latest/concepts/metrics/context\_recall. html. Accessed: 2024-05-28.
- [329] Ragas.io, "Faithfulness." https://docs.ragas.io/en/latest/concepts/metrics/faithfulness. html. Accessed: 2024-05-28.
- [330] Ragas.io, "Summarization score." https://docs.ragas.io/en/latest/concepts/metrics/summarization\_score.html. Accessed: 2024-08-10.
- [331] Ragas.io, "Context entities recall." https://docs.ragas.io/en/latest/concepts/metrics/context\_entities\_recall.html. Accessed: 2024-08-11.
- [332] P. He, X. Liu, J. Gao, and W. Chen, "Deberta: Decoding-enhanced bert with disentangled attention," *arXiv* preprint arXiv:2006.03654, 2020.
- [333] D. Xu, W. Ouyang, X. Wang, and N. Sebe, "Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 675–684, 2018.
- [334] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7482–7491, 2018
- [335] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," *arXiv preprint arXiv:1511.06732*, 2015.
- [336] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks," in *International conference on machine learning*, pp. 794–803, PMLR, 2018
- [337] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *2016 fourth international conference on 3D vision (3DV)*, pp. 565–571, Ieee, 2016.
- [338] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, "Gradient surgery for multi-task learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5824–5836, 2020.

- [339] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," *Advances in neural information processing systems*, vol. 31, 2018.
- [340] G. Zhang, Y. Gao, H. Xu, H. Zhang, Z. Li, and X. Liang, "Ada-segment: Automated multi-loss adaptation for panoptic segmentation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 3333–3341, 2021.
- [341] R. Groenendijk, S. Karaoglu, T. Gevers, and T. Mensink, "Multi-loss weighting with coefficient of variations," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 1469–1478, 2021.
- [342] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.
- [343] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [344] P. Liu, G. Zhang, B. Wang, H. Xu, X. Liang, Y. Jiang, and Z. Li, "Loss function discovery for object detection via convergence-simulation driven search," *arXiv* preprint arXiv:2102.04700, 2021.
- [345] H. Li, T. Fu, J. Dai, H. Li, G. Huang, and X. Zhu, "Autoloss-zero: Searching loss functions from scratch for generic tasks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1009–1018, 2022.
- [346] B. Efron, "The efficiency of cox's likelihood function for censored data," *Journal of the American statistical Association*, vol. 72, no. 359, pp. 557–565, 1977.
- [347] S. C. Hoi, D. Sahoo, J. Lu, and P. Zhao, "Online learning: A comprehensive survey," *Neurocomputing*, vol. 459, pp. 249–289, 2021.
- [348] P. Rajpurkar, J. Irvin, R. L. Ball, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, C. P. Langlotz, *et al.*, "Deep learning for chest radiograph diagnosis: A retrospective comparison of the chexnext algorithm to practicing radiologists," *PLoS medicine*, vol. 15, no. 11, p. e1002686, 2018.
- [349] R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. Bellemare, "Deep reinforcement learning at the edge of the statistical precipice," *Advances in neural information processing systems*, vol. 34, pp. 29304–29320, 2021.
- [350] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature machine intelligence*, vol. 1, no. 5, pp. 206–215, 2019.