Enhancing YOLO through Multi-Task Learning: Joint Detection, Reconstruction, and Classification of Distorted Text Images

by

Reshma Shaji

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2025

© Reshma Shaji 2025

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Robust recognition of alphanumeric text mounted on vehicle surfaces may present significant challenges. These real-world challenges include conditions such as motion blur, out-of-focus imagery, variation in illumination, and compression artifacts. Existing automatic license plate recognition (ALPR) pipelines usually separate detection, enhancement, and recognition into distinct stages, either relying on explicit deblurring networks or extensive augmentation for generalization, each incurring latency, error propagation, or a performance ceiling on severely degraded inputs.

This study introduces YOLO CRNet, a unified end-to-end multi-task framework built upon the YOLO object detector, designed to simultaneously localize characters, enhance text regions, and perform optical character recognition (OCR) within a single network. We integrate two specialized heads into the YOLO backbone: a reconstruction head that restores degraded text regions, and a classification head that directly recognizes alphanumeric characters. Shared feature representations are extracted from multiple depths of the core YOLO network for synergistic learning across complementary tasks.

To inform feature selection for the classifier head, we extract per-character embeddings from five different layer combinations of the YOLO network (ranging from early backbone to deep neck layers) and visualize class separability via t-SNE. This analysis reveals that Configuration A which comprises of early backbone layers (1,2,4) with neck layers (10,13,16) yields the most distinct clusters for the alphanumeric character classes. The YOLO CRNet classifier head trained on Configuration A achieves 95.2% accuracy and a 94.97% F1-score on a held-out set of 10,100 sharp character crops, outperforming alternative layer configurations by up to 18%. Extensive experiments on blurred text datasets demonstrate that combined reconstruction followed by classification of YOLO CRNet significantly outperforms both the baseline YOLO detector and the YOLO CRNet classification head. In particular, the combined reconstruction followed by classification configuration achieves a 23.5% relative improvement in classification accuracy (from 44.5% to 68.0%) and a 15.5% gain in F1-score (from 0.550 to 0.705).

By integrating detection, enhancement, and recognition into a single network guided by t-SNE based feature selection, YOLO CRNet reduces latency, mitigates error propagation, and explicitly handles image distortions. This work lays a foundation for real-time, robust vehicle text detection and illustrates the power of multi-task learning and data-driven feature analysis in fine-grained text recognition tasks.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Prof. Kshirasagar Naik, for his unwavering support and insightful guidance throughout this entire journey. His promptness in addressing any questions and his engaging discussions have continually challenged me to deepen my understanding and broaden my research scope. I would also like to express my gratitude to Dr. Akshaya Mishra and Dr. Ashirbad Pradhan at Eaigle Inc. for their consistent support and valuable guidance throughout the project. Their strategic insights and patient mentorship helped steer the project's direction and ensured the robustness of our methodology. Additionally, I am grateful to Dr. Marzia Zaman and Prof. Ramadan ElShatshat for their invaluable time and feedback on my thesis. I would also like to thank God for granting me the strength and perseverance to complete this work. I sincerely appreciate my friend, Daniel Almeida, for his assistance and thoughtful feedback throughout the project. Last but not least, I extend my heartfelt appreciation to my parents, my brother, my grandparents, and my entire family for their unwavering support and belief in me.

Dedication

This thesis is dedicated to my parents Biji and Shaji, my brother Rohith, my best friend Daniel and my grandparents. I owe my deepest gratitude to my mother, Biji, whose boundless love and sacrifices have shaped me into the person I am today. I am forever thankful to my father, Shaji, whose unwavering belief in my potential gave me the confidence to pursue my dreams with determination. To my brother, Rohith, whose constant support and good humor have lifted my spirits in the toughest moments. I would like to thank my friend, Daniel, whose steadfast encouragement and insightful camaraderie have been a guiding light throughout this journey. I would also like to honor my grandparents, whose wisdom and nurturing care have been the foundation of my life. Their unwavering faith in me continue to inspire every step I take.

Lastly, I extend my thanks to my aunts, uncles and cousins for their warm encouragement and thoughtful guidance. Their collective support has been instrumental in helping me reach this milestone.

All of you have been my unwavering pillars of strength and inspiration. Without your belief in me and your constant support, I would not have had the courage and resilience to relocate to a foreign country and start life anew, nor to seize the opportunity to pursue my education at one of the world's most prestigious institutions.

Table of Contents

Author's Declaration ii

Abstract iii

Acknowledgements iv

Dedication v

List of Figures x

List of Tables xii

1 Introduction 1

1.1 Motivation 1

1.2 Problem Formulation 3

1.3 Contributions 4

1.4 Outline of my thesis 5

2 Literature Review 7

2.1 Scene Text Detection and Recognition 7

2.2 Object Detection Frameworks for Text 8

2.2.1 Evolution of YOLO and its Variants 8

2.2.2 YOLO for Text Detection: Strengths and Limitations 9
2.3 Impact of Image Degradations on Text Recognition 9
2.4 Image Restoration Techniques 11
2.4.1 Classical Image Deblurring and Enhancement 11
2.4.2 Deep Learning-based Image Restoration 12
2.5 Existing Workflows in ALPR and Text Recognition 13
2.5.1 Workflow 1: Multi-Stage ALPR with Explicit Deblurring 13
2.5.2 Workflow 2: Multi-Stage ALPR with Robust Training (No Explicit Deblurring) 14
2.5.3 Workflow 3: End-to-End ALPR with Robust Training 14
2.6 Advanced Learning Strategies and Analysis 15
2.6.1 Multi-Task Learning in Computer Vision 15
2.6.2 Feature Visualization and Analysis with t-SNE 16

3 Detailed Problem Formulation **17**
3.1 System Overview and Unified Mapping 17
3.2 Loss Function Formulation 18
3.2.1 Detection Loss ($\mathcal{L}_{\text{det}}$) 19
3.2.2 Reconstruction Loss ($\mathcal{L}_{\text{recon}}$) 19
3.2.3 Classification Loss ($\mathcal{L}_{\text{cls}}$) 21
3.3 Training Objective 21

4 Exploratory Data Analysis **22**
4.1 Dataset Description 22
4.2 Class Separability Analysis 24
4.2.1 Dataset and Preprocessing 24
4.2.2 Feature Embedding Extraction 25
4.2.3 YOLO Layer Configurations 26

4.2.4 Dimensionality Reduction via t-SNE . . . 26
4.2.5 Evaluation Criteria . . . 28
4.2.6 Visualizations . . . 29
4.3 Imbalance Analysis and Ambiguous Character Separability . . . 31
4.3.1 Overall Label Distribution . . . 31
4.3.2 Ambiguous Character Groups . . . 32
4.3.3 Impact on Model Training . . . 33
4.3.4 Mitigation Strategies . . . 33

5 Methodology **35**
5.1 Pipeline . . . 35
5.1.1 Architecture Overview . . . 35
5.1.2 Training Flow . . . 36
5.2 YOLO: Architecture and Feature Representation . . . 37
5.3 Classifier Head . . . 42
5.3.1 Architecture . . . 42
5.3.2 Design Rationale . . . 45
5.3.3 Loss Function for Training . . . 46
5.3.4 Regularization and Overfitting Prevention . . . 46
5.4 Reconstruction Head . . . 47
5.4.1 Workflow . . . 47
5.4.2 Architecture . . . 49
5.4.3 Design Rationale . . . 55
5.4.4 Loss Functions for Training . . . 57
5.4.5 Measures to Prevent Overfitting . . . 59

6 Experiments and Results **60**
6.1 Experimental Setup . . . 60
6.2 Evaluation Metrics . . . 61
6.3 YOLO CRNet Classification Head Evaluation . . . 63
6.3.1 Experimental Setup . . . 64
6.3.2 Performance Comparison of Layer Configurations . . . 64
6.3.3 Detailed Performance Metrics with Best Layer Configuration A . . . 66
6.3.4 Test Confusion Matrix Analysis . . . 69
6.4 YOLO CRNet Reconstruction Head Evaluation . . . 71
6.4.1 Experimental Setup for Reconstruction Head . . . 71
6.4.2 Performance Comparison of Reconstruction Head Configurations . . . 72
6.4.3 Qualitative Evaluation of Reconstructed Images . . . 73
6.4.4 Ablation Studies on Reconstruction Loss Components . . . 75
6.5 Overall YOLO CRNet Pipeline Evaluation . . . 78
6.5.1 Experimental Design and Models Compared . . . 78
6.5.2 Comparative Classification Performance on Reconstructed Images . . . 79
6.5.3 Analysis and Discussion of Pipeline Evaluation Results . . . 81

7 Conclusion and Future Work **83**
7.1 Conclusion . . . 83
7.2 Challenges and Limitations . . . 84
7.3 Future Work . . . 85

References **87**

List of Figures

2.1 Challenges With Real-world Vehicle Text Detection and Classification . . . 11

2.2 Vehicle text recognition workflow with image enhancement 13

4.1 Sharp Image samples from the dataset 23

4.2 Blur Image samples from the dataset 24

4.3 Feature Embedding Extraction workflow for t-SNE Plot visualization . . . 26

4.4 t-SNE plots for all labels with different YOLO layer configurations (A-E) . 29

4.5 t-SNE plots for all digits with different YOLO layer configurations (A-E) . 30

4.6 Histogram of label occurrences for digits (0–9), lowercase (a–z), and uppercase (A–Z) characters. 31

4.7 t-SNE embeddings of feature vectors for each ambiguous group (A–F). The colored points correspond to the different character clusters. Some groups (e.g. E, F) form well-separated clusters, while others (e.g. B, D) exhibit significant overlap. 33

5.1 Architecture of the proposed YOLO CRNet model 36

5.2 YOLO CRNet flow diagram 37

5.3 YOLOv8 architecture [65] 39

5.4 Figure shows samples of features from YOLO layers 1 (YOLO Early Backbone) and 16 (YOLO Neck) respectively, where 32 channels are plotted. 42

5.5 YOLO CRNet classification flow diagram. 42

5.6 YOLO CRNet Classifier Head Architecture. 45

5.7 YOLO CRNet Reconstruction Head Workflow 47
5.8 Reconstruction Head Configuration A 50
5.9 Illustration of Transposed Convolution 51
5.10 Reconstruction Head Configuration B 52
5.11 Illustration of PixelShuffle [68] 53
5.12 Reconstruction Head Configuration C 54
5.13 SE Block Illustration [22] 55
5.14 Perceptual Loss 58

6.1 YOLO CRNet Classifier Head Accuracy, Precision, Recall & F1-Score Charts for Different Layer Configurations 65
6.2 Test Confusion Matrix for Classifier Head with Configuration A 70
6.3 Reconstructed Result Samples of Model Configuration A 73
6.4 Reconstructed Result Samples of Model Configuration B 74
6.5 Reconstructed Result Samples of Model Configuration C 75
6.6 Ablation Study: Impact of Perceptual Loss 76
6.7 Ablation Study: Impact of L_1 Loss 77
6.8 Ablation Study: Impact of L_2 Loss 77
6.9 Accuracy, Precision, Recall & F1-Score Comparison Chart of Various Models with Blurred Images 80

List of Tables

4.1 YOLO Layer Configurations 27
4.2 Ambiguous character configurations evaluated via t-SNE. 32

6.1 Dataset Split. 61
6.2 Hyperparameters used in the training phase. 61
6.3 Notations for Performance Metrics 62
6.4 YOLO CRNet Classifier Head Performance by Layer Configuration 64
6.5 YOLO CRNet Performance Metrics Classification Report for Digits (0-9) with Configuration A 66
6.6 Overall YOLO CRNet Performance Metrics Classification Report (a-z) . . 67
6.7 Overall YOLO CRNet Performance Metrics Classification Report (A-Z) . 68
6.8 Performance of Reconstruction Head Configurations 72
6.9 Accuracy, Precision, Recall & F1-Score Comparison of Various Models/Configurations with Blurred Input Images 80

Chapter 1

Introduction

This chapter introduces the key challenges in automatic scene-text detection and recognition under real-world degradations and motivates the need for an end-to-end, multi-task approach (Section 1.1). Section 1.2 then briefly defines our problem and sets its composite loss function. Section 1.3 summarizes the main contributions of this thesis. Finally, Section 1.4 provides a roadmap of the remaining chapters.

1.1 Motivation

Automatic scene-text detection and recognition is the task of localizing and transcribing text from real-world images. It is crucial for applications like Automatic License-Plate Recognition (ALPR) [18], assistive technologies [7], and augmented reality [16]. Unlike document Optical Character Recognition (OCR), scene text contends with complex backgrounds, variable illumination, and diverse fonts and orientations [12]. Moreover, severe image degradations such as motion or defocus blur, sensor noise, compression artifacts, glare, occlusions, and adverse weather critically impair detection and recognition accuracy [18, 12, 44].

Convolutional Neural Networks (CNNs) have revolutionized visual recognition by learning hierarchical, translation-invariant features directly from pixels [35]. Early layers capture edges and textures, while deeper layers form semantic concepts. This power drives state-of-the-art object detection and character recognition [24].

In ALPR, a key application, license plates are often captured under challenging conditions. High vehicle speeds or camera motion induce motion blur, streaking characters and

obscuring edges [18, 82]. Limited focus control or incorrect autofocus can lead to defocus blur, causing a general loss of sharpness and character detail [4]. Sensor noise, exacerbated by low-light conditions, introduces random pixel variations that mask fine details [18]. Lossy JPEG compression, used for efficient storage or transmission, can create blocking artifacts (grid-like patterns) and ringing artifacts (spurious oscillations near edges), distorting character shapes. Environmental factors like glare from reflective surfaces can wash out text, while occlusions from dirt or damage can hide characters. Adverse weather (rain, haze, snow) further degrades image quality [12, 18]. These often co-occurring distortions significantly challenge traditional ALPR systems.

Classical deblurring methods typically assume spatially invariant blur kernels and require accurate kernel estimation, where errors lead to artifacts and noise amplification [82]. They also rely on hand-tuned regularization and struggle with non-uniform or mixed degradations [36]. Deep learning-based restoration networks [68, 50] learn data-driven filters but are often applied as standalone pre-processing steps. This introduces latency and can compound errors for downstream tasks if the restoration is not optimized for the end goal of recognition.

Object detection frameworks like "You Only Look Once" (YOLO) have revolutionized the field by recasting object detection as a single, unified regression problem [58, 59]. Instead of complex multi-stage pipelines, YOLO divides the input image into a grid and simultaneously predicts bounding boxes, objectness scores (confidence that a box contains an object), and class probabilities in a single forward pass through a convolutional neural network backbone. This design leads to extremely fast inference speeds. YOLOv8, a recent iteration, achieves state-of-the-art speed-accuracy trade-offs, particularly on embedded hardware, making it an attractive candidate for real-time ALPR systems [24]. However, standard YOLO architectures, while excellent for localizing objects, lack explicit mechanisms for image enhancement to counteract the severe degradations common in vehicle text images.

To address these multifaceted challenges in a more integrated manner, Multi-Task Learning (MTL) offers a principled and powerful framework [10, 63]. MTL involves jointly optimizing a shared network model for several complementary objectives simultaneously, tasks such as text detection, image reconstruction (enhancement), and character recognition. By learning these tasks in parallel, the model can leverage shared representations that are beneficial for all objectives. This often leads to improved data efficiency (as the model learns more from the same data), enhanced robustness of the learned features (as they must be general enough to serve multiple tasks), and better regularization (as each task acts as a regularizer for the others, reducing the risk of overfitting) [8, 3].

For the image reconstruction aspect within an MTL framework, encoder-decoder architectures like U-Net have demonstrated exceptional performance, particularly in biomedical image segmentation and, more broadly, in tasks requiring high-fidelity image-to-image translation [62, 26]. U-Net's characteristic skip connections, which fuse features from the encoder path directly with corresponding layers in the decoder path, allow the network to effectively combine deep semantic information with fine-grained, low-level details, crucial for accurate image restoration. To further refine the reconstruction process, techniques such as sub-pixel convolution layers (often referred to as PixelShuffle) can be employed for artifact-free upsampling, which is often superior to traditional deconvolution or transposed convolution layers in generating sharper, more detailed output images [68]. Additionally, architectural innovations like Squeeze-and-Excitation (SE) blocks can be integrated. SE blocks adaptively recalibrate the channel-wise feature responses within the network by explicitly modeling interdependencies between channels, allowing the model to emphasize informative features and suppress less useful ones on a task-specific basis [22].

By synergistically integrating these advanced deep learning components, it becomes feasible to construct a single, end-to-end trainable network. Such a network could localize text regions, simultaneously restore the imagery within these regions from various degradations, and then transcribe the characters from the enhanced features, all within one efficient forward pass. This holistic approach holds the potential to significantly improve the robustness and accuracy of scene text recognition systems, particularly in challenging, degradation-prone environments like those encountered in vehicle text images.

1.2 Problem Formulation

We consider a blurred input image

$$X_{\text{blur}} \in \mathbb{R}^{H\times W\times 3},$$

with height H, width W, and 3 color channels, and its corresponding sharp image

$$X \in \mathbb{R}^{H\times W\times 3}.$$

Each image contains N text instances, annotated by $\{(b_i^\star, y_i^\star)\}_{i=1}^{N}$, where $b_i^\star \in \mathbb{R}^4$ is the ground-truth box for instance i and $y_i^\star$ the character sequence.

Our model f_θ (YOLO CRNet) outputs

$$f_\theta :\; X_{\text{blur}} \;\longmapsto\; \big\{(b_i, p_i)\big\}_{i=1}^{M_p} \;\times\; \hat{X} \;\times\; \big\{\, c_i\big\}_{i=1}^{M_p},$$

where M_p is the number of predicted text regions, (b_i, p_i) are the predicted box and confidence, $\hat{X}$ is the reconstructed image, and c_i the predicted character sequence for region i.

We train end-to-end by minimizing the composite loss

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{det}} \; + \; \lambda_{\text{recon}} \, \mathcal{L}_{\text{recon}} \; + \; \lambda_{\text{cls}} \, \mathcal{L}_{\text{cls}}. \tag{1.1}$$

where:

- $\mathcal{L}_{\text{det}}$ is the standard YOLOv8 detection loss.
- $\mathcal{L}_{\text{recon}}$ corresponds to the reconstruction loss.
- $\mathcal{L}_{\text{cls}}$ corresponds to the classification loss.

Hyper-parameters $\lambda_{\text{recon}}, \lambda_{\text{cls}}$ balance the three tasks. Training uses Adam on the dataset $\{X^{(k)}_{\text{blur}}, X^{(k)}, \{b^{(k)}_i, y^{(k)}_i\}\}_{k=1}^{M}$.

1.3 Contributions

The research presented in this thesis makes several key contributions to automatic scene-text detection and recognition, focusing on enhancing robustness against real-world image degradations, particularly in applications like Automatic License Plate Recognition (ALPR). The core of these contributions lies in a novel multi-task learning framework and its comprehensive evaluation.

The key contributions are outlined below:

1. **Development of a Novel Integrated Reconstruction Head within the YOLO Framework:** A novel reconstruction head was designed and seamlessly integrated into the You Only Look Once (YOLO) object detection framework. This integration facilitates the simultaneous execution of object detection and image reconstruction tasks. A key aspect of this design is the utilization of shared features from the YOLO backbone network by both the detection and reconstruction pathways. This approach not only promotes computational efficiency but also encourages the learning of feature representations that are robust and beneficial for both understanding image content and restoring image quality from potential degradations.

2. **Development of an Integrated Classification Head Leveraging Enhanced Features:** An integrated classification head was developed to operate in conjunction with the aforementioned detection and reconstruction mechanisms. This classification head is specifically designed to capitalize on the features that are potentially enhanced or refined by the reconstruction task. By processing these improved features, the classification head aims to achieve higher accuracy in identifying the classes of detected objects. The entire architecture, encompassing detection, reconstruction, and classification, is designed for end-to-end training, enabling the system to holistically optimize all components for improved classification performance directly within the YOLO pipeline.

3. **Investigation of Multi-Task Learning Efficacy and Regularizing Benefits:** This research systematically investigated the efficacy of adopting a multi-task learning (MTL) approach for the concurrent objectives of object detection, image reconstruction, and classification. The study demonstrated that jointly training a single model for these related tasks can yield significant regularizing benefits. These benefits manifest as improved generalization capabilities and robustness of the learned model when compared to traditional single-task baselines or sequentially executed tasks, where each component is optimized in isolation.

4. **Feature Representation Analysis using t-SNE Visualization:** t-Distributed Stochastic Neighbor Embedding (t-SNE) visualization was employed as an analytical tool to examine the characteristics of feature representations learned by different layers within the YOLO framework. Specifically, this analysis focused on assessing the class separability of these features, which is crucial for effective classification. The insights gained from t-SNE visualizations helped to elucidate the relationship between the quality of the learned feature representations at various network depths and the resultant performance of the classification head, providing a deeper understanding of the model's internal workings.

1.4 Outline of my thesis

The remaining chapters of this thesis are structured as follows:

Chapter 2: Literature Review
This chapter provides a comprehensive review of existing literature in scene text detection and recognition, object detection frameworks including the evolution of YOLO and its

specific application to text, the impact of image degradations, image restoration techniques (both classical and deep learning-based), and advanced learning strategies relevant to the proposed research.

Chapter 3: Detailed Problem Formulation
This chapter presents a formal definition of the problem addressed, including the system overview, the unified mapping approach for simultaneous detection, reconstruction, and classification, a detailed breakdown of the composite loss function and its components, and the overall training objective.

Chapter 4: Exploratory Data Analysis
This chapter details the datasets utilized in this research, including their description and preprocessing. It further presents an analysis of class separability through feature embedding extraction from various YOLO layer configurations and dimensionality reduction via t-SNE, concluding with visualizations and an analysis of label distribution and ambiguous character groups.

Chapter 5: Methodology
This chapter outlines the proposed YOLO CRNet architecture in detail, covering the overall pipeline, training flow, and specific designs for the YOLO architecture, the classifier head, and the reconstruction head, including their respective design rationales, loss functions, and strategies for regularization and preventing overfitting.

Chapter 6: Experiments and Results
This chapter describes the experimental setup and evaluation metrics used. It then presents a comprehensive analysis of the results obtained from evaluating the YOLO CRNet classifier head with different layer configurations, the reconstruction head with various designs and loss ablations, and finally, the overall integrated pipeline performance.

Chapter 7: Conclusion and Future Work
The final chapter summarizes the key findings and contributions of this thesis, discusses the challenges encountered and limitations of the current work, and proposes potential directions for future research.

Chapter 2

Literature Review

This chapter provides a comprehensive review of the existing literature relevant to automatic text detection and recognition, with a particular focus on vehicle license plates captured in unconstrained environments. It begins by offering an overview of scene text detection and recognition, highlighting its applications and inherent challenges in Section 2.1. Section 2.2 then delves into object detection frameworks suitable for text, with a specific emphasis on the evolution of YOLO and its strengths and limitations in this context. The profound impact of various image degradations on text recognition performance is examined in Section 2.3. Following this, Section 2.4 discusses both classical and contemporary deep learning-based approaches for image restoration and enhancement. Section 2.5 critically reviews existing workflows commonly employed in ALPR and text recognition, categorizing them into multi-stage pipelines with explicit enhancement, multi-stage systems relying on robust training, and end-to-end methodologies. Finally, Section 2.6 explores advanced learning strategies pertinent to this work, including Multi-Task Learning in computer vision and the use of t-SNE for feature visualization and analysis. This review establishes the context and foundation for the research presented in this thesis.

2.1 Scene Text Detection and Recognition

Scene text detection and recognition, the process of localizing and recognizing text from natural images or videos, is a pivotal area in computer vision due to its wide range of applications. These include Automatic License Plate Recognition (ALPR) for intelligent transportation systems [39], assistive technologies for the visually impaired [7], and real-time translation systems [7]. Unlike Optical Character Recognition (OCR) on scanned

documents, scene text presents significant challenges due to variations in fonts, sizes, orientations, complex backgrounds, uneven illumination, and perspective distortions [19, 53, 44]. Furthermore, text in real-world scenes is often subject to various image degradations, which severely impact the performance of recognition systems [18].

The task is typically broken down into two main stages: text detection (localizing text regions) and text recognition (recognizing the text within the localized regions). Some end-to-end approaches aim to solve both simultaneously [46, 7]. The accurate detection and recognition of text, especially small or arbitrarily shaped text, remains an active research area [53].

2.2 Object Detection Frameworks for Text

While generic object detection methods can be adapted for text detection, specialized techniques are often required due to the unique characteristics of text, such as its variable aspect ratios and sequential nature. Beyond the YOLO [57] family, which will be discussed in detail, the field of scene text detection has seen several other influential frameworks. Many early approaches adapted general object detectors like Faster R-CNN [60] or SSD (Single Shot MultiBox Detector) [43] for text by modifying anchor strategies or post-processing steps to handle elongated text regions. More specialized methods emerged focusing on the specific challenges of text. For instance, CTPN (Connectionist Text Proposal Network) [73] was designed to detect text lines by predicting sequences of fine-scale text proposals and connecting them using a recurrent neural network. Segmentation-based approaches like PixelLink [11] and TextSnake [45] treat text detection as a pixel-level segmentation task, identifying text pixels and then grouping them into instances, which allows for the detection of arbitrarily shaped text. EAST (Efficient and Accurate Scene Text Detector) [86] proposed an efficient single-stage model that directly predicts word or text-line quadrilaterals. More recently, Transformer-based architectures have also been explored for text detection, leveraging attention mechanisms to capture long-range dependencies and improve localization accuracy [55]. These frameworks represent various strategies, from regression-based and segmentation-based to more recent attention-based models, each contributing to the advancements in accurately and efficiently localizing text in natural images.

2.2.1 Evolution of YOLO and its Variants

The "You Only Look Once" (YOLO) family of object detectors marked a paradigm shift by framing object detection as a single regression problem, enabling real-time performance

[57]. YOLO processes the entire image in one forward pass, dividing it into a grid and predicting bounding boxes, confidence scores, and class probabilities for each cell simultaneously. This unified architecture contrasts with earlier multi-stage detectors, such as R-CNN [17] and its variants, which involved separate region proposal and classification steps.

Subsequent versions of YOLO, including YOLOv2 (YOLO9000) [58] and YOLOv3 [59], introduced improvements like anchor boxes, multi-scale feature detection, and more sophisticated backbone networks. These enhancements led to increased accuracy while maintaining high processing speeds. The evolution of YOLO has continued through community-driven efforts and new architectural refinements, exemplified by models like YOLOv8 from Ultralytics [74], which strive for state-of-the-art trade-offs between speed and accuracy [24]. These models typically comprise a backbone for feature extraction, a neck for feature aggregation, and a head for making predictions [35].

2.2.2 YOLO for Text Detection: Strengths and Limitations

YOLO's speed and efficiency make it an attractive choice for real-time text detection applications, including ALPR. Its ability to process the entire image at once provides global context, which can be beneficial for distinguishing text from complex backgrounds [24, 49, 34, 6].

However, standard YOLO architectures encounter several challenges when applied specifically to text detection. Text lines often exhibit extreme aspect ratios and can be very small, characteristics that might overlap [23]. Detecting densely packed or arbitrarily shaped text instances, which are common in general scene text [44], can also prove difficult for standard bounding box prediction mechanisms [53]. Furthermore, standard YOLO models, without specific enhancements or tailored training strategies, are susceptible to significant performance degradation when input images are severely affected by blur, noise, or other artifacts commonly found in vehicle text imagery[6].

2.3 Impact of Image Degradations on Text Recognition

The quality of input images is paramount for achieving accurate text detection and recognition. Images captured by vehicle-mounted cameras or surveillance systems frequently occur

under non-ideal conditions, leading to various forms of degradation that can significantly hinder performance [18].

Common degradations in vehicle text imagery include motion blur, defocus blur, sensor noise, compression artifacts, and environmental factors. Motion blur arises from relative movement between the camera and the license plate or from camera shake, especially prevalent at high speeds or with longer exposure times, resulting in streaked characters and indistinct edges [36, 82]. Defocus blur occurs when the plate is outside the camera's optimal depth of field, a common issue with fixed-focus surveillance cameras or incorrect autofocus, leading to a general loss of sharpness and obscured fine character details. Sensor noise, often exacerbated by low-light conditions, introduces random pixel intensity variations that can mask character details and compromise edge detection [19]. Lossy compression techniques like JPEG, employed for efficient storage and transmission, can introduce blocking artifacts (grid-like patterns) and ringing artifacts (oscillations near sharp edges), both of which distort character shapes. Additionally, environmental factors such as glare from headlights or sunlight, reflections, partial occlusions from dirt or physical damage, and adverse weather conditions like rain, haze, or snow can further diminish text legibility [18]. These degradations, frequently occurring in combination, substantially impair both detection recall and recognition accuracy, heavily affecting the structural similarity and overall quality of the image, which are key aspects of image quality assessment [80]. Figure 2.1 provides an overview of challenges faced in real-world vehicle text detection and classification.

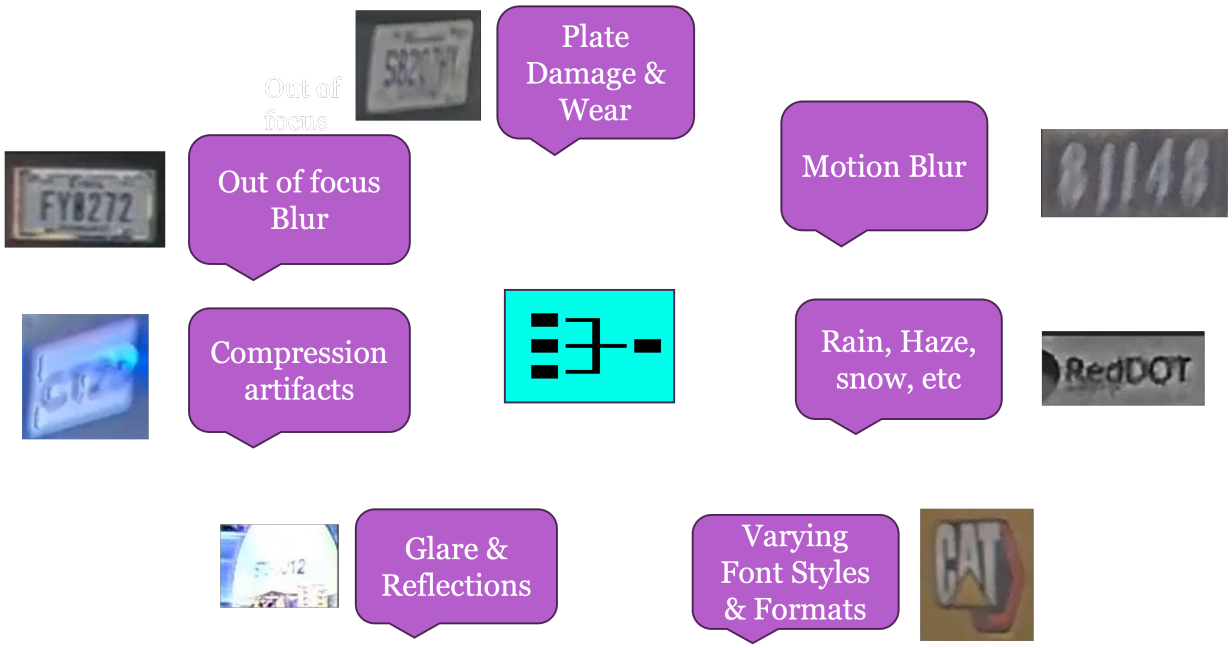


Figure 2.1: Challenges With Real-world Vehicle Text Detection and Classification

2.4 Image Restoration Techniques

Given the detrimental effects of image degradations on text understanding tasks, various image restoration techniques have been developed. These aim to enhance image quality either as a pre-processing step or as an integrated part of the recognition pipeline.

2.4.1 Classical Image Deblurring and Enhancement

Classical deblurring algorithms, such as Wiener filtering and Richardson-Lucy deconvolution, attempt to reverse the blurring process. This typically involves estimating a blur kernel, or Point Spread Function (PSF), and then deconvolving this kernel from the observed image [19]. However, these traditional methods face several significant challenges that limit their effectiveness in real-world scenarios. Many such methods operate under the assumption of a single, spatially invariant blur kernel, an assumption that often does not hold true for images affected by complex, non-uniform, or object-motion blur [36]. The accurate estimation of the blur kernel itself is a critical and extremely difficult task; even

minor inaccuracies in this estimation can lead to substantial deblurring artifacts, such as ringing, or an undesirable amplification of existing image noise [36, 82]. Furthermore, the performance of these classical techniques often depends heavily on meticulous parameter tuning, including regularization weights, which may not generalize effectively across diverse images or varying types of degradation [37]. An additional concern is that inherent image noise can frequently be amplified during the deconvolution process[72]. Other classical enhancement techniques, like histogram equalization and various filtering methods, exist but may offer insufficient improvement for severe or specific degradation types like significant motion blur[19].

2.4.2 Deep Learning-based Image Restoration

Convolutional Neural Networks (CNNs) have demonstrated remarkable success in a variety of image restoration tasks, including deblurring, denoising, and super-resolution, by learning complex mappings from degraded images to their clean counterparts using large datasets [20, 35]. Many CNN architectures for deblurring, often inspired by the U-Net architecture [62], are designed to either estimate the blur kernel or directly regress to a deblurred image, showing capabilities to handle more complex and spatially varying blurs than classical methods [50, 18]. For enhancing image resolution and recovering lost details, super-resolution techniques, such as those employing efficient sub-pixel convolution layers (often termed PixelShuffle), have proven effective [68, 56]. The choice of loss function is critical in training these restoration models; while pixel-wise losses like L_1 and L_2 are common, perceptual losses, which measure similarity in a deep feature space [70], often yield more visually appealing results and achieve better preservation of important textural details [29, 26]. Despite their power, applying these deep learning restoration models as isolated pre-processing steps can introduce computational latency and may not be optimal for subsequent recognition tasks. This is because errors from the restoration stage can propagate, or the restoration process might not specifically emphasize features that are critical for accurate text recognition [13]. Figure 2.2 depicts the deep learning image enhancement application in vehicle text detection.

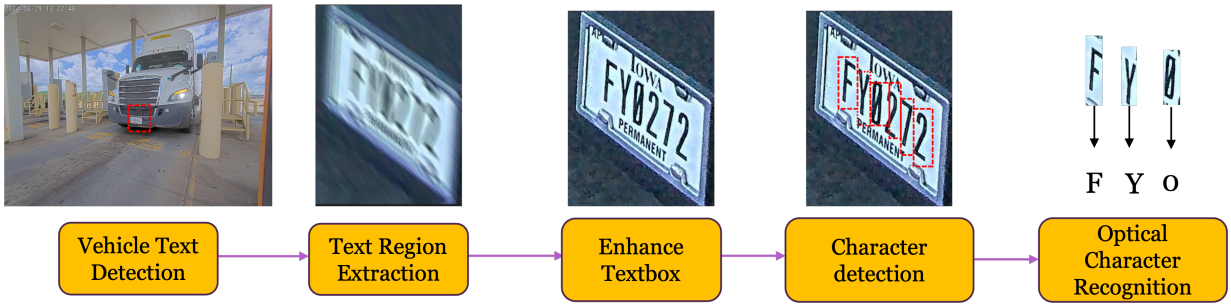


Figure 2.2: Vehicle text recognition workflow with image enhancement

2.5 Existing Workflows in ALPR and Text Recognition

Several distinct workflows are commonly employed for ALPR and general vehicle text recognition, each characterized by its own set of advantages and inherent limitations when dealing with degraded imagery.

2.5.1 Workflow 1: Multi-Stage ALPR with Explicit Deblurring

This sequential approach typically involves first detecting the license plate or text region. Following detection, a dedicated restoration network, which can be based on classical algorithms or, more commonly, deep learning techniques [56, 66, 2], is used to explicitly deblur or enhance the cropped image region. The restored image is then passed to a separate character recognizer for OCR. Pourjam et al. [56] propose a network that first denoises and enhances the cropped plate via an adversarial enhancement GAN, then upsamples it with a super-resolution generative network for sharper input to OCR. Sendjasni et al. [66] introduce a pixel-and-embedding-consistency loss for super resolution via a Siamese network, but recognition remains a downstream, separately trained module and AlHalawani et al. [2] leverage diffusion models to super-resolve plate images. However, this workflow has several common limitations. Restoration networks may introduce new visual artifacts or fail to perfectly reconstruct details, particularly when faced with severe or complex blur types that were not well-represented in their training data [33]. Furthermore, any errors or imperfections remaining after the restoration stage are directly propagated to the recognition module, which can negatively impact the final recognition accuracy [13]. The

sequential execution of multiple complex models inherently increases the overall processing time, potentially creating a bottleneck for real-time applications [16].

2.5.2 Workflow 2: Multi-Stage ALPR with Robust Training (No Explicit Deblurring)

A widely adopted alternative also utilizes separate detection and recognition models but forgoes an explicit image restoration step [38, 49, 34, 85]. Silva et al. [49] and Laroca et al. [34] proposed ALPR systems based on sequence of modified YOLO networks. Silva et al. uses two distinct networks where one detects cars and license plates and the second one to perform OCR. Laroca et al. used five networks for detection, and character recogition. Zherzdev et al. [85] introduces a lightweight end-to-end convolutional network, optional spatial transformer alignment, global context embedding for plate recognition. Li et al. [38] employ a cascade of CNNs comprising of a 37-class character detector, plate/non-plate classifier and a sequence labeller for character recognition. This workflow relies heavily on training both the detector and the recognizer with extensive data augmentation [64]. This augmentation process involves exposing the models to a wide variety of simulated degradations, including different types of blurs, noise, lighting variations, and other distortions, with the aim of making the models inherently robust to such conditions [69]. YOLO is a frequently employed object detector within such robustly trained multi-stage pipelines [49, 34, 6]. This approach has its limitations. The performance is heavily dependent on the augmentation strategy accurately mirroring the diversity and severity of real-world conditions; a mismatch can lead to a domain gap where models still perform poorly on unseen degradation types. While robust training enhances resilience, there may be an inherent performance ceiling when dealing with extremely degraded images [6, 79].

2.5.3 Workflow 3: End-to-End ALPR with Robust Training

This paradigm seeks to use a single, unified deep learning model to perform both text region localization and character sequence recognition simultaneously [46, 7, 53]. Such models are trained end-to-end, often incorporating significant data augmentation strategies to achieve robustness against image degradations [53]. Bartz et al. [7] introduces a unified semi-supervised text spotting model that integrates a spatial-transformer detection head with a recognition branch and is trained end-to-end on both labeled and unlabeled data. Wang et al. [53] present a rapid arbitrary-shaped text detector featuring a lightweight segmentation head and learnable post-processing that plugs into end-to-end

recognition pipelines. Ma et al. [46] proposes an end-to-end light LP detection and recognition network which fuses the detection module and recognition module to generate a parallelogram bounding box with a rotating angle and recognize the characters. Similar to the workflow 2, this approach also doesn't consider the imperfections of the scene text and cause loss of semantic information in scene text regions such as blurring and occlusions [27]. To mitigate this issue, increased training complexity and data requirements are required. Optimizing a single, potentially very deep network for multiple distinct sub-tasks is complex [21]. Moreover, feature sharing, while promoting computational efficiency, might lead to performance trade-offs where the unified model does not achieve the absolute peak performance on both detection and recognition when compared against highly specialized, separate models optimized for each individual task [27]. Interpreting failure modes in these complex end-to-end systems can also present greater difficulty [63].

2.6 Advanced Learning Strategies and Analysis

2.6.1 Multi-Task Learning in Computer Vision

Multi-Task Learning (MTL) is a learning paradigm wherein a single model is trained to perform multiple related tasks concurrently. This is typically achieved by sharing parts of the network architecture, such as early layers for common feature extraction, while employing task-specific heads for generating the final predictions for each task [10, 63]. Key characteristics of MTL include the use of shared representations, where common, generalizable features useful for all tasks are learned. This can be realized through hard parameter sharing, where lower layers are shared before branching out, or soft parameter sharing , where each task has its own model but parameters are constrained to encourage similarity [10, 63]. The model optimizes a combined objective function, usually a weighted sum of losses from individual tasks [3].

There are several benefits for MTL. It can lead to improved data efficiency by leveraging training signals from all tasks, as knowledge gained for one task can implicitly aid related tasks [46]. It encourages the learning of more robust and generalizable representations because these features must be effective across different objectives, thereby acting as a form of regularization that can reduce overfitting compared to single-task training [8]. Furthermore, MTL can result in models with fewer total parameters and potentially faster inference times due to shared computations, compared to deploying multiple independent models. MTL has found successful applications in various computer vision problems, such as joint object detection and recognition [14, 46, 7, 53].

2.6.2 Feature Visualization and Analysis with t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a widely used dimensionality reduction technique designed for visualizing high-dimensional datasets in a low-dimensional space, typically two or three dimensions. It operates by modeling similarities between high-dimensional data points as conditional probabilities and then seeks a low-dimensional embedding that minimizes the divergence between these respective probability distributions [77].

Applying t-SNE allows for the analysis of class separability by visualizing whether features corresponding to different object classes form distinct, well-separated clusters in the embedded space; such clustering indicates that the network has learned discriminative features. It also enables an understanding of layer-wise feature evolution, showing how the network progressively transforms raw input data into more abstract and separable representations [32]. Furthermore, t-SNE can aid in debugging models by helping to identify potential issues, such as classes that are consistently confused by the model, which might manifest as significantly overlapping feature clusters in the t-SNE visualization [75]. However, effective use and interpretation of t-SNE plots require careful consideration of its hyperparameters and an awareness of its limitations, as it primarily focuses on preserving local data structure, meaning the global arrangement and relative sizes of clusters can sometimes be misleading [81].

Chapter 3

Detailed Problem Formulation

This chapter elaborates on the problem addressed in this thesis which is the robust detection and recognition of text from degraded images, with a specific focus on vehicle text detection. It begins by outlining the overall system objectives, defining the inputs, desired outputs, and the unified mapping approach in Section 3.1. Following this, Section 3.2 meticulously breaks down the composite loss function used to train the proposed YOLO CRNet model. This includes detailed explanations of the detection loss ($\mathcal{L}_{\text{det}}$) in Subsection 3.2.1, the reconstruction loss ($\mathcal{L}_{\text{recon}}$) and its components in Subsection 3.2.2, and the classification loss ($\mathcal{L}_{\text{cls}}$) in Subsection 3.2.3. Finally, Section 3.3 describes the overall training objective, the dataset structure, and the optimization strategy employed.

3.1 System Overview and Unified Mapping

The central challenge is to accurately detect and transcribe text from images that are frequently damaged by real-world degradations such as motion blur, defocus blur, sensor noise, and compression artifacts. These issues are particularly prevalent in Automatic License Plate Recognition (ALPR) scenarios and significantly impair the performance of traditional systems. This work proposes a unified framework, YOLO CRNet, designed to concurrently perform text detection, image reconstruction, and character sequence recognition in an end-to-end fashion.

Let $X_{\text{blur}} \in \mathbb{R}^{H \times W \times 3}$ represent an input blurred image, where H and W are its height and width, respectively, and 3 denotes the RGB color channels. The corresponding original, non-degraded (sharp) version of this image is denoted by $X \in \mathbb{R}^{H \times W \times 3}$. Each image X_{blur}

and its sharp counterpart X is assumed to contain $N \geq 0$ text instances. For each text instance i in an image, ground-truth annotations are available, denoted as $(b_i^\star, y_i^\star)$, where $b_i^\star \in \mathbb{R}^4$ is the ground-truth bounding box outlining the spatial dimensions of the text, and $y_i^\star$ is the corresponding ground-truth character (label) for that text instance.

The core objective is to learn a unified mapping function, f_θ, parameterized by a set of learnable weights θ (representing our YOLO CRNet model), that takes the blurred input image X_{blur} and produces a comprehensive set of outputs. This mapping is defined as:

$$f_\theta : X_{\text{blur}} \longmapsto \left\{ (b_i, p_i) \right\}_{i=1}^{M_p} \times \hat{X} \times \{c_i\}_{i=1}^{M_p} \tag{3.1}$$

where M_p is the number of text instances predicted by the model. The specific outputs, corresponding to the different heads of the YOLO CRNet architecture are:

- $\left\{ (b_i, p_i) \right\}_{i=1}^{M_p}$: The output of the text detection task. Each $b_i \in \mathbb{R}^4$ is a predicted bounding box, and p_i is its associated confidence score (indicating text presence and localization precision). This is handled by the YOLOv8 detection head.
- $\hat{X}$: The reconstructed image, $\hat{X} \in \mathbb{R}^{H \times W \times 3}$, generated by the reconstruction model component. It aims to restore X_{blur} to approximate the original sharp image X.
- $\{c_i\}_{i=1}^{M_p}$: The output of the text recognition task. For each detected region b_i, c_i is the predicted character sequence, produced by the classification model component operating on enhanced features.

3.2 Loss Function Formulation

The parameters θ of the entire YOLO CRNet are optimized by minimizing a composite loss function $\mathcal{L}_{\text{total}}$. This loss integrates objectives from the three primary tasks: detection, reconstruction, and classification. The overall structure of this multi-task loss is:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{det}} + \lambda_{\text{recon}} \mathcal{L}_{\text{recon}} + \lambda_{\text{cls}} \mathcal{L}_{\text{cls}} \tag{3.2}$$

Here, $\mathcal{L}_{\text{det}}$, $\mathcal{L}_{\text{recon}}$, and $\mathcal{L}_{\text{cls}}$ represent the loss terms for detection, reconstruction, and classification, respectively. The hyperparameters λ_{recon} and λ_{cls} are crucial scalar weights that balance the contribution of the reconstruction and classification tasks relative to the primary detection task and to each other during joint optimization. Each loss component is detailed in the following subsections.

3.2.1 Detection Loss ($\mathcal{L}_{\text{det}}$)

The text detection loss, $\mathcal{L}_{\text{det}}$, is responsible for training the network to accurately localize text instances within the input image. This component is based on the standard loss function utilized in the YOLOv8 object detection framework [24, 57]. The YOLOv8 loss is itself a multi-component loss that typically includes:

$$\mathcal{L}_{\text{det}} = \underbrace{\sum_{i=1}^{M_p} \mathcal{L}_{\text{CIoU}}(b_i, b_i^\star)}_{\substack{\text{box} \\ \text{regression}}} + \underbrace{\sum_{i=1}^{M_p} \mathcal{L}_{\text{DFL}}(b_i, b_i^\star)}_{\substack{\text{distribution} \\ \text{focal loss}}} + \underbrace{\sum_{i=1}^{M_p} \text{BCE}(p_i, p_i^\star)}_{\substack{\text{objectness} \\ \text{loss}}} \tag{3.3}$$

- **Bounding-box regression (CIoU).** Complete Intersection over Union (CIoU) loss penalizes overlap, center-distance, and aspect-ratio mismatches for tighter localization.
- **Distribution Focal Loss (DFL).** Models each box edge as a discrete distribution over bins, improving boundary precision.
- **Objectness loss.** Binary Cross-Entropy (BCE) loss measures the performance of a binary classification model. BCE on p_i trains the network to distinguish "object" (text) vs. background.

End-to-End Learning Because detection happens on $\hat{X}$ and all YOLOv8 parameters remain trainable, gradients from $\mathcal{L}_{\text{det}}$ flow back through both the detection head and the reconstruction head. This joint update encourages reconstructions that not only look sharp but also yield accurate detections.

3.2.2 Reconstruction Loss ($\mathcal{L}_{\text{recon}}$)

The image reconstruction loss, $\mathcal{L}_{\text{recon}}$, weighted by λ_{recon}, guides the reconstruction head of YOLO CRNet to transform the input blurred image X_{blur} into an enhanced image $\hat{X}$ which is a restoration of the original sharp image X. This loss is a carefully designed combination of pixel-wise losses and a perceptual loss to ensure both fidelity at the pixel level and preservation of higher-level image characteristics. Specifically, L_1 loss and Perceptual loss

are typically weighted higher compared to L_2 loss to encourage sharp reconstructions while maintaining stability. The formulation is:

$$\mathcal{L}_{\text{recon}} = w_1\mathcal{L}_{L1}(\hat{X}, X) + w_2\mathcal{L}_{L2}(\hat{X}, X) + w_3\mathcal{L}_{\text{perc}}(\hat{X}, X) \tag{3.4}$$

where w_1, w_2, w_3 are weighting coefficients for each component, with w_1 and w_3 typically being larger than w_2.

Pixel-wise Losses ($\mathcal{L}_{L1}$ and $\mathcal{L}_{L2}$) These losses measure the direct discrepancy between the pixel values of the reconstructed image $\hat{X}$ and the original sharp image X.

- The L_1 loss (Mean Absolute Error) is defined as:

$$\mathcal{L}_{L1}(\hat{X}, X) = \frac{1}{CHW}\sum_{c=1}^{C}\sum_{h=1}^{H}\sum_{w=1}^{W}|X_{c,h,w} - \hat{X}_{c,h,w}| \tag{3.5}$$

  L_1 loss is known for encouraging sharpness in the reconstructed images and being less sensitive to outliers compared to L_2 loss [15].

- The L_2 loss (Mean Squared Error) is defined as:

$$\mathcal{L}_{L2}(\hat{X}, X) = \frac{1}{CHW}\sum_{c=1}^{C}\sum_{h=1}^{H}\sum_{w=1}^{W}(X_{c,h,w} - \hat{X}_{c,h,w})^2 \tag{3.6}$$

  L_2 loss penalizes larger errors more heavily and can contribute to smoother reconstructions and optimization stability, though it may sometimes lead to blurrier results if used weighted too highly [15].

Perceptual Loss ($\mathcal{L}_{\text{perc}}$) The perceptual loss aims to capture a higher-level notion of similarity based on semantic features rather than just raw pixel values [15, 29]. A pre-trained deep convolutional neural network, typically VGG19 [70], is used as a fixed feature extractor. Both the reconstructed image $\hat{X}$ and the ground-truth sharp image X are passed through this network. Feature maps are extracted from one or more intermediate layers of the VGG19 network. The perceptual loss is then calculated as the squared Euclidean (L_2) distance between these corresponding feature map activations:

$$\mathcal{L}_{\text{perc}}(\hat{X}, X) = \sum_j \frac{1}{C_j H_j W_j}\|\phi_j(\hat{X}) - \phi_j(X)\|_2^2 \tag{3.7}$$

where $\phi_j(\cdot)$ represents the feature map extracted from the j-th selected layer of the VGG19 network, and C_j, H_j, W_j are the dimensions (channels, height, width) of that feature map. This feature-space distance encourages the reconstructed image to be perceptually similar to the ground truth, often leading to more visually realistic results.

3.2.3 Classification Loss ($\mathcal{L}_{\text{cls}}$)

The text-recognition head operates on features cropped from the YOLOv8 feature maps at the ground-truth box locations for each character. Concretely, for each detected instance i and each character position j, we:

Extract the multi-scale feature maps $\{F^n\}_{n=1}^N$ from the YOLO backbone. Crop each $F^n \in \mathbb{R}^{C_n \times H_n \times W_n}$ spatially using the j–th ground-truth box $b_{i,j}^\star$, yielding $\tilde{F}_{i,j}^n$. Apply average pooling to $\tilde{F}_{i,j}^n$ (over its $H_n \times W_n$ spatial dims) to obtain a channel vector $v_{i,j}^n \in \mathbb{R}^{C_n}$. Concatenate $\{v_{i,j}^n\}_{n=1}^N$ into a single 1-D feature $z_{i,j} \in \mathbb{R}^{\sum_n C_n}$. Pass $z_{i,j}$ through the classifier head, producing logits $\ell_{i,j} \in \mathbb{R}^K$, where K is the size of the character vocabulary.

Let the ground-truth character at position j be $y_{i,j}^\star \in \{1, \ldots, K\}$. We apply the standard cross-entropy loss at each position:

$$\mathcal{L}_{\text{cls}} = \sum_{i=1}^{M_p} \sum_{j=1}^{L_i} - \log\big(\text{softmax}(\ell_{i,j})_{\,y_{i,j}^\star}\big) \tag{3.8}$$

where L_i is the number of characters in instance i, and $\text{softmax}(\ell)_k$ denotes the k–th entry of the softmax over ℓ. Finally, we weight by λ_{cls} in the total loss (Eq. 3.3) to balance against detection and reconstruction.

This formulation ensures that the classifier head trained end-to-end along with the YOLO backbone and reconstruction head, learns to map cropped, YOLO features directly to character probabilities via cross-entropy.

3.3 Training Objective

The overall training objective is to find the optimal set of network parameters $\theta^\star$ that minimizes the total composite loss $\mathcal{L}_{\text{total}}$ (defined in Equation 3.3) over the entire training dataset $\mathcal{D}$. The dataset $\mathcal{D} = \{(X_{\text{blur}}^{(k)}, X^{(k)}, \{(b_i^{(k)}, y_i^{(k)})\})\}_{k=1}^M$ consists of M tuples, each containing an input blurred image $X_{\text{blur}}^{(k)}$, its corresponding original sharp version $X^{(k)}$, and the set of ground-truth text bounding boxes and character labels for that image.

Chapter 4

Exploratory Data Analysis

This chapter presents a detailed exploratory data analysis (EDA) foundational to the experimental work in this thesis. The primary aim of this EDA is to understand the characteristics of the dataset used for training and evaluation, and to investigate the inherent separability of character classes based on features extracted from the YOLO architecture. Section 4.1 begins by describing the synthetic dataset constructed for this research, including its generation process and annotation details. Following this, Section 4.2 delves into an extensive class separability analysis; this includes the methodology for extracting feature embeddings from various YOLO layer configurations, the specifics of the t-SNE dimensionality reduction technique employed, the criteria for evaluating cluster quality, and a presentation of the resulting t-SNE visualizations. Finally, Section 4.3 examines data imbalances and the challenges posed by visually ambiguous character groups, discussing the overall label distribution, identifying specific ambiguous character sets, outlining their impact on model training, and briefly touching upon mitigation strategies considered.

4.1 Dataset Description

We constructed a fully synthetic dataset of paired sharp and blurred images to train and evaluate YOLO CRNet. Each image pair consists of a sharp, ground-truth image and a corresponding synthetically blurred version. Sharp images are generated on a 512×512 pixel, 8-bit grayscale canvas (96 dpi) by rendering random sequences of alphanumeric characters drawn from 62 classes (digits [0–9], lowercase [a–z], uppercase [A–Z]). To increase variability, character appearance is randomized through multiple font selections,

scale factors, rotations and spatial placements, and salt-and-pepper noise of varying density is added as background clutter.

Blurred images are produced by applying a synthetic motion-blur kernel to each sharp image. The kernel length is sampled uniformly from the odd integers $\{3, 5, \ldots, 19\}$ pixels, and its orientation is drawn uniformly from the full angular range $[0°, 360°)$. This process emulates real-world motion artifacts and challenges the components of the model.

Ground-truth annotations for each character instance consist of one line per character with five space-separated values: the class index (0–61) followed by the normalized bounding box coordinates $(x_{\text{ctr}}, y_{\text{ctr}}, \text{width}, \text{height})$, where $x_{\text{ctr}}, y_{\text{ctr}}$ denotes the center coordinates of the bounding box and width, height denotes the width and height of the bounding box respsectively.

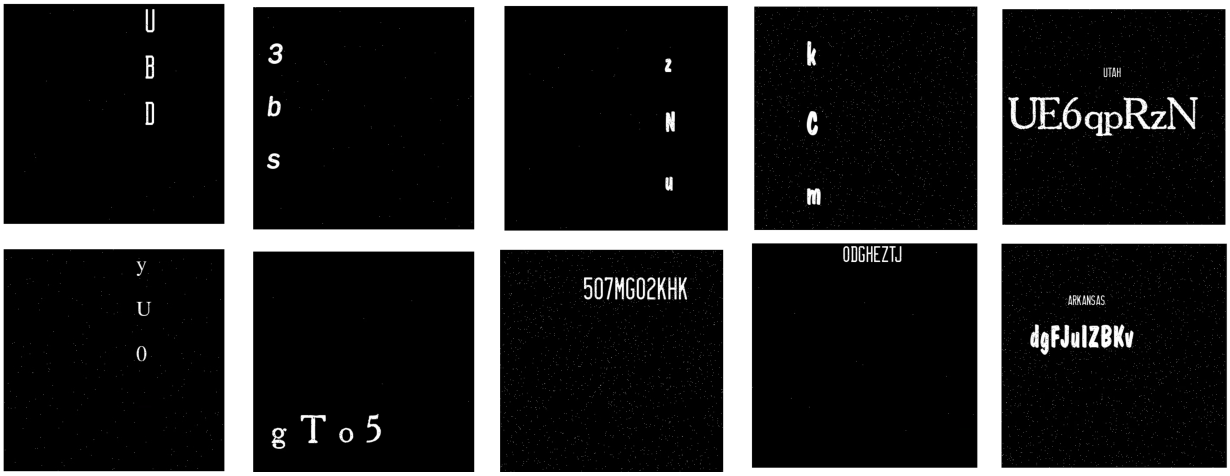


Figure 4.1: Sharp Image samples from the dataset

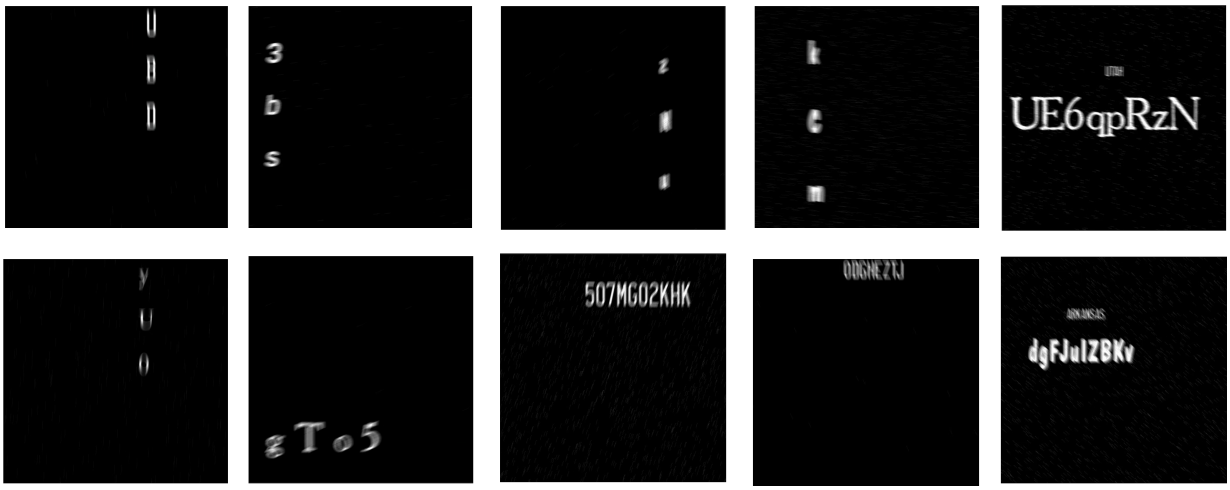


Figure 4.2: Blur Image samples from the dataset

4.2 Class Separability Analysis

To assess the character class separability of our YOLO feature embeddings, we performed a detailed t-SNE visualization study on the processed feature vectors. In what follows, we describe the dataset used for t-SNE visualization, embedding extraction pipeline, YOLO layer configurations experimented, dimensionality reduction procedure, and evaluation criteria.

4.2.1 Dataset and Preprocessing

The training set comprises 10 428 images containing 59 323 annotated characters spanning 62 classes (10 digits, 26 uppercase, 26 lowercase).

1. Each image has corresponding ground truth bounding boxes $b_j = (x_j, y_j, w_j, h_j)$ for character j.
2. Bounding boxes are used to crop the corresponding spatial regions from the YOLO feature maps (see Section 4.2.2).

4.2.2 Feature Embedding Extraction

The input image is passed through a pretrained YOLO detector, and we select a subset of layers

$$\mathcal{L} = \{\, i_1, i_2, \ldots, i_L \}$$

from which to extract feature maps. For each layer $i \in \mathcal{L}$, let feature map be

$$\mathbf{F}_i \;\in\; \mathbb{R}^{H_i \times W_i \times C_i}$$

denote its output, where

- H_i is the feature map height,
- W_i is the feature map width,
- C_i is the number of channels.

Each character instance $j \in \{1, \ldots, N\}$ has a ground-truth bounding box

$$b_j = (x_j,\, y_j,\, w_j,\, h_j),$$

where

- x_j and y_j are the center coordinates of the box in the feature map frame,
- w_j and h_j are the box's width and height (in feature map pixels).

We crop the feature-map to extract

$$\mathbf{F}_i^{(b_j)} \;\in\; \mathbb{R}^{h_j \times w_j \times C_i},$$

the region of $\mathbf{F}_i$ centered at (x_j, y_j) with spatial dimensions $h_j \times w_j$.

We then apply channel-wise average pooling over the cropped spatial dimensions:

$$\mathbf{f}_i^{(j)} = \frac{1}{h_j\, w_j} \sum_{u=1}^{h_j} \sum_{v=1}^{w_j} \mathbf{F}_i^{(b_j)}(u, v, :) \;\in\; \mathbb{R}^{C_i},$$

where this *average pooling* operation computes, for each channel C, the arithmetic mean of the $h_j \times w_j$ activations in that channel's crop. Basically, it sums the feature responses over

all spatial locations (u, v) within the bounding box region and divides by $h_j w_j$, yielding a single representative scalar per channel. The result is a C_i-dimensional vector $\mathbf{f}_i^{(j)}$ that summarizes the spatial content of layer i for character instance j.

Finally, we concatenate these per-layer vectors across all $i \in \mathcal{L}$:

$$\mathbf{f}^{(j)} = \left[\mathbf{f}_i^{(j)}\right]_{i \in \mathcal{L}} \in \mathbb{R}^{\sum_{i \in \mathcal{L}} C_i},$$

so that the length of the final embedding for each character j is $\sum_{i \in \mathcal{L}} C_i$.

Figure 4.3 depicts the Feature Embedding Extraction workflow for t-SNE Plot visualization.

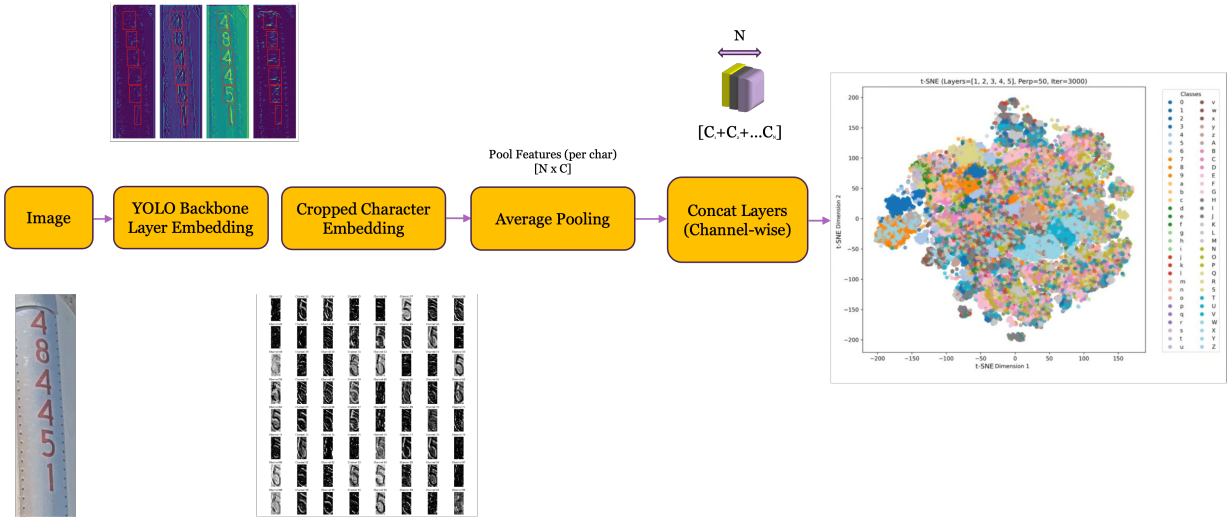


Figure 4.3: Feature Embedding Extraction workflow for t-SNE Plot visualization

4.2.3 YOLO Layer Configurations

We evaluated five distinct YOLO layer configurations $\mathcal{L}$ to understand how shallow vs. deep features affect class separability. Table 4.1 provides the YOLO layer configurations with its ID and depth within the YOLO network architecture.

4.2.4 Dimensionality Reduction via t-SNE

We apply t-SNE to the full set of cropped high-dimensional embeddings

$$\{\mathbf{f}^{(j)}\}_{j=1}^{N}, \quad \mathbf{f}^{(j)} \in \mathbb{R}^D, \quad here, N = 59{,}323,$$

Configuration ID	YOLO Layers	Layer Category
A	[1, 2, 4, 10, 13, 16]	Early Backbone, Early Neck, Deep Neck
B	[1,2,3,4,5]	Early Backbone
C	[8, 9, 10, 13, 16]	Deep Backbone, Neck
D	[9,10,16,19,20]	Deep Backbone, Various Neck Layers
E	[16,17, 19,20]	Various Neck Layers

Table 4.1: YOLO Layer Configurations

where each $\mathbf{f}^{(j)}$ is the j-th feature vector of dimensionality D. We fix the number of gradient-descent iterations to 3000. We sweep the perplexity parameter

$$p \in \{30, 40, 50\},$$

which controls the effective number of local neighbors in the high-dimensional space, and select $p = 50$ for all reported plots as it clusters the best.

t-SNE first converts pairwise Euclidean distances in the original space into conditional probabilities

$$p_{j|i} = \frac{\exp\bigl(-\|\mathbf{f}^{(i)} - \mathbf{f}^{(j)}\|^2/2\sigma_i^2\bigr)}{\sum_{k\neq i}\exp\bigl(-\|\mathbf{f}^{(i)} - \mathbf{f}^{(k)}\|^2/2\sigma_i^2\bigr)},$$

where each bandwidth σ_i is chosen so that the Shannon entropy of the distribution $\{p_{j|i}\}_j$ equals $\log_2 p$, i.e.

$$\mathrm{Perp}(P_i) = 2^{-\sum_j p_{j|i}\log_2 p_{j|i}} \approx p.$$

The joint high-dimensional similarity is symmetrized as

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}.$$

In the low-dimensional map $\{\mathbf{y}_i\}_{i=1}^N \subset \mathbb{R}^2$, similarities are measured with a Student t–distribution (one degree of freedom) to allow heavier tails:

$$q_{ij} = \frac{\bigl(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2\bigr)^{-1}}{\sum_{k\neq\ell}\bigl(1 + \|\mathbf{y}_k - \mathbf{y}_\ell\|^2\bigr)^{-1}}.$$

The embedding $\{\mathbf{y}_i\}$ is found by minimizing the Kullback–Leibler divergence

$$C = \sum_{i\neq j} p_{ij}\,\log\frac{p_{ij}}{q_{ij}}$$

via gradient descent. During the first phase, the p_{ij} are multiplied by a constant (default 4) to encourage formation of tight clusters. To scale to large N, scikit-learn's Barnes–Hut approximation reduces the cost of computing all pairwise forces from $O(N^2)$ to $O(N \log N)$ by organizing points in a quadtree [76].

4.2.5 Evaluation Criteria

The quality of the feature representations learned by the model for individual characters, is primarily assessed through qualitative visual examination. This assessment centers on the structural characteristics of the clusters formed by feature vectors corresponding to characters of the same class when embedded into a 2D space. Two principal aspects of cluster quality are scrutinized: **compactness** and **separation**.

Compactness (Intra-Class Variance) Compactness refers to the degree of cohesion among data points (feature vectors) that belong to the same character class within the t-SNE visualization. A high level of compactness implies that the learned features for a given character are consistent and closer in the feature space, despite variations in the input images such as differing fonts, minor noise levels, or distortions. Visually, in the t-SNE plot, compact clusters manifest as dense, tightly-knit groupings of points sharing a common label. The analysis focuses on the intra-class variance observed in the 2D embedding. Low intra-class variance suggests that the model has effectively learned a feature representation that is robust to variations and can consistently map instances of the same character to a localized region in the feature space. Conversely, scattered or diffuse clusters for a single character class would indicate higher intra-class variance, implying less discriminative for that particular character.

Separation (Inter-Class Margins and Overlap) Separation evaluates how distinct and well-distanced the clusters representing different character classes are from one another in the t-SNE plot. Effective separation implies that the learned features for different characters are sufficiently dissimilar, which is crucial for enabling a subsequent classifier to accurately distinguish between them with high confidence. The visual assessment of separation involves examining the inter-cluster margins and the extent of overlap between clusters. Ideally, clusters corresponding to different characters should be clearly separated by margins or regions of lower point density between them. Wider margins are indicative of superior feature discriminability, suggesting that a decision boundary implemented by a classifier in this feature space would be clearer. Significant overlap or intermingling of

points from clusters of different character classes signals feature ambiguity. For instance, if the t-SNE plot reveals substantial overlap between the clusters for visually similar characters (e.g., 'O' and '0', or 'I' and 'l'), it implies that the learned features for these characters lack sufficient distinction. Such ambiguity would likely correlate with higher confusion rates during the classification stage. The nature and extent of this overlap for various character pairs form a key aspect of this qualitative analysis.

4.2.6 Visualizations

Figure 4.4 and 4.5 shows the different t-SNE plots for all labels and all digits with different YOLO layer configuration respectively.

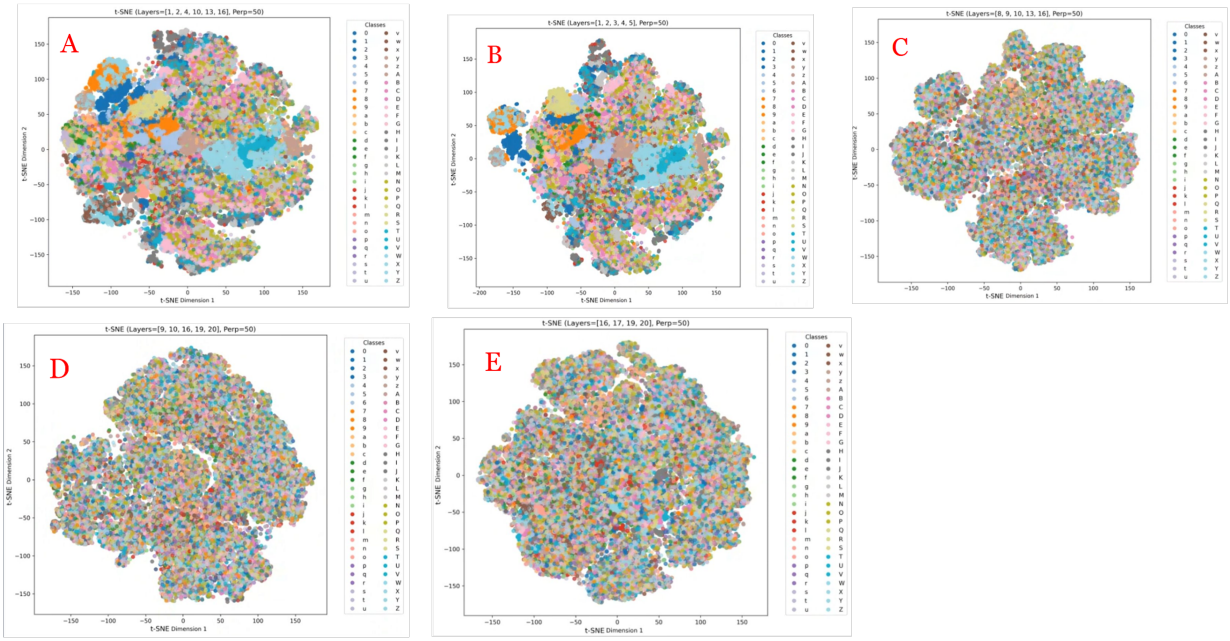


Figure 4.4: t-SNE plots for all labels with different YOLO layer configurations (A-E)

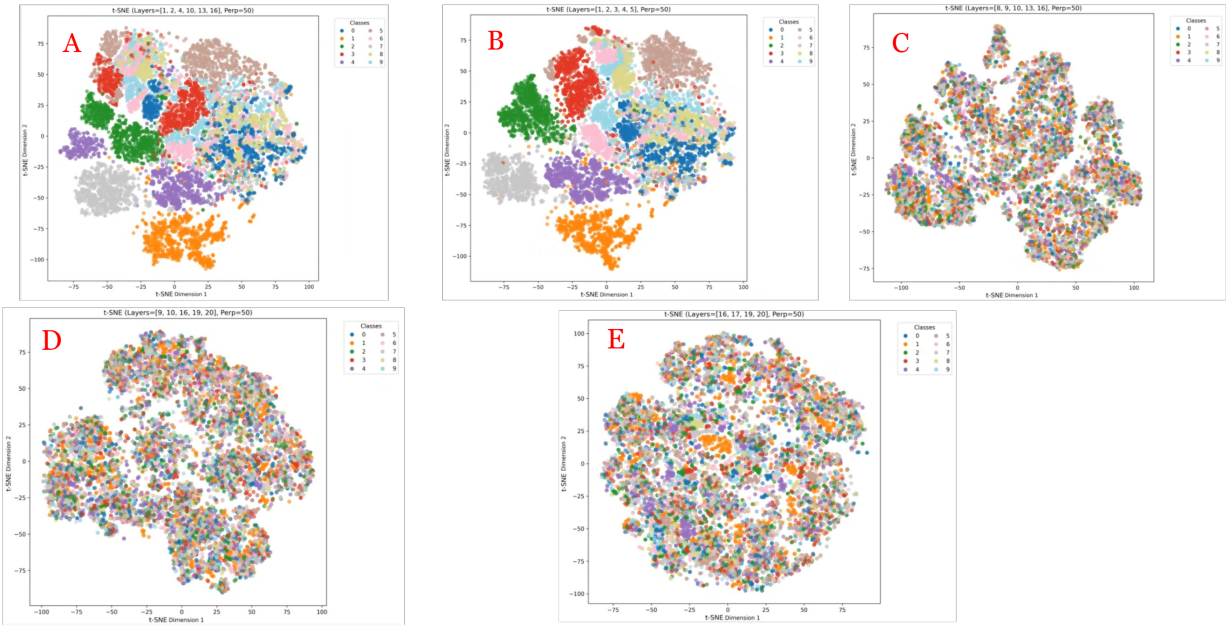


Figure 4.5: t-SNE plots for all digits with different YOLO layer configurations (A-E)

From the figures 4.4 and 4.5 we can observe that Configuration A consistently yielded the tightest, most isolated clusters, guiding our final choice of layers for the classification head. The t-SNE plot analysis indicates that the features that distinguish the classes in the dataset are low-level visual properties like basic outlines, corners, curves, stroke thickness, and serifs that are best captured by the initial layers of the YOLO backbone. The deeper layers (later backbone, neck, head) learn increasingly abstract and complex representations. YOLO's later layers are tailored to feed the detection heads, predicting bounding boxes, objectness, and class probabilities for detected objects. These features encode strong spatial information that aren't as relevant for the classification task. The character classification relies on global texture, color and shape rather than localized objects. The combination of strong signals from early layer and signals from later backbone and neck is ideal to learn features relevant for character classification.

4.3 Imbalance Analysis and Ambiguous Character Separability

In this work, we observe two closely-related challenges in training a generalized classifier: (1) a class imbalance across the labels; and (2) the presence of visually ambiguous character groups that further increase classification errors. Below we quantify both phenomena and discuss their implications.

4.3.1 Overall Label Distribution

Figure 4.6 shows the per-class occurrence counts for all 62 alphanumeric labels in our dataset. On average, uppercase letters (`A`–`Z`) appear roughly 1400-1800 times each, digits (`0`–`9`) around 800-900 times, and lowercase letters (`a`–`z`) only about 300-400 times. This corresponds to a $\sim$5$\times$ imbalance between the most and least frequent classes, which can bias standard training objectives toward majority classes and degrade performance on minority ones.

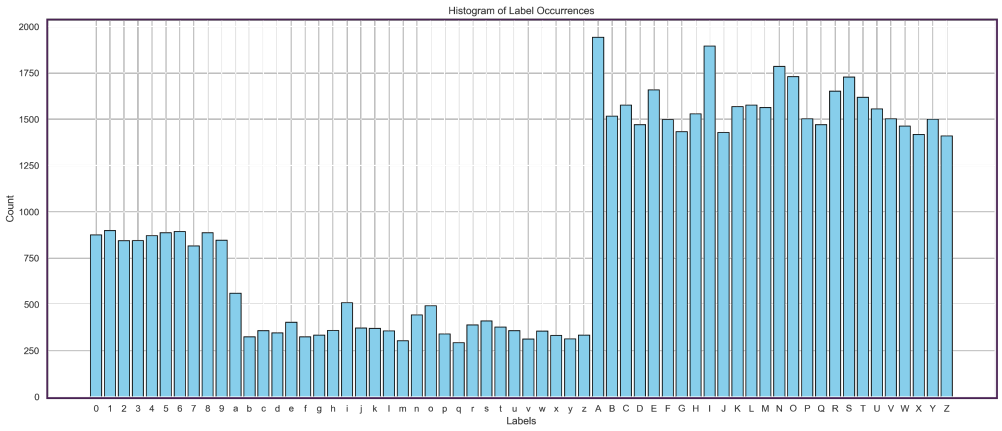


Figure 4.6: Histogram of label occurrences for digits (`0`–`9`), lowercase (`a`–`z`), and uppercase (`A`–`Z`) characters.

4.3.2 Ambiguous Character Groups

Beyond global frequency skew, certain small sets of characters share very similar visual appearance, making them especially prone to misclassification. Table 4.2 lists the six ambiguous groups studied in this work. We projected YOLO feature-map activations into 2D via t-SNE, as discussed in section 4.2, to assess how well each group separates in feature space (Figure 4.7).

Group ID	**Characters analyzed**
A	[O,o,o]
B	[1,1,i]
C	[2,z,Z]
D	[i,I,L]
E	[5,s,S]
F	[c,C,G]

Table 4.2: Ambiguous character configurations evaluated via t-SNE.

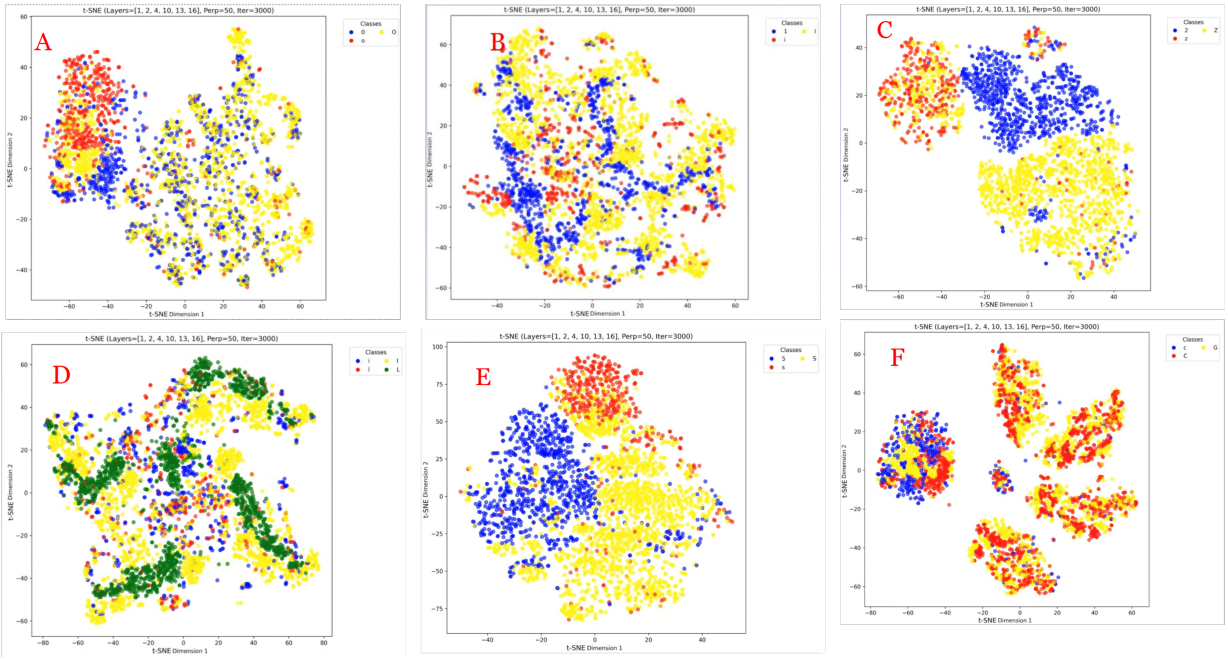


Figure 4.7: t-SNE embeddings of feature vectors for each ambiguous group (A–F). The colored points correspond to the different character clusters. Some groups (e.g. E, F) form well-separated clusters, while others (e.g. B, D) exhibit significant overlap.

4.3.3 Impact on Model Training

The combined effect of class imbalance and ambiguity is two-fold:

- **Majority-class bias.** Standard cross-entropy will be dominated by well-represented uppercase classes, causing the model to underfit minority lowercase characters and digits.
- **Ambiguity-induced confusion.** Overlapping feature clusters in t-SNE indicate that the network's embedding space does not sufficiently discriminate among these visually similar characters, leading to high misclassification rates.

4.3.4 Mitigation Strategies

To address these challenges, we integrate the following measures into our training pipeline:

- **Enhanced Class Separability.** We inspected YOLO's intermediate feature layers via t-SNE, and selectively chose those layers whose embeddings produce the most distinct class clusters for feature extraction and loss computation.
- **Shared Representations via Multi-task Learning.** Multi-task learning forces the model to learn features that are useful across different, related tasks. This shared representation can better capture subtle differences, such as those between ambiguous character pairs.
- **Regularization through Auxiliary Tasks.** By training on detection, reconstruction and classification tasks simultaneously, YOLO model is regularized, guiding the network to learn more general features, not just the dominant classes.
- **Feature Discrimination via Reconstruction Attention.** The reconstruction head compels the model to focus on fine spatial and structural details (e.g. stroke endings, curvature), which are critical for distinguishing between characters like `l` vs. `1` or `c` vs. `G`. This attention helps the network differentiate between very similar characters by emphasizing unique patterns and details that might otherwise be overlooked.

By combining these strategies, we significantly reduce the adverse effects of data imbalance and ambiguity, yielding a more uniform performance across all character classes.

Additionally, the evaluation metrics used to assess model performance should take into account the imbalance in class labels by providing more importance to achieving accurate prediction results on the minority class[20].

Chapter 5

Methodology

This chapter describes the design and implementation of the proposed YOLO CRNet framework. Section 5.1 presents the overall pipeline architecture and training flow; Section 5.2 details the YOLO backbone, neck, and head components and their feature representations; Section 5.3 describes the classifier head architecture, input features, design rationale, and training loss; Section 5.4 outlines the reconstruction head workflow, architectural configurations, loss functions, and overfitting prevention strategies.

5.1 Pipeline

Our proposed YOLO CRNet extends the standard YOLO framework with two parallel heads: a reconstruction head and a classification head which are trained in a unified, multi-task fashion. Figures 5.1 and 5.2 illustrate the high-level architecture and the end-to-end training flow, respectively.

5.1.1 Architecture Overview

Training images are first passed through a standard YOLO backbone, which produces deep feature maps encoding spatial and semantic cues. These shared features then branch into two heads:

- Reconstruction Head A lightweight decoder that upsamples and refines the backbone's feature maps to reconstruct an enhanced image. By learning to recover a

sharp image from degraded input, this head encourages the backbone to capture fine-grained structural details.

- Classification Head A classification head built on the same feature maps. It leverages representations potentially enriched by the reconstruction task, yielding more accurate object class predictions in an end-to-end pipeline.

Embedding the reconstruction head directly into YOLO is our first key contribution, enabling simultaneous image reconstruction and detection from shared features. Our second contribution is the integrated classification head, which benefits from those enhanced features.

Figure 5.1 depicts the architecture overview of YOLO CRNet.

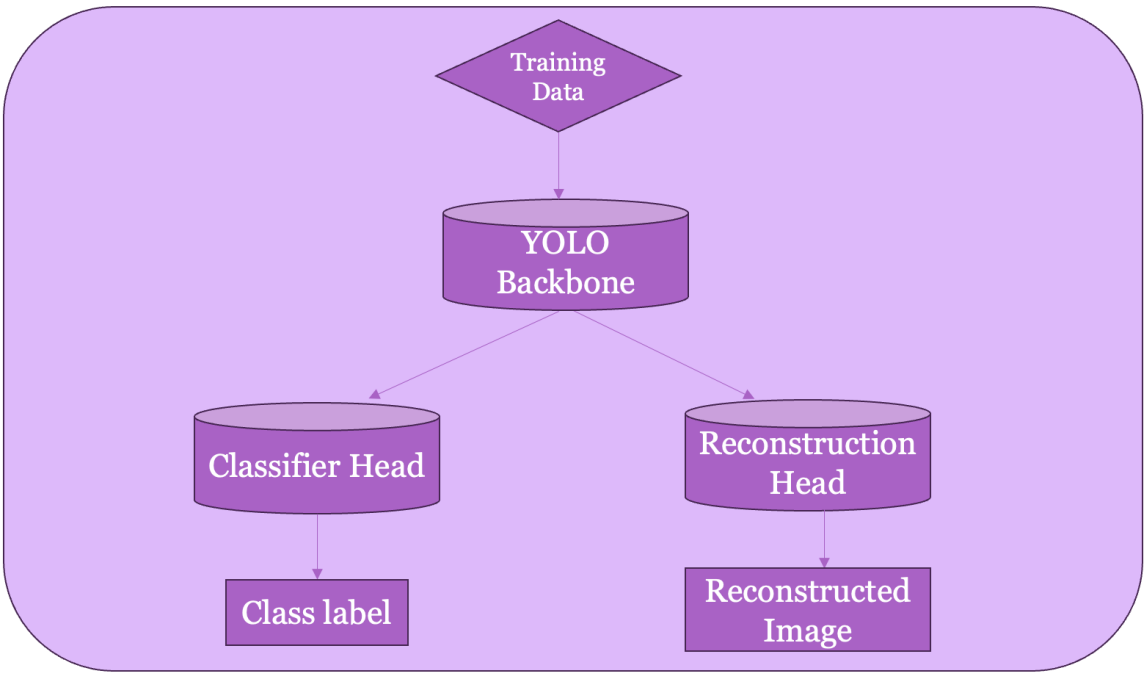


Figure 5.1: Architecture of the proposed YOLO CRNet model

5.1.2 Training Flow

A blurred image is fed into the reconstruction head. The head outputs a reconstructed image, which is compared against the ground-truth sharp image. This reconstruction loss

forces the network to capture fine details and low-level textures. The reconstructed image is re-encoded through the same YOLO backbone into a processed feature vector. The feature vectors are cropped using the ground truth bounding boxes. The classification head transforms these features into predicted labels. A standard cross-entropy loss is computed against the true label. Jointly training both tasks acts as a regularizer, encouraging the backbone to learn features that are simultaneously good for reconstructing fine detail and for discriminative detection.

Figure 5.2 depicts the training workflow of YOLO CRNet.

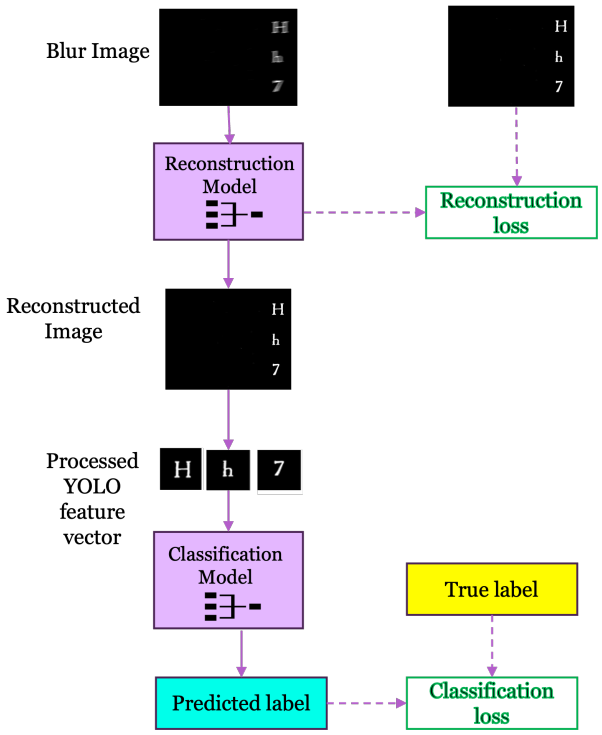


Figure 5.2: YOLO CRNet flow diagram

5.2 YOLO: Architecture and Feature Representation

The You Only Look Once (YOLO) framework represents a significant advancement in real-time object detection by reformulating the detection problem as a single regression task, rather than relying on traditional multi-stage pipelines that involve separate region

proposal generation and subsequent classification [57]. This unified approach processes the entire input image in one forward pass through a convolutional neural network (CNN), making YOLO both exceptionally fast and efficient. YOLOv8 stands out from its predecessors with its anchor-free design. This means that instead of relying on predefined anchor boxes, it directly predicts the likelihood of objects being present using a sigmoid function and then uses a threshold to identify the actual target objects. A key innovation in YOLOv8 is the use of the Task Aligned Assigner, which helps in efficiently assigning targets during the single-stage detection process. To further boost detection accuracy, particularly for the bounding box regression, YOLOv8 employs Distribution Focal Loss [40]. This specific loss function helps the network better handle situations where there's an imbalance between object categories or many background examples, allowing it to concentrate more effectively on learning the precise locations of objects. Alongside this, it uses the Complete Intersection over Union (CIoU) loss function [84] to accurately measure the overlap between the predicted bounding boxes and the actual ground-truth boxes. These advancements collectively contribute to YOLOv8's strong performance in object detection tasks. YOLOv8 consists of three main components: Backbone, Neck and Head [65, 30] as depicted in Figure 5.3.

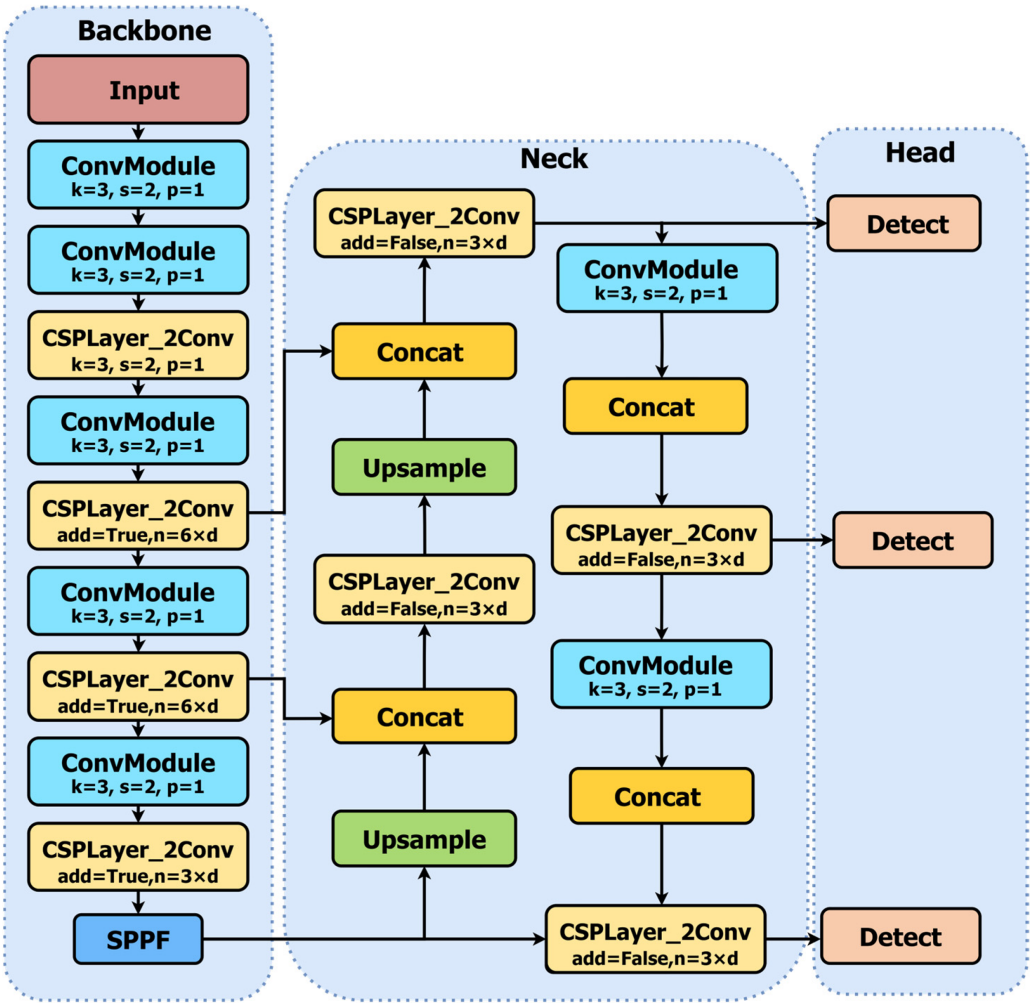


Figure 5.3: YOLOv8 architecture [65]

Backbone: Hierarchical Feature Extraction

The backbone is the foundational component of the YOLO architecture, responsible for extracting essential visual features from the input image at various spatial resolutions and semantic depths. It processes the input image to create hierarchical feature maps. The early layers in the backbone are designed to learn fine-grained spatial information, such as edges, corners, and textures, while, deeper backbone layers, with larger receptive fields, learn more abstract and strong semantic information. The backbone part consists of Conv,

C2f, and SPPF modules, designed to extend network depth and the receptive field, enabling the fusion of local and global features [65, 30].

- Convolutional Layers (Conv): Standard convolutional layers, often referred to as ConvBiSiLU (CBS), form the fundamental building blocks. These layers perform convolution operations on the input images or preceding feature maps, typically using 3x3 kernels for feature learning and 1x1 kernels for channel adjustments. Strided convolutions are used for downsampling, progressively reducing spatial dimensions while increasing the depth of feature representation. These Conv modules are instrumental in initial feature extraction and assist other modules like C2f in their operations. Similar to its predecessors like YOLOv5 [28], these layers are crucial for capturing low-level features such as edges and textures in the early stages, and more complex patterns in deeper stages.

- CSP (Cross Stage Partial) Bottleneck with 2 convolutions (C2f): A significant module in the YOLOv8 backbone is the C2f block. It is designed as a lightweight convolutional structure, offering an advancement over the C3 module found in YOLOv5. The C2f module aims to enrich the gradient flow within the model by establishing more cross-layer connections (shortcuts). This design enhances the ability of the network to express vital features through its dense and residual structures. Operationally, the C2f module often involves splitting feature maps and processing them through parallel branches, followed by concatenation and fusion. It strategically changes the number of channels using split and concatenate operations, guided by scaling coefficients, which effectively reduces computational complexity and model capacity.

- SPPF (Spatial Pyramid Pooling Fast): Located towards the end of the backbone, the SPPF module plays a critical role in increasing the network's sensitivity to objects of varying scales and capturing multi-level contextual information. Consistent with its implementation in architectures like YOLOv5 [28], the SPPF module applies multiple max-pooling operations with different kernel sizes in parallel to the same input feature map. The outputs of these pooling layers are then concatenated and processed, to produce an output feature map that adaptively incorporates information from different receptive field sizes. This enables the model to capture the feature information of different levels in the images to produce an adaptive-sized output.

Neck: Multi-Scale Feature Fusion

Following the backbone, the neck component of YOLOv8 is dedicated to multi-scale feature fusion. The Neck part includes C2f (without shortcut), concatenation and upsampling

modules, integrating feature maps at different scales through a PANet and C2f module, promoting vertical information fusion. It takes the hierarchical feature maps produced by the backbone and combines them to generate feature pyramids that are rich in both semantic information (from deeper layers) and spatial detail (from shallower layers). This fusion enables the detection of objects across various sizes and resolutions. The fusion creates refined feature pyramid maps that contain both spatial and semantic information. The neck produces feature maps at multiple resolution levels (P3, P4, P5 corresponding to strides 8, 16, 32 relative to the input image size) that are fed into the detection heads [65, 30].

YOLOv8 employs structures inspired by Feature Pyramid Networks (FPN) [41] and Path Aggregation Network (PANet) [42]. This FPN-PANet architecture facilitates a bi-directional flow of information. The FPN-PANet structure and C2f (without shortcut) modules fuse feature maps of different scales from the three stages of the backbone, aggregating shallow information to deep features.

Head: Prediction Generation

The head component is responsible for taking the fused multi-scale feature maps from the neck and generating the final predictions, which include bounding box coordinates, objectness scores, and class probabilities. The original YOLOv8 has three detecting heads with respective dimensions in height and width of 20*20, 40*40 and 80*80. YOLOv8 employs several key design choices in its head architecture to enhance detection performance [65, 30].

- Decoupled Head: The head structure is decoupled, meaning that separate branches or sub-networks are used for the classification task and the regression (localization) task. This design choice aims to alleviate the potential conflict between these two distinct objectives, allowing each branch to specialize and potentially leading to improved accuracy for both tasks.
- Anchor-Free Mechanism: YOLOv8 adopts an anchor-free detection mechanism. Unlike anchor-based detectors that rely on a predefined set of anchor boxes with varying scales and aspect ratios, anchor-free methods directly predict object properties (like center points or corner points, and dimensions) without these priors. This approach can simplify the detection pipeline, reduce the number of hyperparameters related to anchors, and improve the detection performance for objects with irregular heights and widths or extreme aspect ratios.

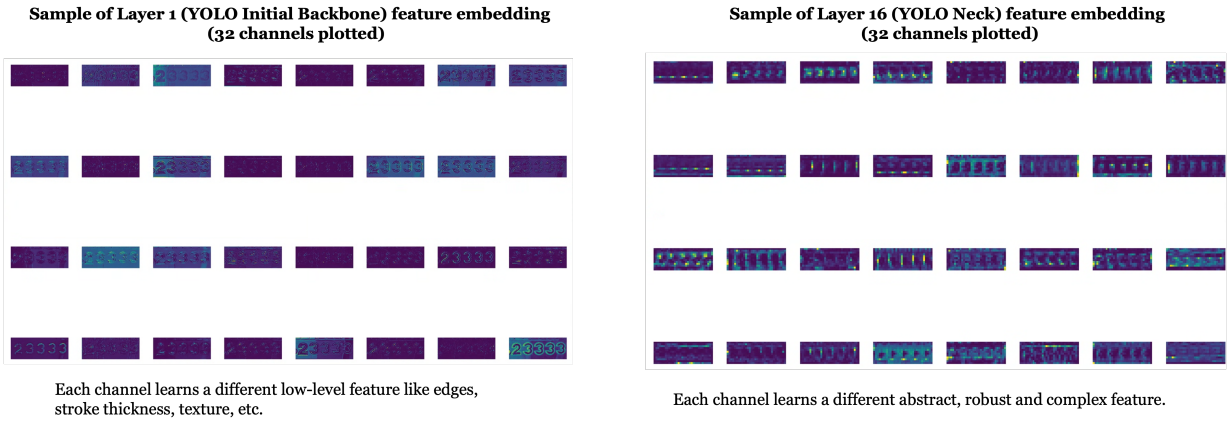


Figure 5.4: Figure shows samples of features from YOLO layers 1 (YOLO Early Backbone) and 16 (YOLO Neck) respectively, where 32 channels are plotted.

5.3 Classifier Head

In this work, we first extract the feature embedding from YOLO as explained in 4.2.2 followed by employing a light weight multi-layer perceptron (MLP) as the classifier head to map each pooled feature vector to one of the character classes. The overall data flow is illustrated in Figure 5.5, and the detailed MLP architecture is shown in Figure 5.6.

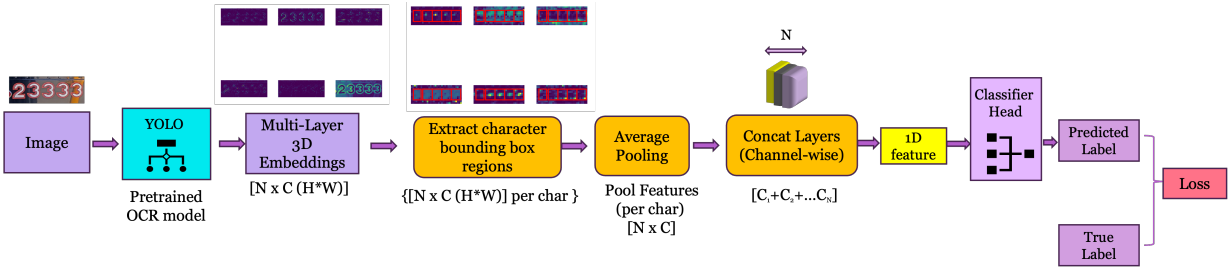


Figure 5.5: YOLO CRNet classification flow diagram.

5.3.1 Architecture

The classifier head takes as input the N pooled channel-wise feature vectors (each of dimension C) obtained by average-pooling the character bounding-box regions (explained

in 4.2.2). These 1D vectors are concatenated into a single feature tensor of shape (N, C) and then flattened into one long vector of length $N \times C$. This vector is fed into a two-stage MLP, each stage comprising:

The classifier head within the YOLO CRNet architecture is designed as a Multi-Layer Perceptron (MLP). Its primary function is to take the N pooled channel-wise feature vectors (each of dimension C) obtained by average-pooling the character bounding-box regions (explained in 4.2.2), and to output class logits suitable for character recognition. As depicted in Figure 5.6, the classifier head consists of repeating hidden blocks, culminating in a final output layer that predicts the character classes. Each hidden block is typically composed of a sequence of a fully-connected layer, batch normalization, a ReLU activation function, and a dropout layer.

Fully-Connected Layer A fully-connected layer, also termed a dense layer, is a foundational element in neural networks where each neuron in the input is connected to every neuron in the current layer. For an input vector x, the output y is computed via a linear transformation

$$y = Wx + b \tag{5.1}$$

where W is the weight matrix and b is the bias vector of the layer. These layers serve as powerful function approximators, capable of learning intricate, non-linear relationships within the data [20]. The dense layers transform the input feature vector into more abstract feature spaces, aiming to enhance the separability of different character classes.

Batch Normalization (BN) Following each dense layer, Batch Normalization (BN) is applied. Introduced by Ioffe and Szegedy [25], BN normalizes the output of the previous layer by re-centering and re-scaling. Specifically, it standardizes the activations to have zero mean and unit variance, and then applies learnable scale (γ) and shift (β) parameters. This technique addresses the issue of internal covariate shift, where the distribution of layer inputs changes during training due to updates in preceding layers. The benefits of BN include acceleration of the training process, allows for the use of higher learning rates, reduces the sensitivity of the network to parameter initialization, and can act as a regularizer.

ReLU (Rectified Linear Unit) Activation The Rectified Linear Unit (ReLU) is the non-linear activation function employed after Batch Normalization. It is defined as

$$f(x) = \max(0, x) \tag{5.2}$$

outputting the input directly if positive, and zero otherwise. The inclusion of non-linear activation functions like ReLU enables the network to learn complex, non-linear mappings from input features to output classes, which linear models alone cannot capture. ReLU has gained widespread adoption due to its computational simplicity and its efficacy in mitigating the vanishing gradient problem, by maintaining a non-saturating gradient for positive inputs [51].

Dropout Dropout is a regularization technique implemented to prevent overfitting in the network [71]. During the training phase, a random subset of neurons in the dropout layer is temporarily ignored with a specified probability. This prevents neurons from co-adapting too much and forces the network to learn more redundant features that are not overly dependent on the presence of any single neuron. This enhances the generalization capability of the character classifier head.

Output Layer The output from the final dropout layer in the last hidden block is then fed into a concluding dense layer. This terminal dense layer has an output dimension equal to the total number of distinct character classes that the model is tasked with recognizing classes. This layer produces the raw, unnormalized scores, commonly referred to as logits, for each character class. During training, these logits are passed through a softmax function to derive a probability distribution over the character classes. The class corresponding to the highest probability is then selected as the model's prediction for a given character position in the sequence.

In summary, the MLP-based classifier head processes the input feature vector through a series of transformations involving linear mappings, normalization, non-linear activations, and regularization. This structured approach enables the learning of discriminative features, culminating in the generation of class logits for the final character recognition task.

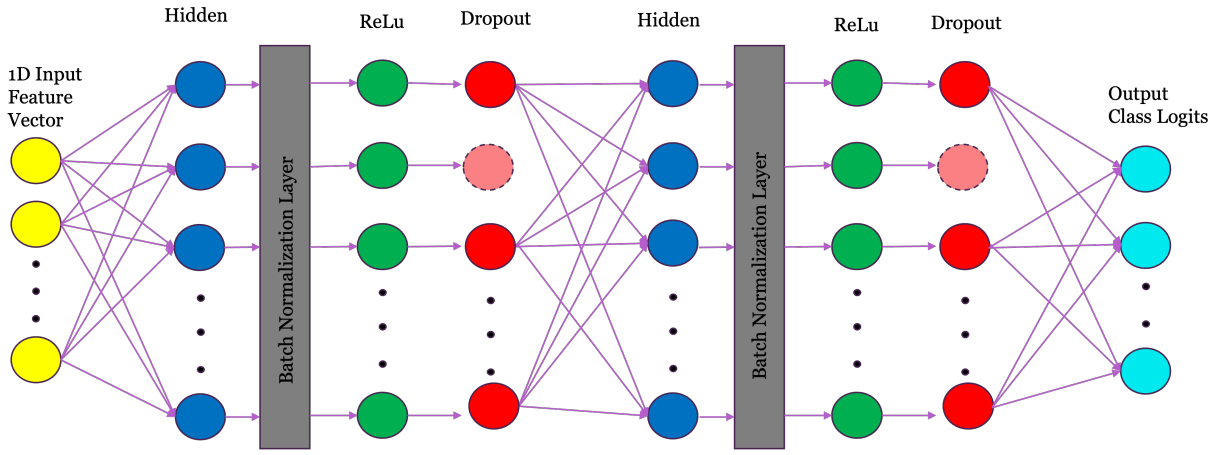


Figure 5.6: YOLO CRNet Classifier Head Architecture.

5.3.2 Design Rationale

The following key design considerations guided the choice of this architecture:

- **Dimensionality Reduction and Efficiency.** Applying Average Pooling reduces high dimensionality to a single 1D vector. Training a relatively simple MLP on this lower dimensional vector is computationally cheaper, faster, and often requires less data to train effectively.
- **Reduced Risk of Overfitting on Spatial Noise.** Discarding the fine spatial layout prevents the classifier head from overfitting on potentially noisy or irrelevant spatial details specific to the training examples. The MLP focuses on the more global feature presence provided.
- **Focus on Channel Information.** We hypothesize that which channels are activated is more discriminative for character identity than the precise arrangement of activations within the bounding box. The model learns channel information (which features are present) rather than their precise spatial arrangement.
- **Fixed-Size Input Vector.** Character bounding boxes extracted from feature map might vary slightly in their spatial size. Average pooling ensures that variations in bounding-box aspect ratios do not alter the classifier input dimension, simplifying implementation.

- **Invariance and Robustness.** Pooling provides a degree of invariance to small translations or distortions of the features.

5.3.3 Loss Function for Training

We train the classifier head using the standard categorical cross-entropy loss:

$$\mathcal{L}_{\text{CE}} = -\sum_{i=1}^{K} y_i \log(p_i) \tag{5.3}$$

where K is the number of character classes, y_i is the one-hot ground-truth indicator for class i, and p_i is the network's predicted probability for class i after softmax. This objective encourages the model to assign high probability to the correct class.

5.3.4 Regularization and Overfitting Prevention

To improve generalization, we incorporate multiple regularization strategies:

- **Dropout** after each ReLU layer to prevent co-adaptation of neurons [71].
- **Batch normalization** before each ReLU, which stabilizes training and acts as a mild regularizer [25].
- **Early stopping** Employed early stopping monitoring validation loss with a patience threshold of 10 epochs as a regularization technique to determine the optimal training duration and avoid overfitting.
- **Adaptive learning-rate scheduler** Employed an adaptive learning rate scheduler that reduces the learning rate when validation loss plateaus to fine-tune convergence and improve final model performance.

5.4 Reconstruction Head

5.4.1 Workflow

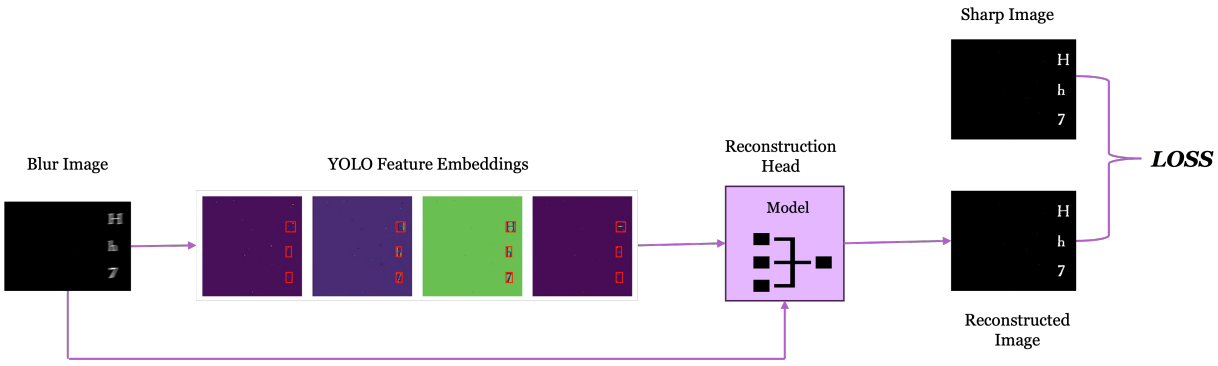


Figure 5.7: YOLO CRNet Reconstruction Head Workflow

The reconstruction head within the YOLO CRNet architecture is specifically designed to mitigate image degradations present in the input. The workflow of this reconstruction head, as illustrated in Figure 5.7, involves leveraging features extracted by the main YOLO and then processing them through a dedicated reconstruction model.

Input to the Reconstruction Head

The reconstruction process begins with two primary inputs:

1. **The Blur Image (X_{blur}):** This is the original, degraded input image that the overall YOLO CRNet system receives. The raw pixel information from the input image is used by the reconstruction network through a skip connection.
2. **YOLO Feature Embeddings:** Concurrently, the blur image is processed through the initial stages of the YOLO generating a set of multi-scale feature embeddings. These embeddings, having been processed by convolutional layers, already encapsulate learned hierarchical features from the input image, ranging from low-level textures and edges to more abstract representations. It allows the reconstruction task to benefit from the rich, context-aware features already learned for detection, potentially leading to more efficient and effective restoration [63].

The Reconstruction Head Model

The core of the restoration process is the Reconstruction Head Model. The model uses an encoder-decoder architecture. A common and highly effective instance of an encoder-decoder architecture for image reconstruction and segmentation is the U-Net [62]. We experiment over several modifications in the U-Net model to achieve better reconstruction quality.

- Encoder Path: The encoder pathway within the reconstruction network systematically processes feature maps sourced from various hierarchical levels of the YOLO pipeline. It initiates with higher-resolution features from YOLO Layer 1, capturing finer details and progressively integrates features from deeper, consequently lower-resolution, YOLO layers progressing towards YOLO Layer 5. This structured, down-sampling like integration allows the encoder to build an increasingly abstract and contextually rich representation of the input, effectively capturing broad contextual information essential for the reconstruction task.
- Decoder Path: The decoder part then progressively upsamples these features, aiming to reconstruct the high-resolution sharp image. Crucially, architecture employ skip connections that concatenate feature maps from the encoder path with corresponding feature maps in the decoder path. These skip connections allow the decoder to directly access fine-grained, low-level features from earlier stages, which is vital for preserving spatial details and producing sharp reconstructions [62, 26].

Output: The Reconstructed Image

The output of the Reconstruction Head is the Reconstructed Image, denoted as $\hat{X}$. This image is the model's attempt to restore the original sharp image X from the blurred input X_{blur} and the features provided. The quality of $\hat{X}$ is critical, as it is hypothesized that a cleaner, sharper image will lead to improved performance in the downstream character recognition task.

Loss Calculation for Reconstruction

To train the reconstruction head, its output $\hat{X}$ is compared against the ground-truth Sharp Image X. The discrepancy between these two images is quantified by a loss function, specifically the reconstruction loss ($\mathcal{L}_{\text{recon}}$ as defined in the problem formulation section,

Equation 3.4). The calculated reconstruction loss is then used, along with the detection and classification losses, to update the parameters of the entire YOLO CRNet model via backpropagation during the end-to-end training process. This joint training allows the feature extraction part of YOLO to also learn features that are beneficial for the reconstruction task.

In summary, the reconstruction head workflow leverages features from the main YOLO pipeline and processes them through an encoder-decoder style network to generate a deblurred image. This reconstructed image is then compared to the ground-truth sharp image to compute a loss, which guides the training of the reconstruction components and, through shared layers, potentially influences the primary detection and recognition tasks.

5.4.2 Architecture

This section details three distinct configurations (A, B, and C) explored for this reconstruction head, each employing an encoder-decoder architecture with variations in upsampling mechanisms and feature refinement strategies. All configurations leverage a U-Net-like structure [62] with skip connections to fuse low-level spatial details from the encoder path with high-level semantic information from the decoder path. All the three configurations share a similar encoder pathway as explained in section 5.4.1. The YOLO layer features are of varying channel depths and resolutions $96 \times 128 \times 128$, $192 \times 64 \times 64$, and $384 \times 32 \times 32$. The deepest part of the encoder, processing YOLO Layer 5 features, forms the bottleneck, representing the most abstract, high-level feature representation before upsampling begins in the decoder.

Common Encoder Structure

All three reconstruction head configurations share a similar encoder pathway. This path progressively processes and downsamples feature maps. It starts by taking features from different stages of the YOLO pipeline (YOLO Layer 1 through YOLO Layer 5, with varying channel depths and resolutions $96 \times 128 \times 128$, $192 \times 64 \times 64$, and $384 \times 32 \times 32$). At each stage of the encoder, features from a specific YOLO layer are typically processed by a DoubleConv block (which consists of two convolutional block sequences which includes a convolutional layer with kernel size 3, a batch normalization layer and RELU activation). These processed features are then concatenated with features from the subsequent YOLO layer at the corresponding resolution before further processing and downsampling in the next encoder stage. This hierarchical feature extraction allows the encoder to capture

multi-scale contextual information from the input image via the YOLO embeddings. The deepest part of the encoder, processing YOLO Layer 5 features, forms the bottleneck, representing the most abstract, high-level feature representation before upsampling begins in the decoder.

Reconstruction Head Configuration A

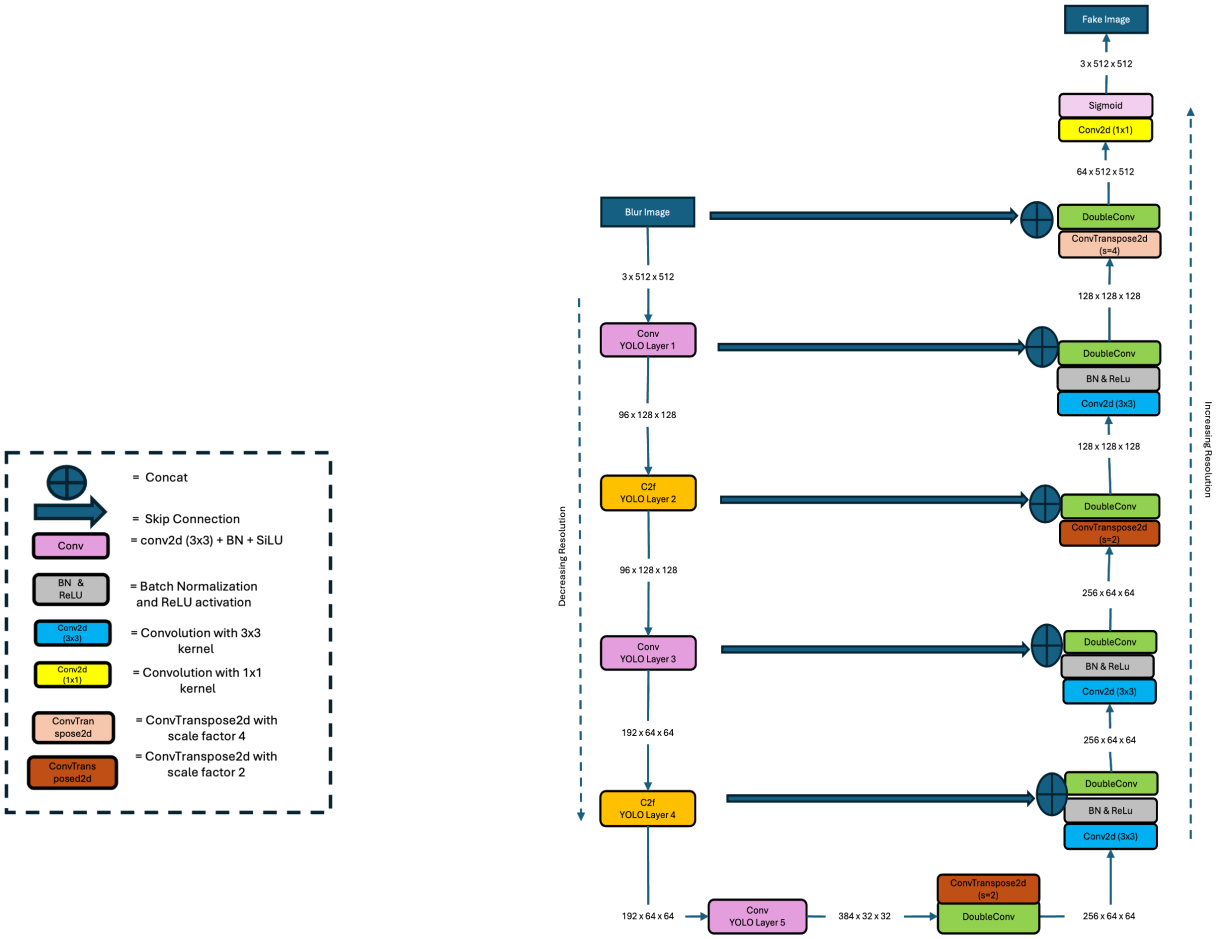


Figure 5.8: Reconstruction Head Configuration A

Configuration A (Figure 5.8) employs Transposed Convolution for its upsampling operations in the decoder pathway.

- Upsampling Mechanism: The decoder path uses Transposed convolution layers ,with a scale factor of 2 or 4 appropriately according to the output resolution required, to progressively increase the spatial resolution of the feature maps, starting from the bottleneck. Transposed convolution (sometimes referred to as deconvolution) is a learnable upsampling method that maps a single point in the input feature map to a larger area in the output feature map, effectively increasing its dimensions (as illustrated in Figure 5.9).

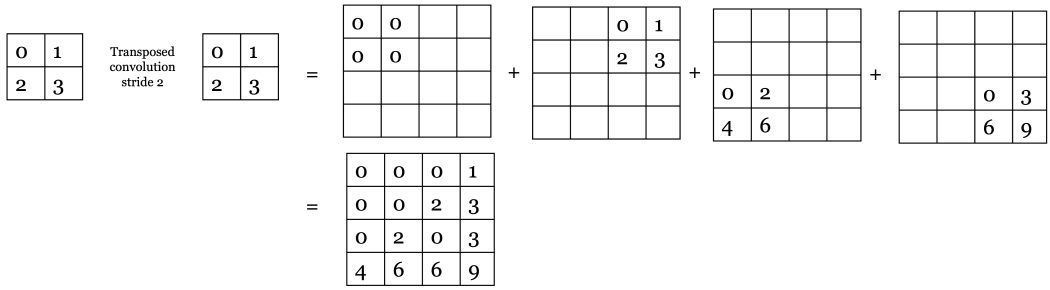


Figure 5.9: Illustration of Transposed Convolution

- Decoder Path: At each decoder stage, the feature map from the previous stage is upsampled using Transposed Convolution [62]. This upsampled map is then concatenated with the corresponding feature map from the encoder path via a skip connection. Following concatenation, a Double Convolution block (DoubleConv) which consists of two convolutional block sequences which includes a convolutional layer with kernel size 3, a batch normalization layer and RELU [51] activation, refines the merged features. This sequence of upsampling, concatenation, and refinement is repeated until the feature maps reach a resolution suitable for the final image reconstruction. The Blur Image is concattenated through a skip connection to this final output generation stage with the decoder output before the final convolution.

- Output Generation: The final feature map from the decoder ($64 \times 512 \times 512$ after the last Upsampling) is passed through a 1×1 Convolutional layer with layer reduce channels to 3 for an RGB image and a Sigmoid activation function to produce the reconstructed image $\hat{X}$ of size $3 \times 512 \times 512$.

Reconstruction Head Configuration B

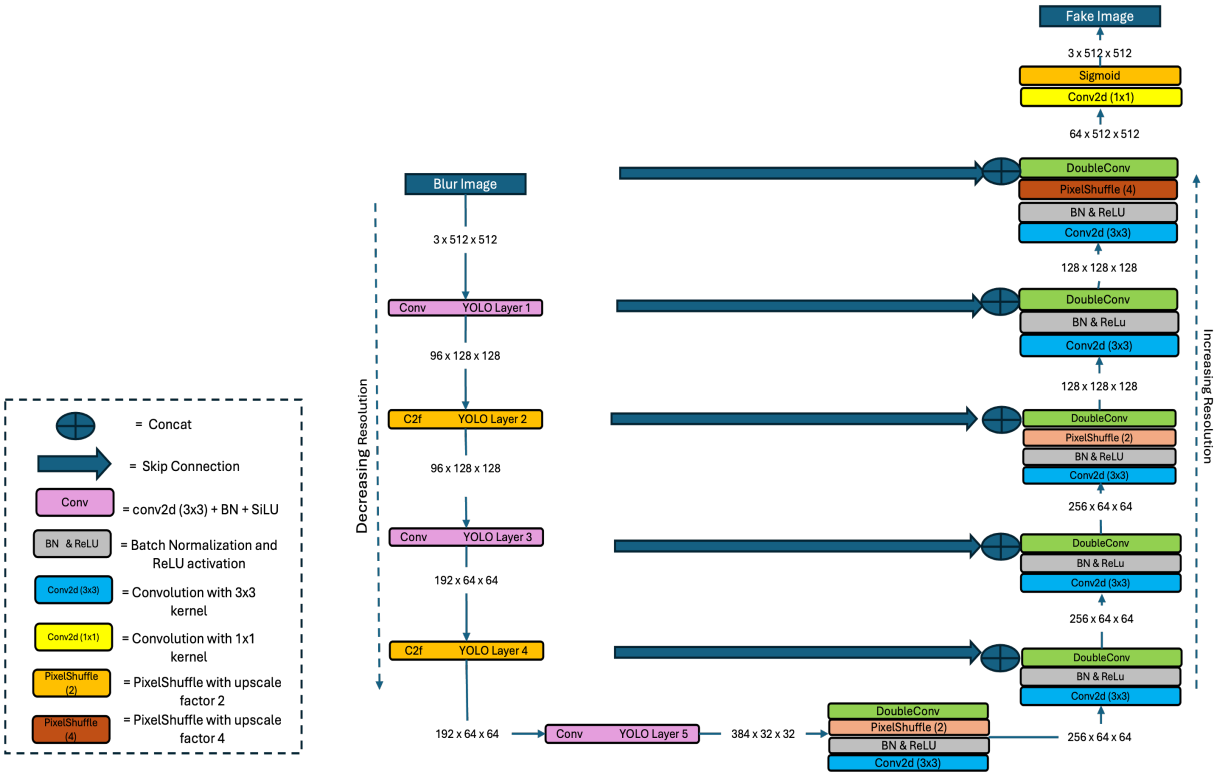


Figure 5.10: Reconstruction Head Configuration B

Configuration B (Figure 5.10) distinguishes itself by utilizing PixelShuffle layers for upsampling in its decoder.

- Upsampling Mechanism: Instead of transposed convolutions, this configuration uses PixelShuffle layers with an upscale factor of 2 or 4 appropriately according to the output resolution required. PixelShuffle is a learned upsampling technique that first uses a convolutional layer to output $C \cdot r^2$ channels (where C is the desired output channel count and r is the upscaling factor). It then rearranges these $C \cdot r^2$ channels into an output feature map of shape $(C, H \cdot r, W \cdot r)$ by unfolding $r \times r$ blocks of channels into $r \times r$ spatial blocks for each location (Figure 5.11) [68]. This method is often preferred as it can avoid checkerboard artifacts sometimes associated with

transposed convolutions and can produce smoother, more visually appealing results [1, 52].

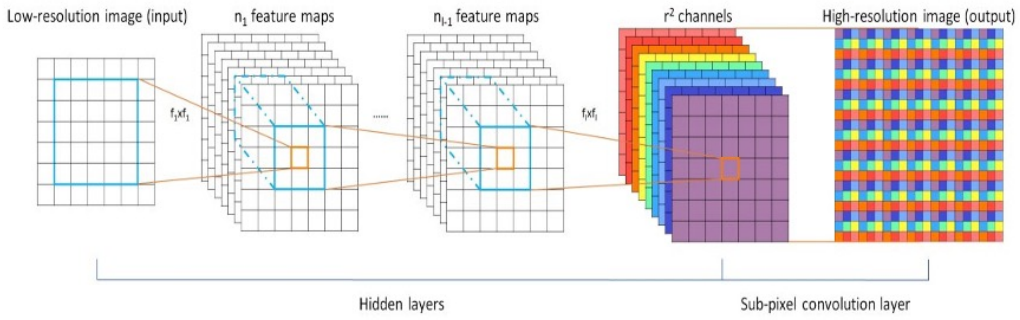


Figure 5.11: Illustration of PixelShuffle [68]

- Decoder Path: Similar to Configuration A, at each decoder stage, the feature map from the previous stage is upsampled using PixelShuffle layers. the rest of the decoder path is similar to as explained in 5.4.2.
- Output Generation: The process mirrors Configuration A for the final output.

Reconstruction Head Configuration C

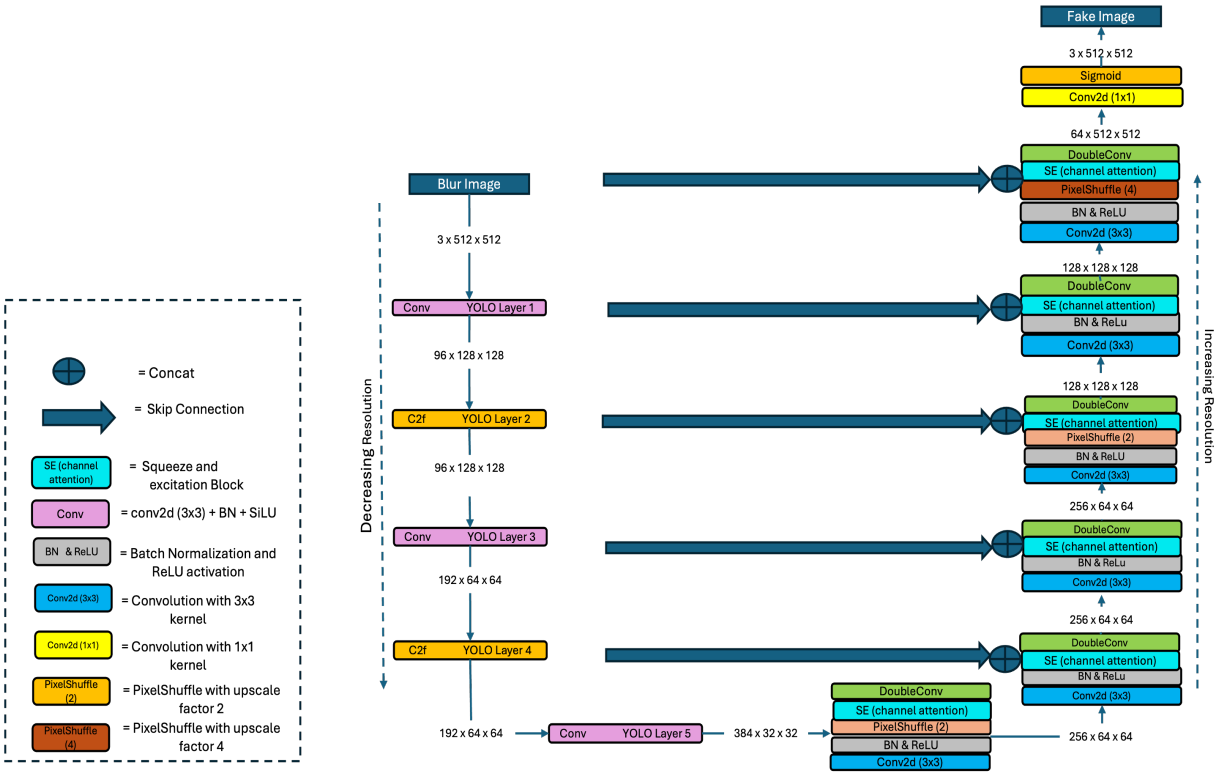


Figure 5.12: Reconstruction Head Configuration C

Configuration C (Figure 5.12) builds upon Configuration B by incorporating Squeeze-and-Excitation (SE) Blocks for channel attention within the decoder's refinement stages, while still using PixelShuffle for upsampling.

- Upsampling Mechanism: This configuration also uses 'PixelShuffle' layers with an upscale factor of 2 or 4 for upsampling, identical to Configuration B.
- Decoder Path with SE Blocks: The key difference lies in the refinement blocks. After upsampling with PixelShuffle and concatenation with encoder features via skip connections, the DoubleConv blocks in Configuration C are augmented with an Squeeze and Excitation(SE) block. (Figure 5.13).

An SE Block is a form of channel attention that allows a network to learn which feature channels are most important for a given input and then enhance those informative channels while suppressing less useful ones[22]. As illustrated in Figure 5.13, it first *squeezes* global spatial information into a channel descriptor using global average pooling. Then, an *excitation* mechanism using two small fully-connected layers and activations learns adaptive channel importance weights. These learned weights are then multiplied channel-wise with the original input feature map to enhance informative channels and suppress less useful ones.

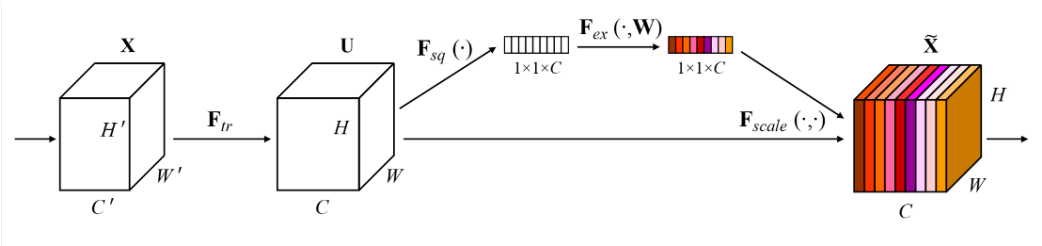


Figure 5.13: SE Block Illustration [22]

By integrating SE blocks into the DoubleConv refinement stages of the decoder, Configuration C aims to improve the quality of the learned features by enabling the network to focus on more relevant channels during the reconstruction process.

- Output Generation: The final output generation remains consistent with Configurations A and B.

In summary, all three configurations leverage the foundational U-Net encoder-decoder structure with skip connections and progressively integrated YOLO layer features. They differ primarily in their choice of upsampling methodology in the decoder (Transposed Convolution vs. PixelShuffle) and the inclusion of advanced feature refinement mechanisms like SE Blocks (in Configuration C), each aiming to produce a high-quality reconstructed image $\hat{X}$ from the blurred input X_{blur}.

5.4.3 Design Rationale

Several key design principles and components were considered and incorporated into the various configurations of the reconstruction head, as previously detailed in Section 5.4.2, to ensure effective image restoration. These choices are summarized as follows:

- **U-Net Structure:** The foundational architecture for the reconstruction head is the U-Net [62]. The U-Net is an effective architecture for image reconstruction and segmentation tasks due to its encoder-decoder structure. The encoder path performs contextual feature extraction through progressive downsampling, while the decoder path gradually upsamples these features to reconstruct the image. A key strength of the U-Net is its use of skip connections, which concatenate feature maps from the encoder directly with corresponding layers in the decoder. These connections allow the network to preserve fine-grained, high-resolution details from earlier stages and combine them with rich semantic information from deeper layers, making it highly effective at generating high-resolution images that are both contextually coherent and spatially accurate.

- **Multi-Scale YOLO Features as Input:** The reconstruction head leverages features extracted from multiple stages of the YOLO object detection pipeline (backbone and neck), along with the original blurred input image, integrated via skip connections and as initial inputs. By taking features from different levels of the YOLO network, the decoder gains access to a hierarchy of information, from low-level spatial details to higher-level semantic content. This multi-scale feature input, a common technique in U-Net like architectures when integrated with a primary feature extractor, aids in preserving high-resolution details and facilitates better gradient flow during training [63].

- **PixelShuffle for Upsampling:** PixelShuffle layers [68] are employed as the upsampling mechanism, replacing traditional transposed convolutions. This method is often preferred because it can effectively reduce or avoid checkerboard artifacts commonly associated with transposed convolutions, leading to smoother and more visually pleasing reconstructed images [1, 52].

- **Channel Attention (Squeeze-and-Excitation Blocks):** To further refine the feature representations within the decoder, Squeeze-and-Excitation (SE) blocks [22] are incorporated. These blocks are typically integrated at multiple stages of the decoder, often after the upsampling step and feature concatenation. This block effectively amplifies informative channels and suppressing less useful ones. This adaptive recalibration of channel features allows the network to focus on the most relevant information for reconstruction.

- **Final Output Layer:** The culmination of the decoder pathway is a final output layer. This typically consists of a 1x1 Convolutional layer used to map the feature channels from the last decoder stage to the desired number of output channels (3

channels for an RGB image). This is followed by a Sigmoid activation function, which scales the output pixel values to the range [0, 1], appropriate for representing a normalized image.

5.4.4 Loss Functions for Training

The training of the reconstruction head is guided by a composite loss function designed to ensure that the reconstructed image $\hat{X}$ is both quantitatively and perceptually similar to the ground-truth sharp image X. As outlined in Section 3.2.2, this involves a weighted combination of pixel-wise losses (L_1 and L_2) and a perceptual loss. The weighting strategy prioritizes L_1 loss and Perceptual loss over L_2 loss to encourage sharpness and visual realism while maintaining training stability.

- **L_1 Loss (Mean Absolute Error):** This loss measures the average absolute difference between the pixel values of the ground-truth sharp image X and the predicted reconstructed image $\hat{X}$. It is defined as:

$$\mathcal{L}_{L1} = \frac{1}{N_p} \sum_{i=1}^{N_p} |X_i - \hat{X}_i|$$

  where X_i is a pixel value in the ground-truth image, $\hat{X}_i$ is the corresponding pixel value in the predicted image, and N_p is the total number of pixels. L_1 loss is known for promoting less blurry results and being robust to outliers.

- **L_2 Loss (Mean Squared Error):** This loss measures the average squared difference between the pixel values of X and $\hat{X}$:

$$\mathcal{L}_{L2} = \frac{1}{N_p} \sum_{i=1}^{N_p} (X_i - \hat{X}_i)^2$$

  While L_2 loss penalizes larger errors more significantly, it can sometimes lead to overly smooth or blurry reconstructions if heavily weighted. Its inclusion, typically with a lower weight, can aid in stabilizing the optimization process.

- **Perceptual Loss:** To address the limitations of pixel-wise losses in capturing human perception of image quality, a perceptual loss is incorporated [29]. As illustrated in 5.14, a pre-trained VGG19 network [70], known for its strong image feature extraction

capabilities, is utilized. Both the reconstructed image $\hat{X}$ and the ground-truth sharp image X are passed through this fixed VGG19 network. Feature maps (ϕ) are then extracted from two intermediate layers . The perceptual loss is computed as the L_2 distance between these corresponding feature maps:

$$\mathcal{L}_{\text{perc}}(\hat{X}, X) = \sum_j \frac{1}{C_j H_j W_j} \|\phi_j(\hat{X}) - \phi_j(X)\|_2^2$$

This feature-space distance encourages the reconstructed image to have similar semantic and textural characteristics to the ground-truth image, often resulting in more visually convincing restorations.

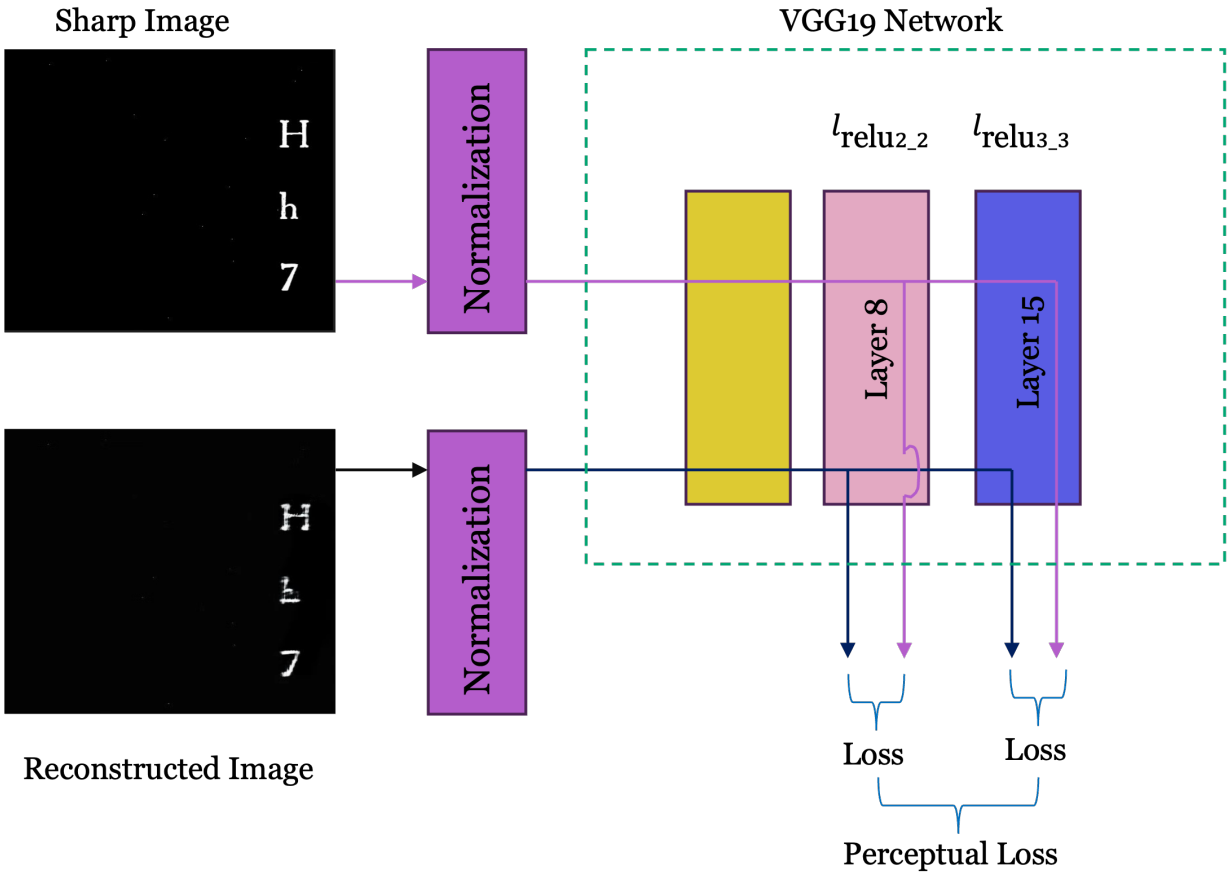


Figure 5.14: Perceptual Loss

5.4.5 Measures to Prevent Overfitting

Training deep neural networks, especially for complex tasks like image reconstruction, requires careful strategies to prevent overfitting, where the model performs well on the training data but poorly on unseen validation or test data. Several measures were employed to ensure robust generalization:

- **Appropriate Weighting of Loss Components:** The individual components within reconstruction are weighted w_1, w_2, w_3 as mentioned in 3.4. Careful tuning of these weights is crucial to ensure balanced training, preventing one loss component from dominating the learning process.
- **Monitoring Validation Loss:** Throughout the training process, the model's performance is continuously monitored on a separate validation dataset, which is not used for updating the model weights. The validation loss serves as a crucial indicator of the model's ability to generalize to unseen data. An increasing validation loss while training loss continues to decrease is a hallmark of overfitting.
- **Early Stopping:** To prevent overfitting and determine an optimal training duration, an early stopping mechanism is employed. This technique involves monitoring the validation loss and terminating the training process if the validation performance does not improve for a predefined number of consecutive epochs, known as the patience threshold (which is 10 epochs). Early stopping acts as a powerful regularization technique by selecting the model iteration that generalizes best to the validation set.
- **Adaptive Learning Rate Scheduler:** An adaptive learning rate scheduler is utilized to fine-tune the convergence of the training process and enhance final model performance. We reduce the learning rate when the validation loss plateaus or stops improving for a certain number of epochs (which is 5 epochs). Lowering the learning rate allows the model to make smaller adjustments to the weights, helping it settle into a better minimum in the loss landscape and avoid oscillating around an optimum.

These measures contribute to training a reconstruction model that is not only accurate on the training data but also generalizes effectively to new, unseen degraded images.

Chapter 6

Experiments and Results

This chapter presents the experimental evaluation conducted to validate the proposed YOLO CRNet methodology, as detailed in Chapter 5. It provides a comprehensive analysis of the results obtained from these experiments. The chapter is organized as follows: Section 6.1 outlines the dataset composition, data preprocessing steps and the training hyperparameters; Section 6.2 defines and formulates the metrics used to assess classification, reconstruction and overall pipeline performance; Section 6.3 investigates different YOLO layer feature-extraction configurations, compares their classification performance, and provides per-label analyses and a confusion matrix; Section 6.4 compares multiple reconstruction head architectures using PSNR and SSIM, presents qualitative examples, and reports ablation studies on loss components; Section 6.5 performs a comparative study of baseline YOLO, the CRNet classifier on blurred inputs, and the full reconstruction+classification pipeline on degraded images. Finally, the results are presented, analyzed, and discussed in detail.

6.1 Experimental Setup

In total, the dataset comprises 12,428 sharp and blur pairs. Of these, 10,428 pairs are used for model development and are dynamically split during training into 80% for training and 20% for validation. A held-out test set of 2000 image pairs is reserved for the final evaluation of model performance. Table 6.1 outlines the number of samples in each category.

Category	Number of Image Pairs (LR/HR)	Number of Characters
Training	8342	47,458
Validation	2086	11,865
Test	2000	10,100
Total	12,428	69,423

Table 6.1: Dataset Split.

All models were trained on a single NVIDIA GeForce RTX 3060 Ti GPU. A uniform set of hyperparameters was used in the training phase of all the models. The Adam optimizer was employed with an initial learning rate of 0.001 and was complemented by a *ReduceLROnPlateau* learning rate scheduler with a factor of 0.1 to adjust the learning rate for faster convergence dynamically. Early stopping monitoring validation loss with a patience threshold of 10 epochs was employed as a regularization technique to determine the optimal training duration and avoid overfitting. The models are developed using PyTorch [54].

Hyperparameter	Value
Optimizer	Adam
Initial Learning Rate (LR)	0.001
ReduceLROnPlateau Patience	5
Batch Size	16
Epoch Size	100
Early Stopping Patience	10

Table 6.2: Hyperparameters used in the training phase.

6.2 Evaluation Metrics

To quantitatively assess the performance of the proposed YOLO CRNet model and its components, a set of standard evaluation metrics are employed. These metrics cover the two primary tasks: character classification and image reconstruction. In the proposed approach, the classification performance of the trained models is evaluated four six metrics: Accuracy, Precision, Recall (Sensitivity), and F1-score, while the reconstructed image quality is evaluated used Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM). The aforementioned metrics are explained below with their mathematical

formulations. The notations used in the mathematical formulations are defined in Table 6.3.

Notation	Description
TP	True Positives: Correctly predicted positive instances
FP	False Positives: Incorrectly predicted positive instances
TN	True Negatives: Correctly predicted negative instances
FN	False Negatives: Incorrectly predicted negative instances

Table 6.3: Notations for Performance Metrics

- **Accuracy:** Accuracy is a measure of the overall correctness of the model. It is defined as the ratio of all correct predictions (both true positives and true negatives) to the total number of instances evaluated. While straightforward to compute, accuracy can be misleading, especially in scenarios with imbalanced class distributions. In such cases, a model predicting the majority class most of the time can achieve high accuracy without having good performance on the minority class of interest.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{6.1}$$

- **Precision:** Precision measures the accuracy of positive predictions. It is the ratio of correctly predicted positive instances (TP) to the total number of instances predicted as positive (TP + FP). A high precision indicates a low false positive rate, ie, the ability of a model to reduce false positives.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{6.2}$$

- **Recall (Sensitivity or True Positive Rate):** Recall measures the ability of the model to identify all actual positive instances. It is the ratio of correctly predicted positive instances (TP) to the total number of actual positive instances (TP + FN). A high recall indicates a low false negative rate. Recall places emphasis on minimizing false negatives, thereby ensuring that all relevant positive instances are found.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6.3}$$

- **F1-Score:** The F1-Score is the harmonic mean of Precision and Recall, providing a single score that balances both. It is particularly useful for datasets with imbalanced classes, where accuracy might be an unreliable measure.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6.4}$$

- **Peak Signal-to-Noise Ratio (PSNR):** PSNR measures the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. Higher PSNR generally indicates better reconstruction quality. It is defined as:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{I_{\text{MAX}}^2}{\text{MSE}} \right) \tag{6.5}$$

  where I_{MAX} is the maximum possible pixel value of the image, and MSE is the Mean Squared Error between the ground-truth and reconstructed images:

$$\text{MSE} = \frac{1}{H \cdot W \cdot C} \sum_{c=1}^{C} \sum_{h=1}^{H} \sum_{w=1}^{W} (X_{c,h,w} - \hat{X}_{c,h,w})^2 \tag{6.6}$$

- **Structural Similarity Index (SSIM):** SSIM measures the perceived similarity between two images based on three components: luminance, contrast, and structural information [80]. It is designed to align better with human visual perception of image quality than PSNR. SSIM values range from -1 to 1, where 1 indicates perfect similarity.

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{6.7}$$

  where μ_x, μ_y are the means of x and y; σ_x^2, σ_y^2 are the variances of x and y; σ_{xy} is the covariance of x and y; and c_1, c_2 are stabilizing constants.

6.3 YOLO CRNet Classification Head Evaluation

This section details the experiments conducted to evaluate the performance of the YOLO CRNet's classification head. The primary goal was to determine the optimal set of YOLO layers from which to extract features for the character recognition task and to comprehensively assess the classifier's performance using these optimal features.

6.3.1 Experimental Setup

The evaluation of the classification head was performed independently of the end-to-end YOLO CRNet system to specifically isolate and analyze its character recognition capabilities based on different feature inputs.

Dataset The training dataset for the classifier model comprised features extracted from various YOLO layers corresponding to 59,323 characters sourced from 10,428 sharp images. For testing and performance evaluation, a distinct test dataset was utilized, consisting of 10,100 characters from 2,000 sharp test images (as detailed in Table 6.1).

Feature Extraction Configurations To identify the most effective feature representations for character classification, features were extracted from different combinations of YOLO layers. Five distinct configurations, labeled A through E, were investigated, as summarized in Table 4.1 . These configurations explore features from different depths and components of the YOLO architecture as explained in section 4.2.3. The classifier model was trained separately using features from each of these configurations and subsequently evaluated on the common test set.

6.3.2 Performance Comparison of Layer Configurations

The performance of the classifier head when trained with features from each of the five configurations (A-E) was evaluated using standard classification metrics: accuracy, precision, recall, and F1-score. The results are summarized in Table 6.4 and visualized in Figure 6.1.

Table 6.4: YOLO CRNet Classifier Head Performance by Layer Configuration

Configuration ID	YOLO Layers	Accuracy	Precision	Recall	F1-Score
A	[1, 2, 4, 10, 13, 16]	**0.9522**	**0.9528**	**0.9467**	**0.9497**
B	[1, 2, 3, 4, 5]	0.9379	0.9277	0.9343	0.9310
C	[8, 9, 10, 13, 16]	0.8757	0.8820	0.8626	0.8722
D	[9, 10, 16, 19, 20]	0.7814	0.8205	0.7415	0.7790
E	[16, 17, 19, 20]	0.8350	0.8432	0.8192	0.8310

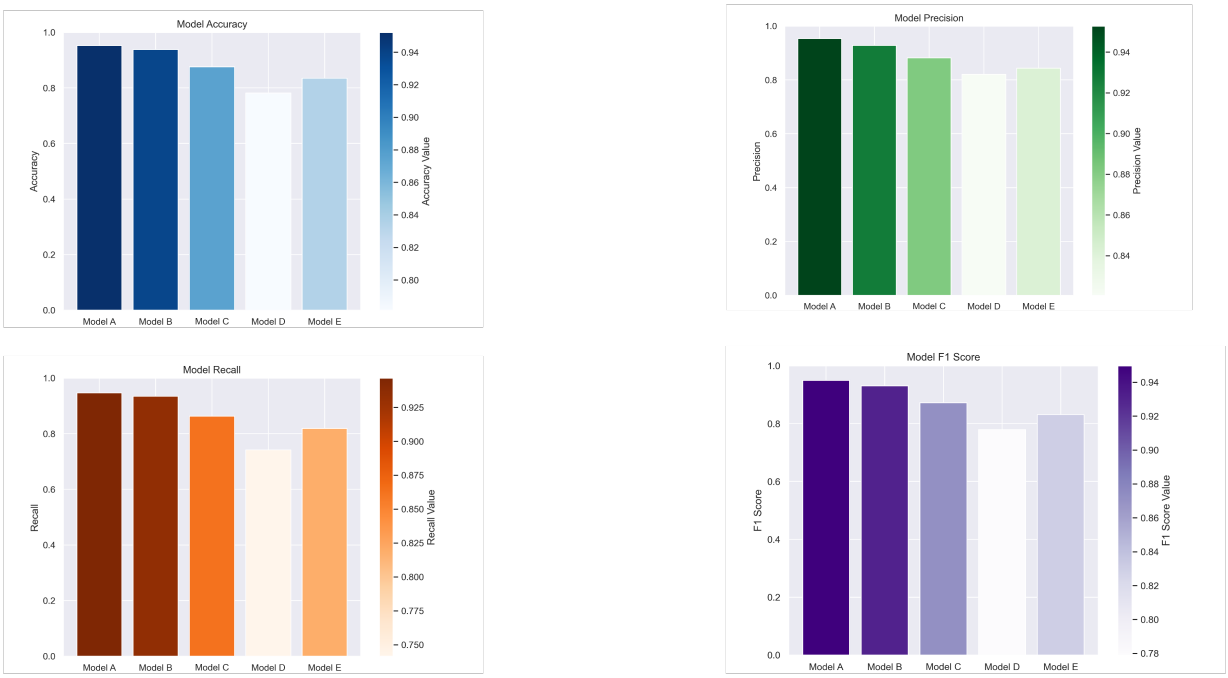


Figure 6.1: YOLO CRNet Classifier Head Accuracy, Precision, Recall & F1-Score Charts for Different Layer Configurations

As clearly indicated in Table 6.4 and Figure 6.1, Configuration A achieved the best performance across all evaluated metrics. It yielded an accuracy of 0.9522, precision of 0.9528, recall of 0.9467, and an F1-score of 0.9497. This suggests that a combination of features from early backbone layers (providing fine-grained spatial details) along with features from early and deep neck layers (providing fused multi-scale semantic context) offers the most discriminative representation for the character classification task.

Configuration B, utilizing only early backbone features, also performed commendably with an F1-score of 0.9310, indicating that low-level features alone carry significant discriminative power. However, it was slightly outperformed by Configuration A, which incorporates richer semantic information from neck layers.

Configurations C, D, and E, which rely more heavily on deeper backbone and various neck layers, showed a noticeable decline in performance. Configuration D, in particular, exhibited the lowest scores (F1-score of 0.7790). This suggests that while deep features are crucial for semantic understanding, relying solely on them without the fine-grained details from very early layers, might be detrimental to distinguishing subtle differences between character shapes.

Based on these results, Configuration A was identified as providing the best set of YOLO layers for feature extraction for the classification head. Subsequent detailed performance analyses are based on this optimal configuration.

6.3.3 Detailed Performance Metrics with Best Layer Configuration A

To further understand the classifier's behavior, a label-wise performance report was generated for Configuration A. Table 6.5 details the precision, recall, F1-score, and support (number of instances) for each digit (0-9).

Table 6.5: YOLO CRNet Performance Metrics Classification Report for Digits (0-9) with Configuration A

Label	Precision	Recall	F1-Score	Support
0	0.83	0.33	0.47	145
1	0.96	0.99	0.98	149
2	1.0	1.0	1.0	136
3	1.0	1.0	1.0	143
4	1.0	1.0	1.0	176
5	1.0	1.0	1.0	128
6	1.0	1.0	1.0	132
7	1.0	1.0	1.0	153
8	1.0	1.0	1.0	162
9	1.0	1.0	1.0	164

The classifier demonstrates exceptional performance for most digits (2-9), achieving perfect precision, recall, and F1-scores of 1.0. The digit *1* also shows very high performance (F1-score of 0.98). However, there is a notable performance drop for the digit *0*, with a precision of 0.83, a significantly lower recall of 0.33, and an F1-score of 0.47. This indicates that while the model is reasonably precise when it predicts a *0* (83% of the time it is correct), it misses a large proportion (67%) of actual *0* instances. This specific difficulty with *0* will be further explored in the confusion matrix analysis (Section 6.3.4).

The label-wise performance for lowercase letters (a-z) is presented in Table 6.6. Similarly, performance for uppercase letters (A-Z) is shown in Table 6.7.

Table 6.6: Overall YOLO CRNet Performance Metrics Classification Report (a-z)

Label	Precision	Recall	F1-Score	Support
a	0.94	0.87	0.90	108
b	0.98	0.86	0.92	58
c	0.94	0.87	0.90	68
d	0.98	0.92	0.95	64
e	0.99	0.82	0.89	83
f	0.98	0.88	0.93	58
g	0.98	0.81	0.89	59
h	0.98	0.87	0.92	68
i	0.96	0.88	0.92	106
j	0.98	0.82	0.89	49
k	0.95	0.87	0.91	71
l	0.59	0.47	0.52	70
m	0.96	0.82	0.88	61
n	0.96	0.89	0.92	95
o	0.93	0.88	0.91	78
p	0.98	0.88	0.93	65
q	1.0	0.84	0.91	50
r	0.97	0.83	0.89	71
s	0.81	0.84	0.82	73
t	0.94	0.80	0.86	83
u	0.98	0.90	0.94	72
v	0.94	0.81	0.87	57
w	0.95	0.80	0.87	66
x	0.90	0.92	0.91	62
y	0.98	0.92	0.95	59
z	0.94	0.82	0.88	61

Table 6.7: Overall YOLO CRNet Performance Metrics Classification Report (A-Z)

Label	Precision	Recall	F1-Score	Support
A	0.96	0.99	0.98	315
B	0.96	1.0	0.98	230
C	0.97	0.99	0.98	270
D	0.97	1.0	0.98	272
E	0.96	0.98	0.97	279
F	0.98	0.99	0.99	247
G	0.95	1.0	0.98	247
H	0.97	1.0	0.98	258
I	0.88	0.89	0.88	262
J	0.97	0.99	0.98	258
K	0.97	0.99	0.98	273
L	0.97	1.0	0.98	234
M	0.96	1.0	0.98	268
N	0.97	1.0	0.99	305
O	0.73	0.96	0.83	292
P	0.97	1.0	0.98	231
Q	0.97	1.0	0.98	259
R	0.96	1.0	0.98	282
S	0.96	0.96	0.96	298
T	0.96	0.99	0.98	265
U	0.97	1.0	0.98	232
V	0.94	0.98	0.96	252
W	0.95	0.99	0.97	272
X	0.98	0.97	0.97	250
Y	0.98	0.98	0.98	236
Z	0.95	0.99	0.97	210

For lowercase letters (Table 6.6), the F1-scores are generally high, mostly ranging from 0.82 to 0.95. A notable exception is the lowercase *l*, which has a significantly lower F1-score of 0.52 (Precision 0.59, Recall 0.47). This suggests the model has considerable difficulty with this character, potentially confusing it with visually similar characters like *1* or *I*. The character *s* also shows a slightly lower F1-score of 0.82.

For uppercase letters (Table 6.7), the performance is generally excellent, with most

characters achieving F1-scores of 0.96 and above, and many having perfect or near-perfect recall. However, similar to the digit *0*, the uppercase *O* exhibits a lower F1-score of 0.83, primarily due to a lower precision of 0.73, despite a high recall of 0.96. This indicates that while most *O*s are found, the model often misclassifies other characters as *O*. The uppercase *I* also has a somewhat lower F1-score of 0.88. These specific confusions will be further investigated with the confusion matrix.

6.3.4 Test Confusion Matrix Analysis

To gain deeper insights into the specific misclassification patterns of the classifier head (Configuration A), a confusion matrix was generated from the test set predictions, as depicted in Figure 6.2. The confusion matrix visualizes the performance by showing the actual versus predicted classes. The diagonal elements represent correctly classified instances, while off-diagonal elements indicate misclassifications.

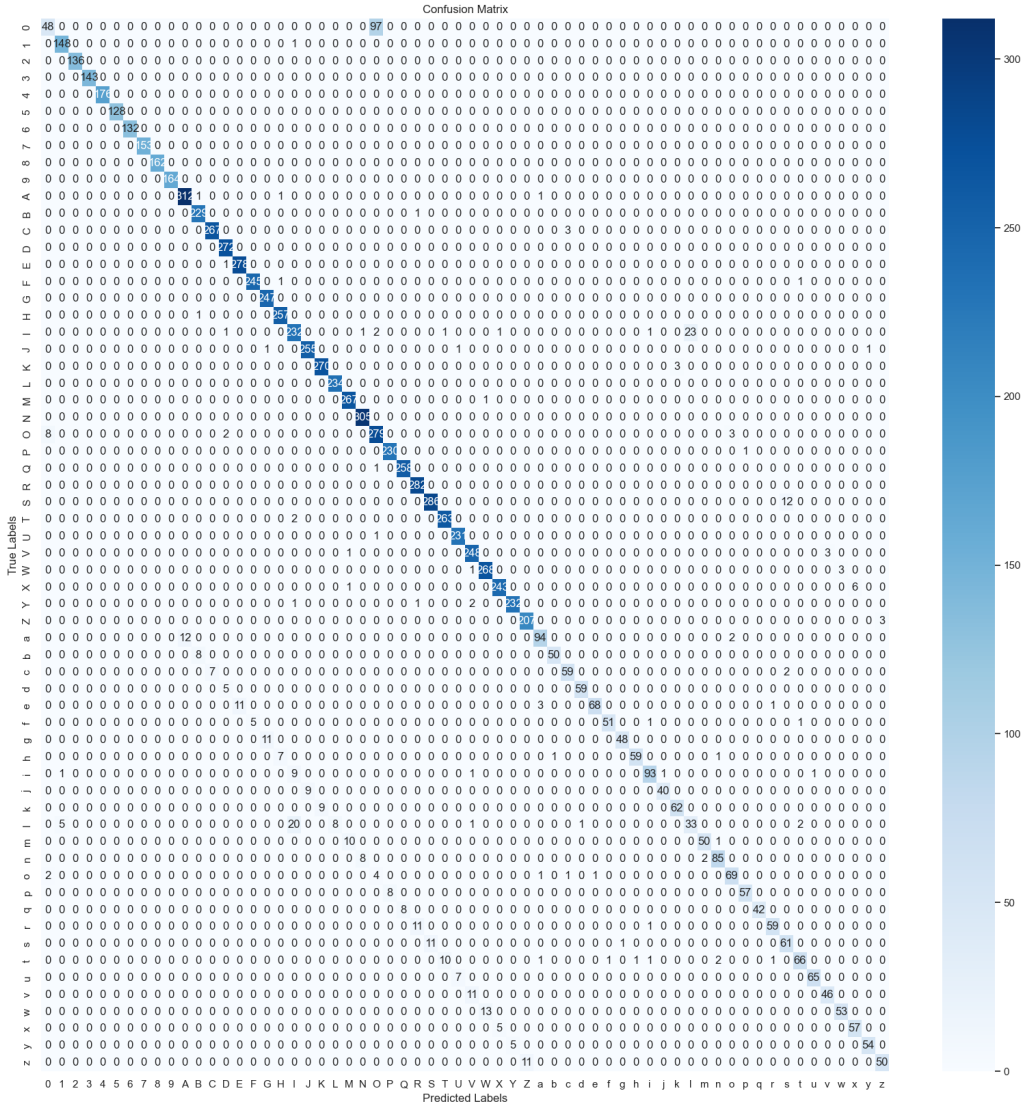


Figure 6.2: Test Confusion Matrix for Classifier Head with Configuration A

The following key misclassification patterns are observed from Figure 6.2:

- **Digit *0* and Uppercase *O*:** A significant number (97 instances) of misclassifications occur where the digit *0* is misclassified as an uppercase *O*. This aligns with the lower recall for *0* and lower precision for *O* observed in the label-wise performance

metrics. This confusion is attributed to the dataset containing characters in several fonts where the letter *o* (lowercase or uppercase) and the number *0* look identical or highly similar.

- **Letters *I*, *i*, and *l*:** Misclassifications are also noted among the characters *I* (uppercase), *i* (lowercase), and *l* (lowercase). These characters are often visually identical or very similar in many common fonts, especially when character height is normalized or in the absence of strong contextual font cues. This explains the lower F1-score for lowercase *l* and the slightly reduced performance for uppercase *I*.
- **Case Mismatches:** Smaller instances of misclassification occur where the model confuses the case of certain letters, for example, between uppercase *W* and lowercase *w*. This suggests that while the features generally capture letter shapes well, subtle differences distinguishing case for some characters might still be challenging under certain conditions or for specific fonts.

Overall, the confusion matrix highlights that the primary sources of error are due to inherent visual ambiguities between certain characters (0/O, I/i/l) that are present across multiple fonts in the dataset, rather than widespread, random misclassifications. This indicates that the feature representation from Configuration A is mostly generalized, but the model's performance is still constrained by these intrinsically confusable character pairs.

6.4 YOLO CRNet Reconstruction Head Evaluation

This section is dedicated to the experimental evaluation of the reconstruction head component of the proposed YOLO CRNet model. The primary objectives of these experiments are to assess the efficacy of different reconstruction head configurations (Configurations A, B, and C, as detailed in Section 5.4.2) in restoring degraded input images and to analyze the impact of various loss function components on the reconstruction quality.

6.4.1 Experimental Setup for Reconstruction Head

Dataset and Evaluation Task

The evaluation of the reconstruction head configurations was conducted on a held-out test set consisting of 2,000 image pairs. Each pair included a sharp, ground-truth image containing diverse alphanumeric characters with varied fonts and sizes, and its corresponding

version synthetically degraded by motion blur. The three primary configurations (A, B, and C) of the proposed YOLO CRNet model's reconstruction head were utilized to perform image reconstruction on these blurred test images.

Evaluation Metrics

The quality of the reconstructed images was quantitatively assessed by comparing them against their respective sharp ground-truth references using two widely recognized image quality metrics PSNR and SSIM defined in Section 6.2.

6.4.2 Performance Comparison of Reconstruction Head Configurations

The quantitative performance of the three reconstruction head configurations (A, B, and C) on the test set is summarized in Table 6.8.

Table 6.8: Performance of Reconstruction Head Configurations

Model Type	PSNR (dB) ↑	SSIM ↑
Reconstruction Model A	22.7353	0.6459
Reconstruction Model B	25.2442	0.8920
Reconstruction Model C	**25.6303**	**0.9044**

As presented in Table 6.8, Reconstruction Model C demonstrates the best performance, achieving the highest PSNR of 25.6303 dB and the highest SSIM of 0.9044. This configuration utilizes PixelShuffle for upsampling and incorporates Squeeze-and-Excitation (SE) blocks for channel attention within its decoder.

Reconstruction Model B, which also uses PixelShuffle for upsampling but lacks the SE blocks, achieved the second-best performance with a PSNR of 25.2442 dB and an SSIM of 0.8920. This indicates that PixelShuffle is an effective upsampling strategy. The improvement seen with Model C over Model B suggests that the inclusion of SE blocks for adaptive channel-wise feature recalibration provides a visually perceivable benefit to the reconstruction quality.

Reconstruction Model A, which employs transposed convolutions for upsampling, yielded significantly lower performance with a PSNR of 22.7353 dB and an SSIM of 0.6459. This

supports the literature suggesting that transposed convolutions can sometimes be suboptimal compared to methods like PixelShuffle, potentially due to issues like checkerboard artifacts or less effective feature propagation during upsampling [1, 52].

Based on these quantitative results, Reconstruction Model C was identified as the optimal configuration for the reconstruction head.

6.4.3 Qualitative Evaluation of Reconstructed Images

Beyond quantitative metrics, a qualitative visual assessment of the reconstructed images provides further insight into the performance of each configuration. Figures 6.3, 6.4, and 6.5 present sample reconstruction results for Model Configurations A, B, and C, respectively.

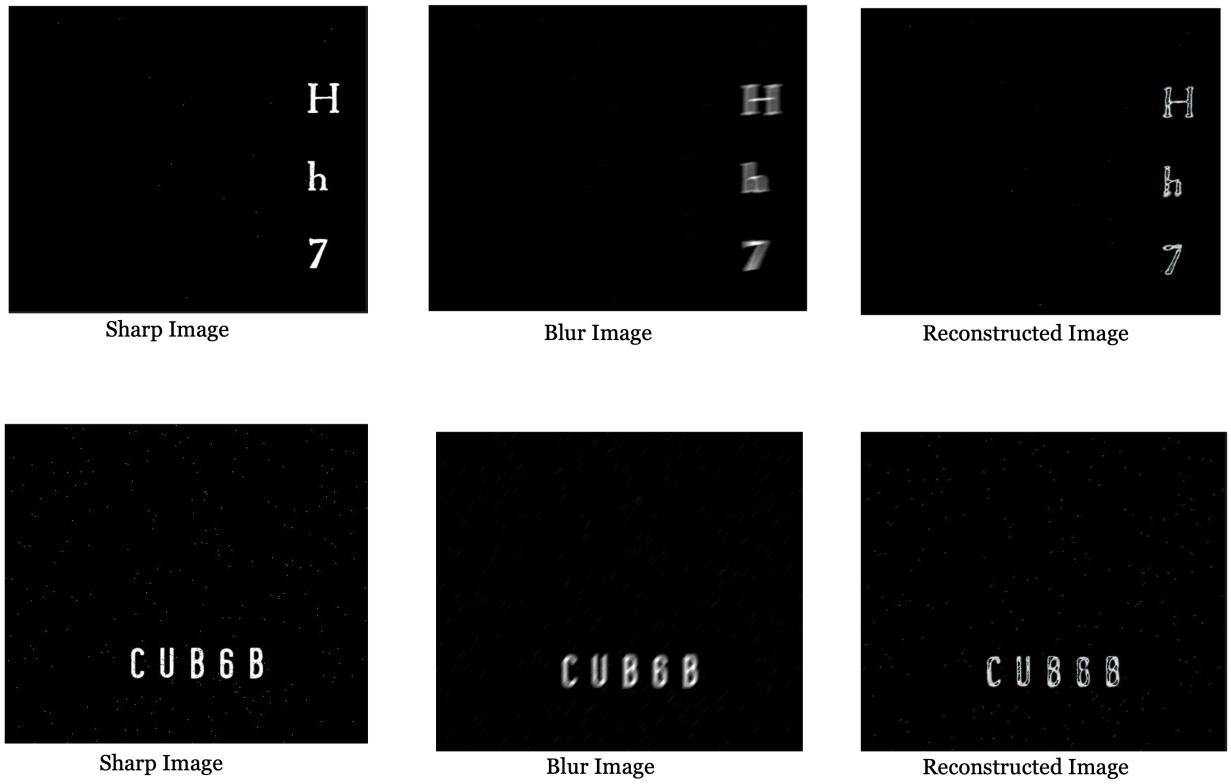


Figure 6.3: Reconstructed Result Samples of Model Configuration A

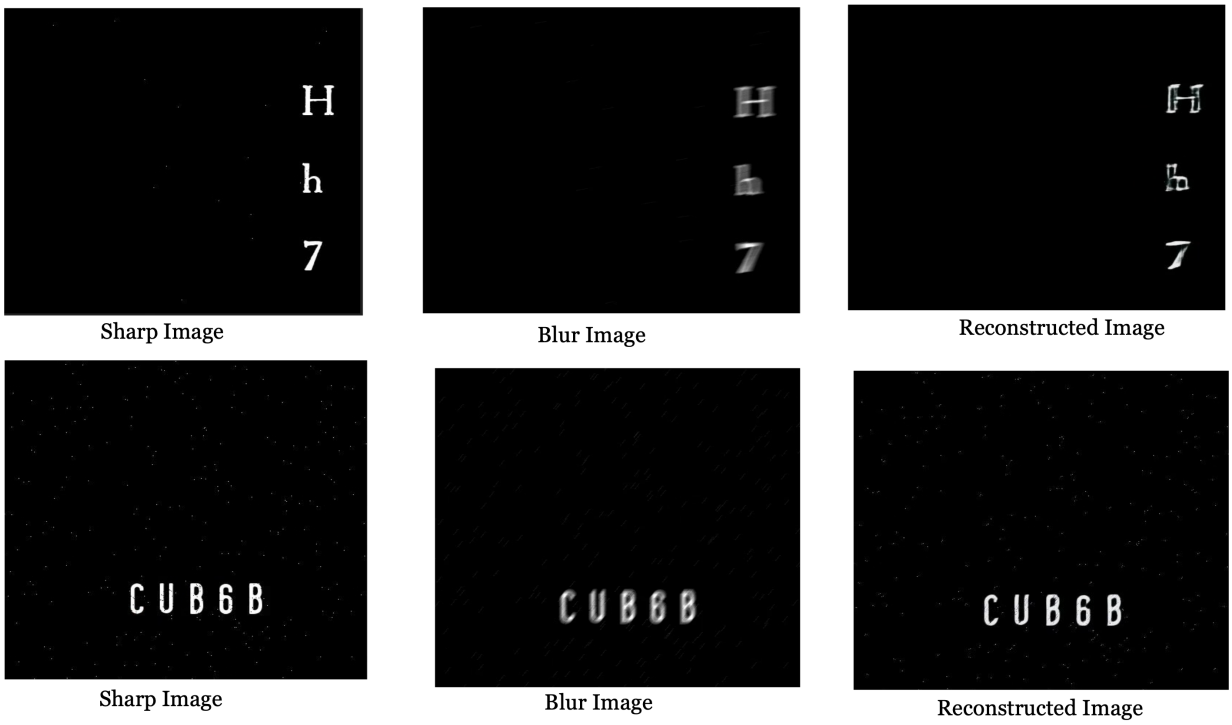


Figure 6.4: Reconstructed Result Samples of Model Configuration B

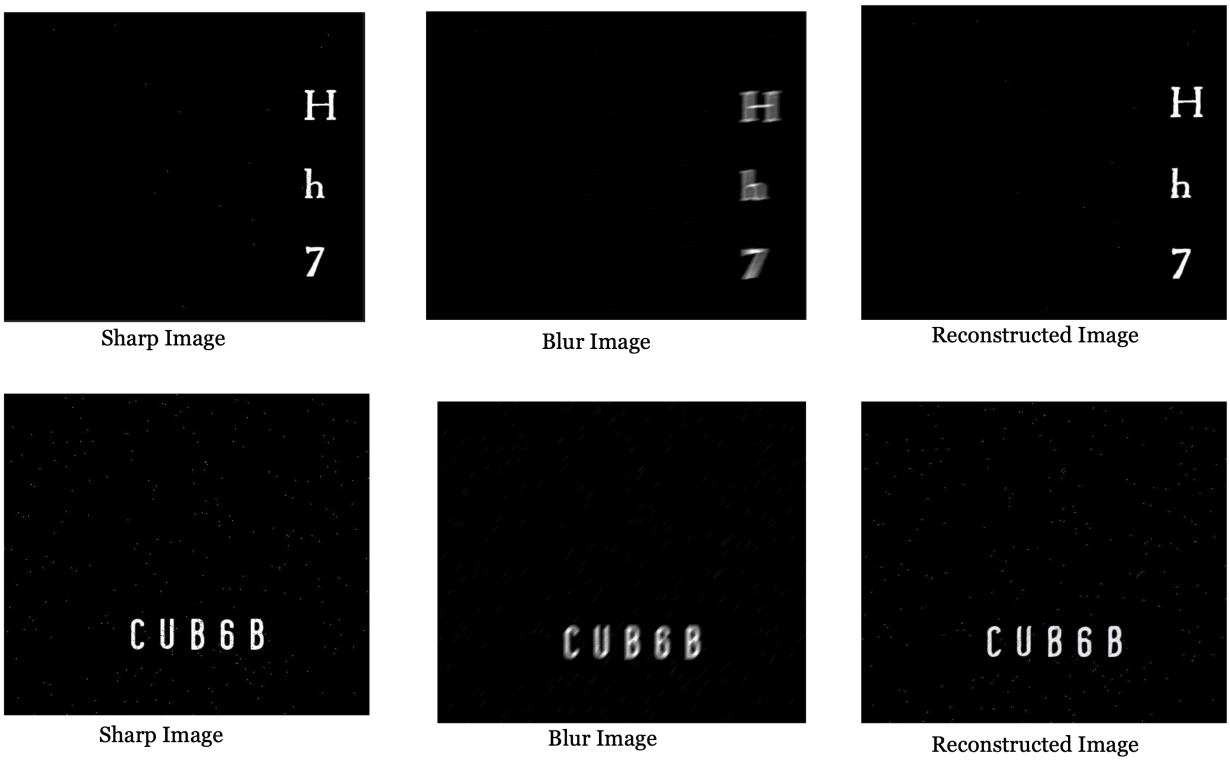


Figure 6.5: Reconstructed Result Samples of Model Configuration C

Visual inspection of the reconstructed samples aligns with the quantitative metrics. Images produced by Model C (Figure 6.5) generally exhibit the best visual quality, with sharper characters and clearer details compared to the input blurred images. Model B (Figure 6.4) also produces good quality reconstructions, though sometimes produce less refined edges (with ringing effect) or textures than Model C. Model A (Figure 6.3), consistent with its lower quantitative scores, tends to produce images that contain checkerboard artifacts and the edges are also not properly formed. The characters in Model C's output appear most sharp and legible.

6.4.4 Ablation Studies on Reconstruction Loss Components

To understand the contribution of individual components within the reconstruction loss function defined in equation 3.4, ablation studies were performed by removing or altering

the weighting of each component and observing the impact on reconstruction quality. These studies were conducted using the best performing architecture (Model C).

Impact of Perceptual Loss

Perceptual loss compares images based on high-level features like textures, edges, and semantic information extracted by a pre-trained deep neural network (VGG19). Its inclusion encourages the generated (reconstructed) image to have similar texture and structural content as the ground-truth sharp image, even if individual pixel values do not perfectly match. The ablation study (visualized in Figure 6.6) demonstrates that when perceptual loss is omitted, the overall texture and edges in the reconstructed image are not learned well. The example shows that background information is lost, and the characters lack the fine details and natural appearance achieved when perceptual loss is included. This underscores the importance of perceptual loss for generating visually realistic and detailed reconstructions beyond simple pixel-level accuracy.

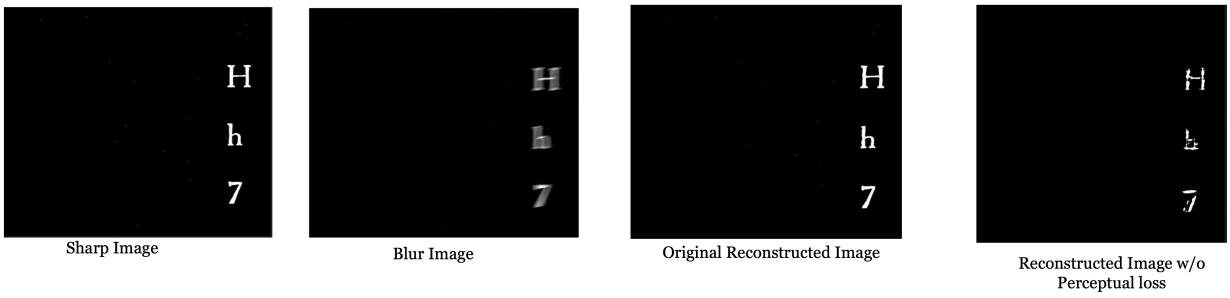


Figure 6.6: Ablation Study: Impact of Perceptual Loss

Impact of L_1 Loss

L_1 loss (Mean Absolute Error) is known for preserving high-frequency details such as edges and textures, and it tends to be more robust to outliers than L_2 loss. It focuses on accurately reconstructing details without overly penalizing sharp transitions, often producing more perceptually pleasing and sharper results, particularly in super-resolution and deblurring tasks [83]. A higher weighting for the L_1 loss component can lead to sharper images with better edges or clearer character boundaries. The ablation study without L_1 loss (visualized in Figure 6.7) shows that when L_1 loss is removed, the edges in the reconstructed characters are not sharp and the image appears visibly smoothed or blurry.

This finding highlights the crucial role of L_1 loss in maintaining sharpness and preventing excessive blurring in the reconstruction output.

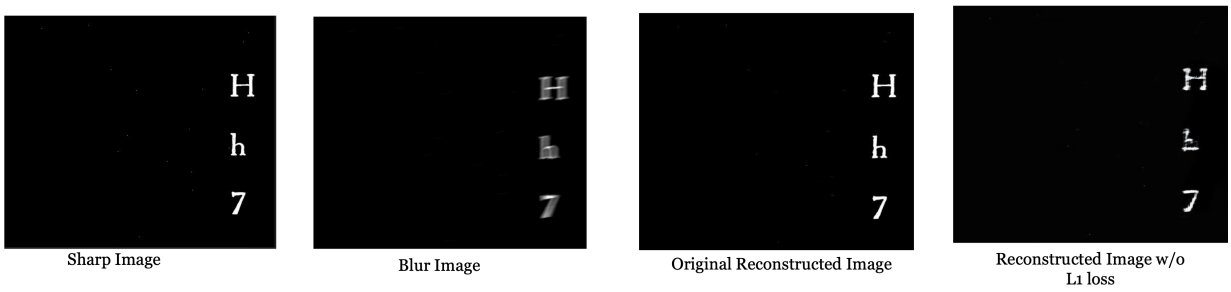


Figure 6.7: Ablation Study: Impact of L_1 Loss

Impact of L_2 Loss

The L_2 loss (Mean Squared Error) function has a smooth gradient everywhere, which can sometimes make the optimization process more stable compared to L_1 loss, especially in the initial stages of training [83]. It strongly penalizes large errors, pushing the model towards accurately predicting low-frequency information, such as the overall structure and smooth areas of an image, ensuring the *big picture* is not wildly off. However, a higher relative weighting for L_2 loss can lead to an overly smooth or blurry image, as it tends to average out fine details to minimize squared errors [83]. The ablation study without L_2 loss (visualized in Figure 6.8) indicates that when L_2 loss is absent, the overall structure of the reconstructed image may not be well learned, and characters can exhibit broken edges or appear somewhat disjointed. This suggests that while L_1 and perceptual losses are key for detail and realism, L_2 loss, even if weighted lower, plays a role in maintaining structural coherence and aiding stable optimization.

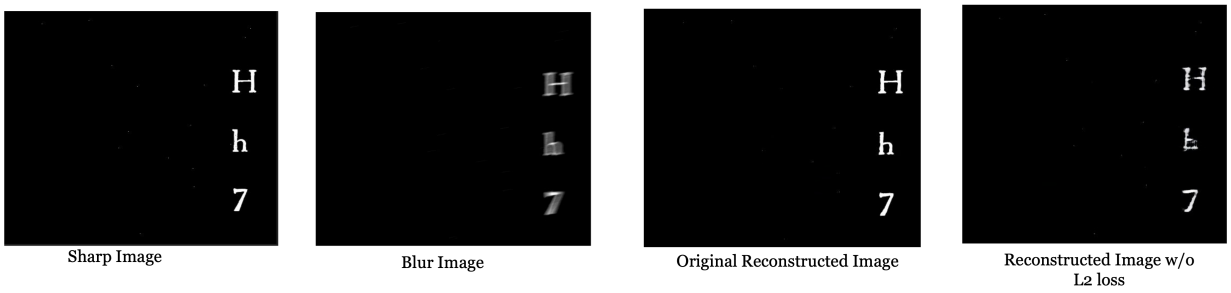


Figure 6.8: Ablation Study: Impact of L_2 Loss

These ablation studies collectively demonstrate that a carefully balanced combination of L_1, L_2, and perceptual losses, with an emphasis on L_1 and perceptual components for detail and realism, is essential for achieving high-quality image reconstruction with the proposed YOLO CRNet architecture.

6.5 Overall YOLO CRNet Pipeline Evaluation

This section presents a comparative evaluation designed to assess the overall impact and effectiveness of the proposed YOLO CRNet framework, particularly focusing on the contribution of its integrated reconstruction and classification stages when dealing with degraded (blurred) input images. The experiments aim to quantify the performance gains achieved by the different operational modes of YOLO CRNet compared to a baseline approach.

6.5.1 Experimental Design and Models Compared

The evaluation was conducted on a held-out test set consisting of 2,000 image pairs. Each pair comprised a sharp ground-truth image, containing diverse alphanumeric characters with varied fonts and sizes, and its corresponding version synthetically degraded by motion blur, as detailed in the dataset description (Section 4.1). All models and configurations were evaluated against these blurry test images to simulate real-world challenging conditions.

A comparative evaluation was structured to assess the impact of the proposed YOLO CRNet framework through the following approaches:

1. **Baseline YOLO:** To establish a performance benchmark and quantify the impact of image degradation, a standard YOLOv8 model was employed. This baseline model was initially trained exclusively on the sharp, non-degraded images from the training partition of the dataset. The model's architecture and training hyperparameters were kept consistent with those intended for the YOLO detection component within the proposed YOLO CRNet, but without any integrated reconstruction capabilities or multi-task objectives involving image restoration.

   During the evaluation phase for this baseline, the model was then directly applied to the *blurred* test images. This experimental condition simulates a common real-world scenario where a pre-trained object detector, optimized for clear imagery, encounters degraded inputs for which it has not been specifically adapted through robust training

on blurred data or by an explicit enhancement module. No image reconstruction pre-processing was performed on these blurred test images before they were fed to the baseline YOLO model. This approach allows for a direct assessment of how much standard detection and recognition performance deteriorates when faced with blurred inputs and, consequently, the extent to which the proposed methods can mitigate this degradation.

2. **YOLO CRNet Classifier (YOLO CRNet C):** The YOLO CRNet model, with its integrated classification head using the optimal feature extraction from Configuration A but with the reconstruction head's output not explicitly fed to the classifier. The classifier head uses features from blurred input processed by shared backbone and was evaluated on the blurred test images. Similar to the baseline method, this experimental condition simulates a common real-world scenario where a multi-task model, optimized for clear imagery, encounters degraded inputs for which it has not been specifically adapted through robust training on blurred data or by an explicit enhancement module. No image reconstruction pre-processing was performed on these blurred test images before they were fed to the integrated classifier head.

3. **YOLO CRNet Reconstruction + Classifier (YOLO CRNet RC):** Subsequently, the YOLO CRNet model was evaluated using an explicit two-stage process. First, the reconstruction head utilizing the best performing Configuration C processed the blurred test images to produce reconstructed versions. Then, the integrated classification head utilizing the best performing feature set from Configuration A processed these reconstructed outputs for character recognition. This setup explicitly tests the benefit of the image restoration stage prior to classification. This assesses the benefit of the multi-task trained feature representation itself.

The performance of these three approaches was quantitatively compared using Accuracy, Precision, Recall, and F1-Score, focusing on the final character classification output.

6.5.2 Comparative Classification Performance on Reconstructed Images

The classification performance results for the three evaluated approaches on the blurred test images are presented in Table 6.9 and visualized in the comparison charts in Figure 6.9.

Table 6.9: Accuracy, Precision, Recall & F1-Score Comparison of Various Models/Configurations with Blurred Input Images

Model Type	Accuracy	Precision	Recall	F1-Score
YOLO Baseline	0.4446	**0.7708**	0.4277	0.5502
YOLO CRNet C	0.5515	0.7511	0.4989	0.5996
YOLO CRNet RC	**0.6800**	0.7609	**0.6558**	**0.7045**

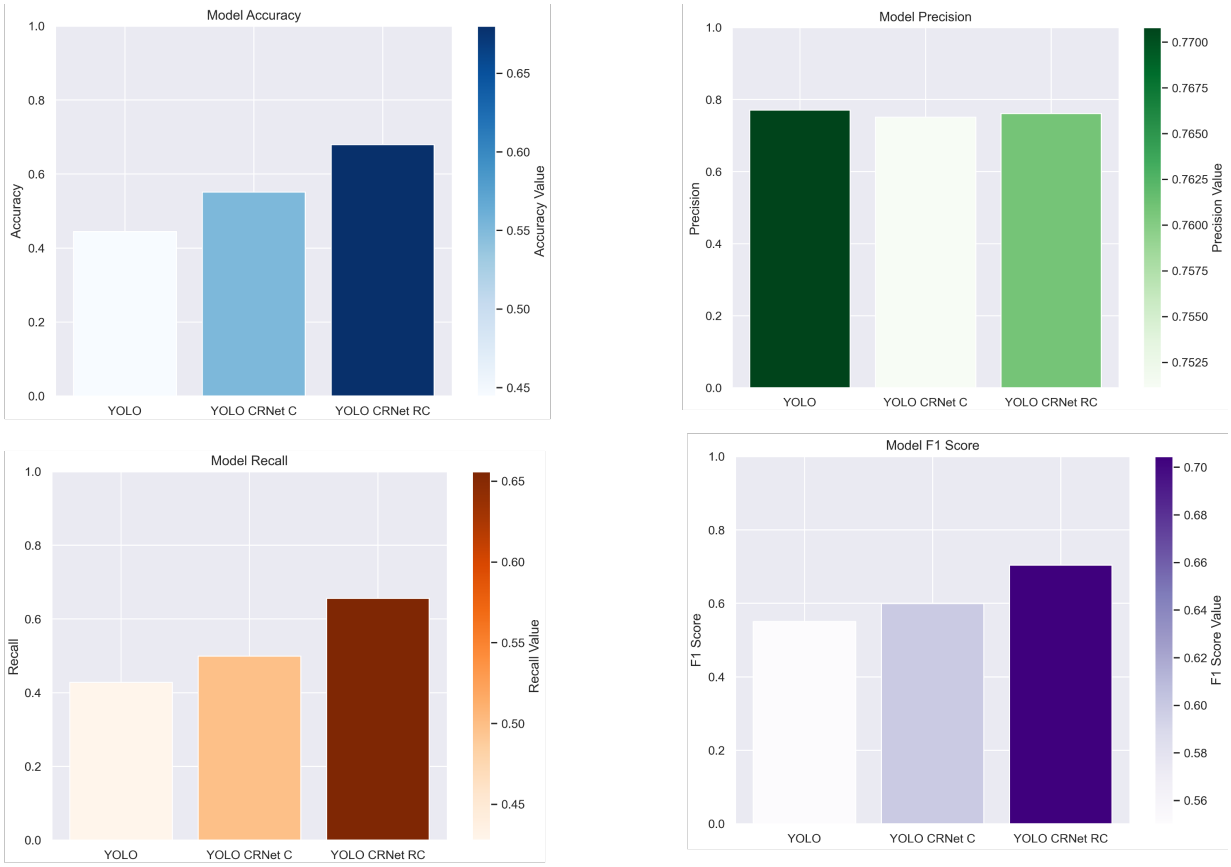


Figure 6.9: Accuracy, Precision, Recall & F1-Score Comparison Chart of Various Models with Blurred Images

6.5.3 Analysis and Discussion of Pipeline Evaluation Results

The results presented in Table 6.9 and Figure 6.9 clearly demonstrate the effectiveness of the proposed YOLO CRNet framework in improving character recognition performance on blurred images.

Baseline Performance The baseline YOLO model, when applied directly to the blurred test images, achieved an accuracy of 0.4446 and an F1-score of 0.5502. This relatively low performance highlights the significant challenge posed by image blur to standard object detection and recognition pipelines that do not incorporate mechanisms to handle such degradations. The precision (0.7708) is notably higher than the recall (0.4277), suggesting that while the predictions made are often correct, the model misses a large number of characters entirely due to the blur.

Impact of YOLO CRNet Direct Classification on Blur (YOLO CRNet C) The YOLO CRNet C configuration, representing the end-to-end trained YOLO CRNet model where the classifier operates on features derived from the blurred input but potentially benefiting from the multi-task learning of the shared backbone, shows an improvement over the baseline. It achieved an accuracy of 0.5515 and an F1-score of 0.5996. The precision dropped slightly to 0.7511, while the recall improved to 0.4989. This suggests that this specialized classification head helps in identifying more characters, but with the cost of slightly lower precise classification.

Impact of Explicit Reconstruction (YOLO CRNet RC) The most significant performance gains are observed with the YOLO CRNet RC approach. In this two-stage configuration, the blurred images were first processed by the best reconstruction head (Configuration C) and then classified using the best classification head (using features from Configuration A). This approach yielded the highest accuracy of 0.6800, recall of 0.6558, and an F1-score of 0.7045. The substantial improvement in all metrics, particularly in recall (from 0.4277 for baseline to 0.6558) and F1-score (from 0.5502 to 0.7045), underscores the critical contribution of the reconstruction stage. By explicitly deblurring the images, the reconstruction head provides cleaner and more discriminative features to the classification head, enabling it to recognize characters more accurately and to identify characters that were previously missed due to severe blur.

Overall Conclusion from Pipeline Evaluation The quantitative comparison demonstrates that the YOLO CRNet pipeline, especially when leveraging its reconstruction capabilities explicitly before classification (YOLO CRNet RC), significantly outperforms the baseline approach of applying a standard model directly to blurred images. This highlights the effectiveness of the integrated reconstruction stage within the YOLO CRNet framework in enhancing image features and the features learned by the MTL setup consequently leads to substantially improved classification performance on degraded images. This validates the core hypothesis of this thesis that integrating image reconstruction directly within the detection and recognition pipeline is a beneficial strategy for handling challenging real-world conditions.

Chapter 7

Conclusion and Future Work

This chapter synthesizes the key findings of the research presented in this thesis in section 7.1, outlines the challenges encountered and limitations of the current work in section 7.2, and proposes potential avenues for future research to further advance the field of robust text recognition in degraded images in section 7.3.

7.1 Conclusion

This research successfully extended the capabilities of the YOLO object detection framework by developing and integrating novel reconstruction and classification heads, thereby enabling a versatile multi-task learning (MTL) approach. We presented and evaluated an architecture, termed YOLO CRNet, capable of simultaneously performing object (text) detection, image reconstruction/enhancement to mitigate degradations, and fine-grained character classification, all while utilizing shared features extracted from the underlying YOLO backbone and neck structures. This integrated system was specifically designed to address the challenges of recognizing text, such as license plates, in images affected by real-world degradations like blur and noise.

Our investigation into the efficacy of multi-task learning versus single-task learning paradigms highlighted the potential regularizing benefits inherent in jointly training these diverse yet complementary tasks. This joint optimization encourages the learning of more generalized feature representations. A significant finding emerged from our systematic analysis of feature extraction strategies for the character classification head. By leveraging t-SNE visualizations to assess feature separability and quantitative classifier performance

metrics, we demonstrated that, for the specific task of alphanumeric character classification, features extracted from a combination of early YOLO backbone layers (rich in spatial detail) and neck layers (providing fused multi-scale context) provided superior class separability and resulted in higher classification accuracy compared to features sourced exclusively from deeper layers. This underscores the importance of appropriate feature selection for fine-grained tasks within a multi-task architecture.

The empirical evaluations confirmed the benefits of the proposed approach. The YOLO CRNet, particularly when employing its reconstruction head to explicitly deblur images prior to classification, demonstrated a substantial improvement in character recognition accuracy on blurred test images compared to a baseline YOLO model operating without such enhancements. Different configurations of the reconstruction head were explored, with a U-Net-like architecture utilizing PixelShuffle for upsampling and Squeeze-and-Excitation blocks for channel attention (Configuration C) yielding the best quantitative (PSNR, SSIM) and qualitative reconstruction results. Ablation studies on the reconstruction loss components further validated the importance of a combined loss incorporating L_1, L_2, and perceptual terms, with L_1 and perceptual losses being critical for sharpness and visual fidelity.

In summary, this work contributes a flexible and effective multi-task YOLO-based architecture (YOLO CRNet) for improved text detection and recognition in degraded conditions. It provides valuable insights into optimal feature extraction strategies from hierarchical networks. The research demonstrates that by thoughtfully integrating task-specific heads (reconstruction and specialized classification) and carefully selecting the feature sources for these heads, the overall performance, generalization, and utility of object detection frameworks can be significantly enhanced for challenging real-world applications like ALPR.

7.2 Challenges and Limitations

Despite the promising results achieved, this research encountered several challenges, and the current implementation of YOLO CRNet has certain limitations that warrant discussion:

1. **Synthetic Data Domain Gap:** A primary challenge in developing and training models for degraded image processing stems from the reliance on synthetically generated training data. This is often necessitated by the scarcity of large-scale, paired real-world datasets containing both sharp ground-truth images and their corresponding naturally degraded versions. While synthetic data allows for controlled generation

of various degradations, it inherently creates a potential domain gap. Models trained primarily on such synthetic data may not generalize optimally to the full complexity, diversity, and nuanced interplay of degradations present in authentic real-world imagery. The specific characteristics of real-world blur kernels or noise patterns, for instance, can differ significantly from their synthetic counterparts.

2. **Reconstruction Fidelity Limits:** The implemented reconstruction head, while demonstrating significant improvements, exhibited limitations in accurately restoring very fine details, particularly for characters characterized by small or thin strokes. This limitation was especially pronounced when characters were subjected to significant levels of motion blur. While the reconstruction successfully deblurred the overall structure, the minute details essential for distinguishing certain characters or ensuring perfect legibility were sometimes not fully recovered, which can impact the subsequent recognition stage.

3. **Fine-Grained Classification Ambiguity:** Distinguishing between visually highly similar character classes (e.g., '0' versus 'O', '1' versus 'l' versus 'I') in different fonts proved to be an inherent challenge for the classification head. Achieving consistently high accuracy for these ambiguous pairs requires the extraction of extremely subtle discriminative details from the feature representations. This task becomes particularly difficult when operating on features derived from degraded images as some critical distinguishing information might be irretrievably lost or distorted by the degradation process itself.

These challenges underscore the complexity of building generalized end-to-end systems for text recognition in unconstrained environments and highlight areas where further improvements are needed.

7.3 Future Work

Building upon the findings and addressing the limitations of this research, several promising avenues for future work can be identified to further enhance the capabilities of the YOLO CRNet framework and advance the field of robust scene text recognition:

1. **Bridging Domain Gaps:** A critical area for future research is to more effectively address the synthetic-to-real domain gap. This could be pursued by:

- Exploring unsupervised or semi-supervised domain adaptation techniques. These methods could help the model adapt its learned representations from synthetic data to better align with the characteristics of real-world degraded images, potentially by leveraging large amounts of unlabeled or sparsely labeled real-world data.
- Enriching synthetic data generation pipelines. This involves developing more sophisticated and realistic degradation models that can more accurately simulate the diverse and complex blurs, noises, lighting variations, and compression artifacts found in real-world scenarios. Incorporating a wider array of background variations and textures into the synthetic data could also improve generalization.

2. **Enhancing Reconstruction Fidelity:** To improve the restoration of fine character details, especially under severe blur conditions, future work could investigate:
   - Advanced reconstruction architectures: Exploring more powerful generative models, such as advanced GANs, diffusion models, or Transformer-based architectures, specifically tailored for image restoration, might yield superior results in recovering intricate details [18].
   - Sophisticated loss functions: Employing loss functions that more explicitly promote sharpness and texture fidelity, such as adversarial losses (if using GANs), gradient profile losses (to enforce edge sharpness), or frequency-domain losses, could guide the reconstruction process more effectively towards producing highly detailed and artifact-free images [48, 18].
3. **Improving Fine-Grained Classification for Ambiguous Characters:** To enhance the classifier's ability to distinguish between visually similar and ambiguous characters (e.g., '0'/'O', '1'/'l'/'I'), several strategies could be explored:
   - Incorporation of sequential context: For character sequences, leveraging contextual information from neighboring characters using advanced sequential modeling architectures (e.g., Transformers, more powerful RNNs) could significantly improve the recognition of ambiguous characters by considering linguistic plausibility [67, 78].

Pursuing these directions could lead to even more generalized and accurate text recognition systems capable of handling a wider array of challenging real-world conditions.

References

[1] Andrew P. Aitken, Christian Ledig, Lucas Theis, Jose Caballero, Zehan Wang, and Wenzhe Shi. Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize. *CoRR*, abs/1707.02937, 2017.

[2] Sawsan AlHalawani, Bilel Benjdira, Adel Ammar, Anis Koubaa, and Anas M. Ali. License plate super-resolution using diffusion models, 2023.

[3] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. In *Advances in Neural Information Processing Systems 20 (NIPS 2007)*, pages 41–48, 2008.

[4] Uditangshu Aurangabadkar, Darren Ramsook, and Anil Kokaram. A sharpness based loss function for removing out-of-focus blur. In *2024 IEEE 26th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE, 2024.

[5] Murat Bakirci. Utilizing yolov8 for enhanced traffic monitoring in intelligent transportation systems (its) applications. *Digital Signal Processing*, 152:104594, 2024.

[6] Halit Bakır and Rezan Bakır. Evaluating the robustness of yolo object detection algorithm in terms of detecting objects in noisy environment. *Journal of Scientific Reports-A*, page 1–25, 2023.

[7] Christian Bartz, Haojin Yang, and Christoph Meinel. See: Towards semi-supervised end-to-end scene text recognition. 2017.

[8] Jonathan Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

[9] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In Grégoire Montavon, Genevieve B. Orr, and Klaus-Robert Müller, editors,

Neural Networks: Tricks of the Trade, volume 7700 of *Lecture Notes in Computer Science*, pages 437–478. Springer, Berlin, Heidelberg, second edition, 2012.

[10] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[11] Dan Deng, Haonan Liu, Xuelong Li, and Deng Cai. PixelLink: Detecting scene text via instance segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[12] E. Deng, G. Zhou, J. Tian, Y. Liu, and Z. Jia. Text enhancement: Scene text recognition in hazy weather. In G. A. Fink, R. Jain, K. Kise, and R. Zanibbi, editors, *Document Analysis and Recognition – ICDAR 2023*, volume 14192 of *Lecture Notes in Computer Science*, pages 122–136. Springer, Cham, 2023.

[13] Steven Diamond, Vincent Sitzmann, Frank Julca-Aguilar, Stephen Boyd, Gordon Wetzstein, and Felix Heide. Dirty pixels: Towards end-to-end image processing and perception. 2021.

[14] Nikolas Ebert, Patrick Mangat, and Oliver Wasenmüller. Multitask network for joint object detection, semantic segmentation and human pose estimation in vehicle occupancy monitoring. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 1473–1480, 2022.

[15] Omar Elharrouss, Yasir Mahmood, Yassine Bechqito, Mohamed Adel Serhani, Elarbi Badidi, Jamal Riffi, and Hamid Tairi. Loss functions in deep learning: A comprehensive review. *arXiv preprint arXiv:2504.04242*, 2025.

[16] Yiming Gan, Yuxian Qiu, Lele Chen, Jingwen Leng, and Yuhao Zhu. Low-latency proactive continuous vision. Association for Computing Machinery, 2020.

[17] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[18] Haoran Gong, Yulin Li, Chen Li, Dong Liu, Junchi Yan, and Wei Feng. A dataset and model for realistic license plate deblurring. *arXiv preprint arXiv:2404.13677*, 2024.

[19] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Pearson, 4th edition, 2018.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[21] Tong He, Zhi Tian, Weilin Huang, Chunhua Shen, Yu Qiao, and Changming Sun. An end-to-end TextSpotter with explicit alignment and attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5020–5029, June 2018.

[22] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7132–7141, 2018.

[23] Wei Hua and Qing Chen. A survey of small object detection based on deep learning in aerial images. *Artificial Intelligence Review*, 58:162, 2025.

[24] Munawar Hussain. YOLO-v1 to YOLO-v8, the rise of YOLO and its complementary nature toward digital manufacturing and industrial defect detection. *Machines*, 11(7):677, 2023.

[25] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 448–456, Lille, France, 2015.

[26] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017.

[27] Jianqiang Jiang, Biaofeng Wei, Mengmeng Yu, Xuefeng Xiao, and Chuangyin Dang. An end-to-end text spotter with text relation networks. *Cybersecurity*, 4, 2021.

[28] Glenn Jocher. Ultralytics yolov5, 2020.

[29] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision (ECCV)*, volume 9906 of *Lecture Notes in Computer Science*, pages 694–711. Springer, 2016.

[30] Ming Kang, Chee-Ming Ting, Fung Fung Ting, and Raphaël C.-W. Phan. Bgf-yolo: Enhanced yolov8 with multiscale attentional feature fusion for brain tumor detection. In Marius George Linguraru, Qi Dou, Aasa Feragen, Stamatia Giannarou, Ben Glocker, Karim Lekadir, and Julia A. Schnabel, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2024: 27th International Conference, Marrakesh, Morocco, October 6–10, 2024, Proceedings, Part VIII*, volume 15008 of *Lecture Notes in Computer Science*, pages 35–45, Cham, Switzerland, 2024. Springer.

[31] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. Published as a conference paper at ICLR 2015.

[32] Dmitry Kobak and Philipp Berens. The art of using t-sne for single-cell transcriptomics. *Nature Communications*, 10(1):5416, 2019.

[33] Xiangtao Kong, Jinjin Gu, Yihao Liu, Wenlong Zhang, Xiangyu Chen, Yu Qiao, and Chao Dong. A preliminary exploration towards general image restoration. *arXiv preprint arXiv:2408.15143*, 2024.

[34] Rayson Laroca, Evair Severo, Luiz A. Zanlorensi, Luiz S. Oliveira, Gabriel Resende Goncalves, William Robson Schwartz, and David Menotti. A robust real-time automatic license plate recognition based on the yolo detector. In *2018 International Joint Conference on Neural Networks (IJCNN)*, page 1–10. IEEE, July 2018.

[35] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[36] Anat Levin, Yair Weiss, Frédo Durand, and William T. Freeman. Understanding and evaluating blind deconvolution algorithms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1964–1971, 2009.

[37] ChuMiao Li. A survey on image deblurring. *arXiv preprint arXiv:2202.07456*, February 2022.

[38] Haoxiang Li, Pin Wang, and Chunhua Shen. Reading car license plates using deep convolutional neural networks and lstms. *arXiv preprint arXiv:1709.04491*, 2017.

[39] Hui Li, Peng Wang, and Chunhua Shen. Towards end-to-end car license plates detection and recognition with deep neural networks, 2017.

[40] Xiang Li, Wenhai Wang, Lijun Wu, Shuo Chen, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. In *Advances in Neural Information Processing Systems*, volume 33 of *NeurIPS*, pages 3741–3752, 2020.

[41] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944. IEEE Computer Society, 2017.

[42] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8759–8768, Salt Lake City, UT, USA, June 2018.

[43] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, volume 9905 of *Lecture Notes in Computer Science*, pages 21–37. Springer, 2016.

[44] Wenjing Liu, Chong Chen, Chunhua Shen, Tao He, Lianwen Jin, and Liang Wang. Scene text detection and recognition: The deep learning era. *ACM Computing Surveys*, 53(6):1–38, 2020.

[45] Shangbang Long, Jiaqiang Ruan, Wenjie Zhang, Xin He, Wenhao Wu, and Cong Yao. TextSnake: A flexible representation for detecting text of arbitrary shapes. In *European Conference on Computer Vision (ECCV)*, volume 11214 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2018.

[46] Ma, Zongfang and Wu, Zheping and Cao, Yonggen. End-to-end light license plate detection and recognition method based on deep learning. *Electronics*, 12(1), 2023.

[47] Syed Zain Masood, Guang Shu, Afshin Dehghan, and Enrique G. Ortiz. License plate detection and recognition using deeply learned convolutional neural networks, 2017.

[48] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015. Published as a conference paper at ICLR 2016.

[49] Sérgio Montazzolli and Claudio Jung. Real-time brazilian license plate detection and recognition using deep convolutional neural networks. In *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 55–62, 2017.

[50] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2535–2543, 2017.

[51] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814. Omnipress, 2010.

[52] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, Oct 2016.

[53] Xuebo Pan, Cuhang Shi, Shancheng Zhang, Kai Zhang, and Jun Xiao. Efficient and accurate arbitrary-shaped text detection with pixel aggregation network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8440–8449, 2019.

[54] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pages 8024–8035. Curran Associates, Inc., 2019.

[55] Dezhi Peng, Zhenhua Yang, Jiaxin Zhang, Chongyu Liu, Yongxin Shi, Kai Ding, Fengjun Guo, and Lianwen Jin. UPOCR: Towards unified pixel-level OCR interface. In *Forty-first International Conference on Machine Learning*, 2024.

[56] Ehsan Pourjam, Hamid Sadoghi Yazdi, Hamid Kourkchi, and Hamidreza Shahamatnia. A new image enhancement and super resolution technique for license plate recognition. *Alexandria Engineering Journal*, 61(11):8799–8809, 2022.

[57] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[58] Joseph Redmon and Ali Farhadi. YOLO9000: Better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017.

[59] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[60] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, pages 91–99, 2015.

[61] Zhang X Wen Z Cheng X Jia L Yang DM Xie Y Zhan X Xiao G. Rong R, Wang S. Enhanced pathology image quality with restore–generative adversarial network. *Am J Pathol.*, 2023.

[62] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer, 2015.

[63] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[64] Utsha Saha, Binita Saha, and Md Ashique Imran. A weather-adaptive convolutional neural network framework for better license plate detection. *Sensors*, 24(23), 2024.

[65] Qingyi Sai, Jin Zhao, Degui Bi, Bo Qin, and Lingshu Meng. Improved yolov8 for gas-flame state recognition under low-pressure conditions. *Sensors*, 24(19), 2024.

[66] Abderrezzaq Sendjasni and Mohamed-Chaker Larabi. Embedding similarity guided license plate super resolution, 2025.

[67] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(11):2298–2304, 2016.

[68] Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1874–1883, 2016.

[69] Sérgio M. Silva and Cláudio R. Jung. License plate detection and recognition in unconstrained scenarios. In *European Conference on Computer Vision (ECCV)*, volume 11215 of *Lecture Notes in Computer Science*, pages 580–596. Springer, 2018.

[70] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. Published as a conference paper at ICLR 2015.

[71] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[72] Sjoerd Stallinga, Bernd Rieger, Johan Vos, Jacob P. Hoogenboom, and Yutong Wang. Noise amplification and ill-convergence of richardson-lucy deconvolution. *Nature Communications*, 16(1):911, 2025.

[73] Zhi Tian, Weilin Huang, Tong He, Pan He, and Yu Qiao. Detecting text in natural image with connectionist text proposal network. *arXiv preprint arXiv:1609.03605*, 2016. Published in ECCV 2016.

[74] Ultralytics. YOLOv8. https://github.com/ultralytics/ultralytics, 2023. Accessed: May 6, 2025.

[75] Inês Valentim, Nuno Antunes, and Nuno Lourenço. Exploring layerwise adversarial robustness through the lens of t-sne. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '24 Companion)*, page 4, Melbourne, VIC, Australia, July 2024. ACM.

[76] Laurens van der Maaten. Accelerating t-sne using tree-based algorithms. *Journal of Machine Learning Research*, page 3221–3245, 2014.

[77] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[78] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, 2017.

[79] Fang Wan, Chen Sun, Hongyang He, Guangbo Lei, Li Xu, and Teng Xiao. Yolo-lrdd: A lightweight method for road damage detection based on improved yolov5s. *EURASIP Journal on Advances in Signal Processing*, 2022(1):98, 2022.

[80] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[81] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. How to use t-SNE effectively. *Distill*, 2016.

[82] Zheng S. Xu, Li and Jiaya Jia. Two-phase kernel estimation for robust motion deblurring. In *Computer Vision – ECCV 2010*, pages 157–170, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[83] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on Computational Imaging*, 3(1):47–57, 2017.

[84] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. Distance-iou loss: Faster and better learning for bounding box regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34 of *AAAI*, pages 12993–13000, 2020.

[85] Sergey Zherzdev and Alexey Gruzdev. LPRNet: License plate recognition via deep neural networks. 2018.

[86] Xinyu Zhou, Cong Yao, He Wen, Yunchao Wang, Shuchang Wei, Jun He, and Jiajun Liang. EAST: An efficient and accurate scene text detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5551–5560, 2017.