

PlateSegFL: A Privacy-Preserving License Plate Detection System using Federated Segmentation Learning

By

Md. Shahriar Rahman Anuvab

21101094

Mishkat Sultana

19201028

Md. Atif Hossain

19201016

Shashwata Das

23141054

Suvarthi Chowdhury

21301718

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
School of Data and Sciences
Brac University
May 2023

© 2023. Brac University
All rights reserved.

Declaration

It is hereby declared that

1. This thesis paper is our/own own original work while studying in Brac University.
2. The thesis paper does not contain any material which was previously used or was written by a third party and any kind of reference was appropriately cited through and accurately referenced.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Anurab

Md. Shahriar Rahman Anuvab

Mishkat

Mishkat Sultana

Atif

Md. Atif Hossain

Shashwata Das

Shashwata Das

Suvarthi

Suvarthi Chowdhury

Approval

This thesis titled “PlateSegFL: A Privacy-Preserving License Plate Detection System using Federated Segmentation Learning” submitted by

1. Md.Shahriar Rahman Anuvab
2. Mishkat Sultana
3. Md. Atif Hossain
4. Shashwata Das
5. Suvarthi Chowdhury

of Spring, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on Jun 25, 2023

Examining Committee:

Supervisor:
(Member)



Rafeed Rahman
Lecturer
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)



Shahriar Rahman Rana
Lecturer
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)



Dibyo Fabian Dofadar
Lecturer
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi
Associate Professor and Chairperson
Department of Computer Science and Engineering
Brac University

Ethics Statement

Through this statement, we declare our research paper did not break any laws, protecting human welfare, rights or dignity. The data was collected with the proper consent and acknowledgement of everyone involved. We also emphasized on proper citation of multiple dependable sources. We aspire that our work would reach our intended community and will benefit them greatly.

Abstract

ALPR is an integral component of an intelligent transport system with extensive applications in secure transportation, vehicle-to-vehicle communication, vehicular networks, passenger protection, and automatic vehicle authorization. In addition, it has numerous applications in the detection of stolen vehicles, traffic violations, and traffic flow management. Focusing on the identification of regions of interest from available recordings or image sources, learning-based APLR system development focuses on the detection of regions of interest. The existing license plate detection system focuses on one-shot learners or pre-trained models that operate with a geometric bounding box, limiting the model's performance to selecting a geometric bounding box and thus providing more or less information than necessary. Furthermore, continuous video data streams uploaded to the central server as a result of an increase in data traffic which in return creates network and complexity issues. To combat this regulatory obfuscation, we proposed a method that implements federated learning via Unet-based segmentation with Federated learning. U-Net is renowned for its capacity to analyze high-resolution images and generate segmentation maps with a high degree of precision. UNet is well-suited for multiclass image segmentation tasks because it can analyze a large number of classes and generate a pixel-level segmentation map for each class. U-Net is capable of segmenting images at the pixel level. Federated learning is used to reduce the quantity of data required while safeguarding the user's privacy. Different computing platforms, such as mobile phones, are able to collaborate on the development of a standard prediction model, it makes efficient use of one's time, it incorporates more diverse data, it delivers projections in real-time, and it requires no physical effort from the user. The F1 dependability of our model hovers around 95%.

Keywords: Federated Learning. U-Net. Convulation Neural Network (CNN).ALPR. F1.

Acknowledgement

We are thankful to the Almighty Allah and his graciousness for helping us in enabling us to finish writing our thesis paper without any hurdles and obstruction. We also would like to thank our Supervisor and Co-Supervisor, Mr. Rafeed Rahman, Mr. Shahriar Rahman Rana and Mr. Dibyo Fabian Dofadar for their most benevolent and accommodating assistance and consultations. We highly appreciate everyone involved in the making of our thesis paper and also appreciate their time and effort. And our parents with their prayer and ongoing support which helped us in reaching where we are now.

Table of Contents

Declaration	i
Approval	ii
Ethics Statement	v
Published Submitted Papers	vi
Abstract	vi
Acknowledgment	vii
Table of Contents	viii
List of Figures	x
List of Tables	xi
Nomenclature	xii
1 Introduction	1
1.1 Problem Statement	1
1.2 Background Information	3
2 Related Work	5
2.1 YOLO	5
2.2 Yololess	8
2.3 U-Net	15
2.4 Federated Learning (FL)	16
3 Research Objectives	18
3.1 Workflow	18
4 Dataset	20
4.1 Data Collection	20
4.2 Data Labelling	21
4.3 Augmentation	22
4.4 Dataset Details	24
4.4.1 Data Description	24
4.4.2 Segment Dataset A	25
4.4.3 Segment Dataset B	27

4.4.4	Segment Dataset C	28
4.4.5	Segment Dataset D	29
4.4.6	Segmented Dataset Merged	30
5	Methodology	32
5.1	U-Net	32
5.2	Federated Learning (FL)	33
5.3	YOLO	34
5.4	OCR	35
5.5	Proposed Method	36
5.5.1	U-Net	36
5.5.2	Federated Learning(FL)	43
5.5.3	OCR	45
6	Experimental Results	47
6.1	System Details	47
6.2	Statistical Metrics	48
6.3	Quantitive Analysis	51
6.3.1	Aggregator comparision	51
6.3.2	Segmented Dataset A	52
6.3.3	Segmented Dataset B	53
6.3.4	Segmented Dataset C	54
6.3.5	Segmented Dataset D	55
6.3.6	Segmented Dataset Merged	56
6.4	Qalitative Analysis	57
6.4.1	Chapter comparison	57
6.5	Model Performance	62
6.5.1	Federated U-Net	62
6.5.2	U-Net	64
6.5.3	YOLO	66
6.6	Model Comparison	68
7	Discussions and Further Improvements	70
7.1	Discussion	70
7.2	Further Improvement	72
8	Conclusion	74

List of Figures

1.1	Analyzing the road incidents involving vehicle vs undetected vehicle .	3
2.1	Graphical representation of literature review statistics	17
3.1	System overview of proposed approach Federated U-Net model	19
4.1	Process of data labelling	22
4.2	Representation augmentation data	23
4.3	Differnt types of vehicle with visible number plate	25
4.4	Segmented dataset A	26
4.5	Segmented dataset B	27
4.6	Segmented dataset C	28
4.7	Segmented dataset D	29
4.8	Segmented dataset merged	30
4.9	Distribution of total dataset	31
5.1	U-Net architecture	33
5.2	Proposed U-Net model	38
5.3	Proposed FL architecture	44
5.4	OCR-1	45
5.5	OCR-2	46
6.1	Segmented dataset A	57
6.2	Segmented dataset B	58
6.3	Segmented dataset C	59
6.4	Segmented dataset D	60
6.5	Merge dataset	61
6.6	Epoch-wise recall, precision, loss, accuracy for FL U-Net model . . .	62
6.7	Epoch-wise recall, precision, loss, accuracy for U-Net model	64
6.8	Epoch-wise recall, precision, loss, accuracy for YOLO model	66
6.9	Model comparison	68
7.1	Workflow of future improvement	73

List of Tables

4.1	Vehicle types and percentages	21
5.1	Layer parameters of encoder	39
5.2	Layer parameter of deccoder	40
5.3	Parameter summary	40
6.1	Model evaluation metrics for FL agregators	51
6.2	Model evaluation metrics for dataset A	52
6.3	Model evaluation metrics for dataset B	53
6.4	Model evaluation metrics for dataset C	54
6.5	Model evaluation metrics for dataset D	55
6.6	Model evaluation metrics for merge dataset	56
6.7	Federated U-Net table	63
6.8	U-Net table	65
6.9	YOLO table	67

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

ANN Artificial Neural Network

CNN Convolutional Neural Networks

DL Deep Learning

FL Federated Learning

KNN K-Nearest Neighbour

OCR Optical Character Recognition

R – CNN Reason-Based Convulation Neural Network

SVM Support Vector Machine

YOLO You Only Look Once

Chapter 1

Introduction

1.1 Problem Statement

Vehicles are one of the most significant inventions of the modern era, as they have made communication and transportation more convenient and effective. However, as the number of vehicles continues to rise, we are also encountering difficulties with surveillance, as there is still the possibility that a vehicle could be involved in larceny, robbery, abduction, or even terrorism. Keeping note of all of these vehicles manually will result in human error, inefficiency, and sluggish response times, among other issues. In this instance, having a license plate detection system will help us avoid these human errors and, as a result, create a much more efficient and secure system that will aid in reducing crime and ensuring the safety of people.

Among the numerous [38], [53], [65], [67], [69], [70], published studies on license plate identification systems, geometric bounding box segmentation is the most popular technique. The model operates by receiving an input image and dividing it into a grid. If the center of an object is contained within a square cell, the cell can identify the object. For each cell, the bounding boundaries for each box and their confidence levels are predicted. However, the use of bounding boxes raises concerns regarding the potential for irregular masking. To address this difficulty, we introduced the U-Net design because it can be implemented even when the masking is inconsistent, such as when there are photos of varying sizes or other variables that could compromise the accuracy of the license plate detection system.

Our proposed license plate detection methodology is a combination of U-Net with federated learning in order to protect the user's privacy. In recent years, federated learning has emerged as a privacy-preserving technique for machine learning. Using this method, multiple stakeholders can train a single model without revealing their training data. Information-theoretically speaking, Security Association(SA) protocols assure that:

- The server and other users are unaware of any user's explicit model update from an encrypted update.
- The server only learns the aggregated model.

The challenges we face during the development of our model include:

- The accumulation of datasets for license plate detection.
- Transmission overhead.
- The existence of irregular masking during image segmentation.

Therefore, the primary contributions of our paper are:

- The compilation of datasets for license plate detection with irregular mask labelling
- Irregular mask segmentation using the proposed U-Net architecture
- The reduction of transmission overhead while maintaining user privacy using Federated Learning
- The text extraction from images using Bangla and English OCR.

Our proposed model, intends to build a for license plate detection using a combination of U-Net and Federated learning. We utilize U-Net as it is more robust and has the capacity to surmount the irregular frames of license plates in undesirable situations. Federated learning will also be implemented due to its ability to maintain the privacy of user data, as it can operate in a decentralized manner that does not compromise the user's essential data, which could result in the loss of privacy for the individuals involved. And ultimately, we will extract the images of the license plates using OCR.

At the end of our research, we tend to:

- Utilize U-Net segmentation for irregular masking as a result of this research.
- Utilize Federated learning for concerns about privacy.
- OCR is used for image extraction.
- Improve knowledge of license plate detection and irregular masking.
- Have a complete comprehension of the deep learning model.
- Evaluate and compare our model to other models.

1.2 Background Information

The significance of vehicle plate detection is obvious, as the number of vehicles has increased exponentially over the last decade. Keeping track of all of these vehicles in public or private areas is not possible only with the capabilities of the human mind. Also, the way vehicle accidents and crime increase in Bangladesh is too much of a threat for us. According to the statistics from 1992–2003, the number of accidents has increased [2] by more than 20%, along with the death rate. Also, the [13] paper claims that in 2009, according to police data, the number of vehicles involved in accidents was 376, whereas the unofficial figure was 390, and the missing car rate was 4%. And the reason for missing vehicles was mostly due to car tracing [13]. Also, [13] states that among the vehicles, the bus and the cars have the highest accident rates. But the serious issue is that, according to the ARI (Accident Research Institute), the accidents and death rates in Bangladesh increase rapidly every year. And identify the reason for that lack of responsible drivers, pedestrians, and police. Also, the most efficient reason is that tracing fails to locate the accident vehicles. That's why criminals do not get punished, and this mentality also creates careless driving and behaviour. In private areas such as military bases, government areas, colonies, and even residential areas, it is very necessary to keep track of vehicles. Any of these vehicles could be a potential threat to terrorists or may involve any kind of illegal activity which could cause any kind of mistreatment. And in public areas, such as roads and highways, there remains the possibility of robbers, kidnappers, and even stolen vehicles. It is far easier to have a license plate detection system keep track of these instead of manually doing so. Which will help achieve a far better level of efficiency with less manpower.

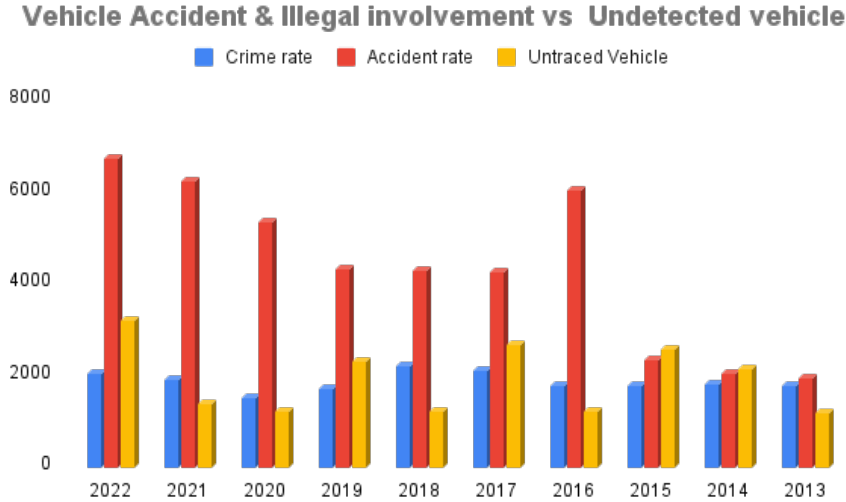


Figure 1.1: Analyzing the road incidents involving vehicle vs undetected vehicle

Figure [1.1] shows that crime rates and accident rates by have been a repeating problem in a country like Bangladesh. In present time, the culprit is not even identified due to insufficient evidence or unable to detect the vehicle becomes common scenario in our daily life. This could be avoided if there is a proper vehicle license plate detection system. Figure [1.1], shows that in the year 2022, the crime rate by

vehicle has been the highest in comparison to the last 3 years. And the accident rate has been the highest, which was almost 6800. Moreover, the undetected vehicle rate has also increased. In 2015 there remained a total of 2587 unsolved crimes.

In Dhaka City in the year 2020 there have been 102 stolen vehicles[75]. These issues could be addressed if there is a proper implementation of vehicle license plate detection systems which could reduce the illegal activities by vehicles.

License plate detection systems can be installed in car dash cams, which can help the authorities detect and find the person of interest far more efficiently. This helps create a safe and secure environment for the general public.

For vehicle plate detection, achieving efficient output with high accuracy is challenging. A license plate could have a variety of sizes, colours, shapes, alphabet, fonts and so on. In computer vision and pattern recognition, it's challenging to segment and detect complex problems. Some traditional methods depend on some basic rule-based algorithms, but they are limited by their inability to handle and detect the complexity of license plates. Moreover, these approaches face difficulties with scenarios like varying lighting conditions and low-resolution images.

To deal with these difficulties, deep learning techniques gained prominent attention and performance in various computer vision tasks. CNN (Convolutional Neural Network) is the best example, which has already shown remarkable success in detection and segmentation and also brought revolutionary changes in car plate detection. There are so many architectures for detection and segmentation based on CNN. Among them, U-Net proved more effective in semantic segmentation and combining high-level and low-level features with an encoder-decoder structure.

On the contrary, federated learning is an approach where multiple data sources collaborate with each other without sharing sensitive information. And so this will be much more suitable for scenarios that actually concern privacy, like vehicle plate detection. In vehicle plate detection, data must be handled with the utmost care. So, through federated learning, we can utilize data from diverse sources to ensure privacy.

So, in this research paper, we propose an approach for vehicle plate detection that combines the efficiency of U-Net with the benefit of federated learning while maintaining privacy. Our proposed method's aim is to achieve more accurate and efficient vehicle plate detection with the best segmentation architecture in real time. Also, federated learning enables training the data collaboratively, distributing the data with privacy, and preserving the data. That's why the proposed system enhances the capability of car plate detection.

Chapter 2

Related Work

2.1 YOLO

For the purpose of car plate detection B. Setiyono et al.[53] used Yolo V3 and Darknet-53 for character identification in the number plate recognition system to recognize license plates. With preprocessing data, the number plate identification rate was 88%, and without preprocessing data, it was 80%. Without preprocessing data, the character recognition accuracy was 97.1%, whereas with preprocessing data, it was 98.2%. Waqar et al.[38] introduced another paper with a model whose recognition rate for license plate images using Yolo V3 and CRNN was 86%. The recognition rate was raised to 96% by using temporal redundancy. The suggested approach performed 9% and 4.9% better than Sighthound and OpenALPR, respectively. Found another paper where, using the Yolo V3 for detecting Egyptian license plates, Youssef el Ata.[65] was able to get an accuracy of 97.89% and 92.46%, respectively, in the case of license plate detection along with character identification. This was done by using Tiny-Yolo v3 as a real-time ALPR system. Their suggested system was assessed using the first publicly accessible data set from the Egyptian Automated License Plate (EALPR).

Furthermore, by following the model of Zou et al.[50], which was made based on YOLO v3, along with the improved License Plate Recognition Net (SIPRNET), a two-stage license plate identification structure was suggested, which resulted in a precision rate of 99.2%. The model is able to perform remarkably well in varying conditions.. Intricate nature landscapes from the Chinese City Parking Dataset (CCPD) include CCPD-DB, CCPD-FN, CCPD-Rotate, CCPDTile, CCPD-Weather, and CCPD-Challenge. These images concern improper localization and low character recognition accuracy for real license plates.

After analyzing [38], [53], [65], [67] papers, some failings in Yolo V3, which are: Its object localization mechanism. Struggle to detect small objects and close them.

For a better approach Se-Ho et al.[62] proposed a Yolo V4-based system for integrating number plate and vehicle type identification. The system identified six different vehicle classifications and license plates, with a featured map (FM) values of 98.0%, 94.0%, 97.1%, and 84.6%, respectively. The system was able to recognize very tiny license plates that were just 100 pixels wide.

A similar research paper by Deshpande et al.[50], on the other hand, proposed a system with Yolo V4 and feature fusion. They achieved 86% accuracy and 45 frames, which were able to recognize tiny things like license plates. Their suggested approach is capable of overcoming various object detection difficulties in the case of detecting the color of the number plate and also recognizing the character of the number plate. They achieved object identification, detection frameworks, object feature representation, and object proposal.

Henceforth, more accuracy get Rishabh et al.[77], used a real-time detection system of vehicle number plates in parking lots using the Yolo V4 deep learning approach. When detecting license plates, their system had a detection accuracy of around 89%. The number was extracted from the number plate picture using an OCR method. But the paper creates an unfavourable environment that causes issues.

Furthermore ,for real time detection and identification approach with YOLO v4 . Sung et al.[41], suggested a model which is used for identification in order to avoid the ROI setup stage without sacrificing recognition accuracy. Direct plate localization on the whole picture is done while accounting for imperfect license plate detection. Using the NVIDIA Jetson TX2 module, the system detects license plates on stationary and moving automobiles traveling at high speeds. Removing the ROI configuration phase makes the license plate recognition system computationally lighter without compromising identification accuracy.

Rathi et al.[32], on the other hand, used the Yolo V4 deep learning approach for car plate detection in parking lots and achieved an accuracy rate of 89%. The number was extracted from the number plate picture using the OCR method.

Though Yolo v4 has more accuracy than Yolo V3, it cannot override all the problems. Low recall is the main identifying problem of Yolo V4. Eventually, Yolo V4 will have more localization and struggle to get small and close objects like Yolo v3.

After Yolo V3 and Yolo V4, the upgraded version of Yolo was also used for vehicle plate detection.

Like Raj et al.[63], in their research for detecting vehicle license plates accurately, they combine an Automatic Number Plate Recognition (ANPR) system that recognizes vehicle license plates accurately using Yolo V5 and a CNN model. The number plate is detected using the Yolo V5 model, and the result is segmented into different areas. For this, image processing and contouring methods were used. Afterward, the various characters are categorized using the CNN model, and the results are combined together. The result from detection is also prepared and segmented into different characters with the help of image processing and contouring techniques.

After that, Laroca et al.[8], have used ALPR as a reliable and effective system that is based on Yolo V5 object detection. Their proposed ALPR model was able to produce remarkable results in two different data sets. In data set one, which consisted of 2000 frames that were collected from 101 vehicle videos, they were able to achieve

a successful identification rate of 93.53% and 47 FPS, and by doing so, their system was able to outperform both commercial systems. On the other hand, Sighthound and OpenALPR gave recognition rates of 89.80% and 93.03%, respectively. On the other hand, the UFPRALPR data set, which was a much bigger public data set, was created for ALPR in an effort to create a scenario that was more plausible. In their data set, 150 movies and 4500 stills are included, along with a variety of moving vehicles. In the data set, their system was able to give an identification rate of 78.33% and 35 FPS.

Though it's fast and more capable of achieving and detecting goals, it's not ready for many real-time experiences and irregular alignments.

Finding paper for license plate detection using crossover yolo versions According to Sun et al.[9], they created and trained a convolutional recurrent neural network (CRNN-12) network, which achieved 98.7% accuracy. For this, both Yolo V2 and V3 were used. The results were measured and then compared in order to comprehend the model with the most accuracy. An end-to-end recognition system was implemented using convolution and recurrent operations.

In the research paper, to detect automobiles and license plates, Reda et al.[57] suggested an end-to-end ALPR pipeline based on Yolo V2 and V4. On a low-end GPU, they managed to retain adequate frames per second (FPS) and an average recognition accuracy of 90.3%. With the use of five available data sets from various areas, they assessed their methodology.

To pinpoint the precise position of license plates in a given photograph, Jimit et al.[28], implemented both Yolo V3 and Yolo V2. The final output text was generated using optical character recognition and image enhancement methods. For Yolo V3, the system obtained an IoU of 58.24% and a mAP of 53.30%, while for Yolo V2, it achieved an IoU of 57.91% and a mAP of 51.28%.

Although combined Yolo can overcome many problems about detection, according to [9], [28], [57], localization and memory allocation cannot even be solved with crossover Yolo.

2.2 Yololess

Except for Yolo, *R-CNN*, *CNN*, *RNN*, *Darknet*, *ALPR*, *SVM*, *C-RNN*, *Sighthound*, *Scale-Invariant Feature(SIFT)*, *Speed Up Robust Feature(SURF)*, *OCR* and others use car plate detection so that it can overrule the basic problems of Yolo.

For that purpose, Siddharth et al.[58], method uses Mask R-CNN, which is a successful method for reading license plates with inclinations greater than 0/60. Compared to the approach of using the Yolo V2 model, Mask R-CNN has significantly improved object identification for objects angled greater than 45 degrees. In the area of plate recognition, the suggested Mask R-CNN approach performs better than existing methods (known as the AOLP data set).

This paper's accuracy is secured, but it uses the Yolo V2 model. For that, the Yolo basic problems like localization and the struggle to detect objects are there.

Another research paper shows that a recognition rate was seen by the proposed ALPR system by Laroca et al.[49], which was achieved across eight different data sets. On the other hand, for the ChineseLP, OpenALPR-EU, SSIG-SegPlate, and UFPR-ALPR data sets, the system was able to outperform both prior works and commercial systems, being able to achieve remarkable frames based on a very powerful 1607 MHz and memory of 1251 MHz, which was done in real time even with four different cars present simultaneously. Using several models, the system was assessed and refined in an effort to achieve the optimal speed/accuracy balance. To make the system stable under many circumstances, the study combined data augmentation techniques with photos from other databases. Yolo V2 and V3 object detectors are the foundation of the system.

Undoubtedly the [49] research paper proposes a good approach for car plate detection yet the paper should be concerned more on better alignment of LP detection and invasion of privacy.

A different approach was used by Jasem et al.[33], in which they used the scale-invariant feature transform (SIFT) and speed-ed up robust features (SURF) algorithm in recognizing Iraqi license plates with the use of an ALPR system that was based on deep learning. The structure achieved an accuracy of 98.17%, with LP detection and identification accuracy of 99.2% and 97.14 percent, respectively. For testing the system, a total of 1300 images and 35 films of Iraqi automobiles in various environments were used. In the initial phase, they propose the DL-ALPR system from the start for the license plate recognition of Iraqi plates, using the most recent (YOLO) convolutional layers to detect single classes. Using a database of Iraqi license plates that the writers had gathered in the first phase, Then, secondly, the detection plates are recognized by drawing out a number of license plate attributes using the SIFT and SURF algorithms and then matching the plates in the database using the K-Nearest Neighbors Algorithm (KNN).

The research paper [33] gives new idea about car plate detection but did not demonstrate the commercial use of the system properly.

At that moment, in order to recognize Taiwanese vehicle registration plates, Hendry et al.[11], had achieved 98.22% accuracy by using the Yolo darknet deep learning framework. The system was made by evaluating several challenging conditions, which included wet backgrounds, complete darkness and other varying types of visual saturation. The system has the ability to perform a single detection and recognition phase, which takes between 800 ms and 1 s to complete. The system is able to achieve an accuracy rate of 78%.

From [11] research paper gets the idea about recognition of car registration plate using YOLO Darknet yet the accuracy is not good enough (only 78%) due to the system does not make use of character segmentation or noise reduction and uses only one phase of detection and recognition, which requires more computing time.

After that, Zhu et al.[44], emphasized the precise identification of license plate character information using the Yolo object detection network. They were able to use the network's real-time location and license plate image recognition capabilities in the system they created. The images in return are located and identified by the network. It can also be used to identify the vehicle's license plate information, which is seen in the output information of the system, in order to construct an intelligent transportation system due to :

- The high loss value.
- The fact that only 200 epochs were trained and is not established properly.

With a developed system that can swiftly and precisely identify Chinese license plates by Shenghu et al.[61], in real time based on the Yolo V7, STN, and LPRNet models. The system's average accuracy is 96.1% and it is durable in challenging conditions. Compared to the conventional license plate detection and identification system, the technology is more effective.

In time, focusing on different objects for car plate detection using many algorithms. In the first paper, Rajik et al.[37], a computationally light graph-based greedy approach for segmenting the Bangladeshi LPR system was suggested. This algorithm was based on over 2000 images that were accumulated from the roads and streets of Bangladesh. Although segmentation techniques and modules have advanced significantly, there are still certain limitations, such as the inability to accurately segment photographs that have been rotated around 20 degrees or more, photos shot at night or in poor light, or those taken from a side perspective. A greedy graph-based division technique was developed that can divide the number plate of Bangladeshi automobiles as one of the paper's main contributions. A rotated picture can be used to segment the number plates via the GGCS method (0–360 degrees). The GGCS method can extract the number plates from a picture that is inclined horizontally (0–85 degrees). The GGCS method can extract the number plates from a picture that is slanted vertically (0–84 degrees). Any related components in a picture can be segmented using the GGCS algorithm. The LRP is made through Yolo v3.

As long as the LRP is made by Yolo V3, it is facing a problem with maximization errors.

Riel et al.[23], using the UCSD-Stills and Caltech Cars datasets, demonstrate the high segmentation and recognition accuracy of the suggested model. Fast inference is made possible by the suggested model's reduced parameter requirements compared to the original SSD and VGG implementation. A per-plate processing time of 66 ms is attained using the suggested inexpensive Raspberry Pi 3 and Intel Neural Compute Stick 2 with OpenVINO embedded systems. But this [23] paper's drawback is the lack of data implementation.

Similarly with the same limitation Tariqul et al.[14], used a faster R-CNN and SSD to create a deep learning program that can instantly recognize Bangla license plates. The suggested model successfully detected the license plate with 100% accuracy and the characters on the license plate with 91.67% accuracy. The model was trained using synthetic character data from an open-sourced dataset of handwritten Bangla characters.

Then research paper on deep learning Ira et al.[16], created an application to identify license plates on Korean cars in photos. Achieved accuracy in detection and recognition of 99.8% and 95.7%, respectively. Vehicle number plate recognition with mixed neural networks using scene text recognition and geometric image transformation (GIT).

[32], Bhutanese license plate localisation using YOLO was accomplished with an overall mean average precision of 98.6% using a single convolutional neural network. Before LP localisation, vehicle detection was carried out to get rid of sign board-related false positives. Vehicle and LP both experienced a training loss of 0.0231.

[32], A different approach was used by Jasem et al. in which they used the scale-invariant feature transform (SIFT) and speeded-up robust features (SURF) algorithm in recognizing Iraqi license plates with the use of an ALPR system that was based on deep learning. The structure achieved an accuracy of 98.17%, with LP detection and identification accuracy of 99.2% and 97.14 percent, respectively. For testing the system, a total of 1300 images and 35 films of Iraqi automobiles in various environments were used. In the initial phase, they propose the DL-ALPR system from the start for the license plate recognition of Iraqi plates, using the most recent (YOLO) convolutional layers to detect single classes. Using a database of Iraqi license plates that the writers had gathered in the first phase. Then, secondly, the detection plates are recognized by drawing out a number of license plate attributes using the SIFT and SURF algorithms and then matching the plates in the database using the K-Nearest Neighbors Algorithm (KNN). This paper's drawback is that commercial use of the system was not demonstrated.

Like the above two papers with the same issue, M. S. Farag et al.[12], introduced the control of the entrance to a smart parking system that required the use of image processing techniques. The proposed approach achieved average detection rates of 97.8%, segmentation rates of 98%, and identification rates of 97%. The suggested approach works well for smart parking entrance control.

Not only the three papers, but also I. Pustokhina et al.[35], made a powerful deep

learning-based VLPR model known as the OKM-CNN model. It uses optimal K-means (OKM) clustering-based segmentation and convolutional neural network (CNN)-based recognition. The three primary stages of the proposed OKM-CNN model are the detection of license plates (LP), segmentation using the OKM clustering technique, and recognition of license plate numbers using the CNN model. Three data sets were used in a thorough experiment, and the results of the simulation ensured that the OKM-CNN model would outperform the other approaches.

Additionally, Bhargavi et al.[18], in their paper came to the conclusion that without segmentation, a CNN-GRU model was created to identify the characters in a license plate number. The proposed model had a training accuracy of 90% and a testing precision of 100%. The model's use of deep learning beats more conventional methods like template matching.

Focusing on using better CNN P. Marzuki et al.[17], suggested using a better convolutional neural network (CNN) algorithm in the identification of number plates. Before feeding the input to CNN, preprocessing methods such as Sobel operator edge detection, morphological operation, and connected component analysis were used to localize the license plate, isolate the characters, and segment them, respectively. The suggested model had a 94.6% accuracy rate for CNN recognition and 74.7% accuracy for preprocessing license plates.

Additionally, Diogo et al.[31], in their research used a convolutional neural network (CNN) system, which is appropriate for embedded systems and can be used to construct a system that can detect and identify Brazilian license plates. While only needing a single forward pass for every single network, the suggested design showed resistance to variations in angle, lightning, and noise. This allowed for quicker processing. The recognition rate was increased to 99.53%, while the average processing time per image increased to 4.88 s after an ensemble of CNN models was evaluated to increase recognition accuracy.

Then working with image capture plates Sandeep et al.[21], constructed a dataset in order to mimic a captured license plate image. The characters were divided into separate segments on the plate and separated on one another from using a variety of binarization approaches. Using this dataset, a convolutional neural network was trained to recognize the characters on license plates.

Focusing on traffic and security requires license plate identification. Access control helps institutions. Yet, hiring security staff to manually restrict vehicle admission at an academic institution is costly, time-consuming, and inaccurate. This study implemented an automated car tracking system with real-time license plate identification utilizing experimental computer vision technologies to regulate vehicle access and improve academic institution security[22]. Vehicle monitoring utilized technology and camera angles. Environmental impacts on OCR accuracy were explored. Design Science Research vehicle monitoring framework artifact. We used the best image enhancement methods. 96% met the vehicle monitoring framework's optimal operating requirements. The findings imply that academic institutions' camera infrastructure, license plate recognition software, algorithms, and camera orientation

may provide a cost-effective solution.[22] .

For license plate detection, AI and computer vision play a vital role. AI and computer vision revolutionized Automatic License Plate Recognition (ALPR), a key transport system application. ALPR cannot directly use IOT due to hardware limits. Tham et al.[55], suggested a lightweight IOT solution with six phases utilizing deep learning. Yolo V4 was used to train the car plate, OpenVino was used to optimize the drilled model in a shorter amount of time, centroid tracking and geofencing were used, multiple images of the same car plate were used, each image's characters were segmented before being fed to the Tesseract optical character recognition (OCR) by OpenCV, a weighted selection algorithm was used to choose the best car plate from a pool of samples, and the solution was deployed in it.

Nevertheless, the paper struggles with OCR accuracy.

For certain applications, automatic license plate recognition (ALPR) may be a challenging undertaking, and the majority of ALPR systems are focused on a single configuration, such as troll control or a single license plate in a particular area (like Europe,Brazil, or the US). Hence, Silva et al.[54], suggest a comprehensive ALPR that focuses on unrestricted catch scenarios where the LP may be significantly altered due to sideways perspectives. The four corners of an LP may be found using an Improved Warped Planar Object Detection Network (IWPOD-NET) under a number of circumstances. comparing two alternative object-based optical character recognition (OCR) techniques using the rectified LP. Criticism about the paper: achieve satisfied outcomes in certain datasets that include a vast number of capture circumstances and vehicle types.

For a license plate identification system, a brand-new algorithm by Vukovic[15], proposes an inventive technique for dimensionality reduction and feature extraction based on compressive sensing methods. Reduction of dimensionality will be able to produce accurate orders with less preparation data while requiring fewer processing resources for the extraction of features, character recognition, and classification using Support Vector Machine (SVM). However, the paper is too complex to implement in real life and the paper may not differentiate the suggested approach from the state-of-the-art, which can create a lack of comparison.

Next is ALPR, which is crucial in intelligent transportation and surveillance systems. This paper provides a hierarchical convolutional neural network-based end-to-end ALPR technique (CNN). The suggested technique uses two CNN passes to verify the vehicle and license plate area, then a second CNN to recognize the characters. The recognition network was extensively used to deal with limited training data sets. Used both synthetic and augmented data, and our findings show that augmentation works. Considerably boosts recognition rates. They also provide temporal coherence to stabilize video OCR output. Our solution outperformed academic and commercial methods using publicly accessible Brazilian and European license plate databases. Silva et al.[40].

In the meantime, introducing deep learning algorithms can separate more perplexing

highlights, subsequently extraordinarily working on identification and acknowledgment exactness[43], and the strengths and weaknesses of deep learning techniques were discussed[30]. Distinguishing proof of vehicle tags has become more viable in the last 10 years in numerous applications. For example, stockpiling and recovery of vehicular inflow records, programmed cost assortment, leaving expense installment, traffic observation, following of moving vehicles, recuperation of taken vehicles, and so on.[43], focuses on convolutional neural networks (CNN) for license plate recognition system implementation. Additionally, areas for additional improvement in a portion of the chosen CNN-based procedures were recommended. Profound learning calculations can separate further elements, extraordinarily working on identification and acknowledgment exactness. Tag acknowledgment frameworks are generally utilized in current shrewd urban communities. For example, cost installment frameworks, stopping charge installment frameworks, and private access control. Such electronic frameworks are helpful for individuals' regular routines, yet they also offer protected and effective types of assistance for administrators. The conventional area acknowledgment calculation is effectively impacted by light, shadow, foundation intricacy or different variables, resulting in the inability to meet the utilization of genuine scenes. With the improvement of profound learning, the tag acknowledgment calculation can remove further highlights, subsequently enormously working on the discovery and acknowledgment precision.

Both paper [30], [43] need Improvement of CNN dependent techniques and better data set implementation are needed.

For the one-stage system, Seungju et al.[60], utilized the identification techniques of Yolo V3, empowering continuous automobile type and tag object acknowledgment with one RGB camera. In the current framework, the number plate region extraction through picture handling and the person acknowledgment strategy utilizing DNN were utilized. These frameworks have the issue of declining acknowledgement rates as the climate changes. Subsequently, the Yolo v3 framework utilized the one-stage object location strategy, Consequences be damned v3, zeroing in on constant discovery and diminishing exactness because of natural changes, empowering continuous automobile type and tag object acknowledgment with one RGB camera. Preparing information comprises genuine information for vehicle type acknowledgment and tag region location, and manufactured information for tag character acknowledgment. The exactness of every module was 96.39% for location of vehicle models, 99.94% for discovery of tags, and 79.06% for acknowledgment of tags. Likewise, exactness was estimated utilizing Consequences Be Damned v3 Small, a lightweight organization of Just Go for YOLOv3.

In the case of detecting license plate detection and recognition in digital, Danilenko et al.[26], proposed a comparative analysis of the detection methods, which was carried out and led to the detecting module's choice of a convolutional neural network. The neural network GoogleNet served as the foundation for the specially designed network architecture. A summary of methods for detecting and identifying license plates in digital photos was developed. The recognition module's ideal solution was found. Latin alphabetic letters and Arabic numerals are both recognized by the system for license plate identification and recognition. The license plate perception

and identification system underwent testing. It demonstrated the system’s dependability and capacity to function successfully even under circumstances that would prohibit the detection and identification of license plates. The system that was put into place has a minimum accuracy of 94%.

Identification concerns still remain whenever a double vehicle line license plate exists. Qin et al.[36], developed a unified deep learning method for this issue. Most systems recognize single-line license plates. Recognition and division are normally used to discover an object in double-line circumstances. These environmentally sensitive methods often distribute errors between segmentation and identification. The writer suggested a joined technique to identify single- and double-line license plates end-to-end without line or character segmentation. Better, lightweight convolutional neural networks extract features. Multi-task learning techniques are used to categorize and characterize license plates. Last but not least, recognition tasks are viewed as sequence labeling issues that may be immediately resolved by connectionist temporal categorization.

Chen et al.[24], give an approach that integrates a fully convolutional network with a massive learning structure for license plate detection, which differs from others. Combining multi-scale and hierarchical information, the fully convolutional network can recognize randomly positioned objects. It uses pixel-level, two-class categorization. A trained AdaBoost cascade classifier finds a key character that represents an administrative region during character segmentation. A symmetric area horizontal projection approach calculates license plate slant angles. For object detection, a large learning system with a stacked auto-encoder of mapped feature nodes is proposed, and two architectures for letters and digits are examined. The proposed strategy outperforms cutting-edge methods in Macau license plate studies. Using the method in different countries helps predict compatibility and generality.

Conclusively, Saidrasul et al.[42] created a deep learning application to identify license plates from Korean cars in photos. achieved accuracy in detection and recognition of 99.8% and 95.7%, respectively. Number plate recognition with mixed neural networks using scene text recognition and geometric image transformation (GIT). From the above[24], [26], [36], [42], [43], [60] papers, one common issue is the lack of proper dataset implementation. Whatever the algorithms used, they are always stuck in the dataset’s implementation.

2.3 U-Net

U-Net is an images segmentation architecture with encoder-decoder approach. Since its work with irregular images, it will be favourable for car plate irregular images also.

Pinaki et al.[25], presented a novel method for enhancing low-light license plate pictures to allow nighttime license plate recognition systems. Because of low light impact, U-Net collects context and symmetric license plate pixels to distinguish them from background pixels independent of content. This enhances images. Text detection is used to verify the improvement's impacts and the recommended system. According to experimental results on their newly constructed data set, which includes night, low light, and limited light images, and the benchmark data set, UCSD, which includes very poor quality and high quality daytime images, the proposed method outperforms the existing methods. Text detection results show the upgrade's efficacy and dependability.

Next, Ruben's et al.[64], fast-moving autonomous vehicle monitoring system uses a roadside surveillance camera. Thus, a single-pass deep neural network that finds the license plate and recognises characters has been proposed. Unlike previous systems, theirs utilizes a single network to fulfill both goals. Processing is accelerated and intermediate errors are eliminated. They capture the footage to identify the car at each frame and read the vehicle's plate. Red license plates are remembered by their personalities. Fictional deep literacy models are trained using Image AI.

Additionally, for identification strategy Hao et al.[66], present an end-to-end license plate to improve location and recognition in demanding conditions. The cyclic generative adversarial network synthesizes the approximate authentic license plate image to enlarge the training set and alleviate data imbalance. This will simplify model training. The U-Net model uses the upgraded VGG structure and feature pyramid to improve feature extraction. MF-RepUnet successfully detects slanted and small-scale license plates. Finally, attention mechanism weighting improves the convolutional recurrent neural network's feature sequence prediction. This enhances license plate character identification and fixes picture degradation-induced hazy semantic structure sequence characteristics.

From the paper [4,5,6,7] find that U-net basically work with irregular and imperfect images for finding the best of it .

2.4 Federated Learning (FL)

Federated Learning is an edge-assisted intelligent traffic system, Zeng et al.[72], introduced FedLVR, a fine grained vehicle identification which is able to maintain privacy by using a lightweight federated learning-based framework. Their model is able to boost the feature discriminative capability of ShuffleNet V2 by inventing parallel dual attention processes and also introducing a better contrastive center loss and also introduced a weighter federated learning which boosts the model capacity for generalization. The results they find were carried on the Stanford Cars data set and the Compcars data set which were shown to be excellent accuracy and also decreases network model complexity and privacy breach risk.

Also to address privacy concerns and model deployment issues, Xong et al.[48], suggested a federated learning-based license plate recognition system (FedLPR). Here the users are able to train their own data in their own mobile devices instead of having a server to train the data. According to their research paper their model has been able to grant great accuracy, a sustainable communication cost and also is able to maintain the user's privacy.

On the other hand Renyou et al.[71] introduced a federated learning system in this study to concurrently identify numerous license plates through several network cameras. Here the semantic segmentation model is used in order to locally extract the most important aspect of an image which has multiple licenses for achieving a high efficiency of multiple license plate identification in real time. In order to translate the relevant information and complete the semantic encoding a semantic auto encoder is created. A multi-task learning technique for multiple license plate identification uses a multi-objective optimisation strategy to train the model with better generalization. Experimental tests on the Chinese City Parking Dataset (CCPD) show that the recommended method improves identification performance while retaining service quality.

Federated learning is[39] used for medical science and has hardly two or three papers. Moreover, U-Net is an image segmentation process using an encode-decode architecture[39], that is very efficient for medical science. We try to merge the both and propose algorithm named FL U-Net for car plate detection. According to our knowledge this can be the first paper where Federated Learning and U-Net are put together for car plate detection so far. So, expect that the issue of lack of implementation in any rough situations can be solved using federated learning with U-Net segmentation..

In the below have shown figure[2.1], the literature related paper that find about vehicle number plate detection.

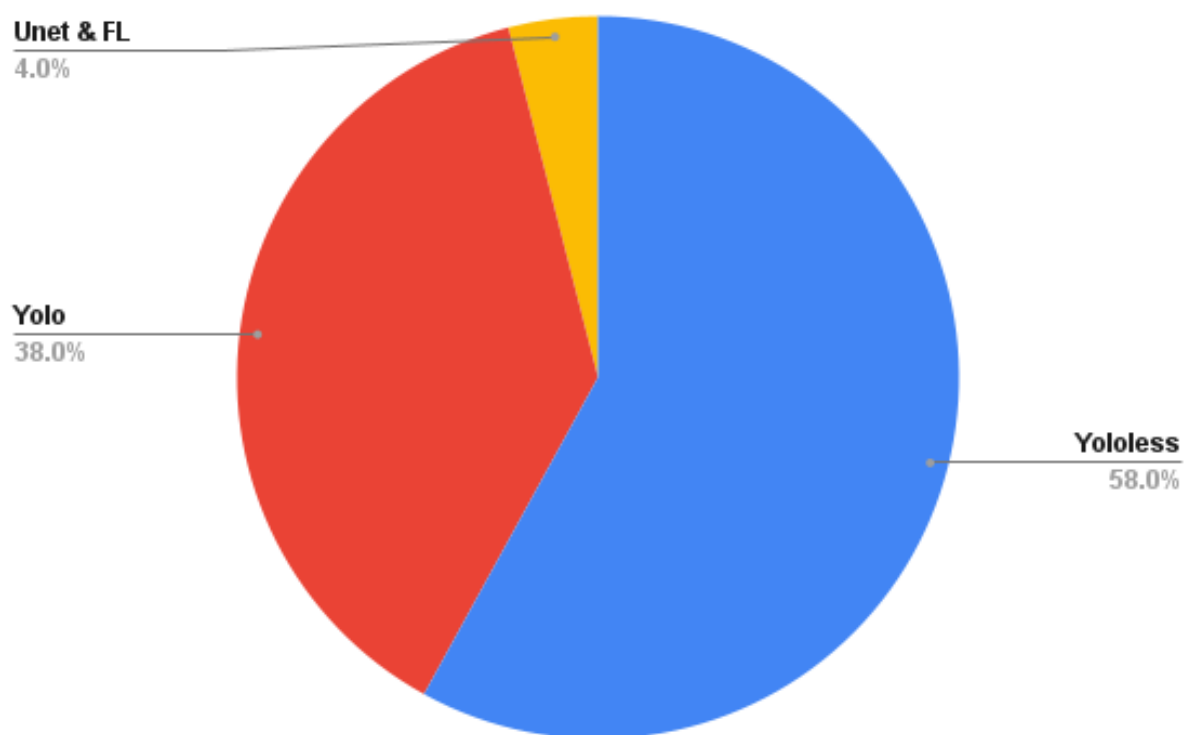


Figure 2.1: Graphical representation of literature review statistics

Chapter 3

Research Objectives

Our purpose is to build a model for license plate detection using a combination of U-Net and Federated learning. We utilize U-Net as it is more robust and has the capacity to surmount the irregular concealment of license plates in undesirable situations. Federated learning will also be implemented to anticipated its ability to maintain the privacy of user data, as it can operate in a decentralized manner that does not compromise the user's essential data, which could result in the loss of privacy for the individuals involved. And ultimately, we will extract the images of the license plates using OCR. At the end of this research, we tend to:

- We will construct a dataset containing a huge amount of data which will be used for our model. We will also label the data.
- Manually label the irregular masking.
- We will develop the U-Net model and also train and test the model.
- We will develop the U-Net with Federated learning for concerns about privacy and also decrease network consumption.
- We will use OCR for image extraction.
- We will evaluate and compare our model to other discussed models which are YOLO, general U-Net and also our Federated learning based U-Net.

3.1 Workflow

In segmented dataset A figure[4.4], there are approximately 433 images. Also, in segmented dataset B figure[4.5], there are approximately 707 images, in segmented dataset C figure[4.6], which was divided into two sections, the first section contained 1928 images and the second section contained 2662 images, and in dataset D figure[4.7], which we collected exclusively, there were 373 still images. Additionally, extracted 267 after cleaning the frames which retrieved from real-time videos.

Our proposed approach figure[3.1] for the Federated U-Net model in a sequential manner. Initially, we labeled the data by manually masking the irregular images. Then the data was then introduced as input into the U-Net segmentation in order to train the model and also test it for data segmentation. Afterwards we integrated Federated learning to our U-Net model for ensuring less consumption of the network and also to prevent the breach of privacy. Afterwards we extracted the license plate inscriptions from the images. Finally, we compared our Federated U-Net model to other models like YOLO and also to standard U-Net.

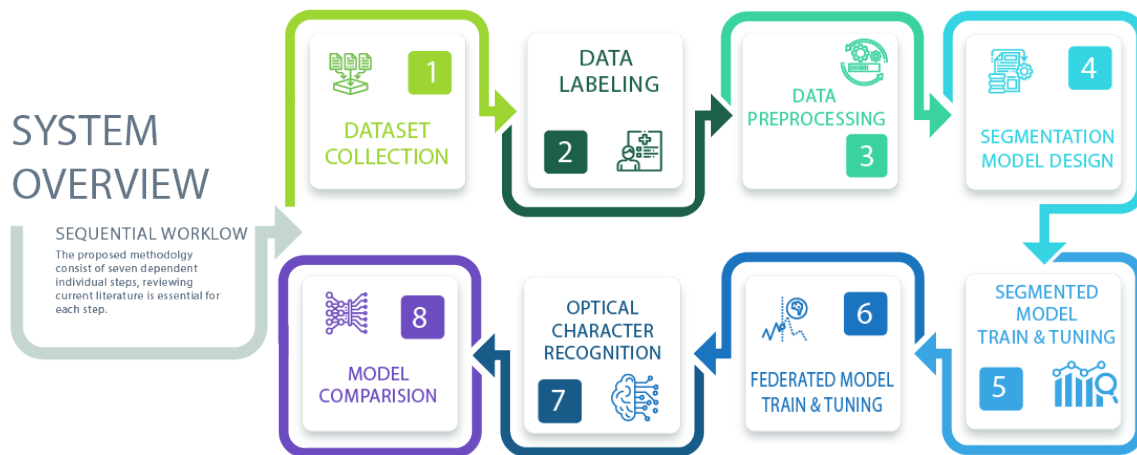


Figure 3.1: System overview of proposed approach Federated U-Net model

Chapter 4

Dataset

4.1 Data Collection

The data collection strategy in this article consists of two general approaches. Firstly, to collect available data and datasets from public online repositories. The other one, is to create a dataset with proper consent from the original car owner. There remain a very limited number of datasets that can be accessed and collected online. The dataset needed to comprise a large number of images, so we captured our data with the proper consent from the respective owners in order to allow us to use them for our research purposes. In addition to that, not to use them publicly for privacy concerns. Even in the case of our own model, in order to thoroughly test and train the model we have proposed, we needed an enormous amount of data. This allowed us to get a better result and understand the benefits and limitations of the model. Due to the fact that our proposed model required a larger quantity of data, that we needed more data for our model. In addition, we had to gather information from additional sources to make up for the data we lacked. To properly assess the effectiveness of our model, the photos were captured under a variety of challenging environmental circumstances and at various times of the day.

Subsequently, data collection was done in the surrounding areas of our educational institution. Due to the possibility of privacy invasion, many people did not give proper authorization for taking images, which made data collection challenging. On the other hand, only a small number of people complied with the request for image collection, which involved taking pictures from various angles, particularly from inappropriate angles, in order to create an unfavorable scenario where the license plates weren't properly displayed in order to test the limits of our model.

Without getting proper consent, we are not able to collect images of licenses; otherwise, it will be a breach of privacy. So for this, we had to get proper permission from the owners of the vehicles to use their license plates. Many were skeptical, but we assured them this was used for research purposes only.

We can observe figure[4.1], that about 65% percent is cars, 28% percent is motor bike and the rest 17% includes auto rickshaws, trucks, mini-vans, cover-vans and other motor vehicles that have number plates.

Vehicle Type	Percentage
Car	65%
Bike	28%
Others	17%

Table 4.1: Vehicle types and percentages

4.2 Data Labelling

In particular for supervised learning, where both input and output data are categorized and labeled to serve as a learning base for additional data processing, data labeling is a crucial component in preprocessing data for machine learning.[5]. Irregular masking has been an issue when it comes to models like Yolo (You Only Look Once). When the license plate is not properly aligned, the bounding boxes are unable to mask the license plate. In order to combat that, we have introduced a segmentation images, by manually masking the images with license plates with adobe photoshop with instruction of our instructors along with the consent of all our group members. The masking was done in order to ensure the labeling was done properly and without any kind of inaccuracy. This will help the model learn the basis of proper labeling and also ensure that, in different conditions, it can distinguish the license plate.

In order to combat that, we have introduced a segmentation model by manually masking the images with license plates with Adobe Photoshop figure[4.1]. The masking was done in order to ensure the labeling was done properly and without any kind of mistake and also for our model to learn the basis of proper labeling and also ensure that, in different conditions, it can distinguish the license plate.

By the proper consent of everyone involved and also by adhering to the instructions of our instructor Rafeed Rahman, we used Adobe Photoshop where we masked the license plate in white color and the rest of the image in black. In Adobe photoshop we used two layer for masking first was used to color the LP in white and the second layer was to color the rest of the image which excluded the LP in black. This was done in order to ensure there was no presence of irregular masking.

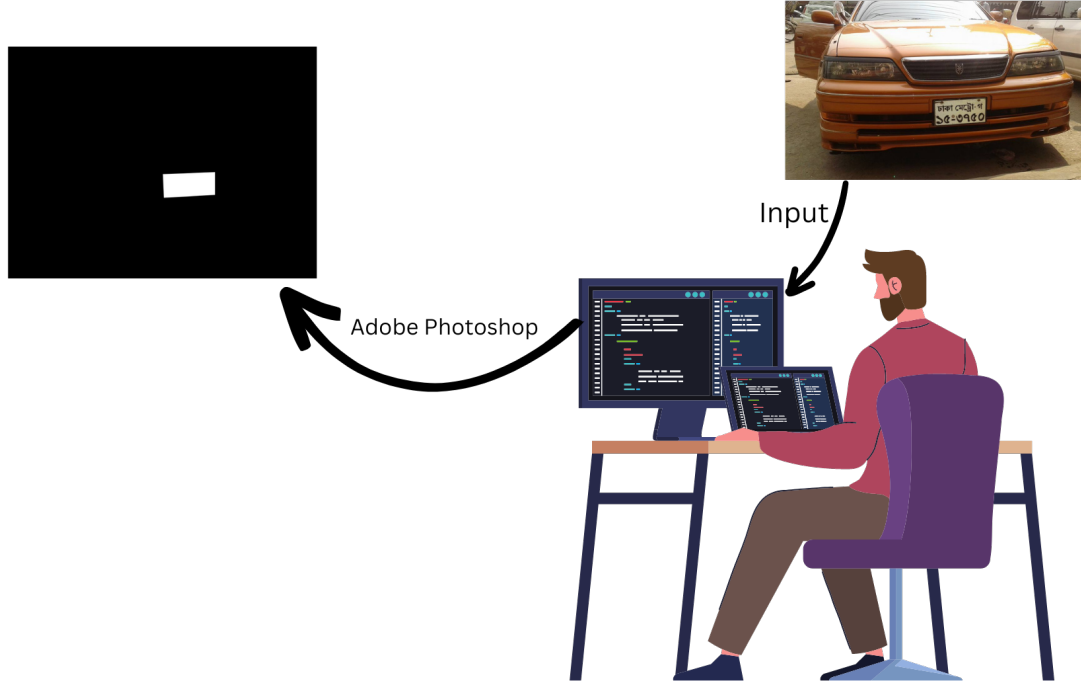


Figure 4.1: Process of data labelling

4.3 Augmentation

To get the best performance from our proposed model, we needed a massive amount of data, as deep learning models perform well when there is a sufficient amount of data. We were able to collect only 6370 images, which could not be beneficial for our model. Moreover, we only collected 373 images and 3 real-time videos from which we get 267, which were taken at different times of day and night. Furthermore, we collected data from different sources, and the data that we collected was not sufficient for us. Due to a lack of data, our proposed model might not be capable of recognizing patterns in the data, so it might not be able to give a commendable result. To address the problem, we all decided to increment the data using the popular approach of data augmentation.

Data augmentation is done in order to improve the model accuracy, generalization and also to control over fitting[4]. Figure[4.2], indicate that 6370 data we had before augmentation and after augmentation, we got 11,472 data that we used to train our model. Among different manner of augmentation, we implemented distinct features for data augmentation which are feature-wise center, feature-wise standard normalization, width shift range, height shift range and also rescale and resize data [34].

Feature-wise center, establish the feature-wise input mean for the data set. Here, the word means refers to the proportion of total values to entire values in a dataset. **Feature-wise standard normalization** uses the dataset's standard deviation to

generate the feature-wise input.

Width shift range, represented as a float ratio of width.

Height shift range, represented as a float ratio of height.

Rescale and resize data provide a compact input size, thereby reducing the amount of computation, memory, and time. Prior to actually attempting a re-size, a rescale is carried out. To rescale the image, divide it by $1/255$. The image is then resized to 160 x 160 pixels.

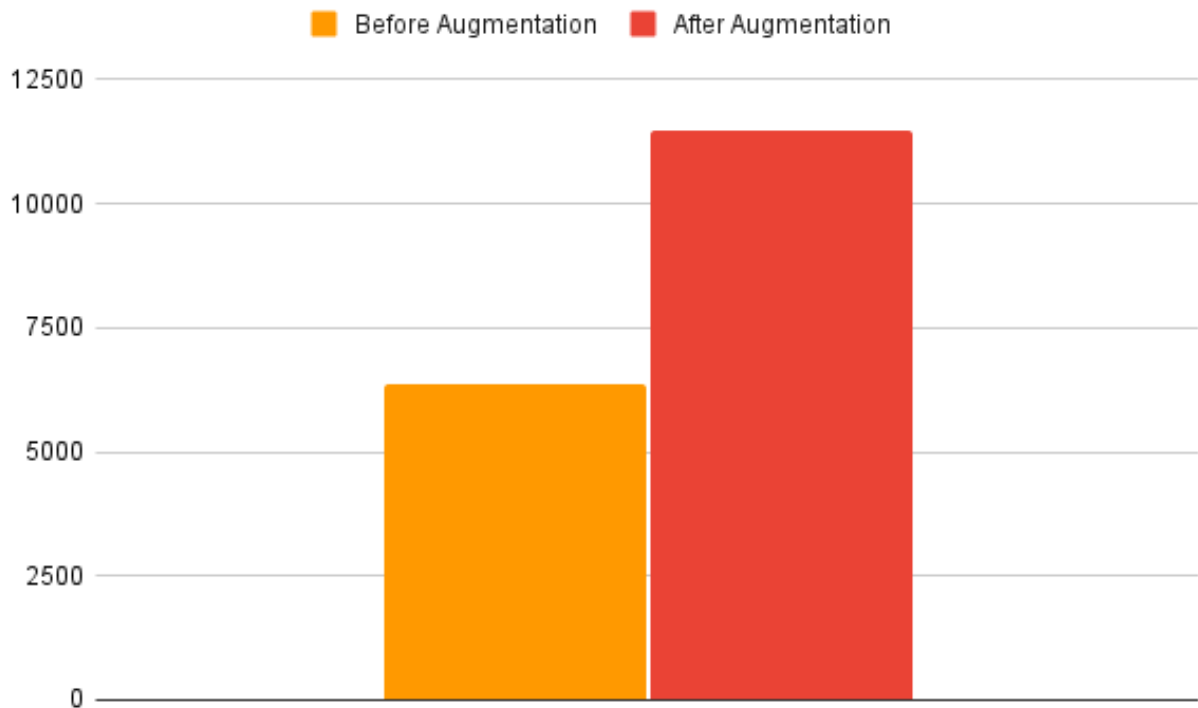


Figure 4.2: Representation augmentation data

4.4 Dataset Details

4.4.1 Data Description

The datasets we collected and gathered for our model included vehicle images. The collected segmented datasets A[74] and B [76], contain vehicle images from different countries in Europe and America. Since, the vehicle originates from different countries, the license plate has a variation in different languages. On the other hand, the segmented dataset C[73] and our segmented dataset D figure[4.3] contain only Bangladeshi vehicles images.

The segmented datasets have images of different vehicles all over the world. The vehicles that have number plates were taken into account. Images of vehicles include bike, car, truck, bus, van etc. etc. During the collection of datasets, we were able to take images of different types of vehicles. In Dataset A and B there were Nissan sylphy, mercedes benz, Toyota fielder, Nissan X-trails, Nissan urvan, Nissan patrol, Suzuki swift, Suzuki Wagon R, Suzuki Ravi etc. There were pickup trucks like the Tata H pickup, the Toyota Dyna, and the Zotye truck. Again, in the segmented dataset C and D, there are all types of vehicle which are commonly seen in the roads of Bangladesh. For example, models like the Toyota Axio, the X Corolla, the Toyota Allion, and the Toyota Premio. There were also motor bikes in our dataset, like honda, Suzuki GSX-R-150, Bajaj pulsar, Bajaj Discover, Bajaj Platina ES etc. Also there are images of bus. track. Even triple-wheeled vehicles like CNG were also taken into account. And these are all mainly Bangladeshi vehicles images.

Different types of vehicles which have a visible number plate figure[4.3] were taken into account for the sole purpose of keeping track of irregular masking, as the vehicles are in different shapes and sizes and also in different positions. As a result, by using them, we can incorporate them into our model to determine the degree to which the license plate's uneven masking is accurate.



Figure 4.3: Differnt types of vehicle with visible number plate

4.4.2 Segment Dataset A

The images in the "Car Licence Plate Detection" dataset[74] named as Segmented dataset A. Include bounding box annotations of the car plates. In segmented dataset A total 433 vehicle images figure[4.4] existed. We were required to change the annotated data in order to create a segmented model. We also used the bounding box data as a starting point for our model, after which we manually labeled the data to conform to our model. The images in this gallery depicted various scenarios. The images were captured in a variety of lighting conditions, including low light, cloudy, broad daylight, and at night. The dataset also included distorted images, which helped to create a situation in which the model could adapt to images that were not always presented in the most ideal format.



Figure 4.4: Segmented dataset A

4.4.3 Segment Dataset B

The Tunisian dataset, "Labeled license plate"[76], named as segmented dataset B. The dataset contains Tunisian license plates and the coordinates of each license plate in the dataset. We found about figure[4.5] 707 data and among them, 141 were used in testing and 566 were used in training. The annotated files were in XML format. We also had to change the data according to our segmented modeling needs. For this, we also use the typical masking approach we have been using. The images that remain here were also taken keeping in mind the concept that the model can adapt to various situations and also test the accuracy of the model. Therefore, the same image was taken from different angles, even at angles where the license plates would be hard to detect. This would, in turn, help to compare the same license plate detected from different angles and also show the accuracy rates of the model that will be used.



Figure 4.5: Segmented dataset B

4.4.4 Segment Dataset C

Bangla LPDB, named as segmented dataset C [73]. This dataset figure[4.6], is the first such dataset in Bangladesh that includes almost every kind of vehicle seen in Bangladesh as represented in the vehicle picture dataset, with the exclusion of army, police, and government vehicles. The first part of the dataset includes 1928 images of Bangladeshi vehicles' noticeable license plates in Bangla. The second part has 2662 photos of a cropped Bangla license plate. The second part only contained cropped images of the license plates.

These images of the segmented dataset C was crucial, as the model needed to be able to detect the license plate, especially in a country like Bangladesh. This would help the model become familiar with the architecture of the Bangladeshi license plate. Here also, the images were taken in different disadvantageous conditions like heavy rain, nightfall, broad daylight, and even a foggy environment. The images were taken from running vehicles to still vehicles in order to get the perfect dataset in order to implement a proper model in both training and testing to combat the situation of the model not being able to detect license plates in unfavorable conditions.



Figure 4.6: Segmented dataset C

4.4.5 Segment Dataset D

The segmented dataset D figure[4.7] is the dataset we collected exclusively with the proper consent of the people involved, with the condition of not sharing the data publicly and only using it for research purposes. We added a total of 373 still frames and three videos in broad daylight and in night. These images were also taken in a heavy downpour, during daylight hours and in a murky situation. The same images were also taken with a diverse inclination to get a concept of the performance of the model.

From the three videos we extracted still frames by using OpenCv implemented from python. We extracted about 847 frames and after cleaning the still frames by manually removing repetition and also irrelevant images. After cleaning the data, we were able to acquire 267 fill frames.



Figure 4.7: Segmented dataset D

4.4.6 Segmented Dataset Merged

For our proposed model, we have collected segmented datasets A, B, C and we have our own collection which is segmented data D. From dataset A figure[4.4], we get 433 images, from B figure[4.5], get 707 datas and from C figure[4.6], we had total 4590 images and D [4.7], get respectively 640 images. So, we had total consist of total data get 6037 data for our proposed model. We mergerd these four dataset and created a new merged dataset named as segmented datasett merged figure[4.8]. After augmentation figure figure[4.2] the all segmented dataset merged we get total 11,472 images.



Figure 4.8: Segmented dataset merged

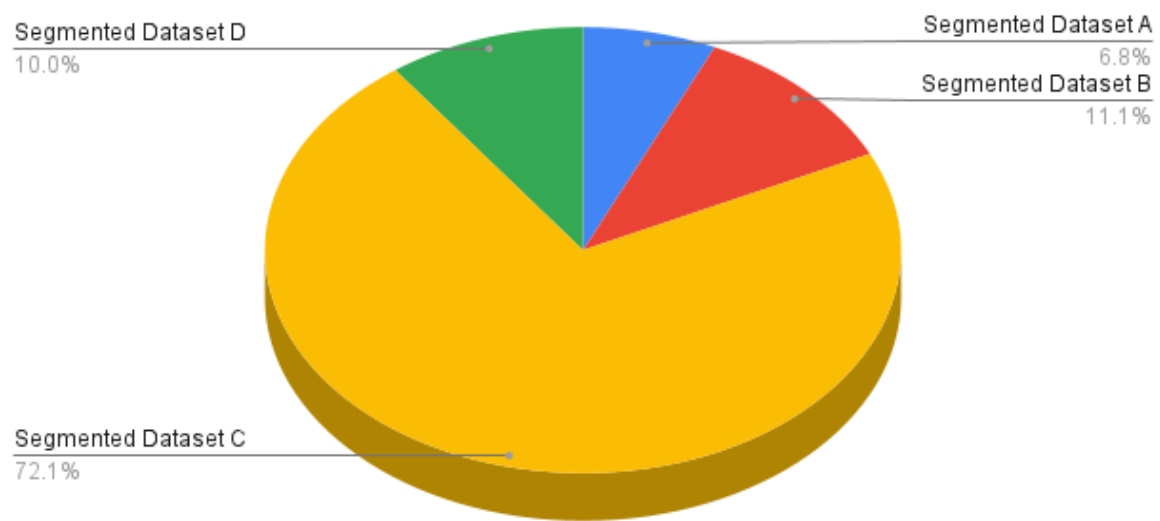


Figure 4.9: Distribution of total dataset

Segmented dataset merge figure[4.9], generated together with segmented dataset A, B, C and D. Where segmented dataset A contributes 6.8%. Between segmented dataset B and C, segmented dataset C has the most contributes of 72.1% and Segmented dataset B contributes 11.1%. Our Segmented dataset D has 10% contribution.

Chapter 5

Methodology

5.1 U-Net

U-Net is an image semantic segmentation architecture[10]. It consists of a path that contracts and one that expands, giving it a U-shaped structure. U-Net is an architecture for semantic segmentation of images. It consists of a contracting path and an expansive path, which gives it the u-shaped architecture figure[5.1]. U-Net has a decreasing route, growing path and pass. Each contracting route convolutional layer adds its own learnable filters to the feature map input. Filters convolve the input feature map with weights to create a set of output feature maps that reflect the image's more abstract features. U-Net max-pooling is a pooling operation that determines the greatest value in each region of each feature map [3]. Despite reducing resolution, max-pooling keeps the map's unique features. A fully linked stratum in the contracting route yields a feature vector for the expanding path. This layer creates a single vector from the contracting path's output feature maps' last convolutional layer[19] .

Transposed convolutional layers make up the enlarged path. These layers compress channels and scale feature maps to picture input size. At each layer, teachable filters train the upsampling algorithm on the supplied feature map. In addition to transposed convolutional layers, the expanding route has skip connections that link the feature maps of the decreasing path to the equivalent layer of the expanded path. The network can learn high-level semantic characteristics while maintaining visual details. The segmentation map, a pixel-by-pixel label map that identifies picture portions, is the last layer in the expanding route. Applying an activation function to the output feature map generates a probability map that assigns probabilities to each pixel in each class [68]. Map segmentation label is the highest likelihood class determines each pixel. Skip connections to connect the sampling track's feature map to the downsampling track's feature map[20]. This enables the model to retain spatial information at a finer resolution.

U-Net uses data augmentation methods including rotation, rotations and elastic deformations to enhance the effective dimension of the training dataset for semantic segmentation models, which need a lot of annotated data[29]. This decreases labeled training data. Picture segmentation divides a picture into smaller pieces. It provides pixel-level information on the picture and object shapes and limits. Im-

age segmentation requires masking[45]. A binary mask has zero and non-zero values.

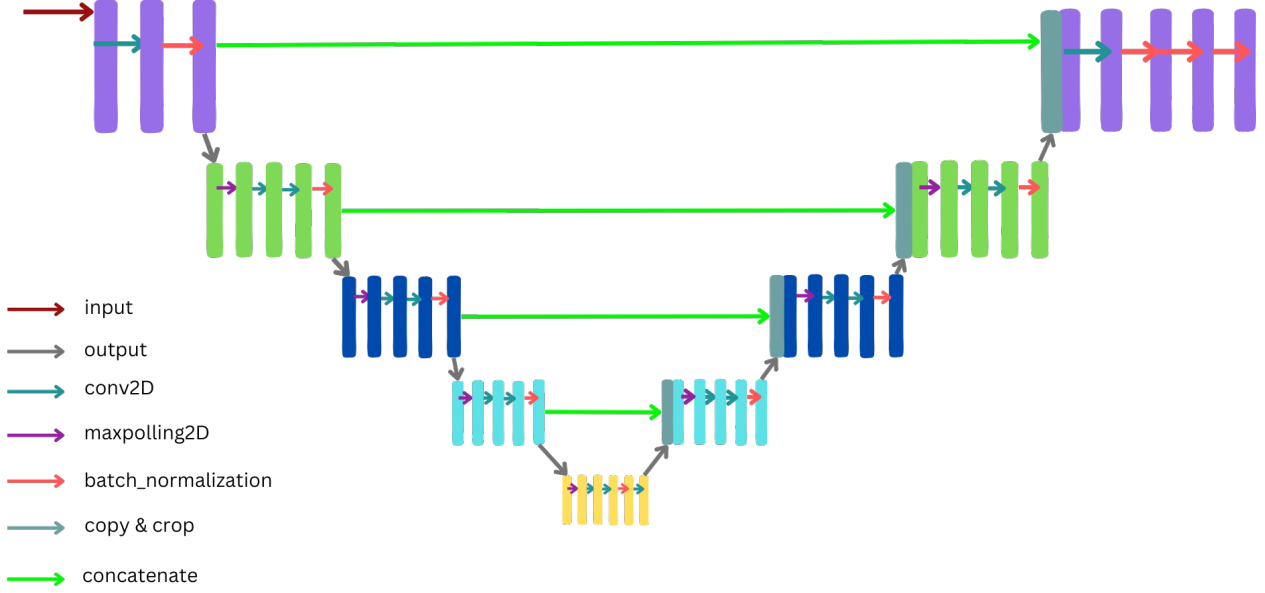


Figure 5.1: U-Net architecture

5.2 Federated Learning (FL)

Federated learning is nowadays used very heavily in machine learning[56]. Traditional model training utilized all data on a single server. Federated learning uses data reduction tactics to assist several entities tackle a machine learning problem. Each entity stores raw data locally and updates a global model with targeted modifications for immediate aggregation. To reduce data exposure, combine user models without preserving the individual models. Secure aggregation and secure enclaves may combine numerous local models without exposing each user's input to the server, providing more dependable guarantees.

Federated learning protects privacy and improves models, latency and power usage. In addition to updating the shared model, devices may utilize the new model immediately[27].

Federated machine learning is decentralized. Machine learning trains on data acquired from edge devices like smartphones, laptops and others and stored on a central server. After training, machine learning algorithms predict outcomes for fresh data.

Users may use applications locally on all devices. Based on human input, the model will learn and train itself, sometimes becoming smarter. The devices return training data from the local machine learning software to the central server. Results are independent of data. Multiple devices with local copies of the app follow this approach. The centralized server will again compile results without user data. The central cloud server uses pooled training data to update the core machine learning model, which is now much more effective. Users update the application with the smarter model built from their own data once the development team changes the model. Only while not in use will gadgets train. While charging, in do not disturb mode, or idle, this might happen.

Federated learning will now encrypt user-sensitive data with a key not held by your cloud server. Zero-sum masking protocols are functional encryptions[46]. Zero-sum masks add to 1 in one direction and 0. The first aggregates and secures encrypted user data while the second decrypts and delivers training results to the server. Repeat until done, then the masks cancel each other out.

5.3 YOLO

You only look once at the model, in short the Yolo model is a widely used model for real time object detection[59]. This algorithm is well-liked due to its speed and precision. In various applications, it has been used to detect traffic signals, pedestrians, parking meters and animals. It is suitable for applications such as video surveillance, autonomous vehicles, and augmented reality[7]. The Yolo algorithm divides an input image into a $S \times S$ grid. If an object's center lies within a grid cell, that cell is responsible for detecting the object. Each grid cell predicts the boundary boxes and the confidence scores for those boxes.

The algorithms operate in two phases. In the first stage, it identifies intriguing regions from the image. Then, these regions are classified using convolutional neural networks. This solution may be very sluggish because prediction must be executed for each selected region. It can identify 20 Pascal object classes, including person, bird, cat, bovine, canine, equine, and livestock an aircraft, a tricycle, a watercraft, a vehicle, a car and a motorcycle. It can predict up to 9000 classes, including previously unseen ones. Multiple objects will be identified in an image by the real-time recognition system, which will also create a boundary frame around each object.

There are a total of 24 layers in Yolo which is followed by 2 fully connected layers and also 4 pooling layers. On the basis of the single-stage detection principle, Yolo integrates all object detection pipeline components into a singular neural network[59]. It utilizes the entire image's features to predict class probabilities and bounding box coordinates. Utilizing the mean average precision (mAP) the accuracy is measured. Comparing the ground-truth bounding box to the detected box, the mAP returns a score. The greater the score, the more precise the model's detections.

5.4 OCR

"OCR," which stands for "Optical Character Recognition" which was invented in the year 1974 for the sake of image character recognition[1]. OCR has been very prominent and also very strong in Japan as the Electronical Laborator (ETL) which is located in Ibaraki has played an important role in the development of OCR. It is a method of technology that can recognise text contained inside a digital picture. Its primary function is to recognise text included inside scanned documents and photos. OCR software has the capability of converting a physical paper document or an image into an electronic format that can be searched for text. Although OCR engines had great contributions in modern times, their performance has not been properly documented in historical texts. Although OCR engines had great contributions in modern times, their performance has not been properly documented in historical texts[51]. One of the greatest advantages of OCR is the generation of less noise. OCR is beneficial for both pattern recognition and feature extraction. There are three steps of OCR :

- Preprocessing
- Character recognition
- Post-processing of output

5.5 Proposed Method

5.5.1 U-Net

Our U-Net model is utilized in the segmentation of images fig[5.2]. Using federated learning, it is intended to detect Bangladeshi license plates in images of cars. The architecture of the model is characterized by an encoder-decoder structure with skip connections between the encoder and decoder sections. After the encoder downsamples the input images into a feature map, the decoder upsamples the feature map into an output segmentation mask.

This U-net is based on the concept of combining encoder and decoder architectures into a single neural network that can learn to precisely divide the input image into distinct regions. Encoder architecture downsampled the input image using convolutional layers with a small kernel size and a large number of filters. Whereas decoder architecture upsamples the encoded features to reconstruct the original image. The proposed model architecture consists of multiple convolutional layers with each convolutional layer learning increasingly intricate image features.

Convolutional filters have kernel dimensions of 3x3, which is a common choice for image segmentation tasks. The amount of coating in the first layer is 32 and this increases to 512 in the deepest layer figure[5.2]. The activation function is used during the ReLU unit since it is known to perform well in deep neural networks. In order to enhance the learning process and protect against overfitting, the model that we have proposed also includes batch normalization layers and dropout layers. Batch normalization layers are added to individual convolutional layers to normalize the resulting features and reduce the internal covariate shift. Dropout layers with a rate of 0.1 to 0.3 are applied after some convolutional layers to prevent overfitting and improve generalization. The encoder architecture of the U-Net model comprises numerous convolutional layers that gradually lower the spatial dimensions of the input picture. The encoder starts with a 2D convolutional layer with 32 filters and a kernel size of 3x3, then adds another layer with 16 filters and the same kernel size. These two layers are then batch normalized and followed by a dropout layer, which has a ratslayer of 0.1. The result of the layer is then sent into a max pooling layer with 2x2 pool dimensions to lower the spatial resolution of the feature maps.

Two convolutional layers, a batch normalization layer, a dropout layer and a max pooling layer make up each of the five blocks of the encoder in our proposed model figure[5.2]. Each subsequent block doubles the amount of cleaner in the first convolutional layer, which ranges from 32 to 64 and the second remains between 16 and 32. In particular, there are 64, 128, 256, and 512 layers in the second, third, and fourth blocks, respectively. On the other hand, in the secondary convolution layer in the second, third, fourth, and fifth blocks (32, 64, 128, and 256 layers, respectively).

The decoder in our suggested model consists of the same four blocks as the encoder. The decoder's layers begin with a concatenation layer, then two transposed convolutional layers with the same number of filters as the first convolutional layer in the encoder block, batch normalization, dropout, and one or more convolutional layers with the same number of filters. The transposed convolutional layers of the first

decoder block include 128 filters; those of the second contain 64; those of the third contain 32; those of the fourth contain 16; and those of the fifth contain 8.

Each block in the encoder and decoder has two convolutional layers, so each block has two sets of weights and, therefore, two sets of neurons. Each group has as many neurons as there are filters in its convolutional layer. Each block of the encoder's first convolutional layer begins with 32 filters and doubles up to 512 filters. Each block's second convolutional layer has half as many filters as its first convolutional layer.

Each block of the decoder's transposed convolutional layers begins with 128 filters and gradually decreases to 8 filters by the last block. There are twice as many filters in the second convolutional layer of each decoder block as there were in the first convolutional layer of the corresponding encoder block.

According to Duan et al.[47], following this architecture for our proposed model can achieve a higher accuracy rate.

In addition to the above, there are also some residual connections in the architecture, such as the one between the concatenation layer and the initial convolutional layer in the subsequent decoder block, which adds the results of the concatenation layer to the first convolutional layer result. These residual connections help the network learn more effectively by allowing gradients to flow more easily through the network. ReLU is used in all the convolutional layers except the last convolutional layer of each block, where no activation function is used.

Besides, in our proposed model, residual connections are added to solve the hurdle of vanishing gradients, which is typical of neural networks with deep layers. Residual connections join the input and output of a layer, which allows the gradients to flow more easily which adds the results of the corresponding encoder layer and the decoder layer, providing the residual connection.

Moreover, in our proposed model, residual connections are added to solve the hurdle of vanishing gradients, which is typical of neural networks with deep layers. Residual connections join the input and output of a layer, which allows the gradients to flow more easily which adds the results of the corresponding encoder layer and the decoder layer, providing the residual connection.

Zhang et al.[6] suggested that the use of residual connections in U-Net model is an effective strategy for ensuring that the model can learn better representations of image features and avoid the vanishing gradient problem. These residual connections enable the information from early layers to be passed directly to later layers, improving the learning capability of the network.

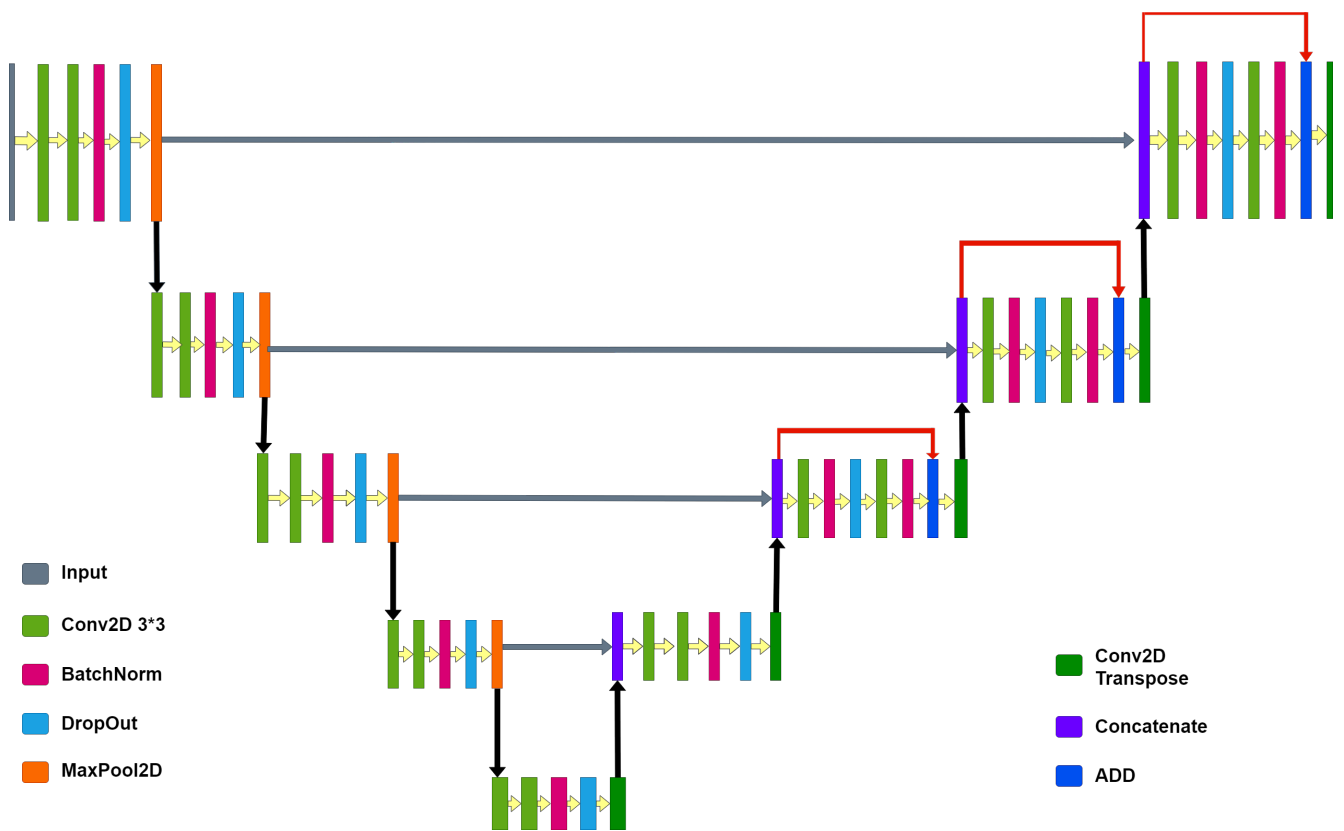


Figure 5.2: Proposed U-Net model

Layer (type)	Output Shape	Parameters
img (InputLayer)	(None, 192, 192, 3)	0
conv2d	(None, 192, 192, 32)	896
conv2d_1	(None, 192, 192, 16)	4624
batch_normalization	(None, 192, 192, 16)	64
dropout	(None, 192, 192, 16)	0
max_pooling2d	(None, 96, 96, 16)	0
conv2d_2	(None, 96, 96, 64)	9280
conv2d_3	(None, 96, 96, 32)	18464
dropout_1	(None, 96, 96, 32)	0
batch_normalization_1	(None, 96, 96, 32)	128
max_pooling2d_1	(None, 48, 48, 32)	0
conv2d_4	(None, 48, 48, 128)	36992
conv2d_5	(None, 48, 48, 64)	73792
batch_normalization_2	(None, 48, 48, 64)	256
dropout_2	(None, 48, 48, 64)	0
max_pooling2d_2	(None, 24, 24, 64)	0
conv2d_6	(None, 24, 24, 256)	147712
conv2d_7	(None, 24, 24, 128)	295040
batch_normalization_3	(None, 24, 24, 128)	512
dropout_3	(None, 24, 24, 128)	0
max_pooling2d_3	(None, 12, 12, 128)	0
conv2d_8	(None, 12, 12, 512)	590336
conv2d_9	(None, 12, 12, 256)	1179904
batch_normalization_4	(None, 12, 12, 256)	1024
dropout_4	(None, 12, 12, 256)	0

Table 5.1: Layer parameters of encoder

Layer Name	Output Shape	Number of Parameters
conv2d_transpose	(None, 24, 24, 128)	131,200
concatenate	(None, 24, 24, 256)	0
conv2d_10	(None, 24, 24, 256)	590,080
conv2d_11	(None, 24, 24, 256)	590,080
batch_normalization_5	(None, 24, 24, 256)	1,024
dropout_5	(None, 24, 24, 256)	0
conv2d_transpose_1	(None, 48, 48, 64)	65,600
concatenate_1	(None, 48, 48, 128)	0
conv2d_12	(None, 48, 48, 128)	147,584
batch_normalization_6	(None, 48, 48, 128)	512
dropout_6	(None, 48, 48, 128)	0
conv2d_13	(None, 48, 48, 128)	147,584
batch_normalization_7	(None, 48, 48, 128)	512
add_1	(None, 48, 48, 128)	0
conv2d_transpose_2	(None, 96, 96, 32)	16,416
concatenate_2	(None, 96, 96, 64)	0
conv2d_14	(None, 96, 96, 64)	36,928
batch_normalization_8	(None, 96, 96, 64)	256
dropout_7	(None, 96, 96, 64)	0
conv2d_15	(None, 96, 96, 64)	36,928
batch_normalization_9	(None, 96, 96, 64)	256
add_2	(None, 96, 96, 64)	0
conv2d_transpose_3	(None, 192, 192, 16)	4,112
concatenate_3	(None, 192, 192, 32)	0
conv2d_16	(None, 192, 192, 32)	9,248
batch_normalization_10	(None, 192, 192, 32)	128
dropout_8	(None, 192, 192, 32)	0
conv2d_17	(None, 192, 192, 32)	9,248
batch_normalization_11	(None, 192, 192, 32)	128
add_3	(None, 192, 192, 32)	0
conv2d_18	(None, 192, 192, 3)	99

Table 5.2: Layer parameter of deccoder

Parameter Type	Number of Parameters
Total Params	4,146,947
Trainable Params	4,144,547
Non-trainable Params	2,400

Table 5.3: Parameter summary

In the encoder table[5.1], at the outset, the input layer receives input tensors of size (None, 192, 192, 3), where None represents the batch size, 192x192 denotes the dimensions of the input image and 3 signifies the three RGB color channels. This input layer serves as the starting point for the U-Net encoder. The model then proceeds with a series of convolutional layers. The first Conv2D layer applies a 2D convolution operation with 32 filters, using a kernel size of 3x3 and a stride of 1. This operation results in an output tensor of dimensions (None, 192, 192, 32), where the number of filters corresponds to the number of learned features in the image. Subsequently, Conv2D 1 further convolves the feature maps with 16 filters, employing the same kernel size and stride as before. This generates an output tensor of size (None, 192, 192, 16). Following this convolutional operation, a Batch normalization layer is applied to normalize the activations along the batch axis, enhancing the stability of the training process. The output shape remains unchanged at (None, 192, 192, 16). To prevent overfitting and improve generalization, a Dropout layer is introduced, randomly setting a fraction of the input units to zero during each training update. Next, a MaxPooling2D layer is employed to downsample the spatial dimensions of the feature maps. It performs max pooling with a pool size of 2x2 and a stride of 2, resulting in halved dimensions. Consequently, the output shape becomes (None, 96, 96, 16), effectively reducing the computational complexity while retaining relevant information. The encoder then proceeds with Conv2D 2 and Conv2D 3, employing increasing numbers of filters (64 and 32, respectively) to extract higher-level features from the downsampled feature maps. Each convolutional layer is followed by Dropout and BatchNormalization layers, maintaining the structure and enhancing the model’s performance. Subsequently, MaxPooling2D 1 is applied to downsample the feature maps further, resulting in (None, 48, 48, 32). Continuing the process, Conv2D 4 and Conv2D 5 are introduced, utilizing more filters (128 and 64, respectively) to extract even more complex features from the downsampled feature maps. BatchNormalization 2 further normalizes the activations, and Dropout 2 aids in regularizing the model. MaxPooling2D 2 is then applied, downsampling the feature maps to (None, 24, 24, 64). This downsampling operation continues to capture high-level information while gradually reducing the spatial dimensions. The U-Net encoder concludes table[], with Conv2D 6 and Conv2D 7, employing a larger number of filters (256 and 128, respectively) to capture the most intricate features from the downsampled maps. BatchNormalization 3 and Dropout 3 continue to normalize and regularize the model, respectively. Lastly, MaxPooling2D 3 downsamples the feature maps to (None, 12, 12, 128), further reducing the spatial dimensions and extracting the highest-level features.

The subsequent layers in the decoder table[5.2], including Conv2D Transpose, Concatenate, Conv2D, Batch Normalization, Dropout, Conv2D, Batch Normalization, and Add, follow a similar pattern as described above. Each layer plays a specific role in further upsampling the feature maps and refining the segmentation output. The decoder starts with a Conv2D Transpose layer. It takes an input with a shape of (None, 24, 24, 128) and performs transposed convolution to produce an output with a shape of (None, 48, 48, 64). This upsampling operation expands the spatial dimensions of the feature maps while reducing their depth. The layer is equipped with 131,200 parameters, which are learnable weights that are adjusted during the training process. Following the upsampling operation, a Concatenate layer is em-

ployed to merge the upsampled feature maps with the corresponding feature maps from the encoder portion of the U-Net. In this case, it takes inputs with shapes (None, 24, 24, 256) and (None, 48, 48, 64) and generates an output with a shape of (None, 48, 48, 128). Subsequently, two Conv2D layers are employed to process the concatenated feature maps. These convolutional layers perform spatial filtering operations to extract higher-level features from the input data. Both Conv2D layers have the same input shape of (None, 48, 48, 128) and output shape of (None, 48, 48, 256). The number of parameters in each Conv2D layer is 590,080, indicating a significant amount of learnable weights that enable the model to capture complex patterns and details in the feature maps. Batch Normalization layers follow the Conv2D layers. The Batch Normalization layer in this model has an input shape of (None, 48, 48, 256) and the same output shape. It introduces 1,024 parameters, which are used to estimate the mean and variance of the inputs. Here, the dropout randomly sets a fraction of the input units to zero during training, reducing the reliance on specific features and promoting robustness. The Dropout layers in this decoder have an input and output shape of (None, 48, 48, 256) and do not contribute any additional parameters. Finally, the model concludes with a Conv2D layer

The table[5.3], provided offers insights into the architecture of a U-Net model encoder. The model encompasses a total of 4,146,947 parameters, which serve as the tunable weights and biases necessary for the network to learn and make predictions. These parameters undergo optimization during the training process, allowing the model to adapt and improve its performance over time. Among the total parameters, 4,144,547 are trainable, meaning they are updated and refined during training to minimize the model's loss function. This significant number of trainable parameters implies that the model possesses the capacity to comprehend intricate patterns and intricate features within the input data, which is of paramount importance for achieving precise image segmentation. Additionally, the model comprises 2,400 non-trainable parameters.

In summary, the U-Net model encoder presented in the table features a total of 4,146,947 parameters, with 4,144,547 of them being trainable and 2,400 non-trainable. With its ability to capture local and global context, this model exhibits a high capacity to accurately segment images. The large number of trainable parameters signifies the model's capability to comprehend complex patterns and features within the input data, contributing to its effectiveness in image segmentation tasks.

5.5.2 Federated Learning(FL)

This model implements Federated learning along with U-Net segmentation, in which the FL model training is decentralized in which the data remains in the devices of the clients. Our U-Net and Federated learning based model takes the advantage of the privacy maintaining factor of FL and along with the task of performing license plate detection of vehicle image dataset. We introduced a dataset where pre processing was done in order to prepare the model for FL. Here we are trying to achieve the aim where we can distribute the training data among the clients while also maintaining the privacy of the users.

First, by distributing training data across multiple clients we divide the training data into separate sets and the sets represent a client. In order to start the data preparation equation[5.1], the total number of images in the training set is divided equally across the four clients.

$$DataPartition = \frac{\text{Total number of images}}{\text{number of client}} \quad (5.1)$$

Data preprocessing was done where the reshaping the pixel data and labels, repeating the dataset for multiple epochs, shuffling the data using a specified buffer size, batching the data according to a predefined batch size and mapping the data to a suitable format. Afterwards we created federated data from the client-specific data and associated clients, which supports the preprocessing steps and also generates a list of preprocessed datasets for each client.

In this model figure[5.3], four clients were used where we equally distributed datasets for training and validation purposes and used the same proposed U-Net model among the clients. The client then conducts local training using the initial model and subsequently, the clients calculate updates or gradients based on their locally trained model figure[5.3]. In order to safeguard data privacy we employed secure aggregation techniques for transmitting the updates to the central server without compromising the raw data and after aggregation this updated model is then distributed back to the clients in order to conduct further iterations.

Data Aggregation: The aggregation of model updates from individual clients is performed by the aggregator figure[5.3]. During each training round, the clients perform local training using their respective datasets. After local training, the clients generate model updates based on their trained models. These updates typically consist of model parameters or gradients. The clients then send their individual model updates to the aggregator. The aggregator receives the model updates from all participating clients and computes their mean. There are three types of aggregators we used in our model. These are:

Mean Factory: The mean factory determines the mean of the model updates received from the clients.

Private Quantile Estimation Process: Here the private quantile estimation is performed, which enables the clients to share their model updates and also preserving privacy.

Differentially Private Factory: This aggregator applies differential privacy techniques to ensure the privacy of the model updates.

Data Distribution: The distributor distributes the updated global model to the clients after the aggregator computes the global model figure[5.3]. By combining the aggregator and distributor components the FL system is able to lump the model updates from each of the clients, calculate the global model and return them back to the clients.

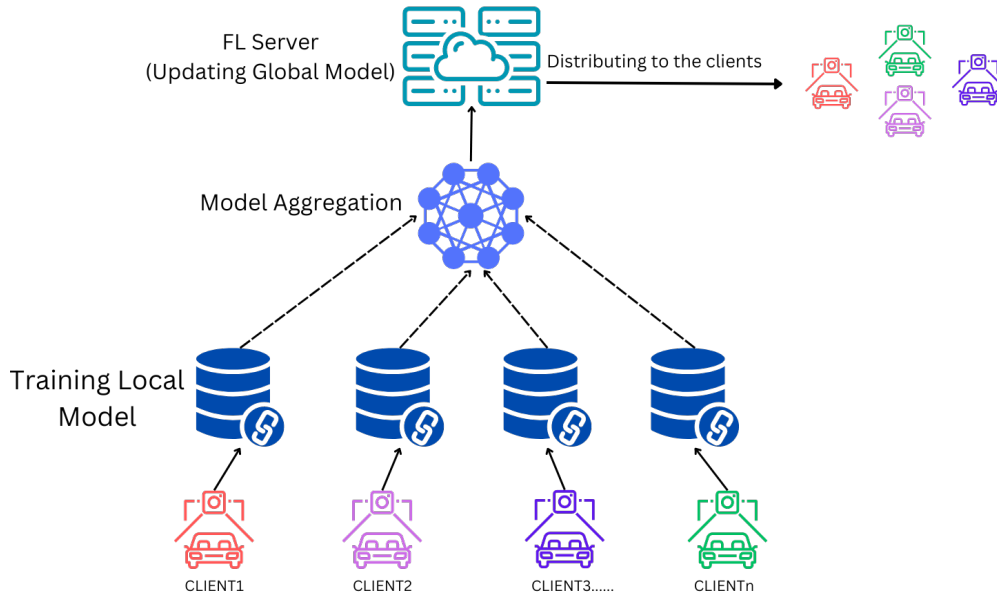


Figure 5.3: Proposed FL architecture

5.5.3 OCR

We are using OCR to extract license plates from the segmented images figure[5.4], fig[5.5]. The image extractor allows us to extract texts from any kind of image. So after completing Federated U-net model, we are using OCR to extract text from images.

The model of OCR we are using is Tesseract, which helps us to achieve significant results and also it is also open source. We have used Tesseract OCR for both English and Bangla characters.

In order to detect Bangla alphabets, we had to install the proper libraries from Tesseract. Different cities have different license plate sequences and along with them exists Bangla alphabets along with English alphabets and so we used tesseract OCR to extract both English and Bangla alphabets from images.

Figure[5.4], is the result of ocr extracting Bangla characters from given input image. Following figure[5.5], shows the result of extraction of English characters from a input frame.

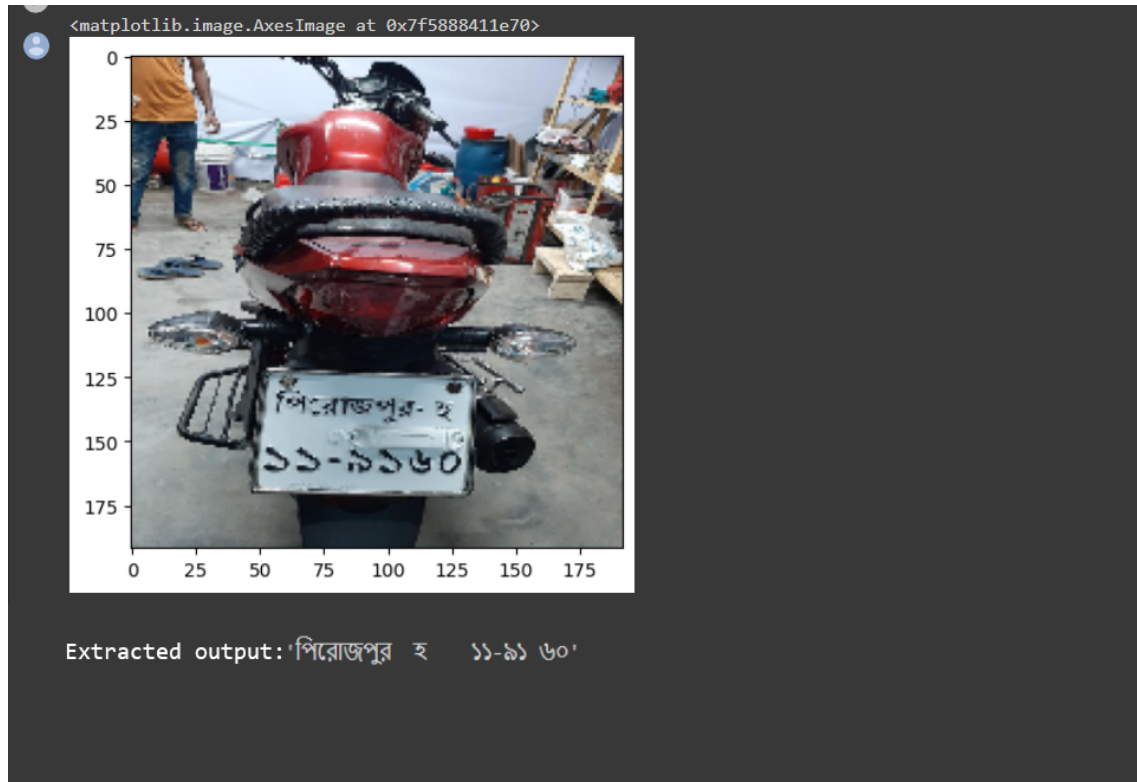


Figure 5.4: OCR-1



Extracted Output: KA G5 MG 1909

Figure 5.5: OCR-2

Chapter 6

Experimental Results

6.1 System Details

Since we utilized a significant amount of data for both our training and our testing and our experiment included a large number of still frames, we required an extremely high performance computing component in order to get the most out of our model. In order to accomplish this goal, we conducted tests on a variety of hardware setups with a view towards achieving the optimal outcome. For training and testing and also for comparison purposes in order to get an accurate perception about the results of the model we used similar hardware to train and test the model. Firstly, we train and test the models in 16GB ram of 3600 MHZ, AMD Ryzen 7 3700xx processor and RTX 3070 GPU. Also, we test and train the models in 8GB ram of 3200 MHZ, gtx 1050 ti and intel i5 seventh processor. This was done to test the model's performance in different configurations and the test was conducted in 300 epochs. We also kept the image in the same size and merged them together throughout the whole experiment.

Our work was carried out using Google Colab and a number of strong personal computers. Colab is perfectly suited for use in areas such as education, machine learning, and data analysis since it gives users the ability to create and run arbitrary Python code through a web browser. For convenience and ease of use, Colab makes use of its own virtual RAM. Because there are four separate clients in our FL U-Net model, we were required to partition the available memory and processing power among the central processing unit (CPU) and the random access memory (RAM). The execution of graphs in TensorFlow's runtime can be parallelized across various dimensions thanks to the runtime's capabilities. Individual operations have parallel implementations leveraging several CPU cores or GPU tasks.

6.2 Statistical Metrics

In the statistical metrics, we are using DICE, cosine distance, BCE dice loss, IOU, SSIM, RMAC, accuracy, recall, precision, AUC and precision F1.

The dice score equation[6.1], which may range anywhere from 0 to 1, is used to assess the performance of models. A score of 1 indicates a pixel-perfect match between the ground truth annotation (represented by green, B, and D) and the deep learning model's (represented by red, A, and D) output. The co-efficient is calculated by the multiplication of the intersected region which is then divided by the sum of regions A and that of B.

$$DSC = \frac{2 * \sum p_{true} * p_{pred}}{\sum p_{true} + \sum p_{pred} + \epsilon} \quad (6.1)$$

The cosine similarity equation[6.2], is a metric that is used in many different machine learning algorithms, such as the KNN, to calculate the distance between neighbors. It is also used in recommendation systems to propose movies that have comparable similarities, and it is used for textual data to identify the similarity of sentences inside the document. Cosine similarity is defined as the cosine of the angle between the vectors, or the dot product of the vectors divided by the product of their lengths.

$$\text{Cosine Similarity}(A, B) = \frac{\text{cosine}(\theta)}{\|A\| \cdot \|B\|} \quad (6.2)$$

AUC equation[6.3], indicates the probability that a random positive (green) example precedes a random negative (red) example. AUC values range between 0 and 1. A model with 100% incorrect predictions has an AUC of 0.0, while one with 100% accurate predictions has an AUC of 1.0. AUC is determined by plotting the ROC curve, which depicts the True Positive Rate (TPR) versus the False Positive Rate (FPR) at various classification thresholds.

$$AUC = \frac{\sum (FPR[i + 1] - FPR[i]) * (TPR[i] + TPR[i + 1])}{2} \quad (6.3)$$

The objective of the binary cross-entropy loss (BCE) equation[6.2], is to determine the degree to which the actual and anticipated picture masks vary in the amount of information that they contain. It is more broadly based on the Bernoulli distribution and it functions most effectively when there is an equal distribution of data across classes. The loss grows by $\log(p(y))$ or the log probability of it being green, for each green point ($y = 1$). For each red point ($y = 0$), it adds $\log(1-p(y))$, which is the log probability of it being red.

$$BCE(y, \hat{y}) = -[y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})] \quad (6.4)$$

In the context of object identification and segmentation, IOU equation[6.5], evaluates the degree of overlap between the ground truth area and the prediction zone. The IOU is calculated by dividing the overlap of the predicted and actual annotations by their union.

$$IOU = \frac{AO}{AU} \quad (6.5)$$

Accuracy equation[6.2], alludes to the total number of instances where the ML model produced accurate predictions and precision means the ability of a model to accurately predict a specific category. Accuracy is determined by summing the total number of accurate predictions and dividing them by the total number of predictions. Accuracy alludes to the total number of instances where the ML model produced accurate predictions and precision means the ability of a model to accurately predict a specific category. Accuracy is determined by summing the total number of accurate predictions and dividing them by the total number of predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (6.6)$$

Recall equation[6.7], is the percentage of data samples that a machine learning model correctly classifies as belonging to a class of interest (the "positive class"), relative to the total number of samples for that class. It is also known as the true positive rate (TPR). Here, the recall equation is calculated by the number of true positives, which is divided by the sum of true positives and false negatives.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (6.7)$$

The level of precision equation[6.2] is used to determine if the percentage of identifications is accurate. For this, take the sum of all predicted positive outcomes (TP) and divide it by the sum of all predicted and actual positive outcomes (TP + FP) to determine accuracy.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (6.8)$$

The Structural Similarity Index (SSIM) equation[6.2], is a perceptual metric that evaluates the decline in picture quality due to either data compression or data transmission losses. SSIM is calculated for the image with respect to the reference image.

$$\text{SSIM}(x, y) = (l(x, y) \cdot c(x, y) \cdot s(x, y))$$

(6.9)

The root mean square error (RMSE) equation[6.2], is a performance metric for regression models. It is determined by taking the square root of the average of the squared differences between predicted and actual values. RMSE assesses the proximity of the predictions to the actual values.

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

(6.10)

The accuracy factor F1 equation[6.2], is a classification model's capacity to return results. The F1 score is a singular metric that integrates recall and precision using the harmonic mean as its foundation. The calculation of F1 precision involves multiplying the values of precision and recall by 2, then dividing the result by the sum of precision and recall.

$$\text{F1 Score} = \frac{2 \cdot (\text{Precision} \cdot \text{Recall})}{\text{Precision} + \text{Recall}}$$

(6.11)

6.3 Quantitive Analysis

6.3.1 Aggregator comparison

Aggregator	Dice Coefficient	bce dice loss	Intersection over Union	RMSE	SSIM	Summation of Cosine Distance
Mean Factory	0.871	0.84	0.781	0.0017	0.014	827.471
Private Quantile Estimation Process	0.7448	0.7810	0.7246	0.028	0.2010	844.5041
Differentially Private Factory	0.8241	0.8054	0.7681	0.010	0.017	825.038

Table 6.1: Model evaluation metrics for FL aggregators

To compare the performance of the aggregators, we measured the performance table[6.1]. Among the three aggregators, The Mean Factory has the highest Intersection over Union of 0.781, SSIM of 0.014, Summation of Cosine Distance of 827.471, and Dice Coefficient values of 0.871, indicating it has the greatest performance in terms of similarity and overlap with the ground truth.

Also, The Private Quantile Estimation Process has the lowest values for Intersection over Union (0.7246), SSIM (0.201), Summation of Cosine Distance (844.5041), and Dice Coefficient (0.7448) among the three machines, indicating that it has the worst performance in terms of similarity and overlap with the ground truth. The Differentially Private Factory, on the other hand, has the lowest RMSE of 0.01 among the three machines, indicating it has the least error in terms of pixel-wise deviation from the ground truth.

Moreover, The Private Quantile Estimation Process, however, has the highest RMSE (0.028) and bce dice loss (0.781) values of the three computers, indicating that it has the most error in terms of pixel-wise difference and combined binary cross entropy and dice loss relative to the ground truth.

6.3.2 Segmented Dataset A

We can analyze from table[6.2], that Federated U-Net has a dice coefficient of 0.860587 and also intersection union of 0.759887, which is higher than the other models for segmented dataset A. It also shows us that the Federated U-Net model also has overlap between the grounded truth masks. On the other hand, Federated U-Net also has a BCE dice loss of 0.830054, a RMSE of 0.00102, and a summation of cosine distance o 486.581116, which in comparison to the other three models is the smallest. That is, it has the lowest error along with dissimilarity between the predicted and ground truth masks

But the bounding box Yolo has table[6.2], a SSIM of 0.1719946249, which is the highest of the three models, which shows us that it has the best structural similarity among the predicted and ground truth masks. Subsequently, it also has the lowest dice coefficient of 0.7899323097 and the lowest intersection over union, which is 0.758018851, which also means that it has the worst overlap between the predicted and ground truth masks.

Semantic U-Net does about the same as FL U-Net table[6.2], when it comes to the dice coefficient, which is 0.8564101827, and the intersection over union, which is 0.757284992. However, it does slightly worse when it comes to the BCE dice loss, which is 0.8240321324; the RMSE, which is 0.006420268101; and the sum of cosine distance, which is 488.514829. It also has an SSIM of 0.0129284735, which is lower than the bounding box Yolo, indicating that the predicted and ground truth masks are less structurally similar.

Model	Dice Coefficient	BCE dice loss	Intersection over Union	RMSE	SSIM	Summation of Cosine Distance
Federated U-Net	0.8605	0.8300	0.7598	0.001	0.0095	486.5811
Bounding box Yolo	0.7614	0.78993	0.75801	0.02705	0.1719	486.830
Semantic U-Net	0.8564	0.8240	0.7573	0.0064	0.0129	488.5148

Table 6.2: Model evaluation metrics for dataset A

6.3.3 Segmented Dataset B

In the case of segmented dataset B, we have a similar results of different components as in segmented dataset A table[6.3], where Federated U-Net has a dice coefficient of 0.861832 and intersection over union of 0.768747, which is the highest of the three models for this dataset, indicating that it has the greatest overlap between the predicted and ground truth masks.

BCE dice loss and RMSE table[6.3], are also low for Federated U-Net, which are 0.827546 and 0.001621, respectively and among the three models for this dataset, it is the lowest, indicating that its predicted and ground truth masks contain the least amount of error.

Bounding box based Yolo has the highest SSIM table[6.3], among the three models for this dataset, which is 0.013538, indicating that its predicted and ground truth masks have the highest structural similarity. The worst overlap between the predicted and ground truth conceals

The performance of semantic U-Net is the same as that of federated U-Net in terms of dice coefficient and intersection table[6.3], which are both 0.8594272124 and 0.7618999159 over union, but slightly worse in terms of BCE dice loss, RMSE, and SSIM, which are all 0.8194762179 and 0.01593166572, respectively. It also has a cosine distance of 851.0384912 that is greater than federated U-Net, indicating that the predicted and ground truth masks are more dissimilar.

Model	Dice Coefficient	bce dice loss	Intersection over Union	RMSE	SSIM	Summation of Cosine Distance
Federated U-net	0.861	0.82	0.768	0.0016	0.013	849.748
Bounding box Yolo	0.7098	0.7818	0.6986	0.032	0.2066	933.7071
Semantic U-Net	0.8594	0.8196	0.7618	0.011	0.015	851.038

Table 6.3: Model evaluation metrics for dataset B

6.3.4 Segmented Dataset C

Similar to segmented dataset A [6.2] and B table[6.3], we have results for segmented dataset C [6.4]. The dice coefficient and intersection over union are also the highest in Federated U-Net, which are 0.868561 and 0.777896, respectively, resulting in the best overlap between the predicted and ground truth masks. BCE dice loss and RMSE are also the lowest in this, and these values are 0.816285 and 0.001721, respectively, which gives it the least error between the predicted and ground truth masks.

Bounding box Yolo also has table[6.4], the lowest SSIM (0.07678744532) of the three models for this segnebted dataset C, which means that its predicted and ground truth masks have the least structural similarity. A higher dice coefficient (0.852533107) and intersection over union (0.7475408588) than semantic U-Net shows that the predicted and ground truth masks overlap more.

Semantic U-Net also table[6.4], does about the same as federated U-Net in terms of dice coefficient (0.8585821967) and intersection over union (0.7683776051), but it does slightly worse in terms of BCE dice loss (0.8162774641), RMSE (0.007230898195), and sum of cosine distance (1749.634687). It also has an SSIM of 0.03202398137, which is lower than Yolo's bounding box, indicating that the predicted and actual masks are less structurally similar.

Model	Dice Coefficient	bce dice loss	Intersection over Union	RMSE	SSIM	Summation of Cosine Distance
Federated U-Net	0.753	0.92	0.6584	0.004	0.040	1657.8394
Bounding box Yolo	0.7218	0.9101	0.632	0.183	0.1114	1701.856
Semantic U-Net	0.746	0.9205	0.6510	0.00870	0.04909	1659.652

Table 6.4: Model evaluation metrics for dataset C

6.3.5 Segmented Dataset D

Similar cases can also be seen in segmented dataset D table[6.5], Federated U-Net has the highest dice coefficient and intersection over union and these values are 0.753287 and 0.6584, respectively, resulting in the greatest overlap between the predicted and ground truth masks. The BCE dice loss of 0.927575 and RMSE of 0.004834 are the lowest in this regard, giving it the lowest error between the predicted and actual masks.

Bounding box Yolo also has table[6.5], a SSIM value of 0.1114913517, which is lower among the three models for this dataset, indicating that its predicted and ground truth masks share the least structural similarity, and its dice coefficient (0.7218338895) and intersection over union (0.6327162293) are both greater than semantic U-Net, indicating that the predicted and ground truth masks overlap more.

Comparable to federated U-Net in terms of dice coefficient of segmentic U-Net table[6.5], (0.7469170493) and intersection over union (0.6510734213), but marginally inferior in terms of BCE dice loss (0.9205578662), RMSE (0.008709447939), and sum of cosine distance (1659.652578). It also has an SSIM of 0.04909789517, which is less than Yolo's bounding box, indicating that the actual and predicted masks are less structurally similar.

Model	Dice Coefficient	bce dice loss	Intersection over Union	RMSE	SSIM	Summation of Cosine Distance
Federated U-Net	0.883	0.852	0.798	0.002	0.018	1049.748
Bounding box Yolo	0.7018	0.8269	0.7156	0.0953	0.181	1101.653
Semantic U-Net	0.876	0.8467	0.7985	0.002	0.0284	1050.618

Table 6.5: Model evaluation metrics for dataset D

6.3.6 Segmented Dataset Merged

Federated U-Net has the maximum dice coefficient (0.883532) and intersection over union (0.798747) among the models, according to table[6.6]. In addition, it demonstrates that the federated U-Net model has overlapping grounded truth masks.

Federated U-Net has the lowest BCE dice loss of 0.852922, RMSE of 0.002321, and cosine distance sum of 1049.748901, which is lower than the other three models. In other words, it produces the least amount of error and dissimilarity between the predicted and actual masks.

Among the three models, however, the bounding box yolo table[6.6], has the highest SSIM of 0.181617917, indicating that it has the greatest structural similarity between the predicted and actual masks. Moreover, it has the lowest dice coefficient of 0.7018329573 and an intersection over union of 0.7156945862, signifying the worst overlap between the predicted and ground truth masks.

Semantic U-Net is similar to federated U-Net in terms of dice coefficient and intersection over union table[6.5], which are both 0.8735968181 and 0.7985557545. However, it is slightly worse in terms of BCE dice loss, which is 0.8467524083, RMSE, which is 0.002870451509, and sum of cosine distance, which is 1050.618069. In addition, it has a lower SSIM than bounding box Yolo, which is 0.02843542861, indicating less structural similarity between the predicted and actual masks.

Based on these metrics, federated U-Net appears to be the best model among the three for this dataset. When comparing our model to others, we also considered other factors such as performance, memory, robustness, etc.

Model	Dice Coefficient	bce dice loss	Intersection over Union	RMSE	SSIM	Summation of Cosine Distance
Federated U-Net	0.883	0.852	0.798	0.002	0.018	1049.748
Bounding box Yolo	0.701	0.826	0.715	0.0953	0.181	1101.653
Semantic U-Net	0.8735	0.8467	0.798	0.002	0.028	1050.618

Table 6.6: Model evaluation metrics for merge dataset

6.4 Qalitative Analysis

6.4.1 Chapter comparison

Segmented Dataset A:

From the figure[6.1] we can see that for irregular masking U-Net and FL U-Net performs substantially well and on the other hand Yolo was not able to detect irregular masking properly and the masking was done correctly. True mask is the masking done manually and when we compare U-Net, FL U-Net, and YOLO we can see that YOLO cannot perform accurately in this regard but the U-Net and FL U-Net perform similarly and quite accurately in detecting irregular mask in the last image masking.

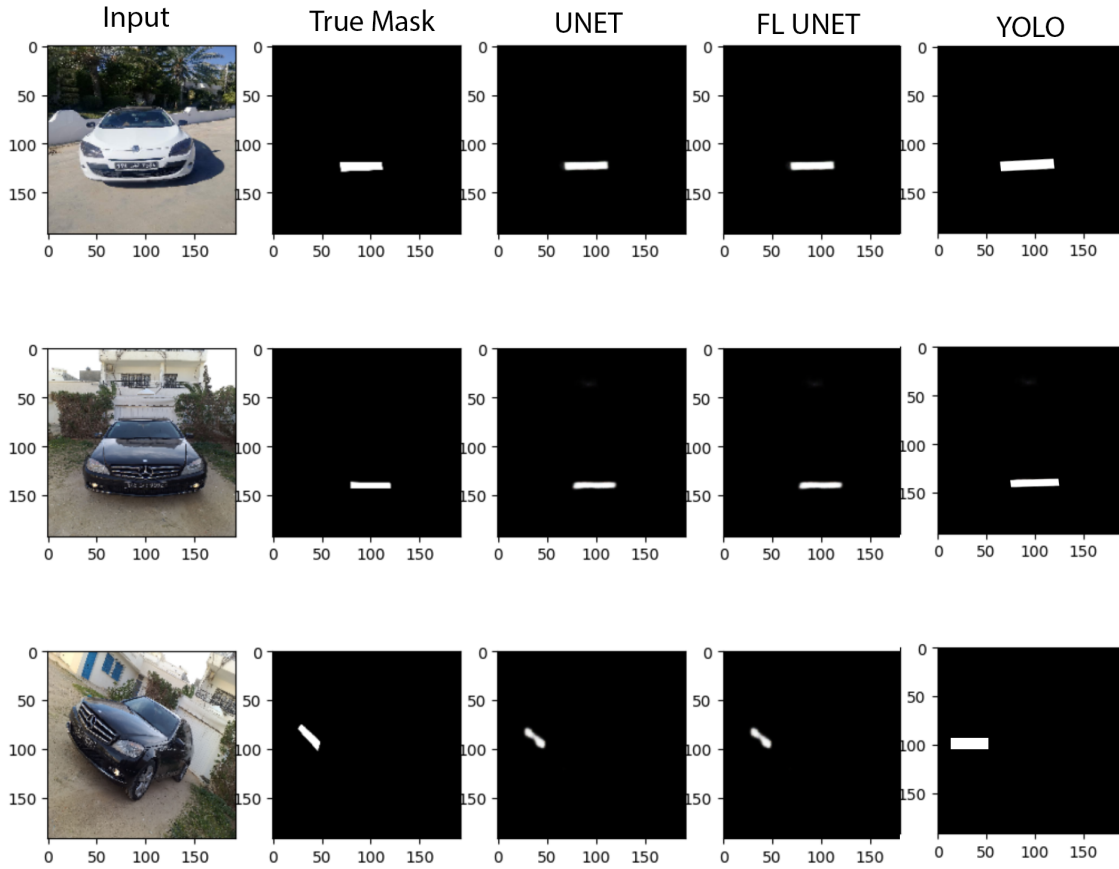


Figure 6.1: Segmented dataset A

Segmented Dataset B: Here the figure[6.2] demonstrates that for irregular masking after true masking where we manually performing the masking of the license plate and comparing U-Net, FL U-Net, and YOLO to the true mask, we can see that YOLO cannot perform accurately in this regard, whereas U-Net and FL U-Net perform similarly and quite accurately in detecting irregular mask in the most recent image masking. But for regular making the three models gave similar accuracy.

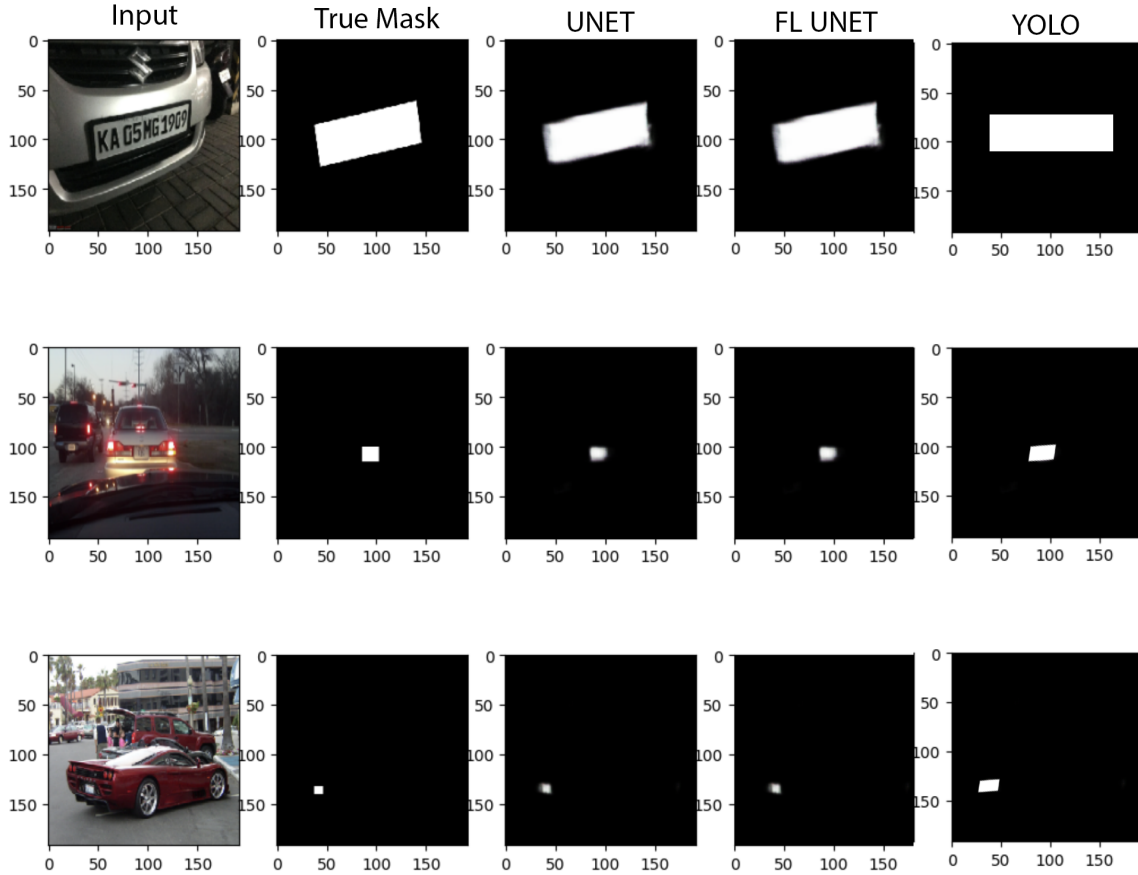


Figure 6.2: Segmented dataset B

Segmented Dataset C: Here figure[6.3], we compare the models to the true mask in the last two images, the models performs quite similarly, but when it comes for the first image where there remains two vehicles and two license plates, in case of YOLO for the regular license plate it performed perfectly but for the irregular license plate masking was not accurately and precisely. On the other hand both U-Net and FL U-Net were able to perform the task properly performing regular and irregular masking.

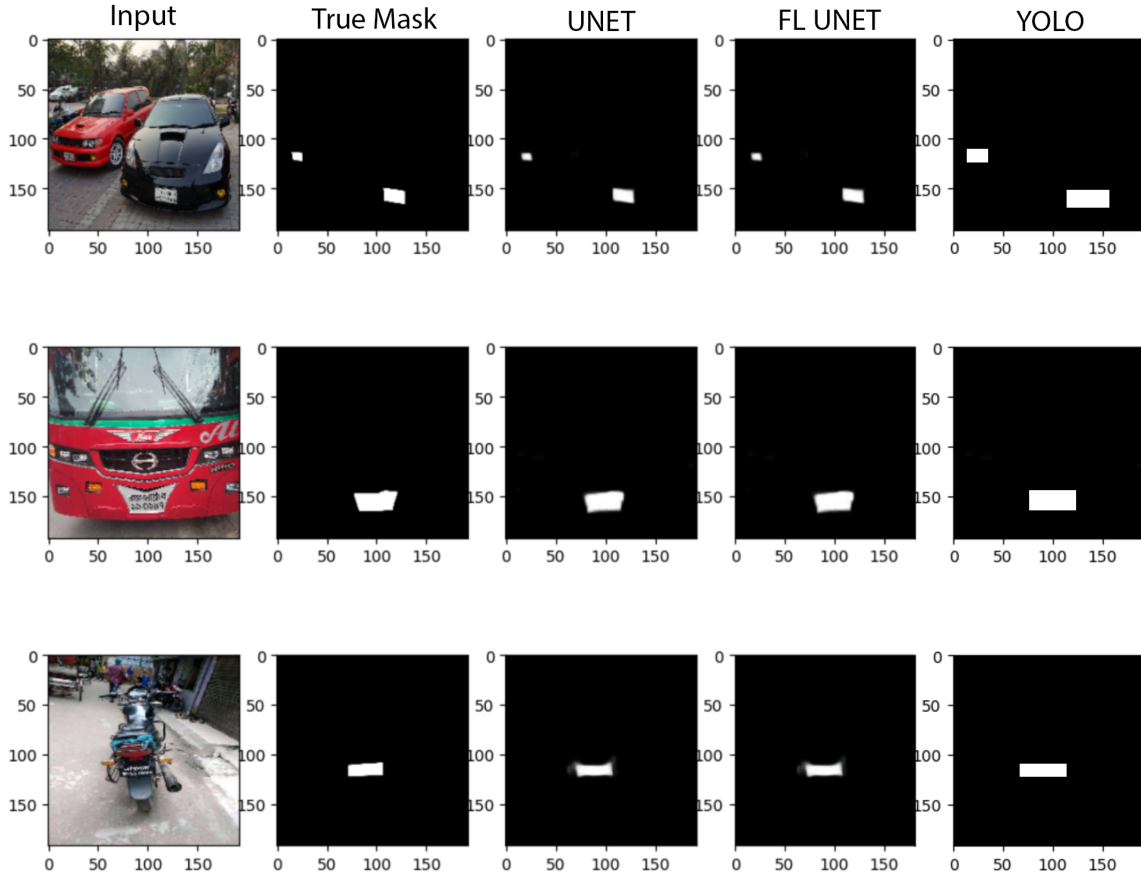


Figure 6.3: Segmented dataset C

Segmented Dataset D: Similarly figure[6.4], when comparing the models to the true mask in the second image, the models perform quite similarly. However, when it comes to the first and third image, which contains irregular masking, YOLO even performed flawlessly for the second image of regular license plate but not so well for the irregular license plate of the first and third. On the other hand, both U-Net and FL U-Net were able to successfully complete the task by conducting both regular and irregular masking.

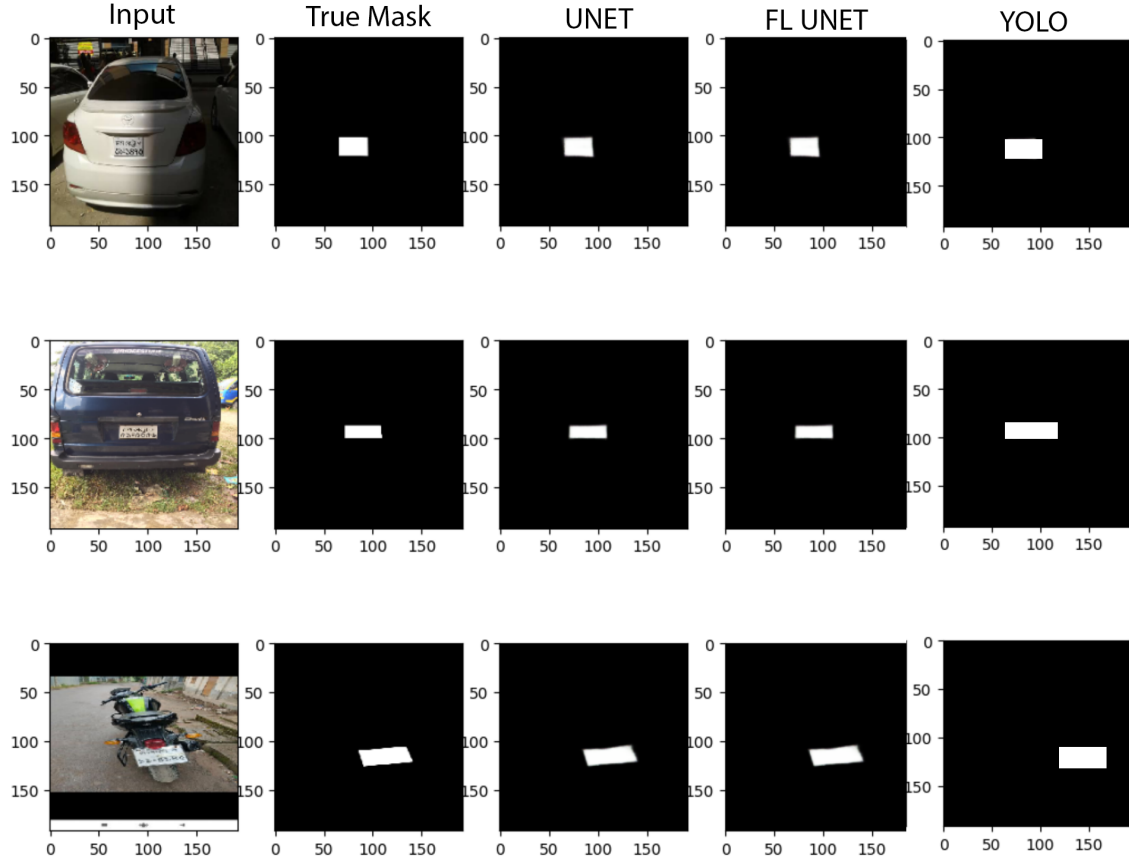


Figure 6.4: Segmented dataset D

Segmented Dataset Merged Similarly figure[6.5], when comparing the models to the true mask in the fifth image, which contains irregular masking, YOLO even performed flawlessly for the second image of regular license plate but not so well for the irregular license plate of fifth image. On the other hand, both semantic U-Net and FL U-Net were able to successfully complete the task by conducting both regular and irregular masking. In proposed model, the irregular masking was detected more precisely than compared to other models.

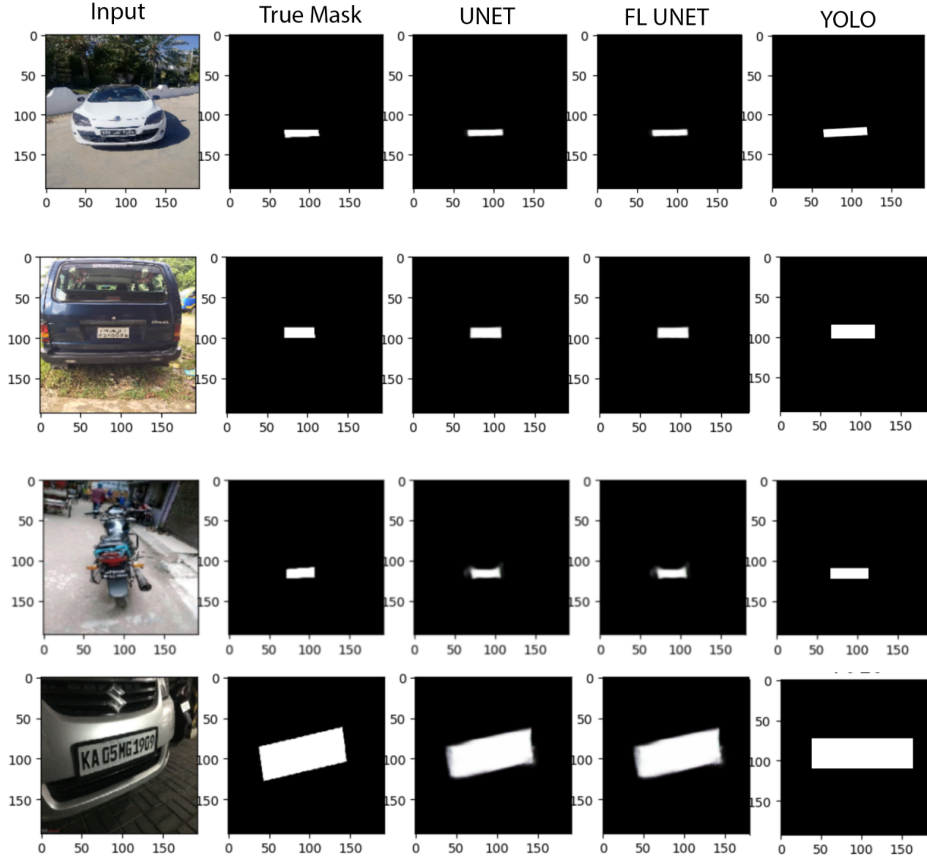
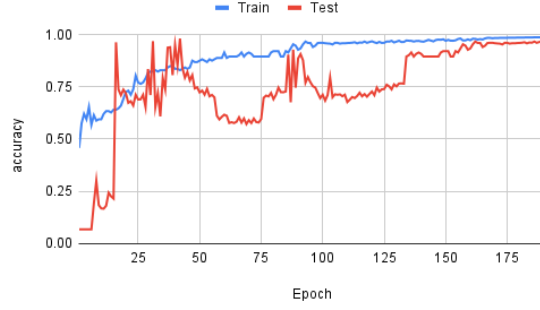


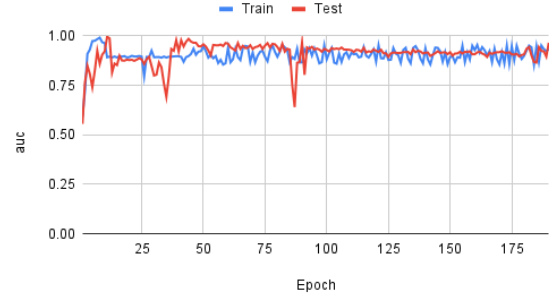
Figure 6.5: Merge dataset

6.5 Model Performance

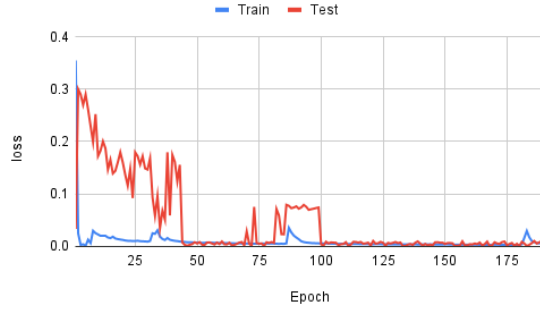
6.5.1 Federated U-Net



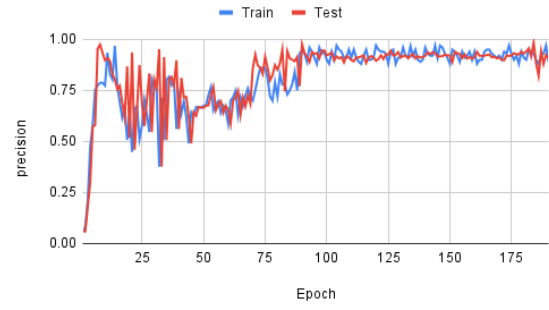
(a) Federated U-Net accuracy



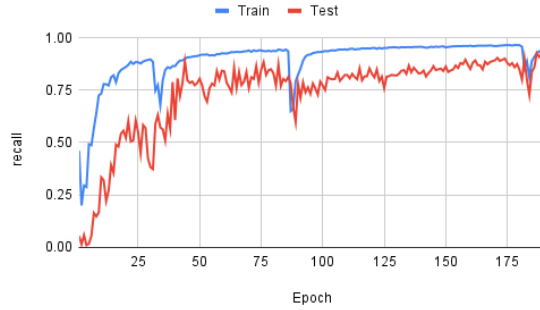
(b) Federated U-Net auc



(c) Federated U-Net loss



(d) Federated U-Net precision



(e) Federated U-Net recall

Figure 6.6: Epoch-wise recall, precision, loss, accuracy for FL U-Net model

Test	Train	Metric
0.96	0.98	Accuracy
0.96	0.97	AuC
0.95	0.97	Recall
0.0016	0.0014	Loss
0.96	0.97	Precision

Table 6.7: Federated U-Net table

In our proposed FL U-Net accuracy graph[6.7], we can see that there is a rapid spike of accuracy at the beginning during testing, which reaches its highest at 25 to 50 epochs, and during training, it reaches its highest at around 175 epochs and maintains a steady increase. And during testing, the flow is not steady; there is a sudden decrease at different epochs, reaching its lowest at around 175 and maintaining that stability.

From the auc graph[6.7], there has been a steady flow of the auc mark, but during testing, there has been a decrease around 50 epoch and 85 epoch, which remains at its lowest.

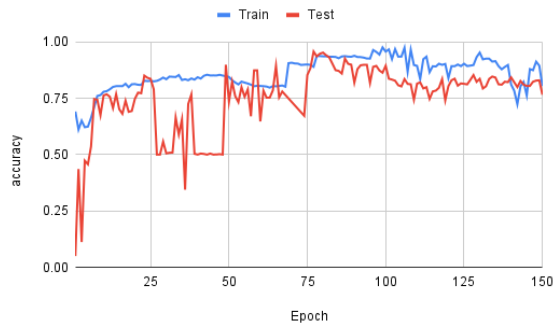
During training, there is an increase, which is the highest, and then afterwards, the loss goes down graph[6.7]. Even in the testing stage, similar results can be seen, but from 75 to 100 epochs, there is a small increase, but it goes down. During testing, the highest value remains at the beginning.

Both during training and testing, there remains a gradual instability where the precision fluctuates from 15 epochs to around 80 epochs and goes to its lowest at around 35 epochs. But remains unsteady but still has a high precision around 100 epochs [6.7].

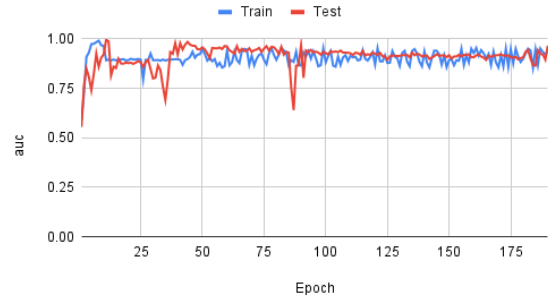
The recall for testing and training is gradually increasing, and during training, the recall sometimes decreases, but the majority of the time it remains closer to the highest. During testing, the recall remains quite unstable from 0 to 100 epochs [6.7], but afterwards it reaches some stability.

In our Unet-FL model table[6.7], we can see during testing we get a score of 0.96 during testing and during training Unet-FL gets a score of 0.98. On the other hand for AUC during testing there is a score of 0.96 and during training 0.97. And for Recall in training and testing there are a score of 0.95 and 0.97 respectively. In case of Loss there remains a score of 0.0016 and 0.0014 during training and testing respectively. Finally, for precision during precision our model generates during training and testing gets 0.96 and 0.97 respectively.

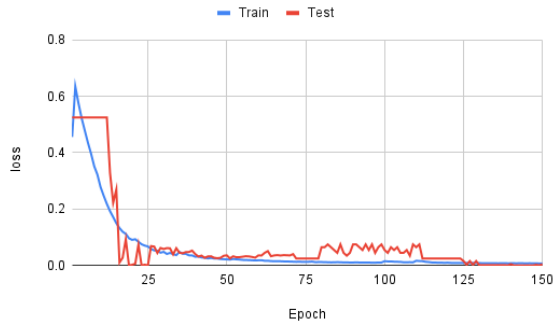
6.5.2 U-Net



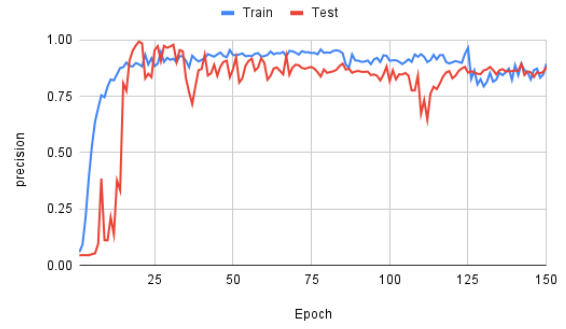
(a) U-Net accuracy



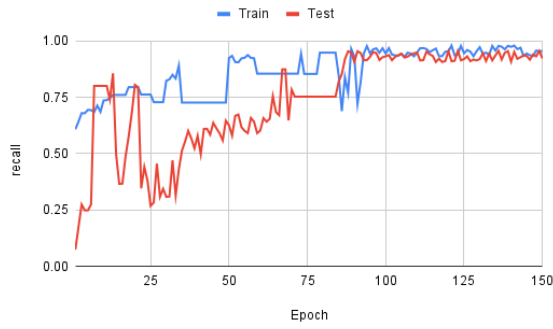
(b) U-Net auc



(c) U-Net loss



(d) U-Net precision



(e) U-Net recall

Figure 6.7: Epoch-wise recall, precision, loss, accuracy for U-Net model

Test	Train	Metric
0.94	0.96	Accuracy
0.93	0.94	AUC
0.92	0.94	Recall
0.0044	0.0034	Loss
0.94	0.95	Precision

Table 6.8: U-Net table

Here in the graph[6.7], we can see that there is a rapid spike of accuracy at the beginning during testing and during training it reaches the highest at around 110. Then it comes down at its lowest at around 130 epoch. And during testing at the 25 epoch mark there is a big downfall in the accuracy and around 35 it is the lowest mark and around 49 the line goes back and it reaches its highest.

The AUC graph[6.7], during the training and testing there has been a gradual increase. During testing there has been a decrease during 50 epoch, 60 epoch, 85 epoch, around 125 its at its lowest and around 150 it remains at its lowest.

During training there is an increase highest and then afterwards the loss goes down graph[6.7]. Even in the testing stage the similar results can be seen and from 100 to 125 epochs there is a small increase but it goes down.

Both during training and testing the precision increases figure[6.7], and the testing reaches its highest at around 25 but has a sudden decrease during 35 and 115 epochs but remains at a steady flow from 125 epochs. During 125 epochs of testing it decreases slightly.

The recall for testing fluctuates quite a bit during the initial stage and near 100 epochs graph[6.7]. During training the recall does not fluctuate that much and goes at its lowest and during 25 to 50 epochs. Also at closer 80 it starts to decrease quite a bit.

During testing, segmented U-Net model table[6.8] receives a score of 0.94 during testing, whereas during training, segmented U-net receives a score of 0.96. In contrast, the AUC score during testing is 0.93 and during training it is 0.94. The scores for Recall in testing and training are 0.92 and 0.94, respectively. In the event of a Loss, the remaining scores during training and testing are 0.0044 and 0.0034, respectively. In terms of precision during testing, our model generates 0.94 and 0.95 respectively during training and testing.

6.5.3 YOLO

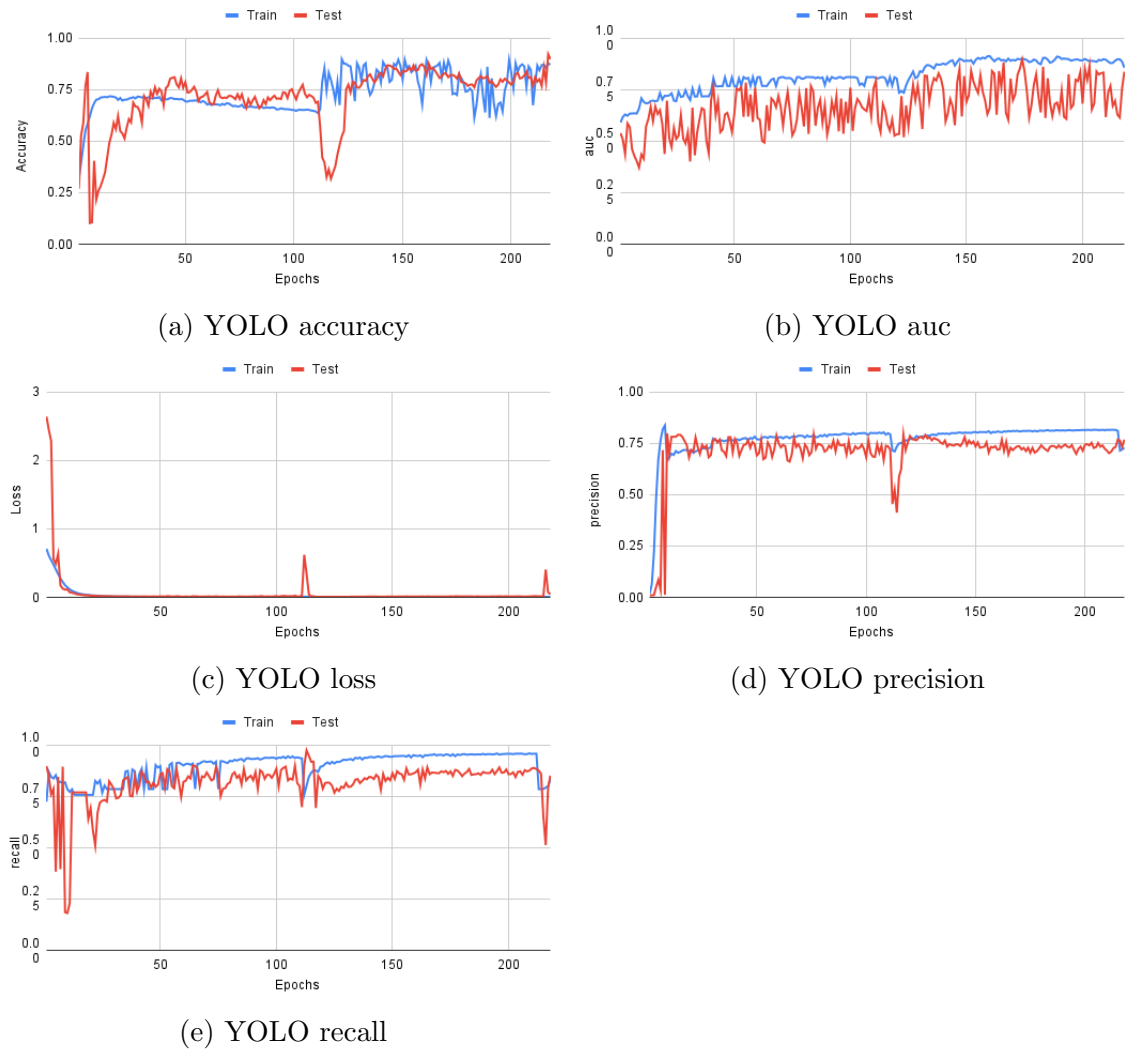


Figure 6.8: Epoch-wise recall, precision, loss, accuracy for YOLO model

Test	Train	Metrics
0.87	0.90	Accuracy
0.85	0.89	Auc
0.87	0.90	Recall
0.07	0.08	Loss
0.79	0.83	Precision

Table 6.9: YOLO table

The accuracy of Yolo graph[6.8], fluctuates quite a bit, but especially during testing, where it goes from its lowest point to its highest point within 30 and over 200 epochs. And during training, the accuracy was quite stable up until 100 epochs, after which it started to become unstable. The highest accuracy was around 130 epochs, and the lowest was from 150 to 200 epochs.

During training, the AUC was steady graph[6.8], and did not have any kind of sudden fluctuation. But during testing, the fluctuation existed by a lot and reached its highest at around 125 epochs and its lowest at 10 to 30 epochs.

Both the training and testing time precision were gradually increasing and closer to the majority of the epochs during testing graph[6.8]; the lowest was suddenly at around 100 epochs and afterwards it went back. During testing, precision had a small amount of instability.

Both during training and testing, the recall reached its highest level and maintained a gradual flow graph[6.8]. But, during testing, there was a sudden decrease in the recall at around 20 epochs and at over 200 epochs, there was a sudden decrease.

During training the loss of YOLO went down and maintained a steady flow graph[6.8]. Even during testing there there was steady flow for most of the periods but there were sudden spikes of increase at around 100 epochs and 200 epochs.

Yolo model table[6.9] receives a score of 0.87 during testing and receives a score of 0.90 during training. During testing, the AUC score is 0.85, while during training it is 0.89. Training and testing recall scores are 0.87 and 0.90, respectively. In case of Loss, the remaining training and testing scores are 0.07 and 0.08, respectively. In terms of precision during testing, the model produces 0.79 and 0.83 during training and testing, respectively.

6.6 Model Comparison



Figure 6.9: Model comparison

Graph[6.9], describe the performance of the three model that we executed, which are Yolo, Federated U-Net and semantic U-Net which is also our model but excluding the federated learning aspect. At first, We analyze the data individually for Yolo, U-Net and FL U-Net. And individually we get graph for Yolo graphs[6.8]for U-net the graphs[6.7] for our model which is U-Net FL model graphs[6.6].

Among the three models U-Net FL, Sementic U-Net and Yolo. Our proposed model FL U-Net has the better peroformance during train and test and accuracy, Auc, Recall, precision, loss has a significiant number. Afterwards the second was U-Net and third was Yolo.

According to graph[6.9], our proposed Federated U-net has better performance compared to other models during this study. There are a number of reasons for our FL U-Net model's improved performance. First, it is likely that the use of four clients with distributed datasets contributed to the observed improvement. Our FL U-Net model gains access to a wider range of data by utilising the data from multiple sources, improving its overall performance. Furthermore, our FL U-Net model's implementation of early stopping late results in superior performance. This method enables training to be stopped when the model's performance on a validation set begins to decline, preventing overfitting. Early stopping late ensures optimal generalisation and boosts the overall accuracy of our FL U-Net model by successfully regulating the model's learning process.

Additionally, when compared to our semantic U-Net model, our FL U-Net model runs faster. This effectiveness is a result of FL U-Net’s distributed dataset, which is split among the four clients. The training process can be parallelized by distributing the data among various clients, greatly reducing the time needed for training and inference. The use of distributed datasets across four clients and the implementation of early stopping late are both responsible for the improved performance of our FL U-Net model.

Moreover, the model has a shorter runtime than our semantic U-Net model, which leads to improved efficiency. This is due to the distributed nature of the dataset. These elements work together to make FL U-Net a viable strategy for enhancing performance and effectiveness in image segmentation tasks.

Chapter 7

Discussions and Further Improvements

7.1 Discussion

The privacy of individual data is very important in modern times. A small mistake can result in the breach of many user data and even can lead to the harm of many as these sensitive data are very important in order to keep the privacy and safety of many. That is why federated learning is very important as video transmission from each car is highly network intensive and requires a lot of cache or temporary storage facilities to bring processed and feed into neural networks. Uploading data to any central server system raises security and privacy concerns. Federated can then help in the prevention of any kind of data breach. With this approach, numerous users can build a centralized, reliable and accurate ML model without disclosing personal user information. As a result, it aids in resolving important concerns including data privacy, data access rights, data protection and access to heterogeneous data. Federated learning addresses crucial challenges including data privacy, data security, data access rights, and access to heterogeneous data by allowing several players to develop a single, strong machine learning model without sharing data. The following are some of the main advantages of federated machine learning. Instead of uploading and storing training data on a central server, FL enables devices like smartphones to develop a shared prediction model cooperatively while keeping the training data on the computer.

Another advantage of Federated learning is that it can work with large amounts of data.[52] Through the use of federated learning, many universities are able to work together to design an algorithm for machine learning without having to share their data. Instead, organizations simply communicate the parameters of their models with one another, which enables them to reap the benefits of a model developed using a bigger dataset while still protecting the confidentiality of their own data. Without compromising users' privacy, federated learning makes it possible to construct models of user behavior by drawing information from a data pool consisting of smartphones. Federated learning is built to accommodate huge datasets that are stored in a variety of locations and on several devices. We used the equation where After calculation we were able to determine for early stopping we get 665 seconds and for full epoch or for 300 epochs we get 10500 seconds. Since there exists four

clients in our model, we divided the results from U-Net by four.

Again, the use of U-Net is very important as it has the ability to detect irregular masking. This is very crucial due to the fact that license plates are not always well in a favorable situation and thus we will not be able to always properly detect irregular masking even with the more widely used YOLO model. That is U-Net is a better alternative as it is more robust and accurate for irregular masks. Mask is the portion that carries the license plate and that's why it's better.

[64] Utilizing a semantic segmentation strategy, U-Net is able to assist in the identification of license plates. This indicates that it is able to differentiate the license plate from the backdrop as well as any other things that may be present in the picture. U-Net is a form of neural network that contains a structure known as an encoder-decoder, which gives it the ability to learn features on a variety of scales and combine them in order to generate accurate segmentation masks. The following is a list of some of the benefits of utilizing U-Net for license plate detection: It is able to deal with challenging conditions such as those including weather, distance, angle and brightness.

Through the use of a multi-scale dilated convolution feature fusion module, it is able to expand the receptive field without negatively affecting the picture quality.

Through the use of spatial attention mechanisms, it is able to improve the attention paid to the license plate area.

By using the focus loss feature, it may make the learning and training process more effective.

U-Net performs well in comparison to other techniques for license plate identification in terms of its accuracy, robustness, and efficiency. It does not need the segmentation of characters, which enables it to avoid the accuracy decrease that might result from segmentation that is either insufficient or inaccurate.

[67] By using an LSTM network to identify the whole sequence of plate features, it is able to access the information that is contained inside the context of the license plate. By using a multi-scale convolution feature fusion module and a spatial attention mechanism, it is able to handle a wide variety of circumstances, including occlusion, uneven lighting, rotation, and vagueness, amongst others.

Using the focused loss function, which lessens the impact of samples that are simple and concentrates on those that are challenging, it may make the process of learning and training more effective.

It is capable of achieving high detection rates and accurate identification on a variety of datasets, including CCPD and SSIG.

In our model, Unet-FL, we can observe that we get a score of 0.96 during testing and a score of 0.98 during training. AUC, on the other hand, scored 0.96 during testing and 0.97 throughout training. Additionally, the scores for Recall through-

out training and testing are 0.95 and 0.97, respectively. When there is a loss, the scores during training and testing stay at 0.0016 and 0.0014, respectively. Finally, our model yields 0.96 and 0.97 for accuracy during training and testing, respectively.

After calculation we were able to determine for early stopping we get 2660 seconds and for full epoch or for 300 epochs we get 42000 seconds.

7.2 Further Improvement

One of our goals figure[7.1], for the not-too-distant future is to evaluate the strengths and weaknesses of our model in relation to those of other models that have already been constructed. In addition to this, one of our goals is to increase the accuracy of the detection rate model we use. In order to broaden the scope of our inquiry, we want to take even more pictures of license plates from other vehicles. Because of this, we will be able to train our model more, study its limits, and make it more robust. Another essential part of our system is the ability to recognize license plates, which may be accomplished by applying our model to digital photographs. As a result of this, we want to put our model on dashboard cameras so that it may be used in real-time recordings, as the videos there will be more unstable and so license plate detection will be more complex in comparison to detection of license plates from still frames. This model is designed to be applied to such cameras. In addition, the only thing that our model can identify at the moment are license plates. In the future, we aim to be able to design a system that, in addition to providing information about the vehicle that is being driven, is also able to offer information about the owner of the vehicle by way of the registration of the license plate.

We also intend to implement web applications as people will also use the model in order to see their own information and also police officers can benefit from this model by finding the information and data they need, in pursuit of any kind of case and they will be very much benefited from the model. Federated learning is very robust in keeping privacy and so having a web implementation will not breach the privacy of the users involved.

We also intend to use OCR to extract information from license plates and see if it matches with any kind of person of interest the authorities will be informed real time about the person of interest or have conducted any kind of crime like speeding, illegal parking or any crimes that can be detected by the model then it will automatically inform the local authorities about the situation and also keep record of the information of the license plate and the person involved.

Our methodology is one that is very respectful of users' privacy, as it makes an effort for both identify licensing regions in every circumstance and counteract any abnormal masking while preserving the users' right to personal anonymity. These goals may be accomplished with our strategy while still protecting the confidentiality of the user. The protection of the general public and the avoidance of the publication of sensitive information were the motivations for the development of this concept.

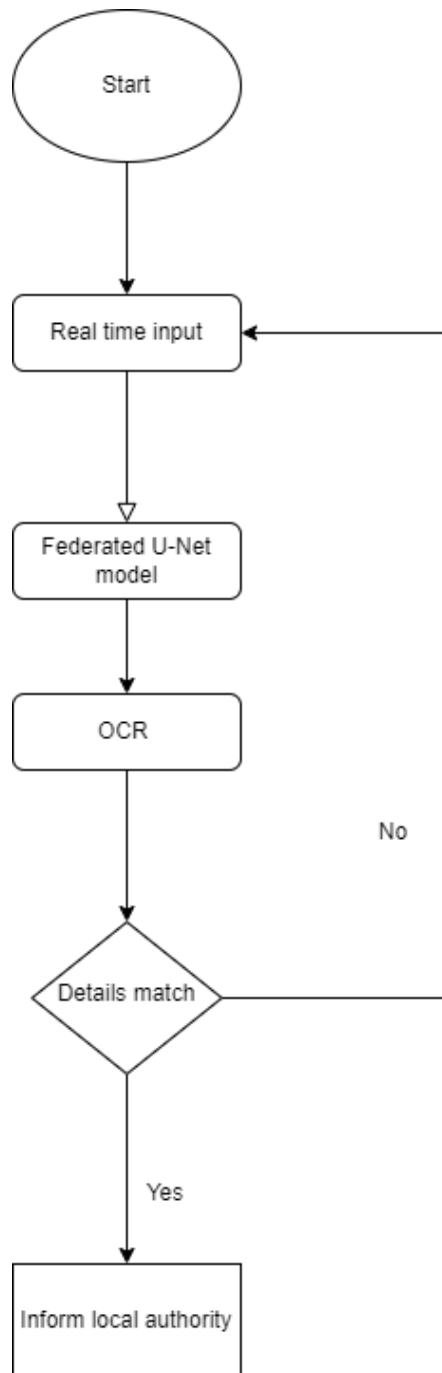


Figure 7.1: Workflow of future improvement

Chapter 8

Conclusion

A license plate detection system is essential in today's world for the monitoring of different cars to guarantee appropriate licensing during traffic, the identification of vehicles involved in any form of crime, and illegal parking. Although it is possible to do these duties manually, doing so will lead to a significant number of errors, and in today's culture, a license plate detecting system is very necessary. And the U-Net in this scenario, although being exceedingly uncommon, is quite important since it can minimize irregular masking under adverse settings and is also more resilient than models utilizing geometric bounding boxes. This is despite the fact that the U-Net is incredibly rare. In addition, with the help of federated learning, we are able to make the whole system more trustworthy and safe, as it will contribute to the protection of data privacy. We are able to create a system that is safe, trustworthy, and strong thanks to the adoption of U-Net with federated learning. This system is capable of withstanding any kind of challenging circumstance. During the process of putting this approach into action, we ran across a number of problems, including the following:

- In Google Colab, there was a discrepancy between versions of Tensorflow Federated, which resulted in extra problems.
- There were issues with the gathering of the data.
- More processing capacity from the central processing unit (CPU) as well as processing power from the graphics processing unit (GPU) was necessary for this paradigm in Google Collaborate.

At the end of our research we were able to utilize the materials present and were able to make a robust model using Federated learning and U-Net for which we were able to overcome the shortcoming of irregular masking and also were able to ensure the privacy of the users involved by using our model. We hope to overcome the shortcomings of our model in the near future by doing more testing and training and also collecting more data for our model.

Bibliography

- [1] S. Mori, C. Y. Suen, and K. Yamamoto, “Historical review of ocr research and development,” *Proceedings of the IEEE*, vol. 80, no. 7, pp. 1029–1058, 1992.
- [2] K. Maniruzzaman and R. Mitra, “Road accidents in bangladesh,” *IATSS research*, vol. 29, no. 2, p. 71, 2005.
- [3] B. Graham, “Fractional max-pooling,” *arXiv preprint arXiv:1412.6071*, 2014.
- [4] M. D. Bloice, C. Stocker, and A. Holzinger, “Augmentor: An image augmentation library for machine learning,” *arXiv preprint arXiv:1708.04680*, 2017.
- [5] Y. Sun, E. Lank, and M. Terry, “Label-and-learn: Visualizing the likelihood of machine learning classifier’s success during data labeling,” in *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, 2017, pp. 523–534.
- [6] Q. Zhang, Z. Cui, X. Niu, S. Geng, and Y. Qiao, “Image segmentation with pyramid dilated convolution based on resnet and u-net,” in *Neural Information Processing: 24th International Conference, ICONIP 2017, Guangzhou, China, November 14-18, 2017, Proceedings, Part II 24*, Springer, 2017, pp. 364–372.
- [7] J. Du, “Understanding of object detection based on cnn family and yolo,” in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1004, 2018, p. 012029.
- [8] R. Laroca, E. Severo, L. A. Zanlorensi, *et al.*, “A robust real-time automatic license plate recognition based on the yolo detector,” in *2018 international joint conference on neural networks (ijcnn)*, IEEE, 2018, pp. 1–10.
- [9] H. Sun, M. Fu, A. Abdussalam, Z. Huang, S. Sun, and W. Wang, “License plate detection and recognition based on the yolo detector and crnn-12,” *Lecture Notes in Electrical Engineering*, 2018.
- [10] Z. Zhou, M. M. Rahman Siddiquee, N. Tajbakhsh, and J. Liang, “Unet++: A nested u-net architecture for medical image segmentation,” in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 20, 2018, Proceedings 4*, Springer, 2018, pp. 3–11.
- [11] R.-C. Chen *et al.*, “Automatic license plate recognition via sliding-window darknet-yolo deep learning,” *Image and Vision Computing*, vol. 87, pp. 47–56, 2019.
- [12] M. S. Farag, M. M. El Din, and H. El Shenbary, “Parking entrance control using license plate detection and recognition,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 15, no. 1, pp. 476–483, 2019.

- [13] M. S. Hossen, “Analysis of road accidents in bangladesh,” *Am. J. Transp. Logist*, pp. 1–19, 2019.
- [14] T. Islam and R. I. Rasel, “Real-time bangla license plate recognition system using faster r-cnn and ssd: A deep learning application,” in *2019 IEEE International Conference on Robotics, Automation, Artificial-intelligence and Internet-of-Things (RAAICON)*, IEEE, 2019, pp. 108–111.
- [15] A. Jokic and N. Vukovic, “License plate recognition with compressive sensing based feature extraction,” *arXiv preprint arXiv:1902.05386*, 2019.
- [16] I. Kusumadewi, C. A. Sari, E. H. Rachmawanto, *et al.*, “License number plate recognition using template matching and bounding box method,” in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1201, 2019, p. 012067.
- [17] P. Marzuki, A. Syafeeza, Y. Wong, N. Hamid, A. N. Alisa, and M. Ibrahim, “A design of license plate recognition system using convolutional neural network,” *International Journal of Electrical and Computer Engineering*, vol. 9, no. 3, p. 2196, 2019.
- [18] B. Suvarnam and V. S. Ch, “Combination of cnn-gru model to recognize characters of a license plate number without segmentation,” in *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, IEEE, 2019, pp. 317–322.
- [19] Y. Weng, T. Zhou, Y. Li, and X. Qiu, “Nas-unet: Neural architecture search for medical image segmentation,” *IEEE Access*, vol. 7, pp. 44 247–44 257, 2019. DOI: 10.1109/ACCESS.2019.2908991.
- [20] X. Zhao, Y. Yuan, M. Song, *et al.*, “Use of unmanned aerial vehicle imagery and deep learning unet to extract rice lodging,” *Sensors*, vol. 19, no. 18, p. 3859, 2019.
- [21] S. Angara and M. Robinson, “License plate character recognition using binarization and convolutional neural networks,” in *Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC), Volume 1*, Springer, 2020, pp. 272–283.
- [22] A. Calitz and M. Hill, “Automated license plate recognition using existing university infrastructure and different camera angles,” *The African Journal of Information Systems*, vol. 12, no. 2, p. 4, 2020.
- [23] R. D. Castro-Zunti, J. Yépez, and S.-B. Ko, “License plate segmentation and recognition system using deep learning and opencv,” *IET Intelligent Transport Systems*, vol. 14, no. 2, pp. 119–126, 2020.
- [24] C. P. Chen and B. Wang, “Random-positioned license plate recognition using hybrid broad learning system and convolutional networks,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 1, pp. 444–456, 2020.
- [25] P. N. Chowdhury, P. Shivakumara, R. Raghavendra, U. Pal, T. Lu, and M. Blumenstein, “A new u-net based license plate enhancement model in night and day images,” in *Pattern Recognition: 5th Asian Conference, ACPR 2019, Auckland, New Zealand, November 26–29, 2019, Revised Selected Papers, Part I*, Springer, 2020, pp. 749–763.

- [26] A. Danilenko, "License plate detection and recognition using convolution networks," in *2020 International Conference on Information Technology and Nanotechnology (ITNT)*, IEEE, 2020, pp. 1–6.
- [27] A. R. Elkordy and A. S. Avestimehr, "Secure aggregation with heterogeneous quantization in federated learning," *arXiv preprint arXiv:2009.14388*, 2020.
- [28] J. Gandhi, P. Jain, and L. Kurup, "Yolo based recognition of indian license plates," in *Advanced Computing Technologies and Applications: Proceedings of 2nd International Conference on Advanced Computing Technologies and Applications—ICACTA 2020*, Springer, 2020, pp. 411–421.
- [29] H. Huang, L. Lin, R. Tong, *et al.*, "Unet 3+: A full-scale connected unet for medical image segmentation," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 1055–1059. DOI: 10.1109/ICASSP40776.2020.9053405.
- [30] O. Ibitoye, T. Ejidokun, O. Dada, and O. Omitola, "Convolutional neural network-based license plate recognition techniques: A short overview," in *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*, IEEE, 2020, pp. 1529–1532.
- [31] D. M. Izidio, A. P. Ferreira, H. R. Medeiros, and E. N. d. S. Barros, "An embedded automatic license plate recognition system using deep learning," *Design Automation for Embedded Systems*, vol. 24, no. 1, pp. 23–43, 2020.
- [32] Y. Jamtsho, P. Riyamongkol, and R. Waranusast, "Real-time bhutanese license plate localization using yolo," *ICT Express*, vol. 6, no. 2, pp. 121–124, 2020.
- [33] N. H. Jasem and F. Mohammad, "Iraqi license plate recognition system using (yolo) with sift and surf algorithm," *Journal of Mechanics of Continua and Mathematical Sciences*, vol. 15, 2020.
- [34] C. Khosla and B. S. Saini, "Enhancing performance of deep learning models with different data augmentation techniques: A survey," in *2020 International Conference on Intelligent Engineering and Management (ICIEM)*, IEEE, 2020, pp. 79–85.
- [35] I. V. Pustokhina, D. A. Pustokhin, J. J. Rodrigues, *et al.*, "Automatic vehicle license plate recognition using optimal k-means with convolutional neural network for intelligent transportation systems," *Ieee Access*, vol. 8, pp. 92 907–92 917, 2020.
- [36] S. Qin and S. Liu, "Efficient and unified license plate recognition via lightweight deep neural network," *IET Image Processing*, vol. 14, no. 16, pp. 4102–4109, 2020.
- [37] R. Rahman, T. S. Pias, and T. Helaly, "Ggcs: A greedy graph-based character segmentation system for bangladeshi license plate," in *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISM-SIT)*, IEEE, 2020, pp. 1–7.
- [38] W. Riaz, A. Azeem, G. Chenqiang, Z. Yuxi, W. Khalid, *et al.*, "Yolo based recognition method for automatic license plate recognition," in *2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*, IEEE, 2020, pp. 87–90.

- [39] N. Rieke, J. Hancox, W. Li, *et al.*, “The future of digital health with federated learning,” *NPJ digital medicine*, vol. 3, no. 1, p. 119, 2020.
- [40] S. M. Silva and C. R. Jung, “Real-time license plate detection and recognition using deep convolutional neural networks,” *Journal of Visual Communication and Image Representation*, vol. 71, p. 102 773, 2020.
- [41] J.-Y. Sung and S.-B. Yu, “Real-time automatic license plate recognition system using yolov4,” in *2020 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, IEEE, 2020, pp. 1–3.
- [42] S. Usmanhujaev, S. Lee, and J. Kwon, “Korean license plate recognition system using combined neural networks,” in *Distributed Computing and Artificial Intelligence, 16th International Conference*, Springer, 2020, pp. 10–17.
- [43] W. Weihong and T. Jiaoyang, “Research on license plate recognition algorithms based on deep learning in complex environment,” *IEEE Access*, vol. 8, pp. 91 661–91 675, 2020.
- [44] G. Zhu, J. Joo, D. Kim, and S. C. Kim, “License plate recognition with yolo network,” pp. 152–153, 2020.
- [45] P. Ahmad, H. Jin, R. Alroobaea, *et al.*, “Mh unet: A multi-scale hierarchical based architecture for medical image segmentation,” *IEEE Access*, vol. 9, pp. 148 384–148 408, 2021. DOI: 10.1109/ACCESS.2021.3122543.
- [46] L. Burkhalter, H. Lycklama, A. Viand, N. Küchler, and A. Hithnawi, “Rofl: Attestable robustness for secure federated learning,” *arXiv preprint arXiv:2107.03311*, 2021.
- [47] J. Duan and X. Liu, “Online monitoring of green pellet size distribution in haze-degraded images based on vgg16-lu-net and haze judgment,” *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–16, 2021.
- [48] X. Kong, K. Wang, M. Hou, *et al.*, “A federated learning-based license plate recognition scheme for 5g-enabled internet of vehicles,” *IEEE Transactions on Industrial Informatics*, vol. 17, no. 12, pp. 8523–8530, 2021.
- [49] R. Laroca, L. A. Zanlorensi, G. R. Gonçalves, E. Todt, W. R. Schwartz, and D. Menotti, “An efficient and layout-independent automatic license plate recognition system based on the yolo detector,” *IET Intelligent Transport Systems*, vol. 15, no. 4, pp. 483–503, 2021.
- [50] V. Meena Deshpande, “License plate detection and recognition using yolo v4,” *Webology (ISSN: 1735-188X)*, vol. 18, no. 4, 2021.
- [51] T. T. H. Nguyen, A. Jatowt, M. Coustaty, and A. Doucet, “Survey of post-ocr processing approaches,” *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–37, 2021.
- [52] Y. Sarcheshmehpour, M. Leinonen, and A. Jung, “Federated learning from big data over networks,” in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2021, pp. 3055–3059.
- [53] B. Setiyono, D. A. Amini, and D. R. Sulistyaningrum, “Number plate recognition on vehicle using yolo-darknet,” in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1821, 2021, p. 012 049.

- [54] S. M. Silva and C. R. Jung, "A flexible approach for automatic license plate recognition in unconstrained scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 5693–5703, 2021.
- [55] M.-L. Tham and W. K. Tan, "Iot based license plate recognition system using deep learning and openvino," in *2021 4th International Conference on Sensors, Signal and Image Processing*, 2021, pp. 7–14.
- [56] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106775, 2021.
- [57] R. Al-Batat, A. Angelopoulou, S. Premkumar, J. Hemanth, and E. Kapetanios, "An end-to-end automated license plate recognition system using yolo based vehicle and license plate detection with vehicle classification," *Sensors*, vol. 22, no. 23, p. 9477, 2022.
- [58] S. Ganjoo, "Yolo and mask r-cnn for vehicle number plate identification," *arXiv preprint arXiv:2207.13165*, 2022.
- [59] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A review of yolo algorithm developments," *Procedia Computer Science*, vol. 199, pp. 1066–1073, 2022.
- [60] M. Li, F. Ren, Z. Zhang, Y. Long, and J. Zeng, "Chinese license plate recognition algorithm based on unet3+," in *2022 3rd International Conference on Computing, Networks and Internet of Things (CNIOT)*, IEEE, 2022, pp. 60–64.
- [61] S. Pan, J. Liu, and D. Chen, "Research on license plate detection and recognition system based on yolov7 and lprnet," *Academic Journal of Science and Technology*, vol. 4, no. 2, pp. 62–68, 2022.
- [62] S.-H. Park, S.-B. Yu, J.-A. Kim, and H. Yoon, "An all-in-one vehicle type and license plate recognition system using yolov4," *Sensors*, vol. 22, no. 3, p. 921, 2022.
- [63] S. Raj, Y. Gupta, and R. Malhotra, "License plate recognition system using yolov5 and cnn," in *2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS)*, IEEE, vol. 1, 2022, pp. 372–377.
- [64] R. J. Tom, A. Kumar, S. B. Shaik, L. D. Isaac, V. Tripathi, and P. Pareek, "Car license plate detection and recognition using modified u-net deep learning model," in *2022 8th International Conference on Smart Structures and Systems (ICSSS)*, IEEE, 2022, pp. 01–06.
- [65] A. R. Youssef, A. A. Ali, and F. R. Sayed, "Real-time egyptian license plate detection and recognition using yolo," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 7, 2022.
- [66] H. Yu, X. Wang, Y. Shao, F. Qin, B. Chen, and S. Gong, "Research on license plate location and recognition in complex environment," *Journal of Real-Time Image Processing*, vol. 19, no. 4, pp. 823–837, 2022.
- [67] Y. Zou, Y. Zhang, J. Yan, *et al.*, "License plate detection and recognition based on yolov3 and ilprnet," *Signal, Image and Video Processing*, vol. 16, no. 2, pp. 473–480, 2022.

- [68] H. Cao, Y. Wang, J. Chen, *et al.*, “Swin-unet: Unet-like pure transformer for medical image segmentation,” in *Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part III*, Springer, 2023, pp. 205–218.
- [69] H. Shi and D. Zhao, “License plate recognition system based on improved yolov5 and gru,” *IEEE Access*, vol. 11, pp. 10 429–10 439, 2023.
- [70] L. Song, W. Min, L. Zhou, Q. Wang, and H. Zhao, “Vehicle logo recognition using spatial structure correlation and yolo-t,” *Sensors*, vol. 23, no. 9, p. 4313, 2023.
- [71] R. Xie, C. Li, X. Zhou, and Z. Dong, “Asynchronous federated learning for real-time multiple licence plate recognition through semantic communication,” in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2023, pp. 1–5.
- [72] J. Zeng, K. Zhang, L. Wang, and J. Li, “Fedlvr: A federated learning-based fine-grained vehicle recognition scheme in intelligent traffic system,” *Multimedia Tools and Applications*, pp. 1–22, 2023.
- [73] *Bangla LPDB - A* — *zenodo.org*, <https://zenodo.org/record/4718238#.ZBHlmXZBy3A>, [Accessed 18-May-2023].
- [74] *Car License Plate Detection* — *kaggle.com*, <https://www.kaggle.com/datasets/andrewmvd/car-plate-detection>, [Accessed 18-May-2023].
- [75] *Car theft rampant in the city* — *tbsnews.net*, <https://www.tbsnews.net/bangladesh/crime/car-theft-rampant-city-154402>, [Accessed 19-May-2023].
- [76] *Labeled licence plates dataset* — *kaggle.com*, <https://www.kaggle.com/datasets/achrafkhazri/labeled-licence-plates-dataset/code>, [Accessed 18-May-2023].
- [77] R. Rathi, A. Sharma, N. Baghel, P. Channe, S. Barve, and S. Jain, “License plate detection using yolo v4,”