LIST OF ACRONYMS

STT	ACRONYMS	FULLY WORD
1	ALPR	Automatic License Plate Recognition
2	OCR	Optical character recognition
3	CNN	Convolutional Neural Network
4	RNN	Recurrent Neural Network
5	CTC	Connectionist temporal classification
6	HMM	Hidden Markov Models
7	YOLO	You-only-look-once
8	FPS	Frame per second
9	LP	License plate
10	HPE	Human Pose Estimation
11	LSTM	Long Short Term Memory
12	GRU	Gate Recurrent Unit

TABLE OF FIGURES

TABLE OF TABLES

CONTENTS

PREFACE

Reasons for choosing the topic

Nowadays, with the rapid increase of means of transport, the management and control of traffic vehicles in a smart way is attracting more and more attention. Automated vehicle monitoring and tracking systems have been installed in many countries for traffic monitoring tasks.

An automatic vehicle tracking and monitoring system is a system that takes images or videos as input. If the input contains a vehicle, the system will output the content of the number plate in the form of text and proceed to store and trace the vehicle's route on the map at points with traffic cameras. Techniques to solve the given number plate recognition problem such as object detection [1], image processing [2], pattern recognition [3]. Automated vehicle tracking and monitoring systems are designed for outdoor deployment and therefore are vulnerable to environmental and weather influences, such as in rainy or foggy conditions. Including changing the light between day and night. In fact, it is also difficult to process when the input image has many license plates, and the image quality is poor. In addition, the camera's viewing angle for different vehicles, lighting conditions and color nature of the variable, so the vehicle variable will have different sizes, colors, fonts and offsets, making it difficult to solve the problem of recognizing license plates for all scenarios. Moreover, there will be separate standards for the license plates of that country between countries, so it is really difficult to apply a system anywhere in the world. Currently, most license plate datasets shared in the community are foreign license plate data. Therefore, these datasets can’t be applied to train Vietnamese license plate recognition models. Vietnamese license plates include 2-line and 1-line license plates with size, color, font and arrangement are set according to government regulations. Since August 2020, yellow number plates has been applied to vehicles used for business transportation, the plate fonts are also different from those before. [4] [5]

conducted two studies of ALPR systems on Vietnamese license plates; However, their datasets were small and they have not addressed problems of unconstrained environment. Therefore, the techniques to solve this problem are very complex [6] [7].

According to the survey [8] [9], there are many studies that pose a limited scenario in terms of background conditions and limit the distance between the camera and the vehicle. However, this makes it difficult to read the number plates of moving vehicles. To deal with light changes, much research has been focused on developing systems that can operate both day and night, and there is a general assessment that performance at night is often worse than during the day [9]. In addition, the ALPR system must also meet the requirements for operability and connectivity. While deep learning methods can provide high rates of accuracy, they are computationally and resource-intensive to deploy on real devices. While classical computer vision approaches are reasonably accurate in specific scenarios and cost little to compute. The implementation of an ALPR system requires a comprehensive balance between the beneficial elements of the methods.

Stemming from the above reasons, I choose the topic: Building a large-scale car tracking and monitoring system using deep learning models.

Purpose of the study

Researching about license plate recognition technologies, applied to building a large-scale car tracking and monitoring system using deep learning models.

Objects and scope of the study

Objects of the study: In this work, objects of the study are license plate recognition technologies, car tracking and monitoring systems.

Scope of the study: In this work, our system receives images or videos as input and recognize the number of the license plates.

Researching methods

Researching the machine learning and deep learning algorithms used for license plate recognition problem and car tracking and monitoring system.

Doing more experiments to evaluate the algorithms.

Setting and deploying our car tracking and monitoring system in real-world conditions.

Goals of the thesis

The problem of automatic license plate recognition (ALPR) is one of the problems with many practical applications and has attracted many studies in recent years. Most of the previous studies on Vietnamese license plates focused on recognizing license plates when the vehicle entered a parking lot or toll station. Then the angle needed to receive the image from the camera to the number plate area, and the light at the entrance of the vehicle can be arranged to obtain an ideal environment for identification (known as constrained environment). Recent studies have been carried out. However, the data set is small, not diverse, has not covered all cases of license plates in Vietnam and has not solved the challenges for license plate recognition in unconstrained environment. In addition, the problem of non-rectangular plate detection has not been taken seriously, leading to low accuracy of license plate detection. From that fact, in this report, we propose an improved model, which focuses on improving the accuracy of license plate detection and license plate recognition in unconstrained contexts. We also build a dataset of Vietnamese car license plates that are diverse in type, color, collection time, weather conditions and camera placement angle to serve training and testing. To solve non-rectangular object detection problem, the number plate detection part is inspired by the Keypoints detection problem, each corner of the license plate is a key-point. Our results show improvements in license plate detection accuracy based on changes from semantic segmentation architecture

DDRNet [10] with mean IOU $mIOU = 95.01\%$ and precision $P_{75} = 99.5\%$. The license plate optical character recognition (OCR) uses a segmentation free approach based on the encoder decoder network architecture. The main advantage of this approach is that character segmentation is not required. The accuracy in license plate OCR was up to $Acc_{seq} = 99.28\%$ at sequence level and $Acc_{char} = 99.7\%$ at character level.

Our proposed methods have been published to the fourth International Conference on Multimedia Analysis and Pattern Recognition (MAPR) supported by VAPR (Vietnamese Association on Pattern Recognition) was held in Ha Noi on October 15-16, 2021. The title of our paper is "*an efficient method to improve the accuracy of Vietnamese vehicle license plate recognition in unconstrained environment*" [11]. We received an award as the best paper in all of the conference papers. These showed that our proposed methods were so prominent for solving ALPR problem.

A large-scale car tracking and monitoring system has the function of tracking and monitoring vehicles based on the traffic camera system in different roads and conditions. The system is deployed with client-server architecture.

The main contributions in this work are:

First, we propose model and build a complete large-scale automotive tracking and monitoring system for practical deployment. The system is integrated with digital map technology, motion detection algorithm, object tracking, and checks whether the identification results are in accordance with the regulations of the Vietnamese number plate issued by the government.

Second, regarding non-rectangular object detection problem, we propose an approach that is based on Keypoints detection. We modify the semantic segmentation architecture DDRNet [10] to match the addressed problem.

Third, our system identifies plate based on segmentation-free approach integrated with Attention mechanism [12]. Feature extraction part is designed based on VGG-19 [13] in more fine-grained level.

Fourth, we build the MTAVLP dataset that includes 15571 photos of vehicles containing license plates and 16012 photos of Vietnamese license plates. This dataset is large and various in image capturing conditions.

To complete the project, I would like to thank the teachers in the Faculty of Information Technology - Military Academy of Engineering for teaching me knowledge about general subjects as well as specialized subjects, helping me had a solid theoretical foundation and facilitated me throughout my studies. In particular, I would like to express my deep gratitude to Ph.D. Nguyen Quoc Khanh, who directly guided, cared, enthusiastically instructed and created all favorable conditions during the time I completed this project. Finally, I would like to thank my family and friends for always facilitating, caring, helping and encouraging me during my study and completion of my graduation thesis.

Despite many efforts, due to limited time and limited knowledge, mistakes are inevitable. Therefore, I look forward to receiving suggestions and guidance from teachers to make my project more complete.

I sincerely thank!

Chapter 1

BACKGROUND KNOWLEDGE.

1.1. Convolutional neural networks (CNN)

A Convolutional neural network (CNN) is a neural network that has one or more convolutional layers and are used mainly for image processing, classification, segmentation and also for other auto correlated data.

A convolution is essentially sliding a filter over the input. Rather than looking at an entire image at once to find certain features it can be more effective to look at smaller portions of the image.

1.1.1. Common uses for CNNs

The most common use for CNNs is image classification, for example identifying satellite images that contain roads or classifying hand written letters and digits. There are other quite mainstream tasks such as image segmentation and signal processing, for which CNNs perform well at. CNNs have been used for understanding in Natural Language Processing (NLP) and speech recognition, although often for NLP Recurrent Neural Nets (RNNs) are used. A CNN can also be implemented as a U-Net architecture, which are essentially two almost mirrored CNNs resulting in a CNN whose architecture can be presented in a U shape. U-nets are used where the output needs to be of similar size to the input such as segmentation and image improvement.

More and more diverse and interesting uses are being found for CNN architectures. An example of a non-image based application is “The Unreasonable Effectiveness of Convolutional Neural Networks in Population Genetic Inference” by Lex Flagel et al. This is used to perform selective sweeps, finding gene flow, inferring population size changes, inferring rate of recombination. There are researchers using CNNs for generative models in single cell genomics for disease identification. CNNs are also being used in astrophysics to interpret radio telescope data to predict the likely visual image to represent the data.

Deepmind's WaveNet is a CNN model for generating synthesized voice, used as the basis for Google's Assistant's voice synthesizer.

1.1.2. The core of CNN: the convolution

The core of a CNN is the convolution, but what's a convolution? In mathematics a convolution is a process which combines two functions on a set to produce another function on the set. We can think of images as two-dimensional functions. Many important image transformations are convolutions where you convolve the image function with a very small, local function called a "kernel." The kernel slides to every position of the image and computes a new pixel as a weighted sum of the pixels it floats over, in order to process the spatial structure of the image.

Convolutions are used to detect simple patterns such as edge detection in an input image. For example, we can detect edges by taking the values −1 and 1 on two adjacent pixels, and zero everywhere else. That is, we subtract two adjacent pixels. When side by side pixels are similar, this gives us approximately zero. On edges, however, adjacent pixels are very different in the direction perpendicular to the edge.

1.1.3. CNNs multilayered architecture

A CNN has a multilayered architecture where an image is used as an input and has multiple layers of convolutions which creates feature maps. In inner layers, a pooling process is performed in order to reduce the resolution of the subsequent feature maps and increase their quantity. The end layer of feature maps is passed through a linear classifier where a one dimensional array called output layer is created containing the tags the image could be associated with. When a pixel in this one dimensional array is activated it means the image was classified as its associated tag.

CNNs can detect complex patterns such as handwritten numbers because the architecture can perform detections over the previous detections (The input of an

inner layer is the output from the previous one). How do we define which convolutions to apply to solve a specific problem? The neural network does it during the training process where we have a set of training images with a tag/category associated with each one of these. Then the neural network decides which are the best convolutions combinations to apply in order to classify the training dataset successfully.

1.1.4. The future of CNNs

Ever since the breakthrough of CNN in 2012, the evolution in Deep Learning techniques has accelerated exponentially, ranging from image classification models used by self-driven vehicles to text generation models like GPT-3 that produces human-like texts. For that reason, we can expect that Deep Learning tools will continue evolving in the next 5 to 10 years to a point this kind technology will be widely democratized and become a standard tool for every person, having a ubiquitous presence in our daily lives.

1.2. Recurrent Neural Networks

1.2.1. What is RNN?

Humans don’t start their thinking from scratch every second. As you read this essay, you understand each word based on your understanding of previous words. You don’t throw everything away and start thinking from scratch again. Your thoughts have persistence. Traditional neural networks can’t do this, and it seems like a major shortcoming. For example, imagine you want to classify what kind of event is happening at every point in a movie. It’s unclear how a traditional neural network could use its reasoning about previous events in the film to inform later ones. Recurrent neural networks address this issue. They are networks with loops in them, allowing information to persist.

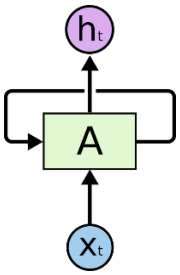

Figure 1.1. A RNN have loops.

In the above diagram, a chunk of neural network, A, looks at some input x_t and outputs a value h_t. A loop allows information to be passed from one step of the network to the next. These loops make recurrent neural networks seem kind of mysterious. However, if you think a bit more, it turns out that they aren't all that different than a normal neural network. A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor. Consider what happens if unroll the loop:

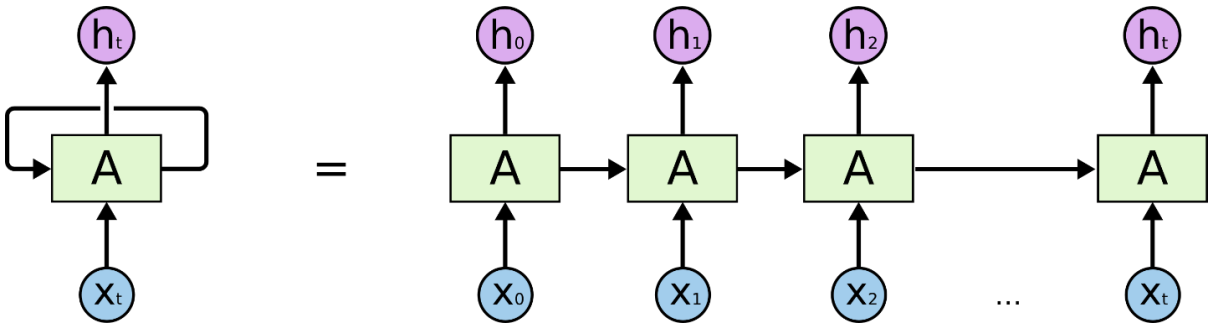

Figure 1.2. An unrolled recurrent neural network.

This chain-like nature reveals that recurrent neural networks are intimately related to sequences and lists. They're the natural architecture of neural network to use for such data. And they certainly are used! In the last few years, there have been incredible success applying RNNs to a variety of problems: speech recognition, language modeling, translation, image captioning… The list goes on.

Essential to these successes is the use of "LSTMs" and its variant a very special kind of recurrent neural network which works, for many tasks, much better than the standard version. Almost all exciting results based on recurrent neural

networks are achieved with them. It's these LSTMs and its variant that this essay will explore.

1.2.2. The Problem of Long-Term Dependencies

One of the appeals of RNNs is the idea that they might be able to connect previous information to the present task, such as using previous video frames might inform the understanding of the present frame. If RNNs could do this, they'd be extremely useful. But can they? It depends.

Sometimes, we only need to look at recent information to perform the present task. For example, consider a language model trying to predict the next word based on the previous ones. If we are trying to predict the last word in "the clouds are in the sky," we don't need any further context – it's pretty obvious the next word is going to be sky. In such cases, where the gap between the relevant information and the place that it's needed is small, RNNs can learn to use the past information.

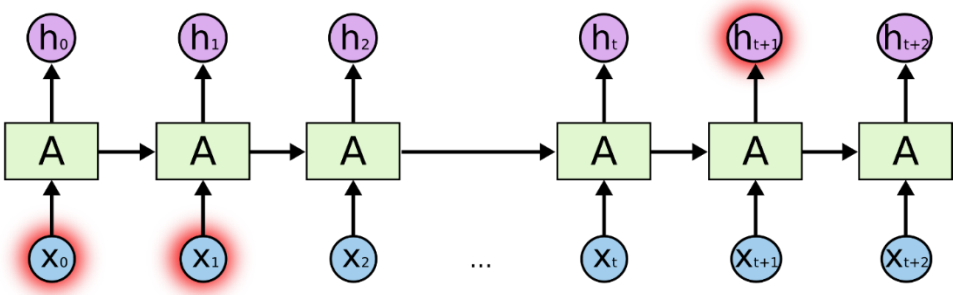

Figure 1.3. Long-term dependencies.

But there are also cases where we need more context. Consider trying to predict the last word in the text "I grew up in France… I speak fluent French." Recent information suggests that the next word is probably the name of a language, but if we want to narrow down which language, we need the context of France, from further back. It's entirely possible for the gap between the relevant information and the point where it is needed to become very large.

Unfortunately, as that gap grows, RNNs become unable to learn to connect the information. In theory, RNNs are absolutely capable of handling such “long-term dependencies.” A human could carefully pick parameters for them to solve toy problems of this form. Sadly, in practice, RNNs don’t seem to be able to learn them.

Thankfully, LSTMs and its variant don’t have this problem!

1.2.3. LSTM Networks

Long Short Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies. LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn! All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

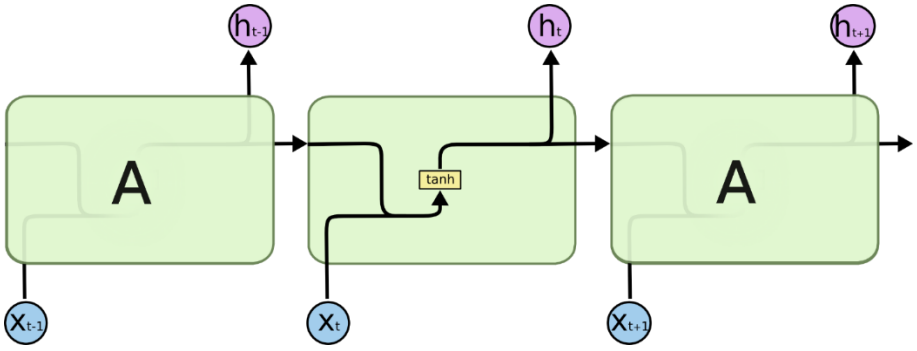

Figure 1.4. The repeating module in a standard RNN contains a single layer.

LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

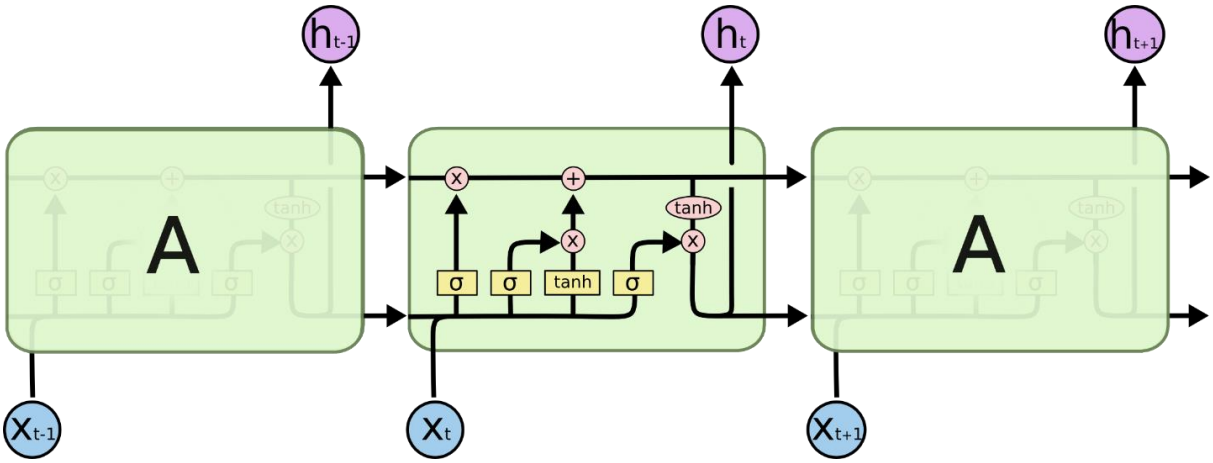

Figure 1.5. The repeating module in an LSTM contains four interacting layers.

In the above diagram, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denote its content being copied and the copies going to different locations.

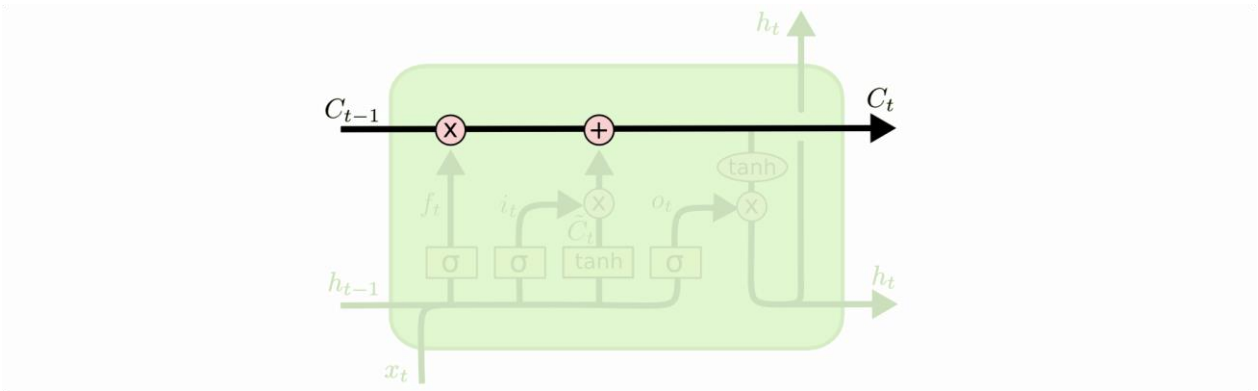

Figure 1.6. Cell state in LSTM.

The key to LSTMs is the cell state, the horizontal line running through the top of the diagram.

The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It’s very easy for information to just flow along it unchanged.

The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates. Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.

The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means “let nothing through,” while a value of one means “let everything through!”. An LSTM has three of these gates, to protect and control the cell state.

1.2.4. Variants on Long Short Term Memory

In fact, it seems like almost every paper involving LSTMs uses a slightly different version. The differences are minor, but it’s worth mentioning some of them.

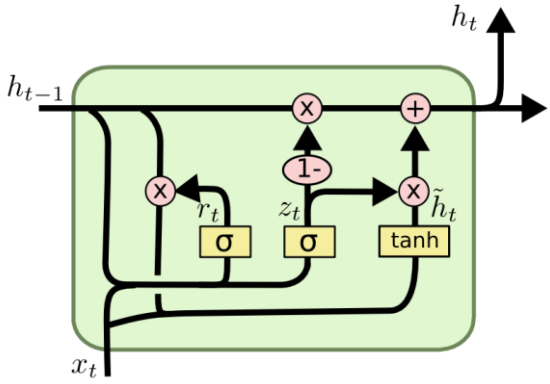

Figure 1.7. A GRU cell.

A slightly more dramatic variation on the LSTM is the Gated Recurrent Unit, or GRU, introduced by [14]. It combines the forget and input gates into a single “update gate.” It also merges the cell state and hidden state, and makes some other changes. The resulting model is simpler than standard LSTM models, and has been growing increasingly popular.

1.3. RNNs for sequence to sequence problem.

In a neural network, you have one output per input. You have an image and you have a label, for instance. There is no way you can input image after image

on Neural Networks and get an output based on all of them. The nature of neural networks makes them unable to process sequential data.

On the other hand, RNNs work very well with sequential data. They have a mechanism of "remembering" the previous inputs and producing an output based on all of the inputs. This makes them well-suited for sequential type of data such as text, audio, video or any time-series data.

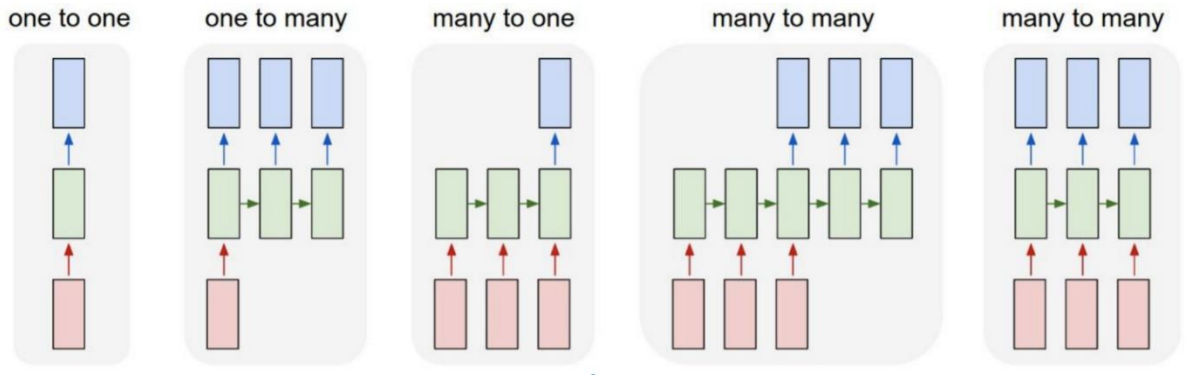

Figure 1.8. RNNs in each sequence to sequence problem.

Figure 1.8 shows five variations of RNNs. Red vectors are inputs whereas blue vectors are the outputs. First one is a one input to one output situation. Which is essentially a neural network. The others are capable of sequential inputs and outputs, such as:

One to many: image -> caption sentence

Many to one: sentence -> sentiment (positive / negative label)

Many to many: a sentence in English -> a sentence in Turkish

The other many to many: frames of video -> coordinates of bounding boxes around an object

Sequence to Sequence Machine Translation is one of the most famous sequence to sequence problems that RNNs can tackle.

RNNs are also capable of doing natural language translation, aka. machine translation. It involves two RNNs, one for the source language and one for the target language. One of them is called an encoder, and the other one decoder. The reason is that, the first one encodes the sentence into a vector and the second one converts the encoded vector into a sentence in target language.

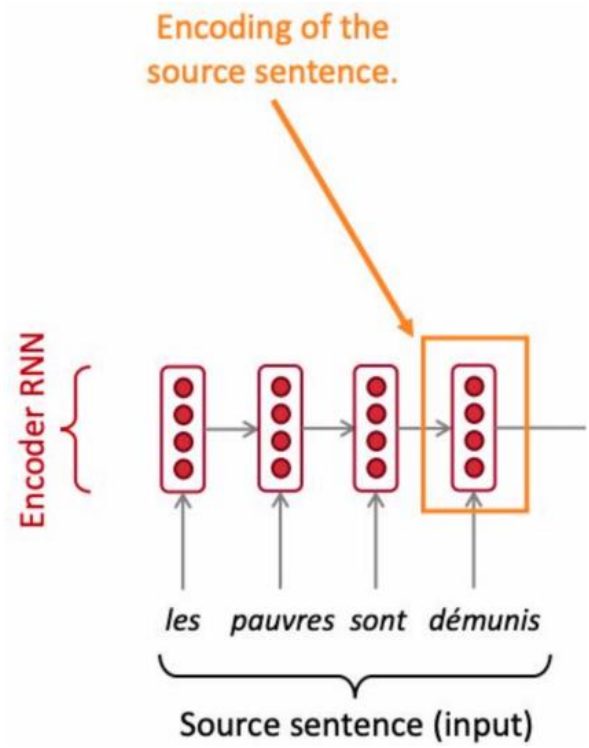

Figure 1.9. Encoder using an RNN.

The encoder is an RNN. Given the source sentence, it produces an encoding. Notice the benefit of RNNs here. The size of input sentence can be any size.

The decoder is a separate RNN. Given the encoded sentence, it produces the translated sentence in target language. Notice the purple, dashed arrows. Every predicted word is an input to the next prediction. This is the recurring nature of RNNs and it is very interesting to me that they can do a significant work such as language translation by being connected like that.

At each step of the decoder, given the hidden state and input word vector as input, the argmax of a probability distribution of words is chosen as the most

probable word. This is called greedy decoding. The problem with that is, the decoder is going one by one. At one point, if the sentence does not make sense, it cannot go back.

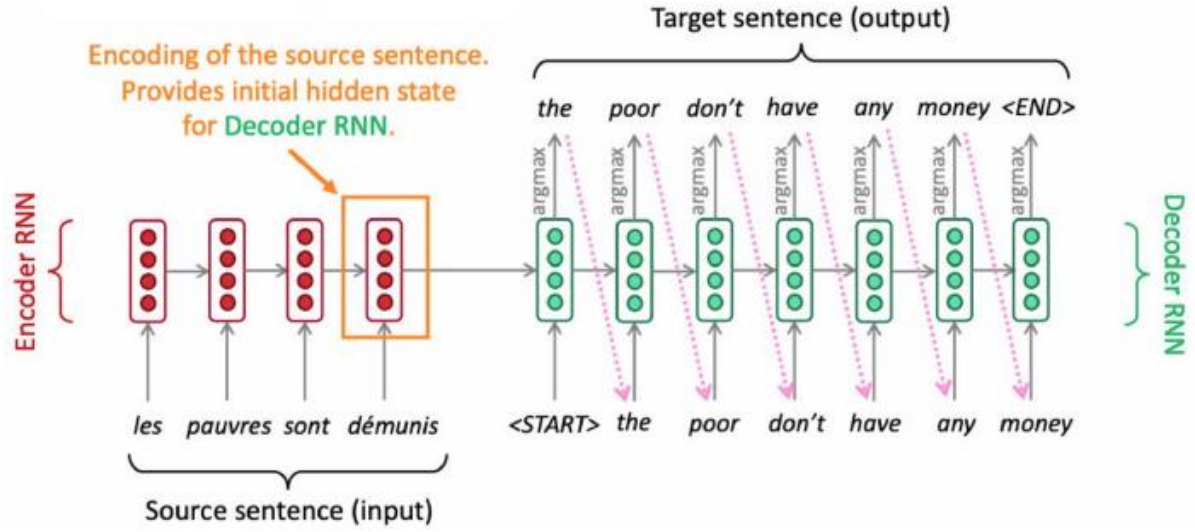

Figure 1.10. The decoder is also a separate RNN.

In addition, the whole sentence being encoded in a single vector is hard to translate into a sentence. After the encoding step, it is essentially a mapping from sentence to a vector. It really hard for the decoder to derive an order for the words in the target language. That's where attention comes into play.

1.4. Attention mechanism.

Attention lets the decoder to focus on specific parts of the input sentence for each output word. This helps the input and output sentences to align with one another.

In the picture, the blue ovals are simply dot products with the sentence encoding and hidden states.

For the French sentence above, the highest attention score belongs to "les". This will cause the decoder to start the translated sentence with "the". When I first saw that diagram, I thought that, if the correct translation started with "pauvres", there is no way the decoder would get it right. However, since this neural network

is supposed to be trained end-to-end, it will also learn to produce correct attention scores along the way.

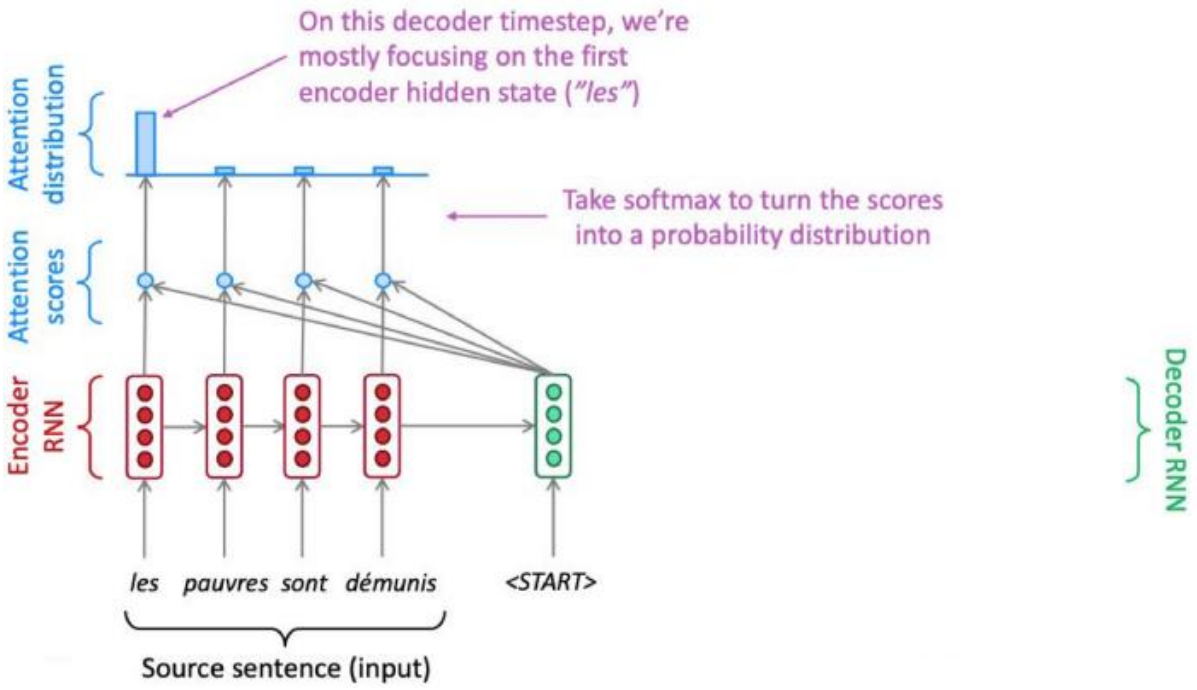

Figure 1.11. A simple use of Attention mechanism.

1.5. Digital map

1.5.1. Concept of digital map

A digital map is an organized collection of map data on a computer-readable device and represented as a map image. Digital maps include the following basic components: Data logging device, Calculator, Database, Map display device.

Digital maps are organized and stored compactly, different from traditional maps in that: Digital maps are only data files recorded in computer memory and can be presented in the same image as traditional maps. system on the computer screen. If using plotters, we can print the map on paper just like a regular map.

Thanks to computers with the ability to store large amounts of information, the ability to synthesize, update, analyze information and process rich map data, digital maps are widely and more widely applied. much more than paper maps.

1.5.2. Google Map API

The Google Maps Api is a gateway to existing Google Map features for developers to develop third-party applications.

1.5.3. Direction API

Directions API is a service that calculates directions between locations. Directions can be searched for several modes of transportation, including vehicle, driving, walking, or biking. The API returns the most efficient routes when calculating directions. Travel time is a major factor in optimization, but the API can also take other factors into account such as distance, number of turns, and more when deciding which route is most efficient.

1.5.4. GMap.Net Platform

GMap.NET is a great and powerful, free, cross-platform, open-source, .NET-controlled tool.

GMap.NET allows routing, geocoding, directions and supports map types from Google, Yahoo!, Bing, OpenStreetMap, ArcGIS, Pergo, SigPac, Yandex, Mapy.cz, Maps.lt, iKarte.lv, NearMap, OviMap, CloudMade, WikiMapia, MapQuest in Windows Forms & Presentation, support caching and run on mobile platforms!

A few things to understand when using this map:

+ GMapControl: is the control that displays the map.

+ GmapOverlay: is a layer on top of the map control. You can have several layers on top of the map, each representing a route with stops, a list of stores, etc.

+ GMapMarker: are points on a layer, each point represents a specific geographical location (Lat, Lon) for example each drop point on a route.

+ GMapRoute: is the path or direction between two or more points.

Some properties of GmapControl:

- CanDragMap = true: user can drag the map by using right mouse button.

- MarkersEnabled = true: any defined markers will be displayed. Otherwise, they will not appear.

- PolygonsEnabled = true: any defined polygon will be displayed.

- ShowTileGridLines = true: GMap.NET will display grid-style coordinates

- Zoom, MinZoom, MaxZoom - Zoom level for Google Maps from 0 (zoom out to global level) to 18 (zoom to street level). Zoom is the current zoom level (5 would be fine for the country level), while MinZoom and MaxZoom must be set to 0 and 18 respectively if users want to be able to fully zoom in and out.

1.6. Conclusion for chapter 1

In this chapter we have covered some background techniques that used in our car tracking and monitoring system.

Chapter 2

CAR LICENSE PLATE RECOGNITION

2.1. Related work of car license plate recognition problem.

There are two main approaches for ALPR: multi-stage and single-stage license plate recognition.

2.1.1. Multi-stage license plate recognition systems

Most of the existing solutions for the ALPR task have considered the multi-stage method, which consists of three main steps. The first stage is the license plate detection or extraction. Existing algorithms use traditional computer vision techniques and deep learning methods with object detection to locate the license plate in an image. Traditional computer vision techniques are mainly based on the features of the license plate such as shape, color, symmetry, texture, etc. In the second stage, the license plate is segmented and the characters are extracted using some common techniques such as mathematical morphology, connected components, relaxation labelling, and vertical and horizontal projection. However, the character segmentation stage is not necessarily performed in every multi-stage ALPR system, because there are some segmentation-free algorithms in which this stage is omitted.

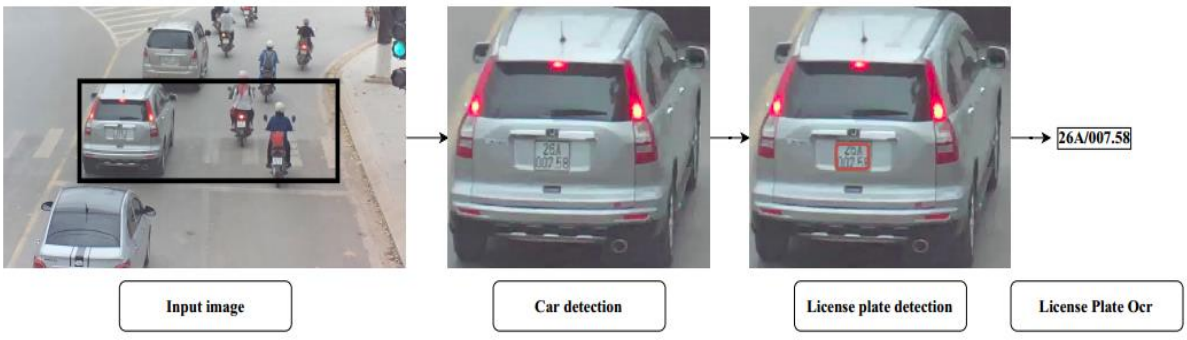

Figure 2.1. Main stages in a multi-stage license plate recognition system

The final stage is the recognition of the characters using pattern matching techniques or classifiers like neural networks and fuzzy classifiers. However, the

main downside of separating detection from recognition is its impact on the accuracy and efficiency of the overall recognition process. This happens mainly due to the imperfection of the detection process such as flaws in the bounding box prediction. For instance, if the detection process misses a part of a license plate, it will affect to reduce the overall accuracy of the recognition process. Thus, in a multi-stage approach, it is important to achieve satisfying results in each stage. Figure 2.1 shows the main processing stages in a multi-stage plate recognition system.

2.1.1.1. License plate detection

The general definition of a license plate is “a metal or plastic plate attached to a vehicle that helps to identify them uniquely”. Yet, this definition is not comprehended by a machine. In order to detect a license plate, a definition that can be understood by a machine is required. Considering its features, the definition for a license plate can be stated as “a rectangular area of a vehicle with a high density of horizontal and vertical edges”. Based on these features, many algorithms have been put forward to solve the license plate detection task and some of them are rooted in traditional computer vision techniques and some in deep learning. Bellow table shows a classification of the license plate detection techniques that are used in existing methods.

Table 2.1 Compares those localization methods along with the benefits and the limitations of each method.

Technique	Advantages	Limitations
Edge-based methods	Simple, faster	Sensitive to unwanted edges, not suitable to be applied with blurry and complex images
Color-based methods	Robust against deformations and rotations	Sensitive to illumination changes
Texture-based methods	Robust against boundary deformations in the license plates	Computationally complex, work poorly with complex backgrounds and illumination changes
Character-based methods	Can be used for rotated plates	Time consuming and not suitable for images with any other texts
Statistical classifiers	Accurate, robustness to environmental changes, efficient	Computationally complex
Deep learning based	Efficient, robustness to environmental changes	Resource consuming, computationally complex

Edge-based methods

Considering that every license plate is rectangular and has a known aspect ratio, most of the studies have relied on edge-based approaches for license plate detection. Since the license plate color is distinct to the color of the vehicle body, the boundary of the license plate appears as an edge in the image. There are two kinds of edges in an image as horizontal and vertical edges. When two horizontal edges are concerned it is known as horizontal edge detection and for two vertical edges, it is called vertical edge detection. Several studies have used Sobel filter

for edge detection. The Sobel filter has two 3×3 convolutional matrices and one is dedicated for vertical and the other for horizontal edge detection. Ease of use is one of the key advantages of this method. One of the major drawbacks of this approach is its responsiveness to noise.

In [3] [15] Luo et al. and Sarfraz et al. have used a Sobel filter to extract only the vertical edges. However, using only the horizontal lines leads to erroneous results as the vehicle bumper can be mistaken as a horizontal edge of the license plate. Sarfraz et al. [3] proposed a method to generate some candidate rectangles in the image using the vertical edge detection. The candidate rectangle that has the same aspect ratio as the license plate was then detected as the plate. This method has claimed a success rate of 96.2% even under different illumination conditions.

Heo et al. [16] have developed a new method for license plate extraction with the combination of a line-grouping algorithm and an edge-density mapping algorithm. The line-grouping algorithm extracts the line segments and groups the line segments at the license plate boundary, while the edge-density map detects the region that is highly dense with lines as the candidate plate. They have shown a success rate of 99.45% for the new algorithm.

In the traditional approach, Hough Transformation is used to identify the straight lines in the images. This approach gives an advantage of detecting lines up to an inclination of 30°. Yet, the method consumes more time and memory. Also, Hough transformation is highly sensitive to boundary deformations. As a result, it is not used when there are any unclear boundaries in the license plate. Therefore, Duan et al. [17] have proposed a new method combining Hough transformation with a contouring algorithm and achieved faster results and higher accuracy of 98.8%. Many studies have used edge-based approaches for license plate detection as they are simple and faster. However, these methods are highly

sensitive to unwanted edges and not suitable to be used with blurry and complex images.

Table 2.2. Comparison of license plate localization methods

Technique	Advantages	Limitations
Edge-based methods	Simple, faster	Sensitive to unwanted edges, not suitable to be applied with blurry and complex images
Color-based methods	Robust against deformations and rotations	Sensitive to illumination changes
Texture-based methods	Robust against boundary deformations in the license plates	Computationally complex, work poorly with complex backgrounds and illumination changes
Character-based methods	Can be used for rotated plates	Time consuming and not suitable for images with any other texts
Statistical classifiers	Accurate, robustness to environmental changes, efficient	Computationally complex
Deep learning based	Efficient, robustness to environmental changes	Resource consuming, computationally complex

Colour-based methods

Colour-based methods rely on the fact that the colour of a license plate is different from the background colour of the vehicle. Also, the colour combination of the plate and its characters is nowhere found in the image other than the plate region. The Hue, Lightness, and Saturation (HLS) colour model can be used to classify the pixels in an input image against different illuminations. Unlike the

Red, Green, and Blue (RGB) model, the HLS model has classified pixels into 13 categories and was tested on Chinese license plates. Yet, the HLS model is sensitive to the noise. Few studies have used the Genetic Algorithm as a search heuristic for identifying the license plate colour.

Zhang et al. [18] have proposed a novel algorithm called Gaussian Weighted Histogram Intersection (GWHI) for license plate classification. Histogram intersection is used to match two colours by matching their colour histograms. The sensitivity to illumination is the major problem in conventional histogram intersection methods. Therefore, they have added a Gaussian function as a modification to the conventional histogram intersection.

The colour-based methods can be used to detect deformed or inclined license plates. Yet, they are rarely used alone for plate detection as they are sensitive to illumination changes. Also, they depend on the specifications of the camera that is used to capture the images. Besides, they make erroneous results if the image contains other regions with the same colour as of the license plate. Therefore, these methods are often combined with some other technique to achieve accurate results.

Texture-based methods

Texture-based methods use the presence of characters on the license plate as the base for plate detection. Due to the significant colour difference between the plate and its characters, it creates a frequent colour transition on the license plate. Hence, if the image is grey-scaled there is a notable difference between the characters and the plate background. Thus, it creates a unique pixel intensity distribution around the plate region. Moreover, the colour transition makes the plate region to have a high edge density.

In texture analysis, the Gabor filter is often used. A key advantage of using a Gabor filter is its ability to analyze texture in infinite directions and scales. Hsieh et al. [19] have presented a method based on the Wavelet Transform for license

plate detection. As shown in Figure 4 there are four sub-bands (or sub-images) in Wavelet Transform namely LL, HL, LH, and HH where H and L stand for high and low frequencies, respectively. The LL sub-band contains the original image passed through a low-pass filter and the rest contain the missing details. HL sub-band has the characteristics in the vertical direction, while LH has the characteristics in the horizontal direction. The method described in [19] was consist of three stages. In the first stage, the Wavelet transform is performed on a binary image using a Haar-scaling function. In the next stage, a reference line with the maximum horizontal variation is found with the help of the LH sub-band. Then the reference line is used to extract the candidate regions and finally, the license plate is accurately located from the extracted candidates. The reported detection accuracy in 92.4%.

All the texture-based methods are robust against license plate deformation and it is a key advantage of using these methods. Still, these methods involve complex computations and work poorly with complex backgrounds and different illumination conditions.

Character-based methods

Examining an image for the presence of characters and locating them is also used for license plate detection. These methods are categorized as character-based methods and consider the region with characters as the possible plate region. Matas and Zimmermann [20] have proposed a method that extracts all character-like regions in an image. A neural network classifier is then used to classify those extreme regions and if any linear spatial configuration is found, it is assumed as the possible region with the license plate. This method is claimed to be robust against different illumination conditions and viewpoints and the reported detection accuracy is 95%.

In [21], Draghici has scanned the image horizontally to detect any repeating contrast changes with a minimum of 15 pixels in length. This approach is made

under three main assumptions; (1) the contrast between the background and the characters is sufficiently good, (2) the license plate contains at least 3-4 characters, (3) the minimum vertical size of a character is 15 pixels. However, the minimum vertical size measure depends on the camera specifications and has to be calibrated for any hardware change. The reported detection rate for this approach is 99% in outdoor conditions.

Moreover, character-based methods are advantageous due to their robustness to license plate rotations. However, these methods are time-consuming and often prone to errors if there are other texts in the input image.

Statistical classifiers

Several studies have used Haar-like features with Adaptive Boosting (AdaBoost) to train cascade classifiers for license plate detection [22]. In [22], a decision tree based cascaded classifier with AdaBoost training is proposed. They used a combination of statistical and Haar-like features for training since statistical features simplify the process. This enhanced algorithm has achieved a detection rate of 94.5% under different illumination conditions and viewpoints.

Kim et al. [23] have proposed a new algorithm based on colour texture for object detection and demonstrated with a license plate localization system. They have extended the previous studies on texture classification by following a Support Vector Machine (SVM) based approach for identifying plate regions. The SVM has used to classify a region as a license plate or non-plate using its colour and texture properties. The SVM-based method was significantly robust and efficient when compared to the previous approaches for texture classification. After the classification stage, each pixel indicates a probability or a score for it being a part of the plate region and these scores are used to predict the bounding box by applying the continuously adaptive mean-shift algorithm (CAMShift). This combination of SVM and CAMShift has provided a high detection rate with efficient processing.

Deep-learning techniques

According to the recent development in computer vision approaches, most of the statistical methods have been replaced by deep learning neural networks due to their high accuracy in object detection. Embracing this fact, many studies in license plate detection have used different types of neural networks. In [24], Selmi et al., have presented a localization method using a Convolutional Neural Network (CNN). In their study, they have followed two major steps in the license plate detection stage. In the first stage, preprocessing techniques were applied to the input image to remove the noise and extracted the finer elements or details from the image. In the next stage, possible bounding boxes for the plate region have extracted and distinguished as a license plate or non-plate using a CNN classifier. The experiment was tested on the Caltech data set and the recall and the f-score accuracy reported is 93.8% and 91.3%, respectively.

The success of state-of-the-art, real-time object detectors such as You-only-look-once (YOLO) [25], have inspired the license plate detection process in many recent ALPR studies. Several studies have used the state-of-the-art YOLO object detector for license plate detection. In [26], Laroca et al., have used YOLO (version 2) object detector to build an efficient and robust system for localization. They have used two separate CNNs for vehicle detection and license plate detection. This study has shown promising results over existing methods for both SegPlate (SSIG) dataset and Federal University of Parana (UFPRALPR) dataset.

The Scene Text Spotting (STS) methods mainly focus on font, lexical and semantic information. There are several word processing methods that can be applied to detect license plates with different offset angles. More recently, research [27] presented an approach to text detection called the Instance Transformation Network (ITN), which can theoretically handle license plate rotations, but only resulting in a bounding box that surrounds the text rather than the bounding corners of the text.

2.1.1.2. License plate optical character recognition.

As the second stage in a multi-stage automated license plate recognition pipeline, this stage is responsible for "reading" the license plate ones the detection stage has localized it. This is a specific case of optical character recognition that considers certain features in the license plate. For instance, many countries have strict regulation regarding the font and color of the license plate and usually, they are selected to be easy to read. However, there are some unique issues associated with the license plates. For instance, since the image is taken outdoors, the system designers have to consider aspects such as variable ambient light, uneven brightness, effect of weather. Despite having a standard license plate, they still could be damaged or rotated. In general, license plate OCR has two main approaches which are: segmentation-based and segmentation-free approach.

Segmentation-based approach

Several pre-processing tasks performs before the character segmentation and recognition to handle unique challenges in license plate recognition. For example, rotation techniques such as bilinear transformations, least square-based methods and line fitting methods have been used in related studies. In many classical machine vision-based techniques for character recognition, the image is binarized before segmentation. The process makes it easier to separate the pixels belong to the characters in the image, compared to grey-scale or colour images. However, the threshold for this binarization must be determined correctly, to avoid the combining of the characters or merging with the license plate frame in the binary image, which makes it difficult to segment. The threshold value can be defined using image enhancement techniques such as noise removal, histogram equalization, contrast enhancement and grey level transformation. However, even with these enhancements, it can be hard to get a single threshold. In such cases, adaptive binarization techniques such as local thresholding and Niblack's binarization method (NBM) have used.

Table 2.3. Comparison of character segmentation methods.

Technique	Advantages	Limitations
Pixel connectivity	Simple, robust to rotation	Not suitable to be applied with joined or broken characters
Projection profiles	Robust to rotation, independent of character positions	Sensitive to noise and font changes, number of characters in the license plate should be known
Using prior knowledge about the license plate	Simple	Specify to the regions where they were designed to operate
Deep neural networks	Reduce the amount of parameters and the computational cost	

In many optical character recognition techniques, the characters are first segmented before the classification. License plate character segmentation techniques consider the attribute of having contrasting colours for the background and characters. Binarization of the image makes this separation easier, as the foreground (character) and background pixels get the opposite "colours" in binarization. Table 2.3 compares the existing methods for segmentation by considering their advantages and limitations in use.

A. Character segmentation

a. Character segmentation using pixel connectivity

Pixel connectivity is a simple method for character segmentation [28]. Here, the connected pixels are labelled and if the pixels of the same label for an object of predetermined size or aspect ratio, then they are extracted as a character. One problem with pixel connectivity based methods is that they fail with broken

characters or when characters are joined due to binarization threshold selection. However, pixel connectivity based methods are robust against rotated license plates and relatively simple to implement. Lack of need for pre-processing to compensate for license plate rotation further simplifies the license plate recognition pipeline.

b. Character segmentation using projection profiles

Projection profiles methods use the fact that having opposite colours for the character and background pixels in the license plate image after image binarization. Typically, vertical projections are used to detect the starting and ending positions of the character and then horizontal projects are used to extract the character [9]. However, project-based techniques are sensitive to image quality and image noise. As a result, a denoising stage has to be included in the pre-processing stage of the recognition pipeline. Although these methods give low robust values compared to pixel connectivity based methods for rotations, the projection-based methods are still robust to rotations and independent on character positions.

c. Character segmentation using prior knowledge

Prior knowledge about the license plate such as the aspect ratio of the characters, the ratio of various coloured pixels in the image is used for character segmentation. For instance, a simple approach was used by Paliy et al. [29], by resizing the extracted license plate to a fixed size, where they had predetermined the position of characters. Approaches based on prior knowledge tend to be simple to implement; however, these methods are usually specific to the regions where they were designed to operate and do not generalize in other instances.

d. Character segmentation using deep learning

Neural network has become a recent approach for character segmentation, which uses CNN for the task associated with computer vision. A localized license

plate is given as the input for the CNN and the bounding boxes of each character are produced as the output. However, depending on the dataset, the CNN execution consumes more time and resources compared to the traditional computer vision-based techniques. In addition, several deep learning-based license plate recognition pipelines have omitted explicit character segmentation in favor of implicit character segmentation in later stages, which leads to reduce the number of parameters and the computational cost [30].

B. Character recognition

Many classification techniques require fixed-size inputs to the learning model. Since the output from the segmentation stage is vary in size, the input segments are re-scaled before the classification. Because the number of characters, their relative position and possible values are known in most cases, each segment is classified individually to be of one of the possible values. This can be considered in three cases, (1) Compare all the pixel values of the raw image data directly with the predefined templates, (2) Use different image processing and machine learning techniques to extract features before classifying the segments, (3) Use deep learning techniques to classify segments.

a. Template and pattern matching techniques

Given that license plate usually have a known font and character size, one popular option is to use template matching techniques to classify characters [3]. Template matching is typically used with binarized images. For each possible character, a predefined template is created and each segment is matched with each template to find the most similar template. Here, the similarity is measured using metrics such as Mahalobian distance, Jaccard value, Hausdorff distance, Hamming distance and normalized cross-correlation. These methods are simple to implement but do not generalize well for different types of license plates due to their nature. Besides, this technique is difficult to use if there are several possible templates in terms of typography. In order to handle the rotations in

characters, additional templates have to be stored which further increases computation time and processing memory.

b. Character recognition using feature extractors

In general, all the pixels are not required to recognize a character. Thus, feature extractors are used to distinct simpler features from the images by reducing the computational costs. Some feature extraction techniques can extract features, which are robust against rotations and image noise. In these methods, a feature vector is generated using a transformation on each segment and then classifies using a machine learning model. Some of the feature extractors techniques are eigenvector transformation [31]. Machine learning models such as SVM and Hidden Markov Models (HMM) are used to classify the extracted features.

c. Character recognition using deep learning

Advantage of using neural networks is that they can be given the raw pixel data directly and act as both feature extractors and classifiers on their own. Various forms of neural networks ranging from simple multi-layer perceptron to Probabilistic Neural Networks (PNN) [32] and discrete-time cellular networks have been used for this task. However, many recent studies have used CNN, which have shown great potential in many computer vision tasks. Another recent approach is to directly use object detection based techniques such as YOLO. Although deep learning-based approaches are relatively computationally expensive than the alternative methods like template matching and statistical feature extractors, they provide better accuracy in general.

In segmentation-based approaches, characters are segmented and then classified. Techniques for character segmentation are projection profiles methods, using pixel connectivity, and prior knowledge. These techniques are highly influenced by characteristics and thresholds of input images. A drawback of

segmentation-based approaches is that the quality of the segmentation stage has high impact to classification results.

Segmentation-free approach

With segmentation-free, the character segmentation step is omitted to reduce the computational cost, assign data labels, take the entire license plate image as input and output as text. To solve this problem, many studies have proposed solutions based on CNN and RNN. The models in [33] are based on variable length sequence decoding controlled by connectionist temporal classification (CTC) loss [34]. The [35] model follows the approach described in [36] but the sliding window method has been replaced by CNN output spatial splitting to the RNN input sequence.

Studies with free segmentation approach are mainly a combination of CNN and RNN, optimized through CTC loss. However, CTC tends to align each label prediction with the corresponding part of the input sequence. This is also its main drawback in choosing a fixed feature of which part of the image to output at each step of the CTC. Also, when using CTC, the output quantity is always less than or equal to the input string. As a result, if the model is trained for license plates whose large font size has a maximum of *m* letters, it will not be applicable to other signs of the same size but smaller font size because there are at most *n* letters, with n > m. Another distinctive feature of CTC is that it does not explicitly model inter-label dependencies, it solves this problem by adding a blank character to solve the problem of duplication.

In this work, we take a segmentation-free approach with an encoder-decoder architecture using Gated Recurrent Neural Network (GRU) [14] to solve the problem of output quantity limitation and inter-label dependency. We integrate an attention mechanism to get ideas from [12] to help the model extract important features in each part of the image, which affect the desired output.

2.1.2. Single-stage license plate recognition systems

While most research has focused on multi-stage processes in license plate recognition, recently there have been some successful studies on single-stage processes. Can be understood as using a single deep neural network, trained for end-to-end detection, localization and recognition of the license plate in a single forward pass. This allows models to share parameters and have fewer parameters than multi-stages models, thus being faster and more efficient [37]. This approach used VGG16 as a feature extractor, and modified VGG16 and removed three layers and the output was fed into a Region Proposal Network (RPN). They use two separate sub-networks for license plate detection and vehicle number recognition. The license plate detection subnet gives bounding box coordinates and probabilities, to avoid character segmentation they modeled a sequence labeling problem using Bidirectional RNNs (Figure 2.2).

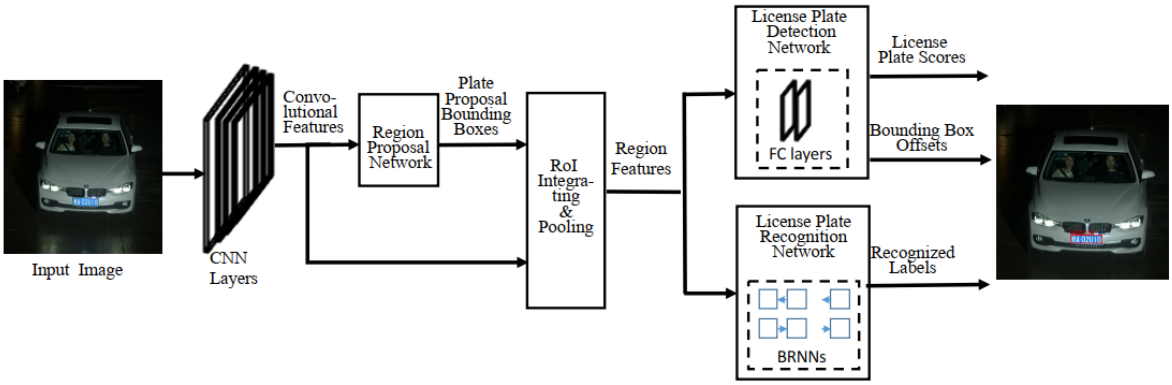

Figure 2.2. The overall structure of [37] single-stage model for ALPR.

In these systems, two separate sub-networks were used for license plate detection and license plate OCR. However, using only one deep neural network will make it more difficult to intervene in every step of the processing. Therefore, it is difficult to apply this approach to Vietnamese LPs that include both 1-line and 2-line plate styles.

2.1.3. Keypoints detection

In keypoints detection problem classes, human pose estimation (HPE) problem has attracted more attentions over last decade. HPE includes two main approaches: coordinate regression and heatmap regression [38]. Heatmap regression approaches have shown to be more effective than coordinate regression approaches [39] (Figure 2.3, Figure 2.4).

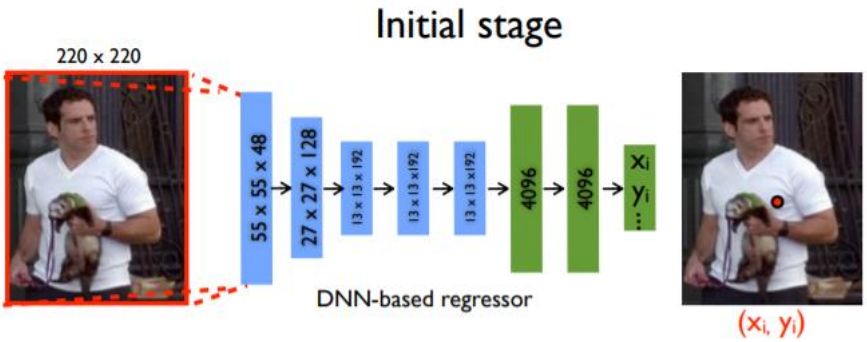

Figure 2.3.A coordinate regression approach

Figure 2.4. Heatmap regression outputs

For coordinate regression approach, they formulate the problem as a direct (continuous) regression to joint location rather than a discrete heat-map output. However, their method performs poorly in the high-precision region and we believe that this is because the mapping from input RGB image to XY location adds unnecessary learning complexity which weakens generalization. For example, direct regression does not deal gracefully with multi-modal outputs (where a valid joint is present in two spatial locations). Since the network is forced to produce a single output for a given regression input, the network does not have enough degrees of freedom in the output representation to afford small errors

which we believe leads to over-training (since small outliers - due to for instance the presence of a valid body part - will contribute to a large error in XY).

Studies based on heatmap regression training includes two main structures: high-resolution structure and feature fusion techniques such as HRNet [38] and DLA network [40]. In which, the first structure recorded the highest accuracy at AP = 74.4 in COCO test-dev human pose estimation dataset.

In this work, we use heatmap regression for LP keypoints detection to handle non-rectangular problem, in which each angle of LP is considered as a keypoint.

2.2. Proposed method of solving the ALPR problem.

Our proposed license plate recognition model consists of three main parts: car detection, license plate detection and license plate OCR, as illustrated in Figure 2.5.

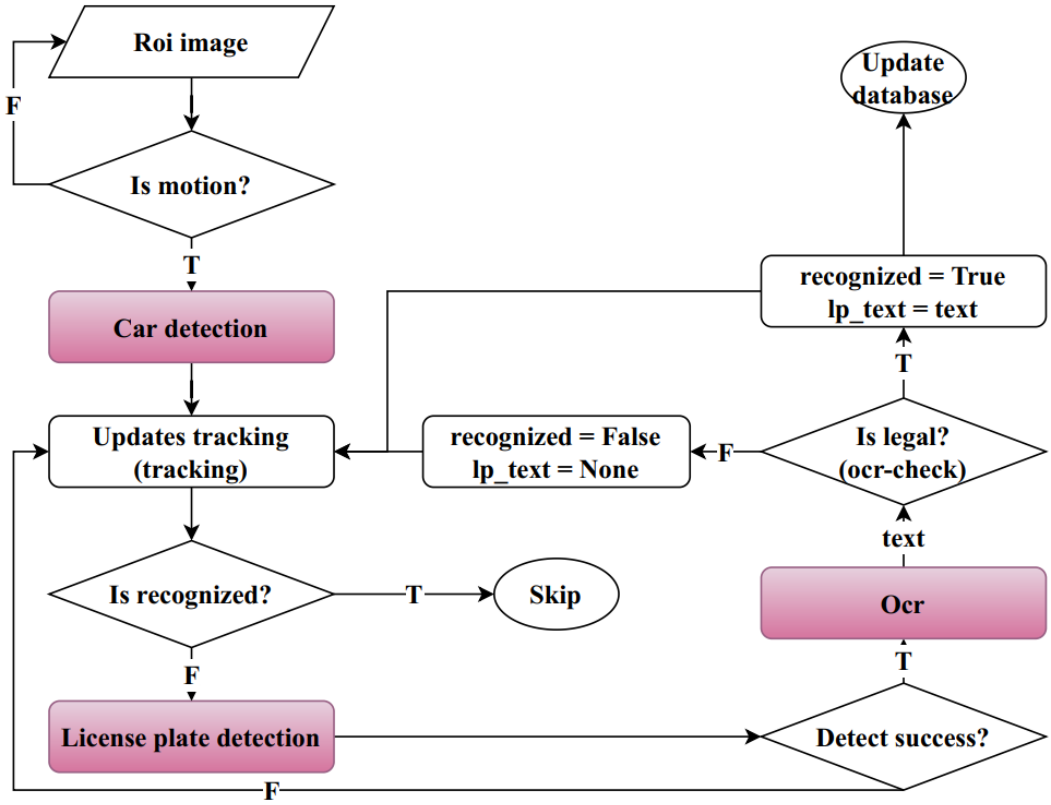

Figure 2.5. The flow of our ALPR core model.

The input is a frame obtained from the surveillance camera. on national highways can predetermine the area of interest (ROI). Motion detection algorithm based on frame difference. If there is movement, the car detection module detects the closing frame of the cars. Each detected closing frame will be updated with the tracking algorithm. The tracking algorithm indicates whether the object has been processed in the previous frame or not, if not, the License plate detection module detects the license plate. The OCR module conducts optical character recognition and goes through an ocr-check for the validity of the obtained results to confirm whether the prediction result is valid or not. If valid, the identified object completes the identification process and the results are stored in the database. In contrast, the object is not yet recognized, and continues to be processed in the following frames.

2.2.1. Car detection

Cars are one of the basic objects present in many large datasets, such as PASCAL-VOC [41], ImageNet [42] and COCO [43], so we decided not to train the generator from scratch, instead choose a known model to perform auto detection based on high call rate criteria, since mistakenly rejecting a car can lead to overall error, and also, precision rate need to be taken care of as the wrong detection of non-automobile objects consumes processing time as it has to go through the following steps. Based on these criteria, we decided to use the YOLOv2 network due to the fast execution speed of about 70 FPS and the accuracy of 76.8% mAP compared to the PASCAL-VOC dataset. We have not made any changes or improvements to YOLOv2.

For each detectable area, the objects will update the tracking algorithm we recommend. The tracking algorithm is based on the Euclidean distance between the centers of detected objects (i.e. between newly discovered regions and previously detected regions), and performs object storage discard after a zero time. appear in the frame. Objects that have not been successfully identified will

continue to be passed through the number plate detection step, the rest will be ignored. Objects that have not been successfully identified include: new objects appear; the subject of the license plate capture has not been successful; OCR identification results are wrong compared to the regulations of the Vietnamese number plate issued by the government.

The tracking algorithm we propose has three meanings. Firstly, moving car objects when captured from traffic cameras in Vietnam only move one way. Therefore, we build a simple but sufficient tracking algorithm to solve the problem and ensure the resources used. Second, each object performs successful identification exactly once, increasing overall performance. Third, objects that have not been successfully recognized will have the opportunity to re-recognize in later frames. This will increase the accuracy and robustness of the system.

2.2.2. License plate detection

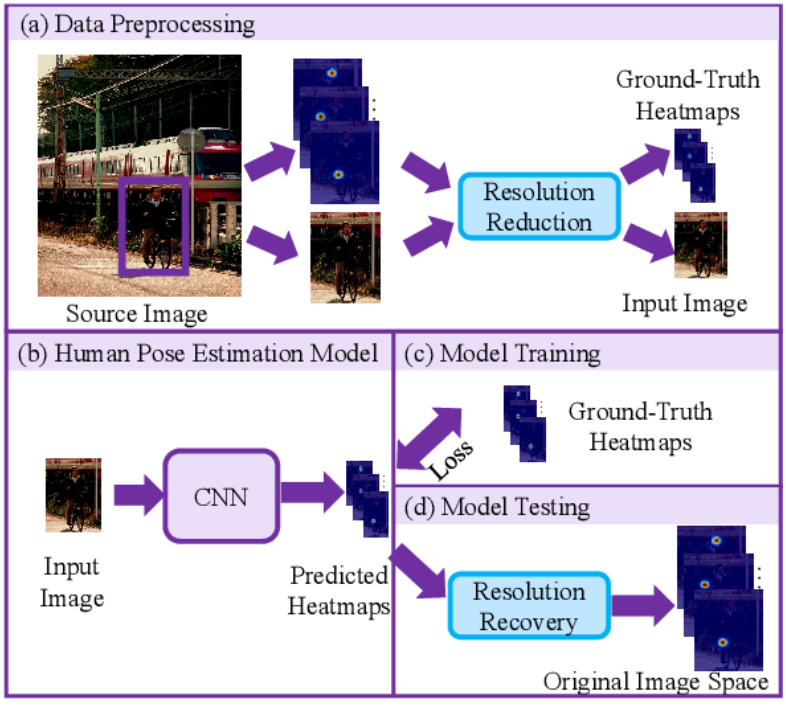

Figure 2.6. Our LP detection was inspired from top-down HPE approach.

For the license plate detection part, we approach the problem in the direction of keypoints detection, each corner of the license plate is considered a keypoint.

Based on the keypoints, we can align the license plates when they are deformed, or tilted at different angles. This is still unresolved by methods using pure rectangular closures for object detection.

The problem of detecting license plate keypoints is very close to the human pose estimation problem. There are two traditional ways to solve this problem: coordinate regression and heatmap regression. The heatmap regression approach achieves state-of-the-art on the benchmark datasets, making it more difficult to train the regression model directly with the coordinates of the keypoints. So we approach the second way heatmap regression. Similar to the steps in the human pose estimation problem (Figure 2.6). The training and inference process is performed as [44].

2.2.2.1. Preprocessing

All vehicle images were normalized to the standard normal distribution, and then resized to fixed sizes according to input shapes of the model architectures respectively. The purpose of this pre-processing is to obtain the same range of values for each input image before feeding to the CNN model. It also helps to speed up the convergence of the model.

2.2.2.2. Keypoints encoding

In our system, the heatmap regression is used for license plate keypoints detection. Considering input image $X \in \mathbb{R}^{W \times H \times 3}$ has width of W and height of H. Then, the output is a heatmap $Y \in [0,1]^{\frac{W}{H} \times \frac{H}{R} \times C}$, in which R is the resolution decrease, C is a number of classes. For each ground truth keypoint $k \in \mathbb{R}^2$ of class C, it is equivalent to $k' = [\frac{k}{R}]$ in Y. The output at (x, y) of a class c in heatmap Y is

$$Y_{xyc} = \exp\left(-\frac{(x - k'_x)^2 + \left(y - k'_y\right)^2}{2r_k 2}\right) \quad (2.1)$$

In which, r_k is the radius of the keypoint k in Y. To diminish $\delta = \frac{k}{R} - k'$, all classes of C shared two offset regression heads $O \in \mathbb{R}^{\frac{W}{R} \times \frac{H}{R} \times 2}$, in which $(O_{xy1}, O_{xy2}) = (\delta_x, \delta_y)$ if $Y_{xyc} > 0, 1 \leq c \leq C$. We choose $R = 4, r_k = 2$.

We found that in addition to detecting the object's keypoints, Zhou [44] also detects the center of the object and regresses the offsets to the object's keypoints as a way to connect the keypoints. other of the object is not necessary in our problem. Because each car we detect has only one license plate and can rely on the shape of the number plate to control the keypoints if there are more than 4 detected keypoints. Therefore, we only detect 4 keypoints that are enough to solve the problem and limit the error rate when the model has to learn more information. At the same time, previous studies consider keypoints as different classes, however, experimental process shows that the keypoints of license plates can be considered as a single class.

Figure 2.7. Keypoints encoding and local offset illustration.

Designing the number of keypoints for a LP and the number of classes C is illustrated in the Figure 2.8.

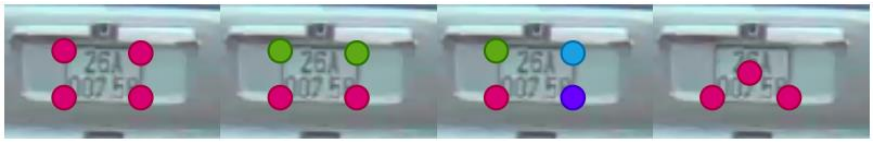

Figure 2.8. Keypoints design for a license plate (colours are according to classes). From left to right showed designs $1c, 2c, 4c, 1c3$ *accordingly.*

The design $1c$ considers all keypoints to belong to a class. The design $2c$ considers 2 upper keypoints to belong to one class while 2 below keypoints belong to another class. The design $4c$ considers 4 keypoints to belong to four different classes. The design $1c3$ considers 2 below keypoints and the centre of the LP to belong to the same class, ignoring two upper keypoints as their background varies a lot in different types of vehicles. It has been proved by experiments that the design $1c$ gives the best result. This is contradicted to human pose estimation where all keypoints belong to different classes [38].

2.2.2.3. Architecture for license plate detection.

The studies that used heatmap regression approach had two main architectures. The first type of architecture maintains high-resolution and various feature fusion techniques such as HRNet [38], DLA network [40]. The second type of architecture takes the form of high-to-low-to-high resolution by symmetric down convolution and up convolution combined with lateral connections such as Restnet-based architecture [45], Hourglass Network [46].

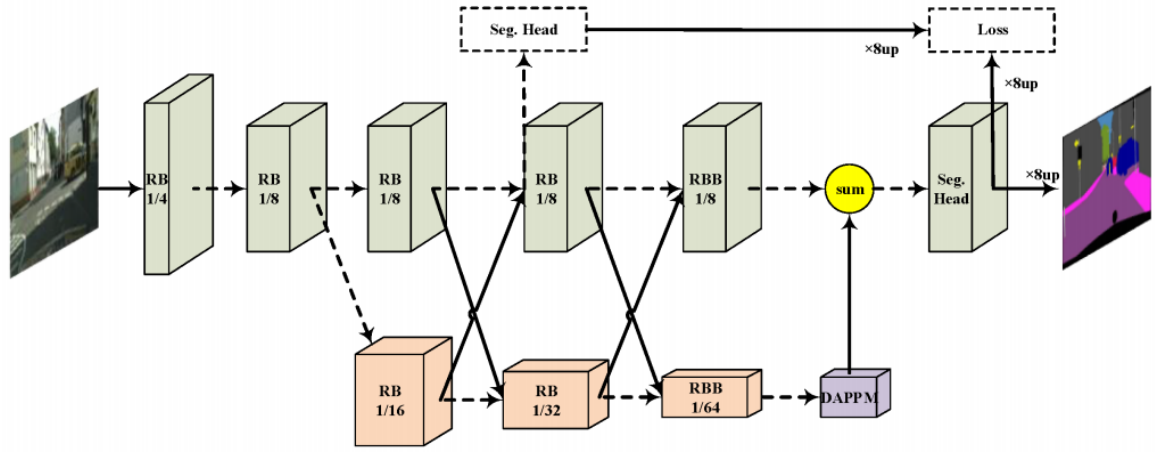

Figure 2.9. Original DDRNet architecture.

The first type of architecture gives good accuracy in this task, HRNet [38] achieves the state-of-the-art on the COCO test-dev human pose estimation dataset with AP = 74.4. We find that the above architectures have a very high relationship

with segmentation network architectures, so we change the segmentation network architectures to suit the problem.

The automatic license plate recognition system from traffic cameras requires processing time as quickly as possible, because the vehicles are constantly moving at high speed. Therefore, we prefer testing lightweight architectures including DLA-34 [40], HRNet-18 [38], Restnet18 [45]. For semantic segmentation architecture, we choose DDRNet [10] and its variants (Figure 2.9) as DDRNet is a semantic segmentation architecture, there is a state-of-the-art trade-off between accuracy and speed when tested on Cityscapes and Camvid, without using any inference optimization techniques or adding other data for training.

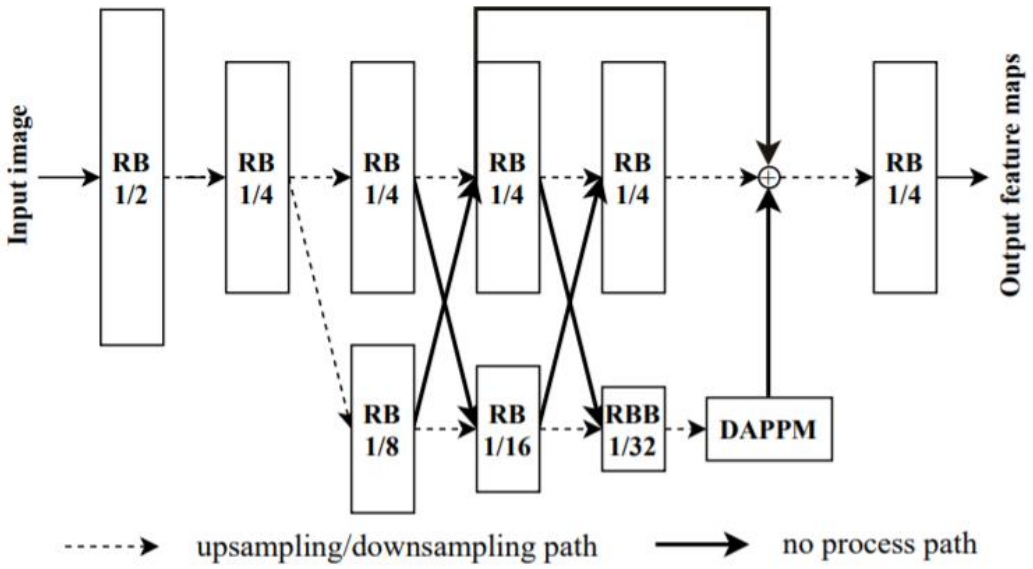

Figure 2.10. DDRNetsh Network, "RB" denotes sequential residual basic blocks. "RBB" denotes single residual bottleneck block. "DAPPM" denotes the Deep Aggregation Pyramid Pooling Module [10].

To be applied into LP keypoints detection problem, we made some changes in the architecture of DDRNet [10]. To inform about lower feature maps, we used residual connection instead of training and collecting information from separated stages as in the original architecture. In addition, the size of an output heatmap that we choose is proportional to the input at R = 4. However, DDRNet maintains

a high-resolution feature map with size of R = 8 compared to the input image. We made two modifications corresponding to two new architectures in order to suit the keypoints detection problem. The first architecture DDRNetsh was obtained via changing the level of down convolution at the first convolution block (Figure 2.10). The second architecture DDRNetup was obtained by adding one block upconvolution at the end of the original architecture (Figure 2.11).

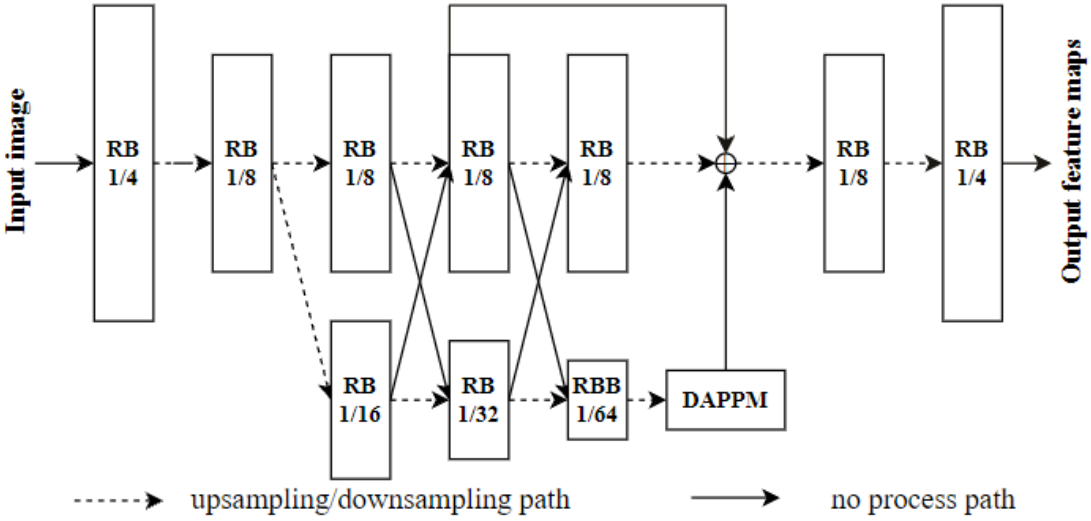

Figure 2.11. DDRNetsup Network, "RB" denotes sequential residual basic blocks. "RBB" denotes single residual bottleneck block. "DAPPM" denotes the Deep Aggregation Pyramid Pooling Module [10].

2.2.2.4. Keypoints decoding

Having obtained feature maps, we used max pooling operator with kernel size of 3×3 according to feature maps of C classes. We removed points that have score less than 0.3 and choose the top 4 points with the highest score. They are the 4 keypoints of a LP needed to find. Then, positions of keypoints in the original image $(x_{\text{or } i}, y_{\text{or } i}) = \big((x + O_x) \times R, y_{\text{ori}} = \big(y + O_y\big) \times R\big)$, in which (x, y) is a position of keypoints in the output feature maps.

2.2.3. License plate optical character recognition

2.2.3.1. The characteristics of Vietnamese license plates

Vietnamese vehicle LPs are designed according to government regulations (Figure 2.12). Vietnamese vehicle LP colors include four main types: blue background or red background with white text, yellow background or white background with black text.

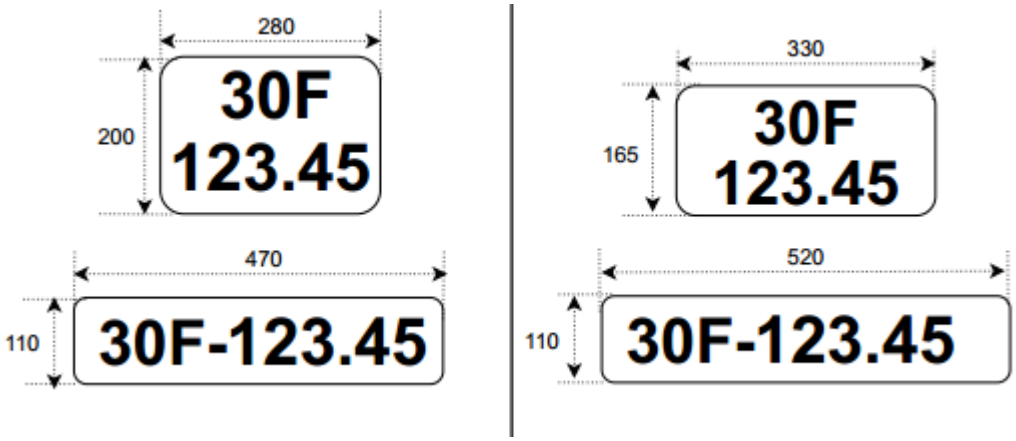

Figure 2.12. Size of old and new Vietnamese LP (unit mm).

The characters in the number plate are in a set of 33 characters {ABCDEFGHKLMNPQSTUVXYZ0123456789-.}. These characters are not randomly arranged, but all belong to one of five template types (Figure 2.13). 1-line LP are obtained from 2-line LP by adding a "-" sign when combining the first and second lines of a 2-line number plate.

Figure 2.13. Color and text formats of Vietnamese license plate.

2.2.3.2. Preprocessing

All LPs images are normalized to the standard normal distribution and then are resized to fixed sizes according to input shapes of the model architectures

respectively. For 2-line LPs, we use the X-Y Cut algorithm [47] to find out where to split a 2-line LP into two 1-line LPs.

2.2.3.3. Architecture

The input of the VOCR is an image of 1-line LP which has size of 32 × 140. The first part of VOCR is VOCR-CNN which extracts local features based on VGG19 [13] (Figure 2.14). Due to small input image size 32 × 140, we choose average pooling instead of max pooling to avoid losing of information. Additionally, the kernel size and stride size were chosen with a small size to compress the information appropriately. The output of VOCR-CNN has 256 feature maps; each feature map has size of 1 × 70. We considered the number of dimensions of each feature map to be the number of time steps, thus time steps T = 70. At each time step t, a feature vector of size 1×256 contains the features of all feature maps at that time step.

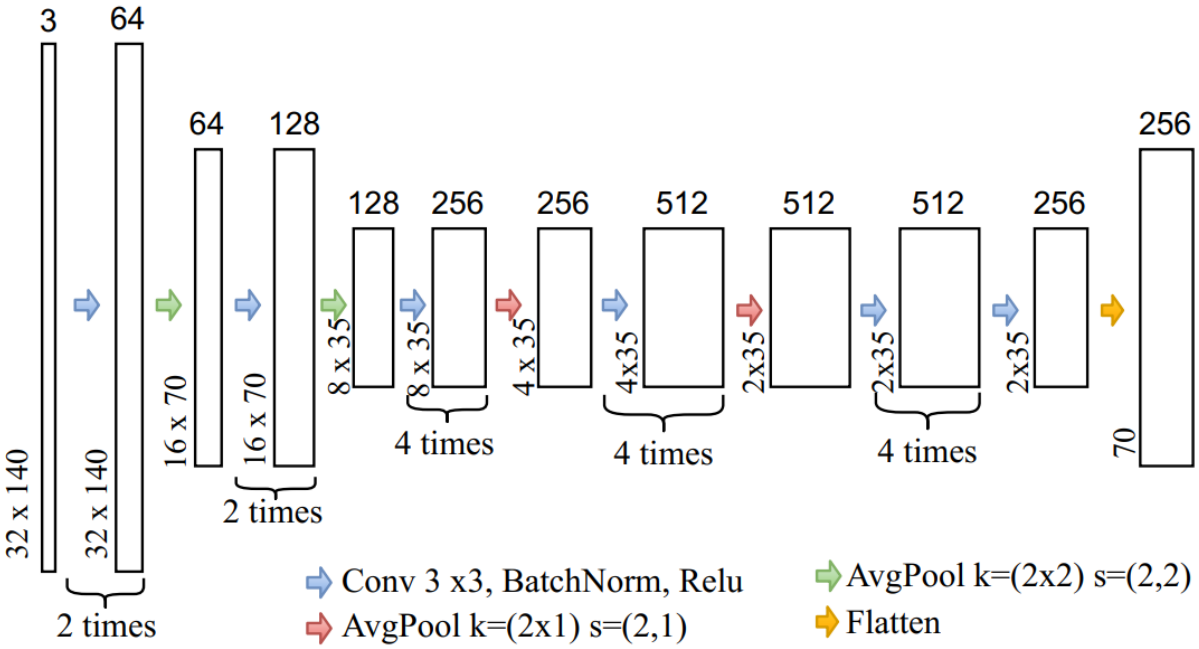

Figure 2.14. VOCR-CNN architecture.

The second part of the VOCR is the VOCR-RNN (Figure 2.15) in the form of an encoder-decoder using the Gated Recurrent Neural Networks (GRU) [14] combined with the attention mechanism [12].

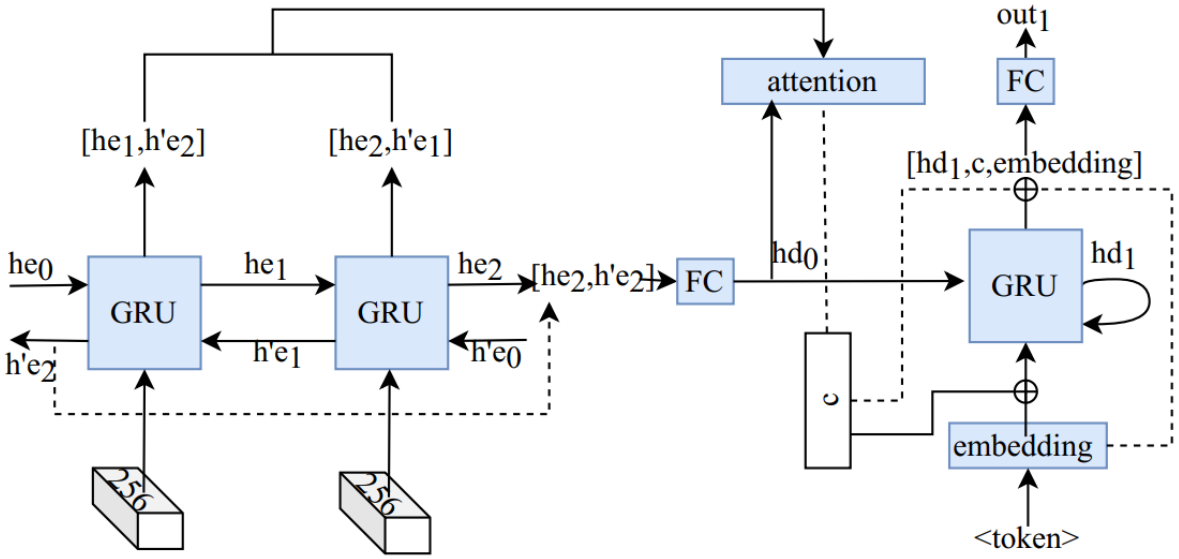

Figure 2.15. VOCR-RNN architecture.

In which [he, $h'e, hd$] is the hidden output feature at every time step of the encoder. In the encoder part, we choose GRU-bidirectional to extract interdependencies in the input string. The output of the encoder at each time step is concatenate of [he, $h'e$] with the dimension 512. The output of the encoder is through the fully connected layer to reduce the number of dimensions equal to the number of hd. The attention mechanism is inspired by [12] consider each hd at each time step as query and all $[he, h'e]$ act as keys and values. Since the dimensions of keys and query are not the same, we use the attention scoring function as Additive attention [48].

$$a(Q, K) = w_v^T \tanh\left(W_q Q + W_k K\right) \in R^{1\times T} \quad (2.2)$$

In which, $Q \in R^{q\times T}, W_q \in R^{h\times q}, K \in R^{k\times T}, W_k \in R^{h\times k}, w_v \in R^h$. In this work, $Q = hd, K = [he, h'e], h = 256, q = 256, k = 512, T = 70$. Therefore, the context vector c can be synthesis like bellow:

$$w_i = \frac{e^{a_i}}{\sum_{i=1}^{T} e^{a_i}} \in R$$
$$w = [w_1, w_2, \dots, w_T] \in R^{1\times T} \quad (2.3)$$
$$C = V \times w \in R^{v\times 1}$$

In which, w is attention weights of each time steps, $V = K, v = k$. Context vector c synthesized will contains important characteristic information from the input sequence that the current decode step needs attention. This makes a lot of sense, for example, predicting that the label is the first letter of the number plate, attention weights w will have a higher weight for time steps containing information for that character.

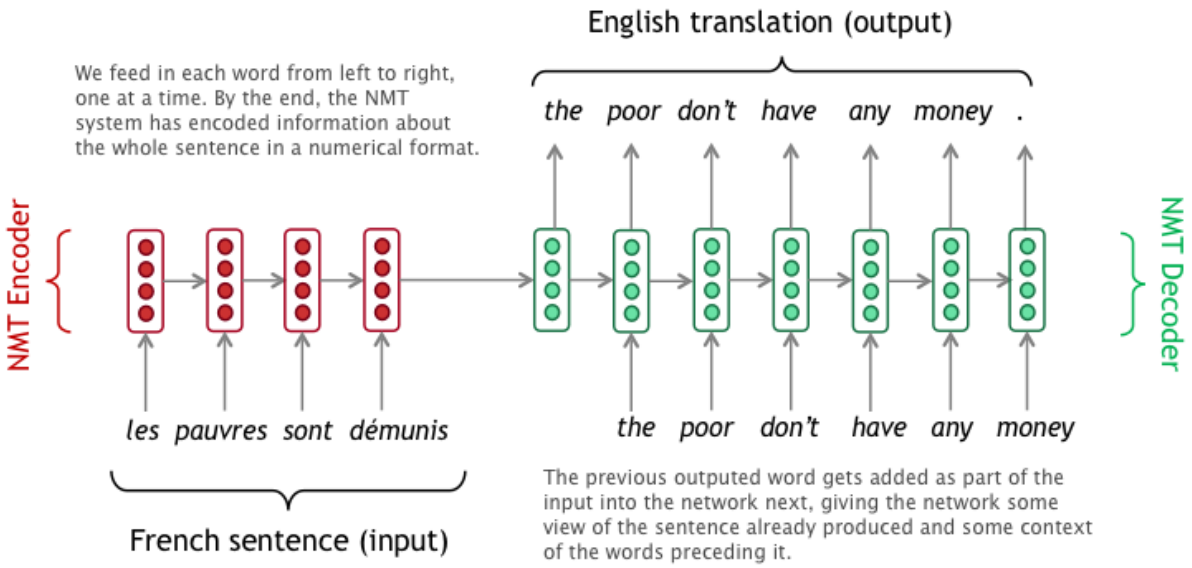

Figure 2.16. Teacher forcing training strategy in Machine Translation.

We train the decoder using the teacher forcing training strategy (Figure 2.16). The network decoder takes as input hd at the previous time step and concatenate $[context\ vector, c, embedding]$ of the label at the current time step. The output of the decoder is the concatenate of $[hd, c, embedding]$ that goes through the fully connected layer with the output dimensions N = 36 (corresponding to the number of characters appearing in Vietnamese LPs plus 3 special characters $<$ sos $>, <$ eos $>, <$ pad $>$). The highest one will the character needed to decode. If we meet $<$ eos $>$ character, the decoding process will complete.

2.2.3.4. Loss function

The loss function used is a combination of the cross-entropy loss and label smoothing technique [49] to increase the generalization for our model.

$$L = -\sum_{i=1}^{N} \left(l_i \times y_i \times \left(\log(\hat{y}_i) \right) \right) \tag{2.4}$$

In which, y_i is a ground truth, $\hat{y}_i$ is a prediction of the model, $l_i = |y_i - \delta|$ is the label smoothing weight at i class. We choose $\delta = 0.1$ based on experimentation of all different thresholds proposed in [49].

To optimize the loss function L, we use the parameter update method based on gradient descent with the Adam algorithm to automatically adjust the learning rate and avoid falling into the local minimum.

2.3. Experiments and results

2.3.1. Dataset

As far as we know, studies in ALPR in Vietnam have been conducted using small private datasets. Vinh et al. [4] tested models in a dataset of only 700 images. Duan et al. [17] used 805 images for test dataset. Understanding the important role of dataset size to the generalization of deep neural networks, we built a large and diverse dataset named MTAVLP. We installed 2 traffic cameras on two different road tracks. We set up a system excluding the LP OCR step–onto a server. The collection process was taken within 3 weeks–from 6:00 a.m. to 6:00 p.m. daily–in different weather and light conditions. Vehicle objects captured were all vehicles moving on roads with both old and new LP styles, and in a variety of plate sizes, font styles, and in different degrees of opacity and noise. Some different images of vehicles and plates are shown in Figure 2.17, Figure 2.18.

Figure 2.17. Vehicle images captured in MTAVLP dataset.

Figure 2.18. License plate images in MTAVLP dataset.

Table 2.4. Number of plates according to colors.

Number of plates	White	Yellow	Blue	Red	Total number
1 -line plate	2717	0	13	7	2737
2-line plate	6933	950	143	10	8036
Total number	9650	950	156	17	10773

To increase the diversity of distribution for the MTAVLP dataset, we added to our dataset a set of 3000 Vietnamese vehicle images that were collected from different conditions and published in [50]. This brings the number of vehicle images in our dataset to 15571 and the number of LP images is 10773. The number of LP is smaller than the number of vehicle images as we removed images captured in redundant times. LP types were summarized in Table 2.4 arranged by

their colors. After preprocessing 2-line plate images, we obtained 16012 images of 1-line plate.

2.3.2. License plate detection results

Conducting Experiment: We randomly divided MTAVLP dataset in to three sets for training, validating and testing according to a rate of 0.8:0.1:0.1.

Validation method: mean IOU ($mIOU$) and precision with $mIOU >= 0.75$ written as P_{75} were chosen as validation criteria [28]. A threshold of $mIOU >= 0.75$ was used instead of $mIOU >= 0.7$ to strengthen the robustness of model. Training: In order to increase the generalization of the model, we randomly performed argumentation operations during training. Adam optimization algorithm was used with an initial learning rate of 0.001, batch size of 8, maximum training epoch number of 140 , and early stopping patience of 10 .

Results and Comparison: We experimented all designs $1c$, $2c, 4c, 1c3$ (Chapter 2-2.2.2.2) with the Resnet18 architecture [37].

Table 2.5. LP detection results for different keypoints considerations.

Techniques	Training size	Testing size	$mIOU$
Resnet18_lc	384 x 384	384 x 384	**93.8**
Resnet18_2c	384 x 384	384 x 384	93.797
Resnet18_4c	384 x 384	384 x 384	91.78
Resnet18_ lc3	384 x 384	384 x 384	90.5

Table 2.5 shows the highest $mIOU$ = 93.8 for Resnet18_1c. It strengthens the hypothesis that all keypoints of a LP can be considered as individual objects. Resnet18_2c is as good as Resnet18_1c because the similarities of the upper and lower keypoints pairs are equivalent. Resnet18_4c based on the 4c design for the human pose estimation problem gives worst results due to the keypoints have common features that cause the model to be confused during training. Resnet18 _3c also produces worse results, indicating that determining a center of a number

plate is more difficult in heatmap regression. Thereafter, we used 1c design for our experiments.

In order to evaluate the effect of input image size on model performance, we studied two image sizes, 384 × 384 and 512 × 512. Results are shown in Table III. We chose DDRNet23sup that was originated from DDRNet-23-slim [3] and made modification as proposed in Section III-B. The results suggest that the performance of a model is slightly decreased when applying the model in a dataset that has smaller image size than that of training set.

Table 2.6. LP detection results in different architectures.

Techniques	**Training size**	**Testing size**	$\boldsymbol{IOU}$	$\boldsymbol{P_{75}}$
DDRNet23sup	384 x 384	384 x 384	94.5	99.2
DDRNet23sh	381x 381	381x 381	**95.01**	**99.5**
DDRNet23up	384 x 384	384 x 384	94.4	99.0
HRNetl8_4s	384 x 384	384 x 384	94.2	98.58
HRNetl8_3s	384 x 384	384 x 384	94.1	98.9
Resnetl8	384 x 384	384 x 384	93.8	99.1
DLA34	384 x 384	384 x 384	90.5	93.6
DLA34	512 x 512	512 x 512	93.0	97.4

Using image size of 384 × 384 for training and test sets, we trained and evaluated different architectures including DDRNet23sh, DDRNet23sup, HRNet18_4s, HRNet18_3s, Resnet18 and DLA34. DDRNet23sh is originated from DDRNet-23-slim [10] with a modification in down sampling convolution rate. DDRNet23sup is originated from DDRNet23-slim and DDRNet-23 with a modification in up sampling convolution. HRNet18_4s and HRNet18_3s are 4-stage and 3-stage versions of HRNet18. Results are showed in Table 2.6.

DDRNet23sh has highest accuracy with $mIOU = 95.01$, P_{75} = 99.5. Although it is an architecture for segmentation problem. This proves the

efficiency of DDRNet23sh when it is meticulously designed with multi-branch architecture and maintenance of high-resolution. DAPPM module in the architecture enriches extracted information. The results also strengthen the hypothesis that maintaining high-resolution since beginning is better than the use of up sampling convolution at the last layer as in DDRNet23sup and DDRNet23up architectures.

2.3.3. License plate OCR results

Conducting Experiment: We divided the set of 16012 LP images into three sets for training, validating and testing with a rate of 0.75:0.1:0.15. Different architectures with segmentation-free approach were studied in order to compare to VOCR, such as LPRnet-STN a lightweight architecture with CNN and CTC loss; CRNN [32]- a combination of CNN and RNN. Besides, we added one more architecture CRNN (VGG19) that is based on CRNN with the use VGG19 as the backbone instead of lightweight CNN.

Validation method: We used LP recognition accuracy at sequence level Acc_{seq} to evaluate LP OCR, in which a LP is considered correctly recognized if all OCR characters are correct without any added characters. However, in cases where only one character is wrongly recognized, the above evaluation criteria cannot give meaning. Therefore, we use Acc_{char}, accuracy at character level to evaluation in addition. A character is said to be correctly identifiable, if its position in the label and in the prediction result is the same.

Training: In order to increase the generality of the model, we randomly perform augmentation operations during the training process, including:

Blurry image: Gaussian Blur ($\sigma \in [0, 1.0]$), Motion Blur ($kernel-size = [3, 3]$)

Increase or decrease Hue and Saturation channel values by random values.

Adjust the contrast of the image.

Invert the values of the image.

Apply transformations like (rotate, crop, translate) to the image.

Reduced image quality.

Truncate the image information by setting the values there to be 0.

Multiply all pixels in an image by a random value.

Increase, decrease the brightness of the image.

Adam optimization algorithm was used with an initial learning rate of 0.001, batch size of 32, maximum iteration number of 10000, and early stopping patience of 250.

Results and Comparison: Table shows that VOCR has dominant efficiency in this task with $Acc_{\text{seq}} = 99.28\%$, $Acc_{\text{char}} = 99.7\%$. As limitations of CTC loss (mentioned in Chapter 1-0), LPRNet-STN, CRNN, and CRNN (VGG19) have worsen results compare to VOCR. The differences are more significant in sequence level. This suggests that CTC based approaches having difficulties in correctly recognizing the character string of an input image.

Table 2.7. Results of LP OCR in MTAVLP dataset.

Techniques	**Training set**		**Validation set**		**Testing set**	
	Acc_{seq}	Acc_{char}	Acc_{seq}	Acc_{char}	Acc_{seq}	Acc_{char}
VOCR	**99.34%**	**99.8%**	**99.25%**	**99.9%**	**99.28%**	**99.7%**
LPRnet STN	92.3%	96.9%	94. 1%	96.4%	94.8%	96.8%
CRNN	92.9%	96.9%	93.6%	97.2%	93.4%	97.1%
CRNN (VGG19)	92.3%	95.5%	92.5%	95.4%	93.0%	96.2%

90% of LP images in the MTAVLP dataset have white background. The imbalance is because white plate is a license plate for individuals. To solve the imbalance data problem, we used image negative and histogram matching

algorithm to generate MTAVLP-color dataset including blue, red, yellow plates from white plates in all 3 training, testing and validation sets.

Table 2.8. Effect of unbalanced data to VOCR.

Dataset	**MTAVLP-color**		**MTAVLP**	
	Acc_{seq}	Acc_{char}	Acc_{seq}	Acc_{char}
Training set	98.8%	99.6%	**99.34%**	**99.8%**
Validation set	98.7%	99.7%	**99.25 %**	**99.9%**
Testing set	98.5%	99.5%	**99.28%**	**99.7%**

We did not retrain, but used the VOCR model trained in the original MTAVLP dataset to test for it performance in the MTAVLP-color dataset. The results show that the accuracy decreases at only 1% in all datasets as in Table 2.8. Most color plates that are miss-recognized are usually those that are incorrect in the original set or not cleared in characters due to the data generation process. This suggests that the colors of LP have a slight effect on the results of LP OCR. It should be noted that these evaluations were performed in different datasets, and our proposed model was performed on the MTAVLP dataset - a dataset of Vietnamese license plates. Applying our proposed models in other datasets is worth to considerer; however, it is out of scope of this paper, so we leave it for future work.

2.3.4. Overall performance

Our proposed ALPR system was set up test using a single GPU NVIDIA GeForce RTX 2080 and without using any inference optimization techniques such as Tensor-RT. We recorded the processing speeds of stages as follows: The processing speed was 38.28 FPS in the stage of car detection using YOLOv2; In the LP detection stage, DDRNet23sh could be run at 103.5 FPS; and in the LP recognition stage, we got 36 FPS with VOCR. Consequently, our ALPR system can be run at 15.73 FPS in a single NVIDIA GeForce RTX 2080 without using any inference optimization techniques. Based on these results, we believe that our

ALPR system can run in real-time when being deployed using additional inference optimization techniques.

Figure 2.19. Results of using DDRNet23sh and VOCR.

2.4. Conclusion for chapter 2

In this chapter, we proposed a model for ALPR in Vietnam in unconstrained environment that focused on improvements on license plate detection and license plate OCR. Detecting the areas containing a plate number and recognizing LP characters. Our improvements have been empirically proved to be effective for Vietnamese ALPR in unconstrained environment. To tackle the problem of nonrectangular object detection, an approach based on keypoints detection was used and DDRNet23sh architecture was modified. This improvement get the highest mean IOU $mIOU = 95.01\%$ and precision $P_{75} = 99.5\%$. In this paper, we also propose VOCR based on segmentation-free that uses CNN architecture

combined with encoder-decoder RNN and attention mechanism. The performance accuracy of our proposed architecture is up to $Acc_{\text{seq}} = 99.28\%$ and $Acc_{\text{char}} = 99.7\%$. We also built a dataset of Vietnamese car license plates named MTAVLP that is large and covers various conditions of image capturing.

Our proposed methods have been published to the fourth International Conference on Multimedia Analysis and Pattern Recognition (MAPR) supported by VAPR (Vietnamese Association on Pattern Recognition) was held in Ha Noi on October 15-16, 2021. The title of our paper is "*an efficient method to improve the accuracy of Vietnamese vehicle license plate recognition in unconstrained environment*" [11]. We received an award as the best paper in all of the conference papers. These showed that our proposed methods were so prominent for solving ALPR problem.

Chapter 3

LARGE-SCALE CAR TRACKING AND MONITORING SYSTEM.

3.1. Related work of large-scale car tracking and monitoring system.

In Hanoi, the Hanoi Traffic Police Department is managing and using nearly 600 cameras of all kinds, including more than 300 cameras to monitor vehicle traffic, more than 100 cameras to serve traffic violations and nearly 100 traffic monitoring devices. In addition, at many intersections, there are cameras of district police; camera of VOV traffic; Thanks to this system, the situation of traffic participants violating errors such as lane encroachment, red light crossing, improper parking... has decreased significantly. Hanoi is urgently researching to integrate not only surveillance cameras of state agencies, but also people and businesses (airports, railway stations, bus stations, administrative agencies, alleys...) to support general traffic management.

In Ho Chi Minh City, the traffic monitoring and control center phase 1 (belonging to the project of upgrading the existing traffic control system) has been completed, with a total investment of about 250 billion VND. traffic control service on main roads, intersections in the city center, with 4 functions: control signal lights; monitoring and providing traffic information; support handling violations. Traffic monitoring and control center operating 24/7 for 365 days can be considered as the "magic eye". Information will be immediately shared on the traffic portal for people to update, avoid traffic jams, prevent congestion, and handle infrastructure problems effectively. In phase 2, after 2020 will complete the city-wide Smart Traffic Operations Center with a total investment of 250 million USD. The intelligent traffic control center will directly manage the entire traffic control system of the city as well as provide information and data to serve the handling of violations; supervise and ensure urban security and order; handling emergency incidents; fire, explosion and flood prevention; yard management.

Hue will sanction violations synchronously throughout the province from November 1, 2019. Violations of traffic order and safety such as: stopping, parking, picking up and dropping off passengers not in accordance with regulations; vehicles run on prohibited roads, at prohibited times; vehicles are not allowed, there is no route into the city; vehicles running in the wrong direction, on the wrong route, running through red lights, oversized vehicles, overloaded vehicles; illegal encroachment on roadways, curbs, construction of works on roads that do not ensure traffic safety... These acts are recorded directly from the image recording system operated by the Center for Supervision and Administration. managed smart city (including devices that record images of other organizations, individuals and units that invest but have been allowed to connect to the smart city monitoring and operating system and images reflected by individuals and organizations through the receiving channel of the Smart City Supervision and Administration Center (which has been verified by the Center) according to regulations, they will be handled and administratively sanctioned according to the provisions of law.

Through the survey, it can be seen that the general situation in big cities has been and is being equipped with surveillance cameras. However, there is no specialized image processing software to process data from these cameras to support functional forces to manage violations and look up when needed. In particular, in emergency cases, with the currently installed cameras, it is not possible to track the violating vehicles when circulating through the camera's observation stations. Therefore, it is very necessary to build a license plate recognition system to monitor the vehicles circulating in the area visually and accurately, to assist the authorities in tracing the vehicles when needed. high efficiency management.

3.2. A large-scale car tracking and monitoring system analysis and design.

3.2.1. The main components, functions and actors of the system

The system is built with 3 components including:

- Client-side software is used to manage the system and monitor vehicle journeys on digital maps.
- The server manages and controls the data flow from the cameras.
- The module recognizes license plates and returns the results to the client-side software.

3.2.1.1. Main components and functions

a. Client-side software.

Manage camera devices (IP address, Port, connection string, Camera location, camera type, ...)

- Camera list and status
- Add new camera
- Edit camera information
- Remove camera from database

Vehicle license plate data management

- Look up the number plate of the vehicle recognized on the camera
- Put license plates on the watch list
- View detailed information of each recognized license plate (time, location, image extracted from the camera)

Manage license plate data by camera

- Look up the recognized license plate on each camera.
- View detailed information of each recognized license plate (time, location, image extracted from the camera).

Look up the route of the vehicle on the digital map

View live camera images as normal grid or in license plate recognition mode.

b. The server manages and controls the data flow from the cameras.

Collect data from clients containing license plate recognition modules and store them in a database.

Returns database query results and images to the client.

Recorded video by time from all cameras and storing management.

c. The license plate recognition module is located on client-side.

Each module is responsible for collecting data from the cameras.

Acquire images from the cameras under their management, then process and return to the server the resulting images and strings from the license plates.

3.2.1.2. System actors

Based on the purpose of the system, we identify two actors:

System-wide manager (Admin): Admin has the role of general management of the entire system. Manage user access rights. Perform all operations on system software.

Operating staff: In charge of the implementation of all the basic functions of the system.

3.2.2. System structure.

3.2.2.1. Camera management subsystem.

View live images transmitted from the camera in a grid or view each camera in detail with functions:

– Control panel with camera controls: Pan, Tilt, Zoom, etc.

- Video operations such as: Play, stop, turn on display with license plate recognition mode, run video playback.
- Display the image of the license plate recognized on that camera.

Camera management on digital map: View information about the camera, operating status, helping users have a visual view of the camera system on a digital map.

Comprehensive and complete management of camera information such as: IP address, connection ports, access information, connection string, camera location on the map. Includes functions:

- Add a camera device with the following contents: Camera name, IP address, connection ports, camera type, login information (Username, password), connection URL string, location of the camera supporting digital maps.
- Update camera device information with the following contents: Camera name, IP address, connection ports, camera type, login information (Username, password), connection URL string, location of camera with map support number.
- Delete camera information.

Manage video playbacks: Manage and review playback videos.

3.2.2.2. Vehicle tracking subsystem.

Track the vehicle's journey over a period of time on a digital map, showing detailed history, time and location according to each camera that the vehicle has passed.

Look up details of history, time, place that the vehicle has passed, review images extracted from the video.

Look up details of history, time, location of license plates by each camera, review images extracted from videos.

3.2.2.3. User management subsystem

Manage user information including information such as usernames, passwords, roles with the following functions:

- Add users
- Update user information
- Remove user from database

Manage user roles with permissions for each role:

- Add role
- Update role information
- Remove user roles

3.2.3. Use case diagram

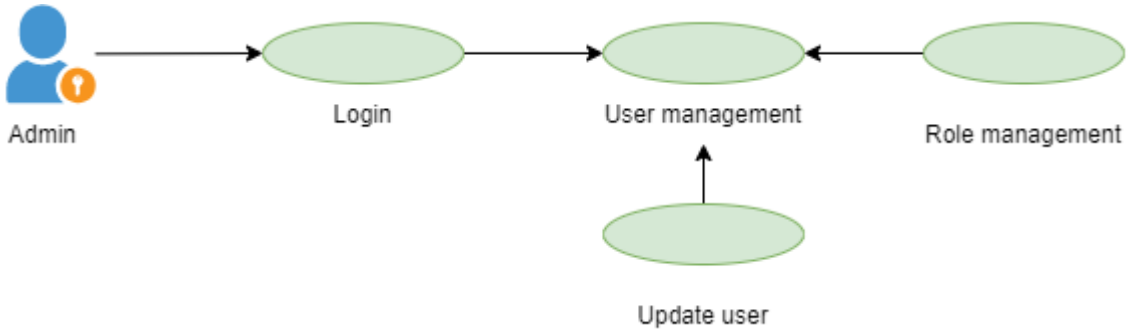

Figure 3.1. Use case model for the system manager (admin).

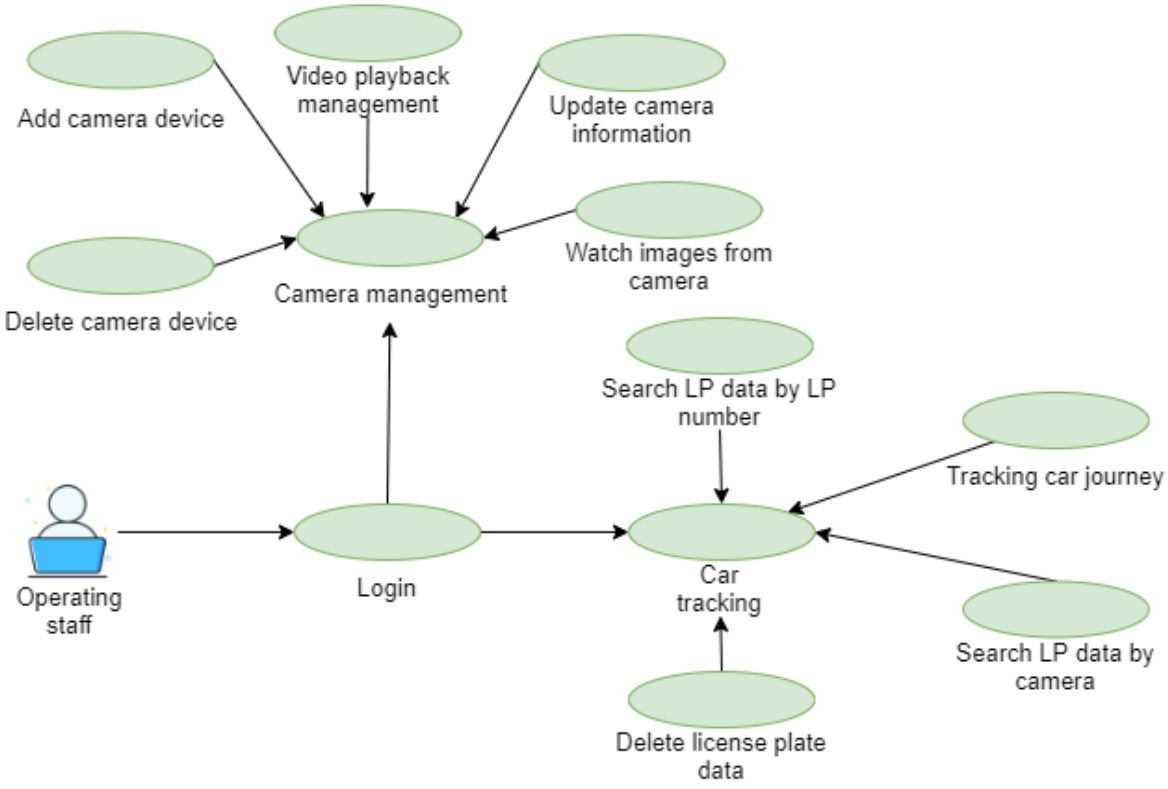

Figure 3.2. Use case model for system operating staff.

3.2.4. Architectural design

The system uses Client-server architecture consisting of 3 main components:

- License plate recognition module
- Main server manages and processes data
- Client-side software

3.2.5. Database design

3.2.5.1. Logical diagram

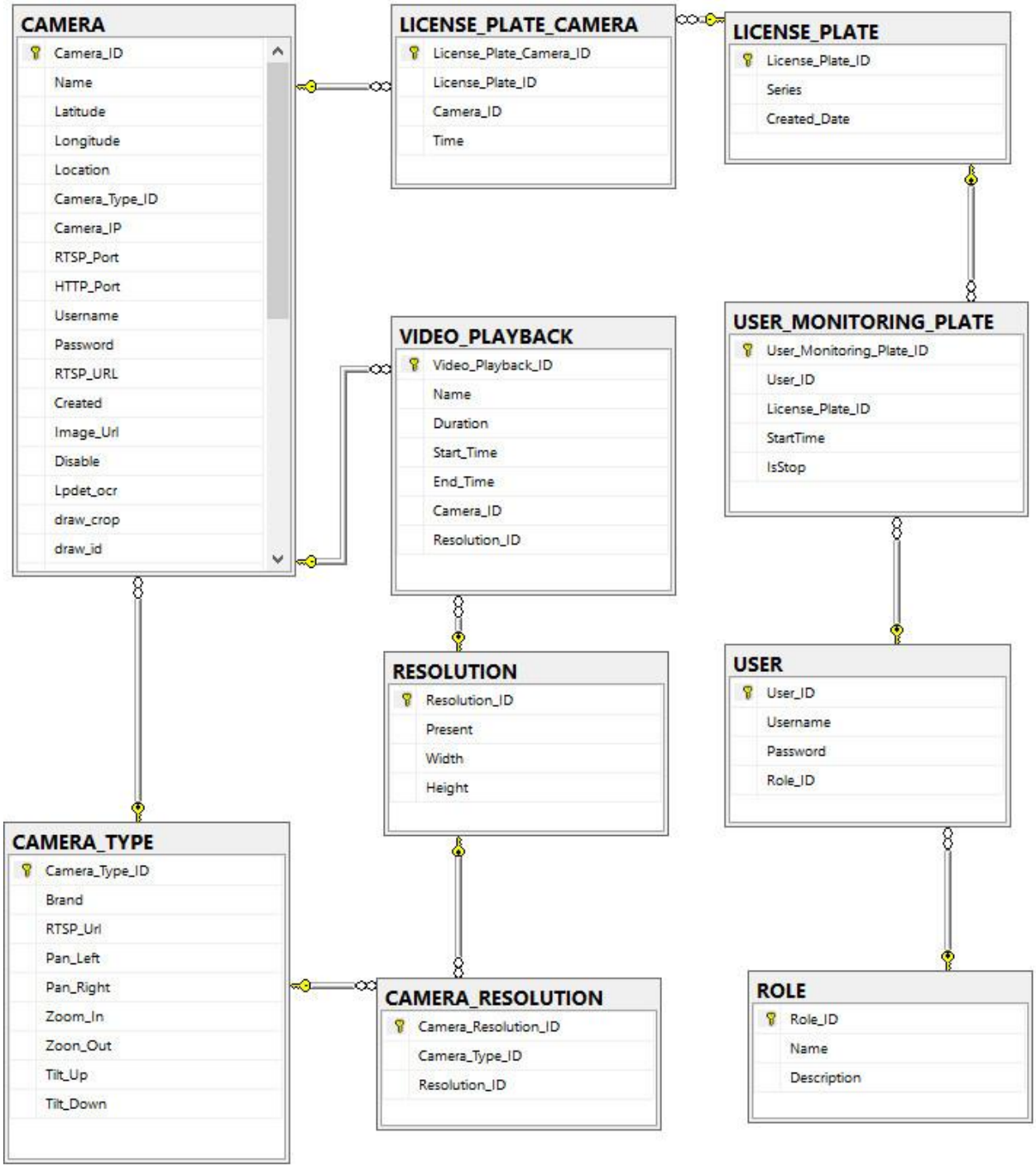

Figure 3.3. Logical diagram.

3.2.6. User interface design

3.2.6.1. Login interface

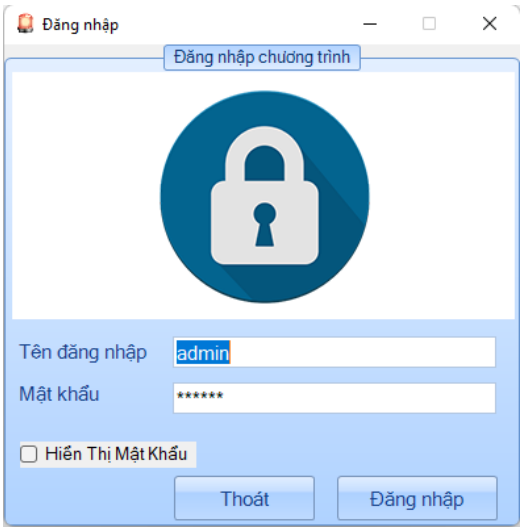

Figure 3.4. Login interface.

3.2.6.2. Grid camera view interface

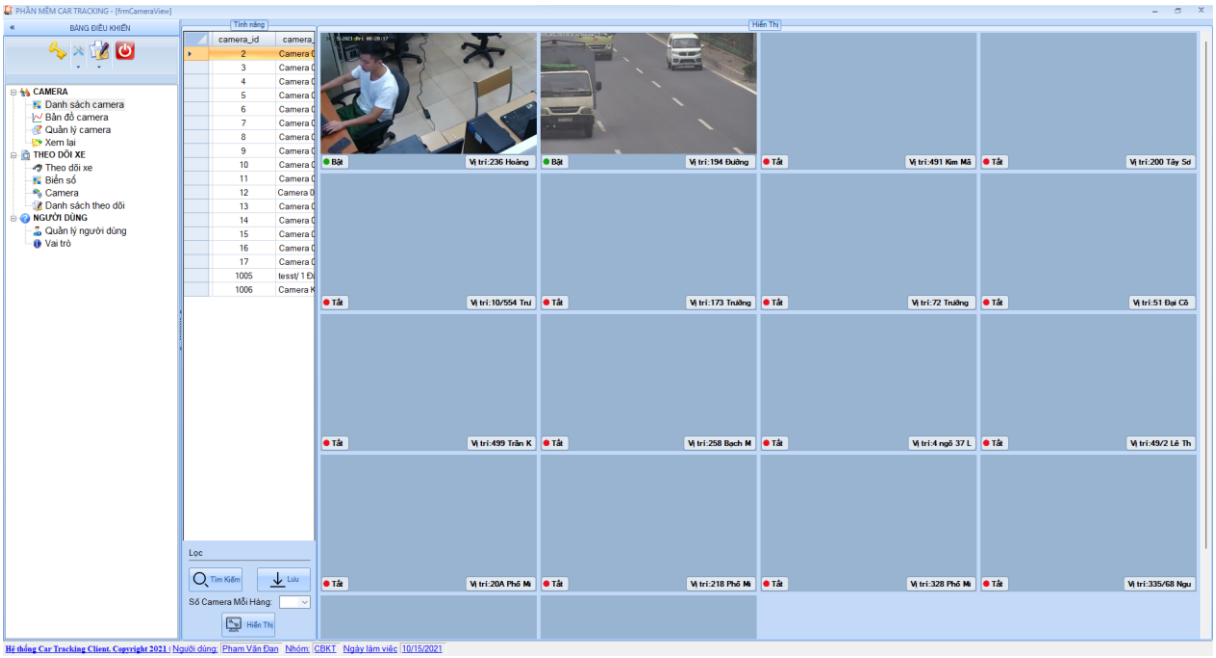

Figure 3.5. Grid camera view interface.

3.2.6.3. Camera map interface

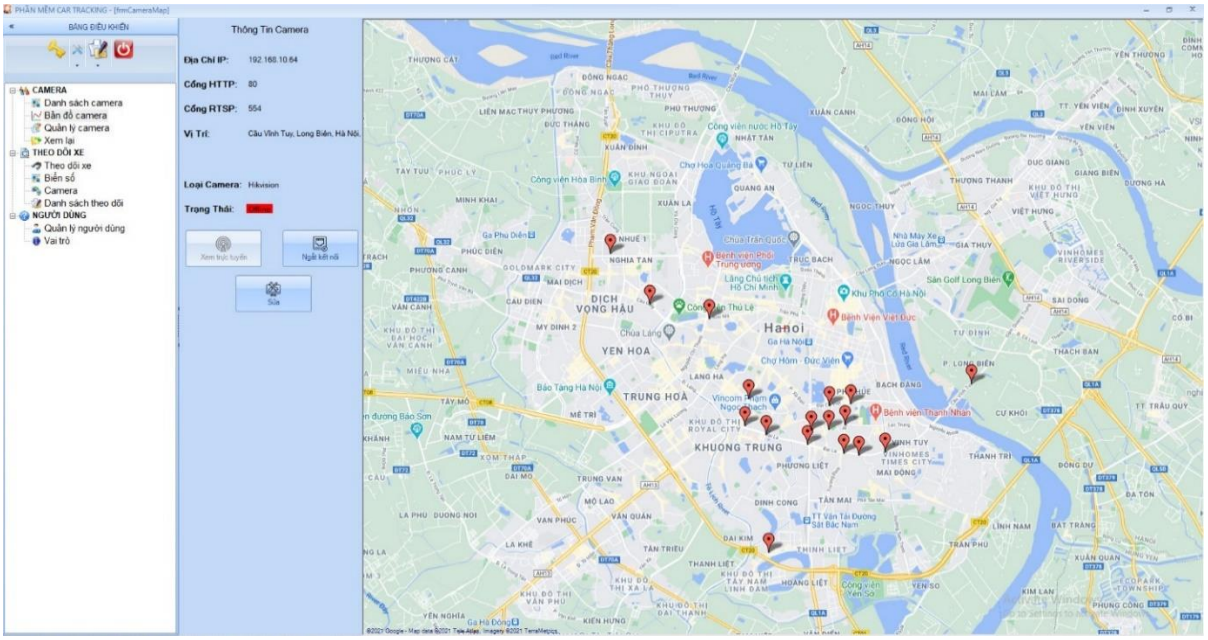

Figure 3.6. Camera map interface.

3.2.6.4. Camera list interface

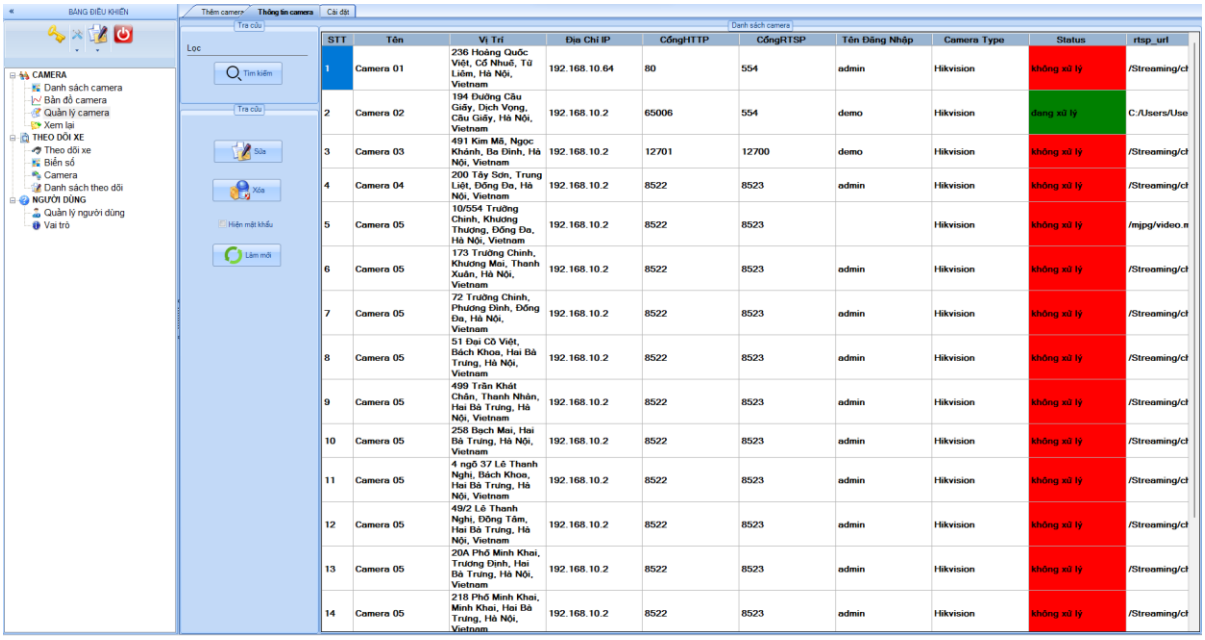

Figure 3.7. Camera list interface.

3.2.6.5. Interface for adding and updating camera information.

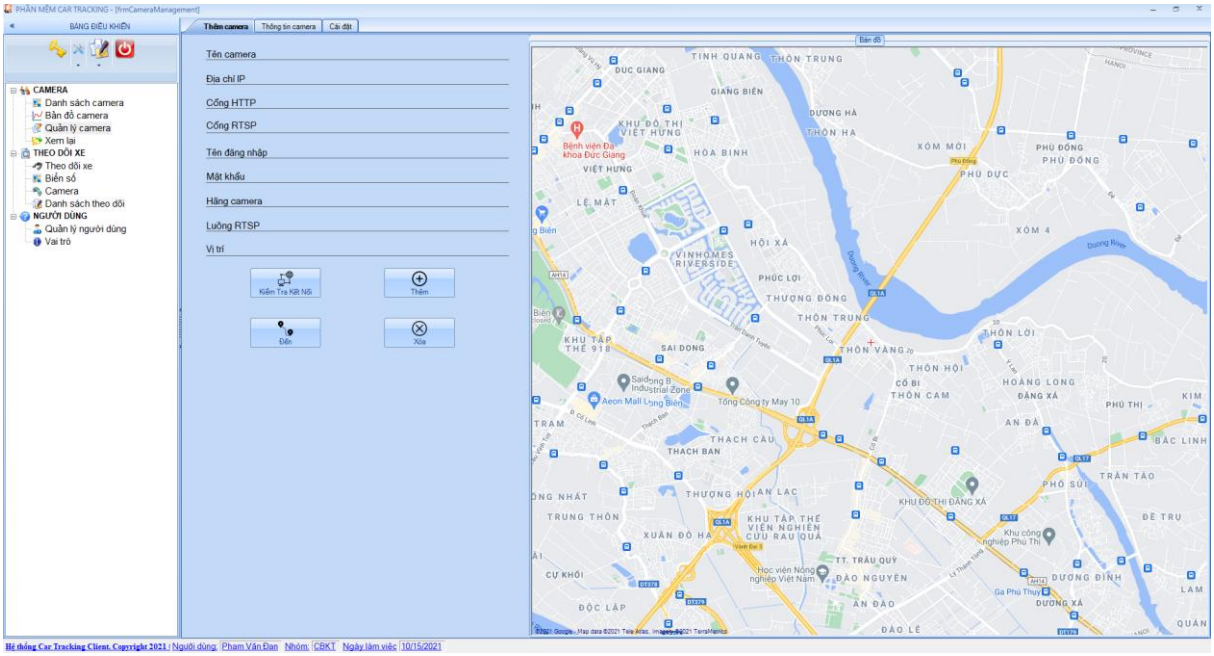

Figure 3.8. Interface for adding and updating camera information

3.2.6.6. Video playback interface

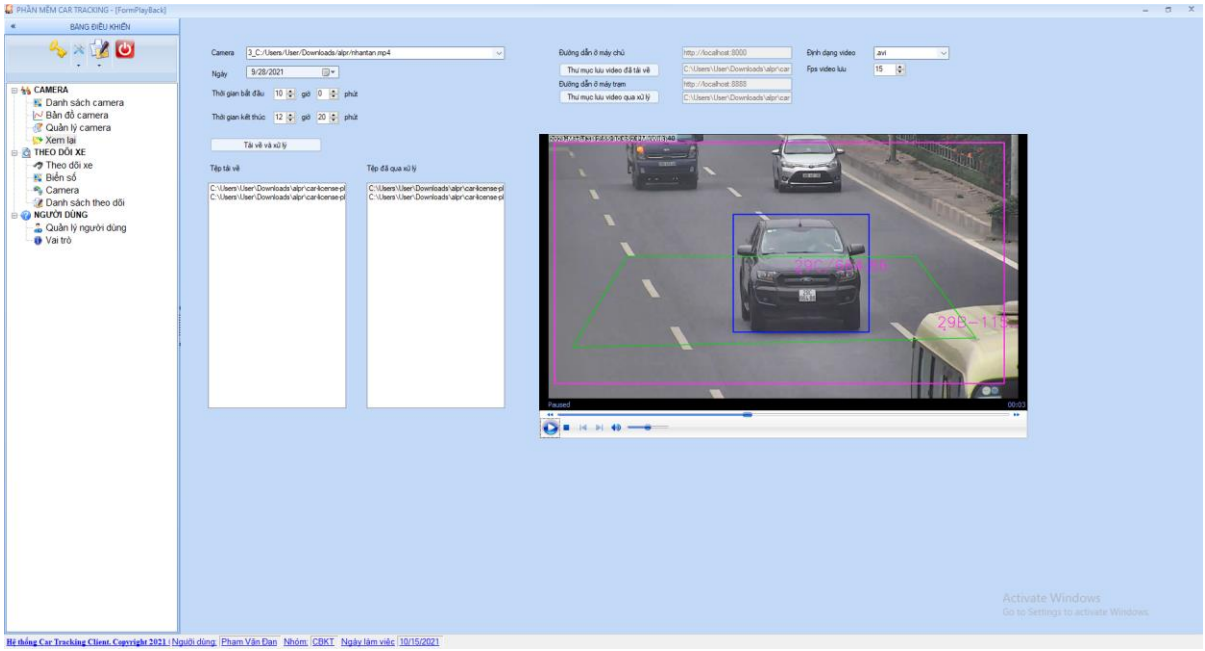

Figure 3.9. Video playback interface.

3.2.6.7. Single camera view interface

Figure 3.10. Single camera view interface

3.2.6.8. Vehicle tracking interface.

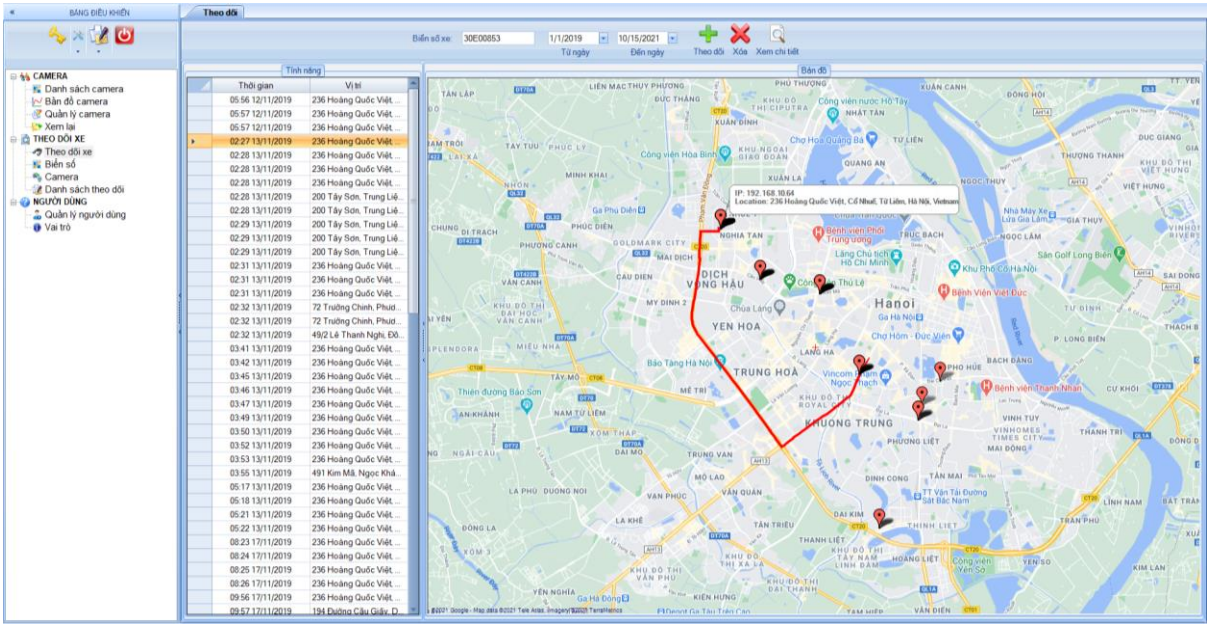

Figure 3.11. Vehicle tracking interface.

3.2.6.9. Interface to view license plate recognition data.

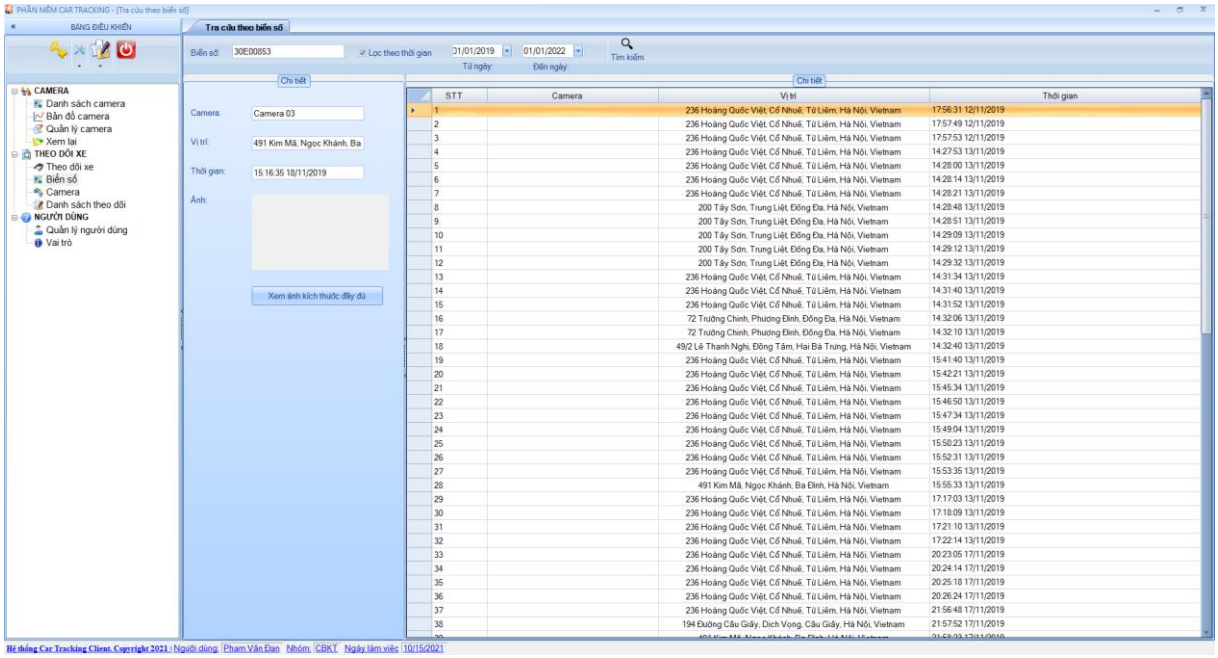

Figure 3.12. Number plate lookup interface.

3.2.6.10. Interface for viewing license plate recognition results by camera

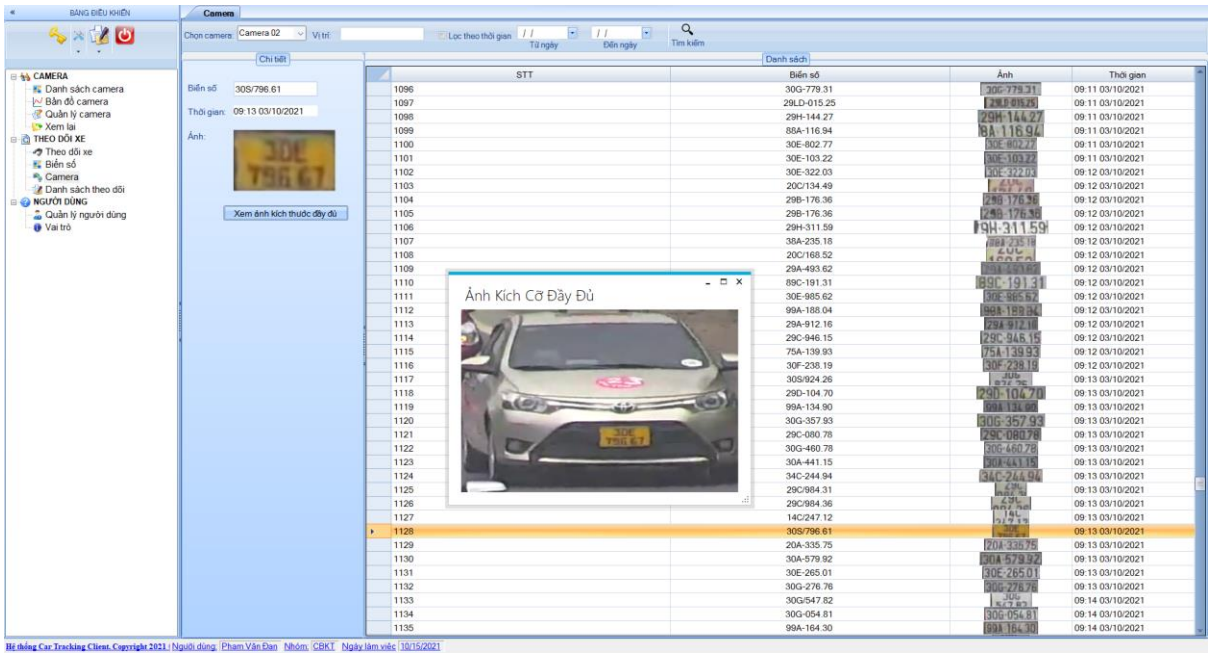

Figure 3.13. Result interface of number plate recognition by camera.

3.3. Model of deployment.

The system is divided into 3 main components:

- License plate recognition module
- Main server manages and processes data
- Client-side software

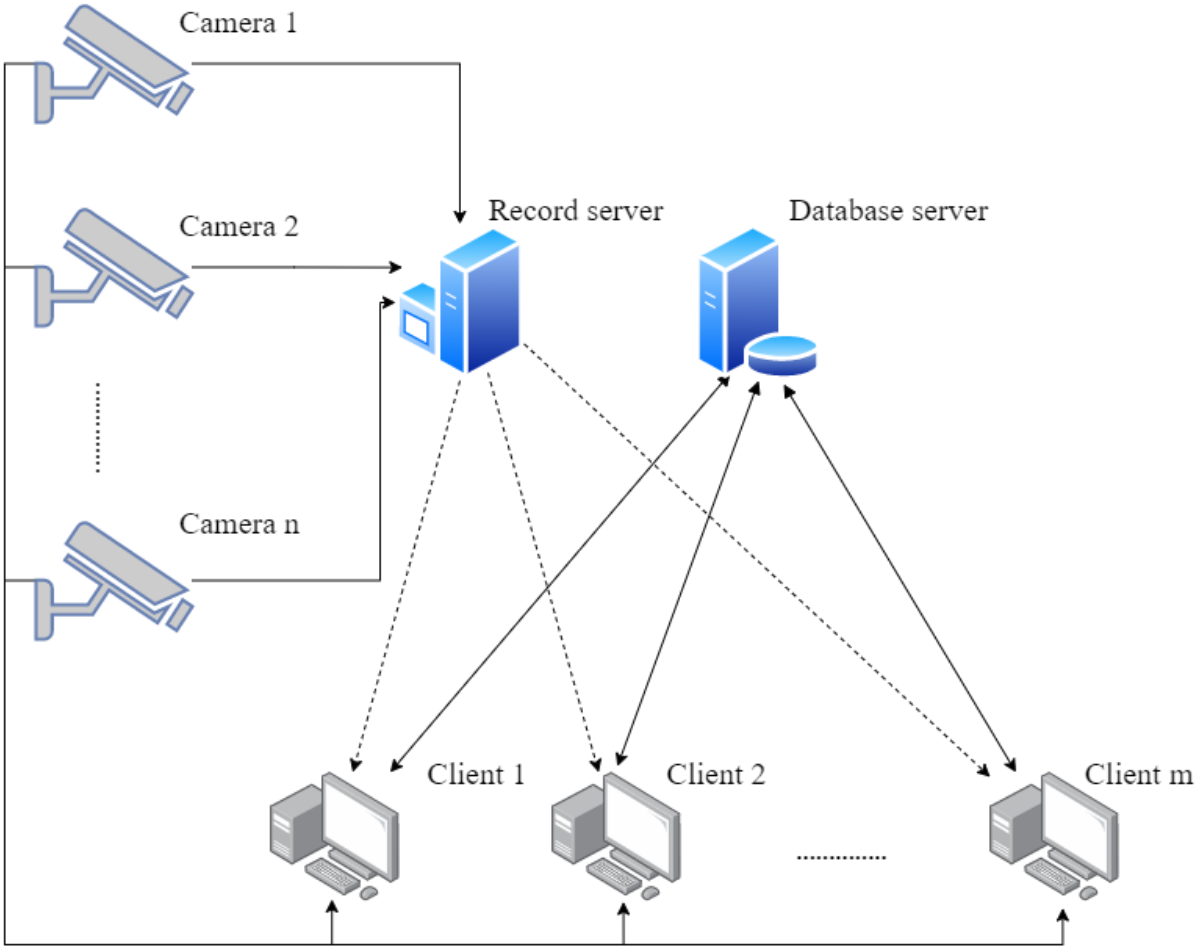

Figure 3.14. System model.

Advantages of our system model (Figure 3.14) can be proved by this example compare to traditional approaches where all processing like ALPR module, camera connection, database placed on a main server.

To demonstrate the benefit of our system model that is better than a main server model, let’s consider this problem: I suppose that we need to process total 90 cameras.

If each Client can process 3 cameras, we need: 30 Client (GPU) * 25 million= 750 million, 1 record server + 1 database server CPU (RAM > 16): 150 million. Therefore, the total cost is **900 million.**

In other case, if each main server can process 18 cameras, we need 30 Client (CPU) * 15 million= 450 million, 5 main servers (GPU: 16-32Gb; RAM: > 32Gb): 750 million. Therefore, the total cost is **1300 million**.

Therefore, our model is much more benefit for customers.

3.3.1. Record server

The Record server is responsible for storing the video from the video stream of the connected camera and also allowing the Client to download certain camera videos in a certain time period to perform video playback for the end user.

3.3.2. Database server

Database server is responsible for synchronizing data related to license plates from all clients:

- Receive and process data sent back from camera management servers, data storage and database management.
- Return query results for client users such as journey history, camera information, list of number plates.

3.3.3. Client

Each Client machine will be installed with a user interface and a license plate recognition module running as a local api. At the same time, each Client machine will be responsible for handling a certain amount of cameras depending on the machine's resources. The Client connects directly to the camera during processing. Users can perform basic system functions, especially allowing users to check the status of the camera, if the camera is in an active state, the Client will initiate a separate data stream for marine identification. vehicle number per camera.

3.4. Conclusion for chapter 3

In this chapter, we have built a system model of a car management system based on license plate recognition technology, built a license plate recognition module based on machine learning techniques, visually displayed on a digital map, built user interface, so that users can easily perform system operations.

CONCLUSION

The results have been achieved.

This work proposed a model for ALPR in Vietnam in unconstrained environment that focused on improvements on license plate detection and license plate OCR. Detecting the areas containing a plate number and recognizing LP characters. Our improvements have been empirically proved to be effective for Vietnamese ALPR in unconstrained environment. To tackle the problem of nonrectangular object detection, an approach based on keypoints detection was used and DDRNet23sh architecture was modified. This improvement get the highest mean IOU $mIOU = 95.01\%$ and precision $P_{75} = 99.5\%$. In this paper, we also propose VOCR based on segmentation-free that uses CNN architecture combined with encoder-decoder RNN and attention mechanism. The performance accuracy of our proposed architecture is up to $Acc_{\text{seq}} = 99.28\%$ and $Acc_{\text{char}} = 99.7\%$. We also built a dataset of Vietnamese car license plates named MTAVLP that is large and covers various conditions of image capturing.

Our proposed methods have been published to the fourth International Conference on Multimedia Analysis and Pattern Recognition (MAPR) supported by VAPR (Vietnamese Association on Pattern Recognition) was held in Ha Noi on October 15-16, 2021. The title of our paper is "*an efficient method to improve the accuracy of Vietnamese vehicle license plate recognition in unconstrained environment*" [11]. We received an award as the best paper in all of the conference papers. These showed that our proposed methods were so prominent for solving ALPR problem.

We have built a system model of a car management system based on license plate recognition technology, built a license plate recognition module based on machine learning techniques, visually displayed on a digital map, built user interface, so that users can easily perform system operations, specifically:

- The program is easy to use, meets the objective needs of vehicle management and monitoring based on license plate recognition technology.
- Design a basic program with basic functions that meet user requirements.
- Beautiful interface attracts users, does not cause boredom during operation.
- Support functions suitable for information search and disaster incident information system management.
- Applying digital maps to building user support systems.

Some limitations

Although this work has achieved certain results, a number of limitations have not yet been resolved due to limited capabilities and time conditions. Some limitations can be seen as follows:

- We have not had enough time to compare our method with other methods on foreign license plate datasets.
- We consider ALPR problem only on daytime, so in low light conditions or night time, the accuracy of our methods will be decreased.
- In case the traffic is congested and moving close together, the system will be slow because it has to be re-processed many times because the tracking algorithm is difficult to track the vehicles.
- The system has not been deployed for a long time, so there are still many errors arising during operation.

Future work

Comparing our method with other methods on foreign license plate datasets.

Considering ALPR problem not only on daytime but also in low light conditions or night time.

Researching license plate recognition models to increase accuracy as well as time processing.

Maintain and fix errors arising when deploying the system for a long time.

Complete, update more functions to improve user experience, as well as ensure reasonable use of system resources, reasonable data management.

REFERENCES

[1] R. Laroca et al. , "A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector," 2018.

[2] A. M. Al-Ghaili et al, "A new vertical edge detection algorithm and its application," 2008.

[3] M. Sarfraz, M. J. Ahmed, and S. A. Ghazi., "Saudi Arabian license plate recognition system," 2003.

[4] Vinh Mai, Duoqian Miao, and Ruizhi Wang, "Building a license plate recognition system for Vietnam tollbooth," 2012.

[5] Tran Duan et al., "Building an Automatic Vehicle License-Plate Recognition System," 2005.

[6] Y. Wen et al., "An Algorithm for License Plate Recognition Applied to Intelligent Transportation System," 2011.

[7] A. Rio-Alvarez et al., "Effects of Challenging Weather and Illumination on Learning-Based License Plate Detection in Noncontrolled Environments," 2019.

[8] Christos-Nikolaos Anagnostopoulos et al., "License Plate Recognition From Still Images and Video Sequences: A Survey," 2008.

[9] S. Du et al., "Automatic License Plate Recognition (ALPR): A Stateof-the-Art Review," 2013.

[10] Yuanduo Hong et al., "Deep Dual-resolution Networks for Real-time and Accurate Semantic Segmentation of Road Scenes," 2021.

[11] Khanh Nguyen, Dan Pham Van, Van Pham Thi Bich, "An efficient method to improve the accuracy of Vietnamese vehicle license plate recognition in unconstrained environment," *MAPR*, 2021.

[12] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning, "Effective Approaches to Attention-based Neural Machine Translation," 2015.

[13] Zisserman, Karen Simonyan and Andrew, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 2014.

[14] Junyoung Chung et al., "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," 2014.

[15] Y.-T. Chen, J.-H. Chuang, W.-C. Teng, H.-H. Lin, and H.-T. Chen, "Robust license plate detection in nighttime scenes using multiple intensity IR-illuminator," 2012.

[16] G. Heo, M. Kim, I. Jung, D.-R. Lee, and I.-S. Oh, "Extraction of car license plate regions using line grouping and edge density methods," 2007.

[17] T. Duan, D. Tran, P. Tran, and N. Hoang, "Building an automatic vehicle license-plate recognition system," 2005.

[18] W. Jia, H. Zhang, X. He, and Q. Wu, "Gaussian weighted histogram intersection for license plate classification," 2006.

[19] C.-T. Hsieh, Y.-S. Juan, and K.-M. Hung, "Multiple license plate detection for complex background," 2005.

[20] Zimmermann, J. Matas and K., "Unconstrained licence plate and text localization and recognition," 2005.

[21] S. Draghici, "A neural network based artificial vision system for licence plate recognition," 1997.

[22] Q. Wu, H. Zhang, W. Jia, X. He, J. Yang, and T. Hintz, "Car plate detection using cascaded tree-style learner based on hybrid object feature," 2006.

[23] K. Kim, K. Jung, and J. Kim, "Color texture-based object detection: An application to license plate localization," 2002.

[24] Z. Selmi, M. B. Halima, and A. M. Alimi, "Deep learning system for automatic license plate detection and recognition," 2017.

[25] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.

[26] R. Laroca, E. Severo, L. A. Zanlorensi, L. S. Oliveira, G. R. Goncalves,, "A robust real-time automatic license plate recognition based on the YOLO detector," 2018.

[27] Max Jaderberg et al., "Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition," 2014.

[28] B.-F. Wu, S.-P. Lin, and C.-C. Chiu, "Extracting characters from real vehicle licence plates out-of-doors," 2007.

[29] I. Paliy, V. Turchenko, V. Koval, A. Sachenko, and G. Markowsky, "Approach to recognition of license plate numbers using neural networks," 2004.

[30] S. M. a. C. Jung, "Real-time Brazilian license plate detection and recognition using deep convolutional neural networks," 2017.

[31] M.-S. Pan, J.-B. Yan, and Z.-H. Xiao, "Vehicle license plate character segmentation," 2008.

[32] C. N. E. Anagnostopoulos, I. E. Anagnostopoulos, V. Loumos, and E. Kayafa, "A license plate-recognition algorithm for intelligent transportation system applications," 2006.

[33] Zied Selmi, Mohamed Ben Halima, and Adel Alimi, "Deep Learning System for Automatic License Plate Detection and Recognition," 2017.

[34] Alex Graves et al., "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural," 2006.

[35] Shaoqing Ren et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," 2015.

[36] Teik Cheang, Yong Shean Chong, and Yong Haur Tay., "Segmentation-free Vehicle License Plate Recognition using ConvNet-RNN," 2017.

[37] Hui Li, Peng Wang, and Chunhua Shen., "Toward Endto-End Car License Plate Detection and Recognition With Deep Neural Networks," 2017.

[38] Feng Zhang et al., "Distribution-Aware Coordinate Representation for Human Pose Estimation," 2019.

[39] Jonathan Tompson et al., "Efficient object localization using Convolutional Networks," 2015.

[40] Fisher Yu et al., "Deep Layer Aggregation," 2019.

[41] Mark Everingham et al., "The Pascal Visual Object Classes (VOC) Challenge," 2010.

[42] Olga Russakovsky et al., "ImageNet Large Scale Visual Recognition Challenge," 2014.

[43] Tsung-Yi Lin et al., "Microsoft COCO: Common Objects in Context," 2015.

[44] Xingyi Zhou, Dequan Wang, and Philipp Kr¨ahenb¨uhl, "Objects as Points," 2019.

[45] Bin Xiao, Haiping Wu, and Yichen Wei., "Simple Baselines for Human Pose Estimation and Tracking," 2018.

[46] Alejandro Newell, Kaiyu Yang, and Jia Deng. , "Stacked Hourglass Networks for Human Pose Estimation," 2016.

[47] Faisal Shafait and Thomas Breuel., "The Effect of Border Noise on the Performance of Projection-Based Page Segmentation Methods," 23.

[48] Dzmitry Bahdanau, Kyunghyun Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," 2014.

[49] Rafael Muller, Simon Kornblith, and Geoffrey E. Hinton., "When Does Label Smoothing Help?," 2019.

[50] "https://thigiacmaytinh.com/tai-nguyen-xu-ly-anh/tonghop-data-xu-ly-anh/," 2014.