A red and black rectangular sign

Description automatically generated

PROJECT REPORT LICENSE PLATE RECOGNITION

GROUP 2

Leader: Hai Anh - 104177513  
Member:

* Nghiem Tuan Linh - 104187363
* Phan Huy Quang - 104177128
* Nguyen Duc Hieu - 10358217

**Table of Contents**

INTRODUCTION 1

THE INITIAL SOLUTION 4

IMPROVEMENT 19

TEAM SUMMARY REPORT

REFERENCES

# INTRODUCTION

License Plate Recognition (LPR) is an advanced technology designed to read and interpret vehicle registration plates. It plays a crucial role in automating the process of vehicle identification and tracking by extracting and analysing licence plate information from images or video sequences.

LPR systems use sophisticated image processing techniques and optical character recognition (OCR) algorithms to detect and decode the alphanumeric characters on a licence plate. This process involves several key stages, including image acquisition, pre-processing to enhance the image quality, plate detection to locate the plate within the image, character segmentation to isolate individual characters, and finally, recognition to translate the segmented characters into readable text.

The technology is widely used in various applications, such as traffic management, law enforcement, parking solutions, and toll collection. For instance, in traffic management, LPR can monitor and manage vehicle flow on highways, while in law enforcement, it assists in identifying stolen vehicles or tracking offenders. In parking management, LPR can automate access control and payment processing, enhancing convenience and efficiency for users.

Additionally, the integration of LPR with other technologies, such as machine learning and artificial intelligence, has further improved its accuracy and adaptability. These advancements enable LPR systems to handle diverse licence plate designs and varying environmental conditions, making the technology increasingly effective in real-world scenarios.

## Steps of Plate recognition

1. **Image Capture:**  
   The process begins with high-speed cameras that capture images of vehicles as they move. These cameras are designed to operate efficiently at varying speeds and conditions, ensuring that clear and accurate images are obtained.
2. **Image Preprocessing:**  
   Once the images are captured, they undergo preprocessing to enhance quality and reduce noise. This step involves techniques such as adjusting contrast, brightness, and sharpness, as well as applying filters to remove distortions or artefacts that could interfere with subsequent processing stages.
3. **Detection:**

* **Licence Plate Detection:** In this stage, the system identifies and locates the licence plate within the image. This involves detecting the plate’s position and orientation, often using machine learning algorithms or traditional image processing methods to isolate the plate from the rest of the image.
* **Character Segmentation/Detection**: After the plate is detected, the next step is to isolate the individual characters on the plate. This involves segmenting the plate into its constituent characters, making it easier for the OCR system to recognize each character separately.

1. **Optical Character Recognition (OCR)**: OCR technology is then applied to convert the segmented characters into readable text. This step involves analysing the shapes and patterns of the characters and matching them to a predefined set of possible characters to produce accurate text output.
2. **Data Analysis**: The recognized text is then analysed and cross-referenced with databases for information retrieval. This could involve checking against vehicle registration databases, law enforcement records, or parking management systems to retrieve relevant information about the vehicle.
3. **Licence Plate Localization**: Throughout the process, the system must effectively detect and localise the licence plate within each input image or video frame. This step ensures that the plate’s location is accurately identified so that the OCR and data analysis stages can proceed effectively.

## Input and output

A black background with white text

Description automatically generated

## Licence plate detection

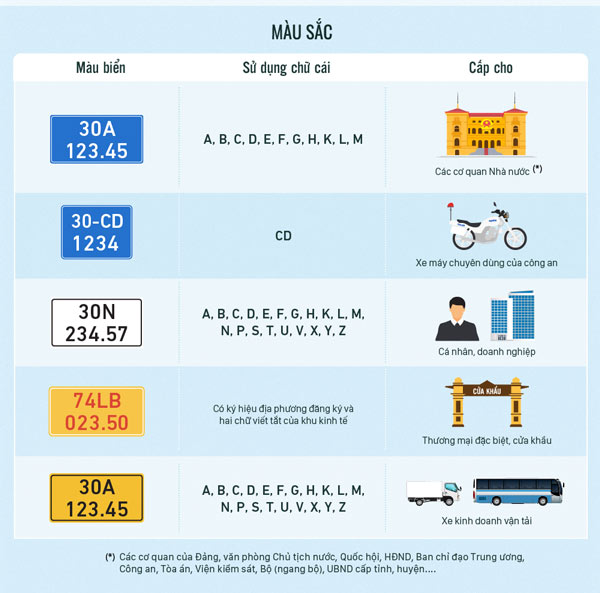
There are 2 types of licence plate that we would like to detect:

| 2 line plate | 1 line plate |
| --- | --- |

## Character recognition

According to the Vietnamese licence plate rules, the characters will be classified into 32 classes:

* 20 letters: A, B, C, D, E, F, G, H, K, L, M, N, P, S, T, U, V, X, Y, Z
* 10 numbers: 1,2,3,4,5,6,7,8,9,0
* 2 special characters: -, .



## Initial Solution

A diagram of a process

Description automatically generated

## Image preprocessing

There are 4 main stages in preprocessing the image:

1. Grey scaling
2. Increasing the contrast level
3. Noise Decreasing by Gaussian filter
4. Adaptive threshold for image binarization

A diagram of a process

Description automatically generated

### Gray Scaling:

The purpose of grey scaling is to simplify the image by reducing its complexity, thereby making it more suitable for processing with various image analysis algorithms, such as the Canny edge detection algorithm. Grey scaling converts images that are initially displayed in colour formats, such as RGB (Red, Green, Blue), CMYK (Cyan, Magenta, Yellow, Black), or HSV (Hue, Saturation, Value), into different shades of grey.

In a colour image, each pixel contains multiple values representing different colour channels (typically three for RGB or four for CMYK). By converting the image to grayscale, the colour information is removed, and each pixel is represented by a single intensity value. This reduction simplifies the image from a multi-dimensional space (e.g., 3 or 4 channels) to a single dimension, where each pixel's intensity represents varying shades of grey.

This simplification has several advantages:

* **Reduced Computational Complexity**: Grey scaled images require less computational power and memory, as they consist of a single channel of intensity values instead of multiple colour channels.
* **Improved Algorithm Performance**: Algorithms like Canny edge detection work more efficiently on grayscale images because they only need to analyse intensity variations rather than complex colour variations. This can lead to faster processing times and more accurate results.
* **Enhanced Focus on Structural Information**: Grayscale images highlight structural and geometric features of the image without the distraction of colour information, making it easier to perform tasks such as edge detection, object recognition, and pattern analysis.

By converting to grayscale, the image’s initial multi-dimensional data is condensed into a more manageable format, which facilitates more effective and efficient processing and analysis.

A comparison of a car

Description automatically generated

A screen shot of a computer screen

Description automatically generated

### Maximise contrast

In the process of image dilation, we enhance the contrast between the elements in the image by making the black elements appear darker and the bright elements become even brighter. This technique is particularly useful in the context of licence plate recognition, where it helps to improve the visibility of the licence plate characters.

Dilation is an image processing operation where pixels in a binary or grayscale image are expanded. During dilation:

* **Dark Elements**: The black or darker elements in the image are intensified, making them even darker. This helps in emphasising the features that are already dark.
* **Bright Elements**: The bright elements are enhanced to become even brighter, which increases their contrast relative to the background.

This enhancement increases the separation between the characters on the licence plate and the background. By amplifying the differences in intensity, dilation makes the edges of the characters more pronounced, which aids in distinguishing them from the plate background. This is crucial for subsequent processing stages, such as character segmentation and optical character recognition (OCR), where clear and well-defined characters are essential for accurate recognition.

Overall, dilation helps to improve the clarity and contrast of the image, facilitating more effective extraction and analysis of the licence plate details.

A close-up of a parking meter

Description automatically generated

| Before | After |
| --- | --- |

### Gauss filter

The Gaussian filter is a widely used image processing technique designed to reduce noise and blur the surroundings, thereby emphasising the central point of interest in an image. This filter works by applying a Gaussian function to the image, which smooths and blurs the image based on a bell-shaped curve.

**Key Aspects of the Gaussian Filter:**

1. **Noise Reduction**: The primary function of the Gaussian filter is to minimise noise in the image. By averaging the pixel values within a neighbourhood defined by the Gaussian function, the filter smooths out variations caused by random noise, making the image cleaner and more uniform.
2. **Blurring**: The Gaussian filter blurs the image by averaging pixel values in a weighted manner. Pixels closer to the centre of the filter’s kernel are given higher weights, while pixels further away are weighted less. This results in a gradual smoothing effect, where the central point remains relatively sharp while the surrounding area becomes increasingly blurred.
3. **Focus on Central Point**: By reducing the sharpness of the surrounding areas, the Gaussian filter helps to focus attention on the central point of the image. This is particularly useful in scenarios where you want to highlight specific features or objects in the centre of the image while minimising the impact of peripheral details.

| Before | After |
| --- | --- |

### 4. Image Binarization

Converting a grey-scaled  image into a binary image, which only contain 0(black) and 1(white)

Parameters:

* M(x,y): Intensity of position x,y on grey scaled
* T: a adaptive threshold - generated with openCV
* MB(x,y): Intensity at position x,y on binary image

Conversion:

* M(x,y) >= T => MB = 0
* M(x, y) < T => MB = 1

A black and white image of a motorcycle

Description automatically generated

## Licence Plate detection

There are 2 main stages in detecting and extracting the licence plate

1. Canny Edge detection
2. Detect the plate by drawing contours and if..else

A black grid with red rectangles and white text

Description automatically generated

### 1. Canny edge detection

Use 3x3 filters applied to each pixel of the image to indicate their gradient: G = √(Gx^2 + Gy^2)

We compare the central pixel with its neighbour pixels in 3 directions:

* 0 degree angle: horizontal
* 90 degree angle: vertical
* 45 degree angle: diagonal

We will only keep its gradient if it is the highest value in the neighbourhood, else we will set it gradient to 0, which represent black

A black and white image of a motorcycle

Description automatically generated

### 2. Drawing contour - openCV

* Using Suzuki’s tracing algorithm in the OpenCV library. The agent will start from the most bottom left pixel and changes direction according to the colour of the pixel it moves into (0 or 255)

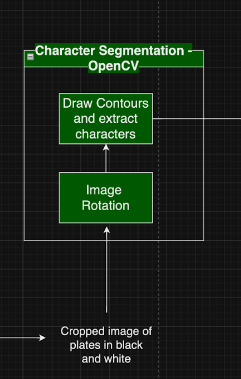


* The routes that looks like a 4 sides polygon will be identified as licences and stored using the (x,y) coordinates of the

A close up of a license plate

Description automatically generated

## Character segmentation



### Rotate the image to the right orientation

To correct the orientation of a licence plate image, we first need to identify the coordinates of two vertices, A and B, located at the bottom of the drawn contour of the number plate. Let's denote these vertices as A(x1, y1) and B(x2, y2).

**Steps to Calculate the Rotation Angle and Adjust the Image Orientation:**

1. **Identify Vertices Coordinates**:
   * Extract the coordinates of the vertices A(x1, y1) and B(x2, y2) from the contour of the licence plate.
2. **Calculate the Rotation Angle**:
   * Use the arctangent function to calculate the angle of rotation. The angle θ\thetaθ can be calculated using the following formula:
   * This angle represents the inclination of the line segment AB with respect to the horizontal axis.
3. **Determine Rotation Direction**:
   * If vertex A is lower than vertex B (i.e., y1 < y2), the angle θ is positive, indicating that the image should be rotated clockwise.
   * If vertex A is higher than vertex B (i.e., y1 > y2), the angle θ is negative, indicating that the image should be rotated counterclockwise (anticlockwise).
4. **Rotate the Image**:
   * Rotate the image according to the calculated rotation angle θ:
     + **Clockwise Rotation**: Apply a clockwise rotation when θ is positive.
     + **Counterclockwise Rotation**: Apply a counterclockwise rotation when θ is negative.
   * The rotation can be performed using image processing libraries such as OpenCV. For instance, in OpenCV, the cv2.getRotationMatrix2D and cv2.warpAffine functions can be used to apply the rotation transformation to the image.

A close up of a license plate

Description automatically generated

### Drawing the contour for the letters

To accurately identify and isolate the characters on a licence plate, we need to reconstruct the contour of the letters from the binary image, where the characters are represented by the white parts. Then, we draw rectangles around these characters. However, accurately locating these contours can be challenging due to potential inaccuracies and the detection of non-character objects. To improve the accuracy of our character detection, we can use specific criteria based on the height/width ratio of the characters and their area in comparison to the overall licence plate.

A close-up of numbers

Description automatically generated

### Character recognition using K nearest neighbour

K-Nearest Neighbors (KNN) is one of the simplest and most intuitive supervised learning algorithms in machine learning. It operates through a straightforward four-step process:

1. **Determine the Parameter K (Number of Nearest Neighbors)**:
   * The first step in implementing the KNN algorithm is to choose the value of K, which represents the number of nearest neighbours to consider when making a classification or regression decision. The choice of K can significantly impact the algorithm's performance. A smaller K makes the model sensitive to noise, while a larger K smooths out the classification boundaries but may include points from other classes.
2. **Calculate the Distance from the Point in Question to All Points in the Data Set**:
   * Next, for a given query point (the point to be classified), calculate the distance between this point and all other points in the training data set. Various distance metrics can be used, such as Euclidean distance, Manhattan distance, or Minkowski distance, depending on the specific requirements of the problem.
   * **Euclidean Distance Formula**:

A black background with white text

Description automatically generated

Where and are the feature vectors of the query point and a point in the training set, respectively.

1. **Sort the Distances in Ascending Order**:
   * Once the distances are calculated, sort them in ascending order. This will arrange the data points such that the closest points to the query point are at the beginning of the list.
2. **Classify the Query Point Based on the Majority Vote of its Neighbours**:
   * Select the K closest points from the sorted list. These K points are the nearest neighbours to the query point.
   * For classification tasks, assign the class to the query point based on the majority class among the K nearest neighbours. If a majority of the nearest neighbours belong to a particular class, the query point is classified as belonging to that class.
   * **Example**: Suppose K=3 and the three nearest neighbours have the classes {0, 1, 1}. Since there are more '1's, the query point is classified as '1'.
   * For regression tasks, the value assigned to the query point can be the average of the values of its K nearest neighbours.

A diagram of a circle with a star and squares

Description automatically generated

**Normalisation**:

To facilitate processing, it is essential to normalise each character in the licence plate image to a consistent size, given that characters can vary in dimensions. We normalise the characters to a fixed height-to-width ratio of 30:20 pixels. This standardisation ensures uniformity and simplifies further processing.

**Labelling Points**:

Our labelled points include the characters that can appear on a licence plate. These characters are stored with their corresponding features, which include the normalised images and rotation angles ranging from 1 to 10 degrees. This helps in recognizing characters that might be slightly rotated due to the angle at which the image was captured.

**Processing Steps**:

1. **Input Image Processing**:
   * Load the input image of the licence plate.
   * Convert the image to grayscale and apply thresholding to obtain a binary image.
   * Detects and extracts individual character contours from the binary image.
2. **Normalisation**:
   * Resize each detected character contour to the standard size of 30 x 20 pixels. This normalisation ensures that each character is processed uniformly, regardless of its original size.
3. **Rotation Correction**:
   * For each normalised character image, consider the possible rotations from 1 to 10 degrees. This helps in accounting for minor misalignments in the captured image.
   * Generate rotated versions of each character image within this range to match against the labelled points.
4. **Distance Calculation**:
   * For the given input image (normalised and potentially rotated), calculate the distance to all labelled points in the dataset. This distance is typically computed using a suitable metric, such as Euclidean distance.
   * The distance calculation helps in determining the similarity between the input character and the labelled characters.
5. **Character Recognition**:
   * The character corresponding to the minimum distance is identified as the matching character.
   * This character is then translated to its ASCII code, representing the recognized character.
6. **Licence Plate Number Extraction**:
   * Repeat the above steps for each character in the licence plate.
   * Concatenate the recognized characters to form the complete licence plate number.
   * Print out the final licence plate number.

A group of letters on a white background

Description automatically generated

A close-up of numbers

Description automatically generated A white sign with black numbers and green border

Description automatically generated

# THE INITIAL SOLUTION

### Challenges and Limitations of License Plate Recognition Using KNN:

1. **Plate Detection Issues**:
   * The current plate detection algorithm is not optimised for plates that have additional elements such as drawings, decorations, or unusual fonts. These elements can interfere with the accurate detection and segmentation of characters, leading to erroneous results.
2. **Low Recognition Ability of KNN**:
   * The K-Nearest Neighbors (KNN) method, while simple and easy to implement, has relatively low recognition accuracy in this context. It often misclassifies characters, particularly when the characters have similar shapes or when the images are not clear. This is because KNN relies heavily on the distance metric and the quality of the training data.
3. **High Runtime Due to Scanning Through All Training Data**:
   * KNN has a high computational cost, especially with large datasets. Since it needs to calculate the distance between the query point and all points in the training dataset, the runtime increases significantly with the size of the training data. This makes real-time processing difficult.
4. **Poor Performance in Adverse Conditions**:
   * The algorithm performs poorly under conditions of poor lighting, reflections, and blurriness. Variations in lighting can cause shadows and highlights that distort the characters. Reflections can create misleading bright spots, and blurriness reduces the clarity of character edges, all of which contribute to inaccurate recognition.

A license plate on a motorcycle

Description automatically generated

A license plate with black numbers and purple lights

Description automatically generated

A white rectangular sign with black numbers

Description automatically generated A close up of a license plate

Description automatically generated A screenshot of a computer screen

Description automatically generated

Does not find character Incorrect recognition: A and 4 Incorrect recognitions B and 8

# IMPROVEMENTS

A black screen with white text

Description automatically generated

A screen shot of a computer

Description automatically generated

## Upgrading the licence plate recognition technology

Initially, our system employed a custom plate recognition technology. However, the accuracy of this model was suboptimal, as it often failed to recognize licence plates in various situations. To address these shortcomings, we decided to upgrade our approach.

After conducting extensive research, we ultimately chose to implement YOLO version 7 for licence plate detection. YOLO v7 is a state-of-the-art object detection model known for its high accuracy and real-time performance. By leveraging YOLO v7, we aimed to significantly enhance the detection capabilities of our system.

To train our YOLO v7 model, we utilised a dataset sourced from Kaggle, specifically curated to improve the recognition of Vietnamese licence plates. This dataset provided a diverse range of licence plate images, allowing our model to learn and generalise better to real-world scenarios.

Despite the substantial improvements achieved with YOLO v7, there are still some limitations. The model sometimes fails to detect licence plates that are too far away or blurred due to reflections. These edge cases highlight the ongoing challenges in achieving perfect detection accuracy under all conditions.

### Implementation Steps:

1. **Data Collection and Preparation**:
   * Gather a diverse dataset of Vietnamese licence plates from Kaggle, ensuring it includes various plate styles, lighting conditions, and angles.
   * Annotate the dataset with bounding boxes around the licence plates.
2. **Model Training**:
   * Use the annotated dataset to train the YOLO v7 model. The training process involves fine-tuning the pre-trained YOLO v7 weights on our specific dataset to enhance its ability to detect Vietnamese licence plates accurately.
3. **Integration and Testing**:
   * Integrate the trained YOLO v7 model into our licence plate recognition system.
   * Conduct extensive testing to evaluate the model's performance, focusing on detection accuracy and real-time processing capabilities.
4. **Addressing Limitations**:
   * Identify and analyse the cases where the model fails to detect licence plates, such as those that are too distant or blurred by reflections.
   * Consider additional preprocessing steps, such as image enhancement techniques, to mitigate these issues.

A diagram of a computer network

Description automatically generated

### YOLO Algorithm Overview:

The YOLO algorithm is a highly efficient and accurate object detection method that uses a single deep convolutional neural network with 26 layers. Here's a detailed breakdown of its structure and training process:

1. **Network Architecture**:
   * The network consists of 24 convolutional layers followed by 2 fully connected layers.
   * The convolutional layers are responsible for feature extraction from the input image, while the fully connected layers predict the class probabilities and bounding box coordinates.
2. **Pre-training**:
   * The first 20 convolutional layers of the model are pre-trained on the ImageNet dataset, which contains millions of images across various categories.
   * During pre-training, these layers are connected to a temporary average pooling layer and a fully connected layer to perform classification tasks on ImageNet.
3. **Conversion for Detection**:
   * After pre-training, the model is adapted for object detection tasks. The temporary average pooling and fully connected layers are removed.
   * The network is modified to include 2 final fully connected layers. These layers predict both the class probabilities of the detected objects and the coordinates of the bounding boxes surrounding the objects.
4. **Detection Process**:
   * The input image is divided into an SxS grid (e.g., 7x7). Each grid cell is responsible for detecting objects whose centre falls within the cell.
   * Each grid cell predicts B bounding boxes and their confidence scores. The confidence score reflects the accuracy of the bounding box and whether it contains an object.
   * Each bounding box consists of 5 predictions: x and y coordinates of the box centre relative to the grid cell, width and height relative to the entire image, and a confidence score.
   * Each grid cell also predicts the class probabilities for the detected objects.
5. **Training**:
   * The model is trained on a large dataset annotated with bounding boxes and class labels.
   * The loss function combines classification loss (measuring the error in predicted class probabilities), localization loss (measuring the error in bounding box coordinates), and confidence loss (measuring the error in the confidence scores).

### Why we choose YOLO version 7:

Licence plate recognition systems need to operate both quickly and accurately, especially in scenarios where vehicles are moving and need to pass through detection points without delay. YOLO v7 is particularly well-suited for this task due to its design and capabilities.

**Speed**

**Fast Detection and Recognition**:

* **Single Stage Object Detection**: YOLO v7 operates as a single stage object detection method, meaning it performs both object localization and classification in a single forward pass through the network. This is in contrast to two-stage detectors like Faster R-CNN, which require multiple passes and hence are slower.
* **Real-time Performance**: YOLO v7 is designed for real-time applications, capable of processing images at high speeds. This is crucial for licence plate recognition, where vehicles are often in motion and rapid detection is essential to minimise delays.

**Accuracy**

**Maintained Accuracy for Licence Plates**:

* **Sufficient for Simpler Classifications**: Although single-stage detectors like YOLO v7 are generally considered less accurate than two-stage detectors, this difference is less significant for tasks with a limited number of classes. Licence plates typically involve recognizing a small set of characters (around 30-40 classes, including letters and digits). This reduces the complexity of the classification task, ensuring that the slight trade-off in accuracy is acceptable.
* **Focused Training**: By training the YOLO v7 model specifically on datasets containing licence plates (e.g., from Kaggle), we can fine-tune the network to achieve higher accuracy for this specific task. This focused training helps the model better understand the nuances of licence plate images, further improving its performance.

## Transition from KNN to PaddleOCR for License Plate Recognition:

Initially, this project employed a custom KNN (K-Nearest Neighbors) model to recognize characters on licence plates. However, the accuracy of this approach was suboptimal, with some licence plates failing to be recognized clearly. Recognizing the need for a more robust solution, we explored various technologies and ultimately decided to switch from the custom KNN model to PaddleOCR for character segmentation and recognition.

**Issues with KNN Model**

**Custom KNN Model**:

* **Low Accuracy**: The custom KNN model struggled with accurately recognizing characters on licence plates. This led to frequent misrecognitions and unreliable results.
* **Recognition Challenges**: The model had difficulty distinguishing between similar-looking characters, which affected its overall performance.

**Switching to PaddleOCR**

**Research and Decision**:

* After thorough research, we identified PaddleOCR as a promising alternative. PaddleOCR is a comprehensive Optical Character Recognition (OCR) toolkit developed by PaddlePaddle, specifically designed for high-accuracy text recognition tasks.

**Implementation and Benefits**:

* **Improved Accuracy**: By integrating PaddleOCR into our system, we achieved a significant improvement in recognition accuracy. The new model can recognize most licence plates with an accuracy of over 90%.
* **Segmentation and Recognition**: PaddleOCR includes advanced segmentation capabilities, which accurately isolate and identify individual characters on the licence plates.
* **Error Reduction**: While the new model is highly accurate, it occasionally confuses the characters "0" and "D". Despite this minor issue, the overall performance is substantially better than the custom KNN model.

**Technical Details**

**PaddleOCR Integration**:

* **Preprocessing**: The input images are first processed to enhance their quality and make them suitable for OCR. This includes steps like resizing, noise reduction, and contrast adjustment.
* **Character Segmentation**: PaddleOCR's built-in segmentation module isolates individual characters from the licence plate.
* **Character Recognition**: The segmented characters are then passed through PaddleOCR's recognition module, which converts the images of characters into text.

## PaddleOCR for Enhanced License Plate Recognition:

PaddleOCR, developed by Baidu's Paddle platform, is an efficient and powerful Optical Character Recognition (OCR) system. Leveraging deep learning, it provides accurate text detection and recognition, making it ideal for various applications, including licence plate recognition.

**Key Features and Capabilities**

**Two-Stage Pipeline**:

* **Text Detection**: PaddleOCR employs a Differentiable Binarization (DB) algorithm for text detection. This method is highly effective in identifying text regions within an image, even under challenging conditions.
* **Text Recognition**: For text recognition, PaddleOCR uses a Convolutional Recurrent Neural Network (CRNN). This network combines convolutional layers for feature extraction with recurrent layers for sequence modelling, ensuring high accuracy in recognizing characters.

**Multi-Language Support**:

* PaddleOCR supports multiple languages, making it versatile for use in various regions and applications. This is particularly beneficial for recognizing licence plates from different countries with diverse character sets.

**Data Augmentation**:

* To enhance robustness, PaddleOCR uses data augmentation techniques during training. This helps the model generalise better to different lighting conditions, distortions, and noise, improving its performance in real-world scenarios.

**Optimised for Cloud and Edge Deployment**:

* PaddleOCR is optimised for both cloud and edge deployment. This flexibility allows it to be used in various environments, from powerful servers to resource-constrained edge devices, ensuring quick and efficient processing.

**Application in License Plate Recognition**

**Integration and Benefits**:

* **Enhanced Accuracy**: By incorporating PaddleOCR into our licence plate recognition system, we have significantly improved accuracy. The system can now recognize most licence plates with an accuracy exceeding 90%.
* **Advanced Segmentation**: PaddleOCR’s segmentation capabilities accurately isolate individual characters on licence plates, addressing the challenges faced with the initial custom KNN model.
* **Error Handling**: While the new system occasionally confuses similar-looking characters (e.g., "0" and "D"), the overall recognition performance is robust and reliable.

## PaddleOCR’s CRNN Architecture:

PaddleOCR follows the CRNN (Convolutional Recurrent Neural Network) architecture, which is highly effective for text recognition tasks. This architecture consists of three key layers:

**1. Convolutional Neural Network (CNN)**

**Feature Extraction**:

* **Convolutional Layers**: These layers apply convolution operations to the input image, extracting essential features such as edges, textures, and patterns. The convolutional layers help in identifying distinct characteristics of the characters in the licence plate.
* **Max-Pooling Layers**: These layers perform down-sampling operations to reduce the spatial dimensions of the feature maps, thereby retaining the most significant features while reducing computational complexity. The output of the CNN layers is a sequence of feature vectors (feature maps).

**2. Recurrent Neural Network (RNN)**

**Sequence Label Prediction**:

* **RNN Layers**: Built on top of the CNN, the RNN layers process the sequential data from the feature maps produced by the CNN. RNNs are well-suited for handling sequence data, making them ideal for recognizing the sequential nature of text.
* **Label Prediction**: The RNN predicts a label for each feature vector or frame in the sequence. Mathematically, for each frame xtx\_txt​ in the feature sequence x=(x1,x2,...,xt)x = (x\_1, x\_2, ..., x\_t)x=(x1​,x2​,...,xt​), the RNN predicts a corresponding label yty\_tyt​. This step is crucial for character-level recognition.

**3. Transcription Layer**

**Final Sequence Translation**:

* **Transcription**: This layer translates the per-frame predictions from the RNN into a coherent final sequence of characters. It uses techniques like the Connectionist Temporal Classification (CTC) to align the predicted labels with the input sequence, ensuring the output is a readable text string.
* **Highest Probability Selection**: The transcription layer selects the sequence with the highest probability, ensuring that the final output is the most likely interpretation of the input image.

### Why PaddleOCR > VietOCR

| Model | VietOcr | Paddle Ocr |
| --- | --- | --- |
| Dataset | Using more hand-written and documentation data to train | Using more street images to train |
| Language support | Vietnamese only | Supports multiple languages |
| Features | Optimised for Vietnamese text | Optimised for more versatile choice of language |
| Ease of use | Easier to use | Harder to use |
| Performance and accuracy | High performance on Vietnamese text | High performance on multiple language text |

### Demo of improvement

A group of arrows pointing to different images

Description automatically generated

# TEAM SUMMARY REPORT

| Name | Tasks |
| --- | --- |
| Tran Hoang Hai Anh | Implementing PaddleOCR, Testing test samples |
| Nghiem Tuan Linh | Implementing PaddleOCR, Design the architecture |
| Phan Huy Quang | Train the YOLO, Design the architecture |
| Nguyen Duc Hieu | Train the YOLO, Collecting test samples |

# REFERENCES

* Marzuki, P., Radzi, Feeza, Wong, Yan Chiew, Abdul Hamid, Norihan, Ali, Nur, & Mat Ibrahim, Masrullizam. (2019). A design of licence plate recognition system using convolutional neural networks. *International Journal of Electrical and Computer Engineering (IJECE)*, 9, 2196-2204. https://doi.org/10.11591/ijece.v9i3.pp2196-2204
* YOLO Object Detection. V7 Labs. Retrieved from <https://www.v7labs.com/blog/yolo-object-detection>
* How Paddle OCR Reads Image Data. Aurigait. Retrieved from <https://aurigait.com/blog/how-does-paddle-ocr-reads-image-data/>