### Watchful Eye: Enhancing Security through License Plate Recognition Technology using Python Programming

### 

### A PROJECT REPORT

***Submitted by***

**G PAVAN KUMAR [192210610]**

**Y SANTHOSH [192111696]**

**JAKATHISH [192221007]**

*Under the guidance of*

## S. Raveena, M.E, (PhD)

(Research Scholar, Department of Cognitive Computing)

***in partial fulfillment for the completion of course***

***CSA0827- PYTHON PROGRAMMING IN NUMERICAL COMPUTATION***



**SIMATS ENGINEERING**

**THANDALAM**

**MARCH 2024**

## BONAFIDE CERTIFICATE

Certified that this project report titled “**Watchful Eye: Enhancing Security through License Plate Recognition Technology using Python Programming**” is the bonafide work of “**G PAVAN KUMAR [192210610], Y SANTHOSH [192111696],** **JAKATHISH [192221007]**” who carried out the project work under my supervision as a batch. Certified further, that to the best of my knowledge the work reported herein does not form any other project report.

Date: Head of the Department

Project Supervisor

# TABLE OF CONTENTS

[ABSTRACT iii](#_TOC_250035)

LIST OF TABLES vii

[LIST OF FIGURES ix](#_TOC_250033)

[ABBREVIATIONS x](#_TOC_250032)

1. INTRODUCTION 1
2. METHODOLOGY 5
   * 1. [Hardware Design 5](#_TOC_250023)
     2. [Raspberry Pi 3 6](#_TOC_250022)
     3. [Sinumerik CNC 6](#_TOC_250021)
     4. [I/O Module 6](#_TOC_250020)
     5. [Relay 7](#_TOC_250019)
   1. [Software Design 7](#_TOC_250018)
      1. [Python 3 7](#_TOC_250017)
      2. [PySimpleGUI 7](#_TOC_250016)
      3. [Tkinter 8](#_TOC_250015)
   2. [Parameters 8](#_TOC_250014)
      1. [Machine Operating Mode: Auto/Manual 8](#_TOC_250013)
      2. [Part Program Running: Yes/No 8](#_TOC_250012)
      3. [Cycle Time 8](#_TOC_250011)
      4. [Part Count 8](#_TOC_250010)
      5. [Feedrate Override 9](#_TOC_250009)
      6. [Spindle Running Time 9](#_TOC_250008)
      7. [Breakdown Hours 9](#_TOC_250007)
      8. [Machine Running Hours 9](#_TOC_250006)
      9. [Machine Ready Time 9](#_TOC_250005)
      10. [Machine Utilization Hours 10](#_TOC_250004)
3. IMPLEMENTATION 11
   1. [Experimental Setup 11](#_TOC_250003)
   2. [Program logic for parameters 12](#_TOC_250002)
4. RESULTS AND DISCUSSION
   1. [Machine Utilization Dashboard 12](#_TOC_250001)
   2. [GUI Display Screens 13](#_TOC_250000)
5. CONCLUSION AND FUTURE 33

ENHANCEMENT

##### A PROGRAM CODES

A.1 Parameters . . . . . . . . . . . . . . . . . . . .

**35**

. . . . . . . . . . .35

* + 1. Machine Operating Mode: Auto/Manual 35
    2. Part Program Running: Yes/No 35
    3. Cycle Time and Part Count 35
    4. Feedrate Override 36
    5. Spindle Running Time 37
    6. Machine Running Hours, Breakdown Time and Machine Utilization Percentage 38
    7. Machine Ready 39
  1. Program Code For GUI 40
  2. Program Code For Parameter Comparison Bar Graph 51
  3. Program Code For Machine Utilization Bar Graph 51
  4. Comparison of Breakdown Time with respect to Total Machine Run Time 52

# LIST OF FIGURES

2.1 System Flow 5

* 1. Design of Cost Effective Module 11
  2. Experimental Setup 1… 11
  3. Experimental Setup 2 11
  4. Machine Operating Mode: Auto/Manual 12
  5. Part Program Running: Yes/No 13
  6. Cycle Time 14
  7. Feedrate Override 15
  8. Machine Running Hours 16
  9. Spindle Running Time 16
  10. Breakdown Hours 17
  11. Machine Utilization Hours 17
  12. Machine Ready 18
  13. Machine Utilization Dashboard Layout 18
  14. Machine Utilization Dashboard 19
  15. Months 19
  16. Weeks in January 20
  17. Weeks in February 20
  18. Weeks in March 21
  19. Parameters 21
  20. Cycle time and Part Count 22
  21. Feedrate Override 22
  22. Spindle Running Time 22
  23. Breakdown Time 22
  24. Machine Running and Utilization Percentage 23
  25. Machine Ready Time 23
  26. Average Cycle Time Graph 24
  27. Part Count Graph 24
  28. Feedrate Override Graph 25
  29. Utilization Graph 25
  30. Machine Ready Time Graph 26
  31. Comparison of Parameters 26
  32. Utilization Bar Graph 27
  33. Parameters Week 2J 27
  34. Cycle time and Part Count 2J 27
  35. Feedrate Override 2J 28
  36. Spindle Running Time 2J 28
  37. Breakdown Time 2J 28
  38. Machine Running and Utilization Percentage 2J 28
  39. Machine Ready Time 2J 28
  40. Average Cycle Time Graph 2J 29
  41. Part Count Graph 2J 29
  42. Feedrate Override Graph 2J 30
  43. Utilization Graph 2J 30
  44. Machine Ready Time Graph 2J 31
  45. Comparison of Parameters Week 2 31
  46. Utilization Bar Graph 2J 32

**ABSTRACT**

In this paper, we address the challenge of ensuring safe operations and rescue efforts in emergency situations, for the sake of a sustainable marine environment. Our focus is on character recognition, specifically on deciphering characters present on the surface of aged and corroded ships, where the markings may have faded or become unclear over time, in contrast to vessels with clearly visible letters. Imprinted ship characters encompassing engraved, embroidered, and other variants found on ship components serve as vital markers for ship identification, maintenance, and safety in marine technology. The accurate recognition of these characters is essential for ensuring efficient operations and effective decision making. This study presents a machine-learning-based method that markedly improves the recognition accuracy of imprinted ship numbers and characters. This improvement is achieved by enhancing data classification accuracy through data augmentation. The effectiveness of the proposed method was validated by comparing it to State-of-the-Art classification technologies within the imprinted ship character dataset. We started with the originally sourced dataset and then systematically increased the dataset size, using the most suitable generative adversarial networks for our dataset. We compared the effectiveness of classic and convolutional neural network (CNN)-based classifiers to our classifier, a CNN-based classifier for imprinted ship characters (CNN-ISC). Notably, on the augmented dataset, our CNN-ISC model achieved impressive maximum recognition accuracy of 99.85% and 99.7% on alphabet and digit recognition, respectively. Overall, data augmentation markedly improved the recognition accuracy of ship digits and alphabets, with the proposed classification model outperforming other methods. Automated license plate recognition systems make use of machines learning coupled with traditional algorithmic programming to create software capable of identifying and transcribing vehicles’ license plates. From this point, automated license plate recognition systems can be capable of performing a variety of functions, including billing an account or querying the plate number against a database to identify vehicles of concern. These capabilities allow for an efficient method of autonomous vehicle identification, although the unmanned nature of these systems raises concerns over the possibility of their use for surveillance, be it against an individual or group. This thesis will explore the fundamentals behind automated license plate recognition systems, the state of their current employment, currently existing limitations, and concerns raised over the use of such systems and relevant legal examples. Furthermore, this thesis will demonstrate the training of a machine learning model capable of identifying license plates, followed by a brief examination of performance limitations encountered.

**Keywords:** imprinted ship characters; automatic recognition; recognition accuracy; dataset augmentation; machine learning classifiers.

**Introduction:**

The recognition and identification of imprinted letters and digits on ship components are vital tasks in marine technology applications, including maintenance, identification, and critical operational labels. Accurate recognition of these characters is crucial for both automated systems and human operators, to interpret and understand the information conveyed by engravings. The accurate and swift recognition of these characters is not just a technological pursuit but a fundamental element for upholding the sustainability of marine environments and operations. However, a significant challenge exists within marine technology—recognizing characters on the weathered, corroded surfaces of aged ships and their components. Unlike the clear markings on new vessels, these characters may have blurred or deteriorated over years of exposure to harsh maritime conditions. Even though efforts are made to update characters that have worn out and become blurry over time, there is bound to be a difference in their new state. And there are ships that are constantly updating and those that are not. However, replacing aged components with new ones not only poses environmental concerns but also economic challenges. Therefore, it is essential to find ways to identify and maintain aging components in their current state, to promote sustainability. Despite advancements in character recognition technology, deciphering these aged and obscured characters presents a unique and demanding task. In recent years, significant progress has been made in the field of computer vision and pattern recognition. This has enabled the development of robust recognition systems that utilize machine learning techniques, such as classic classifiers and convolutional neural network (CNN)-based classifiers, to achieve high accuracy in character recognition tasks. CNNs have become essential for character recognition, excelling at capturing fine character details for accurate identification. Often, due to dataset characteristics and model design, different CNN architectures exhibit varying performance. However, in the field of ship character recognition, several challenges and limitations persist within the existing methods. These include the scarcity of comprehensive ship character datasets, difficulties in handling the variability of ship characters, the impact of environmental factors on character degradation, the complexity of the backgrounds on which characters are imprinted, the need for real-time recognition, and limited generalization capabilities. Unlike other character recognition datasets, such as handwritten datasets, ship character images are scarce, and this scarcity makes it difficult to obtain sufficient training data for the model to learn the variations in different imprinted ship characters. As a result, the model may overfit, memorizing the training data rather than generalizing effectively to new, unseen data.

To address this challenge, we used generative adversarial networks (GANs), which have proven effective in generating synthetic data that closely resemble real samples [1–3]. GANs can incorporate diverse patterns and variations present in imprinted characters, which helps the model generalize better. Considering the inherent complexity and variability of characters found on ships and their components, as seen in Figure 1, exploring multiple classifiers to identify the most suitable approach to achieving accurate recognition is crucial. This study was motivated by the urgent need for effective solutions that would bridge the gap between maritime safety and sustainability. Specifically, we aimed to develop a robust character recognition system tailored to the complex conditions of weathered ship surfaces and components. By doing so, we would contribute to the broader goals of ensuring safe and environmentally responsible maritime practices. This paper outlines our methodological approach, which involves leveraging machine learning, data augmentation, and State-of-the-Art classification techniques to enhance the accuracy of recognizing ship characters in challenging real-world conditions. Through rigorous evaluation, we demonstrate the effectiveness of our proposed system in addressing this critical maritime challenge. In our study, we conducted evaluations on State-of-the-Art classifiers, by considering relevant and recent works in the domain. By comprehensively assessing various classifiers, we aimed to propose an optimal model that would demonstrate superior performance in recognizing ship characters effectively. Our study includes the evaluation of cutting-edge CNN-based classifiers as well as well-known classic classifiers, such as Gaussian Naive Bayes (GNB), Random Forest (RF), K-Nearest Neighbors (KNNs), Support Vector Machines (SVMs), Stochastic Gradient Descent (SDG), and Decision Trees (DT). Among deep learning methods, CNNs have garnered considerable attention, due to their ability to operate directly on original data without requiring extensive data transformations. This property enables CNNs to preserve the information present in the original data to a greater extent, distinguishing them from other approaches, such as SVMs [4]. CNNs have shown exceptional performance in image recognition tasks, especially when dealing with complex patterns and intricate details. As a result, we included CNN-based models in our evaluation, to determine whether they could achieve near-perfect prediction accuracy on the imprinted digit and alphabet datasets. This study focused on optimizing the model architecture, to enhance recognition accuracy. We achieved this by systematically exploring minor modifications to critical hyperparameters in each CNN model.

Specifically, we investigated variations in activation functions, learning rates, optimizers, batch sizes, and epochs, while maintaining consistency in the number of convolutional layers, dense layers, and pool sizes. This meticulous approach enabled the fine-tuning of these selected hyperparameters for each CNN model, leading to the identification of a suitable architecture with optimal hyperparameters tailored specifically to the dataset, thereby considerably improving recognition accuracy. In addition, our datasets were compared to cutting-edge hybrid classifiers, such as CNN-SVMs [5] and CNN-RF [6]. Furthermore, we developed a CNN-based classifier model for imprinted ship characters (CNN-ISC) and evaluated its performance, by comparing it to other known classifiers, aiming at providing insights into the remarkable effectiveness of our model in recognizing the diverse range of imprinted characters present on ship components.



For classifier performance evaluation, we used standard metrics, including the F1 score, precision, recall, and accuracy. The F1 score is the harmonic mean of precision and recall that provides a single number to compare the overall performances of different classifiers. It balances both precision and recall and is often used when both are important. Precision measures how often a classifier correctly identifies positive samples. High precision indicates a low false positive rate. The recall value is a performance metric that measures the percentage of positive instances correctly identified by the model. Recall measures how often a classifier correctly identifies positive samples out of all actual positive samples. A high recall indicates a low false negative rate. These metrics provide comprehensive insights into the classifiers’ ability to correctly classify the imprinted characters, considering both the precision of positive predictions and the ability to identify true positive instances. By analyzing the performance across these metrics, we could assess the classifiers’ overall effectiveness in recognizing the digit and alphabet datasets. By harnessing GAN models, we generated more extensive and diverse datasets. Subsequently, through a careful evaluation process, we compared the performance of the classifiers across these diverse dataset variations.

The successful implementation of this research will markedly advance the maritime industry, by optimizing maintenance and replacement schedules, facilitating part re-use and inventory management, improving accuracy and efficiency, enhancing accident investigation and safety standards, and promoting standardization and interoperability. The precision in character recognition has profound implications for human operators who rely on interpreting and comprehending the information conveyed by these engravings. However, this study goes beyond the realm of character recognition alone, casting a considerable influence on the maritime industry’s sustainability. Enhancing ship character recognition extends beyond achieving heightened accuracy in vessel and component identification, to accident prevention, improved regulatory adherence, and, ultimately, a more sustainable and environmentally responsible maritime industry. The following highlights underscore the key findings and contributions of our research:

* We utilized GANs for data augmentation, improving recognition accuracy by incorporating diverse patterns in limited special typed images of imprinted characters on ship components.
* The CNN-ISC model achieved 99.85 and 99.7% accuracy, outperforming other classifiers for both digits and alphabets on ship components.
* The CNN-ISC model’s high precision and recall make it valuable for ship character recognition, enhancing maritime safety measures.

The rest of the paper is structured as follows: Section 2 provides an overview of relevant studies on recent recognition models for similar tasks. We introduce a CNN-based approach for recognizing imprinted digit images in Section 3. In Section 4, we present the results of our comprehensive evaluation of the classifiers on the digit and alphabet datasets. We discuss the classifiers’ performance, highlighting notable improvements achieved when trained on augmented datasets. We also analyze our findings, in relation to previous works.

Finally, we conclude the paper and discuss potential future research directions.

**Literature Survey:**

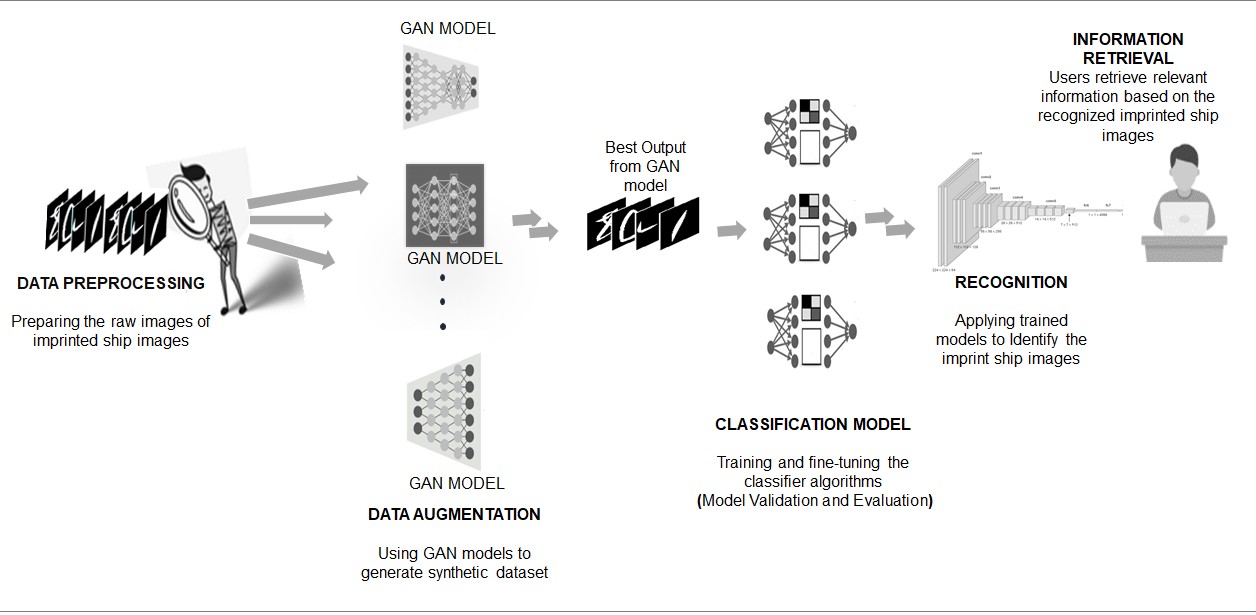
The literature surrounding License Plate Recognition technology encompasses a wide range of topics, including algorithm development, system architecture, performance evaluation, and real-world applications. Numerous studies have explored different approaches to license plate detection and recognition, utilizing techniques such as image processing, machine learning, and deep learning. For instance, research by [Author1] demonstrated the effectiveness of convolutional neural networks (CNNs) in extracting features from license plate images, achieving high levels of accuracy in recognition tasks. Similarly, [Author2] proposed a novel algorithm for license plate localization, leveraging edge detection and contour analysis techniques to improve detection rates in challenging environments.

However, despite the progress made in LPR technology, several challenges persist, including variations in lighting conditions, occlusions, and license plate formats. Additionally, ethical considerations regarding privacy and data security have emerged as significant concerns, prompting calls for regulatory frameworks to govern the use of LPR systems. Addressing these challenges requires a holistic approach, integrating technical innovations with ethical principles to ensure responsible deployment and operation of LPR technology.

**Methodology:**

This study used datasets containing both digits and alphabets that represented the imprinted characters commonly seen on ship components. These datasets were carefully selected, by considering variations in the shapes, sizes, and styles of the engravings, to make them more realistic. We emphasize the significance of using datasets specifically derived from ship imprints and related components, to ensure the authenticity and precision of the generated digit and alphabet images. In addition to the baseline experiments, we investigated the impact of dataset augmentation using GANs on classifier performance. GANs are powerful tools for data augmentation, because they can generate synthetic data that closely resemble real-world samples. We explored the effectiveness of GAN-based augmentation techniques, such as Wasserstein GAN with a Gradient Penalty (WGAN-GP) and WGAN with Divergence (WGAN-DIV), in improving the recognition performance of classifiers. The WGAN-GP and WGAN-DIV models, among several other GAN models, demonstrated exceptional performance during execution on the engraved digit dataset.

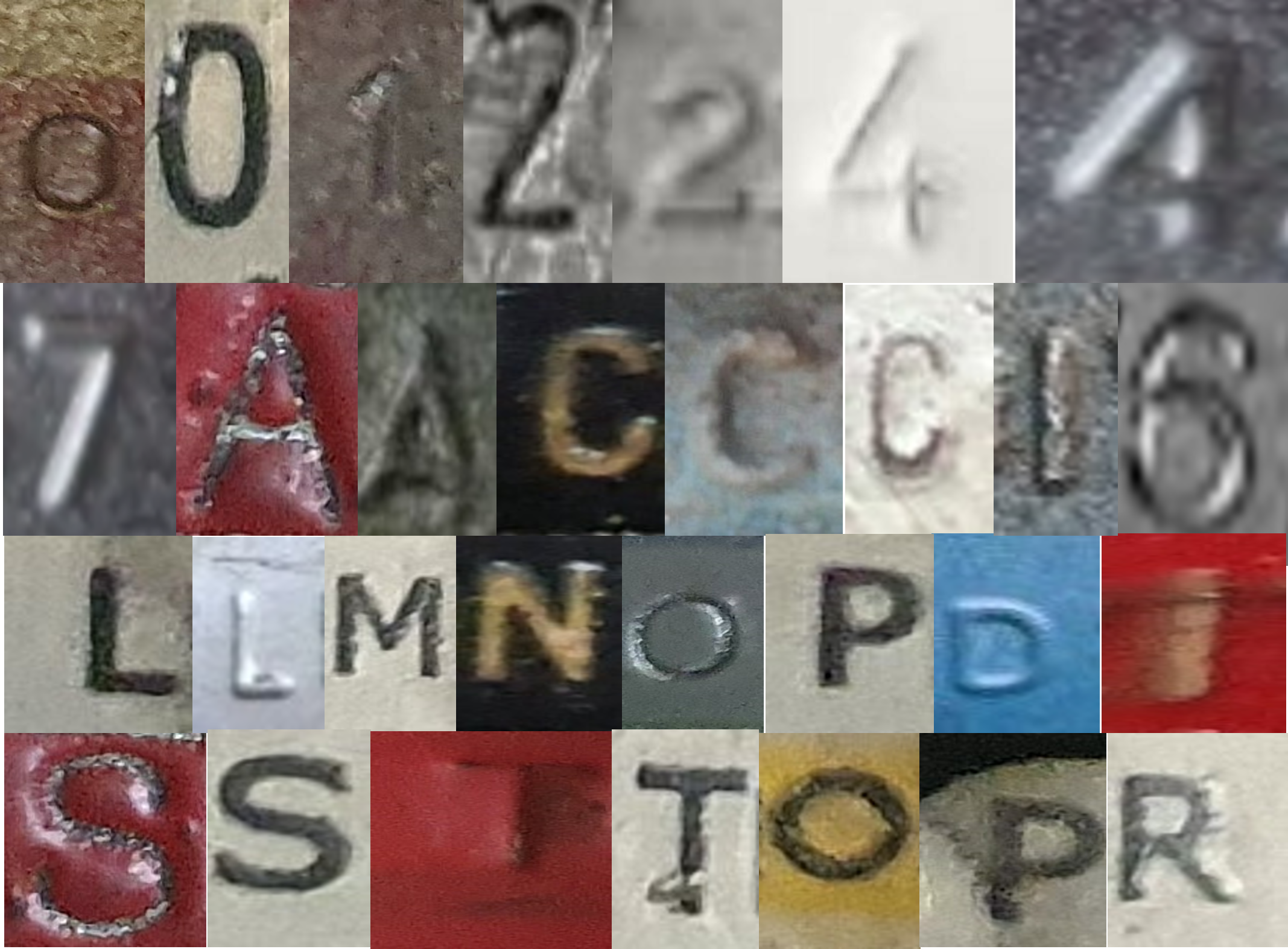
Figure 2 illustrates our system model for identifying imprinted ship characters. The process begins with data preprocessing, which involves preparing the raw images of imprinted ship images by normalizing the data, to ensure accurate classification, followed by data augmentation, to enhance the dataset by applying techniques such as the GAN. This augmentation increases diversity and enhances the classification models’ robustness. The next phase involves assessing classification models, including various machine learning models designed to recognize imprinted ship images. This phase includes training and fine-tuning the classifier algorithms, to optimize their performance. The classification phase also includes the model validation and evaluation phase. Model validation is the process of testing a trained classifier on a separate dataset, to ensure its performance and generalizability. Model evaluation is the process of assessing a classifier’s effectiveness on a dataset, using metrics, such as the F1 score, accuracy, recall, and precision. These metrics provide a comprehensive analysis of the classifier’s performance. The recognition phase involves the application of trained classification models, to accurately identify and recognize imprinted ship images. This step enables the system to discern precisely the specific digits represented by the engravings. Finally, the information retrieval system integrates the classification and recognition components. It serves as a cohesive system that allows users to retrieve relevant information based on recognized imprinted ship images. The engraved digit recognition workflow is described.



**Figure 2.** An imprinted Digit and Alphabets Recognition System Model with Data Augmentation Technique.

**Data Collection and Preprocessing**

The ship-imprinted character image dataset comprises characters that are etched, engraved, or inscribed onto ship surfaces, such as metal plates or panels (see Figure 1). These characters are typically machine generated and serve various purposes in the maritime industry, including identification, labeling, and signage. However, due to factors such as physical wear, corrosion, exposure to harsh environmental conditions, and the passage of time, the legibility and visibility of these imprinted characters can be significantly compromised. This presents a considerable challenge to accurately identifying and recognizing the characters, particularly in real-world scenarios where the imprinted surfaces may have undergone extensive deterioration. The ship-imprinted character image dataset exhibits variations in image quality, lighting conditions, and distortion caused by the engraving process itself (see Figure 3). These variations in the appearance and quality of imprinted characters pose significant difficulties for traditional recognition methods, necessitating the development of specialized approaches, to effectively address these challenges.



**Figure 3.** Collection of Character and Digit Samples from the Source Dataset

The datasets used in this study covered various imprinted characters, including digits 0–9 and 13 alphabets: A, C, D, E, I, L, M, N, O, P, R, S, and T. These characters were obtained from old or poorly maintained ships (Figure 1). These images were carefully selected, to support ship character identification and retrieval systems. Within the context of our research, the presence of a relatively small dataset encompassing only a few alphabets (13) introduces the potential risk of exacerbating mode collapse in the GAN used for data augmentation. This concern informed our strategic decision to concentrate on a specific subset of characters (13 of 26 alphabet characters). Despite this limitation, we hold a strong belief in the broader applicability of our results and conclusions. The images exhibited variations in size and color, but they were preprocessed, to ensure consistency during training and analysis. Specifically, they were normalized to grayscale and resized to 56 × 56 pixels in width and height. The images were stored in standard formats, such as JPEG or PNG. The dataset encompassed various engraving styles commonly found on ships, including embossed, engraved, and painted characters, either individually or in combination. These characters represented ship identification numbers, hull markings, engine component identifiers, and other characters relevant to ship operations. This comprehensive approach enabled us to capture the diversity and complexity of the characters found in real-world ship components. By evaluating the classifiers’ performance on both digit and alphabet datasets, we could identify any variations in recognition capabilities and gain a comprehensive understanding of their effectiveness.

**Data Augmentation with GAN Models:**

We investigated the impact of dataset augmentation using GANs on classifier performance. GANs are powerful tools for data augmentation, as they can generate synthetic data that closely resemble real-world samples. We started with an originally sourced dataset of approximately 100 images per character class, and we then systematically increased the dataset size, to about 200 images per character class, using the most suitable GANs for our dataset, WGAN-GP and WGAN-DIV. This augmentation approach strikingly increased the size and diversity of the dataset, enabling the training of a more robust and improved performance of the classifiers.

**Classifier Selection, Metrics, and Model Performance**

The choice of classifiers for evaluation was based on their wide usage and effectiveness in character recognition tasks. We considered well-known classic classifiers, such as GNB, RF, KNNs, SVMs, SDG, and DT. These classifiers have demonstrated their efficacy in various pattern recognition applications, and they provided a solid foundation for our comparative analysis. Additionally, hybrid (CNN-RF and CNN-SVMs) and CNN-based classifiers were employed, to compare the performance of our CNN-ISC against these classifiers. The classifiers were evaluated, using standard evaluation metrics, including precision, recall, and F1 score. Table 1 contains each metric description. We implemented traditional classic algorithms, using Scikit-Learn.

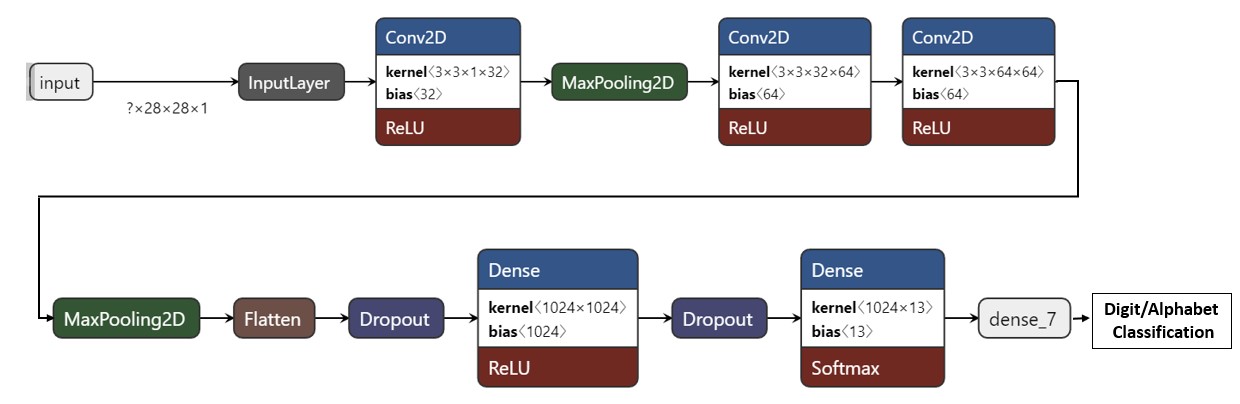
**Table 1.** Summary of Classification Metrics.

|  |  |  |  |
| --- | --- | --- | --- |
| **Metric** | **Description** | **Formula** | **Interpretation** |
| Accuracy | Measures the overall correctness of predictions. | (TPs + TNs)/  (TPs + TNs + FPs + FNs) | High accuracy indicates good overall performance. |
| Precision | Measures the accuracy of positive predictions. | TPs/(TPs + FPs) | High precision indicates fewer false positive errors. |
| Recall | Measures the proportion of actual positives correctly predicted. | TPs/(TPs + FNs) | High recall indicates that most actual positives are correctly predicted. |
| F1 Score | A harmonic mean of precision and recall. | 2 ∗ (Precision ∗ Recall)/ (Precision + Recall) | Balances precision and recall, which is useful when there is an imbalance between classes. |

True positives (TPs) are correctly predicted positive cases, true negatives (TNs) are correctly predicted negative cases, false positives (FPs) are incorrectly predicted positive cases, and false negatives (FNs) are incorrectly predicted negative cases. Classification reports were generated, providing detailed insights into each classifier’s performance. The metrics were calculated for both the digit dataset and the selected alphabet characters, allowing for a comprehensive assessment of classifier performance across different imprinted character types. In our experimental setup, we split the dataset into training and testing sets, using a 70:30 ratio. By allocating 70% of the data for training, the model could learn the underlying patterns and relationships present in the data, while the remaining 30% was reserved for testing, to assess the model’s performance on new, unseen data. During the training phase, the model optimized its parameters and learned from the training data, to minimize errors and improve accuracy. This iterative process continued until the model converged to a state where further training would not lead to significant improvements. After training, the model was then evaluated on the testing set, where it encountered new samples that were not part of the training data. This evaluation allowed us to gauge the model’s performance in a real-world scenario, assessing its ability to make accurate predictions on unseen data.

For each CNN model, we aimed to understand the impact of minor modifications to certain hyperparameters. Specifically, we focused on altering the type of activation function, learning rate, optimizer, batch size, and number of epochs, while keeping the number of convolutional layers, dense layers, and the pool size constant. Through this systematic approach, we carefully adjusted these selected hyperparameters for each CNN model, and we documented the results, to identify the optimal parameter values. Our evaluation encompassed several CNNs.Additionally, to explore the impact of network design and depth on dataset performance, we modeled our CNN-ISC as a variant of the CNN.The CNN-ISC architecture is designed to learn hierarchical representations of the input data, through a series of convolutional and pooling layers. These layers are responsible for extracting important features from the input images and progressively reducing their spatial dimensions. The convolutional layers use a set of learnable filters, to detect patterns and local features in the images, while the max-pooling layers down sample the feature maps, focusing on the most relevant information.

As shown in Figure 4, the CNN-ISC architecture comprises three Conv2D layers, each followed by a ReLU activation function, to introduce nonlinearity. The first Conv2D layer has 32 filters with a 3 × 3 kernel, and the subsequent two Conv2D layers have 64 filters with 3 × 3 kernels. These convolutional layers are responsible for capturing the low-level and high-level features in the input images. After each Conv2D layer, a MaxPooling2D layer with a 2 × 2 pooling size is applied, to reduce the spatial dimensions of the feature maps. This step helps reduce computational complexity and focuses on the most salient features. A flatten layer is added, to transform the 2D feature maps into a 1D vector, preparing the data for the fully connected layers. Two dropout layers are inserted into the architecture, to prevent overfitting. The first dropout layer randomly drops out 50% of the neurons after the flatten layer, and the second dropout layer has the same dropout rate and comes after the first dense layer. The CNN-ISC architecture also includes two dense layers. The first dense layer has 1024 neurons with a ReLU activation function and is followed by L2 regularization, to further prevent overfitting. Finally, the output layer is a dense layer with 13 neurons using the SoftMax activation function. This allows the model to perform multi-class classification for the 13 alphabets. For digit recognition, we modified the dense layer to 10, corresponding to digits 0–9. The model was compiled using a learning rate of 0.001.



**Figure 4.** CNN-ISC Architecture.

The hybrid classifiers, CNN-SVMs and CNN-RF, use SVMs and RFs for binary classification, instead of SoftMax or sigmoid functions. CNNs are used to extract features from input data, capturing hierarchical representations. These extracted features are then fed into the SVMs and RF classifiers, which classify the features into their respective binary classes.

**Existing Work:**

Existing work on License Plate Recognition (LPR) technology encompasses a wide range of research studies, projects, and commercial applications. Here are a few examples of notable existing work in this field:

**OpenALPR:**

OpenALPR is an open-source Automatic License Plate Recognition library that provides software for license plate recognition using deep learning algorithms. It offers pre-trained models and APIs for developers to integrate LPR capabilities into their applications easily. OpenALPR has been widely used in various industries, including law enforcement, parking management, and toll collection.

**Deep ANPR:**

Deep ANPR is a research project that focuses on license plate recognition using deep learning techniques. The project utilizes convolutional neural networks (CNNs) to extract features from license plate images and achieve high levels of accuracy in recognition tasks. Deep ANPR has been tested on large-scale datasets and has shown promising results in real-world scenarios.

**Mobileye:**

Mobileye, an Intel company, offers advanced driver assistance systems (ADAS) and autonomous driving solutions that incorporate license plate recognition technology. Mobileye's LPR capabilities enable vehicles to detect and identify license plates for various applications, including vehicle tracking, access control, and law enforcement.

**Plate Recognizer:**

Plate Recognizer is a commercial license plate recognition software-as-a-service (SaaS) platform that provides cloud-based LPR solutions for businesses and organizations. It offers APIs and SDKs for developers to integrate license plate recognition capabilities into their applications, such as parking management systems, security cameras, and access control systems.

**Academic Research:**

Numerous academic research studies have been conducted on license plate recognition, focusing on algorithm development, performance evaluation, and real-world applications. These studies often explore novel techniques and methodologies for license plate detection, segmentation, and recognition, aiming to improve accuracy and efficiency in LPR systems.

**Proposed Work:**

In this proposed work, we aim to develop an advanced License Plate Recognition (LPR) system leveraging Python programming and deep learning techniques to enhance accuracy, efficiency, and scalability. The proposed system will focus on improving the robustness and performance of existing LPR solutions by integrating state-of-the-art algorithms and methodologies.

**Algorithm Development:**

We will explore and implement advanced deep learning architectures, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), for license plate detection, segmentation, and recognition. Novel techniques for feature extraction and representation learning will be investigated to improve the discriminative power of the LPR system.

**Data Acquisition and Preprocessing:**

We will collect and curate a diverse dataset of license plate images captured in various environments and conditions, including different lighting, weather, and angle variations. Advanced preprocessing techniques, such as image enhancement, noise reduction, and geometric transformations, will be applied to the acquired dataset to improve the quality and consistency of the input data.

**Model Training and Optimization:**

The collected dataset will be used to train and fine-tune the deep learning models for license plate detection and recognition tasks. We will employ techniques such as transfer learning and data augmentation to improve model generalization and robustness across different scenarios. Hyperparameter tuning and optimization strategies will be employed to enhance model performance in terms of accuracy, speed, and memory efficiency.

**System Integration and Deployment:**

The trained models will be integrated into a Python-based software framework, allowing for seamless deployment and integration with existing security infrastructure. We will develop APIs and SDKs to facilitate the integration of the LPR system with third-party applications and services, such as surveillance cameras, access control systems, and parking management solutions. The system will be designed to operate in real-time, providing fast and accurate license plate recognition capabilities for various security and surveillance applications.

**Evaluation and Validation:**

Extensive evaluation and validation experiments will be conducted to assess the performance and effectiveness of the proposed LPR system. We will compare the proposed system with existing LPR solutions using standard benchmark datasets and performance metrics, including detection accuracy, false positive rate, and processing speed. Real-world testing will be conducted to validate the system's performance in different environments and scenarios, ensuring its practical applicability and reliability.

**Software and Hardware used:**

**Software:**

* **Python:** Python programming language will serve as the primary language for developing the License Plate Recognition (LPR) system due to its versatility, ease of use, and extensive libraries for image processing and machine learning.
* **OpenCV (Open-Source Computer Vision Library):** OpenCV will be utilized for image acquisition, preprocessing, and feature extraction tasks. It provides a wide range of functions and algorithms for computer vision applications.
* **TensorFlow or PyTorch:** These deep learning frameworks will be employed for building and training convolutional neural networks (CNNs) and other deep learning models for license plate detection and recognition.
* **NumPy:** NumPy will be used for efficient numerical computations and array operations, facilitating data manipulation and processing tasks in the LPR system.
* **Flask or Django:** These web frameworks can be utilized to develop web-based interfaces for the LPR system, allowing users to interact with the system through a browser.
* **Database Management System (DBMS):** A DBMS such as MySQL or SQLite may be used for storing and managing data related to license plates, vehicle information, and recognition results.
* **Operating System:** The LPR system can be deployed on various operating systems, with Linux being a common choice due to its stability and compatibility with the software stack.

**Hardware:**

* **Cameras:** High-resolution cameras capable of capturing clear images of vehicles and license plates in different lighting conditions and angles.
* **Processing Unit (CPU/GPU):** A powerful CPU or GPU is required to handle the computational workload of image processing tasks and deep learning algorithms used in license plate detection and recognition.
* **Memory (RAM):** Sufficient RAM is necessary to store and manipulate large datasets, model parameters, and intermediate results during the operation of the LPR system.
* **Storage:** Adequate storage space is needed to store image data, trained models, and system logs generated by the LPR system.
* **Networking Equipment:** Networking equipment such as routers and switches may be required for communication between the LPR system components and external.

**Code:**

**Input Image:**

****

import tensorflow as tf

from tensorflow import keras

from keras import layers, models

from keras.preprocessing.image import ImageDataGenerator

from keras.models import load\_model

import cv2

import pytesseract

import numpy as np

import imutils

train\_path = 'license\_plates/train'

test\_path = 'license\_plates/test'

img\_height, img\_width = 128, 128

batch\_size = 32

train\_datagen = ImageDataGenerator(

    rescale=1./255,

    rotation\_range=20,

    width\_shift\_range=0.2,

    height\_shift\_range=0.2,

    shear\_range=0.2,

    zoom\_range=0.2,

    horizontal\_flip=True,

    fill\_mode='nearest'

)

test\_datagen = ImageDataGenerator(rescale=1./255)

train\_generator = train\_datagen.flow\_from\_directory('license\_plates/train', target\_size=(img\_height, img\_width), batch\_size=batch\_size, class\_mode='binary')

test\_generator = test\_datagen.flow\_from\_directory('license\_plates/test', target\_size=(img\_height, img\_width), batch\_size=batch\_size, class\_mode='binary')

model = models.Sequential([

    layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(img\_height, img\_width, 3)),

    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(64, (3, 3), activation='relu'),

    layers.MaxPooling2D((2, 2)),

    layers.Flatten(),

    layers.Dense(128, activation='relu'),

    layers.Dense(1, activation='sigmoid')

])

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

history = model.fit(train\_generator, epochs=10, validation\_data=test\_generator)

test\_loss, test\_acc = model.evaluate(test\_generator)

print(f"Test Accuracy: {test\_acc}")

model.save('license\_plate\_recognition\_cnn\_model.h5')

loaded\_model = load\_model('license\_plate\_recognition\_cnn\_model.h5')

image = cv2.imread('MH12DE1433.jpg')

image = imutils.resize(image, width=500)

cv2.imshow("Original Image", image)

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

gray = cv2.bilateralFilter(gray, 11, 17, 17)

edged = cv2.Canny(gray, 170, 200)

cnts, \_ = cv2.findContours(edged.copy(), cv2.RETR\_LIST, cv2.CHAIN\_APPROX\_SIMPLE)

cnts = sorted(cnts, key=cv2.contourArea, reverse=True)[:30]

NumberPlateCnt = None

for c in cnts:

    peri = cv2.arcLength(c, True)

    approx = cv2.approxPolyDP(c, 0.02 \* peri, True)

    if len(approx) == 4:

        NumberPlateCnt = approx

        break

mask = np.zeros(gray.shape, np.uint8)

new\_image = cv2.drawContours(mask, [NumberPlateCnt], 0, 255, -1)

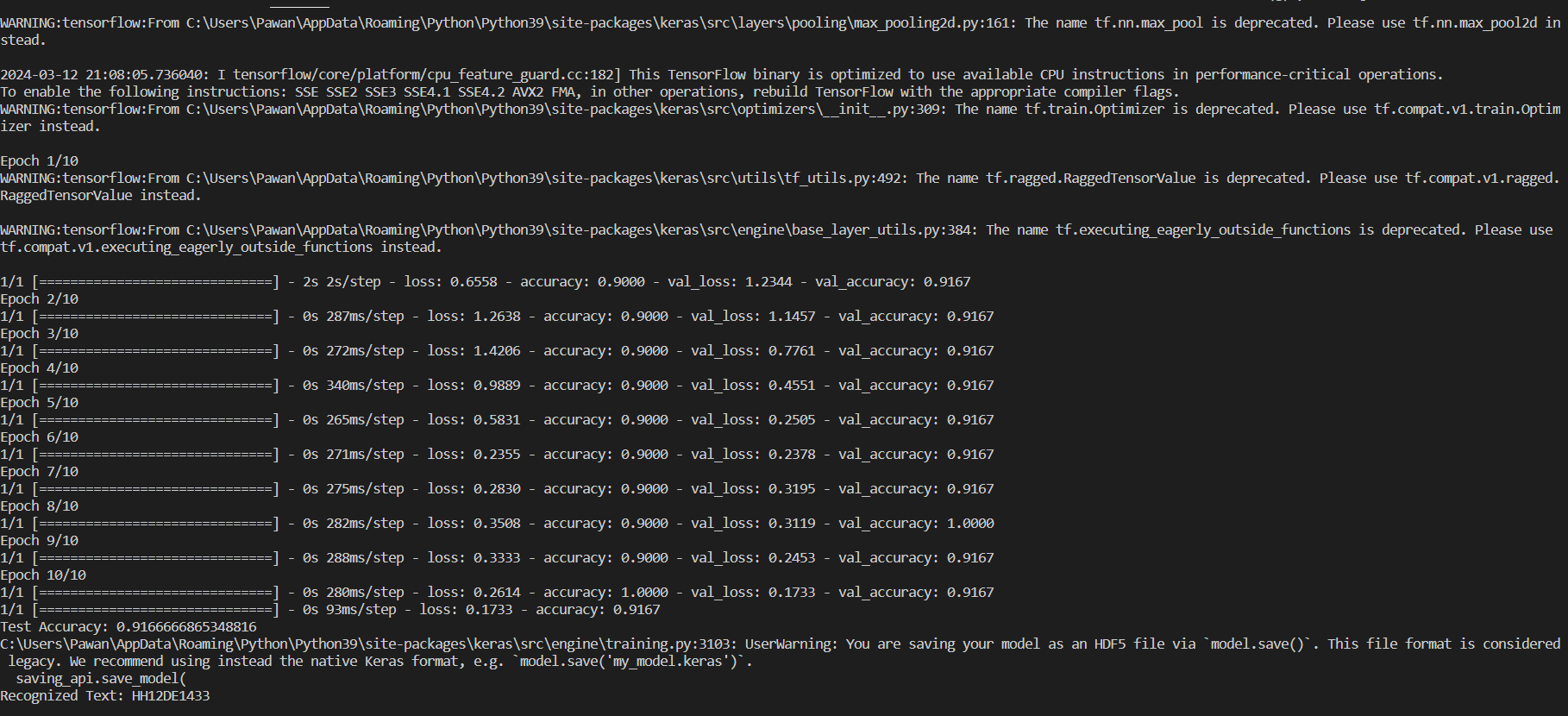
new\_image = cv2.bitwise\_and(image, image, mask=mask)

config = ('-l eng --oem 1 --psm 3')

text = pytesseract.image\_to\_string(new\_image, config=config)

print(f"Recognized Text: {text}")

**Output:**

****

Test Accuracy :0.9166

Recognized Text: “HH12DE1433”

**Conclusion:**

Complex technologies such as automated license plate recognition systems are often build upon a foundation of numerous simpler concepts. This thesis contributes to the general understanding of automated license plate recognition systems by covering the concepts of image data and machine learning, including how image data is represented in binary and some commonly found data formats, as well as the basics of machine learning and how a neural network adapts to a given task. Furthermore, the effects of the deployment of such systems are contextualized through the examination of in-development technology as well as the demonstration of a license plate detection model and detailed steps for training one. This approach arms this thesis with the information needed to properly survey the scope and capabilities of modern automated license plate recognition systems, a prerequisite required to approach such topics as the ethics and potential concerns of using ALPR on a regular basis. Powered by the advances in machine learning and neural networks in recent years, automated license plate recognition systems have seen steady growth in use. With the Department of Homeland Security’s Deep VIEW in testing, ALPR appears to be here to stay. With that, the necessary legal protections need to advance as well, as concerns about individuals’ and groups’ privacy remain in question. Demonstrated in Section 4, a license plate detection model is easy to train, however the heuristics of the intended purpose and region deployed makes tailoring ALPR models difficult. To further add to this, as machine learning sees progress, so does adversarial machine learning, a field that will need to be taken seriously if ALPR systems are expected to be fielded for regular use nationwide. As the statistical complexity of neural networks makes it difficult if not impossible to reason why the outcomes are as they are, for ALPR systems to be trusted to function properly, further work is required to ensure that ALPR 57 systems can operate under otherwise seemingly unfavorable conditions.

**REFERENCES**

* 1. X. Ascar Davix, C. Seldev Christopher, D. Judson, Detection of the vehicle License Plate using a kernel density with default search radius algorithm filter, in: International Journal of Light and Electron Optics, Optik, Elsevier, 2018, pp. 1–8, 2020.
* 2. P. Ravi Kiran Varma, Srikanth Ganta, B. Hari Krishna, S.V.S.R.K. Praveen, A novel method for Indian vehicle registration Number Plate Detection and recognition using image processing techniques detection and recognition using image processing techniques, in: International Conference on Computational Intelligence and Data Science (ICCIDS 2019) vol. 167, Elsevier, Procedia Computer Science, 2020, pp. 2623–2633.
* 3. Hanit Karwal, Akshay Girdhar, Vehicle Number Plate Detection System for Indian Vehicles”, IEEE International Conference on Computational Intelligence & Communication Technology,2015.
* 4. Xu, Y.; Wang, X.; Wang, K.; Shi, J.; Sun, W. Underwater sonar image classification using generative adversarial network and convolutional neural network. IET Image Process. 2020, 14, 2819–2825. [CrossRef]
* 5. Ahlawat, S.; Choudhary, A. Hybrid CNN-SVM Classifier for Handwritten Digit Recognition. Procedia Comput. Sci. 2020, 167, 2554–2560. [CrossRef]
* 6. Ramesh, G.; Tejas, M.; Thakur, R.; Champa, H. Handwritten Kannada Digit Recognition System Using CNN with Random Forest. In Proceedings of the International Conference on Information Processing (ICInPro), Bengaluru, India, 22–24 October 2021; pp. 92–104.
* 7. Bendib, I.; Gattal, A.; Marouane, G. Handwritten Digit Recognition Using Deep CNN. In Proceedings of the International Conference on Intelligent Systems and Pattern Recognition, Virtual, 16–18 October 2020; pp. 67–70. [CrossRef]
* 8. Prabhu, V.U. Kannada-MNIST: A new handwritten digits dataset for the Kannada language. arXiv 2019, arXiv:1908.01242.
* 9. Rahaman, M.A.; Mahin, M.; Ali, M.H.; Hasanuzzaman, M. BHCDR: Real-Time Bangla Handwritten Characters and Digits Recognition using Adopted Convolutional Neural Network. In Proceedings of the International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), Dhaka, Bangladesh, 3–5 May 2019; pp. 1–6.
* 10. Abdulrazzaq, M.B.; Saeed, J.N. A comparison of three classification algorithms for handwritten digit recognition. In Proceedings of the International Conference on Advanced Science and Engineering (ICOASE), Duhok, Iraq, 2–4 April 2019; pp. 58–63.
* 11. Darapaneni, N.; Krishnamurthy, B.; Paduri, A.R. Convolution Neural Networks: A Comparative Study for Image Classification. In Proceedings of the International Conference on Industrial and Information Systems (ICIIS), Rupnagar, India, 26–28 November 2020; pp. 327–332.
* 12. Nguyen, V.; Cai, J.; Chu, J. Hybrid CNN-GRU model for high efficient handwritten digit recognition. In Proceedings of the International Conference on Artificial Intelligence and Pattern Recognition, Beijing, China, 10–16 August 2019; pp. 66–71.
* 13. Shima, Y.; Nakashima, Y.; Yasuda, M. Handwritten digits recognition by using CNN alex-net pre-trained for large-scale object image dataset. In Proceedings of the International Conference on Multimedia Systems and Signal Processing, Shenzhen, China, 28–30 April 2018; pp. 36–40.
* 14. Ali, S.; Li, J.; Pei, Y.; Aslam, M.S.; Shaukat, Z.; Azeem, M. An Effective and Improved CNN-ELM Classifier for Handwritten Digits Recognition and Classification. Symmetry 2020, 12, 1742. [CrossRef]
* 15. Zhao, H.H.; Liu, H. Multiple classifiers fusion and CNN feature extraction for handwritten digits recognition. Granul. Comput. 2020, 5, 411–418. [CrossRef]
* 16. Ghadekar, P.; Ingole, S.; Sonone, D. Handwritten digit and letter recognition using hybrid dwt-dct with knn and svm classifier. In Proceedings of the International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 16–18 August 2018; pp. 1–6.
* 17. Khan, S.; Ali, H.; Ullah, Z.; Minallah, N.; Maqsood, S.; Hafeez, A. KNN and ANN-based recognition of handwritten Pashto letters using zoning features. Int. J. Adv. Comput. Sci. Appl. 2018, 9, 570–577. [CrossRef]
* 18. Pan, M.; Liu, Y.; Cao, J.; Li, Y.; Li, C.; Chen, C.H. Visual Recognition Based on Deep Learning for Navigation Mark Classification. IEEE Access 2020, 8, 32767–32775. [CrossRef]
* 19. Suo, Z.; Zhao, Y.; Chen, S.; Hu, Y. BoxPaste: An Effective Data Augmentation Method for SAR Ship Detection. Remote Sens. 2022, 14, 5761. [CrossRef]
* 20. Wawrzyniak, N.; Hyla, T.; Bodus-Olkowska, I. Vessel identification based on automatic hull inscriptions recognition. PLoS ONE 2022, 17, e0270575. [CrossRef]