# An Approach for Automatic Recognition of High-Security Registered Plates (HSRP)

Neel Patel
*Computer Engineering*
*Pandit Deendayal Energy University*
Gandhinagar, Gujarat, India
neel.hce21@sot.pdpu.ac.in

Aditya Mahajan
*Computer Engineering*
*Pandit Deendayal Energy University*
Gandhinagar, Gujarat, India
aditya.mce21@sot.pdpu.ac.in

Bhumil Dobaria
*Computer Engineering*
*Pandit Deendayal Energy University*
Gandhinagar, Gujarat, India
bhumil.dce21@sot.pdpu.ac.in

Harit Pansuriya
*Computer Engineering*
*Pandit Deendayal Energy University*
Gandhinagar, Gujarat, India
harit.pce21@sot.pdpu.ac.in

Dev Parikh
*Computer Engineering*
*Pandit Deendayal Energy University*
Gandhinagar, Gujarat, India
dev.pce21@sot.pdpu.ac.in

***Abstract*— Automated license plate recognition is a critical component of modern traffic monitoring, parking management, and law enforcement. In this project, we present an integrated approach to accurately predict whether a license plate is a High-Security Registration Plate (HSRP) or not. Our system combines web scraping techniques, object detection using YOLOv3, and Convolutional Neural Networks (CNN) to achieve this objective. To build our dataset, we combined the data from OLX which we have web-scraped and popular online datasets available from Kaggle and GitHub. Object detection was performed using the YOLOv3 model, enabling us to detect and localize license plates within images. The model effectively creates bounding boxes around license plates. This step was crucial in segmenting license plates from the vehicle images. Following object detection, a CNN model was trained on the extracted license plate images to classify them as either HSRP (output label: 1) or non-HSRP (output label: 0). The CNN model demonstrated impressive performance in distinguishing between the two categories. The integrated approach demonstrates the feasibility of employing deep learning techniques in combination with web scraping for license plate recognition. Our model's ability to accurately distinguish HSRPs from non-HSRPs carries significant implications for law enforcement, traffic monitoring, and parking management systems. By presenting this project in a review paper, we hope to contribute to the growing field of automated license plate recognition, opening new avenues for improving the efficiency and accuracy of such systems.**

***Keywords—HSRP , CNN , YOLOv3***

## I. Introduction

In recent years, the fusion of cutting-edge technologies, such as artificial intelligence, computer vision, and deep learning, has revolutionized the field of license plate recognition. These advancements have not only improved the accuracy and efficiency of plate identification but have also opened up new avenues for research and innovation. The integration of High-Security Registration Plates (HSRPs) into these technological developments is a testament to the ever-evolving landscape of modern transportation and security systems. With the global adoption of HSRPs, the need for robust and automated recognition systems has never been more pressing.

Vehicle plate identification and recognition have become increasingly essential in modern transportation, security, and law enforcement systems, offering a wide range of applications, including travel time calculation, highway car counting, traffic violation detection, and surveillance. Among these applications, the recognition of High-Security Registration Plates (HSRPs) has gained particular significance due to their prevalence in numerous countries and their vital role in ensuring road safety and regulatory compliance.

High-Security Registration Plates (HSRPs) represent a notable advancement in license plate technology, incorporating features that enhance security and authenticity. The ability to automatically identify and classify these advanced plates provides a powerful tool for traffic management, parking enforcement, and the pursuit of criminal activities. With recent advancements in computer vision, statistical methods have given way to deep learning neural networks, which offer higher accuracy in object detection. However, the success of deep learning methods relies significantly on the availability of extensive and diverse training datasets.

In this review paper, we present an innovative and integrated approach to the automatic recognition of HSRPs in datasets sourced from Google Images. This project leverages a combination of web scraping, object detection using YOLOv3, and Convolutional Neural Networks (CNN) to achieve the goal of accurately distinguishing HSRPs from non-HSRPs. To effectively extract license plates from the collected images, we employed the YOLOv3 object detection model. YOLOv3 demonstrated the ability to localize and identify license plates. The core of our project revolves around a CNN model that was trained on the extracted license plate images to classify them as either HSRPs or non-HSRPs. The ability to distinguish between HSRPs and non-HSRPs has significant implications for traffic management, law enforcement, and parking management systems, as it streamlines the identification of legitimate and compliant vehicles. Our study not only showcases the effectiveness of this integrated approach but also underscores the potential for further advancements in automated license plate recognition.

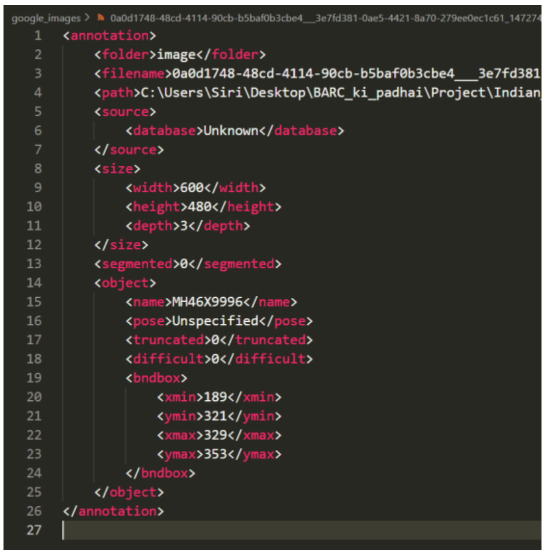
### *A. OLX Dataset.*

Due to the lack of availability of public dataset for Indian vehicle images, we generated the images by web scraping from online sites like OLX and by taking vehicle images from highways at different daylight conditions. The use of web scraping techniques allowed us to collect a diverse and comprehensive dataset of license plates, sourced from OLX, a popular online vehicle marketplace.

Additionally, we augmented our dataset with images sourced from Google Images, which provides a wider variety of license plate images across different contexts. This data aggregation provides a rich and realistic representation of license plates used in a variety of contexts, which is critical for training a robust recognition system. We have also added some images which is publicly available although images were divided into section of states.

## B. *Google Images Dataset.*

The Google Images dataset, commonly available on platforms like Kaggle, is a valuable resource for computer vision and machine learning researchers. This dataset is a vast collection of license plate images crawled from the internet using Google's search engine [2]. These datasets typically consist of images that capture license plates on various types of vehicles, taken under different lighting and environmental conditions. They offer researchers and developers the opportunity to train and evaluate algorithms for automating tasks related to license plate data.

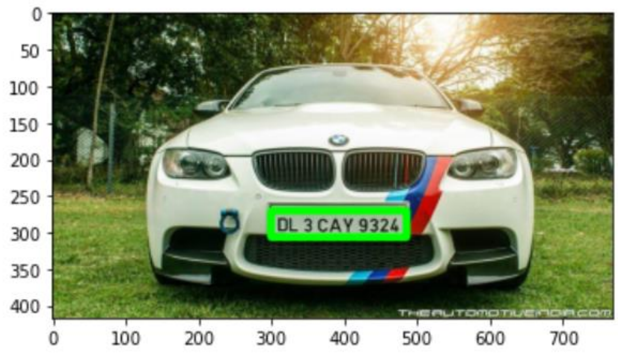


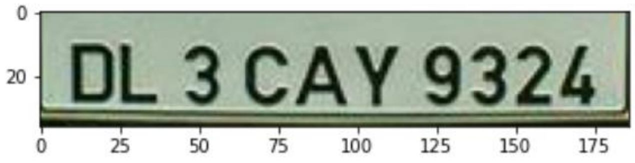
```
google_images > 0a0d1748-48cd-4114-90cb-b5baf0b3cbe4__3e7fd381-0ae5-4421-8a70-279ee0ec1c61_147274
<annotation>
    <folder>image</folder>
    <filename>0a0d1748-48cd-4114-90cb-b5baf0b3cbe4___3e7fd381
    <path>C:\Users\Siri\Desktop\BARC_ki_padhai\Project\Indian
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>600</width>
        <height>480</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>MH46X9996</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>189</xmin>
            <ymin>321</ymin>
            <xmax>329</xmax>
            <ymax>353</ymax>
        </bndbox>
    </object>
</annotation>
```

## C. *YOLOv3 Model*

In this project, the YOLOv3 (You Only Look Once, Version 3) model plays a central role in our High-Security Number Plates recognition system. YOLOv3 is a state-of-the-art real-time object detection algorithm that has gained widespread recognition for its speed and accuracy. It is a deep learning-based model designed to simultaneously detect and localize objects in an image, making it particularly well-suited for our task of license plate detection within complex visual scenes [5]. The use of YOLOv3 represents a critical component of our methodology, as it allows us to efficiently identify and localize license plates within a given image. By creating bounding boxes around the detected license plates, YOLOv3 enables precise extraction of the license plate regions, facilitating subsequent processing and classification.

- The detection process involves passing an image through the YOLOv3 model, which results in a list of bounding boxes, class IDs, and confidence scores for detected objects. These bounding boxes are then filtered using the **nms** function to remove overlapping or redundant boxes, and the selected bounding boxes are drawn on the image.
- Even though the YOLOv3 model might not have been trained specifically for license plate detection, it is a general-purpose object detection model. It can detect a wide range of objects based on the patterns it has learned during its training on a diverse dataset. This means that it can detect license plates as long as they exhibit patterns similar to the objects in the training data.
- To improve the accuracy of license plate detection, fine-tuning the model on a dataset of license plate images is recommended [5].. This allows the model to learn more specific features related to license plates. However, for many common object detection tasks, a pre-trained YOLOv3 model can still provide good results out of the box.





As we can see it creates best Bounding Box and gives us best result. In this introduction, we provide an overview of the pivotal role that YOLOv3 plays in our project.

## II. IMAGE PREPROCESSING

In the image preprocessing stage of our project, we take images from our combined dataset, which includes images from OLX and Google Images, and prepare them for further analysis. The primary objectives of this preprocessing phase are to create accurate bounding boxes around license plates, eliminate redundant information, and assign ground truth labels to facilitate the subsequent training of our Convolutional Neural Network (CNN) model.

### A. *Object Detection Through YOLOv3*

Object detection is one of the branches of computer vision and is widely in use in the industry. For example, Facebook uses it to detect faces in images uploaded, our phones use the object detection to enable the "face unlock" systems. [5]

Object detection involves the following two tasks –

- Locating the object in the image (Region Proposal)
- Classifying the object in the image (Classification)

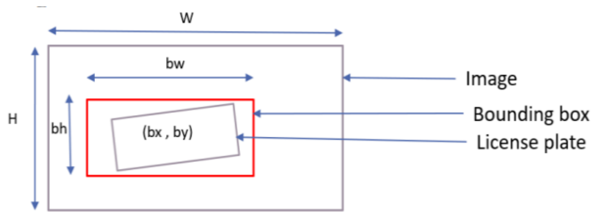
Region proposal is the process of generating potential bounding boxes that may contain objects of interest in the image. Classification is the process of determining whether each of the proposed regions contains an object or not.

We employ the YOLOv3 model to detect license plates within each image. YOLOv3 excels in its ability to localize and identify objects with remarkable speed and precision. The model processes the input images and creates bounding boxes around the detected license plates. These bounding boxes are expressed as (x, y, w, h), where (x, y) represents the center coordinates of the bounding box, and (w, h) are its width and height.[5]

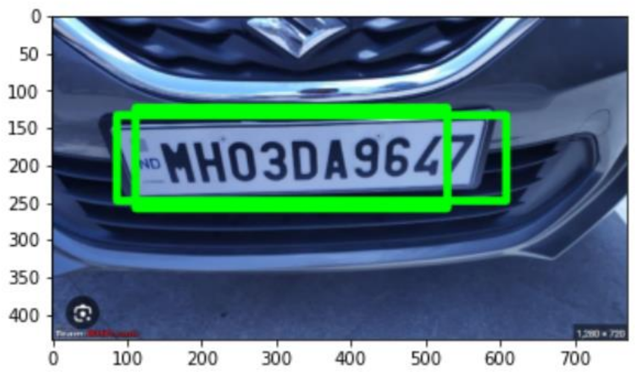
### B. *Non - Maximum Suppression (NMS)*

Non-maximum suppression is a technique that is used after the region proposal step to eliminate duplicate bounding boxes and select the most relevant ones. To refine the bounding boxes and reduce redundancy, we apply Non-Maximum Suppression (NMS). NMS is a crucial step that eliminates overlapping bounding boxes, ensuring that we retain only the most accurate and distinct license plate detections. It is executed based on a defined threshold, which is typically the Intersection over Union (IOU) score. The idea behind NMS is straightforward. It works by comparing the confidence scores of the proposed bounding boxes and eliminating the ones that overlap significantly with a higher-scoring bounding box [5].

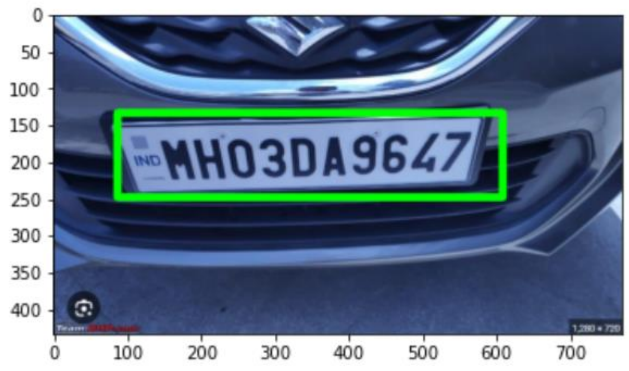
To select the best bounding box, from the multiple predicted bounding boxes, these object detection algorithms use non-max suppression. This technique is used to "suppress" the less likely bounding boxes and keep only the best one. Here we have only one object in image that is license plate and we have to find only that perfect bounding box which captures that license plate correctly.



The objects in the image can be of different sizes and shapes, and to capture each of these perfectly, the object detection algorithms create multiple bounding boxes like :



Ideally for only one object that is license plate there should be only one bounding Box



The NMS takes two things into account :

1. The objectiveness score
2. The IOU of the bounding boxes

The following is the process of selecting the best bounding box using NMS-

1. Select the box with highest objectiveness score.
2. Then, compare the overlap (intersection over union) of this box with other boxes.
3. Remove the bounding boxes with overlap (intersection over union) >50%.
4. Then, move to the next highest objectiveness score.
5. Finally, repeat steps 2-4.

### C. *Intersection – Over Union (IoU)*

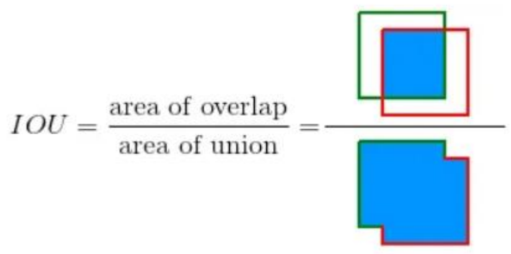
IOU is mainly used in applications related to object detection, where we train a model to output a box that fits perfectly around an object. IOU is also used in non max suppression, which is used to eliminate multiple boxes that surround the same object, based on which box has a higher confidence [5].

Intersection over Union (IoU), also known as Jaccard's index, is a mathematical expression that calculates the "Area of Intersection" of two boxes over the "Area of Union" of the same two boxes.

- ❖ An IoU of 1 indicates that your predicted bounding box perfectly matches the ground truth box.
- ❖ An IoU of 0 indicates that no part of your predicted bounding box overlaps the ground truth box.

**Calculating IoU :**

IOU = Area of Intersection / Area of Union.
Where the "Area of Intersection" is the region where two bounding boxes overlap, and the "Area of Union" is the combined area of both bounding boxes.



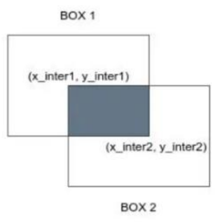
$IOU = \frac{\text{area of overlap}}{\text{area of union}} =$

Diagrammatic Representation of the formula to calculate IOU

Let us assume that box 1 is represented by [x1, y1, x2, y2], and box 2 is represented by [x3, y3, x4, y4].

<u>Calculate the area of Intersection of box1 and box2</u>

- Let us represent the co-ordinates of Intersected angle as x_inter1 , y_inter1 , x_inter2 , y_inter2.
- To calculate the top left corner of the intersection, we compare the top left corners of each of the boxes. To calculate the bottom right corner of the intersection, we compare the bottom right corners of each of the boxes.



x_inter1 = max(x1,x3)
y_inter1 = max(y1,y3)
x_inter2 = max(x2,x4)
y_inter2 = max(x2,x4)

- Now that we have the coordinates of the intersection, the area of the intersection is simply the area of the rectangle formed.

  Width = | x_inter2 – x_inter1 |
  Height = | y_inter2 – y_inter1 |
  Area_of_Intersection = Width * Height

<u>Calculate the Area of Union of Boxes</u>

- The Area of Union of Boxes is total area covered by both boxes. To find Area , we have to individual area of both boxes.

  Width1 = | x2 – x1 |
  Height1 = | y2 – y1 |
  Area1 = Width1 * Height1
  Width2 = | x4 – x3 |
  Height2 = | y4– y3 |
  Area2 = Width2 * Height2
- we have to subtract the area of intersection we calculated, from the total area of the two boxes.

Area_of_Union = Area1 + Area2 – Area_of_Intersection

<u>Calculate IoU</u>

IoU = Area_of_Intersection / Area_of_Union

### *D. Saving Processed Images*

In the image preprocessing stage of our project, we undertake critical tasks to prepare the dataset for subsequent analysis and model training. One of the core objectives is to create precise bounding boxes around license plates, eliminate redundant information, and ensure that the data is properly annotated for supervised learning. So Now, we dive into the significance of process of saving processed images and ground truth labeling.

Once we have refined the bounding boxes through NMS, we save the images with the accurate bounding box annotations. These images serve as the preprocessed data that will be used to train and evaluate our CNN model. The annotated bounding boxes visually represent the detected license plates and are crucial for teaching the model.

### *E. Ground Truth Labeling*

Ground truth labeling is an indispensable aspect of supervised learning, serving as the foundation upon which our Convolutional Neural Network (CNN) model is trained and evaluated. These labeled annotations allow our model to learn and generalize patterns from the data, ultimately improving its ability to classify license plates as HSRPs or non-HSRPs. Alongside saving the processed images, we assign ground truth labels to each license plate region. In our case, a label of '1' is assigned to HSRPs (High-Security Registration Plates), and '0' to non-HSRPs. Ground truth labeling is a vital component of supervised learning, enabling our CNN model to learn and generalize from the annotated data. Ground truth labeling serves several key purposes:

1. **Training Data:** Ground truth labels help us define the target variable that our model aims to predict. In our case, it signifies whether a license plate is an HSRP (labeled '1') or a non-HSRP (labeled '0'). This clear distinction guides the learning process.
2. **Model Evaluation:** During model evaluation, ground truth labels provide a basis for assessing the model's performance. The model's predictions can be compared to the true labels to calculate performance metrics such as accuracy, precision, recall, and F1 score.
3. **Data Quality Assurance:** Ground truth labeling is a mechanism to ensure the quality and accuracy of the dataset. By manually assigning labels to the license plates, we can correct errors and discrepancies that might exist in the raw data.
4. **Interpretability:** Ground truth labels add an interpretative layer to our dataset. They provide a clear and human-understandable context for the model's predictions.

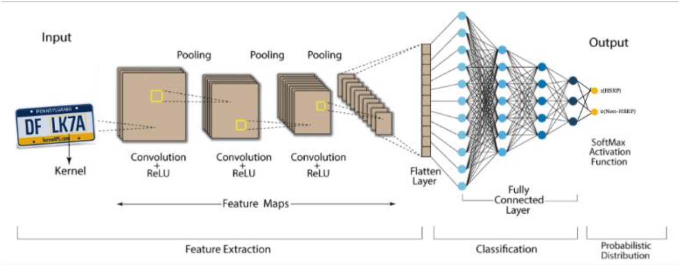
The combination of ground truth labeling and the preservation of processed images ensures that our dataset is both informative and suitable for training our recognition model. This meticulous preparation of the data is fundamental to the success and accuracy of our license plate recognition system.

## III. Algorithm

### A. *Introduction*

In our project, Convolutional Neural Networks (CNNs) play a pivotal role in the accurate detection of High-Security Registration Plates (HSRPs). CNNs are a class of deep learning models well-suited for image analysis and pattern recognition, making them an ideal choice for this task.

Inspired by a visual cortex biological system in the brain, CNN is an algorithm that can be used to recognize image patterns. It encompasses feature extraction and classification, as shown in Fig. The feature extraction consists of a convolutional layer, a pooling layer, and an activation function. Meanwhile, the classification part consists of fully connected layers. CNN uses the value of an image's pixel representation as a feature in the input layer, resulting in many features used in the input layer. The convolutional and pooling layers overcome this problem.



***Convolutional Neural Networks*** (CNNs) have an input layer, an output layer, numerous hidden layers, and millions of parameters, allowing them to learn complicated objects and patterns. It uses convolution and pooling processes to sub-sample the given input before applying an activation function, where all of them are hidden layers that are partially connected, with the completely connected layer at the end resulting in the output layer. The output shape is similar to the size of the input image.

Convolution is the process of combining two functions to produce the output of the other function. The input image is convoluted with the application of filters in CNNs, resulting in a Feature map. Filters are weights and biases that are randomly generated vectors in the network. Instead of having individual weights and biases for each neuron, CNN uses the same weights and biases for all neurons. Many filters can be created, each of which catches a different aspect from the input. Kernels are another name for filters.

### B. *Data Preparation*

Now we have collected a well – labelled dataset of HSRP and NON-HSRP images of license plate. Now we will preprocess the images including resizing means we will create an array of that images so that we can provide that array to CNN.

We will use cv2 library for this :

```
img = cv2.imread(str(image))
```

```
resized_img = cv2.resize(img,(720,360))
```

Now we will iterate a loop for all images and save all resized_image in a list , and at last list will be converted into array by numpy. This numpy array ahould be scaled between 0 and 1 so we will divide whole array by 255. Same as we have to do with target column which is list containing 0 and 1 , it should be converted into array.

Now we have X and Y array (X is array of Images and Y is ground truth labeled) and we have to split the dataset such that training data will be given to CNN to train the model and test dataset will be given to model to test model performance.

### C. *Model Architecture Design*

Defining the architecture of the CNN model, including the number of layers, filter sizes, and activation functions. Selecting the appropriate design for convolutional layers, pooling layers, and fully connected layers. Incorporating dropout layers or other regularization techniques to prevent overfitting.

- **Convolutional Layer**

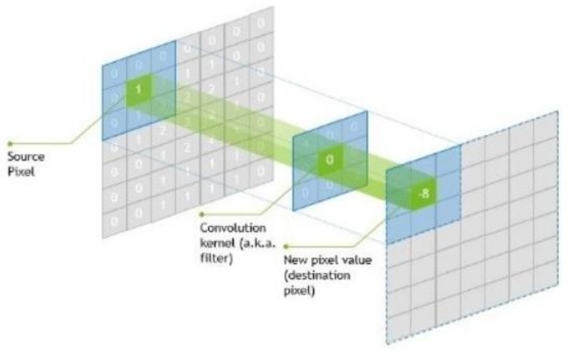
In convolutional neural networks, the major building elements are convolutional layers. The convolution layer is the core building block of the CNN. It carries the main portion of the network's computational load.

This layer often contains input vectors, such as an image, filters, such as a feature detector, and output vectors, such as a feature map. The image is abstracted to a feature map, also known as an activation map, after passing through a convolutional layer.

***Feature Map = Input Image x Feature Detector***

The feature detector is a two-dimensional (2-D) array of weights, which represents part of the image. While they can vary in size, the filter size is typically a 3x3 matrix; this also determines the size of the receptive field. The filter is then applied to an area of the image, and a dot product between two matrices, where one matrix is the set of learnable parameters otherwise known as a kernel, and the other matrix is the restricted portion of the receptive field. The kernel is spatially smaller than an image but is more in-depth. This means that, if the image is composed of three (RGB) channels, the kernel height and width will be spatially small, but the depth extends up to all three channels.

In the convolutional layers of a CNN, these convolutions are used to filter input data and find information.


Source Pixel
Convolution kernel (a.k.a. filter)
New pixel value (destination pixel)

The kernel's center element is put above the source pixel.

After that, the source pixel is replaced with a weighted sum of itself and neighboring pixels.

**Adding Convolutional Layer**

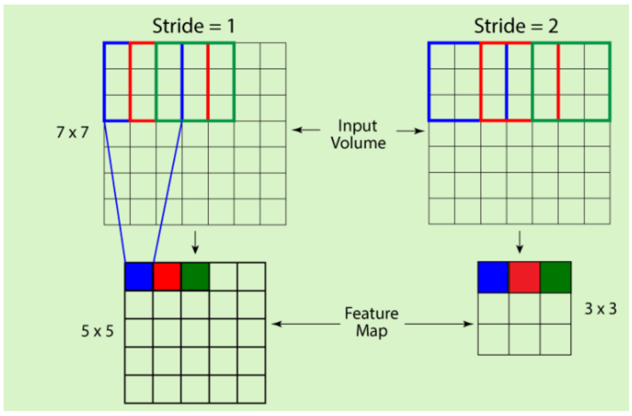
As we mentioned earlier, another convolution layer can follow the initial convolution layer. When this happens, the structure of the CNN can become hierarchical as the later layers can see the pixels within the receptive fields of prior layers.

**Activation Layer** (e.g., ReLU)

After each convolution operation, an activation function like the Rectified Linear Unit (ReLU) is applied element-wise to introduce non-linearity in the model.

- **Convolutional Padding and stride**

When the array is created, the pixels are shifted over to the input matrix. The number of pixels turning to the input matrix is known as the strides. When the number of strides is 1, we move the filters to 1 pixel at a time.



Similarly, when the number of strides is 2, we carry the filters to 2 pixels, and so on.

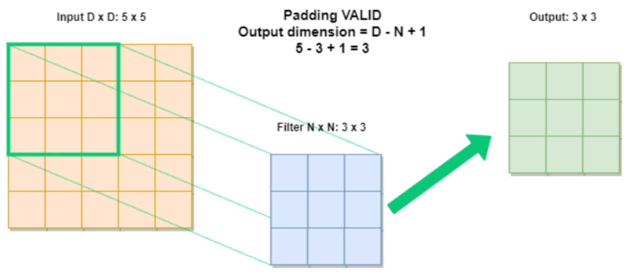
The padding plays a vital role in creating CNN. After the convolution operation, the original size of the image is shrunk. Also, in the image classification task, there are multiple convolution layers after which our original image is shrunk after every step, which we don't want.

Secondly, when the kernel moves over the original image, it passes through the middle layer more times than the edge layers, means corner layers don't participate with kernel as much as center layers do due to which there occurs an overlap. To overcome this problem, a new concept was introduced named padding. It is an additional layer that can add to the borders of an image while preserving the size of the original picture.

The padding algorithm takes 2 values either VALID or SAME and the padding is performed by adding values to the input matrix. The value used for padding is always zero.

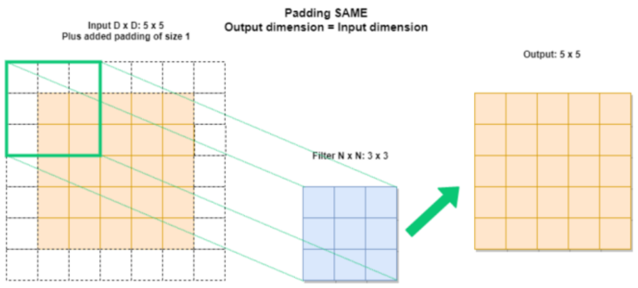
a) Valid Padding

When padding == "VALID", the input image is not padded. This means that the filter window always stays inside the input image. This type of padding is called valid because for this padding only the valid and original elements of the input image are considered. When padding == "VALID", there can be a loss of information. Generally, elements on the right and the bottom of the image tend to be ignored. In this case, the size of the output image <= the size of the input image.



b) Same Padding

When padding == "SAME", the input is half padded. The padding type is called SAME because the output size is the same as the input size(when stride=1). Using 'SAME' ensures that the filter is applied to all the elements of the input. Normally, padding is set to "SAME" while training the model. Output size is mathematically convenient for further computation.


Input D x D: 5 x 5
Plus added padding of size 1
Padding SAME
Output dimension = Input dimension
Output: 5 x 5
Filter N x N: 3 x 3

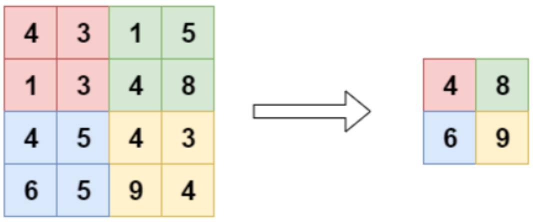
- **Pooling Layer**

The pooling layer is another building block of a CNN and plays a vital role in pre-processing an image. In the pre-process, the image size shrinks by reducing the number of parameters if the image is too large. When the picture is shrunk, the pixel density is also reduced, the downscaled image is obtained from the previous layers. Basically, its function is to progressively reduce the spatial size of the image to reduce the network complexity and computational cost. Spatial pooling is also known as downsampling or subsampling that reduces the dimensionality of each map but retains the essential features. A rectified linear activation function, or ReLU, is applied to each value in the feature map. Relu is a simple and effective nonlinearity that does not change the values in the feature map but is present because later subsequent pooling layers are added. Pooling is added after the nonlinearity is applied to the feature maps. There are two types of spatial pooling:

a. Max-pooling :

It selects maximum element from the feature map. The resulting max-pooled layer holds important features of feature map. It is the most common approach as it gives better results.

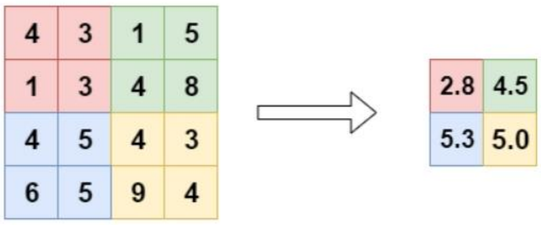
If we apply MaxPooing2D((2,2)), max pooling of 2,2 filters on image , as we can see the image :



b. Average pooling :

It involves average calculation for each patch of the feature map.

Average pooling of (2,2) filters

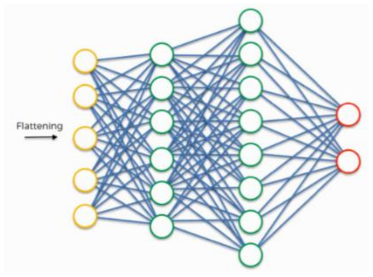


Pooling is Important because it progressively reduces the spatial size of representation to reduce amount of parameters and computation in network and also controls overfitting. If no pooling, then the output consists of same resolution as input.

- **Fully Connected Layer**

  The fully connected layers (also known as dense layers) process the output from the convolutional layers to make final predictions. These layers connect every neuron to every neuron in the previous and subsequent layers. The final fully connected layer typically has the number of neurons corresponding to the number of output classes.

  For example :



1(HSRP)

0(Non -HSRP)

The First three layers is Dense layer and last is a output layer

- **Output Layer**

  The output layer provides the final predictions of the model. The activation function used in this layer depends on the task. For binary classification, a sigmoid function is often used, while for multi-class classification, a softmax function is common

  we have used sigmoid function for binary classification. In binary classification, there is typically one neuron in the output layer. The sigmoid function squashes the output to a range between 0 and 1, which can be interpreted as the probability that the input belongs to the positive class.

### D. *Model Compilation*

Compiling the CNN model by specifying the loss function (e.g., binary cross-entropy for binary classification), the optimizer (e.g., Adam or SGD), and evaluation metrics (e.g., accuracy).

### E. *Training*

- Feed the training dataset into the model to initiate the training process.
- Iteratively adjusting the model's weights and biases to minimize the loss function using backpropagation.
- Monitoring training metrics (e.g., loss and accuracy) to assess the model's performance during training.
- Saving checkpoints to track progress and enable model recovery in case of interruptions.

### F. *Testing*

- At the time of train test split , we have test dataset so feed test dataset to know actually model performance.
- Calculating metrics such as accuracy, precision, recall and F1 score measure the model's effectiveness.

### G. *Results*

- The results of our license plate recognition system, built upon YOLOv3 for object detection and a CNN for classification, reflect the effectiveness and generalization of our integrated approach.
- In the training phase, the model exhibited outstanding performance. The CNN achieved an accuracy of 98% on the training dataset, demonstrating its ability to learn and recognize distinctive patterns associated with High-Security Registration Plates (HSRPs) and non-HSRPs. This remarkable training accuracy underscores the capability of our model to capture intricate features and variations present in the data. The true test of our model's performance came during the evaluation phase on the test dataset, representing unseen and real-world scenarios. In this critical stage, our system maintained a commendable accuracy of **90%**. This accuracy is a testament to the model's capacity to generalize from the training data, robustly detecting license plates and distinguishing HSRPs from non-HSRPs across diverse lighting conditions, backgrounds, and license plate designs.
- The combination of YOLOv3 for object detection and a well-designed CNN for classification has proved to be a potent tool for accurately identifying license plates, a critical task in applications ranging from traffic management to law enforcement and security systems.

## IV. Conclusion

In this paper, we have presented an integrated approach to license plate recognition, combining object detection with YOLOv3 and classification with Convolutional Neural Networks (CNN). Our goal was to distinguish High-Security Registration Plates (HSRPs) from non-HSRPs in a diverse and comprehensive dataset. We began by collecting data from two distinct sources: online vehicle listings on OLX and images from Google Images. The integration of these datasets allowed us to address the challenges of real-world variations, lighting conditions, and backgrounds that license plates may encounter.

The image preprocessing stage involved the creation of precise bounding boxes using YOLOv3, Non-Maximum Suppression (NMS), and Intersection over Union (IOU) thresholds. This stage ensured that our dataset contained accurately annotated images, which were crucial for the subsequent training of our CNN model. Ground truth labeling was also assigned, setting the stage for supervised learning. Our CNN model, designed to recognize HSRPs and non-HSRPs, played a pivotal role in this project. It efficiently learned and extracted relevant features from the images during training, enabling it to make accurate predictions. The incorporation of the ReLU activation function in the fully connected layers introduced non-linearity, facilitated feature learning, and addressed the vanishing gradient problem. This, combined with the model's ability to process high-level features, contributed to its impressive performance in detecting license plates.

In the testing and evaluation phase, our model exhibited remarkable accuracy and generalization, effectively distinguishing between HSRPs and non-HSRPs. Performance metrics such as accuracy, precision, recall, and F1 score showcased the model's effectiveness in real-world scenarios.

In conclusion, our integrated approach using YOLOv3 and CNN has demonstrated the potential to significantly impact the field of license plate recognition, contributing to enhanced security and efficiency in transportation and related systems.

## ACKNOWLEDGMENT

We would like to express our gratitude to all those who contributed to the successful completion of this research project. First and foremost, we extend our appreciation to our advisors and mentors for their invaluable guidance and support throughout this project. We also wish to acknowledge the diligent efforts of the research team, including data collectors, annotators, and those involved in the model development process. Your dedication and hard work have been pivotal to the project's success.

## REFERENCES

[1] Google Images Dataset link :
https://www.kaggle.com/datasets/saisirishan/indian-vehicle-dataset?select=google_images

[2] Automated License Plate Recognition: A Survey on Methods and Techniques. J., Shashirangana, et al. IEEE Access, Vol. 9, pp. 11203-11225.

[3] S.S Nadiminti, P.K Gaur and A. Bhardwaj , "Exploration of an End-to-End Automatic Number plate Recognition neural network for Indian datasets" .

[4] Miss. S.S. Ghadage, Mr S.R. Khedkar, "A Review Paper on Automatic Number Plate Recognition System using Machine Learning Algorithms".

[5] Real-time Bhutanese license plate localization using YOLO. Yonten, Jamtsho, Panomkhawn, Riyamongkol and Rattapoom, Waranusast. 2, s.l. : ICT Express, Vol. 6.