Vehicle Number Plate Detection using custom CNN

(DL Mid Evaluation)

Prepared by

Kushagra Gupta

(2022BTECH051)

(Section – A)

Under the guidance of

Kshitiz Verma

Submitted in
partial fulfillment of the requirements
for the course of

Deep Learning (CS1218)

Institute of Engineering & Technology (IET)
JK Lakshmipat University

(April 2025)

INTRODUCTION

1.1 Objective of the project

Building a **custom CNN** from scratch without relying on pre-built models like YOLO.

1.2 Importance

Importance of number plate detection is in traffic surveillance, law enforcement, smart city, etc.

But the main idea behind this is to make this for the highways such that there is no need for toll plazas and the vehicles are charged automatically, according to the distance they travelled on the highway.

PROBLEM STATEMENT

The increasing vehicular traffic on highways necessitates efficient toll collection systems to minimize congestion and enhance travel experience. Traditional toll plazas often lead to delays, fuel wastage, and increased emissions due to manual processing. While Electronic Toll Collection (ETC) systems like FASTag have addressed some of these issues, they still require dedicated infrastructure and can face challenges like tag misreads or account issues. An alternative approach involves leveraging Automatic Number Plate Recognition (ANPR) systems to identify vehicles and automate toll collection without the need for physical tags. However, existing ANPR solutions often rely on pre-trained models, which may not generalize well across diverse conditions prevalent on Indian highways, such as varying plate designs, lighting conditions, and vehicle types. Therefore, there is a need to develop a robust, custom-trained CNN model capable of accurately detecting number plates under these challenging conditions, facilitating seamless and efficient toll collection.

DATASET DESCRIPTION

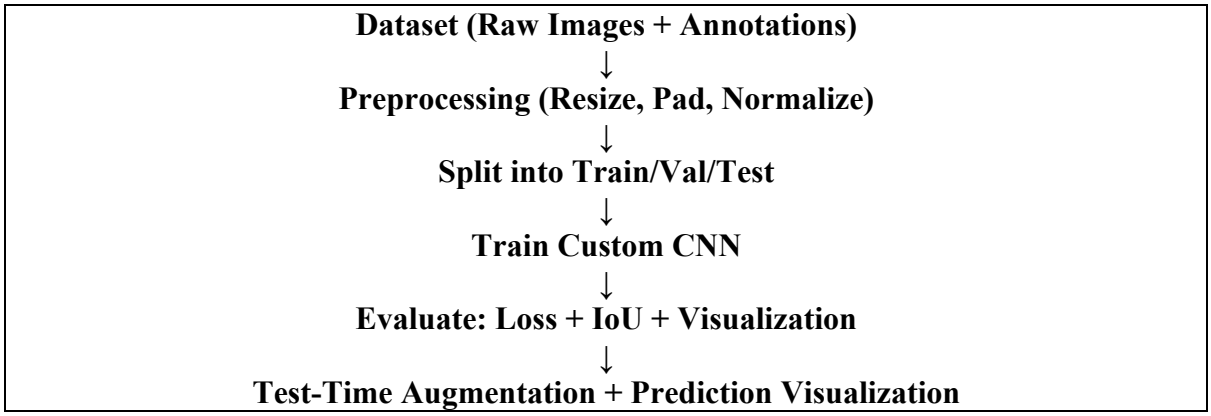
- Type:
  - Images with annotations of number plates
  - Some negative samples also for better model training
- Source:
  - 5 different **Kaggle datasets** having images and annotations in xml (Pascal VOC) and txt (YOLO) format.
  - Having car, bike, scooter, electric, bus, truck number plates with variety of environment.
- Size:
  - Around **3.5 GB** of data
  - **9.6k** images – Around **8k** after cleaning and preprocessing
  - **Split** – Train (80%) Validation (10%) Test (10%)
  - **Dimensions:** Resized to **416 x 416** + padding

PREPROCESSING

- Image resizing + aspect ratio padding
- Bounding box normalization
- For OCR later:
  - Grayscale conversion
  - Edge Detection using Canny

METHODOLOGY

2.1 Workflow Chart



2.3 Model Architecture

Custom CNN

- 6 Convolution Layers with ReLU
- BatchNorm2D added after some Conv layers
- MaxPooling layers
- Flatten → Dense → Output layer (4 neurons for bbox: x_min, y_min, x_max, y_max)
- Dropout(0.3) before Dense
- Total trainable parameters: 52.1M+

2.4 Model Input and Output

- Input: (3, 224, 224) rgb image
- Output: (x_min, y_min, x_max, y_max) in unnormalized or normalized coordinates

2.5 Training

- Adam Optimizer
- Learning rate: 1e-3 with ReduceLROnPlateau
- Evaluation Metric – IoU
- Epochs – 10

ABLATION STUDY

1. **1st Approach** of CNN Model
   3 → 32 → 32 → 64 → 128 → 128 → 256
   512 * 26 * 26 = 346,112 input neurons to the Dense layer. Which is huge
   346k × 1024 = **354M+** parameters just in one FC layer
   Slow training & Possible overfitting

2. **2nd Approach** with better CNN Model
   3 → 16 → 32 → 64 → 128 → 256 → 512
   Faster, deeper, but more parameters and still slower with slight overfitting

   1. (3 x 3 x 3 x 16) + 16 = 448
   2. (3 x 3 x 16 x 32) + 32 = 4640
   3. (3 x 3 x 32 x 64) + 64 = 18496
   4. (3 x 3 x 64 x 128) + 128 = 73856
   5. (3 x 3 x 128 x 256) + 256 = 295168
   6. (3 x 3 x 256 x 512) + 512 = 1180160

   Total = 16,25,768 = **1.6M+**

3. **3rd Approach** with Best fit CNN Model
   3 → 16 → 32 → 64 → 64 → 128 → 512
   Lighter, faster, lower overfitting risk, but less complex features extraction
   Great balance of depth, speed, and accuracy

   1. (3 x 3 x 3 x 16) + 16 = 448
   2. (3 x 3 x 16 x 32) + 32 = 4640
   3. (3 x 3 x 32 x 64) + 64 = 18,496
   4. (3 x 3 x 64 x 64) + 64 = 36,928
   5. (3 x 3 x 64 x 128) + 128 = 73,856
   6. (3 x 3 x 128 x 512) + 512 = 5,90,336

   Total = 7,24,704 = **700k**

   <u>Batch Norm 2D</u>

   32+64+128+128+256+1024 = 1632

   <u>Fully Connected Layer</u>

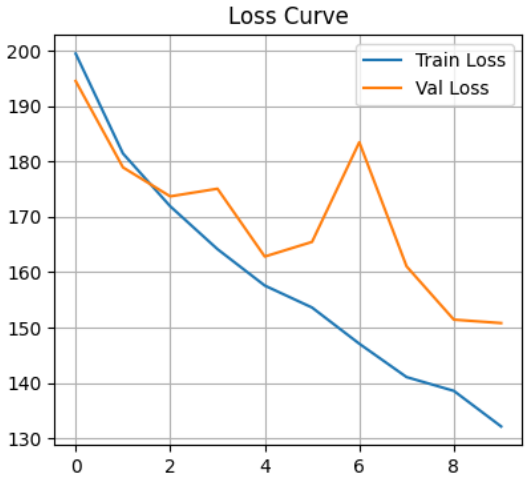
   Linear (512 * 14 * 14, 512) = (100352 × 512) + 512 = 5,13,81,376

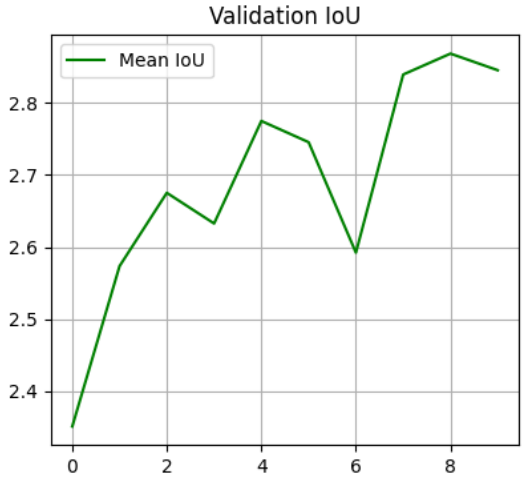
   Linear (512, 4) = (512 × 4) + 4 = 2,052

   Total FC Parameters: 5,13,83,428

Total Parameter = 52.1M+

- Input of (3, 224, 224) rgb image
- 6 Convolutional Blocks
- Each Block having:
  - Conv2D – 3x3 filter with padding=1
  - Batch Norm 2D
  - ReLU Activation
  - Max Pooling after block 2, 4, 5 & 6
- Final feature map size: (512, 14, 14)
- Regressor
  - Flatten – Converts (512, 14, 14) – 100352
  - Linear Layer: 100352 – 512
  - ReLU + Dropout(0.3)
  - Linear Layer: 512 – 4 (bounding box coordinates: [xmin, ymin, xmax, ymax])
- Training
  - Adam Optimizer
  - Loss Function – SmoothL1Loss

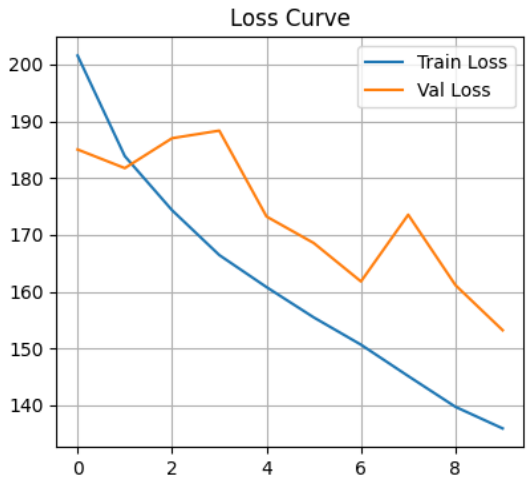


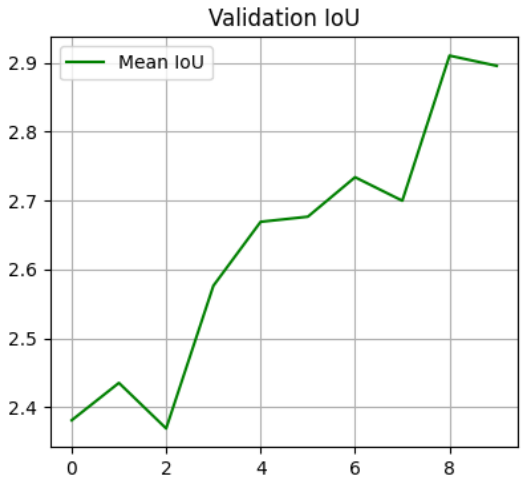


4. 4th Approach

Here, CNN, Optimizer, Loss Function, IoU function and Epoch Loop is same.
Improvements are:

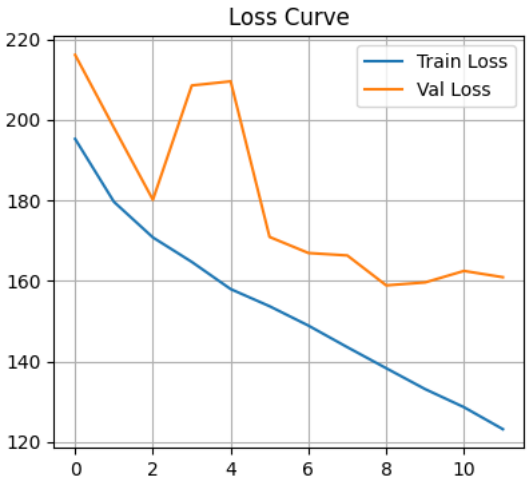
- Better Loss Averaging – More accurate per-epoch loss values vs. summing all and reporting directly.
- Evaluation on Validation Set
  Saves memory by disabling gradient tracking (no_grad())
- Better Mean IoU Computation
  Aggregating IoU per sample rather than skipping it.

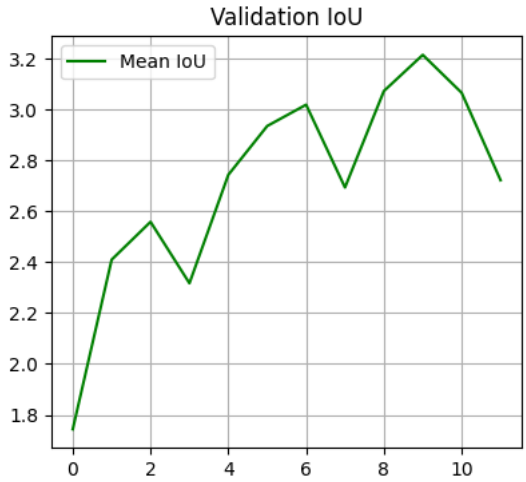




5. 5th Approach

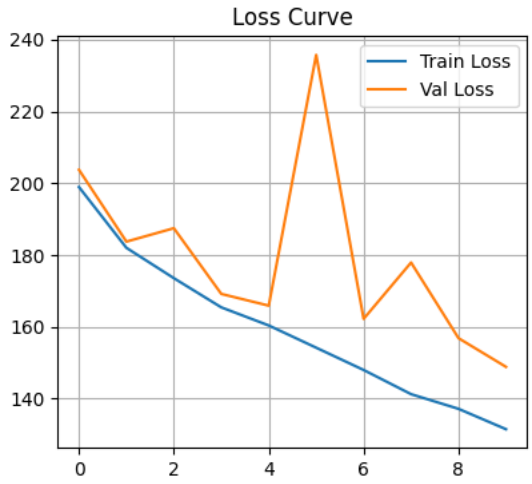
- + Early Stopping with patience=3, minimum delta=1e-4. Called for each epoch.
- Training epoch increased.
- Instead, bounding epochs, it is adaptable to the performance.
- Low risk of over-fitting
- Efficient training

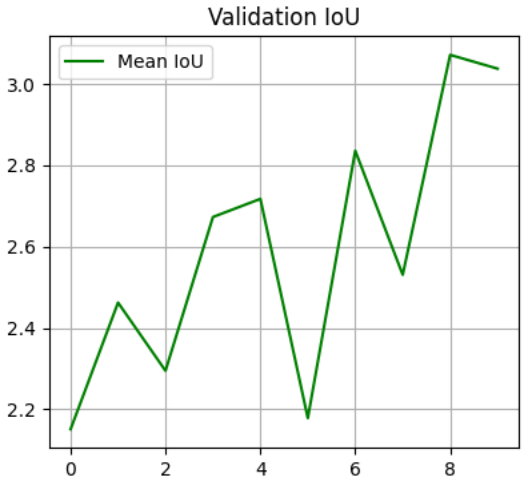




6. 6th Approach

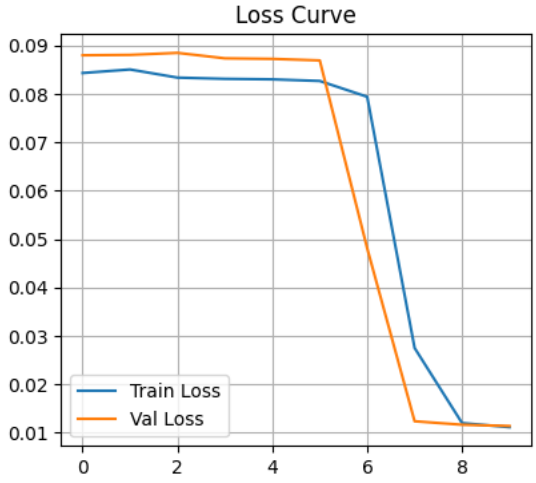
+ Batch Norm after every Conv, Dropout (0.3) and SmoothL1 are more stable.
Better image resolution and feature map sizing – 224 → 112 → 56 → 28 → 14
+Data Augmentation – TTA with inverse mapping and averaging

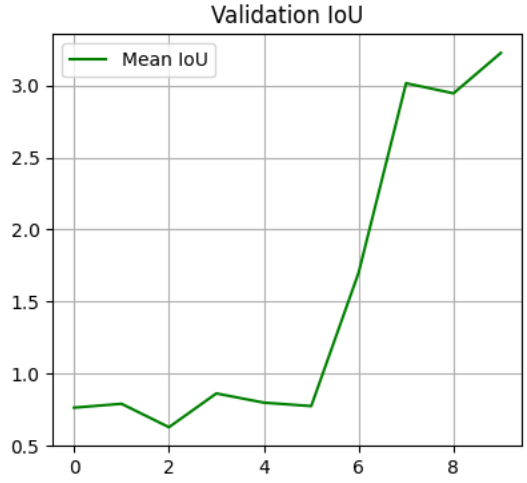




7. 7th Approach

- Architecture
  Conv2d → BatchNorm2d → ReLU → (several times) → MaxPool → Flatten → Dense → Dropout → Dense(4) → Sigmoid
- Final Sigmoid ensures output is between [0, 1] (since boxes are normalized)
- Test-Time Augmentation (TTA)
  3 versions of image: Original, Horizontally flipped, brightness increased. Prediction is made on all 3 averages. inverse_tta() corrects flipped predictions back to original coordinates.
- +Prediction Visualization
- Loads training and Validation sets with batch size 16





RESULTS COMPARISION

Approach	Trainable Parameters	Train Loss	Val Loss	Mean IoU (%)
3	52.1M+	0.0113	0.0125	71.48
4	52.1M+	0.0115	0.0129	72.40
5	52.1M+	0.0101	0.0140	77.60
6	52.1M+	0.0113	0.0121	75.23
7	52.1M+	0.0111	0.0114	73.26

PERFORMANCE VISUALIZATION

- Loss Curve – Train VS Val Loss
- Validation IoU Graph – Over epoch
- Prediction Visualization
  - Input image
  - Ground Truth box (green)
  - Predicted Box (red)
  - For 5 samples

TEST SET EVALUATION

- Mean IoU – 0.2145
- Visual inspection shows poor prediction alignment with ground truth
- Issues Identified:
  - Bounding box predictions are off target even when IoU score is > 0.2
  - Test-time augmentation may have overfit or caused instability

PROBLEMS IDENTIFIED

- Not predicting well and vague IoU even though the validation metrics looked okay.
- A peak is observed in the training-validation loss graph.
- 52.1M+ parameters from just 1 FC layer, which is too large and inefficient. Can Global Average Pooling before the dense layer or replacing FC with convolutional bounding box regressor can help ?
- mAP metric and Precision/Recall for evaluation missing
- Overfitting handling other than Dropout.
- Normalizing bounding boxes
- Try deeper CNN or pretrained models for feature extraction (ResNet features)

FUTURE WORK

- Better detection using pre-built models. Reflecting upon the why this didn't work.
- Implementing approaches more broadly and analyzing using better metrics.
- Implementing OCR.

REFERENCES

1. Jawale, M.A., William, P., Pawar, A.B. and Marriwala, N., 2023. Implementation of number plate detection system for vehicle registration using IOT and recognition using CNN. Measurement: Sensors, 27, p.100761.
2. Dorbe, N., Jaundalders, A., Kadikis, R. and Nesenbergs, K., 2018. FCN and LSTM based computer vision system for recognition of vehicle type, license plate number, and registration country. Automatic Control and Computer Sciences, 52, pp.146-154.
3. Soomro, S.R., Javed, M.A. and Memon, F.A., 2012, October. Vehicle number recognition system for automatic toll tax collection. In 2012 international conference of robotics and artificial intelligence (pp. 125-129). IEEE.
4. Lubna, Mufti, N. and Shah, S.A.A., 2021. Automatic number plate Recognition: A detailed survey of relevant algorithms. Sensors, 21(9), p.3028.
5. Chauhan, R.K. and Chauhan, K., 2022. Intelligent toll collection system for moving vehicles in India. Intelligent Systems with Applications, 15, p.200099.