Automatic Number Plate Recognition

By Reuben Kurian

Aims & Objectives

- The aims and objectives of this project include:
- - Creating an automatic number plate recognition system using machine learning techniques.
- - Creating a web application and integrate the ANPR pipeline into the web application for other people to use.
- - Experimenting with different model types, various methods and transfer learning.
- - Determining how effective machine learning can be with license plate recognition.

Project Requirements

- Aiming to create an accurate number plate recognition system, ensuring that each component in the pipeline performs well and accurately recognises license plates.
- Create a web application that is scalable to open up deployment opportunities.
- Ensure that the web application is user-friendly and the UI is clean and tidy.

Dataset

- Combination of Kaggle and Roboflow dataset
- Utilised for character recognition model
- In total, about 25000+ images (for training, testing and validation sets)
- Downsized due to computational limitations
- Took 210 images at random of each class for training dataset, and 45 images at random of each class for testing and validation separately.
- In total after downsize, there are 300 images per class for 36 classes (10800 total images).
- Subfolders are labelled accordingly, with images matching the label within the subfolder.

Software Platforms

- Character recognition model trained on Google Colab due to resource requirements
- License plate detection, character segmentation, formation of the pipeline and the web app were developed on Visual Studio Code.

Libraries, Frameworks and Languages

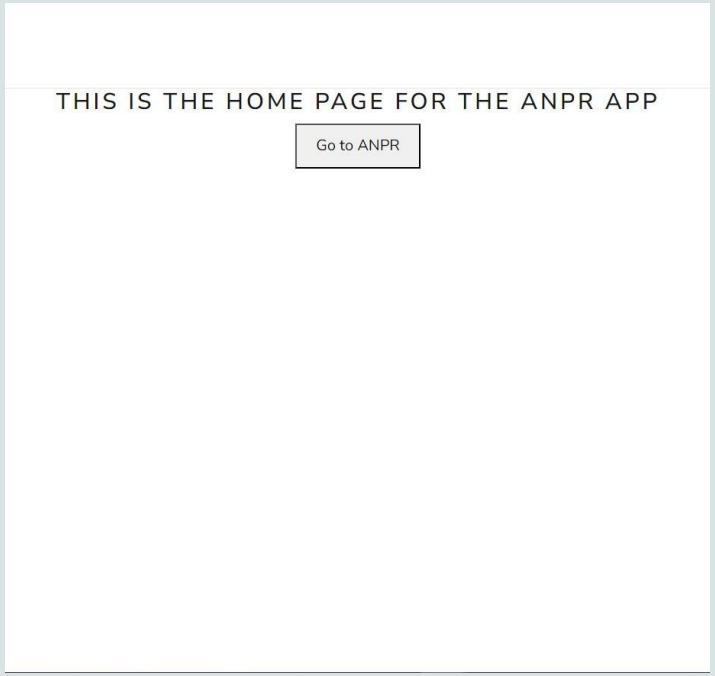
- Python was the language used to develop the code.
- Few libraries were utilised to create the project
- Main libraries include Tensorflow. OpenCV, Numpy.
- ‘Helper’ libraries also used to assist.
- Tensorflow for model development, Numpy and OpenCV for license plate detection and character segmentation.
- Django utilised as framework for the web application.
- HTML, CSS, JavaScript, Jinja2 and Bootstrap were employed to develop the web application.

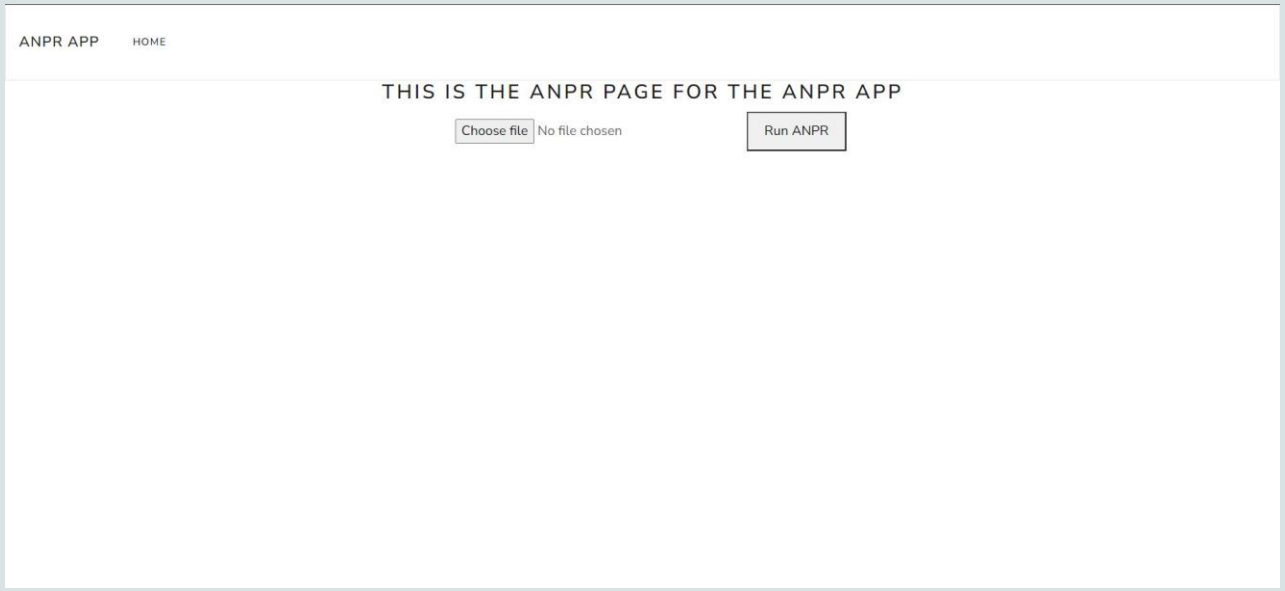
Design of ANPR

- Three components connected in a pipeline-fashion
- First component – license plate detection
- Second component – character segmentation
- Third component – character recognition

- All three components connected in a sequential fashion facilitate the ANPR system.

Design of Web Application





The image to the left displays how the home page of the ANPR would look. The image above displays how the main ANPR page, found on the /run-pipeline path, would look like.
Design I went for is a simplistic, clean, user-friendly UI.

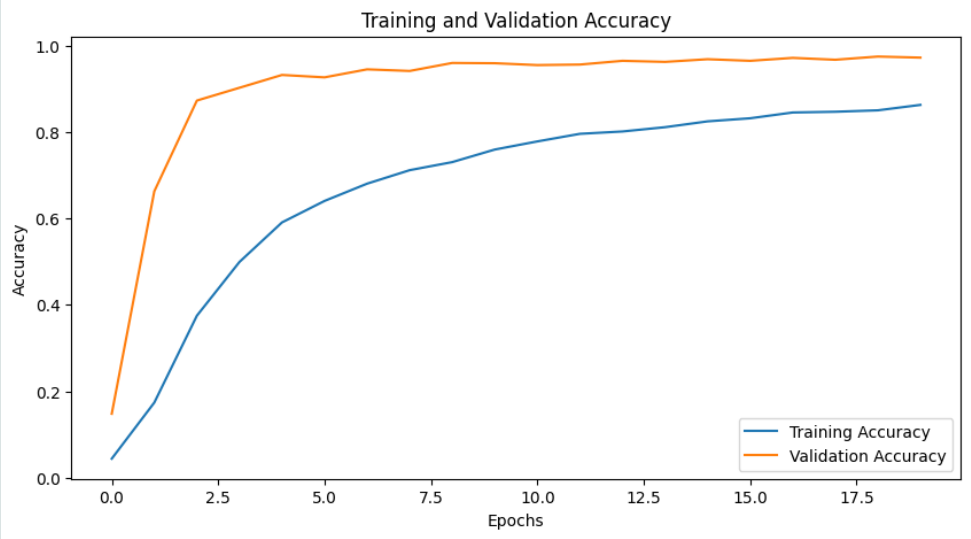
Results:

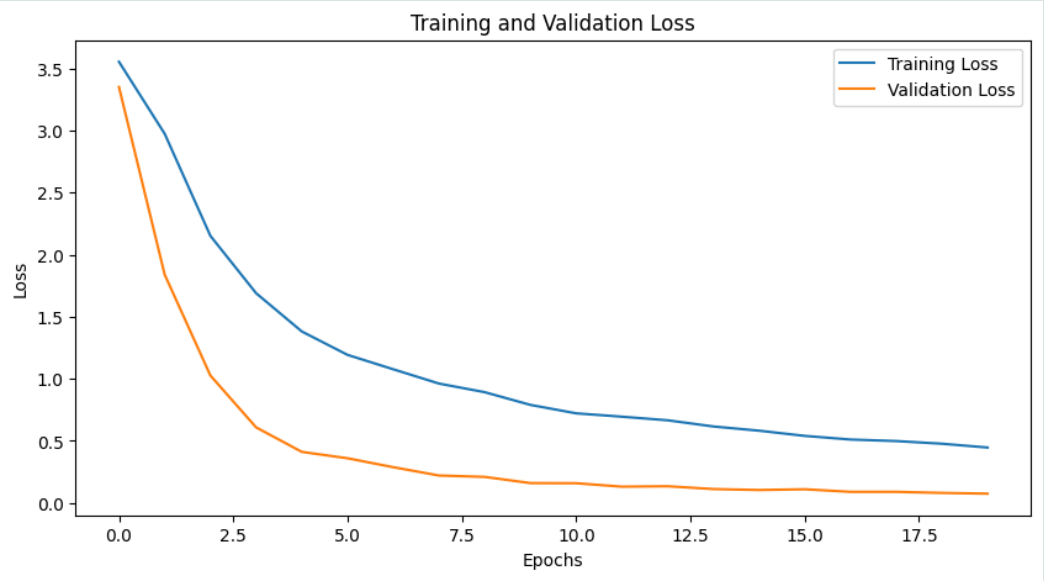
The report on the left shows the classification report's results on the transfer-learned model. The report on the right illustrates the classification report's results on the custom model.

Class	Metrics			Support
	Precision	Recall	F1-score	
0	0.68	0.41	0.51	32
1	0.76	1.00	0.86	34
2	1.00	1.00	1.00	32
3	1.00	0.94	0.97	35
4	1.00	0.97	0.99	35
5	1.00	0.94	0.97	35
6	1.00	0.97	0.98	33
7	1.00	1.00	1.00	34
8	0.95	1.00	0.97	36
9	1.00	0.95	0.97	39
A	1.00	1.00	1.00	37
B	0.97	0.97	0.97	36
C	0.92	0.94	0.93	35
D	1.00	0.97	0.98	33
E	0.97	0.91	0.94	32
F	0.91	0.97	0.94	31
G	0.88	0.88	0.88	33
H	1.00	1.00	1.00	35
I	1.00	0.39	0.56	28
J	0.82	0.96	0.89	28
K	1.00	0.87	0.93	31
L	0.97	1.00	0.99	33
M	1.00	0.97	0.99	34
N	0.94	1.00	0.97	34
O	0.55	0.82	0.66	28
P	1.00	1.00	1.00	35
Q	0.93	1.00	0.97	28
R	0.89	0.97	0.93	33
S	1.00	1.00	1.00	37
T	0.97	1.00	0.98	31
U	1.00	1.00	1.00	37
V	1.00	1.00	1.00	32
W	1.00	1.00	1.00	28
X	0.97	0.97	0.97	37
Y	0.94	1.00	0.97	33
Z	1.00	1.00	1.00	39
Overall Metrics				
Accuracy		0.94		1203
Macro Avg	0.94	0.94	0.94	1203
Weighted Avg	0.95	0.94	0.94	1203

Class	Metrics			Support
	Precision	Recall	F1-score	
0	0.68	0.59	0.63	32
1	0.92	1.00	0.96	34
2	1.00	1.00	1.00	32
3	1.00	0.97	0.99	35
4	1.00	0.97	0.99	35
5	1.00	0.97	0.99	35
6	1.00	0.97	0.98	33
7	1.00	1.00	1.00	34
8	1.00	1.00	1.00	36
9	1.00	1.00	1.00	39
A	0.97	1.00	0.99	37
B	1.00	1.00	1.00	36
C	0.80	1.00	0.89	35
D	1.00	1.00	1.00	33
E	0.97	1.00	0.98	32
F	1.00	0.97	0.98	31
G	1.00	0.70	0.82	33
H	1.00	1.00	1.00	35
I	1.00	1.00	1.00	28
J	1.00	0.96	0.98	28
K	1.00	1.00	1.00	31
L	1.00	1.00	1.00	33
M	0.94	1.00	0.97	34
N	1.00	1.00	1.00	34
O	0.59	0.68	0.63	28
P	1.00	1.00	1.00	35
Q	1.00	1.00	1.00	28
R	0.97	1.00	0.99	33
S	1.00	1.00	1.00	37
T	1.00	1.00	1.00	31
U	1.00	1.00	1.00	37
V	1.00	1.00	1.00	32
W	1.00	1.00	1.00	28
X	1.00	0.97	0.99	37
Y	1.00	1.00	1.00	33
Z	1.00	1.00	1.00	39
Overall Metrics				
Accuracy		0.97		1203
Macro Avg	0.97	0.97	0.97	1203
Weighted Avg	0.97	0.97	0.97	1203

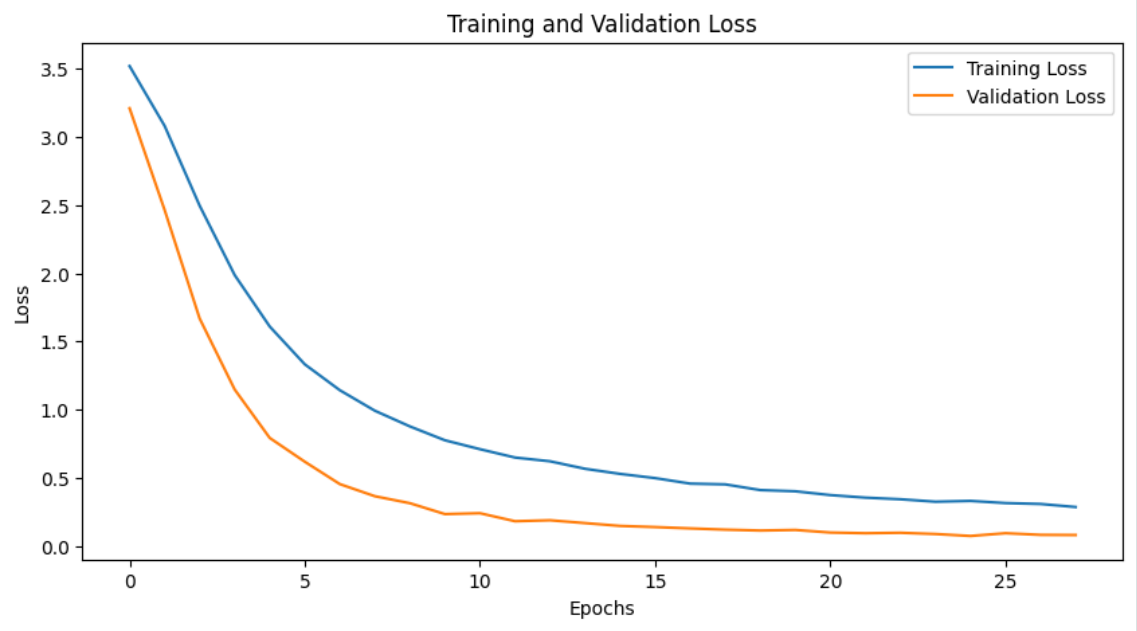
Results





Graph on the left portrays the training and validation accuracy on the custom model. Graph above displays the training and validation loss on the custom model

Results:



Graph on the left portrays the training and validation accuracy on the VGG-16 transfer learned model. Graph above displays the training and validation loss on VGG-16 transfer-learned model

Discussion

- Complex architecture possibly lead to overfitting, which led to slightly worse results.
- Both models have poor performance with the classes 0 and O.

Critical Assessment

- Strengths: ease of obtaining dataset
- Abundance of resources
- Weaknesses: lack of computational power
- Poor time management
- Strengths of project: - user friendly UI
- Strong character recognition model
- Lessons learned: comprise better time management
- Look into more complex architectures.

Links

- Link to video: Video Link
- Link to code: Code Link