Eluvio Challenge 2022 – Logo Recognition Report By Ankita Singh

Table of Contents

1.Objective

Given a custom logo dataset, perform a series of experiments to explore the dataset, develop a model for car brand recognition and analyze the results.

2.Tools and Packages

Language: Python

Packages: numpy, opencv, keras, sklearn, pandas, tensorflow, matplotlib

Tools: Python-tesseract[1]

3.Exploratory Data Analysis

The training dataset consists of 18 classes of car brand names with 50 images per class. The test dataset consists of 103 total images from all classes. Figure 1 shows some sample images from train and test data:

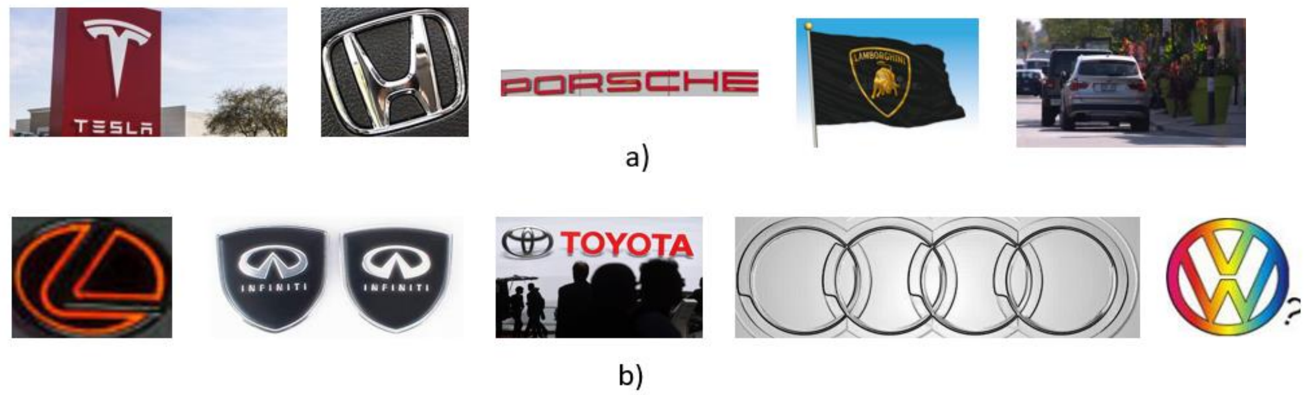


Figure 1.a) Sample Images from the training dataset. b) Sample images from the test dataset

3.1. Dimension Distribution Across Training and Test Data:

The distribution of dimensions of available images in training and testing dataset is examined. Figure 2 contains the scatter plot for image dimensions. As observed, the dimension of images in test dataset is towards the lower range. Training on high resolution images while testing on low resolution can cause performance issues. Resizing of images in a suitable range(according to test dataset) can alleviate this problem.

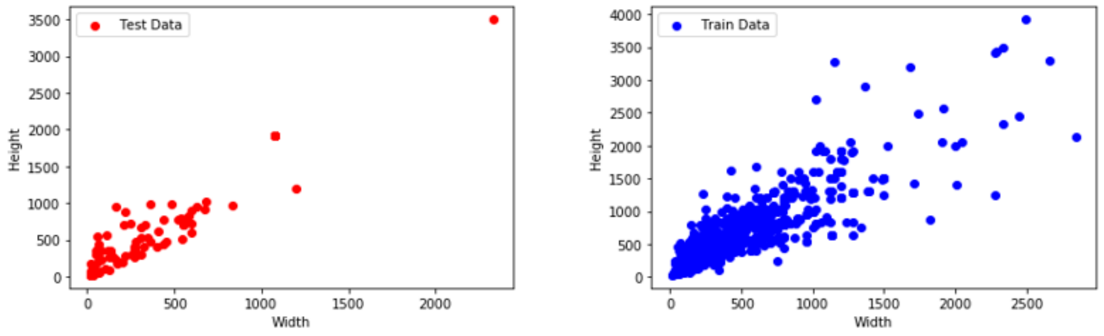


Figure 2 Distribution of dimensions across the training and test data.

3.2. Image Formats

It is useful to determine the formats of the given images as I don't want to run into problems because of an unreadable format during training. According to Figure 3, the dataset contains only jpg, png and jpeg formats.

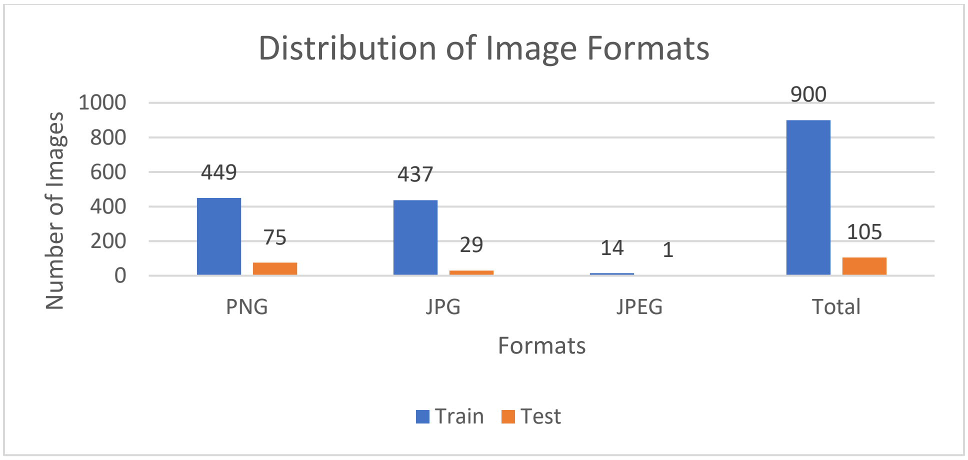
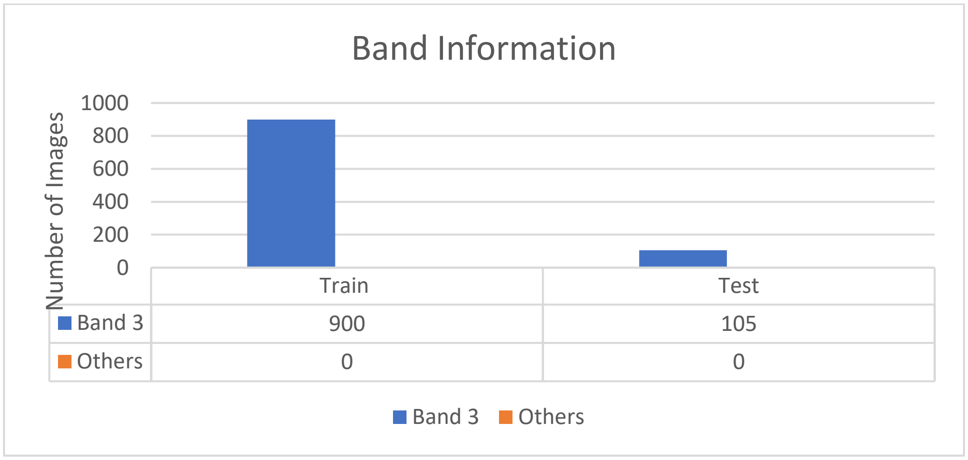


Figure 3. Distribution of image formats across training and test dataset. All the images are in readable format.

3.3. Band Information

A quick look at the dataset points towards presence of RGB images. As seen in Figure 4, the band information is confirmed through count of images with band 3 being equal to the total images present in the given dataset.



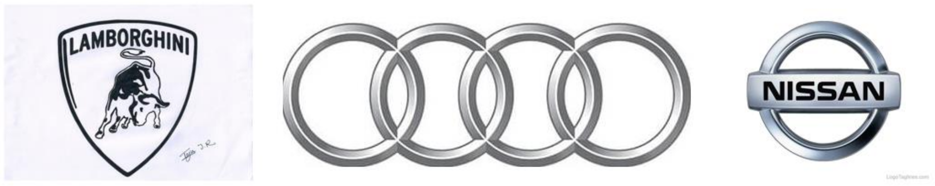
	Train	Test
Band 3	900	105
Others	0	0

Figure 4 Band information of images. All images had RGB band.

3.4. Types of Images

Looking at the sample images, I can categorize the images into 3 types:

1. Recognizing car brand from clean car logo images.



2. Recognizing car brand from images containing text(car brand name).

3. Recognizing car brand from images where the logo or car name is not localized. These images are noisy and contains irrelevant information. For example, images of car showrooms or a snapshot from day traffic.

It is also observed that images in train are cleaner than images in test. Test contains various images from second and third category, representing a dataset closer to real world.

3.5. General Observations

The following observations can help us in identifying data augmentation techniques that can possibly bring the domain of train and test dataset closer to each other.

1. The test dataset contains rotated images of logos at various angles
2. The test dataset contains horizontally flipped images of logos.
3. Multiple images in the test dataset are blurred and of very low resolution.
4. Multiple images in test come from third category as identified in Section 3.4. They require extra step of selective search for object location detection before performing logo recognition.

4.Data Preprocessing

4.1 Resize Data: Since, CNNS take input images of same size, I need to resize our images. I want to choose image size which is in the test domain(i.e., towards the smaller dimension size) and is big enough so that I can compare results of various off the shelf pretrained networks. I chose 224,224 as the image size for this study.

4.2 Normalizing: Image normalization is preferred for neural networks(I will use Convolution Neural Networks for modeling in this study). Without normalization, the shared weights will have different adjustment for different features and may not converge efficiently.

4.3 Segmentation for Images containing text

I am using pretrained Python Tesseract to identify text in images. This requires image segmentation as preprocessing step. For this study, binary threshold segmentation is used. Based on parameter tuning and associated results the threshold for binary segmentation is selected as 160. If time allows, I can study the effect of applying different image segmentation techniques on overall performance in future.

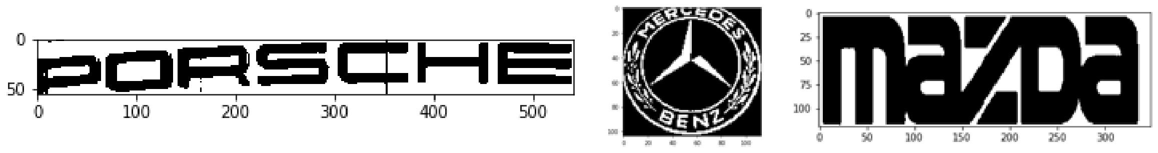


Figure 5: Sample images after segmentation using binary threshold(160)

5.Model Selection

An off the shelf network, pre trained on imagenet dataset is used as a base model for this study. This way I can use the model's weights that has already learnt the task of object detection and use the borrowed knowledge for our study. This will save us both time and resources.

Preprocessed Images
Pre-Trained CNN Model
Predicted Car Brand

Figure 6 Framework for initial experimentation using various pretrained models.

There are multiple models available in the keras API[2]. For initial comparison and selection, I trained these models on the original training dataset by freezing all layers and adding a dense layer on top for task specific classification. Due to resource constraint, I kept no. of epochs = 5 for each model. Our objective here is to choose a model that gives decent accuracy with comparatively smaller number of parameters. Table 1. below shows the model's performances. **The train dataset is split into train and validation(20%) for model selection. The test dataset is not used here.** We can also determine which model to select using test accuracy. Because of time constrain I conducted only one selection experiment. I have highlighted the models that I am going to use further in this study.

Note: I did not pick DenseNet201 because I had already picked similar architecture DenseNet169 with smaller number of parameters.

Table 1 The table shows number of parameters and obtained accuracy for each pretrained model.

Model Name	# Model Parameters(Sorted)	Validation Accuracy
MobileNetV3Small	1529968	0.0898
MobileNetV2	2257984	0.7696
MobileNet	3228864	0.7033
EfficientNetB0	4049571	0.0168
MobileNetV3Large	4226432	0.0898
NASNetMobile	4269716	0.3041

EfficientNetB1	6575239	0.0449
DenseNet121	7037504	0.7191
EfficientNetB2	7768569	0.0617
EfficientNetB3	10783535	0.0674
DenseNet169	12642880	0.8483
VGG16	14714688	0.5730
EfficientNetB4	17673823	0.0730
DenseNet201	18321984	0.8089
VGG19	20024384	0.5449
Xception	20861480	0.7065
InceptionV3	21802784	0.7078
ResNet50V2	23564800	0.7808
ResNet50	23587712	0.2191
EfficientNetB5	28513527	0.0393
EfficientNetB6	40960143	0.0449
ResNet101V2	42626560	0.7833
ResNet101	42658176	0.3426
InceptionResNetV2	54336736	0.7808
ResNet152V2	58331648	0.7921
ResNet152	58370944	0.3033
EfficientNetB7	64097687	0.0561

6.Data Augmentation Techniques

As discussed in Section 3.5, data augmentation techniques to a) get a bigger dataset for model training as we know that CNNs performs better when trained on larger datasets and b) bring train and test domains closer to each other.

6.1 Blurring: Add Gaussian noise to the images.

6.2 Rotation: Rotate the images in the range of 0-180.

6.3 Horizontal Flip: Flip the images horizontally.

6.4 Zoom: Zoom in or out in the range of [-20% - 20%].

5 fold data generated through above augmentation techniques. Augmented data is added to original training data for the training of the models.

7. Performance of selected networks on augmented data set

Epochs = 5, Batch Size = 32, Optimizer = Adam, Learning Rate = 0.01, Base Model trainable = False

Table 2 The table shows number of parameters and obtained accuracy on augmented+original training dataset.

Model	#Parameters	Test Accuracy
MobileNetV2	2257984	0.4380
DenseNet201	18321984	0.5047
ResNet50V2	23564800	0.5433

Resent50V2 gave the highest frequency. We will be using just ResNet50V2 in further experiments.

8. Hyper Parameter Tuning

Performance of network using different parameters is examined. Due to time and resource constraints, only a few selected parameters are tested. I reduced the learning rate with a hope of achieving better accuracy as larger learning rate for small datasets can cause skipping of minima. I increased the batch size for faster convergence. Another tuning method which is tried here is to train the last few layers(last convolution block in case of Resnet50V2) of selected model on the given dataset. Table shows the performance with parameter tuning. Early stopping is used to prevent overfitting.

Batch_size	**Learning Rate**	**Epochs**	**Model's last few layers Trainable?**	**Accuracy**
32	0.01	10	False	0.5809
32	0.001	100	False	0.6285
64	0.01	10	False	0.5714
32	0.01	10	True	0.5619

A quick peak into the set of images in the test dataset that were not recognized correctly: As observed, the model was not able to correctly recognize the images containing car brand names as text or images with noisy surroundings/backgrounds. To address the problem of images containing car brand names a to step methodology is proposed in the next section.

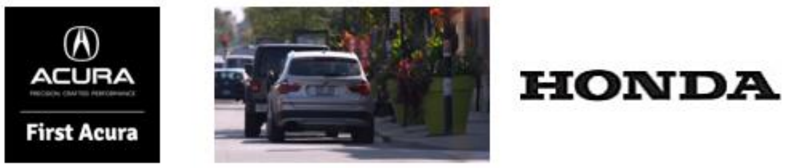


Figure 7 Example of test samples that were not recognized correctly

9. Car brand/logo detection using two step methodology: Text Detection and Logo Classification

In this framework two steps are performed as followed:

Step 1. Images are processed using segmentation method as mentioned in Section 4.3. These are then passed through pretrained Python Tesseract to determine if the image contains brand names from the given classes. If a particular class name is present in the image a probability of 1 is assigned to that class and 0 is assigned to rest of the classes.

Step 2. Images are processed as per section 4.1 and 4.2. These images are then passed to a pretrained Resnet50V2 with task specific fully connected layers. The predicted probabilities are then obtained

The final probabilities are obtained using argmax of sum of probabilities from Step 1 and Step 2. The final accuracy is calculated based on the final predicted probabilities obtained in this manner.

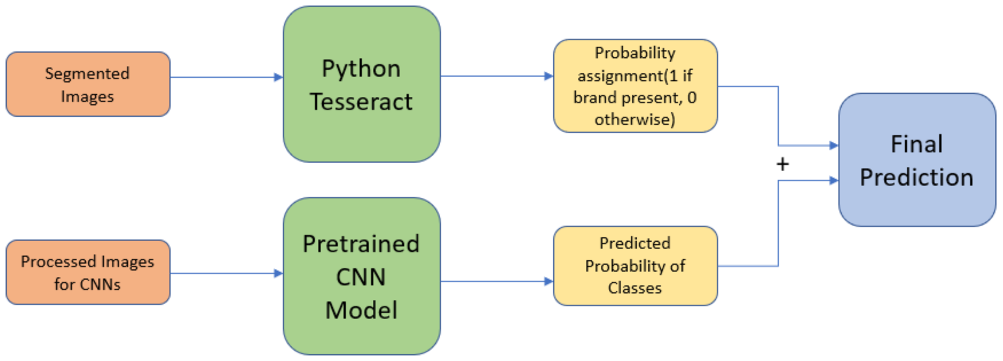

Segmented Images
Python Tesseract
Probability assignment(1 if brand present, 0 otherwise)
+
Final Prediction
Processed Images for CNNs
Pretrained CNN Model
Predicted Probability of Classes

Figure 8: Framework of the two-step methodology

10. Performance of Selected Networks using two step methodology

The accuracy of 70.47% is obtained through the two-step method. The table below shows the parameters used for training.

Table 3 Accuracy obtained using two step method with given parameters.

Learning Rate	Batch Size	Epochs	Accuracy
0.01	32	15	0.7047

A sneak peak into the kind of samples's brand name that were not predicted correctly.

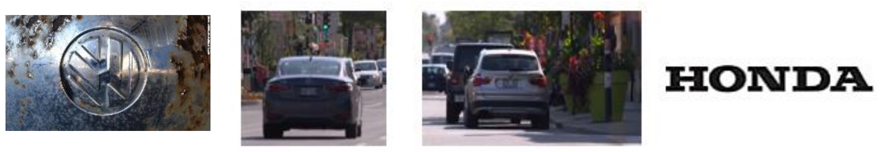


Figure 9 Sample Images from test dataset that were not correctly classified

11. Challenges and possible solutions:

1. **Challenge:** Python Tesseract could not recognize all kinds of text in given images.

   Possible Solutions:

   a) Improving the performance of tesseract by training it on a similar dataset containing fonts used in brand names.

b) Different segmentation methods or other preprocessing methods for preparing the images for tesseract.

2. **Challenge:** Most of the images in third category as per section 3.4 were not predicted correctly.

   Possible Solutions:

   a) Selective Search can be used to localize the brand logos as another step. We can use networks like YOLO, RCNN for localization of objects and train them on a dataset containing bounding boxes (ex. Flickr Logo dataset) in supervised setting. The challenge that can occur in this solution is the domain discrepancy as we do not have a car logo localization dataset containing bounding boxes. The model may not give expected results without proper training.

12. Other Observations and ideas:

a) Further hyper parameter tuning can improve the performance.
b) Studies on multiple data augmentation techniques, can provide insights on resolving the domain discrepancy between test and train dataset.
c) Similar networks as in Table 2 but improved versions(with higher number of parameters) can be used to extract complex features.

Bonus question: If you are a team lead to construct an end-to-end project (logo detection and recognition), what experiments would you design, what are the criteria to do a model selection?

Experiments:

1. Further image data exploration to get more insights into the dataset if possible.
2. Develop a model to recognize text from the images or how can we improve the use of Python tesseract.
3. Develop a model to detect logos in the images containing noise or multiple objects. Pretrained models like YOLOv3, Faster RCNN can be used. An issue could be the use of dataset for training the model as the Coco dataset on which these models are trained does not contain logos of the cars that are present in our dataset.
4. Study different data augmentation techniques and analyze which are most suitable for our task.

Criteria to do Model Selection:

1. Consider the constraints if any(time, resources, data availability)
2. Select a model that can work well with the test data.
3. Consider image preprocessing required for such models. See if the preprocessing step is causing any loss of information.

References

1. Python Tesseract[Source Code]
2. Keras Pretrained Models(Applications)[Source Code]