Machine Learning - A2

Faith Ha s3890479 & Chau Le s3928470

Loading Dataset:

- Read in data by unzipping the traffic sign and test dataset and loading it using the OS module
- Found the subfolders of each category by iterating over each file using the OS module
- Created a dataframe by iterating over main subfolders and sub-subfolders then appending the image name and label to the list
    - Created dataframe from list
- Expanded dataframe by splitting the label into shape and type

EDA of traffic sign dataset:

- Split the label column into shape and type and found the number of images for each category
    - Significant class imbalance

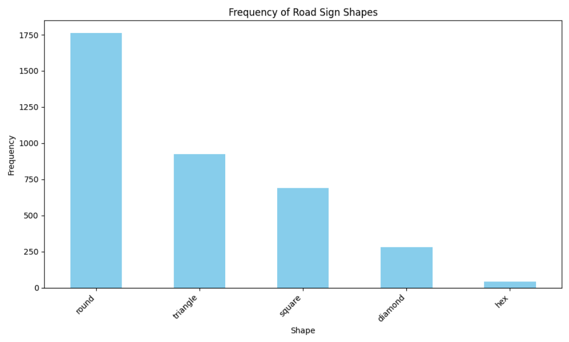


Figure 1: Number of images for Shape

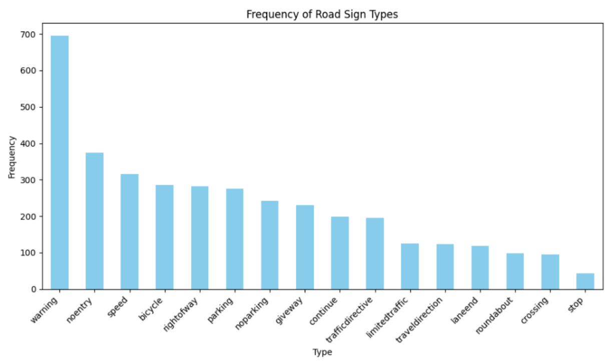


Figure 2: Number of images for Type

- Almost 700 images in the warning category with label count decreasing
- Random images were selected using the random image function, which was set to a sample size of 4.

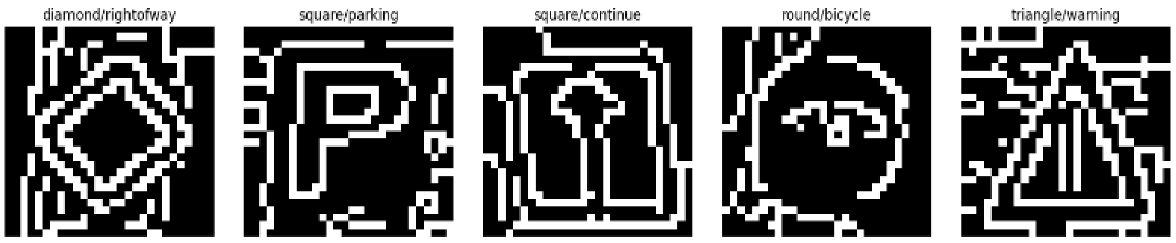
Data Preprocessing:

- The shape and type columns were encoded by sorting shape and type and assigning numerical values to each label, then adding the columns to the dataframe
- These labels were then added to the dataframe

Edge Detection:

- Edge detection is when edges are identified using gradient-based algorithms, which exploit the gradient of the image's intensity (Edge Detection Definition | Encord, 2023).

- We used the canny edge detection algorithm that combines gradient-based and non-maximum suppression techniques to produce a clean and accurate edge map.
    - First uses Gaussian blur to help remove noise.
- Initial hope is that it cleans irrelevant noise and information, while preserving important structural properties of an image
    - Sample shows that while edge detection can identify triangles and diamonds more cleanly it lacks finesse when dealing with round shapes



-

Figure 3: Sample of edge detection results

Smote:

- Because of large class imbalance we decided to try smote.
- Smote oversamples the minority class by synthetically adding new images to the minority class, helping to reduce class imbalance (Brownlee, 2020).
- However, although this approach helps mitigate class imbalance for sign types, it creates an even bigger class imbalance for shape sign types, as there are a different number of subclasses for each shape label.

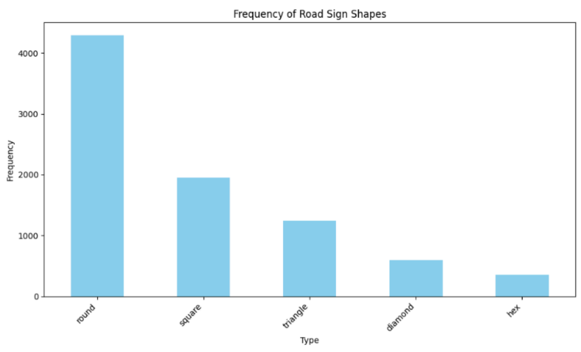


Figure 4: Smote Class Size Shape

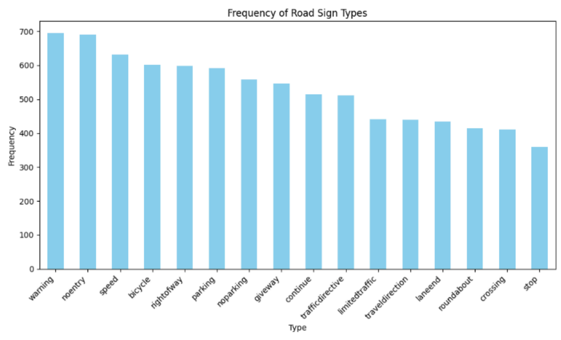


Figure 5: Smote Class Size Type

Testset:

- The test set was created using the European Traffic Sign dataset from Belgium, Croatia, France, Germany, The Netherlands, and Sweden.
- Using the class folder provided with the dataset we were able to distinguish which classes matched the Belgium dataset classes.
- 50 images for each category were then converted to .png files, greyscaled, and downsampled to 28 x 28 pixels to match the belgium dataset.
    - We chose 50 as we wanted to approximately match the size of the validation dataset which was 740 images.
    - The continue label only had 35 test images, but is 17.5% of the train dataset continue label and so considered adequate.

- As there might be overlapping images with the train dataset, the test images were then compared to the Belgium dataset by computing a unique hash key for both the test set and belgium dataset.
    - Images were compared, but no duplicates were found.
- Next, random traffic signs pictures taken from around melbourne were added to the testset.
    - Then greyscaled and downsized to 28 x 28 pixels.
    - There are some legal and ethical concerns:
        - There might be intellectual property rights for some of the signs we took pictures off
        - Introducing melbourne image signs might add some bias into the model's evaluation
- Total images in test set is 796

Splitting the Dataset:

- The dataset was split 80% train and 20% validation

Justification of accuracy metrics used:

- Accuracy could not be used as the classes are so imbalanced, a model can achieve high accuracy simply by predicting the majority class most of the time.
- Models were then trained on the f1 score as it combines precision and recall into a single value providing a balanced measure of a classifier's performance.
    - Useful metric when you have imbalance data as it takes into account the type of errors - false positive and false negative
        - Not just the number of predictions that were incorrect
- Other Supporting Metrics:
    - Confusion Matrix
        - Allows us to visualise the performance of the classification model based on different categories
    - Precision
        - Metric that measures how often a machine learning model correctly predicts the positive class.
        - Works well to identity imbalance classes since it shows the model correctness in identifying the target class
        - Useful when cost of false positives are high (Accuracy vs. Precision vs. Recall in Machine Learning: What's the Difference?, n.d.)
    - Recall
        - The ability of a model to find all the relevant cases within a data set
        - Focuses on capturing all positive instances and minimising false negatives

Model 1: CNN model for Shape classification

Table of Results (Shape):

- Note: the train, val f1-scores, train loss, and val loss are taken from the last epoch's results

	Train F1	Val F1	Weighted F1	Train Loss	Val loss	Weighted Precision	Weighted Recall
Baseline CNN	.4825	.4939	.31	4.4721	4.1820	.23	.48
Smote CNN	.2081	.1465	.04	7.9441	12.6124	.02	.015
Edge CNN	.4001	.3984	0.01	12.9097	13.0252	0.00	0.07
Modified Baseline CNN	.9993	.9889	.99	.0031	.0333	.99	.99
Modified CNN Regularised 1	.9973	.9941	.99	.0166	.0157	.99	.99
Modified CNN Regularised 2	.9707	.8349	.90	1.1961	1.6074	.84	.84
Modified CNN Augmentation	0.00	.9383	0.01	4.3162	1.9877	0.01	0.07
Modified CNN Tuned Learning Rate	.9965	.9987	.99	.02725	.2529	.99	.99

Table 1: Model Results for Shape

Baseline Shape Model:

- Model was trained using the tensorflow module which is a conventional cnn model
- Last epoch f1-score was .4825 for Train and .4939 for val
- The weighted f1 score was .31
- Precision and Recall scores of .23 and .48
- All scores were misleading as the model predicted that every image was from the largest label class round.

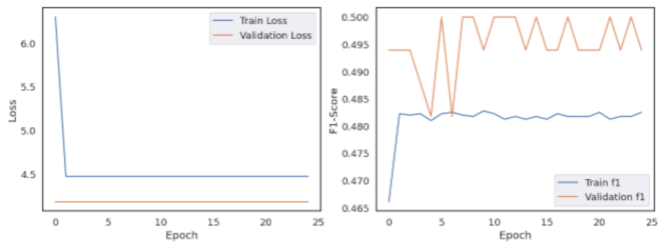


Figure 6: Baseline Loss and F1-Score over Epochs

Smote Model:

- The model performed worse to the baseline choosing to predict all images as triangles.
    - It is noted that the class imbalance would be greater as there are a different number of subclasses for each type
- The weighted F1, precision, and recall scores dropped to below 1%.

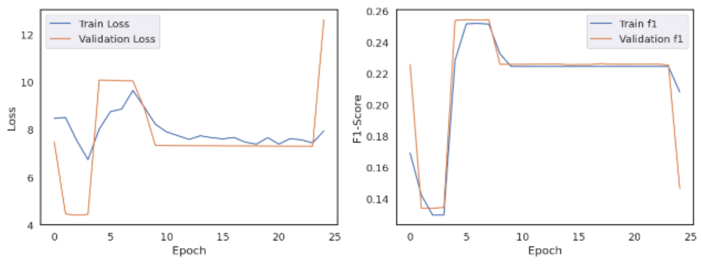


Figure 7: Smote Loss and F1-Score over Epochs

Edge Detection Model

- The model was trained using the edge detected dataset.
- Weighted precision, recall, and f1 were all lower than the baseline scores.
- Model chose to predict all images to diamond.

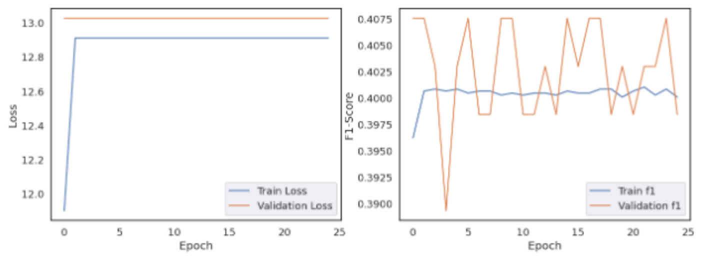


Figure 8: Edge Loss and F1-Score over Epochs

Modified Shape CNN:

- As the previous model's layers were not working, we tried a different arrangement of convolutional layers, activation functions, batch-normalisation layers, and max-pooling layers.
- Convolutional Layers:
    - Two convolutional layers with 16 filters each, kernel size (3, 3), and 'same' padding.
    - ReLU activation functions applied after each convolutional layer.
    - Batch normalization applied along the channel dimension after each activation.
- Max Pooling Layers:
    - Two max-pooling layers with pool size (2, 2) applied after every two convolutional layers.
- Flattening:
    - Flatten layer to convert the 2D output from the convolutional layers into a 1D feature vector.
- Dense Layers:
    - Two dense layers with 128 units each, followed by ReLU activation and batch normalization.

    - Dropout with a rate of 0.5 applied after each dense layer for regularization.
- Output Layer:
    - Dense layer with classes number of units, where classes represents the number of output classes.
- Results were much better, with a high train and validation f1 scores, low loss, and almost perfect precision and recall.
    - However, the train f1-score was slightly higher than the validation score leading to concerns of overfitting.
    - When looking over the graphs of accuracy and loss there were concerning high spikes occurring during the later half of training.

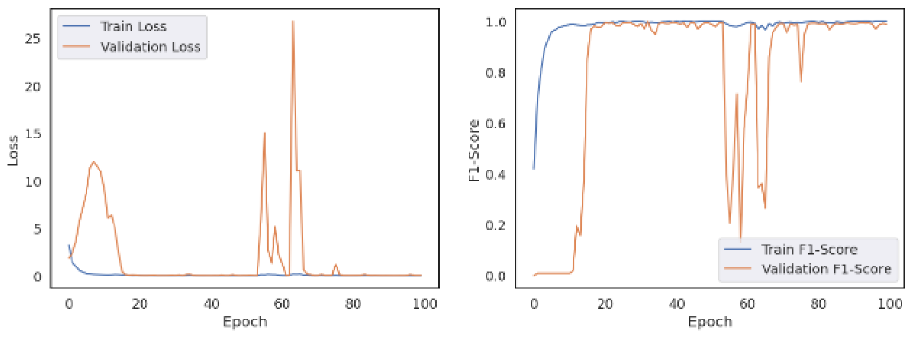


Figure 9: Loss and F1-Score over Epochs - Baseline

Modified Shape CNN with Regularisation:

- To prevent overfitting the model we used regularisation with a lambda value of .001.

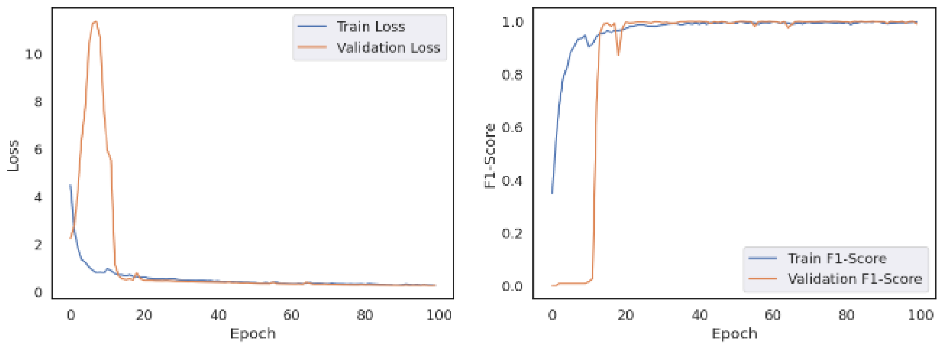


Figure 10: Loss and F1-Score over epochs - Regularised

- As seen from Figure 5, spikes were drastically lowered and training and validation scores were very similar
- Decision was made to tune the regularised parameters further:
    - Regularisation parameter was then tuned and best regularisation parameter found was found to be 0.1
    - The regularised model with the lambda value of 0.1 was then tested and performed worse with a lot of spikes and variations.
        - Cross Validation was applied to both models with lambda values of .001 and 0.1 to decide which model to use.
            - For model with lambda value of .001:
                - Mean F1 Score: 0.9918, Std Deviation: 0.0079
            - For model with lambda of 0.1:
                - Mean F1 Score: 0.2385, Std Deviation: 0.3498
        - Model with lambda of .001 was chosen to continue further analysis

Dynamic Lambda Learning Rate

- Next the lambda parameter was tuned with a dynamic learning rate schedule function to adjust the learning during training to find the most optimal value.
- Overall scored the highest evaluation metrics compared to the rest of the models.

Modified Shape CNN with Regularisation and Augmentation:

- Augmentation was used to enhance model generalisation
- Balance classes, prevent overfitting, and explore feature space
- However, based on the f1-score graph, the model failed to classify the images correctly, with large amounts of fluctuations occurring in the loss and accuracy graphs.
    - Very low weighted precision, recall and f1-scores

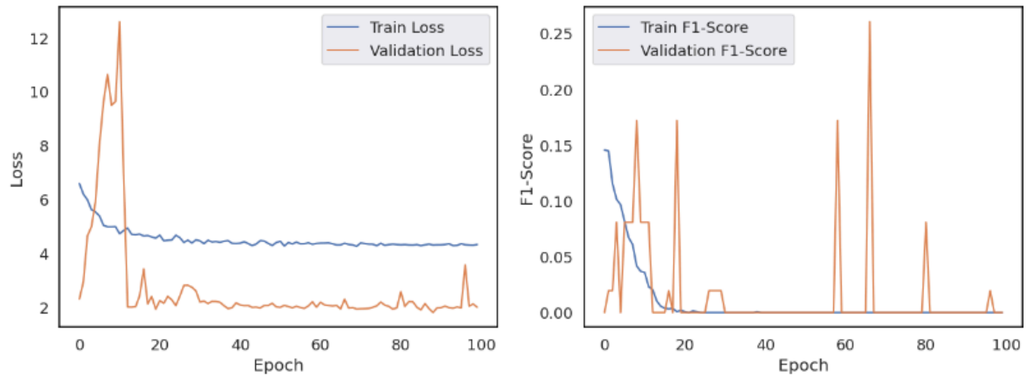


Figure 11: Augmentation loss and f1-Scores

Test Set Results:

- The regularised shape model of .001 with the dynamically tuned learning rate was chosen to use on the test set.
- Overall test results were quite high with a precision weighted average of .90 and recall of .89.
- Weighted f1-score was .89 and the model achieved an accuracy of .89 on test set

Model 2: MLP for Type classification

Baseline model for Type:

- MLP Neural Network
- One single layer with 126 neurons
- Activation mode: relu
- Optimiser: SGD
- Learning rate of optimiser: 0.01 (default)

Baseline model with SMOTE dataset:

- Use the SMOTE dataset to train the baseline model
- Evaluate the model using the validation set from SMOTE dataset

Hypertune number of neurons and learning rate:

- For all hyper tuning of this model, the objective is Maximum validation F1 score

- Tune the Single-layer baseline model
- Test the model with different numbers of neurons in the single hidden layer: 256, 400, 512
    - 256 and 512 are commonly used numbers
    - 400 because it’s the mean of input and output since most problems can be solved using a single-layer model with the number of neurons being the mean of input and output layers (Krishnan, 2021)
- Tested learning rate: 0.01 and 0.001,
- Tuning is performed using keras_tuner package
- Best model:
    - Number of neurons: 256
    - Learning rate: 0.01

MLP model with 2 hidden layers:

- Add an extra layer
- Parameter values to be tested:
    - Number of neurons in Layer 1: [256,400,512]
    - Number of neurons in Layer 2: [256,400,512]
    - Learning rate: [0.01, 0.001]
- Use keras tuner to find the best-fitting number of neurons in each layer and learning rate of the optimizer
- Best mode:
    - Hidden layer 1: 400 neurons
    - Hidden layer 2: 248 neurons
    - Learning rate: 0.01

2 Hidden Layers with regularisation

- Similar to the previous model, we tune the number of neurons and the lambda used for L2 regularisation of each layer
- Tested values: [0.01, 0.001]
- Type of regularisation: L2
- Best model:
    - Layer 1: 400 neurons, lambda = 0.001
    - Layer 2: 512 neurons, lambda = 0.01

Data Augmentation:

- Use the tuned single-layer model to be trained and evaluated as it is the best-performing model so far
- Methods:
    - Rotation_range = 90 -
    - Brightness_range = [0.8,1.3] - to help the model predict signs better under different lighting
    - Vertical flip = True

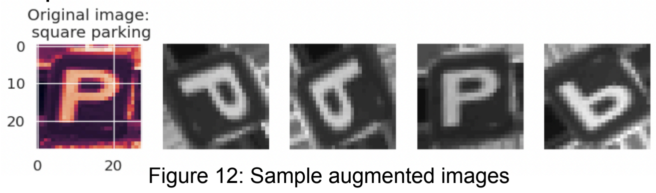


Figure 12: Sample augmented images

Comparing all the models using F1

	Train F1	Val F1
Baseline	0.411	0.415
Smote	0.322	0.311
Tuning single-layer model	0.428	0.413
2 Hidden Layers with hyperparameter tuning	0.402	0.357
2 Hidden Layers with regularisation	0.356	0.323
Data Augmentation	0.232	0.230

Table 2: Type Model Results

Conclusion:

- Model 2 does not perform well
- We chose not to proceed with it to classify the test set

Using Model 1 for Type Classification

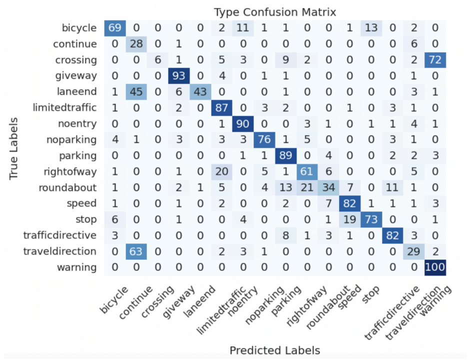
- Since Model 1 had a very good performance in classifying the traffic signs by their shapes, we will use it to classify types of traffic signs as well
- The model will be trained and tuned for Type, but with the same layers setting as the one used for shape classification.

Performance on Training Dataset

- Performance of last epoch:
    - Training F1: 0.995
    - Validation F1: 0.979
- The scores are high, but we do not want the model to overfit the training set then perform poorly on the Test Dataset
- We will regularise the 4 2D Convolation layers and use Keras tuner to tune the value of lambda
- Tested values of lambda: [0.01, 0.001]
- The best set of lambda values from the tuner (in order): [0.01, 0.001, 0.01, 0.001]
- Validation F1: 0.98

Performance on Test dataset

- Use the best model from the tuner to classify the test dataset:
- Weighted F1: 0.66
- Confusion matrix:



	bicycle	continue	crossing	giveway	laneend	limitedtraffic	noentry	noparking	parking	rightofway	roundabout	speed	stop	trafficdirective	traveldirection	warning
bicycle	69	0	0	0	0	2	11	1	1	0	0	1	13	0	2	0
continue	0	28	0	1	0	0	0	0	0	0	0	0	0	0	6	0
crossing	0	0	6	1	0	5	3	0	9	2	0	0	0	0	2	72
giveway	0	0	0	93	0	4	0	1	1	0	0	0	0	0	1	0
laneend	1	45	0	6	43	0	0	0	1	0	0	0	0	0	3	1
limitedtraffic	1	0	0	2	0	87	0	3	2	0	0	1	0	3	1	0
noentry	0	0	0	0	0	1	90	0	0	3	1	0	1	1	4	1
noparking	4	1	0	3	0	3	3	76	1	5	0	0	0	3	1	0
parking	0	0	0	0	0	0	1	1	89	0	4	0	0	2	2	3
rightofway	1	0	0	1	0	20	0	5	1	61	6	0	0	0	5	0
roundabout	1	0	0	2	1	5	0	4	13	21	34	7	0	11	1	0
speed	1	0	0	1	0	2	0	0	2	0	7	82	1	1	1	3
stop	6	0	0	1	0	0	4	0	0	0	1	19	73	0	0	1
trafficdirective	3	0	0	0	0	0	0	0	8	1	3	1	0	82	3	0
traveldirection	0	63	0	0	0	2	3	1	0	0	0	0	0	0	29	2
warning	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100

Figure 13: Correlation Matrix of the model's classification of Test Dataset

- Observations from the confusion matrix:
    - Many Travel Direction signs are classified as Continue signs. By visualizing some of the predictions, we determine a potential cause of this: Continue signs and Travel Direction signs that tell drivers to go ahead both have a straight arrow in the middle of the sign. The only difference is the shape of the sign.

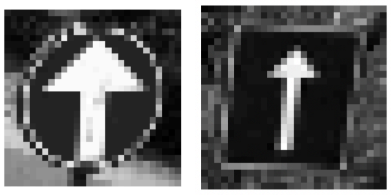


Figure 14: A Travel Direction sign vs a Continue sign

    - Many Crossing signs are classified as Warning signs. By inspecting the dataset, a potential reason is found to be that Crossing signs strongly resemble Warning signs due to both having big white triangles. Even tho the shape of Crossing signs is square, they have a white triangle in the middle of the sign that makes the model confuse them with Warning signs

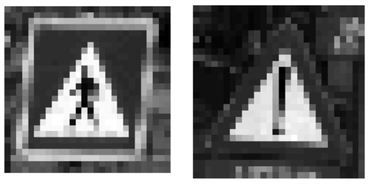


Figure 15: A Crossing sign vs a Warning sign

Ultimate Judgment and Conclusion:

- Our ultimate judgement in which model to be used on real-world problems is the Experimental CNN models
- In a real world context, the model should take consider both shape and type of a traffic sign to classify it to avoid confusion between signs that resemble each other
- In conclusion, we experimented with 2 models: CNN and MLP
- To deal with class imablance, we used techniques like SMOTE or Class Weight
- To refine the models, we used different methods such as Edge detection, Synthetic images (SMOTE), Hyperparameter Tuning, Regularisation, Modifying the configuration or layers of the baseline model.
- The models are evaluated and compared against each other using a set of metrics, including F1 score, Precision, and Recall

References:

Brownlee, J. (2021, March 17). *SMOTE for Imbalanced Classification with Python - MachineLearningMastery.com*. Machine Learning Mastery. Retrieved May 17, 2024, from https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

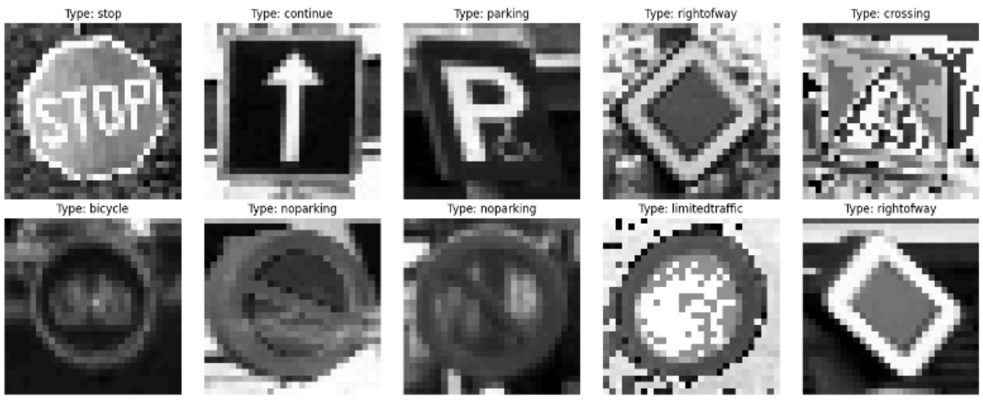
Encord. (n.d.). *Edge Detection Definition*. Encord. Retrieved May 18, 2024, from https://encord.com/glossary/edge-detection-definition/

Evidently AI. (n.d.). *Accuracy vs. precision vs. recall in machine learning: what's the difference?* Evidently AI. Retrieved May 15, 2024, from https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall

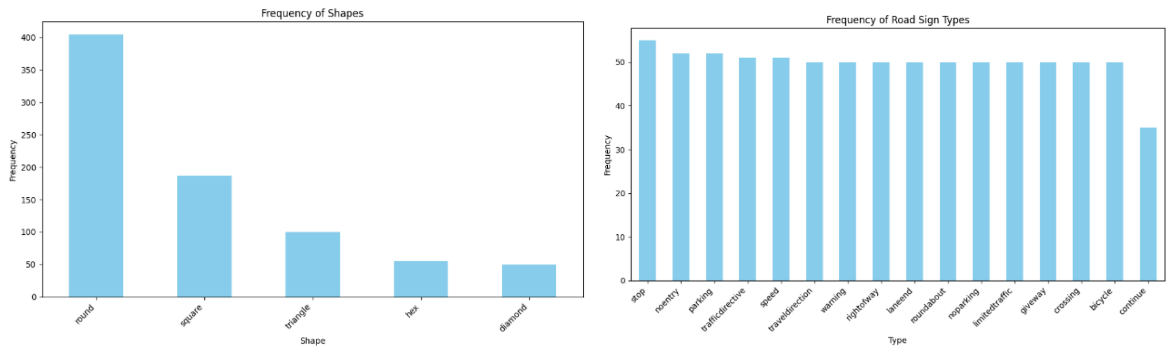
Krishnan, S. (2021, September 9). *How do determine the number of layers and neurons in the hidden layer?* Medium. Retrieved May 18, 2024, from https://medium.com/geekculture/introduction-to-neural-network-2f8b8221fbd3

O'Malley, T., Bursztein, E., Long, J., Chollet, F., Jin, H., Invernizzi, L., & Others. (2019). *KerasTuner*. Retrieved from https://github.com/keras-team/keras-tuner

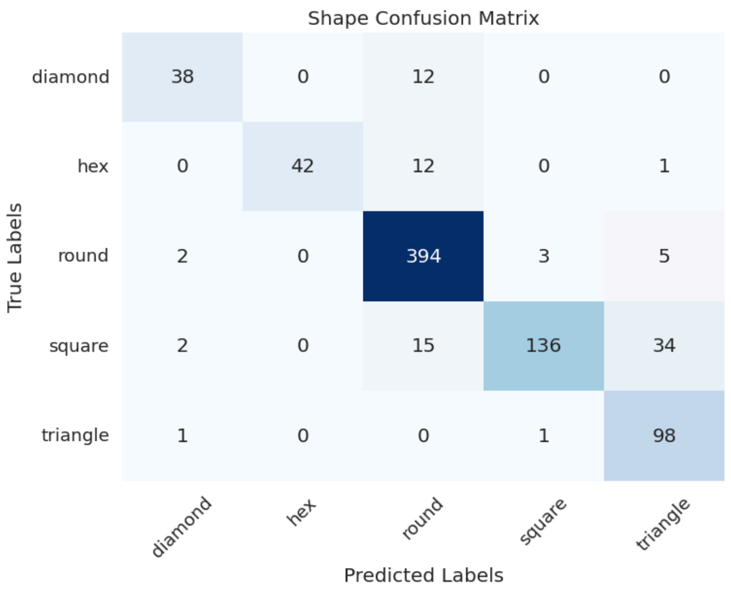
Appendix:



Smote random sample of images

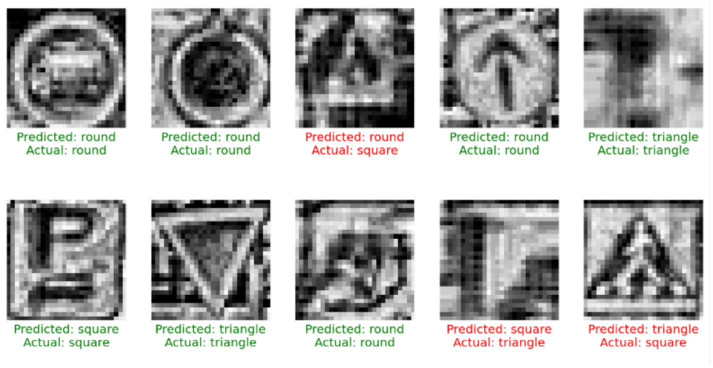


Bar Chart of number of Test Images for Shape and Size



	diamond	hex	round	square	triangle
diamond	38	0	12	0	0
hex	0	42	12	0	1
round	2	0	394	3	5
square	2	0	15	136	34
triangle	1	0	0	1	98

Test Set Confusion Matrix - Shape



Sample of Test Image Predictions - Shape