ASSIGNMENT 2

COSC2673
MACHINE LEARNING
UNDERGRADUATE

By
Oisin Aeonn (s3952320)
&
Vince Quach (s3900481)

Table of Contents

Introduction

For *Assignment 2* in **COSC2673 Machine Learning** at **RMIT University**, we selected '**Project 1: Classify Images of Road Traffic Signs**'. The primary goal of this project was to develop an end-to-end machine learning algorithm capable of performing an n-ary classification of a road sign's shape and type. We used the provided modified version of the **BelgiumTS Dataset** for training and validation. Additionally, we aimed to create a robust and generalisable model that can accurately classify road signs from an independently sourced dataset, ensuring its effectiveness in a real-world scenario.

Road sign recognition plays a crucial role in intelligent transportation systems and autonomous vehicles, enhancing overall road safety. By employing advanced machine learning techniques, including deep learning architectures such as Convolutional Neural Networks (CNNs), and leveraging insights from domain experts, we developed a reliable road sign classification model that could be integrated into these systems. In the subsequent sections of this report, we will detail our approach, present our ultimate judgement, discuss the challenges encountered and insights gained throughout the project.

General Approach

Our approach to develop a robust and generalisable road sign classifier followed the comprehensive Machine Learning Workflow methodology (Datacamp, 2022 & Alavi et al. RMIT 2024), with some key elements borrowed from Cross-Industry Standard Process for Data Mining (CRISP-DM) (RunAI, 2024).

As seen in the **Introduction**, we began by defining the problem of road sign classification and its significance in the context of intelligent transportation systems. Next, we conducted an extensive Exploratory Data Analysis (EDA), outlier handling, labelling of the dataset, before pre-processing the dataset. We could finally then split the dataset into training and validation sets. Additionally, we utilised Linear Discriminant Analysis (LDA) to explore the relationships between different road signs, as suggested in the original 2013 paper by Mathias et al. [pp. 4-8] on traffic sign recognition. As well as utilised a few unsupervised techniques such as K-Means to extrapolate, and cluster images together to extract relevant topics.

We investigated various machine learning algorithms and justified our selection based on their suitability for image classification. The selected models were optimised using dimensionality reduction and hyperparameter tuning techniques. Our hyperparameter tuning involved several approaches, including manual tuning, Bayesian optimisation, and grid search. Primarily, we tuned regularisation parameters (L1 and L2) to prevent overfitting, batch size for optimal performance, adjusting tree and neural network architectures, dropout rates to encourage robust feature learning, transfer learning for improved generalisation, and early stopping to mitigate overfitting. Model evaluation was performed using the appropriate classification metrics: F1-score, accuracy, precision, and recall. These metrics enabled us to assess model performance on our validation set, identifying the strengths and weaknesses of each model and informing our final model selection.

Finally, for our independent evaluation, which simulates real-world implementation, we rigorously collected an independent dataset using the scientific method. This dataset consisted of road sign images from various countries, to assess the generalizability and performance of our trained models in practical scenarios. By applying our models to this unseen data, we aimed to evaluate their effectiveness and robustness in accurately classifying road signs across diverse geographical locations and imaging conditions.

By following this machine learning workflow methodology, we developed a robust and accurate road sign classification model, emphasising the iterative nature of model development and continuous refinement to achieve the best possible performance and generalisation capability. We will now go into depth explaining the rationale behind each step.

Exploratory Data Analysis (EDA)

We began the EDA phase by examining the dataset, which consists of 3,699 images categorised using a folder hierarchy. To understand the dataset, we plotted and made observations of the following key features:

- ***Sharpness: The distribution is positively skewed, indicating most images have lower sharpness.***
- ***Pixel Intensity and Image Similarity: Both follow a roughly Gaussian distribution.***
- ***Entropy: The distribution is negatively skewed, with most images having higher entropy.***
- ***Image Size and Format: All images are consistently 28x28 pixels, grey scale, and in .PNG format.***

The image similarity analysis demonstrated significant diversity within the dataset, suggesting that it has already undergone pre-augmentation. This solidified the recommendation not to augment the dataset further, as it has been carefully curated and pre-augmented for our task. No images in the dataset are exactly equal to another, but there are some that are very similar, as indicated by the similarity score.

This initial EDA provided valuable insights into the dataset's characteristics and diversity, guiding our subsequent steps in the machine learning workflow. In the next step, we will go further into detail of how we ingested the data and further explore the variance in the labelling.

Handling Low Quality Data

In the dataset there are examples of road signs that are obstructed or irregular (see Figure 1), creating edge cases for certain classes. We removed this low quality data to attempt to reduce bias and maintain consistency across different sign types and shapes. Including challenging examples is often beneficial, but inconsistencies across classes can create bias, leading the model to potentially over predict a class that has an edge case when given an abnormal input leading to a misclassification. For example in the Figure 1 case any blurry input may be classified as a Hexagon Stop Sign. Removing edge cases from certain categories during training could make the model struggle with similar cases during evaluation. To balance data quality and consistency, we removed identified edge cases, benchmarking the performance on the original and cleaned datasets, and verified performance improvement while maintaining parity across all traffic sign categories.

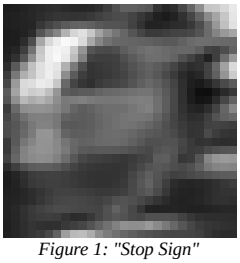

Figure 1: "Stop Sign"

Data Ingestion and Pre-processing

After conducting the initial EDA, we proceeded to the data ingestion and pre-processing stage. We noticed that the image names themselves were insignificant for our task, and the crucial information was contained in the directory structure where the images were stored. The directory hierarchy represented the labels for each image, with the first level indicating the 5 shapes and the second level indicating the 17 types of road sign.

To prepare the data for a supervised machine learning model, we needed to create a data structure that explicitly captures the true labels for each image. We opted to create a Pandas DataFrame containing the following three lists:

1. ***image_paths: Stores the relative path, allowing access to the pixel data.***
2. ***shapes: Stores the corresponding shape label for each image.***
3. ***types: Stores the corresponding type label for each image.***

We implemented a custom data labelling function to iterate through the dataset directory and populate these lists. The function traversed the directory structure, extracting the image paths and their corresponding shape and type labels.

To convert the categorical labels into numerical representations suitable for our model, we utilised the LabelEncoder from the sklearn.preprocessing module. We encoded the shape and type labels separately, ensuring that each unique label was assigned a numerical value.

To ensure the integrity of our dataset labelling process, we performed a verification step. We confirmed that the entire training dataset was successfully labelled, satisfying the following condition: $\forall$ DataFrame $\exists$ Label (image_path) $\wedge$ Label (shape) $\wedge$ Label (type).

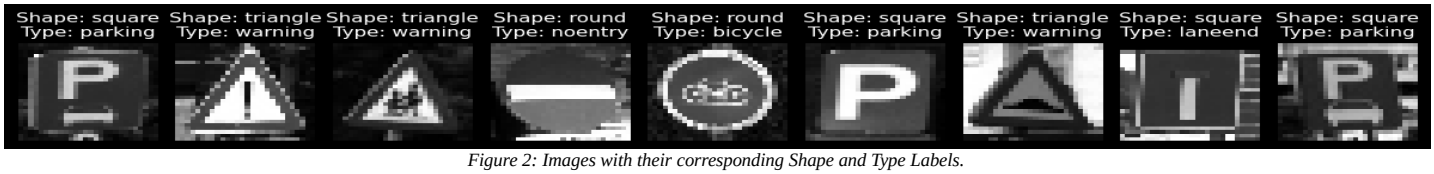

Figure 2: Images with their corresponding Shape and Type Labels.

By completing the data ingestion and pre-processing stage, we transformed the raw dataset into a structured format suitable for training and evaluating a supervised machine learning model as requested in the brief. Upon further exploration of the DataFrame (refer to Figure 2), we can observe that all images are classified correctly.

EDA (Part 2)

Now that we have a labelled DataFrame, we can do some further EDA on our dataset to identify any class imbalances and make further observations. Upon analysing the class distribution within the dataset, we observe a significant class imbalance across both shape and type categories as seen in Figure 3. Some classes have a considerably higher number of samples compared to others, which can potentially lead to biased predictions if not addressed properly. To address the class imbalance issue, we will test techniques like:

- ***Class Weighting: Assigning higher importance to minority classes during training.***
- ***Oversampling: Increasing the number of minority class samples through duplication or synthetic data generation.***
- ***Undersampling: Reducing the number of majority class samples to match the minority classes.***

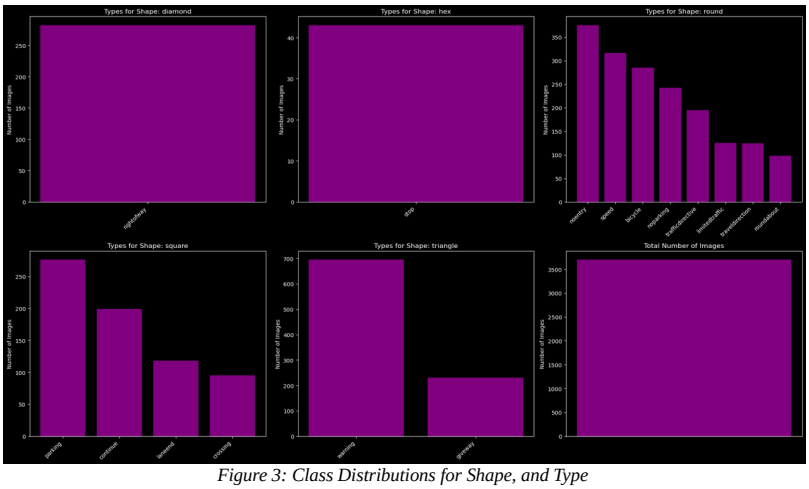

Figure 3: Class Distributions for Shape, and Type

Data Splitting and Leak Checking

We split the dataset into training and validation sets using an 80/20 ratio to ensure reliable evaluation of our model's performance. It's crucial to check for data leakage between the sets, as it can lead to overly optimistic performance estimates.

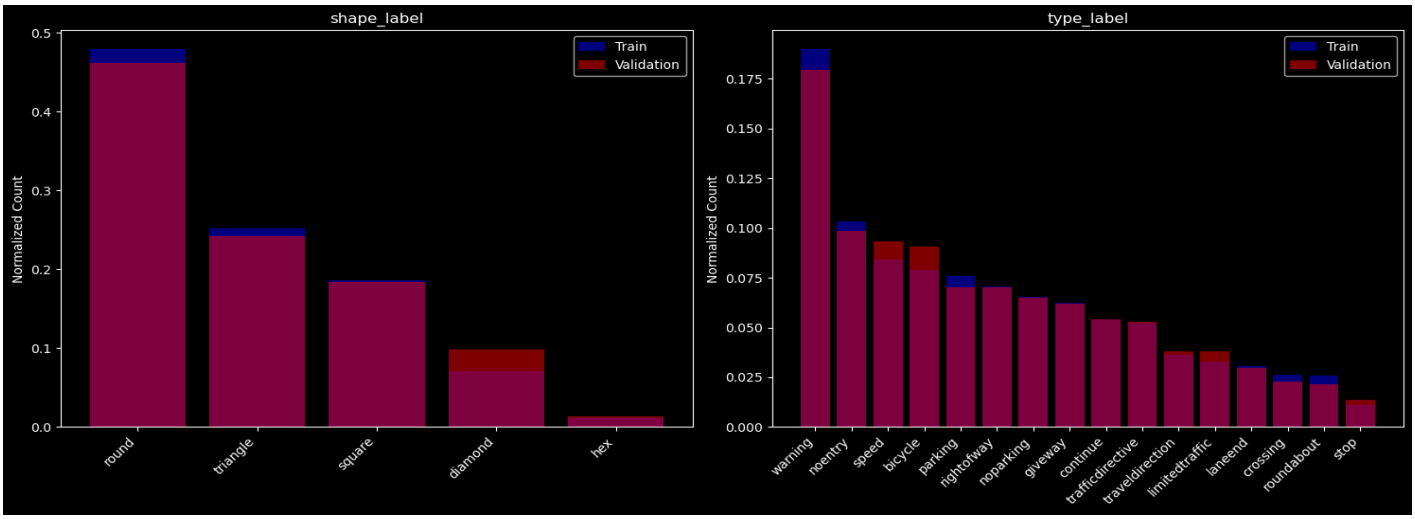

Figure 4: Training and Validation Data Leakage

Figure 4 illustrates the normalised distribution of samples, where purple indicates overlap, blue represents more training data, and red represents more validation data. The analysis shows minimal data leakage, with small discrepancies considered acceptable.

Dimensionality Reduction with Principal Component Analysis (PCA)

PCA is explored as an optional step to reduce the dimensionality of the high-dimensional image data. By extracting essential features and discarding irrelevant information, PCA aims to improve the predictability and efficiency of our traffic sign classification model. Experimenting with different numbers of principal components and evaluating their impact on model performance helps determine the optimal level of dimensionality reduction for our specific task. We predict this will make a small impact on performance, and is worth implementing.

Fixing Class Imbalances

Due to the relatively small size of our dataset and the presence of significantly underrepresented classes, such as the Hexagon Stop Sign, which constitutes only around 1% of the total samples, undersampling is not feasible as it would result in insufficient data for generalisation. Therefore, to address class imbalance, we are left with two optional steps in our notebook: class weights and oversampling. Class weights involve assigning higher importance to minority classes during model training, allowing them to have a greater influence on the model's learning process. On the other hand, oversampling techniques aim to increase the number of samples in underrepresented classes to achieve a more balanced class distribution. However, it's important to be aware of the potential drawbacks associated with these methods. Oversampling, in particular, can lead to overfitting if not used carefully, as the model may learn to memorise the duplicated samples instead of generalising well to unseen data. Additionally, these techniques assume that the class distribution follows a Gaussian distribution in the real world, which may not always be the case. To make an informed decision on whether to incorporate oversampling or class weights into our final model, we will compare the performance of models trained with and without these techniques.

Evaluation Metrics

Before diving into our model development, it's crucial to establish a clear understanding of the metrics used to evaluate a model's performance. We will primarily focus on utilising confusion matrices, which provide a comprehensive view of the model's performance by presenting the counts of true positive, true negative, false positive, and false negative predictions for each class. The major benefit of confusion matrices is the ability to quickly glance at them and look for a diagonal trend from top left to bottom right to see the correct predictions. This allows us to calculate various evaluation metrics such as accuracy, precision, recall, and F1-score.

In addition to confusion matrix-based metrics, we will also consider other evaluation metrics such as the training error, loss function, and number of epochs. Monitoring the loss over epochs helps us assess the model's learning progress and identify potential issues like overfitting or underfitting.

Primarily we will focus on accuracy as it is the overall correctness of all predictions. As optimists, we put our target accuracy high aiming for 90% on shape, and 80% for type in our validation set. We predicted that when using an unseen dataset for our independent evaluation that performance would go down by 10-15% as we would have a fair amount of variance introduced with road signs from different countries. For further details on evaluating classification models, please refer to the **COSC2673 Machine Learning** Lecture 1 Discussion Forum Post by Me (**Oisin**).

Non-Neural Network Models

Before establishing a baseline model, we explored several non-neural network algorithms to ensure we didn't overlook a simple solution for road sign classification. These models, such as logistic regression and tree-based algorithms (Decision Trees, Random Forests, and XGBoost), are generally more simple, interpretable, and less computationally expensive compared to neural networks. They provide insights into the relationship between input features and the target variable, making them valuable in the initial stages of model development.

Evaluating the performance of these models on our specific classification task allowed us to assess their suitability and determine if they could achieve satisfactory results without the need for more complex architectures like neural networks. Some models, for example logistic regression, can effectively handle high-dimensional data like that which is present our image dataset without extensive feature engineering or dimensionality reduction techniques.

On the other hand tree-based ensemble methods, such as Random Forests and XGBoost, leverage multiple weak learners to create a strong and robust classifier, often outperforming individual models. By exploring non-neural network models first, we gained valuable insights and made informed decisions about the need for more advanced techniques as well as what performance metrics we should minimally get given the computational cost.

Hyperparameter Tuning

To find the optimal hyperparameters for each model, we employed a grid search, which is a brute-force method that exhaustively searches through a specified parameter grid cross validating the results to find the optimal combination. We also tried a Bayesian search, but found we got the optimal results using a grid search using the default of 5 for cross validation. The hyperparameters tuned for each model, and their associated best combination for shape, and type were:

- ***Decision Tree: criterion=entropy, max_depth=none, min_samples_leaf=1, min_samples_split=5***
- ***Random Forest: max_depth=20, min_samples_leaf=1, min_samples_split=2, n_estimators=200***
- ***XGBoost: subsample=0.1, n_estimators=5, max_features=sqrt, max_depth=100, learning_rate=X***

Where X is 0.8 for shape, or 1.0 for type.

The rationale for the selected hyperparameters to tune, and the chosen parameter grid was based on a few key considerations. We aimed to keep the number of grid parameters between 2 and 4 values, with a preference for 3, to strike a balance between computational efficiency and thorough exploration of the hyperparameter space. For more computationally expensive models like XGBoost, we opted for 2 values to manage the training time. The parameter inputs were varied to cover a spectrum from default settings to more aggressive configurations, allowing us to find the optimal trade-off between model complexity and generalisation performance. In the case of the simpler Decision Tree model, we were able to explore a wider range of inputs within a reasonable amount of time, enabling a more comprehensive search for the best hyperparameter combination.

Results

The validation results in Figure 5 show that Gradient Boosting, Random Forest, and Logistic Regression achieve very high accuracy scores all above 95% for both shape and type classification tasks, while Decision Trees exhibit a relatively lower accuracy of ~85% compared to the other models which is explainable by the ensemble combination of weak learners being optimal for tasks dealing with images. However, these high validation scores are difficult to take seriously as our model includes highly augmented images that could contribute to our model just remembering answers, verses being generalisable.

When applying PCA we don’t find a significant difference on our validation dataset, but we predict this will improve when applying our model to the independent dataset.

Model	Shape Accuracy	Type Accuracy
Gradient Boosting	0.96	0.95
Random Forest	0.95	0.96
Decision Tree	0.86	0.84
Logistic Regression	0.95	0.96

Figure 5: Non-Neural Network Validation Results

Neural Networks

After exploring various non-neural network models and evaluating their validation performance on our road sign classification task, we proceeded to investigate neural network architectures. Neural Networks are an obvious next step to solving our problem as they are often praised to be able to solve any problem being able to "approximate any function to any desired degree of accuracy" (**George Cybenko 1991**). Given the success of neural networks, particularly CNNs, in image classification tasks, we decided to that we would leverage their ability to learn hierarchical representations and capture complex patterns in visual data. However, we'll start off small, and simple with our baseline model using an MLP instead.

Baseline Model: Multi-Layer Perceptron (MLP)

An MLP is a type of feedforward neural network that can learn to classify images by automatically extracting relevant features from the input data. MLPs consist of multiple layers of interconnected neurons that transform the input image through a series of non-linear operations, enabling them to learn complex decision boundaries and capture intricate patterns.

Layers

Optimisation

When we were optimising our neural network models, we employed various techniques to improve performance and prevent overfitting. Here are some of the key optimisation techniques and evaluation metrics we utilized:

We monitored the loss function during training to assess how well the model is learning and converging.

Tracking the loss over epochs, we gained insights into the model's learning progress and identified potential issues like plateauing or divergence.

We experimented with increasing the number of neurons in the hidden layers of our neural networks. Adding more neurons can enhance the model's capacity to learn complex patterns and capture intricate relationships in the data. However, we were cautious not to add too many neurons, as it can lead to overfitting and increased computational complexity.

We implemented early stopping to prevent overfitting and find the optimal point to stop training.

By monitoring the validation loss or accuracy, we determined when the model's performance on unseen data started to degrade.

Early stopping helped us strike a balance between model complexity and generalisation ability, avoiding unnecessary training iterations.

We applied regularisation techniques, such as L1 and L2 regularisation, to reduce overfitting and improve model generalisation.

As demonstrated in Assignment 1: L1 regularization (Lasso) adds a penalty term proportional to the absolute value of the weights, encouraging sparsity and feature selection.

L2 regularization (Ridge) adds a penalty term proportional to the square of the weights, keeping the weights small and preventing individual features from dominating.

We performed manual hyperparameter tuning to find the optimal combination of hyperparameters for our neural network models.

Hyperparameters such as learning rate, batch size, number of layers, and number of neurons in each layer were adjusted iteratively.

Due to the computational expense of training neural networks, we opted for manual tuning instead of using automated methods like Bayesian optimization or grid search.

We incorporated dropout layers in our neural networks to reduce overfitting and improve generalization.

Dropout randomly sets a fraction of the input units to 0 during training, preventing the network from relying too heavily on specific features.

By applying dropout, we forced the network to learn more robust and distributed representations of the data.

We experimented with different batch sizes during training to find the optimal balance between computational efficiency and model performance.

Larger batch sizes can lead to faster training but may result in less precise weight updates, while smaller batch sizes can provide more accurate updates but may take longer to train.

We also increased the number of epochs to allow the model to learn from the data for a longer duration, especially when dealing with complex patterns or large datasets.

By employing these optimization techniques and monitoring the relevant evaluation metrics, we aimed to improve the performance and generalization ability of our neural network models. Manual hyperparameter tuning allowed us to efficiently explore the parameter space and find the best configurations for our specific classification task, given the computational constraints of our hardware setup.

Incremental Improvement

Advanced Model: Convolutional Neural Networks (CNNs)

CNNs on the other hand are more powerful than MLPs. They have emerged as an efficient architecture for image classification tasks. CNNs introduce convolutional layers that apply learned filters to the input image, enabling them to capture local spatial patterns and hierarchical features while being invariant to translations and distortions. These features of CNNs make them particularly well-suited for road sign classification, as they can effectively handle the variations in shape, orientation, and appearance of traffic signs. Thus, why we opted, and knew this would be the best methodology for the task.

Layers

Incremental Improvement

Transfer Learning

Independent Evaluation

To collect our independent evaluation dataset, we followed a rigorous scientific method to ensure the data's quality and diversity. We sought insights from a domain expert, **Professor Simon Jones**, **School of Science** at **RMIT University**. **Professor Jones** has extensive experience in collecting data and classifying images for various applications, including fire detection and prediction using advanced machine learning techniques like Random Forests, ensemble methods, and U-Net CNNs.

Based on the guidance from our domain expert, as well as several scientific journals we researched we focused on gathering data from various sources to create a diverse and representative dataset. We collected 511 road sign images from the **German Traffic Sign Recognition Benchmark (GTSRB)** dataset, which is a sister dataset to the **BelgiumTS** dataset used for training our model. Additionally, we captured 68 Greater Melbourne Region road sign images using our mobile phones: an iPhone 12 (used by Oisin) and an iPhone 13 (used by Vince). See Tom’s Guide for information on Camera difference. We also utilised Google Maps to source 106 Belgium Road Signs to test the dataset against unseen data from the same country. In total, we created a dataset of 689 images for our Independent Evaluation which was comparable to the Validation Dataset which equates to 20% of the size of the training dataset which is what was recommended by **Professor Alavi.**

During the data collection process, we prioritised capturing images with varying angles, brightness levels, and sharpness/blurriness. We also as previously mentioned we’re using different devices, which further simulated real-world scenarios. This approach helped us create a dataset that accounts for the diverse conditions in which road signs are encountered in real life. We also made sure to include both correct, poor-quality, and edge case examples of road signs to thoroughly test our model's ability to distinguish between different classes and handle challenging cases. Ensuring no overlap with our training / validation dataset was relatively easy is just meant we had to be very careful to not find images that directly matched the training dataset, and find ones comparable to what a real-world test may look like.

To ensure compatibility with our trained model, we developed a custom pre-processing script that standardised the collected images. This script processes all .png and .ppm images that haven't already been transformed by resizing them to 28×28 pixels and converting them to Grey scale. By applying this pre-processing step, we guarantee that our test data is in the same format as the training data, enabling a fair evaluation of the model's performance. After pre-processing, we conducted an Exploratory Data Analysis (EDA) on the test dataset, similar to the analysis performed on the training data. This step allowed us to verify data consistency and identify any anomalies or irregularities. We ensured that the images were centred within the 28×28 frame and exhibited a varying degree of angles and slight differences in the object's appearance, reflecting real-world variations.

Independent Dataset EDA

CHECK THE AND SHOW THE Normalised THING. ALSO SHOW THE NUMBER OF SAMPLES PER COUNTRY ONE.

MENTION HOW THE IMAGES ARE REFERENCED.

By following this meticulous data collection process, leveraging insights from domain experts, and applying appropriate pre-processing and EDA techniques, we created a high-quality and diverse independent evaluation dataset. This dataset will enable us to assess our trained model's generalisation capabilities and robustness when exposed to unseen examples, providing a reliable measure of its performance in real-world scenarios.

Summary of Results

Ultimate Judgement

The criterian we are using for the optimal model is

Looking at the raw results we can say that:

For Shape the best model choice is:

For Type the best model choice is:

However taking into consideration the simplicity of having one model the best overall is:

If you are looking for a “Budget Model” (computationally inexpensive), but still performs very well we would recommend:

Discuss insights gained in comparing these two related tasks

Comparison to Existing Classifiers

Comparing our results to Scientific Journals, Papers, and Articles we found:

Conclusion

Overall success

Summarize your key findings and the outcomes of your traffic sign classification model development

Discuss potential next steps or improvements to the system

What do next time.

Would've liked larger images, with colour, more data, been able to apply it more generally to perhaps indentifying people, and what not.

I wish the data was pre-augmented so that we could do more of the ML steps.

Appendix

"I did <xyz> because it is more efficient. It is more efficient because...

Introduction to what a CNN is – pitfalls of them are they work best with more data, with more parameters and independent variables.

• As part of your evaluation, you should discuss challenges you face in combining this independent data and

your models.

Problems:

generators feeding in the file paths instead of the actual pixel data, transfer learning, encoders losing the actual label for each class. Having numbers 0 – 16 is very difficult to read compared the actual name.

References

References for Data:

https://menshealth.com.au/australian-road-signs-for-dummies/

The baseline model that we chose for this assignment was Multi-Layer Perceptron (MLP). MLP is a type of neural network with a number of pre-determined layers. A number of different models were tested before settling on MLP as our baseline.

"Traffic Sign Recognition – How far are we from the solution?" by Markus Mathias, Radu Timofte, Rodrigo Benenson, and Luc Van Gool, published in 2013: https://btsd.ethz.ch/shareddata/publications/Mathias-IJCNN-2013.pdf?ref=hackernoon.com

https://www.tomsguide.com/face-off/iphone-13-vs-iphone-12

https://rmit.instructure.com/courses/125015/discussion_topics/2248913