OMP3314 Group Project Report

Image Classification Competition

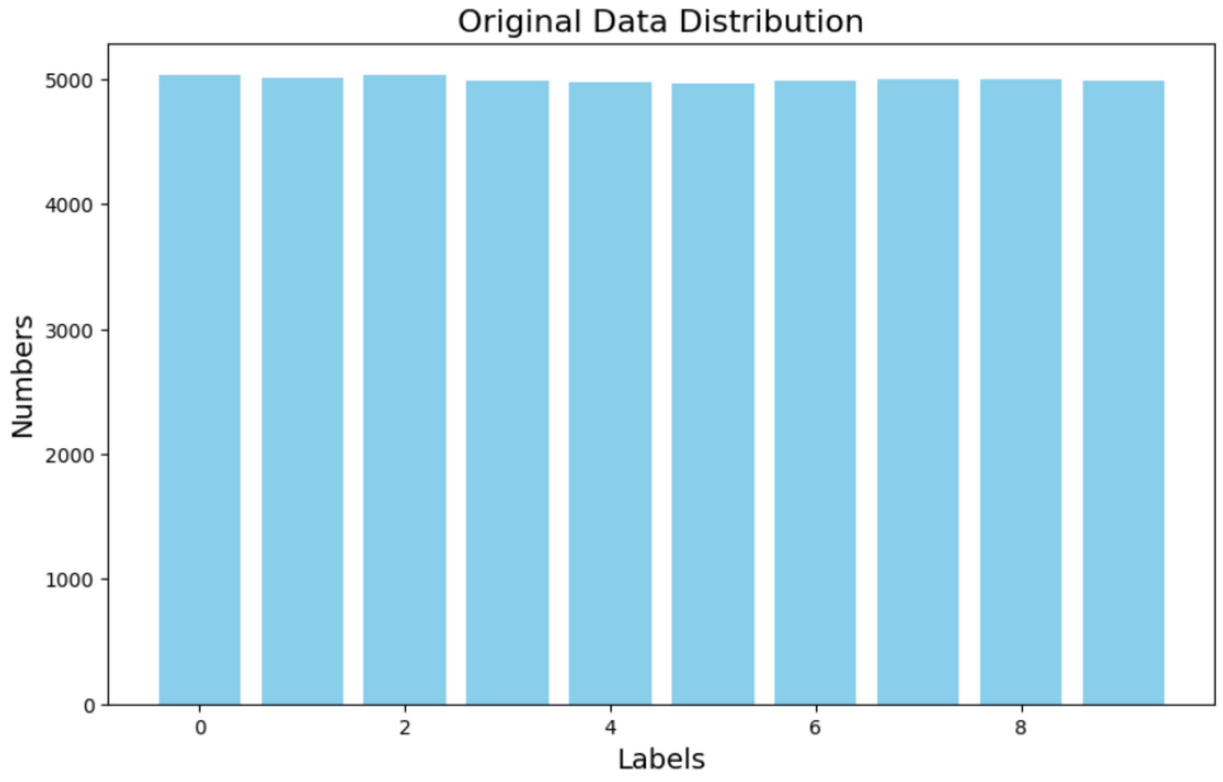
Fok Po Hin **Shen Wenqi** **Ma Xinyue**

1. Dataset Analysis

The dataset used for image classification is the dataset provided from the COMP3314 Kaggle competition. During preprocessing, the data from the decompressed pickle file and the given test, valid and train CSVs are combined into one CSV file for ease of process. In the below section, all descriptions are for the concatenated file. It is during the assessment of models that the CSV is split into the given orders (train.csv, valid.csv, test.csv).

Moreover, it is noted that the given dataset is balanced. In the total 50000 images with 10 classes, each class have around 5000 samples. The class distribution is shown in figure below.

```
Original data category statistics:
[2, 5, 7, 3, 4, 8, 0, 1, 6, 9]
[5032, 4967, 4998, 4991, 4982, 5002, 5038, 5016, 4985, 4989]
```

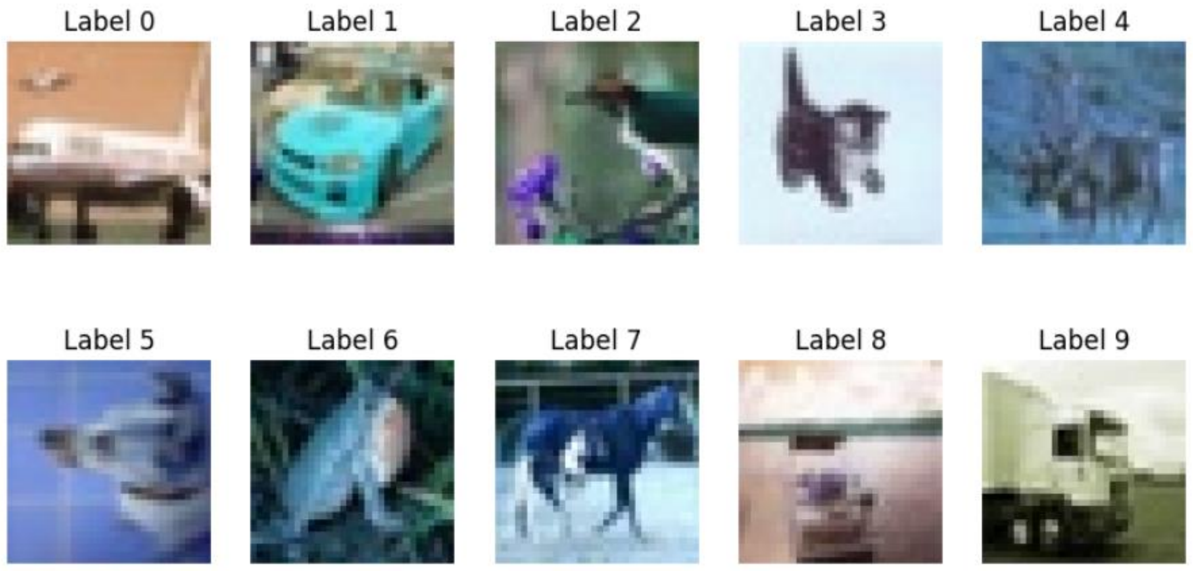


Hence, oversampling or under-sampling is not needed to balance the dataset. However, it is still important to argument the dataset with different image transformations such as horizontal flipping or color casting so that our classifiers can detect the objects independent of their orientations or colors.

The following table, based on human observations of several images with the same label, summarizes an overview of the possible explanations for each label category:

Label	Description
0	Airplane
1	Car
2	Bird
3	Cat
4	Deer
5	Dog
6	Frog
7	Horse
8	Car
9	Truck

The figure below shows a visualization for each label to support the summarizations made before:



2. Feature Extraction

Before applying classifiers, we split the training set into 40000 training samples and 10000 validation samples respectively. Various transformations are applied to the training set including horizontal flipping, vertical flipping, color casting, etc. This triples the size of training set from 40000 samples to 120000 samples. Meanwhile, the validation set, and test set is not augmented.

Since each image contains many pixels, it is impractical to use all of them for classification. As such, we extracted several image features including:

- Histogram of Oriented Gradients (HOG)
- Local Binary Patterns (LBP)
- Dense Application of Local Image Descriptors (DAISY)
- Edge-Oriented Histogram (EOH)
- Colour features (RGB and HSV)
- Blurred features

As shown in the data analysis, the data set consists of vehicles (land, air, sea) and animals, each with different geometries and contours. Hence, HOG, EOH and DAISY are chosen for extracting edge and contour features that will allow the model to know the shape of objects. LBP is used to differentiate surface textures. Colour features, RGB and HSV, are used to capture the colour distribution of the model through different colour channels. Since the images have a low resolution (32x32), blurred features are extracted to help generalization of minor variations caused by the low resolution. As a result, a total of 1,373 number of features were extracted, before applying dimension reduction, using the above techniques.

Before extracting features from all images, we need to verify whether the features are useful or not. To ensure this, we sample 100 images from dataset and check the proportion of images that obtain the same extracted features. If the proportion is high, the feature has low variability, and we need not waste memory to extract them.

Then we experimented a variety of dimensionality reduction techniques such as PCA, LDA, NDA for all classifiers to further prevent the curse of dimensionality. Using these features, we implemented SVM, KNN, Logistic Regression and an ensemble classifier.

3. Classifier Exploration

3.1. KNN

KNN was chosen based on an intuition that images of same class should appear similarly and thus having similar vector representations after feature extraction. We performed hyper parameters grid search with a range of values for n_neighbor and p. We found that higher values of n_neighbors result in higher validation accuracy, while the values for p does not have high influence on the performance. Also, it is better to employ KNN on only the LDA projected data instead of PCA projected data or a combination of both. However, the performance does not exceed 0.66 in validation accuracy. This can be explained by the tradeoff between capturing data variations and number of dimensions. With lower projected dimensions, the curse of dimensionality is prevented but the data does not have sufficient variation for KNN to distinguish the classes well. With higher projected dimensions, the data have sufficient variation for KNN to distinguish the classes, but curse of dimensionality prevented the algorithm to perform well.

3.2. Logistic Regression

Like KNN, we applied Logistic Regression with hyper parameters grid searched by differing the values of C. We found that Logistic Regression with C equals to 0.01 results in best performance. But the performance of this classifier is still unsatisfactory with validation accuracy not exceeding 0.66 like KNN.

3.3. Support Vector Machine (SVM)

As seen in the preprocessing section, 1,373 number of features were extracted, resulting in a high-dimensional feature space. Hence, SVM classifier is initially chosen for its kernel function, which allows it to project data into higher dimensions in order to separate them for classification non-linearly. SVM is also chosen for its ability to maximize the margin between classes and thus, improves generalization (80% of final testing data are private) and reduces the risk of overfitting.

Experimental results showed that although SVM took the longest time to train, it yields the best performance with testing accuracy (public dataset on Kaggle) reaching 0.74. This is our current best model for this competition. The hyperparameters for the model are as follow:

- Kernel: RBF
- Regularization parameter: 7
- Gamma: scale (scaling based on feature variance)

The RBF kernel is chosen for non-linear decision boundaries. Scaling for gamma ensures flexibility for the kernel. For the regularization parameter, it is found that while increasing the regularization parameter increased training time, it did not always translate into an increase in the accuracy, with 7 being the optimal value.

3.4 Random Forest

Random forest was chosen for the image-classification task as it can handle complex non - linear relationships in images, resist noise in large datasets, manage high - dimensional data, and offer good generalization.

Like other models, tuning its hyperparameters is essential for optimal performance:

- n_estimators: A higher value generally improves performance as more trees capture diverse patterns, but it also raises computational cost. We tested values from 50 to 1000 and found 500 to be optimal, as a balance between pattern-capturing and over-complexity.
- max_depth: Greater depth allows trees to capture intricate patterns but risks overfitting. Testing values from 2 to 10, we found little performance difference. Larger values led to overfitting and excessive time, as the tree becomes too tailored to the training data.

We augmented training data with transformations. The model reached 0.7494 accuracy on the training set after 1h15min, but only 0.609 on the testing set. Possible reasons for this drop are:

- Overfitting: Despite data augmentation, a complex model may learn augmented data too specifically, failing to generalize to the testing set.
- Randomness in the algorithm: The random selection of samples and features during training can lead to models overly influenced by training - set patterns, resulting in poor testing performance.

In summary, the overall performance of Random Forest was not as well as expected, then we tried to ensemble the models.

3.5 Light Gradient Boosting Machine (LightGBM)

Besides the classic models taught in lecture, we tried LightGBM for image classification. It fits the data by building an ensemble of decision trees with leaf wise growing strategy in a sequential order to minimize prediction errors. Similar to random forest, it can model complex non-linear relationships, but it also allows improving inaccurately predicted classes through boosting, which is why it is of our consideration. As with other models, it is essential to tune its hyper parameters for it to work efficiently:

- Max_depth: Similar to random forest, a higher depth allows the tree to capture more complex patterns in the data at the risk of overfitting. We found that higher value of max_depth greatly increases training accuracy but only increases validation accuracy slowly. At last, we found a max_depth value of 300 to be optimal, which is relatively high but not the highest max_depth experimented.
- Learning rate: A high learning rate fasten training process but too high of a value risks the loss oscillating between the optimal points. A smaller learning rate generally increases the chance for the algorithm to safely converge to a stationary point but increases the training time. We found a learning rate of 0.01 to be optimal for this dataset.

Using the above hyper parameters, it’s validation accuracy reaches around 0.65 to 0.69 depending on how many features used to train, which is higher than the previous models. It could be due to LightGBM optimizing the objective better through boosting.

3.6. Ensemble Model

Using the previous models, we constructed an ensemble classifier with the weights of included classifier being proportional to their validation accuracies. Surprisingly, the validation accuracy of ensemble classifier is only 0.7, which is lower than our single best classifier (SVM). This could be due to these models performs well in similar classes, but does not complement each other in classes that they do not exceed in.

4. Final Solution Description

As shown in Classifier Exploration, SVM achieved the highest accuracy. Hence, for our final model, SVM is chosen with the aforementioned hyper-parameters.