

Task 4.3 – Build Your Own Image Recognition System

Contents

Introduction.....	1
Task Contributions	1
Dataset	1
Base Dataset	1
Image Examples	2
Pre-Processing.....	2
Bag of Words Model.....	3
Elbow plot (k selection & optimisation).....	3
Algorithm Benchmarking.....	4
k-NN	4
Support Vector Machine	5
AdaBoost.....	5
Benchmarking Results.....	6

Introduction

In this task we were asked to find an image dataset and perform SIFT detection and create a bag of words model for the images. The word histograms were then used to train and tune k-NN, SVM and Adaboost models and compare the test set performance. This was completed as a group and the table below shows each group members contributions.

Task Contributions

Task	Contributor
Dataset – Sourcing, preprocessing and Train/Test/Validation split & description	Justin Tomlinson
BoW model build & k optimisation	Sagar Prakesh
k-NN Model	Justin Tomlinson
SVM Model	Sagar Prakesh
AdaBoost Model	Sagar Prakesh
Model Evaluation Comparison	Justin Tomlinson
Report formatting and compilation	Justin Tomlinson

Table 1 – task contributions by student

Dataset

For this task we selected the Lego Brick Dataset from Kaggle (<https://www.kaggle.com/joosthazelzet/lego-brick-images>) The complete dataset contained 16 classes with about 400 images per class totalling approximately 12700 images.

The dataset consists of consistently sized images taken while a gray Lego brick is slowly rotated. The resulting dataset therefore has many different variations of the same class including various angles and perspectives.

Base Dataset

The original Kaggle dataset contains approximately 12700 images. This was larger than we needed so we decided to select 4 classes to reduce the size. This left us with about 1600 images (400/class). This was still a bit large, so we further reduced to around 200/class by manually removing images which showed the underside of the brick. This was largely due to simplify the process of reducing the class image count. This resulted in a total of 847 images across all classes (~200/class). The total count per class can be seen in image 1 below.



Image 1 – Images per class

Samples images for each selected class can be seen in table 2.

Image Examples

Class Name	Sample images
brick_2x2	
brick_corner_1x2x2	
flat_tile_1x2	
plate_1x1	

Table 2 – sample images from the selected classes

Pre-Processing

The task requirements stipulate that the images must be varying in resolutions (sizes), viewpoints, lighting conditions, occlusions etc. The Lego Brick dataset images are all consistently sized (200x200) and the same contrast and colour (gray). To fit with the task requirements, a pre-process script was developed and applied to the dataset.

This pre-processing script selected 50% of the dataset and randomly resized and/or brightened the images using OpenCV. The resized images were randomly scaled down between [0.5, 0.6, 0.7, 0.8]. A random subset were brightened using cv.convertScaleAbs with alpha=1.3 and beta=35. Examples of the resized and brightened images can be seen in Table 3 below.

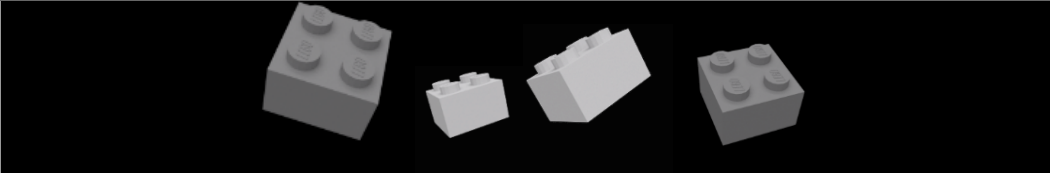
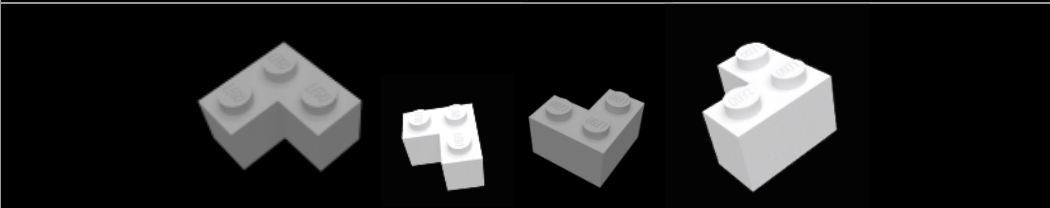
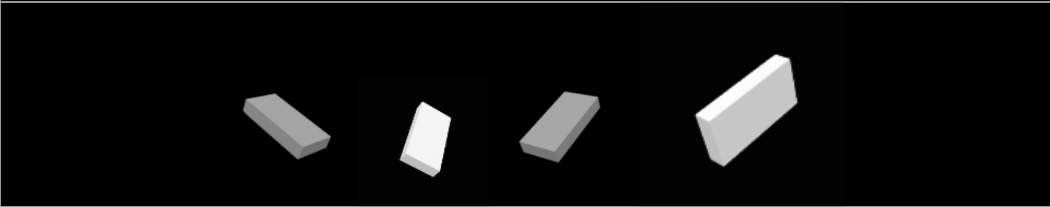
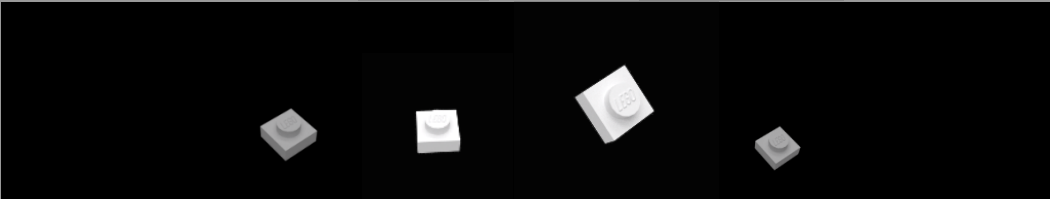
Class Name	Sample images after random transformations
brick_2x2	
brick_corner_1x2x2	
flat_tile_1x2	
plate_1x1	

Table 3 – sample images after random transformations.

Train / Validation / Test splits.

Once the randomised image transformations were performed, the dataset was then split into train/validation/test sets using the splitfolders python library using ratios (4:3:3). The resulting split of images can be seen in Image 2.

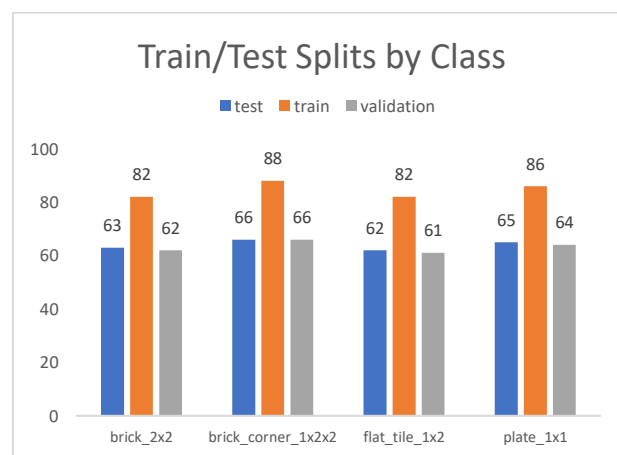


Image 2 – Train/Validation/Test split by class

Bag of Words Model

Elbow plot (k selection & optimisation)

The bag of Words model requires a k parameter for the number of words to use as part of the underlying k-means model. To select the best k value, we applied the elbow method whereby a range of k values are used iteratively, and the results plotted. A range of 10 – 200 with a step size of 10.

The optimal value identified by the “elbow” or kink in the chart was hard to identify. We decided to run 2 experiments with k=50 and k=125 against the validation set and see which one produced better results. The k=125 resulted in the better performing model across all 3 so the rest of the task was completed using this k-value. The resulting plot can be seen below in image 3.

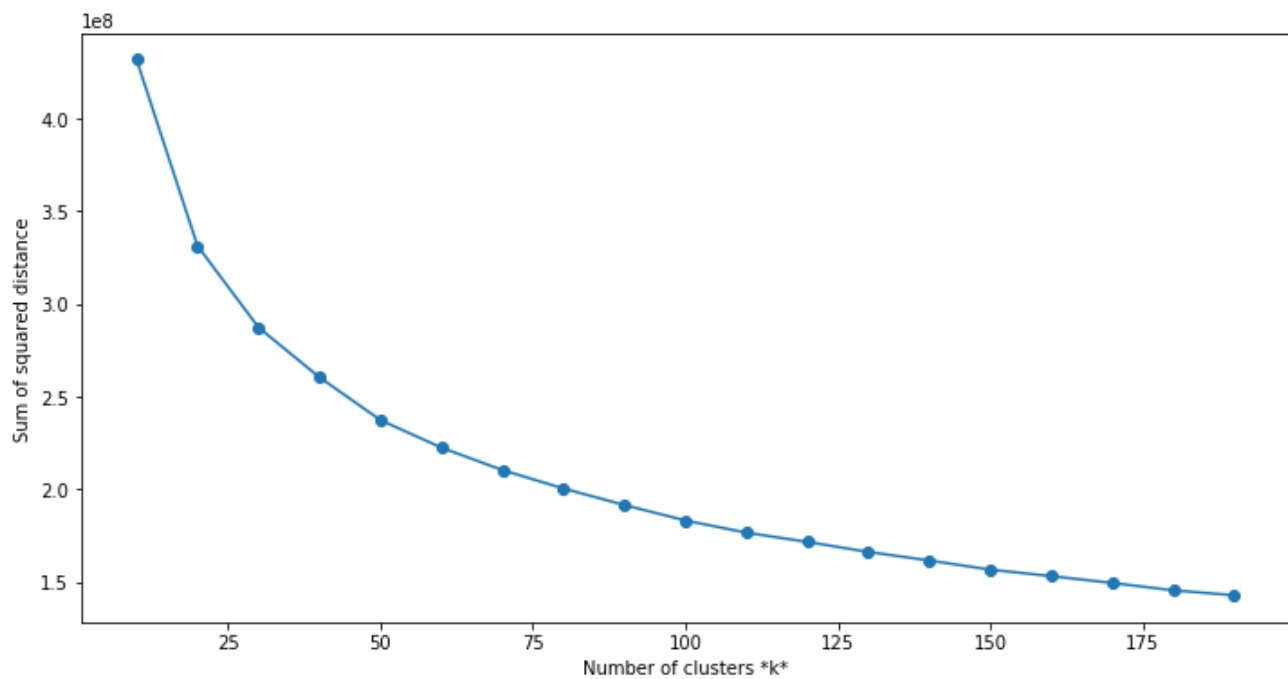


Image 3 – Elbow plot for k-selection.

Algorithm Benchmarking

For this task we experimented with k-NN, SVM and Adaboost models. Several training runs were performed while varying the main parameters.

k-NN

The k-NN model's primary parameter is the k-value. To determine the best value, we trained models using: [1, 3, 5, 7, 10, 13] and then plotted the accuracy and confusion matrix using the validation set. The results can be seen in table 4. The best performance was achieved with k=1.

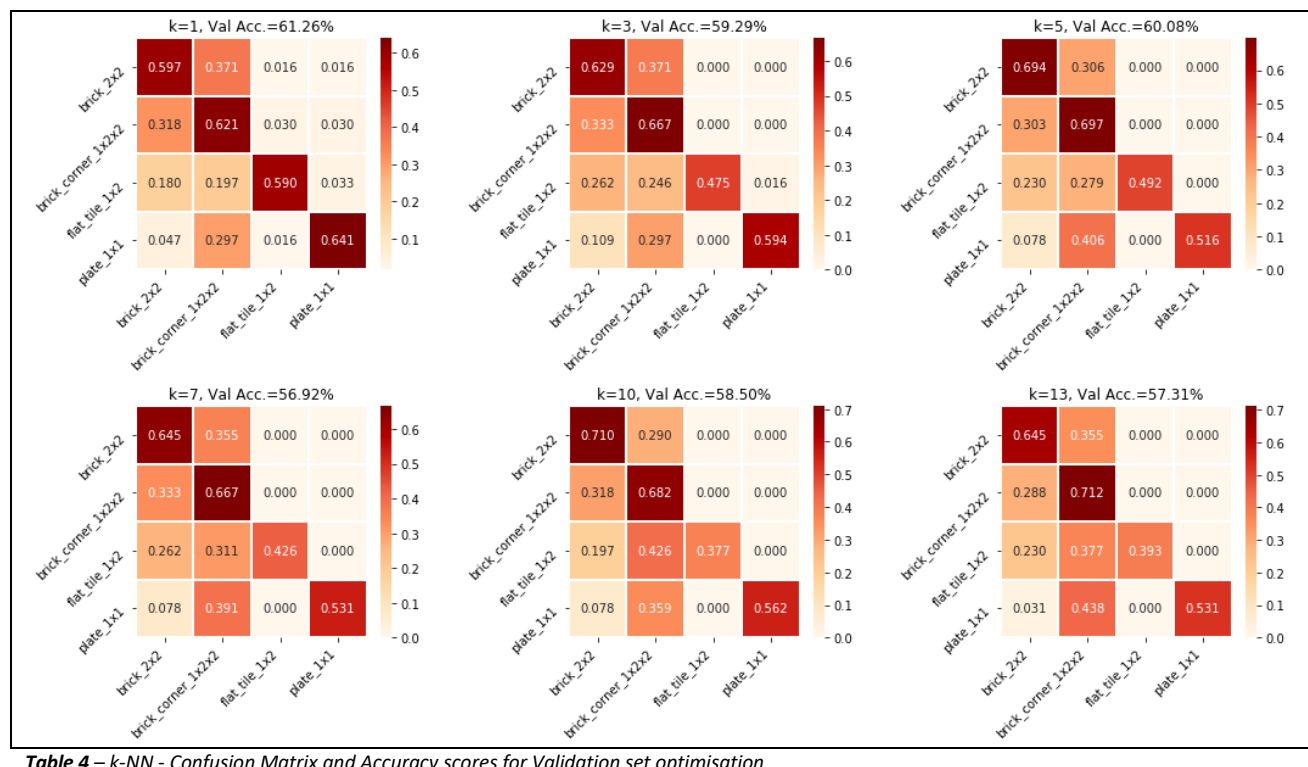


Table 4 – k-NN - Confusion Matrix and Accuracy scores for Validation set optimisation

Support Vector Machine

The SVM model's primary parameter is the C-value. To determine the best value, we trained models using: [10, 20, 30, 40, 50, 75] and then plotted the accuracy and confusion matrix using the validation set. The results can be seen in table 5. The best performance was achieved with C=40.

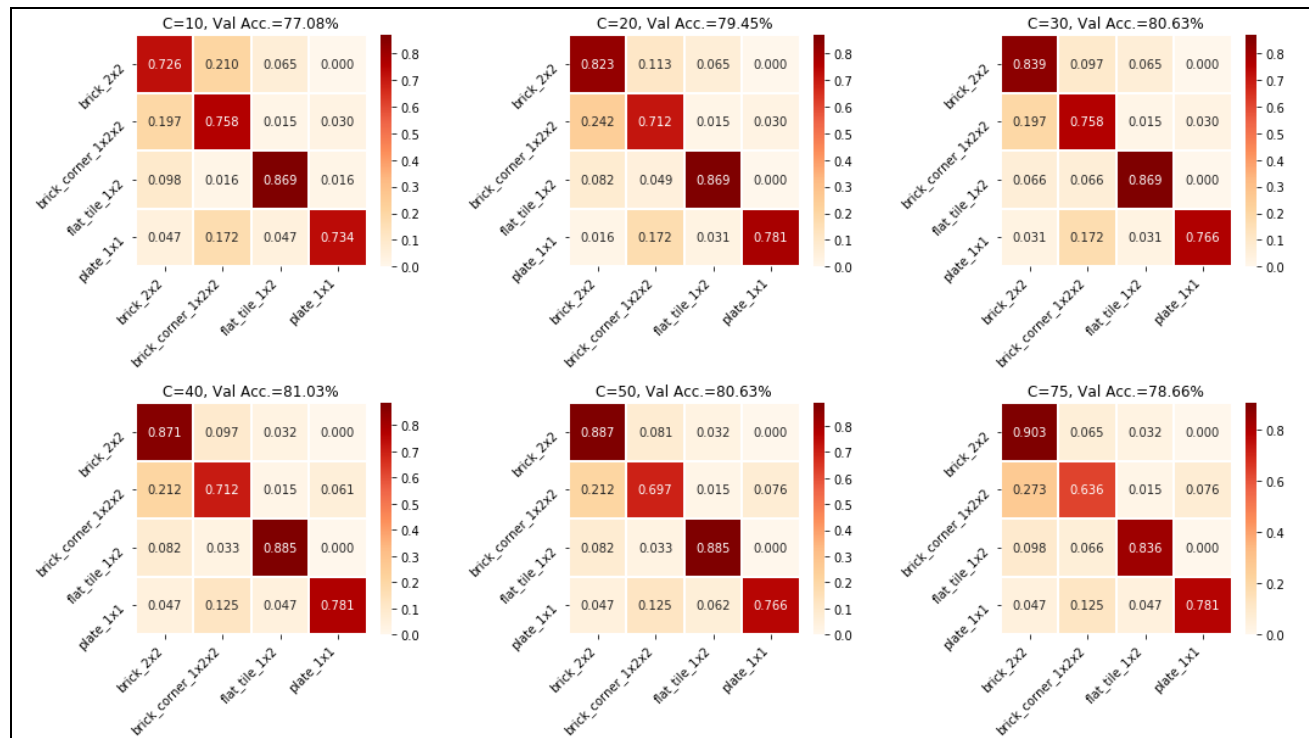


Table 5 – SVM - Confusion Matrix and Accuracy scores for Validation set optimisation

AdaBoost

The Adaboost model's primary parameter is the n_estimators. To determine the best value, we trained models using: [150, 200, 250, 300, 400, 500] and then plotted the accuracy and confusion matrix using the validation set. The results can be seen in table 6. The best performance was achieved with n_estimators =500.

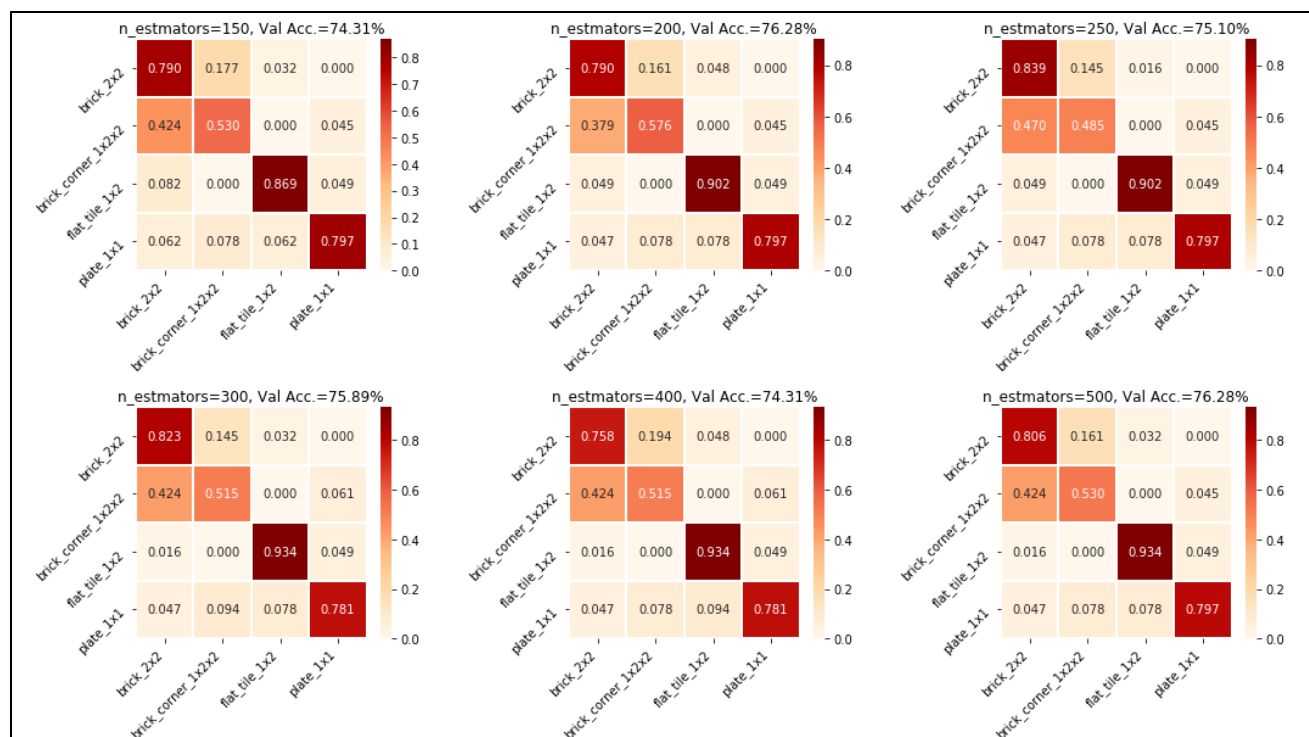


Table 6 – AdaBoost - Confusion Matrix and Accuracy scores for Validation set optimisation

Benchmarking Results

The optimal parameters determined using the validation set were used to build models and score using the test set. The validation vs test set accuracies were closest for the SVM model. The k-NN and AdaBoost models performed better on the test set.

The confusion matrix plots shown in table 9 indicated that the k-NN model was the least accurate and was often confusing classes brick_2x2 and brick_corner_1x2x2. This is not surprising as when you compare some of the images in table 7 you can see how the confusion would occur. The other classes were more distinct in shape and size and resulted in less confusion. Overall, the AdaBoost model performed the best with a test set accuracy of 81.25% followed closely by the SVM at 80.86%. The k-NN performed significantly worse achieving only 65.62%. The validation vs test accuracies are summarised in table 8.

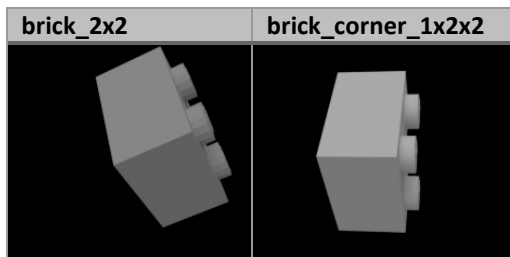


Table 7 –easily confused from the right angle

Model	Validation	Test
k-NN (k=1)	61.26%	65.62%
SVM (C=40)	81.03%	80.86%
AdaBoost (n_estimators=500)	76.28%	81.25%

Table 8 –Validation vs Test set scores

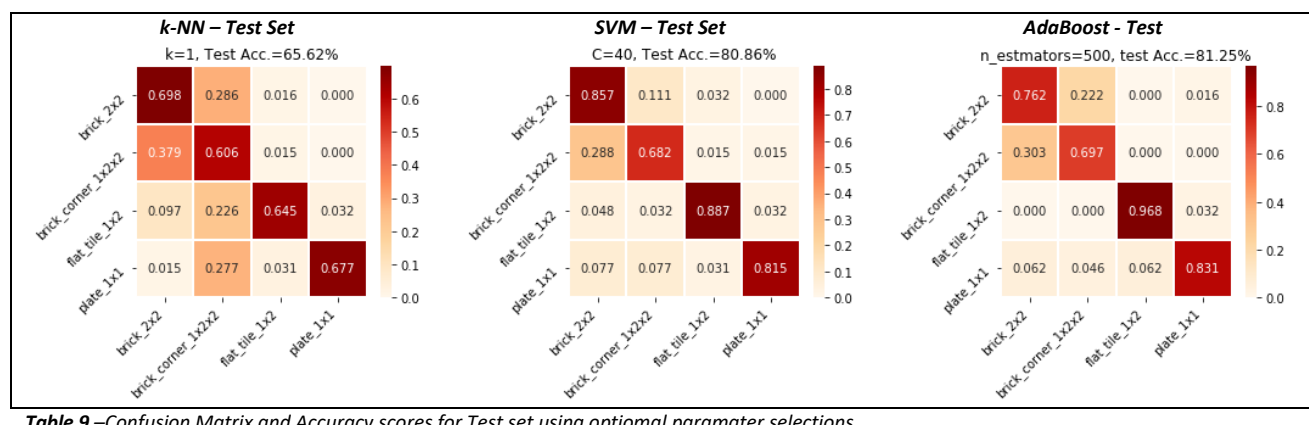


Table 9 –Confusion Matrix and Accuracy scores for Test set using optimal paramater selections