Pass Task 4.3: Build your own image recognition system – group task

Image Recognition of Mechanical Tools

Student ID 221382131
chatterjeeso@deakin.edu.au

Student ID 221023977
mskashif@deakin.edu.au

ABSTRACT

In this task, we have built an image recognition system to identify images of Mechanical Tools like Hammer, Wrench, Plier, and Screwdriver. A system such as this could be useful to household hardware retailers in order to tag images of various brands of their hardware tools to facilitate a smooth search and image retrieval experience for inventory staff as well as e-commerce clients.

1. Introduction

Comprehending mechanical tools using their images and tagging them could be very useful to both the internal and external stakeholders of any household hardware tool retailer. The image recognition system we have created, uses a Bag-of-Words model for extracting features from images and then classifies the tools based on the extracted features using 3 classifiers: k-NN, SVM, and AdaBoost.

2. Dataset

The dataset is based on the Mechanical Tools dataset available on Kaggle. A small subset of this dataset has been adopted to perform all the classification activities. The dataset is located at the URL: *https://www.kaggle.com/datasets/salmaneunus/mechanical-tools-dataset*. The dataset lists Google images as its primary source of data collection along-side other unnamed repositories.

To facilitate the image recognition of Mechanical Tools, 4 different type of Mechanical Tool images have been selected to be the **4 Classes** – **Hammer, Pliers, Screwdriver, and Wrench** and there are slightly more than 100 images for each type of these type of Classes totalling to **440 images**. The dataset has been further sub-divided into more than 40 Training Images, 32 Validation Images, and 30 Test Images for each Class. The exact split is shown in the table below:

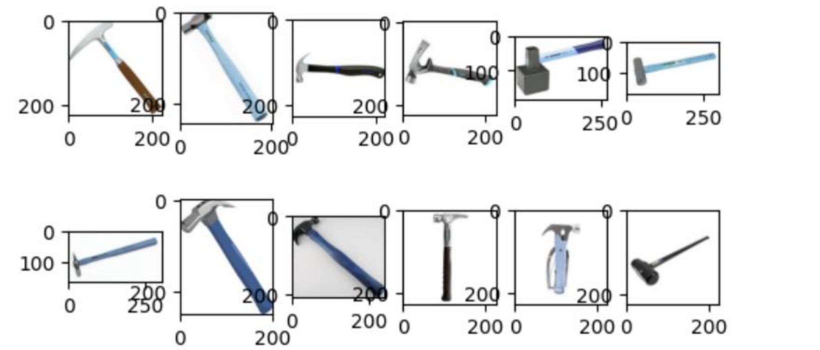
Hammer (Total = 106)			**Screwdriver (Total = 110)**	
Directory	*Count*		*Directory*	*Count*
Train	*44*		*Train*	*48*
Test	*32*		*Test*	*32*
Val	*30*		*Val*	*30*
Pliers (Total = 108)			**Wrench (Total = 116)**	
Directory	*Count*		*Directory*	*Count*
Train	*46*		*Train*	*54*
Test	*32*		*Test*	*32*
Val	*30*		*Val*	*30*

Table 1 – Count of image files for each of the 4 classes across Train, Test, and Val directories

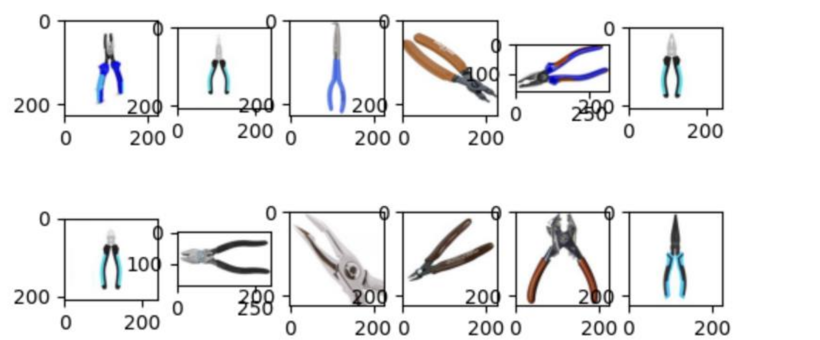
Following are a few sample images from each of the 4 types of Classes of Mechanical Tools data, generated using code. Most of the images have different viewpoints and sizes and some have different backgrounds. Its URL is https://github.com/sovikc/SIT789_4_3_P/tree/main/ToolImages

Sample Training Images

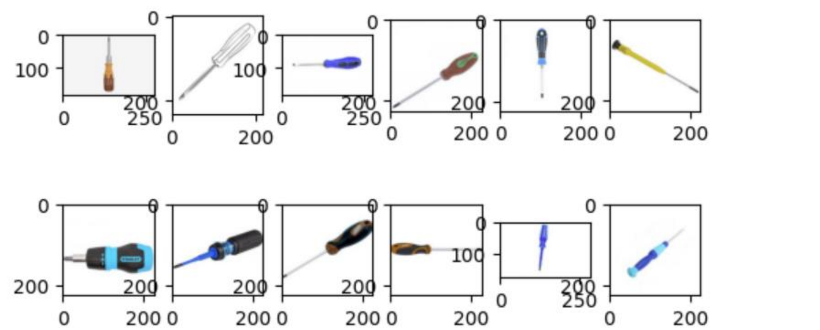
Hammer



Plier



Screwdriver



Wrench

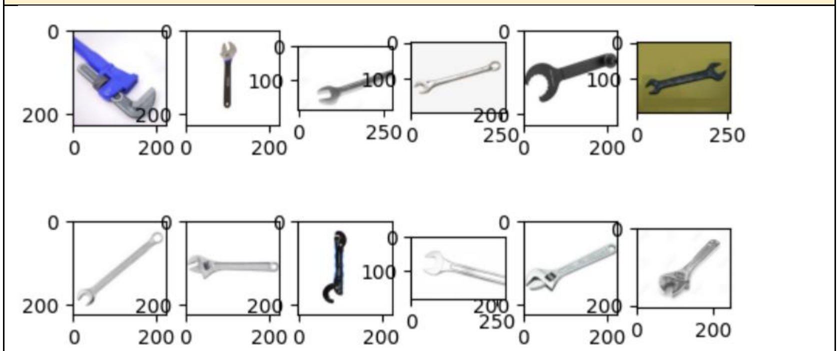


Fig 1. Sample Training Images of all the 4 classes of Mechanical Tools

Sample Validation Images

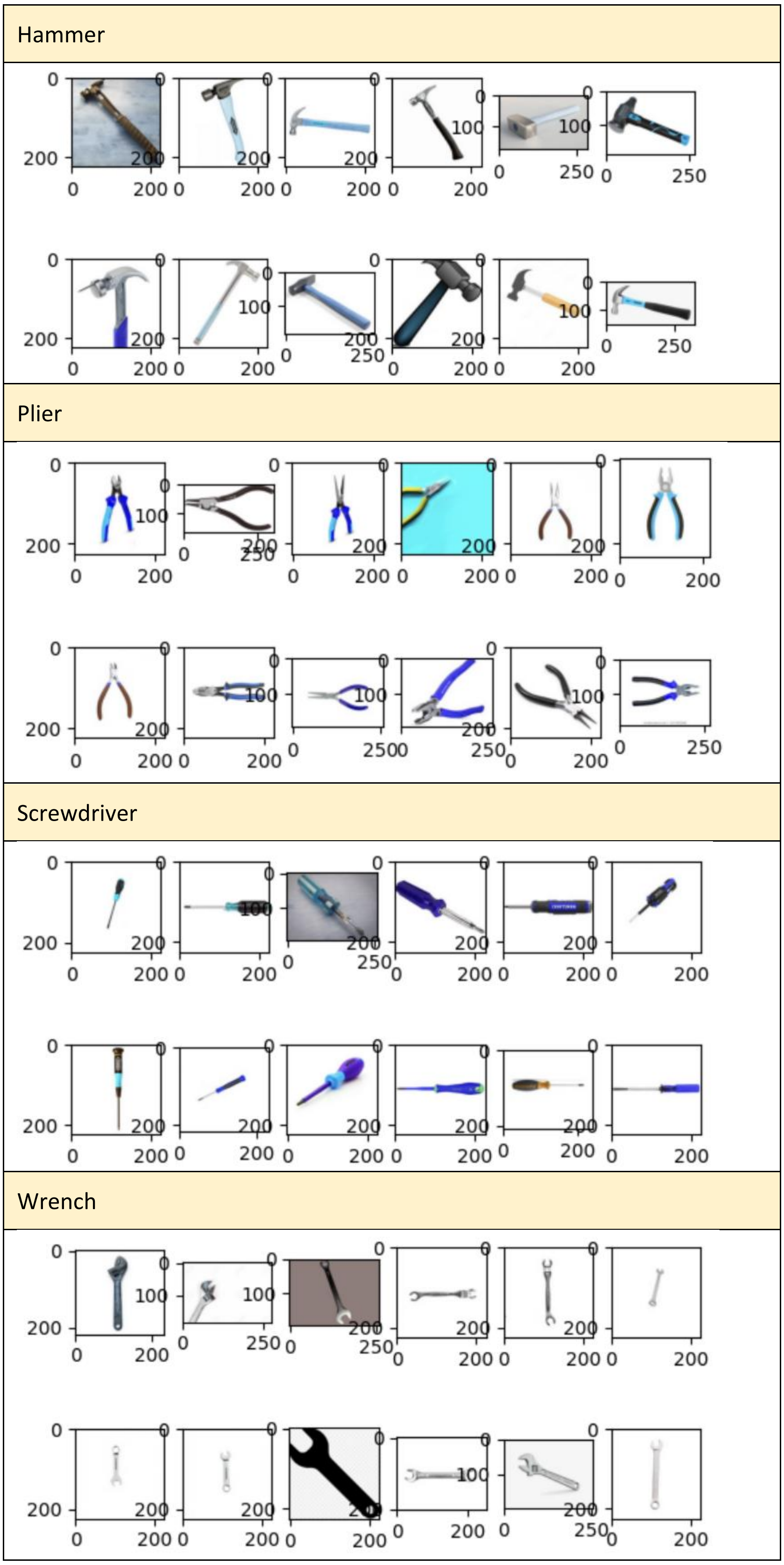
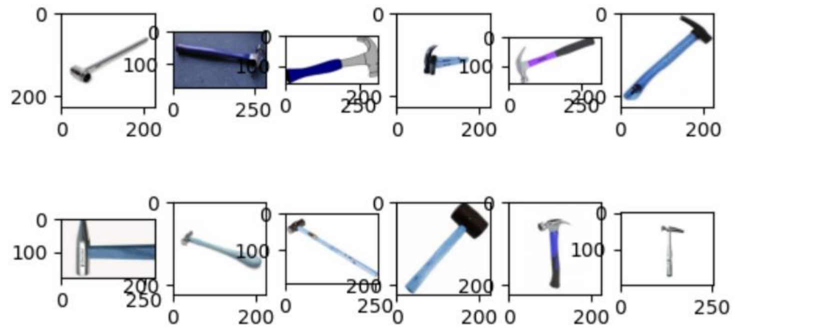


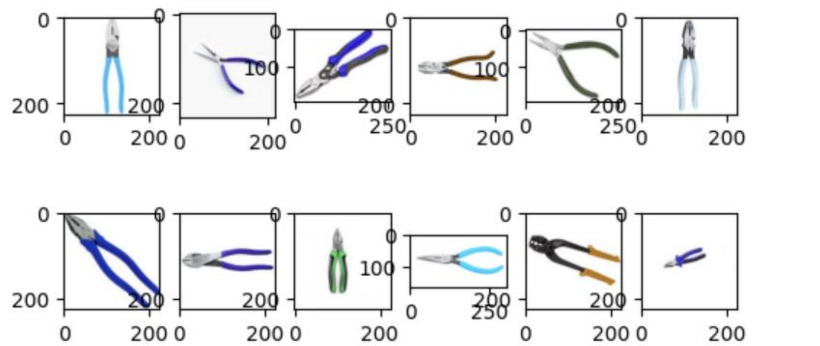
Fig 2. Sample validation Images of all the 4 classes of Mechanical Tools

Sample Test Images

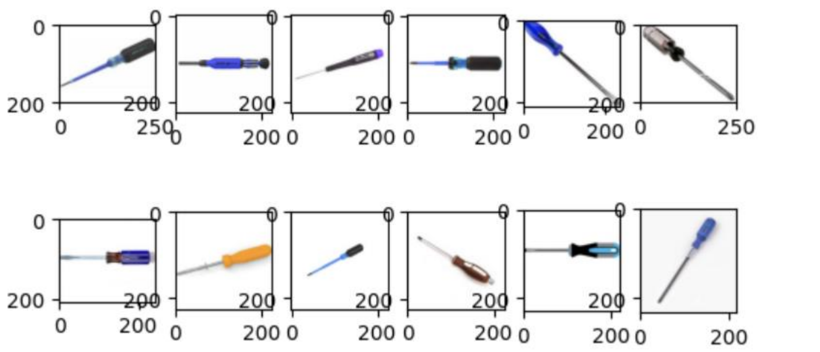
Hammer



Plier



Screwdriver



Wrench

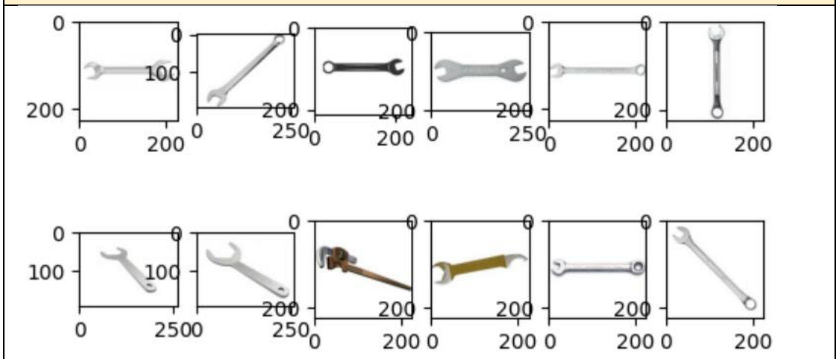


Fig 3. Sample Test Images of all the 4 classes of Mechanical Tools

It can be seen from the figures Fig. 1, 2, and 3, above that the images of the tools have different viewpoints, sizes, and backgrounds.

3. Elbow Method and Silhouette Analysis

Both Elbow Method and Silhouette Analysis were performed on the Training data. The best value for `K` was sought in a range of cluster values from 50 to 500. An Elbow method was initially applied on a set of `K values = [50, 75, 100, 125, 150, 175, 200, 225, 250, 275, 300, 325, 350, 375, 400, 425, 450, 475, 500]`. The outcome of the Elbow Method with these sets of values is shown below:

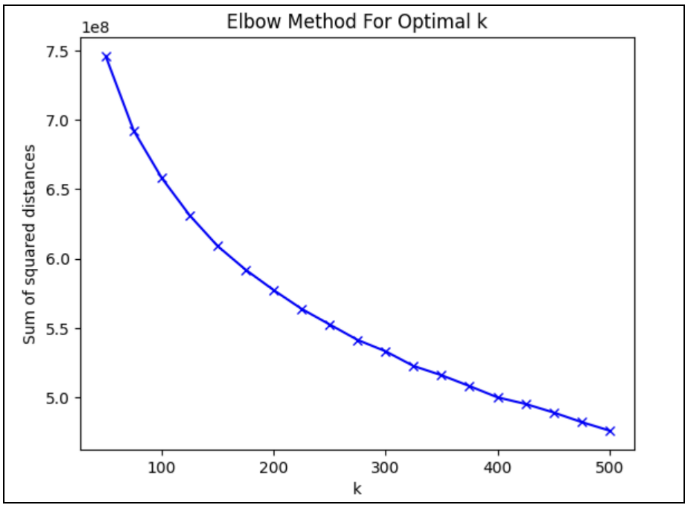


Fig 4. Elbow Diagram 1

It can be seen from Fig 4 above, there is no clear elbow in the diagram, but there looks like a steady bend somewhere half-way between the 100 and 200 marks, closer to 200. Based on this observation a second run of Elbow method was performed on the training data using an expanded set with an interval of 10 on a slightly reduced range of `K values = [50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250]` and the outcome is shown below:

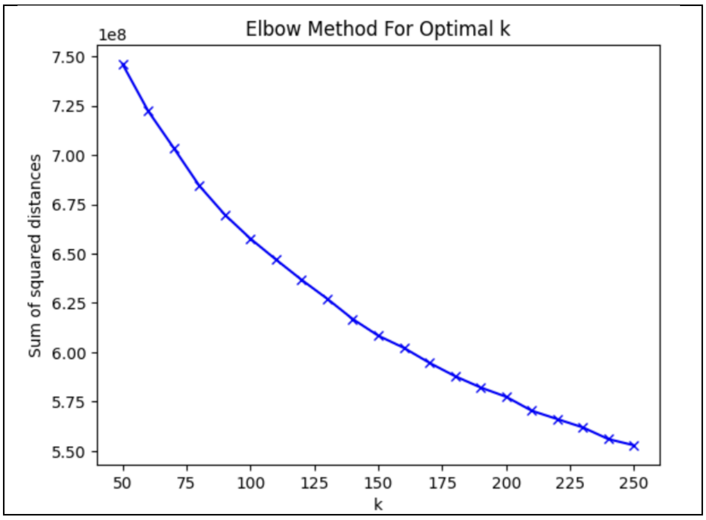
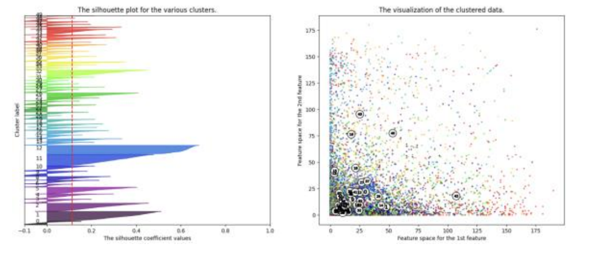
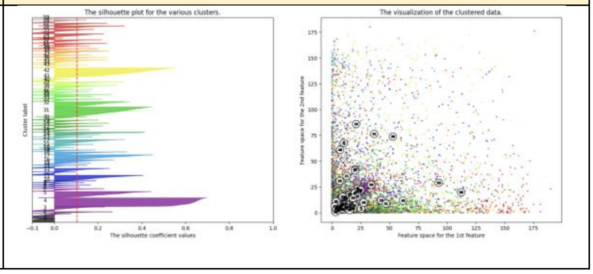
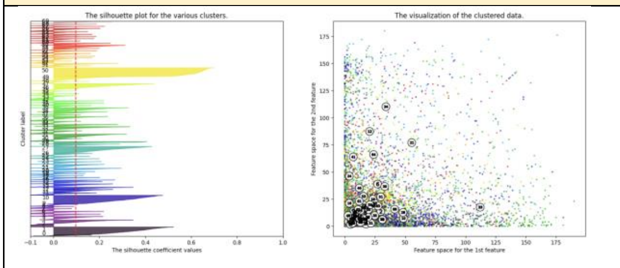
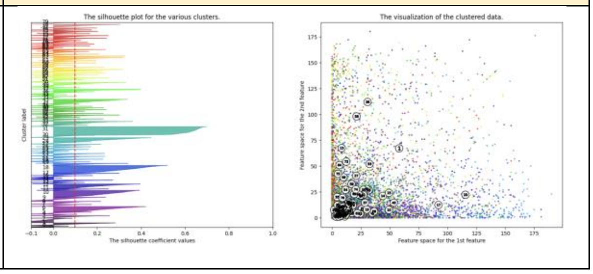


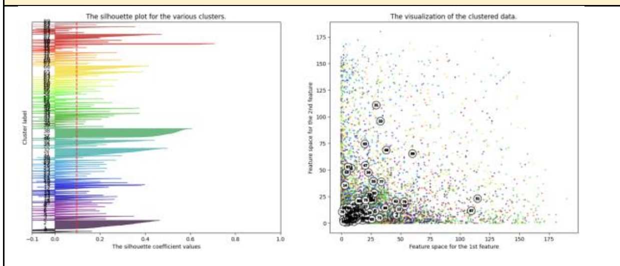
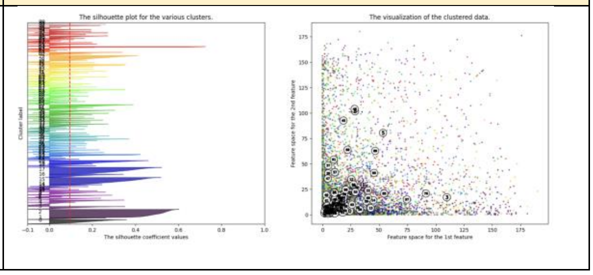
Fig 5. Elbow Diagram 2

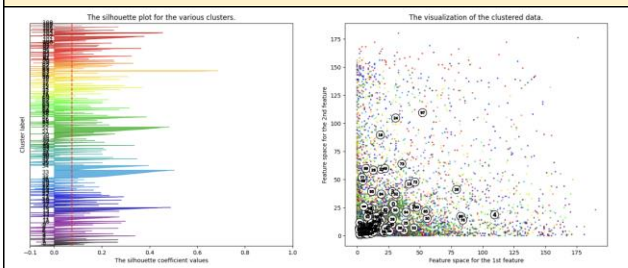
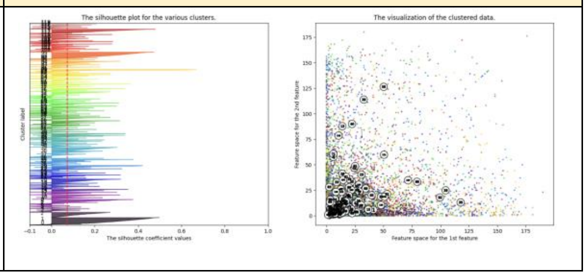
Since this showed a steady decline rather than a clear elbow, a silhouette analysis was performed on the training data to get a clearer understanding of the clusters.

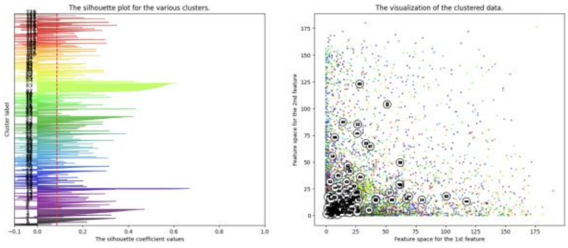
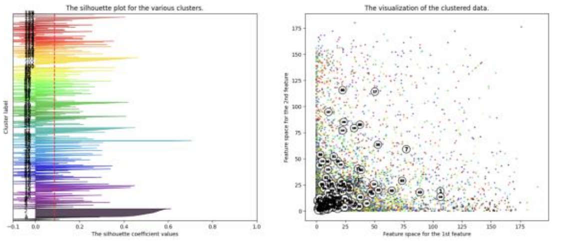
Following are the findings based on the Silhouette analysis on the training dataset for a set of cluster values = `[50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250]`

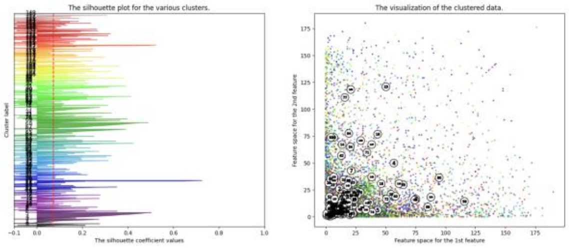
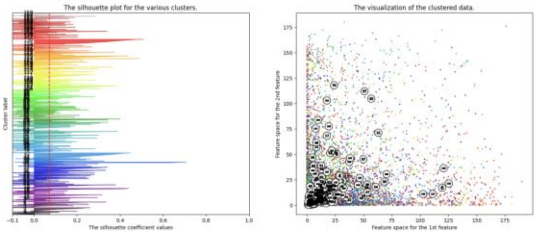
Number of Clusters = 50	*Number of Clusters = 60*
Silhouette Score = 0.1122023	*Silhouette Score = 0.10374667*
	

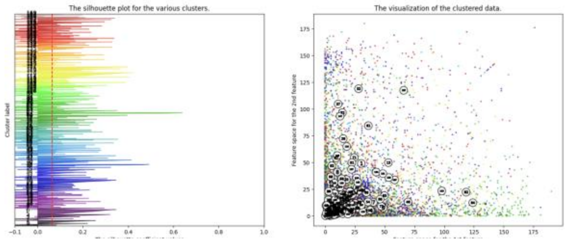
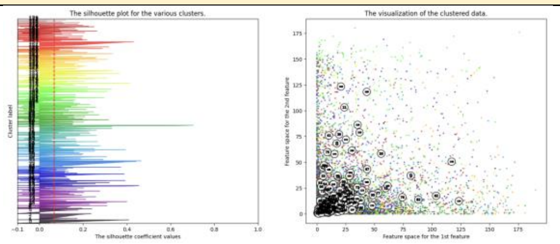
Number of Clusters = 70	*Number of Clusters = 80*
Silhouette Score = 0.097595625	*Silhouette Score = 0.09794261*
	

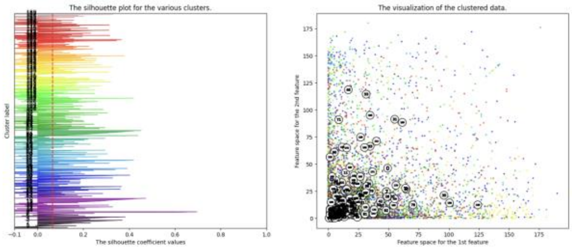
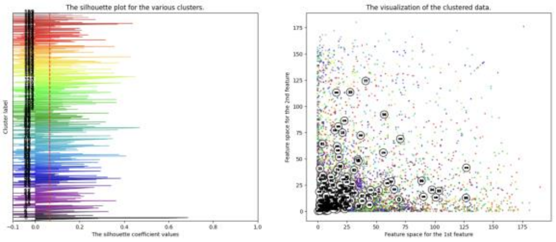
Number of Clusters = 90	*Number of Clusters = 100*
Silhouette Score = 0.09787407	*Silhouette Score = 0.094843164*
	

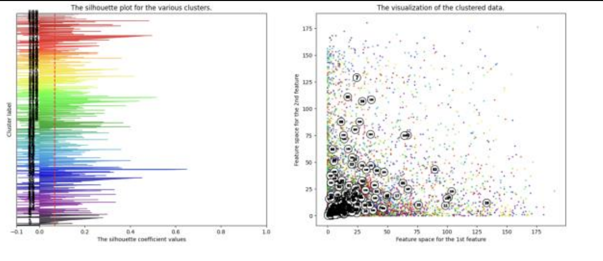
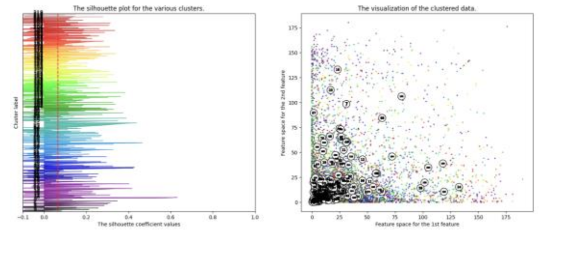
Number of Clusters = 110	*Number of Clusters = 120*
Silhouette Score = 0.07458802	*Silhouette Score = 0.072990835*
	

Number of Clusters = 130	*Number of Clusters = 140*
Silhouette Score = 0.086040564	*Silhouette Score = 0.0870144*
	

Number of Clusters = 150	*Number of Clusters = 160*
Silhouette Score = 0.07199616	*Silhouette Score = 0.07058725*
	

Number of Clusters = 170	*Number of Clusters = 180*
Silhouette Score = 0.06440761	*Silhouette Score = 0.066796996*
	

Number of Clusters = 190	*Number of Clusters = 200*
Silhouette Score = 0.06630591	*Silhouette Score = 0.06644189*
	

Number of Clusters = 210	*Number of Clusters = 220*
Silhouette Score = 0.06803277	*Silhouette Score = 0.06581328*
	

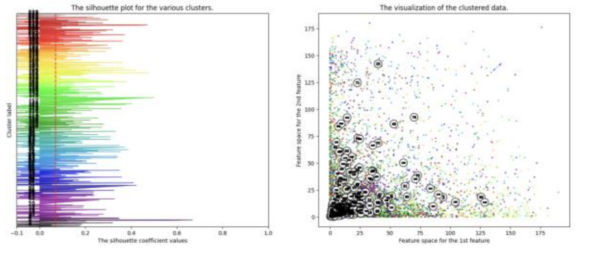
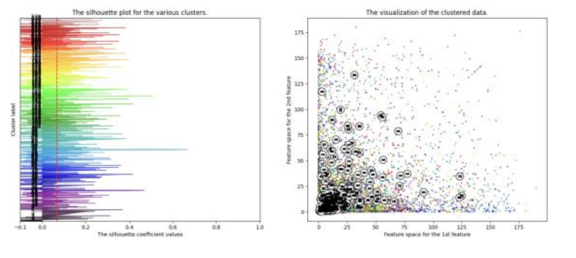
Number of Clusters = 230	*Number of Clusters = 240*
Silhouette Score = 0.06956155	*Silhouette Score = 0.06670767*
	

Fig 6. Set of images showing silhouette plot and visualization of clustered data

After comparing all the values obtained from both the methods – Elbow and Silhouette, shown in Fig. 4, 5, and 6 above a best value of 100 was selected to be the number of clusters.

4. Classifiers

After creating the Bag-of-Words Model, we were now to analyse the three classifiers that will help in assessing the word histograms derived from the tool images. All our models were implemented in a consistent manner. First, the three models had all the hyperparameters tuned with the Validation Dataset. This hyperparameter tuning was implemented with the help of a technique called the Grid Search, which essentially compares the accuracies of the models (fed the Validation Dataset), under K-Folds Cross Validation (implemented on the same dataset) . Once we get the best hyperparameters, we apply a new model and train it with the training dataset. Once the model is trained, we will evaluate its performance with the test dataset. With this dataset, we will observe the confusion matrix and generate the classification report as well.

The classifiers we have used in our case, are as follows:

- K- Nearest Neighbours
- Linear SVM
- Adaptive Boosting (AdaBoost)

4.1 k-NN

Hyperparameter tuning was performed on the Validation set using a range of values for `n_neighbors = [4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 30, 35]`. The best value returned was `9` and following are the performance metrics when `n_neighbors = 9` was used.

Recognition Accuracy	Confusion Matrix	Classification Report
0.4375	[[13 13 1 5] [5 27 0 0] [7 20 3 2] [8 10 1 13]]	precision recall f1-score support hammer 0.39 0.41 0.40 32 pliers 0.39 0.84 0.53 32 screw_driver 0.60 0.09 0.16 32 wrench 0.65 0.41 0.50 32 accuracy 0.44 128 macro avg 0.51 0.44 0.40 128 weighted avg 0.51 0.44 0.40 128

Table 2. Performance metrics of k-NN

4.2. SVM

Hyperparameter tuning was performed on the Validation set with a range of `C values = [0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 30, 50]`. The best value returned was `30`, however it was empirically found that `50` returned the best result. Following is the table of metrics that was observed on applying this hyperparameter.

Recognition Accuracy	Confusion Matrix	Classification Report
0.671875	[[16 1 8 7] [2 24 5 1] [4 6 21 1] [3 4 0 25]]	precision recall f1-score support hammer 0.64 0.50 0.56 32 pliers 0.69 0.75 0.72 32 screw_driver 0.62 0.66 0.64 32 wrench 0.74 0.78 0.76 32 accuracy 0.67 128 macro avg 0.67 0.67 0.67 128 weighted avg 0.67 0.67 0.67 128

Table 3. Performance metrics of SVM

4.3. AdaBoost

Hyperparameter tuning was performed with a range of estimator values = `[150, 160, 165, 170, 175, 180, 200, 250]` and the best `n_estimator` turned out to be `250`. However, it performed quite poorly with it. Whereas, reducing the `n_estimator` value up to `80` gave a better result. Hence, a value of `80` was chosen empirically.

Following are the performance metrics obtained with AdaBoost

Recognition Accuracy	Confusion Matrix	Classification Report
0.61718	[[25 2 3 2] [7 18 4 3] [13 4 15 0] [8 1 2 21]]	precision recall f1-score support hammer 0.47 0.78 0.59 32 pliers 0.72 0.56 0.63 32 screw_driver 0.62 0.47 0.54 32 wrench 0.81 0.66 0.72 32 accuracy 0.62 128 macro avg 0.66 0.62 0.62 128 weighted avg 0.66 0.62 0.62 128

Table 4. Performance metrics of AdaBoost

5. Comparison and Conclusion

Based on the performance metrics shown in Tables 2, 3, and 4, the following comparison table has been created.

Model	Hyperparameters	Recognition Accuracy	Confusion Matrix
k-NN	n_neighbors = 9	0.4375	[[13 13 1 5] [5 27 0 0] [7 20 3 2] [8 10 1 13]]
SVM	C = 50	0.671875	[[16 1 8 7] [2 24 5 1] [4 6 21 1] [3 4 0 25]]
AdaBoost	n_estimator = 80	0.61718	[[25 2 3 2] [7 18 4 3] [13 4 15 0] [8 1 2 21]]

Table 5 – Comparison of all the 3 classifiers on the whole dataset
(row with best recognition accuracy shaded)

As is clear from Table 5, the SVM classifier with a hyperparameter value of `C = 50` has performed the best with a recognition accuracy of 0.67 (approx.), followed by AdaBoost classifier at 0.62 (approx.) and lastly k-NN having a recognition accuracy of 0.44 (approx.), with their hyperparameter values of `n_estimators = 80` and `n_neighbors = 9` respectively on this dataset of Mechanical Tools.

Furthermore, here are some other conclusions we can also draw out from this entire task:

- Before implementing the models, the best technique to find the value of K for the K-Means model, was the 'Silhouette Scores'. These scores gave values distinct enough to determine from the ranges of K, unlike the 'Elbow Method' (This method gave a smoother trend, which made it difficult for the model to yield a better accuracy).
- The confusion Matrix of the KNN Model happens to be the most distributed among all the models. This fact also solidifies the point that this model was the least accurate model among the three models mentioned.
- All the models gave very low accuracies (through both the validation and test datasets), simply due to the important fact, that the data images we had were insufficient. This could also be a potential case of the underfitting of our models.

6. Contributions

The development of this Task was largely a collaborative effort with both the team members working together and equally on coding, model parameter tuning and writing. Following is the task breakdown of the students.

Student ID	**Activities**
221382131	• Dataset selection and creating the project dataset of 440 images • Elbow Method and Silhouette Analysis • Hyperparameter tuning and building the AdaBoost Classifier • Report Writing
221023977	• Dataset selection (suggested a dataset of casual shoes) • Hyperparameter tuning and building the SVM Classifier • Hyperparameter tuning and building the k-NN Classifier • Report Writing

The codebase along with the dataset is located at the URL https://github.com/sovikc/SIT789_4_3_P