FACULTY OF INFORMATION SCIENCE & TECHNOLOGY

TPR2251 Pattern Recognition

Trimester 2, 2025

Bridge Recognition System

Group Name: ohyeah

Lab Section: 1A

24 June 2025

	Name	Student ID
1	SIA POH XIANG	1211108363
2	ONG QI REN	1211108357
3	TANG JUN HONG	1211111767
4	LOH JIE HAO	1211108626

Table of Contents

1.0 Introduction 3

2.0 Description of design and implementation 4

2.1 Flowchart 4

2.1.1 Flowchart for Machine Learning Pipeline Used 4

2.1.2 Flowchart for Data Preprocessing 5

2.1.3 Flowchart for Part Recognition Model 6

2.1.4 Flowchart for System GUI 7

2.2 Features Extraction 8

2.3 Classification 9

3.0 Description of your input data 10

4.0 Experiment setup 13

4.1 Part Recognition 13

4.1.1 Data Collection 13

4.1.2 Experiment control 15

4.1.3 Image Preprocessing 17

4.1.4 Algorithm used to train the model 20

5.0 Performance evaluation 23

5.1 Part Recognition 23

5.1.1 Performance Evaluation of **KNN** 23

5.1.2 Performance Evaluation of **SVM** 35

6.0 Discussion and conclusions 47

7.0 References 49

8.0 Task declaration and distribution form 50

1.0 Introduction

Pattern recognition powers advancements in artificial intelligence, such as enabling machines to detect, classify, and quantify different objects regardless of changes in lighting, partially obscured views, or novel perspectives. This capability is essential for several infrastructure-related tasks, such as recognizing bridge patterns, associated with the global transportation network. Bridges present a challenging task for automated recognition in computer vision, as they can be designed as a suspension bridge, arch bridge, beam bridge, or truss bridge among others. The Bridge Recognition System in this project templates emerging models from Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) to classify the different types of bridges from images, with important applications such as monitoring of infrastructure, urban planning, or identification of disaster risk.

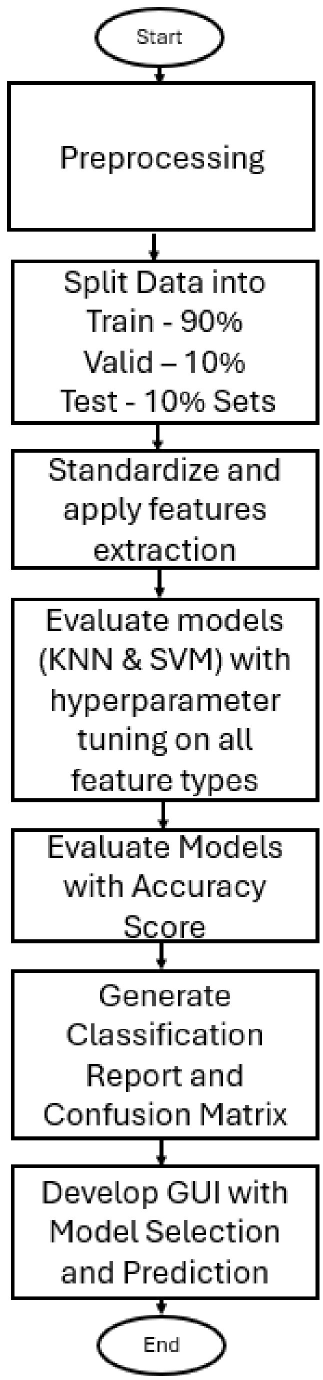
This project is in response to the need for automated systems with the ability to process large numbers of images for infrastructure management purposes. Currently, the problem with manual bridges is that they take a long time to process and there is a high element of trial and error—especially when images are captured during different lighting conditions or weather events. PCA dimensionality reduction provides us with an ability to extract information about an image and LDA distinguishes classes—in most cases giving us a very high level of precision and flexibility that a non-automated bridge cannot provide! The project proposal framework includes a formal data collection system, a highly developed image processing system, and a user-friendly interface. PCA is used to reduce data to a lower dimension, maximizing the amount of variance in the data, while LDA will maximize the variation between classification for reliable processing efforts.

The main aim is to create a Bridge Recognition System that accurately classifies types of bridges, based on at least 30 high-quality photos for each group of bridge type. The specific objectives include collating datasets with different overall designs, angles and lighting conditions; performing useful pre-processing to improve the quality of the images; taking advantage of PCA and LDA to extract features and classify; creating an intuitive user interface; documenting the system thoroughly; and determining the accuracy, reliability and scalability through a rigorous testing process.

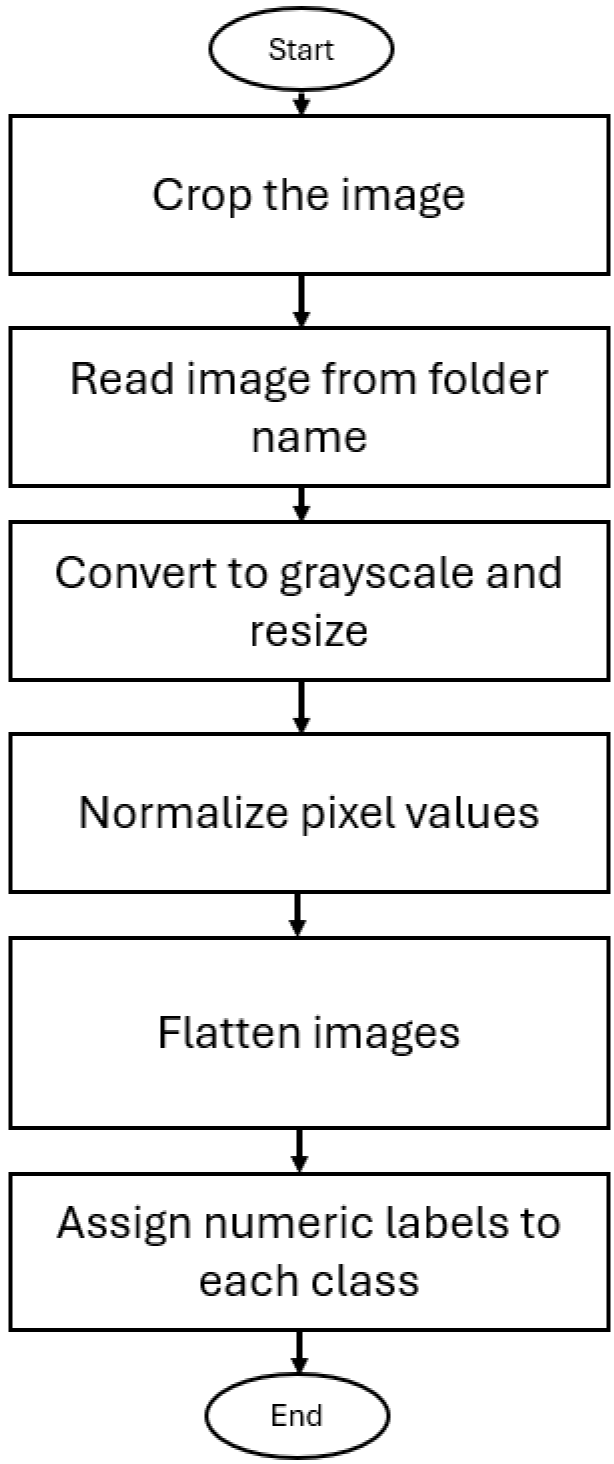
2.0 Description of design and implementation

2.1 Flowchart

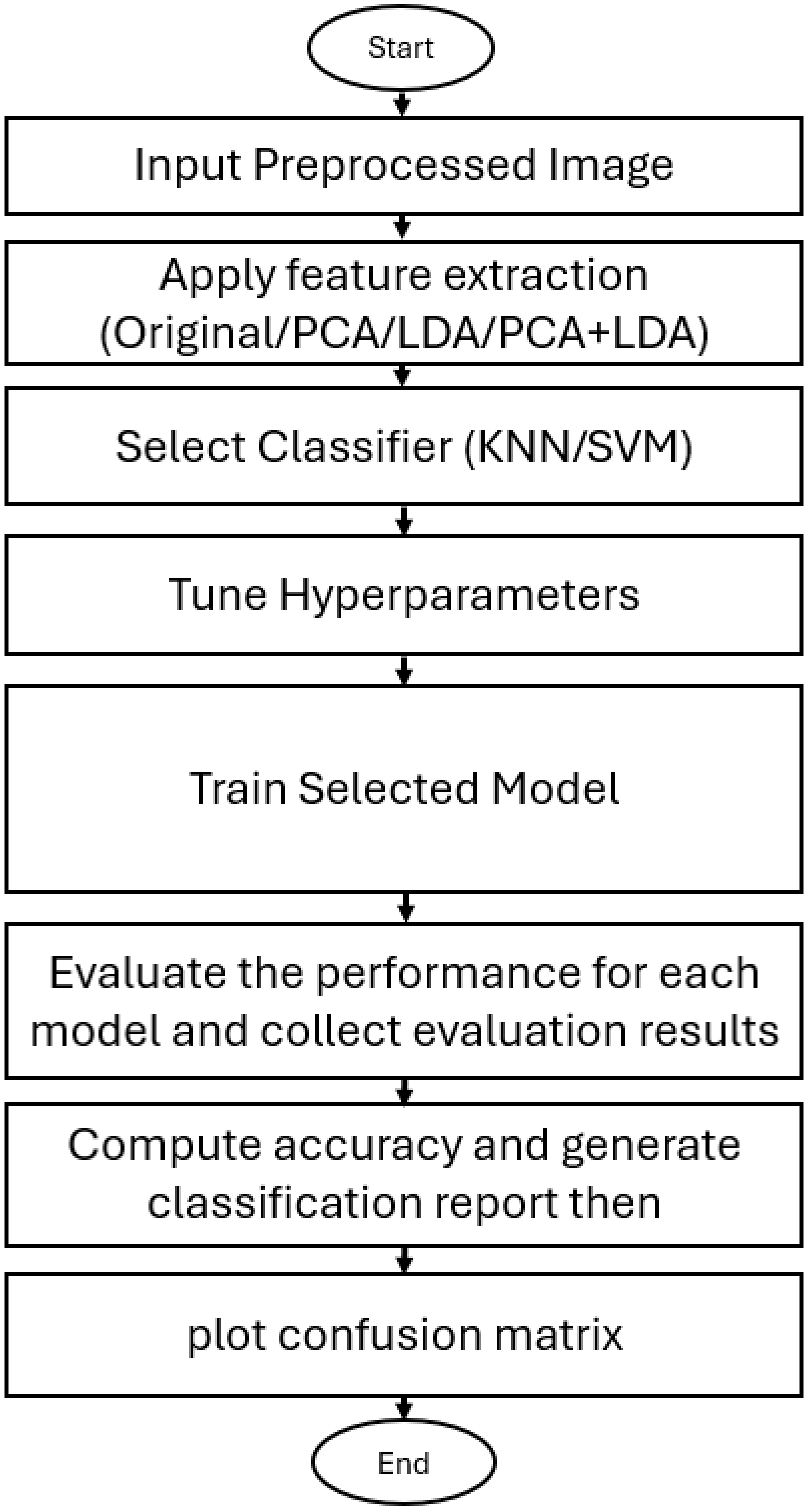
2.1.1 Flowchart for Machine Learning Pipeline Used



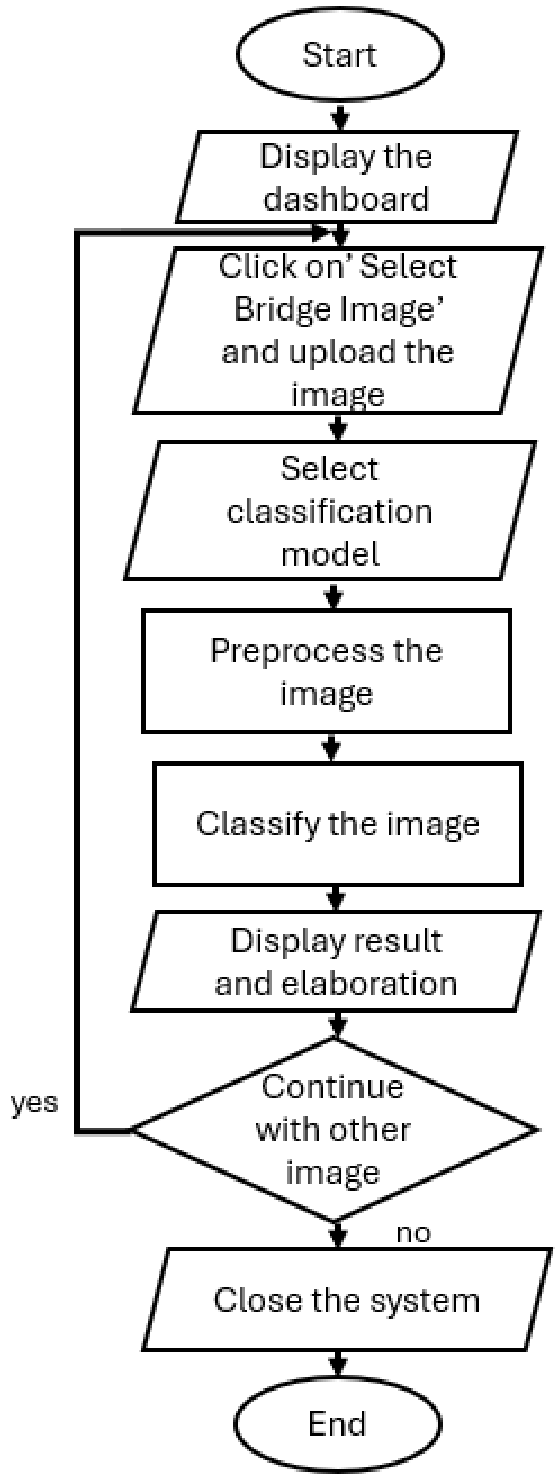
2.1.2 Flowchart for Data Preprocessing



2.1.3 Flowchart for Part Recognition Model



2.1.4 Flowchart for System GUI



2.2 Features Extraction

Feature extraction is an important component of the Bridge Recognition System, as it changes high-dimensional image data into a lower-dimensional vector that could then be used for classification. In this project, PCA and LDA will be used to reduce dimensionality and minimize surface separation, allowing us to quickly and accurately assign bridge types.

PCA is an unsupervised learning technique that takes high-dimensional and complex image data and decreases the dimensionality by finding directions of maximum variance. It works by collapsing the flattened image data (in our case a 128x128 pixel image, or 16,384-dimensional vector) onto an orthogonal axis called a principal component. The first principal component is the direction of maximum variance. All subsequent principal components are orthogonal to previous components and capture less and less variance. This dimensionality reduction greatly decreased the complexity of its computations while retaining much of the variability within the image data. PCA permitted the retention of the critical features from the original image for the bridge recognition task while also making additional processing and visualization of the data easier occur in lower dimensional data space.

Unlike PCA, LDA is a supervised algorithm that seeks to maximize classification, which is used here to maximize the separation of bridge types. LDA maximizes a composite of between-class variance and within-class variance with many covariances and means, by finding the linear combination of features. LDA uses the PCA transformed data projected onto the same measurement units that were used for previous steps, projecting the feature into discriminant axes, with respect to minimizing within-class variation (maximizing distance between classes). This transformation provides a feature set that can be used to classify the bridge type in order to maximize separation of the classes, such as if the classes are well-separated.

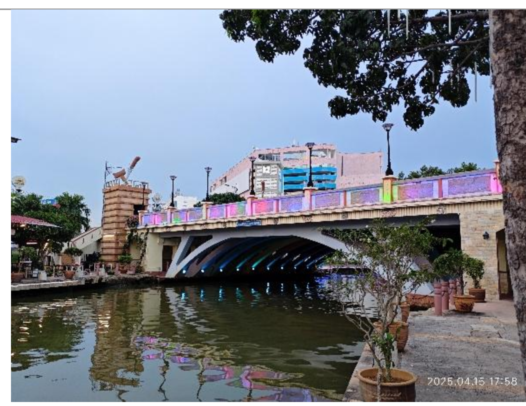
2.3 Classification

K-Nearest Neighbours (KNN) is a non-parametric, instance-based learning algorithm that is widely used for classification in machine learning. The method works by classifying an unseen data-point based on the majority class of the k nearest neighbours in feature space, where distance is typically measured using distance metrics such as Euclidean distance. When training KNN, it retains the entire dataset, and it does not build a model; it simply computes a distance to all of the training points, and then selects the k closest instances (eg, k=2 or k=3) and assigns a label based on majority voting from the neighbours. KNN is powerful, easy to understand, and versatile. KNN is useful when data are clustered, which are well separated features. KNN performance is affected by the selection of k, and distance from features, thus requires standardized preprocessing steps. Support Vector Machines (SVMs) classify using hyperplanes to best separate classes in the feature space using the largest space between the classes. SVM can be executed using either linear kernels for linearly separable data, or non-linear kernels, such as the Radial Basis Function (RBF) for separation using more complex, non-linear boundaries using transformation of data to higher dimensions. In the illustrated example datasets SVM models are fitted and trained using linear and RBF kernels on transformed features as part of the data separation process. Both KNN and SVM work with transformed features (e.g., PCA or LDA), improving the speed, and complexity and on the whole, performance of the computational train assessment over its comparative original feature set, and without detracting or improving the properties of class separation performance. With these methods i.e., meaning in terms of implementations into a GUI as evidenced through the preprocessing notebook there are very minimal to non-existent limitations on what you wish to see represented from real-time classification assessment of modelling fit outcomes. In how this is relevant now points to dis-severality of applications more broadly across numerous applications including for modelling data in structural engineering in regard to bridge classification in a involving hierarchical modelling also creates beneficial links in current bridge classification. Libraries like Scikit-learn represents good implementations of practices of research effectively improving the effectiveness of implementing such algorithms into practice.

3.0 Description of your input data

TPR

In terms of pattern recognition training, photographs are the principal input data, as they illustrate all diversity and richness needed to create a solid model. The photographs were collected between April 7, 2025 - May 2, 2025. Within a one-hour timeframe, the dataset contains photographs of four different bridge types, each of which has distinctive shapes and colors, and are used as sample parts for training the Bridge Recognition System.

Bridge	Image
Jambatan Datuk Mohd Zin	
Jambatan Hang Tuah	
Jambatan Kampung Morten	

Jambatan Old Bus Station	

Table 3.1

Photographs are the primary source of data for pattern recognition training, offering the possibilities and richness necessary to create strong models. Data was collected between April 7th, 2025, and May 2nd, 2025, taking photographs all within one hour. The dataset consists of photographs of four different types of bridges, each with distinctive shapes and colours, and used as sample parts for training the Bridge Recognition System.

To develop the generalisation capabilities of the Bridge Recognition System we took photographs in several different scenarios and conditions. Photographs were taken in different orientations and lighting situations, such as bright sun, cloudy skies, and low-light situations. The photographs were also taken where bridges were partially obscured by trees and/or buildings. We took photographs from different backgrounds to capture real-world variability, ensuring, the model is robust and with the ability to process and respond to a variety of new and unencountered data.

Bridge	Image
Jambatan Datuk Mohd Zin	

Jambatan Hang Tuah	
Jambatan Kampung Morten	
Jambatan Old Bus Station	

Table 3.2

Overall, the entire data collection process spanned a three-week lab period. We augmented the dataset by taking additional photographs of each part. This allowed us to enrich the dataset further, providing more samples for training and thereby improving the model's robustness.

4.0 Experiment setup

4.1 Part Recognition

4.1.1 Data Collection

In order to create a model for part recognition and classification algorithms, we have gathered a large number of photos. Four separate individuals took a total of 1344 photos, which varied in terms of backdrop colour and viewing perspectives (front, side, and rear). The photos were taken between April 7, 2025, and May 2, 2025. To make our model smarter and more accurate at identifying the various elements, we have used all of these diverse photos to train it.

The table below shows 4 different devices used to capture and collect the images along with specifications.

Device	Main Camera Specification
realme 11 pro plus	**Triple:** 200 MP, f/1.7, 23mm (wide), 1/1.4", 0.56um, PDAF, OIS 8 MP, f/2.2, 16mm, 112° (ultrawide) 2 MP, f/2.4, (macro) **Features:** LED flash, HDR, panorama **Video:** 4K@30fps, 1080p@30/60/120/480fps, 720p@960fps, gyro-EIS
POCO F6 Pro	**Triple:** 50 MP, f/1.6, (wide), 1/1.55", 1.Oum, multi-directional PDAF, OIS 8 MP, f/2.2, (ultrawide) 2 MP, f/2.4, (macro)

	Features: LED flash, HDR, panorama **Video:** 8K@24fps, 4K@24/30/60fps, 1080p@30/60/120/240/960fps, 720p@1920fps, gyro-EIS
infinix note 40 pro	**Triple:** 108 MP, f/1.8, (wide), PDAF, OIS 2 MP, f/2.4 2 MP, f/2.4 **Features:** Dual-LED flash, HDR, panorama **Video:** 1440p@30fps, 1080p@30/60fps
Oppo Reno 11 pro	**Triple:** 50 MP, f/1.8, 24mm (wide), 1/1.56", 1.Oum, multi-directional PDAF, OIS 32 MP, f/2.0, 47mm (telephoto), 1/2.74", 0.8um, PDAF, 2x optical zoom 8 MP, f/2.2, 16mm, 112° (ultrawide), 1/4.0", 1.12um **Features:** LED flash, HDR, panorama **Video:** 4K@30fps, 1080p@30/60/120/480fps, gyro-EIS, 720p@960fps, HDR

Table 4.1

4.1.2 Experiment control

Part	Bridge	Image	Raw data	Data after filter
0	Jambatan Datuk Mohd Zin		151	60
1	Jambatan Hang Tuah		168	60
2	Jambatan Kampung Morten		196	60

3	Jambatan Old Bus Station		152	60

Table 4.2

The Bridge Recognition System obtained total 667 images for four bridge classes, in order to have a broad range of visual conditions for each bridge, including variations in lighting, angle, and surroundings. After filtering, this dataset was reduced to 60 per class (using a total of 240) to maintain a good high-quality set of images that were not redundant, while also maintaining a balanced dataset for each bridge class to train the model optimally.

Filtering was important as we aimed to improve the quality of the dataset by removing images that were blurry, or low resolution, or occluded, we removed redundant images or images that were too similar in the hopes of maintaining a high-quality dataset to prevent overfitting, we made sure that we kept images in a consistent format (grayscale, 128x128), and we tried to keep each class at a similar number of images to lessen the chance of biasing the classification models (e.g., KNN, SVM). From a processing point of view, removing images helped reduce the workload when it came to handling the images for preprocessing, feature extraction (PCA, LDA), and model training.

4.1.3 Image Preprocessing

4.1.3.1 Assigning label and Conversion into .jpg format

The first preprocessing task is to organize the images and assign labels for supervised learning. The images are organized in folders, which are believed to be named after the different classes (for example, the names of bridges). The script parses through the list of folder names, assigns each class a numeric label (integer starting at zero), and creates a dictionary mapping the class name to the integer label. The script makes sure that each image can be associated with a class during training. Note that we are not explicitly converting the images into a .jpg format; rather, we assume the images are already compatible with .jpg, .png format etc. and we used libraries such as the Python Imaging Library (PIL) that will handle them. If we included rare formats, we would have to have a conversion step, but it is not needed in this case.

4.1.3.2 Image Resizing and Greyscale Conversion

In order to acknowledge consistency and reduced computational complexity, images will be resized and converted to greyscale with mainly the following steps. Each image is opened with PIL's Image library and converted to greyscale [Image.convert('L')], which gives the image array a single channel (as opposed to three channels which would put an image into RGB format, while still capturing the features of the image). Each image is also resized; it is resized into a meaningfully standardized resolution of 128x128 pixels. The combination of the two methods certainly ensures consistency in terms of the input dimensions of the dataset, which is an important part of preparing images for machine learning algorithms and is a less complex data structure. As the dataset is processed, these methods are sequentially completed for all images resulting in a streamlined data structure for the next series of steps.

4.1.3.3 Storing images into h5 file format

A resource efficient solution for data management is to save preprocessed images as an HDF5(Hierarchical Data Format version 5) file. Once images have been resized, converted to greyscale, and normalized, NumPy arrays (e.g., the image data, and labels) are created. NumPy arrays are written to an HDF5 file, using the recommended h5py library, with dedicated datasets for images and segments. The solution includes code to automatically create the directory if it does not already exist, making this solution relatively error prone. HDF5 is ideal since it can efficiently work with large datasets and provides quick access to the data whilst the model is being trained, while at the same time providing a compact and organized format for storing datasets.

4.1.3.4 Train -Test Split

The dataset was split into training, validation, and testing for unbiased evaluation of model performance with the function train_test_split from sklearn.model_selection by initially splitting off 10% of the data for testing, and keeping 90% for training, while setting a random state equal to 42 for reproducibility purposes. Subsequently, the training set was split into two parts again, reserving approximately 11.11% as a validation set (which at the same time reduced the training set to 80% and leaving the test set at 10%). The independent sample of the dataset to hold out for testing the model is critical since there are relatively balanced 240 images in the dataset, with 60 images in each bridge class, across training, validation and test samples - for example: 192 training images, and 24 validation images, and 24 test images. The 192 training images should give us a reasonably sized sample to learn from, the 24 validation images can be used for hyper-parameter tuning and model selection, and the last 24 of test images can independently evaluate the generalization performance of the model using the performance metrics of accuracy. This split takes into account the need to have enough training data, while still allowing the evaluation of the performance of the model using unseen and independent data

4.1.3.5 Normalize and Flatten the image data

In order to convert the image data for model training, a preprocessing pipeline was used to normalize and flatten the images. Each image was converted to grayscale and resized to a uniform size of 128x128 pixels using the PIL library. The purpose of converting to grayscale was to minimize computations required for model training while still maintaining the important structural elements of the bridge images. Each image's pixel values were then normalized to use values within range of [0, 1]. Pixel values were normalized from 1 to 255 to 0 to 1 by dividing the pixel values by 255.0. The normalization would have benefits when model training begins. The first is consistent scaling within the dataset, the second is helping with model convergence, and lastly can help the achieve better model performance by avoiding fluctuations due to variations in pixel intensity. Finally, the images were flattened into one-dimensional arrays of size 16,384 (128x128). At this point, the data was reshaped into a format that could be input to machine learning algorithms, KNN and SVM. The process we have followed to preprocess the images was applied to all 240 images. The resulting feature matrix will be named X and have the shape of (240, 16384). The labels for image classification will exist in another array we will write to memory, which we named y. Pre-processed data will be written to a HDF5 file for quick storage of data and retrieval of data and as a result can be used to run our training and test models.

4.1.4 Algorithm used to train the model

4.1.4.1 KNN with PCA

Principal Component Analysis (PCA) was performed to decrease the dimensionality of the standardized training, validation, and test image data while keeping 95% of the variance. The purpose of this transformation was to decrease computational complexity while capturing the most important features of the data. Both the PCA was fit on the training data only applying PCA to the validation and test datasets. A K-Nearest Neighbors (KNN) classifier with 1 neighbor was used to classify the labels of each image. Each hyperparameter, such as the number of neighbors, these were determined by prior experimentation or tuning. The KNN model was trained by predictions of the PCA training, validation, and testing dataset.

4.1.4.2 KNN with LDA

Linear Discriminant Analysis (LDA) was used to reduce the standardized training, validation, and test image data into a reduced dimensional space of 3 components, due to the restrictions of 4 bridge classes (LDA is limited to the number of classes - 1). LDA maximizes class separability which makes it suitable for classification approaches. A KNN classifier was trained with 1 neighbor on the LDA reduced data, where the hyperparameter (number of neighbors) was based on performance remonstration. The model built on the LDA data was then used to provide labels for the LDA reduced datasets.

4.1.4.3 KNN with PCA and LDA

A two-stage transformation involved the use of principal component analysis (PCA) and linear discriminant analysis (LDA). In the first stage, PCA was conducted with the aim of reducing the dimensionality of group standardized data so that 95% of the variance was retained. In the second stage, LDA was performed on the PCA-transformed data in order to project it into 3-dimensional space, maximizing the capacity for class separation. A KNN classifier was trained, which uses 6 neighbours to make predictions. The number of neighbours was chosen as it maximized the predictive performance in the transformed feature space. The final LDA model used a single classifier trained on the PCA + LDA-transformed data to predict labels for training, validation and test data processed through both transformations.

4.1.4.4 KNN without PCA or LDA

A KNN classifier with 13 neighbors was trained on the standardized (but not dimensionally reduced) training set, which used the full 16,384 dimensions. The choice of 13 neighbors was based on manipulation of bias and variance, so that we didn't perform feature extraction, and thus simply worked with the raw pixel data after standardization. The trained model was used to predict labels for the training, validation, and test datasets.

4.1.4.5 SVM with PCA

A PCA was conducted on the training, validation, and test datasets that had been standardized to retain 95% of the variance. Next, a Support Vector Machine (SVM) with a radial basis function (RBF) kernel was constructed of the PCA-transformed data. The hyperparameters were C=10 and gamma='scale' as this was the optimized parameterization for use with the feature space once it was reduced. The trained SVM model was then used to predict labels using the PCA-transformed data.

4.1.4.6 SVM with LDA

Following the pre-defined training, validation, and test datasets were projected into a 3-dimensional space using LDA, with class separation maximized. Data in LDA form was then used to train an SVM classifier with RBF kernel. The hyperparameters C=0.1 and gamma=0.1 were selected for optimized classifier performance. The data LDA-trained model was then used to predict labels for every dataset in LDA form.

4.1.4.7 SVM with PCA and LDA

The data was first standardized and then subjected to PCA to extract 95 percent of the variance, followed by LDA to represent the data in three-dimensional space. We trained an SVM with an RBF kernel using the data represented with PCA and LDA. Hyperparameters for the SVM were set to optimal performance values of C=10 and gamma=0.01. The trained model was used to predict labels for the training, validation, and test datasets, both PCA + LDA transformation.

4.1.4.8 SVM without PCA or LDA

A support vector machine (SVM) with a radial basis function (RBF) kernel was trained on the standard input training dataset (it trained directly on the entire 16,384-dimensional feature space directly with no dimensionality reduction applied) with C = 100 and gamma = "auto". Once the model was trained the SVM was used to predict the training, validation, and test data labels.

4.1.4.9 Saving Trained Models:

All trained models, the KNN and SVM classifiers were trained for each of the configurations (with only PCA, LDA, PCA+LDA, and no transformation), and the corresponding transformations (PCA, LDA, and StandardScaler), have been stored in memory through the BridgeClassifierGUI application. The methods are loaded upon initialization to allow on-the-fly classification, obviating the need for retraining the models, thereby saving the computational overhead and creating for an efficient interaction to the end user.

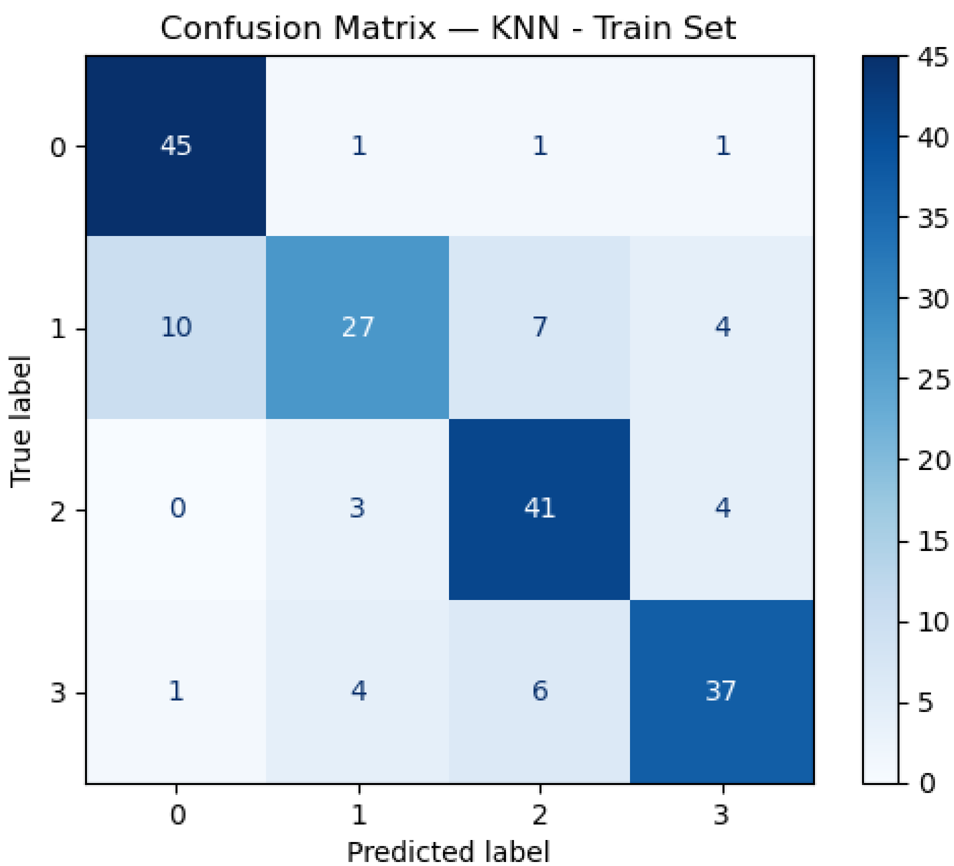
5.0 Performance evaluation

5.1 Part Recognition

5.1.1 Performance Evaluation of **KNN**

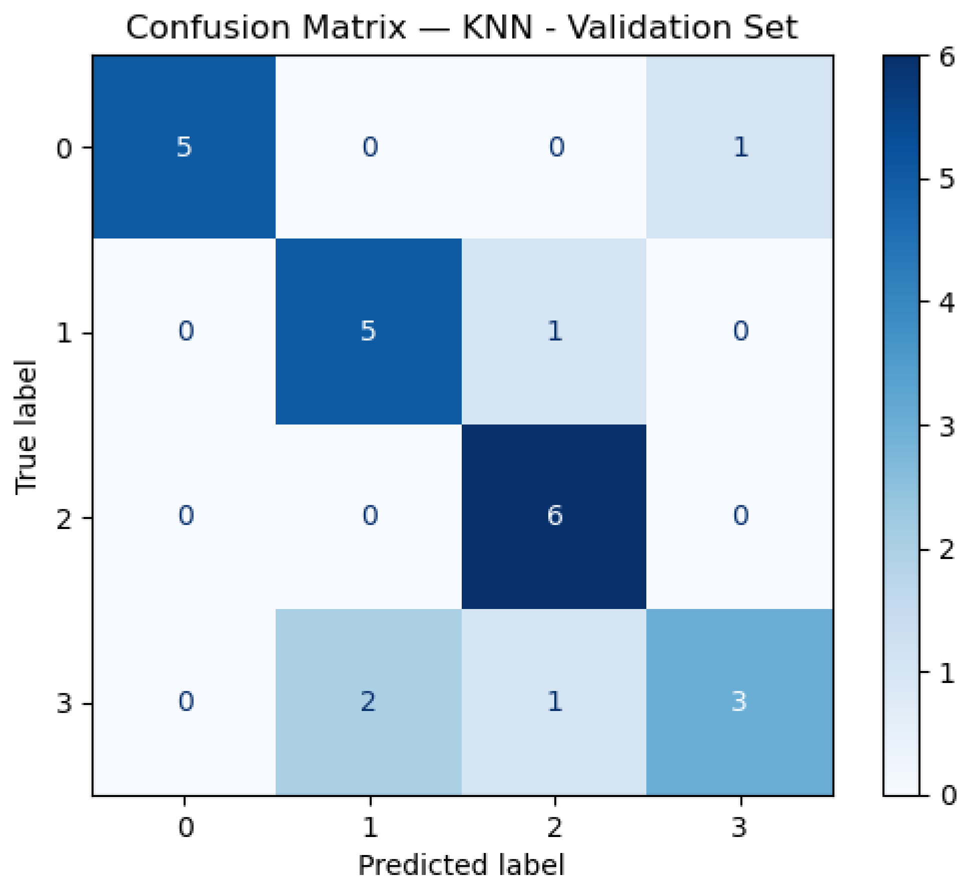
5.1.2.1 KNN Training Set Classification Report (Original Feature)

Class	Precision	Recall	F1-Score	Support
0	0.80	0.944	0.87	48
1	0.77	0.56	0.65	48
2	0.75	0.85	0.80	48
3	0.80	0.77	0.79	48
Accuracy			0.78	192
Macro Avg	0.78	0.78	0.77	192
Weighted Avg	0.78	0.78	0.77	192



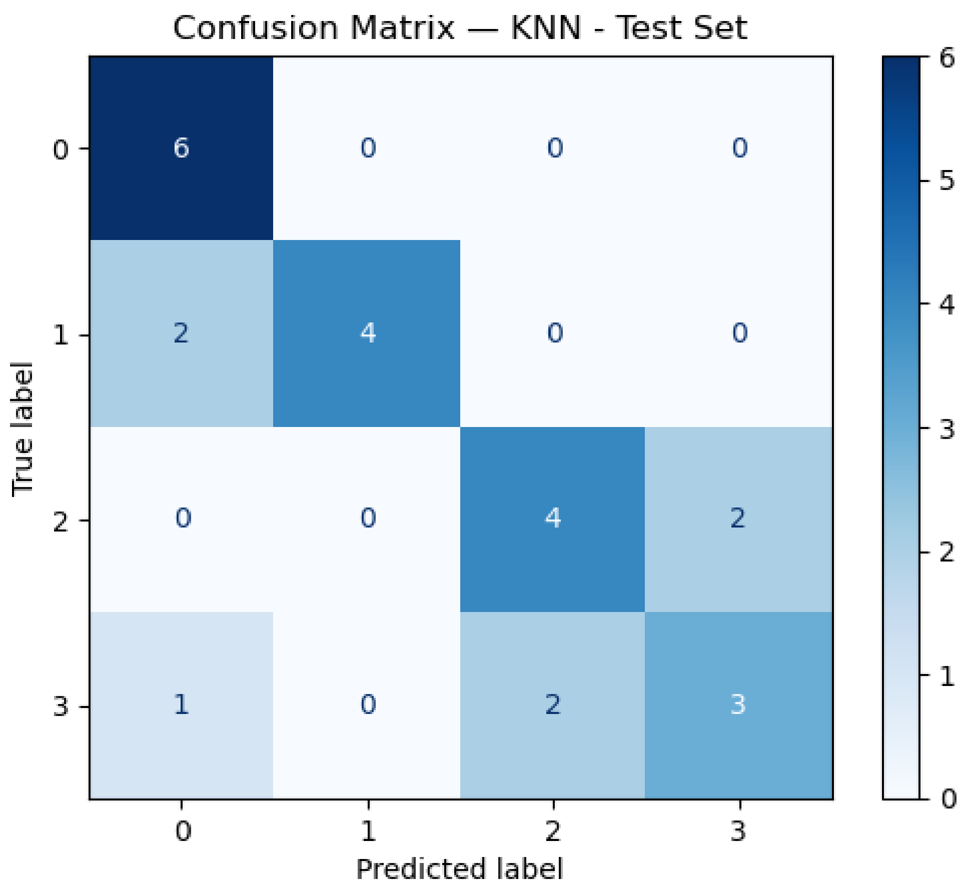
5.1.2.2 KNN Validation Set Classification Report (Original Feature)

Class	Precision	Recall	F1-Score	Support
0	1.00	0.83	0.91	6
1	0.71	0.83	0.77	6
2	0.75	1.00	0.86	6
3	0.75	0.50	0.60	6
Accuracy			0.79	24
Macro Avg	0.80	0.79	0.78	24
Weighted Avg	0.80	0.79	0.78	24



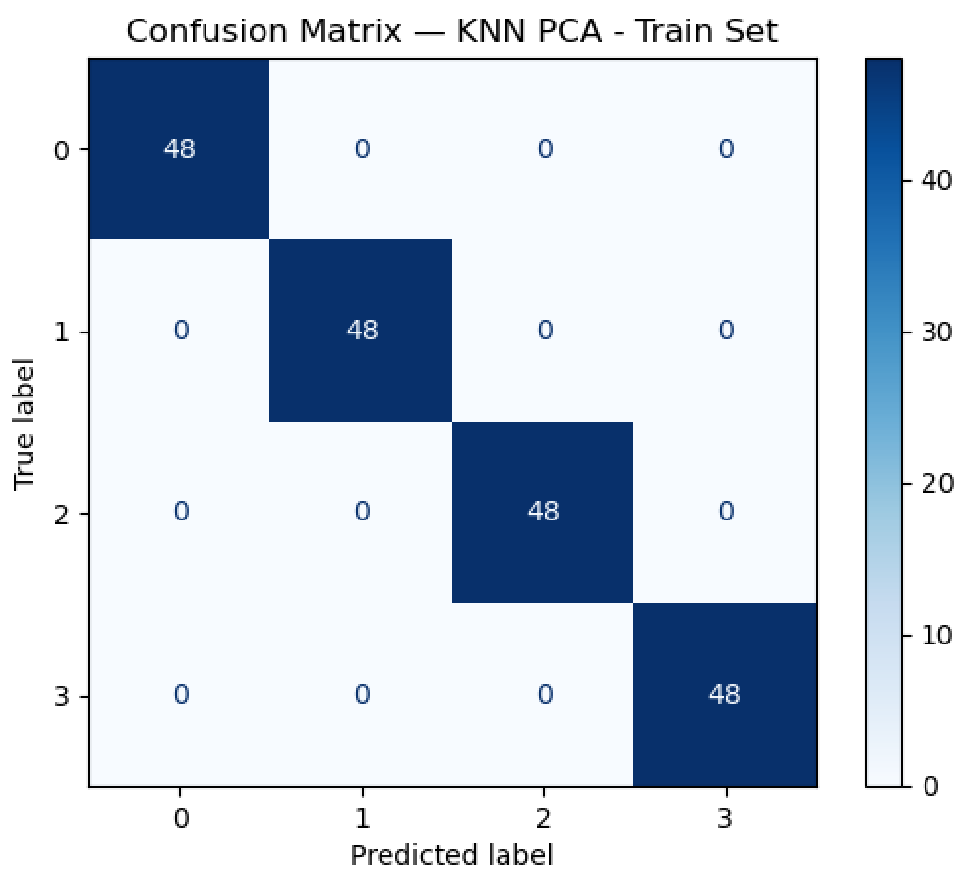
5.1.2.3 KNN Test Set Classification Report (Original Feature)

Class	Precision	Recall	F1-Score	Support
0	0.67	1.00	0.80	6
1	1.00	0.67	0.80	6
2	0.67	0.67	0.67	6
3	0.60	0.50	0.55	6
Accuracy			0.71	24
Macro Avg	0.73	0.71	0.70	24
Weighted Avg	0.73	0.71	0.70	24



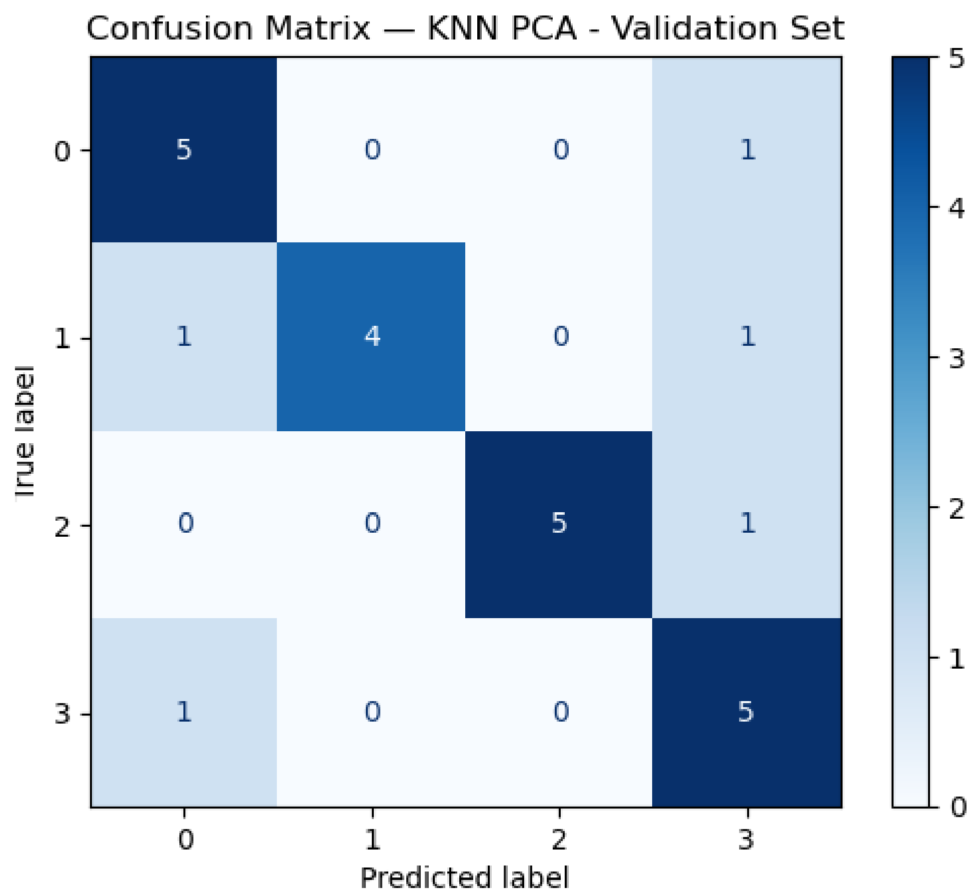
5.1.2.4 KNN Training Set Classification Report (PCA Feature)

Class	Precision	Recall	F1-Score	Support
0	1.00	1.00	1.00	48
1	1.00	1.00	1.00	48
2	1.00	1.00	1.00	48
3	1.00	1.00	1.00	48
Accuracy			1.00	192
Macro Avg	1.00	1.00	1.00	192
Weighted Avg	1.00	1.00	1.00	192



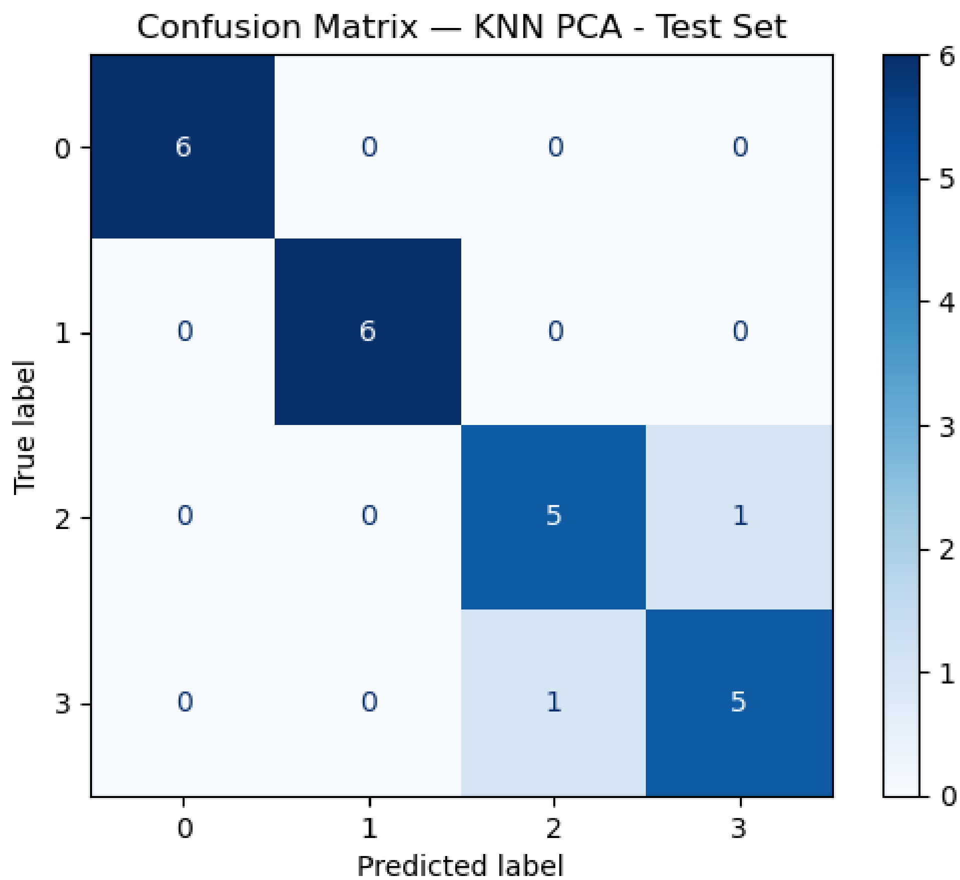
5.1.2.5 KNN Validation Set Classification Report (PCA Feature)

Class	Precision	Recall	F1-Score	Support
0	0.71	0.83	0.77	6
1	1.00	0.67	0.80	6
2	1.00	0.83	0.91	6
3	0.62	0.83	0.71	6
Accuracy			0.79	24
Macro Avg	0.83	0.79	0.80	24
Weighted Avg	0.83	0.79	0.80	24



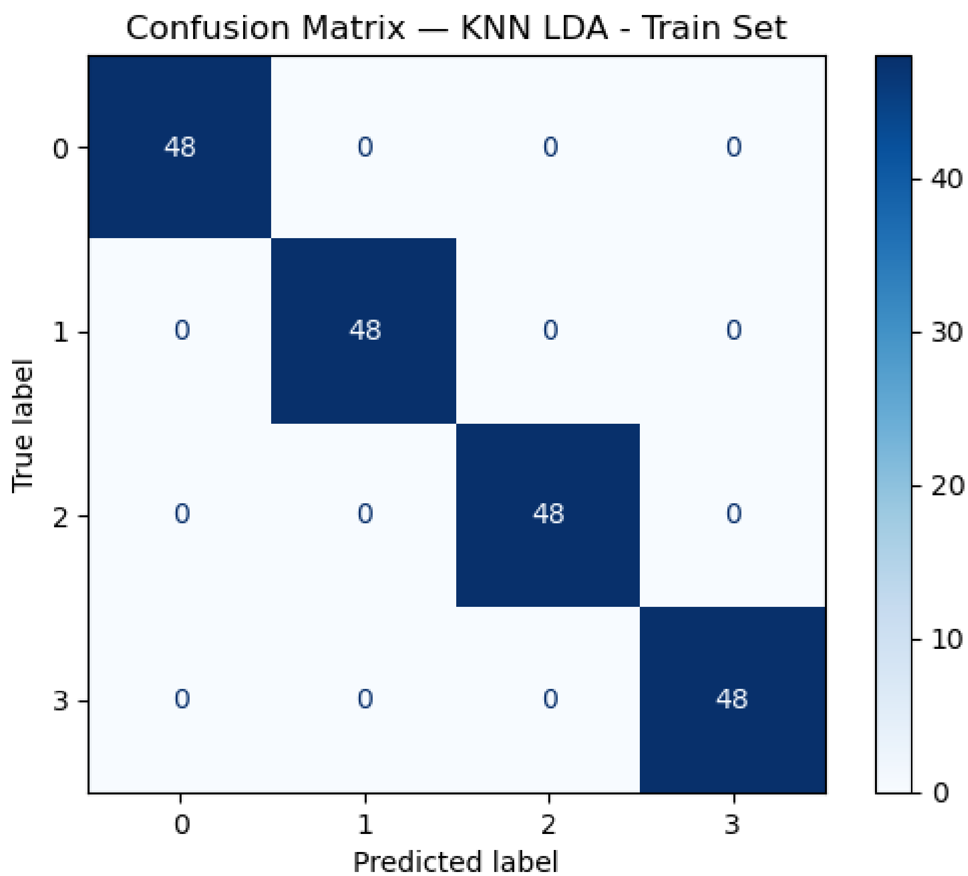
5.1.2.6 KNN Test Set Classification Report (PCA Feature)

Class	Precision	Recall	F1-Score	Support
0	1.00	1.00	1.00	6
1	1.00	1.00	1.00	6
2	0.83	0.83	0.83	6
3	0.83	0.83	0.83	6
Accuracy			0.92	24
Macro Avg	0.92	0.92	0.92	24
Weighted Avg	0.92	0.92	0.92	24



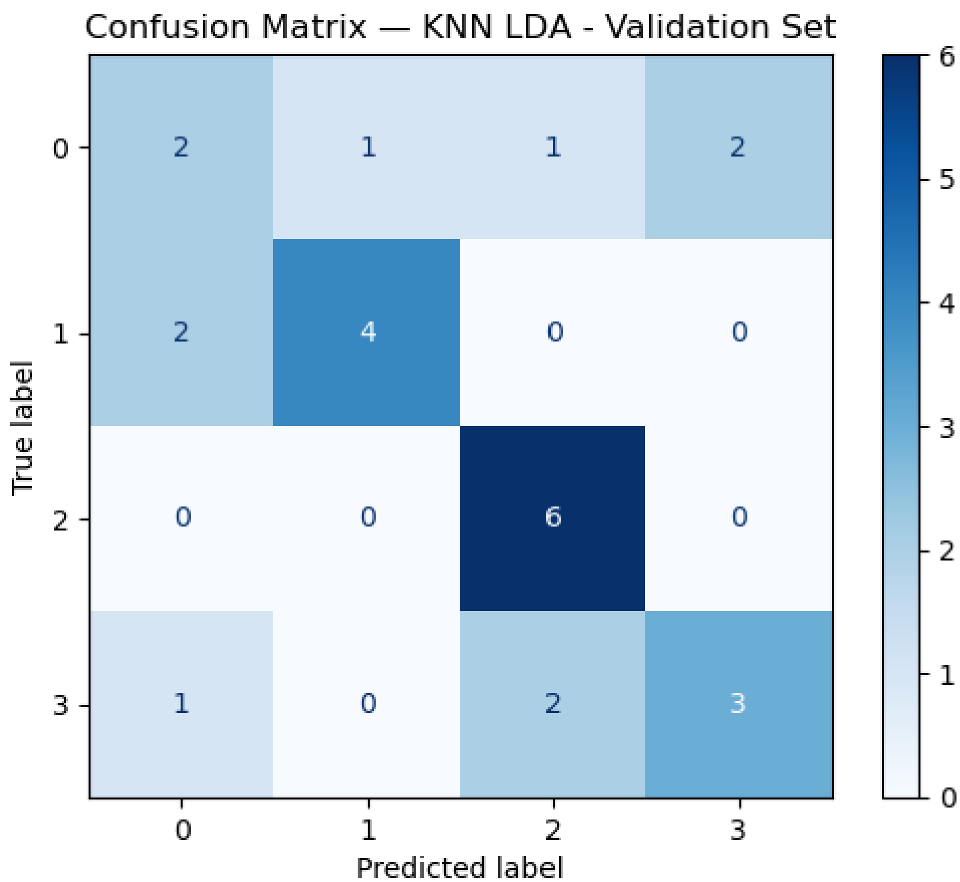
5.1.2.7 KNN Train Set Classification Report (LDA Feature)

Class	Precision	Recall	F1-Score	Support
0	1.00	1.00	1.00	48
1	1.00	1.00	1.00	48
2	1.00	1.00	1.00	48
3	1.00	1.00	1.00	48
Accuracy			1.00	192
Macro Avg	1.00	1.00	1.00	192
Weighted Avg	1.00	1.00	1.00	192



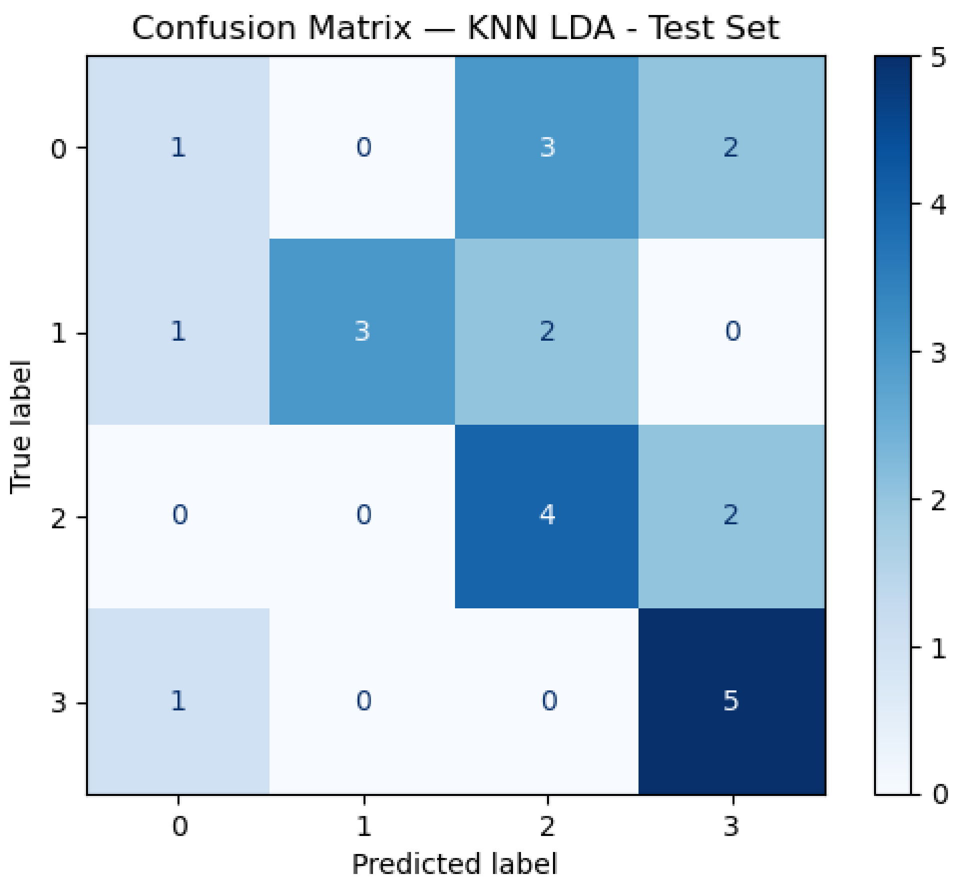
5.1.2.8 KNN Validation Set Classification Report (LDA Feature)

Class	Precision	Recall	F1-Score	Support
0	0.40	0.33	0.36	6
1	0.80	0.67	0.73	6
2	0.67	1.00	0.80	6
3	0.60	0.50	0.55	6
Accuracy			0.62	24
Macro Avg	0.62	0.62	0.61	24
Weighted Avg	0.62	0.62	0.61	24



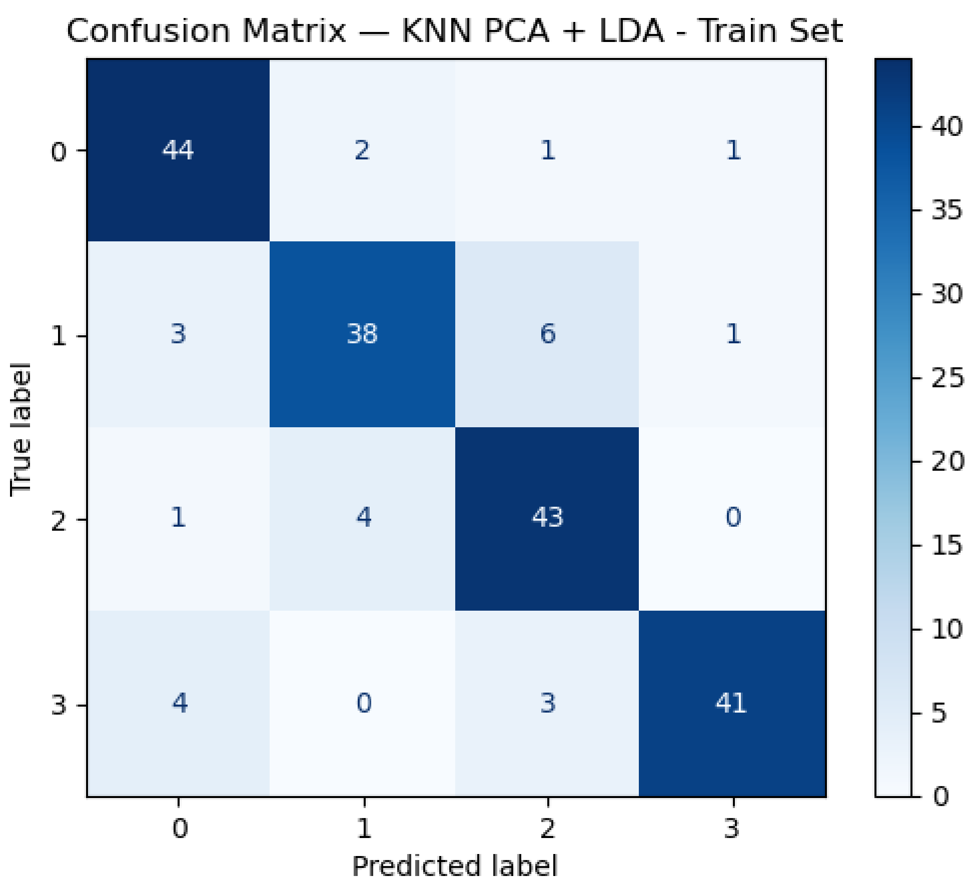
5.1.2.9 KNN Test Set Classification Report (LDA Feature)

Class	Precision	Recall	F1-Score	Support
0	0.33	0.17	0.22	6
1	1.00	0.50	0.67	6
2	0.44	0.67	0.53	6
3	0.56	0.83	0.67	6
Accuracy			0.54	24
Macro Avg	0.58	0.54	0.52	24
Weighted Avg	0.58	0.54	0.52	24



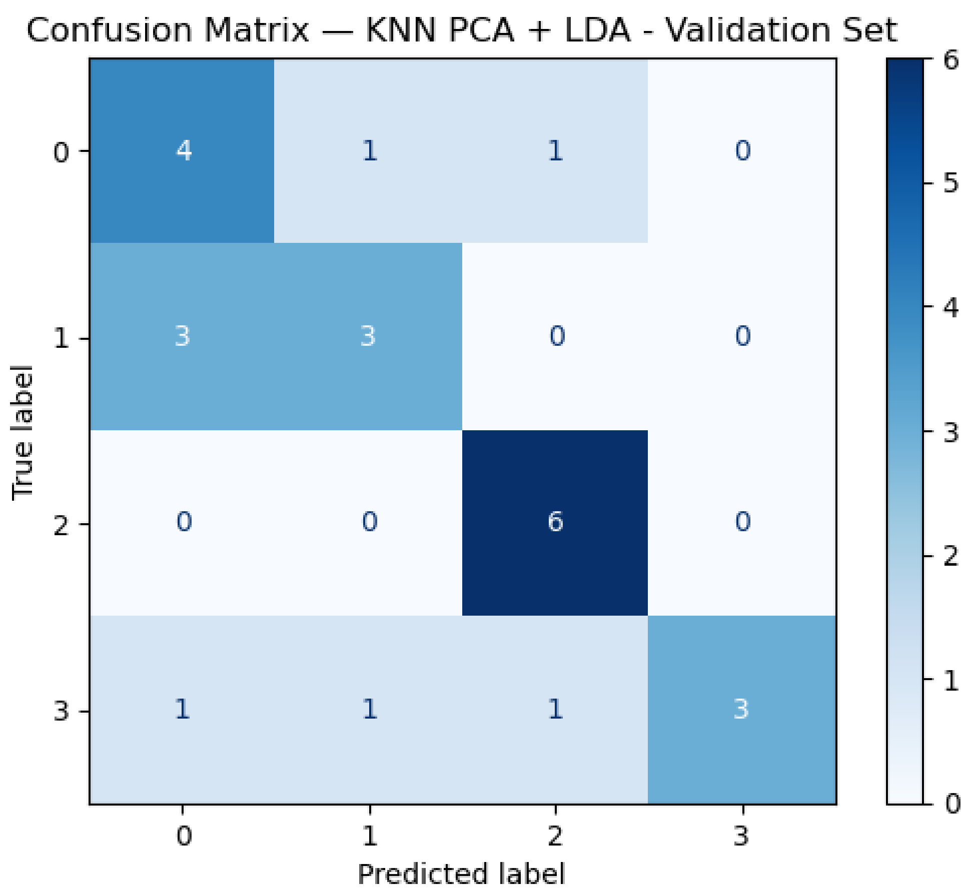
5.1.2.10 KNN Train Set Classification Report (PCA + LDA Feature)

Class	Precision	Recall	F1-Score	Support
0	0.85	0.92	0.88	48
1	0.86	0.79	0.83	48
2	0.81	0.90	0.85	48
3	0.95	0.85	0.90	48
Accuracy			0.86	192
Macro Avg	0.87	0.86	0.86	192
Weighted Avg	0.87	0.86	0.86	192



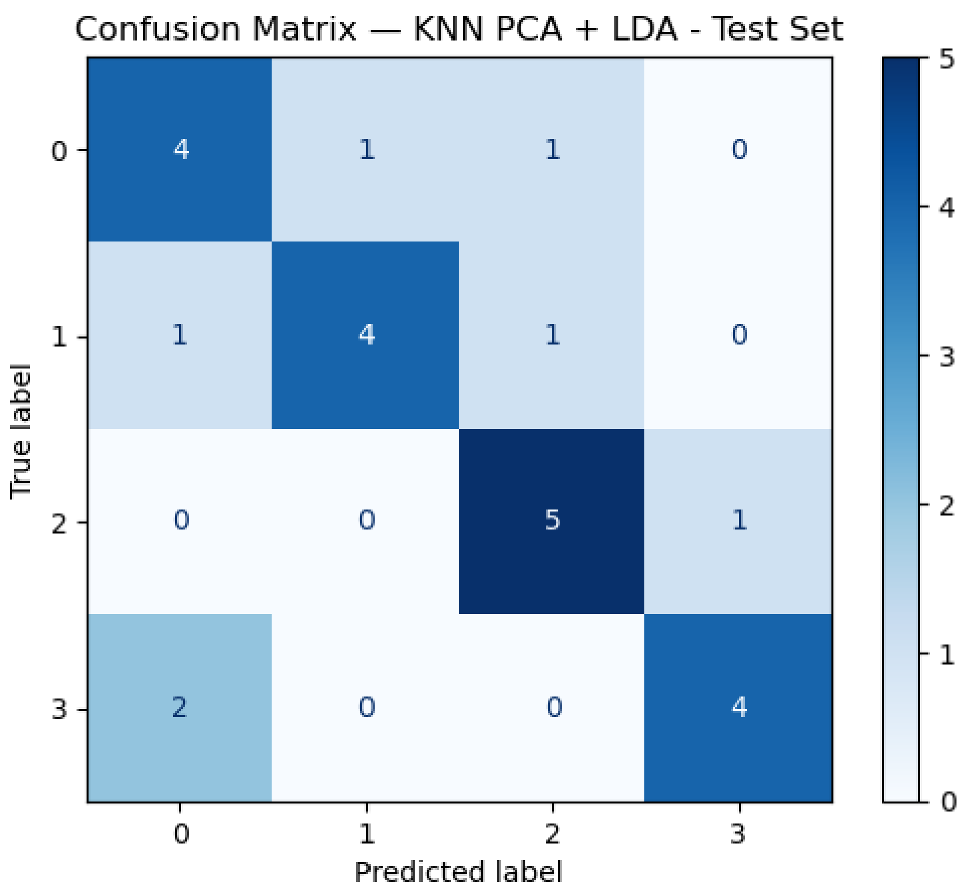
5.1.2.11 KNN Validation Set Classification Report (PCA + LDA Feature)

Class	Precision	Recall	F1-Score	Support
0	0.50	0.67	0.57	6
1	0.60	0.50	0.55	6
2	0.75	1.00	0.86	6
3	1.00	0.50	0.67	6
Accuracy			0.67	24
Macro Avg	0.71	0.67	0.66	24
Weighted Avg	0.71	0.67	0.66	24



5.1.2.12 KNN Test Set Classification Report (PCA + LDA Feature)

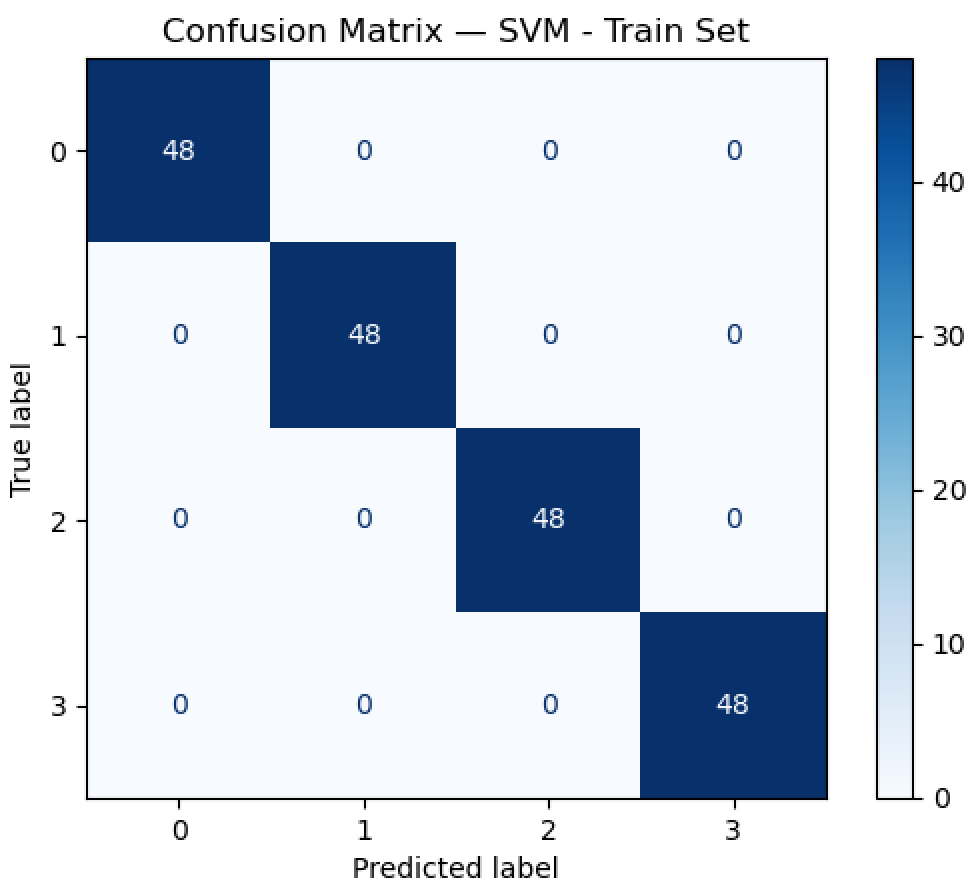
Class	Precision	Recall	F1-Score	Support
0	0.57	0.67	0.62	6
1	0.80	0.67	0.73	6
2	0.71	0.83	0.77	6
3	0.80	0.67	0.73	6
Accuracy			0.71	24
Macro Avg	0.72	0.71	0.71	24
Weighted Avg	0.72	0.71	0.71	24



5.1.2 Performance Evaluation of **SVM**

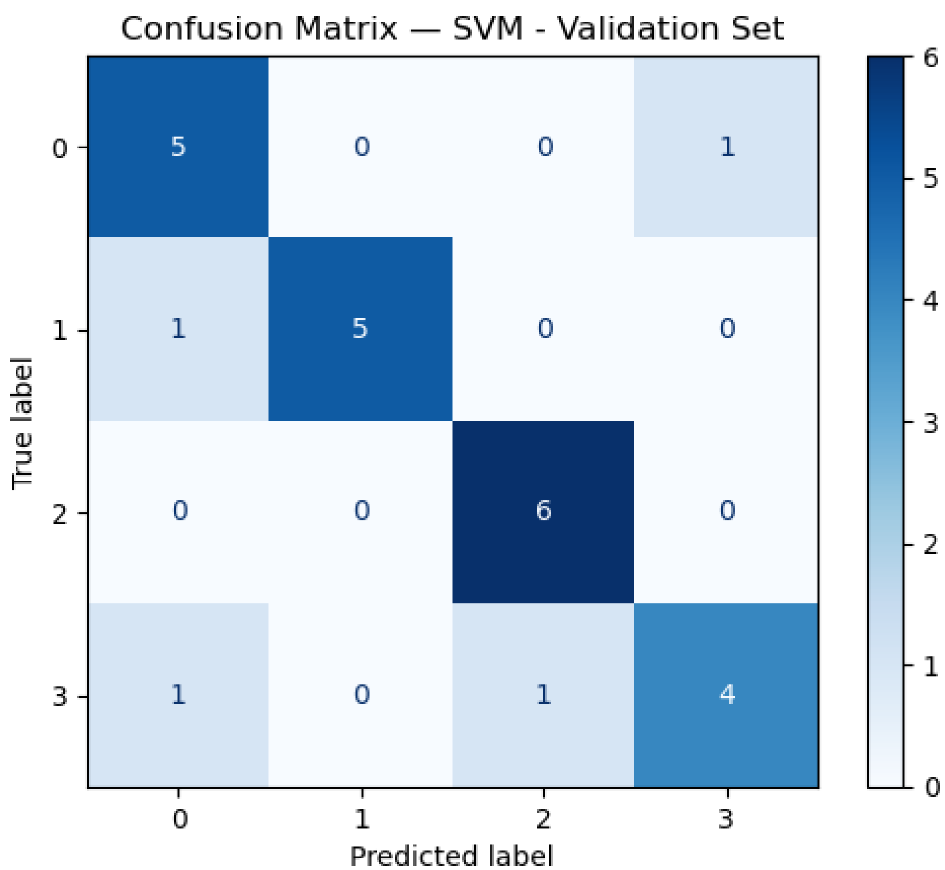
5.1.2.1 SVM Train Set Classification Report (Original Feature)

Class	Precision	Recall	F1-Score	Support
0	1.00	1.00	1.00	48
1	1.00	1.00	1.00	48
2	1.00	1.00	1.00	48
3	1.00	1.00	1.00	48
Accuracy			1.00	192
Macro Avg	1.00	1.00	1.00	192
Weighted Avg	1.00	1.00	1.00	192



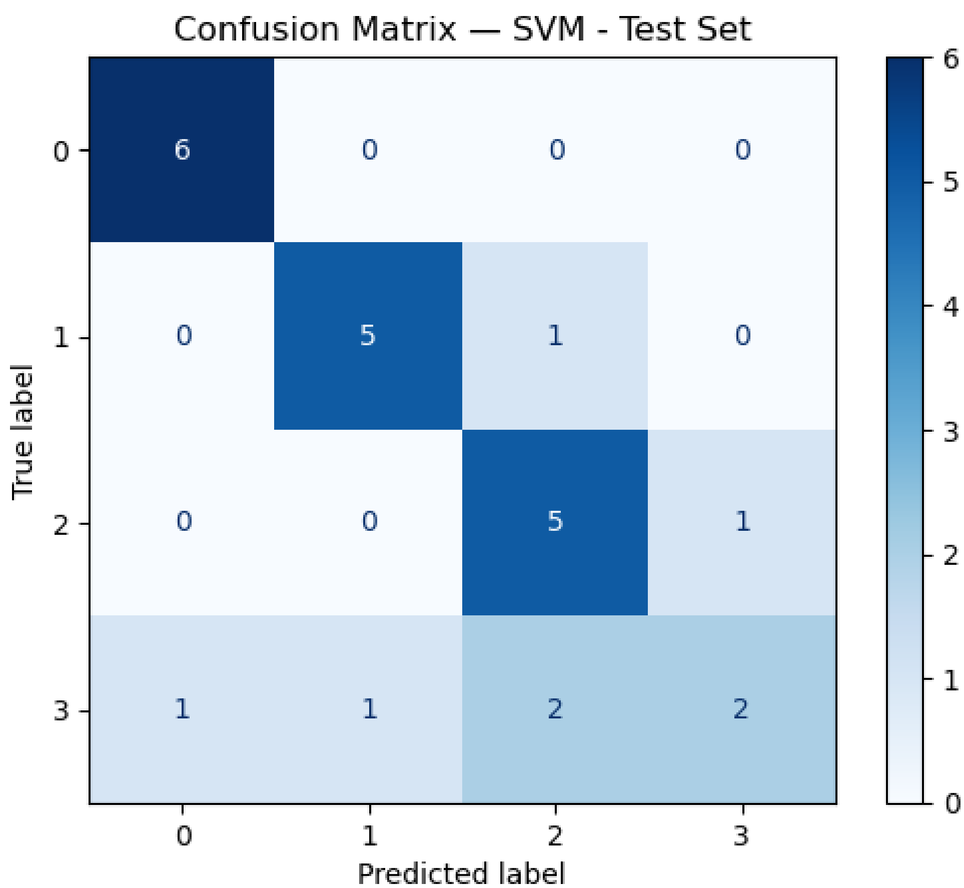
5.1.2.2 SVM Validation Set Classification Report (Original Feature)

Class	Precision	Recall	F1-Score	Support
0	0.71	0.83	0.77	6
1	1.00	0.83	0.91	6
2	0.86	1.00	0.92	6
3	0.80	0.67	0.73	6
Accuracy			0.83	24
Macro Avg	0.84	0.83	0.83	24
Weighted Avg	0.84	0.83	0.83	24



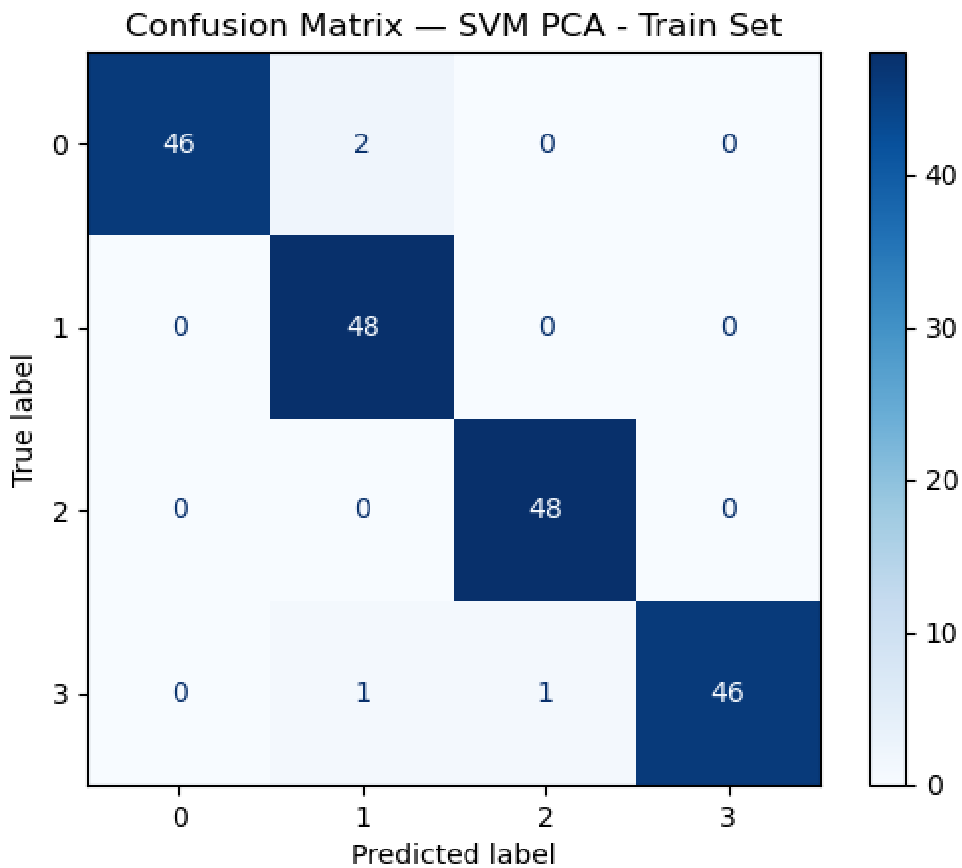
5.1.2.3 SVM Test Set Classification Report (Original Feature)

Class	Precision	Recall	F1-Score	Support
0	0.86	1.00	0.92	6
1	0.83	0.83	0.83	6
2	0.62	0.83	0.71	6
3	0.67	0.33	0.44	6
Accuracy			0.75	24
Macro Avg	0.75	0.75	0.73	24
Weighted Avg	0.75	0.75	0.73	24



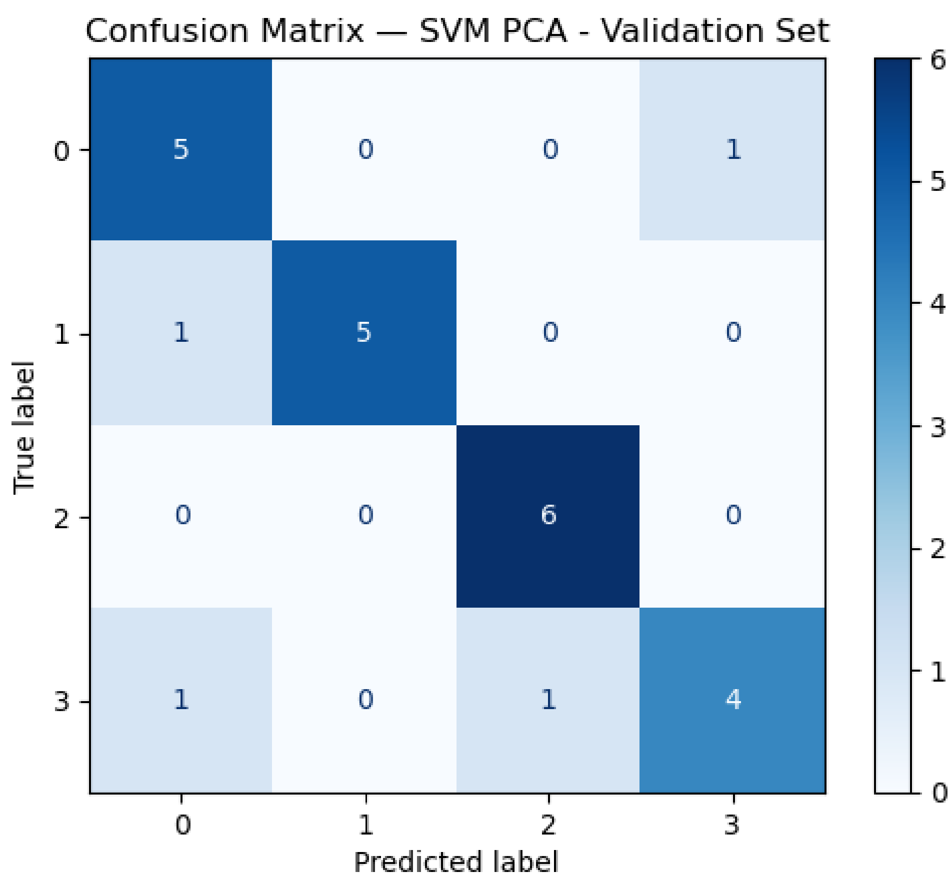
5.1.2.4 SVM Train Set Classification Report (PCA Feature)

Class	Precision	Recall	F1-Score	Support
0	1.00	0.96	0.98	48
1	0.94	1.00	0.97	48
2	0.98	1.00	0.99	48
3	1.00	0.96	0.98	48
Accuracy			0.98	192
Macro Avg	0.98	0.98	0.98	192
Weighted Avg	0.98	0.98	0.98	192



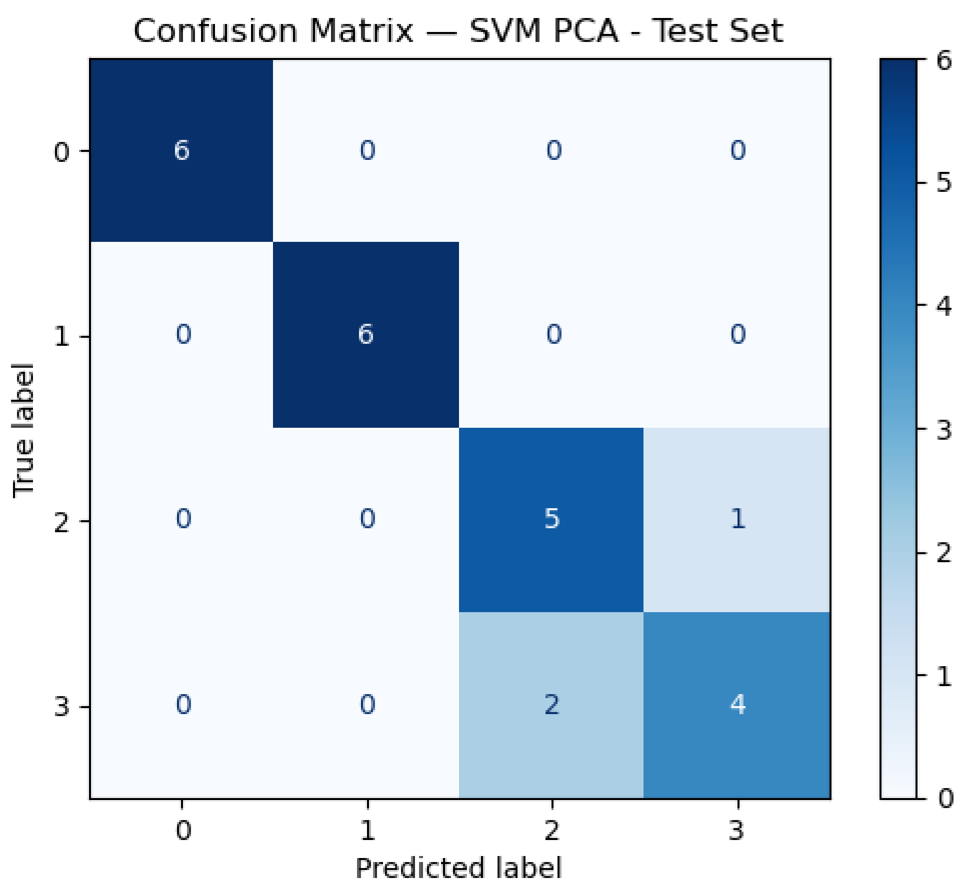
5.1.2.5 SVM Validation Set Classification Report (PCA Feature)

Class	Precision	Recall	F1-Score	Support
0	0.71	0.83	0.77	6
1	1.00	0.83	0.91	6
2	0.86	1.00	0.92	6
3	0.80	0.67	0.73	6
Accuracy			0.83	24
Macro Avg	0.84	0.83	0.83	24
Weighted Avg	0.84	0.83	0.83	24



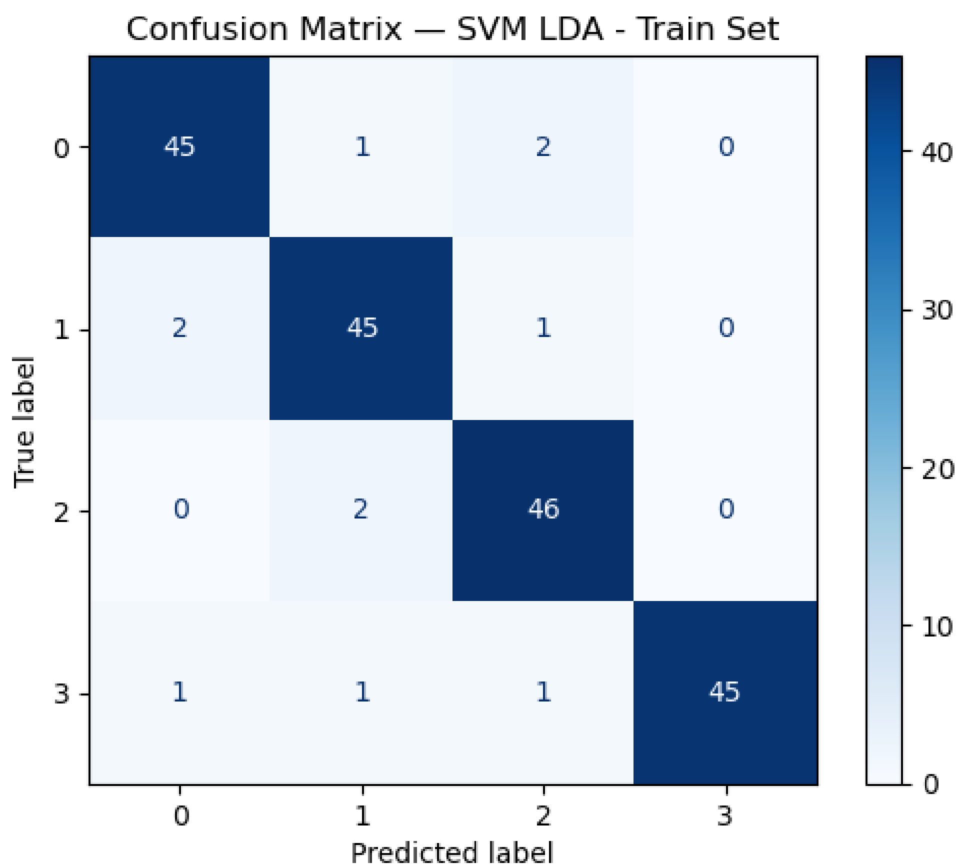
5.1.2.6 SVM Test Set Classification Report (PCA Feature)

Class	Precision	Recall	F1-Score	Support
0	1.00	1.00	1.00	6
1	1.00	1.00	1.00	6
2	0.71	0.83	0.77	6
3	0.80	0.67	0.73	6
Accuracy			0.88	24
Macro Avg	0.88	0.88	0.87	24
Weighted Avg	0.88	0.88	0.87	24



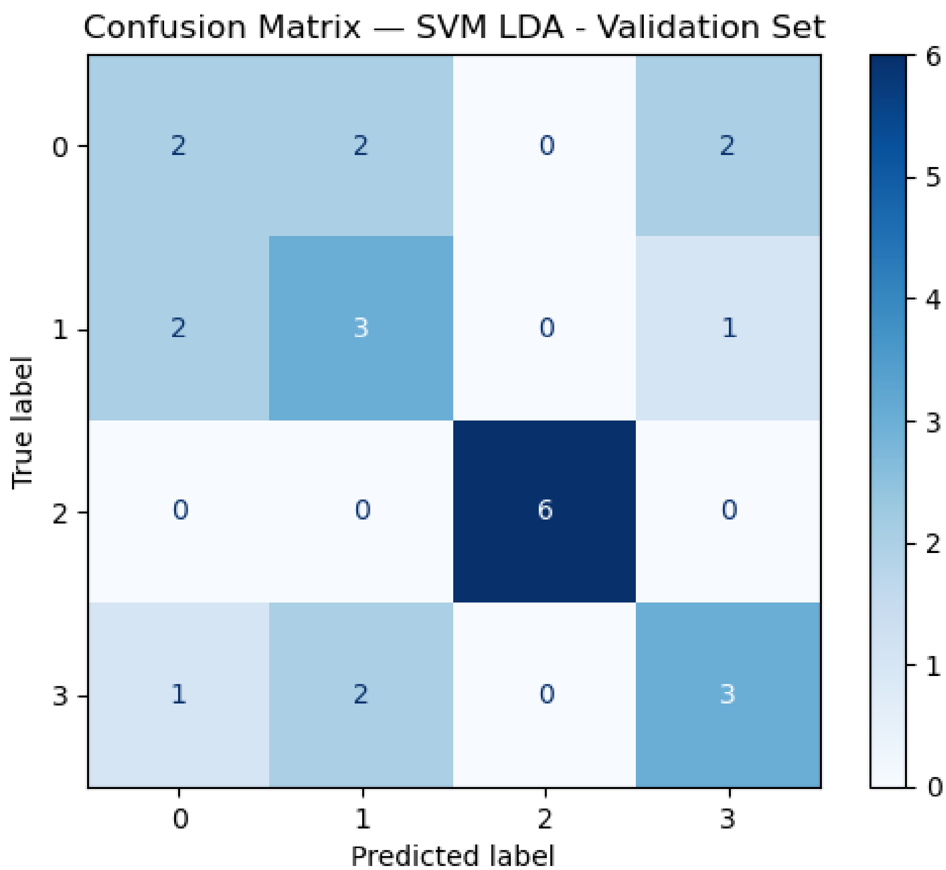
5.1.2.7 SVM Train Set Classification Report (LDA Feature)

Class	Precision	Recall	F1-Score	Support
0	0.94	0.94	0.94	48
1	0.92	0.94	0.93	48
2	0.92	0.96	0.94	48
3	1.00	0.94	0.97	48
Accuracy			0.94	192
Macro Avg	0.94	0.94	0.94	192
Weighted Avg	0.94	0.94	0.94	192



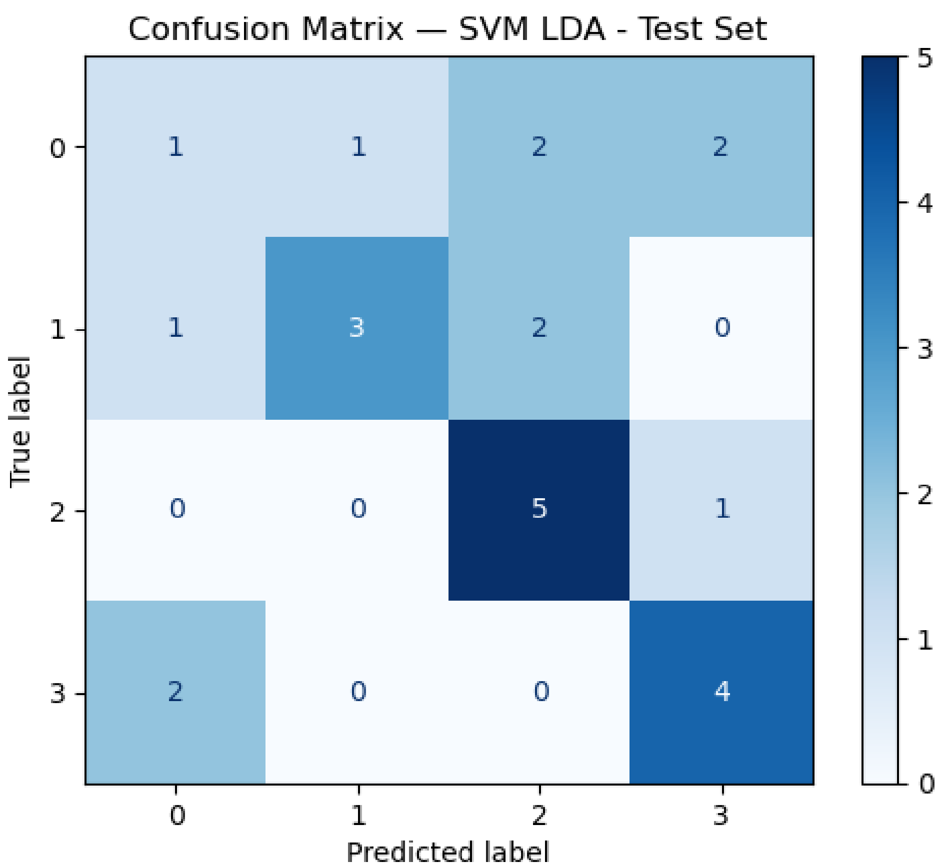
5.1.2.8 SVM Validation Set Classification Report (LDA Feature)

Class	Precision	Recall	F1-Score	Support
0	0.40	0.33	0.36	6
1	0.43	0.50	0.46	6
2	1.00	1.00	1.00	6
3	0.50	0.50	0.50	6
Accuracy			0.58	24
Macro Avg	0.58	0.58	0.58	24
Weighted Avg	0.58	0.58	0.58	24



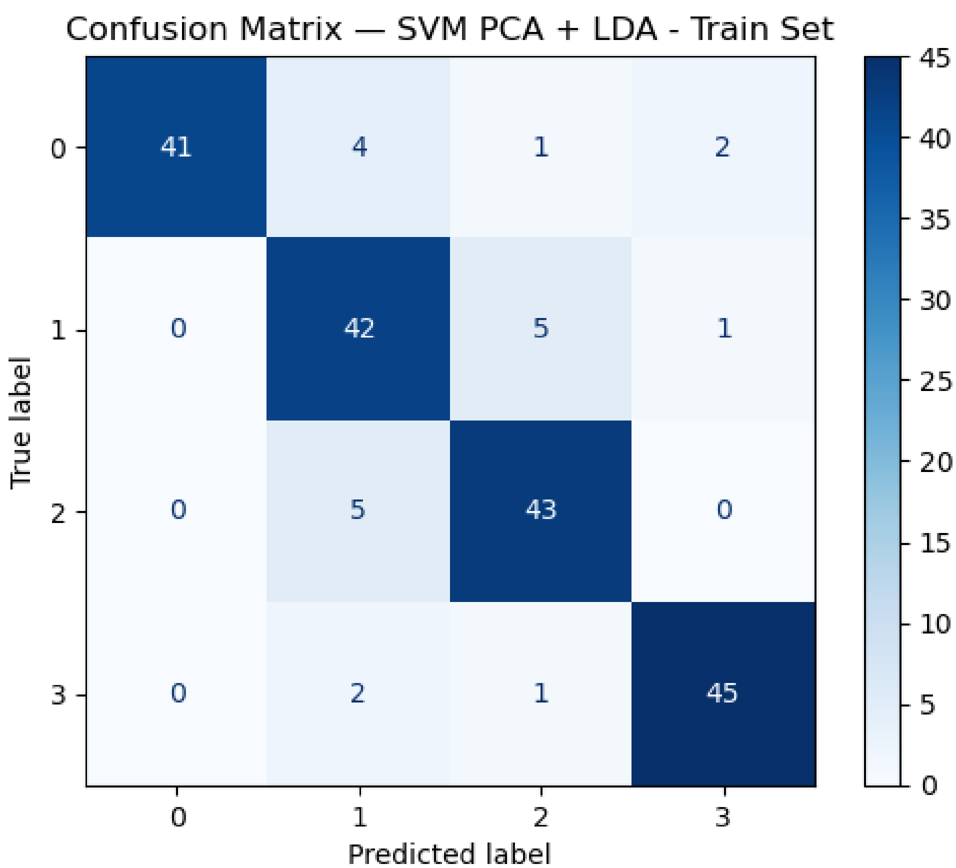
5.1.2.9 SVM Test Set Classification Report (LDA Feature)

Class	Precision	Recall	F1-Score	Support
0	0.25	0.17	0.20	6
1	0.75	0.50	0.60	6
2	0.56	0.83	0.67	6
3	0.57	0.67	0.62	6
Accuracy			0.54	24
Macro Avg	0.53	0.54	0.52	24
Weighted Avg	0.53	0.54	0.52	24



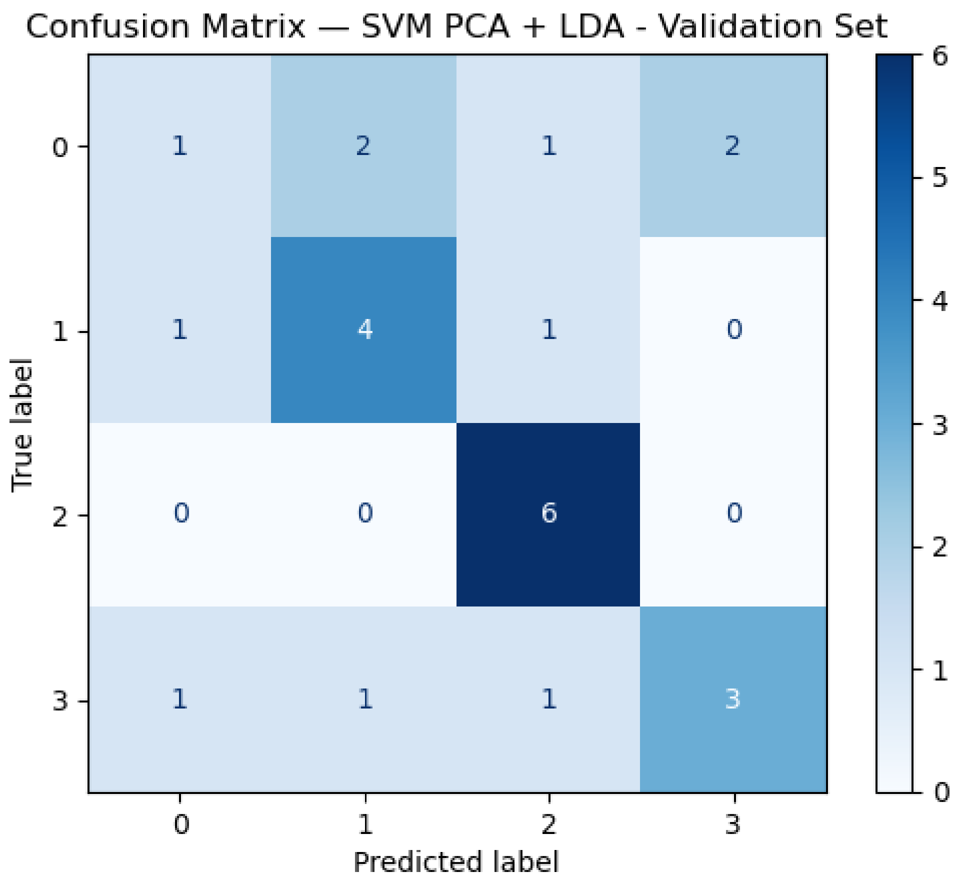
5.1.2.10 SVM Train Set Classification Report (LDA+PCA Feature)

Class	Precision	Recall	F1-Score	Support
0	1.00	0.85	0.92	48
1	0.79	0.88	0.83	48
2	0.86	0.90	0.88	48
3	0.94	0.94	0.94	48
Accuracy			0.89	192
Macro Avg	0.90	0.89	0.89	192
Weighted Avg	0.90	0.89	0.89	192



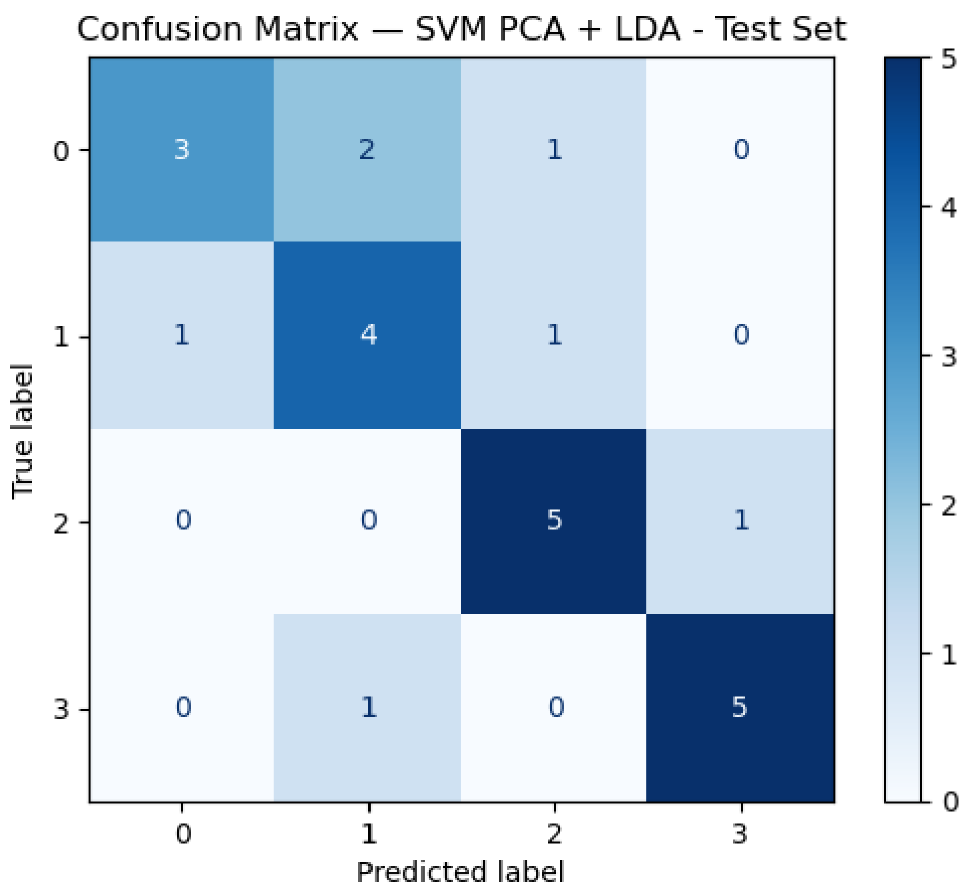
5.1.2.11 SVM Validation Set Classification Report (LDA+PCA Feature)

Class	Precision	Recall	F1-Score	Support
0	0.33	0.17	0.22	6
1	0.57	0.67	0.62	6
2	0.67	1.00	0.80	6
3	0.60	0.50	0.55	6
Accuracy			0.58	24
Macro Avg	0.54	0.58	0.55	24
Weighted Avg	0.54	0.58	0.55	24



5.1.2.12 SVM Test Set Classification Report (LDA+PCA Feature)

Class	Precision	Recall	F1-Score	Support
0	0.75	0.50	0.60	6
1	0.57	0.67	0.62	6
2	0.71	0.83	0.77	6
3	0.83	0.83	0.83	6
Accuracy			0.71	24
Macro Avg	0.72	0.71	0.70	24
Weighted Avg	0.72	0.71	0.70	24



6.0 Discussion and conclusions

Through this project, we have gained valuable hands-on experience in implementing a real-world pattern recognition system using machine learning techniques. One of the most important lessons that we have learned about was the importance of proper data collection and preprocessing. From the beginning, we gathered over 1300 images of four different types of bridge and with different angles but through filtering and standardization, we refined the dataset to a manageable and high-quality collection of 240 images. This process taught us that data quality often has a more significant impact on model performance than the model complexity itself. In this process, we learned to identify and remove blurry and redundant images and convert them to a uniform size and format which can greatly improve model training efficiency.

In terms of methodology, we explored and compared multiple classification algorithms such as K-Nearest Neighbors (KNN) and Support Vector Machine (SVM) by using dimensionality reduction techniques like Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA). This allowed us to understand about different models will perform under the various feature extraction strategies. For example, KNN combined with PCA achieved excellent results especially on the test set with a classification accuracy of up to 92%. We also observed that using raw features without dimensionality reduction could lead to overfitting or longer processing times. These comparisons helped us to appreciate the trade-offs between simplicity, accuracy and computational cost when designing machine learning pipelines.

Moreover, the experimentation phase gave us the insight into model evaluation using metrics like precision, recall, F1-score and accuracy. We learned how to interpret classification reports and make informed decisions about model performance. It was evident that PCA often helped retain essential features while simplifying the input data and LDA improved class separation in certain cases. However, we also realized that, with LDA alone, it might not always outperform PCA, depending on the dataset distribution. These findings helped us understand how feature space transformation can affect classification results.

Building the graphical user interface (GUI) for our bridge recognition system also taught us how to integrate back-end machine learning models with front-end user interaction. We also had made the system user-friendly and efficient by saving trained models and allowing real-time classification through the GUI. This aspect of the project emphasized the practical importance of system design in delivering AI-based applications that are accessible to users.

In conclusion, this project has enhanced our knowledge of pattern recognition and machine learning by allowing us to apply theory in a meaningful context. We also gained experience in data collection, preprocessing, feature extraction, model training, evaluation and system integration. We also learned the importance of experimentation, optimization and clear documentation. Most importantly, we developed critical thinking and problem-solving skills that are essential for working with real-world data. This project has prepared us to handle more complex AI tasks in the future, especially those related to image classification, infrastructure monitoring, or intelligent systems design.

7.0 References

[1] Fu, Y., Xing, K., Huang, Y., & Xiao, Y. (2009). Recognition of Bridge over Water in High-Resolution Remote Sensing Images. 2009 WRI World Congress on Computer Science and Information Engineering, 621–625. https://doi.org/10.1109/csie.2009.2

[2] Wu, F., Wang, C., Zhang, H., Zhang, B., & Zhang, W.-S. (2006). Knowledge-based bridge recognition in high resolution optical imagery. 28(4), 587–591. https://www.researchgate.net/publication/291739383_Knowledge-based_bridge_recognition_in_high_resolution_optical_imagery

[3] Kim, H., Narazaki, Y., & Spencer Jr., B. F. (2023). Automated bridge component recognition using close-range images from unmanned aerial vehicles. Engineering Structures, 274, 115184. https://doi.org/10.1016/j.engstruct.2022.115184

[4] Wang, J., Liu, H., Han, Z., & Wang, Y. (2023). Automatic identification method of bridge structure damage area based on digital image. Scientific Reports, 13(1). https://doi.org/10.1038/s41598-023-39740-z

[5] Zhu, M. (2006). A NewWay for the Detection and Recognition of Bridges. 2006 8th International Conference on Signal Processing. https://doi.org/10.1109/icosp.2006.345676

8.0 Task declaration and distribution form

FACULTY OF INFORMATION SCIENCE & TECHNOLOGY

Project Declaration Form

TPR6223 Pattern Recognition

Trimester March/April 2025 (Term 2510)

No.	Student ID	Student name	Programme	Lab Section
1	1211108363	SIA POH XIANG	AI	1A
2	1211108357	ONG QI REN	AI	1A
3	1211111767	TANG JUN HONG	AI	1A
4	1211108626	LOH JIE HAO	AI	1A

Declaration by Group Leader

I hereby declare that all group members' names are correctly included in the above section. I hold a copy of this assignment which I can produce if the original is lost or damaged. I certify that not part of this assignment has been copied from any other student's work or from any other source except where due acknowledgement is made in the assignment/project/etc.

Group Leader's Signature :

Group Leader's Name : SIA POH XIANG

Group Leader's ID : 1211108363

Group Member's Declaration

(Each group member, including the group leader, must individually fill up and submit this form. This form has to be attached together with submission.)

Group member's name : SIA POH XIANG

Student ID : 1211108363

For the purpose of completing this assignment, I have performed the following tasks:

1. I organize all meetings for the team.
2. I assign tasks to team members.
3. I complete part of the report.
4. I manage the training dataset and handle debugging.
5. I check the work of team members.
6. I make sure all tasks are completed and submitted on time

I hereby declare that I have assessed the final submission and I take full responsibility should there be any inaccuracies, incompleteness, omissions, delays or non-submission.

Group member's signature :

Group member's name : SIA POH XIANG

Group member's ID : 1211108363

Date : 23/6/2025

Group Member's Declaration

(Each group member, including the group leader, must individually fill up and submit this form. This form has to be attached together with submission.)

Group member's name : ONG QI REN

Student ID : 1211108357

For the purpose of completing this assignment, I have performed the following tasks:

In the role of project team member, I will participate in every meeting with the, as well as bringing ideas, to ensure we are working together collaboratively. I will contribute to coding, that means, I will be supporting the development and debugging of the program and scripts we need to deliver as part of the project. I have been requested to coordinate and provide transport for the data collection to assist with capturing the dataset as efficiently and quickly as possible. Finally, the write-up of the report sections from 1.0 to 4.0 is my responsibility. This means that all the project information (objectives, scope, methods, results and analysis) will be comprehensively recorded.

I hereby declare that I have assessed the final submission and I take full responsibility should there be any inaccuracies, incompleteness, omissions, delays or non-submission.

Group member's signature : 

Group member's name : ONG QI REN

Group member's ID : 1211108357

Date : 23/6/2025

Group Member's Declaration

(Each group member, including the group leader, must individually fill up and submit this form. This form has to be attached together with submission.)

Group member's name : TANG JUN HONG

Student ID : 1211111767

For the purpose of completing this assignment, I have performed the following tasks:

1. I attend all the meeting and collect our dataset every week
2. I had chosen the feature extraction and classification as well as train the model.
3. I had created and designed our interface
4. I had done the flow chart for our report
5. I had participated in presentation
6. I had submitted this project on time

I hereby declare that I have assessed the final submission and I take full responsibility should there be any inaccuracies, incompleteness, omissions, delays or non-submission.

Group member's signature : 

Group member's name : TANG JUN HONG

Group member's ID : 1211111767

Date : 23/6/2025

Group Member's Declaration

(Each group member, including the group leader, must individually fill up and submit this form. This form has to be attached together with submission.)

Group member's name : LOH JIE HAO

Student ID : 1211108626

For the purpose of completing this assignment, I have performed the following tasks:

1. I have taken part in group discussion
2. I have taken part in completed the documentation
3. I have taken part in completed presentation slide
4. I have taken part in video presentation
5. I have taken part in completed the system

I hereby declare that I have assessed the final submission and I take full responsibility should there be any inaccuracies, incompleteness, omissions, delays or non-submission.

Group member's signature : ________

Group member's name : LOH JIE HAO

Group member's ID : 1211108626

Date : 23/6/2025