Classification of Caenorhabditis Elegans Genotypes using Locomotory Behavioural Patterns

by

Samuel Apigi Ikirigo

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the
Faculty of Physical Sciences and Engineering
School of Electronics and Computer Science

24th February 2022
Understanding the neural basis of decision making is a major challenge in many disciplines. One way to study the behaviour of a model organism is through their movement patterns in a well-defined environment. The *Caenorhabditis elegans* (*C. elegans*) is an example of a model organism used for many biological investigation. *C. elegans* often moves from one location to another especially in search of food or due to environmental changes or threat. As *C. elegans* moves, the changes that occurs in their body shapes provide information that could aid in understanding the worm’s locomotory activity. In this work, we have built a computational model that automates the retrieval and processing of features from the skeleton body shapes of the worms in a video sequence. These measurements are used to quantify and classify between a wild-type worm (AQ2947) and different classes of mutant worms (OW939, OW940, OW949, OW953 and OW956). In the previous works, the worm genotypes classification was done using features extracted from the worm’s posture phenotypes as input to classifiers such as random forest, decision tree and a deep classification model. However, it is not clear if these methods provide a compact representation of the worm’s movement activity. Here, we used both supervised and unsupervised dimensionality reduction methods such as Principal Component Analysis (PCA), Non-negative Factorisation Matrix (NMF), Linear Discriminant Analysis (LDA), Kernel Principal Component Analysis (KPCA), and Kernel Fisher Linear Discriminant Analysis (KLDA) to extract low-dimensional representations of the skeleton angle data derived from each image frame in a movie. These low-dimensional features serves as input to standard machine learning algorithms such as $k$-nearest neighbour ($k$NN), random forest (RF), and support vectors machine (SVM), and a deep classification model. The outcome of this investigation shows that postural features of the worms retrieved over a period of time can be use to classify their genotypes.
Contents

Nomenclature xix
Abbreviations xx
DECLARATION OF AUTHORSHIP xxv
Acknowledgements xxv

1 Introduction 1
  1.1 Background .................................................. 1
  1.2 The research goal and questions .............................. 2
  1.3 Contribution .................................................. 3
  1.4 Report structure ............................................. 5

2 Background and Methods 7
  2.1 Introduction .................................................. 7
  2.2 Background ................................................... 7
    2.2.1 C. elegans as model organism .......................... 8
    2.2.2 C. elegans phenotypic features ......................... 9
    2.2.3 Extraction of phenotypic features from C. elegans images 9
  2.3 Methods ...................................................... 10
    2.3.1 The machine-learning classification algorithms .......... 10
      2.3.1.1 Support Vector Machines ........................... 11
      2.3.1.2 Random Forest .................................... 11
      2.3.1.3 k-Nearest Neighbour ............................... 12
      2.3.1.4 Deep learning-based algorithms ................... 13
    2.3.2 Evaluation metrics ...................................... 14
      2.3.2.1 Confusion matrix ................................... 15
      2.3.2.2 Receiver Operating Characteristics curve ......... 16
      2.3.2.3 t-SNE ............................................. 17
      2.3.2.4 Mean Absolute Error ................................ 18
      2.3.2.5 Root Mean Squared Error ........................... 18
    2.3.3 Dimensionality Reduction Methods ...................... 19
      2.3.3.1 Principal Component Analysis ....................... 19
      2.3.3.2 Non-negative Matrix Factorization ................ 20
      2.3.3.3 Kernel Principal Component Analysis .............. 21
      2.3.3.4 Linear Discriminant Analysis ...................... 22
      2.3.3.5 Kernel Fisher Linear Discriminant Analysis ....... 24
## CONTENTS

2.4 Summary .......................................................... 25

3 Literature Review .................................................. 27
3.1 Tracking system requirements .................................. 27
3.2 Worm Trackers ....................................................... 27
3.3 Worm Detection and Tracking Algorithm ....................... 28
   3.3.1 Object tracking procedure .................................. 30
   3.3.2 Image processing .............................................. 32
3.4 Feature Extraction Techniques .................................. 35
3.5 Worm classification techniques .................................. 43
3.6 Summary ............................................................ 45

4 Video Frames Preprocessing ......................................... 49
4.1 Introduction ........................................................ 49
4.2 Worm videos collection .......................................... 50
4.3 Image object detection ............................................ 52
   4.3.1 Canny edge detection ....................................... 52
4.4 Image object segmentation ....................................... 53
   4.4.1 Otsu Thresholding Algorithm ............................... 55
   4.4.2 Niblack Thresholding Algorithm ............................ 58
   4.4.3 Sauvola Thresholding Algorithm ............................ 58
   4.4.4 Adaptive Thresholding Algorithm ........................... 58
   4.4.5 Kapur Thresholding Algorithm .............................. 59
   4.4.6 Performance measures for the image binarisation methods ........................................ 60
   4.4.7 Recall, Precision, and F-measure metrics .................. 61
4.4.8 $MSE$, $PERR$, $SNR$ and $PSNR$ metrics .................... 61
4.5 Image Object Skeletonization ..................................... 63
   4.5.1 Iterative thinning algorithm ................................. 64
      4.5.1.1 Zhang and Suen Algorithm .............................. 64
      4.5.1.2 Guo and Hall Algorithm ............................... 66
      4.5.1.3 Stentiford Thinning Algorithm .......................... 68
      4.5.1.4 Hilditch Thinning Algorithm ........................... 69
   4.5.2 Non-iterative thinning algorithm ............................ 72
4.5.3 Thinning algorithm performance measurement ............... 72
   4.5.3.1 Skeletonisation challenges ............................... 74
4.5.4 Spline generation and features collection ................... 75
4.6 Summary ............................................................ 76

5 C. Elegans Phenotypic Feature Extraction and Classification ....... 79
5.1 Introduction ........................................................ 79
5.2 High-dimensional Skeleton Angles Extraction ................... 80
5.3 Dimensionality Reduction Techniques ........................... 82
   5.3.1 Dealing with dropped unsegmented or unskeletonised frames ........................................ 82
   5.3.2 To confirm the number of dropped frames due to failed segmentation and skeletonisation .......... 83
5.3.3 Data Imputation .................................................. 84
   5.3.3.1 Linear imputation ........................................ 85
Appendix G  Grid-search  177
  G.1 PCA gridsearch  ............................................................................. 177
  G.2 NMF grid-search ........................................................................ 180
  G.3 KPCA grid-search ....................................................................... 183
  G.4 KPCA grid-search ....................................................................... 184
  G.5 KPCA grid-search ....................................................................... 185
  G.6 LDA grid-search ......................................................................... 186
  G.7 KLDA Gridsearch ....................................................................... 189

Appendix H  3D plots of the low-dimensional skeleton angles embeddings  193
  H.1 NMF 3D plots ............................................................................. 194
  H.2 KPCA 3D plots ........................................................................... 195
  H.3 LDA 3D plots ............................................................................. 196
  H.4 KLDA 3D plots ........................................................................... 197

Appendix I  Low-dimensional embeddings projections per worm type  199
  I.1 PCA projections .......................................................................... 200
  I.2 NMF projections .......................................................................... 204
  I.3 KPCA projections ....................................................................... 208
  I.4 LDA Projections .......................................................................... 212
  I.5 KLDA projections ....................................................................... 216

Bibliography  221
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>A simple version of the receiver operating characteristics (ROC) curve. The curve shows the plot of the true positive rate against the false positive rate.</td>
<td>17</td>
</tr>
<tr>
<td>3.1</td>
<td>Worm tracking system (Yemini [2011])</td>
<td>28</td>
</tr>
<tr>
<td>3.2</td>
<td>An overview of a worm tracking procedure (Husson et al. [2013]).</td>
<td>29</td>
</tr>
<tr>
<td>4.1</td>
<td>Pipeline for worm video analysis and classification</td>
<td>50</td>
</tr>
<tr>
<td>4.2</td>
<td>Worm videos split into training and test sets, where k denote the split point, N denote the number of videos per strain and p denote the number of strains in the dataset</td>
<td>51</td>
</tr>
<tr>
<td>4.3</td>
<td>Image object detection using Canny edge detection algorithm</td>
<td>53</td>
</tr>
<tr>
<td>4.4</td>
<td>Extraction of threshold value using Otsu algorithm (Yemini [2011])</td>
<td>57</td>
</tr>
<tr>
<td>4.5</td>
<td>Evaluation of image object segmentation techniques</td>
<td>60</td>
</tr>
<tr>
<td>4.6</td>
<td>Thinning algorithm classification (Harish and Paramjeet [2011])</td>
<td>64</td>
</tr>
<tr>
<td>4.7</td>
<td>A $3 \times 3$ window with nine pixels according to Zhang and Suen</td>
<td>64</td>
</tr>
<tr>
<td>4.8</td>
<td>Pixel considered for deletion during the first scan</td>
<td>66</td>
</tr>
<tr>
<td>4.9</td>
<td>Pixel considered for deletion during the second scan</td>
<td>66</td>
</tr>
<tr>
<td>4.10</td>
<td>The 8 neighbours forms that needs to be satisfied for the deletion of pixel $P$ during odd iterations</td>
<td>67</td>
</tr>
<tr>
<td>4.11</td>
<td>The 8 neighbours forms that needs to be satisfied for the deletion of pixel $P$ during even iterations</td>
<td>67</td>
</tr>
<tr>
<td>4.12</td>
<td>The four $3 \times 3$ templates used to locate pixels for deletion using the Stentiford thinning algorithm. The empty white boxes represent locations in which the colour value of the pixel does not matter or has to be checked</td>
<td>69</td>
</tr>
<tr>
<td>4.13</td>
<td>When not to mark pixel $P$ for deletion using the Hilditch thinning algorithm because it will affect the pattern connectivity. The 1’s represent the foreground or object pixels and the 0’s represent the background pixels</td>
<td>70</td>
</tr>
<tr>
<td>4.14</td>
<td>Image pattern skeletonization using iterative thinning algorithms</td>
<td>71</td>
</tr>
<tr>
<td>4.15</td>
<td>Image pattern skeletonization using non-iterative medial axis transform</td>
<td>73</td>
</tr>
<tr>
<td>4.16</td>
<td>Complete and partial removal of unwanted objects during image object skeletonisation</td>
<td>75</td>
</tr>
<tr>
<td>4.17</td>
<td>The worm skeleton coordinates extraction</td>
<td>76</td>
</tr>
<tr>
<td>5.1</td>
<td>Videos and the count of failed segmented and skeletonised frames</td>
<td>83</td>
</tr>
<tr>
<td>5.2</td>
<td>A worm video and the count of failed segmented and skeletonised frames</td>
<td>84</td>
</tr>
<tr>
<td>5.3</td>
<td>Variance captured by retaining fifteen principal components</td>
<td>88</td>
</tr>
</tbody>
</table>
5.4 Worm shape reconstruction using PCA. The figure at the top represents the original worm image retrieved from a video sequence, the next figure depicts the binary image of the worm, the third figure shows the skeletonised image and the bottom figure depicts the reconstructed worm shape using PCA components.

5.5 Comparison between the reconstruction of a worm shape using high-dimensional raw angles and PCA reconstructed angles.

5.6 Image skeletonization reconstruction using PCA imputation.

5.7 Confusion Matrices for the SVM, $k$NN and RF classifiers based on the angles data with missing points approximated using linear imputation method.

5.8 Confusion Matrices for the SVM, $k$NN and RF classifiers based on the angles data with missing points approximated using spline imputation method.

5.9 The ROC AUC curve for the SVM, $k$NN and RF classifiers using one-versus-all scoring metric based on the angles data, in which the missing angles values are approximated using linear interpolation technique.

5.10 The ROC AUC curve for the SVM, $k$NN and RF classifiers using one-versus-all scoring metric based on the angles data, in which the missing angles values are approximated using spline interpolation technique.

5.11 Classification scores obtained based on each PCA component on the test set.

5.12 Confusion Matrices for the SVM, $k$NN and RF classifiers based on PCA embeddings derived from angle data in which linear interpolation was used to calculate the missing values.

5.13 Confusion Matrices for the SVM, $k$NN and RF classifiers based on PCA embeddings retrieved from angle data in which quadratic spline interpolation was used to compute the missing values.

5.14 The ROC AUC curve for the SVM, $k$NN and RF classifiers using one-versus-all scoring metric based on the PCA embeddings, in which the missing angles values are approximated using linear interpolation technique.

5.15 The ROC AUC curve for the SVM, $k$NN and RF classifiers using one-versus-all scoring metric based on the PCA embeddings, in which the missing angles values are approximated using spline interpolation technique.

5.16 PCA 2D training and test embeddings extracted from the high-dimensional skeleton angles data.

5.17 PCA 3D training and test embeddings extracted from the high-dimensional skeleton angles data.

5.18 t-SNE 2-dimensional embeddings of the PCA training and test features.

5.19 Classification scores obtained based on each NMF component on the test set.

5.20 Confusion Matrices for the SVM, $k$NN and RF classifiers based on NMF embeddings.

5.21 Confusion Matrices for the SVM, $k$NN and RF classifiers based on NMF embeddings.

5.22 The ROC AUC curve for the SVM, $k$NN and RF classifiers using one-versus-all scoring metric based on the NMF embeddings, in which the missing angles values are approximated using linear interpolation technique.

5.23 The ROC AUC curve for the SVM, $k$NN and RF classifiers using one-versus-all scoring metric based on the NMF embeddings, in which the missing angles values are approximated using spline interpolation technique.

5.24 NMF 2D training and test embeddings of the high-dimensional skeleton angles data.

5.25 t-SNE 2D embeddings of the NMF training and test features.
5.26 Classification scores obtained based on each KPCA components on the test set 122
5.27 Confusion Matrices for the SVM, $k$NN and RF classifiers based on KPCA embeddings 123
5.28 Confusion Matrices for the SVM, $k$NN and RF classifiers based on KPCA embeddings 124
5.29 The ROC AUC curve for the SVM, $k$NN and RF classifiers using one-versus-all scoring metric based on the KPCA embeddings, in which the missing angles values are approximated using linear interpolation technique. 125
5.30 The ROC AUC curve for the SVM, $k$NN and RF classifiers using one-versus-all scoring metric based on the KPCA embeddings, in which the missing angles values are approximated using spline interpolation technique. 126
5.31 KPCA 2D training and test embeddings of the high-dimensional skeleton angles data 127
5.32 t-SNE 2D embeddings of the KPCA training and test features 128
6.1 Classification scores obtained based on each LDA component on the test set 132
6.2 Confusion Matrices for the SVM, $k$NN and RF classifiers based on LDA embeddings 134
6.3 Confusion Matrices for the SVM, $k$NN and RF classifiers based on LDA embeddings 135
6.4 The ROC AUC curve for the SVM, $k$NN and RF classifiers using one-versus-all scoring metric based on the LDA embeddings, in which the missing angles values are approximated using linear interpolation technique. 136
6.5 The ROC AUC curve for the SVM, $k$NN and RF classifiers using one-versus-all scoring metric based on the LDA embeddings, in which the missing angles values are approximated using spline interpolation technique. 137
6.6 LDA 2D training and test embeddings of the high-dimensional skeleton angles data 138
6.7 t-SNE 2-dimensional embeddings of the LDA training and test features 139
7.1 Classification accuracy scores achieved as the KLDA test embedding dimension increases 143
7.2 Confusion Matrices for the SVM, $k$NN and RF classifiers based on KLDA embeddings 144
7.3 Confusion Matrices for the SVM, $k$NN and RF classifiers based on KLDA embeddings 145
7.4 The ROC AUC curve for the SVM, $k$NN and RF classifiers using one-versus-all scoring metric based on the KLDA embeddings, in which the missing angles values are approximated using linear interpolation technique. 146
7.5 The ROC AUC curve for the SVM, $k$NN and RF classifiers using one-versus-all scoring metric based on the KLDA embeddings, in which the missing angles values are approximated using spline interpolation technique. 147
7.6 KLDA 2D training and test embeddings of the high-dimensional skeleton angles data 148
7.7 t-SNE 2-dimensional embeddings of the LDA training and test features 149
F.1 Multiple C. elegans detection and tracking in order to quantify and classify food leaving event 174
F.2 Tracked path of the worms 175
| G.1 | PCA Grid-search plots | 177 |
| G.2 | PCA Grid-search plots | 178 |
| G.3 | PCA Grid-search plots | 179 |
| G.4 | NMF Grid-search plots | 180 |
| G.5 | NMF Grid-search plots | 181 |
| G.6 | NMF Grid-search plots | 182 |
| G.7 | KPCA Grid-search plots | 183 |
| G.8 | KPCA Grid-search plots | 184 |
| G.9 | KPCA Grid-search plots | 185 |
| G.10 | LDA Grid-search plots | 186 |
| G.11 | LDA Grid-search plots 2 | 187 |
| G.12 | LDA Grid-search plots | 188 |
| G.13 | KLDA Grid-search plots | 189 |
| G.14 | KLDA Grid-search plots | 190 |
| G.15 | KLDA Grid-search plots | 191 |
| H.1 | NMF 3D training and test embeddings of the high-dimensional skeleton angles data | 194 |
| H.2 | KPCA 3D training and test embeddings of the high-dimensional skeleton angles data | 195 |
| H.3 | LDA 3D training and test embeddings of the high-dimensional skeleton angles data | 196 |
| H.4 | KLDA 3D training and test embeddings of the high-dimensional skeleton angles data | 197 |
| I.1 | PCA training projections per worm class, in which linear interpolation was used to compute the missing frame values | 200 |
| I.2 | PCA training projections per worm class, in which quadratic spline interpolation was used to compute the missing frame values | 201 |
| I.3 | PCA test projections per worm class, in which linear interpolation was used to compute the missing values | 202 |
| I.4 | PCA test projections per worm class, in which quadratic spline interpolation was used to compute the missing values | 203 |
| I.5 | NMF training projections per worm class, with linearly imputed missing frames values | 204 |
| I.6 | NMF training projections per worm class, with quadratic spline imputed missing frames values | 205 |
| I.7 | NMF test projections per worm class, with linearly imputed missing frames values | 206 |
| I.8 | NMF test projections per worm class, with quadratic spline imputed missing frame values | 207 |
| I.9 | KPCA training projections per worm class, in which quadratic linear interpolation was used to compute the missing values | 208 |
| I.10 | KPCA training projections per worm class, in which quadratic spline interpolation was used to compute the missing values | 209 |
| I.11 | KPCA test projections per worm class, in which linear interpolation was used to compute the missing values | 210 |
I.12  KPCA test projections per worm class, in which quadratic spline interpolation was used to compute the missing values ........................... 211
I.13  LDA training projections per worm class, in which linear interpolation was used to compute the missing values ........................... 212
I.14  LDA training projections per worm class, in which quadratic spline interpolation was used to compute the missing values ......................... 213
I.15  LDA test projections per worm class, in which linear interpolation was used to compute the missing values ................................. 214
I.16  LDA test projections per worm class, in which quadratic spline interpolation was used to compute the missing values ......................... 215
I.17  KLDA training projections per worm class, in which linear interpolation was used to compute the missing values .......................... 216
I.18  KLDA training projections per worm class, in which quadratic spline interpolation was used to compute the missing values ................... 217
I.19  KLDA test projections per worm class, in which linear interpolation was used to compute the missing values .............................. 218
I.20  KLDA test projections per worm class, in which quadratic spline interpolation was used to compute the missing values ...................... 219
## List of Tables

<table>
<thead>
<tr>
<th>Table No</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Confusion matrix</td>
<td>15</td>
</tr>
<tr>
<td>2.2</td>
<td>Confusion matrix scores computation</td>
<td>16</td>
</tr>
<tr>
<td>4.1</td>
<td>Performance evaluation scores for image object segmentation techniques</td>
<td>62</td>
</tr>
<tr>
<td>4.2</td>
<td>Evaluation of iterative thinning algorithms</td>
<td>74</td>
</tr>
<tr>
<td>5.1</td>
<td>Missing data imputation using linear and quadratic spline interpolation methods</td>
<td>87</td>
</tr>
<tr>
<td>5.2</td>
<td>Deep model classification accuracy for the skeleton angles with missing values computed using linear or spline interpolation</td>
<td>99</td>
</tr>
<tr>
<td>5.3</td>
<td>Deep model classification accuracy for the skeleton angles with missing values computed using linear and spline interpolation techniques</td>
<td>105</td>
</tr>
<tr>
<td>5.4</td>
<td>Deep model classification accuracy for the NMF test embeddings with missing values computed using linear and spline interpolation techniques</td>
<td>114</td>
</tr>
<tr>
<td>5.5</td>
<td>Deep model classification accuracy for the skeleton angles with missing values computed using linear and spline interpolation methods</td>
<td>122</td>
</tr>
<tr>
<td>6.1</td>
<td>Deep model classification accuracy for the skeleton angles with missing values computed using linear or spline interpolation</td>
<td>133</td>
</tr>
<tr>
<td>7.1</td>
<td>Deep model classification accuracy for the skeleton angles with missing values computed using linear or spline interpolation</td>
<td>143</td>
</tr>
<tr>
<td>F.1</td>
<td>Food leaving event between the N2 and mgl2 worms</td>
<td>173</td>
</tr>
</tbody>
</table>
Listings
Nomenclature

$C$ Covariance matrix
$D$ dimension of $x$
$d$ feature dimension
$\frac{\partial (T(x))}{\partial (x)}$ partial derivative of $T(x)$ with respect $x$
$\lambda_i$ eigenvalue number $i$
$\mu_\mu$ eigenvector number $\mu$
$\alpha$ eigenvector matrix
$I$ identity matrix
$k$ number of classes
$K$ Kernel matrix
$\tilde{K}$ Gram matrix
$K(P||Q)$ Kullback-Leibler divergence between the probability distribution $P$ and $Q$
$\tilde{\mu}_i$ mean of $m$
$\Phi(x)$ map of $x$ to a new feature space
$\mu$ mean vector
$N$ number of samples
$P(i|j)$ conditional probabilities
$P(ij)$ joint probability
$\mathbb{R}^N$ real number of dimensionality
$\mathbb{R}^{M \times N}$ $M$ samples with $N$ dimensions
$\sigma$ variance
$\sum$ summation
$S_w$ within-class scatter matrix
$S_b$ between-class scatter matrix
$S_w^\phi$ within-class scatter matrix in $\phi$ feature space
$S_b^\phi$ between-class scatter matrix in $\phi$ feature space
$T$ threshold
$T_n$ number of images or video frames
$X$ matrix whose column are $x_i$ with $i = 1, \ldots, N$
$x^T$ transpose of $x$
$||x_i - x_j||^2$ pairwise squared Euclidean distance
$\epsilon$ classification error
NOMENCLATURE

\[ \theta \] angle
\[ < \theta > \] mean angle
\[ y(x) \] project of \( x \)
Abbreviations

False Negative (FN)
False Positive (FP)
Independent Component Analysis (ICA)
Kernel Fisher Discriminant Analysis (KFDA)
Kernel Linear Discriminant Analysis (KLDA)
Kernel Principal Component Analysis (KPCA)
Linear Discriminant Analysis (LDA)
Mean Absolute Error (MAE)
Neighborhood Component Analysis (NCA)
One Versus All (OVA)
Principal Component (PC)
Principal Component Analysis (PCA)
Random Forest (RF)
Root Mean Squared Error (RMSE)
Receiver Operating Characteristics (ROC)
Support Vector Machines (SVM)
True Negative (TN)
True Positive (TP)
Vector Quantisation (VQ)
Yellow Fluorescent Protein (YFP)
2-dimensional (2D)
3-dimensional (3D)
DECLARATION OF AUTHORSHIP

I Samuel Apigi Ikirigo declare that this thesis entitled Classification of Caenorhabditis Elegans Genotypes using Locomotory Behavioural Patterns and the work presented in it is my own and has been generated by me as the result of my own original research.
I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;

2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;

3. Where I have consulted the published work of others, this is always clearly attributed;

4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;

5. I have acknowledged all main sources of help;

6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;

7. None of this work has been published before submission;

Signature: .......................... Date: ..........................
Acknowledgements

I thank God Almighty for His wisdom, strength, provision and love throughout my PhD study. His presence kept me going even in difficult times. I specially want to thank my PhD supervisor Dr. Srinandan Dasmahapatra for his meticulous feedback, financial support and patience even when it seems like no progress had been made. To my father Late. Elder Roland Apigi Ikirigo, you will always be in my heart and may you rest in peace. To my mother Mrs. Inibo Mercy Ikirigo, thank you for your love, support and prayers. I wish to thank my brothers, sisters and other family members for their words of encouragement and display of love. Special thanks to the family of Mr. and Mrs. Robert Pool, words cannot express how grateful I am for all you did for me; God bless you richly. Thanks to my agile pastor Dr. John Paul Oddoye and all members of Covenant Church Southampton, I sincerely appreciate your prayers, fellowship and love; God bless you all immensely. My humble thanks goes to all staff and student in ECS, I cherish your support and friendship. I also wish to thank the management team of Federal University of Otuoke and Tertiary Education Trust Fund (TETFund) for your sponsorship, I sincerely appreciate your support. To all my friends thank you so much for the support, I am grateful.

To … God Almighty
Chapter 1

Introduction

1.1 Background

The nematode Caenorhabditis elegans (C. elegans) is a model organism used for behavioural study for a long time, because its genome is well characterised. It consists of 302 neurons and with approximately 7000 synaptic connections (Moy et al. [2015] and White et al. [1986]). Due to the simplicity of its neural architecture, several biological investigations done in the field of neuroscience involve the use of nematode worms (Nah and Baek [2003]). As a result, thousands of articles have been published based on studies performed on them. The motivation for using C. elegans in this study is that, extensive knowledge of the organism’s genes and the ability to intervene in the organism’s physiology has been proven successful in different laboratories across the world. Even though the worm’s genome is well established, the relationship between structure and organismal function is largely lacking. The ability to introduce precise genetic mutation which affect specific parts of neural anatomy and signalling circuitry offers the possibility to make the association between changes in the neural architecture and consequent changes in behaviour. The ability to identify changes in locomotory behaviour correlated to genetic mutants would offer a point of entry into identifying the key functional modules that link genes to neuro-muscular and behavioural function. This study focuses on the characterisation of different strains of C. elegans based on the changes in the body posture of the worm recorded over a period of time in a well established and confined environment. Many studies have shown that, the changes that occurs in the body posture of the worm during movement, contain useful information that can be used to characterise the locomotory behaviour of the worm. This in turn can be used to describe the decision-making process of the worm in the case of a threat or in the presence of food. This is achieved through the quantification and characterisation of the locomotion patterns of the worm using computer vision techniques and machine learning algorithms. In previous works, experienced observers subjectively quantify these behaviours manually with the aid of a microscope (Bendesky et al. [2011] and Nah and Baek [2003]). However, this approach of quantifying the worm’s behaviour is very tedious and contain many challenges that cannot be resolved via manual process,
but the use of automated system. For example, some expert observers subjectively analyse the state of a worm for a short period of time and use that information to categorise the worm type. This method of quantification and classification of the worm is prone to human errors, because some subtle behaviour exhibited by the worm takes a longer time to be seen and if not captured by the observer may affect the overall goal of the study. Another challenge is that, it will be difficult to reproduce the locomotory activities displayed by the worm which led to the observer’s conclusion for further analysis, especially when the investigation is conducted with the use of a microscope only. Finally, this manual approach is time consuming, especially when you have to study the behaviour of large number of strains. By automating most of the steps used in gathering and analysing data from this model organism, we can objectively classify phenotypic behaviour exhibited by \textit{C. elegans} with distinctive genotypes. Therefore, the main goal of this research was based on the quantification and characterisation of the movement patterns between distinctive types of \textit{C. elegans} namely, the wild-type strains (worms with normal locomotion patterns) and the \textit{mutant} strains (worms with abnormal patterns of locomotion). Here, we built a classification model for quantifying and characterising ageing \textit{C. elegans} worms with different genotypes, by utilising techniques derived from computer vision and machine learning. These techniques not only provide meaningful ways of gathering and analysing large datasets, but can be adapted in other scientific applications. The machine learning aspect involves using algorithms to discover meaningful biological patterns in a large dataset, in our case a dataset of worm videos. On the other hand, computer vision is concerned with developing algorithms that enables the computer to process image data similar to the task of the human visual system. In this work, the computer vision techniques are used to retrieve the image frames from the video stream, thus automating the process of extracting useful shape-based features (Geng et al. [2004]).

1.2 The research goal and questions

The main goal of this research is to identify a known phenotype that is linked to specific genetic alteration for the purpose of interpreting complex neural functions involve in the worms movement patterns. We aim to achieve this goal using meaningful low-dimensional features obtained from measurements that describes the locomotive activities of the different strain types over a period of time. Here, we investigate the locomotory activity of nematode worms previously used to study the difference between physiological (normal) and pathological (disease-model) ageing process Martineau et al. [2019]. This is based on the assumption that, at different stages in the worms (mutant) life cycles their movement patterns are affected by the fused genetic materials in them. In the previous study, 171 features derived from 57 core features were used to perform the strain classification problem. However, their result was not quantified but represented using clustering analysis. That we aim to achieve in this work using different learning algorithms and evaluation metrics. Based on the outcome of Stephens et al. [2008] experiment which reveals that with less than six (6) principal components, we can
account for over 95% of the worm shape variance. And this was achieved with 100 tangent angles (posture feature) measured along equally spaced points in the worm’s skeleton body in each image frame of a movie. Then arose the question of what set of features are best suitable for the animal behavioural studies such as the classification of the movement patterns of \textit{C. elegans} strains with different genotypes. This leads to the following research questions that needs to be addressed:

1. What set of features are generally suitable for the study of nematode \textit{C. elegans} behaviour confined to a certain environment?

2. In order to overcome the curse of dimensionality, which low-dimensional features representation methods can best estimate the high-dimensional features retrieved from the images of the single worms tracked over a period of time?

3. Will the chosen low-dimensional feature representation method be applicable to all genotype classification problems and still yield a good classification accuracy?

To address these questions, we performed the \textit{C. elegans} genotype classification problem using unsupervised and supervised learning methods, to extract low-dimensional skeleton angles representations. Thereafter, we compared our results with other related works.

\subsection*{1.3 Contribution}

In this work, we propose a semi-automated system for the study of locomotory activities in \textit{C. elegans} worms. The motivation behind the contribution made in this work was derived from an initial study conducted on multiple \textit{C. elegans} worm videos. We use data gathered and analysed from video records of multiple worms crawling freely on an agar plate that contains food, to classify different worm strains (wild type and mutant). We investigate the food leaving behaviour between the wild type (N2) and mutant (\textit{mgl2}) worms. A food leaving behaviour is defined as an event where the worm’s body is completely out of a marked boundary (in this case a circle) in the food patch. The food leaving behaviour describes the number of times a particular worm will leave the food patch in search of a new food source. The results from our study shows that the wild-type worms travel more in search of other food source compared to mutant worms. Interestingly, the method applied in this work, can eliminate the manual method used in related work done by Bendesky et al. [2011]. Furthermore, this method processes large amounts of video data within a short period. The results from this experiment have provided a quantitative way of analysing food behaviour between these two types of nematode worms. Although this study is not considered as a major part of this report, however brief information about the techniques and results of the food leaving event is found in appendix F of this report. Due to the challenges encountered while trying to extract motion features from the multiple worm videos, we switched to the use of single worm videos in this
work. A major challenge experienced with the multiple worm videos is that, when the worms cross (occlude) each other we lose track of both worms in that particular video frame. If this problem persists for subsequent frames, it affects the food leaving count process. Therefore, the contribution made in this work is based on features extracted from high magnification single worm videos. We applied dimensionality reduction methods to generate shape-based features from high magnification video records of single worms moving in a confined environment with food. The reason is that, when the nematode worms are searching for food or responding to an external factor such as threat, they move rapidly within a short period. With unsupervised dimensionality reduction methods such as principal component analysis (PCA), independent component analysis (ICA) and non-negative matrix factorisation (NMF), it has been shown that the changes in the body shape of the worms can be described using four or five basic shapes (Stephens et al. [2008], Berman et al. [2014], Szigeti et al. [2015], and Gyenes and Brown [2016]). For example, the features extracted using PCA technique shows that, 95% of the worms shape variance can be captured by retaining just four dimensions (Stephens et al. [2008]). Here, we have not only demonstrated that four basis shapes account for 94% of the worms shape variance, but also shown the reconstruction of a worm shape using these basis shapes. We will discuss the steps and results of this achievement later in the report. The use of dimensionality reduction techniques in this work is very important, because it is used to learn patterns that can generalise to unseen data more reliably. Reduction of features dimension help to save a lot of storage memory and also reduce the computational time required to analyse large video dataset. The contributions outlined in this work are useful for *C. elegans* behavioural study and the research community:

1. We have demonstrated that unsupervised linear and nonlinear dimensionality reduction technique such as NMF and kernel principal component analysis (KPCA), can be used to improve classification accuracy of different strains of *C. elegans* previously used to investigate the physiological and pathological diseases in ageing worms. A related work done with different sets of strains produced classification accuracy score of 49.40% using time series of PCA eigenworms to train a deep classifier (Javer et al. [2019]). We performed similar experiment using different set of *C. elegans* strain. The outcome of our experiment using the same set of features representation, produced an accuracy score slightly below half the score obtained in their work. Thereafter, we tried the unsupervised NMF representation to extract low-dimensional features from our dataset. The classification accuracy score achieved was slightly above the score obtained with the PCA features and a little above half the score obtained in the previous work done by Javer et al. [2019]. Furthermore, we extracted the low-dimensional postural features using the KPCA technique. The prediction score achieved using the KPCA features was again slightly above the score obtained with PCA features but less than the score got with the NMF representation. However, the NMF and KPCA features has shown to improve the classification accuracy score for our chosen sets of worm strains. The results are also presented in different format (t-SNE, confusion matrix and receiver operating characteristic ROC curve) that will enable biologists to interpret the outcome.
2. Next, we decided to use Fisher’s linear discriminant analysis (LDA) to extract postural features, since the linear and non-linear techniques has failed to capture genotype specific differences from the worm’s movement patterns without any form of supervision. Although the performance score achieved with the LDA was below the scores obtained with NMF and KPCA features, however it was slightly above classification score produced when PCA embedding was used. In addiction, the visualisation of the LDA training embedding using t-SNE 2D plots shows that, there was a clear separation between the different classes of worms in the dataset, but failed to generalise.

3. Finally, we applied the supervised kernel linear discriminant analysis (KLDA) technique to extract phenotypic locomotory features of the worms and performed the same classification task. the accuracy score obtained outperforms the scores achieved with PCA,KPCA and LDA representation, but slightly lower than the NMF score. However, it performs better when trained with a deep classifier compared with the other postural shape representation methods (PCA,NMF,KPCA and LDA)

Overall, the different time series locomotory feature representation methods (NMF, KPCA, LDA and KLDA) has shown to improve the *C. elegans* genotype classification accuracy score.

### 1.4 Report structure

In Chapter 2, we provide background information about the nematode *Caenorhabditis elegans*, different machine learning algorithms and evaluation metrics utilised in this study. Chapter 3 summarises related work done in the classification of *C. elegans* based on behavioural phenotypes and other findings. Here, we considered the basic requirements to setup a tracking system, existing feature extraction and classification methods. In Chapter 4, we explain the data used in this work and the various image processing techniques utilised for the extraction of meaningful phenotypes in the video images. In Chapter 5, we present the extraction of the high-dimensional behavioural phenotypes, the application of dimensionality reduction techniques such as principal component analysis (PCA), non-negative matrix factorisation (NMF) and kernel principal component analysis (KPCA). Subsequently, we performed the classification of different worm genotypes using each low-dimensional principal components. Finally, we discussed the results and challenges. Next, we present the use of supervised feature extraction technique known as linear discriminant analysis (LDA) to classify worms with different genotypes in Chapter 6. In Chapter 7, we show how a kernel function was combined with supervised linear dimensionality reduction method (referred to as kernel Fisher discriminant analysis (KLDA)) to extract low-dimensional features useful for the classification of disease-model worms with distinct genotypes. In Chapter 8, we summarise the work done and provides steps to take in the future work.
Chapter 2

Background and Methods

2.1 Introduction

In this chapter, we provide basic information about the experiment data, the machine learning algorithms used for the feature extraction and classification task, and the evaluation metrics use to measure the quality of the extracted features and the learning models prediction performance. This chapter is divided into two parts. In the first part, we provide the basic biological information about the Caenorhabditis elegans (C.elegans) including information about their phenotypic features and the ways through which they are measured. In the second part, we first described the machine learning classification algorithms implemented to classify C. elegans based on postural features extracted from successive video images that describes their movement patterns. Next, we discussed some statistical performance metrics use to measure the quality of the image object extraction process, missing values imputation method and the performance of the different machine learning algorithms in categorising worms with subtle phenotypic differences based on their distinctive genotypes. Finally, we explained different supervised and unsupervised dimensionality reduction techniques use for the extraction of meaningful features from the C. elegans skeleton angles data.

2.2 Background

In this section, we provide a brief introduction of the C. elegans worms and why this particular biological model organism is suitable for the study of behavioural attributes as a function of their movement patterns. Furthermore, we looked at useful phenotypic features that could be extracted for the purpose of studying their movement behaviours. In the end, we explain ways in which postural or shape-based features are extracted from successive still images of C. elegans video files.
2.2.1 C. elegans as model organism

C. elegans is a nematode that lives in soil and feeds on bacteria such as Escherichia coli. They belong to the class of phylum nematoda also referred to as roundworms. They have transparent body with unsegmented length of 1 mm. C. elegans are considered model organism because their genome and nervous system are well mapped (Lin et al. [2017] and Hodgkin [2001]), which consists of 302 neurons and approximately 7000 synaptic connections. Furthermore, they have been used for several genetic investigations in domains such as development biology, cell biology and neurobiology (Hodgkin [2001]). Further studies show that, they have a simplified version of complex nerve system that contains all the components needed to explain complex behaviour in higher organisms such as humans (Brenner [1974]). The importance of C. elegans as a model for genetic study was first introduced by Nigon and Dougherty [1949], however C. elegans became widely known and accepted as an experimental organism due to the work of Sydney Brenner in the early 60’s. Currently, C. elegans are now used for several screening experiments (Gosai et al. [2010], Hulme and Whitesides [2011] and Partridge et al. [2018]). Worms without any form of defect in their nerve system during developmental stages are referred to as the wild-type or normal worms. On the contrary, worms with defects in the nerve system during developmental stages or artificially created using a process called gene knockouts are referred to as mutant worms. There are different kind of mutant worms as a result of the affected gene or genes, which in most cases determines their names. The nomenclature for naming C. elegans was applied first by Brenner [1974], but is still dynamic. The C. elegans nomenclature system become the main focus of WormBase under the leadership of Hodgkin. They assign unique laboratory/strain code for naming strains, and an allele code for naming genetic differences (mutation) and transgenes. The laboratory/strain code is made up of 2 to 3 uppercase letters while the allele consists of 1 to 3 lowercase letters. An example of a strain name or number is CB429, where CB represent the laboratory name (Cambridge) and is not italicised. For more information about the naming convention refer to WormBase (Horvitz et al. [1979] and Tuli et al. [2018]).

In this work, the goal is to use different low-dimensional phenotype representations to classify worms with different genotypes previous used in the investigation of physiological and pathological ageing of C. elegans (Martineau et al. [2019]). The dataset consists of 5 different classes of mutant strains (OW939 (zglIs113[Pdat-1::alpha-Synuclein::YFP]), OW940 (zglIs128[Pdat-1::alpha-Synuclein::YFP]), OW949 (zglIs125[Pdat-1::alpha-Synuclein::YFP]), OW953(zglIs138[Pdat-1::YFP]), and OW956(zglIs144[Pdat-1::YFP]) ) and a single wild type strain (AQ2947). The mutant worms were created by fusing alpha-synuclein to yellow fluorescent protein (YFP) which is attached to the 8 dopaminergic neurons of the worm. The alpha-synuclein is an aggregated-prone protein that causes Parkinson’s disease (neurodegenerative disorder) in human. In the strain names, the Pdat-1 denote the promoter and the YFP serves as the acceptor for the genetical-coding sequence of the gene.
2.2.2 C. elegans phenotypic features

It has been established that C. elegans performs actions that show they learn and store information. Lin et al. [2017] and Sasakura and Mori [2013] describe these behaviors as non-associative and associative learning and memory based on how they react to mechanical, thermal, and chemical stimuli. The non-associative behaviors are changes in their behaviors as a result of a stimulus, but do not require any form of reinforcement. On the other hand, the associative behaviors are responses as a result of different factors or stimuli. For example, the avoidance of heat by a worm after pre-exposure to a similar environment is regarded as non-associative learning. On the other hand, the ability of C. elegans to evaluate food quality and leave a pathogenic bacterial infected environment is described as associative learning (Sasakura and Mori [2013]). Therefore, the mechanisms responsible for the learning process and memory are vital for the survival of the animal in the real world. For example, nematode worms exhibit an escape behavior when heat is introduced to its environment, thereby making them to move faster away from the heat source. As they move faster, the changes which occur along their body shape contain meaningful information that can be used to infer the worm’s behavior at that instance. These physical but subtle changes that occur along the worm body shape or posture as they move are referred to the phenotypic features. The set of phenotypic features extracted from a given worm body shape depends on the researcher’s interest. These observable characteristics or phenotypic features are important because they can be used to infer the worm’s genotype (these are set of genes the organism carries) and the effect of environmental changes on these genes. Examples of phenotypes include but are not limited to worm’s size, colour, shape (posture), behavior and biochemical properties. To conclude, C. elegans has the tendency to display plasticity (change in body shape) to different kinds of environmental factors (stimuli) which can be described as phenotypic behaviors, but the underlying internal mechanisms that brings about these observable characteristics are referred to as their genotypes. Hence, a worm’s phenotype is controlled or determined by its genotype.

2.2.3 Extraction of phenotypic features from C. elegans images

The extraction of features from an image depends on the region or object of interest. To extract useful features from the C. elegans images requires tracking the worm over a period of time. This has been successfully achieved using worm trackers (Geng et al. [2004]), in which a region of interest is followed by automated computer vision and image processing methods (Baek et al. [2002]). There are two main types of trackers: single and multiple worm trackers. The single worm trackers are used to track only one worm over a user-defined time interval, while the multiple worm trackers are used to track two or more worms simultaneously over a time period. Videos created with the worm tracker can be analyzed further using image analysis techniques. Features like the image texture, pixel intensities or shapes can be collected from the images with a computer program written in a specific language. As mentioned earlier, the features extracted depends on the researchers’ interest. In this work, our focus was
based on features measured from the shape of the worms. To achieve this, we applied image processing techniques such as image segmentation (binarisation) and image skeletonisation (thinning). Image segmentation involves the detection and conversion of a region of interest in an image to a particular colour (in our case white and referred to as the foreground object) and other regions to a different colour (in our case black and referred to as the image background). The image segmentation task requires mapping out or selecting areas in the image object for which measurement can be done. Image skeletonisation involves reducing the size of the identified image object to a single pixel width, but still preserve the properties of the original shape. The image object skeletonisation helps to reduce the amount of information to be stored for a single image. In this research, we measured tangent angles along the worm’s skeleton body shape and stored them as features for further processing. We referred to the extracted skeleton angles data as the high-dimensional worm video frames data. The retrieved features are then used directly or transformed into a lower dimensional embeddings for the classification of the different types of worms. There are classical and deep learning-based image segmentation and skeletonisation models used for the extraction of object or region of interest in images. Examples of classical image segmentation models are: Otsu thresholding algorithm (Nobuyuki [1979]), Bernsen local adaptive thresholding algorithm (Bernsen [1986]), Sauvola method (Sauvola and Pietikainen [2000]), Niblack algorithm (Niblack [1986]) and Kapur algorithm (Kapur et al. [1985]) etc. Examples of deep algorithms used for image segmentation are: Fully convolutional network (FCN)(Milletari et al. [2016]), Dilated convolutional neural network (DenseNet), Mask R-CNN (Wang et al. [2018a]), SegNet, U-Net and Generative adversarial networks (GAN), DeepYeast (Pärnamaa and Parts [2017]), WorMachine (Hakim et al. [2018]), DeepLoc (Kraus et al. [2017]), and Deep Q-net (Wang et al. [2018b]) etc. Examples of classical image skeletonisation methods are: Zhang and Suen algorithm (Zhang and Suen [1984]), Guo and Hall model (Guo and Hall [1989]), Stentiford algorithm (Alberto and Tosunoglu [1986]) and Hilditch algorithm (Hilditch [1969]) etc. Examples of deep networks used for image skeletonisation are skeleton generative adversarial network (skelGAN) and structure correcting adversarial network (SCAN) etc.

2.3 Methods

In this section we briefly discuss the machine-learning methods, evaluation metrics and dimensionality reduction techniques applied in this research.

2.3.1 The machine-learning classification algorithms

These are learning algorithms that are trained using a subset of a given dataset (training data) and use to make target (class) predictions on an unseen data (test data) drawn from the same distribution. The classification algorithms used in this work are supervised learning algorithms. This means that, the algorithms are used to find hidden patterns in a given data
distribution with the aid of a user predefined class labels. They work by creating a function that maps each input data point to a known output label using examples of the input-output pairs. Here, we used three standard machine learning algorithms such as Support Vector Machines (SVM) (Guyon et al. [2002]), Random Forest (RF) (Breiman [2001]), and $k$-Nearest Neighbour (KNN) (Cover and Hart [1967]), and a deep learning model (ResNet) (He et al. [2016] and Javer et al. [2019]) to characterise the features extracted from worms with different genotypes. Although there are other learning algorithms including deep learning models, but we choose to implement the three types of estimators mentioned above.

2.3.1.1 Support Vector Machines

The support vector machine (SVM) was introduced by Cortes and Vapnik [1995] and it was designed to find a decision boundary or hyperplane that separate data points into distinctive classes with a maximal margin. The SVM model in its simplest form uses a linear hyperplane to separate the data points into different classes. This linear approach leads to a lot of misclassifications of the data points, because not all points belong to different classes can be separated by a linear hyperplane. Hence, most SVM model implementations use a soft margin which although permit the misclassification of some data points but at a specific cost value. The SVM model can also be used to create non-linear decision boundaries by first transforming the data points to a new feature space using a mapping function such as Gaussian kernel or polynomial kernel. The SVM model is robust to noisy features and computationally efficient. SVMs have been applied to accurately classify different growth stages in *C. elegans* in high-resolution image data (White et al. [2010]). In their work, the image objects are segmented into large and small objects. Thereafter, the segmented objects are decomposed into parts and features are extracted from these parts. The extracted features were used as input to the SVM learner for classification. SVM have been used successfully to solve classification problems in cell biology (Guyon et al. [2002]) and (Sommer and Gerlich [2013]).

2.3.1.2 Random Forest

The random forest (RF) is a supervised machine learning algorithm that was first introduced by Breiman [2001] for the purpose of training an ensemble of decision trees based on a random influencer (Breiman et al. [1984]). The RF classifier utilises the output of multiple decision trees (ensembles of trees) based on majority voting rule to determine the class to assign an input vector (data point). The overall performance of the RF classifier is determined by the strength of each tree in the forest and the correlation that exist between the trees. The algorithm generates random vectors that determines how each tree in the ensemble will grow. One way to grow each tree in the forest was proposed by Breiman [1996], in which random selection without replacement is applied on the training data. Another method was suggested by Dietterich [1998], used random split selection such that at each split node, a split rule is selected at random
from a set of $k$ best splits (where $k$ is the number of tree in the forest). The following steps are taken to decide the class of an input vector:

1. for each $k$th tree, generate a random vector $\Theta_k$ which does not rely on the past random vectors $\Theta_1, \ldots, \Theta_{k-1}$, but are all drawn from the same distribution (independent identically distributed random vectors)
2. grow a tree using the random vector $\Theta_k$ and the training set ($X$), which results in a classifier $h(x, \Theta_k)$ and $x$ denotes an input vector
3. after the creation of a large number of trees, voting is done to determine the class of $x$ based on the popular vote

The method used to generate the random vector $\Theta$ in step 1 was described as the counts in $N$ boxes obtained, when $N$ number of darts are thrown into $N$ random boxes ($N$ denotes the number of training samples) (Breiman [1996]). In the case of the random split selection method, $\Theta$ is determined by the number of independent random integers between 1 and $K$. The structure (dimension) of $\Theta$ is determined during the tree construction. Instead of popularity vote (step 3), the algorithm uses an average to measure the prediction outcome and this help to minimise the overall variance while it maintains a low bias among the decision trees. The RF model uses a feature selection method in the prediction process, which makes it robust to high dimensions data. It is computationally efficient and widely used in $C. elegans$ phenotypic feature classification problems as well as in cell biological applications (Sommer et al. [2011]).

2.3.1.3 $k$-Nearest Neighbour

The $k$-nearest neighbour ($k$-NN) classifier is a non-parametric machine learning algorithm first proposed by Fix and Hodges [1951], and thereafter improved (Cover and Hart [1967]) to solve classification and regression problems. When used to solve either classification or regression problem, the input consists of $k$ nearest training samples in the feature space, whereas the output is based on the nature of the problem being solved (classification or regression). In the case of classification problem, the output is a class identifier or label and a feature vector (data point) is classified (assign a class label) based on a popularity vote of its closest $k$ neighbours. This implies that, a query data point is assigned a class based on the labels of the highest number of its $k$-nearest neighbours. The number of nearest neighbour $k$ is always a positive integer and in most cases predefined by the user. If $k=1$, it simply means that the data point is assigned a class based on a single closest point. The output for a regression problem is a real values that shares the same or similar property to the query data point. The $k$-NN algorithm depends on a distance metric (such as Euclidean, Mahalanobis) for classification and works well when the training data (especially when dealing with features measured with different units) are normalised on the same scale (Piryonesi and El-Diraby [2020] and Hastie [2001]). The $k$-NN model works well when the data are normalised on the same scale. For a high dimensional dataset, the
use of dimensionality reduction or feature selection may be necessary to improve the model performance. A unique property of this algorithm is the sensitivity to the local structure of the dataset. Furthermore, the \( k \)-NN uses a technique that assigns more weight to data points that are close to each other and a smaller weight to points that are far apart. During voting, points that are closer to each other contribute more in the selection of a class (classification) or property value (regression) for a query test point. One disadvantage of the \( k \)-NN classifier is that it does not work well when the class distribution in the data is skewed (Coomans and Massart [1982]). This means that, \( k \)-NN classification performance drops when there are more data points belonging to a particular class referred to as the majority class than the other classes in the dataset. A possible solution to this problem is to weight the classification process such that the distance between a test point and its nearest neighbours is considered. This can be achieved by multiplying the class (classification) or value (regression) of each of the \( k \) nearest points to the test point by a weight that is proportional to the distance between each close point to the test point. The choice of the value of \( k \) influences the model output and is predefined at the time of model building. The optimal (best) value of \( k \) is learned from the training data itself. The larger the value of \( k \) the smaller the effect of noise on the classification generalisation results, but affects the decision boundaries of the different classes (classes are hard to separate) (Everitt et al. [2011]).

### 2.3.1.4 Deep learning-based algorithms

Due to the rapid increase in modern technologies with high storage capacity and speed, deep learning-based models for image analysis and classification are been developed. In most computer vision applications, the deep learning machines has outperformed the existing traditional machine learning algorithms without rigorous feature selection, and has improved the performance of deep automated systems such as self-driven cars and medical images diagnostic systems. The basic deep learning model is the convolutional neural network (CNN) designed for image object detection and classification. This deep machine consists of three layers: input, hidden and output layers. The input layer accept the original images (the pixel matrix) into the deep network. The hidden layer or layers transform the input images into feature maps by the learning algorithm and the output layer returns the features learned by the model in image format with the same size as the input image. Currently, many deep learning networks are used for image object classification. Examples of deep learning-based networks used for image classification are LeNet-5 (Wang et al. [2018a]), AlexNet (Krizhevsky et al. [2012]), VGGNet16 (Simonyan and Zisserman [2015]) and ResNet (He et al. [2016]). In this work, we applied the fully convolutional classification model implemented by Javer et al. [2019] and it was used for the classification \( C.\ elegans \) strains.

In this research, we considered using the three classifiers: Random Forest (RF), \( k \)-Nearest Neighbour (\( k \)NN), Support Vector Machine (SVM) and deep classifier model. The reason for
multiple classifiers was to determine which pair of data representation and classifier gives a better classification accuracy. The implementation codes for these learning algorithms are based on the python Scikit-learn packages. However, for each classifier, we tune some of the hyper-parameters and choose to report the model configurations that produced the best classification accuracy for each data representation. For $k$-NearestNeighbour classifier, we used different values of $k = [1, \ldots, 10]$ on the training data and record the prediction performance on the test data. For the random forest classifier, after testing different values for the model parameters, we initialised the number of estimators ($n\_estimators$) in the range of 2 to 25, and the maximum depth ($max\_depth$) in the range of 1 to 15. The number of estimators determine the number of trees in the forest and maximum depth determines the number of nodes allowed from the root to the farthest leaf of a tree. For the SVM classifier, we initialised the degree of tolerance $C$ in the range of 5 to 150, and with different kernels ($kernel = ['linear',\gamma]$). The deep classification model consists of a 7 convolution followed by $53 \times 3$ strided convolutions, 4 dilated convolutions in the order of 2-4-2-1, and a classification layer made up of Global average pooling and a $1 \times 1$ convolution. A unique feature of the model is that, it can be applied to input of arbitrary size. The deep model parameters are: learning rate $= 0.001$, momentum $= 0.9$, weight decay $=0.0001$, optimiser= SGD (Stochastic gradient decent), epoch $=1000$ and mini-batch size $=64$. The output of the deep model was modify to display the top 5 accuracy scores, instead of the top 1 and 5 scores only. The top k accuracy is implemented such that, a prediction is considered true if its original label is among the top k labels predicted by the model. For instance, if $k=1$, it simply means that original label must correspond with the first label predicted by the learning model.

2.3.2 Evaluation metrics

Evaluation metrics which are also referred to as loss functions are quantitative assessment methods used to measure the performance of a machine learning model (classification and regression models). Most machine learning classification problems involves assigning a class or label to specific data points in the test data. This is achieved by using a function that is tuned to map input observations to their corresponding discrete output variables or classes. A classification problem that involves only two classes of data points is referred to as binary classification problems. Classification problems that needs to assign data points to more than two classes are referred to as multi-class classification problems. The performance of any classification model is determined by its ability to predict correctly the classes to which data points are assigned. Hence, evaluation metrics are performance assessment methods used to measure the quality of predictions made by the learning models. The evaluation results are used to choose the best performing model that can solve a specific classification or regression problem. Evaluation metrics are also use to measure the quality of features extracted from raw input images during image preprocessing. The evaluation method used depends on the problem the machine learning algorithm is trying to solve. Evaluation methods suitable for classification problems are confusion matrix, receiver operating Characteristics (ROC) curve and percent classification
accuracy. Whereas, for regression problems which output continuous values requires the use of MAE or RMSE evaluation methods. For a regression problem, the learning algorithm that returns the smallest MAE or RMSE is considered the optimal algorithm. Examples of evaluation methods used for image processing (segmentation and skeletonisation) are dice index, Jaccard index, segmentation accuracy (SA), oversegmentation rate (OR), undersegmented rate (UR), F-measure, recall, precision, mean squared error (MSE), pixel error rate (PERR), signal-to-noise ratio (SNR) and peak signal-to-noise ratio (PSNR).

2.3.2.1 Confusion matrix

A confusion matrix is a table which describes the quality of predictions made by a classification model based on a number of observations for which their true values are known (Fawett [2006]). The rows of the confusion matrix represent the actual class a data point belongs to while the columns in the table represent the predicted class for that particular data point. It is a performance metric widely used to measure the quality of predictions produced by machine learning models. It is suitable for both binary and multi-class classification problems. Different scores can be computed from the confusion matrix which are used to measure the performance of the classification model based on the application domains. Examples of these scores as seen in Table 2.1 are classification accuracy \((Acc)\), classification error rate \((\epsilon)\), recall \((r)\), precision \((pr)\), false positive rate \((fpr)\), false negative rate \((fnr)\), sensitivity \((se)\), specificity \((sp)\); where TP represents the number of positive samples correctly classified (true positives), TN represents the number of negative samples correctly classified (true negatives), FP represents the number of negative samples incorrectly classified as positive (false positive) and FN represents the number of positive samples incorrectly classified as negative (false negative). Classification accuracy and classification error rate are the most widely used performance metrics in many classification problems. Accuracy and classification error rate have been applied in many application because they both have theoretical meaning, easy to compute and interpret. However, these performance metrics are not suitable for problem in which there are missing or incomplete class labels. Table 2.2 shows how the various scores are calculated using information from the confusion matrix.

<table>
<thead>
<tr>
<th>Actual</th>
<th>Predicted positive</th>
<th>Predicted negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>positive</td>
<td>TP</td>
<td>FN</td>
</tr>
<tr>
<td>True positive</td>
<td>False negative</td>
<td></td>
</tr>
<tr>
<td>negative</td>
<td>FP</td>
<td>TN</td>
</tr>
<tr>
<td>False positive</td>
<td>True negative</td>
<td></td>
</tr>
</tbody>
</table>
Table 2.2: Confusion matrix scores computation

<table>
<thead>
<tr>
<th>Evaluation metrics</th>
<th>Score equation</th>
<th>Evaluation metrics</th>
<th>Scores equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>classification error ($\epsilon$)</td>
<td>$\epsilon = \frac{FP + FN}{TP + FP + TN + FN}$</td>
<td>Accuracy</td>
<td>$Acc = \frac{TP + TN}{TP + FP + TN + FN}$</td>
</tr>
<tr>
<td>recall ($r$)</td>
<td>$r = \frac{TP}{TP + FN}$</td>
<td>Precision ($pr$)</td>
<td>$pr = \frac{TP}{TP + FP}$</td>
</tr>
<tr>
<td>Specificity ($sp$)</td>
<td>$sp = \frac{TN}{TP + TN}$</td>
<td>Sensitivity ($se$)</td>
<td>$se = \frac{TN}{TN + FN}$</td>
</tr>
<tr>
<td>False positive rate ($fpr$)</td>
<td>$fpr = 1 - sp$</td>
<td>False negative rate ($fnr$)</td>
<td>$fnr = 1 - r$</td>
</tr>
</tbody>
</table>

2.3.2.2 Receiver Operating Characteristics curve

The Receiver Operating Characteristics (ROC) curve is a graphical method for performance evaluation of classification models. This graphical method takes into consideration all possible classification thresholds when displaying the performance of the learning model. The ROC curve provides a summary of learning model performance based on a certain compromise between the true positive rate ($tpr$) and the false positive rate ($fpr$) ([Fawett [2006]]). The x-axis of an ROC curve represents the false positive rate $fpr = \frac{FP}{TN + FP}$ while the y-axis represents the true positive rate $tpr = \frac{TP}{TP + FN}$. An example of the ROC curve is shown in Figure 2.1, where the line $fpr = tpr$ in the curve represent the case of randomly guessing the class. If the ROC curve moves far to the left, it means the classifier is able to distinguish the positive points from the negatives. However, if it moves further to the right, it signifies that the classifier is either predicting positives instead of negatives or vice versa. This graphical method of performance evaluation is suitable for an imbalanced or skewed dataset where the classification error rate may not be a good performance metric to use. A conventional performance metric for the ROC curve is the Area Under the Curve (AUC). The AUC is a useful performance metric to measure the quality of predictions obtained from learning algorithms, because it does not depend on the decision criterion used and prior probabilities. The consider various thresholds to retrieve the performance of the classifier(s). For this reason, the ROC curve evaluation method is different from the accuracy method which only shows a classifier performance based on a single threshold. The ROC convex hull is a robust method for identifying potentially optimal classifiers. It is worth noting that, for a multi-class classification problems, the ROC curve uses a One-Versus-All (OVA) approach to evaluate the performance of the classifiers. This OVA approach involves training one classifier per class, with the data points belonging to that class considered as positive points while all other points in the dataset are considered negative points. This process is repeated for all classes in the dataset, and the trained classifiers are then used to make predictions for an unseen data point.
For the ROC curve, one major challenge is how to determine the optimal operating point (OOP) also known as the optimal decision threshold. The reason for the OOP is to enable the classifier choose the best trade off between the cost of not being able to give the right prediction on the positive data points compared to the cost of producing high false alarm. Examples of evaluation metrics use to determine the OOP of the ROC curve are Youden index, highest sensitivity plus specificity, distance to corner, harmonic mean of the sensitivity and specificity, and anti-harmonic mean of the sensitivity and specificity etc. The sensitivity and specificity defined in Table 2.2, are two important pair of variables use to determine the OOP of the ROC curve. For instance, the Yoden index for each cutoff value is defined as:

\[
\text{Youden index} = \text{sensitivity} + \text{specificity} - 1
\]

(2.1)

### 2.3.2.3 t-SNE

The t-distributed Stochastic Neighbour Embedding (t-SNE) is a machine learning algorithm widely use for the visualisation of high-dimensional data in a low-dimensional feature space. The algorithm is an extension of the existing Stochastic Neighbour Embedding (SNE) introduced by Hinton and Roweis [2002], for easy optimisation and better visualisation of high-dimensional data. For a high-dimensional data points \( x_1, \ldots, x_N \) of size \( N \), the t-SNE algorithm start by computing the probabilities \( P_{ij} \) that determines how close point \( x_i \) is to point \( x_j \). The conditional probabilities for \( i \neq j \) is defined as:

\[
P_{ji} = \frac{\exp\left(-||x_i - x_j||^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-||x_i - x_k||^2 / 2\sigma_i^2\right)}\]

(2.2)

where \( \sigma_i \) represents the variance of the Gaussian centred at data point \( x_i \), \( P_{ii} = 0, \sum_j P_{ji} = 1 \) for all \( i \), and \( ||x_i - x_j||^2 \) represents the Euclidean distance between points \( x_i \) and \( x_j \) in the
set. This implies that, the similarity between two data points \( x_i \) and \( x_j \) is the conditional probability \( P_{ji} \) \cite{Van+Hinton:2008}. The probabilities \( P_{ij} \) is given as:

\[
p_{ij} = \frac{P_{ji} + P_{ij}}{2N}
\]  

such that \( \sum_j P_{ij} > \frac{1}{2N} \) for all data points \( x_i \). Because distances between points in high-dimensional data are difficult to discriminate as a result of curse of dimensionality which in turn reduces the \( P_{ij} \). The t-SNE algorithm learn a \( d \)-dimensional map \( y_1, \ldots, y_N \) with \( y_i \in \mathbb{R}^d \) that represents the similarities \( P_{ij} \) as close as possible. The similarities between point \( y_i \) and \( y_j \) in the map is defined as:

\[
q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k\neq l}(1 + ||y_k - y_l||^2)^{-1}}
\]  

For \( i \neq j \) and \( q_{ii} = 0 \). To find the positions of points in the map, the Kullback-Leibler divergence is used to minimise the distribution \( P \) and \( Q \). The Kullback-Leibler divergence given as:

\[
KL(P_i || Q_i) = \sum_i \sum_j p_{ji} \log \frac{p_{j|i}}{q_{j|i}}
\]  

The minimisation of \( KL(P_i || Q_i) \) with respect to the point \( y_i \) is achieved using a gradient descent.

### 2.3.2.4 Mean Absolute Error

The mean absolute error (MAE) is the average of the difference between the actual values and the predicted values \cite{Hyndman:2006}. The MAE metric shows how far the predicted values are from the actual values.

\[
MAE = \frac{1}{N} \sum_{i=1}^{N} |y_i - \tilde{y}_i|
\]  

where \( y_i \) represents the actual value, \( \tilde{y}_i \) represents the predicted values and \( N \) the total number of data points in the data. This metric will be use to evaluate different data imputation methods and choose the method with the least score.

### 2.3.2.5 Root Mean Squared Error

The root mean squared error (RMSE) is the squared root of the average of the square difference between the actual values and predict values \cite{Hyndman:2006}. This metric is different form the MAE method because it takes the average of the square between the actual values and the predicted values. The RMSE metric is considered not biased in terms of the direction of error since it takes the squared difference between the actual and predicted values.
That is, the squared difference prevents the cancellation of positive and negative error values. One limitation of the RMSE evaluation method is that, it is sensitive to outliers.

\[
RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2}
\]  

(2.7)

where \(y_i\) represents the actual value, \(\hat{y}_i\) represents the predicted values and \(N\) the total number of data points in the data.

Both the MAE and RMSE metrics are use later in this work to evaluate the performance of missing data (dropped video frame) imputation techniques. The imputation method that returns the lowest error value is selected. Similar to the MAE, the RSME metric will be use to evaluate different data imputation methods and then choose the one that gives the least score.

### 2.3.3 Dimensionality Reduction Methods

Dimensionality reduction techniques are learning algorithms used for the extraction of discriminant features in data sets with minimal loss of information and reduction of the feature dimension. These techniques are widely used feature engineering (feature extraction) in fields like computational biology, machine learning, and computer vision (Sommer and Gerlich [2013], O’Farrell et al. [2005], Bucinski et al. [2005]). Dimensionality reduction techniques are classified into two main groups: unsupervised and supervised learning methods. The unsupervised methods are learning algorithms which aimed at finding undiscovered patterns in a data without the use of pre-defined labels and with minimum human intervention. Examples of unsupervised learning methods used for dimensionality reduction are Principal Component Analysis (PCA), Non-negative Matrix Factorisation (NMF), Kernel Principal Component Analysis (KPCA), Vector Quantisation (VQ), Independent Component Analysis (ICA), autoencoder and other variants of these methods. The supervised methods are learning algorithms that aimed at finding hidden patterns in given data with the help of user pre-defined labels. It requires defining a function that maps a specific data point to a known output label thereby learning from examples of input-output pairs. Examples of supervised learning algorithms use for dimensionality reduction are Fisher Linear Discriminant Analysis (LDA), Kernel Fisher Discriminant Analysis (KLDA) and Neighbourhood Component Analysis (NCA).

In this research, we based our discussions mostly on the supervised and unsupervised dimensionality reduction methods we implemented, but provide references for other methods where necessary.

#### 2.3.3.1 Principal Component Analysis

Principal component analysis (PCA) is an unsupervised linear transformation technique widely used to transform high-dimensional data into a low-dimensional data with minimal loss of information (Jolliffe [1986]). PCA is sometimes referred to as Hotelling transformation or
Karhunen-loeve transformation (Gonzalez and Woods [1993]). The PCA algorithm has been applied in many object recognition and classification problem (Mather [1999] and Turk and Pentland [1991]). PCA utilises the fact that significant information about the features is contained in the directions along which the variations are the largest (Wang and Paliwal [2003]). The algorithm transform an input data with $D$-dimensions to $d$-dimensional linear feature space in the direction of the largest variance. An important application of PCA in the extraction of features in C. elegans images was done by Stephens et al. [2008]. The study shows that with only four (4) principal components we can capture 95% of the worm shape variance. This discovery has triggered the application of PCA in model organism behavioural studies. More information about the work done by Stephens et al. [2008] is found in Chapter 3. In this research, we applied PCA to extract low-dimensional shaped-based features, similar to the work done by Stephens et al. [2008]. The extracted features are then used to categorise worms with distinctive genotypes on the basis of their locomotory activity.

Given an input data $x \in \mathbb{R}^{N \times D}$, the goal of PCA is to find $d$ principal axes such that $d \leq D$. The covariance matrix $C$ is defined as:

$$C = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)^T (x_i - \mu),$$

(2.8)

where $\mu$ is defined as:

$$\mu = \frac{1}{N-1} \sum_{i=1}^{N-1} x_i$$

(2.9)

This result to an eigendecomposition of the covariance matrix given as:

$$Cv_i = \lambda_i v_i$$

(2.10)

where $i \in d$, $\lambda$ and $v_i$ represent the eigenvector and eigenvalues respectively. After computing the eigenvalues and eigenvectors, select the $d$ eigenvectors with the largest eigenvalues. More details on PCA is found in appendix A.

2.3.3.2 Non-negative Matrix Factorization

The Non-negative Matrix Factorisation (NMF) algorithm is an unsupervised learning technique mostly used for feature extraction and dimensionality reduction (Lee and Seung [1999]). The NMF algorithm has been widely utilised to solve problems in computational biology and machine learning related problems. This NMF method decompose a matrix $V \in \mathbb{R}^{n \times m}$ into two nonnegative matrices $W \in \mathbb{R}^{n \times r}$ and $H \in \mathbb{R}^{r \times m}$ such that, their product act as an approximation of the matrix $V \approx WH$. The matrices $W$ and $H$ are referred to as the basis (weight) and encoding (component or coefficient) matrices respectively. The rank $r$ is selected so that $(n+m)r < nm$. The values of $W$ and $H$ are updated using an update rule while minimising the approximation error using an objective function. The non-negativity constraints of the NMF algorithm and its parts-based local representation has been proven useful in the extraction
meaningful features in image data analysis and classification (Lee and Seung [1999], Hassani et al. [2019] and Brunet et al. [2004a]). NMF is different from PCA, because the NMF algorithm prohibits the entries of negative values in both matrix factors \( W \) and \( H \). For this reason, NMF is interpreted as the combining of parts to form a whole. More information about the NMF algorithm is found in appendix B. In addition, more information about the application of NMF algorithm is found in Chapter 3. The NMF parts-based representation will be useful in the analysis and classification of different type of worms, because this property of NMF makes it possible to capture shape-based features of the worms during movement. The NMF algorithm has been applied to skeletal angle data of \( C. elegans \) and the result shows that four (4) basis capture 97.6\% of the worm shape variance (Gyenes and Brown [2016]). The captured features can then be used to identify the type of worm and categorise them based on their distinctive genotypes. Due to the application of image processing operations such as segmentation and skeletonisation on the worm images before the extraction of the body angles, it is difficult to interpret the features captured by the NMF algorithm. Similar to the facial images parts-based learning task in which the algorithm was to capture facial features such as mouths, noses and eyes (Lee and Seung [1999]); the NMF algorithm can be used to learn parts-based representation of the worm images such as the head, tail and sharp body bends. For this reason, we applied the NMF algorithm for parts-based representation of the worms.

### 2.3.3.3 Kernel Principal Component Analysis

The kernel principal component analysis (KPCA) is an unsupervised nonlinear dimensionality reduction technique used for the extraction of significant features in a given dataset. The KPCA algorithm was inspired after the successful application of kernel trick in the support vector machine domain. This unsupervised nonlinear dimension reduction method transform a linearly inseparable input data into a linearly separable feature space using a nonlinear kernel function. In the created new feature space, low dimensional significant features are extracted using linear PCA algorithm. In other words, the KPCA algorithm can be described as the application of PCA in a linear input space created with a kernel function. Given an input data \( X \in \mathbb{R}^{N \times D} \).

The following steps are taken in order to extract features from an input data using KPCA:

1. Compute the kernel matrix \( k \) from the input data \( X \in \mathbb{R}^{N \times D} \) using:

\[
k_{i,j} = k(x_i, x_j) \quad (2.11)
\]

where \( k(x_i, x_j) \) denotes a kernel matrix. The application of the kernel function is performed here. Examples of a kernel functions are:

(a) The polynomial kernel defined as:

\[
k(x, y) = (1 + x^T y)^d \quad (2.12)
\]
where \( d \) denotes the order or degree of the polynomial, and \( x \) and \( y \) denotes two feature vectors.

(b) The Gaussian kernel defined as:

\[
k(x, y) = \exp\left(-\frac{|x - y|^2}{2\sigma^2}\right)
\]  

(2.13)

where \( \sigma \) is a free parameter which can be optimised and \(|x - y|\) is the pairwise squared Euclidean distance matrix of size \( N \times N \).

2. Compute the Gram matrix \( \tilde{K} \) (Bishop [2006], Wang [2012]) defined as:

\[
\tilde{K} = k - \frac{1}{N} k - k \frac{1}{N} + \frac{1}{N} k \frac{1}{N}
\]  

(2.14)

This step is required only when the projected data is assumed to be not centred. Otherwise, the kernel matrix \( k \) is used directly. The \( \frac{1}{N} \) is an \( N \times N \) matrix in which the values of all elements are \( \frac{1}{N} \). It is important to centre the data to prevent any form of rotational variant.

3. Calculate the eigenvectors \( a_k \) using

\[
K a_k = \lambda_k N a_k
\]  

(2.15)

where \( N \) represents the number of data points in the set and we replace \( k \) with \( \tilde{K} \) if the projected dataset is not centred.

4. Compute the kernel principal components using

\[
y_k(x) = \sum_{i=1}^{N} a_k k(x, x_i)
\]  

(2.16)

For classification problem, the generated new features \( y_k(x) \) serves as input to the machine learning classifiers.

2.3.3.4 Linear Discriminant Analysis

The Fisher linear discriminant analysis (LDA) is a supervised dimensionality reduction technique used for the extraction of low-dimensional significant features from a given input space and can be used to solve classification problems when implemented as a classifier. As a dimension reduction method, the algorithm reduces the number of variables or dimension in a given dataset while preserving meaningful information about the data. The LDA algorithm was design to learn a set of variables \( \omega \in \mathbb{R}^{D \times d} \) in which when projected on the original input data \( X \in \mathbb{R}^{N \times D} \) would yield a set of low-dimensional features set \( y \in \mathbb{R}^{N \times d} \). To achieve this, Fisher suggested finding a projection that maximises the separation between the means
Chapter 2 Background and Methods

of the data classes while minimising the within-class covariance. Fisher defined the objection function as:

\[ \hat{\omega} = \arg \max_\omega \frac{\omega^T S_B \omega}{\omega^T S_W \omega} \]  

(2.17)

where \( S_B \) and \( S_W \) denote the between-class and within-class scatter matrices respectively. These matrices of same dimension \( D \times D \) are define as

\[ S_W = \sum_{k=1}^{K} \sum_{n=1}^{N_k} (x_{nk} - \mu_k)(x_{nk} - \mu_k)^T \]  

(2.18)

and

\[ S_B = \sum_{k=1}^{K} N_k (\mu_k - \mu)(\mu_k - \mu)^T \]  

(2.19)

where \( x_{nk} \) represents the \( n \)th samples belonging to the \( k \)th class in the set, \( K \) represents the number of classes in the dataset, \( N_k \) represents the number of samples in each \( k \)th class, \( \mu \) denotes the global mean of all samples in the dataset; and \( \mu_k \) denotes the mean of the \( k \)th class. Based on Lagrangian dual and the KKT conditions, equation 2.17 can be transformed into eigenvalue decomposition problem as presented in equation 2.20

\[ S_B \omega = \lambda S_W \omega \]  

(2.20)

where \( \lambda \) denote the eigenvalues. Then the transformation matrix \( W \) is defined as

\[ \omega = \max_d (\text{eig}(S_W^{-1} S_B)) \]  

(2.21)

The discriminative features \( y \) in equation 2.22 is obtained by projecting the input data onto the eigenvectors \( W \), after selecting the \( d \) eigenvectors with the largest eigenvalues \( \lambda \).

\[ y = \omega^T X \]  

(2.22)

Here, \( d \) represent the reduced dimension, which is at most equal to \( K - 1 \) and the transformation matrix \( \omega \) represent the \( d \) reduced dimension eigenvectors with the largest eigenvalues. Since \( S_B \) is the sum of \( K \) matrices of rank 1, meaning only \( K - 1 \) of these are independent and \( S_B \) is of rank \( K - 1 \) or less. This also implies that, only \( K - 1 \) of the computed eigenvalues are nonzero and the weight vectors \( \omega \) are the eigenvectors of these nonzero eigenvalues. For more information on LDA implementation refer to appendix D.
2.3.3.5 Kernel Fisher Linear Discriminant Analysis

The kernel Fisher linear discriminant analysis (KLDA) can be categorised as a nonlinear supervised dimensionality reduction technique used for the extraction of significant features in data sets. The KLDA algorithm is an extension of the Fisher linear discriminant analysis (LDA), which transforms a given input samples into a new feature space $F$ via a kernel function (Wei et al. [2005]). Subsequently, the LDA algorithm is applied in the new $F$ space to extract lower dimension discriminant features which is used to solve pattern recognition or classification problems. In other words, the KLDA algorithm start by mapping the input data vector $X$ into different feature space via a nonlinear mapping function $\phi(X)$ and utilise Fisher’s linear discriminant to extract low-dimensional discriminative features in the mapped space. Given a dataset $x \in \mathbb{R}^{N \times D}$ with $K$ number of classes and the number of samples in each class be $N_{ki}$, the KDFA algorithm transform the input data $x$ into a $K - 1$ feature space via a $K - 1$ discriminant functions. Like LDA method, the KLDA algorithm aimed at finding a projection that maximises the class means separation in the data and at the same time minimises the within class variance.

The between-class covariance matrix is given as:

$$S^\phi_b = \sum_{i=1}^{K} N_{ki} (m_i^\phi - m^\phi)(m_i^\phi - m^\phi)^T,$$

(2.23)

where $m^\phi$ denotes the global mean for the data points in the map feature space and $m_i^\phi$ denotes the mean for each class in the dataset. The within-class variance is defined as:

$$S^\phi_w = \sum_{k=1}^{K} N_{ki} \sum_{i=1}^{N_k} (\phi(x^k_i) - m_k^\phi)(\phi(x^k_i) - m_k^\phi)^T,$$

(2.24)

$$m_k^\phi = \frac{1}{N_{kj}} \sum_{j=1}^{N_{kj}} \phi(x_j)$$

(2.25)

where $m_k^\phi$ denotes the mean for each class $k \in K$. Find the function $J(.)$ that maximise the ratio between $S^\phi_w$ and $S^\phi_b$ using:

$$J(w) = \frac{w^T S^\phi_b w}{w^T S^\phi_w w}$$

(2.26)

By applying the kernel trick, KLDA objective for multi-class becomes

$$A^* = \frac{A^T M A}{A^T S A}$$

(2.27)

By maximising $A$ in equation 2.27 and compute the $K - 1$ leading eigenvectors of $S^{-1}M$. The projection of a new point $(x_k)$ is defined as

$$y(x_k) = (A^*)^T K_k$$

(2.28)
where the \(i^{th}\) component of \(K_k\) is simply \(K(x_i, x_k)\). A kernel function that mostly used in the KLDA algorithm is the Gaussian RBF kernel as is defined as:

\[
K(x, y) = \exp\left(-\frac{||x - y||^2}{2\sigma^2}\right)
\]  

(2.29)

where \(K(x, y)\) denote the kernel matrix of \((N \times N)\) size, \(||x - y||^2\) denote the squared Euclidean distance between the two feature vectors \(x\) and \(y\), and \(\sigma\) a free parameter which can be tuned. The KLDA algorithm has been applied in the extraction and classification of malignant and Benign cluster microcalcification (Wei et al. [2005]). For more information on KLDA refer to appendix E.

### 2.4 Summary

In this chapter, we first introduced *C. elegans* as model organisms suitable for the study of behaviour in animals with known disorder such as gene mutation or knockout or malnutrition at the early stages of their life cycle. Neurobiologist could benefit from experiments that bridge the gap between the neural circuitry responsible for the visible actions of the worms when faced with situations that determines their survival. Information gathered form these experiments are important for the interpretation of neural activities in other organisms that share similar nervous system including humans. Here we considered six types of worms (AQ2947, OW939, OW940, OW949, OW953, and OW956) used previously in the investigation of ageing process in worms, with the aim of categorising them based on shape-based features extracted form their skeleton images. Next, we briefly discussed the features of three standard machine learning models and other modern (deep) learning algorithms used to quantify and characterise worms with distinctive genotypes based on their locomotory behaviour. Furthermore, we explained various evaluation metrics that was used at each stage to measure the quality of the extracted data and the predictions made by the learning algorithms. Finally, we discussed both supervised and unsupervised dimensionality reduction methods used for the extraction of meaningful biological information which act as an approximate representation of the high-dimensional features extracted from the successive images in a worm video file. Both the low and high dimensional features serves as input to the machine learning algorithms to perform the worm classification task.
Chapter 3

Literature Review

3.1 Tracking system requirements

Many automated *C. elegans* tracking systems consists of a microscope, a camera, an agar plate, a motorised stage, and a computer program (Husson et al. [2013]). The camera and the microscope are used to monitor and create the videos of the worm over a period of time based on the scientist research interest. The camera mounted on the microscope is used to produce the video record of the worm as it crawls in the agar plate. In some experiments, the tracking system is designed such that the camera is in-built into the microscope. The magnification of microscope and the resolution of the camera are key components that can highly influence the performance of the tracking device and it can also reduce or increase the number of steps required to process the image data. The motorised stage and the controller program in the tracking system, help to place the worm in the camera field of view and facilitate the feature extraction process. Figure 3.1 shows a worm tracking system. The digital camera is used to create the worm videos with the help of the microscope, the stepper motor and motion controller are used to place the worm under the camera field of view and to track the worm. The petri dish holder is used to position the dish where the worm(s) are placed.

3.2 Worm Trackers

The review done in this work, only considered automated worm tracking systems that are relevant to the current work and therefore, not a complete representation of all existing worm tracking systems. Generally, there are two kinds of automated worm tracking systems: the single-worm tracker and the multi-worm tracker (Husson et al. [2013]). As their names implies, the single-worm trackers are used to monitor and study the behaviour of a single type of worm. On the other hand, the multi-worm trackers are used to investigate multiple worms at the same time. For the single-worm tracker, the magnification of the microscope is usually high since
a single worm is being tracked and all focus is on the same worm. However, for multiple-worm trackers, the magnification of the microscope is usually low because the focus is on multiple worms and there is a need to place each worm in the field of view of the camera. Most tracking systems are developed based on the researchers’ experimental needs (extracting features) or specifications. Dusenbery [1985] develop a multi-worm tracking system to study the behaviour of multiple worms as they react to carbon dioxide (CO2) concentration. This automated worm tracker utilised 10-second windows to compute binary events of motion and turns independently. A few years later, Hardaker et al. [2001] implement the first single-worm tracker that enables the automatic quantification of locomotion behaviour and measurement of features such as velocity, change in direction and egg laying with a precision of 1Hz. Baek et al. [2002] extend the functionality of the single-worm tracker such that, they were able to measure 94 features at 2Hz precision. The capability of the tracker was later increased to measure 161 features at 10Hz. There are several types of multi-worm and single-worm trackers in existence (Wang and Wang [2013] and Yemini et al. [2011]). Examples of single-worm trackers are Worm Tracker 2.0, Nemo (Nematode movement), and CoLBeRT (Control locomotion and behaviour in real time). Examples of multi-worm trackers are the Parallel Worm Tracker (PWT), Opto Tracker, WormLab and The Multi Worm Tracker. Finally, many worm tracking and analysing system softwares are coded in C, C++, Java, LabView 7.0 and MATLAB programming languages (Husson et al. [2013]).

3.3 Worm Detection and Tracking Algorithm

The study of animal behaviour has triggered the use of automated tools that can store, process and analyse high-dimensional data. Therefore, to provide an accurate quantification and characterisation of C. elegans behavioural phenotypes, it is important to automate the image data creation and features extraction processes. In this study we focus on external factors that trigger the change in behaviour of the worms. For example, some factors that can modulate
the behaviour of the nematode worms are chemical, thermal and mechanical stimuli (Stephens et al. [2008]). The worms also respond differently in the presence of these stimuli and therefore exhibit behaviours such as fast movement and body coiling. One determinant used in the study of locomotion behaviour in C. elegans is food (Yemini et al. [2011]). This animal reacts differently in the presence and absence of food. They move very fast in search for food and move slowly when in contact with food. As the worm moves in response to these factors, they exhibit different types of behaviour of which some are subtle to the human eye. More details about the factors that trigger the change in behaviour of C. elegans can be found in Ben-Yakar et al. [2009], Clark et al. [2013] and Omura et al. [2012]. In one experiment, behavioural choice has been studied based on manipulation genes with the use of method known as cell-ablation (Gray et al. [2005]). The manipulated genes introduces observable abnormalities in the animal’s movement pattern which can be tracked, quantified and characterised (Geng et al. [2004]). In another experiment, Yemini et al. [2013a] developed a software for tracking worms called Worm Tracker 2.0 (WT2). The automated tracker software was designed to compute C. elegans behavioural and morphological features. The tracker can measure at least 700 distinct phenotypic features and is available for download at http://www.mrc-lmb.cam.ac.uk/wormtracker/. Recently, Javer et al. [2018a] implemented the tierpsy tracker software which can capture interpretable features as well as extract enough information about the phenotypic differences between the worms been analysed. The software can be used to track, segment and skeletonise raw single worm video files. The tierpsy tracker can measure up to 2000 features, which can be used for strains classification problem. However, they recommended the use of tierpsy.256 which can compute at most 256 features and still yield high classification accuracy. The features extracted by their tracker achieved a better classification accuracy than the features extracted using the WT2 software by Yemini et al. [2013a]. The tierpsy tracker software is available for download at http://ver28.github.io/tierpsy-tracker/.

![Figure 3.2: An overview of a worm tracking procedure (Husson et al. [2013]).](image)

Figure 3.2, illustrate the steps involve in tracking and studying the behaviour of nematode C. elegans. Figure 3.2 A and B each consists of a microscope, a camera, a motorised (x, y) stage
and worms feeding on an agar plate. The difference between the labels A and B is that, in A the camera is physically attached to the microscope, whereas in B the camera is in-built into the microscope and therefore, not detachable from the microscope. Figure 3.2 C represents the video records of the worm feeding on a food patch. Subsequently, the video file is converted into multiple grayscale image frames. Figure 3.2 D depicts the grayscale images and by performing morphological operations on the images, gives the binary images as shown in Figure 3.2 E that is noise free. Furthermore, the application of a specific threshold on the binary images produces the skeletonised images of the worm shape. The label F contain the skeletonised images which are later marked to obtain the label G referred to as the segmented images. Finally, from the segmented images we can extract features and then store them in a database for further processing as shown in Figure 3.2 H. The process of retrieving meaningful information from the image object pixels is referred to as feature extraction. In previous works, the worm to study is cleaned, placed in the agar plate, and allowed to habituate for at least 30 minutes before the start of the recording process. Finally, a standalone computer program performs the feature extraction and analysis tasks and does not require in some cases, a worm tracker (Gray et al. [2005]).

3.3.1 Object tracking procedure

Tracking requires detection of a moving object location over a period in a video stream. In other words, tracking of an object or multiple objects in a video sequence relies upon object detection across multiple frames. Therefore, object detection is a vital step in tracking an object. Object detection is the process of identifying and perhaps retrieving the exact location of an object of interest from an image sequence or video frame. We can describe object detection in two ways: the detection of moving objects using a stationary camera and the detection of moving objects with a moving camera. The first method is less complicated than the second is because, it requires a more robust algorithm to quickly identify the same object location in the current video frame. The simplest way to detect an object in motion with a stationary camera is by using a technique known as background subtraction (Nixon and Aguado [2014] and Shaikh et al. [2014]). This process involves subtracting the image background from the current image containing the moving object and what is left is the moving object known as the foreground. After the retrieval of the foreground pixel map, morphological operations such as erosion, dilation and closing are performed to remove noise or unwanted features while enhancing the image quality. In this approach, the production of the background image is accomplished by averaging images after some pre-defined period. For instance, a pixel at position (a, b) in the present frame $I_t$ is confirmed as foreground if

$$|I_t(a, b) - B_t(a, b)| > T$$  \hspace{1cm} (3.1)

Where $B_t$ in equation 3.1 is the background at location (a, b) and $T$ is a predefined threshold. The background image ($B_T$) is updated utilising:
\[ B_{t+1} = \alpha I_t + (1 - \alpha) B_t \] 

(3.2)

where \( \alpha \) in equation 3.2 is a small value used for corrections. Background subtraction method is less effective if sudden changes in illumination happen. To detect a moving object with a moving camera is quite a complex task. A typical technique utilised for moving object detection with a moving camera is known as temporal differencing. This scenario is different from the method used in the detection of moving objects with the same background using a static camera because, in this case both the image background and camera position are changing with time. Therefore, the background subtraction model is not suitable for this operation. Hence, to detect the moving object, we have to measure the differences between two consecutive image frames (previous \((t_{i-1})\) and current \((t_i)\)). Nonetheless, the movement of the camera and the movement of the object are balanced in the moving camera. In most cases, the movement of the camera is computed first. A limitation to this method is that, it wrongly identifies trailing areas as moving objects especially when an object moves quickly between frames. To overcome this problem, Collins et al. [2000] applied a hybrid algorithm utilising a three-frames differencing model combined with an adaptive background subtraction technique. In this study, a video record produced from a stationary camera was used to determine if an object is moving between consecutive frames. For instance, at a particular time \(t = i\) the intensity value at pixel \(x\) is \(I_i(x)\). The assumption was that, a pixel \(x\) is moving if the intensity has changed significantly between the present frame \((I_i(x))\) and the last frame \((I_{i-2}(x))\), or between the present frame \((I_i(x))\) and the next to the last frame \((I_{i-1}(x))\) as shown in equation F.1:

\[ |I_i(x) - I_{i-1}(x)| > T_i(x) \text{ and } |I_i(x) - I_{i-2}(x)| > T_i(x) \] 

(3.3)

Where \(T_i(x)\) in equation F.1 is the threshold representing a measurable significant intensity changes at pixel location \(x\). A fundamental issue with the frame-differencing model is that pixels inside the moving object body are excluded from the group of “moving” pixels. To solve this problem, an adaptive background subtraction is perform in order to retrieve all “moving” pixels within the boundary of the moving object \((R)\). For instance, if the present background intensity value at pixel position \(x\) is \(B_i(x)\), obtained from series of updates, the “missing” pixels \(b_i\) in the object body can be filled by collecting and comparing all pixels with the region \(R\) that are highly different from the background model.

\[ b_i = \{ x : |I_i(x) - B_i(x)| > T_i(x), x \in R \} \] 

(3.4)

Note that both \(B_i(x)\) and \(T_i(x)\) in equation 3.4 are measurable properties collected from the intensities learned from series of consecutive images. Subsequently, the values of \(B_i(x)\) and \(T_i(x)\) are updates as follows:
Chapter 3 Literature Review

\begin{equation}
B_{i+1}(x) = \begin{cases} 
\alpha B_i(x) + (1 - \alpha)I_i(x), & \text{if } x \text{ is non-moving} \\
B_i(x), & \text{if } x \text{ is moving} 
\end{cases}
\end{equation}

\begin{equation}
T_{i+1}(x) = \begin{cases} 
\alpha B_i(x) + (1 - \alpha)(5x|I_i(x) - B_i(x)|), & \text{if } x \text{ is non-moving} \\
T_i(x), & \text{if } x \text{ is moving} 
\end{cases}
\end{equation}

Where \( \alpha \) is a fixed time that states when new information replaces the former records (equations 3.5 and 3.6). Although this approach yields useful results in detecting and tracking rigid and fast moving objects, however, this model is not suitable for tracking deformable objects. This method is suitable for this research, but will require some modifications. Hence, to develop a tracking system; first, we have to detect the objects of interest in the video frame. Second, we should be able to predict their locations in the next frame by computing the distance they have traveled between two successive frames. This way we can identify each of them. Finally, we use the predicted object locations to create tracks by comparing the detected locations of the object across the image frames.

### 3.3.2 Image processing

Two vital image processing techniques used for the extraction and analysis of most video data are segmentation and skeletonisation techniques. Segmentation is used to find and separate a desired object of interest in an image from the image background. Whereas, skeletonisation is used to extract a simple and more compact representation of the image object. Segmentation techniques are referred to as thresholding algorithms, because threshold are applied to divide the image into foreground (image object) and background (not interested part). On the other hand, most skeletonisation techniques are referred to as thinning algorithms, because the skeletal shape obtained from the binary image are seen as a thin line drawn from the head to the tail (in the case of worm image). To process worm video data, the first operation is to convert the grayscale images to its corresponding binary images. This is done using a user-defined or automatically selected threshold intensity. The applied threshold is used to distinguish between the image object (in this case nematode \textit{C. elegans} worm) and the image background. This threshold is usually calculated from the image statistics. In a study done by Baek et al. [2002] to retrieve the threshold intensity level from an image, they first retrieved the pixel intensity level at the four corners of the video frame based on the assumption that those areas represent the image background. Thereafter, they scanned the entire image using a \( 5 \times 5 \) moving window and calculate the mean and standard deviation of the 25 pixel within the window. If the mean of the pixel within the \( 5 \times 5 \) is less than 70\% of the background intensity level or the computed standard deviation within the window is greater than 30\% of the mean, then that pixel is assigned the value 1 and considered part of the worm’s body. This process of extracting the binary image of the worm create holes in the body of the worm in
some cases. To close the holes in the body of the worm, a morphological closing operation was used (Gonzalez and Woods [2002]). To remove unwanted objects a sequential component labelling algorithm was applied (Jain et al. [1995]). This method of removing unwanted objects involve scanning the entire image in the $x$ and $y$ directions sequentially and at the same time label all object with certain number of connected components. At the end of the scanning operation, every detected objects in the image are removed except the object with the largest size (worm). Yemini [2011] used Otsu binarisation method to retrieve binary images and thereafter, the worm skeleton. In another study, thresholding was applied to extract the binary images of *Drosophila Melanogaster* (fruit fly) and *Caenorhabditis elegans* (worm) in each video frame from the background (Szigeti et al. [2015]). The binary images are later skeletonised to obtain image object skeleton. To extract the worm body skeleton from the binary image, a fast parallel skeletonisation algorithm was used in another related work (Wang et al. [2009]). For a clean image background, the worm skeleton has only two ends (Head and tail). However, in cases where there are spurious branches in the worm skeleton due to noise in the image background, an image fitting technique known as cubic smoothing spline was applied to rectify the problem (Wang et al. [2009]). Let $x_1 < x_2 < \cdots < x_n$ and $y_1 < y_2 < \cdots < y_n$ represent an ordered sets of $x$ and $y$ coordinates on the worm skeleton body respectively. The cubic spline $s$ that minimises the objective function was defined as:

$$f(s) = p \sum_{i=1}^{n} w_i [y_i - s(x_i)]^2 + \int_{-\infty}^{\infty} (d^2 s/dx^2) dx$$

(3.7)

where $p$ denotes a positive smoothing parameter and $w_i$ denotes a positive weight for each coordinate point. Wang et al. [2009] implemented an automated tracking system that is capable of extracting real-time locomotive features of the worm. In order to obtain quality locomotive features, a cubic smoothing spline fitting was also applied. For the image binarisation task, an adaptive thresholding method was applied to the grayscale image. In this binarisation method, each pixel position $(x, y)$ in the image is assigned a local threshold based on the mean of its $m \times n$ neighbours. They suggested the use of a $9 \times 9$ neighbour window and defined the threshold for each point as:

$$T(x, y) = \frac{1}{mn} \sum_{i=-(m-1)/2}^{(m-1)/2} \sum_{j=-(n-1)/2}^{(n-1)/2} f(x+i, y+j)$$

(3.8)

And derived the binary image from:

$$B(x, y) = \begin{cases} 
1 & \text{if } f(x, y) \leq T(x, y) \\
0 & \text{otherwise}
\end{cases}$$

(3.9)

Recently, due to the recent improvement of storage facilities and high speed computing systems, the development of deep learning algorithms for image object segmentation in large
image datasets has increased rapidly. Some of these algorithms has performed very well compared with traditional methods. A major benefit for the use of deep neural networks for image object segmentation is that, it does not require manual image feature extraction which in turn reduces the steps involve in the image preprocessing. However, most of the deep learning segmentation algorithms requires lots of images to train the model in order to achieve a quality segmented image. Deep learning-based image segmentation methods was inspired by the pixel-level prediction algorithms Wang et al. [2018a]. Image object segmentation methods using deep learning has been broadly grouped into two: semantic and instance image segmentations. The semantic segmentation methods segment an image object based on the pixel level and can also group similar object in an image together using the same colour, whereas the instance segmentation perform the image object segmentation based on regions of interest (ROI) and allows the assignment of different colours to the same image object. The first successful application of deep learning for image semantic segmentation was achieved using a fully convolutional network (FCN) (Long et al. [2015]). This deep learning model for image semantic segmentation utilises a supervising learning technique (Liu et al. [2021]) and output an outstanding results. Examples of deep models use for image segmentation are: Fully convolutional network (FCN) (Long et al. [2015]), U-Net (Ronneberger et al. [2015]), DeepLab1 (Chen et al. [2016]), deep generative adversarial network (GAN) (Luc et al. [2016]), an encoder-decoder deep network called SegNet (Badrinarayanan et al. [2017]), combination of U-Net and GAN called SegAN (Xue et al. [2018]), instance segmentation network Masked R-CNN (He et al. [2018]) and a deep active contour model (ACM) (Chen et al. [2019]). For instance, Chen et al. [2019] implemented a deep learning-based active contour model (ACM) for medical image segmentation task. Unlike the traditional active contour models, this deep active contour learning models uses the area inside and outside of the region of interest (ROI) as well as the size of the boundaries in the learning process. Their main contribution was the use of a loss function which takes into consideration the area and size information of the image object of interest and integrate them into the dense deep learning architecture in order to improve the segmentation performance.

Object skeletonisation gives a compact morphological/phenotypical representation that describe the relationship/connection between parts of the object. Shen et al. [2017] suggested a holistically-nested network (HED) with multiple scale-associated side outputs for skeleton extraction. The holistically-nested network is a deep fully convolutional network (FCN)[20], which enables holistic image training and prediction for per pixel tasks. They connected a scale-associated side output to each convolutional layer in the holistically-nested network to address the unknown-scale problem in skeleton extraction. In a related work, the vanilla U-Net architecture (Isola et al. [2017]) was modified for extracting skeletons from binary images. Shen et al. [2016], proposed a scale-associated side output layer, which enables target learning and fusion in a scale-associated way. Therefore, our holistically-nested network is able to localise skeleton pixel with multiple scales. The skeleton scale is basically the distance between a skeleton point and nearest boundary point in the object. They supervised the side output of the network with different scale-associated ground truths, while the ground truth in HED
are the same. Wang et al. [2018] (Wang and Liu [2018]) applied a deep learning-based model for the extraction and recognition of the skeleton of handwritten Chinese characters. A fully convolutional network (FCN) was used to perform the handwritten Chinese characters skeletonisation task and a convolutional neural network (CNN) was used for the skeleton characters recognition task. In another work, the holistically-nested networks (Xie and Tu [2015]) and the scale-associated networks (Shen et al. [2017]) has been successfully used to predict the contour maps and skeleton maps of generic objects in natural images respectively. However, these models do not assure that it is one-pixel width in the output contour or skeleton. Recently, Ko et al. [2021] implemented an end-to-end deep adversarial network for font image skeletonisation known as SkelGAN. This deep skeletonisation model reduces a binary image to a one-pixel width representation. The skeleton generator in SkelGAN output a font character skeleton with a one-pixel width structure and do not need any post-processing techniques. The proposed SkelGAN was trained with a paired dataset which consists of a reference image (original font image) in the reference domain and the ground truth skeleton image in the target domain. The ground truth skeleton images are generated either using a mathematical thinning algorithms by inverse mapping function to generate font from skeletonised font images. Deep learning skeletonisation models requires large amount of annotated training data. The performance of the deep models including the FCN model are affected by the limited amount of the training data (Dai et al. [2016]). To solve this problem, some researchers transform pre-trained deep classifiers into FCNs (Long et al. [2014]). Due to the difficulties encountered in retrieving an object skeleton with a single pixel width, we decided to investigate and implement the classical image segmentation and skeletonisation algorithms mentioned in chapter 2 and discussed in the next chapter (4).

3.4 Feature Extraction Techniques

Feature extraction is a vital component of any pattern recognition and classification system (Wang and Paliwal [2003]). This action help to transform raw data into a meaningful feature vectors. Although video records simultaneously provide information about the worms movement dynamic and capture morphological differences, majority of the pixel intensities in the image frame are background pixels. This implies that, when quantifying region of interest in the image, most part of the image pixel intensities are irrelevant. For this reason, most C. elegans genotype classification problems involve the extraction of features from the video images and subsequently use them to categorise the worm types. However, most of the features extracted from the images are still high-dimensional features and contain some redundant or irrelevant information, hence the need for dimensionality reduction technique. Dimensionality reduction techniques are sets of machine learning algorithms used for the purpose of extracting significant or discriminative features from the data with minimal loss of information. Dimensionality reduction methods make it easier to visualise high-dimensional data sets. These dimension reduction methods remove redundant or irrelevant features from the data.
Dimensionality reduction techniques are classified into two groups: supervised and unsupervised methods. The unsupervised method search for hidden patterns in the data without the guide of a user pre-defined class labels. Examples of unsupervised dimension reduction methods are: Principal Component Analysis (PCA), Non-negative Matrix Factorisation (NMF) and Kernel Principal Component Analysis (KPCA) which is an extension of the PCA method. On the other hand, the supervised dimensional reduction methods search for hidden patterns in a data using user pre-defined class labels. This method uses a function that map each input to specific output label to form an input-output pair. Examples of supervised methods are Linear Discriminant Analysis (LDA) and Kernel Fisher Discriminant Analysis (KLDA). Here, we provide a brief explanation on the logic behind these learning algorithms. However, we deliberately focused on the methods implemented in this work. In this work, the high-dimensional features are the skeleton angle data extracted from each image of the worm body posture, and the low-dimensional features are the principal components (bases) extracted from the high-dimensional skeleton angle data.

Principal component analysis (PCA) is a linear transformation technique used to transform high-dimensional input data into low-dimensional features space with minimal loss of information (Jolliffe [1986]). PCA method utilises the fact that significant information about the features is contained in the directions along which the variations are the largest (Wang and Paliwal [2003]). In this method, the first principal component captures the greatest variation in the original data. For information on PCA implementation refer to appendix A. It has been proven that the different body shapes exhibited by the nematode *C. elegans* can be represented in a low-dimensional space using four (4) PCA dimensions (Stephens et al. [2008]). They demonstrated that with only 4 dimensions in the low-dimensional feature space, we can account for 95% of the worm shape variance. This implies that, any shape displayed by the worm, is a combination of four basis postures. They quantify and derived equations of motion that describes the worm locomotory activity in this low-dimensional space. It was shown that, the worm rapidly exhibit specific set of body postures (referred to as attractors) as a response to stimuli. Examples of these attractors are the omega shape with ventral side orientation and the delta shape with dorsal reorientation (Broekmans et al. [2016]). Based on these familiar modes the worm usually visit, they were able to interpret the worm’s locomotory activity by synchronising stimuli to the worm’s shape space. Stephens et al. [2008] represents the worm skeleton as a curve with length $s$ that passes through the centre of the worm’s body. Thereafter, they described the curve by the tangent angles $(\theta(s))$ measured from the head to the tail of the worm after using 100 equally spaced points on the curve. The covariance matrix of angles was defined as:

$$C(s, s') = \langle (\theta(s) - \langle \theta \rangle)(\theta(s') - \langle \theta \rangle) \rangle$$  (3.10)
where \(<\theta\>\) represents the mean. The eigenvalues \(\lambda_{\mu}\) and its corresponding eigenvectors \(\mu_{\mu}\) are calculated from the covariance matrix using:

\[
\sum_{s'} C(s, s') \mu_{\mu}(s') = \lambda_{\mu} \mu_{\mu}(s) \tag{3.11}
\]

The fractional variance obtained using the \(k\) selected eigenvectors was given as:

\[
\sigma^2_k = \frac{k}{\sigma^2} \sum_{\mu=1}^{k} \lambda_{\mu} \tag{3.12}
\]

where \(\sigma^2 = \sum_{\mu} \lambda_{\mu}\) represents the total variance calculated. It was confirmed that only 4 eigenvalues as referred here as either principal components or eigenworms account for 95% of the total variance. This means, if only \(k = 4\) eigenvalues are significant (or nonzero), then the shape of the worm can be represented as superposition of the eigenworm shapes:

\[
\theta(s) \approx \sum_{\mu=1}^{k} a_{\mu} \mu_{\mu}(s) \tag{3.13}
\]

where \(k\) denotes the number of significant eigenvalues (in this case 4) and the \(a_{\mu}\) denotes the amplitude of motion for each principal components which was defined as:

\[
a_{\mu} = \sum_{s} \mu_{\mu}(s) \theta(s) \tag{3.14}
\]

This work by Stephens et al. [2008] shows how the body postures of the worm can be transformed into meaningful features in a low-dimensional space that describes the worm’s dynamics. These low-dimensional representation of the worm shape have been used to make qualitative descriptions of the worms behaviour (Croll [1975]). In another study, a deep learning model was used to characterised different C. elegans strains based on its locomotory activity using different embeddings for both multiple worms (MW) and single worm (SW) videos (Javer et al. [2019]). The outcome of this investigation outperforms the score achieved with hand-crafted features defined by Yemini et al. [2013b]. Input to the deep model are time series of the worm postures. Linear interpolation technique was used to impute the missing values for the dropped video frames, as a result of failed skeletonisation. To smooth the extracted worm skeleton in space and time, they applied Savitzky-Golay filter. For the multiple worms (MW) videos, autoencoder was used to extract features directly from the images, whereas for the single worm (SW) dataset traditional computer vision techniques were used. Similar to the work of Stephens et al. [2008], they created 49 evenly spaced segments on the single worm skeleton body and then measured 49 tangent angles. The tangent angle between two consecutive points on the skeleton was defined as:

\[
\theta^i_k = \arctan \frac{x_{i+1} - x_i}{y_{i+1} - y_i} \tag{3.15}
\]
where \((x_i, y_i)\) and \((x_{i+1}, y_{i+1})\) represent two coordinate points. The normalise angle was defined as:

\[
\theta_i = \theta'_i - \frac{1}{N_s - 1} \sum_{n=1}^{N_s-1} \theta'_n
\]  

(3.16)

where \(N_s = 49\) represents the total number of segments and \(\theta_i\) represents the corresponding segment angle of \(\theta'_i\). In this work, they used six (6) eigenvectors to account for 98% of the total shape variance. The embeddings were stacked to form time series of postural maps. Here, we only report the classifier used and the scores obtained for the SW videos. A fully connected layer followed by a softmax layer was used to perform the strain classification on their skeleton angle data and hand-crafted features from Yemini et al. [2013b]. The input features were scaled using z-transform which simply subtract the mean from the data and divide it by the standard deviation. The classification result shows that, the best score of 58.44% was obtained when the single worm skeleton angles was used as input. With six (6) eigenworms, the top-1 accuracy obtained was 49.40%, whereas for the manually hand-crafted features the top-1 accuracy achieved was 43.20%. We choose the top-1 accuracy score of 49.40% obtained in Javer et al. [2019] as our baseline score. Javer et al. [2019] has demonstrated that worms with different genotypes can be characterised using each worm postural dynamics alone. In this research, the accuracy score of 49.40% obtained for the single worm genotype classification using the low-dimensional PCA components as our benchmark. In a similar work, Szigeti et al. [2015] describes a novel method that applies unsupervised learning to discover behavioural motifs in two model organisms: larval Drosophila Melanogaster (fruit fly) and Caenorhabditis elegans (worm). They defined motif as a notable sequence of postures that each animal exhibit frequently, and posture was defined as observable components of an animal behaviour (Szigeti et al. [2015]). The changes that occurs in the animal’s postures was represented as an eigenshape time series and analysis was performed on the time series to find motifs. To achieve this, they segmented the eigenshape time series and applied spline regression to cluster the segments. An interesting aspect of their work was that, they also demonstrated that four (4) eigenworms account for 92% of the worm shape variance. Furthermore, posture was mathematically defined as a superposition of the eigenworms as:

\[
Posture(t) = \sum_{i=1}^{n} \alpha_i(t)eigenworm_i
\]  

(3.17)

where \(\alpha_i(t)\) represents the coefficient associated with the \(i\)th eigenworm at time \(t\) and \(n\) represents the number of significant eigenvectors or principal components. This implies that behaviour can be translated as change in posture over time. They also derived an equivalent of the eigenworms for the larval Drosophila known as the eigenmaggots, which also capture above 90% of the fly’s shape variance using four (4) principal components. The angles measured on the fly’s skeleton were restricted to values between \(-\pi < \theta_i \leq \pi\), calculated from 71 points equally spaced on the skeleton. They measures 70 angles for each frame and store the angle data as 70-dimensional vectors. These vectors are concatenated to form an \(n \times 70\) data matrix, where \(n\) is the number of frames. One of the major challenge they encountered was that
certain amount of video frames were dropped due to curled-up postures and the quality of the image background. In this work, we considered different data imputation methods to generate estimate for dropped frames as explained in chapter 5. Berman et al. [2014] develop a method for quantifying postural dynamics that describes the behaviour of Drosophila Melanogaster using mapped stereotype actions (motifs). They start by extracting the fly’s posture from the image and rotate the image in order to avoid rotational invariance. Thereafter, they decomposed the fly’s shape into postural time series and convert the time series into a wavelength spectrograms, which serves as a spatio-temporal representation of the fly’s dynamics within the extracted images. The spectrogram was embedded into two dimensional spectral feature vectors, which was visualise using t-distributed stochastic neighbour embedding (t-SNE). Another important finding in this work is the use of Canny’s method for the edge detection before the application of the thresholding algorithm (Canny [1986]). The Canny’s algorithm makes it easier to identify the fly’s (in our case worms) in the video frames. Currently, feature selection is done using both automated and manual methods. These features are not only interpretable but are also reliable to distinguish behavioural differences between mutant strains and wild-type (Javer et al. [2018b]).

In our case, to measure the high-dimensional angles data, we marked 51 equally spaced points along the worm’s skeleton shape and measured 50 tangent angles. Thereafter, we performed PCA on the angles data after horizontally stacking the angles data to form time series of the worms movement patterns.

The Non-negative Matrix Factorisation (NMF) is an unsupervised dimensionality reduction technique that decomposes a matrix $V$ into two non-negative matrices $W$ and $H$ whose product is an approximation of $V$ (Lee and Seung [1999]). Suppose $V \in \mathbb{R}^{n \times m}$, the dimensions of the two non-negative matrices $W$ and $H$ can be represented as $W \in \mathbb{R}^{n \times d}$ and $H \in \mathbb{R}^{d \times m}$ respectively. The NMF decomposition of the matrix $V$ is expressed in equation 3.18

$$V \approx WH \quad \text{or} \quad (V)_{i\mu} = \sum_{a=1}^{d} W_{ia} H_{a\mu} \quad (3.18)$$

where the $d$ represents number of column in the basis images $W$ and is chosen based on the conditions that $(n + m)d < nm$ or $d < \min(n, m)$. The columns of the matrix factor $H$ represents the required coefficients through which the image object can be represented as a linear combination of the basis image. Lee and Seung [1999] explain the difference between NMF and PCA based on an experiment conducted using a dataset made up of facial images. The aim of the experiment was to discover which (PCA or NMF) is better in terms of learning parts of an image object. Each method was used to find an approximate factorisation of the $V$ from the product of $WH$ as implied in equation 3.18. Lee and Seung [1999] explains that the NMF algorithm is different from the PCA method as a result of the constraints it imposes on the matrix factors $W$ and $H$. In the case of PCA, the algorithm conditioned the columns of the matrix $W$ to be orthonormal and the rows of the encodings $H$ to be orthogonal to each other. This implies that, each face in matrix $V$ is approximated by allowing the linear
combination of all the basis images in \( W \). In other words, each column of \( W \) contributes in the approximation of a face in \( V \) since no columns in \( H \) has zero elements. However, PCA allows the columns of \( W \) and \( H \) to contain both positive and negative values, which in turn results to cancellation of the basis images when representing a face as a linear combination of its basis images (eigenfaces). For this reason, some of the eigenfaces appear distorted when compared with the whole image. In the case of the NMF algorithm, it prohibits the entries of negative values in both matrix factors \( W \) and \( H \). Unlike the PCA method, the non-negativity constraints imposed by NMF permit only addition of multiple basis images to represent a face. In other words, the non-negative constraints of NMF is interpreted as the combination of parts to form a whole. The basis learn by the NMF are localised features of the faces in the database, which can be interpreted as parts of the faces. This property of NMF describes how it learns parts-based representation. For the \textit{C. elegans} movement pattern classification task, the continuous changes that occurs along the body shapes of the worms as they move can be captured by the NMF part-based additive algorithm and utilise to solve the classification problem. The part-based local representation property of the NMF method has made it possible to identify and interpret biological subsystems, which has helped in the detection of functional genetic relationships. \textit{Gyenes and Brown} [2016] derived a compact representation of the worm locomotion using NMF (parts-based representation). They also demonstrated that at least 97\% of the worms shape variance can be account for with only 5 basis shapes. More details on the implementation of the NMF algorithm as well as the initialisation method is found in appendix B.

The Kernel Principal Component Analysis (KPCA) is an unsupervised nonlinear dimensionality reduction technique used for the extraction of low-dimensional features from an input data (\textit{Scholkopf and Smola} [2002] and \textit{Mika et al.} [1999a]). KPCA is an extension of PCA that utilises the kernel methods implemented in the support vector machines domain. In other words, the KPCA algorithm transform an input data to a kernel-defined space and thereafter extract low-dimensional features from the kernel-defined space. For \( X \in \mathbb{R}^{N \times D} \), we define the kernel as:

\[
K = K(x, y) = (\Phi(x), \Phi(y)) = \Phi(x)^T \Phi(y)
\] (3.19)

where \( K \) represent the kernel feature space. Rather than compute the eigenvalues and the eigenvectors in the mapped space \( \Phi(x) \), we compute the projection of a point in the mapped space \( \Phi(x) \) on the \( d \)th principal components \( v^k \) as:

\[
v^d \Phi(x) = \sum_{i=1}^{N} \alpha^d_i \Phi(x_i)
\] (3.20)

where \( d \) denotes the number of principal components and \( \alpha^d_i \) the eigenvector in the mapped space. Since \( \Phi(x)^T \Phi(x) \) represent the dot product in the kernel \( (K) \), compute the eigenvectors and eigenvalues using:

\[
\lambda N a = Ka
\] (3.21)
where \( a \) and \( \lambda \) represent the eigenvectors and eigenvalues of \( K \) respectively. For an input data that is not centred use:

\[
K' = K - 1_N K - K 1_N + 1_N K 1_N
\]

(3.22)

where \( 1_N \) represents a \( N \times N \) matrix and each element is \( \frac{1}{N} \). The projection \( x_{new} \) can be defined as:

\[
x_{new} = K' a / \sqrt{\lambda}
\]

(3.23)

KPCA was used by Scholkopf and Smola [2002] for object recognition. The KPCA method is different from PCA and NMF, because the eigenvalues and eigenvectors are not calculated directly from the input data but in the kernel space. More details on KPCA refer to append C.

Linear discriminant analysis (LDA) is a supervised method used for performing dimensionality reduction in multidimensional dataset (Duda et al. [2001]). The term ”supervised” is used because this method uses the class-labels and the data in the feature extraction or feature selection process. LDA algorithm can be used for dimensionality reduction and to solve classification problems when used as a classifier. Initially, the LDA algorithm was implemented to solve 2-class problem but has now be generalised to solve multiple-class problem as well. The aim of Multi-class LDA is to find the axes that improves the separability between multiple classes. In contrast to PCA, which computes the component axes that maximise the variance of data, the LDA finds the direction (known as linear discriminant) or axes that maximise the separation between multiple classes. In order to categorise different objects in a dataset, Fisher proposed a function that gives a large separation between the projected class means and a small variance within each class, which reduces the overlap between the different classes. This was achieved by maximising the ratio between the between-class variance and the within-class variance. This ratio is defined as:

\[
\hat{W} = \arg \max_w \frac{W^T S_B W}{W^T S_W W}
\]

(3.24)

\( S_B \) denotes the between-class variance and \( S_W \) denotes the within-class variance. For 2-classes \( \hat{W} \propto S_W^{-1} (\mu_1 - \mu_2) \) where \( \mu_1 \) and \( \mu_2 \) represent the sample means for the two classes. For k-classes problem, the problem is solved using eigendecomposition of the matrix-multiplication between the inverse of the within-class and the between-class scatter matrices.

\[
S_B W = \lambda S_W W
\]

(3.25)

Where \( \lambda \) denote the eigenvalues.

\[
W = \max_d (eig(S_W^{-1} S_B))
\]

(3.26)

Here, \( d \) represent the reduced dimension and is at most equal to \( k - 1 \) and the weight vectors or transformation matrix \( W \) represent the \( r \) reduced dimension eigenvectors with the largest eigenvalues. Since \( S_B \) is the sum of \( k \) matrices of rank 1 or less, meaning only \( k - 1 \) of these are
independent and $S_B$ is of rank $k - 1$ or less. This also implies that, only $k - 1$ of the computed eigenvalues are nonzero and the weight vectors $W$ are the eigenvectors of these nonzero eigenvalues. The LDA method is different from PCA, NMF and KPCA techniques, because it uses pre-defined class-labels for the extraction of low-dimensional features from the original input space. Another difference is that, the LDA algorithm is implemented such that the number of classes in the data determines the number of low-dimensional features available. For more information on LDA implementation refer to appendix D.

The Kernel Fisher Discriminant Analysis (KLDA) is a nonlinear supervised dimensionality reduction technique used to extract features from a dataset. KLDA technique applies kernel function to transform the input data to a new feature space and thereafter, uses a supervised dimensionality reduction method such as LDA to extract discriminative features. In other words, the input data is mapped into a new feature space using a nonlinear function such as Gaussian RBF kernel (Mika et al. [1999a]), then linear discriminant analysis is used to extract low-dimensional features useful for pattern recognition or classification. After mapping the input data to the new feature space $F$, the KLDA algorithm decompose the Gram matrix into within ($S_w$) and between ($S_b$) scatter matrices. This leads to the generalised eigenvalue problem defined as:

$$S_b \alpha = \lambda S_w \alpha$$

This problem can be solved by introducing a regularisation method, because the $S_w$ is often singular. $S_w$ is singular because the determinant of the matrix is equals zero. In other word, the matrix does not have an inverse. To solve this problem, we replace $S_w$ with $S'_w$ as defined as:

$$S'_w = S_w + \mu I$$

where $\mu$ is a small positive number and I is an identity matrix with the same dimension as $S_w$.

We can now rewrite equation 3.27 as:

$$(S_w + \mu I)^{-1} S_b \alpha = \lambda \alpha$$

(3.29)

Compute the eigenvectors ($\alpha_i$) and eigenvalues ($\lambda$) from equation 3.29 and select the $d$ number of principal components. The projection of a point $x$ onto the $K$ space is defined as:

$$y(x) = \sum_{i=1}^{N} \alpha_i K(x_i, x)$$

(3.30)

where $N$ represents the number of samples. For more information on KLDA algorithm check appendix E. The KLDA algorithm has been applied in face recognition problems to extract unique facial features (Liu et al. [2004] and Li et al. [2003]). The KLDA algorithm has not be applied in the extraction of C. elegans features. This study tries to investigate the benefit of KLDA algorithm in the analysis and classification of worms with different genotypes.
3.5 Worm classification techniques

Worm classification techniques define how different worm strains with the same measurement features are classified. Initially, worms are often classified subjectively and include lots of pitfalls. Due to the invention of modern technologies, automatic classification and analysis methods are now applied. Nah and Baek [2003] described how different worm types were classified using Classification and Regression Tree (CART). The CART applies a binary classification tree based on the features learnt from the sample. In the beginning, the root node of the CART consists of equal amount of all types of worm mixed together and the aim is to subdivide the data continuously using binary splits such that no two terminal nodes in the tree contain different types of worm. In another work, Geng et al. [2004] applied a Random Forest classification method to classify different worm types. We notice that, most investigation done on genotype classification represent their results using clustering analysis or t-SNE plots (Geng et al. [2003], Berman et al. [2014] and Martineau et al. [2019]). For example, in one of the investigation, they applied \( k \)-means algorithm to classify the behavioural patterns of eight mutant types of \textit{C. elegans} strains (Geng et al. [2003]). In another study, a fully connected layer followed by a softmax layer was used to perform the strain classification on their skeleton angle data and hand-crafted features from Yemini et al. [2013b]. Due to the advance in modern technology, deep learning models such as convolutional neural networks (CNN) are applied in studies that involves \textit{C. elegans}. Deep learning-based image object classification models are used to detect, recognise and classify unique objects in images with the help of a trained deep neural networks. Examples of deep neural networks are LeNet-5, AlexNet, VGG-Nets and ResNet. Deep learning models and their variants have been proven useful in many image analysis and classification tasks.

Javer et al. [2019] implemented a deep learning model that could categorise different strains of \textit{C. elegans} based on video records of their movement patterns using different input representations referred to as embeddings. These embeddings are stacked over time to create the postural maps. The classification accuracy obtained using their model outperformed a different approach in which the input features are hand-crafted. In their method, the input to the deep learning model are time series of the worms postures. The aim of their research was to determine if significant biological motion information can be learned from \textit{C. elegans} video data. The inputs to the deep learning model are features extracted directly from the worm skeletons using classical computer vision techniques (single worm (SW) videos) and features learned using autoencoder on the input images (multiple worms (MW) videos). For the single worm videos, they obtained an accuracy of 49.40%, when features directly obtained from the worm’s postures (PCA time series coefficients) were used as input to the neural networks. This result outperformed the score obtained when hand-crafted features were used as input to the same model. Furthermore, when the same sets of hand-crafted features were used for the MW dataset based on their selected strains, high performance accuracy was reported and no explicit explanation was given for this improvement. The parameters of the deep learning model used for the SW dataset experiment are: learning rate \(10^{-5}\) and was reduced to \(10^{-4}\),
a momentum of 0.9, weight decay of $10^{-4}$, and the optimiser was SGD (Stochastic Gradient Descent). Two-third of the data was used for training and the rest for testing.

Lin et al. [2020], utilise CNN to determine the physiological age of multiple *C. elegans*. In their research, raw images serves as inputs to 5 different CNN models such as 50-layer residual network (ResNet50), inceptionV3, inceptionResNetV2, 16-layer visual geometry group network (VGG16) and MobileNet, and the output from these networks are the ages of the wild-type worms in days. Each CNN models were trained with 70% of randomly selected images in the dataset, validated with 20% of the data and their performance tested with 10% of images in the set. In another approach, due to lack of data related to age changes in *C. elegans* worms, they added another global feature referred to as curved or straight and applied only the top 2 CNN models (inceptionv3 and inceptionv2) models in the first experiment separately. The reason for the additional attribute was to reduce the number of incorrect matches of both the straight and curvy *C. elegans* worms in order to improve the CNN object recognition performance. They applied two types of performance test: for regression analysis and for logistic regression. For the logistic regression, the inputs are *C. elegans* image and the curved or straight attribute, and the output is a discrete value of age. For the regression analysis, the input to the CNN model are *C. elegans* images and the attribute curved or straight, and the output is a continuous value of age. However, attribute curved or straight was manually added depending on the shape of the worm in the microscopic image. In both cases, the rectified linear unit (ReLU) was applied in all layers as the activation function. They achieved an accuracy of 89.92% with the regression analysis and 89.13% with logistic regression. Although both approaches yield high classification or recognition accuracy, however these methods does not take into account the changes that occurs in the different segments of the worm body when they move. In other words, their methods does not consider the worms genotypes but their morphological features. As mentioned earlier, their approach also use a manual method to input an attribute into the CNN models, which may be time consuming for a large image or video dataset.

García Garví et al. [2021], implemented an automated model that combines traditional computer vision techniques with convolutional (ResNet18) and recurrent neural (LSTM) networks to determine the state (alive or dead) of *C. elegans* strains from low-resolution image sequences. The traditional computer vision techniques are used to extract the image sequences, perform segmentation, detect the movement of worms at the edges and post-processing. On the other hand, the convolutional neural network (ResNet-LSTM) was used to extract feature maps from the images such as segmentation error or dirt in the worm plate or aggregation, which in turn are passed to the fully connected layer for the classification task. A limitation to this approach is that, some images (1108) are manually labelled in order to train the classifiers and is time-consuming for a larger image datasets. In this experiment, a worm is considered dead if it remains in the same position and posture for more than a day; otherwise it is considered alive. To achieve this, three images sequences are compared: the current day image, image before the current day and image after the current day. For the cascaded classifiers, the ResNet18 was used for feature extraction while the LSTM was used to obtain temporal (dynamic) information.
Both the extracted features and the temporal information serves as input to the fully connected layer that does the alive and death categorisation. The original dataset contain 108 image sequences with 10 to 15 *C. elegans* worms in each Petri dish. To evaluate the performance of the suggested model, they used confusion matrix as the validation metrics.

### 3.6 Summary

From the investigation carried out, most literature deals with different aspect of quantifying the worm’s morphological skeleton feature extraction and characterisation methods. The findings also revealed that, to obtain quality or high throughput features from the video data, robust binarisation and skeletonisation algorithms must be used. For example, in this work we applied the widely used Otsu’s method for the conversion of the grayscale image into its corresponding binary image. Although, in the next chapter we described other binarisation algorithm suitable for transforming grayscale images into its binary equivalent. We also discovered that most literatures did not explicitly describe the skeletonisation (thinning) algorithm used for the skeletal shape extraction. Although Wang et al. [2009] did mention the used of fast parallel algorithm, but the algorithm was not described. For these reasons, we also investigated different types of thinning algorithms in chapter 4 and choose the most effective methods. As regards to the choice of features suitable for the classification of worms with different genotypes, there are no specific rules that determine the set of phenotypes to focus on when characterising *C. elegans* worms. Furthermore, deep neural networks such as CNN has shown remarkable performance in many computer vision application that involves image object segmentation, skeletonisation and classification. In this work, we applied performance evaluation metrics such as t-SNE, ROC curve, accuracy and confusion matrix in order to classify the worm types and as well as evaluate each machine model performance. For the genotypes classification problem, we first transformed the high-dimensional angles data to its corresponding low-dimensional representation methods and arrange the vectors as time series for each movie. This time series serves as input to the machine learning classifiers. Second, we represent the angle data as time series of the skeleton angles for each movie before the application of dimensionality reduction methods such as PCA, NMF, KPCA, LDA and KLDA on each time series. The low-dimensional representation of each principal component then serves as input to the machine learning algorithms. Although there are other dimensionality reduction techniques, but we cannot clearly say which of these representations is most informative for the analysis and classification of worm with different genotypes. For the worm classification problem, the goal is to achieve high classification performance using these different low-dimensional features and then compare the accuracy scores obtained to determine which is better in terms of representation. However, the accuracy score alone is not sufficient if the low dimensional features cannot be interpreted. To analyse the worms with different genotypes both (Stephens et al. [2008] and Gyenes and Brown [2016]) used a low-dimensional representation of the worm body postures (tangent skeleton angles measured along the worm’s body). These data-driven
<table>
<thead>
<tr>
<th>Task</th>
<th>Useful techniques</th>
<th>Techniques applied</th>
</tr>
</thead>
</table>
| 1. Object detection | - Canny edge detection  
- Ridge | - Canny edge detection |
| 2. Image Segmentation | - Semantic segmentation  
- Active contour  
- Otus algorithm  
- Sauvola algorithm  
- Kapur algorithm  
- Niblack  
- Adaptive | - Otus algorithm  
- Sauvola algorithm  
- Kapur algorithm  
- Niblack  
- Adaptive |
| 3. Image Skeletonisation | - Zhang and Suen algorithm  
- Guo and Hall algorithm  
- Stentiford algorithm  
- Hilditch algorithm | - Zhang and Suen algorithm  
- Guo and Hall algorithm  
- Stentiford algorithm  
- Hilditch algorithm |
| 4. Feature extraction | - Manually defined  
- Principal Component Analysis (PCA)  
- Non-negative Factorisation Matrix (NMF)  
- Kernel Principal Component Analysis (KPCA)  
- Independent Component Analysis (ICA)  
- Linear Discriminant Analysis (LDA)  
- Linear Discriminant Analysis (KLDA) | - PCA  
- NMF  
- KPCA  
- LDA  
- KLDA |
| 5. Classification | - ResNet  
- Support Vector Machine (SVM)  
- Random Forest (RF)  
- $k$-Nearest Neighbour ($k$NN)  
- CART | - ResNet  
- SVM  
- RF  
- $k$NN |
| 6. Missing values interpolation | - Quadratic spline  
- Linear  
- Mean  
- Principal Component Analysis (PCA) | - Quadratic spline  
- Linear  
- PCA |
approach to dimensionality reduction depends mostly on the dataset used to train them. This implies that the basis shapes derived may appear different for a different training set especially when the sample size is small.

Finally, to the best of our knowledge, the classification of *C. elegans* with different genotypes has not been performed using low-dimensional discriminative features derived from the application of KPCA, LDA and KLDA techniques. For the worms genotype classification problem, we used three standard machine learning algorithms (SVM, RF and $k$NN) and we used the deep neural network implemented by Javer et al. [2019], with a little modification to the deep classifier output.
Chapter 4

Video Frames Preprocessing

4.1 Introduction

The detection and extraction of relevant information from successive worm images in a video file is a vital step that determines how accurate a machine learning algorithm can interpret the extracted features or categorise the features into meaningful classes. For this reason, image processing is an important step in every image object recognition and classification problem. It involves the preparation of the image for the extraction of desired features. To retrieve meaningful information from the raw single worm images, we ensure that each image frame is of good quality by removing and suppressing all unwanted features. In this chapter, we considered several image processing techniques use for image object segmentation and skeletonisation task. In Figure 4.1, we outline the main stages required for the single worm feature extraction and classification. We explained all stages in Figure 4.1, except the worm classification task which is implemented in the next chapter. The first stage involves the selection of the single worm video files required for the worm genotype classification task. In the next stage, successive still images are retrieved from each worm video file. To separate the object or region of interest (ROI) in the image from the background, requires image segmentation operation. The image skeletonisation stage involves reducing the size of the object of interest to a thin line, but still maintain the properties of the original image object. Here, image object segmentation and skeletonisation are two important image preprocessing steps used for finding and extracting a region of interest from a given image for further processing or classification. These image processing techniques have been applied in numerous vision-based systems, shape-based recognition and classification problems (Chaki et al. [2014]). Examples of applications where these two methods have been utilised includes video object detection and tracking, medical image analysis and classification, gesture recognition, face recognition, digital handwriting recognition and classification. Therefore, in this chapter we discuss and compare different image object segmentation and skeletonisation algorithms. To select robust algorithms for our worm shape segmentation and skeletonisation task, we use different performance metrics to evaluate the quality of the segmented and skeletonised image object obtained. Finally, we explain the
approach used in this work for finding the region of interest and the extraction of meaningful shapes that gives an approximate representation of the original image objects in our video data.

![Figure 4.1: Pipeline for worm video analysis and classification](image)

### 4.2 Worm videos collection

The high-resolution single worm videos used in this work were carefully collected from the open worm movement database (http://movement.openworm.org). The video data set consists of 1200 videos of 6 types of worms with different genotypes. There are 200 videos per worm type and the length of the videos varies from 5 to 15 minutes at the rate of 25 frames per second. These worms are tracked over a time period with a camera mounted on a microscope as the worms move around in a food patch. Five out of the six worms (OW939 (zgIs113[Pdat-1::alpha-Synuclein::YFP]), OW940 (zgIs128[Pdat-1::alpha-Synuclein::YFP]), OW949 (zgIs125[Pdat-1::alpha-Synuclein::YFP]), OW953(zgIs138[Pdat-1::YFP]), and OW956(zgIs144[Pdat-1::YFP])) are categorised as mutant types (Martineau et al. [2019]) and the sixth worm AQ2947 is categorised as a wild-type worm. As mentioned earlier, *C. elegans* with mutations in their genetic architecture are referred to as the mutant types and they posses abnormalities in their nerve system. These abnormalities usually occur as a result of mixed effects-gene or environment, nature or nurture, and can also be created in the laboratory through a process called transgenesis. Transgenesis is the process in which foreign genetic substance are fused into the genome of the host cells (Chien [2004]). The mutant worms were created via the insertion of alpha-synuclein to yellow fluorescent protein (YFP) which is attached to the 8 dopaminergic neurons of the worm (Martineau et al. [2019]). The alpha-synuclein is an aggregated-prone protein
that causes Parkinson’s disease (neurodegenerative disorder) in human. In the strain names, the Pdat-1 (Phospholipid:diacylglycerol acyltransferase 1) denotes the promoter and the YFP serves as the acceptor for the genetic-coding sequence of the gene.

The wild-type or healthy worms, are worms without any engineered defects in the nerve system, whereas the mutant worms has engineered defects in their nerve system. For the purpose of this work and to balance the differences in the length of the videos, we choose to extract an equal number of frames from each video. This approach helps to prevent any form of bias or imbalance in the dataset which may also lead to bias in the data analysis phase. The size of each video frame is $480 \times 640$ pixels. The video data is divided into training and test sets. There are 840 videos in the training set and 360 videos in the test set. We ensure that there are 140 videos of each worm in the training set and 60 videos per worm type in the test set. For each video, we extracted 400 successive video frames taking into consideration dropped video frames that could not be segmented or skeletonised. The dropped frames are initialised with the “NaN” value and then estimated using data imputation methods during the data analysis phase. We ensure that, frames from the same worm movie is used only for training or testing to avoid information leakage. Based on the number of frames extracted per video, the training (70% of video frames) and testing (30% of video frames) sets contain 336000 and 144000 video frames respectively. Figure 4.2 gives an illustration of the train-test separation method used.

![Figure 4.2: Worm videos split into training and test sets, where k denote the split point, N denote the number of videos per strain and p denote the number of strains in the dataset](image)

Furthermore, in order to carry out more experiments, we split each worm video file in the dataset into two equal parts and used them as two separate videos with the same class label. By splitting each video record into two, we have increased the number of observations in the dataset from 1200 to 2400. This train-test separation method can be considered as a form of data augmentation, but has its own benefit and limitation when used to classify different strain types. The limitation for splitting a single video file into two video files with the same label is that, if the worm remain in the same position and posture for a very long time both files will contain the same information and therefore introduce redundancies in the data. This implies that, it possible to have both data points created from the same video in either the training set or testing set. However, the benefit of splitting the video into two parts is to provide more data points which allow the classification models to learn from a large number of shape-based features.
4.3 Image object detection

Before we can track the worms, first we have to find them in the video record. Image preprocessing is an important phase in an automated tracking system and it is the next stage after collection of the recorded video file. For recorded video, the starting point will be the extraction of multiple image frames from the video using customised software and then converting them to grayscale images. These images can be stored in different file formats and may contain some noise. With image processing operations (filtering and thresholding), we can transform the grayscale images into binary images. After the removal of any unwanted features or noise from the image using a Gaussian filter, the grayscale image are transformed into a binary image (black or white). The main reason for the conversion is to separate the image foreground (desired image object) from the image background.

4.3.1 Canny edge detection

Canny edge detection operator is a popular and powerful algorithm used in detecting edges in an image Canny [1986]. There are four main stages in canny edge detection (Nixon and Aguado [2014] and Gonzalez et al. [2004]):

1. Removal of noise from the image by smoothing it using a Gaussian filter.

2. Calculate the edge gradient \( g(x, y) = \sqrt{G_x^2 + G_y^2} \) and edge direction \( g(x, y) = \tan^{-1}(G_y/G_x) \). Here, \( G_x \) and \( G_y \) represent the x and y derivatives of the points in the image being considered. By using Sobel or Prewitt or Robert technique, we can obtain the values of \( G_x \) and \( G_y \). This operation helps to identify those points or edges where the strength of the gradient changes are high in the direction of gradient.

3. The edges identified in step 2 above forms ridges in the gradient magnitude image. With the use of non-maximum suppression algorithm (thinning algorithm), those points that are not part of the ridge are set to zero, while those on top of the ridges are retained. Thereafter, the ridges retained are thresholded by applying two thresholds, the upper threshold (UT) and the lower threshold (LT). Edges with pixel values above UT are recognised as strong edges otherwise, they are referred to as weak edges.

4. The final step is to connect strong edges together and suppress all weaker edges not connected to strong edges.
Chapter 4 Video Frames Preprocessing

Figure 4.3: Image object detection using Canny edge detection algorithm

(a) Worm image
(b) Canny edge detection

The main purpose of the Canny edge detection is to pick out or isolate the detected object from the image background. We applied the Canny edge detection operator to identify the worms in each successive still image frames. Figures 4.3a and 4.3b, show the original image frame and the application of Canny edge operator on the image object respectively. It is usually applied during the image object segmentation and useful for the removal of unwanted objects in images.

4.4 Image object segmentation

Image object segmentation is a crucial preprocessing step in any image pattern or shape recognition applications. Segmentation is the process of finding and separating a desired object of interest in an image from the image background. This object of interest in the image is often referred to as the foreground pixels and the regions of no interest can be termed as the image background. Furthermore, the desired image object or region of interest in a given image usually contain meaningful properties of the image that are required to analyse and solve a domain specific problem. Therefore, image segmentation operations are used to find and extract vital image objects in several classification applications. Image segmentation operations are also referred to as image binarisation operations, and there are several traditional and deep learning-based binarisation algorithms suitable for different kinds of segmentation tasks. Binarisation operations are used to segment objects in a grayscale image into two distinctive set of pixels known as the foreground pixels (areas of interest or targeted object) and the background pixels (areas not interested in). In some traditional or classical methods, binarisation operations are achieved by applying a certain threshold value to the grayscale image. The threshold value which in most cases can be obtained from the grayscale image is used to distinguish between the light and dark regions in an image based on pixel intensities. Hence, pixel values below the selected threshold value are set to zeros (0) and pixels values equal to or greater
than the threshold values are set to one (1) or vice versa. On the other hand, deep learning-based image binarisation operations apply deep neural networks to perform the image object segmentation task. The deep learning methods, accept images as input and they return the corresponding binary or segmented images as output. As mentioned in Chapters 2 and 3, we did not implement any of the deep learning-based segmentation methods in this work because of lack of training data and the nature of the features to be extracted. The use of binary images rather than grayscale images in image recognition or classification tasks is that, it reduces the computational load required to process large image dataset. A good binarisation algorithm can also improve the accuracy of image object recognition in most pattern recognition tasks. The binarisation operations can be broadly classified into two groups: global thresholding and local thresholding. For global thresholding or global binarisation, a single derived or selected threshold value is used to process all pixels in a particular image. An example of global thresholding algorithm is the Otsu thresholding algorithm (Nobuyuki [1979]). In the case of local thresholding or local binarisation, different sets of threshold values are used to process different regions in the same image. Examples of local thresholding algorithms are Niblack method (Niblack [1986]), Sauvola method (Sauvola and Pietikainen [2000]), local adaptive method by Bernsen [1986], and Kapur entropy method (Kapur et al. [1985]). These algorithms will be discussed later in this chapter. Chaki et al. [2014] describe a simple approach that can be used to compute a threshold value for a particular segmentation task:

1. Set an initial threshold value $T$. It was suggested that an initial estimate for $T$ should be the average of the minimum and maximum grayscale intensity values in the image.

2. Use the initial threshold value $T$ to segment the image. This operation produce two sets of pixels: the first set $S_1$, consists of all pixels with intensity values equal to or greater than the initial threshold value ($S_1 \geq T$) and $S_2$ consists of pixels with intensity values less than the threshold ($S_2 < T$).

3. Calculate the average intensity values $\mu_1$ and $\mu_2$ for each group of pixels $S_1$ and $S_2$ respectively.

4. Calculate the new threshold value defined as

$$T = \frac{(\mu_1 + \mu_2)}{2} \quad (4.1)$$

5. Repeat steps 2 to 4 until the threshold value $T$ is less than a predefined parameter $T_0$ for each successive iterations.

However, this simple method of obtaining a threshold value is not always robust for many image preprocessing tasks. Also, selecting a wrong threshold value may affect the overall performance of the system. For this reason we choose to discuss and compare a few but well known image binarisation algorithms here. The usefulness of the thresholding operations are domain based and may not be suitable for all image processing applications.
4.4.1 Otsu Thresholding Algorithm

The Otsu thresholding algorithm (Nobuyuki [1979]) is the most popular, successful and frequently used global binarisation operation in image analysis applications. Nobuyuki suggests that the optimal threshold is selected based on discriminant criteria (Nobuyuki [1979]). This method takes into consideration the zeroth and the first-order cumulative moment of the gray-level histogram. The steps used by Nobuyuki in the automatic selection of a threshold value is defined as:

1. Let the pixels of a given image be given as \( L \) gray levels \([1, 2, \ldots, L]\). Assumed that the number of pixels at level \( i \) is represented by \( n_i \) and the sum of the pixels in the image be given as

\[
N = n_1 + n_2 + \cdots + n_L
\]  

(4.2)

2. Normalise the gray level histogram and regard it as a probability distribution using:

\[
P_i = \left( \frac{n_i}{N} \right)
\]  

(4.3)

such that \( P_i \geq 0 \) and \( \sum_{i=1}^{L} P_i = 1 \)

3. Divide the image pixels into two classes \( C_0 \) and \( C_1 \) (background and foreground (object)) by a threshold at level \( k \). Assumed \( C_0 \) represents pixels with level \([1, 2, \ldots, k]\) and \( C_1 \) represents pixels with levels \([k+1, k+2, \ldots, L]\). Calculate the probabilities of class occurrence and the class mean for both classes as:

\[
\omega_0 = Pr(C_0) = \sum_{i=1}^{k} P_i = \omega(k)
\]  

(4.4)

\[
\omega_1 = Pr(C_1) = \sum_{i=k+1}^{L} P_i = 1 - \omega(k)
\]  

(4.5)

and

\[
\mu_0 = \sum_{i=1}^{k} iPr(i|C_0) = \frac{\sum_{i=1}^{k} iP_i}{\omega_0} = \frac{\mu(k)}{\omega(k)}
\]  

(4.6)

\[
\mu_1 = \sum_{i=k+1}^{L} iPr(i|C_1) = \frac{\sum_{i=k+1}^{L} iP_i}{\omega_1} = \frac{\mu_T - \mu(k)}{1 - \omega(k)}
\]  

(4.7)

where \( \omega(k) = \sum_{i=1}^{k} P_i \) and \( \mu(k) = \sum_{i=1}^{k} iP_i \) are zeroth and first-order cumulative moments of the histogram up to \( k \)th level respectively and \( \mu_T = \mu(L) = \sum_{i=1}^{L} iP_i \) is the total mean level of the original image.

4. For any choice of \( k \), verify that the following constraints are true \( \omega_0 \mu_0 + \omega_1 \mu_1 = \mu_T \) and \( \omega_0 + \omega_1 = 1 \)
5. Compute the class variance for both the background and object using:

\[ \sigma_0^2 = \sum_{i=1}^{k} (1 - \mu_0)^2 P_T(i|C_0) = \frac{\sum_{i=1}^{k} (1 - \mu_0)^2 P_i}{\omega_0} \] (4.8)

\[ \sigma_1^2 = \sum_{i=k+1}^{k} (1 - \mu_1)^2 P_T(i|C_1) = \frac{\sum_{i=k+1}^{L} (1 - \mu_1)^2 P_i}{\omega_1} \] (4.9)

6. To determine the goodness of the threshold (at level k), the following measurement discriminant criteria are used:

\[ \lambda = \frac{\sigma^2_W}{\sigma^2_B} \]

\[ k = \frac{\sigma^2_T}{\sigma^2_W} \]

\[ \eta = \frac{\sigma^2_B}{\sigma^2_T} \] (4.10)

where

\[ \sigma^2_W = \omega_0 \sigma^2_0 + \omega_1 \sigma^2_1 \] (4.11)

\[ \sigma^2_B = \omega_0 (\mu_0 - \mu_T)^2 + \omega_1 (\mu_1 - \mu_T)^2 = \omega_0 \omega_1 (\mu_1 - \mu_0)^2 \] (4.12)

\[ \sigma^2_T = \sum_{i=1}^{L} (i - \mu_T)^2 P_i \] (4.13)

are the within-class variance, the between-class variance and total variance of levels, respectively.

7. Compute the optimal threshold that maximises the objective functions in equation \( \eta(k) = \frac{\sigma^2_B(k)}{\sigma^2_T} \). \( \sigma^2_B(k) = \frac{(\mu_T - \mu(k) - \mu(k))^2}{\omega(k)[1 - \omega(k)]} \) and the optimal threshold \( K^* \) is\( \sigma^2_B(k^*) = \max_{1 \leq k \leq L} \sigma^2_B(k) \). The range of \( k \) over which the maximum is determine is restricted to \( S^* = \{ k : \omega_0 \omega_1 = \omega(k)[1 - \omega(k)] > 0 \text{ or } 0 < \omega(k) < 1 \} \). Note that, minimising the within-class variance is the same as maximising the between-class variance.

These seven steps can be summarised as:

1. Compute histogram and probabilities of each intensity level
2. set up initial \( \omega_i(0) \) and \( \mu_i(0) \)
3. Step through all possible thresholds \( k=1 \) to the maximum intensity
   (a) Update \( \omega_i \) and \( \mu_i \)
   (b) Compute \( \sigma^2_B(k) \)
4. The desired threshold corresponds to the maximum \( \sigma^2_B(k) \)
For multithresholding, Nobuyuki suggests the selection of multiple \( k \) values. For instance, in the case of three-thresholding we should initialise two thresholds such that: \( 1 \leq k_1 \leq k_2 < L \) for three different classes, \( C_0 \), for three different classes, \( C_0 \) for \( [1, \ldots, k_1] \), \( C_1 \) for \( [k_1 + 1, \ldots, k_2] \) and \( C_2 \) for \( [k_2 + 1, \ldots, L] \). The criterion measure \( \sigma_B^2(\eta) \) depends on the function of two variables \( k_1 \) and \( k_2 \), and the optimal set of threshold \( k_1^* \) and \( k_2^* \) is selected by maximising \( \sigma_B^2 \):

\[
\sigma_B^2(k_1^*, k_2^*) = \max_{1 \leq k_1 < k_2 < L} \sigma_B^2(k_1, k_2) \tag{4.14}
\]

Figure 4.4 shows the region suitable for the selection of a threshold value using the Otsu algorithm. Figure 4.4 A represents the grayscale image extracted from the video record. Figure 4.4 B depicts the histogram of the grayscale image, which indicates that the image is bimodal. The histogram shows that the grayscale image consists of two modalities: the worm pixels is represented by the dark modality and is enclosed with the green ellipse, whereas the grayscale image background is represented by bright modality which is enclosed with the blue ellipse. The value of the threshold that optimally splits the image into two modalities is indicated with the red arrow. The threshold value minimises the variance within the modalities and at the same time maximise the variance between the modalities. Figure 4.4 C shows the application of the selected threshold value to the grayscale image to retrieve the worm shape (foreground pixels).
In this study, we applied the same algorithm in calculating the threshold values for each image and this method works even for low quality videos.

4.4.2 Niblack Thresholding Algorithm

The Niblack’s algorithm (Niblack [1986]) is a local thresholding method that computes pixel-wise threshold values using a rectangular sliding window over the grayscale image. This algorithm selects the thresholds based on the local mean $m(i, j)$ and standard deviation $\sigma(i, j)$, and returns threshold values of window size $b \times b$. The threshold $T$ is defined as:

$$T(i, j) = m(i, j) + k\sigma(i, j)$$  \hspace{1cm} (4.15)

where $k$ is a fixed value within the range of 0 to 1 and it determines the amount of the object edges to preserve. The performance of this algorithm is largely influenced by the size of the sliding window and the value of $k$ used. In several applications, a sliding window of $b \times b$ (with $b=25$) and $k = 0.6$ seems to produce optimal result (Chaki et al. [2014]). It was suggested that size of the sliding window should be small enough to capture the local illumination level and large enough capture both the image object and the image background.

4.4.3 Sauvola Thresholding Algorithm

Sauvola’s algorithm is also a local thresholding algorithm that computes the threshold values using a local-variance-based method (Sauvola and Pietikainen [2000]). This method acts as an improved version of the Niblack method especially for images with unevenly distributed illumination or light texture background. The Sauvola method modifies Niblack’s method by adding a constraint to the contribution of the standard deviation $\sigma$. The Sauvola’s formula is defined as:

$$T(i, j) = m(i, j) \times [1 + k(\sigma(i, j)/R - 1)]$$  \hspace{1cm} (4.16)

where $k$ is a constant value, $m(i, j)$ and $\sigma$ denote the mean and standard deviation of the entire window respectively. It was suggested that values of $k = 0.5$ and $R = 128$ yield a better result. It was also discovered that, the value of $R$ has negligible effect on the quality of the method’s performance. Similar to Niblack’s algorithm, the value of $k$ and the size of the rectangular window affects the algorithm’s performance significantly. A good binary image is obtained by selecting an optimal combination of $k$ and the rectangular window size.

4.4.4 Adaptive Thresholding Algorithm

The adaptive algorithm is a local thresholding method that utilises the image contrast and was suggested by Bernsen [1986]. In this implementation, the threshold value was defined as the middle or median value between the mean of the minimum $I_{\text{min}}(i, j)$ and the maximum...
\( I_{\text{max}}(i,j) \) gray-level pixel values within a given local window. It was suggested that an optimal local window size is \( w = 31 \). In this method, if the contrast \( C(i,j) = I_{\text{max}}(i,j) - I_{\text{min}}(i,j) \) is below a given contrast threshold \( k \), all pixels within the window under consideration are either set to zero (background) or set to one (foreground) based on the class of pixels that best describe the local window. This method is significantly influenced by the value of \( k \) and the local window size \( w \). The threshold \( T \) is defined as

\[
T(i,j) = 0.5 \times (\max_w(I(i,j)) + \min_w(I(i,j)))
\]  (4.17)

where \( \max_w(I(i,j)) \) and \( \min_w(I(i,j)) \) denote the maximum and minimum image gray-level values within the local window respectively, and \( w \) represents the local window size of \( w \times w \).

### 4.4.5 Kapur Thresholding Algorithm

This local threshold algorithm (Kapur et al. [1985]) focuses on entropies between the image foreground and background considered as two distinctive sources of signal. The optimal threshold is obtained when the sum of the two classes of entropies reaches its maximum. This method utilizes two probability distributions derived from the original gray-level to represent the image foreground and background distributions. For a threshold value \( t \), let \( P(\leq t) \) and \( P(>t) \) represent the background (below threshold) and foreground (above threshold) probability distributions respectively. These probabilities are defined as:

\[
P(\leq t) = \sum_{i=0}^{t} P_i
\]  (4.18)

\[
P(>t) = \sum_{i=t+1}^{l} P_i
\]  (4.19)

and the corresponding entropies of the distributions below and above the threshold are defined as:

\[
E_1(t) = - \sum_{i=0}^{t} P_i P(\leq t) \ln \frac{P_i}{P(\leq t)}
\]  (4.20)

and

\[
E_2(t) = - \sum_{i=t+1}^{l} P_i P(>t) \ln \frac{P_i}{P(>t)}
\]  (4.21)

The optimal threshold \( t^* \) that will produce a good binary image is taken to be the gray-level value that maximizes the sum of the two entropies \( E_1(t) + E_2(t) \) i.e \( t^* = \arg\max[ E_1(t) + E_2(t) ] \) where \( t \) is drawn from all gray values in the image.
Image object segmentation (binarisation) is an important task in biological image analysis. Therefore, a robust image segmentation technique will improve the image object recognition and classification task. It is difficult to select a particular image segmentation technique that is suitable for all image analysis and classification task. Hence the need to evaluate the performance of different image segmentation techniques and determine the best method applicable for a specific task. In this study, we considered seven performance measures popularly use for the evaluation of image segmentation techniques. The evaluation measures are mean squared error (MSE), recall, precision, F-measure, signal-to-noise ratio (SNR), peak signal-to-noise ratio (PSNR) and pixel error rate (PERR). To perform the evaluation, two images are required: a binary image and a reference image. The binary image is usually obtained from using any of the binarisation methods, whereas the reference image is created based on a majority voting.
scheme that depends on the output of different segmentation methods Chaki et al. [2014]. The following steps are taken to create the reference image:

1. Create the binary images using the binarisation methods such as Otsu, Niblack, Sauvola, Kapur, and Adaptive thresholding algorithms.

2. To obtain the reference image, consult all the binarised images pixel by pixel.

3. Set each pixel in the reference image to 1 if the majority of the binarisation methods output agrees that the pixel position considered is a 1. Otherwise, set the pixel value to 0 in the reference image.

4. The reason for creating the reference image using the various binarisation techniques is that, there is no ground truth *C. elegans* binary image for the performance measurement.

### 4.4.7 Recall, Precision, and F-measure Metrics

To compute the recall, precision and F-measure scores, let $N_f$=total number of object pixels in the reference image, $N_b$=total number of object pixels in the binary image, $A$=number of common pixel in both the reference and binary images, $B$= number of object pixels in the reference image that are not in the binary image $= N_f - A$, and $C$= number of object pixels in the binary image that are not in the reference image $= N_b - A$.

The recall is defined as:

$$\text{recall} = \frac{A}{A + B} \times 100 \quad (4.22)$$

The precision is defined as:

$$\text{precision} = \frac{A}{A + C} \times 100 \quad (4.23)$$

The $F$-measure is defined as:

$$F - \text{measure} = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}} \times 100 \quad (4.24)$$

The higher value of recall, precision and F-measure implies a good image object segmentation performance. The recall, precision and F-measure performance scores are express in percentage.

### 4.4.8 MSE, PERR, SNR and PSNR metrics

To describe how the parameters $MSE$, $PERR$, $SNR$ and $PSNR$ are computed from the reference image $R(i,j)$ and binary image $B(i,j)$, let $i$ and $j$ denote the $i$th row and the $j$th column pixel locations in both images respectively. If the sizes of both the reference and binary
images is given as $M \times N$, then the $MSE$ is defined as

$$MSE = \frac{\sum_i \sum_j e(i, j)^2}{M \times N} \quad (4.25)$$

where the local error $e(i, j) = R(i, j) - B(i, j)$. The lower the MSE score, the better the image object segmentation performance. In other words, the lower the value of $MSE$, the lower the error. In the case, the $MSE$ unit is pixel square.

The pixel error rate ($PERR$) is defined as

$$PERR = \frac{MSE}{255^2} \quad (4.26)$$

The lower the $PERR$ score, the better the image object segmentation performance. The unit of $PERR$ is pixel square.

The signal-to-noise ratio ($SNR$) is defined as

$$SNR(dB) = 10 \log_{10} \frac{\sum_i \sum_j R(i, j)}{\sum_i \sum_j (R(i, j) - B(i, j))^2} = 10 \log_{10} \frac{\sum_i \sum_j R(i, j)}{MSE} = 10 \log_{10} \frac{\sum_i \sum_j R(i, j)}{PERR \times 255^2} \quad (4.27)$$

In this case, the higher the $SNR$ metric evaluation score, the better the image object segmentation performance.

The peak signal-to-noise ratio ($PSNR$) is defined as

$$PSNR(dB) = 10 \log_{10} \frac{255^2 \times M \times N}{\sum_i \sum_j (R(i, j) - B(i, j))^2} = 10 \log_{10} \frac{255^2 \times M \times N}{MSE} = 10 \log_{10} \frac{M \times N}{PERR} \quad (4.28)$$

The higher the $PSNR$ score, the better the image object segmentation performance. The unit for both the $SNR$ and $PSNR$ evaluation metrics is decibel (dB).

<table>
<thead>
<tr>
<th>Method</th>
<th>Recall</th>
<th>Precision</th>
<th>F Measure</th>
<th>PSNR</th>
<th>SNR</th>
<th>MSE</th>
<th>PERR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sauvola</td>
<td>100.00</td>
<td>95.75</td>
<td>97.83</td>
<td>110.78</td>
<td>27.10</td>
<td>0.07</td>
<td>1.08E-06</td>
</tr>
<tr>
<td>Niblack</td>
<td>97.91</td>
<td>62.50</td>
<td>76.30</td>
<td>108.87</td>
<td>25.19</td>
<td>0.11</td>
<td>1.68E-06</td>
</tr>
<tr>
<td>Adaptive(L)</td>
<td>87.38</td>
<td>98.05</td>
<td>92.41</td>
<td>111.51</td>
<td>27.82</td>
<td>0.06</td>
<td>9.15E-07</td>
</tr>
<tr>
<td>Otus</td>
<td>76.86</td>
<td>97.06</td>
<td>85.78</td>
<td>112.21</td>
<td>28.53</td>
<td>0.05</td>
<td>7.79E-07</td>
</tr>
<tr>
<td>Kapur</td>
<td>100.00</td>
<td>89.15</td>
<td>94.26</td>
<td>110.30</td>
<td>26.62</td>
<td>0.08</td>
<td>1.21E-06</td>
</tr>
</tbody>
</table>

The higher the performance score the better the image segmentation technique, except for the $MSE$ metric in which a lower score is preferred. As seen in Table 4.1, the Otsu and Sauvola algorithms performed better than the other methods. The Otsu method perform better than the Sauvola method in terms of precision, $PSNR$, $SNR$, $MSE$ and $PERR$ evaluation metrics. On the other hand, the Sauvola method outperform the Otsu model in terms of recall and F-measure computations. In this work, we choose the Otsu image segmentation method, because it is robust and works very well for images with poor illumination compared to the other methods.
4.5 Image Object Skeletonization

Image object thinning, also referred to as skeletonization by Abu-Ain et al. [2013], is the process of reducing the width of the targeted or desired image object pixels to a single pixel (referred to as skeleton). The method used in extracting the skeleton of an object involves removing all the contour points in the object except those points that belong to the skeleton. These thinning operations are usually done in such a way that the successive removal of the outer layers of the object pixel does not in any way affect the pixel connectivity of the image skeleton. Sometimes, these steps are repeated several times until an image skeleton of 1 pixel thick with full pixel connectivity is achieved. A good thinning algorithm have the following properties (Harish and Paramjeet [2011]):

1. the obtained object skeleton must maintain or preserve the connectivity possessed by the original image object
2. the obtained object skeleton must have a unit width pixel
3. the algorithm should be insensitive to noise
4. the algorithm should have a short execution time

Some of the reasons for image object thinning as described by Abu-Ain et al. [2013] and Harish and Paramjeet [2011] are:

1. the thinning process reduces the amount of data to be processed and saves a lot of memory space required to store vital structural information about the image object
2. it helps to reduce the amount of time required to process the image patterns
3. thinning makes it easier to extract basic features of characters or shapes in most character or shape recognition applications.
4. shape analysis can be more easily performed on object skeletons than on the complete object pixels

Thinning algorithms can be categorised into two broad groups: Iterative and non-iterative thinning algorithms as shown in Figure 4.6.
4.5.1 Iterative thinning algorithm

This method of thinning the targeted image object involves pixel-by-pixel operations on the object until the desired skeleton is achieved. In other words, this method involves analysing all the pixels in the image in order to obtain a suitable skeleton image (Abu-Ain et al. [2013]). The iterative thinning method is further divided into sequential and parallel algorithms as shown in Figure 4.6. For sequential algorithms, the thinning operation is performed on one image pixel at a time and the outcome of the pixel currently being processed depends on previously analysed pixels. On the other hand, the parallel algorithms performs the thinning operations on all or a subset of all pixels in the image simultaneously. In most cases, the parallel thinning algorithms execute faster than their sequential counterparts especially when a parallel processor is available. However, the parallel algorithms in some cases fail to preserve the skeletal object connectivity after the thinning operation. For this reason, some iterative thinning algorithms combine both the parallel and sequential operations for each iteration or sub-iterations.

4.5.1.1 Zhang and Suen Algorithm

The Zhang and Suen [1984] thinning algorithm, is an iterative method that skeletonises a binary image pattern using two successive steps. In this method, a $3 \times 3$ pixels window or template is used in the thinning operation. The current pixel to be removed or retained is placed at the centre of the window and the other pixels in the $3 \times 3$ window are referred to as the eight connecting neighbours of the centred pixel. In this implementation, the neighbouring pixels are arranged in a clockwise order around the current pixel as shown in Figure 4.7, and the pixel considered for deletion are described with respect the neighboring pixel positions as shown in Figures 4.8 and 4.9.

<table>
<thead>
<tr>
<th>$P_1(x+1,y-1)$</th>
<th>$P_2(x+1,y)$</th>
<th>$P_3(x+1,y+1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_8(x,y-1)$</td>
<td>$P(x,y)$</td>
<td>$P_4(x,y+1)$</td>
</tr>
<tr>
<td>$P_7(x-1,y-1)$</td>
<td>$P_6(x-1,y)$</td>
<td>$P_5(x-1,y+1)$</td>
</tr>
</tbody>
</table>

Figure 4.7: A $3 \times 3$ window with nine pixels according to Zhang and Suen
For all $P_i = 0, 1$, where $i = 1, \ldots, 8$ are the eight neighbouring pixels to the current or centred pixel $P$. The $x-1, x, x+1$ and $y-1, y, y+1$ are used to represent the $x-$ and $y-$ coordinates of the pixel’s location respectively. Therefore to access the centred pixel value in Figure 4.7, we use location $(x, y)$. In this fast thinning algorithm implemented by Zhang and Suen [1984], they defined $N(P)$ as the number of non-zeros pixel neighbours around the centred pixel ($P$). The parameter $N(P)$ is given as

$$N(P) = \sum_{i=1}^{8} P_i$$  \hspace{1cm} (4.29)$$

They also defined other parameter $S(P)$ which is the number of changes in neighbourhood pixel values from 0 to 1 in the sequential order $P_1 \rightarrow P_2, P_2 \rightarrow P_3, P_3 \rightarrow P_4, \ldots, P_7 \rightarrow P_8, P_8 \rightarrow P_1$ around pixel $P$. In the first step of their algorithm, all pixels in the image are analysed based on equation 4.30 which consists of 4 conditions using a $3 \times 3$ pixels window.

The first and second conditions in step 1 states that the number of pixels surrounding the central pixel $P$ must have at least 2 white (in this case) pixels and at most 6 white (in this case) pixels neighbours. In addition, this implies that the central pixel $P$ cannot be deleted if it has only one neighbour because it will shorten the skeleton limb. The second condition in step 1, checks if the pixels surrounding the central pixel within the $3 \times 3$ neighbourhood has only one connected component (the number of 0 $\rightarrow$ 1 transition equals 1). The third and fourth conditions in step 1, implies that the centred pixel $P$ belongs to South ($P_6 = 0$), East ($P_4 = 0$) and North-West ($P_2 = 0, P_8 = 0$) boundaries as shown in Figures 4.8a,4.8b and 4.8c respectively. This mean that during the first pass, the algorithm will flag pixel $P_4$, $P_6$ and ($P_2$, $P_8$) for deletion.

2. Also applies a local $3 \times 3$ window to identify pixels border for deletion. After the entire image has been scanned, the identified border pixels are deleted.

$$ (P_2 \leq N(P) \leq P_6) \land (S(P) = 1) \land (P_2 \times P_4 \times P_6 = 0) \land (P_4 \times P_6 \times P_8 = 0) \hspace{1cm} (4.31)$$

The first and second conditions in step 2 are the same as in step 1. The third and fourth conditions in step 2, implies that the centred pixel belongs to North ($P_2 = 0$), West ($P_8 = 0$) and South-East ($P_4 = 0, P_6 = 0$) boundaries as shown in Figures 4.9a,4.9b and 4.9c respectively. This mean that during the second pass, the algorithm flag pixel $P_2$ or $P_8$ or ($P_4$ and $P_6$) for deletion.
In both steps, all the conditions are checked simultaneously (the logic $\land$ operator) and the pixel values that meet all conditions are noted. After all pixels in the image are tested, all noted or flagged border pixel values are set to zero and removed. This delay in the pixel deletion helps to prevent the removal of crucial pixel(s) that may affect the next pixel to be tested (to maintain skeleton connectivity). This process is repeated until there are no more boundary or surface pixels to remove.

### 4.5.1.2 Guo and Hall Algorithm

The Guo and Hall [1989] thinning algorithm, is an iterative parallel thinning algorithm with two sub-iterations. This algorithm uses a $3 \times 3$ template window as shown in Figure 4.7, to test and remove boundary pixels from an image object. In their implementation, they introduced two new variables $C(P)$ and $N(P)$. The variable $C(P)$ represent the number of unique 8-connected components with pixel value 1 within the $3 \times 3$ window surrounding the current pixel $P$. The variable $N(P)$ represent the minimum number of pairs of pixels around $P$ that contain ones (1’s).

$$N(P) = \min(N1(P), N2(P))$$  \hspace{1cm} (4.32)

Where

$$N1(P) = (P_1 \lor P_2) + (P_3 \lor P_4) + (P_5 \lor P_6) + (P_7 \lor P_8)$$  \hspace{1cm} (4.33)

and

$$N2(P) = (P_2 \lor P_3) + (P_4 \lor P_5) + (P_6 \lor P_7) + (P_8 \lor P_1)$$  \hspace{1cm} (4.34)

where $N1(P)$ and $N2(P)$ each divide the 8 neighbours around pixel $P$ into four pairs of connecting pixels. The variable $N(P)$ is essential for detecting endpoints and can also be use to improve the quality of the thinning algorithm. $N(P)$ is used to preserve endpoints while removing unnecessary pixels in the middle of curves. In this method, a boundary pixels also referred to as edge points are deleted if the following conditions are satisfied:
1. \( C(P) = 1 \)

2. \( 2 \leq N(P) \leq 3 \)

3. Choose to perform only one test:

   (a) for odd iterations test for: \((P_2 \lor P_3 \lor \tilde{P}_5) \times P_4 = 0\) and

   (b) for even iterations test for: \((P_6 \lor P_7 \lor \tilde{P}_1) \times P_8 = 0\)

Where the operator "\( \lor \)" denotes the logic OR operation and "\( \sim \)" denotes logical complement. When \( C(P) = 1 \), the current pixel \( P \) is said to be 8-simple which further implies that pixel \( P \) is a boundary pixel. The condition 3(a) determines when the pixel \( P \) can be deleted during odd iterations. In other words, it ensure that the pixel \( P \) is not deleted if its removal is going to affect skeletal object connectivity for odd iterations. Subsequently, condition 3(b) determines when pixel \( P \) should be deleted for even iterations. This condition ensures that, the skeletal end points are retrieved even if the points have only one or two foreground object pixel(s) (1’s) in the 8 neighbours. For both odd and even iterations, Figure 4.10 and 4.11 shows when it is possible to delete pixel \( P \) when the 8 neighbours take any of the forms. Figure 4.11a and 4.11b was obtained by rotating the conditions in Figure 4.10a and 4.10b by 180 degree respectively. The "\( \cdot \)" in both Figures 4.10 and 4.11 means don’t care pixel and can either be 0 or 1 value. The conditions in 3(a) and 3(b) can be used to identify the pixels at the north and east boundary, and the pixels at the south and west boundary of the objects respectively.

For each unique iteration, the object thinning operation is performed by deleting pixels in the north and east, then followed by south and east of the boundary or edge pixels. The thinning
operation is terminated when there are no pixels to delete. Guo and Hall in their paper, suggest that, an effective way of evaluating condition 1 \( (C(P)) \) is by introducing a new variable \( C_s(P) \) which is defined as:

\[
C_s(P) = \bar{P}2 \land (P3 \lor P4) + \bar{P}4 \land (P5 \lor P6) + \bar{P}6 \land (P7 \lor P8) + \bar{P}8 \land (P1 \lor P2) \quad (4.35)
\]

The operator “ \( \land \) ” represent logical AND operator. The value of \( C_s(P) \) is between 0 to 4 because it counts the number of times a side zero with a one in at least one of the two adjacent pixels in the clockwise direction around pixel \( P \) 8 neighbourhood. It was also stated that \( C_s(P) = C(P) \) for all neighbouring pixels of \( P \) except in instances where \( P \) has four sides with 1’s. In that case, \( N(P) = 4 \), which implies that pixel \( P \) is not deletable by the algorithm. Guo and Hall suggest the use of \( C_s(P) \) in computing condition 1 above \( (C(P)) \).

### 4.5.1.3 Stentiford Thinning Algorithm

This iterative parallel thinning algorithm by [Alberto and Tosunoglu (1986)]{}, extracts the image object skeleton by using four \( 3 \times 3 \) templates to scan the whole image pixels. Figure 4.12 shows the four templates used to determine if the pixel currently being processed should be deleted or retained. The following steps are used to retrieve the object skeleton:

1. For a given image, find a pixel location \((i, j)\) in which its surrounding pixel neighbours matches the template \( T1 \). Based on the identified template, delete all pixels from the left to right and from the top to bottom.

2. Using the template, mark the central pixel for deletion if it is not an endpoint pixel and has a connectivity number of 1. Here, an endpoint pixel is described as an object pixel location that is connected to one other object pixel. In other words, if an object (in this case black) pixel has only one black neighbour among its 8-neighbours, it is considered an endpoint pixel. The connectivity number was defined as the number of object pixels that are connected to a particular pixel under consideration.

\[
C_n = \sum_{k \in S} N_k - (N_k \times N_{k+1} \times N_{k+2}) \quad (4.36)
\]

Where \( N_k \) denote the colour value of the 8-neighbourhood pixel analyzed and \( S = \{1, 5, 3, 7\} \). \( N_0 \) is used to represent the template central pixel colour value and \( N_1 \) represent the colour value of the pixel to the right of the central pixel when the remaining 7-neighbours are labelled in counterclockwise order around the centered pixel.

3. Repeat step 1 and 2 for all pixel locations that matches the template \( T1 \)

4. Repeat step 1 to 3 using the other templates \( T2, T3 \) and \( T4 \). The template \( T2 \) is used to match pixels on the left side of the object, moving from the bottom to top and then from the left to right. The template \( T3 \) is used to match pixels along the bottom of the image.
and then move from right to left and then from bottom to top. The temple \( T4 \) is used to identify pixels on the right side of the object, then move from top to bottom and then from right to left.

5. Set pixels marked for deletion as white

![Templates](image)

Figure 4.12: The four \( 3 \times 3 \) templates used to locate pixels for deletion using the Stentiford thinning algorithm. The empty white boxes represent locations in which the colour value of the pixel does not matter or has to be checked

### 4.5.1.4 Hilditch Thinning Algorithm

The Hilditch’s thinning algorithm (Hilditch [1969]), combines both parallel and sequential iterative steps in the retrieval of the image object skeleton. The algorithm is parallel because at a single pass, all pixels are tested simultaneously and the decisions whether to keep or remove each of the tested pixels are made. The algorithm is sequential because the object skeleton extraction are repeated several times until there are no changes in the pixel pattern. There two versions of this algorithm; the first method use a \( 3 \times 3 \) window for scanning all pixels in the image and the second method use a \( 4 \times 4 \) window to achieve the same task. Here, we only considered the version of Hilditch thinning algorithm that applied the \( 3 \times 3 \) template as shown in Figure 4.7 for testing the image pixels.

Similarly, \( P_1, P_2, P_3, P_4, P_5, P_6, P_7 \) and \( P_8 \) are the eight neighbour pixels to the central pixel \( P \). The following steps describe how the Hilditch thinning algorithm is applied to a binary image in order to extract the object skeleton.

1. \( 2 \leq B(P) \leq 6 \).
2. \( A(P) = 1 \)

3. \( P_2.P_4.P_6 = 0 \) or \( A(P_2) \neq 1 \)

4. \( P_2.P_4.P_6 = 0 \) or \( A(P_4) \neq 1 \)

5. Repeat steps 1 to 4 until no more pixels can be deleted

The first step consists of two conditions: the number of non-zero 8-neighbours of \( P \) denoted as \( B(P) \) is greater than or equal to 2, and \( B(P) \) is less than or equal to 6. The first condition ensures that no endpoint pixel or no isolated one is deleted (any pixel with one white (1) neighbour is considered an endpoint pixel). The second condition in the first step ensures that boundary pixels are not deleted. The second step is used to test for pixels’ connectivity. \( A(P) \) is the number of 0 to 1 transitions in the sequence \( P_1, P_2, P_3, P_4, \ldots, P_7, P_8, P_1 \) neighbours around the central pixel \( P \). In Figure 4.13 where \( A(P) \geq 1 \), if the value of pixel \( P \) is changed to 0, this will affect the image object connectivity. The third step ensures that 2-pixel wide vertical lines are not completely deleted by the algorithm. The \((,\) between \( P_2, P_4 \) and \( P_8 \) implies that the values of these pixels are all 0 at the same time for a given instance. \( A(P_2) \) means the number of 0 to 1 transitions of the 8-neighbours around pixel \( P_2 \). A pixel is marked for deletion if this value is not equal to 1. The fourth step ensures that 2-width horizontal pixel are not deleted by the algorithm. \( A(P_4) \) means the number of 0 to 1 transitions of the 8-neighbours around pixel \( P_4 \).

\[
\begin{array}{ccc}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{array}
\]

\( A(P) = 2 \)

\[
\begin{array}{ccc}
0 & 0 & 1 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{array}
\]

\( A(P) = 2 \)

\[
\begin{array}{ccc}
1 & 0 & 0 \\
0 & 1 & 1 \\
0 & 1 & 0 \\
\end{array}
\]

\( A(P_2) = 3 \)

Figure 4.13: When not to mark pixel \( P \) for deletion using the Hilditch thinning algorithm because it will affect the pattern connectivity. The 1’s represent the foreground or object pixels and the 0’s represent the background pixels.
Figure 4.14: Image pattern skeletonization using iterative thinning algorithms
4.5.2 Non-iterative thinning algorithm

The non-iterative thinning algorithms are non-pixel based methods. However, the same properties of a good skeleton as defined with the iterative thinning algorithms also applies to the non-iterative methods. That is, the extracted skeleton must maintain the same topological properties of the object shape and can act as an approximate representation of the original image object. Examples of non-iterative methods are: Symmetric axis transform, medial axis transform and the grassfire transform. The most popular non-iterative thinning algorithm is the medial axis transform. The medial axis-based method directly extracts a central line also known as the median line from the desired image object only in one iteration without testing all the pixels in the image. The output of the medial axis method always gives a skeletal shape with lots of distortion. Other non-iterative skeletonisation methods retrieve the object skeleton by first detecting and extracting the object contour. Thereafter, they place points around the extracted contour based on the 8-connectivity chain code. Finally, with the help of the chain code and scanning over the whole image, the object skeleton is extracted. Like the case of the iterative thinning method, there are several implementations of the non-iterative thinning algorithm; with each newly developed algorithm trying to improve on a previous existing method. Here, we show the result of the median axis-based algorithm on the same image used in testing the performance of the different iterative thinning algorithm. It is important to note that, the reconstruction of the original image object or pattern can be easily achieved from the extracted skeleton object using the non-iterative thinning algorithms as compared to the iterative thinning algorithms. The reason is that, most non-iterative methods store information about the distance between the extracted skeleton points and their corresponding contour or boundary points. Therefore, to reconstruct the original image object requires drawing the object contour using the two parallel points, the corresponding skeleton point and the distance between each point from the skeleton point. However, this method is computationally expensive for a large image dataset.

4.5.3 Thinning algorithm performance measurement

Thinning or skeletonisation algorithms are different from each other based on their individual performance and implementation method. Two major criteria for any good thinning algorithm is that, the algorithm should converged or be terminated without altering the connectivity of the skeletal image object and the width of the skeletal image should be a single pixel thick uniformly. The following performance measurements were used to measure how good the skeletonisation algorithms are:

1. The Size Ratio

The size ratio is the ratio of the number of foreground pixels in the skeleton to the number of foreground pixels in the input binary image (after segmentation). This measurement
Figure 4.15: Image pattern skeletonization using non-iterative medial axis transform

is considered reliable only when the quality of the skeleton is acceptable. The size ratio is defined as

\[ S_r = \frac{|S_p|}{|B_p|} \]  

(4.37)

Where \(|x|\) is the operation that counts the number of foreground pixels in the pattern \(x\), \(S_p\) denote the resultant skeleton and \(B_p\) denote the input binary image pixels. The lower the size ratio, the greater the reduction in the foreground data.

2. Measure of Convergence of Unit Width

This measurement indicates how thin or how close to 1-pixel width the obtained skeleton is. It is defined as

\[ m_t = 1 - \left( \frac{\text{Area}[u_1 \leq k \leq 4S_pQ^k]}{\text{Area}[S_p]} \right) \]  

(4.38)

Where \(\text{Area}[x]\) is the operation that counts the number of foreground pixels, \(S_p\) is the resultant skeleton and \(Q\) is the set of structuring elements used for hit or miss transform. The value of \(m_t\) is a non-negative integer in the range \([0,1]\). The greater this value, the better is the convergence.

3. Number of Iterations (n)

This measure is the total number of iterations undergone by the procedure until the result is obtained. Although this measure does not give an absolute measure of performance; a low value indicate that the method is robust.
The result of each algorithm performance is shown in Table 4.2. The results shows that, the algorithm proposed by Guo and Hall [1989] produce a better skeleton image with zero convergence unit, lowest size ratio and highest sensitivity. However, the Guo and Hall [1989] algorithm involves a large number of iterations which affect its robustness. In this research, we choose the Zhang and Suen [1984] algorithm which also has a high sensitivity, low size ratio and low number of iterations than the Guo and Hall [1989] algorithm. The chosen algorithm is also not computationally expensive as the non-iterative thinning algorithm (medial axis transform).

<table>
<thead>
<tr>
<th>Input Data</th>
<th>Convergence unit</th>
<th>Execution time (seconds)</th>
<th>Iterations</th>
<th>Size Ratio</th>
<th>Sensitivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhang and Suen</td>
<td>0.0005</td>
<td>0.0352</td>
<td>28</td>
<td>0.0149</td>
<td>0.9704</td>
</tr>
<tr>
<td>Steniford</td>
<td>0.0035</td>
<td>0.0091</td>
<td>107</td>
<td>0.026</td>
<td>0.9481</td>
</tr>
<tr>
<td>Hilditch</td>
<td>0.0005</td>
<td>0.0142</td>
<td>69</td>
<td>0.0195</td>
<td>0.9613</td>
</tr>
<tr>
<td>Guo and Hall</td>
<td>0</td>
<td>0.0246</td>
<td>40</td>
<td>0.0143</td>
<td>0.9715</td>
</tr>
</tbody>
</table>

4.5.3.1 Skeletonisation challenges

During the image object skeleton retrieval process, we encountered some challenges while getting rid of the unwanted objects in the image frames. The reason is that, it is difficult to completely get rid of the unwanted object when it is in contact with the body of the worm. Therefore, one solution to this problem will be to remove the affected frames from the video sequence and either interpolate the values of the dropped frames or replaced them with the next valid successive frame.
Chapter 4 Video Frames Preprocessing

Figure 4.16: Complete and partial removal of unwanted objects during image object skeletonisation

Figure 4.16a shows the transformation of the original "noisy" worm video frame to a frame containing the shape of the worm skeleton, as depicted in Figure 4.16b. Figure 4.16c shows when the worm is in contact with an unwanted object that cannot be completely removed, as seen in Figure 4.16d.

4.5.4 Spline generation and features collection

The spline is created using a cubic interpolation of midpoints of the worm body. The spline represents the worm skeleton and it is used to extract the coordinates of the points along the worm body. The spline is divided into a given number of segments by placing markers at equal intervals. In the previous work, Stephens et al. [2008] and Brown et al. [2013] divide the worm skeleton into 100 and 48 segments respectively. In this report, we consider 50 segments by placing 51 markers on the skeleton. The reason is that, the total number of points along the worm skeleton varies and therefore, it is important to specify the number of segments to used across all frames in the video sequence. By measuring the tangent angles between two consecutive points along the spline, we collect and store data for the quantification of the worm body posture. For example, with 100 segments we can measure and store 100 tangent angles along the worm body. This process also facilitates the identification of the head and tail of the worm, because the coordinates of the two ends of the skeleton are the potential head and tail positions. To move along the spline, we can start from the head and move towards the tail of the worm or vice versa. At this point, it is important to state that the number of points along
Chapter 4 Video Frames Preprocessing

the worm skeleton differs for each frame and therefore, there is need to specify the number of coordinates to consider and that should be uniform across all frames in the video sequence for proper analysis. First, we calculate the number of points along the skeleton for all frames. Next, we compute the minimum and maximum points along the skeleton body and with these values, we can then consider the number of segments to use. By default, the arrangement of skeleton coordinates is in increasing order of the $x$-coordinate. However, this does not give a true representation of the points along the skeleton body in most cases. Hence, we followed a well-defined steps that provides a means of moving from one end of the worm body to the other end:

1. Identify and retrieve the coordinates of the head and tail of the skeleton body.
2. Starting from one end, compute the nearest point, move to it, and store the former coordinate.
3. Repeat step 2 until you get to the other end of the skeleton.

Figures 4.17a and 4.17b illustrate the difference between using the default skeleton coordinates and the ordered coordinates. After computing the tangent angles between two successive segment, we stored the angles for further processing.

4.6 Summary

This chapter discussed the importance of image preprocessing steps in the extraction of meaningful biological features that describes the shape of the worms during movement. Two aspect of the image preprocessing steps considered here are: image segmentation and image skeletonisation. Although there are deep image segmentation and skeletonisation algorithms in existence, but we choose to implement the classical methods based on our data size and other requirements by the deep learning algorithms as mentioned in section 3.3.2. After the implementation of several image segmentation and image skeletonisation algorithms, we found
the Nobuyuki [1979] (Otsu) image segmentation and the Zhang and Suen [1984] image skeletonisation algorithms as the right choices for this experiment. The Otsu’s method was used by Stephens et al. [2008] for the worms image segmentation task, however there is no report where the Zhang and Suen [1984] skeletonisation method was applied to retrieve worm skeleton body shapes. Our choice of the image segmentation and image skeletonisation algorithms was based on the results obtained using different performance evaluation metrics. For the image segmentation task, another alternative to the Otsu’s method is the Sauvola and Pietikainen [2000] algorithm. Similarly, the Guo and Hall [1989] algorithm is an alternative to the Zhang and Suen [1984] algorithm used here for the image object thinning task. We discovered that, no report has clearly state or shown the evaluation method(s) used to determine the right image segmentation or image skeletonisation algorithms applied to C. elegans images as done here.
Chapter 5

C. Elegans Phenotypic Feature Extraction and Classification

5.1 Introduction

*C. elegans* genotype prediction task based on visual inspection by expert is time consuming and prone to errors. To tackle these problems, computer vision and machine learning techniques can be used to automate the feature extraction and classification process. These techniques help to reveal subtle features that are not visible to the human eyes and facilitate the classification task. In this study, we extracted time series features from the worm skeleton body in each successive still image frame in a video record. The skeleton images are obtained after the application of thinning algorithm to their corresponding grayscale images retrieved from the video record ([Zhang and Suen 1984](#)). There are 6 types of worms with different genotypes in the dataset and for each genotype there are 200 video records. For each worm video frame of dimension $640 \times 480$ pixels, we extracted 400 image frames and retrieved the equivalent binary worm skeleton images. Subsequently, we further processed the worm’s skeleton images by extracting meaningful information about their posture. This was done by first marking predefined equally spaced number of points along the worm’s body skeleton posture for each image frame. Thereafter, we measured 50 tangent skeleton angles for each image frame based on the marked points and stored the angle data as feature vector. We repeated the same process for all extracted image frames and labelled each feature vector according to the worm type. We referred to these tangent skeleton angles feature vectors as the high-dimensional skeleton angles data set. Furthermore, we reduced the high-dimensional angles data to a more compact low dimensional feature space using both supervised and unsupervised dimensionality reduction techniques such as linear discriminant analysis (LDA), principal component analysis (PCA), kernel PCA, non-negative matrix factorisation (NMF) and kernel Fisher linear discriminant analysis (KLDA). In this work, we used both the higher- and lower-dimensional shape-based features representation to characterise the movement behaviour of different worm
types. Each feature representation method serves as input to different standard machine learning algorithms such as K-Nearest Neighbour (kNN), Support Vector Machine (SVM), Random Forest (RF) classifiers. The objective here is to discover patterns in the locomotory behaviour of the worms that are predictive of their genotypes taken as the classification target. This chapter is organised as follows: first, section 5.2 describe the steps taken to transform a given binary skeletonised image object to an array of tangent angles. Next, section 5.3 explains how significant low dimension shape-based features were retrieved from the high-dimensional tangent angles data using unsupervised learning techniques. Subsequently, in section 5.4 we discuss the classification results as related to the different input representations implemented here. Finally, section 5.7 contain the summary of the work done.

5.2 High-dimensional Skeleton Angles Extraction

To obtain the high-dimensional tangent skeleton angles data, a series of image preprocessing steps were performed on the original video frames. These include image smoothing for the removal of noise or the isolated one points (image foreground pixel with no connectivity to other pixels), image binarisation (or image thresholding) for the conversion of the grayscale image to a binary image in which white pixels represent the image foreground and black pixels represent the image background, and image skeletonisation (or image thinning) for the extraction of the worm skeleton shape from the binary images. The next phase after the retrieval of the worms skeleton shapes, is the extraction of meaningful information from the shape structure that can act as an approximate representation of the original worm shapes and useful for the characterisation of the movement patterns of the worms across multiple video frames. If these movement patterns are peculiar to a specific C. elegans strain, this can be use to distinguish one type of worm from another. The reason is that, the worm’s body shape changes as it moves and may contain useful information about the decision-making process in the worm’s neural circuit which in turn can be use to interpret the worm’s behaviour. In this work, we measured \( m = 50 \) tangent angles along the worm’s body skeleton shape. To achieve this, we applied a \( 3 \times 3 \) kernel filter to the skeletonise the worm images in order to retrieve the skeleton head, tail and body coordinates. We then fit a spline through the identified skeleton body coordinates and divide the coordinates into equally spaced number of segments. We refer to points that make up a segment as markers. In other words, a segment consist of two markers. A tangent angle is the angle between two successive markers along the worm’s skeletal shape. The total number of angles measured for a given frame is one less than the number of markers in the skeletal shape. We used equation 5.1 to compute the tangent angle between two successive markers.

\[
\theta_i = \arctan \left( \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \right) \tag{5.1}
\]
where \((x_i, y_i)\) and \((x_{i+1}, y_{i+1})\) are two consecutive coordinates or points along the worm skeleton body, \(i = 1, \ldots, m\) and \(m + 1 = 51\) is the number of markers along the worm skeletal shape. The angles for each frame are normalised by subtracting the mean angle value from each angle. This is to ensure that the skeleton angles data are rotation invariant.

\[
\theta_i^{(t)} = \theta_i - \langle \theta \rangle
\]  

(5.2)

where \(\theta_i^{(t)}\) represent the normalized angle data at frame \((t)\) and the mean angle \(\langle \theta \rangle\) is defined by

\[
\langle \theta \rangle = \frac{1}{m - 1} \sum_{i=1}^{m-1} \theta_i.
\]  

(5.3)

An image frame can be represented as a sequence of skeleton angles at time \((t)\), \(frame(t) = (\theta_1^{(t)}, \theta_2^{(t)}, \ldots, \theta_m^{(t)})\) and a video file can be represented as \(v(n) = \{frame_{(n)}(t)| t = 1, \ldots, T_n\} = \{(\theta_1^{(t)}, \theta_2^{(t)}, \ldots, \theta_m^{(t)})| t = 1, \ldots, T_n\}\). The \(m\) denotes the number of tangent angles along the worm skeleton body for each video frame, \(T_n\) denotes the total number of frames extracted from a video file and \(n\) represents the \(n^{th}\) video file. This implies that a video frame of size \(640 \times 480\) has been transformed into a sequence or array of size \(1 \times m\) and each skeleton angle in a frame represents a feature variable. The skeleton angles extracted for each movie are arranged and use as feature vector. The high-dimensional skeleton angles data consists of feature vectors extracted from the single worm movie files. Therefore, the size of the skeleton angles dataset is \(N \times D\), where \(D = m \times T_n\) and \(N\) is the total number of videos in the input space. Furthermore, all video frames of the same worm type are assigned the same label. In this study, we used \(X \in \mathbb{R}^{N \times D}\) to serve as the high-dimensional skeleton angles data representation. The high dimension angles data can serves as input to a machine learning classifier for worm genotype classification task or can be further processed to retrieve a more meaningful low-dimensional representation. If each image frame is treated as a data point, then the number of observation will be \(N \times T_n\) and the dataset can be represented as \(X \in \mathbb{R}^{(N \times T_n) \times m}\). Here, \(N \times T_n\) represents the total number of extracted image frames in the dataset. The tangent skeleton angles retrieved from a single still image cannot be used only to characterise the worm’s behaviour, but are useful in terms of the worm shape reconstruction using a low-dimensional shape-based representation. Furthermore, the single frame skeleton angles are useful for interpolating or finding an approximation of a missing frame angles. In this work, the high-dimensional skeleton angles data serves as input to the unsupervised and supervised dimensionality reduction algorithms to obtain a low-dimensional feature subspace. In this chapter, we only considered the unsupervised dimensionality reduction methods. The skeleton angles data in the form of a time series are use to categorise the worms into different classes, when serve as input to the different machine learning classifiers.
5.3 Dimensionality Reduction Techniques

To further reduce the dimension of the skeleton angles data, we applied unsupervised learning algorithms such as principal component analysis (PCA), non-negative matrix factorisation (NMF) and kernel principal component analysis (KPCA). These dimensionality reduction techniques were used to reduce the dimension of the worms body skeleton angles data to a more compact representation. These new representations contain meaningful information about the shapes of each worm in a given video record and can be used to characterise a worm movement pattern. Due to the problem of missing values in the skeleton angle dataset, these dimension reduction techniques can be used directly to impute the missing angles data or applied to the complete skeleton angles dataset after the application of other imputation methods. Here, our first approach was to perform the dimension reduction task on the high-dimensional tangent angles data, after all missing frame angles have been imputed using different imputation methods. The second approach was to directly compute for the missing values and at the same time perform the dimension reduction task on the high-dimensional tangent skeleton angle dataset. Here, we only describe the missing data imputation method using PCA.

5.3.1 Dealing with dropped unsegmented or unskeletonised frames

During the process of segmentation or skeletonisation, some video frames were dropped because they could not be segmented or skeletonised. The reason for the dropped video frames are as a result of:

1. The surrounding illumination when creating the videos
2. Inadequate contrast
3. The difference in the gray-level between the object (foreground) and the image background
4. Frames which capture fast movement of the camera when trying to place the worm under the camera view
5. Frames in which the worm body cross itself (for a single worm)

These problems are inevitable. For example, we cannot control the shapes exhibited by worm while moving in order to avoid self-occlusion (body parts crossing itself). Similarly, it is hard to control the ambient illumination for all videos in order to achieve the same result. One way to tackle these problems is to apply image processing techniques that are less sensitive, but robust to these issues as discussed in the previous chapter. However, it is difficult to resolve the problems completely, hence we still have missing values as a result of dropped frames in the dataset. In this work, we approach the problem in two ways: The first method involve taking into account the locations of the failed skeleton frames and then use imputation techniques to
compute the values for dropped frames. In the second approach, we replaced all video frames that could not be skeletonise with the next successfully skeletonised frame and then extract the tangent angles from the video frames. The limitation with the second method is that, different video length is been used. We compared both missing frame interpolation methods to see which is more suitable for the *C. elegans* locomotory behaviour characterisation.

### 5.3.2 To confirm the number of dropped frames due to failed segmentation and skeletonisation

The worm image segmentation and skeletonisation algorithms are implemented such that any image frame that could not be segmented or skeletonised are dropped and the frame position in the video record as well as the video name (which consists of the worm type and video number) are saved. At the first stage, we selected 3000 frames from each video record and videos in which 1000 frames were unable to be segmented or skeletonised were removed. In the next stage, we remove videos in which more than 100 successive image frames out of 400 frames failed the segmentation or skeletonisation process. In the final stage, videos in which at least 400 out of 500 image frames were unable to be segmented or skeletonised were also dropped. The remaining videos were used for the *C. elegans* classification experiment.

![Figure 5.1: Videos and the count of failed segmented and skeletonised frames](image)

As seen in Figure 5.1, it shows a plot of the worm videos comprising of the worm video name and the video number in the horizontal axis, while the vertical axis represents the count or number of dropped image frames in that video. As demonstrated, the video OW956_10 would
be removed because more than 2000 image frames were unable to be segmented or skeletonised by the traditional segmentation or skeletonisation algorithms. When we look at the image frames in that particular video, we discovered that large number of images were blurred and contained a lot of unwanted image objects is in contact with the worm body shape. It is difficult to segment or skeletonise the image object when in contact with an unwanted object or truncated by the frame boundary. We also investigated the number of successive dropped image frames for each video as illustrated in Figure 5.2.

![Figure 5.2: A worm video and the count of failed segmented and skeletonised frames](image)

As shown in Figure 5.2, more than 150 image frames out of 500 successive images were dropped and therefore not used in the experiment.

### 5.3.3 Data Imputation

Due to the video frames that could not be segmented or skeletonised, there are missing values in the skeleton angles data. Data imputation is the task of computing an approximate substitute or proxy for a missing value in a given dataset. The performance of an imputation method depends on how close the computed value is to the original value. In this research, we applied linear and spline interpolation and dimensionality reduction methods to impute the missing values in the skeleton angles dataset. First, we described how the linear interpolation methods was used to impute the missing tangent skeleton angles due to the dropped video frames that could not be segmented or skeletonised. Second, we performed the same missing values imputation task...
using the spline interpolation method. Third, we estimated the missing skeleton angles data using the unsupervised PCA dimensionality reduction method and at the same time extract a low dimensional representation of the skeleton angles data. Finally, we evaluate the different imputation methods using two loss functions such as root mean squared error (RMSE) and mean absolute error (MAE).

5.3.3.1 Linear imputation

The linear imputation method uses information from data points before and after a missing data point to estimate a value for the missing data. This imputation method uses a linear function to compute the value of the missing data. In other words, this process involves using known or valid observations in a data to compute unknown or missing values before and after them. For a time series with two data points \( x_a \) and \( x_b \) for \( a < b \), the estimate \( x_t \) for the unobserved time \( t \), with \( a < t < b \) is defined as:

\[
x_t = x_a + \frac{x_b - x_a}{b - a} (t - a)
\]  

(5.4)

where \( x_t \) represents the unknown data point at time \( t \), \( x_a \) and \( x_b \) represents the known values before and after the missing data at time \( a \) and \( b \) respectively. The linear interpolation method is mainly used to estimate the missing values per variable.

5.3.3.2 Spline imputation

Let \( q(x) \) represents the quadratic interpolant of the function \( f(x) \) (Vanderbobert [2017]). This implies that, we define \( q(x) \) as

\[
q(x) = ax^2 + bx + c
\]  

(5.5)

where \( a, b, c \in \mathbb{R} \). The values of the coefficients (constants) can be obtained by bracketing the critical point of \( f \), whose endpoints are \( x_1 \) and \( x_2 \). Therefore, we have

\[
f(x_1) = ax_1^2 + bx_1 + c
\]  

(5.6)

\[
f(x_2) = ax_2^2 + bx_2 + c
\]  

(5.7)

\[
f'(x_1) = 2ax_1 + b
\]  

(5.8)

Note that \( f(x_1) = q(x_1), f(x_2) = q(x_2) \), and \( f'(x_1) = q'(x_1) \). Here we have three equations (5.6, 5.7, and 5.8) with three unknowns \( a, b, \) and \( c \). If \( f_i = f(x_i) \) and \( f'_i = f'(x_i) \), then we can rewrite equations (5.6, 5.7, and 5.8) as equations (5.9, 5.10, and 5.11) respectively.

\[
f_1 = ax_1^2 + bx_1 + c
\]  

(5.9)

\[
f_2 = ax_2^2 + bx_2 + c
\]  

(5.10)
\[ f_1' = 2ax_1 + b \] (5.11)

By solving equations (5.9, 5.10, and 5.11) for \( a \) and \( b \) we have:

\[ a = \frac{(f_1 - f_2 - f_1'(x_1 - x_2))}{(x_1 - x_2)^2} \] (5.12)

\[ b = f_1' + 2\left(\frac{f_1 - f_2 - f_1'(x_1 - x_2))}{(x_1 - x_2)^2}\right)x_1 \] (5.13)

The minimiser of \( q \) denoted as \( x_{\text{min}} \) is obtained from \(-b/2a\) by setting \( q'(x) = 0 \) in equation 5.8. Hence we have:

\[ x_{\text{min}} = \frac{-b}{2a} = x_1 - \frac{1}{2} \frac{(x_1 - x_2)f_1'}{(f_1 - f_2)(x_1 - x_2)} \] (5.14)

Hence, we can rewrite equation 5.14 as an iteration formula which can be used to impute (estimate) missing data in a time series. That is, if the minimiser \( x_{\text{min}} = x_k \) which lie between points \( x_{k-1} \) and \( x_{k+1} \), then equation 5.14 becomes

\[ x_k = x_{k-1} - \frac{1}{2} \frac{(x_{k-1} - x_{k+1})f_{k-1}'}{f_{k-1} - f_{k+1}} \] (5.15)

This iteration approach can also be used to interpolate missing data in a truncated time series say \( x_{k+1} \) which is outside the two endpoints \( x_{k-1} \) and \( x_k \) defined as:

\[ x_{k+1} = x_{k-1} - \frac{1}{2} \frac{(x_{k-1} - x_k)f_{k-1}'}{f_{k-1} - f_k} \] (5.16)

There are different variant of quadratic splines which applies different techniques in estimating the missing data such as the Lagrange and Newton interpolation methods. We performed the linear and spline missing frame angles interpolation using the pandas package function called interpolate. The missing values interpolation was done per single worm video.

The following steps were taken to evaluate the missing values interpolation methods:

1. Randomly select a given number of videos without missing skeleton angles.
2. Define the percentage of missing values (frames) and the number of consecutive missing values in each video to use.
3. Randomly select the starting locations for the frames to be removed and store them for future use (ensure that the last frame position is not included).
4. Remove the values of the selected locations and store them for future use (these values are the original valid data).
5. Replace the remove data with not a number (NaN).
Table 5.1: Missing data imputation using linear and quadratic spline interpolation methods

<table>
<thead>
<tr>
<th>Consecutive missing frames</th>
<th>Video number</th>
<th>Impute method</th>
<th>Percentage missing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RMSE</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>linear</td>
<td>0.0705</td>
</tr>
<tr>
<td></td>
<td></td>
<td>spline</td>
<td>0.0523</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>linear</td>
<td>0.0576</td>
</tr>
<tr>
<td></td>
<td></td>
<td>spline</td>
<td>0.0464</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>linear</td>
<td>0.0704</td>
</tr>
<tr>
<td></td>
<td></td>
<td>spline</td>
<td>0.0546</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>linear</td>
<td>0.0549</td>
</tr>
<tr>
<td></td>
<td></td>
<td>spline</td>
<td>0.0427</td>
</tr>
</tbody>
</table>

6. Perform the linear or quadratic spline interpolation method

7. Measure the difference between the estimated values and the original valid data using the RMSE and MAE performance metrics.

Here, the percentage of missing frames considered are 5%, 10% and 20%, and the number of consecutive missing frames used are 5 and 10. For instance, if a video has 400 frames, 20 images would be considered missing if 5% of a video data is assumed to be missing. In that case, 5 consecutive frames would be removed in 4 different locations without overlapping.

As shown in Table 5.1, the performance of each interpolation method is evaluate using root mean squared error (RMSE) and mean absolute error (MAE) depending on the percentage of missing values and the number of successive missing values. The evaluation result shows that, the quadratic spline interpolation method produced the least RMSE and MAE values. In both worm videos, the quadratic spline data imputation method outperformed the linear imputation method. As also seen in the Table 5.1, even as the percentage of missing data increases the spline imputation method performed better when compared with linear method. However, as the percentage of missing data increases for both video 1 and 2, the RMSE and MAE values decreases for both the linear and spline interpolation methods. This suggest that, the approximated values are very close to each other but far from the already existing values before the imputation. That is, the estimated values are not within the same range as the valid data used for the imputation. Overall we preferred the quadratic spline interpolation method over the linear based on the performance evaluation results. For this reason, we choose to use the spline interpolation method over the linear method.

### 5.3.4 Principal Component Analysis

In this section, we first describe the extraction of meaningful features from the high-dimensional skeleton angles data \( X \in \mathbb{R}^{(N \times T_n) \times m} \) using PCA, with the assumption that no video frames
were dropped during the segmentation or skeletonisation process. We considered each still worm image skeleton angles as a data point. With the PCA dimension reduction method, we transformed every skeleton angles in a video frame into low dimensional time series coefficients. That is, a given video frame \( frame(t) = (\theta_1^{(t)}, \theta_2^{(t)}, \ldots, \theta_m^{(t)}) \) is transformed into \( x^{(t)} = (x_1^{(t)}, \ldots, x_d^{(t)}) \). The \( d \) denotes the number of features or principal component(s) retained and \( x^{(t)} \in \mathbb{R}^d \). Similarly, the \( n \)th video can be defined as \( v(n) = (x_1^{(n)}, x_2^{(n)}, \ldots, x_{T_n}^{(n)}) \), where \( x_t^{(n)} \in \mathbb{R}^d \) for \( t = 1, \ldots, T_n \) and \( T_n \) denotes the total number of image frames extracted from a given worm video record. This means that, we reduced the dimension of \( X \) from \( m \) to \( d \), where \( d < m \).

![Cumulative Explained Variance as a Function of the Number of Components](image)

**Figure 5.3:** Variance captured by retaining fifteen principal components

After performing the dimension reduction task on the skeleton angles data using the PCA algorithm described in section 2.3.3.1 or appendix A, we achieved 94% of the total shape variance by using only four principal components \( (d = 4) \) as shown in Figure 5.3. This result is same as the result obtained by Stephens et al. [2010]. This implies that, for a suitable choice of basis, with just 4 coefficients we can reconstruct any worm shape to a reasonable degree of accuracy.

Furthermore, we described the equations for reconstructing the skeleton shapes of the worms in each image frame based on the PCA time series coefficients. Let the \( n \)th video file be defined as:

\[
 v(n) = (x_1^{(n)}, \ldots, x_{T_n}^{(n)}),
\]

where
\[ \mathbf{x}^{(t)}_{(n)} = \begin{pmatrix} \tilde{\theta}^{(t)}_{(n)1} \\ \vdots \\ \tilde{\theta}^{(t)}_{(n)m} \end{pmatrix} \]

and

\[ [\mathbf{x}^{(t)}_{(n)}]_i = \tilde{\theta}^{(t)}_{(n)i} \]

\( \mathbf{x}^{(t)}_{(n)} \) denotes the skeleton angles vector for each image frame \( t \) in video file \( n \), \( m \) denotes the number of tangent angles measured along the worm’s skeleton body (Here \( m = 50 \)), and \( T_n \) denotes the total number of image frames extracted from each video. The following steps are taken to reconstruct the worm skeleton shape for each video frame

1. Normalise or center the skeleton angles data :

\[
\tilde{\theta}^{(t)}_{(n)i} = \theta^{(t)}_{(n)i} - \frac{1}{m} \sum_{i=1}^{m} \theta^{(t)}_{(n)i} \quad (5.18)
\]

where \( \tilde{\theta}^{(t)}_{(n)i} \) represents the centred angles for each image frame and \( \theta^{(t)}_{(n)i} \) represents the extracted tangent angles from the worm’s skeleton body in each image frame.

2. Compute the mean skeleton angles from the data :

\[
\langle \tilde{\theta} \rangle_i = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{T_n} \sum_{t=1}^{T_n} \tilde{\theta}^{(t)}_{(n)i} \quad (5.19)
\]

where \( \langle \tilde{\theta} \rangle_i \) denotes the mean vector (global) for the data.

3. Compute the covariance matrix :

\[
[C_{\theta\theta}]_{ij} = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{T_n} \sum_{t=1}^{T_n} \delta\theta^{(t)}_{(n)i} \delta\theta^{(t)}_{(n)j} \quad (5.20)
\]

where

\[
\delta\theta^{(t)}_{(n)i} \equiv \tilde{\theta}^{(t)}_{(n)i} - \langle \tilde{\theta} \rangle_i
\]

where \( \delta\theta^{(t)}_{(n)i} \) represents the centred angle values (vector) for the images and \( \delta\theta^{(t)}_{(n)j} \) is the transpose of the centred angles values.

4. Reconstruct for each frame the skeleton angles

\[
\theta^{(t)}_{(n)i} = \tilde{\theta}^{(t)}_{(n)i} u^T_\mu + \langle \tilde{\theta} \rangle_i \quad (5.21)
\]
where $\theta^{(t)}_{(n)j}$ denotes the reconstructed angles, $\tilde{\theta}^{(t)}_{(n)j}$ denotes the normalised or centred skeleton angles, $u_\mu$ are the eigenvectors of $C_{\theta\theta}$ and represents the selected eigenvectors with the highest eigenvalues of the $m \times m$ angle covariance matrix and $\langle \tilde{\theta} \rangle$ is the mean angle.

5. For each frame skeleton angle data $\theta_i$, reconstruct the $x-$ and $y-$ coordinates along the worm skeleton body using equations 5.22 and 5.23 respectively

$$x_i = x_{int} + \sin(\theta^{(t)}_{(n)j})$$  \hspace{1cm} (5.22)

$$y_i = y_{int} + \cos(\theta^{(t)}_{(n)j})$$  \hspace{1cm} (5.23)

Where $x_{int}$ and $y_{int}$ denotes the initial starting points. In our case, $x_{int} = 0$ and $y_{int} = 0$. For each angle in a frame, update the values of $x_{int}$ and $y_{int}$ to compute the next coordinates. $x_i$ and $y_i$ are estimated coordinates along the worm body shape. The $\theta^{(t)}_{(n)j}$ represents the reconstructed angles for each image frame.

6. Plot the $x-$ and $y-$ coordinates for a given frame to reconstruct the skeleton shape of the worm.
Figure 5.4: Worm shape reconstruction using PCA. The figure at the top represents the original worm image retrieved from a video sequence, the next figure depicts the binary image of the worm, the third figure shows the skeletonised image and the bottom figure depicts the reconstructed worm shape using PCA components.

Figure 5.4 shows the original video image, binary equivalent image, the skeletonised image and the PCA reconstructed skeleton shape respectively (top to bottom). The blue circles in the reconstructed image represent the number of angles measured along the worm skeleton body. Furthermore, we compared the worm shape reconstruction using the high-dimensional skeleton angles and the PCA reconstructed angles. As demonstrated in Figure 5.5, the PCA reconstructed shape (Figure 5.5d) of the worm looks smoother than the reconstructed skeleton shape using the high-dimensional raw angles (Figure 5.5c). Figures 5.5a and 5.5b represent the original worm image and the skeletonised image of the worm respectively.
To distinguish between the nematode worms with different genotypes using the PCA features, the input to the machine learning classifiers are the projection of the normalised skeleton angles data onto the $d$ selected eigenvectors with the largest eigenvalues.

For the low dimensional time series features representation with dropped video frames, the unsupervised PCA learning algorithm was used to compute the missing values and at the same time reduce the dimension of the feature set. The following steps were taken to compute the PCA time series coefficients.

1. Compute the tangent angles for all frames in the same video and note the locations of the dropped frames.

2. For all missing values in the same video record, initialise the missing values with their corresponding feature mean angles.

3. Compute the low dimensional time series features using PCA per video record.

4. Repeat the same process for all video records.
5. Stack the time series features for all video records.

6. The derived PCA time series coefficients serves as input representation to standard machine learning classifier.

This process prevents the use of skeleton angle values from a different video record in the imputation of missing values and the computation of the lower dimensional features in another video record. In other words, this method uses the low dimensional principal components derived from the observed $50 \times 50$ high dimensional angles data to impute the missing values for both the high and low-dimensional missing features for each worm video file.

To demonstrate the effectiveness of imputing missing values using PCA, the following steps was taken:

1. Collect a subset of the high-dimensional tangent skeleton angle data in which all frames are properly skeletonised

2. Randomly remove a sequence of $n$ number of frames from different locations in each video data

3. Using PCA, perform the dimension reduction task to obtain the PCA time series coefficients

4. Use the computed PCA time series coefficients data to reconstruct the skeleton shapes of the randomly removed tangent skeleton body angles data

5. Compare the artificially reconstructed (imputed) skeleton shapes of the worm with that of the worm skeleton shape obtained when the complete high-dimensional subset angles data was used

The visual inspected result in Figure 5.6 shows that this method is reliable especially when the number of sequentially dropped frames $n \leq 5$. As the value of $n$ increases the shape of the reconstructed worm skeleton body looks less similar to the original worm body shape in the video frame. The Figures 5.6c, 5.6e, 5.6g and 5.6i are frames in the data which were initially segmented and skeletonised, but were removed from the data in order to test how the PCA technique can be use to impute values for the missing frames. Figure 5.6a is the frame before the removed frames, and Figure 5.6b shows the corresponding skeleton shape reconstruction. The reconstruction of the missing frames using the PCA imputation method are shown in Figures 5.6d, 5.6f, 5.6h and 5.6j. The PCA imputation method has proven to be a useful technique for estimating the values of missing frames in C. elegans dataset. However, when the number of successively dropped or failed frames increases the performance of this method decreases.
Figure 5.6: Image skeletonization reconstruction using PCA imputation
### 5.3.5 Non-negative Matrix Factorisation

Here we consider the transformation of the worm’s skeleton tangent angles data to their corresponding NMF representation. Similar to the PCA transformation, this method also aim at reducing the dimension of the tangent angles data while retaining meaningful information about the original data. Although both the PCA and NMF methods represent an image object as a linear combination of the basis images, but with different results in terms of quality (Lee and Seung [1999]). As explained in Chapter 3, one major difference between the NMF and the PCA skeleton angles representation is that, the NMF algorithm prohibit the entering of negative values in the matrix factors $W$ and $H$ whereas, the PCA algorithm permit negative values in the matrix factors. For the PCA method, this implies that when the eigenworms are used for linear combination, there are cancellations between the positive and negative values. This results in many eigenworms not having intuitive meaning. For the NMF method, due to the non-negativity constraint, no subtraction occurs when the eigenworms are used for the linear combination of the worms skeleton images. To retrieve the NMF representation of the high dimension skeleton angles data, let the skeleton angles data $X$ contain $N$ samples each with $D$ dimensions. Therefore, the structure of the worms skeleton angles data can be defined as $X \in \mathbb{R}^{N \times D}$. The goal is to reduce the $D$ dimension data to $d$ low dimension data using NMF algorithm. The projection of the angles data unto the coefficient matrix $H$ can serve as input to a standard machine learning classifiers for the strain genotypes classification problem.

The NMF algorithm decomposes $X$ to $W \in \mathbb{R}^{N \times d}$ and $H \in \mathbb{R}^{d \times D}$. Each column of the matrix $W$ can be referred to as the bases tangent angles such as the head, tail and shape body bends angles, and represent the main building block for the reconstruction of all the original skeleton angles data. On the other hand, Each column of $H$ can be referred to as the coordinates of each frame skeleton angles that defines how an approximation of the original frame skeleton angles can be reconstructed based on a linear combination of the main building blocks in $W$. This matrix captures features that are in each frame skeleton angles. The single worm skeleton angles reconstruction can be defined as $\tilde{X} \in WH$, where $\tilde{X}$ is the approximation of $X$. We used the Frobenius norm in equation 5.24 to measure the accuracy of the approximation of $X$ with respect to $WH$, such that $W \geq 0$ and $H \geq 0$.

\[
\|X - WH\|_F^2 = \sum_{ij} (X - WH)_{ij}^2
\]  

(5.24)

One major issue in the use of NMF algorithm for data compression is how the values of the variables $W$ and $H$ are initialised and updated. Here, we use the Non-negative Double Singular Value Decomposition (NNDSVD) (Boutsidis and Gallopoulos[2008]) applied on the raw skeleton tangent angles to initialise $W$ and $H$. The NNDSVD method is faster and guarantees convergence when compared to the use of random initialisation method. Another major challenge in the transformation process is how to choose the factorisation rank $(d)$ value. The optimal number of components $d$ can be derived using methods such as cross validation (Kanagal and
Sindhwani [2010]) and cophenetic correlation (Brunet et al. [2004b]). Both methods are computationally expensive for a large dataset. In this work, we choose to use the same number of NMF components as that used in PCA for easy comparison of both representations. For each video frame \( f(t) = (\theta_1^{(t)}, \theta_2^{(t)}, \ldots, \theta_m^{(t)}) \) we transformed it into \( x(t) = (x_1^{(t)}, \ldots, x_d^{(t)}) \) low-dimensional NMF time series representation. The \( d \) denote the number of features retained and \( x(t) \in \mathbb{R}^d \). Also, the \( n \)th video can be defined as \( v(n) = (x_{(1)}^{(n)}, x_{(2)}^{(n)}, \ldots, x_{(T_n)}^{(n)}) \), where \( x_{(t)}^{(n)} \in \mathbb{R}^d \) for \( t = 1, \ldots, T_n \) and \( T_n \) represents the total number of videos extracted from each video file. Note that the worm skeleton angles data was normalised using the Min-MaxScalar (Pedregosa et al. [2011]) preprocessing method to contain values within the range of 0 and 1, this helps to remove the negative angles before the application of the NMF algorithm. Similar to PCA, we have reduced the dimension of the skeleton angles data from \( N \times D \) to \( N \times d \), where \( d < D \). To perform the worm genotypes classification, the input the machine learning classifiers are time series of the skeleton angles data projected unto the \( d \) basis coefficient matrix \( H \).

### 5.4 C. Elegans Genotypes Classification

In this section, we describe the computational tools and the structures of the learned representations used for the classification of the distinctive C elegans strains. The goal is to classify the C elegans strains as either wild-type (AQ2947) or one of the mutants (OW939, OW940, OW949, OW953 and OW956) using classifiers such as SVM, Random Forest (RF), \( k \)NN, and a deep learning model. The data used for the classification task are features (tangent skeleton angles) extracted from the body postures of the worms in 1200 single worm video records. The computational tools refers to the standard machine learning algorithms and the performance measurement metrics used in evaluating how good the algorithms can distinguish between the different strain types. The learned representations refers to the high-dimensional skeleton angles data and the low-dimensional PCA and NMF time series coefficients.

#### 5.4.1 Train-Test separation method

The training and testing sets split method is an important step that can influence the performance of any machine learning algorithm. Thus, the objective is to identify the class of worm and not an individual strain. This implies that, the test set cannot contain segments of the time series of an individual worm that was in the training set. A random split of the frames of all videos would not do. In this work, a proper train-test split must take into consideration these two points:

1. the identity of a worm is preserved over the length of a video
2. the strain of worm should be preserved across videos
In order not to compromise steps 1 and 2 by incorrect train-test split.

The dataset used for this experiment contain 1200 videos of six classes of worm with distinct genotypes. For the worm classification task, the dataset is always split into 840 samples (70% of 1200) training set and 360 samples (30% of 1200) testing set. We always ensure that each set contain equal number of videos per worm class. For each worm movie, we extracted 400 video frames, and for each video frame, we calculated 50 tangent skeleton angles along the worm’s body. Skeleton angles belonging to the same worm movie are horizontally stacked together to form a single time series observation and then stored as a feature vector for that worm movie. This implies that, the skeleton angles extracted from successive image frames in a single worm movie can be represented as $\mathbf{x} \in \mathbb{R}^{400 \times 50}$. Therefore, for the training set, there are 840 observations $\in \mathbb{R}^{20000}$. Similarly, there are 360 observations $\in \mathbb{R}^{20000}$ in the test set. We refer to this particular training and testing sets as high-dimensional skeleton angles data sets, because of its dimension. For the strains classification problem, the training data serves as input to the learning algorithms and the test data was used to evaluate the performance of the algorithms in predicting the class of the worms. The classification accuracy results using the high-dimensional skeleton angles data as input to the classifiers are shown and discussed in section 5.4.3. Furthermore, we transformed the high-dimensional skeleton angle data set into its corresponding low dimension representations using PCA and NMF dimension reduction techniques as discussed in sections 5.3.4 and 5.3.5 respectively. The high-dimensional angles data was reduced to $d$-dimensions, which account for certain percentage of the shape variance. This means that, each data point in the PCA or NMF representation is $\mathbf{x} \in \mathbb{R}^{d}$, where $d < 20000$. For easy comparison between the dimensionality reduction methods, we choose to maintain the same feature vector dimensions for PCA and NMF representations. Similarly, the sizes of the training and testing sets for the low-dimensional representations (PCA and NMF) are 840 and 360 samples respectively. The training set serves as input to the machine learning algorithms and the test set was used to evaluate the performance of the algorithms. The classification accuracy results using the PCA and NMF time series coefficients are also shown and discussed in section 5.4.3.

### 5.4.2 Learning model parameters

To characterise the *C. elegans* strains with well defined genotypes based on phenotypic features extracted from the changes that occurs in their body shape during movement, requires the use of an estimator or estimators that group or separate data points into their respective classes. These estimators also referred to as classifiers or learning algorithms, has the ability to learn from the data and make predictions on unseen data. In this work, we utilise three standard machine learning algorithms such as Support Vector Machine (SVM), Random Forest (RF) and $k$-Nearest Neighbour ($k$NN), and a deep neural network. These learning algorithms are usually use for the classification of biological data. Our goal is find a reliable model that suit
our data and can generalise when presented with unseen data. These estimators have hyper-
parameters that can be tuned in order to boost their prediction performance. However, these
hyper-parameters are not learnt within the estimators and therefore need to be specified. The
hyper-parameters tuned for the SVM model are:

1. kernel: this parameter transforms a given data from one feature space to another via a
   kernel function
2. penalty parameter (C): it is used to tell the SVM optimisation how much error is acceptable
3. gamma: determines the line of separation. If gamma is high, points that are close have
   high influence and if low, far points are also considered in order to define the decision
   boundary.

The hyper-parameters tuned for the RF model are:

1. Number of estimators (n_estimators): determines the number of trees in the forest.
2. Maximum depth (max_dept): determines the number of nodes allowed from the root to
   the farthest leaf of a tree.
3. Minimum sample leaf (min_sample_leaf): determines the minimum number of leaf
   needed at a leaf node.

For the kNN and deep classification model, we only tuned the number of nearest neighbours
to consider and the learning rate respectively. To tune the parameters of the models, we ap-
plied cross-validation using grid search. For the RF model: n_estimators: [5, 10, 50, 100, 150, 200],
max_depth: range(2, 10), and min_samples_leaf: [2, 3]. For SVM: C: range(2, 150, 2), kernel:[γ,
linear] and gamma was set to the default after using other values. For kNN: n_neighbors: range(2, 10),
and for the deep model: lr = [0.0001, 0.001].

5.4.3 C. elegans genotype using high-dimensional angles data

The section focuses on the use of dynamic motion behaviour of the worms monitored within a
specific time interval to infer their genotypes. Hence, the goal is to maintain a balance between
finding a suitable posture representation method and choosing the right predictive model with
the capability to understand the differences in the extracted features. Due to the missing val-
ues in the worm body skeleton angle dataset, there are two versions of the high-dimensional
angles data. First, is the angle data in which the missing values were estimated using quad-
ratic spline interpolation method and the second, is the angle data in which the missing values
were approximated using the linear interpolation method. The reason for using both versions,
is to investigate how the estimated missing values can influence the outcome of the classifier’s
predictions, with the aim to achieve high classification accuracy score. In addition, the focus
is not to choose a specific classifier based only on its ability to make good predictions about the worm classes, but to find a combination of meaningful feature representation and predictive model that can yield a better classification accuracy, and can generalise when presented with an unseen data. For simplicity and clarity, we used "Linear" or "Spline" in parentheses to mean, high- or low-dimensional features extracted from the angle data in which the linear or quadratic spline interpolation algorithm was used to estimate the missing values respectively. To perform the worm genotype classification task, we scaled both sets of data using the Scikit-learn standard scaler class and use them separately as input to the learning algorithms. The dataset used for this experiment consists of six classes of worms with distinct genotypes and therefore it is expected that any machine learning model should be able to achieve at least 16.67% classification accuracy by randomly guessing the points in the data. As shown in Table 5.2, is the genotype classification accuracy scores produced by the learning algorithms, when the high-dimensional angle data were used to train and test the prediction performance of the classifiers.

Table 5.2: Deep model classification accuracy for the skeleton angles with missing values computed using linear or spline interpolation

<table>
<thead>
<tr>
<th>Feature Type</th>
<th>Impute method</th>
<th>SVM (%)</th>
<th>RF (%)</th>
<th>KNN (%)</th>
<th>Deep classifier model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Top1 Acc. (%)</td>
<td>Top2 Acc. (%)</td>
<td>Top3 Acc. (%)</td>
<td>Top4 Acc. (%)</td>
</tr>
<tr>
<td>Angles</td>
<td>Linear</td>
<td>20.28</td>
<td>18.89</td>
<td>19.44</td>
<td>19.44</td>
</tr>
<tr>
<td>Angles</td>
<td>Spline</td>
<td>20.28</td>
<td>19.72</td>
<td>18.06</td>
<td>18.61</td>
</tr>
</tbody>
</table>

As seen in Table 5.2, the SVM classifier produced the best accuracy score of 20.28% for both data sets. The best accuracy score produced by the SVM estimator is slightly above the scores achieved by the other classifiers. As also seen in Table 5.2, the deep learning model performed better when linear imputation method was used to calculate the missing values in the data as compared to the accuracy got when quadratic spline was used to impute the missing values. To further investigate the performance of the individual standard machine learning algorithms, we considered the confusion matrix produced by each classifier for both sets of data, as shown in Figures 5.7 and 5.8. As demonstrated in Figures 5.7 and 5.8, all three classifiers performed badly in identifying features specific to the OW956 mutant worms for both the linear and spline imputed missing angle datasets. As also seen in Figures 5.7a and 5.8a, the SVM classifier yield the same accuracy score, however different features of the worm classes are picked out. This implies that, the method used to estimate the missing angle values can influence the classifier’s performance.
Figure 5.7: Confusion Matrices for the SVM, $k$NN and RF classifiers based on the angles data with missing points approximated using linear imputation method.
As illustrated in Figures 5.9 and 5.10, the ROC curve shows that all three classifiers failed to identify genotype specific features of the OW940 mutant worms, except for the kNN classifier when the missing angles in the data are interpolated using the quadratic spline method. The closeness of the curve for each worm class to the diagonal line indicates low predictions accuracy (slightly above the line) and failed predictions when below the line. As also seen in both figures, the performance of the SVM and kNN classifiers are the same for both the linear and spline imputed missing value dataset.
Figure 5.9: The ROC AUC curve for the SVM, $k$NN and RF classifiers using one-versus-all scoring metric based on the angles data, in which the missing angles values are approximated using linear interpolation technique.
Chapter 5 C. Elegans Phenotypic Feature Extraction and Classification

5.4.4 C. elegans genotypes classification using PCA embedding

Because of the low prediction accuracy scores produced by the learning algorithms and the curse of dimensionality using the high-dimensional angles data as input to the classifiers, we moved to PCA technique to extract compact but meaningful information from the high-dimensional angle data. This new skeleton angle representation still possesses relevant information about the original data and can serve as an approximate version of the complete data. Before the transformation of the angle data into the PCA feature subspace, we scaled the angle data using the Scikit-learn standard scaler class and thereafter split the data into 70% training and 30% testing sets. We did this for both the linear and quadratic spline imputed skeleton
angles data sets. To get the PCA principal components for the training data, first we computed
the mean of each feature in the data using equation A.1 and thereafter subtract the mean from
each observation in the data using equation A.2. Subsequently, we calculated the covariance
matrix of the centred data using equation A.3. From the covariance matrix, we extracted the
eigenvalues and its corresponding eigenvectors using equation A.4. Next, was to project the
centred training angle data onto the $d$ selected eigenvectors with the highest eigenvalues to
retrieve the PCA training embedding. To retrieve the equivalent PCA test embedding, we first
subtracted the mean of the training data from each point in the test data and then project the
centred test data onto the same $d$ selected eigenvectors previously calculated from the training
data. Finally, the training embedding is used to train the learning models, and we measured
their classification accuracies based on how accurately they predict the unseen test embedding.

Figure 5.11 shows the prediction accuracy scores achieved by the estimators on the unseen
PCA test embedding as the feature dimension increases from $d = 1$ to $d = 5$. In this work,
we choose to use only 5 dimensions, in order to compare with other feature extraction meth-
ods. As seen in the Figure, the PCA test embeddings with linearly imputed missing angles
data achieved the highest accuracy score of 21.39% when trained and evaluated using the $RF$
classifier. The same estimator (RF) produced the best score of 21.11% when the test embedding
derived from the skeleton angles, in which we computed the missing values using the quad-
artic spline interpolation method. Similar to the result achieved when the high-dimensional
features serves as inputs to the deep neural network, the PCA embeddings extracted from
the angles data in which the linear interpolation method was used to calculate the missing or
dropped frame values produced the highest score when compared to the input PCA embed-
dings generated from the angles data, with spline imputed missing values. The deep learning
model prediction score obtained with the PCA test embedding derived from angles with lin-
early imputed missing values, is slightly above the random guess value of 16.67%, as depicted
in Table 5.3. On the other hand, the accuracy score achieved when the principal components of
the angles with spline computed missing value serves as input to the deep model, falls below
the random guess value. We have also provided the 5-fold cross-validation grid-search plots
for key model parameters using the PCA training and test embeddings as seen in appendix
G.1. The shadow represents the score plus or minus the standard error for the tune parameter.

Table 5.3: Deep model classification accuracy for the skeleton angles with missing values com-
puted using linear and spline interpolation techniques

<table>
<thead>
<tr>
<th>Feature</th>
<th>Impute method</th>
<th>Deep classifier model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Top1 Acc. (%)</td>
</tr>
<tr>
<td>PCA embedding</td>
<td>Linear</td>
<td>17.50</td>
</tr>
<tr>
<td>PCA embedding</td>
<td>Spline</td>
<td>16.39</td>
</tr>
</tbody>
</table>

Similar to the experiments performed with the angles data, we studied the confusion mat-
rix produced by each standard machine learning algorithms in order to gain more insight as
regards the reason for low accuracy scores. Figure 5.12 shows that, the learning algorithms
were able to correctly classify some data points belonging to at least three different classes of
C. elegans strains. It was observed that, points from the mutant worm OW953 were more ac-
curately grouped by all three classifiers compared to other class points. Just like the confusion
matrices produced by all three estimators using the PCA embedding derived from angles with
the missing values imputed via linear imputation, all three classifiers accurately predicted few
points representing at least four different worm types when the PCA test postural features
derived from the spline imputed high-dimensional angles data was used, as depicted in Figure
5.13. Also, the confusion matrices in Figures 5.12c and 5.13c shows that, the mutant strain
OW956 points were highly misclassified by the \( k \)NN model. In fact this was observed even
with the high-dimensional angles data using the same \( k \)NN classifier.
Figure 5.12: Confusion Matrices for the SVM, kNN and RF classifiers based on PCA embeddings derived from angle data in which linear interpolation was used to calculate the missing values.
### Figure 5.13: Confusion Matrices for the SVM, kNN and RF classifiers based on PCA embeddings retrieved from angle data in which quadratic spline interpolation was used to compute the missing values

<table>
<thead>
<tr>
<th>SVM (Spline)</th>
<th>RF (Spline)</th>
<th>kNN (Spline)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.17 0.15 0.13 0.27 0.20 0.07</td>
<td>0.25 0.10 0.13 0.13 0.23 0.12</td>
<td>0.15 0.15 0.17 0.20 0.27 0.07</td>
</tr>
<tr>
<td>0.03 0.20 0.20 0.20 0.20 0.07</td>
<td>0.15 0.08 0.22 0.20 0.17 0.10</td>
<td>0.13 0.22 0.18 0.08 0.17 0.22</td>
</tr>
<tr>
<td>0.13 0.18 0.22 0.20 0.20 0.07</td>
<td>0.15 0.10 0.27 0.17 0.10 0.07</td>
<td>0.10 0.25 0.18 0.08 0.17 0.22</td>
</tr>
<tr>
<td>0.12 0.10 0.27 0.30 0.20 0.02</td>
<td>0.15 0.10 0.27 0.17 0.10 0.07</td>
<td>0.10 0.25 0.18 0.08 0.17 0.22</td>
</tr>
<tr>
<td>0.10 0.17 0.20 0.23 0.25 0.05</td>
<td>0.15 0.10 0.27 0.17 0.10 0.07</td>
<td>0.13 0.22 0.18 0.08 0.17 0.22</td>
</tr>
<tr>
<td>0.15 0.15 0.17 0.20 0.27 0.07</td>
<td>0.25 0.10 0.13 0.13 0.23 0.12</td>
<td>0.15 0.15 0.17 0.20 0.27 0.07</td>
</tr>
</tbody>
</table>

**Figure 5.14 and 5.15** shows the receiver operating characteristic (ROC) area under the curve (AUC) for multi-class classification problem using the PCA time series postural features derived from the angles data in which the missing values were estimated using the linear and spline interpolation methods respectively. In each plots, the black diagonal line indicates the chance level (50% since one-versus-rest classifier setting was used) and curves below this line indicate false predictions by the classifier. As illustrated in **Figure 5.14a**, the SVM classifier failed to predict correctly the points in the OW939, OW949 and OW956 strain classes. Hence, their AUC scores falls below the chance level (0.50 or 50%). On the other hand, points in the OW940 mutant strain were more correctly classified compared to the other worm classes. For the RF classifier ROC curve as shown in **Figure 5.14b**, the AUC score for all strains are slightly
above the chance level, except for the OW939 mutant worm in which the AUC score is below the chance level. Hence, the high performance achieved when trained and evaluated using the PCA embeddings. As demonstrated in Figures 5.14 and 5.15, all three classifiers seems to predict correctly the points in the AQ2947 worm class. As seen in Figure 5.15, the ROC curve for the OW940 worm is below the diagonal for all three classifiers. This indicates that points in this class were misclassified by all three classifier.

![ROC AUC curves for SVM, kNN, and RF](image)

Figure 5.14: The ROC AUC curve for the SVM, kNN, and RF classifiers using one-versus-all scoring metric based on the PCA embeddings, in which the missing angles values are approximated using linear interpolation technique.
Figure 5.15: The ROC AUC curve for the SVM, \(k\)NN and RF classifiers using one-versus-all scoring metric based on the PCA embeddings, in which the missing angles values are approximated using spline interpolation technique.

To further investigate the features learnt by the PCA algorithm from both sets of data, we plotted the first two principal components together, as seen in Figure 5.16. By comparing Figures 5.16a and 5.16b, it is observed that the training data points are tightly close to each other for the first two PCA embedding in which the missing angle values were approximated using the quadratic spline interpolation compared to the PCA training embedding derived from the angle data in which linear interpolation method was used. This may be a significant factor that influence the classification performance of the classifiers using both sets of PCA embeddings. Similarly, the points are more closer to each other for the PCA test embedding in which spline imputation was used to complete the missing angle data. The same results was observed for the 3D plots of the PCA training and testing embeddings as shown in Figure 5.17 (Linear and Spline). The 3D plots also shows that as the feature dimension increases, the points from different classes are strongly mixed with each other. Hence, it is difficult for the classifiers to categorise them into their respective classes.
Figure 5.16: PCA 2D training and test embeddings extracted from the high-dimensional skeleton angles data
Figure 5.17: PCA 3D training and test embeddings extracted from the high-dimensional skeleton angles data

The t-SNE 2D plots of the PCA training and testing embeddings for both sets of data shows that, points from different classes are strongly overlapped with each other, as seen in Figure 5.18. This means that, combination of the PCA and t-SNE embeddings did not reveal distinctive information about the differences in the locomotory behaviour of the different classes of worms.
The projections of each worm class data onto the selected eigenworms for both the training and test sets are found in Appendix I. For each plot, the points represent a worm time series video data for a particular class of worm. As characterised in both the training and test projections shown in Figures I.1 and I.2, and Figures I.3 and I.4 respectively, it is observed that the points in each worm class are tightly close to each other. However, the projections onto the first eigenvector is slightly separated from the other projections. Consequently, the reason for the high classification accuracy scores when the first projection serves as input to the machine learning classifiers. As also seen in the Figures, the projections onto the selected eigenvectors are close to zero. This indicate that the second to the fifth principal components did not contribute much to the variance like the first component. Overall, the PCA projections for the training and test embeddings shows that, there is no linear association between the tangent skeleton angles and the calculated eigenvectors. Therefore, it is hard for the classifiers to group the worms based on their unique genotypes. Due to the low performance of the classifiers when the PCA postural dynamic of each worm type was served as input, especially when compared with the score of 49.40% obtained in a related work done by Javer et al. [2019], we decided to use another unsupervised dimensionality reduction method like the non-negative factorisation matrix (NMF). The outcome of the experiment is discussed in the next section 5.5.
5.5 C. elegans genotypes classification using NMF embedding

Just like the C. elegans genotypes classification task performed using the PCA time series embeddings, we solved the same problem using the equivalent NMF time series embeddings. The main goal is to find a better approximation of the high-dimensional skeleton angles data that can further improve the classification accuracy scores (above the random guess of 16.67% and the best score achieved using the PCA embeddings) of the learning models in terms of grouping the worms into their distinct classes. To derive the NMF training and test embeddings, we first scaled the angle data using the Scikit-learn MinMaxScaler class and thereafter, split the data into training (70%) and testing (30%) sets. Subsequently, we transformed both sets into their respective NMF training and test embeddings using the NMF algorithm. Same as the PCA embeddings, we trained the classifiers with the NMF training embedding and evaluated their individual performance on an unseen data using the NMF test embedding. Figure 5.19 shows the prediction scores achieved by the classification models as the dimension of the NMF test embedding increases from 1 to 5. Same as the PCA implementation, we choose to use only 5 dimensions in order to compare with other dimensionality reduction methods. As demonstrated in Figure 5.19a, an accuracy score of 24.47% was achieved when the NMF embeddings derived from the skeleton angle data with linearly estimated missing values was trained using the kNN classifier. With the same NMF embeddings extracted from the linearly imputed skeleton angles, the accuracy scores produced by all three standard learning algorithms are slightly above the corresponding accuracy scores achieved when the missing angles were calculated using the quadratic spline interpolation method.

![NMF scores for different dimensions](image1)

(a) NMF test embedding (Linear imputed missing values)

![NMF scores for different dimensions](image2)

(b) NMF test embedding (Spline imputed missing values)

Figure 5.19: Classification scores obtained based on each NMF component on the test set

When both sets of NMF features (embeddings derived from the high-dimensional angle data using either linear or spline interpolation method to approximate the missing values) were used separately as input to the deep learning neural network, the same top 1 accuracy score of 20.56% was achieved as shown in Table 5.4. As done with the PCA embeddings, we looked
at the confusion matrix produced by the standard learning algorithms in order to know the proportion of points belonging to the different classes of *C. elegans* that were correctly or incorrectly classified. Also, we have provided the 5-fold cross-validation grid-search plots for key model parameters using the NMF training and test embeddings as evidenced in appendix G.2. The shadow represents the score plus or minus the standard error for the tune parameter.

Table 5.4: Deep model classification accuracy for the NMF test embeddings with missing values computed using linear and spline interpolation techniques

<table>
<thead>
<tr>
<th>Feature</th>
<th>Interpolation</th>
<th>Deep classifier model</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Top1 Acc. (%)</td>
<td>Top2 Acc. (%)</td>
<td>Top3 Acc. (%)</td>
<td>Top4 Acc. (%)</td>
<td>Top5 Acc. (%)</td>
</tr>
<tr>
<td>NMF</td>
<td>Linear</td>
<td>20.56</td>
<td>36.38</td>
<td>53.06</td>
<td>68.89</td>
<td>82.22</td>
</tr>
<tr>
<td>NMF</td>
<td>Spline</td>
<td>20.56</td>
<td>34.72</td>
<td>52.78</td>
<td>66.39</td>
<td>83.33</td>
</tr>
</tbody>
</table>

In Figure 5.20, we provided the confusion matrix for each learning model when the NMF features extracted from the linearly estimated missing skeleton angle data. As illustrated in the figure, all three classifiers to some extend correctly predicted unique features of mutant worm OW939 compared to the other classes of worms. Furthermore, features of the wild-type (AQ2947) and mutants (OW940, OW949, OW953 and OW956) were incorrectly classified as features of the mutant worm OW939. With the NMF test embedding derived from angles with linearly calculated missing values, the $k$NN classifier captures more feature specific information about the different classes of worms in the data compared to the other classifiers.
Similarly, Figure 5.21 shows the confusion matrix produced by the machine learning algorithms, when the NMF test embedding retrieved from the angle data with missing values imputed using quadratic spline interpolation method. As seen in the figure, class specific features of the wild-type (AQ2947) and the mutant strains (OW940, OW949, OW953 and OW956) are significantly misclassified as distinctive features of the YFP in fused mutant worm OW939, especially by the SVM classifier. Both the SVM and $k$NN classifiers performed badly in terms of predicting features specific to worm classes OW949, OW953 and OW956 as evidenced in Figures 5.21a and 5.21c respectively.
Figure 5.21: Confusion Matrices for the SVM, kNN and RF classifiers based on NMF embeddings

Next, we considered the ROC AUC curve to gain insight on how the sensitivity and specificity change for different threshold, as demonstrated in Figures 5.22 and 5.23. The AUC score for the mutant strains fused with alpha-Synuclein (OW953 and OW956) falls below the chance level, for the RF and kNN classifiers as characterised in Figures 5.22b and 5.22c respectively. All AUC scores for all worm classes are slightly above the chance level for the SVM estimator as shown in Figure 5.22a. For all classifiers, the OW939 mutant strains are more correctly classified when compared with the other worm classes. As seen in Figure 5.23, the performance of all learning models dropped when the NMF features derived from the angles data in which the missing values were approximated using quadratic spline technique. In this case, the RF classifier produced the least result. Using the ROC metric, we have seen that for different threshold, the performance of the learning models varies. For this reason, their predictions for
each worm class in the dataset may vary too. In addition, with the ROC-AUC metric, we have an ideal of which specific worm features that are not captured.

Figure 5.22: The ROC AUC curve for the SVM, $k$NN and RF classifiers using one-versus-all scoring metric based on the NMF embeddings, in which the missing angles values are approximated using linear interpolation technique.
Chapter 5. C. Elegans Phenotypic Feature Extraction and Classification

Figure 5.23: The ROC AUC curve for the SVM, $k$NN and RF classifiers using one-versus-all scoring metric based on the NMF embeddings, in which the missing angles values are approximated using spline interpolation technique.

Figure 5.24 shows the 2D plots of the NMF train and test embeddings derived from the high-dimensional angles data. As seen in the Figures 5.24a and 5.24b, the NMF train and test points extracted from the linearly estimated missing angles data are more separated from each other compared to the NMF train and test embeddings obtained via quadratic spline approximated missing angle data, as demonstrated in Figures 5.24c and 5.24d.
Figure 5.24: NMF 2D training and test embeddings of the high-dimensional skeleton angles data

The NMF 3D plots shows similar result as the 2D plots and can be found in Appendix H.1. As demonstrated in Figure 5.25, the t-SNE 2D embedding plots of the NMF train and test embeddings for both sets of data, shows that data points from different classes of worms are strongly mixed together. The NMF training and testing data projections can be found in Appendix I.2. Each worm class projections unto the eigenvectors looks similar, which indicates that the points are clustered together.
We observed that, the projections of the angle data (with spline approximation for missing values) onto the first NMF component for each worm type was clearly separate from the other point projections, as seen in Figures 5.6 and 5.8.

Although there was a slight improvement in the worm strain prediction accuracy using the NMF postural embeddings, but the best result was still below the 49.40% achieved by Javer et al. [2019] for their chosen strain types. For this reason, we switched to non-linear unsupervised features extraction method known as kernel principal component analysis (KPCA). The implementation and classification result obtained using the KPCA embeddings is considered in the next section (5.6).

5.6 *C. elegans* genotypes classification using KPCA embedding

In order to improve the classification accuracy scores obtained with the PCA and NMF linear data transformation techniques, we applied the unsupervised non-linear feature transformation method known as kernel principal component analysis (KPCA). As mentioned in section
3.4, this technique is an extension of PCA method and allows the extraction of low-dimensional embeddings from the kernel feature space. To extract the KPCA train and test embeddings, first the skeleton angle data was scaled using the MinMaxScaler class in Scikit-learn. Next, the high-dimensional angle data was split into 70% training set and 30% test set. Furthermore, the training data was transformed to a new feature space using the kernel function as defined in equation C.15. Subsequently, the features in the kernel space are normalised using the Gram formula as defined in equation C.12 to obtain the normalise Gram kernel matrix. Thereafter, we extract the eigenvalues and its corresponding eigenvectors from the Gram kernel matrix as given in equation C.10. Subsequently, we selected the $d$ dimensional eigenvectors with the highest eigenvalues. To get the KPCA train embedding, we projected the normalised Gram kernel matrix onto the $d$ selected eigenvectors. To get the KPCA test embedding, first we calculated the square difference between each point in the test data from every point in the high-dimensional training data. Thereafter, we compute the kernel matrix for the test data. Finally, the test kernel data is projected onto the $d$ selected eigenvectors previously calculated from the training data using equation C.11, to retrieve the required KPCA test embedding. In order to categorise the locomotory behaviours of the worms based on their known distinctive genotypes, we trained the learning models using the KPCA train embedding and quantify their performance using the KPCA test embedding. Figure 5.26 depict the test accuracy scores produced by each classifier as the dimension of the test features increases from $d = 1$ to 5. Like the PCA and NMF algorithms, we only maintain 5 feature dimensions in order to compare with other methods later. As expressed in Figure 5.26, the best accuracy score of 23.06% was obtained when the KPCA embeddings derived from the skeleton angles with linearly estimated missing values were used to train and make predictions on the test embedding. Just like the classification task conducted using the low dimensional NMF embeddings, when the KPCA embeddings derived from the linearly calculated missing angle data was used train the classifiers, the accuracy scores produced by all estimators outperforms the scores achieved when the KPCA embeddings was computed from the angle data with missing values estimated using the quadratic spline imputation method.
Chapter 5 C. Elegans Phenotypic Feature Extraction and Classification

Figure 5.26: Classification scores obtained based on each KPCA components on the test set

Similar result was obtained when the two sets of KPCA embeddings (linear and spline) served as inputs to the deep learning model as demonstrated in Table 5.5. Again, the best top 1 accuracy score was obtained from the KPCA embeddings extracted from the angle data in which linear interpolation method was used to approximate the missing values. As done with other methods, we have provided the 5-fold cross-validation grid-search plots for key model parameters using the KPCA training and test embeddings as seen in appendix G.5. The shadow represents the score plus or minus the standard error for the tune parameter.

Table 5.5: Deep model classification accuracy for the skeleton angles with missing values computed using linear and spline interpolation methods

<table>
<thead>
<tr>
<th>Feature</th>
<th>Interpolation</th>
<th>Deep classifier model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>Top1</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Acc. (%)</td>
</tr>
<tr>
<td>KPCA</td>
<td>Linear</td>
<td>21.94</td>
</tr>
<tr>
<td>KPCA</td>
<td>Spline</td>
<td>17.78</td>
</tr>
</tbody>
</table>

To further investigate the performance of each classifier in terms of correctly predicting the points specific to a class of worm using the KPCA test embedding, we analyse the confusion matrix produced by the three standard machine learning algorithms (SVM, RF and kNN). Figure 5.27 shows the confusion matrix produced by the three estimators, when the low-dimensional KPCA features were extracted from the high-dimensional angles data in which linear interpolation algorithm was used to compute the missing values in the data. Similarly, Figure 5.28 shows the confusion matrix obtained via the KPCA embeddings when the missing angles data were estimated using the quadratic spline interpolation method. As seen in Figure 5.27b, the RF classifier capture more distinctive features between the different classes of worms in the dataset.
than the SVM and kNN classifiers did. The mutant worms OW949 (YFP) and OW956 (Alpha-Synuclein) were not recognised by the SVM classifier, since points from these worm classes were significantly misclassified as mutants of OW940 and OW953 classes. Similar result was produced by the SVM classifier when the KPCA features were derived from the angle data in which the missing values were approximated using the spline method, as shown in Figure 5.28a. Although strains from the wild-type (AQ2947) and mutants (OW939, OW940, OW949, and OW953) were incorrectly classified as OW956 mutant worm by the RF classifier, however the RF model still recognises more distinctive features in the data than the other learning classifiers, as confirmed in Figure 5.28b.

Figure 5.27: Confusion Matrices for the SVM, kNN and RF classifiers based on KPCA embeddings
Figure 5.28: Confusion Matrices for the SVM, $k$NN and RF classifiers based on KPCA embeddings

Figures 5.29 and 5.30 shows the ROC curve for the multi-class classification problem using KPCA embeddings derived from the high-dimensional angle data, in which linear and spline interpolation methods were use to impute the missing values respectively. As illustrated in Figure 5.29, the AUC scores for all worm type are slightly above the chance score (0.50 or 50%) for all three estimators. This implies that, for the different pairs of sensitivity and specificity, the prediction accuracy of the three classifiers can be improved. Furthermore, all three classifiers the features of the OW939 and OW949 mutant worms are more rightly classified compared to the other classes of worms. On the other hand, as seen in Figure 5.30, the prediction performance of all three classifier dropped. However, the $k$NN model captured more distinctive locomotory behavioural information from the OW939 worm class than the other two classifiers. Overall, we have seen that the nature of the input representations influence the prediction
accuracy. And for each low-dimensional representations, the classes of worms picked out by the classifiers are slightly different.

Figure 5.29: The ROC AUC curve for the SVM, $k$NN and RF classifiers using one-versus-all scoring metric based on the KPCA embeddings, in which the missing angles values are approximated using linear interpolation technique.
As illustrated in Figure 5.31 for both sets of data, points belonging to a specific class of worm are mixed with points from other classes and therefore, difficult for the learning models to categorise them into their respective classes. The KPCA 3D plots can be found in Appendix H.2. Similarly, the t-SNE 2D embeddings of the KPCA training and test features, also shows that the points are not separable despite the transformation as demonstrated in Figure 5.32. This suggest that the KPCA method was unable to pick out features from the locomotory behaviour that distinguish the different worm genotypes.
Figure 5.31: KPCA 2D training and test embeddings of the high-dimensional skeleton angles data.
The KPCA training and testing data projections onto the selected eigenvectors can be found in Appendix I.3. As seen in the figure, the projections of each worm class onto the eigenvectors looks similar, hence indicate that the points are mixed together. Each worm class projections onto the eigenvectors, shows that the first component capture the difference between the points in the same class.

As evidenced in Figure 5.26, the best classification accuracy score of 23.06% achieved with the KPCA time series postural embeddings, was still lower than the 49.40% score obtained by Javer et al. [2019]. Although the KPCA score was slightly higher than the best PCA score, but it was less than the score obtained using the NMF pose features. Because the linear and non-linear transformation of the locomotory data has failed to capture genotypic specific differences of the postural dynamic of the worm without any form of supervision, next we investigated supervised learning techniques.
5.7 Summary

This chapter described the various steps taken in the classification of \textit{C. elegans} with distinctive genotypes using phenotypic features extracted from the body shape of the worms in each successive image frame in a movie file. The worm genotype classification task was first performed using time series of high-dimensional tangent skeleton angles measured along the skeleton body of the worm in each still image frame. The high-dimensional skeleton angles data was used to train the machine learning models and thereafter, evaluate their performance on a separate unseen skeleton angles data. Before using the high-dimensional angles data for the classification task, the missing values as a result of the dropped frames during the image segmentation and skeletonisation stages were imputed using linear and spline interpolation techniques. We observed that the spline interpolation method produced lesser errors when used to estimate the missing data compared to the linear method. When both interpolation methods were used to reconstruct the shape of the worm in a given frame, the shape produced by the linear method is closest to the original shape than the shape obtained using the spline method. Here we used the linear data imputation method for computing the missing values in the high-dimensional skeleton angles data. Thereafter, we performed the worm genotype classification problem using low-dimensional feature representations retrieved from the high-dimensional data. The reduced dimension data were retrieved using PCA, NMF and KPCA dimensionality reduction techniques. The compact feature data were used to train the machine learning models and measure their prediction accuracy on the unseen data. The classification accuracy score obtained shows that the KPCA time series features captured more meaningful information about the movement behaviour of the worms than the other representation methods. The accuracy score achieved using the compact but meaningful KPCA features outperformed the score obtained when the raw high-dimensional angles was used for the same classification task.
Chapter 6

C. Elegans Classification Using Fisher’s Linear Discriminant Analysis

6.1 Classification of C. elegans genotype using LDA embeddings

Based on the outcome of the classification results produced when the unsupervised linear and non-linear transformation of the C. elegans locomotory data failed to reveal genotypic specific differences between the strains without any supervision, we decided to investigate supervised feature learning techniques. Here, we considered the characterisation of the C. elegans worms with distinctive genotypes using the Fisher linear discriminant analysis (LDA). This supervised linear dimensionality reduction technique was used to transform the high-dimensional tangent skeleton angles data into a low-dimensional feature subspace, while preserving the class-discriminatory information. First, the skeleton angles data was split into training set (70%) and test set (30%), and scaled using the Scikit-learn MinMaxScaler class. Next, we compute the between-class ($S_B$) and within-class ($S_W$) scatter matrices from the training data using equations D.2 and D.4 respectively. Thereafter, we compute the eigenvalues and its corresponding eigenvectors from the $S_B$ and $S_W$ scatter matrices using equation D.6. Furthermore, we sort the eigenvectors in descending order of the eigenvalues and select the $d$ eigenvectors with the largest eigenvalues. Subsequently, the $d$ selected eigenvectors are use to transform the original training data into a new feature subspace (train embedding). To obtain the test embedding, we projected the scaled test data onto the selected eigenvectors obtained during the training data transformation using equation D.7. Finally, we perform the worm genotype classification task by training each classifier using the LDA train embedding and compute the prediction scores using the LDA test embedding. Figure 6.1 shows the classification accuracy scores achieved as the LDA test feature dimension increases from $d = 1$ to $d = 5$. As seen in Figure 6.1a, the best accuracy score of 21.94% was achieved when the RF classifier was used
to make predictions on the LDA test embedding (for the linearly estimated high-dimensional angles data). Similarly, the RF classifier produced the best accuracy score of 22.22% when the LDA test embedding was extracted from the high-dimensional shape-based data, in which spline interpolation technique was used to estimate the missing values. By comparing the performance of the SVM model in Figures 6.1a and 6.1b, it was observed that the method in which the missing values in the high-dimensional data are calculated can influence the quality of predictions produced by the learning algorithm. For instance, the SVM model performed better when the LDA test embedding was retrieved from the high-dimensional shape-based feature data, in which linear interpolation technique was used to compute the missing values. Whereas, in the case of the $k$NN classifier, it performs better when the LDA test features are derived from the angle data in which the spline interpolation method was used to estimate the missing values.

![Figure 6.1: Classification scores obtained based on each LDA component on the test set](image)

(a) LDA test embedding (Linear imputed missing values)  
(b) LDA test embedding (Spline imputed missing values)

We have provided the 5-fold cross-validation grid-search plots for key model parameters using the LDA training and test embeddings as seen in appendix G.6. The shadow represents the score plus or minus the standard error for the tune parameter. Table 6.1 shows the top 5 accuracy scores obtained when the deep learning model was trained and used to make predictions on the LDA test embeddings for both the linear and spline estimated missing angles data. The performance of the deep model was below the random guess score in both instances. However, the score (15.83%) obtained using the features extracted from the linearly interpolated missing angles data.
Table 6.1: Deep model classification accuracy for the skeleton angles with missing values computed using linear or spline interpolation

<table>
<thead>
<tr>
<th>Feature</th>
<th>Interpolation</th>
<th>Deep classifier model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Top1 Acc. (%)</td>
</tr>
<tr>
<td>LDA</td>
<td>Linear</td>
<td>15.83</td>
</tr>
<tr>
<td>LDA</td>
<td>Spline</td>
<td>14.72</td>
</tr>
</tbody>
</table>

Figures 6.2 and 6.3 shows the confusion matrices produced by the learning algorithms for the LDA embeddings obtained via the angles data with missing values interpolated using linear and spline methods respectively. As seen in Figure 6.2, the OW939, OW949, OW953 and OW956 mutant worms are highly misclassified as the OW940 mutant strain, except for the kNN classifier in which distinctive features of the wild-type(AQ2974) and the mutant (OW939) worms were also captured.
Figure 6.2: Confusion Matrices for the SVM, $k$NN and RF classifiers based on LDA embeddings
The ROC curve for multi-class classification task using the LDA features extracted from the high-dimensional angle, in which the missing values were imputed using the linear and spline interpolation methods are shown in Figures 6.4 and 6.5 respectively. The three classifiers (SVM, RF and \( k \)NN) performed better when predictions were made on the LDA features extracted from the angles data in which spline imputation method was used to estimate the missing values. This is different from the previous results obtained using the PCA, NMF and KPCA feature extraction methods. In this case, the \( k \)NN model performed better than the SVM and RF classifiers as demonstrated in Figures 6.4 and 6.5.
Chapter 6 C. Elegans Classification Using Fisher’s Linear Discriminant Analysis

Figure 6.4: The ROC AUC curve for the SVM, kNN and RF classifiers using one-versus-all scoring metric based on the LDA embeddings, in which the missing angles values are approximated using linear interpolation technique.
Chapter 6 C. Elegans Classification Using Fisher’s Linear Discriminant Analysis

Figure 6.5: The ROC AUC curve for the SVM, kNN and RF classifiers using one-versus-all scoring metric based on the LDA embeddings, in which the missing angles values are approximated using spline interpolation technique

As depicted in Figures 6.6a and 6.6b, five clusters of C. elegans worms were seen in the 2D LDA training embedding plots, for both the linear and spline estimated missing angle data. However, in the 2D LDA test embedding plots shown in Figures 6.6c and 6.6d, data points from different classes of worms are mixed together and hence hard to distinguish. This indicates that the LDA dimension reduction method did not capture meaningful information from the test data. The 3D LDA training and testing embedding can be found in Appendix H.3. The 3D plots are similar to the 2D plots shown here.
Chapter 6 C. Elegans Classification Using Fisher’s Linear Discriminant Analysis

Figure 6.6: LDA 2D training and test embeddings of the high-dimensional skeleton angles data

The LDA training and testing data projections for each worm class video onto the selected eigenvectors can be found in Appendix I.4. It was observed that each worm video data projection varies from the projection of another worm video in the same class. The projection plots capture the variance between each worm video in a particular class from another worm video in the same class. Because each component projections per worm class is different from another worm, hence the clear separation between the points as seen in the 2D t-SNE training plot.

The t-SNE 2D plots for the LDA training and test embeddings extracted from the linear and spline estimated missing angles dataset is shown in Figure 6.7. As shown in Figures 6.7a and 6.7b, the t-SNE 2D training embedding plot of the LDA training features, clearly shows six clusters representing the six classes of C. elegans worm with unique genotypes. Whereas, in the t-SNE 2D embedding plots of the LDA test features, the points are strongly mixed together and difficult to distinguish, as shown in Figures 6.7c and 6.7d.
The best prediction score of 22.22% obtained when predictions were made on the LDA test embedding, was less than the scores achieved with the unsupervised learning techniques such as NMF and KPCA were used. In addition, looking at the t-SNE 2D training embedding of the locomotory data as displayed in Figure 6.7, clear clusters of the six classes of worms in the dataset was seen. This implies that, the LDA algorithm was able to capture discriminative features of the different classes of worm in the training data but failed to generalise on the LDA test data. Just like we did with the PCA techniques, we decided to use a non-linear version of the LDA algorithm known as kernel linear discriminant analysis (KLDA). The outcome of the experiment is discussed in the next chapter (7).

6.2 Summary

This chapter provides information on the classification of *C. elegans* worms with distinctive genotypes using a supervised linear dimensionality reduction method known as Fisher’s linear discriminant analysis (LDA). Like the linear PCA and NMF feature dimension reduction method, the LDA algorithm extract the low-dimensional representation of the high-dimensional
skeleton angles data but with the help of the class label for each data point. Each machine learning algorithm was trained using some part of the LDA features and the quality of the predictions were evaluated on the unseen LDA test data. The classification accuracy score obtained using this approach is slightly less than the scores achieved with the NMF and KPCA features representation. The dimension of the LDA features used as input to the classifiers is restricted to number of classes in the dataset. At most, only $k - 1$ feature dimension can be used as input to the classifiers. The $k$ denotes the number of classes in the dataset. Due to the failure of the LDA algorithm to generalise based on the fact that clear separation between the different classes of worm was seen in the t-SNE 2D training plot of the LDA training feature, we implemented the KLDA feature extraction technique.
Chapter 7

C. Elegans Geneotypes Classification Using KLDA Features

7.1 Introduction

Here we considered a nonlinear feature extraction technique with dimensionality reduction referred to as kernel Fisher linear discriminant analysis (KLDA) for the characterisation of the locomotive behaviour of *C. elegans* worms with different genotypes. The KLDA algorithm is an extension of the standard Fisher linear discriminant analysis (LDA) that utilises a kernel function in the extraction of meaningful features in a given input space. The core ingredient of the KLDA method to feature extraction is the use of the kernel trick, in which significant features are extracted from the original input space via the use of a nonlinear kernel function. The kernel function uses a nonlinear mapping method to extract nonlinear discriminant features from the input data. The input data to the KLDA algorithm in this work are time series sequence of tangent angles measured along the skeleton body of the worms in successive video frames. The significant low-dimensional features extracted using the KLDA method serves as input to three standard machine learning classifiers such as SVM, kNN and RF, and a deep classifier model. To increase the classification accuracy, the KLDA algorithm finds a projection that maximises the class separation in the data while minimising the variance within each class. The performance of this method is compared with the previously used methods such as PCA, NMF, KPCA and LDA.

7.2 Classification of *C. elegans* genotypes using KLDA embedding

We conducted the *C. elegans* genotype classification task using the supervised kernel linear discriminant analysis (KLDA) to transform the high-dimensional skeleton angles data. First,
the skeleton angles data which describes the dynamic motions of the worms over a period of time was split into training and testing sets. Thereafter, the data was scaled using the Scikit-learn MinMaxScaler class. Subsequently, the KLDA dimensionality reduction technique was applied on the training set to extract the KLDA training embedding. To achieve this, we first transformed the training data to a kernel feature space (kernel matrix) using equation E.3, and compute the within-class ($S_W$) and between-class ($S_B$) scatter matrices from the kernel feature space using equations E.16 and E.19. Furthermore, we calculated the eigenvectors and its corresponding eigenvalues using equation E.22. Next, we projected the kernel matrix onto the $d$ selected eigenvectors with the largest eigenvalues. To get the KLDA test embeddings, we compute the kernel RBF between each point in the test data and every point in the original training data, and projected the kernel test data onto the $d$ selected eigenvectors with the largest eigenvalues using equation E.25. For the genotypes classification task, we trained the machine learning algorithms using the KLDA train embeddings and evaluate their performance using the KLDA test embeddings. In this experiment, we used the Gaussian Radial Basis Function (RBF) as the kernel function. The value of the free parameter sigma ($\sigma$) in the kernel function was set to 15 and the value of $\mu$ which ensure that the within-class scatter matrix ($S_W$) is not singular was set as 0.8. Figure 7.1a shows the classification accuracy results as the dimension of the test embedding increases from $d = 1$ to $d = 5$. At most only $d = 5$ features dimension can be used for the classification task, since there are only 6 classes of worms present in the dataset. As seen in both Figures 7.1a and 7.1b, the first dimension of the KLDA test embedding (Linear and Spline) contributes more to the individual classifier performance. As seen in Figure 7.1a, the kNN and RF classifiers produced the best accuracy score of 24.17% when the KLDA test embedding (Linear) dimension equals 5. On the other hand, the best accuracy score of 24.44% was achieved by the SVM classifiers when the KLDA test embedding (Spline) dimension equals 3, as illustrated in Figure 7.1b. The results shows that, the performance of the classifiers varies based on the process used to retrieved the transformed low-dimensional KLDA features.
We have provided the 5-fold cross-validation grid-search plots for key model parameters using the KLDA training and test embeddings as seen in appendix G.7. The shadow represents the score plus or minus the standard error for the tune parameter. Table 7.1 shows the classification accuracy scores achieved when the deep learning model was trained with the KLDA training embedding (linear and spline) and predictions were made for the unseen KLDA test embedding (linear and spline). The best top 1 accuracy score of 21.39% was achieved, when predictions were made for the KLDA test embedding in which the missing values in the high-dimensional angle data were estimated using the linear interpolation method.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Interpolation</th>
<th>Deep classifier model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>Top1</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Acc. (%)</td>
</tr>
<tr>
<td>KLDA</td>
<td>Linear</td>
<td>21.39</td>
</tr>
<tr>
<td>KLDA</td>
<td>Spline</td>
<td>20.00</td>
</tr>
</tbody>
</table>

In Figure 7.2, we provide the confusion matrices produced by the three standard machine learning classifiers (SVM, RF, and kNN), when the KLDA embeddings were derived from the high-dimensional angles data in which the missing values are estimated using linear interpolation technique. As seen in the figure, all three classifiers produced similar result for each worm class in the data. The wild-type (AQ2947), the mutant fused with YFP (OW949) and the mutants fused with alpha-Syncrein YFP (OW953 and OW956) are significantly misclassified as either OW939 or OW940 mutant worms by the learning algorithms (SVM, kNN and RF). As also
shown in all three confusion matrices in Figure 7.2, few discriminative information about the OW939, OW940 and OW953 mutant worms are identified by the three classifiers. A slightly different results was obtained for the confusion matrices produced by the three standard classifiers, when the KLDA embeddings were retrieved from the high-dimensional angle data in which the missing values are approximated via quadratic spline interpolation technique.

![Confusion Matrices](image)

(a) SVM  
(b) RF  
(c) kNN

Figure 7.2: Confusion Matrices for the SVM, kNN and RF classifiers based on KLDA embeddings

A slightly different confusion matrices were produced by the three classifiers, when the KLDA embedding were extracted from the angles data for which the missing values are approximated using the quadratic spline interpolation technique, as demonstrated in Figure 7.3. As shown in Figure 7.3, discriminative features of the OW940 mutant worms were more correctly classified.
by all three classifiers when compared with the proportion of data points that are correctly classified for other classes of worms. As depicted in Figures 7.3a and 7.3b, the wild-type (AQ2947) and the mutant strains (OW939, OW949, OW953 and OW956) are misclassified as the OW940 mutant worm. Similar result was obtained with the $k$NN classifier as shown in Figure 7.3c, however some distinctive KLDA features of the mutant worm (OW9393) was captured by the machine learning model.

By comparing Figures 7.2 and 7.3, all three classifiers failed to capture discriminative KLDA features that are specific to the wild-type (AQ2974), the mutant fused with YFP (OW949) and mutants fused with alpha-Synuclein YFP (OW953 and OW956) strains. Figures 7.4 and 7.5 depicts the ROC curve for three estimators based on the KLDA embeddings extracted from the
angle data from which linear and spline imputation methods were used to approximate the missing values respectively. As shown in Figure 7.4, the \( k \)NN classifier produced the best AUC scores when compared with the scores of other predictive models. As the threshold increases the AUC scores are slightly above the chance level for each worm class as evidenced in Figure 7.4c. As seen in Figures 7.4a and 7.4b, the SVM and RF models failed to learn enough distinctive features of the OW956 mutant worms. Similarly, as shown in Figure 7.5, all three classifiers performed badly while categorising the points in the wild-type strain (AQ2947). Unlike the result obtained with the LDA algorithm, the KLDA features extracted from the linearly estimated missing angle values produced highest AUC scores for all three classifiers.

Figure 7.4: The ROC AUC curve for the SVM, \( k \)NN and RF classifiers using one-versus-all scoring metric based on the KLDA embeddings, in which the missing angles values are approximated using linear interpolation technique.
Figure 7.5: The ROC AUC curve for the SVM, $k$NN and RF classifiers using one-versus-all scoring metric based on the KLDA embeddings, in which the missing angles values are approximated using spline interpolation technique.

As evidenced in Figures 7.6a and 7.6b, the KLDA 2D training embedding shows that the points representing the wild-type (AQ2947) and the mutants (OW949 and OW956) were partially separated from the other class points in the data, except in Figure 7.6b in which the mutant (OW949) was mixed with the other points. This indicates that, the KLDA algorithm was able to extract genotype specific features of the AQ2947, OW949 and OW956 strains. As demonstrated in Figures 7.6c and 7.6d, the points in the KLDA 2D test embedding plots are strongly mixed together (inseparable).
Figure 7.6: KLDA 2D training and test embeddings of the high-dimensional skeleton angles data

The 3D plots of the KLDA training embeddings also show a partial separation between the different data points as seen in Appendix H.4a and H.4b. However, for the KLDA 3D test embeddings plots, the data points from different classes are strongly mixed together as evidenced in Appendix H.4c and H.4d.

Figure 7.7 depicts the t-SNE 2D training and testing embeddings of the KLDA training and testing features. As shown in Figure 7.7a, six clusters representing the six classes of worms was seen for the t-SNE 2D training embedding of the KLDA features in which the missing angles were estimated using linear interpolation technique. However, this is different from the t-SNE training plot of the KLDA feature in which the spline technique was used to input the missing values. As seen in the t-SNE 2D test embedding plots in Figures 7.7c and 7.7d, points from the different classes of worms strongly cluster together. Therefore, did not reveal features of the locomotory behaviour that can be used to distinguish the different worm genotypes.
Figure 7.7: t-SNE 2-dimensional embeddings of the LDA training and test features

The KLDA training and test angles data projections onto the selected eigenvectors can be found in I.5. Like the LDA training data projections onto the selected eigenvectors, similar result was obtained with the KLDA training set projections. That is, projections for each worm class is slightly from the projections in another worm class, especially for the angle data in which linear interpolation technique was used to estimate the missing values, as evidenced in Figures I.17 and I.18. This explain the partial separation between some worm class points from other worm class points as characterised in the t-SNE 2D training plot. However, for the KLDA test data projections as shown in Figures I.19 and I.20, the projections are similar for each worm class and therefore, difficult for the classifiers to distinguish between them.

7.3 Discussion

In this study, we have demonstrated that the supervised kernel linear discriminant analysis can be used to extract low-dimensional features that can improve the classification accuracy score of distinct *C. elegans* strains previously done using raw angles data, PCA, KPCA and LDA features. Although the accuracy score obtained in this work using NMF features outperformed the
score obtained with the KLDA embedding, but we believe that score obtained with the KLDA features can be further improved. This can be achieved by searching for a pair of $\sigma$ value in the Gaussian RBF kernel as defined in equation E.3 and the $\mu$ parameter in the within-class scatter matrix regularisation formula as defined in equation E.23. Both parameters determines how the points in the data are separated. Using a robust method to extract the values of these parameters can improve the classification accuracy. Also, we have demonstrated that, the KLDA features extracted from the high-dimensional angles data in which linear interpolation technique was used to estimate the missing values, can boost the *C. elegans* genotype classification accuracy.

### 7.4 Summary

This chapter discussed the use of supervised kernel Fisher linear discriminant analysis for the classification of *C. elegans* locomotory behaviour based on the extracted shape-based features from successive image frames in each video file. Like the LDA method, the feature extraction process was done using the class-labels. Also, the number of meaningful feature dimension extracted is restricted to the number of classes in the data. For instance, if there are $k$ classes in the data, only $k-1$ significant features can be used for the classification task. The low-dimensional KLDA features are divided into training and testing sets. The KLDA training data is use train the classifiers and the KLDA test data is use to measure the quality of predictions produced by the individual classifier. During the low-dimensional KLDA feature extraction phase, two parameters influences the kind of features obtained. The first and most important is the $\sigma$ value in the Gaussian RBF kernel function as defined in equation E.3. The second parameter the $\mu$ value in the within-class regularisation formula in equation E.23. In this work, the choice of $\sigma$ and $\mu$ values are based on the quality of predictions produced by the classifiers since both variables are used simultaneously in the feature extraction process. The classification accuracy score obtained using the feature extraction approach outperforms the scores obtained when PCA, KPCA and LDA were used to extract features from the high-dimensional skeleton angles data. Although the accuracy score obtained with the NMF features was higher than the score obtain with the KLDA features, we believe that could be change by choosing the right combination of the kernel function parameter $\sigma$ and the $\mu$ parameter in the within-class scatter matrix regularisation equation.

Similar to the t-SNE 2D training embeddings of the LDA postural features, the same result was seen with the KLDA t-SNE 2D training embedding as characterised in Figure 7.7. Six distinctive clusters representing the classes of worm genotypes were partially formed, especially for the t-SNE 2D embedding of the KLDA training feature in which the missing data were linearly interpolated. However, the KLDA algorithm did not also generalise on the corresponding KLDA test data. A possible solution is to increase the size of the data. Alternatively, is to adjust the values of $\sigma$ and $\mu$ parameters used for the kernel function and the within-class scatter matrix regularisation respectively.
Chapter 8

Conclusions and Future Works

8.1 Conclusions

Due to the advent of modern technologies, researchers have developed automated tools that has the capacity to store, process and analyse large image data sets. This has facilitate the study of locomotory behaviour in model organism like *Caenorhabditis elegans* (*C. elegans*) in many biological laboratories across the world. This model organism has been used to study diseases such as Parkinson’s (neurodegenerative disorder) that affects humans. This is possible because their genome has been completely analysed and the ability to intervene in the organism’s genes that lead to changes in its nervous system has been proven successful. In this study, we have implemented a computational approach that characterises the locomotory behaviour of different class of *C.elegans* strains (AQ2947, OW393, OW940, OW949, OW953 and OW956). The AQ2947 is a wild-type worm, while the others are mutant worms. Techniques in computer vision and machine learning has made it easier to extract features from video images of these worms and process the data in order to discover hidden information. Here we examine and analyse the movement patterns of the worms based on their body posture in the movie frames. The videos used in this work are high magnification videos of single nematode worms tracked over a time period. We performed image segmentation, image skeletonisation, and feature extraction on the retrieved image frames for all videos. Next, we extracted high-dimensional tangent angles from the worms skeleton body shapes for each movie file. Subsequently, we applied linear and quadratic spline interpolation techniques to impute the missing values in the angles dataset. Thereafter, dimensionality reduction techniques such as PCA, NMF, KPCA, LDA and KLDA are use to extract compact but significant information from the high-dimensional angle data. For instance, related work done in this area using PCA technique for feature extraction has shown that by preserving four principal components, we can achieve 95% of the worm’s shapes variance (Stephens et al. [2008] and Gyenes and Brown [2016]). We have also shown that with four principal components, we achieved 94% worm shape variance. Recently, the extracted PCA features are use to conduct worm genotype classification task (Javer et al. [2019]). In this study, we aimed at conducting the same experiment
but with different sets of *C. elegans* strains. Using the classification score obtained by Javer et al. [2019] as our baseline, we solved the same classification problem by first using PCA time series coefficients. The outcome of the experiment shows that, there was no clear separation between the different classes of worms using these representations for our chosen dataset. Thereafter, we conducted the same experiment using time series of NMF phenotypic features. Although the performance score (24.72%) obtained was slightly higher than the score achieved using PCA technique, but significantly lower than the baseline score of 49.40%. In order to improve the accuracy scores obtained using PCA and NMF, we implemented the KPCA algorithm to extract shape-based features that describes the worm’s locomotory behaviour and thereafter, performed the genotype classification task using these features. The classification accuracy score got outperforms the PCA performance score, but was slightly less than the NMF score. However, we a deep learning classifier, the KPCA prediction score outperforms the NMF result. The best scores obtained so far is still less than our baseline score of 49.40%,and the linear and nonlinear unsupervised techniques has failed to identify genotype specific differences between the locomotory behaviour of the worms without any supervision. For these reasons, we moved at using the supervised linear discriminant analysis (LDA) to extract the shape-based features and performed the classification task. Again, the result achieved with the LDA embedding of the skeleton angles was slightly above the PCA score, but lower the NMF and KPCA scores. However, the t-SNE 2D embedding plot of the LDA training feature, shows a clear separation between the points from different classes and a tight cluster for points in the same class. This indicate that, the LDA algorithm in this case failed to generalise. Finally, we the performed the classification problem using the nonlinear variant of LDA such as KLDA. The accuracy score produced by the classifiers, outperforms the scores got using PCA, KPCA and LDA features, but slightly less than the score obtained when the NMF time series feature was used. Although the best score obtained in the work slightly above half the baseline score achieved by Javer et al. [2019] for their chosen class of *C. elegans* strains, however we have shown that different feature representation methods rather than PCA time series feature can improve the worms genotype classification accuracy. In addition, the prediction score obtained when the KPCA features was used to trained the deep classifier outperforms the scores achieved with the other representation methods. Finally, we have also shown in this work that, the missing value imputation method can also influence the outcome of the classifier’s predictions. Therefore, the implementation of a robust missing value interpolation method can help boost the classification accuracy. We notice that the use of linear interpolation technique to estimate the missing values of the skeleton angles, increases the performance of the deep model.

### 8.2 Future Works

Based on the difficulties we encountered amid this investigation, most of the tasks we recommend here consists of some currently existing issues that has note been completely resolved,
some newly identified problem and as well new ideas. A lot can still be done in terms of improving the classification accuracy of \textit{C. elegans} with distinct genotype using phenotypic postural features. This involves the careful selection of worms with well-defined or known locomotory disabilities to the implementation of robust predictive model with the capacity to generalise when presented with unseen data or different feature representation. The quality of the worms video determines the amount of image processing that has to be done in order capture or reveal genotype specific differences between the various classes of worm in the dataset. The creation of quality single and multiple worms video will inspire the implementation of more deep learning-based image processing algorithms with the ability to detect and extract meaningful postural information about the locomotory behaviour of the worms. Although the traditional image object segmentation and skeletonisation techniques are still useful to aid the extraction relevant data from images, however the recent introduction of robust deep learning models has made the job more exciting such that, the extraction of meaningful information from images is now earlier and faster. However, to tackle the problem dropped image frames during feature extraction process is not completely resolved. It is difficult to extract angles data from the worm skeleton body when the worm’s touches itself or the worm shape form a close-circle, hence that frame is dropped. These set of frames dropped also introduce bias in the animal’s behaviour analysis process, because the close-circle shapes also represent a motion state in the worm’s motion dynamics. A possible solution to this problem is to implement an algorithm that can identify a close-hole skeleton shape, randomly choose two unequal coordinate points in the worm’s body and measure the tangent angles starting from one end to the other end. An alternative solution is to use a robust missing value interpolation method to estimate the closest value to the dropped image frame. As mentioned earlier, the missing value interpolation method can influence the performance of the predictive model. Therefore, a robust missing data imputation method would minimise the loss of visible phenotypic information about the worm’s movement patterns. In this work, we have shown how the data interpolation method effects the prediction accuracy of three standard learning algorithm (SVM, RF and \textit{k}NN) and a deep model. Recently, there are deep learning-based object detection, segmentation and skeletonisation models (Long et al. [2015], Ronneberger et al. [2015], Chen et al. [2016], Luc et al. [2016], Badrinarayanan et al. [2017], Xue et al. [2018], (He et al. [2018] and Chen et al. [2019]), which can facilitate the extraction of relevant features or region of interest (ROI) from images. To improve the prediction accuracy of the deep network, requires quality images to train it. To improve classification accuracy of the deep model, also require large dataset to train it, which is hard to get. Furthermore, we have shown that, the use of different feature extraction method such as NMF, KPCA, LDA and KLDA can improve the worm class prediction accuracy. For instance, different combination of dimensionality reduction techniques can provide a robust feature extraction method that can generalise and produced good classification results. Finally, both traditional and deep classifiers perform differently for different input data. The implementation of a robust classification model that can generalise for different worm dataset will be a welcome development. The implementation done here shows that, the prediction accuracy of the selected classifier is determine by the structure of the feature representation
method.
Appendix A

Principal Component Analysis (PCA)

A.1 The PCA problem formulation

Let $X$ represents a dataset of images $[X_1, X_2, \ldots, X_M]$ and each image is of size $N \times N$. The task is to reduce each image in the dataset to an image vector of size $N^2 \times 1$ and then transform the images into a low-dimensional space of size $M \times k$, where $k \ll N^2$. In order to capture a significant amount of variability in the data using a small value of $k$.

A.2 The PCA algorithm

In order to solve the problem, the following steps were taken:

1. Collect the dataset which consists of the image vectors
2. Calculate the mean vector $\mu$ defined as:
   \begin{equation}
   \mu = \frac{1}{M} \sum_{i=1}^{M} X_i \tag{A.1}
   \end{equation}

3. Calculate the difference between the mean vector and each image vector using:
   \begin{equation}
   \Phi_i = X_i - \mu \tag{A.2}
   \end{equation}

4. Calculate the covariance matrix $C$ using:
   \begin{equation}
   C = \frac{1}{M} \sum_{n=1}^{M} \Phi_n \Phi_n^T \tag{A.3}
   \end{equation}
The size of the covariance matrix $C$ is $N^2 \times N^2$ matrix.

5. Calculate the eigenvectors $u$ and its corresponding eigenvalues $\lambda$ from the covariance matrix $C$ by solving the eigenvalue decomposition problem:

$$ Cu_i = \lambda_i u_i $$

(A.4)

In some cases it is unrealistic to compute the eigenvectors from the covariance matrix $C$ calculated using the product $\Phi \Phi^T$, especially for a large image dataset with high dimensions. Hence, the covariance matrix is usually calculated using the product $\Phi^T \Phi$ which returns a matrix of size $M \times M$ in this case. If $v$ denotes the eigenvectors calculated from the covariance matrix using the product $\Phi^T \Phi$, then the eigenvectors $u$ are related to $v$ through the expression $u_i = \Phi v_i$ since the eigenvalues are the same in both instances. In this case we can calculate the $M$ eigenvectors using $u_i = \Phi v_i$.

6. Arrange both the eigenvalues $\lambda_i$ and eigenvectors $u_i$ in the descending order of the eigenvalues. Select only $k$ eigenvectors that corresponds to the $k$ highest eigenvalues.

The various steps outlined has helped to reduce the dimension of the original image vectors from $M \times N^2$ size to a more manageable $M \times k$ dimension. For a facial image dataset, the eigenfaces can be computed using:

$$ \tilde{u}_\mu = \sum_{j=1}^{M} u_{jk} \Phi $$

(A.5)

The values of $\tilde{u}_\mu$ are usually normalise such that $||\tilde{u}_\mu|| = 1$. A new face image $(X_{\text{new}})$ is transformed into its equivalent eigenface components using:

$$ \omega_j = u_j^T (X_{\text{new}} - \mu) $$

(A.6)

where $j = 1, \ldots, k$ and $k$ denotes the number of significant eigenvectors selected. For a facial image recognition task, Turk and Pentland [1991] suggest the use a weight vector $\gamma = [\omega_1, \omega_2, \ldots, \omega_3]$ which describes the contribution made by each eigenface in the approximation of the original face images. The weight vector is also obtained by projecting the original face images onto the derived low-dimensional basis images or eigenfaces space. Then the distance between a given face image and the low-dimensional face space is defined as:

$$ \varepsilon^2 = ||\Phi - \Phi_w||^2 $$

(A.7)

where $\Phi = X - \mu$ and $\Phi_w = \sum_{i=1}^{k} \omega_k u_k$. The error parameter $\varepsilon$ can be used to solve a classification problem which involves two different classes of image objects.
Appendix B

Non-negative Matrix Factorisation (NMF)

B.1 The NMF problem formulation

Given a positive matrix $V \in \mathbb{R}^{N \times M}$ and a desired rank $p$. The goal is to find nonnegative matrix factors $W$ and $H$ such that

$$V \approx WH \quad (B.1)$$

where $W \in \mathbb{R}^{N \times p}$, $H \in \mathbb{R}^{p \times M}$ and $p < \min(N, M)$.

B.2 The NMF algorithm

There are many solutions to the NMF problem defined in equation (B.1) as a result of different cost functions been used to measure the difference between $V$ and the product $WH$. Here we focused on two optimisation problems and their possible solutions as suggested by Lee and Seung [2001]. To find a suitable approximation factorisation $V \approx WH$, Lee and Seung [1999] suggest two cost functions which took into consideration the distance between the two nonnegative matrices $V$ and $WH$. One of these cost function is the square of the Euclidean distance between the matrices $V$ and $WH$. The cost function is defined as:

$$\min_{W,H} f(W, H) = ||V - (WH)||^2 = \sum_{i=1}^{N} \sum_{j=1}^{M} (V_{ij} - (WH)_{ij})^2 \quad (B.2)$$
where $W \geq 0$ and $H \geq 0$

Another useful cost function referred to here as divergence is defined as:

$$\min_{W,H} f(W, H) = D(V || (WH)) = \sum_{i=1}^{N} \sum_{j=1}^{M} \left( V_{ij} \log \frac{V_{ij}}{(WH)_{ij}} - V_{ij} + (WH)_{ij} \right)$$ \hfill (B.3)

where $W \geq 0, H \geq 0$ and $D(V || (WH))$ can be interpreted as the divergence of $V$ from the product $WH$.

The functions in equations ?? and ?? are not convex in both $W$ and $H$ together except only when one of them is fixed, hence it is difficult to solve them. Although a possible solution to these problems is the use of gradient descent (Lee and Seung [2001]), however this method is slow to converged. Another solution that is fast to converged to at least a local minima is known as conjugate gradient. However, the method is very difficult to implement when compared with the implementation of gradient descent. Lee and Seung [1999] suggest the use of multiplicative update rules to solve the problems in equations ?? and ??.

The multiplicative update rules that solved the problem in equation ?? are defined as:

$$H_{kj} \leftarrow \frac{H_{kj}(W^TV)_{kj}}{(W^TH)_{kj}} \quad W_{ik} \leftarrow \frac{W_{ik}(VH^T)_{ik}}{(WHH^T)_{ik}}$$ \hfill (B.4)

It is important to state here that the update process are done on an element-by-element basis and not matrix multiplication. The multiplicative rules used to solved equation ?? are defined as:

$$H_{kj} \leftarrow H_{kj} \frac{\sum_i W_{ik}V_{ij}/(WH)_{ij}}{\sum_u W_{uk}} \quad W_{ik} \leftarrow W_{ik} \frac{\sum_j H_{kj}V_{ij}/(WH)_{ij}}{\sum_h H_{hv}}$$ \hfill (B.5)

For proof of convergence by both multiplicative update rules refer to Lee and Seung [2001]. Boutsidis and Gallopoulos [2008], suggested non-random methods for the initialisation of the nonnegative factors $W$ and $H$ referred to as NNDSVD. The NNDSVD method which is an extension of another non-random initialisation method called SVD-based, is said to outperformed the iterative multiplicative methods in some cases (Wild et al. [2004]). The NNDSVD initialisation method utilises singular value decomposition of $V$. The NNDSVD method start by first creating $k$ group of clusters based on the columns of $V$ and thereafter, initialise the factors $W$ and $H$ by the positive left and right singular vectors corresponding to the highest singular values of for each group. Suppose $V \in \mathbb{R}^{N,M}$ with a desired positive rank $r \leq (N, M)$, SVD expresses $V$ as sum of its $r$ highest singular factors as defined by:

$$V = \sum_{j=1}^{r} \sigma_j a_j b_j^T$$ \hfill (B.6)

where $\sigma_1 \geq \cdots \geq \sigma_r > 0$ represents the nonzero singular values of $V$ and $\{a_j, b_j\}_{j=1}^{r}$ represents the respective left and right singular vectors. Thereafter, for all $k \leq r$ the optimal rank-k
approximation of \( V \) with reference to Frobenius norm is defined as:

\[
V^{(k)} := \sum_{j=1}^{r} \sigma_j C^{(j)} = \arg \min_{\text{rank}(G) \leq k} \| V - G \|_F
\]  

(B.7)

where \( C^{(j)} = a_j b_j^T \) and \( G \) the approximation of \( V \) for each \( k \). The extension of the SVD-based method in NNDSVD is as a result of the additional approximation of every matrix \( C^{(j)} \) to its corresponding nonnegative section represented as \( C^{(j)}_+ \). Thereafter, the matrix factors \( W \) and \( H \) are initialised by choosing singular triplets of \( C^{(j)}_+ \). The NNDSVD initialisation steps can be summarised as:

1. compute the \( k \) leading singular triplets \((\sigma, a, b)\) of \( V \)
2. form the unit ranks matrices \( \{C^{(j)}\}_{j=1}^{k} \) obtained from the vector pairs \((a, b)\)
3. extract their positive section \( C^{(j)}_+ \) and corresponding singular triplet information
4. use the triplet information to initialise the factors \( W \) and \( H \)

The NNDSVD initialisation method converges faster with rapid decrease in the approximation error. This method of initialisation also returns the approximation matrix as well as the initialised nonnegative factors \( w \) and \( H \). We utilised this approach for the initialisation of \( W \) and \( H \) in this work.
Appendix C

Kernel Principal Component Analysis (KPCA)

C.1 The KPCA procedure

Suppose we have a dataset \( x = [x_1, x_2, x_3, \ldots, x_M] \in \mathbb{R}^N \). The kernel function \( \Phi \) maps the data \( \Phi(x) \) into a k-dimensional feature space, where \( k > N \). The kernel function \( \Phi \) creates nonlinear combinations of the data \( x \) by computing the dot product between the vectors.

Therefore, the projected high dimensional feature space with a zero mean as defined by Wang [2012] is:

\[
\frac{1}{M} \sum_{i=1}^{M} \Phi(x_i) = 0 \tag{C.1}
\]

The covariance matrix for the mapped feature space is defined as

\[
C = \frac{1}{M} \sum_{i=1}^{M} \Phi(x_i)\Phi(x_i)^T \tag{C.2}
\]

where \( C \) is \( M \times M \) matrix. To compute the eigenvalues and its corresponding eigenvectors we use

\[
C v_k = \lambda_k v_k \tag{C.3}
\]

where \( k = 1, 2, \ldots, M \)

Substitute \( C \) in equation C.2 into equation C.3

\[
\frac{1}{M} \sum_{i=1}^{M} \Phi(x_i)\Phi(x_i)^T v_k = \lambda_k v_k \tag{C.4}
\]
solve equation C.4 for \( v_k \) (Wang [2012]) we have:

\[
v_k = \sum_{i=1}^{M} a_{ki} \Phi(x_i)
\]  

(C.5)

Substitute \( v_k \) in equation C.5 into equation C.4, we have

\[
\frac{1}{M} \sum_{i=1}^{M} \Phi(x_i) \Phi(x_i)^T \sum_{j=1}^{M} a_{kj} \Phi(x_j) = \lambda_k \sum_{i=1}^{M} a_{ki} \Phi(x_i)
\]  

(C.6)

The kernel method which is the dot product of two vectors or data points is defined as

\[
k(x_i, x_j) = \Phi(x_i)^T \Phi(x_j)
\]  

(C.7)

Multiply both sides of equation C.6 by \( \Phi(x_l)^T \), we have

\[
\frac{1}{M} \sum_{i=1}^{M} k(x_i, x_l) \sum_{j=1}^{M} a_{kj} k(x_i, x_j) = \lambda_k \sum_{i=1}^{M} a_{ki} k(x_i, x_l)
\]  

(C.8)

By expressing equation C.8 using matrix notation we have

\[
K^2 a_k = \lambda_k M K a_k
\]  

(C.9)

where \( k_{i,j} = k(x_i, x_j) \) and \( a_k = [a_{k1}, a_{k2}, \ldots, a_{kM}]^T \), a vector with M-dimensions. Equation C.9 can be further reduced to

\[
K a_k = \lambda_k M a_k
\]  

(C.10)

Solve equation C.10 to obtain \( a_k \).

The resulting kernel principal components can be obtained using

\[
y_k(x) = \Phi(x)^T v_k = \sum_{i=1}^{M} a_k k(x, x_i)
\]  

(C.11)

For most classification problems, the inputs to the machine learning algorithms are the kernel principal components. It is important to note that, if the projected dataset \( \Phi(x_i) \) does not have zero mean, the Gram matrix \( \tilde{K} \) is used for the computation of the kernel principal components instead of the kernel matrix \( k \) (Bishop [2006], Wang [2012]). The Gram matrix \( \tilde{K} \) is defined as:

\[
\tilde{K} = k - 1_M k - k 1_M + 1_M k 1_M
\]  

(C.12)

Where \( 1_M \) is the matrix of size \( M \times M \) with all elements equal to \( \frac{1}{M} \) and \( M \) equals to the total number of samples.

A unique property of the kernel method is that, the mapping of the original dataset \( x_i \) to the high dimensional feature space \( \Phi(x_i) \) is not explicitly calculated. The kernel matrix \( k \) derive directly from the training dataset and its corresponding eigenvectors and eigenvalues
are obtained from the kernel matrix (Weinberger et al. [2004]). Thereafter, the kernel principal components $y_k(x)$ is calculated. Three popular kernel methods are

1. The polynomial kernel defined as:

$$k(x, y) = (x^T y)^d$$  \hspace{1cm} (C.13)

or

$$k(x, y) = (x^T y + c)^d$$  \hspace{1cm} (C.14)

where the constant $c$ is greater than zero, $d$ denotes the order or degree of the polynomial, and $x$ and $y$ denotes two feature vectors.

2. The Gaussian kernel defines as:

$$k(x, y) = \exp\left(-\frac{||x - y||^2}{2\sigma^2}\right)$$  \hspace{1cm} (C.15)

where $\sigma$ is a free parameter which can be optimised. The Gaussian kernel is a function that compute the distance $||x - y||$ between two vectors $x$ and $y$. To select the optimal value for $\sigma$, equation C.16 was suggested by Wang [2012]:

$$\sigma = 5 \times \text{mean}(d_{MM})$$  \hspace{1cm} (C.16)

Where $\langle d_{MM} \rangle$ is defined as the distance between a data point $x_i$ and its nearest neighbours. The method allows the value of $\sigma$ to large in order to capture neighbouring data points and ensures that it is smaller than the inter-class distances.

3. The hyperbolic tangent defined as:

$$k(x, y) = \tanh(\rho x y + \rho)$$  \hspace{1cm} (C.17)

Where $\rho$ is a constant, $x$ and $y$ denotes two vectors in the feature space.

Unlike the standard PCA, it is difficult to reconstruct or estimate the original data points $x_i$ from the kernel principal components. However, we can estimate the projections $\Phi(x)$ using

$$\Pi_m \Phi(x) = \sum_{i=1}^M y_k(x) v_k$$  \hspace{1cm} (C.18)

Where $\pi_m$ denotes a projection operator and $v_k$ is the eigenvectors of the covariance matrix (C) as defined in equation (C.2). If $m$ is very large, then $\Pi_m \Phi(x)$ is an approximation of the mapped points $\Phi(x)$. Since finding an estimate of $x$ is hard, we can obtain an approximation referred to as $z$, such that

$$\Phi(z) \approx \Pi_m \Phi(x)$$  \hspace{1cm} (C.19)
The equation (C.19) can be estimated by minimising equation (C.20)

\[ p(z) = \| \Phi(z) - \Pi_m \Phi(x) \| \]  \hspace{1cm} (C.20)

Mika et al. [1999b] defined a technique for computing the estimate \( z \) based on the kernel function used. For a Gaussian (\text{RBF}) kernel as defined in equation (C.15), the value of \( z \) is obtained using

\[ z = \frac{\sum_{i=1}^{M} \gamma_i \exp\left(-\frac{||z-x_i||^2}{2\sigma^2}\right)x_i}{\sum_{i=1}^{M} \gamma_i \exp\left(-\frac{||z-x_i||^2}{2\sigma^2}\right)} \]  \hspace{1cm} (C.21)

Where \( \gamma_i \) is defined as

\[ \gamma_i = \sum_{i=1}^{M} y_k(x)a_{ki} \]  \hspace{1cm} (C.22)

We can compute \( z \) iteratively using

\[ z_{t+1} = \frac{\sum_{i=1}^{M} \gamma_i \exp\left(-\frac{||z_t-x_i||^2}{2\sigma^2}\right)x_i}{\sum_{i=1}^{M} \gamma_i \exp\left(-\frac{||z_t-x_i||^2}{2\sigma^2}\right)} \]  \hspace{1cm} (C.23)

Wang [2012] suggest that the mean of the training data be used as the initial value for \( z \).

\[ z_0 = \text{mean}(x_i) \]  \hspace{1cm} (C.24)
Appendix D

Linear Discriminant Analysis (LDA)

D.1 The LDA problem formulation

Suppose the training set \( X = [x_1, x_2, \ldots, x_N] \in \mathbb{R}^D \) with corresponding class-labels \( Y = [y_1, y_2, \ldots, y_N] \), where \( N \) denotes the number of samples and \( D \) denotes the number of feature in the set. The aim is to transform the high-dimensional training data \( \{X\} \) into a new feature subspace with \( d \)-dimension, where \( d < D \).

The following LDA steps was used to perform the feature reduction

1. Compute the \( D \)-dimensional mean vectors \( (m_k) \) for each distinctive worm class

\[
m_k = \frac{1}{N_k} \sum_{n \in X}^{N_k} x_n \quad (D.1)
\]

Where \( k = [1, 2, \ldots, K] \), \( K \) denotes the total number of classes, \( N_k \) denotes the number of samples for each class in the training data and \( x_n \) denotes the data points in class \( k \).

2. Compute the within class covariance matrix \( (S_W) \), which is defined as:

\[
S_W = \sum_{k=1}^{K} S_k \quad (D.2)
\]

Where \( S_k \) is given as

\[
S_k = \sum_{n \in X} (x_n - m_k)(x_n - m_k)^T \quad (D.3)
\]

Each \( S_k \) denote the scatter matrix for that class and each \( m_k \) is the same as defined in equation \( (D.1) \). The between-class covariance matrix \( (S_B) \) is defined as

\[
S_B = \sum_{k=1}^{K} N_k (m_k - \bar{m})(m_k - \bar{m})^T \quad (D.4)
\]
Where $\tilde{m}$ is obtained using

$$
\tilde{m} = \frac{1}{N} \sum_{i=1}^{N} x_i
$$

(D.5)

Where $\tilde{m}$ denotes the global mean for the complete training data $(X)$, $x_i$ represent each point in the training data, $N_k$ denote the total number of observations in each class and $m_k$ denotes the mean vector for each class.

3. Solve the generalised eigenvalue problem for the weight matrix $(W)$ using

$$
W = \max_d \left( \text{eig}(S_W^{-1} S_B) \right)
$$

(D.6)

4. Select the desired number of linear discriminants $(d)$ for the new feature subspace after sorting the eigenvectors in descending order of the eigenvalues. The number of discriminants available is at most $K - 1$. This restriction on the dimension retained, is one of the main difference between the LDA and the standard PCA technique.

5. Compute the projected new feature subspace referred to as $X_{train}$ using

$$
X_{train} = XW^T
$$

(D.7)

Where $X_{train} \in \mathbb{R}^{N \times d}$. Similarly, the new test features can be represented as $X_{test} \in \mathbb{R}^{N \times d}$.
Appendix E

Kernel Linear Discriminant Analysis (KLDA)

Suppose we have a dataset $X \in \mathbb{R}^{N \times D}$ with $k$ classes. The number of samples in each class is denoted $N_i$ and the total number of samples $N$ in the set is defined as

$$N = \sum_{i=1}^{k} N_i \quad (E.1)$$

The LDA algorithm search for an optimal projection vector $w \in \mathbb{R}^d$ that maximises the function

$$J(\omega) = \frac{\omega^T S_b \omega}{\omega^T S_w \omega} \quad (E.2)$$

where $S_w$ is the within-class scatter matrix and $S_b$ is the between-class scatter matrix. Since, the LDA algorithm is a linear method, it makes separating the nonlinear separable data extremely challenging. Through the implementation of the kernel trick, the KLDA algorithm was developed to identify nonlinear separable data. Two widely used Mercer kernel used to transform the input data into the high-dimensional feature space $F$ are

$$K(x, y) = \exp\left(-\frac{||x - y||^2}{2\sigma^2}\right) \quad (E.3)$$

and

$$K(x, y) = (x^T y + 1)^d \quad (E.4)$$

where $\sigma$ (the width of the kernel) and $d$ are positive parameters for the Gaussian Radial basis function (RBF) and polynomial kernel respectively.

To accomplish this, the low-dimensional input data $X$ is mapped $\Phi(X)$ into a higher dimension feature space $F$ using a kernel function. Therefore, to find the projection vector $\omega$ that
maximises the $J(\omega)$ in the high dimensional feature space $F$, we rewrite equation (E.2) as

$$J(\omega) = \frac{\omega^T S_b^\Phi \omega}{\omega^T S_w^\Phi \omega}$$ \hspace{1cm} (E.5)

where $S_b^\Phi$ and $S_w^\Phi$ denotes the between-class and within-class scatter matrices in the mapped space $F$ respectively. The scatter matrices $S_b^\Phi$ and $S_w^\Phi$ in the mapped $F$ space are defined as:

$$S_b^\Phi = \frac{1}{N} \sum_{j=1}^{k} N_j (m_j^\Phi - m^\Phi)(m_j^\Phi - m^\Phi)^T,$$ \hspace{1cm} (E.6)

and

$$S_w^\Phi = \frac{1}{N} \sum_{j=1}^{k} \sum_{i=1}^{N_j} (\Phi(x_i^j) - m_j^\Phi)(\Phi(x_i^j) - m_j^\Phi)^T,$$ \hspace{1cm} (E.7)

where $m^\Phi$ denotes the global mean for all points in the mapped feature space $F$ and $m_j^\Phi$ denotes the mean for each class in the dataset. The global mean $m^\Phi$ and class mean $m_j^\Phi$ are defined as

$$m^\Phi = \frac{1}{N} \sum_{i=1}^{N} \Phi(x_i)$$ \hspace{1cm} (E.8)

and

$$m_j^\Phi = \frac{1}{N} \sum_{i=1}^{N_j} \Phi(x_i^j)$$ \hspace{1cm} (E.9)

Since is difficult to extract the best discriminant vector directly as a result of the high dimension feature space $F$, a dot product is applied in the $F$ space instead. This is represented as:

$$K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$$ \hspace{1cm} (E.10)

Based on the theory of reproducing kernel, the projection vector can be obtained from the $F$ space involving all mapped data points, which is defined as

$$\omega = \sum_{i=1}^{N} \alpha_i \Phi(x_i)$$ \hspace{1cm} (E.11)

By multiplying equation (E.11) by $m_j^\Phi$ in equation (E.9)

$$\omega^T m_j^\Phi = \frac{1}{N_j} \sum_{k=1}^{N_j} \sum_{i=1}^{N} \alpha_i \langle \Phi(x_k^i), \Phi(x_i) \rangle$$ \hspace{1cm} (E.12)

$$\omega^T m_j^\Phi = \alpha^T M_j$$ \hspace{1cm} (E.13)
Appendix E Kernel Linear Discriminant Analysis (KLDA)

where

\[(M_j)_i = \frac{1}{N_j} \sum_{k=1}^{N_j} K(x_k^j, x_i) \quad (E.14)\]

We can rewrite the numerator of equation (E.5) using equation (E.13) as

\[\omega^T S^\Phi \omega = \alpha^T M \alpha \quad (E.15)\]

where \(M\) is given as

\[M = \sum_{j=1}^{k} N_j(M_j - M_t)(M_j - M_t)^T \quad (E.16)\]

where \(M_j\) is same as equation (E.14) and \(M_t\) is the overall mean in the kernel space, which is given as

\[(M_t)_k = \frac{1}{N} \sum_{i=1}^{N} K(x_k, x_i) \quad (E.17)\]

Similarly, we can combine the denominator of equation (E.5) with equations (E.9) and (E.11) gives

\[\omega^T S^\omega \omega = \alpha^T S \alpha \quad (E.18)\]

where

\[S = \sum_{j=1}^{k} K_j(I - 1_{N_j})K_j^T \quad (E.19)\]

We can then rewrite equation (E.5) as

\[J(\alpha) = \frac{\alpha^T M \alpha}{\alpha^T S \alpha} \quad (E.20)\]

Equation (E.20) can be express as the generalised eigendecomposition problem given as

\[M \alpha = \lambda S \alpha \quad (E.21)\]

\[S^{-1} M \alpha = \lambda \alpha \quad (E.22)\]

In most cases, the \(S\) is a singular matrix and cannot be applied directly except it is regularised. The regularisation is represented as

\[S_\mu = S + \mu I \quad (E.23)\]

where \(\mu\) is a small positive number representing the regularisation parameter and \(I\) is the identity matrix. We can rewrite equation (E.22) as:

\[(S + \mu I)^{-1} M \alpha = \lambda \alpha_i \quad (E.24)\]

We can extract the required discriminant features from equation (E.24) by computing and selecting the \(d\) eigenvectors \(\alpha_i\) with the highest eigenvalues \(\lambda_i\). The projection of a test sample
$x$ onto $\omega$ in the feature space $F$ is defined as

$$<\omega, \Phi(x)> = \sum_{j=1}^{N} \alpha_j K(x_j, x)$$  \hfill (E.25)

For a binary classification problem, equation (E.16) becomes

$$M = (M_1 - M_2)(M_1 - M_2)^T$$  \hfill (E.26)

where $M_1$ and $M_2$ are the class means in the transformed feature space. Similarly equation (E.19) becomes

$$S = \sum_{j=1}^{2} K_j (I - 1_{N_j}) K_j^T$$  \hfill (E.27)

The KLDA steps can be summarised as

1. Compute the kernel matrix $K$ of size $N \times N$ using any of the kernel functions in equations (E.3) and (E.4)
2. Compute the between-class scatter matrix $M$ of size $N \times N$ in the kernel feature space using equation (E.16)
3. Compute the within-class scatter matrix $S$ of size $N \times N$ in the kernel feature space using equation (E.19)
4. Extract the $d$ eigenvalues ($\lambda_i$) and the eigenvector($\alpha_i$) using equation (E.22). However, if the matrix $S$ is singular (matrix with zero determinant) use equation (E.24)
5. Obtain the projection of a new point $x$ onto $\omega$ in the kernel space using equation (E.25)
Appendix F

Food leaving event

F.1 Introduction

To quantify and characterise C. elegans locomotory behaviour requires studying the movement patterns of the worm and its body postures over a period of time. In this chapter, we analyse the movement behaviour of two distinct worms types (wild-type and mutant type) in the presence of food using low-resolution microscopic videos of the worms. The wild-type worm in this case N2 are worms with functional nervous system without any defects in their genes. On the other hand, the mutant worm in this case mgl2 are worms that have defects in their genes either during the development stages or as a result of genes knockouts in the laboratory. The Institute for Life Science (IFLS), University of Southampton provided the videos used in this work. The videos are 5 minutes long at the rate of 10 frames per seconds. The videos contain multiple worms of the same genotypes freely crawling in an agar food plate. The goal is to quantify and characterise the foraging behaviours of these two distinct worms using an automated system as opposed to the manual system currently in use in most laboratories.

F.2 Extraction and Preprocessing of Video frames

The first stage in the automatic detection and tracking system of the worm movement is the retrieval and analysing of the video frames sequentially. In this work, our customised software written in Python combined with the OpenCV built-in functions was used for the extraction of video frames in the movies. In the process of extracting the video frames, we applied image processing operations on each video frame in order to get rid of noise or any unwanted features. The following steps was taken to clean each video frame before the detection and tracking of the worms:

1. Extract and convert the coloured video frames into its grayscale equivalent image frames
2. Apply a gaussian filter on the grayscale images to remove the noise.

3. Transformation of the clean grayscale images into its binary equivalent (black and white) using a thresholding operations (Otsu’s method). The reason for this conversion is to separate the foreground objects (worms) from the video background.

F.3 Quantification of C. elegans movement behaviour

Here we describe the steps taken to characterise the behaviour of the worms in a given video based on their locomotion behaviour on a food lawn. Worms in a given video were detected and tracked over time as each worm crawl and feed on a food patch. The following steps was taken to detect, track and quantify the worms movement patterns:

1. To detect the worms in any frame, the Canny edge detection algorithm was applied to identify the shapes of the worms and the OpenCV find contours function was used to retrieve the positions of the worms. Information about each worm position between successive videos frames are stored and used for further analysis.

2. To track a worm, we calculate the distance travelled by the worm between two successive image frames using the stored coordinates. We applied Euclidean distance between two points to compute the distance moved by a given worm between successive frames. This distance is used to track and identify the worm position in subsequent frames. The shorter the distance, means the worm in the current frame is the same worm in the previous frame.

3. To quantify the nematode worm food behaviour, we marked a circle on the food patch and use it as the rationale for characterising a worm behaviour. For a given frame, if a worm is within or at the marked circular boundary on the food patch, we count it as inside event. Similarly, if the worm has completely cross the marked line, it is counted as outside or food leaving event. For the purpose of investigating the time spend by the worms at the food boundary, we choose to compute the boundary count separately. The automated system has the capability to detect and retrieve the parameters of the marked circle. This was achieved using Hough transform function for image circle detection. Let \((x_0, y_0)\) denotes the centre of the circle with radius \((r)\). Equation F.1 was used to compute the distance between a worm position \((x_c, y_c)\) from the circle centre. Therefore, If this distance \((d)\) is less than or equal to the radius \((r)\), that means the worm is within the food plate and is counted as not leaving event. Otherwise, it is counted as food leaving event. As mentioned earlier, we decided to compute the time spend at the boundary separately although it can be described as not a leaving event.

\[
d = \sqrt{(x_c - x_0)^2 + (y_c - y_0)^2}
\]  (F.1)
Furthermore, equation F.2 can be used to measure the total distance $d_t$ covered by a given worm in a video.

$$d_t = \sum_{i=1}^{n} \sqrt{\left(x_{i+1} - x_i\right)^2 + \left(y_{i+1} - y_i\right)^2}$$  \hspace{1cm} (F.2)

Here $n = k - 1$ where $k$ denotes the total number of frames in the movie, $(x_i, y_i)$ and $(x_{i+1}, y_{i+1})$ represent the previous and current coordinates of the worm between two successive frames respectively. Finally, the instantaneous speed for a given worm is defined as the ratio of the distance covered and the snapped time $\delta$ in seconds. The snapped time can be described as the time interval between the creation of two successive frames. The speed of a worm is calculated using equation F.3.

$$\text{Speed} = \frac{\sqrt{\left(x_t - x_{t-\delta}\right)^2 + \left(y_t - y_{t-\delta}\right)^2}}{\delta}$$  \hspace{1cm} (F.3)

Where $(x_{t-\delta}, y_{t-\delta})$ and $(x_t, y_t)$ are the coordinates of the worm position in the previous and current frames respectively.

### F.4 Classification results

The result in Table F.1 shows that, the N2 worms cross the food boundary more when compared with the mgl2 worms. The result also shows that the N2 worms covered more area compared to their mgl2 counterpart. We may infer that, the wild type worms tries to explore other opportunities around whereas, the mutant lacks that desire.

<table>
<thead>
<tr>
<th>Events</th>
<th>Wild-type (N2)</th>
<th>Mutant type (mgl2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of frames</td>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td>Leaving count</td>
<td>294</td>
<td>112</td>
</tr>
<tr>
<td>Boundary count</td>
<td>548</td>
<td>457</td>
</tr>
<tr>
<td>Inside count</td>
<td>2487</td>
<td>2536</td>
</tr>
<tr>
<td>Area under the curve</td>
<td>293.00</td>
<td>122.00</td>
</tr>
<tr>
<td>Average area under the curve</td>
<td>0.586</td>
<td>0.224</td>
</tr>
</tbody>
</table>

Table F.1: Food leaving event between the N2 and mgl2 worms

Figures F.1 shows the stages involve in quantifying and characterising food leaving event. Figure F.2 shows the tracked paths of multiple worms in a movie.

### F.5 Conclusion and Future work

The built automated detection and tracking system has shown that, the movement behaviour of two distinct *C. elegans* strains can be quantify and characterised when confined in given environment or in the presence of food. The implemented tracking system was able to provide
quantitative data about the food leaving behaviour of the wild type and mutant strains. The result shows that the food leaving event of the N2 (wild type) strain is significantly high compared to the *mgl2* (mutant) strain. A major challenge encountered while tracking and quantifying the worms behaviour is that, it is hard to keep track of the worms when two or more worms crosses each other. For this reason, we ignored frames where two or more worms cross each other. One possible solution to this problem, is ignore frames where worms are in occlusion and maintain the previously identified coordinates of the worms and use that information to track them after separation.
Appendix F Food leaving event

Figure F.2: Tracked path of the worms
Appendix G

Grid-search

G.1 PCA gridsearch

Figure G.1: PCA Grid-search plots
Appendix G Grid-search

Figure G.2: PCA Grid-search plots
Appendix G Grid-search

Figure G.3: PCA Grid-search plots

(a) RF train (Linear)  
(b) RF train (Spline)  
(c) RF test (Linear)  
(d) RF test (Spline)
G.2 NMF grid-search

(a) SVM train (Linear)

(b) SVM train (Spline)

(c) SVM test (Linear)

(d) SVM test (Spline)

Figure G.4: NMF Grid-search plots
Appendix G Grid-search

Figure G.5: NMF Grid-search plots

(a) kNN train (Linear)  
(b) kNN train (Spline)

(c) kNN test (Linear)  
(d) kNN test (Spline)
Figure G.6: NMF Grid-search plots
G.3 KPCA grid-search

Figure G.7: KPCA Grid-search plots
**G.4 KPCA grid-search**

![KPCA Grid-search plots](image)

(a) $k$NN train (Linear)  
(b) $k$NN train (Spline)  
(c) $k$NN test (Linear)  
(d) $k$NN test (Spline)

Figure G.8: KPCA Grid-search plots
Appendix G Grid-search

G.5 KPCA grid-search

Figure G.9: KPCA Grid-search plots
G.6 LDA grid-search

Figure G.10: LDA Grid-search plots
Figure G.11: LDA Grid-search plots2
Figure G.12: LDA Grid-search plots

(a) RF train (Linear)  
(b) RF train (Spline)

(c) RF test (Linear)  
(d) RF test (Spline)
G.7 KLDA Gridsearch

Figure G.13: KLDA Grid-search plots
Appendix G Grid-search

Figure G.14: KLDA Grid-search plots
Figure G.15: KLDA Grid-search plots
Appendix H

3D plots of the low-dimensional skeleton angles embeddings

H.1 NMF 3D plots

Figure H.1: NMF 3D training and test embeddings of the high-dimensional skeleton angles data
H.2 KPCA 3D plots

Figure H.2: KPCA 3D training and test embeddings of the high-dimensional skeleton angles data
H.3 LDA 3D plots

Figure H.3: LDA 3D training and test embeddings of the high-dimensional skeleton angles data
Appendix H 3D plots of the low-dimensional skeleton angles embeddings

H.4 KLDA 3D plots

(a) 3D train (Linear)  
(b) 3D train (Spline)  
(c) 3D test (Linear)  
(d) 3D test (Spline)

Figure H.4: KLDA 3D training and test embeddings of the high-dimensional skeleton angles data
Appendix I

Low-dimensional embeddings projections per worm type

I.1 PCA projections

Figure I.1: PCA training projections per worm class, in which linear interpolation was used to compute the missing frame values.
Figure I.2: PCA training projections per worm class, in which quadratic spline interpolation was used to compute the missing frame values.
Appendix I: Low-dimensional embeddings projections per worm type

Figure I.3: PCA test projections per worm class, in which linear interpolation was used to compute the missing values.
Figure I.4: PCA test projections per worm class, in which quadratic spline interpolation was used to compute the missing values.
Appendix I Low-dimensional embeddings projections per worm type

### I.2 NMF projections

(a) AQ947 class (Linear)

(b) OW939 class (Linear)

(c) OW940 class (Linear)

(d) OW949 class (Linear)

(e) OW953 class (Linear)

(f) OW956 class (Linear)

Figure I.5: NMF training projections per worm class, with linearly imputed missing frames values
Appendix I Low-dimensional embeddings projections per worm type

Figure I.6: NMF training projections per worm class, with quadratic spline imputed missing frames values
Appendix I Low-dimensional embeddings projections per worm type

Figure I.7: NMF test projections per worm class, with linearly imputed missing frames values
Appendix I Low-dimensional embeddings projections per worm type

Figure I.8: NMF test projections per worm class, with quadratic spline imputed missing frame values
I.3 KPCA projections

Figure I.9: KPCA training projections per worm class, in which quadratic linear interpolation was used to compute the missing values
Appendix I Low-dimensional embeddings projections per worm type

Figure I.10: KPCA training projections per worm class, in which quadratic spline interpolation was used to compute the missing values
Appendix I Low-dimensional embeddings projections per worm type

Figure I.11: KPCA test projections per worm class, in which linear interpolation was used to compute the missing values.
Appendix I Low-dimensional embeddings projections per worm type

Figure I.12: KPCA test projections per worm class, in which quadratic spline interpolation was used to compute the missing values.
Appendix I Low-dimensional embeddings projections per worm type

I.4 LDA Projections

Figure I.13: LDA training projections per worm class, in which linear interpolation was used to compute the missing values.
Appendix I Low-dimensional embeddings projections per worm type

Figure I.14: LDA training projections per worm class, in which quadratic spline interpolation was used to compute the missing values
Figure I.15: LDA test projections per worm class, in which linear interpolation was used to compute the missing values
Appendix I Low-dimensional embeddings projections per worm type

Figure I.16: LDA test projections per worm class, in which quadratic spline interpolation was used to compute the missing values
I.5 KLDA projections

Figure I.17: KLDA training projections per worm class, in which linear interpolation was used to compute the missing values
Appendix I: Low-dimensional embeddings projections per worm type

(a) AQ2947 class (Spline)

(b) OW939 class (Spline)

(c) OW940 class (Spline)

(d) OW949 class (Spline)

(e) OW953 class (Spline)

(f) OW956 class (Spline)

Figure I.18: KLDA training projections per worm class, in which quadratic spline interpolation was used to compute the missing values
Appendix I Low-dimensional embeddings projections per worm type

Figure I.19: KLDA test projections per worm class, in which linear interpolation was used to compute the missing values
Appendix I Low-dimensional embeddings projections per worm type

219

(a) AQ2947 class (Spline)  (b) OW939 class (Spline)

(c) OW940 class (Spline)  (d) OW949 class (Spline)

(e) OW953 class (Spline)  (f) OW956 class (Spline)

Figure I.20: KLDA test projections per worm class, in which quadratic spline interpolation was used to compute the missing values
Bibliography

E. Yemini. High-throughput, single-worm tracking and analysis in caenorhabditis elegans. 


Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. 2018.


