

Winter 2018

Data Visualization and Classification of Artificially Created Images

Dmytro Dovhalets

Central Washington University, dovhaletsd@gmail.com

Follow this and additional works at: <https://digitalcommons.cwu.edu/etd>



Part of the [Artificial Intelligence and Robotics Commons](#), and the [Numerical Analysis and Scientific Computing Commons](#)

Recommended Citation

Dovhalets, Dmytro, "Data Visualization and Classification of Artificially Created Images" (2018). *All Master's Theses*. 891.

<https://digitalcommons.cwu.edu/etd/891>

This Thesis is brought to you for free and open access by the Master's Theses at ScholarWorks@CWU. It has been accepted for inclusion in All Master's Theses by an authorized administrator of ScholarWorks@CWU. For more information, please contact scholarworks@cwu.edu.

DATA VISUALIZATION AND CLASSIFICATION
OF ARTIFICIALLY CREATED IMAGES

A Thesis
Presented to
The Graduate Faculty
Central Washington University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Computational Science

by
Dmytro Dovhalets
March 2018

CENTRAL WASHINGTON UNIVERSITY

Graduate Studies

We hereby approve the thesis of

Dmytro Dovhalets

Candidate for the degree of Master of Science

APPROVED FOR THE GRADUATE FACULTY

Dr. Răzvan Andonie

Dr. Boris Kovalerchuk

Dr. Szilárd Vajda

Dean of Graduate Studies

ABSTRACT

DATA VISUALIZATION AND CLASSIFICATION OF ARTIFICIALLY CREATED IMAGES

by

Dmytro Dovhalets

March 2018

Visualization of multidimensional data is a long-standing challenge in machine learning and knowledge discovery. A problem arises as soon as 4-dimensions are introduced since we live in a 3-dimensional world. There are methods out there which can visualize multidimensional data, but loss of information and clutter are still a problem. General Line Coordinates (GLC) can losslessly project n -dimensional data in 2-dimensions. A new method is introduced based on GLC called GLC-L. This new method can do interactive visualization, dimension reduction, and supervised learning. One of the applications of GLC-L is transformation of vector data into image data.

This novel approach of transforming vector data into images using lossless visualization introduces a new method for classification of data in vector format. Having images which are in raster format instead of vector format allows it to be classified with a Convolutional Neural Network (CNN). Experiments conducted on datasets of different sizes show that these artificially created images provide useful information for the CNN. The CNN can classify these artificially created images with competitive results to other analytic machine learning algorithms for vector data. The artificially created images were also classified with a Support Vector Machine (SVM) and a Multilayer Preceptron (MLP).

ACKNOWLEDGEMENTS

I am grateful to the School of Graduate Studies and Research for providing me with a graduate research assistantship for the 2016-2017 academic year. I am also grateful to the Computer Science Department and School of Graduate Studies and Research for providing me grants for travel related expenses to the BHI 2018 conference.

I would like to thank Dr. Vajda who provided me feedback and advice throughout my graduate career, arranging for me to attend and present at my first conference.

Thank you to Dr. Kovalerchuk who introduced me to n-dimensional visualization which became a big part of my thesis and guided me to co-author my first journal paper.

Thank you to Dr. Andonie who provided me with endless advice and pushed me outside my comfort zone, introducing me to the world of machine learning and artificial intelligence.

TABLE OF CONTENTS

Chapter		Page
I	INTRODUCTION	1
	General Line Coordinates	2
	Data Transform with GLC-L	3
	Motivation and Contribution	5
II	CONVOLUTIONAL NEURAL NETWORK	6
	Convolutional Layers	6
	Pooling Layers	8
	Dropout	9
	CNN Achievements	10
III	GENERAL LINE COORDINATES	11
	Comparison of GLC-L and Parallel Coordinates	11
	GLC-L	13
	GLC-AL	16
	GLC- IL	18
	GLC-DRL	19
	GLC-L Limitations	20
IV	DATA TRANSFORM	25
	GLC-L Data Transform	25
	Thickness of Line in Artificially Created Images	30
	Size of Artificially Created Images	31
	Random Search Algorithm	34
V	EXPERIMENTAL RESULTS	36
	CNN Experiments	37
	MLP Experiments	41
	SVM Experiments	43
	Comparison of CNN, MLP and SVM	45
VI	CONCLUSION	47

Chapter	Page
REFERENCES CITED	49
APPENDIX: DATASETS	52
Swiss Roll Dataset	52
Wisconsin Breast Cancer	54
Wine Quality	54
Diabetic Retinopathy Debrecen	55
MNIST-Subset	56

LIST OF TABLES

Table	Page
1 Line Thickness Experimental Results	30
2 Image Size Experimental Results	32
3 GLC-L with Random Search	34
4 CNN Result Comparison	39
5 Transform+CNN Runs	40
6 MLP Result Comparison	42
7 Transform+MLP Runs	42
8 SVM Result Comparison	44
9 Transform+SVM Runs	44
10 CNN, MLP and SVM Result Comparison	46
11 Overview of Datasets	52

LIST OF FIGURES

Figure	Page
1 Visualization of 10-D data with GLC-L	2
2 Examples of artificially created images	4
3 CNN architecture	6
4 Example of Pooling	8
5 Dropout example	9
6 Comparison of Parallel Coordinates and GLC-L	12
7 Visualization of 4-D data with GLC-L	14
8 Visualizing GLC-L	15
9 Interacting with GLC-L	19
10 Dimension reduction with GLC-DRL	20
11 1-D points from Wisconsin Breast Cancer, Class 1 and Class 2	22
12 1-D points from Wisconsin Breast Cancer, both classes together	22
13 1-D points from FERET Database, Class 1 and Class 2	23
14 1-D points from FERET Database, both classes together	23
15 Dataset split into 2 new sets	26
16 Diagram of the data transform system	27
17 World view coordinates	28
18 Projection line thickness	31
19 Examples of different image sizes	33
20 Swiss Roll Dataset 2-D	53
21 Swiss Roll Dataset 3-D	53
22 Breast Cancer Wisconsin dataset	54

LIST OF FIGURES (CONTINUED)

Figure		Page
23	Red Wine Quality dataset	55
24	Diabetic Retinopathy Debrecen dataset	56
25	Images from the MNIST-Subset	56

CHAPTER I

INTRODUCTION

The majority of people work with multidimensional data without even realizing it. Anyone who has ever used a spreadsheet to enter information has created multidimensional data. This multidimensional data may contain patterns and valuable information, but it is not easy to interpret it without extensive data analysis. It is difficult to extract general rules and information from an Excel sheet with dozens of columns and hundreds, if not thousands of rows. Data visualization is very powerful and is used by the masses daily, some examples include Cartesian coordinate systems, histograms, pie charts, etc. Those visualization tools are very useful and powerful when dealing with 2-D or 3-D data. However, when the data is of 4 or more dimensions those tools cannot be used. There are tools out there which can visualize data with 4 or more dimensions, but clutter and loss of information is still a challenge [1].

Linear General Line Coordinates (GLC-L) project multidimensional data onto 2-D graphs. Having the n-dimensional data projected on to 2-D graphs allows us to see the high dimensional data. Once the data are visualized they can be used to extract rules and information. However this can be a hard and tedious task due to the complexity of the data and abundant amount of information. Such tasks can be automated to better explore the visualization and the data itself. For classification of data, Dr. Kovalerchuk and I have implemented an interactive visualization method. This method applies machine learning to GLC-L in order to separate the data and visualize it [1].

GLC-L is further explored in its potential to transform vector data into images. This data transform allows vector data to be classified using a Convolutional Neural Network (CNN).

General Line Coordinates

General Line Coordinates (GLC) proposed by Kovalerchuk [2] can visualize multidimensional data losslessly. Lossless visualization presents the data without losing any information, and the visualization is reversible to the original data. Allowing users to extract rules and information from specific attributes within the dataset. One of the methods of GLC is GLC-L; it is a visualization algorithm for a linear function.

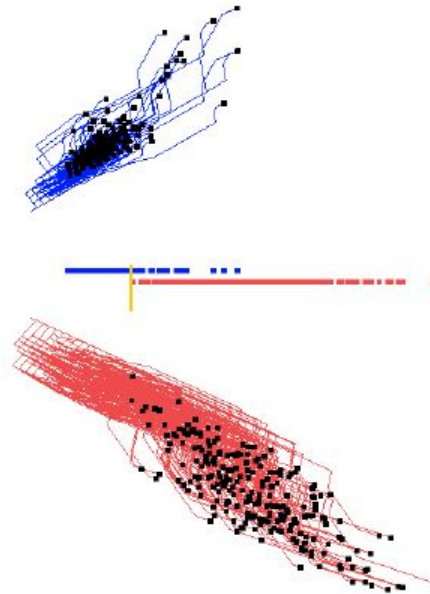


FIGURE 1: Visualization of Wisconsin Breast Cancer dataset, which is of 10 dimensions. Instances from one class are projected above in blue while instances from the other class are below in red. The visualization is optimized to separate the two classes with GLC-L.

Fig. 1 is visualizing a dataset which has over 600 instances and 10 attributes. It is essentially an Excel sheet with over 600 rows and 10 columns, where one of the columns specifies to which class that specific row belongs to. It would be extremely difficult and nearly impossible to see that the data can be separated between the two classes based on some rules within the dataset. Not only can GLC-L visualize this dataset, but it does it in a way which separates the data between the 2 classes.

GLC-L can be used as a preprocessing step to algorithm selection in machine learning. Since it visualizes a linear function, it separates data very well, which is linearly separable. Testing to see if the data is linearly separable can narrow down the number of algorithms to choose from. If GLC-L can separate the data, then other linear methods will be able to separate it as well. When the data can't be separated with GLC-L, it means that the data are not linearly separable, and it's best to choose a non linear method to solve the classification task. This provides information about the dataset and avoids guessing and checking different types of methods.

The visualization itself provides information about the dataset which is very difficult to spot in the raw format. With a quick glance it can be seen how the data clusters in the n-dimensional space. It makes it easy to see if instances from different classes have unique patterns.

The base GLC-L method was extended to be an interactive, machine learning, visualization tool. Supervised learning was added to optimize the separation of data between classes. Dimensionality reduction was implemented for a cleaner visualization and more optimized classification. Interactivity was incorporated for a more friendly visualization, which allowed the users to interact with the visualization for a deeper analysis of the data [1].

Data Transform with GLC-L

GLC-L can be used to transform vector data into images, visualizing one data instance at a time and saving the visualization as an image. These saved images which are artificially created by the visualization can later be used for extraction rules from the dataset. Instead of an Excel sheet with over 600 rows and 10 columns, images are created with labels corresponding to their class label. Looking at images is much easier than an

Excel sheet. Patterns of the line can be spotted by looking at samples from both classes and generalizing based on those patterns.

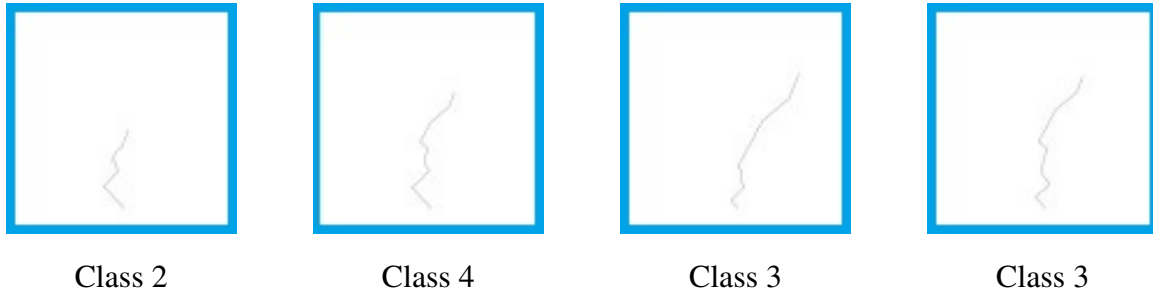


FIGURE 2: Examples of artificially created images using GLC-L. These examples are from the Wine Quality Dataset which contains 12 attributes. Labels below each figure correspond to the class label.

Fig. 2 has examples of numerical data transformed with GLC-L into images. The original numerical data has 12 attributes with one being the class label. Looking at the images in Figure 2, visual patterns can be seen. Images from Class 3 have different patterns in the line than images from Classes 2 and 4. It can also be seen that the line in the image from Class 2 is much shorter.

Visualizing only one data instance at a time eliminates clutter and provides a clean visualization. Images can be compared side by side for extraction of rules and information. This task can also be automated by putting the images through a CNN for supervised learning. The CNN model can be trained on these images for classification.

Other algorithms such as Support Vector Machines (SVM) [3] and Multilayer Perceptron (MLP) [4] can be trained for classification. SVM and MLP are capable of producing competitive results to CNN in image classification tasks [5], [6]. The artificially created images were put through all three algorithms for classification. Experimenting with multiple machine learning algorithms provides valuable information on how these systems handle artificially created images.

Motivation and Contribution

Visualizing vector data of high dimension and being able to see it in 2-D allows us to gain insight of the data and extract information. Extracting information is not an easy task due to the complexity of the data and an abundant amount of information. Being able to automate the task of extracting information from the visualizations has inspired us to use visualization as a data transformation, visualizing the entire dataset one instance at a time and capturing the visualizations as images, which can be classified by machine learning algorithms.

The goal of this study is to gain insight if the artificially created images from the visualization can be used for classification by machine learning algorithms. CNN is designed for natural images. These artificially created images of lines which are created by the visualization contain very few informational pixels. The majority of the pixels are white, representing the background. Having a sparse number of informational pixels raised many questions. Whether the CNN would be able to generalize was one of the main questions. Exploration of CNNs and artificial images is yet to be done.

Contribution of this study is filling the gap between n-dimensional data visualization and automated classification of the visualizations. Transforming data using a visualization method has yet to be done. This novel approach of creating images from the visualization allows vector data to be classified with a CNN. The following chapters contain explanations of the visualization method and the data transformation, along with the classification experiments and results of the transformed data.

CHAPTER II

CONVOLUTIONAL NEURAL NETWORK

Large datasets and advances in computing power have made breakthroughs in machine learning. Convolutional Neural Networks (CNN) [7], which are a category of Artificial Neural Networks, have gained popularity and reached performance levels similar to humans at image recognition tasks. CNNs have trainable parameters which learn to extract features from images and are able to recognize objects with very low error rates. The architecture of the CNN can be constructed in many different ways to best accommodate the task at hand. The architecture has an input layer, convolutional layers, pooling layers, followed by fully connected layers. There can be a few convolutional layers and pooling layers in the same CNN. The majority of the CNNs which produce best results have multiple convolutional and pooling layers in their architecture [8].

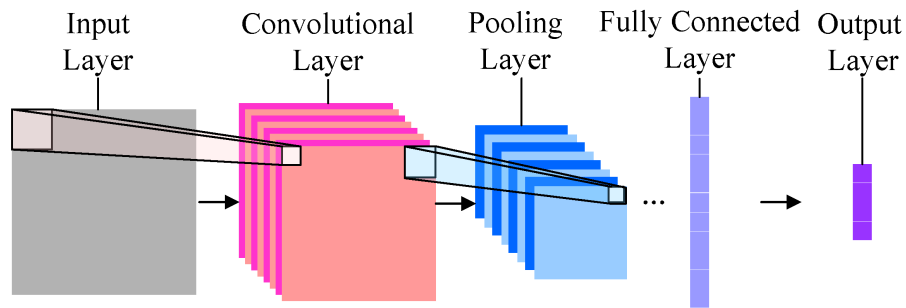


FIGURE 3: CNN architecture [9].

Convolutional Layers

2-D convolutions are designed for 2-D data format such as images. Using 2-D convolutions lets the network find patterns and relations within the data, using neighboring pixels to extract useful information. Other machine learning algorithms

such as a SVM [3] and MLP [4] get outperformed by a CNN in image recognition tasks because they do not use relational input.

Convolutional layers extract information from the 2-D input by passing a filter throughout the 2-D input. The filter size can vary from 2x2 to 11x11 depending on the architecture of the network and the data itself. The filter is passed through the image starting at the top left corner and moving along to the right bottom corner. The number of pixels the filter moves over is referred to as the *stride*. The stride is set to be greater than 1 but no larger than the pooling neighborhood [7]. A filter of size 2x2 and stride of 1 can be viewed as a box with the size of 2x2 which is being placed over the image and shifting over 1 pixel every time it's applied.

The number of filters which should be applied depends on the complexity of the data. Images with people in them would require more filters than images with digits in them. The filter produces feature maps which act as feature detectors. The more feature detectors the model has the more information it is able to extract. However, more filters does not directly mean it will perform better. There are other factors in the architecture of the network which play a huge role in the performance of the final model.

The number of convolutional layers in the network varies on the task at hand. Networks which have more than a few layers are considered deep networks. Deep networks have performed very well in image recognitions tasks. For face recognition, a network with 37 layers achieved an accuracy of 99.13% [8].

Pooling Layers

Pooling layers, also referred to as sub-sampling or down-sampling, reduce the feature maps as the input makes its way down the network. There are different types of pooling. It can be done by taking the sum, average, or max. Max pooling with filters 2x2 means for each of the 2x2 regions only the max value will be taken [10].

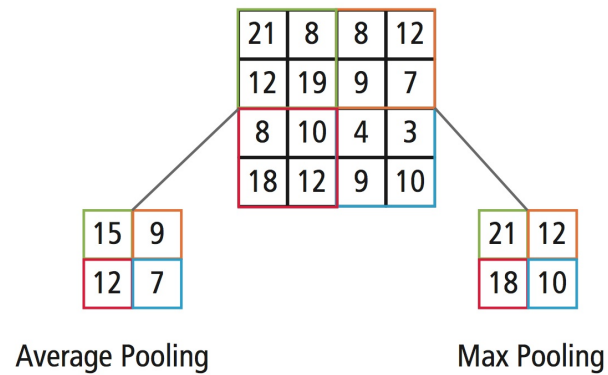


FIGURE 4: Example of how pooling is done in a CNN. Pooling can be done several ways, including by taking the average or max [11].

In Fig. 4, pooling is presented in two different ways, by taking the average or the max. In this example, 2x2 pooling is applied to a 4x4 input space. The input space is divided into 4 subparts, and for each of the subparts one value will be used for representation of the entire subpart. Average pooling is done by adding all of the numbers in the subpart and dividing it by the number of elements in the subpart. In the original 4x4, the input space of the top left subpart contains 21, 8, 12, and 19. Adding them up and dividing by 4 results in 15, which can be seen as the number in the left top corner in the 2x2 space after pooling. Max pooling is done by simply taking the largest value from the subpart, which is 12 for the top right subpart in Fig. 4. One of the advantages of pooling is reduction of hidden nodes as the network gets larger and deeper.

Dropout

Training CNN models with small datasets may lead to over-fitting. This happens when models perform well on the validation set during training but perform poorly on the testing set. The testing set is not seen by the model during training and is used to evaluate the model once it has been trained. Having the model perform well during training but poorly during testing is due to the model learning the training set instead of generalizing. To deal with over-fitting, a technique called dropout was introduced [12]. Dropout works by dropping (ignoring) random nodes of the network at each iteration. By dropping certain nodes at each iteration, the network is required to learn other features from the input data. Dropout prevents the nodes in the network from co-adapting and helps with generalizing [13]. Dropout implementation takes in a decimal value between 0 and 1, which represents the fraction of nodes which should be ignored during the iteration. Dropout with the value of 0.5 would be considered harsh, as at each iteration half of the input nodes would be ignored.

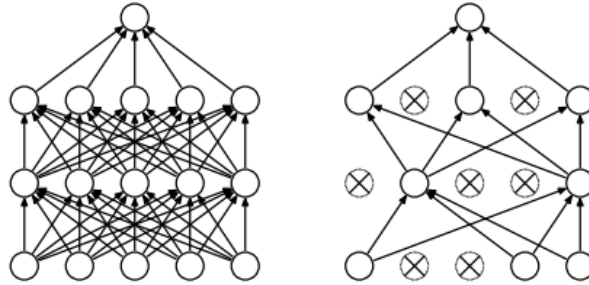


FIGURE 5: Dropout being applied to a neural network [12].

Dropout increases the number of epochs it takes to train a strong model. The trade off in longer training time to prevent over-fitting is sufficient. Training multiple models and using them together can also prevent over-fitting. However, training multiple models requires more time than using dropout during training for only one model [14].

CNN Achievements

CNNs outperform other machine learning algorithms in image recognition tasks. Comparison of the machine learning algorithms on gender classification using front facial images shows a CNN gets an accuracy of 96.1%, while the second highest accuracy (produced by a SVM) is 77.4% [6]. A CNN trained on a dataset of 1.2 million images produced an error rate of 17%, while the next best method got an error rate of 28.2% [14]. Exploration of CNNs has been a very popular topic. Combining multiple CNNs and having deeper networks has gained a lot of interest in the machine learning community. Making deeper networks has shown a significant improvement in classification accuracy for large image datasets [15].

CNNs have been around for some time now, but have only gained popularity after large image datasets became available and more computing power was introduced with the GPUs. Since then, CNNs have become a standard for image recognition tasks and widely adopted by the machine learning community.

CHAPTER III

GENERAL LINE COORDINATES

General Line Coordinates (GLC) visualize multidimensional data losslessly. There is no dimensionality reduction or loss of information during the process of visualization. The n-dimensional data is projected onto 2-D graphs without losing any information in the process. One of the methods from GLC is GLC-Linear (GLC-L). The base GLC-L algorithm was extended to GLC-DRL to do automatic dimensionality reduction. It was also extended to be interactive GLC-IL and to do automatic optimization for separation GLC-AL. All of these algorithms are explained in the following subsections of this chapter, along with comparison of GLC-L with Parallel Coordinates.

Comparison of GLC-L and Parallel Coordinates

Parallel Coordinates can visualize n-dimensional data in 2-D without losing any information, similar to GLC-L. However, when Parallel Coordinates are used to visualize large datasets, the visualization gets cluttered due to the abundant amount of information.

Parallel Coordinates visualization has a number of vertical lines (axis) corresponding to the number of attributes in the dataset. Those axis are presented parallel to each other. Axis have dashes (marks) on them with values in range corresponding to the data value themselves. Plotting data samples onto Parallel Coordinates is done by setting marks on the axis at the location corresponding to the data values and connecting lines between the marks on the axis. Each instance in the dataset will get one line across all the axis, connecting each axis at the location of the marks, which are determined by the data values. The color of the lines is often used to distinguish between instances from different classes.

Parallel Coordinates can be used to extract rules and information from n-dimensional data in 2-D graphs. The visualization with Parallel Coordinates is easy to understand when the number of dimensions is not very large and the number of data samples visualized stays relatively low. As the number of dimensions increases, the number of axis also increases, squeezing them closely together. Having a large number of axis makes the visualization very cluttered and makes it difficult to extract rules and information.

GLC-L visualization produces less clutter by drawing line segments in the length only corresponded to the data value. Not having an axis drawn with a length to accommodate for the entire range of data values produces a visualization with much less clutter.

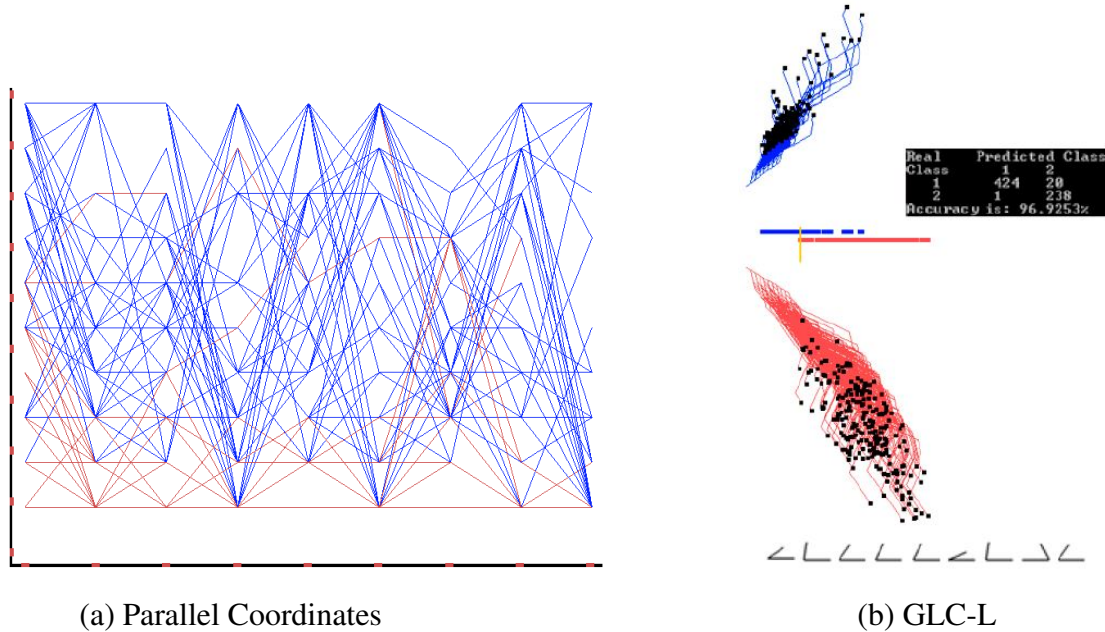


FIGURE 6: Comparison of Parallel Coordinates and GLC-L on Wisconsin Breast Cancer dataset. Parallel Coordinates visualizing only 100 instances while GLC-L is visualizing the entire dataset with 683 instances.

Fig. 6 shows visualizations of Wisconsin Breast Cancer dataset with Parallel Coordinates and GLC-L. Parallel Coordinates visualize 100 instances, while GLC-L is visualizing the entire data set with 683 instances. GLC-L provides a visualization with separation of classes, making it easier to see the patterns. Parallel Coordinates visualization is quite messy making it difficult to use for extraction of rules and useful information.

Parallel Coordinates, although visualizing nearly 6 times less data instances than GLC-L, produce a more cluttered visualization of the 2 classes. GLC-L does not have lines from one class overlap with lines from a different class. The overlapping lines in GLC-L form a pattern, which can be used to identify instances between different classes. Parallel Coordinates have lines overlapping from different classes, preventing the discovery of patterns between classes. Avoiding cluttered visualizations is one of the main challenges in lossless n-dimensional data visualization. Having a cluttered visualization defeats the entire purpose of n-dimensional data visualization. Extracting useful information and rules from a messy visualization is as difficult as it would be to extract it from an Excel sheet with a large number of rows and columns.

An in-dept study has been conducted by Kovalerchuk, where Parallel Coordinates are compared with GLC. It concludes that GLC-L provides a less cluttered visualization among other advantages, such as the ability to separate classes automatically [1], [16].

GLC-L

GLC-L is used to visualize a linear function. Each attribute (x) from the data instance makes a line segment, and the line segments are stacked one on top of the other. The length of the line segment is determined by the value of the attribute. A large value

produces a longer line segment than a smaller value. The angle of the line is based on the coefficients (C) which are normalized between [-1,1]. Having coefficients :

$$C = (c_1, c_2, \dots, c_{n+1}),$$

normalizing $C = (c_1, c_2, \dots, c_{n+1})$ by creating as set of normalized parameters:

$$K = (k_1, k_2, \dots, k_{k+1}) : k_i = c_i / c_{\max},$$

the equation is the following:

$$y = k_1x_1 + k_2x_2 + k_3x_3 + \dots + k_nx_n + k_{n+1}x_{n+1},$$

where $k_i = \cos(\arccos(k_i))$ and x_i is a data attribute at i . $Q_i = \arccos(|k_i|)$ which is the computed angle. Fig. 7 contains more information.

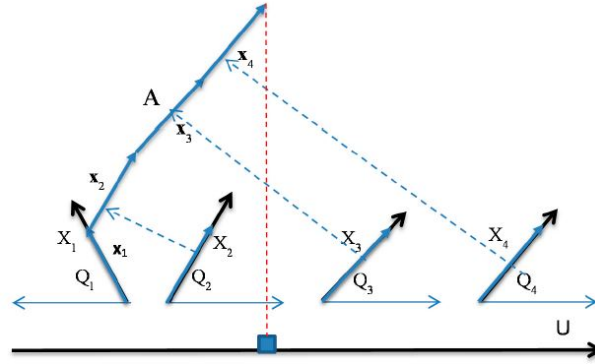


FIGURE 7: Visualization of 4-D data with GLC-L. The data attributes (X_1 - X_4) are all positive numbers with the value of 1. Having the same values for all attributes makes all of the line segments to be equally long. The first angle (Q_1) is a negative which turns the line to the left and the other angles (Q_2 - Q_4) are positive, turning the line to the right [1].

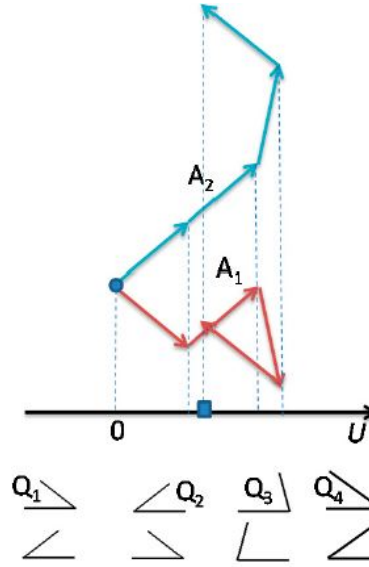


FIGURE 8: Visualizing 4-D data $A = (-1, 1, -1, 1)$ in two different representations A_1 and A_2 with angles labeled (Q_1 - Q_4). The flipped angles below are a representation of how the line segments will turn if the data value is negative [1].

When the data value is positive and the angle is positive, the line segment turns to the right. If the data value is positive and angle is negative, then the line segment turns to the left. And if the data value and the angle is negative, then the line segment turns to the right. The end point of the last line segment is projected onto the line below. This end point is then used for classification. An alternative way to visualize GLC-L would be to go down when the data value is down. However, going up and down creates a messier visualization. Examples of alternative representations can be seen in Fig. 8.

Algorithm 1 contains the pseudo code for the base GLC-L algorithm. All of the line segments start to be drawn from the same location. The class labels are excluded from being a line segment but rather determine the color of the line. In Fig. 1 the class label is used to determine if the samples are drawn above the median or below. Drawing samples

Algorithm 1 Pseudo code for GLC-L

```
1: procedure GLCLINEAR(data, coefficients)
2:    $numRows \leftarrow data.size()$  ▷ Number of data instances
3:    $numColumns \leftarrow data[0].size()$  ▷ Number of attributes
4:   for  $i \leftarrow 0$  to  $i < numRows$  do
5:      $x \leftarrow 0$ 
6:      $y \leftarrow 0$ 
7:     for  $j \leftarrow 0$  to  $j < numColumns$  do
8:        $radius \leftarrow data[i, j]$ 
9:        $angle \leftarrow calculateAngle(coefficients[j])$  ▷ Coefficient to radians
10:       $new\_x \leftarrow x + radius * cos(angle)$ 
11:       $new\_y \leftarrow y + radius * sin(angle)$ 
12:       $drawLine(x, y, new\_x, new\_y)$  ▷ Draws line segment between 2 points
13:       $x \leftarrow new\_x$ 
14:       $y \leftarrow new\_y$  ▷ Update starting position
15:     end for
16:   end for
17: end procedure
```

above and below the median also provides a cleaner visualization, opposed to drawing them in the same space with different colors.

GLC-AL

Automatic search for best coefficients is optimized based on the separation of classes. The search for the coefficients is done by a random search algorithm. The random search algorithm tries different sets of coefficients on the training dataset and the best coefficients are then evaluated on the test dataset. Training and testing sets are created by splitting the original dataset into 2 sub sets after shuffling.

Algorithm 2 Pseudo code for GLC-AL

```
1: procedure COEFFICIENTSSEARCH(data, epochs)
2:    $n \leftarrow 0$ 
3:    $best\_coefficients \leftarrow []$ 
4:    $best\_accuracy \leftarrow 0$ 
5:   while  $n < epochs$  do ▷ Number of epochs
6:      $coefficients \leftarrow random(-1, 1)$ 
7:      $current\_accuracy \leftarrow 0$ 
8:     for  $i \leftarrow 0$  to  $i < data.size()$  do ▷ Number of data instances
9:        $x \leftarrow 0$ 
10:      for  $j \leftarrow 0$  to  $j < data[0].size()$  do ▷ Number of attributes
11:         $radius \leftarrow data[i, j]$ 
12:         $angle \leftarrow calculateAngle(coefficients[j])$  ▷ Coefficient to radians
13:         $new\_x \leftarrow x + radius * cos(angle)$ 
14:         $x \leftarrow new\_x$ 
15:      end for
16:    end for
17:     $current\_accuracy \leftarrow evaluateCoefficients()$  ▷ Calculate accuracy
18:    if  $current\_accuracy > best\_accuracy$  then
19:       $best\_coefficients \leftarrow coefficients$ 
20:       $best\_accuracy \leftarrow current\_accuracy$ 
21:    end if
22:     $n \leftarrow n + 1$ 
23:  end while
24: end procedure
```

GLC-AL optimizes the coefficients based on the classification accuracy of the separation. 1-D points, which are the end points from the lines created by GLC-L, are used for separation of classes. A threshold selected based on best separation of classes is evaluated with a confusion matrix producing a classification accuracy. During training the best coefficients are selected and a corresponding threshold separation is found. Using the selected coefficients during training the test data is projected and evaluated based on the threshold from training.

Algorithm 2 has the pseudo code for GLC-AL. The algorithm does a random search to find the best set of coefficients, evaluating which set of coefficients separates the data

between classes more effectively. The algorithm takes in a dataset and also a number of epochs, which stands for the number of iterations to do. Having 100 iterations means the algorithm will try 100 different sets of coefficients and select which ones best separate the data between classes.

GLC- IL

Several interactive functionalities have been implemented for GLC-IL: moving the separation threshold, selecting a specific zone for further exploration, and switching coefficients for a different representation of data. When the separation threshold is moved, a new confusion matrix is calculated. The trade of in accuracy can be seen as a result of misclassifying some samples due to the new threshold. In medical data false positives could be fatal and a trade of in accuracy for less false positives is important.

GLC-AL is run again if a different visualization of the dataset is needed. Visualizing data with different coefficients provides different perspectives and insights of the data. The projection of the data can be toggled until a satisfactory visualization is projected. Being able to select an overlap area of 1-D points for further exploration can lead to interesting findings. Points which are within the overlap zone are put through GLC-AL to find a better separation.

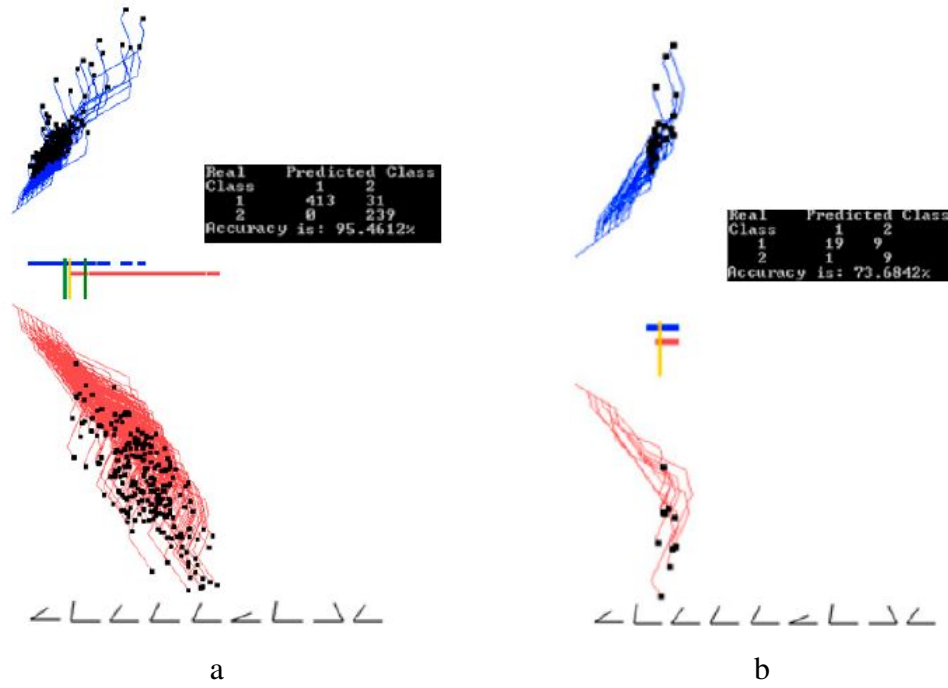


FIGURE 9: Interacting with Wisconsin Breast Cancer dataset. The malignant cases are drawn in red and benign in blue. In Fig. 9a, two green lines are set interactively to specify the overlap zone. Fig. 9b has a visual of only projecting the selected overlap zone.

GLC-DRL

Two different dimensionality reduction methods were implemented for GLC-DRL. The first one is an automated way of reducing dimensionality by removing attributes which do not contribute to the overall classification of the dataset. The second one is an interactive method, allowing for a specific dimension to be removed. The interactive method would remove the selected attribute from the dataset and make a new visualization excluding the removed attributes.

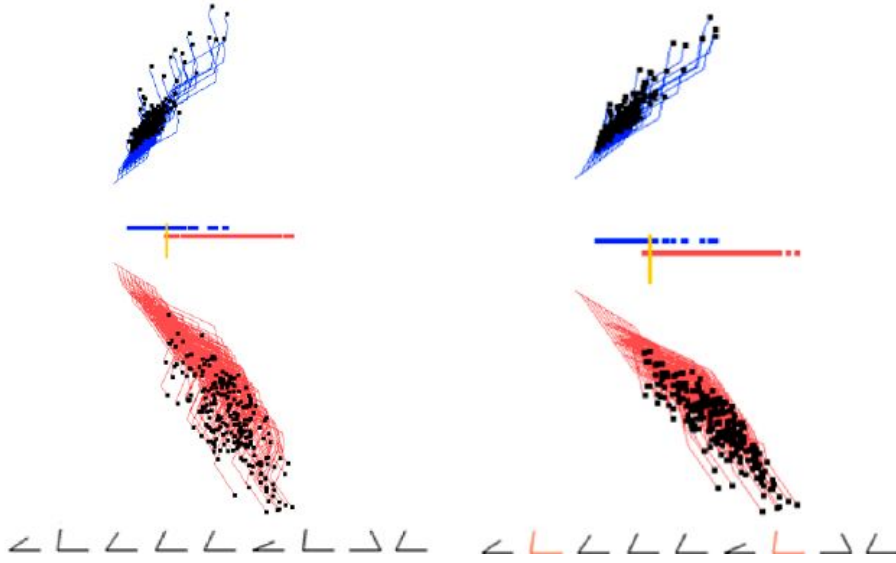


FIGURE 10: Dimension reduction and visualization of Wisconsin Breast Cancer dataset. Showing how the visualization changes when 2 dimensions are removed. 2nd and 7th dimension is removed, the corresponding angles are highlighted in red.

The automated algorithm for removing dimensions works by keeping track of the line as it stretches in the x direction. The classifier does not take into account the y position of the end point but only the x position. The automatic dimensionality reduction method keeps track of how much on average each dimension contributes to the overall length of x . A threshold is set manually determining where the cutoff should be. Dimensions are removed based on the threshold and the average contribution of each dimension. Dimensions with low contributions get removed.

GLC-L Limitations

GLC-L classifies the 1-D points which are the x locations of the end points from the projection. When only taking the x location of the last line segment, we disregard a lot of useful information provided by the visualization. In cases where 1-D points are sufficient for successfully separating data points from different classes, the disregard of other

information is not a problem. However, in cases where 1-D points are not sufficient to separate data from different classes, it makes sense to use other information provided by the visualization. Experiments were conducted on 1-D points created by the projection to understand what those 1-D points look like for two different datasets. One of the datasets is Wisconsin Breast Cancer, where GLC-L is able to separate the data from different classes very well. The other dataset is the FERET Database [17], where the GLC-L does not perform well due to the complexity of the data. FERET Database contains labeled frontal photographs of males and females.

Visualizing these two data sets with GLC-L produces two very different results. One visualization provides a clear separation of classes, while the other one does not. The 1-D points from both visualizations were saved to text files for further analysis. Histograms were created with those 1-D points in order to understand how the data from different classes overlaps. The 1-D points were normalized by taking the minimum 1-D point from a dataset and adding it to every other 1-D point in the corresponding dataset. This resulted in having the minimum 1-D point at 0.

Three different histograms were generated for each dataset. One for each class and one for the two classes together. The histograms have two axis, “Frequency” and “Bin”. The “Frequency” corresponds to how many 1-D points are in a certain “Bin,” where “Bin” is the range of the 1-D points.

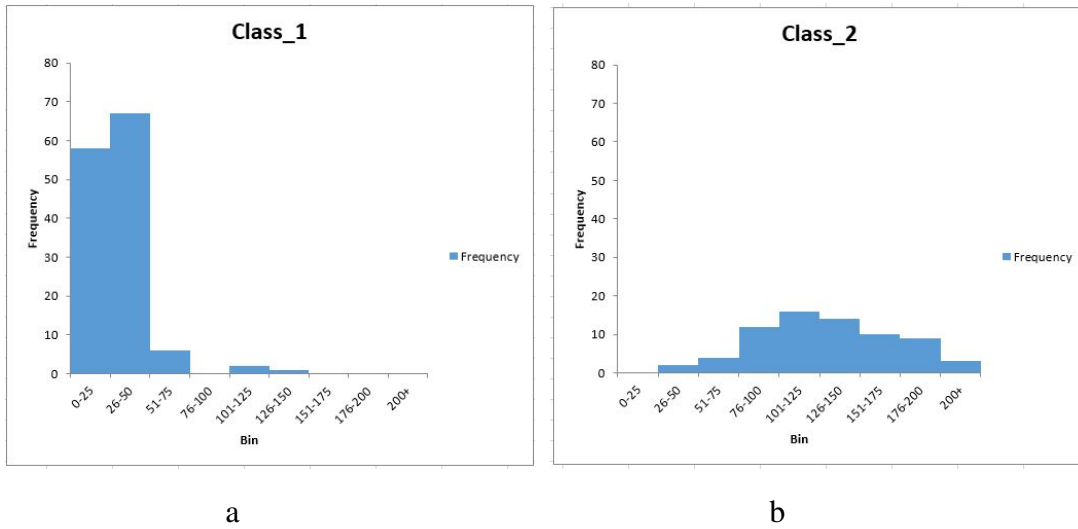


FIGURE 11: 1-D points from the Wisconsin Breast Cancer dataset. Fig. 11a has 1-D points from class 1 and Fig. 11b has points from class 2.

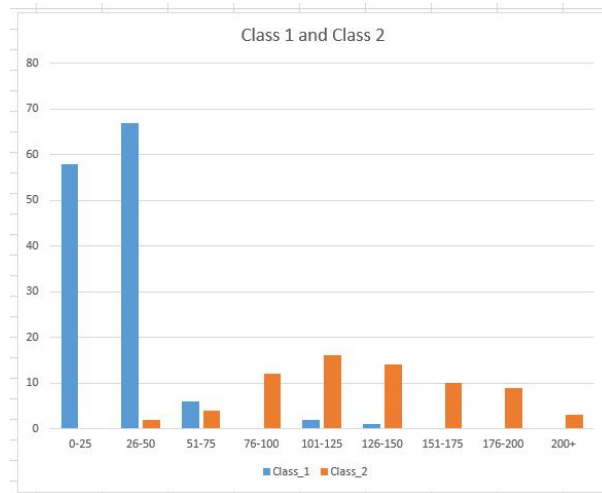


FIGURE 12: 1-D points from both classes from the Wisconsin Breast Cancer dataset. Class 1 samples are in blue and Class 2 samples are in orange.

Looking at Fig. 12, it can be seen that the data is easily separated with 1-D points. A simple rule can be extracted from these 1-D points to successfully separate the two different classes. 1-D points which are less than 50 belong to Class 1 and points greater than 50 belong to Class 2. With such a simple rule the data will be classified correctly with very high accuracy.

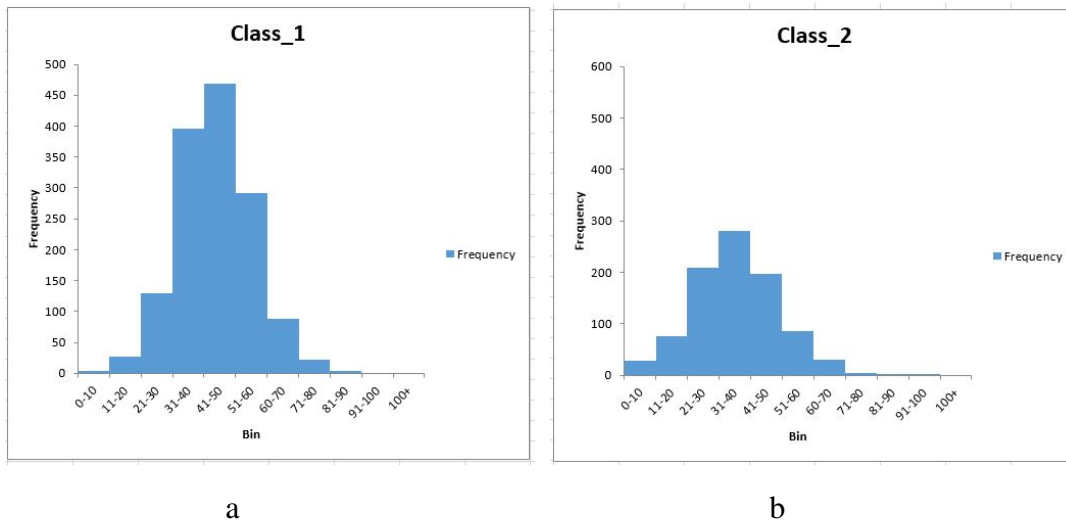


FIGURE 13: 1-D points from the FERET Database. Fig. 13a has 1-d points from class 1 and Fig. 13b has points from class 2.

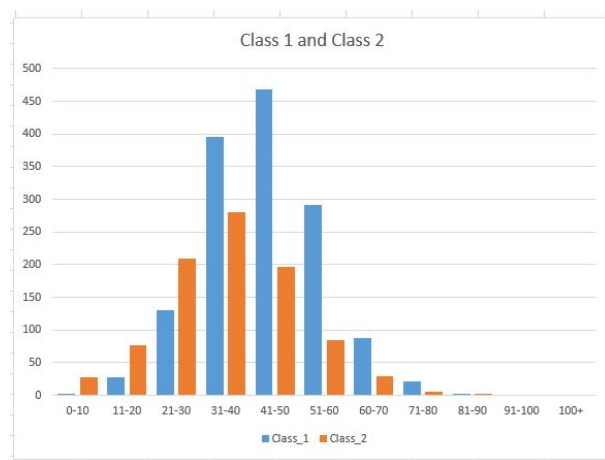


FIGURE 14: 1-D points from both classes from the FERET Database. Class 1 samples are in blue and Class 2 samples are in orange.

Fig. 14 contains 1-D points of both classes from the FERET Database. It can be seen that the 1-D points are not easily separated between the two classes. The separation of classes does not exist with the 1-D points; the overlap is too great to be able to make rules for separation.

Experiments with GLC-L such as these motivated the data transformation method. GLC-L produces a lossless visualization of n -dimensional data. Discarding the visualization and only using a 1-D point to represent an entire data instance has a very large compression rate. Making use of the entire visualization for classification seemed very promising.

CHAPTER IV

DATA TRANSFORM

Using GLC-L to visualize only one data instance at a time produces a 2-D graph (an image) with that specific data instance in it. The process can be repeated for every single instance in the dataset using the same coefficients (angles for line segments) to transform the entire dataset. Those artificially created images can be used for extraction of rules and information from the dataset. Transforming numeric multidimensional datasets into 2-D graphs (images) provides a unique lossless transformation of data.

The purpose of this study is to see if these artificially created images can be used by machine learning algorithms such as Support Vector Machine (SVM) [3], Multilayer Perceptron (MLP) [4], and Convolutional Neural Network (CNN) [7] for classification. SVM and MLP can be used to train models with data in a format of a vector and a raster (image). For these two algorithms, image data gets reshaped to a vector format and is then used for training/testing the model. The CNN is, however, designed specifically for raster input. It uses adjacent attributes to extract relations from the data, meaning two neighboring pixels have more relation with each other than a pixel from one corner of the image has with a pixel from a different corner.

The GLC-L Data Transform is described in this chapter along with experiments and findings for some of the components of the GLC-L Data Transform system. Experiments with datasets and their results are presented in a later Chapter.

GLC-L Data Transform

GLC-AL does n number of epochs to find which coefficients best transform the data for classification. This process is done by choosing a random set of coefficients and

transforming the data with those coefficients. This means using the same set of randomly chosen coefficients. Each data instance gets visualized with GLC-L and an image is created by capturing the visualization. The image captures of the visualizations get their corresponding labels from the original data. Once the data is transformed and images are created they are put through a machine learning algorithm for classification. The images train a model with the training set and the model gets evaluated with the testing set producing a result, which is the classification accuracy. This process is repeated n number of times to find the best set of coefficients.

For the process of finding the best coefficients, a training set and a testing set is needed. And for the purpose of testing the system and the best coefficients, a second testing set is required. The second testing set is required to properly test the data transformation system. This data can't be part of the process for coefficient selection because it would lead to overfitting. Overfitting means the model learns the training data and learns how to separate it without generalizing. If the model does not generalize well, it will not perform well on unseen data since it just learned the training data. For this purpose the system takes in a dataset, shuffles it, and splits it into two new sets.

The first set (set-1) is 80% of the original dataset, and the second set (set-2) is the remaining 20% of samples. Refer to Fig. 15 for a visual representation of how the data gets split into two different sets.

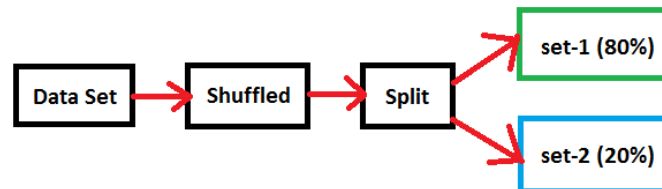


FIGURE 15: Dataset gets shuffled and split into two new sets for the purpose of training and testing the system.

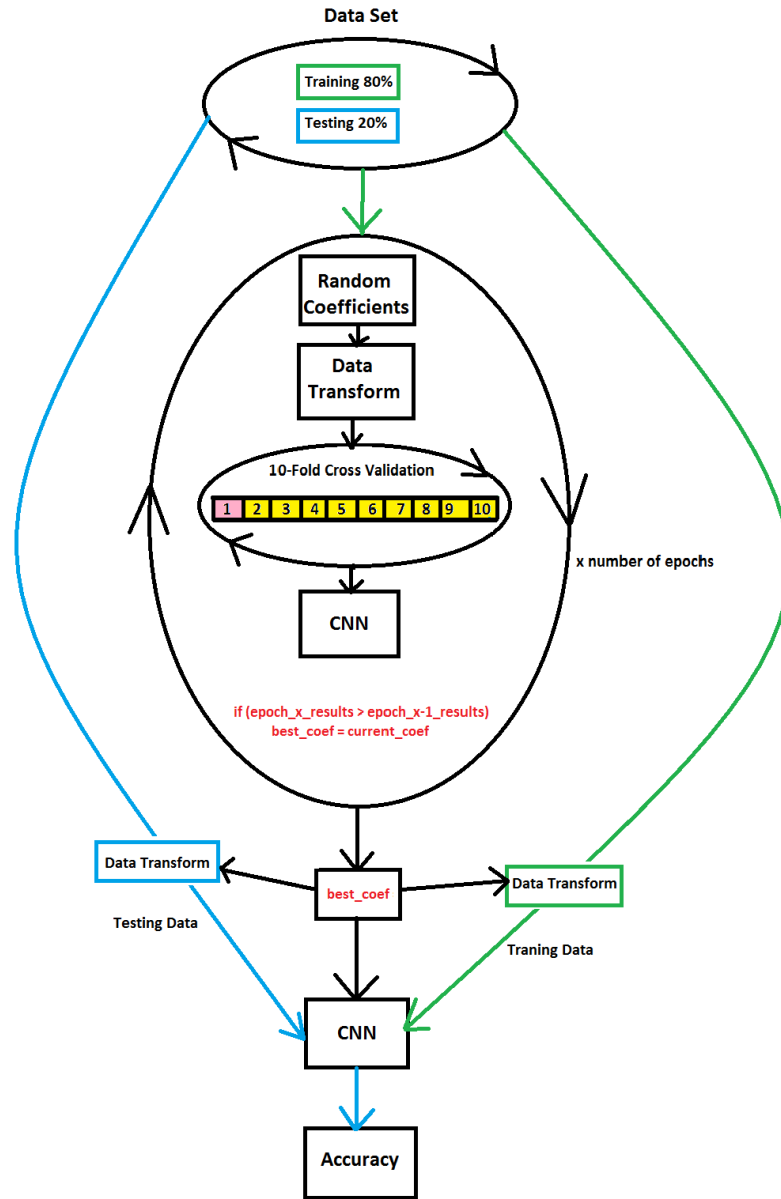


FIGURE 16: Diagram of the data transform system. Shows how set-1 and set-2 are used to train and evaluate the system. "best_coef" stands for best coefficients.

10-fold cross validation [18] is done on set-1 during the search of best coefficients. 10-fold cross validation means that the data gets split equally into 10 subsets. First of the 10 subsets gets chosen to be the testing set, while the other 9 subsets are used for training. This process is repeated 10 times with changing the testing set every time to the next one,

while including the previous testing set for training. Doing this results in training and testing on the entire dataset. The accuracy from each testing subset is recorded and the overall accuracy is reported as an average of the 10 runs. Set-2 is used to test the system after the best coefficients are established.

Refer to Fig. 16 for a visual representation of the system diagram. In this system diagram, a CNN is displayed as the classification algorithm. In place of it, a SVM or MLP can be used. The initial split of data to set-1 and set-2 is done multiple times to fully use the entire dataset. The results for the testing set are recorded for each run of the system and an average is reported as the final classification result.

When the line is being produced for visualization with GLC-L, it is not certain how far up, left, and right it will reach. This all depends on the number of attributes in the dataset, the value of each attribute which determines the length of the line segment and which angles are used. Angles of 90 degrees make the line go more up than angles which are closer to 0 degrees. Low angles make the line go more left or right depending on the attribute value. Dimensions of the window in which the line gets drawn are referred to as the world view coordinates. Those coordinates are different from the size of the image created. Refer to Fig. 17 for more information about world view coordinates.

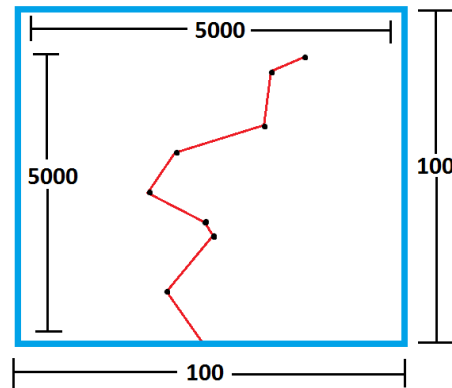


FIGURE 17: World view coordinates are 5000x5000, while the image they are converted in is 100x100.

The world view coordinates of the visualization are dynamic for each experiment. They are set dynamically for each transformation after the coefficients have been chosen. Having the same world view coordinates for each data instance is crucial in order to maintain lossless visualization. If each image was to get different world view coordinates based on how far the line reaches, it would result in loss of information. The world view coordinates are set by finding the largest world view coordinates for a specific experiment. This is done by randomly shuffling the training set and taking a subset of 10%, visualizing the subset with the already chosen coefficients and recoding the extremes. The extremes are the largest world view coordinates used during the projection of the subset. Once those extremes are recorded, they are used for the entire dataset for visualization.

After some preliminary results have shown, a CNN performs better than SVM and MLP at classifying artificially created images. Further experiments to better understand the data transform were only conducted using the CNN. The data transform has many parts which can be optimized, including the architecture of the CNN itself. Optimization of hyperparameters of a CNN are out of scope for this study. These hyperparameters include the number of convolutional layers, the number of pooling layers, filter size, and other parts which make up the architecture of a CNN. Experiments were focused on how different parts affect the data transformation while evaluating the artificially created images on the same CNN. The architecture of the CNN used for these experiments can be found in section **CNN Experiments** in Chapter **Experimental Results**. Experiments were generated to understand how image size of the artificially created images and line thickness affects classification. Experiments were also done regarding random search and how increasing the number of epochs for trying different sets of coefficients affects classification.

These experiments were all done on the same dataset using the same split for training, validation, and testing subsets. The split for the subsets was 80% for the training set, 10% for the validation set, and 10% for the testing set. The Diabetic Retinopathy Debrecen [19] dataset was chosen because the classes are balanced, with class 1 having 540 instances and class 2 having 611 instances. This dataset has 18 attributes and is not easily separable due to the complexity of the data. For experiments with line thickness and size of the artificially created images, the same coefficients were used for all experiments. Essentially only 1 epoch was done by the system in search of coefficients.

Thickness of Line in Artificially Created Images

The thickness of the line directly corresponds to how many pixels are being used in the image created. Having a thin line corresponds to having a small number of pixels actually being used in the image. Having more pixels used corresponds to having more information in the image. Experiments were done to test this hypothesis. Three different experiments were conducted to test different line widths and their effect on classification. The value t , which is how thick the line is, was the only variable being changed between experiments. For each experiment, 5 different runs were performed and the average of the 5 runs is reported on the test set.

TABLE 1: Line Thickness Experimental Results

Run #	$t = 0.1(\%)$	$t = 1.0(\%)$	$t = 2.0(\%)$
1	48.24	71.92	71.05
2	72.80	71.92	73.68
3	75.43	70.17	71.05
4	48.24	74.56	72.80
5	71.92	76.31	73.68
Average	63.23	72.97	72.45

Notes: Columns with t values correspond to the t value used for the experiment and the accuracy it produced on the testing set.










Image 1	Image 2	Image 3	t value
			0.1
			1.0
			2.0

FIGURE 18: Projection line thickness with different t values. Image 1 ,2, and 3 were randomly saved from the experiment. The 4th column contains the t value used for each visualization in the corresponding row.

The first experiment had line thickness at $t = 0.1$, second experiment with $t = 1.0$, and the third experiment with $t = 2.0$. Examples of how different t values change the line thickness and the visualization itself can be seen in Fig. 18.

Table 1 contains the results produced from the experiments done to test how the line thickness affects classification accuracy using a CNN. It can be seen that there is a dramatic improvement in accuracy going from $t = 0.1$ to $t = 1.0$. After that there is not much gain in classification accuracy going to $t = 2.0$. These experiments show that the thicker lines produce higher results, having more information leads to an improvement in accuracy.

Size of Artificially Created Images

Four different experiments were conducted to see how the size of artificially created images affects classification accuracy with the CNN. For this set of experiments the size

of artificially created images was the only variable being changed. The t value was set to 1.0. In the CNN, the input dimension had to be changed to the corresponding image size, but all other hyperparameters for the CNN remained unchanged. For each of the 4 experiments, 5 different runs were performed and the average of the 5 runs are reported on the test set.

For the first experiment images of 50x50 pixels were used. This means no matter what world view coordinates are, the projection ended up being compressed into an image of 50x50 pixels. For the next experiments, the size was enlarged to 100x100, 200x200, and finally 300x300 pixels.

TABLE 2: Image Size Experimental Results

Run #	50x50 (%)	100x100 (%)	200x200 (%)	300x300 (%)
1	74.56	71.92	51.75	51.75
2	65.78	71.92	48.24	51.75
3	74.56	70.17	51.75	51.75
4	72.80	74.56	72.80	51.75
5	76.31	76.31	75.43	48.24
Average	72.80	72.97	59.99	51.05

Notes: First column contains the run number out of 5. The remaining columns represent the size of images used in the experiment and accuracy on the test set.

Table 2 contains experimental results for different images sizes. It can be seen that there is no significant gain in performance going from 50x50 pixel images to 100x100 pixel images. Having images larger than 100x100 makes the accuracy drop significantly. Accuracy such as 51.75% and 48.24% is the result of the CNN not being able to converge during training. This is due to the hyperparameters of the CNN. The CNN hyperparameters, such as number of filters, filter size, and number of layers, were tuned manually by guessing and checking on much smaller images. Optimization of hyperparameters for a CNN are out of scope for this study.

Fig. 19 contains examples of artificially created images of different size. It can be seen how the lines get sharper and provide more details as the image size increases.






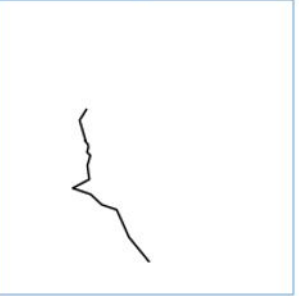
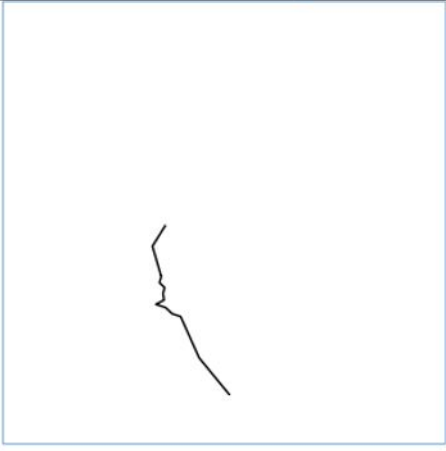
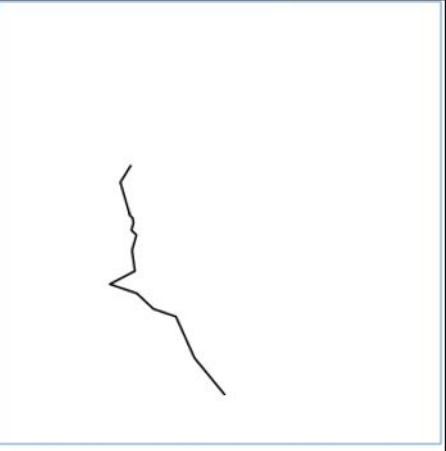
Image 1	Image 2	Image Size
		50x50
		100x100
		200x200
		300x300

FIGURE 19: Examples of different image sizes used for this set of experiments. Larger image sizes produce much more detailed lines.

Random Search Algorithm

GLC-AL does random search [20] to select the best coefficients for the data transformation. Random search is not a structured search. A random set of coefficients is chosen every iteration no matter how well or poorly the previous set of coefficients performs. Random search is very effective at trying a large variety of coefficients in a short amount of time.

Experiments were done to show that a large search is not needed for satisfactory results. For this set of experiments the number of epochs was the only variable being changed. Three different experiments were conducted with 5 runs for each experiment. In the first experiment, epochs was set to 1. Only one set of randomly selected coefficients was used to do the transformation and evaluate the system. For the second and third experiments, epochs was set to 20 and 100 respectively. The experiment with 20 epochs means there was 20 different sets of randomly chosen coefficients to transform the data and the system was evaluated on the coefficients producing the highest classification accuracy.

TABLE 3: GLC-L with Random Search

Run #	Epochs=1 (%)	Epochs=20 (%)	Epochs=100 (%)
1	64.03	66.78	66.78
2	68.42	71.05	65.78
3	68.42	64.03	69.29
4	63.15	73.68	68.42
5	67.54	69.29	65.78
Average	66.31	68.96	67.21

Notes: First column contains run numbers out of 5. The following 3 columns have the number of epochs used in the experiment. The last row is the average for each column.

Table 3 contains the experimental results for the random search algorithm which is used for selecting coefficients. The last row has the averages of the 5 runs for each of

the experiments, where different number of epochs was used. It can be seen that there is a small improvement going from epochs = 1 to epochs = 20. Experiment with epochs = 20 had CNN models trained on 20 different representations of data, rather than just 1. However, going to epochs = 100 does not further improve the results.

The random search experiments show that the search for coefficients does not have to be large. There is abundant amount of linear functions which can be used for the transformation of data. Although there is a possibility that there exists a linear function which produces the best separation of data, it is not required for the transformation of data.

CHAPTER V

EXPERIMENTAL RESULTS

Case studies and their results with the pure GLC-L can be found in the article [1], [21].

Experimental results were gathered on the data transformation system. Three different machine learning algorithms were used to evaluate the transformation of numerical data with lossless visualization provided by the GLC-L. The artificially created images were put through CNN, MLP, and SVM for training and testing. In order to have a fair comparison on how these three machine learning algorithms perform with artificially created images, two sets of experiments were conducted for each of the algorithms. Each of the algorithms was evaluated on the raw numerical data and on the transformed images which were created by the data transformation system using GLC-L.

Three different sets of experiments were gathered for comparison, two for each of the algorithms. In the first set of experiments, a CNN was used to evaluate the raw numerical data and the artificially created images. For the second set of experiments, a MLP was used, and for the third a SVM.

For all of the experiments, the hyperparameters of the machine learning algorithms remained the same. In the data transformation system the following were used for final experiments:

- t value of 1.0
- size of artificial images 50x50
- 20 epochs with the data transformation system
- 10-fold cross validation

Experiments were carried out on the CWU supercomputer, taking advantage of the GPU clusters for the CNN. The data transformation system was implemented in Python. For the CNN model, a Python library Keras [22] was used. For the MLP and SVM, a Python machine learning library scikit-learn was used [23]. The following subsections contain the experimental results for each of the three machine learning algorithms. The details of each of the machine learning algorithms used for experiments are provided in the corresponding subsection.

CNN Experiments

For this set of experiments two different CNN architectures had to be used. CNN is designed for 2-dimensional input such as images. CNN was used for classification of artificially created images. However, in order to use a CNN to classify 1-dimensional data (a vector), 1-D convolutional neural network was constructed.

Both of the architectures used the same Adam optimizer [24] with the learning rate set to 0.0001. The network was set to do 1,000 epochs with an early stopping check point. If the validation accuracy during training is not improved over 100 epochs, the training would stop.

The architecture of the CNN used for classification of images is the following:

- Convolutional Layer with 64 output channels, a kernel shape of 2x2, stride of 2x2, and RELU activation [25]
- Convolutional Layer with 64 output channels, a kernel shape of 2x2, stride of 2x2, and RELU activation
- Pooling layer with pooling size of 2x2
- Drop out layer with fraction of input units to drop set to 0.4
- Convolutional Layer with 128 output channels, a kernel shape of 2x2, stride of 2x2, and RELU activation
- Convolutional Layer with 128 output channels, a kernel shape of 2x2, stride of 2x2, and RELU activation
- Pooling layer with pooling size of 2x2
- Drop out layer with fraction of input units to drop set to 0.40
- Fully Connected Layer with 256 output nodes and RELU activation
- Drop out layer with fraction of input units to drop set to 0.40
- Fully Connected Layer with number of output nodes equal to the number of classes, with a softmax activation

The original MNIST-Subset was not put through the 1-D CNN because the original data is an image. Instead, the original images were evaluated with the CNN architecture used for classification of the artificially created images.

The architecture of the 1-D CNN which is used to classify vector data is the following:

- 1-D Convolutional Layer with 64 output channels, filter length of 2, and RELU activation
- Drop out layer with fraction of input units to drop set to 0.4
- Fully Connected Layer with 1024 output nodes and RELU activation
- Fully Connected Layer with number of output nodes equal to the number of classes, with a softmax activation

TABLE 4: CNN Result Comparison

Dataset Name	CNN-1D (%)	Transform+CNN (%)
Swiss Roll 2-D	72.50	97.43
Swiss Roll 3-D	96.18	97.55
Breast Cancer Wisconsin	96.92	97.22
Red Wine	60.65	60.93
White Wine	55.91	60.76
Diabetic Retinopathy Debrecen	73.75	69.04
MNIST-Subset (Regular CNN)	99.53	92.93

Notes: Comparison of results for raw input data with 1-D CNN and artificially created images with regular CNN. The table is sorted with respect to number of attributes in the dataset.

Column 1 in Table 4 has the dataset names, column 2 has the results for the raw input data with 1-D CNN. Column 3 contains the experimental results of transforming numerical data with GLC-L and classifying images with a regular CNN. The results for “1-D CNN” are gathered using 10-fold cross validation, and the results for “Transform + CNN” are an average of 5 runs using the data transformation system where 10-fold cross validation is used for training the CNN.

TABLE 5: Transform+CNN Runs

Run #	Swiss 2-D (%)	Swiss 3-D (%)	Wisconsin (%)	Red (%)	White (%)	Diabetic (%)	MNIST (%)
1	98.12	97.81	97.08	61.87	62.34	66.52	91.33
2	95.93	97.81	97.08	59.68	59.18	68.69	94.00
3	96.87	96.56	98.54	61.87	63.36	70.43	94.16
4	99.37	97.18	96.35	59.06	58.16	72.17	92.00
5	96.87	98.43	97.08	62.18	60.76	67.39	93.16
Average	97.43	97.55	97.22	60.93	60.76	69.04	92.93

Notes: Results for each of the 5 runs with the GLC-L Transform and CNN. The average results are reported in Table 4. Names of the datasets have been abbreviated.

Table 5 contains the results for each of the 5 runs of the data transformation with GLC-L and CNN classification of artificially created images. Names of the datasets have been abbreviated. The average of the 5 runs is reported for comparison in Table 4. The datasets are arranged left to right with respect to the number of dimensions in the dataset.

Comparison of the results between the 1-D CNN and the GLC-L Transform + CNN is presented in Table 4. 1-D CNN did not perform well with the Swiss Roll 2-D dataset with only 72.50% classification accuracy, while the GLC-L transformation + CNN did 97.43%. Transformation with GLC-L+ CNN performed slightly better for Swiss Roll 3-D dataset than the 1-D CNN, getting 97.55% versus 96.18%. 1-D CNN classification accuracy is slightly lower with the Breast Cancer Wisconsin dataset than the transformation with GLC-L + CNN. 1-D CNN produced 96.92%, while the transformation with GLC-L + CNN produced 97.22% classification accuracy for the Breast Cancer Wisconsin dataset.

Both methods performed poorly with the Red Wine Quality dataset, with 60.65% with the 1-D CNN and 60.93% with the GLC-L transformation + CNN. White Wine Quality dataset got higher classification accuracy with the GLC-L transform + CNN 60.76% , while the 1-D CNN got 55.91%. 1-D CNN performed better with the Diabetic

Rectionopathy Debrecen dataset than the transformation with GLC-L + CNN. 1-D CNN produces 73.75% classification accuracy, while the GLC-L transformation + CNN 69.04%. MNIST-Subset was evaluated on the regular CNN for both sets of experiments. Original images were put through the regular CNN and then the artificially created images with the GLC-L transformation were put through the same CNN. The original images produce classification accuracy of 99.53%, while the transformed images produced 92.93%. MNIST-Subset was used to test the data transformation system in very high dimensions. Even though there is a drop in accuracy, it still shows that the data transformation with GLC-L produces useful information, even in very high dimensions.

MLP Experiments

For this set of experiments MLP was used as the machine learning algorithm for classification of the raw and the transformed data. The MLP classifier from scikit-learn Python library was used with all of the default hyperparameters. The default number of hidden nodes is 100 and the number of epochs is 200. The default number of epochs is fairly low compared to the 1000 with the CNN. However, the CNN had an early stopping checkpoint. The parameter was left default since optimization of the hyperparameters was not in the scope of this study.

TABLE 6: MLP Result Comparison

Dataset Name	MLP (%)	Transform+MLP (%)
Swiss Roll 2-D	97.50	95.87
Swiss Roll 3-D	98.43	63.87
Breast Cancer Wisconsin	94.83	89.36
Red Wine	60.78	46.12
White Wine	53.57	46.03
Diabetic Retinopathy Debrecen	69.84	53.47
MNIST-Subset	99.10	83.29

Notes: Comparison of results for raw input data with MLP and artificially created images with MLP. The table is sorted with respect to number of attributes in the dataset.

Column 1 in Table 6 has the dataset names, column 2 has the results for the raw data with MLP. Column 3 contains the experimental results of transforming numerical data with GLC-L and classifying images with a MLP. The results for “MLP” are gathered using 10-fold cross validation, and the results for “Transform + MLP” are an average of 5 runs using the data transformation system where 10-fold cross validation is used for training the MLP.

TABLE 7: Transform+MLP Runs

Run #	Swiss 2-D (%)	Swiss 3-D (%)	Wisconsin (%)	Red (%)	White (%)	Diabetic (%)	MNIST (%)
1	97.18	74.06	95.72	43.75	47.12	52.17	87.00
2	97.18	78.43	93.43	46.87	46.02	48.26	80.33
3	96.25	36.87	97.81	45.93	46.12	47.39	82.83
4	91.87	36.87	65.69	46.56	45.81	49.56	86.50
5	96.87	90.62	94.16	47.50	45.10	70.00	79.83
Average	95.87	63.37	89.36	46.12	46.03	53.47	83.29

Notes: Results for each of the 5 runs with the GLC-L Transform and MLP. The average results are reported in Table 5. Names of the datasets have been abbreviated.

Table 7 contains the results for each of the 5 runs of the data transformation with GLC-L and MLP classification of artificially created images. Names of the datasets have

been abbreviated. The average of the 5 runs is reported for comparison in Table 6. The datasets are arranged left to right with respect to the number of dimensions in the dataset.

GLC-L transformation + MLP do not seem to perform well together, producing results which vary dramatically between different runs. Looking at Table 8 in columns labeled “Swiss 3-D” and “Wisconsin,” a dramatic difference can be seen between the runs. “Swiss 3-D” has 2 runs with classification accuracy of 36.87% and one with 90.62%. Such dramatic difference between runs is not normal. I think it is because a simple MLP classifier is not being able to learn the parse number of active pixels in the artificially created images.

Comparison of the results between the MLP and the GLC-L transform + MLP is presented in Table 6. The raw data with MLP performed better than the GLC-L transformation + MLP with each dataset. Swiss Roll 2-D dataset is the only dataset which did not have a dramatic decrease in classification accuracy between the two methods. MLP produced 97.50% while the GLC-L transformation + MLP produced 95.87% . All the other datasets had a dramatic decrease in classification accuracy with the GLC-L transformation + MLP.

SVM Experiments

For this set of experiments a SVM was used as the machine learning algorithm for classification of numerical and transformed data. The SVM classifier from scikit-learn Python library was used with all of the default hyperparameters. The default kernel is “rbf”.

TABLE 8: SVM Result Comparison

Dataset Name	SVM (%)	Transform+SVM (%)
Swiss Roll 2-D	98.50	96.43
Swiss Roll 3-D	98.25	70.24
Breast Cancer Wisconsin	96.33	95.91
Red Wine	56.66	52.31
White Wine	51.53	44.98
Diabetic Retinopathy Debrecen	61.25	59.21
MNIST-Subset	98.76	84.79

Notes: Comparison of results for raw input data with SVM and artificially created images with SVM. The table is sorted with respect to number of attributes in the dataset.

TABLE 9: Transform+SVM Runs

Run #	Swiss 2-D (%)	Swiss 3-D (%)	Wisconsin (%)	Red (%)	White (%)	Diabetic (%)	MNIST (%)
1	97.81	74.68	94.89	50.31	45.40	63.91	86.50
2	95.62	61.87	97.08	53.12	44.50	53.47	86.00
3	97.18	70.00	94.16	48.75	46.12	66.08	84.33
4	97.81	69.68	97.08	56.56	44.38	53.47	82.16
5	93.75	75.00	96.35	52.81	44.50	59.13	85.00
Average	96.43	70.24	95.91	52.31	44.98	59.21	84.79

Notes: Results for each of the 5 runs with the GLC-L Transform and SVM. The average results are reported in Table 5. Names of the datasets have been abbreviated.

Column 1 in Table 8 has the data set names, column 2 has the results for the raw input data with SVM. Column 3 contains the experimental results of transforming numerical data with GLC-L and classifying images with a SVM. The results for “SVM” are gathered using 10-fold cross validation, and the results for “Transform + SVM” are an average of 5 runs using the data transformation system where 10-fold cross validation is used for training the SVM.

Table 9 contains the results for each of the 5 runs of the data transformation with GLC-L and SVM classification of artificially created images. Names of the datasets have

been abbreviated. The average of the 5 runs is reported for comparison in Table 8. The datasets are arranged left to right with respect to the number of dimensions in the dataset.

SVM on the raw data performs better than GLC-L transformation + SVM for all of the datasets in the experiments. SVM produces 98.50% on the raw Swiss Roll 2-D dataset, while GLC-L transformation + SVM produces 96.43%. The drop in accuracy isn't as significant as the drop between the two methods for the Swiss Roll 3-D dataset. The raw numerical data with SVM gets 98.25% compared to 70.24% with the GLC-L transformation + SVM for the Swiss Roll 3-D dataset. Breast Cancer Wisconsin and Diabetic Retinopathy Debrecen datasets get similar classification accuracies with both methods.

Comparison of CNN, MLP and SVM

Table 10 contains comparisons of results between all of the algorithms described in this chapter: CNN, Transform+CNN, MLP, Transform+MLP, SVM, and Transform+SVM. The highest accuracy is highlighted for each of the datasets. Three out of the seven datasets produced highest classification accuracy with GLC-L transformation + CNN. MLP produced the highest accuracy for the raw Swiss Roll 3-D dataset and SVM performed best on the raw Swiss Roll 2-D dataset. 1-D CNN performed best on the Diabetic Retinopathy Debrecen dataset, and the regular CNN performed best on the original MNIST-Subset.

TABLE 10: CNN, MLP, and SVM Result Comparison

Dataset Name	1-D CNN (%)	T+CNN (%)	MLP (%)	T+MLP (%)	SVM (%)	T+SVM (%)
Swiss 2-D	72.50	97.43	97.50	95.87	98.50	96.43
Swiss 3-D	96.18	97.55	98.43	63.87	98.25	70.24
Wisconsin	96.92	97.22	94.83	89.36	96.33	95.91
Red	60.65	60.93	60.78	46.13	56.66	52.31
White	55.91	60.76	53.57	46.03	51.53	44.98
Diabetic	73.75	69.04	69.84	53.47	61.25	59.21
MNIST(CNN)	99.53	92.93	99.10	83.29	98.76	84.79

Notes: Transform has been abbreviated to “T”. Comparison of results between CNN, Transform+CNN, MLP, Transform+MLP, SVM, and Transform+SVM. The highest accuracy for each dataset has been highlighted.

CHAPTER VI

CONCLUSION

In this thesis, Linear General Line Coordinates (GLC-L) are described along with their potential. GLC-L can losslessly visualize n-dimensional data on 2-D graphs. An application of GLC-L as a data transformation method is presented. Results are gathered with various machine learning algorithms to evaluate the artificially created images for classification. Datasets in varying number of instances and attributes are used in the experiments. The method is tested on data up to 484 dimensions, showing that it is scalable.

Experimental results show that a lossless visualization method for n-dimensional data can be used as a data transformation method, by visualizing one data sample at a time with GLC-L and capturing the visualizations as images. No loss of information is done by the transformation. These artificially created images contain useful information which can be used for classification of data. Classification of artificially created images can be automated with a machine learning algorithm such as a CNN, MLP, and SVM.

CNN outperformed the other methods in classifying the artificially created images. CNN is designed for relational data such as images, which explains why it performed better than MLP and SVM. Showing that a CNN can correctly classify images with such sparse number of active pixels is useful for further studies.

Further experiments can be done to improve the classification results of the GLC-L transformation. The random search for coefficients can be redesigned to a structured search for finding the optimal angles for the visualization. The architecture of the CNN can be optimized to better suit the sparse number of active pixels in the artificially created images. The hyperparameters of the SVM and MLP can also be optimized.

The ability to classify artificially created images created by a visualization method with a CNN raises many interesting questions. Perhaps there are other visualization methods which can be used to transform data and produce better classification accuracy with machine learning algorithms.

REFERENCES CITED

- [1] B. Kovalerchuk and D. Dovhalets, “Constructing Interactive Visual Classification, Clustering and Dimension Reduction Models for n-D Data,” in *Informatics*, vol. 4, p. 23, Multidisciplinary Digital Publishing Institute, 2017.
- [2] V. Grishin and B. Kovalerchuk, “Multidimensional Collaborative Lossless Visualization: Experimental Study,” in *International Conference on Cooperative Design, Visualization and Engineering*, pp. 27–35, Springer, 2014.
- [3] C. Cortes and V. Vapnik, “Support-Vector Networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [4] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford university press, 1995.
- [5] R.-F. Chang, W.-J. Wu, W. K. Moon, Y.-H. Chou, and D.-R. Chen, “Support Vector Machines for Diagnosis of Breast Tumors on US Images,” *Academic radiology*, vol. 10, no. 2, pp. 189–197, 2003.
- [6] J. Lemley, S. Abdul-Wahid, D. Banik, and R. Andonie, “Comparison of Recent Machine Learning Techniques for Gender Recognition from Facial Images,” 2016.
- [7] Y. LeCun, K. Kavukcuoglu, and C. Farabet, “Convolutional Networks and Applications in Vision,” in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pp. 253–256, IEEE, 2010.
- [8] O. M. Parkhi, A. Vedaldi, A. Zisserman, *et al.*, “Deep Face Recognition,” in *BMVC*, vol. 1, p. 6, 2015.
- [9] M. Peng, C. Wang, T. Chen, and G. Liu, “NIRFaceNet: A Convolutional Neural Network for Near-Infrared Face Identification,” *Information*, vol. 7, no. 4, p. 61, 2016.
- [10] K. Jarrett, K. Kavukcuoglu, Y. LeCun, *et al.*, “What is the Best Multi-Stage Architecture for Object Recognition?,” in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 2146–2153, IEEE, 2009.
- [11] S. Hijazi, R. Kumar, and C. Rowen, “Using Convolutional Neural Networks for Image Recognition,” *Cadence Design Systems Inc.: San Jose, CA, USA*, 2015.
- [12] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

- [13] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” *arXiv preprint arXiv:1207.0580*, 2012.
- [14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.
- [15] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [16] B. Kovalerchuk and V. Grishin, “Adjustable general line coordinates for visual knowledge discovery in n-d data,” *Information Visualization*, 2017.
- [17] P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss, “The FERET Evaluation Methodology for Face-Recognition Algorithms,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1090–1104, 2000.
- [18] R. Kohavi *et al.*, “A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection,” in *Ijcai*, vol. 14, pp. 1137–1145, Montreal, Canada, 1995.
- [19] B. Antal and A. Hajdu, “An ensemble-based system for automatic screening of diabetic retinopathy,” *Knowledge-based systems*, vol. 60, pp. 20–27, 2014.
- [20] J. Bergstra and Y. Bengio, “Random Search for Hyper-Parameter Optimization,” *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.
- [21] B. Kovalerchuk, *Visual Knowledge Discovery and Machine Learning*, vol. 144. Springer, 2018.
- [22] F. Chollet *et al.*, “Keras.” <https://github.com/fchollet/keras>, 2015.
- [23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [24] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [25] B. Xu, N. Wang, T. Chen, and M. Li, “Empirical Evaluation of Rectified Activations in Convolutional Network,” *arXiv preprint arXiv:1505.00853*, 2015.
- [26] D. Dheeru and E. Karra Taniskidou, “UCI Machine Learning Repository,” 2017.

- [27] D. Surendrani, “Swiss Roll Dataset,” 2014.
- [28] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, “Modeling wine preferences by data mining from physicochemical properties,” *Decision Support Systems*, vol. 47, no. 4, pp. 547–553, 2009.
- [29] Y. LeCun and C. Cortes, “MNIST handwritten digit database,” 2010.

APPENDIX

DATASETS

In this chapter the datasets used for experiments are explained, along with overview of the datasets and preprocessing done to them. These datasets vary in number of classes, number of instances, and number of attributes per instance. A few of these datasets are from the UCI Machine Learning Repository [26]. Table 11 contains a quick glance at the datasets.

TABLE 11: Overview of Datasets

Dataset Name	Instances	Attributes	Classes
Swiss Roll 2-D	1600	2	4
Swiss Roll 3-D	1600	3	4
Breast Cancer Wisconsin	683	9	2
Red Wine	1599	11	6
White Wine	4898	11	7
Diabetic Retinopathy Debrecen	1151	18	2
MNIST-Subset	3000	484	3

Notes: Overview of the datasets used for experiments. Table includes the number of samples, attributes, and classes in each dataset. The table was organized from lowest to highest with regards to number of attributes.

Swiss Roll Dataset

This is an artificially created dataset to test various dimensionality reduction algorithms. This dataset has a total of 1600 instances equally spread out among 4 classes. There is two subsets of the Swiss Roll Dataset, 2-D and 3-D [27]. The labels column was joined with the data columns, being added as the 3rd column for 2-D data, and 4th column for 3-D data. The data comes normalized and no further preprocessing was done to it.

2-D Manifold

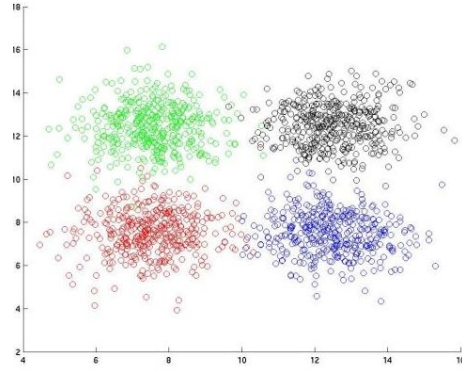


FIGURE 20: Swiss Roll Dataset 2-D [27]

3-D Manifold

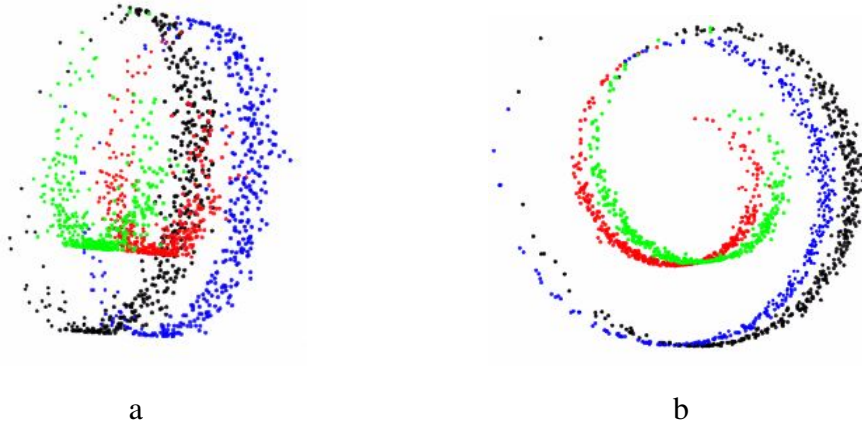
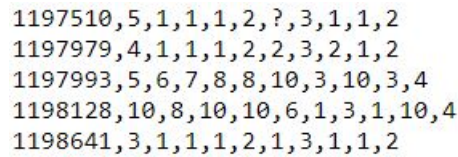


FIGURE 21: Swiss Roll Dataset 3-D, screen shots taken from 3-D visualization on the dataset website [27].

Swiss Roll 3-D was converted from the 2-D data by mapping $(x, y) = (x \cos x, y, x \sin x)$. Fig. 21 contains two screen shots in 3-D, taken from the dataset website.

Wisconsin Breast Cancer

Wisconsin Breast Cancer dataset is from the UCI machine learning repository [26]. The dataset consists of 699 instances with 11 attributes. In the preprocessing step instances with missing values “?” were removed from the dataset, resulting in 683 instances of which 444 were benign cases and 239 were malignant cases. Patient ids were also removed from the attributes, resulting in 10 attributes with one being the class label. The dataset is originally normalized between 1 and 10, no other normalization was done.



```
1197510,5,1,1,1,2,?,3,1,1,2
1197979,4,1,1,1,2,2,3,2,1,2
1197993,5,6,7,8,8,10,3,10,3,4
1198128,10,8,10,10,6,1,3,1,10,4
1198641,3,1,1,1,2,1,3,1,1,2
```

FIGURE 22: Wisconsin Breast Cancer dataset. First column contains the patient ids which were removed in the preprocessing step.

Fig. 22 contains a screen shot of the Wisconsin Breast Cancer dataset. In the first row a “?” value can be seen in attribute 7. Rows with “?” values were removed from the dataset in the preprocessing step.

Wine Quality

This dataset is also from the UCI machine learning repository [26]. Wine quality dataset [28] is composed of two different datasets, red wine and white wine. Red wine quality subset has 1599 instances and the white wine quality subset has 4898 instances. Both of the subsets have 11 attributes followed by the class label. Red wine quality has 6 different classes, but it is very unbalanced with most of the instances being only in 2 classes. White wine quality dataset has 7 classes and it is also very unbalanced. Both of the subsets were normalized between 0 and 1 columnwise.

Red Wine

6.2;0.56;0.09;1.7;0.053;24;32;0.99402;3.54;0.6;11.3;5
6.1;0.715;0.1;2.6;0.053;13;27;0.99362;3.57;0.5;11.9;5
6.2;0.46;0.29;2.1;0.074;32;98;0.99578;3.33;0.62;9.8;5
6.7;0.32;0.44;2.4;0.061;24;34;0.99484;3.29;0.8;11.6;7
7.2;0.39;0.44;2.6;0.066;22;48;0.99494;3.3;0.84;11.5;6

FIGURE 23: Red Wine Quality dataset before any preprocessing.

Fig. 23 contains a screen shot of the Red Wine Quality dataset. The data had to be normalized to the large range in values between different attributes (columns). Attribute number 6 has values below 1, while attribute number 7 had values larger than 50. Such range of values would greatly distort the visualization.

White Wine

The White Wine Quality dataset is similar in structure to the Red Wine Quality with only 2 differences. The first difference is that it has many more instances , 4898 instances as opposed to 1599 in the Red Wine Quality data. And the other difference being it has an extra class label.

Diabetic Rectionopathy Debrecen

Diabetic Rectionopathy Debrecen dataset [19] is from the UCI machine learning repository [26]. This dataset has labels for two different classes, one of the classes having signs of Diabetic Rectionopathy (DR) and the other one which shows no signs of DR. This data set has 1151 instances with 18 attributes. The classes are balanced with 540 instances with no signs of DR and 611 instances with signs of DR. The dataset was normalized columnwise between 0 and 1. Originally the dataset had 19 attributes, excluding the label class, but attribute number 19 was removed because it was the result of binary classification provided from the author.

```

1,1,11,11,11,11,11,8,31.657567,15.759608,2.239425,0.207526,0,0,0,0,0.52008,0.097052,0,0
1,1,6,6,6,6,5,5,67.222908,5.2789,1.201355,0.148482,0.01246,0,0,0,0.562787,0.104872,0,1
1,0,25,23,23,21,20,14,10.902544,7.388796,0.204684,0,0,0,0,0.556941,0.093038,1,1
1,1,34,34,34,32,26,17,18.248935,10.179572,1.431258,0.095209,0,0,0,0.569702,0.11706,0,0
1,1,20,20,20,19,16,14,8.342012,1.374588,0.129534,0,0,0,0,0.530497,0.111247,1,0
1,1,51,51,51,50,47,43,17.525446,5.303288,0.653304,0,0,0,0,0.565561,0.099917,0,0

```

FIGURE 24: Diabetic Rectionopathy Debrecen dataset.

Fig. 24 contains a screen shot of the Diabetic Rectionopathy Debrecen dataset before preprocessing. It can be seen that the data is not normalized with some of the columns having value range between 0 and 1, while others range between 0 and over 100. The last two columns contain labels, one being the data label. The other one as a result of prediction from the experiments gathered by the author of the dataset.

MNIST-Subset

In order to test the data transform method on data of high dimension, a subset of Modified National Institute of Standards and Technology (MNIST) Database [29] was used. MNIST Database contains images of digits 0 through 9. The subset consists of 1000 instances from the digits 0, 1, and 2, totaling 3000 instances altogether. The original images are 28x28 pixels (784 dimensions). The images are centered with black padding around them. In the preprocessing step, the padding was removed by cropping the images to 80% of the original size. After preprocessing, the images were 22x22 (484 dimensions).



FIGURE 25: Images from the MNIST-Subset. One image from each class was randomly selected. These examples are after cropping has been applied to remove extra black padding, resulting in 22x22 (484 dimensions).