2023-03

ON EUCLIDEAN NETWORKS FOR IMPROVING CLASSIFICATION ACCURACY

Slaughter, Jacob W.

Monterey, CA; Naval Postgraduate School

https://hdl.handle.net/10945/72055

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ON EUCLIDEAN NETWORKS FOR IMPROVING CLASSIFICATION ACCURACY

by

Jacob W. Slaughter

March 2023

Thesis Advisor:	Armon C. Barton
Second Reader:	Marko Orescanin

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC, 20503.

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** March 2023	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE ON EUCLIDEAN NETWORKS FOR IMPROVING CLASSIFICATION ACCURACY	**5. FUNDING NUMBERS**
6. AUTHOR(S) Jacob W. Slaughter	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Machine learning is found in nearly every facet of daily life. Large amounts of data are required but not always available for specific problems, precluding the use of advanced methods such as deep learning and convolutional neural networks. The Euclidean Network (EN) can be used to mitigate these issues. The EN was thoroughly tested to prove its viability as a classification algorithm and that its methods may be used to augment data and transform the input data to increase its feature space dimensionality. Originally, it was hypothesized that the EN could be used to synthetically generate data to augment a data set, though this method was proven to be ineffective. The next area of research sought to expand the dimensionality of the input feature space to improve performance with additional classifiers. This area showed positive results, which supported the hypothesis that more complex, dense input would give algorithms more insight into the data and improve performance. The EN has been found to perform exceptionally well as an independent classifier, as it achieved the highest accuracy for 12 of the 21 data sets. For the remaining 9, though it did not have the highest accuracy, the EN performed comparably to more sophisticated algorithms. The EN also proved capable to expand a data set's feature space to further improve performance. This tactic provided a more robust classification technique and saw an average increase in accuracy of 3% between all data sets.

14. SUBJECT TERMS EN, Euclidean Network, machine learning, classification	**15. NUMBER OF PAGES** 69
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

ON EUCLIDEAN NETWORKS FOR IMPROVING CLASSIFICATION ACCURACY

Jacob W. Slaughter
Captain, United States Marine Corps
BSME, Virginia Polytechnic Institute and State University, 2015

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 2023

Approved by: Armon C. Barton
Advisor

Marko Orescanin
Second Reader

Gurminder Singh
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Machine learning is found in nearly every facet of daily life. Large amounts of data are required but not always available for specific problems, precluding the use of advanced methods such as deep learning and convolutional neural networks. The Euclidean Network (EN) can be used to mitigate these issues. The EN was thoroughly tested to prove its viability as a classification algorithm and that its methods may be used to augment data and transform the input data to increase its feature space dimensionality. Originally, it was hypothesized that the EN could be used to synthetically generate data to augment a data set, though this method was proven to be ineffective. The next area of research sought to expand the dimensionality of the input feature space to improve performance with additional classifiers. This area showed positive results, which supported the hypothesis that more complex, dense input would give algorithms more insight into the data and improve performance. The EN has been found to perform exceptionally well as an independent classifier, as it achieved the highest accuracy for 12 of the 21 data sets. For the remaining 9, though it did not have the highest accuracy, the EN performed comparably to more sophisticated algorithms. The EN also proved capable to expand a data set's feature space to further improve performance. This tactic provided a more robust classification technique and saw an average increase in accuracy of 3% between all data sets.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

CNN	convolutional neural network
EN	Euclidean Network
GPU	graphics processing unit
KNN	k-nearest neighbors
RBF	radial basis function
RF	random forest
SVM	support vector machine

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

1.1 Purpose

The proliferation of machine learning has led to more advanced and sophisticated learning algorithms, specifically neural networks. This work discusses a new type of network that utilizes an alternate approach to the traditional perceptron used in traditional neural networks. As explained in [1], a typical neural network calculates the hidden layer values via the dot product of the input and weights. The Euclidean Network (EN), however, calculates the hidden layer values with the squared differences of the input and weights. This work demonstrates that the EN shows exceptional performance in classification tasks, but also discusses additional uses for the algorithm to alter the input data to improve performance. Two methods used to alter the data were synthetic point generation and feature space expansion. The research discussed in this thesis showed that the means by which additional points are synthetically generated was not conducive to performance, but upsampling the feature space proved to be an effective method to improve classification accuracy. Further research with the EN can provide valuable insight into neural network weight calculations to improve classification accuracy. This research is also significant because it utilizes feature space transformations to increase a data set's dimensionality. This transformation can expand the problem sets for machine learning and enable more complex algorithms to benefit from an expanded feature space.

The following three research questions provided the impetus for this research:

1. Can the EN perform well as an independent classifier?
2. What are the effects of synthetic point generation?
3. Can higher dimensional feature space improve accuracy?

1.2 Research Questions and Hypotheses

The order of this work followed three specific areas of interest: performance of the EN as an independent classifier; effects of data augmentation with synthetically generated points from the EN's hidden layer; and transformation of the input data into higher dimensional space and how it affects classification accuracy.

1.2.1 EN as an Independent Classifier

The first set of experiments were performed under the hypothesis that the EN would outperform other distance-based approaches for classification problems. The goal of these tests was to determine whether the accuracy results are comparable to or greater than the results of existing classifiers. Additionally, they sought to identify whether the EN performs better or worse for specific data set characteristics or types. Multiple classification models were trained of various types for each data set used in this research. To respond to this research question, the EN was trained on the same data sets to compare performance metrics.

1.2.2 Effects of Synthetic Point Generation

Secondly, this work discusses the effects of data augmentation and whether multiple iterations of synthetic point generation improve performance when used to train another EN or alternate classifier. Synthetic data points were generated with the EN and combined with the original data set to create a new, hybrid data set. The hypothesis was that classification accuracy would increase when the hybrid data set was used as the input data for an untrained EN or other classifier. A method built into the EN allows data points to be extracted and plotted in the same feature space as the input data. These data points are used to augment the existing data set. This augmented data set was then used to retrain multiple classification models. The results of the augmented data were compared with the results of the raw input data to conclude the effects of synthetic point generation.

1.2.3 Effects of Increased Dimensionality

Finally, the effects of feature space expansion were thoroughly tested to discover if higher dimensional data improved classification accuracy. These tests were performed under the hypothesis that models would benefit from input data that had an expanded feature space.

A specific focus was given to identify whether data sets with a limited number of features benefited from this upsampling to give algorithms more information from which to extract useful data patterns to improve performance. As with synthetic point generation, the EN contains a built-in method that produces higher dimensional data from the input data set. The new, expanded data set was used to train multiple classifiers to discern its effects on accuracy when compared to the raw data set results.

1.3 Chapter Summary

Chapter 2 discusses the background of the research and other work related to applicable methods used with the EN. Chapter 3 describes the EN's derivation and functionality. It also discusses the data acquisition process and how the EN was applied to perform classification, synthetic point generation, and feature space transformation. Chapter 4 contains the results from each experiment and provides metrics with accuracy percentages and comparisons. Finally, Chapter 5 contains the discussion of results and drawn conclusions, as well as potential future tasks to continue the research.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2: Background and Related Work

This chapter provides background information on machine learning, specifically classification tasks. The EN utilizes concepts from traditional classification approaches, which are described in Section 2.1. These concepts include the method of classification, training process, logistic regression, neural networks, and other classification algorithms used for comparison with the EN.

Additionally, this chapter compares the applicable research questions with similar areas of work. No published research was identified with regard to altering the calculation of a neural network's hidden layer, affecting its backpropagation and gradient calculations. However, much research has been conducted in the realm of data augmentation and feature engineering, both of which are applications for which the EN was tested.

2.1 Machine Learning

Géron [1] described machine learning as the art and science of "programming computers so they can learn from data." A plethora of data is needed to effectively train machine learning algorithms. This potential issue is compounded for supervised learning algorithms, where that data must have accurate labels. Acquiring and labeling data can be cumbersome, but it is necessary so that the appropriate amount of data can be provided to teach an algorithm the desired outcomes.

Classification is a common task performed with supervised learning, wherein an algorithm is trained on input data that contains a set of features that correspond to a label, or class. The model is trained to associate feature values and relationships with a specific class. This thesis utilized multiclass classification via a one-vs-all (or one-vs-rest) method. This method works by creating a binary classifier for each class. For example, if a model is trained to classify vehicles with labels: sedan, truck, van, it will create a classifier for sedan, truck, and van, independently. In [1], Géron described that once the model is trained and test data is used, the resultant prediction from the model will be the output with the highest score of the three classifiers. This score is calculated with the model's parameters, which are tuned using a method call gradient descent.

Gradient descent is used during training to optimize the parameters of a model. The objective of gradient descent is to minimize the cost function by iterating through parameter values that are often randomly initialized. The minimum is identified by a derivative (gradient) that is equal zero. As described in [1], when the gradient is zero, a minimum has been found, but additional steps are required to ensure that training does not converge to local minima, but instead continues to find the global minimum to produce the best-trained model. There are multiple types of gradient descent, but for the purposes of this thesis, stochastic gradient descent is used. This method uses random instances of the data to compute the gradient. Therefore, it is capable of training larger data sets, since only one instance needs to be saved in memory for each training iteration. Géron [1] stated that the stochastic method also produces a more irregular descent towards optimization and never quite reaches *the* optimal solution though, on average it will produce results very close to it.

A hyperparameter to control the rate of the descent is learning rate. Learning rate controls how quickly (or slowly) the model trains. A high learning rate may skip over desired minima and cause the model to diverge. Alternatively, it is warned in [1] that if the learning rate is too small the model can take much longer to converge and it may get stuck in a local minimum, failing to find the optimal parameter values.

In addition to selecting an appropriate learning rate, one must be aware of overfitting or underfitting throughout the training process. In [1], overfitting is said to result from a model that may be too complex for the data and train specifically to the training set. This results in a high training accuracy score, but the model will fail to generalize well, which will result in poor results beyond the training set. Alternatively, a model that underfits is too simple for the data and will not produce accurate predictions. It is often more favorable to work with a model that initially overfits the data. This is because regularization techniques and other tuning can be done to improve the model. An overfitting model is evidence that the data can be learned and that there are patterns to be discerned for predictions. However, underfitting models provide little insight to analyzing the data beyond the application of more complex models or data.

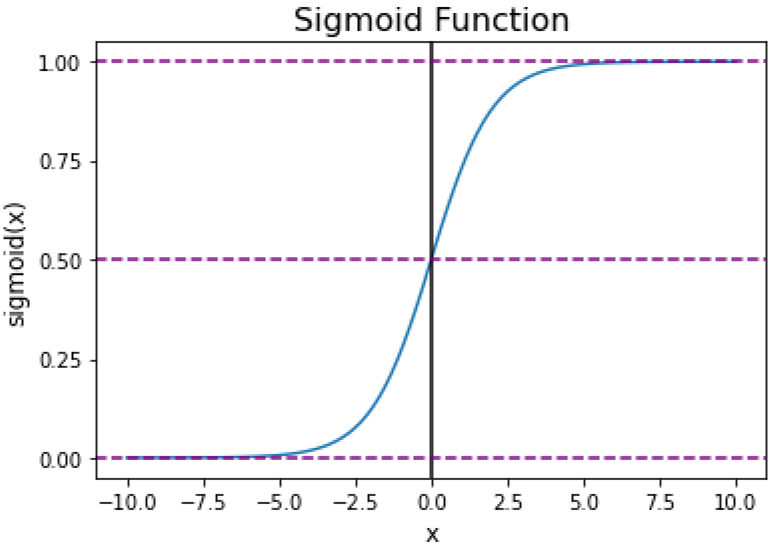

Figure 2.1. The sigmoid function forces values to be between 0 and 1, allowing them to be treated as probabilities. These probabilities are then used to predict the class label for a given input.

2.1.1 Logistic Regression

The EN borrows concepts from logistic regression notably, the log loss and sigmoid functions. These functions are discussed in depth in Section 3.1.2. Logistic regression works by computing a probability for a particular class instance. This probability comes from the sigmoid function (Figure 2.1), since all output values will be between 0 and 1. As described in [1], the class prediction depends on if the output of the sigmoid is less than or greater than 0.5. This break occurs where $x = 0$, so if the input is positive, logistic regression will predict 1 and 0 if the input is negative. The EN also utilizes the sigmoid function to form its predictions in a similar manner.

Logistic regression makes predictions with the sigmoid function, but it is trained using the log loss function. The main concept is that this function outputs a higher cost for incorrect label predictions and a near zero cost for correct predictions. As explained in [2], the log loss function is convex, so gradient descent can be used to find the global minimum . The EN utilizes the same log loss function for its forward pass during training only.

2.1.2 Neural Networks

The simplest example of an artificial neural network is the perceptron. The perceptron calculates the weighted sum of the inputs, then applies some activation function to that sum to get an output. A multilayer perceptron expands this idea by stacking layers of perceptrons to produce a network consisting of an input layer, hidden layer(s), and output layer. This architecture is discussed further in Section 3.1.1. Typically, a softmax layer is added to the output layer for purposes of classification. Each possible class label will have its own output neuron and the softmax activation function will ensure that the probabilities of each neuron add up to 1. Multiclass classification, as described in [1] is performed by predicting the class label with the highest probability.

A major strength of a dense neural network is its use of backpropagation during training. Backpropagation allows the network to adjust each of its parameters based on the error from the desired outcome. Géron [1] summarizes the backpropagation algorithm as follows: the network's forward pass makes a prediction, which is then used to calculate the error; the algorithm then goes backwards through the network to measure the contribution to that error from each connection. It then adjusts the weights to reduce the error via gradient descent. Backpropagation relies on the chain rule of calculus to propagate the error through each connection in terms of the respective parameters of that layer. The EN conducts its forward and backward pass in a similar fashion to a fully connected neural network, which are shown in detail in Sections 3.1.1 and 3.1.2.

A more advanced application of a dense neural network is the CNN. CNNs work by introducing convolutional layers to the hidden layers of a neural network. As described in [3], a filter, or kernel is used to slide across the input, convolve the features in the receptive field, and produce an output of a new size, based on the kernel size. Multiple convolutional layers can be stacked to extract more information from low-level features so that they become larger in the next layer. This process enables the CNN to learn more information from a given input than a typical classifier. A benefit of CNNs is their variety of architectures and the ability to fine tune a model for very specific problem sets. Many architectures exist, but for the purposes of this thesis, a simple architecture (two to three convolutional layers) was used.

2.1.3 Additional Classification Algorithms

Throughout this thesis, a number of classifiers were used for comparison in addition to the previously described logistic regression and neural network algorithms. These algorithms include, linear SVM, Gaussian RBF kernel SVM, k-nearest neighbors (KNN), and random forest. The linear SVM algorithm attempts to separate data points within a certain feature space with a line, fitting the widest margin possible between instances of the classes. The linear SVM can also take advantage of different kernels, as seen with the Gaussian RBF kernel SVM. This SVM uses the kernel trick to improve performance on data that is not linearly separable. In [1], the kernel trick is explained to improve performance by creating additional similarity features between instances. With these added features, the data may become linearly separable to achieve high classification accuracy. Another algorithm that checks for instance similarity is KNN. The hyperparameter, k is used to tell the algorithm how many surrounding instances to consider for each point. KNN classifies an instance with a plurality vote between those surrounding points. Effectively, points that are closer to other points of a particular class are labeled as that respective class. The next algorithm tested against the EN was a random forest classifier. A random forest operates as an ensemble of decision trees. According to Géron [1], these decision trees use attributes of specific features to create nodes that branch through additional features until they reach a classification node. The random forest has an advantage over decision trees due to its ensemble nature and introduction of randomness, which results in greater tree diversity to produce a higher-performing model.

2.2 Data Augmentation

Data augmentation can be used as a regularization technique by adding realistic variants of the data to the training set. In [4], it is stated that affine transformations such as, rotation, translation, and scaling can be used for image classification to force a model to learn additional variations of acceptable images, while preserving labels. This method easily augments image data by creating additional samples of specified classes and helps the model accurately classify test samples given its more robust training set.

Schlüter and Grill [5] applied a similar process to sound data, where input data was augmented by applying random frequency filters and shifting the pitch. An issue with the previous two examples is that they are both domain specific. For example, rotation and cropping can only be applied in the image domain. Another approach suggested a process that was agnostic to the domain from which the data came. DeVries and Taylor's [6] approach utilized a system of encoders to transform encoded data in the feature space, rather than the raw input to improve model performance.

This thesis also used a data agnostic approach, as synthetic points were generated with the first layer of weights once the network had been fully trained. Instead of altering known samples for more data, the EN would create new data points based on existing data points' weights with their respective labels.

2.3 Feature Engineering

Much research has been conducted on methods of feature engineering. A large part of feature engineering occurs in the data acquisition and cleaning process. As described in [7], this is where important features are identified and thus chosen to be used for training models. This process, feature selection, goes hand-in-hand with feature extraction, where certain features are combined into more useful training data. All of these methods involve reducing the feature space, whereas the EN seeks to improve classification accuracy by increasing the feature space. An area of this research was conducted in [8] for reinforcement learning models, where it was examined that higher dimensional input data allowed learning algorithms to extract more complex functions and states, which improved training. This idea is reflected in the goal of the EN to increase input dimensionality.

Additionally, research in [4] was conducted to augment data sets within their feature space. Though this thesis does not necessarily augment the data with its feature space extractions, it does use the activations as an independent data set to train new models. As discussed by Wong and Gatt [4], augmentation within the feature space proved to be beneficial due to the activations' ability to "extract salient information on which the classifier performs learning." This observation supports the hypothesis that higher dimensional data would enable more complex algorithms to extract previously hidden, underlying information that can be used to improve classification performance.

CHAPTER 3:
Methodology

3.1 Euclidean Network Design

3.1.1 Contrast to Conventional Neural Networks

Before delving into the specifics of the EN, a brief overview of a conventional neural network may be helpful. A simple, single layer neural network is shown in Figure 3.1. The input, X is passed to the network, which calculates the hidden layer using weights, W. From there, the hidden layer, z_h is activated using a sigmoid function to produce a_h. Those values are then used with the hidden weights, W_h to produce the output node, z_{out}. That final node is then activated to produce the output or prediction.

Traditional neural networks calculate the hidden layer using Equation 3.1 and the desired activation function. The subscript h denotes the hidden layer. The node denoted by z_{out} in Figure 3.1 is similarly calculated, substituting a_h for X and W_h for W as seen in Equation 3.2.

$$z_h = W^T X \tag{3.1}$$

$$z_{out} = W_h^T a_h \tag{3.2}$$

The EN differs in its calculation of the hidden layer. As shown in Equation 3.3, the weights are subtracted from the input rather than multiplied, and the difference is squared. This causes the weights to fit the data differently than a NN by converging in and around data points within the data manifold during training.

$$z_h = (X - W)^2 \tag{3.3}$$

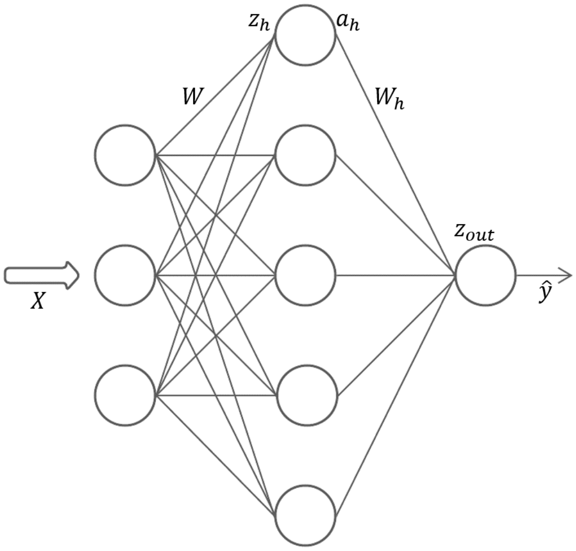

Figure 3.1. A simple neural network with three input nodes that receive the input data X, five hidden nodes, and a single output. Subscript h denotes elements of the middle, hidden layer.

Another key difference between the EN and a typical neural network is how it makes its predictions. A conventional neural network may use a softmax function at the output to produce a probability distribution for each class. With this method, the class with the highest probability is used as the network's prediction. However, the EN creates distinct binary classification networks for each class, which it uses to perform multiclass classification via a one-vs-all technique. The weights for each network of the EN exist in the same space as the input data and are updated to be further or closer to the input points as the network is trained. This process creates a push and pull dynamic between input data points and network weights.

After training, the EN acts similar to a distance measurement function that measures distance between a sample input and the EN's weights. During test time, a given point will be passed through each network. The network that produces the lowest value will denote the class prediction.

3.1.2 Gradient Derivation

Familiar Functions and Forward Pass

The EN uses two functions that are common in classification algorithms. These two functions are the logistic regression loss function (Equation 3.4) and the sigmoid activation function, along with its derivative (Equation 3.5).

$$J = \begin{cases} -log(1-\hat{y}), & \text{if } y == 0 \\ -log(\hat{y}), & \text{if } y == 1 \end{cases} \tag{3.4}$$

$$\begin{aligned} \sigma(x) &= \frac{1}{1+e^{-x}} \\ \sigma'(x) &= \sigma(x)\cdot(1-\sigma(x)) \end{aligned} \tag{3.5}$$

Each network instantiated by the EN computes its forward pass with the calculations found in Equation 3.6, with the same naming convention seen in Figure 3.1.

$$\begin{aligned} z_h &= (X-W)^2 \\ a_h &= \sigma(z_h) \\ z_{out} &= (a_h - W_h)^2 \\ \hat{y} &= \sigma(z_{out}) \end{aligned} \tag{3.6}$$

Each set of weights (W and W_h) is updated using the learning rate hyperparameter and the partial derivatives of the loss function, depending on the data point's true label. The following equations display the calculations for the partial derivatives of each set of weights and for each true label. The only possible labels are 0 and 1 since the EN uses a one-vs-all technique.

Backpropagation

The variable J represents the loss, which is minimized with respect to the weights of the corresponding layer. The below equations show the calculations for these partial derivatives ($\frac{\delta J}{\delta W_h}$ and $\frac{\delta J}{\delta W}$) for each label value (0 or 1).

If the class label is 0 ($y == 0$), W_h is updated with Equation 3.7. The calculations for each partial derivative are shown in Equation 3.8.

$$\frac{\delta J}{\delta W_h} = \frac{\delta J}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta z_{out}} \frac{\delta z_{out}}{\delta W_h} \tag{3.7}$$

$$\frac{\delta J}{\delta W_h} = \frac{1}{ln(10)(1-\hat{y})} \cdot \sigma'(z_{out}) \cdot (-2(a_h - W_h)) \tag{3.8}$$

Continuing with a true label of 0, the set of weights W is updated using Equation 3.9, expanded in Equation 3.10.

$$\frac{\delta J}{\delta W} = \frac{\delta J}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta z_{out}} \frac{\delta z_{out}}{\delta a_h} \frac{\delta a_h}{\delta z_h} \frac{\delta z_h}{\delta W} \tag{3.9}$$

$$\frac{\delta J}{\delta W} = \frac{1}{ln(10)(1-\hat{y})} \cdot \sigma'(z_{out}) \cdot (-2(a_h - W_h)) \cdot \sigma'(z_h) \cdot (-2(X - W)) \tag{3.10}$$

Next we calculate the partial derivatives if the label is 1. The set of weights W_h is updated with Equation 3.11 with the substitutions shown in Equation 3.12.

$$\frac{\delta J}{\delta W_h} = \frac{\delta J}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta z_{out}} \frac{\delta z_{out}}{\delta W_h} \tag{3.11}$$

$$\frac{\delta J}{\delta W_h} = -\frac{1}{\hat{y} \cdot ln(10)} \cdot \sigma'(z_{out}) \cdot (-2(a_h - W_h)) \tag{3.12}$$

Finally, when $y == 1$, the set of weights W is updated using Equation 3.13, expanded in Equation 3.14.

$$\frac{\delta J}{\delta W} = \frac{\delta J}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta z_{out}} \frac{\delta z_{out}}{\delta a_h} \frac{\delta a_h}{\delta z_h} \frac{\delta z_h}{\delta W} \tag{3.13}$$

$$\frac{\delta J}{\delta W} = -\frac{1}{\hat{y} \cdot ln(10)} \cdot \sigma'(z_{out}) \cdot (-2(a_h - W_h)) \cdot \sigma'(z_h) \cdot (-2(X - W)) \tag{3.14}$$

Algorithm 1 EN Training

Variables	
X_train	training set of input values
y_train	true labels of training set
$epochs$	number of training epochs
η	learning rate
W	first set of weights, prior to hidden layer
W_h	weights from hidden layer, preceding output layer
z_h	hidden layer output, prior to activation
a_h	sigmoid activation of z_h
z_{out}	final output prior to activation
Functions	
$init\_weights(X\_train, y\_train)$	initializes network weights, given training data
$len(x)$	returns number of samples in x
$randint(x)$	produces random integer in range 0 to $x - 1$
$model.predict(x)$	produces class prediction from input vector
$\sigma'(x)$	applies derivative of sigmoid function

```
1:  $init\_weights(X\_train, y\_train)$
2:  for $e$ in $range(epochs)$ do
3:      for $i$ in $range(len(X\_train))$ do
4:          $random\_index \leftarrow randint(len(X\_train))$
5:          $x_i \leftarrow X\_train[random\_index : random\_index + 1]$
6:          $y_i \leftarrow y\_train[random\_index : random\_index + 1]$
7:          $\hat{y} \leftarrow model.predict(x_i)$
8:          if $y_i == 0$ then
9:              $\nabla_{W_h} \leftarrow \frac{1}{ln(10)(1-\hat{y})} \cdot \sigma'(z_{out}) \cdot (-2(a_h - W_h))$
10:             $\nabla_W \leftarrow \frac{1}{ln(10)(1-\hat{y})} \cdot \sigma'(z_{out}) \cdot (-2(a_h - W_h)) \cdot \sigma'(z_h) \cdot (-2(x_i - W))$
11:             $W_h \leftarrow W_h - (\eta * \nabla_{W_h})$
12:             $W \leftarrow W - (\eta * \nabla_W)$
13:         end if
14:         if $y_i == 1$ then
15:             $\nabla_{W_h} \leftarrow -\frac{1}{\hat{y} \cdot ln(10)} \cdot \sigma'(z_{out}) \cdot (-2(a_h - W_h))$
16:             $\nabla_W \leftarrow -\frac{1}{\hat{y} \cdot ln(10)} \cdot \sigma'(z_{out}) \cdot (-2(a_h - W_h)) \cdot \sigma'(z_h) \cdot (-2(x_i - W))$
17:             $W_h \leftarrow W_h - (\eta * \nabla_{W_h})$
18:             $W \leftarrow W - (\eta * \nabla_W)$
19:         end if
20:     end for
21: end for
```

3.1.3 EN Training Algorithm

The procedure for training the EN is shown in Algorithm 1. Training begins by first initializing the weights. Line 2 begins the **for** loop to iterate through the number of training epochs. The inner **for** loop is performed for the number of samples in the training set, as shown in

Line 3. A random index is selected from the training set, which is saved into the temporary variables x_i and y_i per Lines 4-6. The model's prediction is acquired from a forward pass through the EN and saved in Line 7. Lines 8 and 14 contain **if** statements to decide which form of the partial derivative to use as described in Section 3.1.2. The respective contents of each **if** statement apply the appropriate updates to each set of weights. ∇_W and ∇_{W_h} are calculated via the partial derivative (Lines 9-10 and 15-16), which is then multiplied by the learning rate, η. That product is then subtracted from the current weight values to produce the updated weights (Lines 11-12 and 17-18). This process is repeated for the assigned number of epochs to complete training the EN.

3.1.4 Multiclass Classification

As noted in Section 3.1.1, the EN creates a separate network for each class present in the data set. For example, if a data set contains three classes, the EN will create three separate networks (0, 1, and 2) of an identical structure. All three would have the same number of hidden nodes and train over the same number of epochs, but each network updates its own weights independently. Each network trains by iterating through every data point. Network 0 would imply Class 0, so for the purposes of the loss function, a data point with a true label of 0 would follow backpropagation where $y == 0$ and a data point with a true label of 1 or 2 would perform backpropagation for $y == 1$. The same process would be followed for Networks 1 and 2 for classes 1 and 2, respectively. In this way, each network is trained as a binary, one-vs-all classifier and later aggregated during test time to perform multiclass classification.

3.1.5 Weights, Thresholds, and Decision Function

The EN accepts three hyperparamters: number of hidden neurons, epochs, and learning rate. The number of hidden neurons is the number of nodes in the hidden layer of each network which determines the size of the weight matrix. The EN initializes its weights by randomly selecting data points throughout each class. This sets the first set of weights to be equal to random samples of the data. Throughout training, the weights are updated, but they persist in the same space as the data set.

Each network has a threshold t that is first learned over the training set to maximize each network's binary classification score, and later tuned using the validation set to maximize accuracy over the multiclass prediction.

$$\hat{y}_i = \begin{cases} 0 & \text{if } f_i(x) < t_i \\ 1 & \text{if } f_i(x) \geq t_i \end{cases} \tag{3.15}$$

The binary prediction for each network is taken by comparing its raw output with its learned threshold, as shown in Equation 3.15. $\hat{y}_i$ is the prediction for network f_i given that network's threshold t_i. In multiclass classification, with n classes, an output vector Φ is built by iterating over n networks and applying Equation 3.15 such that $\Phi = \{\hat{y}_1, \hat{y}_2, \hat{y}_3 ... \hat{y}_n\}$. Then, the multiclass prediction is taken by:

$$\hat{Y} = \underset{\hat{y}_i}{\operatorname{argmin}}(\Phi) \tag{3.16}$$

For most representative cases in the data, Φ should only have one 0 for the predicted class while the rest of the values should be 1. For example, if $\Phi = \{1, 1, 0, 1, 1\}$, then the prediction would be 2. However, some edge cases were found that produced prediction vectors with all ones, or with more than one zero. For example, if $\Phi = \{1, 1, 1, 1\}$ or $\Phi = \{1, 1, 0, 1, 0\}$, then Equation 3.16 would produce errors. In these instances, $\hat{y}_i$ is calculated by taking the absolute difference between the network prediction $f_i(x)$ and its learned threshold t_i, as shown in Equation 3.17. Then, Equation 3.16 is applied to take the minimum argument of Φ for the class prediction.

$$\hat{y}_i = |f_i(x) - t_i| \tag{3.17}$$

3.1.6 Training Example

In this Section, we provide an example of training an EN on the make_moons and make_circles data sets from the scikit-learn library [9]. These data sets were chosen specifically because they are non-linear and 2D. Figure 3.2 shows how the EN's weights converge

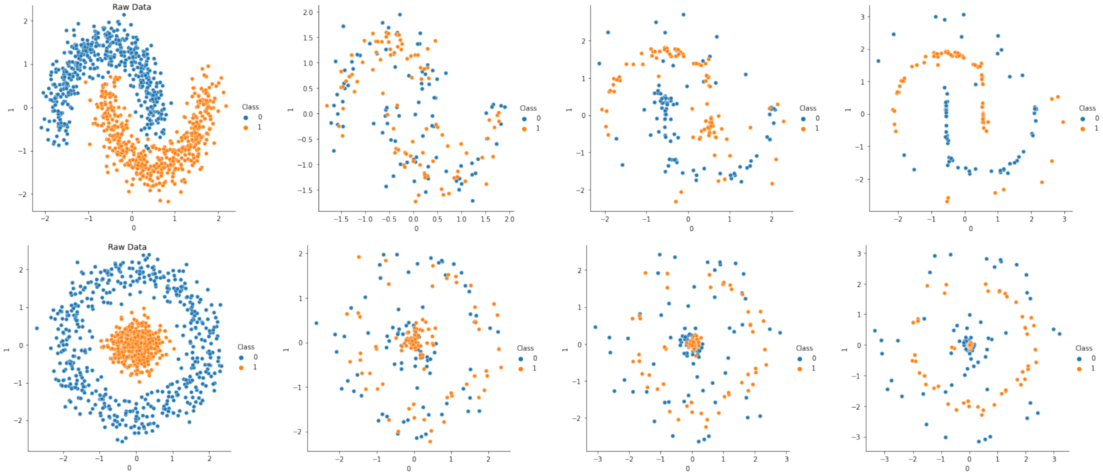

Figure 3.2. From left to right, the raw data and each network's weights at 1, 50, and 100 epochs.

to fit the non-linear data over three training epochs of 1, 50, and 100 respectively. The raw data set is shown on the far left. As epochs increase, the figure shows how weights begin to converge to their optimal location that minimizes loss during gradient descent.

The final set of weights (i.e., the far right images in Figure 3.2) are plotted in green over the raw data in Figure 3.3. Note that the moons data set weights for Class 0 converged to surround the data points for Class 1. This is counterintuitive because one would expect the weights for Network 0 to converge around data points from Class 0. Similarly, the circles data set weights appear to be interlaced throughout both classes, yet each data set achieved a 100% prediction accuracy. To better understand the final position of the weights, edge cases with Class 0 labels were selected from each data set. In addition to the three edge cases, a single, unambiguous point was selected for analysis. The edge points were chosen based on their position and vicinity in relation to the alternate class. Figure 3.3 displays the selected points in red. Each point was passed through Network 0 and Network 1 (associated with Class 0 and Class 1, respectively).

Table 3.1 displays the information for each point. Based on the performance of the EN, it is clear that it is correctly classifying these points, even though the weights converge to an unexpected location. For example, point 103 from Figure 3.3 and Table 3.1 is an instance

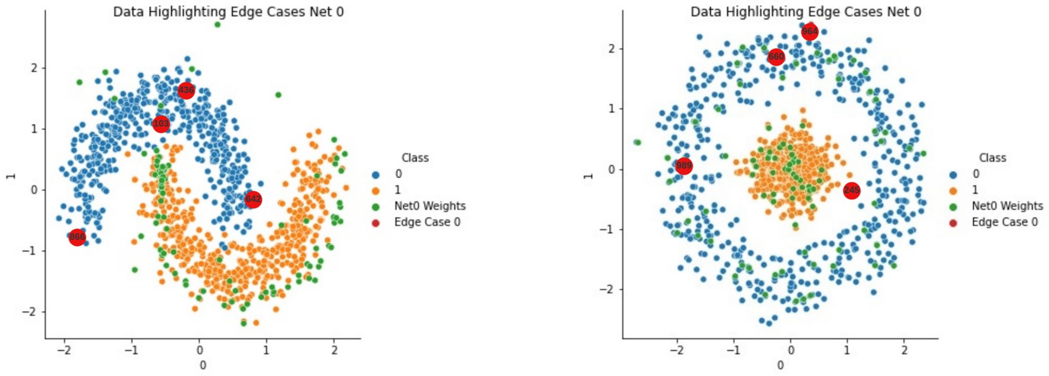

Figure 3.3. The example moons (left) and circles (right) data sets. The red points are near potential decision boundaries, where more uncertainty exists between each class. The edge cases and weights are shown for only one class for simplistic purposes.

of Class 0. Despite its vicinity to Class 1 instances, its label was accurately predicted by the model.

When the thresholds are taken into account, it becomes more clear as to how each network handles these points and why the algorithm is able to make an accurate prediction. Table 3.2 shows the threshold values for each network and data set. To classify each point, the model compares the raw output from each network to that network's threshold. If the output is less than the threshold then a 0 is recorded; otherwise, a 1 is recorded. The minimum value's index is then taken as the model's prediction. For sample point 436 from Table 3.1, Network 0's raw output is 0.056. Since this is less than Network 0's threshold (0.19), 0 is recorded. Alternatively, Network 1's output is 4.637, which is greater than its threshold, scoring a 1. The algorithm returns the index of the 0 prediction as the model's predicted label. Because only one 0 was recorded for this sample, no further heuristic is required.

However, in cases where there are multiple 0's or 1's recorded, a tie breaker must be held to produce a single label prediction. An example of this tie breaker is seen in the first edge case of the moons data (sample 866). This point produced raw outputs of 0.929 and 3.519 for Networks 0 and 1, respectively. To correctly classify this point as Class 0, we take the minimum absolute difference between the raw output and the threshold for each network.

Table 3.1. Network Raw Output Compared to Threshold

Moons Edge Point Data				
Sample index	(x, y)	Net0 Raw Out	Net1 Raw Out	Predicted Class
436	(-0.243, 1.398)	0.056	4.637	0
866	(-1.783, -0.797)	0.929	3.519	0
103	(-0.800, -0.173)	0.808	3.467	0
642	(-0.509, 1.022)	0.676	3.118	0
Circles Edge Point Data				
Sample index	(x, y)	Net0 Raw Out	Net1 Raw Out	Predicted Class
660	(-0.180, 2.034)	0.473	4.083	0
989	(1.165, -0.379)	1.794	4.229	0
964	(0.362, 2.392)	0.692	10.62	0
245	(-1.902, 0.007)	0.395	9.985	0

Table 3.2. Network Thresholds

Thresholds	Net0	Net1
Moons	0.19	0.25
Circles	0.06	0.08

For example, $\hat{y}_0 = |0.929 - .19|$, and $\hat{y}_1 = |3.519 - 0.25|$. Thus, the model predicts Class 0 because $\hat{y}_0 < \hat{y}_1$. This comparison of raw outputs to thresholds is completed for each data point that has multiples of the same label prediction. This calculation enables the EN to assign the appropriate class to each label and make accurate predictions.

3.2 Data Acquisition and Preparation

21 diverse real-world data sets were selected from the University of California Irvine Machine Learning Repository [10]. A complete list of the chosen data sets and their characteristics can be found in Table 3.3. Most of the data sets contain very few features, with the largest feature space being 64. All data sets contain no missing values and each sample contains one label for classification purposes. Minimal adjustments of the data sets were required in order to prepare them for input into the EN and additional classification models.

In order for the EN to process the class labels, they had to be adjusted to the values 0 to $n-1$, where n is the number of classes. However, it is important to note that this is not relabeling the classes into ordinal data; it is only to assign classes with a string data type to an integer. The data was then separated into its x and y values, where x contained all of the feature information and y contained the label. These values were then split into a train and test set. Next, the feature columns were passed through a pipeline to scale and normalize their values. All inputs were normalized, but the scaling applied by the pipeline varied slightly depending on the model performance for a particular data set. The feature data was scaled using either a standard score (standard scaler) or a minimum-maximum range (min-max scaler). The feature data of the test set was then transformed using the pipeline parameters of the training data. Once this data preparation was complete, the data sets were passed through multiple iterations of models in accordance with the described hypotheses.

3.3 Euclidean Network as an Independent Classifier

Initial tests of the EN showed promising results. The EN classification performance was compared to other popular classification algorithms. The metric used for this comparison was the accuracy score. Precision and recall were used in the tuning process for each model, but the final results comparison used only accuracy. The following other classifiers were used for comparisons: Gaussian RBF kernel SVM, linear SVM, random forest, KNN, logistic regression, dense neural network, and CNN. The hyperparameters were tuned for each model and both scaling techniques were tested to achieve the best observed accuracy score with the test set.

The process of tuning the EN mostly included adjustments to the number of hidden neurons, epochs, and learning rate. The most sensitive hyperparameter for the EN is the number of hidden neurons, so this value was adjusted first. Most data sets were tested with the following number of neurons: 100, 300, 500, and 1000. Some data sets were tested with more, with the highest value tested at 10,000. Once the optimal number of hidden neurons was identified, the number of epochs and the learning rate were tuned. The number of epochs used was typically 100, 300, 500, or 1000. Initially, the learning rate was set to 0.1 and generally remained in the 0.01 to 0.3 range. The learning rate was also tuned according to the epochs to ensure that the model reached convergence. Though an exhaustive grid search method

Table 3.3. Data Sets

Data Set	**Features**	**Classes**	**Samples**
Australian Credit	14	2	690
Balance Scales	4	3	625
Breast Cancer	30	2	569
German Credit	24	2	1000
Glass	9	7	214
Heart	13	2	270
Image Seg	19	7	2100
Ionosphere	34	2	351
Iris	4	3	150
Landsat	36	6	6435
Letter Recog	16	26	20000
OptDigits	64	10	5620
Pima Diabetes	8	2	768
Sonar	60	2	208
Soybean	35	4	47
TAE	5	3	151
Vehicle Silhouettes	18	4	846
Vowel Recog	13	11	990
Waveform	21	3	5000
Waveform+noise	40	3	5000
Wine	13	3	178

was not used to find the optimal hyperparameter values, enough test runs were performed to produce high accuracy scores with roughly tuned parameters.

Once all models had been tuned and test runs completed, the accuracy scores for each model were compared for all data sets. Through this comparison, it was evident that the EN could perform well as an alternative classification algorithm and could also outperform well-established algorithms. Metrics from these tests are discussed further in Section 4.1.

3.4 Synthetic Point Generation

Synthetic point generation is a useful technique to augment data sets that may not contain many training samples. The EN is able to augment a data set by adding its first layer of weights to the original data set. With this method, a small data set can be grown to an adequate size for effective training. The goal is to use the additional samples to effectively train a model that can continuously perform well as more data is added. To perform data augmentation via the EN, the first layer of weights, as seen in Figure 3.4, is extracted from a trained model. Because the EN uses independent networks to train each class, the weights from each network are assigned the class label of that particular network. This process creates new data points within the same feature space as the input data. These data points are assigned a label based on the network from which they came. To test the effects of synthetic points, the EN was trained and the weights from the first layer were extracted. These were then concatenated with the input data set to create a new, hybrid data set. This data set consisted of the original, raw data and the newly generated synthetic points. Once the hybrid data set was created, it was used as the input data set for a number of classifiers. The KNN, Gaussian RBF SVM, and the EN were retrained with the hybrid data set. The accuracy was then calculated for each model with the same test set, but with the newly trained parameters based on the hybrid data.

Additionally, the effects of multiple iterations of synthetic point generation were examined. This process consisted of performing the above augmentations, retraining the EN model with the hybrid data, then augmenting that hybrid data with additional synthetic points. This method could potentially vastly increase the available samples for a data set. However, the accuracy of the models continued to decrease after implementing data augmentation. This decrease was seen whether a single or multiple iterations were performed. This is further discussed in Section 4.2 to explain why the hybrid data produced poor results and also to provide a better understanding of how the EN weights converge.

3.5 Feature Space Expansion

3.5.1 Extracting Activations

The last area of research involved expanding the input data set's feature space to increase its dimensionality. Many data sets contain very few features and may limit the learning of more intensive algorithms, particularly CNNs. Many of the data sets chosen for this thesis possess a limited number of features, which made them prime candidates to test the effects of increased dimensionality. The additional features are not arbitrarily added. Instead, their

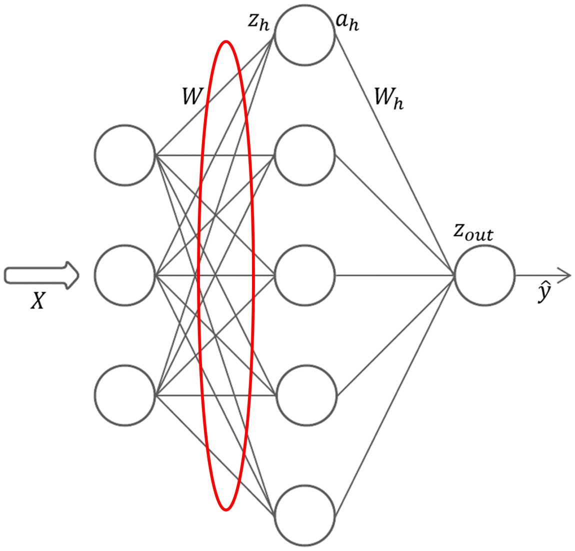

Figure 3.4. A simple network highlighting the first layer of weights. They possess the same dimensionality of the input data, so they may be treated as individual data points for the purposes of data augmentation.

values come from the second layer of weights, and the magnitude of the expansion depends on the number of classes and the hidden neurons hyperparameter.

$$\begin{aligned} z_h &= (X - W)^2 \\ a_h &= \sigma(z_h) \end{aligned} \tag{3.18}$$

In order to extract the required weights from the model, the EN was trained on the raw, input data set. The values are calculated with Equation 3.18, previously described in Section 3.1.2. Once the EN was trained, the values of the activations (i.e., second layer of weights) were saved into a new data set. The size of this new data set depended on the number of classes and hidden neurons. A visual representation of the section of the network that is being used to produce the new data set is shown in Figure 3.5. Because the EN creates a network for each class, the feature space of the activations is equal to the number of classes multiplied by the number of hidden neurons. While tuning the EN, many data sets achieved high accuracy results with 100 to 1000 hidden neurons. These values produced

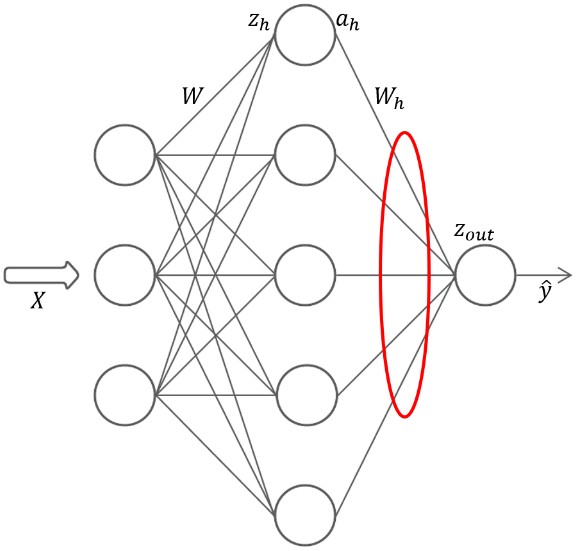

Figure 3.5. Encircled here is the second layer of weights, also called activations, for a single network. Note how the number of activations is directly linked to the number of neurons in the hidden layer.

activations that contained a few hundred to a few thousand features, much more than the original data set. For example, the Glass Identification data set contained 7 classes and the EN was tuned to contain 1000 hidden neurons. This combination produced a new data set with 7000 features, much more than the original 9.

Because feature expansion was performed with the EN, it was important to also test a similar process with traditional neural networks. Single layer, dense networks were also trained and tuned with each data set. Traditional neural networks train their parameters differently, so the shape of the output data set was different. After the activation data was extracted, the number of new features equaled just the number of hidden neurons. To compare to the same data set (Glass Identification), a neural network was tuned to 1000 hidden nodes. This produced activations consisting of 1000 features. The same, second layer of weights is taken from each architecture, but as described in Section 3.1.1, the EN creates a separate network for each class. This results in an even larger feature space than the traditional neural network's activation data.

3.5.2 Testing Scheme

The initial experiments with activations compare the classification accuracy of multiple classifiers using the EN activations and raw data sets. An EN was trained and tuned for each data set, saving the activation data into a separate file for later use. It was then used as a new input data set for a CNN, random forest (RF), and a Gaussian RBF kernel SVM. Each model was tuned for the raw data and activations separately in order to achieve the highest score for each. Particular attention was given to testing CNNs due to their ability to identify complex functional relationships. A basic architecture was used, but tuning was accomplished by adjusting the number of convolutional layers, dense layers, and dropout ratio. Finally, the accuracy score for each model with the activation data was compared to the accuracy of the models with the raw data. Section 4.3 discusses complete results. Though the results were mixed, most of the data sets saw an increase in accuracy when they used the activation data from the EN, dense neural network, or both.

CHAPTER 4: Experimental Results

4.1 Euclidean Network Classification Results

All data sets were tested with each classifier (Gaussian RBF kernel SVM, linear SVM, random forest, KNN, logistic regression, dense neural network, and CNN) and the model with the highest accuracy was recorded for comparison with the EN. The EN had an accuracy greater than or equal to the next highest scoring model for 12 of the 21 data sets. The results for each data set are shown in Table 4.1. The EN still performed well on the other seven data sets, achieving accuracy scores comparable to the other classifiers. The largest disparities between the best classifier and the EN were 6.5% and 13.1% for the Teacher Assistant Evaluation and Vowel Recognition data sets, respectively. The EN's accuracy for the remaining five data sets was within 5% of the best model's accuracy as shown in Figure 4.1. A bar plot was also created for each data set to better visualize accuracy differences between each of the models. Figures 4.2 and 4.3 provide the accuracy results for all 21 data sets.

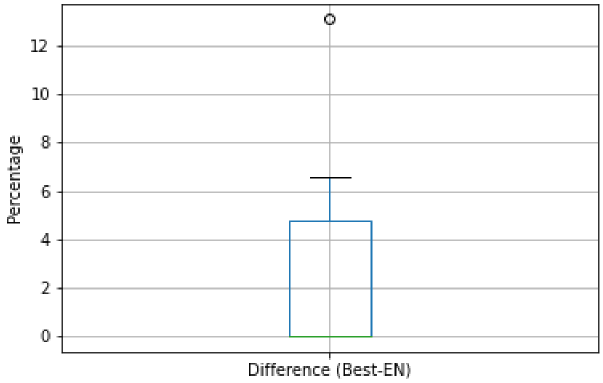

Figure 4.1. The difference in accuracy percentage between the best-performing model and the EN. Most models had a difference of 0, meaning the EN was the best model.

Table 4.1. Highest Accuracy Models

Data Set	**Best Model**	**Best Accuracy (%)**	**EN Accuracy (%)**
Australian Credit	EN Tied	70.3	70.3
Balance Scales	CNN	98.4	89.6
Breast Cancer	EN Tied	100.0	100.0
German Credit	EN	80.0	80.0
Glass	EN	76.7	76.7
Heart	EN Tied	81.5	81.5
Image Seg	CNN	96.7	94.0
Ionosphere	EN	97.2	97.2
Iris	EN Tied	100.0	100.0
Landsat	CNN	90.4	88.7
Letter Recog	SVM	97.4	90.8
OptDigits	SVM/CNN	99.0	96.8
Pima Diabetes	EN Tied	77.9	77.9
Sonar	Tied(SVM/RF/NN)	88.1	83.3
Soybean	EN Tied	100.0	100.0
TAE	LR	71.0	64.5
Vehicle Silhouettes	SVM	90.6	84.7
Vowel Recog	SVM	99.0	85.9
Waveform	EN	88.4	88.4
Waveform+noise	EN	89.6	89.6
Wine	EN Tied	100.0	100.0

The EN showed promising results and was able to produce the highest accuracy score for most, but not all, of the data sets. However, it performed comparably in those other cases. These results show the viability of using the EN as an independent classifier. Though the EN performed well on all data sets, further research is required to discern if there are particular types of data sets for which the EN performs better or worse. From the above results, the answer is unclear as to which types of data sets would favor the use of an EN.

4.2 Effects of Synthetic Point Augmentation

As mentioned in Section 3.4, the augmented data consistently produced lower accuracy scores than the raw data. Only eleven data sets were fully tested, but each model produced

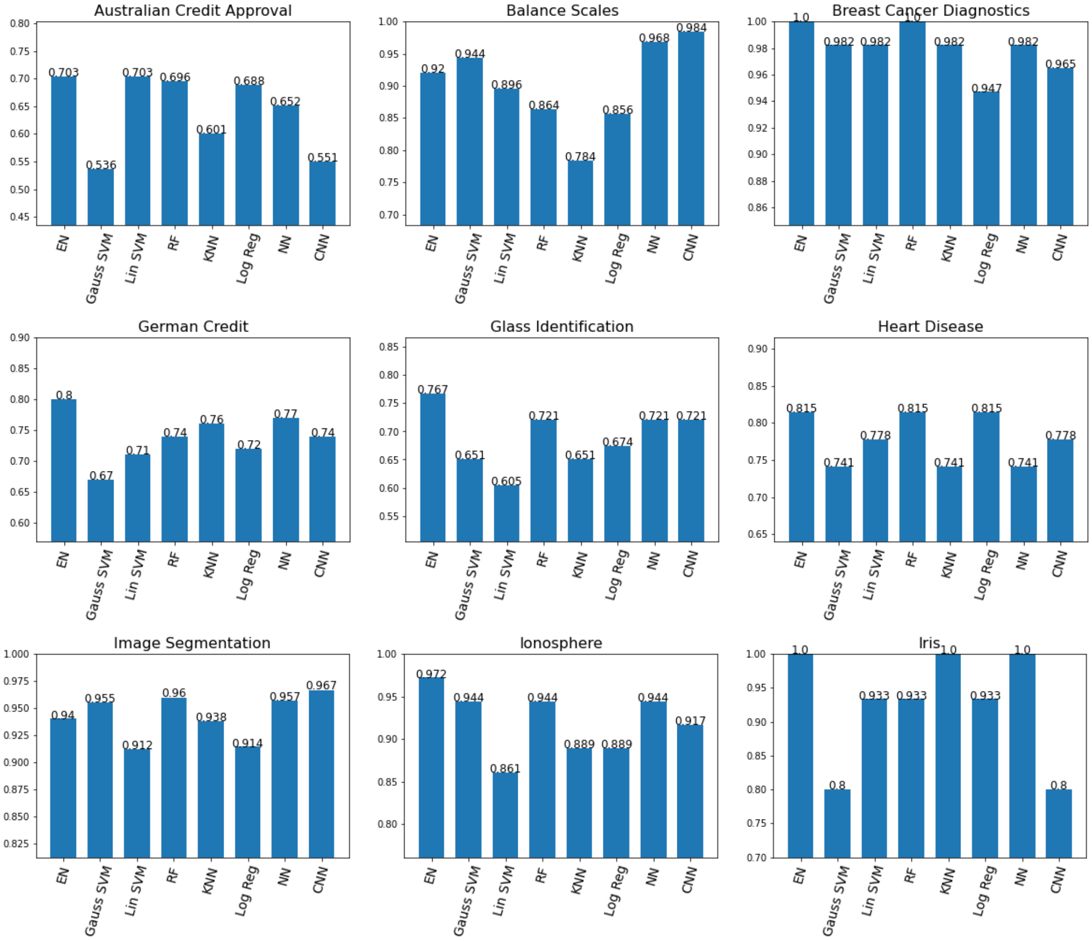

Figure 4.2. The first 9 of the 21 plots (remainder shown in Figure 4.3). Each plot displays the results of each algorithm for comparison with the EN.

lower accuracy scores, except for two, where no change was observed. However, additional iterations of data augmentation further decreased model performance and produced lower accuracy in all cases. Due to the low performance, this particular area of testing was cut short due to the computing time required for each data set and model. Table 4.2 shows the single iteration results for the tested data sets. These data sets were augmented with synthetic points generated by the EN, which were then joined with the original data set to produce the hybrid data set. New models were then trained on this hybrid data. Displayed are the comparison results for the raw data's accuracy for the KNN, Gaussian RBF kernel SVM, and EN. Though inauspicious, the poor results provided the impetus to conduct further

Table 4.2. Synthetic Point Augmentation Results

Data Set	Raw KNN	Augmented KNN	Raw SVM	Augmented SVM	Raw EN	Augmented EN
Australian Credit	60.1	48.6	53.6	54.3	70.3	69.6
Balance Scale	78.4	71.2	94.4	93.6	92.0	57.6
Glass Identification	65.1	14.0	65.1	60.5	76.7	48.8
Heart Disease	74.1	46.3	74.1	61.1	81.5	61.1
Image Segmentation	94.3	94.3	95.2	95.7	95.0	81.4
Landsat Satellite	89.5	89.4	87.8	87.9	88.7	84.2
Optical Recognition of Digits	98.0	97.1	99.5	98.8	98.6	73.3
Sonar	83.3	69.0	88.1	78.6	83.3	81.0
Teaching Assistant Evaluation	58.1	29.0	67.7	22.6	64.5	38.7
Waveform+noise	71.0	48.4	84.0	75.4	89.6	56.0
Wine	91.7	51.4	100.0	70.8	100.0	91.7

analysis as to why accuracy suffered when synthetic points were introduced. These tests led to a greater understanding of how the EN's weights converge and, when plotted in the same space as the original data, why they are not effective for augmentation.

To understand how the synthetic points affected the data set, the moons data set was used for its simplicity [9]. Its two-dimensional space made visualization much easier, so moons were created, an EN was trained, and the first set of weights were plotted in the same space as the original data. The weights for each of the two networks are shown on the left-hand side of Figure 4.4. The raw data set consists of the two masses near the center of the plot, with the synthetic points surrounding them. This plot easily distinguishes the weights (synthetic points) from the raw data and shows where they converged once fully trained. Once clearly identified, the weights were again plotted with the raw data, but with the same color scheme for class labels. The right-hand plot of Figure 4.4 shows the hybrid data set (raw data and synthetic points) with common class label colors. It shows that the synthetic points converged to surround the opposite class. Visually, this seemed counterintuitive, but as discussed in Section 3.1.5, the calculation of the weights and classification based on the network thresholds confirmed the final location of the synthetic points. From Figure 4.4, it is clear that the hybrid data creates more ambiguity by adding data points with the opposite label near existing points, causing the models' performance to suffer. Once the synthetic points were visualized, it became clear why multiple iterations of synthetic point generation further degraded accuracy. These findings negate the hypothesis that data augmented with EN weights will improve accuracy. These results also showed that EN augmentation had a negative effect on all model and data types.

4.3 Euclidean Network Activations Results

4.3.1 Activations and Raw Data

Initially, EN models were trained on the data sets to produce the activation data. These activations have a larger feature space than the original data. This higher-dimensional data was used as input to train additional models for each data set. Three classification algorithms were selected to be trained with the activation data: CNN, random forest, and Gaussian RBF kernel SVM. With 21 data sets, this resulted in a total of 63 new models. The EN activations produced the same or better accuracy than the raw data for two-thirds of the models, with nearly 60% being strictly better. Figures 4.6 and 4.7 show the comparison plots produced for each data set. Some models showed significant disparity between the raw data and activation results, while others had near identical performance. It was expected that some models would have similar performance because when the feature space was expanded, the new features were highly correlated with the original since they were created based off of the original features' weights. This expectation of highly correlated values led to further examination of the feature space via heat maps. The expanded feature space was analyzed for common patterns between the original features and those created by the EN. Figure 4.5 shows examples from three data sets. The expected output was heavily correlated values throughout the entire heat map. However, each data set produced a unique map with differing levels of correlation.

Another analysis included the use of grayscale images to visualize the expanded feature space. This process involved the selection of random samples from each class and plotting the features in a nearly square image. The purpose of this analysis was to see if samples from a specific class showed common patterns in their feature space images. Figure 4.8 shows an example from the Waveform data set. This data set contained three classes, from which three samples were taken. Each plot shows three clear separations in the feature space, though a pattern may not be immediately clear. The top three plots contain three samples of data points labeled Class 0. When visually compared, one can see that the topmost row of each of these plots shares a more similar pattern than the two bottom rows for each sample. The same can be said for the second row of the Class 1 plots (middle row of plots), and the bottom row of Class 2's plots (bottom row of plots). The expanded feature space creates a data set with greater separation between the classes, as seen by the clear delineation between each plot's rows. This point is made more evident when quantitative analysis is performed for each set of plots. The average value was taken for each row of each sample. For the Class 0 samples, the top row (corresponding to a predicted label of 0) showed very little variance, 3.88e-5, while the variances for the other class labels for those same samples were 6.30e-4 and 1.16e-3. This shows how the EN activations produced an image where the corresponding, accurately predicted class is more similar between samples than the incorrect labels.

The EN is able to better classify the data set because it can produce activation data that has greater separation between classes, while maintaining likeness between samples of the same class. Based on the performance for this particular data set (Waveform, Figure 4.7), one can see that this separation is favorable so that a model can more accurately classify samples. Alternatively, Figure 4.9 shows the Heart Disease activation images. The delineations between the rows and classes in the feature space image are less evident and, consequently, the models did not perform as well when trained on the activation data. Again, quantitative analysis was performed in a similar manner as with the Waveform data set. This time, the Heart Disease activation images produced similar variances between each sample's predicted class (1.84e-6 and 3.77e-5). From the activation comparison results in Figure 4.6, the CNN can be seen to have still performed well with the EN activations, while the other two models show no improvement or lower performance. This may be explained by the CNN's ability to extract more information from each data set, despite the closer variances between the predicted classes of each sample. The ability for the EN to transform the data to higher dimensions can, in some cases, create greater separation between the classes in that data. This greater separation can enable models to perform better than when trained on the original data set. However, it is not a guarantee for all data sets and, in some cases, the use of activation data decreased model accuracy, though these instances were in the minority (one-third of models tested).

4.3.2 Euclidean and Neural Network Activations

Additionally, a traditional neural network was used to expand the feature space of the data sets. This was done to enable a direct comparison between the EN's activations and those from a conventional architecture. The previous section discussed the effects of higher-dimensional data on training, but it did not compare the results of an alternate method for creating that higher-dimensional space. To account for this, each data set was also passed through a dense neural network to extract the hidden layer activations. The neural network's activation data showed similar results to the EN when compared to the raw data, so the objective then was to see if the EN activations produced a better accuracy score than its traditional counterpart. Due to the computational demand of some of the larger data sets, only 42 models were compared for the EN and neural network. Of these 42, two-thirds showed equal or better performance, with one-third producing strictly better results with the EN. Comparison plots were produced to show the performance of the models with the EN and neural network activations, shown in Figures 4.10 and 4.11. As with the previous test, there was no clear correlation between a data set's characteristics and its activation performance. Both the EN and neural network improved the models' performance for most of the data sets, which supported the claim that a higher-dimensional feature space can improve classification.

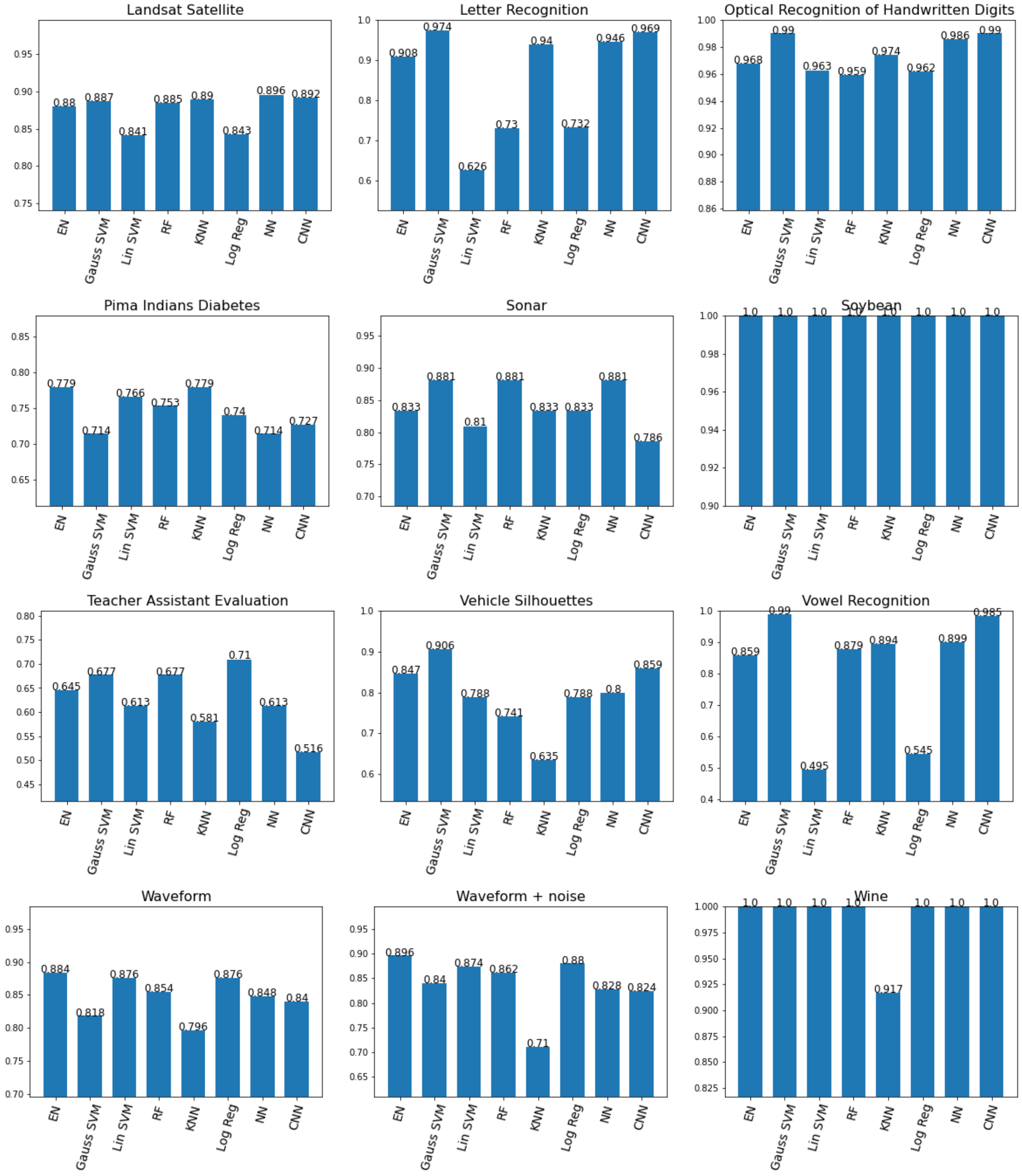

Figure 4.3. The final 12 plots for each data set shows the accuracy results for each model. The EN tied or out performed the next best model for 66.67% of the data sets.

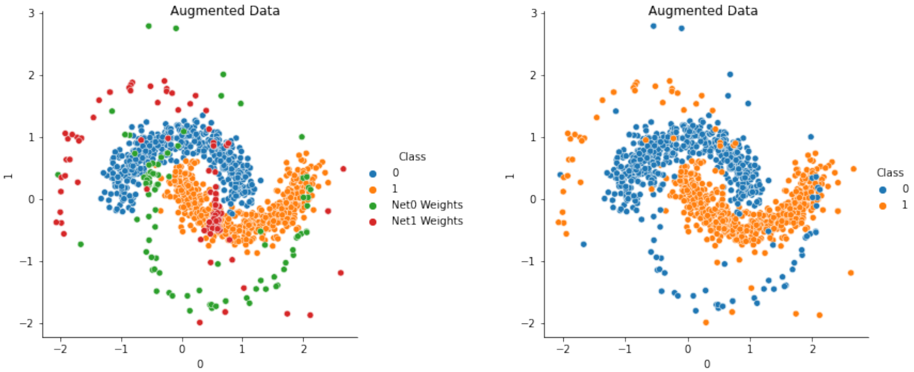

Figure 4.4. Left: synthetically augmented data with highlighted weights (red and green). Right: hybrid data with corresponding class labels.

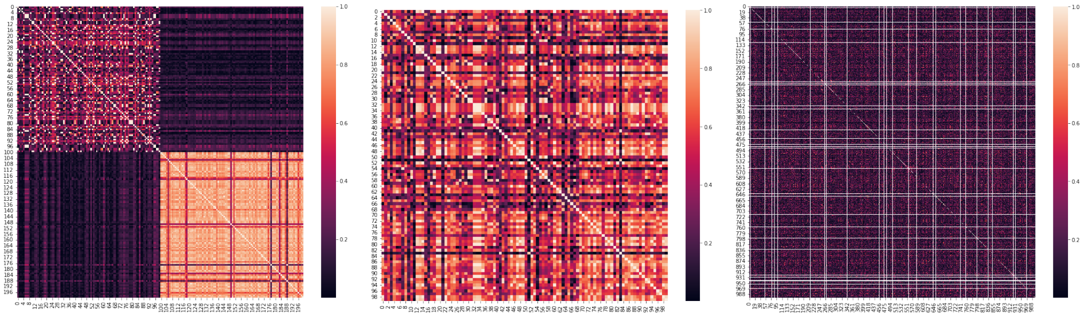

Figure 4.5. Correlation heat maps of higher-dimensional feature space. Lighter colors were expected throughout the image to show greater correlation between the created features, but each data set produced unique patterns of correlation.

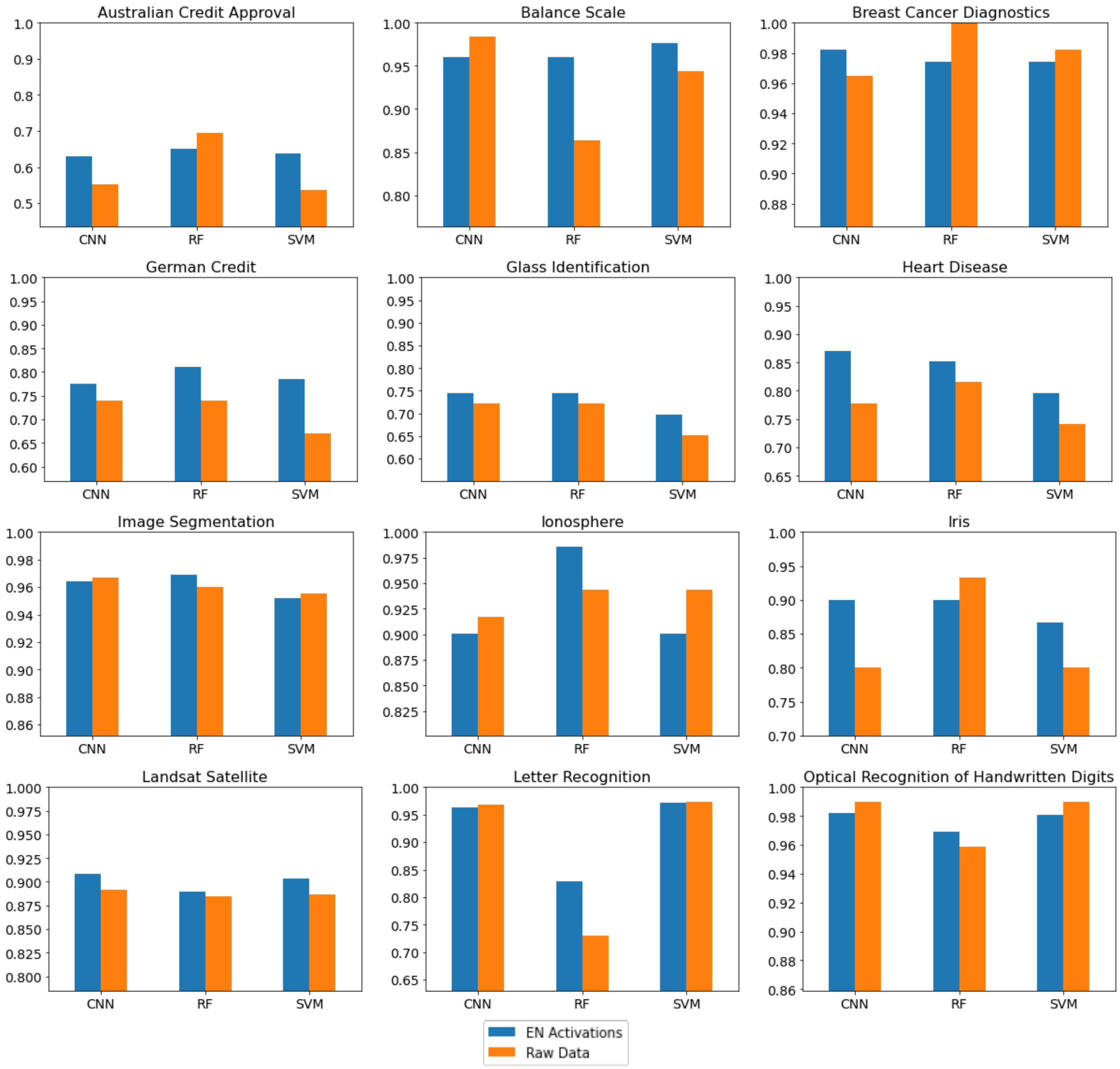

Figure 4.6. The first 12 of the 21 plots (remainder shown in Figure 4.7). Each plot shows the comparison between the results of the raw data with the EN activations for the CNN, random forest, and Gaussian RBF kernel SVM.

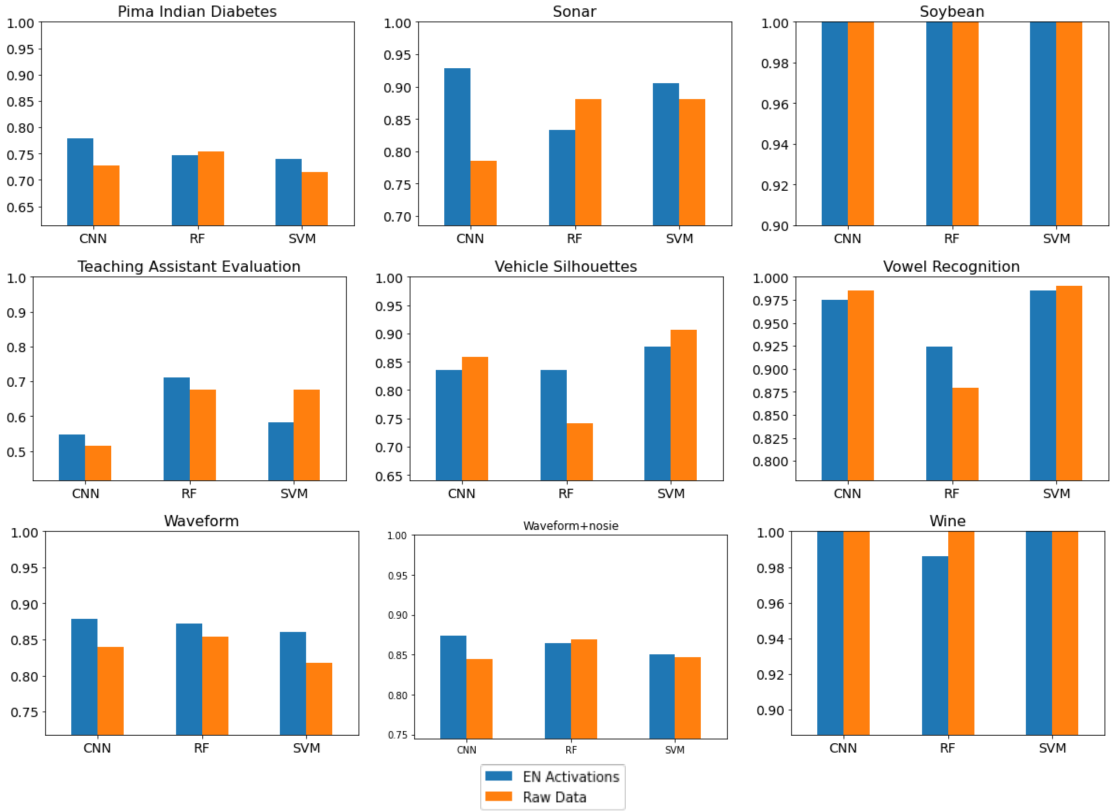

Figure 4.7. The final 9 plots for each data set comparing the accuracy results for the raw data and EN activations.

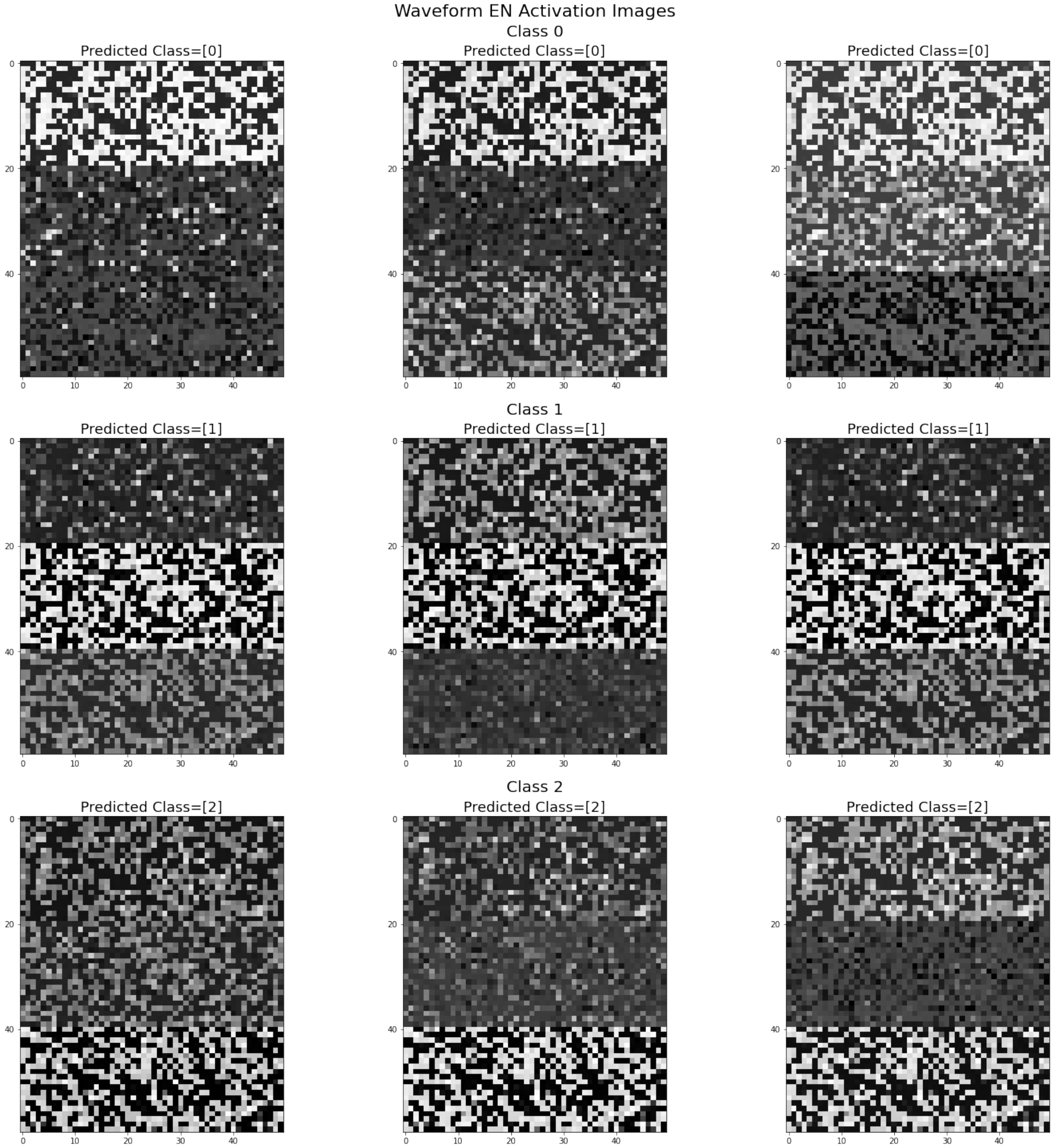

Figure 4.8. Visualization of the expanded feature space for the Waveform data set. The clear separation into three rows corresponds to the three class labels.

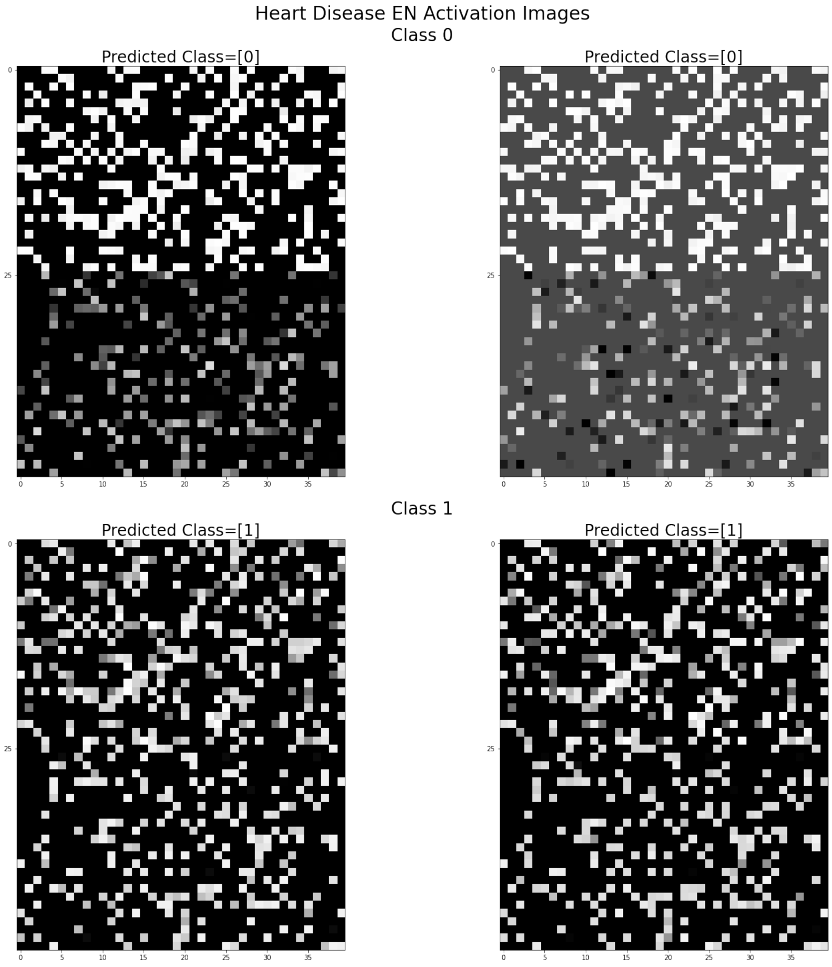

Figure 4.9. Activation data image for the Heart Disease data set. Less obvious delineation between the class rows corresponds with lower performance.

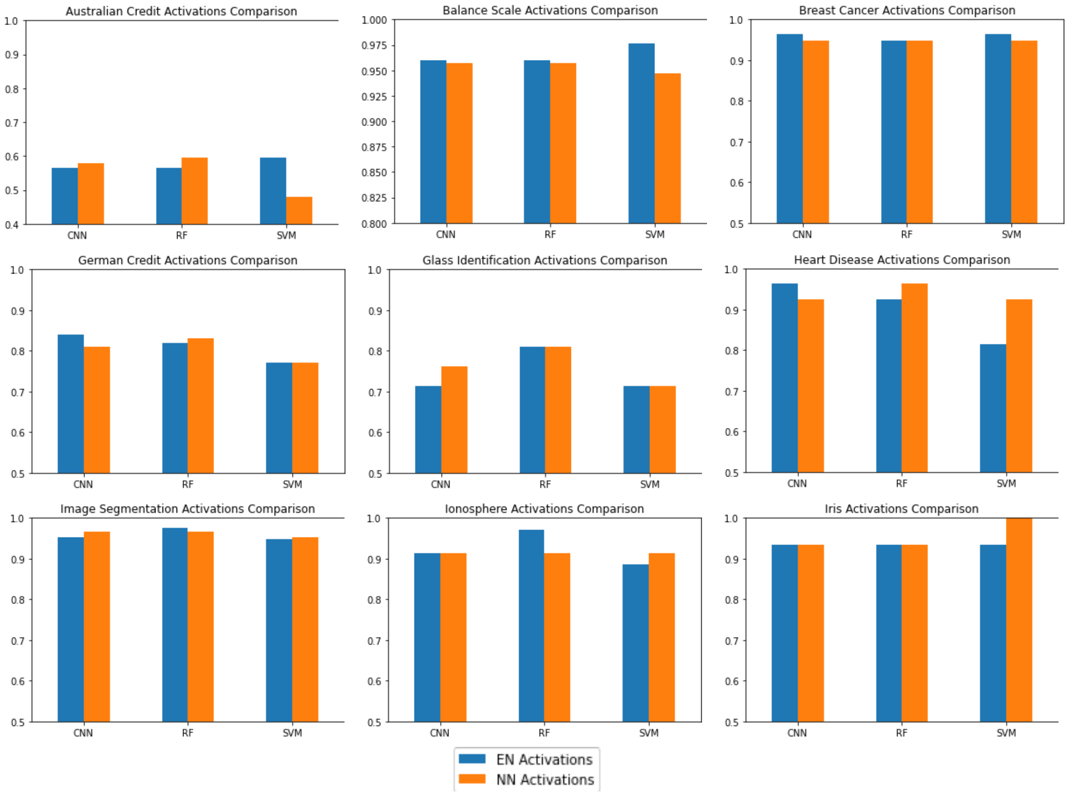

Figure 4.10. The first 9 of 20 plots where the EN and neural network activations were compared.

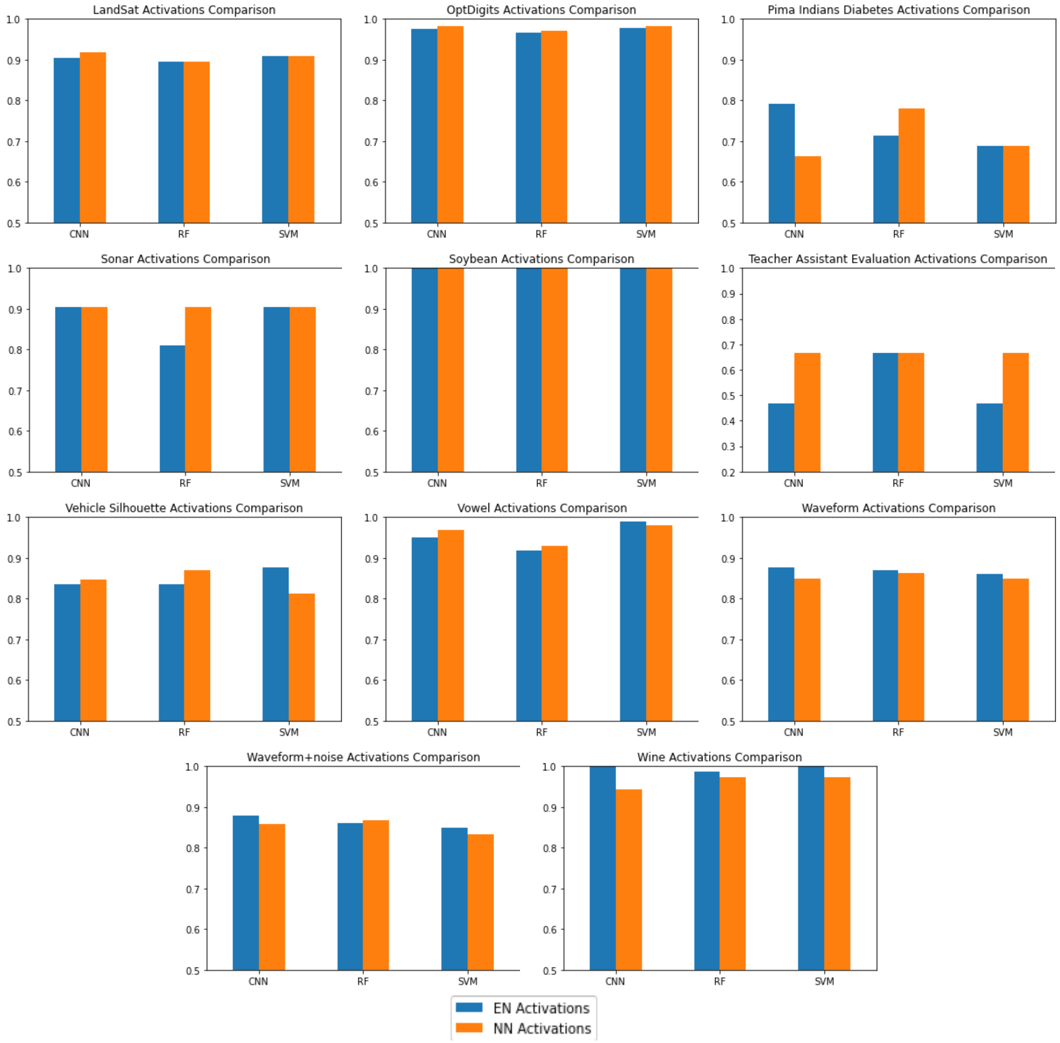

Figure 4.11. The final 12 plots showing EN and neural network activations comparisons. The Letter Recognition data set was omitted due to memory requirements and computational complexity.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5: Conclusions and Future Work

5.1 Conclusions

In two of the three areas of research, the EN showed positive results. These areas were the performance of the EN in terms of classification accuracy as an independent algorithm and the EN's ability to transform the data into higher dimensionality to improve performance with existing classifiers. The analysis of synthetic point generation via the EN's first layer of weights showed that this method of augmentation was ineffective and degraded results for all data sets.

As a standalone classifier, the EN performed comparably to CNNs and neural networks for all 21 data sets and achieved the highest accuracy score for 12. If all models are compared instead of individual data sets, the EN achieved the highest accuracy score for 114 of the 147 models (77.5%). Additionally, the EN's activations produced higher accuracy for 42 of the total 63 models. The activations were able to improve the performance of at least one model for 19 data sets; the remaining two data sets had already achieved 100% accuracy, so allowed no further improvement from EN activations.

To better quantify the positive results for each of the successful areas, percent difference was used to show the EN's success. The left plot of Figure 5.1 shows the percent difference of the EN against four of the top-performing algorithms (Gaussian RBF kernel SVM, random forest, CNN, and a fully connected neural network). As an independent classifier, the EN showed an average of more than a 3.5% increase in classification accuracy versus the other four displayed algorithms.

The next area of interest, expansion of the feature space, is shown in the right plot of Figure 5.1. This plot shows the percent difference of the Gaussian RBF kernel SVM, random forest, and CNN when the EN activation data is used as input versus the raw data. Here, the EN activations resulted in a 3% average improvement across the data sets for these three models. The overall positive percent difference when EN methods were applied shows

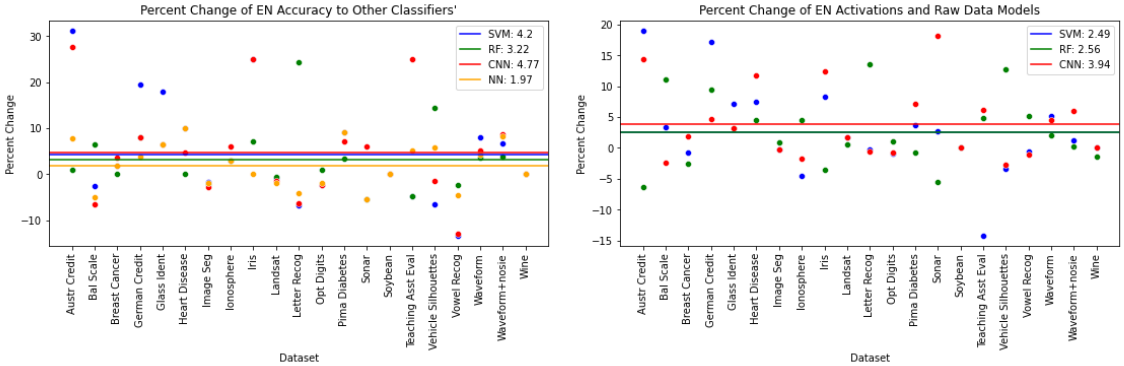

Figure 5.1. Percent increase of the EN compared to other algorithms (left) and the percent increase of the displayed models when using the EN activation data compared to raw data as input (right). The horizontal lines display the average percent increase between the EN and the corresponding model for all data sets.

that it can perform well for classification tasks and produce comparable results to more advanced architectures. The results also support feature space expansion as a promising area of research and a technique to consider to improve performance for smaller data sets.

5.2 Discussion

One goal of this thesis was to determine if certain data set characteristics could provide insight into whether or not certain functions of the EN could improve performance. In this pursuit, the EN's accuracy was plotted against each data set's characteristics to include the number of features, classes, samples, and the product of the number of features and classes. The average accuracy between the CNN, Gaussian RBF kernel SVM, and random forest with the expanded feature space data was also plotted against the same characteristics. The plots showed a slight upward trend, but nothing strong enough to provide definitive explanation as to why some data sets performed well and others did not. There was also a lot of variation in results between data sets with similar characteristics, i.e., same number of features and/or classes.

From this analysis, it remains unclear what types of data sets benefit most from the EN. Increasing the dimensionality of the input data revealed a more positive trend in smaller data sets (low number of features and classes). Though performance was not increased across the board for all models and data sets, using hidden layer activations to expand the feature space of some data sets was shown to be effective. This improvement occurs more often for data sets with a small number of features and classes. This fits with the original goal of the research to create a more complex data set from which better functional relationships could be extracted. A very simple data set would benefit more from expanding its feature space than an already complex (high dimensionality) data set.

5.3 Future Work

Immediate future work would apply to the code and functionality of the EN. The EN architecture is nascent and therefore not optimized for full implementation. Currently, its runtime is significantly longer than other classifiers because it does not incorporate parallelization or graphics processing unit (GPU) support. Because the EN creates multiple networks that train independently for each class, they could each be trained simultaneously, in parallel. Training each network simultaneously would greatly reduce runtime and enable the EN to be applied to data sets with a larger number of classes. Additionally, introducing GPU support would further increase the training speed.

Another option to improve the algorithm's efficiency would be the inclusion of generators. Python generators would allow the models to work with batches and mitigate some of the memory issues experienced with larger data sets. This would be especially useful for those cases where the activations of an already large data set have extensive memory requirements. With optimized code, the run time and memory needs of the EN could be vastly decreased to make it more competitive with CNN and other architectures in terms of operating requirements.

Due to the magnitude of features present in the activation data sets, additional and more complex CNN architectures may be used to monitor performance. This research used a basic CNN architecture, consisting of two to three hidden layers to expedite testing, but it is likely that there are other architectures that would be more adept at extracting key relational data from the higher-dimensional feature space.

The success of the EN as a standalone classifier suggests that it could also perform well on its own higher-dimensional data. A possible experiment would be to use the EN to produce a larger feature space (as performed in this thesis), then recycle that activation data into a new, untrained EN. Unfortunately, additional work would need to be invested into the EN architecture to optimize it. In its current state, the EN does not take advantage of parallelization or GPU support. Once the algorithm is optimized, additional and more complex experiments may be performed.

APPENDIX: Full Derivation

Sigmoid function and its derivative.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \qquad \sigma'(x) = \sigma(x) \cdot (1 - \sigma(x)) \tag{A.1}$$

Activations of input, X and weights, W.

$$\begin{aligned} z_h &= (X - W)^2 \\ a_h &= \sigma(z_h) \\ z_{out} &= (a_h - W_h)^2 \\ \hat{y} &= \sigma(z_{out}) \end{aligned} \tag{A.2}$$

Loss Function.

$$J = \begin{cases} -log(1 - \hat{y}), & \text{if } y == 0 \\ -log(\hat{y}), & \text{if } y == 1 \end{cases} \tag{A.3}$$

$\frac{\delta J}{\delta W_h}$ if y==0.

$$\frac{\delta J}{\delta W_h} = \frac{\delta J}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta z_{out}} \frac{\delta z_{out}}{\delta W_h}$$

$$\frac{\delta J}{\delta \hat{y}} = \frac{1}{ln(10)(1 - \hat{y})} \tag{A.4}$$

$$\frac{\delta \hat{y}}{\delta z_{out}} = \sigma'(z_{out})$$

$$\frac{\delta z_{out}}{\delta W_h} = -2(a_h - W_h)$$

$\frac{\delta J}{\delta W_h}$ if y==1.

$$\frac{\delta J}{\delta W_h} = \frac{\delta J}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta z_{out}} \frac{\delta z_{out}}{\delta W_h}$$

$$\frac{\delta J}{\delta \hat{y}} = -\frac{1}{\hat{y} \cdot ln(10)} \tag{A.5}$$

$$\frac{\delta \hat{y}}{\delta z_{out}} = \sigma'(z_{out})$$

$$\frac{\delta z_{out}}{\delta W_h} = -2(a_h - W_h)$$

$\frac{\delta J}{\delta W^1}$ if y==0.

$$
\begin{gathered}
\frac{\delta J}{\delta W^1} = \frac{\delta J}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta z_{out}} \frac{\delta z_{out}}{\delta a_h} \frac{\delta a_h}{\delta z_h} \frac{\delta z_h}{\delta W^1} \\
\frac{\delta J}{\delta \hat{y}} = \frac{1}{ln(10)(1-\hat{y})} \\
\frac{\delta \hat{y}}{\delta z_{out}} = \sigma'(z_{out}) \\
\frac{\delta z_{out}}{\delta a_h} = -2(a_h - W_h) \\
\frac{\delta a_h}{\delta z_h} = \sigma'(z_h) \\
\frac{\delta z_h}{\delta W^1} = -2(X - W^1)
\end{gathered}
\tag{A.6}
$$

$\frac{\delta J}{\delta W^1}$ if y==1.

$$
\begin{aligned}
\frac{\delta J}{\delta W^1} &= \frac{\delta J}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta z_{out}} \frac{\delta z_{out}}{\delta a_h} \frac{\delta a_h}{\delta z_h} \frac{\delta z_h}{\delta W^1} \\
\frac{\delta J}{\delta \hat{y}} &= -\frac{1}{\hat{y} \cdot ln(10)} \\
\frac{\delta \hat{y}}{\delta z_{out}} &= \sigma'(z_{out}) \\
\frac{\delta z_{out}}{\delta a_h} &= -2(a_h - W_h) \\
\frac{\delta a_h}{\delta z_h} &= \sigma'(z_h) \\
\frac{\delta z_h}{\delta W^1} &= -2(X - W^1)
\end{aligned}
\tag{A.7}
$$

List of References

[1] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras and Tensorflow*, 2nd ed. Sebastopol, CA, USA: O'Reilly, 2019.

[2] K. Lyu and J. Li, "Gradient descent maximizes the margin of homogeneous neural networks," Institute for Interdisciplinary Information Sciences Tsinghua University, Beijing, CN, Tech. Rep., 2020 [Online]. Available: https://arxiv.org/abs/1906.05890

[3] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," Department of Computer Science, Aberystwyth University, Ceredigion, Wales, UK, Tech. Rep., 2015 [Online]. Available: https://arxiv.org/abs/1511.08458

[4] S. Wong, A. Gatt, V. Stamatescu, and M. McDonnell, "Understanding data augmentation for classification: when to warp?" University of South Australia, South Australia, AU, Tech. Rep., 2016 [Online]. Available: https://arxiv.org/abs/1609.08764

[5] J. Schlüter and T. Grill, "Exploring data augmentation for improved singing voice detection with neural networks," Austrian Research Institute for Artificial Intelligence, Vienna, Vienna, AT, Tech. Rep., 2015 [Online]. Available: https://www.ofai.at/~jan.schlueter/pubs/2015_ismir.pdf

[6] T. DeVries and G. W. Taylor, "Dataset augmentation in feature space," School of Engineering, University of Guelph, Guelph, ON, CA, Tech. Rep., 2017 [Online]. Available: https://arxiv.org/abs/1702.05538

[7] I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh, *Feature Extraction Foundations and Applications*. Heidelberg, DE: Springer, 2006.

[8] K. Ota, T. Oiki, D. K. Jha, T. Mariyama, and D. N. Nikovski, "Can increasing input dimensionality improve deep reinforcement learning?" Mitsubishi Electric Research Laboratories, Inc., Cambridge, MA, USA, Tech. Rep., 2020 [Online]. Available: https://arxiv.org/abs/2003.01629

[9] scikit-learn, "sklearn.datasets," July 25, 2022 [Online]. Available: https://scikit-learn.org/stable/modules/classes.html#module-sklearn.datasets

[10] Center for Machine Learning and Intelligent Systems, Bren School of Information and Computer Science, University of California, Irvine, "Machine Learning Repository," Accessed July 2022 [Online]. Available: https://archive.ics.uci.edu/ml/datasets.php

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California