LEVERAGING HIDDEN NODE LAYERS WITHIN NEURAL NETWORKS FOR MNIST DATA DIGIT CLASSIFICATION IN MACHINE LEARNING

Jamia Russell[1,†]

[1] Northwestern University School of Professional Studies
Master of Science in Data Science Program
633 Clark St. Evanston, IL 60208

Address to which correspondence should be addressed:
jamialashe@gmail.com

Abstract

Exploring the performance and behavior of neural networks as it relates to digit classification using MNIST data, this paper evaluates how hidden layer nodes within neural networks characterize aspects of input data and determine the difference in the impact of hidden nodes on model performance. Dimensionality and feature selection are utilized to manipulate network architecture to present the most effective and efficient model classification for hand-written numbers. Specifically, preprocessing techniques Principal Component Analysis (PCA), and Random Forests are used to further evaluate and optimize model performance.

This paper examines the model performance through backpropagation, revealing how interacting with neural networks through sequence of experiments each altering the node amount and effectively changing the function of and results provided with the aim to improve predictions and better classify digits over time.

The model itself was created and examined using Python Scikit Learn, TensorFlow, and Keras packages. Key findings uncovered that increases in hidden nodes improve model performance causing model accuracy in classification of written numbers to increase.

Results indicate that the Principle Component Analysis (PCA) reduction network presented the best comparability accuracy and efficiency.

.

Keywords: Neural Networks, Hidden Layer Nodes, MNIST Digit Classification, Principal Component Analysis, Random Forest

Table of Contents

I. Introduction & Problem Statement

In a period where digitized forms of information collection and sharing are considered most efficient and preferred by a majority of the population, automation and classification of handwritten numbers holds extreme usefulness and value to many industries, decreasing manual labor hours and increasing productivity and resolve. A primary example of this is Optical Character Recognition (OCR), which is used to recognize printed or handwritten text and characters providing the capability of searching for specific terms, phrases, number clusters, and symbols that can represent people, themes, objects, and numerical identifiers.

In addition, the legal industry often uses web based e-discovery platforms that allow for categorizing documents by topic, date, and sender in addition to the ability to search documents for specific details that may support case arguments. These platforms rely on handwritten character and digit recognition to designate use and storage of legal documents which can include sensitive handwritten financial information and notes.

Using the MNIST data set the goal to develop a neural network classifier model able to predict and classify with accuracy handwritten digits (0-9) the use of back-propagation is critical. Back-propagation offers progressive improvement on accuracy when dealing with large data sets. In a system of trial and error where recognizing numbers deriving from individuals with differing chirography, neural networks take on the task of learning from these differences which can be regional, digit presentation using the left or right hand, or style of native language.

An existing challenge outside of training the neural network, is interpreting hidden node behavior revealing the effectiveness of the internal feature process. By manipulating hidden node layers with consistent input data insight is gained on how the network responds.

II. Literature Review

Researchers have no shortage of character and digit availability to limit the amount of insights to build from numerical and textual data – especially as it relates to automation and recognition for performance improvement. Many resources have been used before to assess neural network behavior and performance for digit classification. For instance, Zhu (2018) used multi-layer neural networks to determine classification through training and plot of loss. Kaesner (2013) used Back Propagation Neural Networks (BPNN) with three different methods to determine the effect of input parameters on performance and recognition rate of handwritten digits.

Neural Networks and Digit Classification

Classification of text and characters using neural networks is a popular area to investigate using deep learning techniques. In 2017, Guo et al. investigated deep learning and convolutional neural network design and framework's effect on feature selection. The MNIST data was input through three layer types: convolutional, pooling, and fully connected. They found that network depth, computational efficiency, optimization and activation packages ReLU and Dropout can improve the convolutional neural network performance in image classification (Guo et al., 2017).

Zhao et al. (2019) used MNIST data to train the then digits classified as (0-9) with a CNN based feature extraction with an algebraic fusion of multiple classifiers trained on different feature sets.The authors used CNNLeNet-5 to extract features and an ensemble learning framework for classification containing fully connected layers. From there, learning algorithms are training base classifiers on the feature set creating an ensemble from the features in the primary fusion layer.

ReLU and Softmax Activation

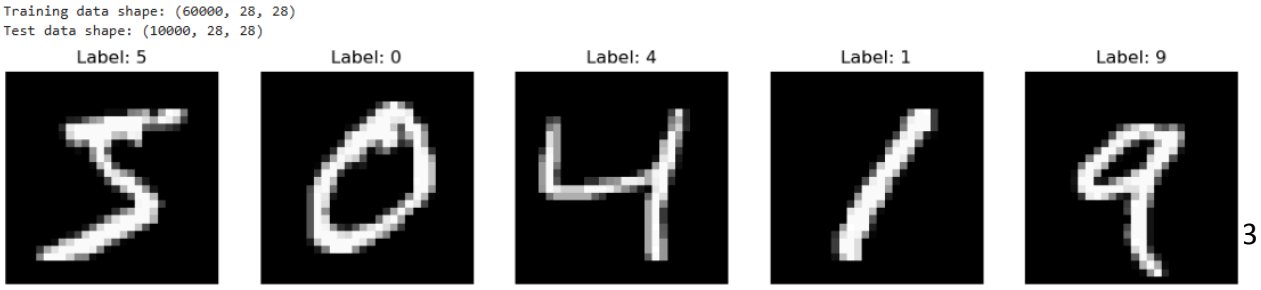
Throughout the literature the importance of non-linearity to capture patterns and relationships between the input is stressed when aiming to learn from model results. Non-linearity offers complex problem solving for image processing and speech recognition (Nair, 2023). To achieve this activation functions are used to learn about model performance and adjust weight and bias to prevent error in prediction. Rectified Linear Unit (ReLU) and Softmax used in combination in the hidden layer and output layer transform the input to a computationally efficient form allowing the output vector to represent the probability of a given class.

In 2020, Asadi and Jiang analyzed neural network approximation power using ReLU activation and Softmax output layer. They found that on a significantly large network implementing ReLU or softmax can approximate integral and indicator functions. Results being arbitrary to the size of the network. Findings show that ReLU and softmax are capable of use in multi-class classification and approximation problems that reflect real life.

III. Methods

Overview of Data

For this model and sequence of experiments Modified National Institute of Standards and Technology (MNIST) data containing 28x28 pixel images of handwritten digits from 0-9 were used. This size and extensive pixel values of the data made it the perfect choice for these experiments. Accuracy and ease of the data allows assessment of the performance of different methods and machine learning techniques to draw insights relating to image processing and text identification.



In machine learning, diverse and accurate data serves as a benchmark for training algorithms providing a comprehensive analysis of model and network performance and informing the efficacy of model experimentation. In addition, compared to larger more complex data MNIST proves itself to be an excellent learning tool for beginner analysis.

To facilitate data use within this model Python was used to preprocess the data. This included splitting the data into training and validation sets, normalization, and one-hot encoding. Image data was divided by the maximum pixel value (255) normalizing the data to values between 0 and 1. Target classes were encoded to allow for multi-class identification through labeling. From there, the images originally 2D arrays with 28x28 pixels are flattened to a singular 784 row 1D array treating each pixel as an individual feature. The 784 elements represent the intensity of each pixel and allow for input into dense neural networks that anticipate one dimensional inputs.

These preprocessing steps assured network model inputs were consistently formatted and provided successful training of the neural networks.

Network Construction

Python packages Keras and TensorFlow are combined to produce the model for the neural network. The model code contains a singular argument allowing for customization of neurons within the hidden layers. Using Keras Sequential API the model stacks layers in a linear order inputting the model to the first layer where it is flattened and reshaped. The input layer contains 784 nodes reflecting the pixel values of each image and an output of 10 nodes. To determine the effectiveness of the model, experiments with varying nodes ranging from 0 to 100 within the hidden layer were altered to examine node impact on model performance.

Back-propagation paired with an Adam optimizer trained the model enabling the cross-entropy loss function.

The first dense layer with a modifiable hidden node parameter, implementing the Rectified Linear Unit (ReLU) activation function, provided non-linearity to the network by outputting 0 or 1 values based on positivity. ReLU activation is a popular choice as it enables models to determine patterns and avoid vanishing gradients. The second dense layer, the output layer contains 10 nodes representing each output class. Softmax is used in this layer to predict output values across the 10 classes, depicting the model's confidence that the image is categorized correctly.

Experimental Methods

A sequence of five experiments were conducted to examine the network's behavior. The first experiment contained a single hidden node analysing activation values and providing insights on the node’s capacity to present similar features across classes and if present the overlap between activation ranges.

Experiment 2 built off the network by adding to the hidden layer amount. Two nodes were input while input and output layer configurations remained the same.The model is evaluated after each of the 10 epochs and metrics are provided for training and validation sets including accuracy and loss comparison and success in the activation function and hidden layer ability to separate inputs into digit classes.

Experiments 3-8 trained a series of models using hidden layers containing nodes ranging from 0-100. The model configurations were maintained focusing on hidden layer size impact on the network’s ability to learn from and classify input data.

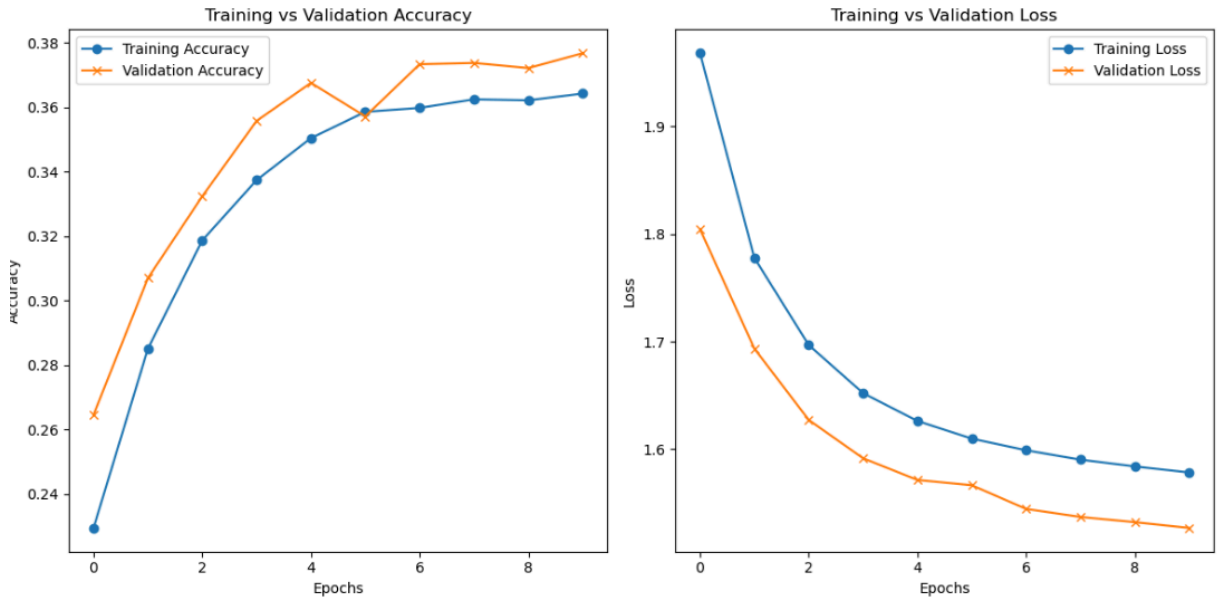
Experiment 9, altering dimensionality and determining the impact it has on network performance the MNIST images are re-configured to lower dimensional renditions using Principal Component Analysis (PCA). Images were reduced to 154 components aiming to capture 95% variance in the original training data. PCA is implemented to simplify the input data by removing repetitive and low variance features improving computational efficiency while also reducing overfitting. Focusing on notable features the model processes the data more efficiently overlooking irrelevant details.

Experiment 10, examines feature selection using a Random Forest classifier to determine the essential input features for image classification and how limiting the input space affects the network's performance. The 784 dimension dataset is used to train the random forest and provide the importance of each feature. The model determines the top 70 influential features. From there, combined with the best performing node amount in experiment 3 the model is re-trained using the 70 features as input. Estimator scores were used to determine these features, reducing the dataset and creating a new neural network with a single 32 node hidden layer implementing ReLU activation and an output layer of 10 utilizing softmax for multi-class identification. To observe accuracy and loss for training and validation the Adam optimizer and categorical cross-entropy function were trained on the datasets.

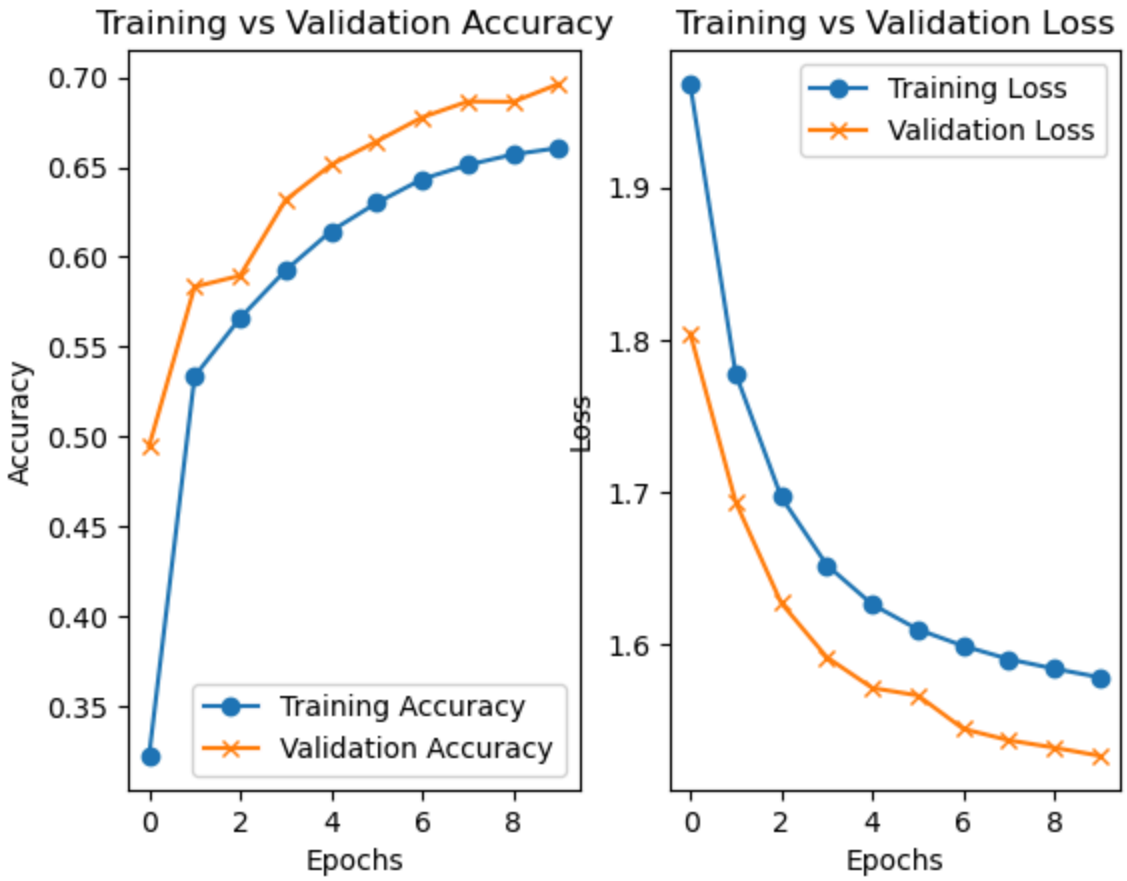
IV. Results

The network training log for experiment one reveals the accuracy and efficiency of the network's performance over 10 epochs. The model processes the entire training set delivering a 22.95% training accuracy and 26.46 validation accuracy during the first epoch. Losses are 1.9769 for training and 1.8043 for validation showing that the model has not learned from the

input features. By the 10th epoch the model has learned from the training, and accuracy and loss metrics improve slowly to 36.43% and 1.5785 respectively. Improvement in results indicate the model is effectively learning patterns within the data. Despite this, validation accuracy remaining higher than training across select epochs reveals the models have not reached optimal performance. In addition, model accuracy and loss diminishing improvements reveal it approaches a plateau as it moves towards optimal weight suggesting underfitting and insufficient capacity to capture full detail of the data. At this level increasing epoch number may contribute to performance improvement as the model seems to not yet fully stabilized.

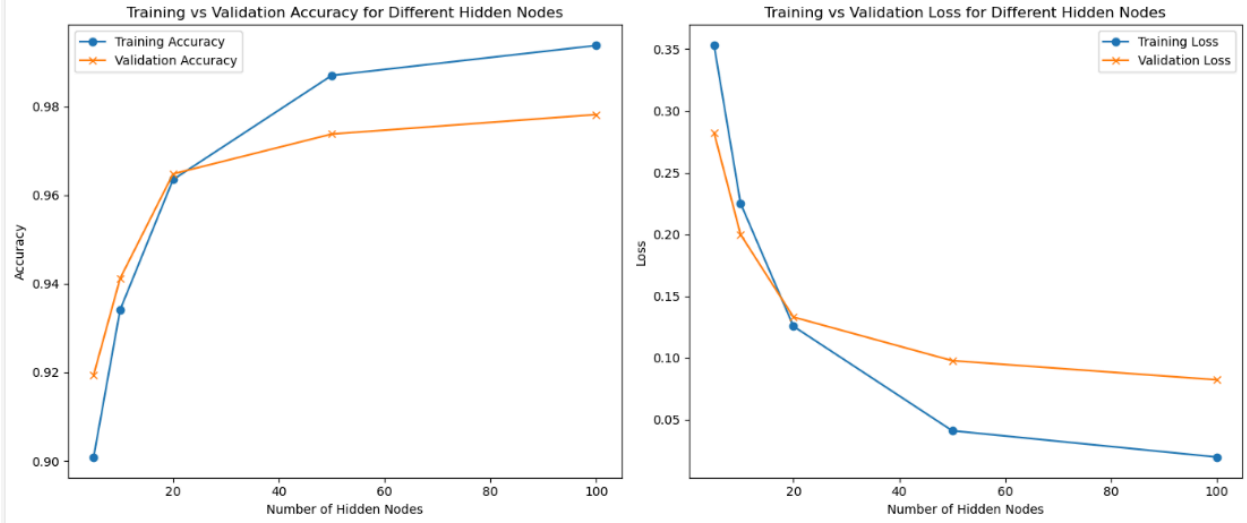


During experiment two there are improvements in training and validation metrics over the 10 epochs with the first epoch's training accuracy 32.22% and validation accuracy 49.46%.This demonstrates the network's progress in learning patterns from the data. Training and validation losses at 1.7653 and 1.403, respectively reflect the model's error minimization. As the model epochs increase and weight adjustments take place metrics improve, suggesting model optimization and generalization performance is suited between training and validation data.

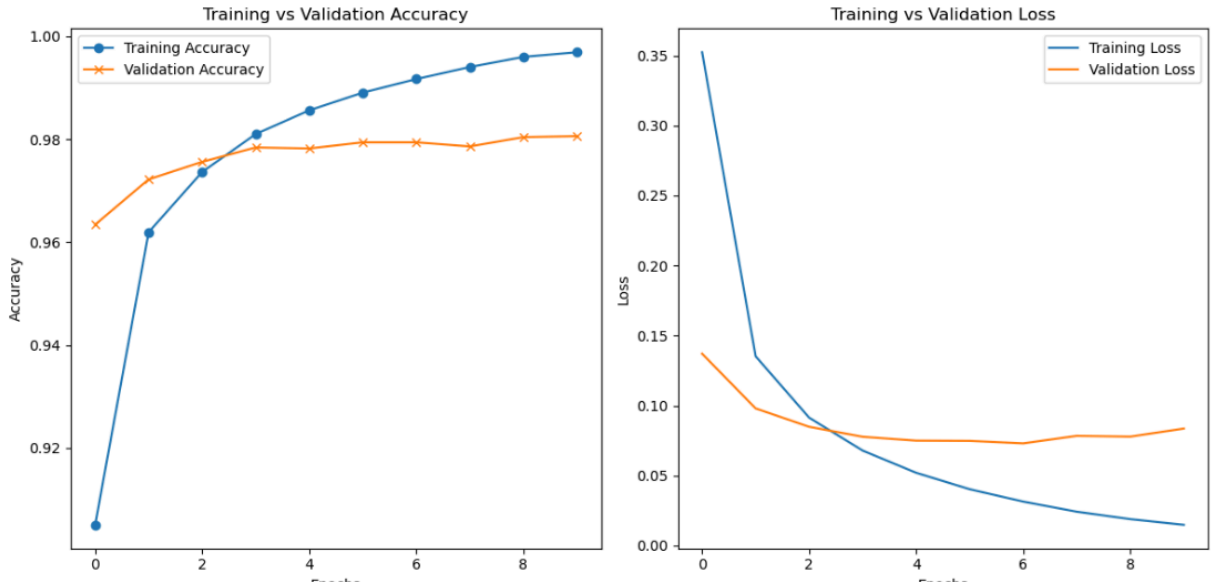


Experiments 3-8, investigate the impact of the network with nodes spanning from zero to one hundred within the hidden layer on training and validation accuracy. The 5 varying hidden node amounts depict the relationship between an increased node amount and accuracy metrics. At 5 hidden nodes the network reaches a training accuracy of 90.09% and validation accuracy of 91.94%, here the model shows itself capable of identifying basic patterns in the data, but meets difficulty when approaching complex features due to the limiting number of nodes. At 20 nodes, training accuracy increases by a small margin to 96.35% and validation jumps to 96.48%. At this stage the network identifies more complex patterns and increases precision in classification. Increasing the nodes to 50, the model training and validation accuracy improve significantly reaching 98,70% and 97.38 respectively proving the model successful in learning and generalizing input data. 100 nodes offers a near perfect training accuracy at 99.38%, while

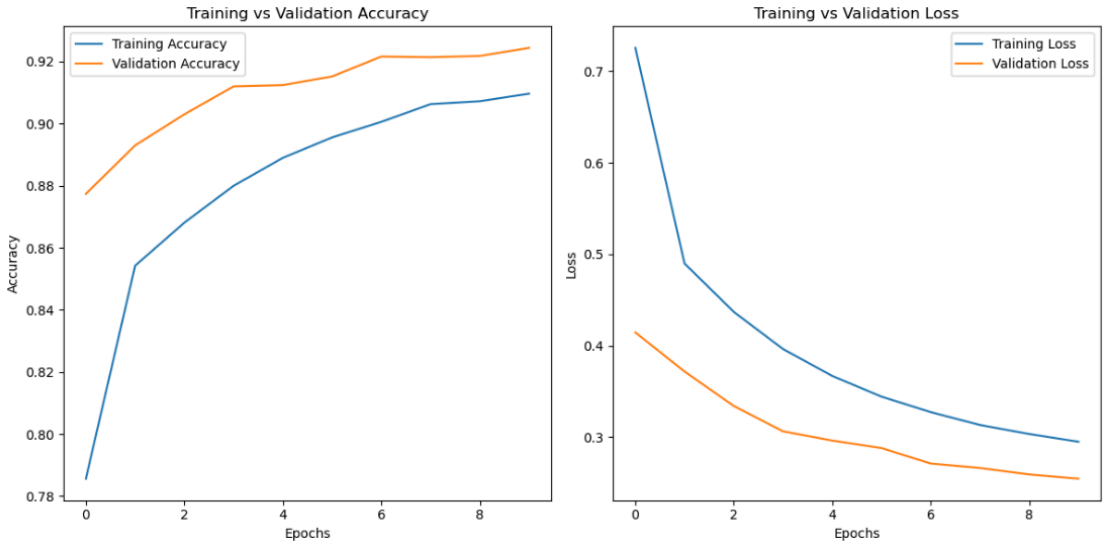
validation accuracy is 97.82%. The model now has the capacity to identify finer details of the data. However, the small margin of increase suggests that increasing complexity may lead to diminishing returns for generalization which may lead to overfitting. The incremental changes in nodes to the hidden layer refines model performance and learning ability increasing complexity and efficiency while simultaneously mitigating overfitting.



The use of Principal Component Analysis (PCA) in experiment 9 highlights the effectiveness of dimensionality reduction in the network's classification of handwritten numbers. Trained on PCA data containing 95% of the dataset's variance, the network's first epoch achieved a training accuracy of 90.50%, increasing to 99.69% by the 10h. Validation accuracy increased from 96.34% to 98.06% across all 10 epochs indicating the reduced input space increased model generalization. With steadily decreasing training and validation losses the model demonstrated its ability to generalize unseen data without overfitting. Decreasing the dimensional input from 784 to 154 principal components we see that training per epoch decreased whilst not sacrificing accuracy. This indicates PCA reduced data robustness allowing the model to distinguish significant patterns without overwhelming complexity.



The last experiment evaluated the effect feature selection using a Random Forest Classifier on the 70 most important feature pixels. The results indicate that strong classification performance improved with feature selection. The model's accuracy and validation increased from 79.09% and 73.38%, respectively to 91.46% and 98.0%. Increases across the 10 epochs depict the random classifiers effect on the models' ability to generalize unseen data and learn from reduced feature sets without overfitting.



V. Conclusion

The sequence of experiments provide an extensive understanding of neural networks and the process of classifying handwritten data. The research informs how structural choices, data preprocessing, feature selection, and dimensionality reduction impact model performance. Exploring node variation within the hidden layer, dimensionality with PCA, and Random Forest feature selection, we observed and analyzed the fluctuations of computational efficiency, performance, and complexity. The experiments revealed a higher number for hidden nodes within a single layer network increased training and validation accuracy across 10 epochs.Larger hidden layer networks were able to precisely classify inputs across complex features. PCA dimensionality reduction's high validation and training accuracy emphasize the role of simplifying dimensional data whilst preserving critical information. With Random Forest Classification, we see the 70 most important features informing the model's sensitivity to design.

```
Final Model Accuracy: 0.973800003528595
PCA Model Accuracy: 0.9782999753952026
Random Forest Feature Model Accuracy: 0.9243999719619751
```

Experiment results demonstrate the power in versatility that neural networks are able to offer. At every stage, data preprocessing, architecture construction, and node variation testing, selected inputs inform model performance, computational efficiency, and the optimal solution. Thus, results determine the PCA model to be most effective in accurate handwritten digit classification.

References

Asadi, B., & Jiang, H. (2020, February 10). *On Approximation Capabilities of ReLU Activation and Softmax Output Layer in Neural Networks*. ArXiv.org. https://doi.org/10.48550/arXiv.2002.04060

Chayaporn Kaensar. (2013). *Analysis on the Parameter of Back Propagation Algorithm with Three Weight Adjustment Structure for Hand Written Digit Recognition*. https://doi.org/10.1109/icsssm.2013.6602610

Guo, T., Dong, J., Li, H., & Gao, Y. (2017). Simple convolutional neural network on image classification. *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)(*. https://doi.org/10.1109/icbda.2017.8078730

Nair, G. (2023, August 22). *Activation Function in Neural Networks: Sigmoid, Tanh, ReLU, Leaky ReLU, Parametric ReLU, ELU, Softmax, GeLU | Medium*. Medium. https://medium.com/@gauravnair/the-spark-your-neural-network-needs-understanding-the-significance-of-activation-functions-6b82d5f27fbf

Zhao, H., & Liu, H. (2019). Multiple classifiers fusion and CNN feature extraction for handwritten digits recognition. *Granular Computing*. https://doi.org/10.1007/s41066-019-00158-6

Zhu, W. (2018). Classification of MNIST Handwritten Digit Database using Neural Network.