Exploration of Neural Network Architectures and Hyperparameters

Donovin Emerson

Artificial Intelligence and Deep Learning

MSDS – 458

Abstract

This study uses the MNIST handwriting dataset to explore the performance effects of architectural changes and dimensionality reduction techniques in simple neural networks (NN) with a single hidden layer and small number of nodes {1, 2, 64, 512}. Activation weights are examined using four visualization techniques including t-SNE and gradient ascent. Hyperparameters are analyzed statistically using ANOVA. As expected, the number of nodes is important however, there was a diminishing return when additional nodes were added. Intriguingly, the sigmoid activation function was a statistically significant leading negative effect against both test accuracy and test multi-class F1 scores. Further, methods are suggested to improve the performance of NN – namely, increasing training data through augmentation and decreasing model complexity.

Introduction

Identifying objects in an image and assigning them to a class is an important but difficult task to explicitly program a computer to complete and a particularly challenging subset of this problem is the classification of handwritten characters (LeCun et al., 1998). This work is motivated by the wide-ranging applications of image classification applied to handwriting. For example, the United States Postal Service routes mail by reading handwritten addresses, the Internal Revenue Service digitizes handwritten paper forms, and hospitals digitize handwritten patient forms or medical records (Internal Revenue Service, 2021; Jaszczak, 2011; Rasmussen et al., 2011). It is easy to see the importance and value of enabling computers to “read” handwriting when considering the millions of documents that would otherwise be keyed into a system by hand.

Thomas and Rajan (2019) suggest that handwriting styles are exclusive to each individual and, when combined with the myriad of available writing instruments in various thicknesses and styles,

classifying handwriting stands out as a unique challenge to solve programmatically. A successful solution must find a way to generalize the outcome of these many disparate writing styles and instruments to accurately classify examples it has never seen before. A NN can model complex decision surfaces and classify high-dimensional patterns which makes them well suited to classify handwriting since they are able to learn by example though low precision inputs and require little preprocessing to produce "confident" predictions (Denker et al., 1988; Lecun et al., 1998).

While there are many methods to predict individual handwritten characters with reasonable accuracy (>95%), the focus of this paper is to explore the effects of architecture and additional hyperparameter modifications using design of experiments (DOE) on the simplest NN possible: a single hidden layer feed forward NN with backpropagation trained on the MNIST dataset (Baldominos et al., 2019).

Literature Review

The benefit of using the MNIST dataset to explore neural network architecture and other hyperparameters is that this dataset is well studied and generally considered a "solved" problem (Baldominos et al., 2019). Simard et al. (2003) suggested that performance of a NN is directly linked to the size of the training set – more training data produces better results. They suggested that the MNIST training set is not large enough for models to accurately generalize without data augmentation. Many studies using the MNIST dataset employed data augmentation techniques by creating "new" samples through distortions like slight skewing, warping, rotating, or shifting (Baldominos et al. 2019; Simard et al., 2003). Expanding the training set takes advantage of the property LeCun et al. (1998) described where the gap between the test set error E_{test} and the training set error E_{train} can be reduced simply by increasing the amount of training data approximately as

$$E_{test} - E_{train} = k(h/P)^{\alpha}$$

P represents the number of training samples, *h* is a measure of "effective capacity" of the system, *k* is a constant, and $\{\alpha \mid \alpha \in \mathbb{R}, 0.5 \leq \alpha \leq 1\}$. They described an alternate method to close this gap by increasing the effective capacity, *h*, which is the complexity of the system however, there is a tradeoff that occurs because an increase in *h* decreases E_{train}.

It is well defined in the literature that for image classification problems, convolutional neural networks (CNN) outperform NN by exploiting the "input topology" or spatial relationship among pixels (Baldominos et al., 2019; LeCun et al., 1998; Simard et al., 2003). However, reasonable classification accuracy is attainable with a small number of nodes in a single hidden layer. In fact, more layers and nodes can degrade performance through exponentially vanishing gradients – bigger does not always mean better (Cireşan et al., 2010).

Methods

The MNIST dataset containing 60,000 training and 10,000 test images of individual handwritten Arabic numbers from zero and nine was used in this study (Deng, 2012). Over 250 participants submitted handwriting for the training set (NIST Special Database 1) and none of the samples in the test set (NIST Special Database 2) include handwriting from any participant that contributed to the training set (Baldominos et al., 2019).

Training and test data were restricted to the publicly available Keras MNIST dataset with 60,000 training and 10,000 test samples with 5,000 samples from the training set held out as a validation set. All pixel data were normalized $\{x \mid x \in \mathbb{R}, 0 \leq x \leq 1\}$. No additional training examples were generated through augmentation or other means.

To determine the impact that the number of input features may have on predictive power, two dimensionality reductions approaches were used – principal component analysis (PCA) and random forest (RF) feature importance. MNIST data have 784 input features when vectorized (28x28 pixels). Using PCA, the first 154 principal components explained approximately 95% of the variance in our training images which reduced the total number of input features to our NN by 630 features. Using random forest, the 70 most important features were extracted which reduced total input features by 714.

Six experiments were conducted:

Experiment 1: single hidden layer with one node, ReLU activation, optimized with Nadam, and trained on data without dimensionality reduction.

Experiment **2** same network as above with the addition of a second node to the hidden layer.

Experiment **3** best model identified included 64 hidden layer nodes, used ReLU activation, was optimized with Nadam, and trained on data without dimensionality reduction.

Experiment 4 first 154 principal components the number of input features was reduced to 154 using PCA and was compared to the best performing model without dimensionality reduction identified in experiment three as well as the RF model from experiment five

Experiment 5 random forest feature importances used to identify the 70 most important features and compared to models in experiment three and four.

Experiment 6 incorporated full factorial mixed level design of experiments to search the design space and evaluate the best performing small network with and without dimensionality reduction. All other hyperparameters were fixed at the TensorFlow Keras defaults.

Feedforward network variables included input layers with nodes {70, 154, 784} varying in size to the number of input features. A hidden layer with nodes {1, 2, 64, 512}, activated with functions {ReLU, SELU, Tanh, Sigmoid}, and a static L2 regularization penalty of 0.001 across all cases. The output layer consisted of 10 nodes for each of the digits from zero to nine using the softmax activation function. Back propagation hyperparameters included optimizers {RMSprop, SGD, Adam, Nadam} and gradient batch sizes {16, 32, 128, 256}.

Factor	Level 1	Level 2	Level 3	Level 4
Dimensionality Reduction	None	PCA	Random Forest	----------
Hidden Nodes	1	2	64	512
Activation Function	ReLU	SELU	Tanh	Sigmoid
Optimizer	RMSprop	SGD	Adam	Nadam
Batch size	16	32	128	256

All experiments were evaluated using both average accuracy and a custom multiclass F_β score using macro precision and recall measures across all batches in each epoch. To evenly weight precision and recall the β term is set to 1, which is equivalent to the F1 score typically used in binary classification. For simplicity, this measure will be referred to as the F1 score.

$$Average\ Accuracy \quad \frac{\sum_{i=1}^{l} \frac{tp_i + tn_i}{tp_i + fn_i + fp_i + tn_i}}{l}$$

$$Precision_M \quad \frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i + fp_i}}{l}$$

$$Recall_M \quad \frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i + fn_i}}{l}$$

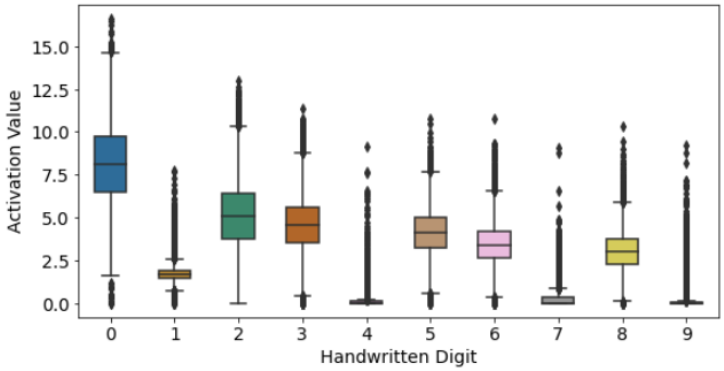
$$F_\beta = \frac{(1 + \beta^2) \cdot (\text{precision} \cdot \text{recall})}{(\beta^2 \cdot \text{precision} + \text{recall})}$$

Results were generated using Python 3.7, TensorFlow 2.3, Keras 2.4.0, cuDNN 7.6.5, CUDAtoolkit 10.1.243, jupyter_core 4.7.1, and JMP 16.2

Results

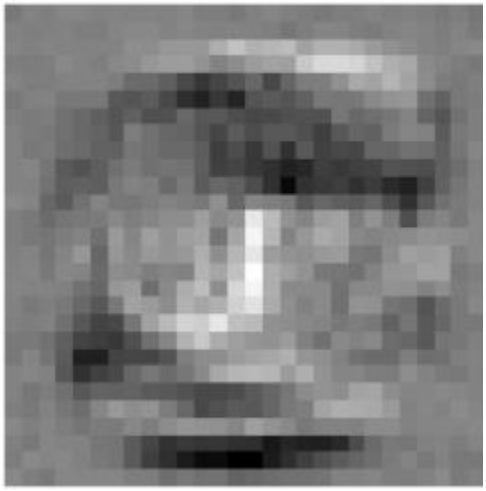
Experiment 1

Intriguingly, with only a single nonlinear decision boundary, predictive power on the unseen test data was nearly four times the benchmark of a random guess – 10%. This architecture demonstrated accuracy of 37.9% and a F1 score of 27%. Because this network used a single hidden layer node, the distributions of the activation values below have considerable meaning. If this were a perfect model, there would be very little overlap in the distributions for each digit and very likely the class could be determined by this plot without the need for a softmax output layer to compute the probability of class belonging.



Gradient ascent can be used in networks with many nodes and convolution layers but the advantage of using this approach in a single node network is that it was possible to use the activation values from the hidden layer and reverse the learning process to discover the entire input pattern that produced the greatest activation of the node. This image represents the input layer, where the vectorized image is fed into, and is the pattern that elicits the greatest response from the hidden layer. This NN asserts with a 46% probability this image represents the number three. A drawback of NN

models is that there is not a high degree of human interpretability, as humans would likely not be confidently interpret this image as a number.



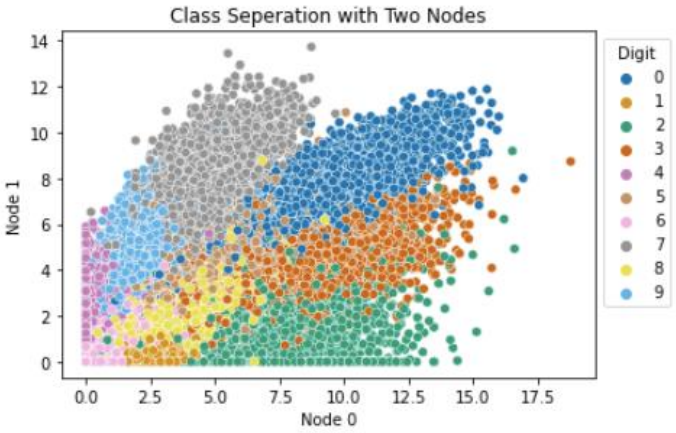
Input image recreated with gradient ascent.

Gradient ascent is achieved by iteratively adding the loss of the layer to the gradient of the error function $\nabla E\left(\boldsymbol{w}^{(\tau)}\right)$ with respect to the loss value and a learning rate η to travel up the error surface in the direction of the gradient to search for the space where the node is maximally activated across the input parameters thereby highlighting what the node is paying the most "attention" to in the input image.

$$\mathbf{w}^{(\tau-1)} = \mathbf{w}^{(\tau)} + \eta \nabla E\left(\mathbf{w}^{(\tau)}\right)$$

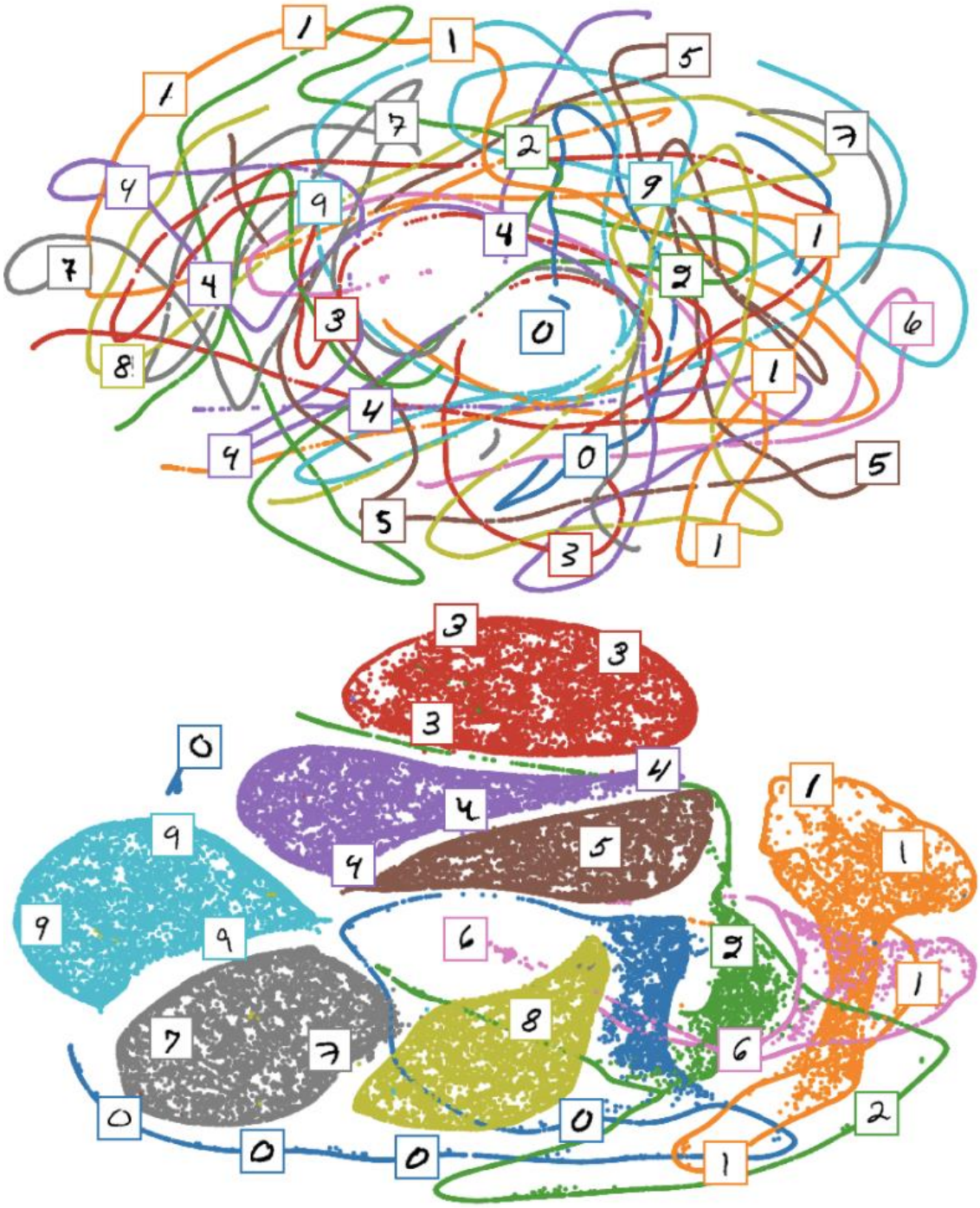
Experiment 2

Adding a second node provided an additional non-linear separation boundary on the decision surface and increased accuracy to 70.6% and F1 to 62.3%. Further, with two nodes, it was still possible to visualize the class separation with conventional plotting methods using the activation values from each node. In this visualization, some overlap occurs however, clusters can be identified. A perfect model would likely show clear separation between groups.



Activation value plot of two nodes separated by class

There are additional ways to visualize class separation as we leave the one- and two-dimensional space. T-distributed Stochastic Neighbor Embedding or t-SNE uses joint probabilities to separate classes and is favored for high dimensional classification visualizations. Below are examples of this method with results from experiment one and two that are labeled using samples of the handwritten training data they represent. The top plot is from experiment one with one-dimensional single node activation values and the bottom plot shows experiment two's two-dimensional activation values.



Experiment 1: single vector t-SNE plot (top)

Experiment 2: two-dimensional t-SNE plot (bottom)

Another important visualization method for classification is the confusion matrix. Experiment one is on the left with experiment two on the right. Experiment one failed to create the familiar diagonal

of correct predictions, called true positives, and both models struggled with accurately predicting the similarly shaped digits: zero, eight, and nine.

Confusion matrix comparison – Experiment 1 (left) and Experiment 2 (right)

Experiment 3

With 64 nodes, many more non-linear decision boundaries produced significantly-improved accuracy and F1 score – both at 97.5%. This predictive power represents a misclassification of 250 of the 10,000 images in the unseen test dataset. For the digits predicted successfully, this model displayed a high degree of confidence >99%. The confusion matrix shows the model struggled with identifying numbers two, five, and seven. For a simple model, it generalizes this complex problem well.

Experiment 4

LeCun et al. (1998) suggested that we should see an improvement in accuracy after reducing model complexity, which was confirmed, however the improvement in accuracy and F1 score was 0.2%

after a reduction in approximately 600 input parameters. Remarkably, the model was able to produce higher quality predictions with fewer inputs.

In experiment two, using two nodes, the model struggled to identify digits with loop patterns – zero, eight, and nine. This could be explained by the limited ability of two nodes to respond well enough to all needed structures in the image and be limited by their own bias. Notably, the only difference between experiment three and four is the data used to train the model and it was intriguing to find that the same model structure trained on different transforms of the same data struggled with the same digits.

Experiment 5

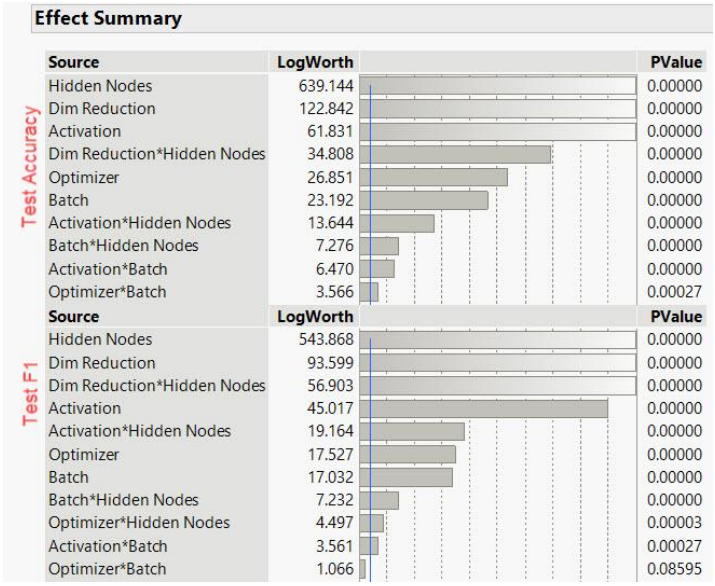
With input dimensionality halved from experiment four without the benefit of subspace mapping to that would incorporate a larger spatial area of the image than a single pixel, the proposal from LeCun et al. (1998) related to a reduction in complexity seemed to have passed an inflection point as our test accuracy and F1 scores decreased to 92.5% and 92.4% respectively. Reviewing the t-SNE plots from both experiments three, four, and five. PCA outperformed and had tight clusters with clear boundaries. Experiment three was a close second, however experiment five was noticeably less well defined and considerable overlap was observed.

Experiment 6

The best performing NN had PCA transformed data, 512 nodes, was activated with ReLU, and optimized by RMSprop using a batch size of 256. This model attained 98.3% test accuracy with an F1 score of 98.2%.

Using ANOVA though the design of experiments module in JMP, the following effect summary plot was created. The blue line denotes statistical significance, and the p-value is found along the right

edge. This chart identifies both main and interaction effects of model hyperparameters and their interactions, ranked by their impact to either test accuracy or test F1 metrics. Interestingly, activation appears near the top of the list over the learning algorithms.



Effect Summary

Test Accuracy

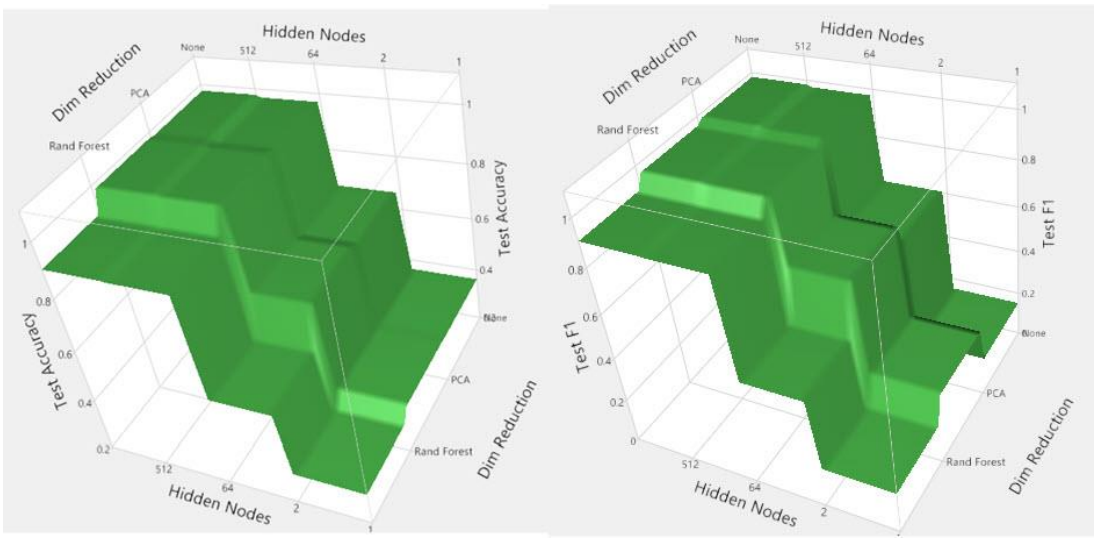
Source	LogWorth	PValue
Hidden Nodes	639.144	0.00000
Dim Reduction	122.842	0.00000
Activation	61.831	0.00000
Dim Reduction*Hidden Nodes	34.808	0.00000
Optimizer	26.851	0.00000
Batch	23.192	0.00000
Activation*Hidden Nodes	13.644	0.00000
Batch*Hidden Nodes	7.276	0.00000
Activation*Batch	6.470	0.00000
Optimizer*Batch	3.566	0.00027

Test F1

Source	LogWorth	PValue
Hidden Nodes	543.868	0.00000
Dim Reduction	93.599	0.00000
Dim Reduction*Hidden Nodes	56.903	0.00000
Activation	45.017	0.00000
Activation*Hidden Nodes	19.164	0.00000
Optimizer	17.527	0.00000
Batch	17.032	0.00000
Batch*Hidden Nodes	7.232	0.00000
Optimizer*Hidden Nodes	4.497	0.00003
Activation*Batch	3.561	0.00027
Optimizer*Batch	1.066	0.08595

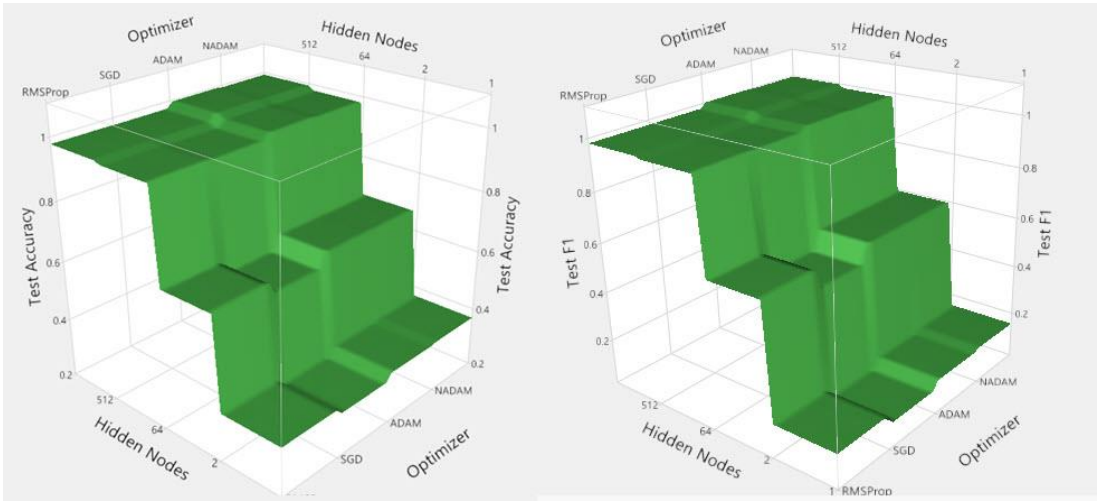
Effect Summary comparing the top 10 effects for test accuracy and F1 metrics - from JMP

Surface plots were examined, and it seemed that for MNIST, PCA provided a boost to performance when nodes remained low, but as additional nodes were added it appeared to lose its edge over the untransformed dataset. Also, there are two clear step changes between one node and 64, however, the change between 64 and 512 is almost imperceptible on these plots.



Surface plot of Hidden Nodes and Dimensionality Reduction by test set scoring metric

A similar pattern emerged with optimizers, where Adam and Nadam had an outsized performance gain in low complexity networks that seemed to diminish as nodes were added.



Surface plot of Hidden Nodes and Optimizer by test set scoring metric

The coefficient summary details the predicted magnitude that each of the levels contributed to either test accuracy or F1 score. Unsurprisingly, the number of nodes produced the greatest effect followed closely by dimensionality reduction. However, many of the top 10 effects by metric have a negative

influence on performance: one node, two nodes, RF top 70, and sigmoid activation. This plot provides additional statistical evidence of the findings from experiments one through five.

Test Accuracy Term	Scaled Estimate	Std Error	t Ratio	Prob>|t|
Intercept	0.6983281	0.001136	614.67	<.0001*
Hidden Nodes[1]	-0.379833	0.001968	-193.03	<.0001*
Hidden Nodes[64]	0.2452604	0.001968	124.64	<.0001*
Hidden Nodes[512]	0.2381094	0.001968	121.00	<.0001*
Hidden Nodes[2]	-0.103536	0.001968	-52.62	<.0001*
Dim Reduction[Rand Forest]	-0.047777	0.001607	-29.74	<.0001*
Activation[Sigmoid]	-0.036318	0.001968	-18.46	<.0001*
Dim Reduction[PCA]	0.0254688	0.001607	15.85	<.0001*
Dim Reduction[None]	0.0223086	0.001607	13.88	<.0001*
Dim Reduction[Rand Forest]*Hidden Nodes[2]	-0.036889	0.002783	-13.26	<.0001*
Optimizer[SGD]	-0.023245	0.001968	-11.81	<.0001*

Test F1 Term	Scaled Estimate	Std Error	t Ratio	Prob>|t|
Intercept	0.6342682	0.002008	315.93	<.0001*
Hidden Nodes[1]	-0.486039	0.003477	-139.77	<.0001*
Hidden Nodes[64]	0.3087786	0.003477	88.80	<.0001*
Hidden Nodes[512]	0.3015443	0.003477	86.72	<.0001*
Hidden Nodes[2]	-0.124284	0.003477	-35.74	<.0001*
Dim Reduction[PCA]	0.0627357	0.002839	22.10	<.0001*
Dim Reduction[Rand Forest]	-0.057745	0.002839	-20.34	<.0001*
Activation[Sigmoid]	-0.049727	0.003477	-14.30	<.0001*
Dim Reduction[None]*Hidden Nodes[1]	-0.063238	0.004918	-12.86	<.0001*
Dim Reduction[Rand Forest]*Hidden Nodes[2]	-0.058286	0.004918	-11.85	<.0001*
Activation[SELU]	0.0326693	0.003477	9.39	<.0001*

Conclusions

With as few as two nodes, these experiments demonstrated NN could effectively generalize written Arabic numbers with limited resources and become an accurate handwriting classifier. While these experiments explored very simple NN, there was a diminishing return when additional nodes were added to a network. It was discovered that the sigmoid activation function was one of the leading negative effects on NN models trained on the MNIST dataset.

Reducing model complexity by limiting input features through dimensionality reduction, as LeCun et al. (1998) suggested, improved performance – up to a point. To further exploit this property, data augmentation should be implemented to expand the training set. Data augmentation could grow a training set by many multiples without the time or dollar cost of securing additional samples. Additional

performance improvements could be realized through tuning hyperparameters that were not included in these experiments.

It is important to underscore, as discussed in experiment one, that NN have limited human interpretability. If a high degree of model transparency is needed, other machine learning methods like logistic regression or tree-based methods should be explored. Also, these results are specific to the MNIST dataset and these specific network parameters are not directly transferable to another problem

References

Baldominos, Alejandro, Yago Saez, and Pedro Isasi. “A Survey of Handwritten Character Recognition with Mnist and EMNIST.” *Applied Sciences* 9, no. 15 (August 4, 2019): 3169. https://doi.org/10.3390/app9153169.

Cireşan, Dan Claudiu, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. “Deep, Big, Simple Neural Nets for Handwritten Digit Recognition.” *Neural Computation* 22, no. 12 (2010): 3207–20. https://doi.org/10.1162/neco_a_00052.

Deng, Li. “The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web].” *IEEE Signal Processing Magazine* 29, no. 6 (October 18, 2012): 141–42. https://doi.org/10.1109/msp.2012.2211477.

Denker, J. S., W. R. Gardner, H. P. Graf, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, H. S. Baird, and I. Guyon. “Neural Network Recognizer for Hand-Written ZIP Code Digits.” Neural network recognizer for hand-written zip code digits | Proceedings of the 1st International Conference on Neural Information Processing Systems, January 1, 1988. https://dl.acm.org/doi/10.5555/2969735.2969773.

Internal Revenue Service. “3.41.274 General Instructions for Processing via Service Center Recognition Image/Processing System: Internal Revenue Service.” 3.41.274 General Instructions for Processing via Service Center Recognition Image/Processing System | Internal Revenue Service. Internal Revenue Service, November 21, 2021. https://www.irs.gov/irm/part3/irm_03-041-274.

Jaszczak, Kaz. “Optical Character Recognition: A Backbone for Postal and Mail Sorting Applications.” Mailing Systems Technology. Mailing Systems Technology, April 7, 2011. https://mailingsystemstechnology.com/article-2813-Optical-Character-Recognition-A-Backbone-for-Postal-and-Mail-Sorting-Applications.html.

Lecun, Y., L. Bottou, Y. Bengio, and P. Haffner. “Gradient-Based Learning Applied to Document Recognition.” *Proceedings of the IEEE* 86, no. 11 (1998): 2278–2324. https://doi.org/10.1109/5.726791.

Rasmussen, Luke V, Peggy L Peissig, Catherine A McCarty, and Justin Starren. “Development of an Optical Character Recognition Pipeline for Handwritten Form Fields from an Electronic Health Record.” *Journal of the American Medical Informatics Association* 19, no. e1 (2011). https://doi.org/10.1136/amiajnl-2011-000182.

Simard, P.Y., D. Steinkraus, and J.C. Platt. “Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis.” *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, August 6, 2003. https://doi.org/10.1109/icdar.2003.1227801.

Thomas, Mebin Wilson, and Santhosh Kareepadath Rajan. “Genuine Handwriting Variations in 10 Years: A Pilot Study.” *Egyptian Journal of Forensic Sciences* 9, no. 1 (2019). https://doi.org/10.1186/s41935-019-0154-2.