Comparing Machine Learning Methods in their Application to the MNIST Dataset

Spencer Lee
Department of Electrical Engineering
University of British Columbia
Vancouver, CA
SID: 88207360

Abstract

The MNIST database (Modified National Institute of Standards Technology) is a large collection of handwritten digits commonly used for training and testing the efficiency of different machine learning methods. This work applies different methods of machine learning and deep learning algorithms to the database. The performance and computational efficiency of these methods are compared to each other and to themselves when different data preprocessing methods are introduced. Optimal hyper-parameters for each method are presented alongside their performance metrics.

1 Introduction

In machine learning, classification is a common application. Many different methods are efficient at sorting datasets into a set of categories through either supervised training or unsupervised training. Given the many different classifiers, the MNIST database [1] is an ideal testing setup since many different methods can be used to classify the handwritten digits. With each classifier, there are also many different hyper-parameter choices, structures, and preprocessing that change performance and computational time. The comparison between different classifiers under different setups gives a perspective on when certain methods are better than others and under what circumstances.

Neural networks have recently been hailed as the top performer in classifying the MNIST dataset [2]. However, there are a few drawbacks of using neural nets. Their data structures make it very difficult to interpret classification reasoning; it is difficult to parse what features have the largest impact without post-processing of the neural net structure. This is very different than regression classification where it can be easily understood what features impacted classification based on weights and feature values. An important aspect of machine learning methods is the interpretability of them. In the application of machine learning, it is important to understand the trade off between performance and complexity. A more complex model is not necessary if it offers the same performance as simpler models.

Another benchmark related to complexity is the computational time required to apply a machine learning method. With the rising popularity of machine learning, many users might not have access to computers with high-level CPUs or GPUs. Not all machine learning methods require the same level of computation, which means that in order to select and apply classifiers, it is necessary to understand what options are available and what options are restricted by high computational load.

As an overview to the contents of this work, eight different learning environments are applied to the MNIST dataset and compared in both accuracy and computation time. The eight methods involve two neural net structures, 4 decision tree structures, and two regression structures.

2 Methods

2.1 Neural Nets

2.1.1 Basic Neural net

The MNIST dataset is composed of thousands of handwritten digits, which come in the form of 28x28 grayscale bitmaps as seen in Figure 1. Each pixel has an integer value between 0 and 255, while the integer itself can range between zero and ten. These three dataset factors influence the structure of the suited neural net. The input must be able to take 28*28 values, which are first normalized on a continuous range between zero and one. This is done by simply dividing the pixel-value by 255. The output must also be able to distinguish ten different values. This is implemented with 10 output nodes, which form a one-hot encoded vector of length 10.

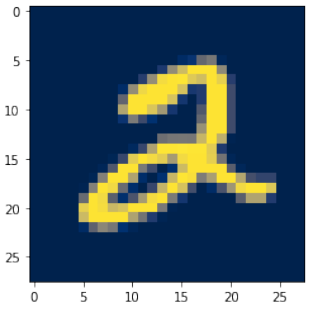

Figure 1: Example of MNIST data input visualized.

The neural net uses one input layer, 2 ReLU layers, and one output layer. The number of nodes in the hidden layer was optimized by testing different numbers of nodes in either layer. Four different layouts were used, where each hidden layer had either 256 or 512 nodes. The neural net was trained over the course of 5 epochs, which is sufficient given the size of the dataset.

2.1.2 Convolutional Neural Net

A standard convolutional neural net (CNN) uses feature extraction on the data before it performs classification. Similar to the basic neural net, the data is preprocessed to be normalized to a unit magnitude. The output data (labels) are also preprocessed with one-hot encoding. The last three layers in the CNN are the same as the basic neural net, two hidden ReLU layers with a SoftMax output layer. The feature extraction portion of the CNN involves two convolution layers and two sub-sampling layers. The two convolutional layers use ReLU activation and the sub-sampling layer uses a max pooling down-sampling method to reduce the dimensionality of the data. This layout of convolution and sub-sampling should introduce some degree of spatial invariance and allow for the agent to learn more robust features that are less sensitive to small translations or deformations in the input data.

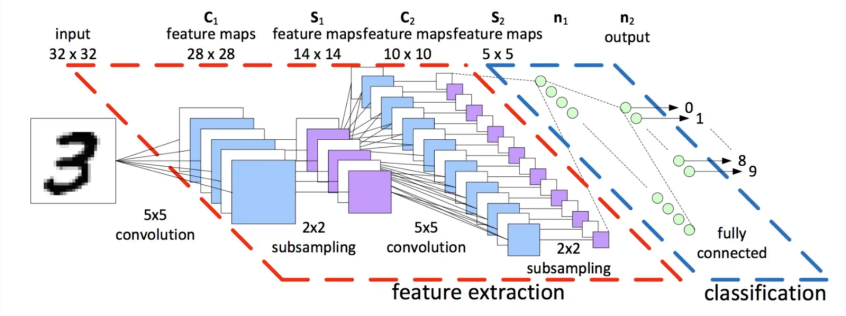

Figure 2: CNN has multiple layers involving convolution and sub-sampling, which allow for feature extraction.

2.2 Decision Trees

With 784 features in a single 28x28 image input, a decision tree (DTR) would be far too deep and complex. It would be prone to overfitting and result in a low accuracy. This means that a high level of dimensionality reduction is needed to simplify the number of features. This method is analyzed because constructing a basic DTR is relatively quick and doesn't require much computational power. The single DTR offers a perspective on the trade-offs between performance and computational speed. In order to analyze the performance between different DTR structures and the performance between them and other machine learning methods, four different levels of complexity are tested.

The first model is the basic singular DTR with no dimensionality reduction, which uses all 784 pixel features. This model also limits the depth of the tree to avoid the aforementioned problem with complexity. The second model is a DTR made from compressed MNIST data, but using the same construction method as the first model. The third model is a DTR also made from the compressed MNIST data by using t-distributed stochastic neighbour embedding (t-SNE). The final DTR structure is a random forest, which is an ensemble method that should produce a very high accuracy at the cost of high computational time.

The method of dimensionality reduction is truncated singular value decomposition (TSVD), which is written as:

$$A = U\Sigma V^T \tag{1}$$

where A is an $m \times n$ matrix, U is an $m \times m$ matrix, Σ is an $m \times n$ diagonal matrix, and V is an $n \times n$ matrix. The columns of U and V are called the left and right singular vectors, respectively, and the diagonal entries of Σ are called the singular values.

Truncated SVD is a variant of SVD that is used to approximate the original matrix A by retaining only the top k singular values and corresponding singular vectors. It can be written as:

$$A \approx U_k \Sigma_k V_k^T \tag{2}$$

where U_k is an $m \times k$ matrix, Σ_k is an $k \times k$ diagonal matrix, and V_k is an $n \times k$ matrix. Truncated SVD is often used for dimensionality reduction, as it can effectively capture the most important features in the original matrix while significantly reducing its size.

There are two main reasons why this dimensionality reduction technique is used. It is a linear method, which means it does not require any complicated non-linear transformations of the data. This makes it easy to interpret and understand the resulting reduced dimensions. The second reason is that it can handle missing values and handle large sparse matrices effectively.

2.3 Regressions

The last machine learning method compared is more of a statistical method. Regression analysis is used to provide a baseline model. It's unlikely to be nearly as accurate as the other methods but is computationally simple and very easy to interpret machine decision making. Specifically a basic logistic regression model is created and compared to itself when using a raw MNIST dataset or a scaled MNIST dataset.

3 Results

3.1 Neural Nets

Performance results indicate that training a basic neural network on the MNIST dataset typically results in a model that can achieve high accuracy on the test set. As visualized in Figure 3, the basic neural network achieves an accuracy of $\sim 97.7\%$ within five training epochs.

An epoch is a measure of the number of times the model has seen the entire training dataset. In general, the more epochs a model is trained for, the better it will perform on the training data. However, training for too many epochs can lead to overfitting, where the model starts to memorize the training data and performs poorly on unseen test data. This is likely what is happening as the accuracy for the basic neural network declines for the third epoch.

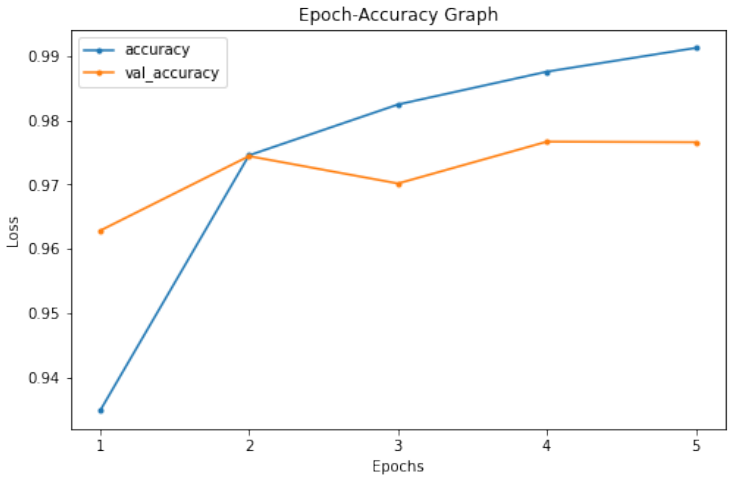

Figure 3: The basic NN training accuracy continues to increase with the number of training epochs even though validation accuracy levels off.

As summarized in Table 1, any moderate number of hidden layer nodes allows the basic neural network to achieve a proficient accuracy. The number of hidden layer nodes does not have a significant impact on the model's performance.

Table 1: Performance of the basic neural net with different hidden layer nodes.

Nodes (Layer 1, Layer 2)	**Accuracy**	**Computation Time**
(256, 256)	0.9758	11.21s
(512, 256)	0.9767	21.99s
(256, 512)	0.9768	16.52s
(512, 512)	0.9782	28.03s

As seen in Figure 4, the training accuracy of the CNN after one epoch is similar to the basic neural network. However, the validation accuracy is constantly high throughout all the training, indicative of the model's robust nature. Similarly, the validation accuracy does not vary as widely as the basic neural network does throughout its training, which is likely due to its resilience to overfitting.

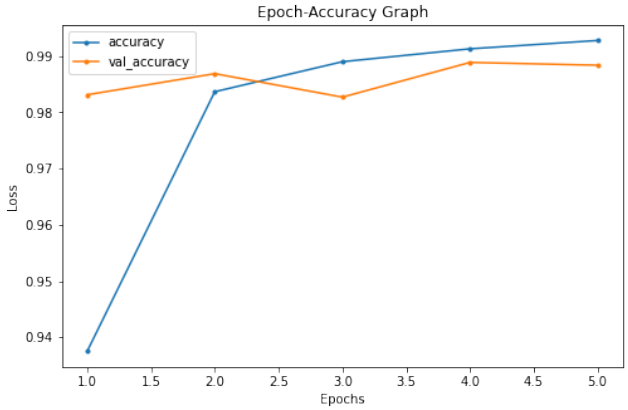

Figure 4: Similar to the basic NN, the CNN training accuracy continues to increase with the number of training epochs even though validation accuracy levels off. The CNN validation accuracy begins very close to its maximum accuracy in contrast to the basic NN.

3.2 Decision Trees

Using a basic singular DTR, the training accuracy was $\sim 98.4\%$ and the testing accuracy was $\sim 88.29\%$. This was done using all pixel features. The optimization for tree depth was done using a cross-validation method visualized by Figure 5. The decline after a depth of 15 is likely due to overfitting, similar to the neural network.

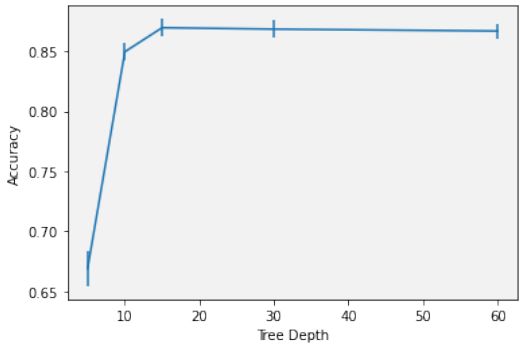

Figure 5: The accuracy of a DTR increases drastically until a depth of $\sim$15 where it begins to slowly decline.

When a DTR is created with a dimensionality reduction, the training accuracy with TSVD features (50 components) is $\sim 99.82\%$, and the validation accuracy with is $\sim 85.23\%$. This is a difference of -2.46% in validation accuracy compared to the baseline. However, this simply provides a baseline to compare with the t-SNE DTR. The first step of training the t-SNE DTR is the unsupervised training. This clusters data into a two-dimensional visualization that qualitatively shows its performance. This can be seen in Figure 6.

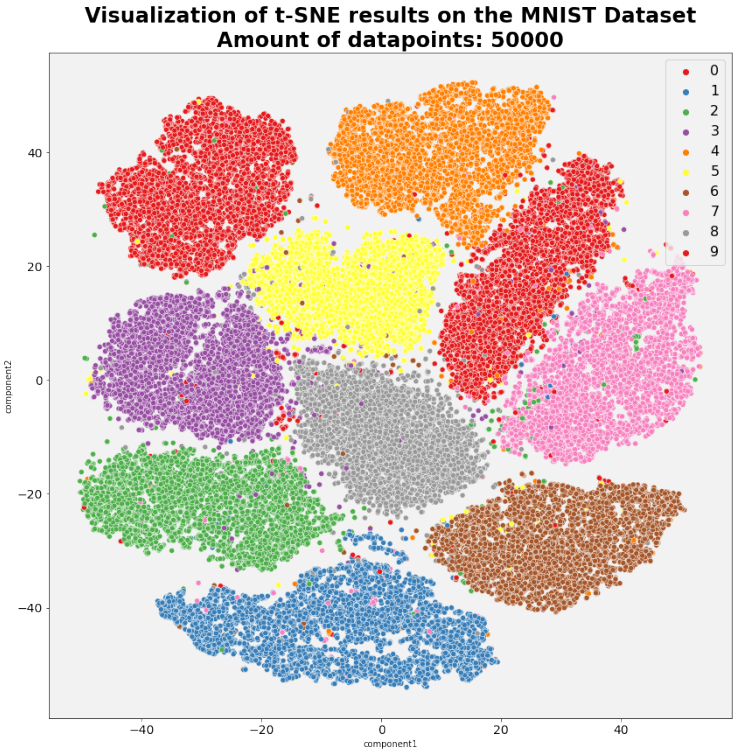

Figure 6: The unsupervised clustering of t-SNE upon a TSVD dataset. The dataset is well clustered into separate groups. The axes indicate the two features created by the t-SNE algorithm

After using the t-SNE features to train a new DTR, the training accuracy with t-SNE features (2 components) is $\sim 97.91\%$, and the validation accuracy is $\sim 97.14\%$. This is an improvement of 9.45% in validation accuracy over the baseline.

The random forest model gave a similar resulting validation accuracy of 97.13%. Using cross-validation, 500 estimators was the optimal hyper-parameter.

3.3 Regressions

The basic logistic regression with no data preprocessing gave an accuracy of $\sim 87.86\%$, which it was able to do in 28.2s. The logistic regression on the dimensionality reduced data gave an accuracy of $\sim 89.68\%$ in a computation time of 20.7s. The higher accuracy for the simpler model is likely due to the robustness of the model, which does not see the same overfitting that the basic logistic regression model has.

It is interesting to note that while the other overarching machine learning families have shown methods within the same category have positively correlating computation time and accuracy, the regression models show a negatively correlating trend. Albeit, there are only two regression methods, and both of them take relatively little time to complete.

4 Discussion

Table 2 summarizes the results of testing different machine learning models on the MNIST dataset. It is clear that although there are some models that result in a higher accuracy, when comparing overarching machine learning families, a better accuracy performance does not necessarily correlate to a worse computation time. For example, the basic NN and CNN were both able to achieve accuracies higher than any other method without the large computation time seen by the t-SNE and random forest models. However, within a machine learning family, there does seem to be a correlation between computation time and accuracy; a more complex DTR will result in a more accurate model.

Table 2: Performance metrics for different machine learning implementations.

Method	**Accuracy**	**Computation Time**
Neural Net	0.9741	21.5s
CNN	0.9870	59.5s
Decision Tree	0.8769	15s
TSVD Tree	0.8523	4.78s
TSVD + t-SNE Tree	0.9714	5min 4s
Random Forest	0.9697	2min 43s
Logistic Regression	0.8786	28.2s
Scaled LR	0.8968	20.7s

For all the different models compared, computation time[1] was calculated based on the fitting function.

There are two main observations with the comparison of machine learning methods. The convolutional layers of the CNN grant it an increased accuracy even though the other models are approaching 100% accuracy. This remaining gap is an important problem to overcome if machine learning classification is to be used in higher risk situations like the medical field. Training using the MNIST dataset gives an insight to what models can be relied upon and what models can reasonably be used in commercial implementations.

The second main observation is the significant gap between the DTR models. The models that incorporate a stochastic selection of data show a better accuracy performance.

[1] Computation time was measured relatively by the time it takes to run the code cell. It is not an exact CPU time.

5 Conclusion

Neural networks, decision trees, and regression models are all machine learning algorithms that can be applied to the MNIST dataset with varying success. Neural networks out perform the other two in both accuracy and computation time, although decision trees are not far behind in the former aspect. Regression models show inferiorities in accuracy, but provide a solid frame of reference between statistical methods and emerging machine learning techniques.

Regardless of the model, the importance of data preprocessing should be emphasized. Preprocessing can improve the performance of a machine learning model by making the data more amenable to the learning algorithm. Normalizing or standardizing the data can help ensure that the model is not biased towards any particular range of values. Preprocessing can also help reduce the complexity of the data, making it easier to work with and understand. This can be especially useful when dealing with large and complex datasets such as the MNIST dataset. Lastly, preprocessing can make the training process more efficient by reducing the amount of data that needs to be processed, and by making it easier to extract meaningful features from the data. In summary, preprocessing is an important step in the machine learning process because it can help improve the performance, reduce the complexity, and increase the efficiency of a model.

It is worth noting that the MNIST dataset is a relatively simple dataset and is often used as a "hello world" example for machine learning. As such, it is relatively easy to achieve high accuracy on this dataset, even with a basic neural network. In more complex tasks, such as image classification with larger and more diverse datasets, it may be necessary to use more advanced neural network architectures or techniques to achieve good results. This is noted not to take away from the comparisons presented in this work, but to acknowledge that machine learning infrastructure is often specialized and tuned to a specific task.

References

[1] DeepAI. MNIST Dataset, September 2019. URL `https://deepai.org/dataset/mnist`.

[2] Bencharks.AI. MNIST: Classify handwritten digits, 2022. URL `https://benchmarks.ai/mnist`.