Report: MNIST Classifiers

Comparing simple CNNs and MLPs

Milo Gerhard

Contents

Introduction

This report provides a brief summary comparing two approaches to the classification of handwritten letters from the MNIST letters database. The genesis of this experiment comes from the standardized use of convolutional neural networks in image processing versus the widespread use of simple dense networks when introducing new users to neural networks. One of the most common examples used to explore deep learning is the MNIST numbers database, almost always using a simple dense network. The contents of this report take that a small step further, using a more complex output space with 26 possible classifications. This report seeks to answer the question of whether, at the 27 pixel scale, a convolutional network solves the classification problem more or less effectively than the standard teaching tool.

Methodology

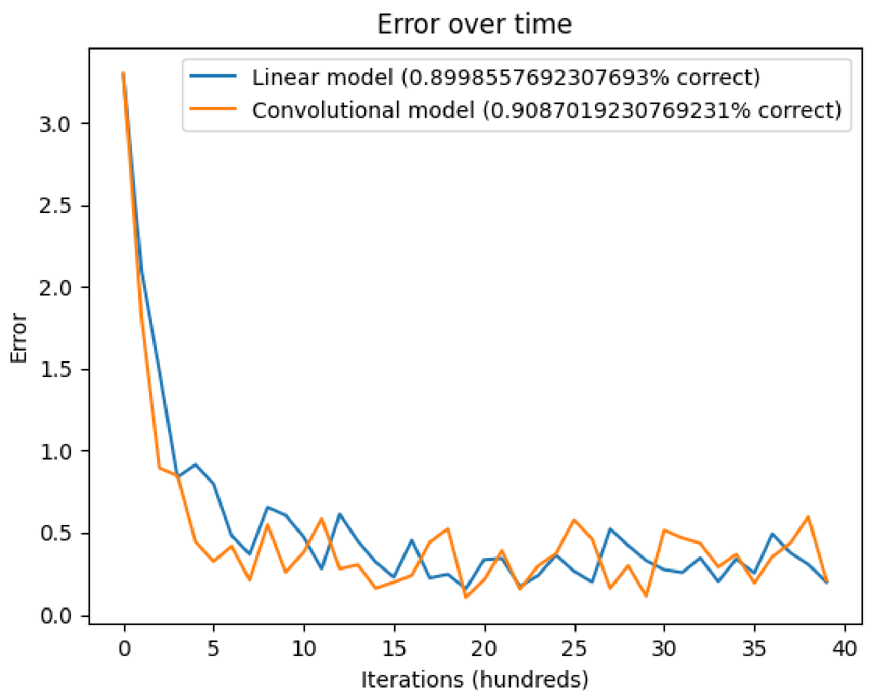
The models were created using the PyTorch default library with default data extraction and sampling methods. The networks were tested with a variety of different architectures using a variety of different learning rates and epoch numbers. See Appendix A for source code. Note that only the most recent tested architectures and values are present in the code.

Results

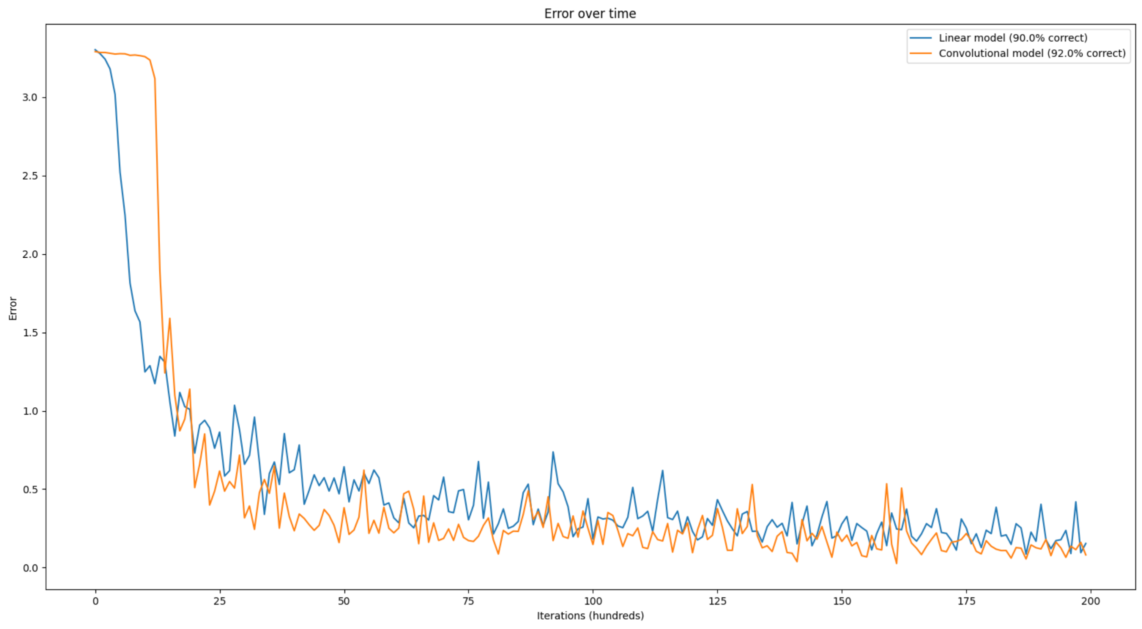
The first pair of networks were the dense network (this one did not change) and the convolutional network with two convolutional layers changing the input dimension from 3 to 10 and one 512 node hidden layer.

The following graphs show the loss graphed against the number of iterations, along with the percentage correct at evaluation.

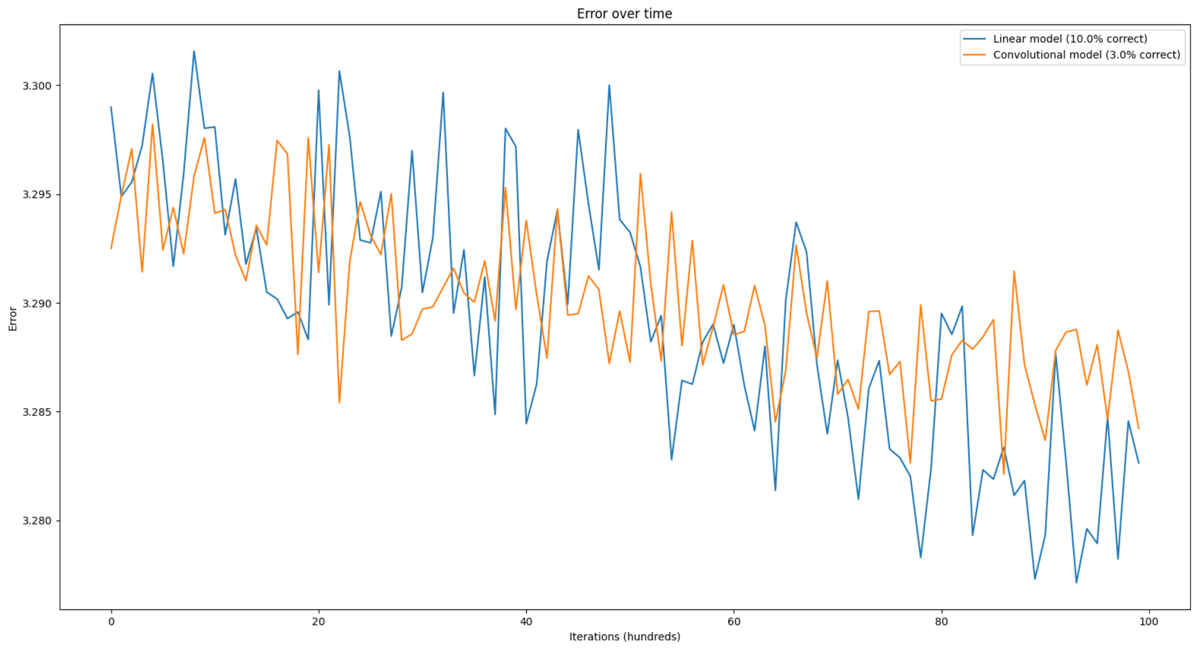
lr=0.3 with 2 epochs (percent correct 1/100 the actual number in this graph only):



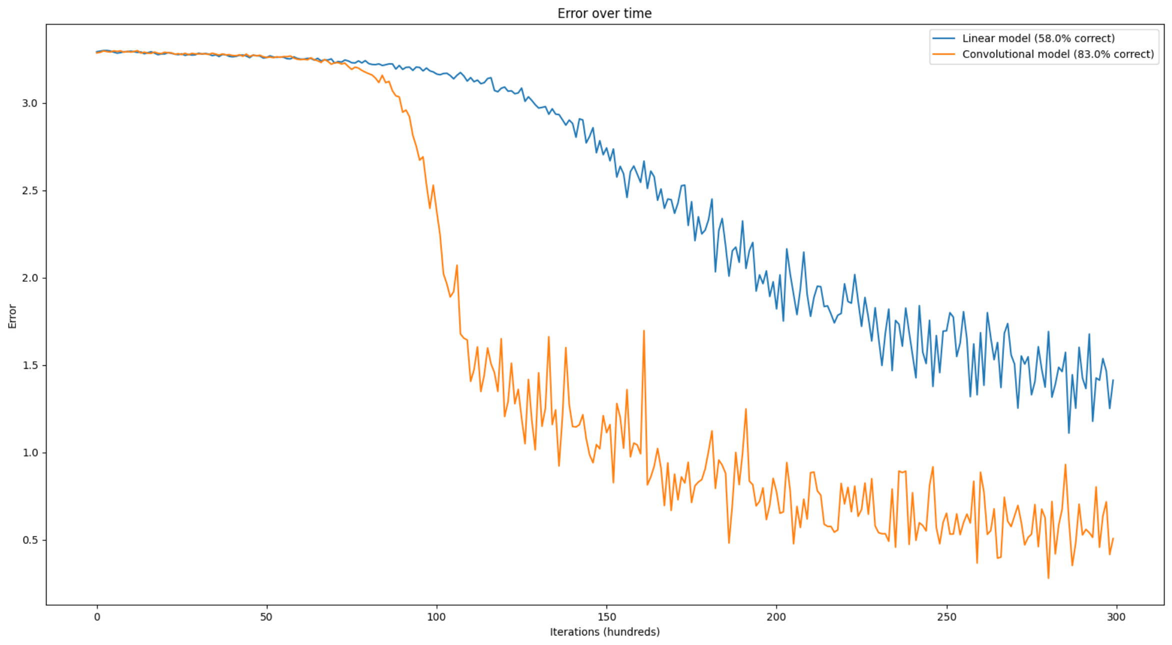
lr=e-2 with 10 epochs:



lr=e-4 with 5 epochs:

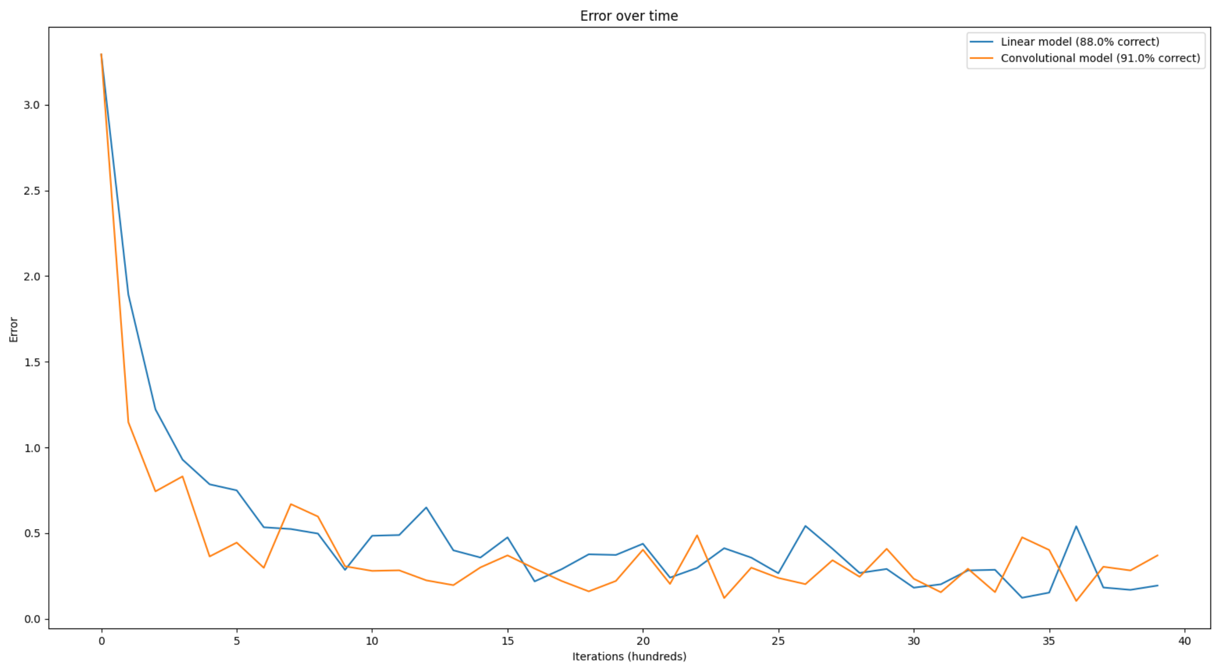


lr=e-3 with 15 epochs:

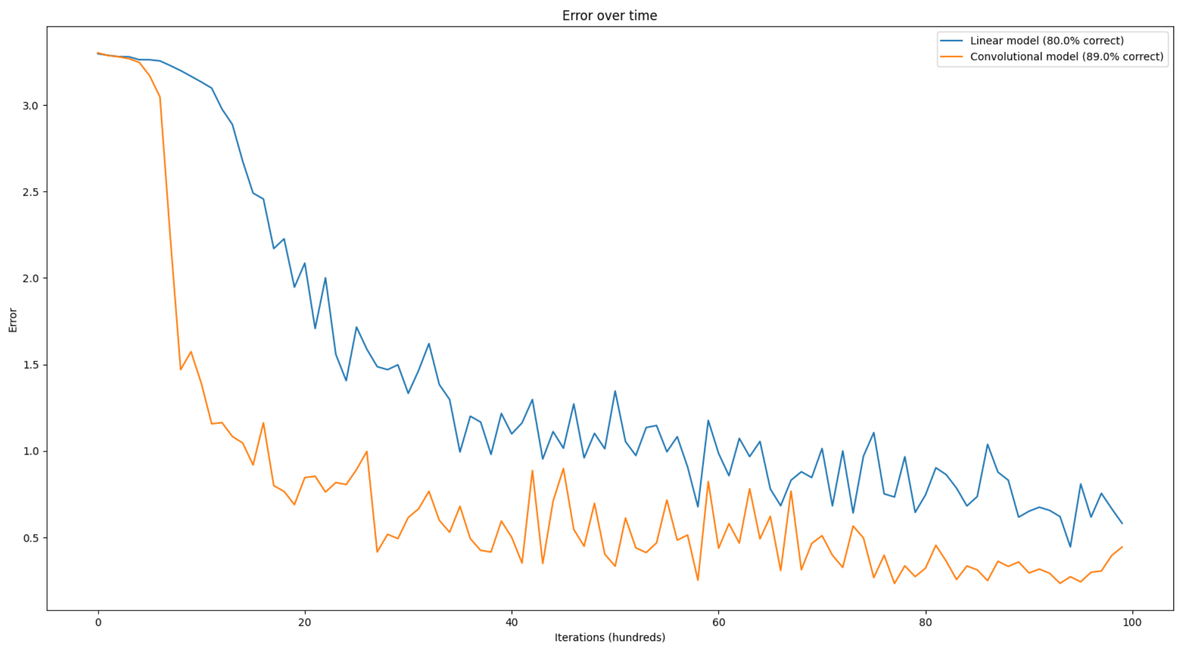


This point marks the first alteration. The convolutional network was altered to use (1,5,5), (5,16,3) as parameters for the convolutional layers and two 256 node dense layers.

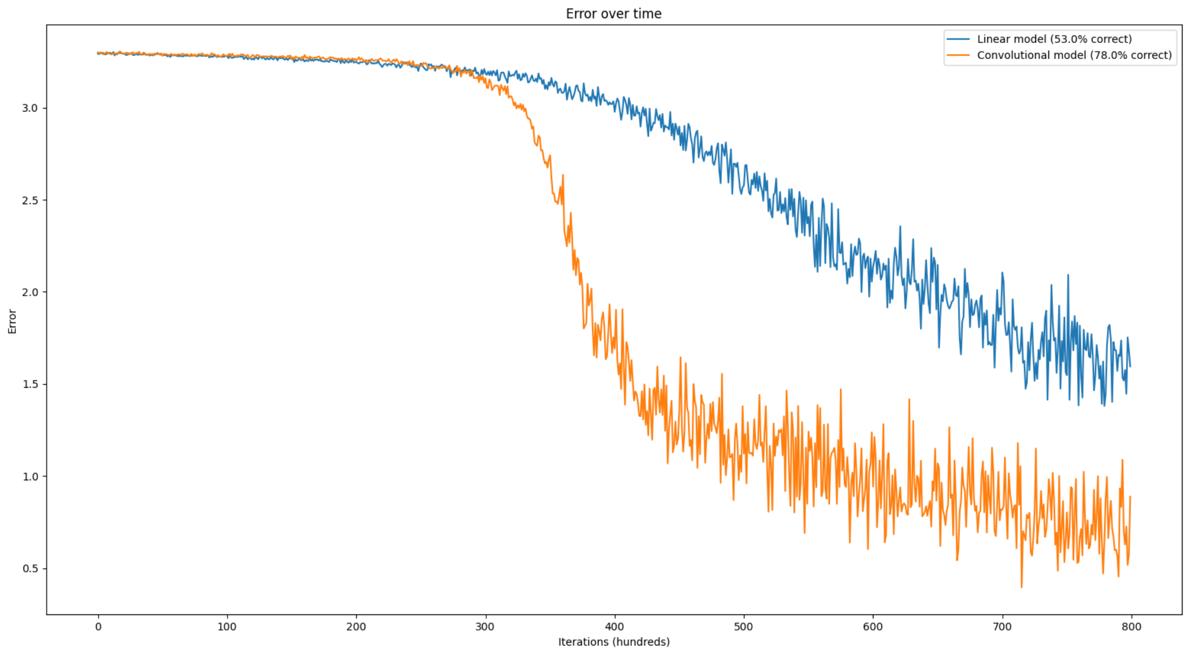
2 epochs, lr=0.3:



5 epochs e-2:

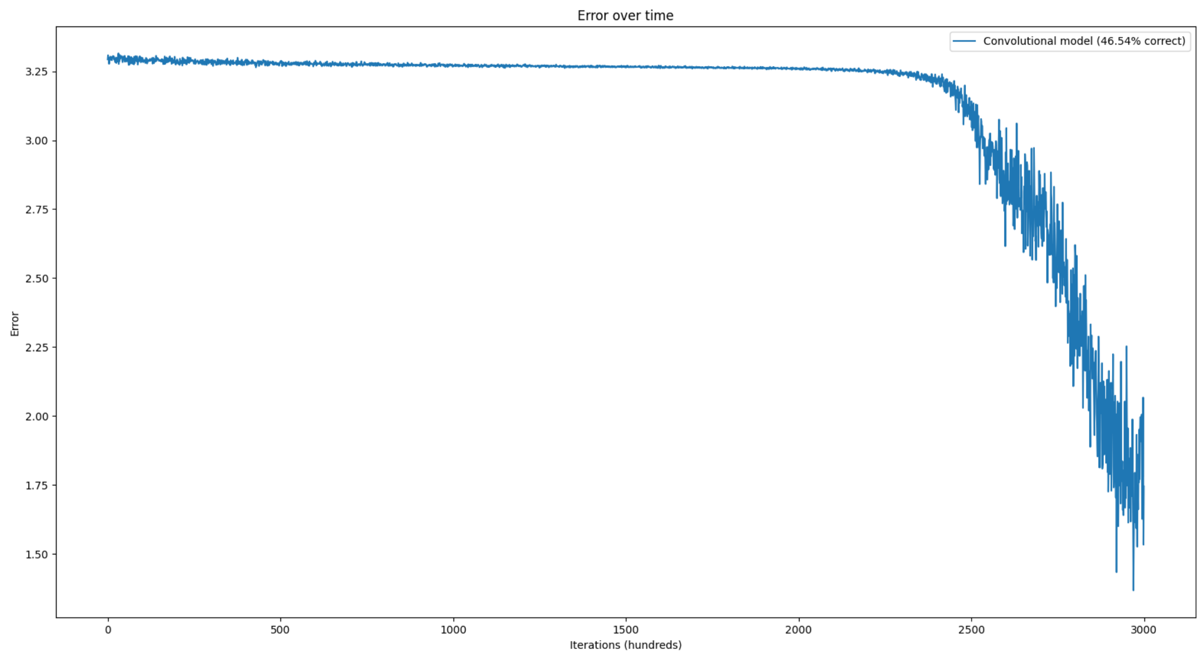


40 epochs 3e-4:

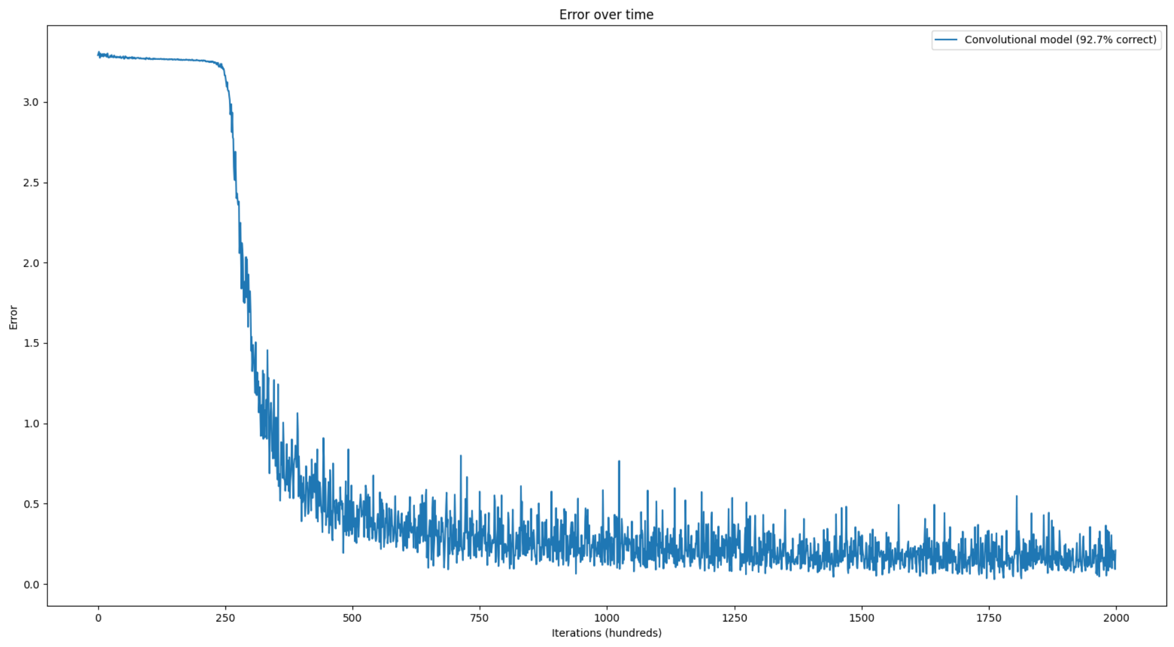


This point marks the second alteration. The convolutional network was altered to include a 16 to 27 dimension convolutional layer (making dimension of the output array 1x27) and to include two 512 node dense layers.

150 epochs, 3e-4:



100 epochs, 3e-3:



Analysis

The convolutional network performed better on average than the basic model and tended to converge faster. However, neither reached a better accuracy than about 93%. Looking at the output data shows that four to five letters tended to have a significantly lower success rate each time. For both models, there was usually a very fast initial learning process and then significant difficulties learning the remainder of the data set. Nothing appears to mitigate this issue. Perhaps with a very large number of epochs and relatively low learning rate these networks would eventually converge to a higher success rate. However, the hairiness of the equilibria implies learning is far less productive after the initial learning curve. What was unusual was the fact that extremely low epoch count and high learning rate models were nearly as effective as extensively trained ones, and better in some cases. This further supports the consideration that the data set is generally easy to learn but that several of the letters are much more difficult. Mathematically, this implies that the solution space has local minima that are easy for the stochastic process to locate but that aren't the best solutions. Classically, this is solved with a longer learning time and lower learning rate or further refinement of the possible solution space. This could imply searching for different dense architectures, activations, and loss calculation methods.

In conclusion, the convolutional network performed better than the basic model, evaluating more correctly and converging faster. However, the increase in performance was barely substantial. These findings do not support convolutional neural networks as a better teaching tool for analysis of simple written character datasets.

Appendix A

https://github.com/DereferenceMyPointer/MNIST-Classifier