

## A Appendix / supplemental material

Two grid searches for the publicly available MNIST dataset were performed to corroborate the learning rate and momentum weight derived in the main paper (LeCun et al., accessed May 21, 2024). The MNIST dataset contains gray-scale images of handwritten digits and is one of the prominent datasets used to evaluate machine learning methods. It is split into a training and a test set, where the latter serves as a standard of comparison. Figure 2 shows an example of the MNIST data.

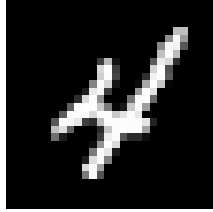


Figure 2: A slightly enlarged example from the MNIST dataset showing a handwritten digit (4).

### A.1 Experiments

The grid searches were performed on the full-size MNIST dataset and a smaller version of MNIST containing only 50% of the training data. In the latter case, a stratified sampling method named *StratifiedShuffleSplit* was used to create a stratified random subset of the training samples (Scikit-learn developers, BSD License; Pedregosa et al., 2011; Buitinck et al., 2013). This ensured that the class distribution in the training subset was the same as in the original full-size training set. The degradation in dataset size allowed observing how each optimizer performed under varying amounts of training data, assuming that providing less training data posed a harder problem.

A deep learning model was trained based on a convolutional neural network (CNN). The model consisted of two convolutional layers, each followed by a ReLU activation function and a max pooling operation. The first convolutional layer had a single-channel input (grayscale image) and applied 16 filters, followed by a second convolutional layer that expanded the channel size to 32. Both convolutional layers used a 3x3 kernel size, a stride of one, and a padding of one. After each convolution, a ReLU activation function introduced non-linearity, and a max pooling operation with a 2x2 kernel and stride reduced the spatial dimensions by half. A dropout layer with a rate of 0.25 was applied after flattening the output to prevent overfitting. The network concluded with two fully connected layers with a final output of 10 classes, where the maximum output value determined the class of an input image. The number of parameters was around two hundred thousand for an MNIST input image of size 28x28. A weight initialization was performed using the Kaiming uniform method. No data augmentation techniques were applied; however, the input was normalized to the range [-1,1]. The training used a batch size of 64 and was conducted over 30 epochs, employing cross entropy as the loss function. The sizes of the training, validation, and test datasets were 54,000, 6,000, and 10,000, respectively. Finally, the model’s performance was assessed through 10-fold cross-validation.

### A.2 Results

The results of both grid searches are shown in Figure 3 for the full-size training set and in Figure 3 for the smaller training set with 50% of the size. The following values were used as momentum weights for each grid search: 0, 0.2, 0.4, 0.6, 0.8, 0.825, 0.85, 0.874, 0.9, and 0.925. On the other hand, the following values were used as learning rates: 0.0001, 0.001, 0.01, 0.016, 0.1, 0.2. These values included the momentum weight derived in the paper ( $\alpha \approx 0.874$ ) and the derived learning rate ( $\eta \approx 0.016$ ). Other values were chosen based on their use in the literature or to increase the resolution around the derived theoretical values. All possible combinations of values span a 6x10 grid. The color of each square in the grids of Figure 3 and Figure 4 represent the performance of the corresponding pair of momentum weight and learning rate, with lighter colors representing higher performance. Green rectangles indicate the top ten performing pairs, whereas blue rectangles show the best-performing pair. Note that more than one pair can share the best performance, as in Figure 3.

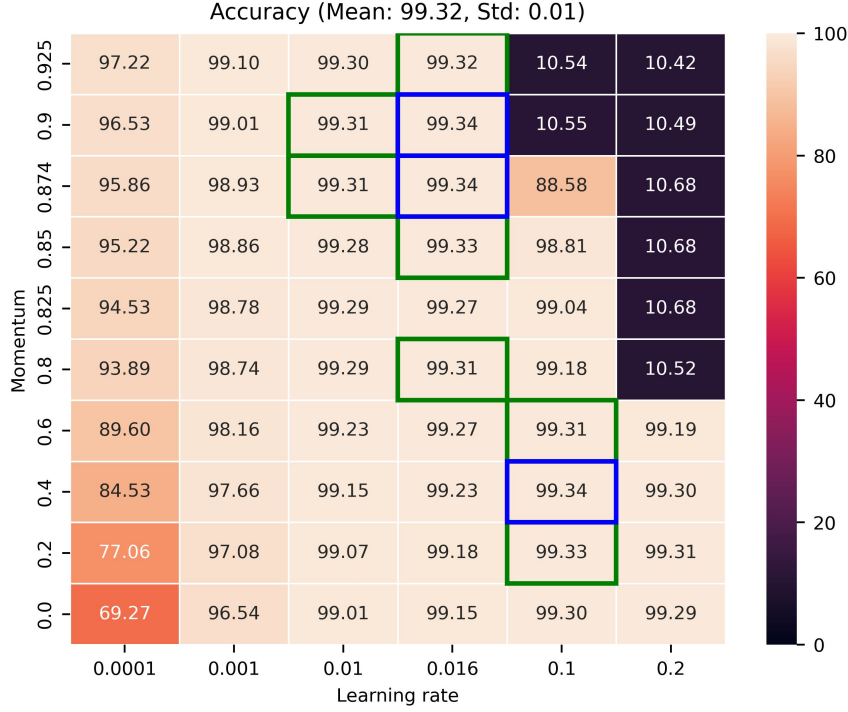


Figure 3: Grid search results for MNIST

Figure 3 shows that no pair of momentum weight and learning rate provides better performance on the full-size MNIST set than the pair derived in the paper, (0.016, 0.874), although this pair has to share its first place with other pairs. The classification accuracies for the reduced training set size are slightly lower in the table of Figure 4, as one would expect for a problem with less training data. Nevertheless, the theoretical values derived in the paper for momentum weight and learning rate show again the best performance.

### A.3 Computational environment and runtime

The software was developed using Python 3.10, and the Convolutional Neural Network (CNN) model was implemented in Pytorch 2.2.2. For each combination of learning rate and momentum weight (60 combinations in total), the training time was approximately three hours for 100% of the training set size and about 1.5 hours for 50% of the training set. Consequently, the cumulative GPU time for all experiments was approximately  $(3 + 1.5) \times 60$  hours, which is 270 hours. The average memory usage was roughly 1 GB for each combination. For more information about the software requirements and workflow, see the Readme file uploaded as supplemental material together with the code.

### A.4 Computing cluster

Figure 5 shows an overview of the GPU computing cluster that was available for the experiments, including the type of GPUs among which the processing was distributed.

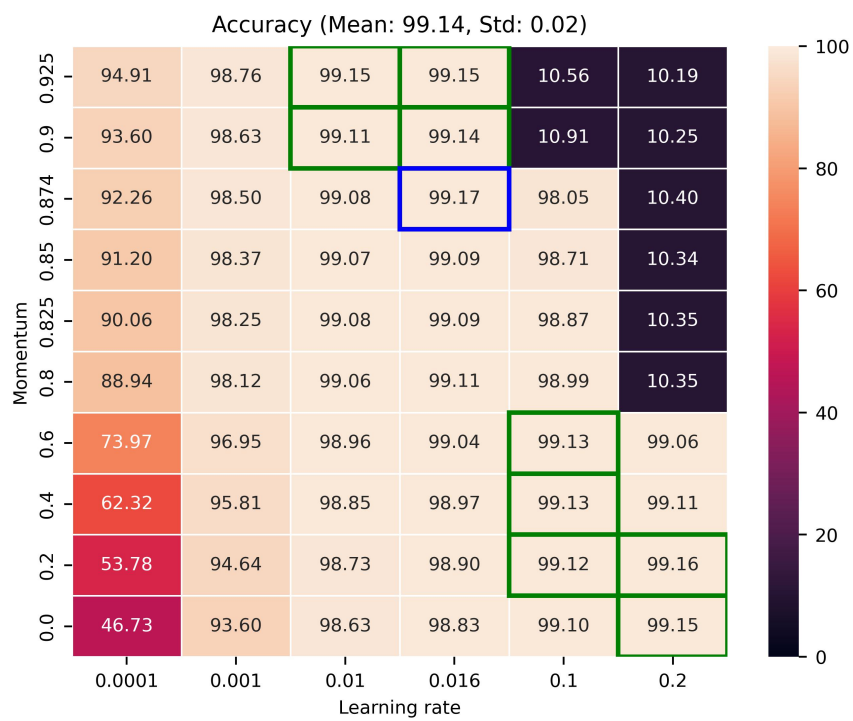


Figure 4: Grid search results for MNIST using only 50% of the training data

GPU nodes	Processor cores per node	Memory	Network
36	32 x 2.8 GHz (AMD Epyc 7543p) hyperthreading enabled 256 MB level 3 cache <b>4 x NVIDIA A100 GPUs</b> (80 GB VRAM, 6912 cores, 432 Tensor cores) NVLINK	256 GB	200 Gb/s HDR Infiniband (1:1)
56	36 x 2.3 GHz (Intel Gold 6140) hyperthreading enabled 25 MB secondary cache <b>4 x NVIDIA V100-SXM2 GPUs</b> (32 GB VRAM, 5120 cores, 640 Tensor cores) NVLINK	384 GB	200 Gb/s HDR Infiniband (1:1)
8	28 x 2.4 GHz (Intel E5-2680v4) hyperthreading enabled 35 MB secondary cache <b>4 x NVIDIA V100 GPUs</b> (16 GB VRAM, 5120 cores, 640 Tensor cores)	128 GB	56 Gb/s FDR Infiniband (1.11:1)
48	28 x 2.4 GHz (Intel E5-2680v4) hyperthreading enabled 35 MB secondary cache <b>4 x NVIDIA P100 GPUs</b> (16 GB VRAM, 3584 cores)	128 GB	56 Gb/s FDR Infiniband (1.11:1)
72	28 x 2.4 GHz (Intel E5-2680v4) hyperthreading enabled 35 MB secondary cache <b>2 x NVIDIA K80 GPUs</b> with 2 x GK210 GPUs each (24 GB VRAM, 4992 cores)	256 GB	56 Gb/s FDR Infiniband (1.11:1)

Figure 5: GPU computing cluster