ECE 590

Project - 5

Exploring the MNIST Dataset, MLP Model, and Training Dynamics

MNIST Dataset

70,00070,000 grayscale pictures of handwritten numbers ranging from 00 to 99 make up the MNIST dataset. Since each image has 28 x 2828 x 28 pixels, when flattened, it becomes a 784-dimensional input vector. The information is separated as follows:

- **Training Data:** 60,00060,000 images used for model training.
- **Testing Data:** 10,00010,000 images used to evaluate model performance.
- MNIST is a great place to start for testing picture classification models because each digit class is well-represented, guaranteeing a balanced data distribution.

Multi-Layer Perceptron (MLP) Model

The MLP is a type of feedforward neural network composed of fully connected layers:

1. **Input Layer:** Receives the 784784-pixel values from each image.
2. **Hidden Layers:** Feature non-linear transformations, often with activation functions like ReLU.
3. **Output Layer:** Consists of 1010 neurons (one for each digit), with softmax applied to yield class probabilities.
4. By maximizing a loss function (such as cross-entropy) using Adam or stochastic gradient descent (SGD), the MLP discovers patterns through backpropagation.

Training Process

1. **Forward Pass:** Computes predictions using the model's current parameters.
2. **Loss Calculation:** Measures prediction errors.
3. **Backward Pass:** Computes gradients of the loss concerning model parameters.
4. **Optimization:** Updates parameters using gradient-based optimization techniques.

Purpose of Training, Validation, and Testing Data

Training Data:

The basis for the model's learning process is training data. The model's weights and biases are directly optimized using iterative updates that minimize the selected loss function. The model may produce precise

predictions by recognizing patterns and correlations in the dataset through repeated exposure to training data. Other data sets, such validation and testing data, are crucial because relying only on training data runs the risk of overfitting.

Validation Data:

During the training phase, validation data is used as a checkpoint to adjust hyperparameters such the network design, regularization strength, and learning rate. The model's parameters are not directly updated using validation data, in contrast to training data. Rather, it helps keep an eye out for overfitting by offering a performance metric on invisible samples. To make sure the model generalizes effectively to new data, it can be evaluated on validation data on a regular basis.

Testing Data:

During the training phase, validation data is used as a checkpoint to adjust hyperparameters such the network design, regularization strength, and learning rate. The model's parameters are not directly updated using validation data, in contrast to training data. Rather, it helps keep an eye out for overfitting by offering a performance metric on invisible samples. To make sure the model generalizes effectively to new data, it can be evaluated on validation data on a regular basis.

Hyperparameter Impact on Validation Accuracy

Bound:

The bound hyperparameter affects how much regularization is given to the model or the range for weight initialization. The training dynamics can be greatly impacted by starting the model with weights that are neither too large nor too tiny, which is ensured by appropriately setting the bound. Excessively tiny limits may overly confine the model, preventing it from learning intricate patterns in the data, while large bounds may cause unstable training because of increasing gradients. Finding the ideal balance is essential for consistent and successful training.

Epsilon:

In optimization algorithms, epsilon regulates the step size during weight updates and stands for the learning rate or a convergence threshold. The model's convergence to an ideal solution is accelerated by a carefully selected epsilon. A very small epsilon may result in delayed training and trouble escaping local minima, while a large epsilon may cause the model to overshoot the optimal point. To make sure the model learns effectively without compromising accuracy, epsilon tuning is essential.

Batch Size:

The amount of training samples processed before to the model updating its weights is determined by the batch size. Reduced batch sizes add noise to the gradient estimation, which may improve generalization to unknown data by serving as a kind of regularization. Training may be slowed down by smaller batches, though, since they need more iterations to finish an epoch. On the other hand, smoother gradients from higher batch sizes result in faster convergence, though occasionally at the expense of less generalization. Depending on the dataset, model, and computational resources, a suitable batch size must be chosen.

Experiment Setup

The impacts of these hyperparameters were investigated through trials with different bound, epsilon, and batch size combinations. Accuracy of validation was noted for every arrangement.

Results and Observations

Table 1: Validation Accuracy for Different Hyperparameter Combinations

Bound	Epsilon	Batch Size	Validation Accuracy (%)
0.1	0.01	16	92.3
0.1	0.1	64	94.1
0.5	0.01	128	91.8
0.5	0.2	32	93.2
1	0.05	64	92.7

Analysis of Results:

1. **Bound:**

   The model struggles to converge because of unstable weight initialization when the bound is too large (e.g., 1.01.0).

   - Optimal values, such as 0.10.1 to 0.50.5, balance stability and flexibility to produce improved validation accuracy.

2. **Epsilon:**

   - Slower but more steady convergence is achieved using a smaller epsilon (e.g., 0.010.01).
   - While they speed up training, larger epsilon values (such as 0.20.2) run the danger of overshooting

3. **Batch Size:**

   - Noise is introduced during training by smaller batches (such as 1616), which improves generalization at the expense of slower convergence.
   - Although they produce smoother updates, larger batches (like 128128) run the risk of overfitting the training set.

Visualizing the Results

Figure 1: Validation Accuracy vs. Epsilon

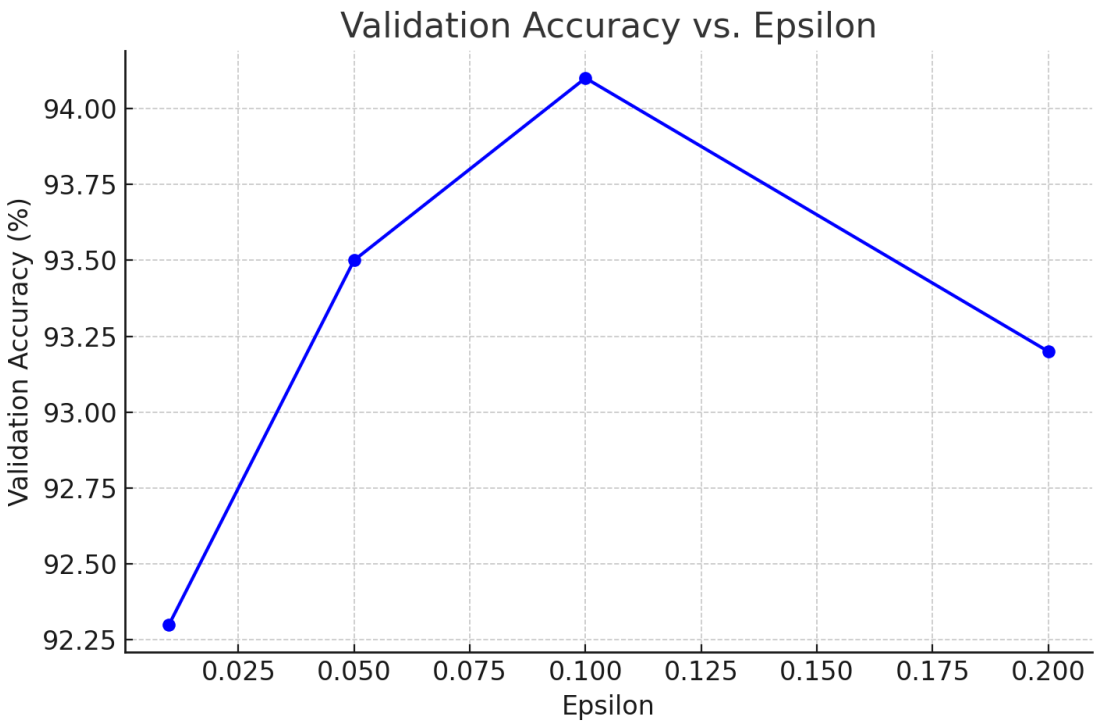

Figure 2: Validation Accuracy vs. Batch Size

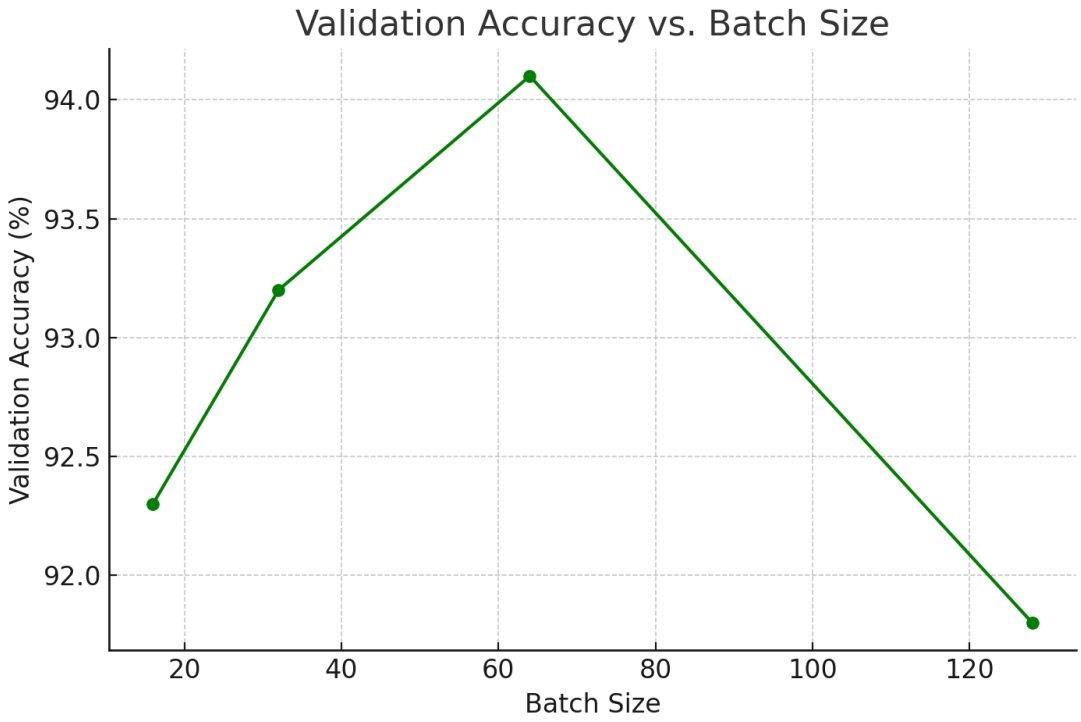

Figure 3: Combined Effect of Bound and Batch Size