A brief description of the MNIST dataset, the MLP model, and the training process.

**The MNIST dataset** is a dataset consisting of pictures of handwritten digits and their corresponding numerical values to be used for testing machine learning algorithms. It is a combination of two former lists from the National Institute of Standards and Technology (NIST).

**The MLP model** used in project 5

Inputs: x

Outputs: z

x = flatten(images)

g = f1(x)

h = ReLU(g)

z = f2(h)

f1 and f2 are linear functions with weight matrices W and bias matrices b.

g = Wf1\*x + bf1

z = Wf2\*h + bf2

**The training process**

1. Initialize theta (weights and biases)
2. Shuffle training images
3. For each epoch
   1. Select batch images and labels based on batch size
   2. Obtain x, g, h, and z with forward propagation
   3. Calculate nabla J with backpropagation
   4. Update theta with stochastic gradient descent (SGD)
   5. Obtain new z using new theta with forward propagation
   6. Use z to predict labels for validation images
   7. Compare predictions with actual labels and print accuracy

Discuss the different purposes of the training, validation, and testing data.

**Training data** ⇒ data with known labels that is used to modify the algorithm’s weights during training through backpropagation

**Validation data** ⇒ data that is not part of the training itself but is used to test the algorithm’s accuracy throughout training

**Testing data** ⇒ data used to test the algorithm’s accuracy after it has already been trained

How the three hyperparameters bound, epsilon, and batch\_size affect the accuracy through validation? You should experiment with different combinations of the three hyperparameters and use tables and figures to explain your discovery.

Bound

The bound seemed to most affect the initial accuracy obtained after the first epoch. Generally, the smaller the bound, the better the accuracy.

Bound = 100, Epsilon = 1e-5, Batch size = 4

| **Epoch** | **Accuracy** |
| --- | --- |
| 1 | 0.360 |

Bound = 1, Epsilon = 1e-5, Batch size = 4

| **Epoch** | **Accuracy** |
| --- | --- |
| 1 | 0.406 |

Bound = 0.001, Epsilon = 1e-5, Batch size = 4

| **Epoch** | **Accuracy** |
| --- | --- |
| 1 | 0.876 |

However, very small bound values nearing zero had an adverse effect.

Bound = 1e-9, Epsilon =1e-5, Batch size = 4

| **Epoch** | **Accuracy** |
| --- | --- |
| 1 | 0.265 |

Also see epsilon section for how the bound affected the efficacy of epsilon values.

Epsilon

Epsilon seemed to affect the rate at which the data was trained (as measured by increase in accuracy over time). The greater the epsilon, the faster the data was trained, and the more quickly the accuracy values increased.

Bound = 1, Epsilon = 1e-9, Batch size = 20

| **Epoch** | **Accuracy** |
| --- | --- |
| 1 | 0.075 |
| 2 | 0.075 |
| 3 | 0.074 |
| 4 | 0.076 |
| 5 | 0.076 |
| 6 | 0.077 |
| 7 | 0.077 |
| 8 | 0.077 |
| 9 | 0.078 |
| 10 | 0.078 |

Bound = 1, Epsilon = 1e-6, Batch size = 20

| **Epoch** | **Accuracy** |
| --- | --- |
| 1 | 0.197 |
| 2 | 0.293 |
| 3 | 0.343 |
| 4 | 0.393 |
| 5 | 0.422 |
| 6 | 0.440 |
| 7 | 0.462 |
| 8 | 0.474 |
| 9 | 0.484 |
| 10 | 0.489 |

However, excessively large epsilon values led to overtraining of the data, and successive epochs decreased or wavered in accuracy rather than increasing.

Bound = 1, Epsilon = 1e-5, Batch size = 20

| **Epoch** | **Accuracy** |
| --- | --- |
| 1 | 0.488 |
| 2 | 0.525 |
| 3 | 0.485 |
| 4 | 0.466 |
| 5 | 0.410 |
| 6 | 0.445 |
| 7 | 0.388 |
| 8 | 0.434 |
| 9 | 0.451 |
| 10 | 0.401 |

One way to remedy this was to decrease the bound. Smaller bounds tolerated greater epsilon values.

Bound = 0.01, Epsilon = 1e-5, Batch size = 20

| **Epoch** | **Accuracy** |
| --- | --- |
| 1 | 0.152 |
| 2 | 0.303 |
| 3 | 0.476 |
| 4 | 0.580 |
| 5 | 0.640 |
| 6 | 0.683 |
| 7 | 0.716 |
| 8 | 0.744 |
| 9 | 0.769 |
| 10 | 0.779 |

Batch Size

Batch size seemed to affect the accuracy values overall, and smaller batch sizes led to greater accuracy.

Bound = 0.01, Epsilon = 1e-5, Batch size = 2

| **Epoch** | **Accuracy** |
| --- | --- |
| 1 | 0.897 |
| 2 | 0.920 |
| 3 | 0.933 |
| 4 | 0.938 |
| 5 | 0.947 |
| 6 | 0.957 |
| 7 | 0.953 |
| 8 | 0.956 |
| 9 | 0.955 |
| 10 | 0.960 |

Bound = 0.01, Epsilon = 1e-5, Batch size = 16

| **Epoch** | **Accuracy** |
| --- | --- |
| 1 | 0.799 |
| 2 | 0.853 |
| 3 | 0.879 |
| 4 | 0.882 |
| 5 | 0.890 |
| 6 | 0.890 |
| 7 | 0.893 |
| 8 | 0.894 |
| 9 | 0.898 |
| 10 | 0.903 |

Bound = 0.01, Epsilon = 1e-5, Batch size = 64

| **Epoch** | **Accuracy** |
| --- | --- |
| 1 | 0.488 |
| 2 | 0.693 |
| 3 | 0.773 |
| 4 | 0.799 |
| 5 | 0.821 |
| 6 | 0.836 |
| 7 | 0.845 |
| 8 | 0.851 |
| 9 | 0.857 |
| 10 | 0.861 |

Here’s a table of all the tests I conducted in order to pass the grading script:

\* → passed all four test cases of grade\_p5.py

| **Bound** | **Epsilon** | **Batch Size** | **Accuracy after 10 Epochs** |
| --- | --- | --- | --- |
| 1 | 1e-5 | 4 | 0.340 |
| 1 | 1e-5 | 20 | 0.401 |
| 1 | 1 | 20 | 0.117 |
| 1 | 5e-5 | 20 | 0.313 |
| 1 | 5e-5 | 4 | 0.258 |
| 1 | 1e-6 | 20 | 0.489 |
| 1 | 1e-6 | 4 | 0.468 |
| 1 | 1e-6 | 50 | 0.389 |
| 50 | 1e-6 | 20 | 0.488 |
| 0.1 | 1e-6 | 20 | 0.514 |
| 100 | 1e-6 | 20 | 0.488 |
| 0.1 | 1e-7 | 20 | 0.197 |
| 1 | 3e-7 | 20 | 0.512 |
| 0.01 | 3e-7 | 20 | 0.861 |
| 1 | 3e-7 | 2 | 0.351 |
| 0.01 | 3e-7 | 4 | 0.905 |
| 0.01 | 5e-7 | 4 | 0.927 |
| 0.001 | 5e-7 | 4 | 0.925 |
| 0.01 | 1e-5 | 4 | 0.952 |
| 0.01 | 1e-5 | 2 | 0.960\* |
| 0.001 | 5e-5 | 2 | 0.965\* |