CMPT 423-820 MINI PROJECT 2 REPORT

Alireza Falamarzi (ylw576), Baptiste Rouquette (dly490), Princess Tayab (prt898), Talha Mansoor (kgy284)

1 The Loss Objective

We have used the sum of cross-entropy loss as our loss objective. By minimizing this summed loss, we maximize the likelihood of the correct class. Given a dataset $D = \{(x_i, y_i)\}_{i=1}^{D}$, where each image x_i is labeled with a formula y_i (3 characters: digit + operator + digit), we label each image as $y_i = (y_{i1}, y_{i2}, y_{i3})$, with y_{i1} representing the first digit, y_{i2} the operator, and y_{i3} the second digit. The likelihood of the dataset D given the parameters θ is then:

$$L(\theta) = \prod_{i=1}^{D} P(y_i \mid x_i; \theta) \tag{1}$$

where $y_i = (y_{i1}, y_{i2}, y_{i3})$ represents the true labels for the first digit, operator, and second digit. Now we use the log trick to turn the product into a sum:

$$\log L(\theta) = \sum_{i=1}^{D} [\log P(y_{i1} \mid x_i; \theta) + \log P(y_i \mid x_i; \theta) + \log P(y_{i3} \mid x_i; \theta)] \tag{2}$$

And then to convert the maximization problem into a minimization problem, we take the negative log-likelihood and normalize by the dataset size D to obtain the average loss:

$$L(\theta) = -\frac{1}{D} \sum_{i=1}^{D} [\log P(y_{i1} \mid x_i; \theta) + \log P(y_{i2} \mid x_i; \theta) + \log P(y_{i3} \mid x_i; \theta)] \tag{3}$$

This is the same as minimizing the sum of three cross-entropy losses, one for each character (digit1, operator, digit2) so for each prediction, we calculate the cross entropy using softmax with the formula given below (this is done internally by nn.CrossEntropyLoss in pytorch):

$$L = -\log(\frac{e^{z_k}}{\sum_{j=1}^{C} e^{z_j}}) \tag{4}$$

Where:

- z_j denotes the logit (raw output score) for class j.
- z_k is the logit for the true class (i.e., where the correct label is).
- C represents the total number of classes. For digits, C is from 0 to 9 (i.e., $C = 10$), and for letters, $C = 4$.

Thus, the total loss is the sum of the cross-entropy losses for the three characters: $L(\theta) = L_{y1} + L_{y2} + L_{y3}$.

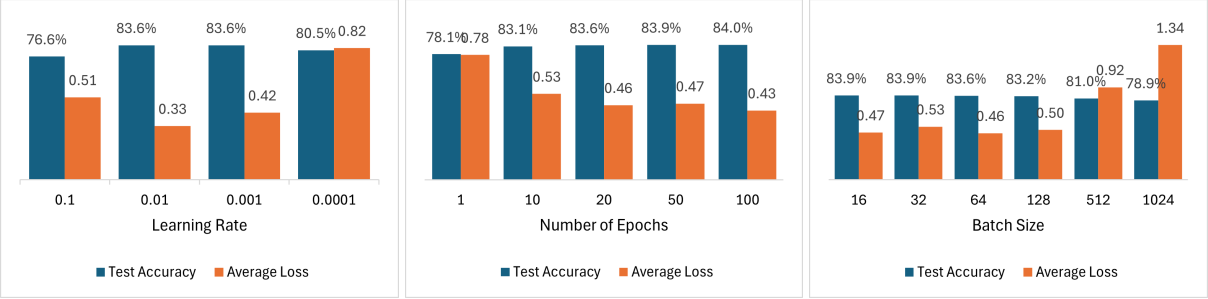
1.1 Why This Approach is Appropriate

In our base model we have used multiclass logistic regression. The EMNIST dataset contains images of simple handwritten letters and characters. These images are relatively simple and a linear model is enough for distinguishing their features after we flatten the images. We have used a separate linear classifier for each of the 3 characters to break the problem down into smaller, easier parts. Instead of trying to predict the whole formula at once, it predicts each character separately, such as the first digit, the operator, and the second digit. This makes the problem simpler and avoids the complexity of handling all possible combinations of digits and operators, which would require way more data and computation. By using cross-entropy loss for each character, the model learns to predict each part accurately, and by combining the losses, it gets better at predicting the entire formula correctly. It also allows for managing longer or more complex formulas by simply adding more output heads.

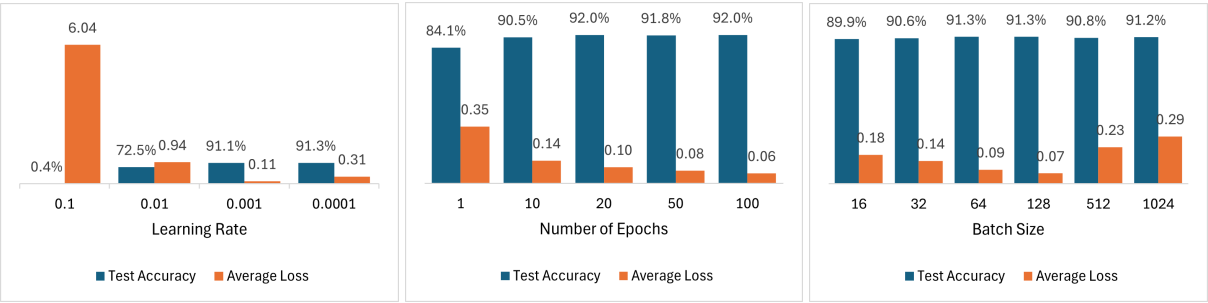
What makes it even better is the use of CNNs (Convolutional Neural Networks) for the improved model. CNNs are great at recognizing patterns in images, like the shapes of digits and operators, which is crucial for this task. This approach is also easier to debug because you can identify which part of the formula the model struggles with, e.g., if it performs poorly with operators but accurately identifies digits. Overall, it's simpler, more efficient, and more accurate than trying to solve the whole problem at once.

2 Performance Report and Results

Hyper Parameters: Learning Rate (lr), Number of Epochs (ne), Batch Size (bs), Optimizer Type (ot)
A. Baseline Model: Best Parameters are lr=0.01 or 0.001, ne=100, bs=16 or 32, ot=SGD



B. Improved Model: Best Parameters are lr=0.001, ne=20 or 100, bs= 64 or 128, ot=Adam



* for the complete details, please see `baseline_outputs.txt` and `improved_outputs.txt` for all outputs, and `summary.pdf` for a summary of findings.

2.1 Comparison with the Baseline

For the final performance comparison between the improved model and the base model using the central limit theorem, we chose a set of best performing hyper parameters for both and ran each model 10 time (We could have ran more times but that was beyond our time and computation power contraints). Then we calculated the average of each of the 10 sets accuracies. And finally we calculated the SEM value for each average that resulted in the 68 percent likely range of accuracies. By observing the gap these two ranges we can confirm that our CNN model has statistically significant improvement than our baseline model:

Model	Mean Accuracy	SEM	68% Confidence Interval ($Mean \pm SEM$)
Baseline	83.849	0.0099	[83.839, 83.859]
Improved	91.111	0.0817	[91.029, 91.193]

Table 1: Performance Comparison of Baseline and Improved Models

Best hyper parameters used:
Baseline: [Epoch: 20], [LR: 0.001], [Batch: 32], [Optimizer: SGD]
Improved: [Epoch: 20], [LR: 0.001], [Batch: 64], [Optimizer: Adam]

3 How to Run The Code and Reproduce experiments

Open the p2/formula.py and follow the instructions at the top of the file to change the desired parameters. See summary.pdf to find the desired settings and compare results. To run the code, open terminal and go to root directory and call one of these commands:

3.1 To train and test baseline model:

```
$ python p2/formula.py train-baseline
```

3.2 To train and test the improved model:

```
$ python p2/formula.py train-improved
```

3.3 To run the the performance comparisons:

The performance statistics output will be saved to a file called performance_stats.txt

```
$ python p2/formula.py performance-diff
```