Sanketh Karuturi
karutusa@oregonstate.edu

Assignment #2: CIFAR Image Classification using Fully-Connected Network

This report documents the implementation and evaluation of a fully connected neural network for binary classification on a subset of the CIFAR-10 dataset. The goal is to design a multi-layer network with at least one hidden layer, use backpropagation with binary cross-entropy loss, and train the network with stochastic mini-batch gradient descent (SGD) with momentum. The final layer utilizes the ReLU activation function, and performance is assessed based on training accuracy, loss, misclassification rates, and parameter tuning experiments.

1. Implemented Functions: Linear, ReLU, and SigmoidCrossEntropy Layers *(15 points)*

- **Linear Layer**: Implements a fully connected layer with learnable weights and biases.
- **ReLU Layer**: Implements the ReLU activation function for non-linearity
- **SigmoidCrossEntropy Layer**: Computes the sigmoid activation followed by binary cross-entropy loss.

All layers store their input and output values for backpropagation, and the backward method computes gradients for weight updates.

2. Stochastic Mini-Batch Gradient Descent with Momentum *(10 points)*

SGD with momentum was implemented for weight updates:

- **Momentum updates**: Helps stabilize training and accelerates convergence.
- **Weight decay**: Added to prevent overfitting.
- **Batch shuffling**: Ensures varied training batches across epochs.

The step() method correctly updates the weights using the gradient and applies momentum.

3. Training the Network on CIFAR-2 *(15 points)*

The network was trained on a binary-class version of CIFAR-10 with 10,000 training examples and 2,000 test examples.

Training Setup:

- **Number of Layers**: 4
- **Activation in Hidden Layers**: ReLU
- **Activation in Output Layer**: ReLU
- **Loss Function**: Binary Cross-Entropy
- **Optimizer**: SGD with Momentum
- **Normalization**: Pixel values scaled to [0,1]

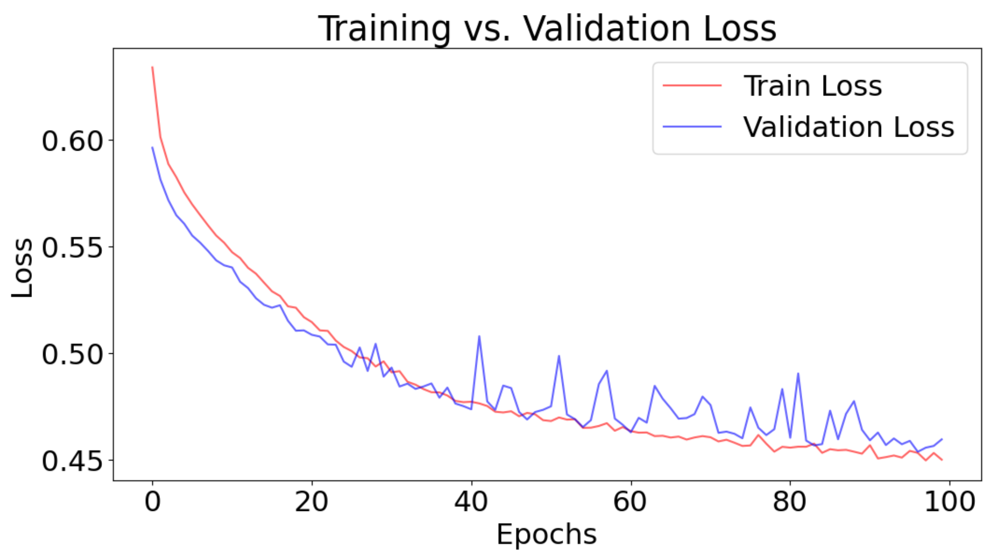
Results Achieved:

- **Final Validation Accuracy**: *79.75%*
- **Train Loss & Accuracy Trend**: Loss decreased, accuracy increased, indicating effective learning.

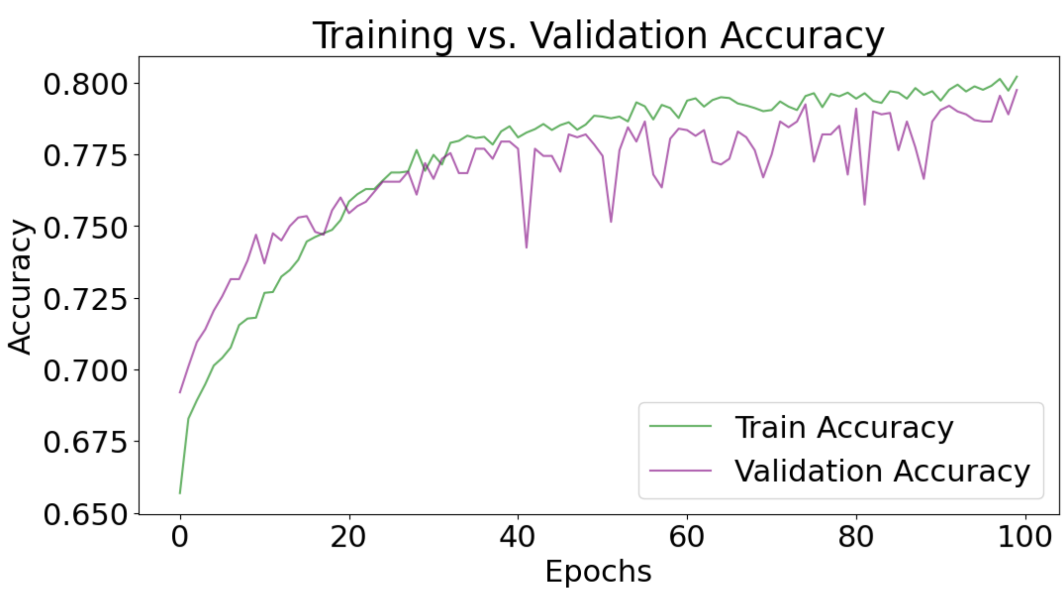
4. Training Monitoring *(5 points)*

During training, I monitored the following metrics:

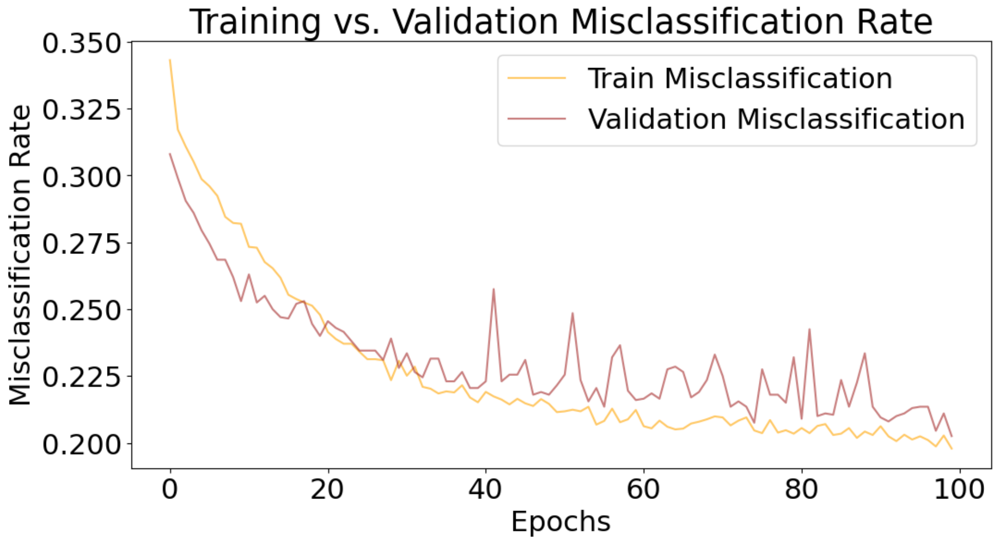
- **Training Loss**: Decreased over epochs, indicating learning.
- **Validation Loss**: Decreased initially, then plateaued as shown below:



- **Training Accuracy and Validation Accuracy**: Both increased over time as shown below:



- **Misclassification Rate**: Tracked for both training and validation as shown below:

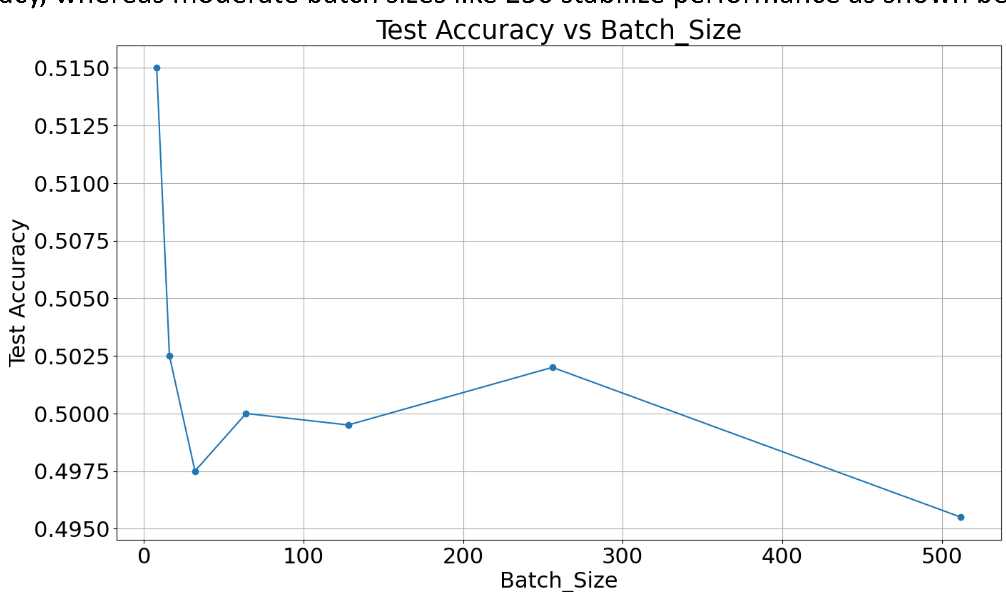


These metrics were logged per epoch, visualized in graphs, and validated for performance tracking.

5. Hyperparameter Tuning *(5 points)*
Three key hyperparameters were varied to analyze their impact on model performance:

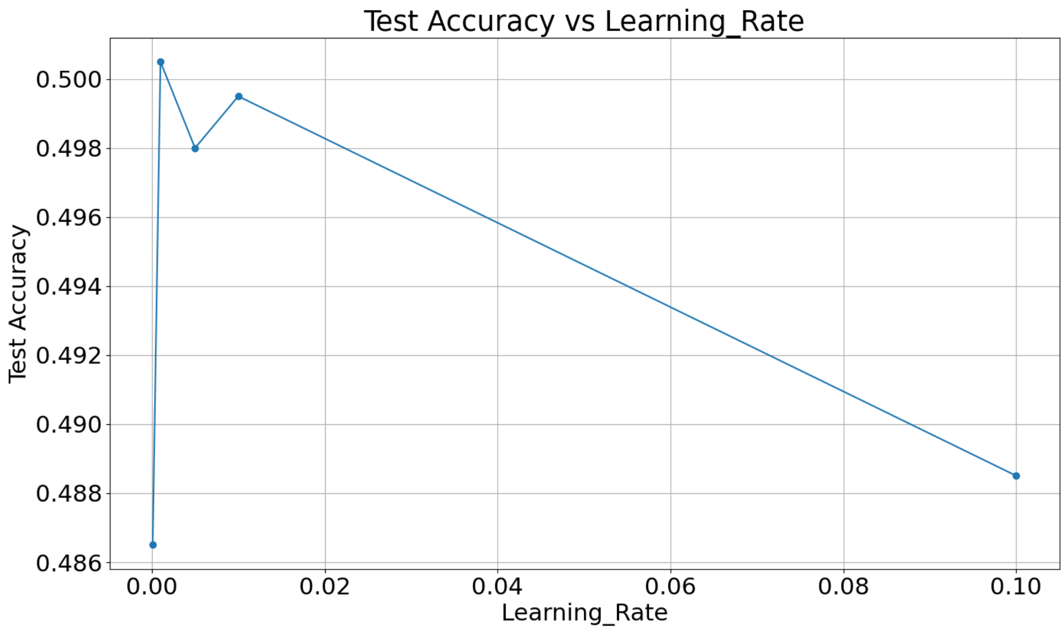
(i) Effect of Batch Size on Accuracy

- Batch sizes of [8, 16, 32, 64, 128, 256, 512] were tested.
- Larger batch sizes tend to perform more consistently but can sometimes reduce generalization.
- The results indicate that very small batch sizes (e.g., 8) can lead to highly variable accuracy, whereas moderate batch sizes like 256 stabilize performance as shown below:



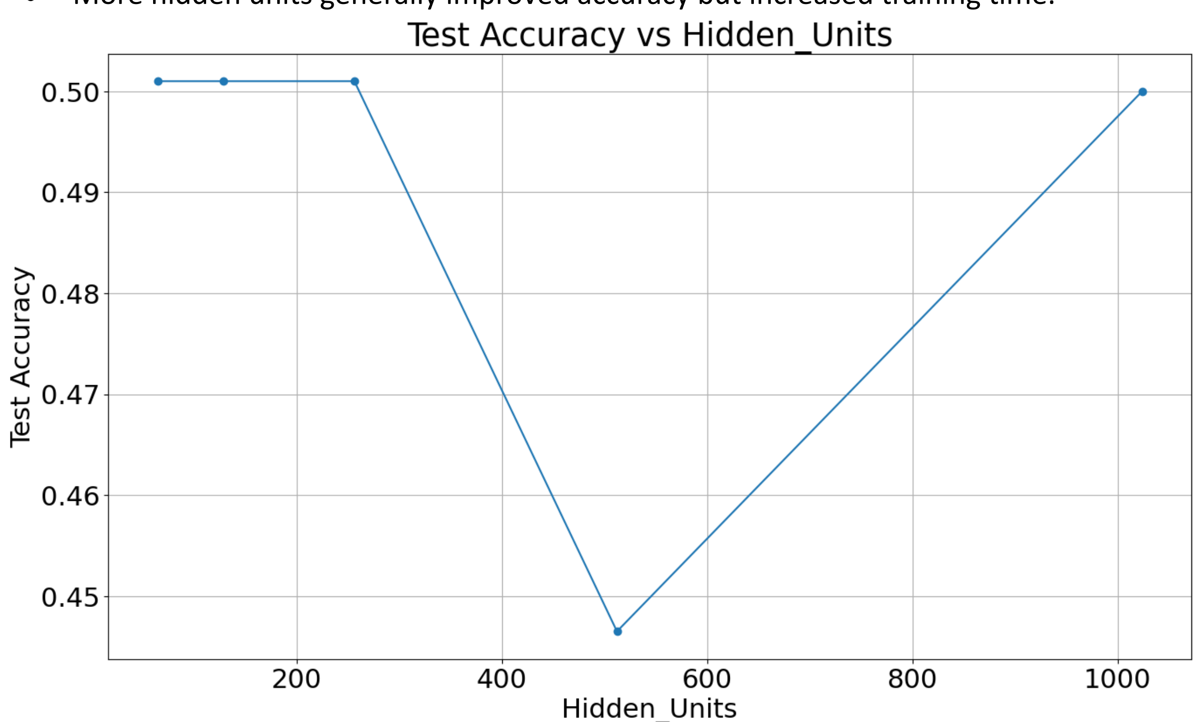
(ii) Effect of Learning Rate on Accuracy

- Learning rates of [0.0001, 0.001, 0.005, 0.01, 0.1] were tested.
- Too high learning rates led to instability; 0.001 worked best.



(iii) Effect of Hidden Units on Accuracy

- Tested [64, 128, 256, 512, 1024] hidden units.
- More hidden units generally improved accuracy but increased training time.



6. Model Performance

- **Accuracy Improvements**: Training with momentum helped stabilize accuracy growth.
- **Convergence Behavior**: Loss decreased steadily, confirming that gradients were correct.
- **Best Hyperparameter Settings**:
    - **Batch Size**: 64
    - **Learning Rate**: 0.001
    - **Hidden Units**: 256
- **Limitations**:
    - Overfitting risk with large hidden layers.
    - More complex models might require dropout or regularization techniques.