

COMP5313 Artificial Intelligence

Department of Computer Science

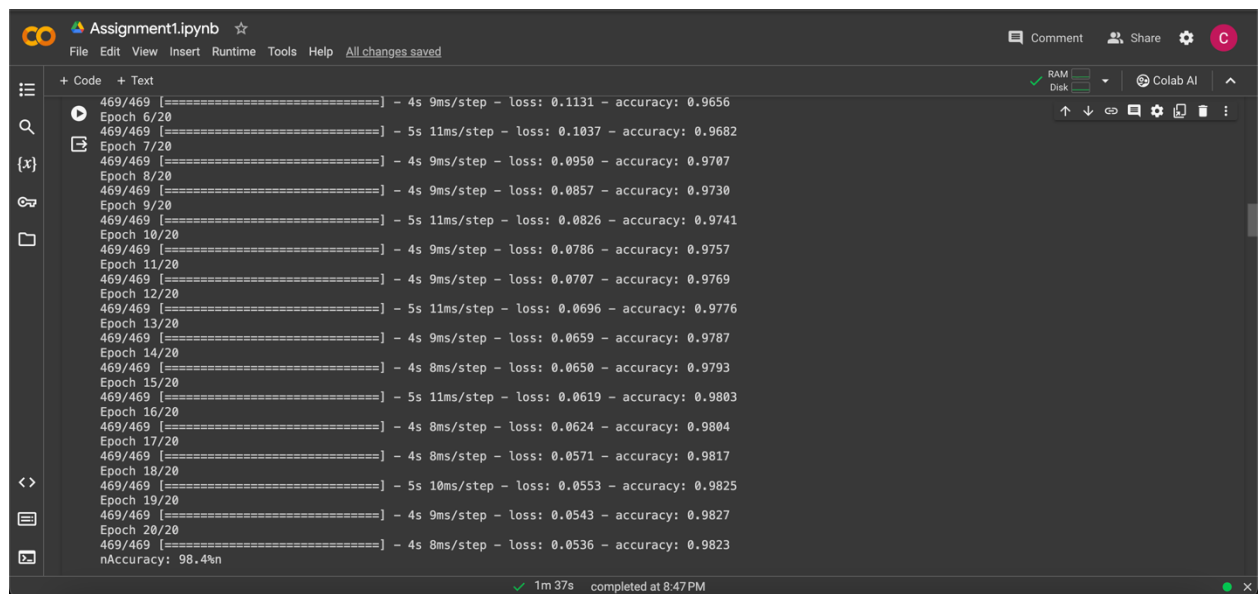
Assignment 1: MLP for Image Classification: Connectionist AI

1. Simple Three-Layer MLP

a. With Keras (As per tutorial [1])

The code implements a simple Multi-Layer Perceptron (MLP) using TensorFlow and Keras for classifying handwritten digits from the MNIST dataset. Key steps include loading the MNIST dataset, converting labels to one-hot encoding, reshaping the input images, and normalizing pixel values. The MLP consists of two hidden layers with ReLU activation and dropout for regularization, followed by an output layer with softmax activation. The model is compiled using categorical crossentropy loss and the Adam optimizer. It is then trained on the training set for 20 epochs with a batch size of 128. Finally, the model's accuracy is evaluated on the test set. The code uses Keras functionalities for model creation, training, and evaluation.

The obtained accuracy is 98.4%



```
469/469 [=====] - 4s 9ms/step - loss: 0.1131 - accuracy: 0.9656
Epoch 6/20
469/469 [=====] - 5s 11ms/step - loss: 0.1037 - accuracy: 0.9682
Epoch 7/20
469/469 [=====] - 4s 9ms/step - loss: 0.0950 - accuracy: 0.9707
Epoch 8/20
469/469 [=====] - 4s 9ms/step - loss: 0.0857 - accuracy: 0.9730
Epoch 9/20
469/469 [=====] - 5s 11ms/step - loss: 0.0826 - accuracy: 0.9741
Epoch 10/20
469/469 [=====] - 4s 9ms/step - loss: 0.0786 - accuracy: 0.9757
Epoch 11/20
469/469 [=====] - 4s 9ms/step - loss: 0.0707 - accuracy: 0.9769
Epoch 12/20
469/469 [=====] - 5s 11ms/step - loss: 0.0696 - accuracy: 0.9776
Epoch 13/20
469/469 [=====] - 4s 9ms/step - loss: 0.0659 - accuracy: 0.9787
Epoch 14/20
469/469 [=====] - 4s 8ms/step - loss: 0.0650 - accuracy: 0.9793
Epoch 15/20
469/469 [=====] - 5s 11ms/step - loss: 0.0619 - accuracy: 0.9803
Epoch 16/20
469/469 [=====] - 4s 8ms/step - loss: 0.0624 - accuracy: 0.9804
Epoch 17/20
469/469 [=====] - 4s 8ms/step - loss: 0.0571 - accuracy: 0.9817
Epoch 18/20
469/469 [=====] - 5s 10ms/step - loss: 0.0553 - accuracy: 0.9825
Epoch 19/20
469/469 [=====] - 4s 9ms/step - loss: 0.0543 - accuracy: 0.9827
Epoch 20/20
469/469 [=====] - 4s 8ms/step - loss: 0.0536 - accuracy: 0.9823
nAccuracy: 98.4%
```

b. Without Keras

The 2nd code implements a three-layer Multi-Layer Perceptron (MLP) using NumPy to classify handwritten digits from the MNIST dataset. It preprocesses the

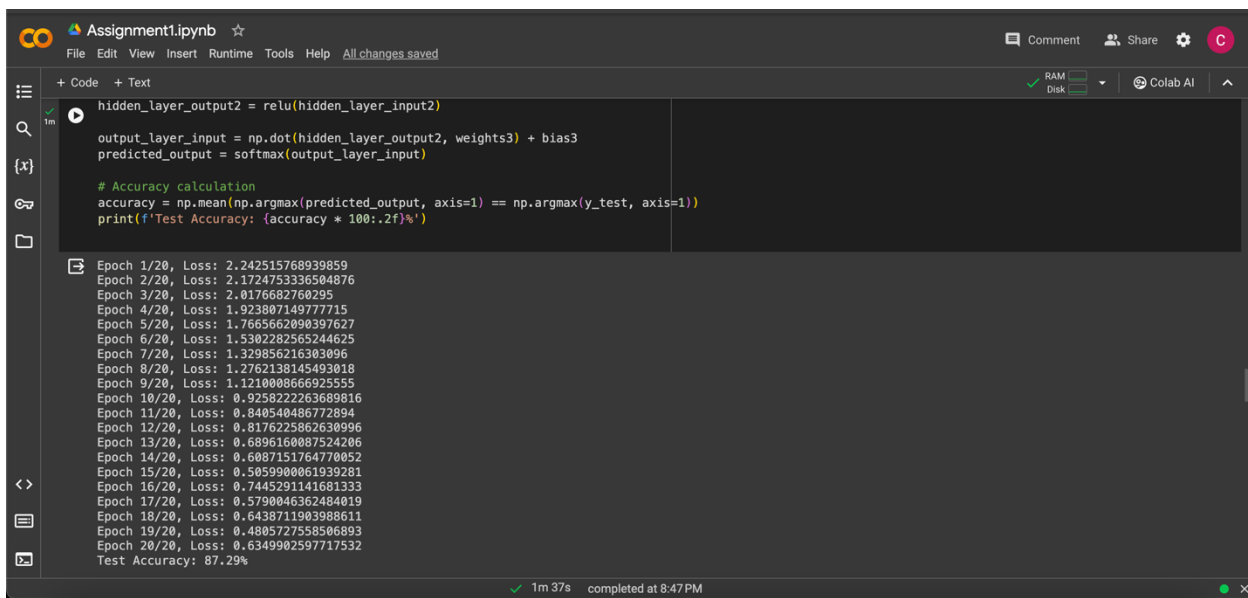
Chandana Sree Krishna

1226847

07 February 2024

data, initializes weights using Xavier/Glorot initialization, and employs mini-batch gradient descent for training. ReLU is used as the activation function for hidden layers, softmax for the output layer, and dropout for regularization. The model is trained for 20 epochs, and the accuracy is evaluated on the test set, achieving digit classification with neural network techniques. The code doesn't rely on deep learning frameworks like TensorFlow or Keras.

The obtained accuracy is 87.29%



The screenshot shows a Jupyter Notebook titled 'Assignment1.ipynb'. The code cell contains the following Python code:

```
hidden_layer_output2 = relu(hidden_layer_input2)

output_layer_input = np.dot(hidden_layer_output2, weights3) + bias3
predicted_output = softmax(output_layer_input)

# Accuracy calculation
accuracy = np.mean(np.argmax(predicted_output, axis=1) == np.argmax(y_test, axis=1))
print(f'Test Accuracy: {accuracy * 100:.2f}%')
```

The output cell displays the training progress over 20 epochs, showing the loss for each epoch and the final test accuracy:

```
Epoch 1/20, Loss: 2.242515768939859
Epoch 2/20, Loss: 2.1724753336504876
Epoch 3/20, Loss: 2.0176682760295
Epoch 4/20, Loss: 1.923807149777715
Epoch 5/20, Loss: 1.7665662090397627
Epoch 6/20, Loss: 1.5302282565244625
Epoch 7/20, Loss: 1.329856216303096
Epoch 8/20, Loss: 1.2762138145493018
Epoch 9/20, Loss: 1.121000866925555
Epoch 10/20, Loss: 0.9258222263680816
Epoch 11/20, Loss: 0.840540486772894
Epoch 12/20, Loss: 0.8176225862630996
Epoch 13/20, Loss: 0.6896160087524206
Epoch 14/20, Loss: 0.6087151764770052
Epoch 15/20, Loss: 0.5059900061939281
Epoch 16/20, Loss: 0.7445291141681333
Epoch 17/20, Loss: 0.5790046362484019
Epoch 18/20, Loss: 0.6438711903988611
Epoch 19/20, Loss: 0.4805727558506893
Epoch 20/20, Loss: 0.6349902597717532
Test Accuracy: 87.29%
```

2. ResMLP

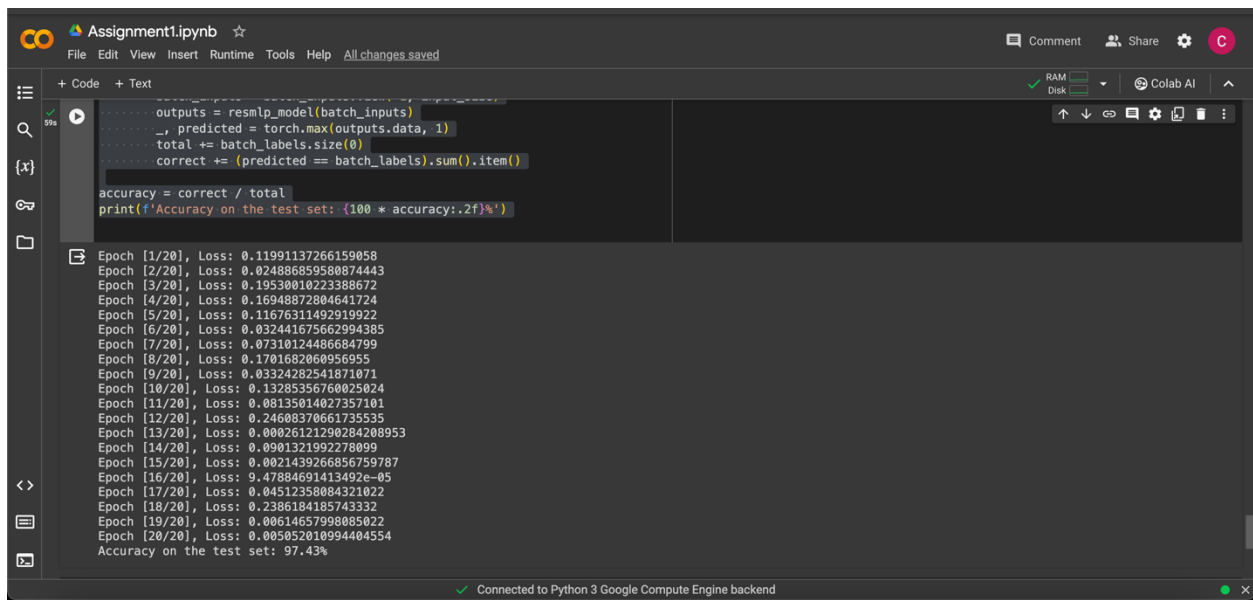
The code implements a Residual Multi-Layer Perceptron (ResMLP) model using PyTorch for classifying the MNIST dataset, which consists of handwritten digits (0-9). The ResMLP architecture includes a sequence of linear layers with GELU activations and residual connections through ResMLPBlock instances. The model is trained using the Adam optimizer and cross-entropy loss. The MNIST dataset is loaded using torchvision, transformed to tensors, and normalized. The training loop iterates over batches, computes the forward and backward passes, updates the model parameters, and prints the loss for each epoch. After training, the model is evaluated on the test set, and the accuracy is calculated and printed. The model achieves accuracy results on the MNIST test set.

The obtained accuracy is 97.43%

Chandana Sree Krishna

1226847

07 February 2024



```
outputs = resmlp_model(batch_inputs)
_, predicted = torch.max(outputs.data, 1)
total += batch_labels.size(0)
correct += (predicted == batch_labels).sum().item()

accuracy = correct / total
print(f'Accuracy on the test set: {100 * accuracy:.2f}%')
```

Epoch [1/20], Loss: 0.11991137266159058
Epoch [2/20], Loss: 0.024886859580874443
Epoch [3/20], Loss: 0.19530010223388672
Epoch [4/20], Loss: 0.16948872804641724
Epoch [5/20], Loss: 0.11676311492919922
Epoch [6/20], Loss: 0.032441675662994385
Epoch [7/20], Loss: 0.07310124486684799
Epoch [8/20], Loss: 0.1701682060956955
Epoch [9/20], Loss: 0.03324282541871071
Epoch [10/20], Loss: 0.13285356760025024
Epoch [11/20], Loss: 0.08135014027357101
Epoch [12/20], Loss: 0.24608370661735535
Epoch [13/20], Loss: 0.00826121290284208953
Epoch [14/20], Loss: 0.0981321092270099
Epoch [15/20], Loss: 0.0021439266856759787
Epoch [16/20], Loss: 0.47884691413492e-05
Epoch [17/20], Loss: 0.04512358084321022
Epoch [18/20], Loss: 0.2386184185743332
Epoch [19/20], Loss: 0.00614657998085022
Epoch [20/20], Loss: 0.005052010994404554
Accuracy on the test set: 97.43%

Comparison:

The Simple Three layer MLP model built with Keras performed the best with the highest accuracy of 98.4%.

The MLP model without Keras had a lower accuracy of 87.29%.

The ResMLP model with PyTorch achieved a good accuracy of 97.43%.

In summary, the Keras-based model stood out for having the highest accuracy, but other factors like simplicity and efficiency should also be considered depending on the specific use case.

Simple Three-Layer MLP with Keras:

Pros:

- Utilizes the high-level API Keras, making it concise and easy to understand.
- Achieves a high accuracy of 98.4% on the MNIST dataset.

Cons:

- May not provide as much flexibility for customization as lower-level implementations.

Simple Three-Layer MLP without Keras (NumPy-based):

Pros:

- Provides a low-level implementation using NumPy, offering more control over the model.
- Achieves a decent accuracy of 87.29% on the MNIST dataset.

Cons:

- Requires more manual coding compared to the Keras-based implementation.
- May not be as efficient or scalable for larger datasets.

Chandana Sree Krishna

1226847

07 February 2024

Residual Multi-Layer Perceptron (ResMLP) with PyTorch:

Pros:

- Implements a more advanced ResMLP model with residual connections using PyTorch.
- Achieves a good accuracy of 97.43% on the MNIST dataset.
- Utilizes PyTorch's automatic differentiation for efficient backpropagation.

Cons:

- Involves a slightly more complex implementation compared to the simple MLP models.

Conclusion:

If simplicity and high-level abstraction are prioritized, the Simple Three-Layer MLP with Keras might be preferable, especially with its impressive accuracy.

If a lower-level implementation with more control is preferred over the model, the Simple Three-Layer MLP without Keras (NumPy-based) is an option.

The Residual Multi-Layer Perceptron (ResMLP) with PyTorch offers a balance between abstraction and control, achieving good accuracy with the added benefit of using PyTorch's features.

References:

1. <https://www.analyticsvidhya.com/blog/2020/12/mlp-multilayer-perceptron-simple-overview/>
2. https://keras.io/examples/vision/mlp_image_classification/
3. <https://sh-tsang.medium.com/review-resmlp-feedforward-networks-for-image-classification-with-data-efficient-training-4eeb1eb5efa6>
4. Lecture slides of Dr. Sabah Mohammad