Identifier: jpenaarc
Highest score: 0.98

HW2: Classifying digit images (MNIST)

In this project, we're building neural networks (ANNs) to recognize handwritten digits. Each module is based on a simple ANN with two hidden layers, forming the basis for four distinct models. Neural networks function by simulating the brain's neural connections, processing input data through layers of interconnected neurons and adjusting weights to minimize prediction error during training. Each model in our project differs by incorporating a specific regularization technique, such as L2, Dropout and Early Stopping, which are crucial for preventing overfitting in neural networks. In general, these techniques work by penalizing large parameter values (complexity), improving the network's ability to generalize to unseen data. Through empirical analysis, we'll explore how each regularization technique affects the accuracy of our basic neural network in digit recognition.

Sections in the Report:

Method: The methodology section summaries the process of constructing our digit recognition models. It covers the architecture of simple artificial neural networks (ANNs), data preprocessing steps including normalization and augmentation and implementation of regularization techniques.

Result:

In the results section, we present the accuracy of each model when evaluated with validation data distinct from the training dataset. This section offers insights into the generalization performance of our models. Additionally, we provide visual representations of the confusion matrices in the form of heat maps for each model. These heat maps offer a comprehensive view of the model's performance in distinguishing between different digit classes, aiding in the interpretation of classification errors and overall model behavior.

Conclusion:

In this section, we'll discuss why we believe the best model performed better than the others and the factors that may have contributed to its success.

Method

ANN Architecture

Initialization: The neural network is created as a sequential model, meaning layers are stacked sequentially. This facilitates the flow of data from input to output layers.

Hidden Layers: Two hidden layers are added to the model, each containing 500 neurons. These layers serve as feature extractors, transforming the input data into a more meaningful representation through weighted connections and activation functions.

Activation Functions: The ReLU (Rectified Linear Unit) activation function is used in both hidden layers. ReLU is a non-linear function that introduces non-linearity to the network, allowing it to learn complex patterns and relationships in the data.

Output Layer: The output layer consists of 10 neurons, corresponding to the 10 possible digit classes (0 to 9). The sigmoid activation function is applied here, which squashes the output values between 0 and 1, effectively representing the probability that a given input belongs to each digit class.

Training Setup: The model is compiled with the Adam optimizer, a popular variant of stochastic gradient descent (SGD) known for its efficiency and effectiveness. The loss function is set to binary cross-entropy, which measures the difference between predicted and true labels for binary classification tasks. Additionally, accuracy is specified as a metric to monitor during training, providing insight into the model's performance.

Preprocessing

In the preprocessing section, the data obtained from the MNIST dataset is prepared for training the neural network. The provided function, `processTestData(X, y)`, conducts several essential preprocessing steps. Firstly, the input images (X) are reshaped into a two-dimensional array, with each row representing a flattened image of dimensions (28 x 28) converted to a single row of 784 pixels. This reshaping is necessary to match the input requirements of the neural network. Next, the MinMaxScaler from the preprocessing module is applied to normalize the pixel values of the input data, ensuring that they fall within the range [0, 1]. Normalization is crucial for enhancing the convergence and performance of the neural network during training.

Regularization

Regularization is a crucial technique in the context of neural networks, it is used for preventing overfitting, where the model learns the training data too well, impeding its ability to generalize unseen data. Regularization methods work by imposing constraints on the model's parameters during training, thereby reducing its capacity to fit noise in the training data. We used 3 commonly used regularization techniques in our base model: L2, Dropout, and Early Stopping.

L2 regularization penalizes large weights in the network by adding a term to the loss function proportional to the square of the weights. This additional term encourages the model to prefer smaller weights, effectively reducing the complexity of the model and

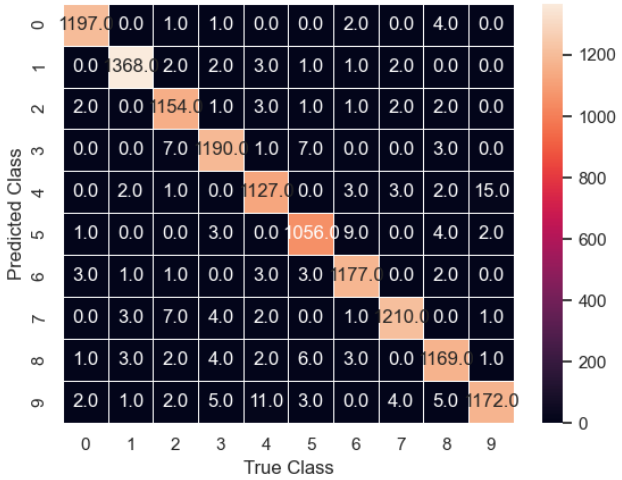
preventing overfitting. The strength of this regularization is controlled by a hyperparameter λ (lambda). By experimenting with different values of λ, such as 0.01, 0.001, and 0.1, we aim to find the optimal regularization strength for our neural network. This involves observing how the model's performance changes with varying levels of regularization, allowing us to strike a balance between reducing overfitting and maintaining good performance on unseen data.

Dropout is a technique used in training neural networks where, during each iteration, a random subset of neurons is temporarily removed from the network. This means their outputs are ignored, and their connections are skipped. By doing this randomly, Dropout prevents neurons from becoming overly dependent on each other and encourages them to learn more independently. This process helps prevent overfitting by promoting robustness and generalization, making the neural network better at handling new, unseen data. We tested by applying dropout to the input layer and then to all layers to see if results improved.

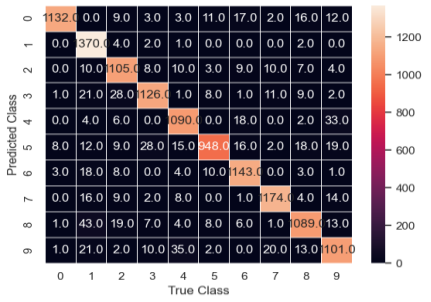
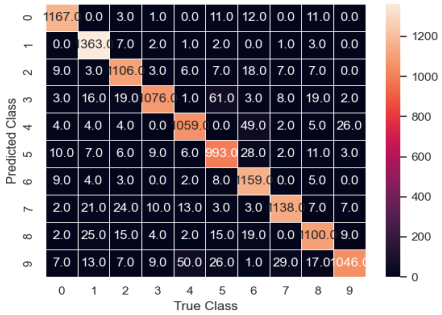
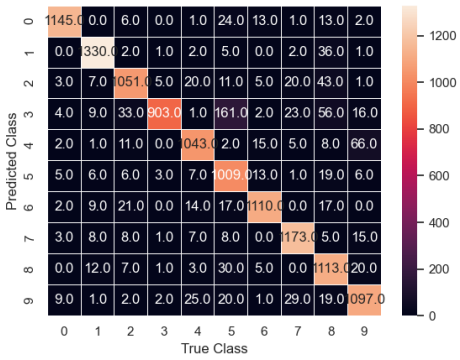
Early Stopping involves monitoring the model's performance on a separate validation dataset during training. Training stops when the accuracy stops improving, indicating that the model is beginning to overfit the training data. This prevents further optimization on noise in the training data and ensures better generalization to unseen data.

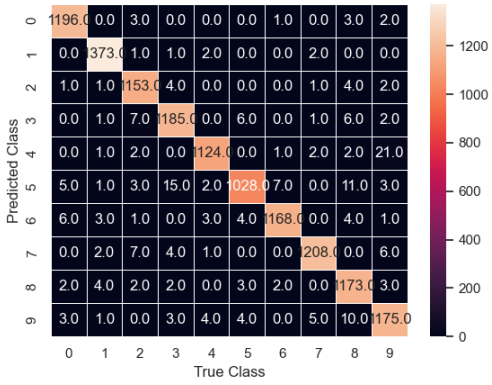
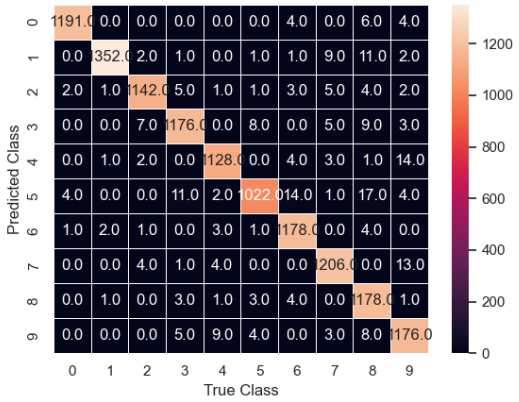
Result

Model 1(Basic architecture, no regularization)
Accuracy: 98.5%

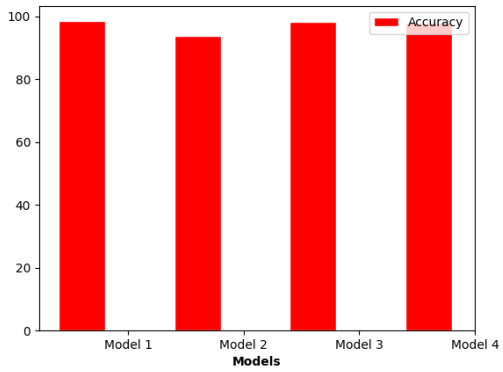


Model 2 (L2 Regularization)

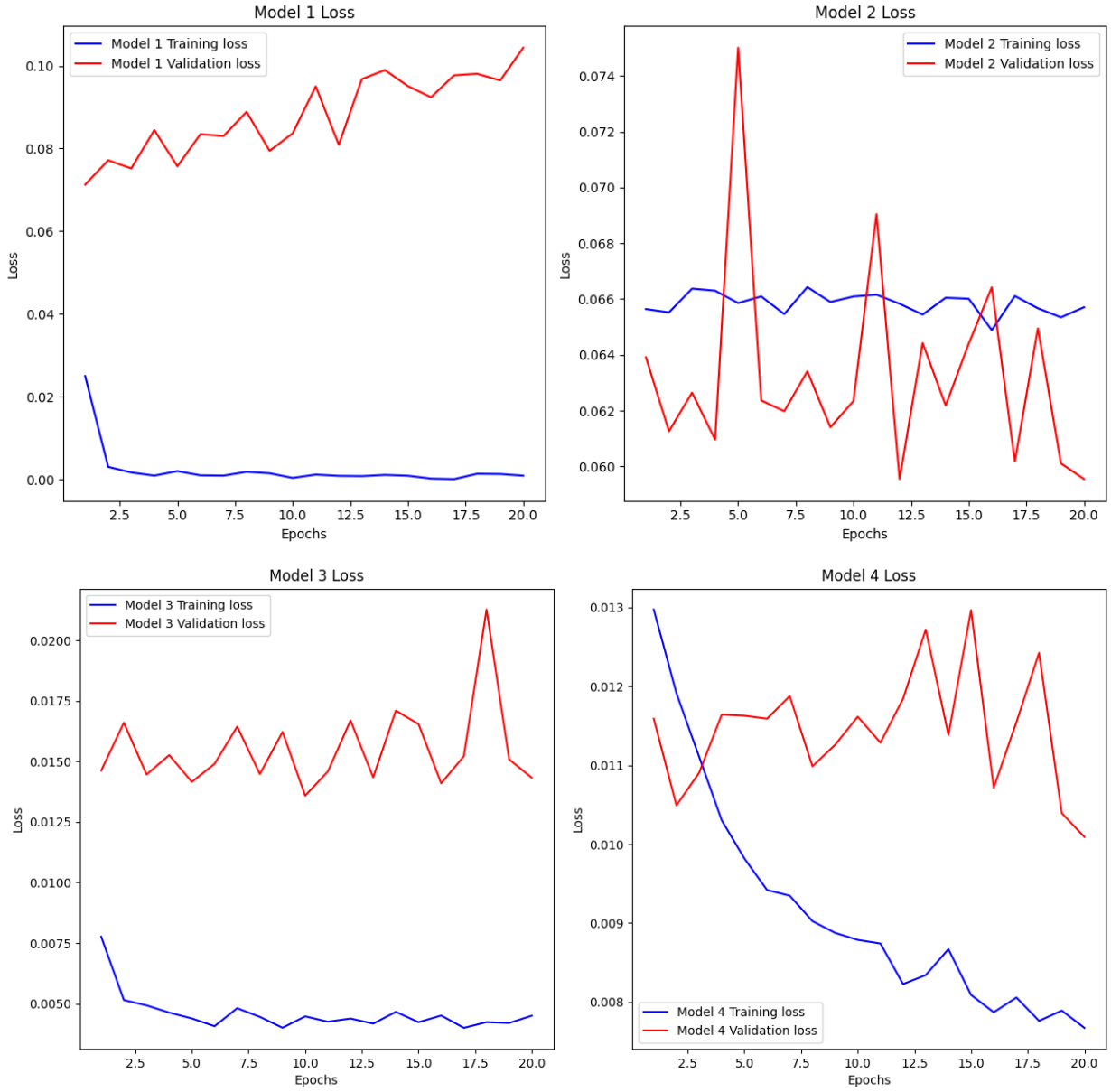
Lambda = 0.001	Lambda = 0.01	Lambda = 0.1
		
Accuracy = 93.98%	Accuracy = 93.38%	Accuracy = 91%

Model 3(Dropout) Accuracy = 98.35%	Model 4 (Early stopping) Accuracy = 97.9 Stopped at epoch 6
	

Accuracy of All models



Plots of the evolution of loss over training and validation data



Conclusion

Through our testing of various regularization techniques in artificial neural networks (ANNs), we found that the best-performing model did not employ any form of regularization. This outcome suggests that the parameters we used over-penalized the models and further experimentation was necessary, or the ANN architecture itself may inherently mitigate overfitting. These results highlight the need for tailored regularization approaches and ongoing refinement to fully leverage the potential of ANNs for specific tasks.