OCTMNIST Classification

1. **Describe the NN you have defined.**
   The neural networks described are convolutional neural networks (CNNs) for image classification tasks. They have two convolutional layers followed by max pooling, and then two fully connected layers for classification.
   a. SimpleCNN: This network has two convolutional layers with ReLU activation and max pooling, followed by two fully connected layers with ReLU activation. It is a basic CNN architecture.

```
SimpleCNN(
  (conv1): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu1): ReLU()
  (pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu2): ReLU()
  (pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=3136, out_features=128, bias=True)
  (relu3): ReLU()
  (fc2): Linear(in_features=128, out_features=4, bias=True)
)
```

1. Input: images.
2. Convolutional Layers:
Two convolutional layers with 32 and 64 filters, respectively.
ReLU activation after each convolutional layer.
3. Pooling Layers:
Two max pooling layers to reduce spatial dimensions.
4. Fully Connected Layers:
One hidden layer with 128 neurons.
ReLU activation after the hidden layer.
Output layer with 4 neurons for classification.

   b. CNNwRegularization: Similar to SimpleCNN, but likely includes regularization technique- weight decay to prevent overfitting.
   In this code snippet, L2 regularization, also known as weight decay, is added to the optimizer.

```python
# Create the model, loss function, and optimizer with weight decay for L2 regularization
model = CNNwRegularization()
criterion = nn.CrossEntropyLoss()
weight_decay = 1e-5  # weight decay
optimizer = optim.Adam(model.parameters(), lr=0.001, weight_decay=weight_decay)

# Learning rate scheduler
scheduler = lr_scheduler.ReduceLROnPlateau(optimizer, mode='min', factor=0.1, patience=3, verbose=True)
```

In the optimizer definition, the weight_decay parameter is set to 1e-5, which determines the strength of the regularization. When the optimizer updates the model's parameters, it adds a term to the gradient that is proportional to the weight value itself, scaled by the weight_decay parameter. This encourages the model to prefer smaller weights, which can help prevent overfitting by reducing the complexity of the model.

```
CNNwRegularization(
  (conv1): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu1): ReLU()
  (pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu2): ReLU()
  (pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=3136, out_features=128, bias=True)
  (relu3): ReLU()
(fc2): Linear(in_features=128, out_features=4, bias=True)
)
```

c. NN_regularization_dropout: This network includes dropout regularization after the last ReLU activation in the fully connected layers. Dropout helps prevent overfitting by randomly setting a fraction of input units to zero during training.

```
NN_regularization_dropout(
  (conv1): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu1): ReLU()
  (pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu2): ReLU()
  (pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=3136, out_features=128, bias=True)
  (relu3): ReLU()
  (dropout): Dropout(p=0.5, inplace=False)
  (fc2): Linear(in_features=128, out_features=4, bias=True)
)
```

d. NN_regularization_dropout_earlystopping: Similar to NN_regularization_dropout, but may also include early stopping as a regularization technique. Early stopping stops training when the performance on a validation dataset starts to degrade, preventing overfitting.

```
NN_regularization_dropout_earlystopping(
  (conv1): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu1): ReLU()
  (pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (relu2): ReLU()
  (pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=3136, out_features=128, bias=True)
  (relu3): ReLU()
  (dropout): Dropout(p=0.5, inplace=False)
  (fc2): Linear(in_features=128, out_features=4, bias=True)
)
```

How early stopping works in the code:

1. During each epoch, the validation loss is compared to the best validation loss (best_val_loss). If the current validation loss is lower (better) than the best validation loss so far, the current model weights are saved as the best model weights (best_model_wts).
2. If the validation loss does not improve (i.e., it does not decrease) for patience number of epochs, the no_improve_count is incremented.
3. If no_improve_count reaches patience, indicating that there has been no improvement in the validation loss for patience consecutive epochs, training is stopped early by breaking out of the training loop.

2. Describe how the techniques (regularization, dropout, early stopping) have impacted the performance of the model.

How each technique impacted the performance of the SimpleCNN model:

1. Regularization:
   - Impact: After applying regularization, the test accuracy slightly decreased from 0.8977 to 0.8973. The test loss increased from 0.3115 to 0.3245. However, the precision, recall, and F1-score improved slightly.
   - Explanation: Regularization adds a penalty term to the loss function, discouraging the model from learning complex patterns that may be noise in the training data. This can help prevent overfitting and improve generalization.

2. Dropout:
   - Impact: After introducing dropout, the test accuracy decreased further to 0.8959. The test loss decreased to 0.2949. The precision improved, but recall and F1-score decreased.
   - Explanation: Dropout randomly sets a fraction of input units to zero during training, which helps prevent units from co-adapting and improves the generalization of the model. However, it can also lead to a slight decrease in performance due to the reduction in model capacity during training.

3. Early Stopping:
   - Impact: After applying early stopping, the test accuracy decreased significantly to 0.8753. The test loss increased to 0.3446. The precision, recall, and F1-score also decreased.
   - Explanation: Early stopping stops training when the performance on a validation dataset starts to degrade, preventing the model from overfitting to the training data. However, sometimes it can lead to a decrease in performance if the model stops training before it has converged to the optimal solution.

Overall, regularization and dropout improved the generalization of the model slightly. The impact of these techniques can vary depending on the specific dataset and model architecture.

3. Discuss the results and provide relevant graphs:

a. Report training accuracy, training loss, validation accuracy, validation loss, testing accuracy, and testing loss.

i. Simple CNN : Test Accuracy: 0.8977 Test Loss: 0.3115 Precision: 0.8340, Recall: 0.7924, F1-score: 0.8088 Val Loss: 0.3085 Acc: 0.8993 Train Loss: 0.1647 Acc: 0.9442

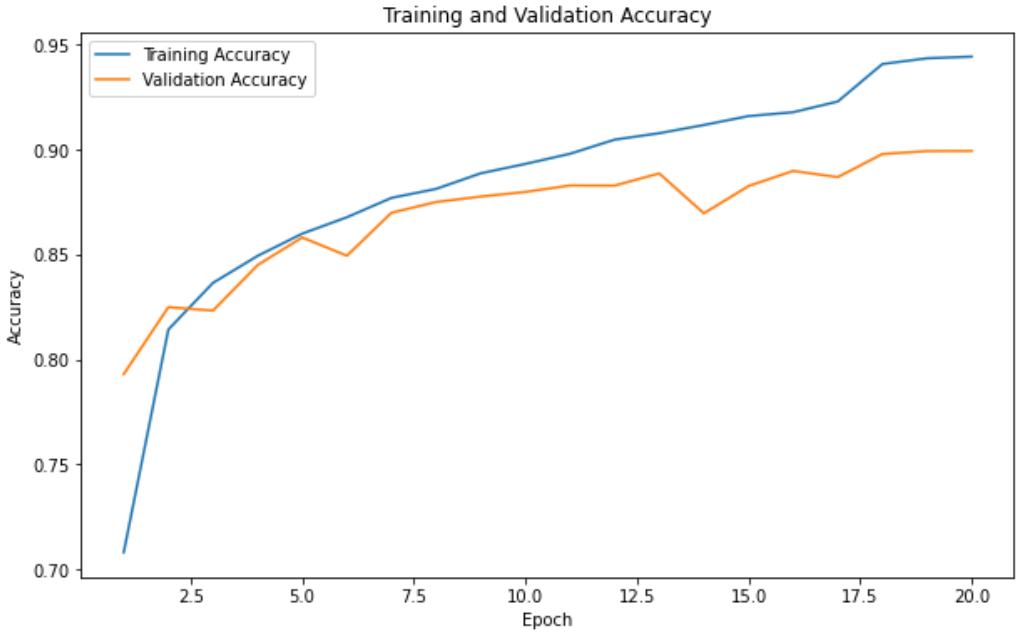
ii. Regularization: Test Accuracy: 0.8973 Test Loss: 0.3245 Precision: 0.8279, Recall: 0.7998, F1-score: 0.8106 Train Loss: 0.1571 Acc: 0.9454 Val Loss: 0.3214 Acc: 0.8982

iii. Dropout: Test Accuracy: 0.8959 Test Loss: 0.2949 Precision: 0.8403, Recall: 0.7715, F1-score: 0.7934 Train Loss: 0.2683 Acc: 0.9055 Val Loss: 0.3052 Acc: 0.8977

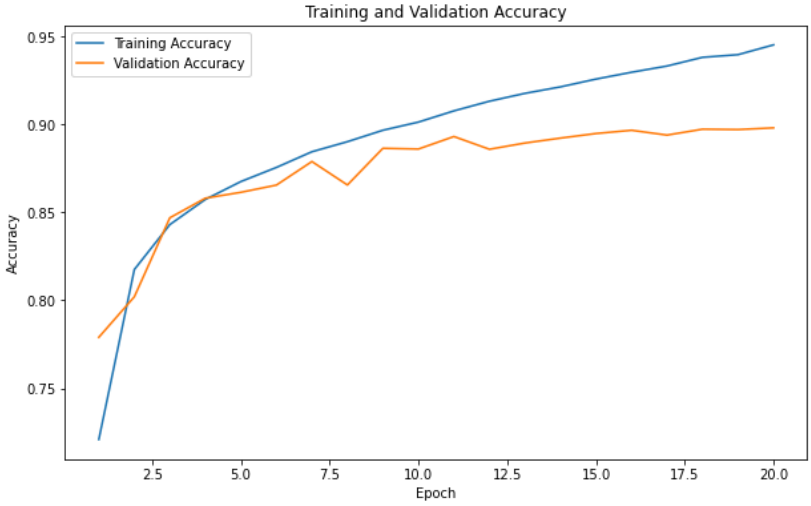
iv. Early Stopping: Test Accuracy: 0.8753 Test Loss: 0.3446 Precision: 0.8109, Recall: 0.7075, F1-score: 0.7183 Train Loss: 0.3553 Acc: 0.8762 Val Loss: 0.3516 Acc: 0.8779

b. Plot the training and validation accuracy over time (epochs). (Observations for b and c written after c)

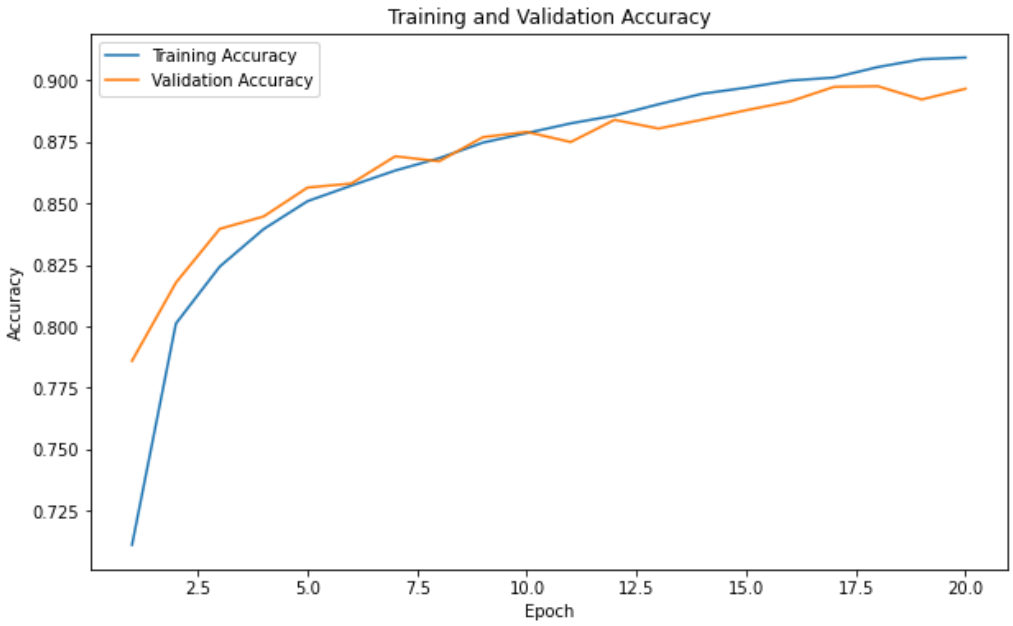
1. Simple CNN:



2. After regularization:

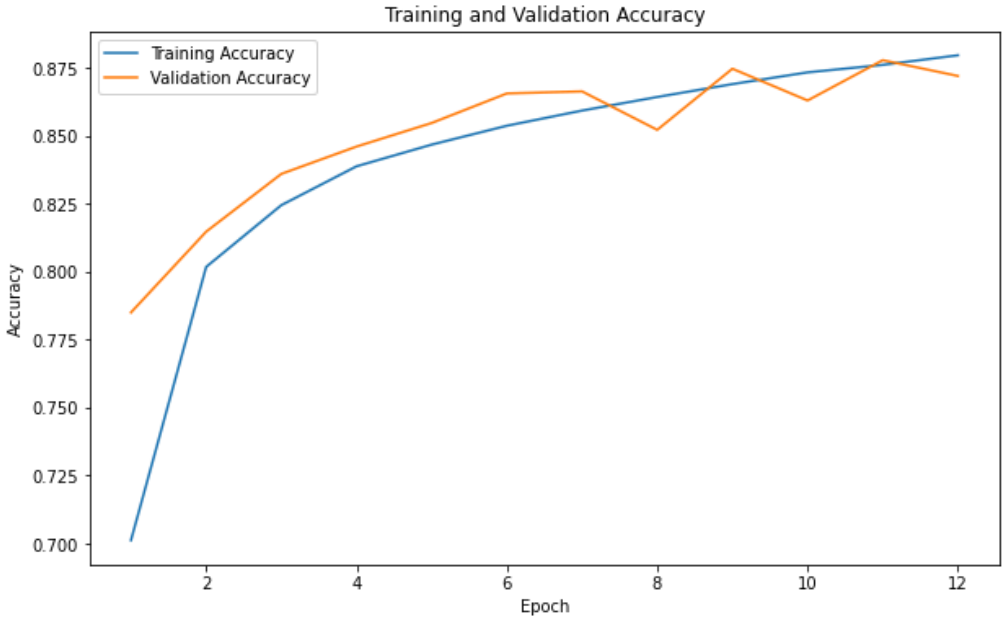


3. After Dropout:



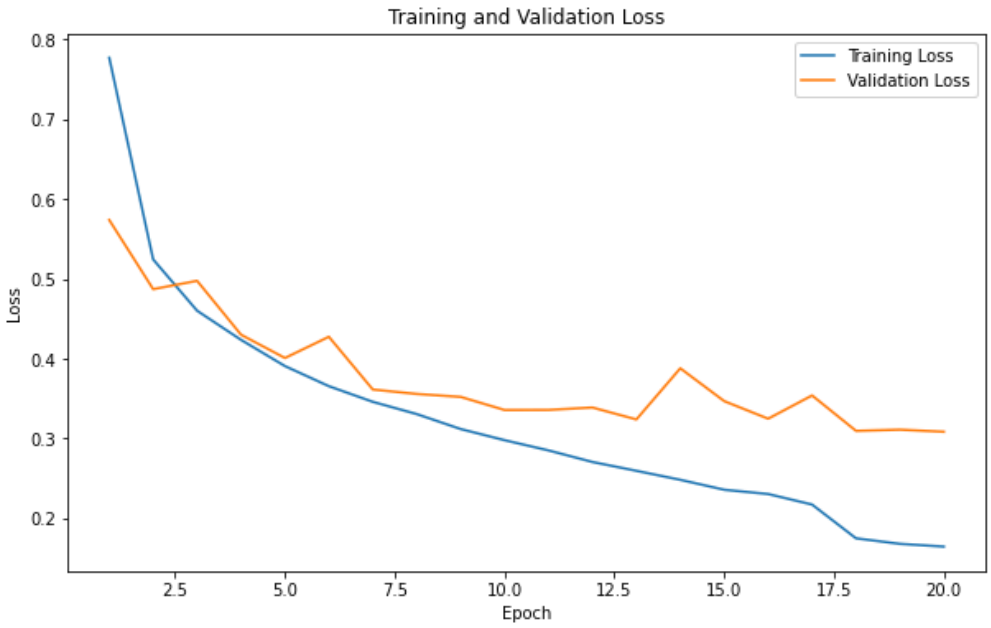
4.

After early stopping:

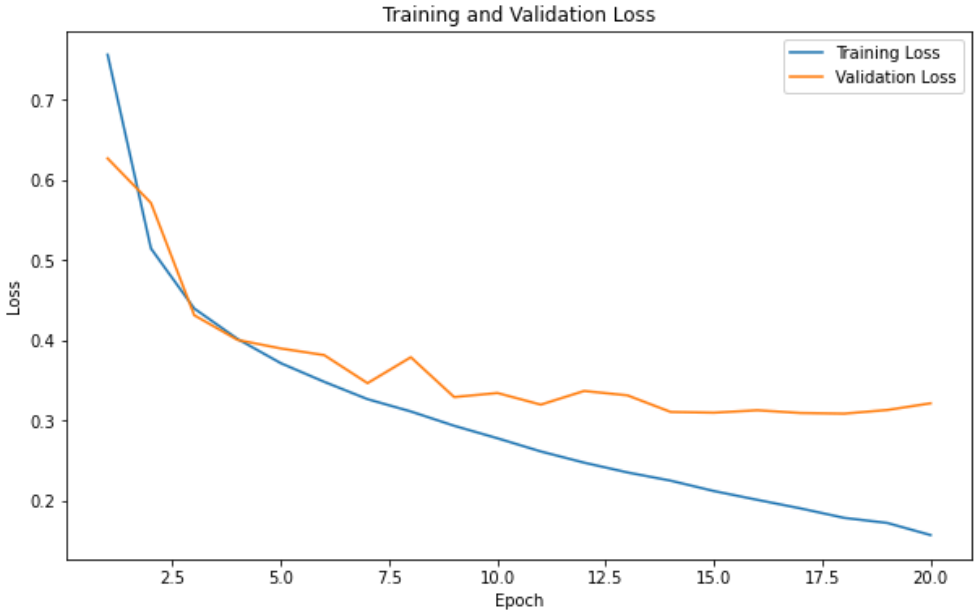


c. Plot the training and validation loss over time (epochs).

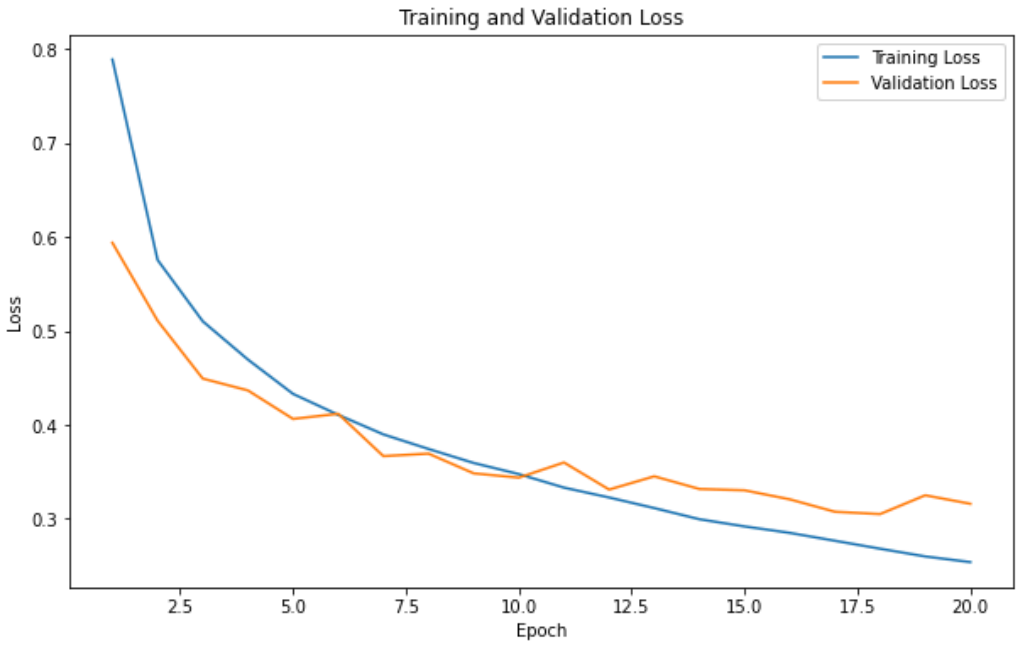
1. Simple CNN:



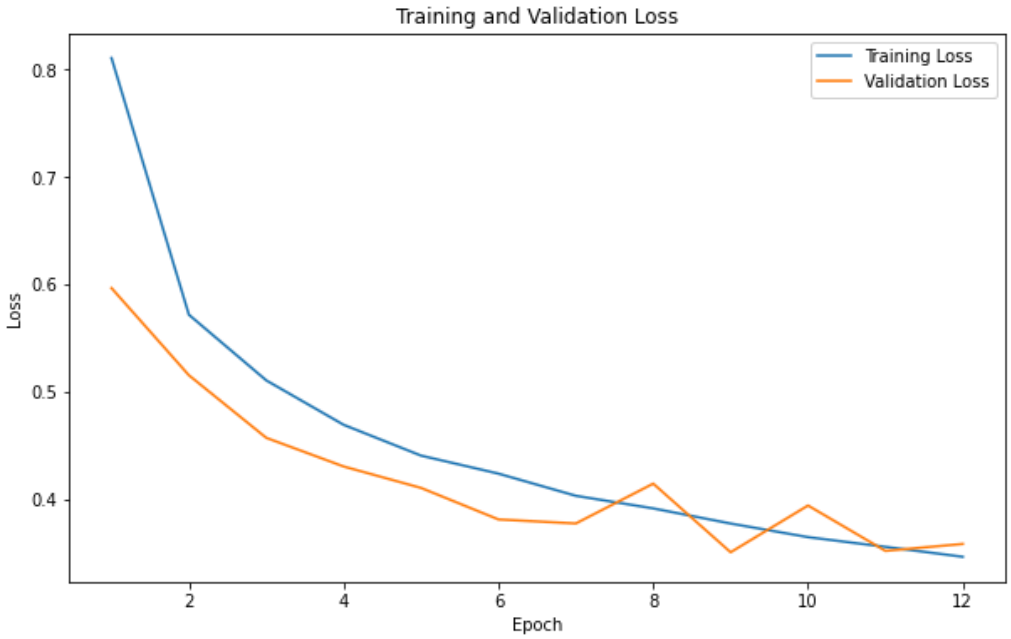
2. After Regularization:



3. After Dropout:



4. After early stopping:



Observations:

1. **Simple CNN**
    a. The training shows a consistent improvement in both training and validation accuracy over the 20 epochs, indicating that the model is learning from the data.

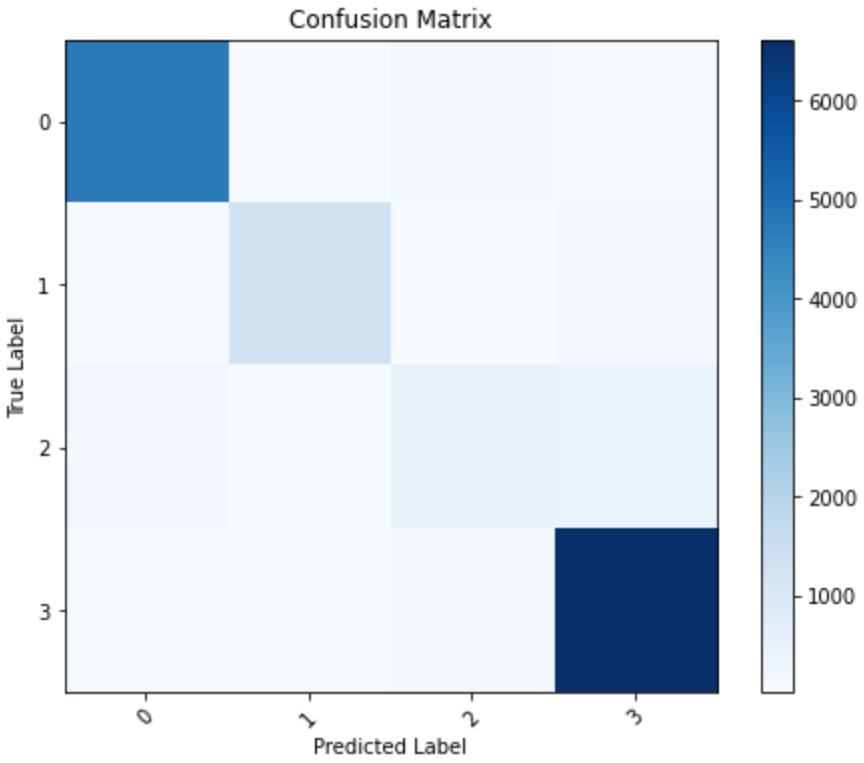
b. he training loss steadily decreases, indicating that the model is fitting the training data better.
c. However, the validation loss initially decreases but then stabilizes, suggesting that the model may start to overfit the training data towards the later epochs.

2. **After regularization**
   a. The training trend for this model shows a consistent improvement in both training and validation accuracy over the 20 epochs, indicating that the model is learning effectively from the data.
   b. The training loss steadily decreases, indicating that the model is fitting the training data well.
   c. The validation loss also decreases initially but then stabilizes, suggesting that the model is not overfitting and generalizing well to unseen data.
3. **After Dropout**
   a. The trend in this training shows a consistent improvement in both training and validation accuracy over the 20 epochs.
   b. Initially, there is a significant decrease in both training and validation losses, indicating that the model is learning well.
   c. As the training progresses, the rate of improvement slows down, but the model continues to learn, as evidenced by the decreasing losses and increasing accuracy.
4. **After early stopping**
   a. In this training trend, the model starts with a relatively high training loss and accuracy, indicating that it is not yet effectively capturing the patterns in the data. However, over the epochs, both the training and validation losses decrease, and the accuracies increase, suggesting that the model is learning and improving its performance.
   b. The training loss decreases steadily, indicating that the model is fitting the training data better. However, the validation loss shows some fluctuations, indicating that the model's performance on the validation set is not consistently improving.
   c. After epoch 9, there is no significant improvement in the validation loss for three consecutive epochs, leading to early stopping. This suggests that the model may have started to overfit the training data, as it is not generalizing well to the validation set.

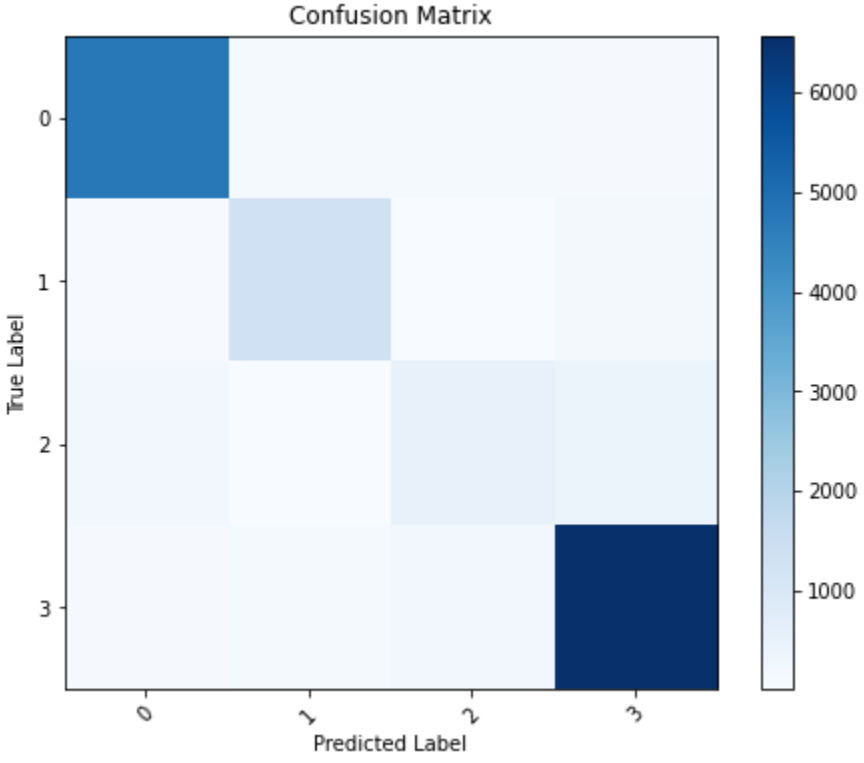
d. Generate a confusion matrix using the model's predictions on the test set.

(Observations of d and e written after e)

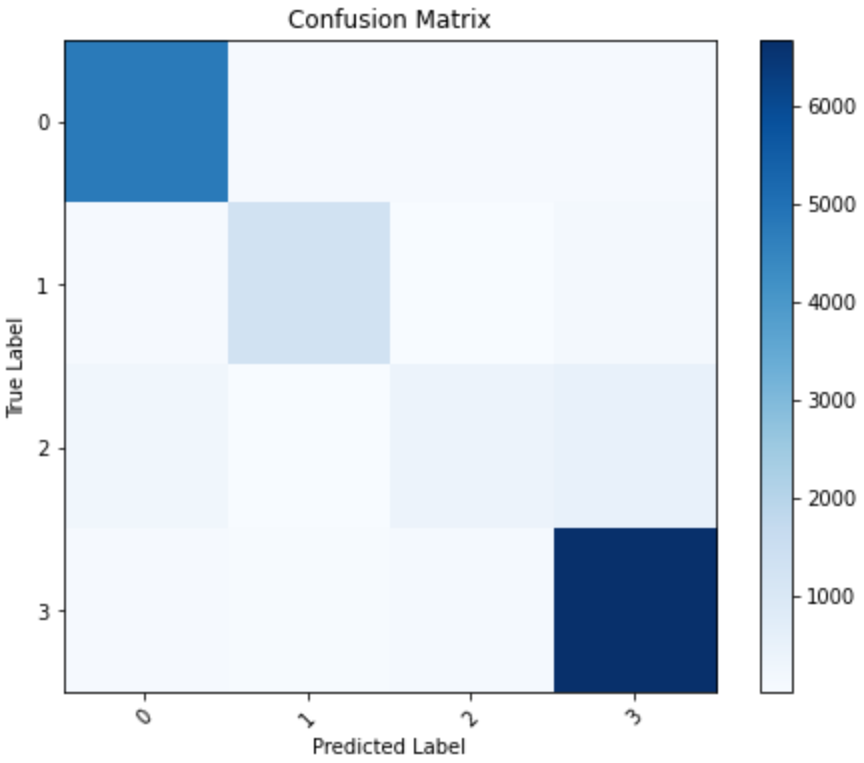
1. Simple CNN:



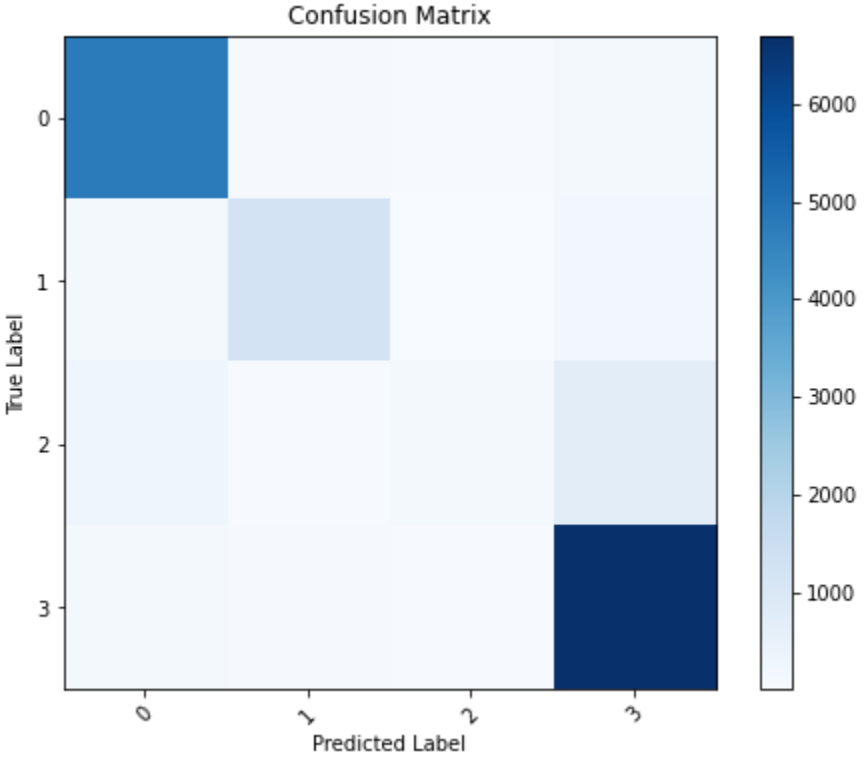
2. After regularization



3. After dropout:



4. After early stopping:



a. **Report any other evaluation metrics used to analyze the model's performance on the test set.**

1. Simple CNN:

```
Confusion Matrix:
[[4717   60  133   91]
 [  74 1308   23  136]
 [ 181   29  488  456]
 [  77   79  157 6614]]
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.94      0.94      5001
           1       0.89      0.85      0.87      1541
           2       0.61      0.42      0.50      1154
           3       0.91      0.95      0.93      6927

    accuracy                           0.90     14623
   macro avg       0.83      0.79      0.81     14623
weighted avg       0.89      0.90      0.89     14623
```

2. After Regularization:

```
Confusion Matrix:
[[4723   95  110   73]
 [  65 1330   18  128]
 [ 199   37  514  404]
 [  76   95  202 6554]]
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.94      0.94      5001
           1       0.85      0.86      0.86      1541
           2       0.61      0.45      0.51      1154
           3       0.92      0.95      0.93      6927

    accuracy                           0.90     14623
   macro avg       0.83      0.80      0.81     14623
weighted avg       0.89      0.90      0.89     14623
```

3.

After Dropout:

```
Confusion Matrix:
[[4758   67   88   88]
 [  73 1287   15  166]
 [ 223   22  389  520]
 [  89   57  114 6667]]
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.95      0.94      5001
           1       0.90      0.84      0.87      1541
           2       0.64      0.34      0.44      1154
           3       0.90      0.96      0.93      6927

    accuracy                           0.90     14623
   macro avg       0.84      0.77      0.79     14623
weighted avg       0.89      0.90      0.89     14623
```

4.

After early stopping:

```
Confusion Matrix:
[[4744   70   54  133]
 [ 112 1222    3  204]
 [ 278   33  141  702]
 [ 128   69   37 6693]]
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.95      0.92      5001
           1       0.88      0.79      0.83      1541
           2       0.60      0.12      0.20      1154
           3       0.87      0.97      0.91      6927

    accuracy                           0.88     14623
   macro avg       0.81      0.71      0.72     14623
weighted avg       0.86      0.88      0.85     14623
```

Trend analysis of how the performance metrics change after adding different modifications (regularization, dropout, and early stopping) to the Simple CNN model:

1. Accuracy:
Simple CNN: 0.90
Regularization: 0.90
Dropout: 0.90

Early Stopping: 0.88
Trend: Accuracy remains relatively stable with regularization and dropout but decreases slightly with early stopping.

2. Loss:
Simple CNN: 0.3115
Regularization: 0.3245
Dropout: 0.2949
Early Stopping: 0.3446

Trend: Loss fluctuates slightly with each modification, with dropout showing the lowest loss.

3. Precision:
Simple CNN: 0.8340
Regularization: 0.8279
Dropout: 0.8403
Early Stopping: 0.8109

Trend: Precision remains relatively stable with regularization and dropout, but decreases slightly with early stopping.

4. Recall:
Simple CNN: 0.7924
Regularization: 0.7998
Dropout: 0.7715
Early Stopping: 0.7075

Trend: Recall remains relatively stable with regularization and dropout, but decreases significantly with early stopping.

5. F1-score:
Simple CNN: 0.8088
Regularization: 0.8106
Dropout: 0.7934
Early Stopping: 0.7183

Trend: F1-score remains relatively stable with regularization and dropout, but decreases with early stopping.

Overall, regularization and dropout generally have a stabilizing effect on the model's performance, maintaining or slightly improving most metrics as the dataset is clean and was giving good accuracy without any modifications. However, early stopping tends to decrease performance, particularly in terms of recall and F1-score, indicating that it may have used less resources, leading to underfitting.

Conclusion:

The trends observed in the different training scenarios highlight the effectiveness of regularization techniques in improving model performance.

1. The Simple CNN model shows signs of overfitting, but regularization helps stabilize the model and prevent overfitting, leading to better generalization.
2. Dropout further enhances the model's ability to generalize by reducing over-reliance on specific features.
3. Early stopping prevents overfitting but suggests that the model may not have fully converged, indicating potential for further improvement.