lab 2

January 29, 2025

1 Step 1: Baseline Model

1. Import Libraries:

```
[1]: import numpy as np
     import matplotlib.pyplot as plt
     from tensorflow.keras.datasets import mnist
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Dense, Flatten
     from tensorflow.keras.utils import to_categorical
```

2. Load and Preprocess Data:

```
[2]: (x_train, y_train), (x_test, y_test) = mnist.load_data()
     x_train = x_train.astype('float32') / 255.0
     x_test = x_test.astype('float32') / 255.0
     y_train = to_categorical(y_train, 10)
     y_test = to_categorical(y_test, 10)
```

3) Build Baseline Model:

```
[3]: baseline_model = Sequential()
     baseline_model.add(Flatten(input_shape=(28, 28)))
     baseline_model.add(Dense(128, activation='relu'))
     baseline_model.add(Dense(10, activation='softmax'))

     baseline_model.compile(optimizer='adam', loss='categorical_crossentropy',␣
      ↪metrics=['accuracy'])
```

```
c:\Users\eakes\AppData\Local\Programs\Python\Python311\Lib\site-
packages\keras\src\layers\reshaping\flatten.py:37: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
```

4) Train Baseline Model

```
[4]: history_baseline = baseline_model.fit(x_train, y_train, epochs=10,␣
      ↪batch_size=32, validation_split=0.2)
```

```
Epoch 1/10
1500/1500          4s 2ms/step -
accuracy: 0.8654 - loss: 0.4723 - val_accuracy: 0.9582 - val_loss: 0.1478
Epoch 2/10
1500/1500          5s 1ms/step -
accuracy: 0.9612 - loss: 0.1365 - val_accuracy: 0.9659 - val_loss: 0.1169
Epoch 3/10
1500/1500          2s 1ms/step -
accuracy: 0.9727 - loss: 0.0906 - val_accuracy: 0.9694 - val_loss: 0.1030
Epoch 4/10
1500/1500          2s 1ms/step -
accuracy: 0.9811 - loss: 0.0662 - val_accuracy: 0.9726 - val_loss: 0.0989
Epoch 5/10
1500/1500          2s 1ms/step -
accuracy: 0.9853 - loss: 0.0497 - val_accuracy: 0.9725 - val_loss: 0.0918
Epoch 6/10
1500/1500          3s 2ms/step -
accuracy: 0.9898 - loss: 0.0355 - val_accuracy: 0.9709 - val_loss: 0.0980
Epoch 7/10
1500/1500          2s 1ms/step -
accuracy: 0.9917 - loss: 0.0279 - val_accuracy: 0.9766 - val_loss: 0.0868
Epoch 8/10
1500/1500          2s 1ms/step -
accuracy: 0.9943 - loss: 0.0199 - val_accuracy: 0.9732 - val_loss: 0.0990
Epoch 9/10
1500/1500          2s 1ms/step -
accuracy: 0.9944 - loss: 0.0193 - val_accuracy: 0.9758 - val_loss: 0.0882
Epoch 10/10
1500/1500          2s 1ms/step -
accuracy: 0.9954 - loss: 0.0151 - val_accuracy: 0.9752 - val_loss: 0.0942
```

5) Evaluate Baseline Model:

[5]:
```python
baseline_loss, baseline_accuracy = baseline_model.evaluate(x_test, y_test)
print(f'Baseline Test Loss: {baseline_loss}, Test Accuracy:␣
 ↪{baseline_accuracy}')
```

```
313/313          0s 1ms/step -
accuracy: 0.9759 - loss: 0.0859
Baseline Test Loss: 0.07402931153774261, Test Accuracy: 0.9793000221252441
```

2 Step 2: L1 and L2 Regularization

1) Modify Model with L1 and L2 Regularization:

[6]:
```python
from tensorflow.keras.regularizers import l1, l2

def create_model_l1(lam):
    model = Sequential()
    model.add(Flatten(input_shape=(28, 28)))
    model.add(Dense(128, activation='relu', kernel_regularizer=l1(lam)))
    model.add(Dense(10, activation='softmax'))
    return model

def create_model_l2(lam):
    model = Sequential()
    model.add(Flatten(input_shape=(28, 28)))
    model.add(Dense(128, activation='relu', kernel_regularizer=l2(lam)))
    model.add(Dense(10, activation='softmax'))
    return model
```

2) Train and Evaluate Models with Different Regularization Strengths:

[7]:
```
l1_strengths = [0.01, 0.1, 0.5]
for lam in l1_strengths:
    model = create_model_l1(lam)
    model.compile(optimizer='adam', loss='categorical_crossentropy',␣
  ↪metrics=['accuracy'])
    model.fit(x_train, y_train, epochs=10, batch_size=32, validation_split=0.2)
    loss, accuracy = model.evaluate(x_test, y_test)
    print(f'L1 Regularization ( ={lam}) - Test Loss: {loss}, Test Accuracy:␣
  ↪{accuracy}')
```

```
Epoch 1/10
1500/1500            4s 2ms/step -
accuracy: 0.7523 - loss: 5.7128 - val_accuracy: 0.8571 - val_loss: 1.2544
Epoch 2/10
1500/1500            2s 1ms/step -
accuracy: 0.8449 - loss: 1.2445 - val_accuracy: 0.8662 - val_loss: 1.0908
Epoch 3/10
1500/1500            2s 1ms/step -
accuracy: 0.8571 - loss: 1.1147 - val_accuracy: 0.8568 - val_loss: 1.0938
Epoch 4/10
1500/1500            3s 2ms/step -
accuracy: 0.8597 - loss: 1.0650 - val_accuracy: 0.8708 - val_loss: 0.9913
Epoch 5/10
1500/1500            5s 2ms/step -
accuracy: 0.8681 - loss: 1.0181 - val_accuracy: 0.8705 - val_loss: 0.9917
Epoch 6/10
1500/1500            2s 1ms/step -
accuracy: 0.8660 - loss: 1.0107 - val_accuracy: 0.8729 - val_loss: 0.9676
Epoch 7/10
1500/1500            2s 2ms/step -
accuracy: 0.8690 - loss: 0.9746 - val_accuracy: 0.8738 - val_loss: 0.9450
Epoch 8/10
1500/1500            2s 2ms/step -
```

```
accuracy: 0.8688 - loss: 0.9800 - val_accuracy: 0.8844 - val_loss: 0.9140
Epoch 9/10
1500/1500            2s 2ms/step -
accuracy: 0.8730 - loss: 0.9616 - val_accuracy: 0.8863 - val_loss: 0.9152
Epoch 10/10
1500/1500            2s 2ms/step -
accuracy: 0.8688 - loss: 0.9552 - val_accuracy: 0.8914 - val_loss: 0.8928
313/313             1s 1ms/step -
accuracy: 0.8723 - loss: 0.9497
L1 Regularization ( =0.01) - Test Loss: 0.8949846625328064, Test Accuracy:
0.8899000287055969
Epoch 1/10
1500/1500            4s 2ms/step -
accuracy: 0.1553 - loss: 39.5955 - val_accuracy: 0.1060 - val_loss: 3.5277
Epoch 2/10
1500/1500            2s 2ms/step -
accuracy: 0.1411 - loss: 3.5334 - val_accuracy: 0.2495 - val_loss: 3.4474
Epoch 3/10
1500/1500            2s 2ms/step -
accuracy: 0.3707 - loss: 3.3586 - val_accuracy: 0.5738 - val_loss: 3.0696
Epoch 4/10
1500/1500            2s 1ms/step -
accuracy: 0.5919 - loss: 3.0481 - val_accuracy: 0.6593 - val_loss: 2.9098
Epoch 5/10
1500/1500            2s 2ms/step -
accuracy: 0.6377 - loss: 2.9204 - val_accuracy: 0.6692 - val_loss: 2.8237
Epoch 6/10
1500/1500            2s 1ms/step -
accuracy: 0.6642 - loss: 2.8478 - val_accuracy: 0.6817 - val_loss: 2.7863
Epoch 7/10
1500/1500            2s 2ms/step -
accuracy: 0.6850 - loss: 2.8021 - val_accuracy: 0.7344 - val_loss: 2.7227
Epoch 8/10
1500/1500            2s 2ms/step -
accuracy: 0.7270 - loss: 2.7291 - val_accuracy: 0.7567 - val_loss: 2.6623
Epoch 9/10
1500/1500            2s 2ms/step -
accuracy: 0.7444 - loss: 2.6783 - val_accuracy: 0.7399 - val_loss: 2.6677
Epoch 10/10
1500/1500            2s 2ms/step -
accuracy: 0.7588 - loss: 2.6449 - val_accuracy: 0.7810 - val_loss: 2.5668
313/313             0s 1ms/step -
accuracy: 0.7538 - loss: 2.6120
L1 Regularization ( =0.1) - Test Loss: 2.5592005252838135, Test Accuracy:
0.7799000144004822
Epoch 1/10
1500/1500            4s 2ms/step -
accuracy: 0.1217 - loss: 187.7072 - val_accuracy: 0.1060 - val_loss: 8.2536
```

```
Epoch 2/10
1500/1500            3s 2ms/step -
accuracy: 0.1135 - loss: 8.2745 - val_accuracy: 0.1060 - val_loss: 8.2684
Epoch 3/10
1500/1500            2s 2ms/step -
accuracy: 0.1144 - loss: 8.2383 - val_accuracy: 0.1060 - val_loss: 8.2036
Epoch 4/10
1500/1500            2s 1ms/step -
accuracy: 0.1115 - loss: 8.2215 - val_accuracy: 0.1060 - val_loss: 8.2159
Epoch 5/10
1500/1500            2s 2ms/step -
accuracy: 0.1161 - loss: 8.2033 - val_accuracy: 0.1060 - val_loss: 8.1731
Epoch 6/10
1500/1500            2s 1ms/step -
accuracy: 0.1124 - loss: 8.1903 - val_accuracy: 0.1060 - val_loss: 8.1770
Epoch 7/10
1500/1500            2s 1ms/step -
accuracy: 0.1146 - loss: 8.1864 - val_accuracy: 0.1060 - val_loss: 8.1811
Epoch 8/10
1500/1500            2s 1ms/step -
accuracy: 0.1140 - loss: 8.1761 - val_accuracy: 0.1060 - val_loss: 8.1483
Epoch 9/10
1500/1500            2s 1ms/step -
accuracy: 0.1152 - loss: 8.1664 - val_accuracy: 0.1060 - val_loss: 8.1697
Epoch 10/10
1500/1500            2s 1ms/step -
accuracy: 0.1151 - loss: 8.1707 - val_accuracy: 0.1060 - val_loss: 8.1778
313/313            0s 923us/step -
accuracy: 0.1160 - loss: 8.1765
L1 Regularization ( =0.5) - Test Loss: 8.176519393920898, Test Accuracy:
0.11349999904632568
```

3 Step 3: Dropout

1) Introduce Dropout Layers:

[8]:
```python
from tensorflow.keras.layers import Dropout

def create_model_with_dropout(dropout_rate):
    model = Sequential()
    model.add(Flatten(input_shape=(28, 28)))
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(dropout_rate))
    model.add(Dense(10, activation='softmax'))
    return model
```

2) Train and Evaluate Models with Different Dropout Rates:

```
[9]: dropout_rates = [0.2, 0.5]
     for rate in dropout_rates:
         model = create_model_with_dropout(rate)
         model.compile(optimizer='adam', loss='categorical_crossentropy',␣
      ↪metrics=['accuracy'])
         model.fit(x_train, y_train, epochs=10, batch_size=32, validation_split=0.2)
         loss, accuracy = model.evaluate(x_test, y_test)
         print(f'Dropout Rate {rate} - Test Loss: {loss}, Test Accuracy: {accuracy}')
```

```
Epoch 1/10
1500/1500           4s 2ms/step -
accuracy: 0.8480 - loss: 0.5219 - val_accuracy: 0.9563 - val_loss: 0.1546
Epoch 2/10
1500/1500           3s 2ms/step -
accuracy: 0.9523 - loss: 0.1591 - val_accuracy: 0.9653 - val_loss: 0.1158
Epoch 3/10
1500/1500           2s 1ms/step -
accuracy: 0.9636 - loss: 0.1208 - val_accuracy: 0.9691 - val_loss: 0.0997
Epoch 4/10
1500/1500           2s 1ms/step -
accuracy: 0.9717 - loss: 0.0923 - val_accuracy: 0.9715 - val_loss: 0.0919
Epoch 5/10
1500/1500           2s 2ms/step -
accuracy: 0.9750 - loss: 0.0815 - val_accuracy: 0.9737 - val_loss: 0.0884
Epoch 6/10
1500/1500           2s 1ms/step -
accuracy: 0.9777 - loss: 0.0726 - val_accuracy: 0.9749 - val_loss: 0.0872
Epoch 7/10
1500/1500           2s 2ms/step -
accuracy: 0.9801 - loss: 0.0611 - val_accuracy: 0.9771 - val_loss: 0.0783
Epoch 8/10
1500/1500           2s 2ms/step -
accuracy: 0.9820 - loss: 0.0555 - val_accuracy: 0.9755 - val_loss: 0.0815
Epoch 9/10
1500/1500           2s 1ms/step -
accuracy: 0.9844 - loss: 0.0470 - val_accuracy: 0.9769 - val_loss: 0.0771
Epoch 10/10
1500/1500           2s 2ms/step -
accuracy: 0.9854 - loss: 0.0452 - val_accuracy: 0.9755 - val_loss: 0.0867
313/313           0s 1ms/step -
accuracy: 0.9723 - loss: 0.0938
Dropout Rate 0.2 - Test Loss: 0.07935527712106705, Test Accuracy:
0.9771000146865845
Epoch 1/10
1500/1500           3s 2ms/step -
accuracy: 0.7944 - loss: 0.6584 - val_accuracy: 0.9480 - val_loss: 0.1797
Epoch 2/10
1500/1500           3s 2ms/step -
```

```
accuracy: 0.9284 - loss: 0.2476 - val_accuracy: 0.9622 - val_loss: 0.1325
Epoch 3/10
1500/1500              2s 2ms/step -
accuracy: 0.9386 - loss: 0.2028 - val_accuracy: 0.9653 - val_loss: 0.1202
Epoch 4/10
1500/1500              2s 1ms/step -
accuracy: 0.9503 - loss: 0.1669 - val_accuracy: 0.9699 - val_loss: 0.1031
Epoch 5/10
1500/1500              3s 2ms/step -
accuracy: 0.9538 - loss: 0.1532 - val_accuracy: 0.9726 - val_loss: 0.0953
Epoch 6/10
1500/1500              2s 1ms/step -
accuracy: 0.9550 - loss: 0.1429 - val_accuracy: 0.9720 - val_loss: 0.0958
Epoch 7/10
1500/1500              2s 1ms/step -
accuracy: 0.9582 - loss: 0.1321 - val_accuracy: 0.9736 - val_loss: 0.0909
Epoch 8/10
1500/1500              3s 2ms/step -
accuracy: 0.9609 - loss: 0.1292 - val_accuracy: 0.9742 - val_loss: 0.0990
Epoch 9/10
1500/1500              3s 2ms/step -
accuracy: 0.9631 - loss: 0.1200 - val_accuracy: 0.9745 - val_loss: 0.0880
Epoch 10/10
1500/1500              2s 1ms/step -
accuracy: 0.9638 - loss: 0.1176 - val_accuracy: 0.9749 - val_loss: 0.0883
313/313              0s 1ms/step -
accuracy: 0.9731 - loss: 0.0976
Dropout Rate 0.5 - Test Loss: 0.08266211301088333, Test Accuracy:
0.9768999814987183
```

4 Step 4: Early Stopping

1) Implement Early Stopping:

[10]:
```python
from tensorflow.keras.callbacks import EarlyStopping

early_stopping = EarlyStopping(monitor='val_loss', patience=3, min_delta=0.001)

model = create_model_with_dropout(0.5)  # Example with dropout
model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
history = model.fit(x_train, y_train, epochs=10, batch_size=32,␣
 ↪validation_split=0.2, callbacks=[early_stopping])
loss, accuracy = model.evaluate(x_test, y_test)
print(f'Early Stopping - Test Loss: {loss}, Test Accuracy: {accuracy}')
```

```
Epoch 1/10
1500/1500              4s 2ms/step -
```

```
accuracy: 0.7974 - loss: 0.6569 - val_accuracy: 0.9492 - val_loss: 0.1800
Epoch 2/10
1500/1500              2s 2ms/step -
accuracy: 0.9265 - loss: 0.2484 - val_accuracy: 0.9601 - val_loss: 0.1390
Epoch 3/10
1500/1500              2s 1ms/step -
accuracy: 0.9393 - loss: 0.2038 - val_accuracy: 0.9652 - val_loss: 0.1193
Epoch 4/10
1500/1500              2s 1ms/step -
accuracy: 0.9470 - loss: 0.1707 - val_accuracy: 0.9670 - val_loss: 0.1113
Epoch 5/10
1500/1500              2s 1ms/step -
accuracy: 0.9513 - loss: 0.1597 - val_accuracy: 0.9673 - val_loss: 0.1073
Epoch 6/10
1500/1500              2s 2ms/step -
accuracy: 0.9540 - loss: 0.1475 - val_accuracy: 0.9716 - val_loss: 0.0993
Epoch 7/10
1500/1500              2s 2ms/step -
accuracy: 0.9565 - loss: 0.1360 - val_accuracy: 0.9743 - val_loss: 0.0945
Epoch 8/10
1500/1500              2s 1ms/step -
accuracy: 0.9605 - loss: 0.1274 - val_accuracy: 0.9731 - val_loss: 0.1001
Epoch 9/10
1500/1500              2s 1ms/step -
accuracy: 0.9599 - loss: 0.1222 - val_accuracy: 0.9754 - val_loss: 0.0898
Epoch 10/10
1500/1500              2s 1ms/step -
accuracy: 0.9635 - loss: 0.1141 - val_accuracy: 0.9739 - val_loss: 0.0941
313/313              0s 1ms/step -
accuracy: 0.9696 - loss: 0.1063
Early Stopping - Test Loss: 0.08979681879281998, Test Accuracy:
0.9736999869346619
```

5 Step 5: Data Augmentation

1) Applying Data Augmentation

[11]:
```
# Reshape x_train to include the channel dimension
x_train = np.expand_dims(x_train, axis=-1)  # Shape will be (60000, 28, 28, 1)
x_test = np.expand_dims(x_test, axis=-1)    # Shape will be (10000, 28, 28, 1)
```

[12]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=False,
    fill_mode='nearest'
)

# Fit the data generator on the reshaped training data
datagen.fit(x_train)
```

2) Train Model Using Augmented Data:

[13]:
```
model = create_model_with_dropout(0.5)  # Example with dropout
model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
history = model.fit(datagen.flow(x_train, y_train, batch_size=32), epochs=10,␣
 ↪validation_data=(x_test, y_test))
loss, accuracy = model.evaluate(x_test, y_test)
print(f'Data Augmentation - Test Loss: {loss}, Test Accuracy: {accuracy}')
```

```
c:\Users\eakes\AppData\Local\Programs\Python\Python311\Lib\site-
packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121:
UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in
its constructor. `**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will be
ignored.
  self._warn_if_super_not_called()
```

```
Epoch 1/10
1875/1875             16s 8ms/step -
accuracy: 0.6148 - loss: 1.1750 - val_accuracy: 0.9390 - val_loss: 0.2318
Epoch 2/10
1875/1875             12s 6ms/step -
accuracy: 0.8184 - loss: 0.5775 - val_accuracy: 0.9559 - val_loss: 0.1625
Epoch 3/10
1875/1875             11s 6ms/step -
accuracy: 0.8525 - loss: 0.4881 - val_accuracy: 0.9620 - val_loss: 0.1328
Epoch 4/10
1875/1875             11s 6ms/step -
accuracy: 0.8670 - loss: 0.4368 - val_accuracy: 0.9688 - val_loss: 0.1161
Epoch 5/10
1875/1875             12s 6ms/step -
accuracy: 0.8705 - loss: 0.4263 - val_accuracy: 0.9677 - val_loss: 0.1074
Epoch 6/10
1875/1875             13s 7ms/step -
accuracy: 0.8765 - loss: 0.4075 - val_accuracy: 0.9695 - val_loss: 0.1059
Epoch 7/10
1875/1875             15s 8ms/step -
accuracy: 0.8819 - loss: 0.3873 - val_accuracy: 0.9688 - val_loss: 0.1065
Epoch 8/10
```

```
1875/1875            13s 7ms/step -
accuracy: 0.8833 - loss: 0.3901 - val_accuracy: 0.9721 - val_loss: 0.1006
Epoch 9/10
1875/1875            12s 7ms/step -
accuracy: 0.8843 - loss: 0.3860 - val_accuracy: 0.9730 - val_loss: 0.0977
Epoch 10/10
1875/1875            12s 7ms/step -
accuracy: 0.8867 - loss: 0.3725 - val_accuracy: 0.9716 - val_loss: 0.0955
313/313             0s 1ms/step -
accuracy: 0.9674 - loss: 0.1100
Data Augmentation - Test Loss: 0.09545360505580902, Test Accuracy:
0.9715999960899353
```

6 Step 6: Combined Regularization

1) Combine Regularization Techniques:

[14]:
```python
def create_combined_model():
    model = Sequential()
    model.add(Flatten(input_shape=(28, 28)))
    model.add(Dense(128, activation='relu', kernel_regularizer=l2(0.01)))
    model.add(Dropout(0.5))
    model.add(Dense(10, activation='softmax'))
    return model
```

2) Train and Evaluate Combined Model:

[15]:
```python
model = create_combined_model()
model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
history = model.fit(datagen.flow(x_train, y_train, batch_size=32), epochs=10,␣
 ↪validation_data=(x_test, y_test))
loss, accuracy = model.evaluate(x_test, y_test)
print(f'Combined Regularization - Test Loss: {loss}, Test Accuracy: {accuracy}')
```

```
Epoch 1/10

c:\Users\eakes\AppData\Local\Programs\Python\Python311\Lib\site-
packages\keras\src\layers\reshaping\flatten.py:37: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
c:\Users\eakes\AppData\Local\Programs\Python\Python311\Lib\site-
packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121:
UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in
its constructor. `**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will be
ignored.
  self._warn_if_super_not_called()
```

```
1875/1875            13s 7ms/step -
accuracy: 0.5874 - loss: 1.8306 - val_accuracy: 0.9203 - val_loss: 0.6689
Epoch 2/10
1875/1875            10s 5ms/step -
accuracy: 0.7530 - loss: 1.0703 - val_accuracy: 0.9249 - val_loss: 0.5851
Epoch 3/10
1875/1875            11s 6ms/step -
accuracy: 0.7741 - loss: 0.9911 - val_accuracy: 0.9379 - val_loss: 0.5503
Epoch 4/10
1875/1875            11s 6ms/step -
accuracy: 0.7863 - loss: 0.9474 - val_accuracy: 0.9371 - val_loss: 0.5427
Epoch 5/10
1875/1875            11s 6ms/step -
accuracy: 0.7915 - loss: 0.9316 - val_accuracy: 0.9375 - val_loss: 0.5230
Epoch 6/10
1875/1875            11s 6ms/step -
accuracy: 0.7979 - loss: 0.9183 - val_accuracy: 0.9426 - val_loss: 0.5150
Epoch 7/10
1875/1875            11s 6ms/step -
accuracy: 0.7966 - loss: 0.9118 - val_accuracy: 0.9421 - val_loss: 0.5041
Epoch 8/10
1875/1875            11s 6ms/step -
accuracy: 0.7994 - loss: 0.9031 - val_accuracy: 0.9470 - val_loss: 0.4791
Epoch 9/10
1875/1875            11s 6ms/step -
accuracy: 0.8059 - loss: 0.8894 - val_accuracy: 0.9426 - val_loss: 0.4970
Epoch 10/10
1875/1875            11s 6ms/step -
accuracy: 0.8005 - loss: 0.8933 - val_accuracy: 0.9404 - val_loss: 0.5046
313/313            0s 911us/step -
accuracy: 0.9300 - loss: 0.5335
Combined Regularization - Test Loss: 0.504585325717926, Test Accuracy:
0.9404000043869019
```