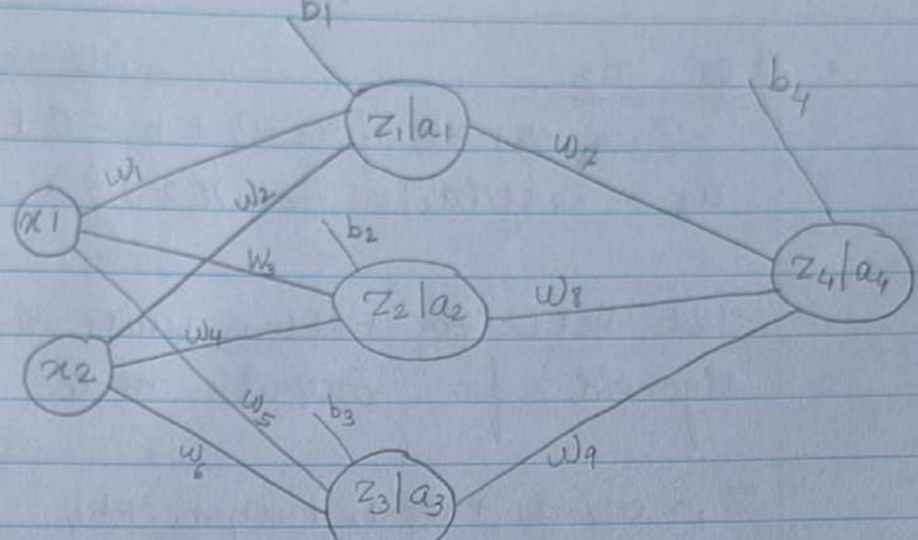
Problem 1

a)



Given :-

- Learning Rate $(\alpha) = 0.1$
- Inputs $x_1 = 0$, $x_2 = 1$
- Output $y = 1$
- All weights & biases $= 1$

i.e. $w_1 = w_2 = w_3 = w_4 = w_5 = w_6 = w_7 = w_8 = w_9 = b_1 =$
$b_2 = b_3 = b_4 = 1$

Step I Forward Pass :- Using ReLU $\Rightarrow f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$ & $f(x) = \max(0, x)$

- For n_1

$Z_1 = x_1 w_1 + x_2 w_2 + b_1 = 0 \cdot 1 + 1 \cdot 1 + 1 = 2$

$a_1 = \text{ReLU}(z_1) = \text{ReLU}(2) = \underline{2}$ as it is +ve & $f(2) = \max(0, 2) = 2$

- For n_2

$Z_2 = x_1 w_3 + x_2 w_4 + b_2 = 0 \cdot 1 + 1 \cdot 1 + 1 = 2$

$a_2 = \text{ReLU}(z_2) = \text{ReLU}(2) = \underline{2}$

- For x_3

$$z_3 = x_1 \cdot w_5 + x_2 w_6 + b_3 = 0.1 + 1.1 + 1 = 2$$

$$a_3 = ReLU(a_3) = ReLU(2) = 2 \quad [\text{as } ReLU(x) = max(0, x)]$$

Used ReLU for hidden units and now using sigmoid for output node

$$z_4 = w_7 \cdot a_1 + w_8 \cdot a_2 + w_9 \cdot a_3 + b_4$$
$$= 1.2 + 1.2 + 1.2 + 1$$
$$= 7$$

Using sigmoid Activation $\boxed{\sigma(x) = \frac{1}{1+e^{-x}}}$

- $$\text{output } (\hat{y}) = \sigma(z_4) = \frac{1}{1+e^{-7}} \quad [\text{Using Sigmoid Activ}^n]$$

$$= \frac{1}{1+0.000911}$$

$$= \frac{1}{1.000911}$$

$$= 0.9990889488$$

$$\hat{y} \approx 0.9991$$

Step II Error

$$\text{Loss Function} = MSE = \frac{1}{2}(y - \hat{y})^2$$

$$MSE = \frac{1}{2}(1 - 0.9991)^2 = \frac{1}{2}(0.0009)^2$$

$$= \cancel{4.05A0} \; 4.05 \times 10^{-7}$$

Step III Backpropagation

$$L = \frac{1}{2}(y-\hat{y})^2$$

derivative w.r.t $\hat{y}$

$$\frac{\partial L}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}}\left(\frac{1}{2}(y-\hat{y})^2\right) = \frac{1}{2}\cdot 2(y-\hat{y})\cdot\frac{\partial}{\partial \hat{y}}(y-\hat{y})$$

with chain Rule

$$\frac{\partial L}{\partial \hat{y}} = -(y-\hat{y}) = \underline{\underline{\hat{y}-y}}$$

where we have y (ground truth) $= 1$ & $\hat{y} = 0.9991$

$\therefore$ we get,

$$\frac{\partial L}{\partial \hat{y}} = 0.9991 - 1 = \underline{\underline{-0.0009}}$$

- Gradient of Loss w.r.t. z_4

$$\frac{\partial L}{\partial z_4} = \frac{\partial L}{\partial \hat{y}}\cdot\sigma'(z_4)$$

where

$$\sigma'(z_4) = \sigma(z_4)\cdot(1-\sigma(z_4)) = 0.9991\cdot(1-0.9991)$$
$$\approx 0.0009$$

$$\therefore \frac{\partial L}{\partial z_4} = -0.0009\times 0.0009 = -8.1\times 10^{-7}$$

- Gradient wrt w_7, w_8, w_9 are

- For w_7:

$$\frac{\partial L}{\partial w_7} = \frac{\partial L}{\partial wz_4} \cdot a_1 = -8.1 \times 10^{-7} \times 2 = -1.62 \times 10^{-6}$$

- For w_8:

$$\frac{\partial L}{\partial w_8} = \frac{\partial L}{\partial wz_4} \cdot a_2 = -8.1 \times 10^{-7} \times 2 = -1.62 \times 10^{-6}$$

- For w_9:

$$\frac{\partial L}{\partial w_9} = \frac{\partial L}{\partial z_4} \cdot a_3 = -8.1 \times 10^{-7} \times 2 = -1.6 \times 10^{-6}$$

To compute updated weights using formula

$$\boxed{w' = w - \alpha \frac{\partial L}{\partial w}}$$

we have $\alpha = 0.1$

$\therefore$ for $\underline{w_7}$:

$$w_7' = w_7 - \alpha \frac{\partial L}{\partial w_7} = 1 - 0.1(-1.62 \times 10^{-6})$$

$$= 1 + 1.62 \times 10^{-7}$$

$$= 1.000000162$$

- For w_8:

$$w_8' = w_8 - \alpha \frac{\partial L}{\partial w_8} = 1 - (0.1)(-1.62 \times 10^{-6})$$
$$= 1 + 1.62 \times 10^{-7}$$
$$= 1.000000162$$

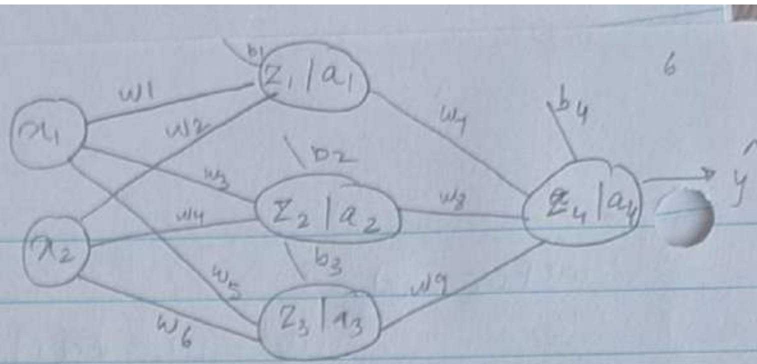
- For w_9:

$$w_9' = w_9 - \alpha \frac{\partial L}{\partial w_9} = 1 - 0.1(-1.62 \times 10^{-6})$$
$$= 1 + 1.62 \times 10^{-7}$$
$$= 1.000000162$$

Comparison:

As all the updated values i.e. w_7', w_8' and w_9' are same i.e. 1.62×10^{-7}

This is the root cause of symmetry-breaking issue. Symmetry breaking in neural networks refers to situation where weights or neurons in network are initialized or updated where the end up performing same function hindering learning process.



(b)

→ Given :-

- weights ⇒ $w_1 = w_3 = w_5 = -0.5$

  $w_2 = w_4 = w_6 = 0.25$

  $w_7 = 1$, $w_8 = -1$, $w_9 = 0$

- biases ⇒ $b_1 = b_2 = b_3 = b_4 = 0.1$.
- inputs ⇒ $x_1 = 1$, $x_2 = 1$
- Learning Rate $(\alpha) = 0.1$
- Using ReLU for hidden layers & sigmoid for output node
- As my student ID is 017516785, here the last digit is 5 (odd) hence computing w_7, w_1

– For n_1 =

$z_1 = w_1 x_1 + w_2 x_2 + b_1$

$= -0.5 \times 1 + 0.25 \times 1 + 0.1$

$= -0.5 + 0.25 + 0.1$

$= -0.15$

$\therefore a_1 = ReLU(z_1) = ReLU(-0.15) = 0$

as $ReLU(x) = \max(0, x)$

$\therefore \boxed{a_1 = 0}$

- For n_2 =>

$z_2 = w_3 x_1 + w_4 x_2 + b_2$
$= -0.5 \cdot 1 + 0.25 \cdot 1 + 0.1$
$= -0.5 + 0.25 + 0.1$

$z_2 = -0.15$

$a_2 = \text{ReLU}(z_2) = \text{ReLU}(-0.15) = 0$

- For n_3 =>

$z_3 = w_5 x_1 + w_6 x_2 + b_3$
$= -0.5 \cdot 1 + 0.25 \cdot 1 + 0.1$

$z_3 = -0.15$

$a_3 = \text{ReLU}(z_3) = \text{ReLU}(-0.15) = 0$

- For $\hat{y}$

$z_4 = w_7 a_1 + w_8 a_2 + w_9 a_3 + b_4$
$= 1 \cdot 0 + (-1) \cdot 0 + 0 \cdot 0 + 0.1$
$= 0 + 0 + 0 + 0.1$

$z_4 = 0.1$

using sigmoid activation

$$\hat{y} = \sigma(z_4) = \frac{1}{1 + e^{-0.1}} = \frac{1}{1 + 0.9048} = \frac{1}{1.9048}$$
$$\approx 0.52497$$

- Loss

MSE => $L = \frac{1}{2}(y-\hat{y})^2 = \frac{1}{2}(1-0.52497)^2$

$= \frac{1}{2}(0.22565)$

≈ 0.112826

- Backpropagation

- Gradient Loss wrt $\hat{y}$

$$\frac{\partial L}{\partial \hat{y}} = \hat{y} - y = 0.52497 - 1 = -0.47503$$

- Gradient Loss wrt z_4

using ~~derivative~~ sigmoid $\sigma(z) = \frac{1}{1+e^{-z}}$

$\therefore$ derivative of $\sigma(z) \Rightarrow$

$$\sigma'(z) = \sigma(z) \cdot (1 - \sigma(z))$$

$$\therefore \sigma'(z_4) = \sigma(z_4) \cdot (1 - \sigma(z_4)) = 0.52497 \cdot (1 - 0.52497)$$

$$\therefore \sigma'(z_4) \approx 0.2494$$

$$\therefore \frac{\partial L}{\partial z_4} = \frac{\partial L}{\partial \hat{y}} \cdot \sigma'(z_4) = -0.4751 \times 0.2494$$

$$\approx -0.1181$$

- For w_7

$$\frac{\partial L}{\partial w_7} = \frac{\partial L}{\partial z_4} \cdot a_1 = -0.1181 \cdot 0 = 0$$

- For w_1

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial z_4} \cdot x_1 = -0.1181 \cdot 1 = -0.1181$$

Hence ~~by~~ using learning Rate $\alpha = 0.1$ we get,

~~$w_7' = w$~~ Formula $\boxed{w' = w - \alpha \frac{\partial L}{\partial w}}$

For w_7

- $$w_7' = w_7 - \alpha \frac{\partial L}{\partial w_7} = 1 - 0.1 \times 0 = 1$$

- For w_1

$$w_1' = w_1 - \alpha \frac{\partial L}{\partial w_1} = -0.5 - 0.1 \times (-0.1181)$$

$$= -0.5 + 0.01181$$

$$= -0.48819$$

$\therefore$ For last id odd

$\underline{w_7' = 1}$ and $\underline{w_1' = -0.48819}$

gov98xbx2

February 24, 2025

CODING!

Problem 2-4: We will develop Artificial Neural Networks using MNIST digit data, you can directly download the data using https://keras.io/api/datasets/mnist/. The dataset contains 10 classes where each of the image sizes is (28×28). Train for minimum number of = 100; You should split the training data into training and validation sets, and for training the model use these datasets. Test data should be kept separated and used only for evaluation purpose. Your ANN must contain minimum of 2 hidden layers. Apply early stopping criteria based on validation loss with patience 3 and with restoring best weights = true. You may use any regularizes to avoid overfitting.

Problem 2 (8 pts): You should select last two digits of your student ID – meaning that if your student id is 006000104, then you should select 0 and 4 for developing the binary classification model. If both last digit is identical, then select first and last digit. For this task, you must prepare (filter) your data at first to convert the multiclass classification into a binary classification system.

[177]:
```
# Importing necessary libraries
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report
import warnings

# ignore warnings
warnings.filterwarnings("ignore")
```

[178]:
```
# to load the MNIST dataset (28x28)
(X_train_full, Y_train_full), (X_test_full, Y_test_full) = mnist.load_data()
```

[179]:
```python
# to check original shape
print("Original shape of training set:", X_train_full.shape)
print("Original shape of test set:", X_test_full.shape)
```

```
Original shape of training set: (60000, 28, 28)
Original shape of test set: (10000, 28, 28)
```

[180]:
```python
# to flatten images to 1D vectors [28x28=784]
X_train_full = X_train_full.reshape(X_train_full.shape[0], 784).
↪astype('float32')
X_test_full = X_test_full.reshape(X_test_full.shape[0], 784).astype('float32')
```

[181]:
```python
# new shape
print("Flattened shape of training set:", X_train_full.shape)
print("Flattened shape of test set:", X_test_full.shape)
```

```
Flattened shape of training set: (60000, 784)
Flattened shape of test set: (10000, 784)
```

[182]:
```python
# min and max values before normalization
print("Min pixel value (train set):", np.min(X_train_full))
print("Max pixel value (train set):", np.max(X_train_full))
print("Min pixel value (test set):", np.min(X_test_full))
print("Max pixel value (test set):", np.max(X_test_full))
```

```
Min pixel value (train set): 0.0
Max pixel value (train set): 255.0
Min pixel value (test set): 0.0
Max pixel value (test set): 255.0
```

[183]:
```python
# normalize pixel values to the range [0,1]
X_train_full = X_train_full / 255.0
X_test_full = X_test_full / 255.0
```

[184]:
```python
# min and max values after normalization
print("Min pixel value (train set):", np.min(X_train_full))
print("Max pixel value (train set):", np.max(X_train_full))
print("Min pixel value (test set):", np.min(X_test_full))
print("Max pixel value (test set):", np.max(X_test_full))
```

```
Min pixel value (train set): 0.0
Max pixel value (train set): 1.0
Min pixel value (test set): 0.0
Max pixel value (test set): 1.0
```

[185]:
```python
#selecting 8 and 5 i.e the last two digits from my Student ID = 017516785
selected_digits = [8, 5]
train_filter = np.isin(Y_train_full, selected_digits)
```

```
test_filter = np.isin(Y_test_full, selected_digits)

# filtering
X_train = X_train_full[train_filter]
Y_train = Y_train_full[train_filter]
X_test = X_test_full[test_filter]
Y_test = Y_test_full[test_filter]
```

[186]:
```
# new dataset shape
print("Filtered training set shape:", X_train.shape)
print("Filtered test set shape:", X_test.shape)
```

```
Filtered training set shape: (11272, 784)
Filtered test set shape: (1866, 784)
```

[187]:
```
# here convert 5 as 0 , 8 as 1
Y_train = np.where(Y_train == 5, 0, 1)
Y_test = np.where(Y_test == 5, 0, 1)
```

[188]:
```
# to check unique values in Y_train and Y_test
print("Unique values in Y_train:", np.unique(Y_train))
print("Unique values in Y_test:", np.unique(Y_test))
```

```
Unique values in Y_train: [0 1]
Unique values in Y_test: [0 1]
```

2a) A. Build an ANN for binary classification. Evaluate your model on the test data. Construct a confusion matrix. Present learning curve (showing training loss and validation loss against number of epochs) and include some examples of your prediction. (3pts)

[189]:
```
# Splitting the training set into 80% training and 20% validation
X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size=0.
↪2, random_state=42, stratify=Y_train)

#  new dataset shapes
print("Training set shape:", X_train.shape)
print("Validation set shape:", X_val.shape)
print("Test set shape:", X_test.shape)
```

```
Training set shape: (9017, 784)
Validation set shape: (2255, 784)
Test set shape: (1866, 784)
```

[190]:
```
X_train1=X_train
Y_train1=Y_train
X_test1=X_test
Y_test1=Y_test
X_val1=X_val
```

```
Y_val1=Y_val
```

[191]:
```
# ANN model with 2 hidden layer and one output layer
model = Sequential([
    Dense(128, activation='relu', input_shape=(784,)),
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid')
])

# model compilation
model.compile(loss='binary_crossentropy', metrics=['accuracy'])
```

[192]:
```
# early stopping
early_stopping = EarlyStopping(monitor='val_loss', mode='min', patience=3,␣
 ↪restore_best_weights=True, verbose=1)
```

[193]:
```
model.summary()
```

```
Model: "sequential_50"


 Layer (type)                    Output Shape                          ␣
 ↪Param #

 dense_194 (Dense)               (None, 128)                           ␣
 ↪100,480

 dense_195 (Dense)               (None, 64)                              ␣
 ↪8,256

 dense_196 (Dense)               (None, 1)                                 ␣
 ↪ 65



 Total params: 108,801 (425.00 KB)


 Trainable params: 108,801 (425.00 KB)


 Non-trainable params: 0 (0.00 B)
```

[194]:
```
# model train
history = model.fit(
    X_train1, Y_train1,
    validation_data=(X_val1, Y_val1),
```

```
    epochs=100,
    batch_size=1000,
    callbacks=[early_stopping],
    verbose=1
)
```

```
Epoch 1/100
10/10            3s 118ms/step -
accuracy: 0.6726 - loss: 0.5494 - val_accuracy: 0.9370 - val_loss: 0.2417
Epoch 2/100
10/10            1s 67ms/step -
accuracy: 0.9442 - loss: 0.2101 - val_accuracy: 0.8905 - val_loss: 0.2731
Epoch 3/100
10/10            1s 74ms/step -
accuracy: 0.9383 - loss: 0.1816 - val_accuracy: 0.9707 - val_loss: 0.1051
Epoch 4/100
10/10            1s 86ms/step -
accuracy: 0.9724 - loss: 0.0966 - val_accuracy: 0.9344 - val_loss: 0.1721
Epoch 5/100
10/10            1s 88ms/step -
accuracy: 0.9521 - loss: 0.1248 - val_accuracy: 0.9627 - val_loss: 0.1049
Epoch 6/100
10/10            1s 97ms/step -
accuracy: 0.9677 - loss: 0.0917 - val_accuracy: 0.9596 - val_loss: 0.1140
Epoch 7/100
10/10            1s 67ms/step -
accuracy: 0.9760 - loss: 0.0748 - val_accuracy: 0.9738 - val_loss: 0.0762
Epoch 8/100
10/10            1s 70ms/step -
accuracy: 0.9796 - loss: 0.0599 - val_accuracy: 0.8652 - val_loss: 0.3313
Epoch 9/100
10/10            1s 79ms/step -
accuracy: 0.9481 - loss: 0.1241 - val_accuracy: 0.9836 - val_loss: 0.0506
Epoch 10/100
10/10            1s 59ms/step -
accuracy: 0.9872 - loss: 0.0404 - val_accuracy: 0.9809 - val_loss: 0.0537
Epoch 11/100
10/10            1s 77ms/step -
accuracy: 0.9834 - loss: 0.0500 - val_accuracy: 0.9854 - val_loss: 0.0456
Epoch 12/100
10/10            1s 60ms/step -
accuracy: 0.9875 - loss: 0.0381 - val_accuracy: 0.9792 - val_loss: 0.0585
Epoch 13/100
10/10            1s 50ms/step -
accuracy: 0.9843 - loss: 0.0447 - val_accuracy: 0.9871 - val_loss: 0.0392
Epoch 14/100
10/10            1s 57ms/step -
```

```
accuracy: 0.9912 - loss: 0.0276 - val_accuracy: 0.9814 - val_loss: 0.0522
Epoch 15/100
10/10            1s 59ms/step -
accuracy: 0.9912 - loss: 0.0297 - val_accuracy: 0.9880 - val_loss: 0.0345
Epoch 16/100
10/10            1s 64ms/step -
accuracy: 0.9917 - loss: 0.0261 - val_accuracy: 0.9867 - val_loss: 0.0345
Epoch 17/100
10/10            1s 68ms/step -
accuracy: 0.9933 - loss: 0.0223 - val_accuracy: 0.9885 - val_loss: 0.0326
Epoch 18/100
10/10            1s 118ms/step -
accuracy: 0.9875 - loss: 0.0385 - val_accuracy: 0.9907 - val_loss: 0.0304
Epoch 19/100
10/10            1s 73ms/step -
accuracy: 0.9938 - loss: 0.0188 - val_accuracy: 0.9889 - val_loss: 0.0325
Epoch 20/100
10/10            1s 28ms/step -
accuracy: 0.9905 - loss: 0.0268 - val_accuracy: 0.9902 - val_loss: 0.0304
Epoch 21/100
10/10            1s 27ms/step -
accuracy: 0.9959 - loss: 0.0147 - val_accuracy: 0.9907 - val_loss: 0.0310
Epoch 22/100
10/10            0s 23ms/step -
accuracy: 0.9892 - loss: 0.0293 - val_accuracy: 0.9898 - val_loss: 0.0306
Epoch 23/100
10/10            0s 30ms/step -
accuracy: 0.9964 - loss: 0.0128 - val_accuracy: 0.9477 - val_loss: 0.1541
Epoch 23: early stopping
Restoring model weights from the end of the best epoch: 20.
```

[195]:
```python
test_loss, test_accuracy = model.evaluate(X_test1, Y_test1)
print(f"Test Loss: {test_loss:.4f}")
print(f"Test Accuracy: {test_accuracy:.4f}")
```

```
59/59            0s 3ms/step -
accuracy: 0.9894 - loss: 0.0283
Test Loss: 0.0252
Test Accuracy: 0.9909
```

Learning Curves (Training & Validation Loss)

[196]:
```python
#  loss and accuracy data
train_loss = history.history['loss']
val_loss = history.history['val_loss']
train_acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
epochs = range(1, len(train_loss) + 1)
```

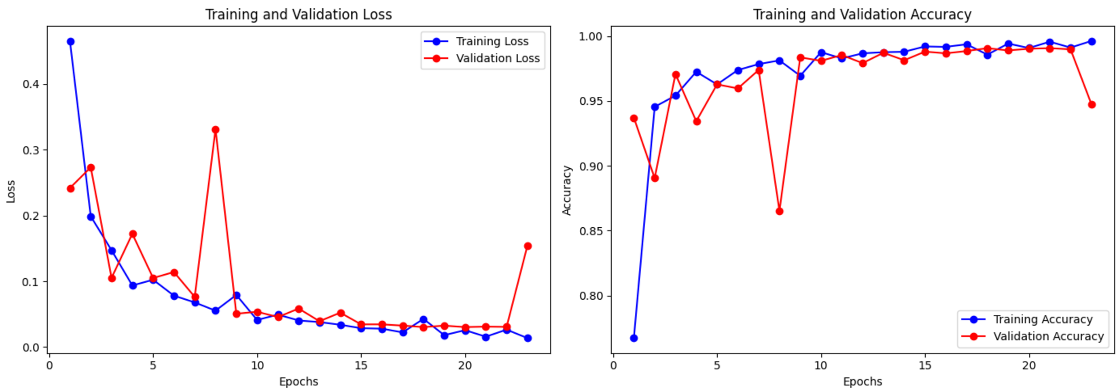
```python
print("Training Loss:", train_loss)
print("Validation Loss:", val_loss)
print("Training Accuracy:", train_acc)
print("Validation Accuracy:", val_acc)
```

```
Training Loss: [0.4651414752006531, 0.19865594804286957, 0.14709831774234772,
0.09359128773212433, 0.10222889482975006, 0.07810667902231216,
0.06783109158277512, 0.05527894943952560, 0.07918457686901093,
0.04114745184779167, 0.04929564520716667, 0.04039256274700165,
0.0379407629370689, 0.03386365622282028, 0.028625624254345894,
0.027960525825619698, 0.022256115451455116, 0.042767032980918884,
0.018059607595205307, 0.025622360408306122, 0.015745608136057854,
0.026451002806425095, 0.013609780929982662]
Validation Loss: [0.2416839301586151, 0.27312207221984863, 0.10508830100297928,
0.1720648556947708, 0.10490819811820984, 0.11402373015880585,
0.07620465010404587, 0.3313034474849701, 0.05062020942568779,
0.05365185812115669, 0.045562803745269775, 0.058458361774683,
0.03924311324954033, 0.052226755768060684, 0.03446607664227486,
0.034502990543842316, 0.032592177391052246, 0.030359461903572083,
0.032513637095689774, 0.03035561926662922, 0.031041506677865982,
0.03064480423927307, 0.15407925844192505]
Training Accuracy: [0.7671065926551819, 0.945436418056488, 0.9540867209434509,
0.9724963903427124, 0.9627370238304138, 0.9738272428512573, 0.978374183177948,
0.9812576174736023, 0.9693911671638489, 0.9875789880752563, 0.9828102588653564,
0.9866918325424194, 0.9875789880752563, 0.9879117012023926, 0.9920150637626648,
0.9916823506355286, 0.993678629398346, 0.9856936931610107, 0.9942331314086914,
0.9907951354980469, 0.9956748485565186, 0.9911278486251831, 0.996229350566864]
Validation Accuracy: [0.9370288252830505, 0.8904656171798706,
0.9707317352294922, 0.9343680739402771, 0.9627494215965271, 0.9596452116966248,
0.9738359451293945, 0.865188479423523, 0.9835920333862305, 0.980931282043457,
0.9853658676147461, 0.9791574478149414, 0.9871397018432617, 0.9813747406005859,
0.9880266189575195, 0.9866962432861328, 0.9884700775146484, 0.990687370300293,
0.9889135360717773, 0.9902439117431641, 0.990687370300293, 0.9898004531860352,
0.9476718306541443]
```

[197]:

```python
# two subplots
fig, ax = plt.subplots(1, 2, figsize=(14, 5))

# plot for training & validation Loss
ax[0].plot(epochs, train_loss, 'bo-', label='Training Loss')
ax[0].plot(epochs, val_loss, 'ro-', label='Validation Loss')
ax[0].set_title('Training and Validation Loss')
ax[0].set_xlabel('Epochs')
ax[0].set_ylabel('Loss')
ax[0].legend()
```

```
# plot for training & validation accuracy
ax[1].plot(epochs, train_acc, 'bo-', label='Training Accuracy')
ax[1].plot(epochs, val_acc, 'ro-', label='Validation Accuracy')
ax[1].set_title('Training and Validation Accuracy')
ax[1].set_xlabel('Epochs')
ax[1].set_ylabel('Accuracy')
ax[1].legend()

plt.tight_layout()
plt.show()
```



Loss Graph:

- The training loss starts high and gradually decreases which mean that the model is learning well.
- The validation loss shows somewhat a similar decreasing trend, telling that the model is generalizing well to unseen data.
- The loss stabilizes after approx 10 epochs which meanss that training beyond this point has little improvement.

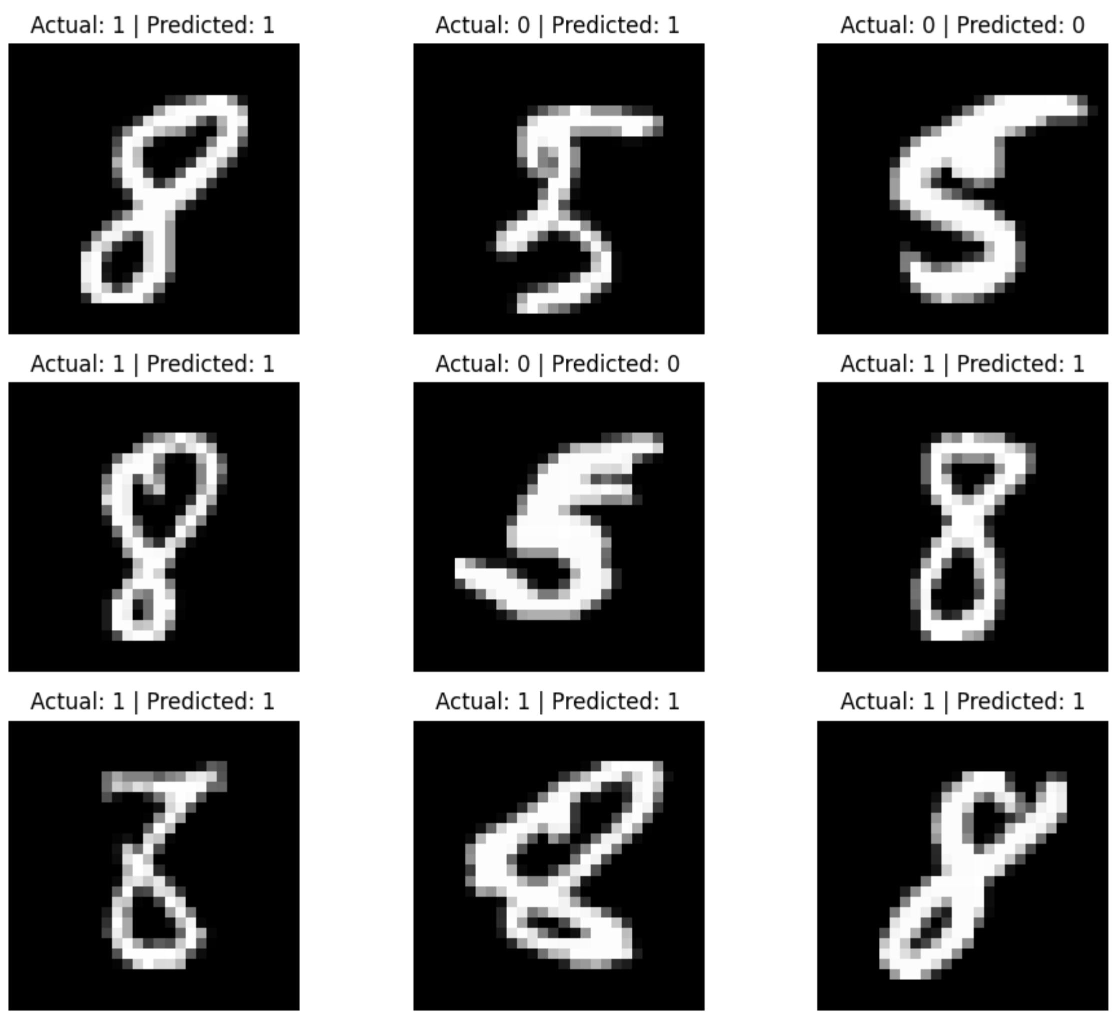
Accuracy Graph: - Here the training accuracy increases very sharply in the first few epochs and this indicates rapid learning. - Also the validation accuracy closely follows the training accuracy which means that there there is no significant overfitting. - Both training and validation accuracy reach nearly 98 to 99% this suggests that the model performs well on both seen and unseen data.

1 Predictions

[198]:
```
import random

# random test images
num_samples = 9
indices = random.sample(range(len(X_test1)), num_samples)
sample_images = X_test1[indices].reshape(-1, 28, 28)
sample_labels = Y_test1[indices]
```

```
sample_predictions = (model.predict(X_test1[indices]) > 0.5).astype("int32").
↪flatten()

# images with predictions
plt.figure(figsize=(10, 8))
for i in range(num_samples):
    plt.subplot(3, 3, i + 1)
    plt.imshow(sample_images[i], cmap='gray')
    plt.title(f"Actual: {sample_labels[i]} | Predicted:␣
↪{sample_predictions[i]}")
    plt.axis("off")

plt.tight_layout()
plt.show()
```

```
1/1              0s 97ms/step
```



Predictions on Randomly Selected Images to test: - From above image results we can see that he model has correctly identified digits the didgits of 8 and 5. - It correctly classifies both 5s and 8s as per their respective labels.

Interpretation: - The Model was bulit with two hidden layers and one output layer. The first hidden layer has 128 units, the second hidden layer has 64 units. Each of these hidden layers uses the ReLU activation function, which is one of the efficient activation function in neural networks.
- The model is well-trained and achieves high accuracy in both training and validation.
- It has a high test accuracy percentage of approx 99% and a low test loss of 0.0252 - No signs of overfitting is obseerved as the validation loss remains low indicating the model is highly effective in distinguishing between the digits 5 and 8.

[198]:

```
```

B. Build ANNs for binary classification using combinations of weight initializers (Normal, He, and Xavier) and activation functions (ReLU, Sigmoid, and tanh). You may use early stopping callback function. Construct confusion matrices and show learning curves for each combination. Create a table, showing combinations, and accuracy. Now, from your experimental results – write a comparative analysis on – impact of different combinations of initializers and activation functions in terms of performance and learning curves. (5 pts)

[199]:

```
from tensorflow.keras import initializers
from sklearn.metrics import confusion_matrix, precision_score, recall_score,␣
↪f1_score
```

[200]:

```
# to load the MNIST dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# filtering out classes for '8' and '5'
is_eight_or_five_train = (y_train == 8) | (y_train == 5)
is_eight_or_five_test = (y_test == 8) | (y_test == 5)
X_train, y_train = X_train[is_eight_or_five_train],␣
↪y_train[is_eight_or_five_train]
X_test, y_test = X_test[is_eight_or_five_test], y_test[is_eight_or_five_test]

# to flatten and normalize
X_train = X_train.reshape((-1, 784)).astype('float32') / 255
X_test = X_test.reshape((-1, 784)).astype('float32') / 255

# convert to 1d
y_train = (y_train == 8).astype(int)
y_test = (y_test == 8).astype(int)
```

[201]:

```
def build_and_train_model(initializer, activation_function):
    model = Sequential([
        Dense(128, input_shape=(784,), activation=activation_function,␣
↪kernel_initializer=initializer),
```

```
        Dense(64, activation=activation_function,␣
↪kernel_initializer=initializer),
        Dense(32, activation=activation_function,␣
↪kernel_initializer=initializer),
        Dense(1, activation='sigmoid')
    ])
    model.compile(optimizer='adam', loss='binary_crossentropy',␣
↪metrics=['accuracy'])
    early_stopping = EarlyStopping(monitor='val_loss', patience=3,␣
↪restore_best_weights=True)
    history = model.fit(X_train, y_train, validation_split=0.2, epochs=100,␣
↪callbacks=[early_stopping], verbose=0)
    return model, history
```

[202]:

```
model.summary()
```

```
Model: "sequential_50"


 Layer (type)                       Output Shape                        ␣
↪Param #

 dense_194 (Dense)                  (None, 128)                         ␣
↪100,480

 dense_195 (Dense)                  (None, 64)                            ␣
↪8,256

 dense_196 (Dense)                  (None, 1)                               ␣
↪ 65


 Total params: 217,604 (850.02 KB)

 Trainable params: 108,801 (425.00 KB)

 Non-trainable params: 0 (0.00 B)

 Optimizer params: 108,803 (425.02 KB)
```

[203]:

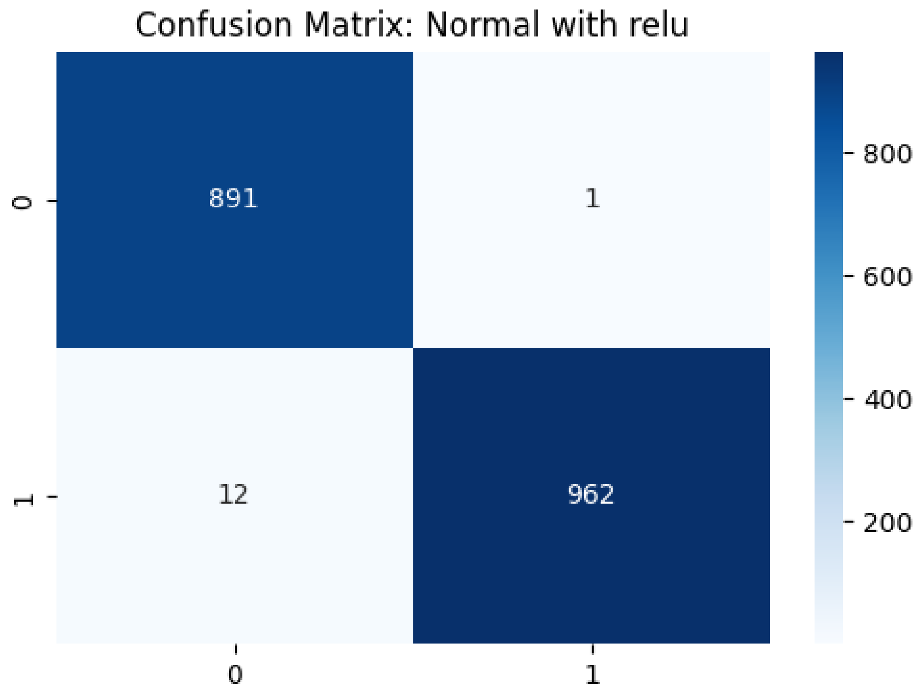
```
initializers_dict = {
    'Normal': initializers.RandomNormal(mean=0.0, stddev=0.05, seed=42),
    'He': initializers.HeNormal(seed=42),
```

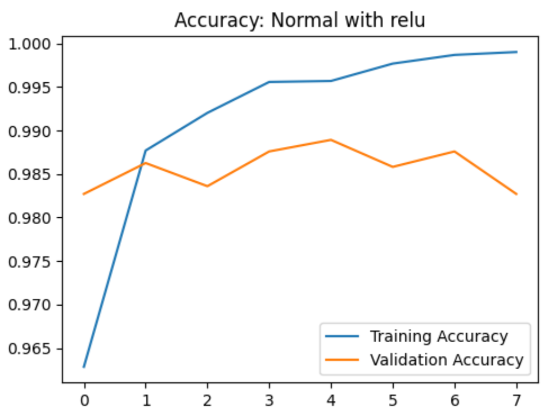
```
    'Xavier': initializers.GlorotUniform(seed=42)
}

activations = ['relu', 'sigmoid', 'tanh']
```

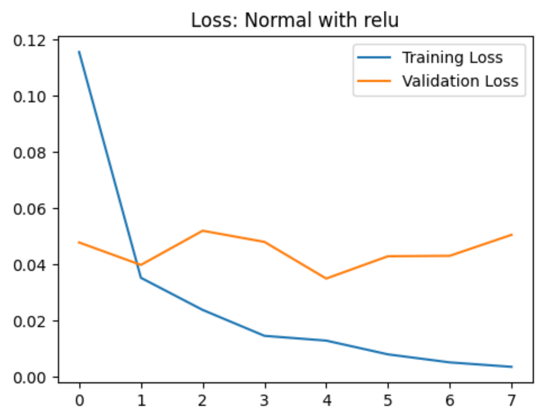
[204]:

```
results = []
for name, init in initializers_dict.items():
    for activation in activations:
        model, history = build_and_train_model(init, activation)
        test_loss, test_acc = model.evaluate(X_test, y_test, verbose=0)
        predictions = model.predict(X_test)
        predictions = (predictions > 0.5).astype(int)
        cm = confusion_matrix(y_test, predictions)

        # displaying the confusion matrix
        plt.figure(figsize=(6, 4))
        sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
        plt.title(f'Confusion Matrix: {name} with {activation}')
        plt.show()

        # for plotting the learning curves
        plt.figure(figsize=(12, 4))
        plt.subplot(121)
        plt.plot(history.history['accuracy'], label='Training Accuracy')
        plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
        plt.title(f'Accuracy: {name} with {activation}')
        plt.legend()

        plt.subplot(122)
        plt.plot(history.history['loss'], label='Training Loss')
        plt.plot(history.history['val_loss'], label='Validation Loss')
        plt.title(f'Loss: {name} with {activation}')
        plt.legend()

        plt.show()

        results.append({
            'Initializer': name,
            'Activation': activation,
            'Accuracy': test_acc,
            'Confusion Matrix': cm
        })
```

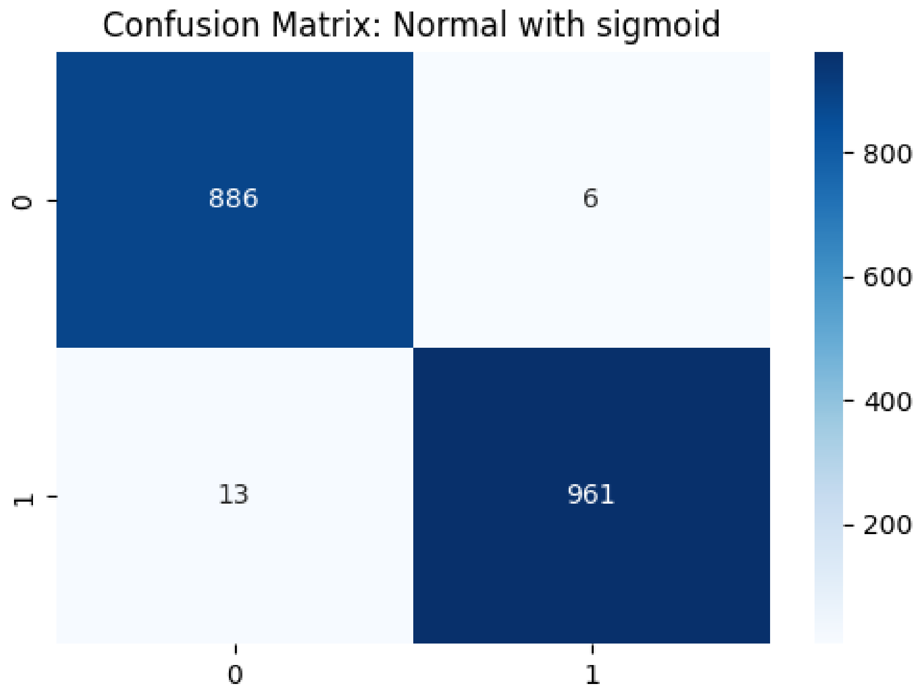
```
59/59              0s 3ms/step
```

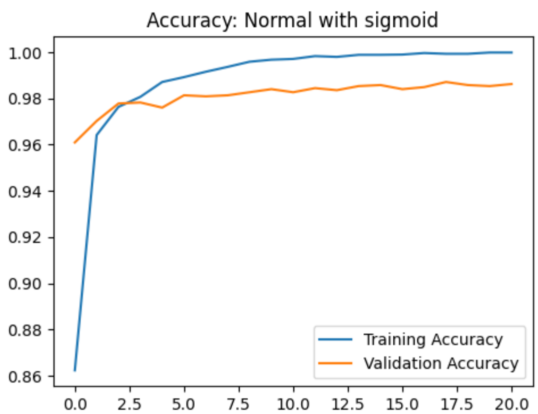


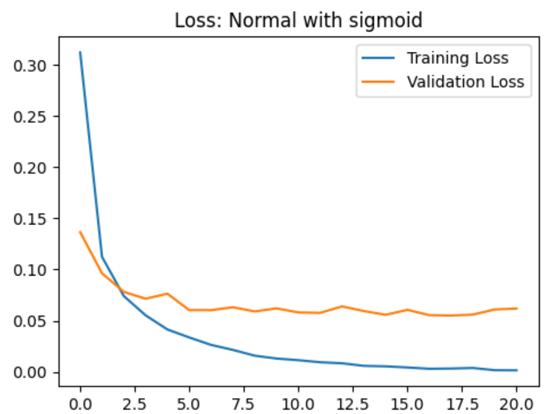




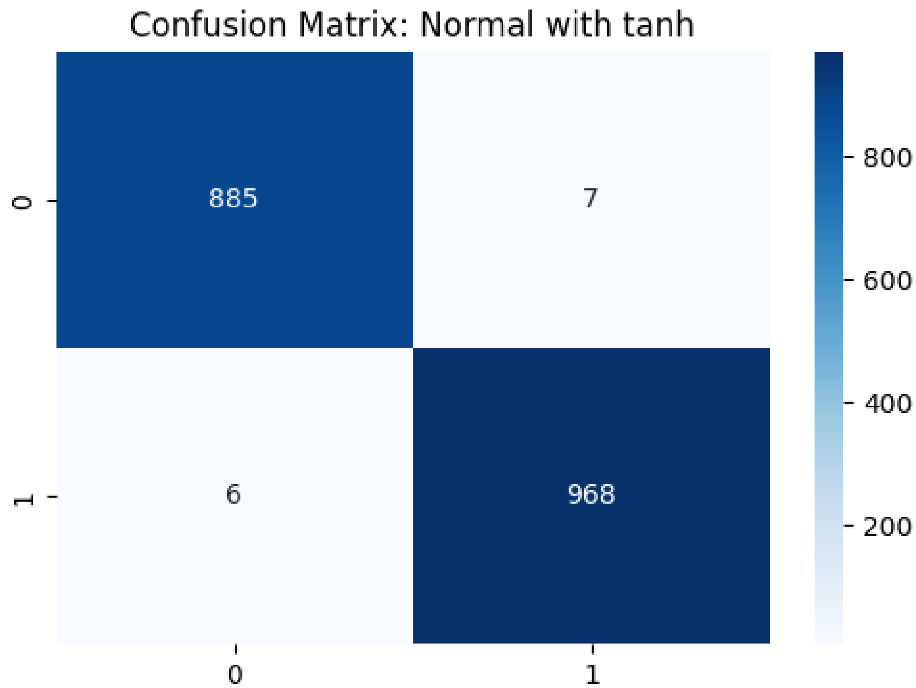

```
59/59                  0s 3ms/step
```

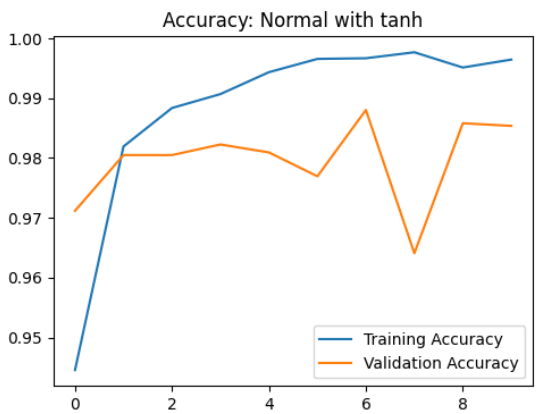


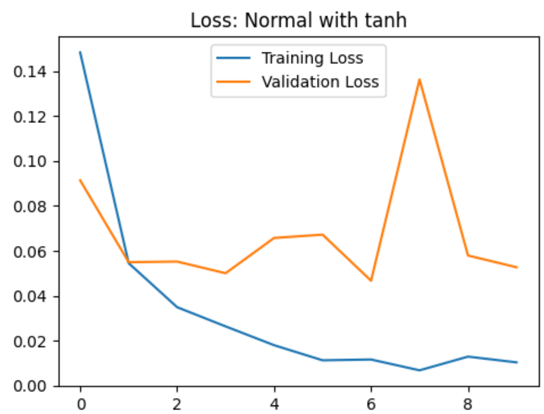




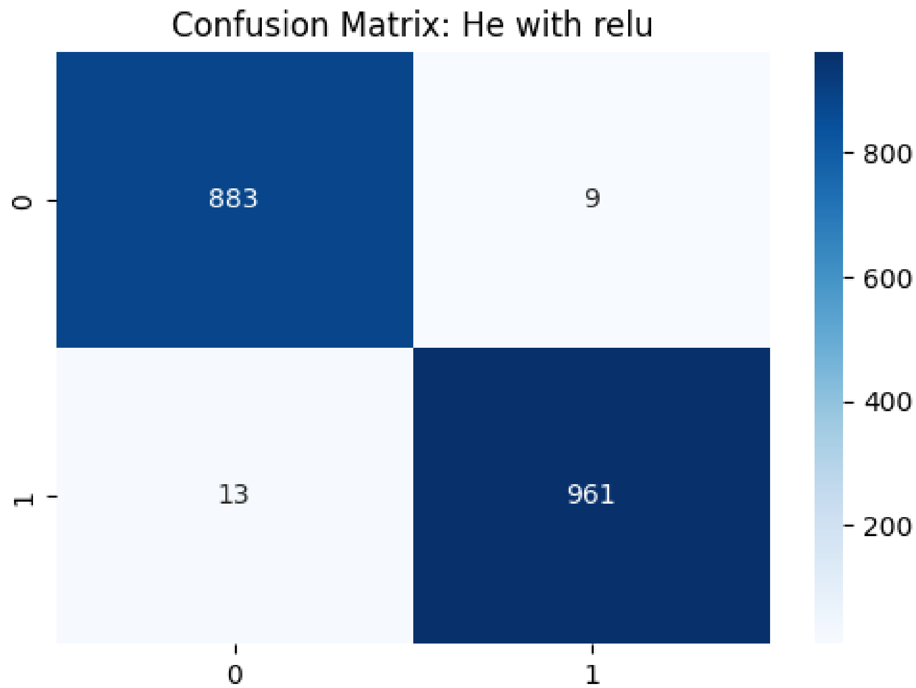

```
59/59                    0s 3ms/step
```

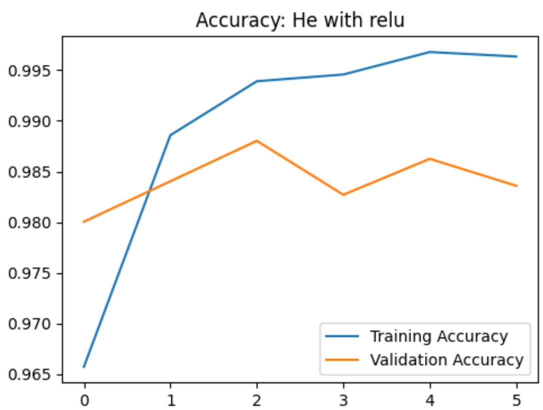


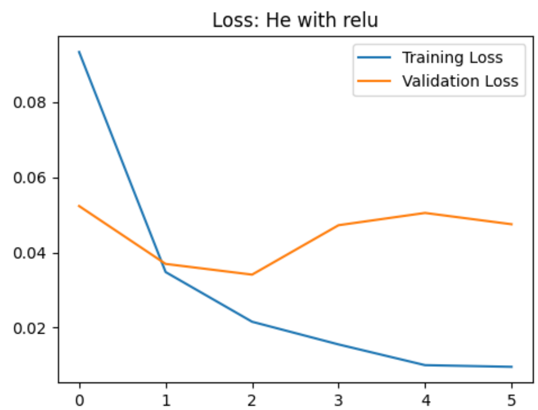




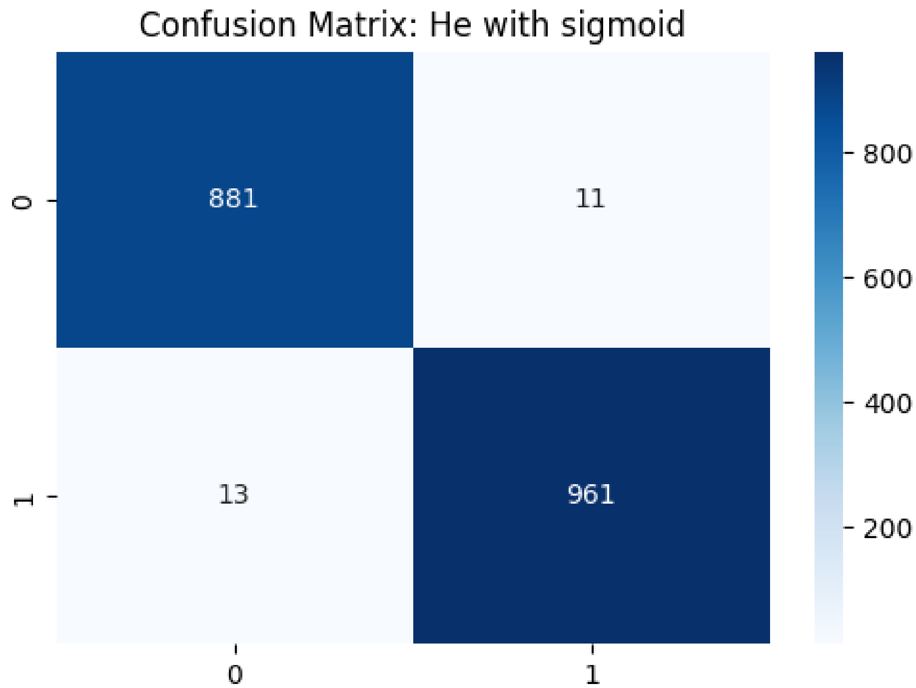

```
59/59                  0s 3ms/step
```

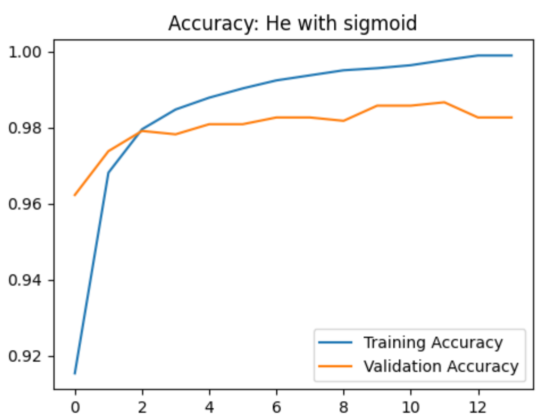


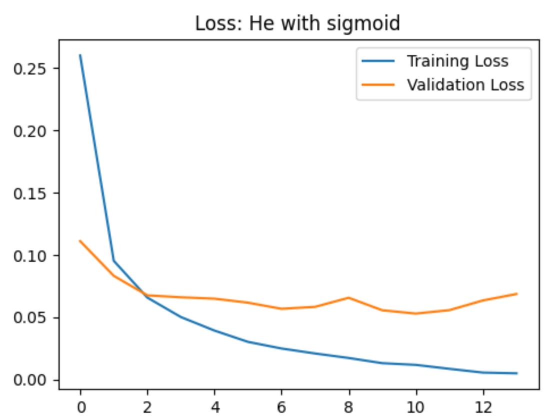




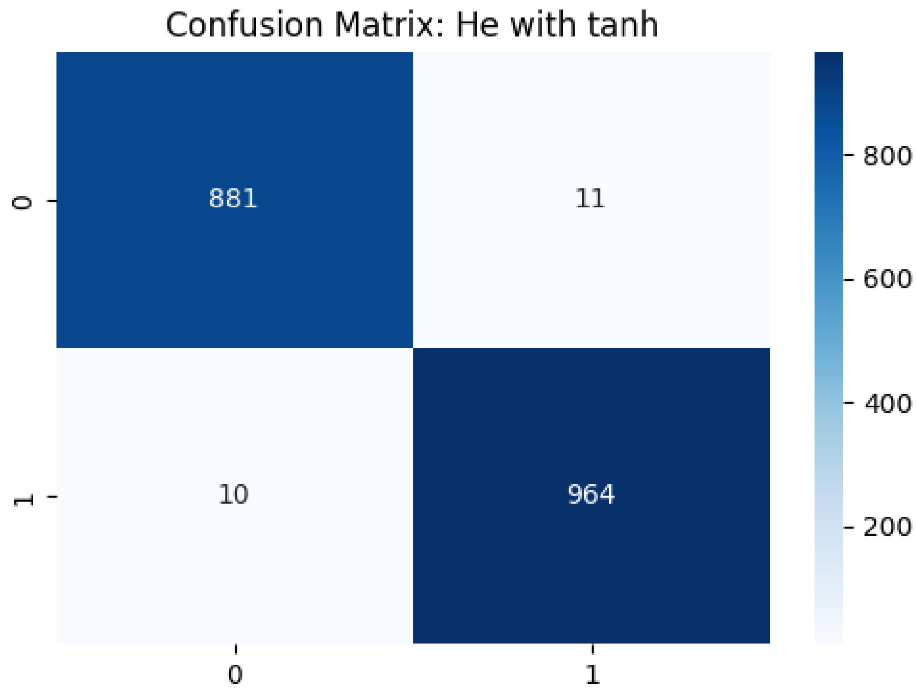

```
59/59                 0s 4ms/step
```

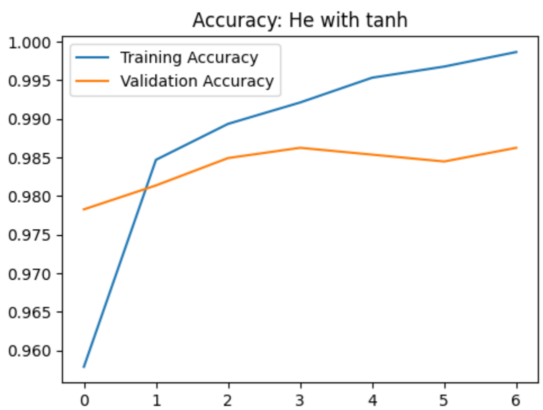


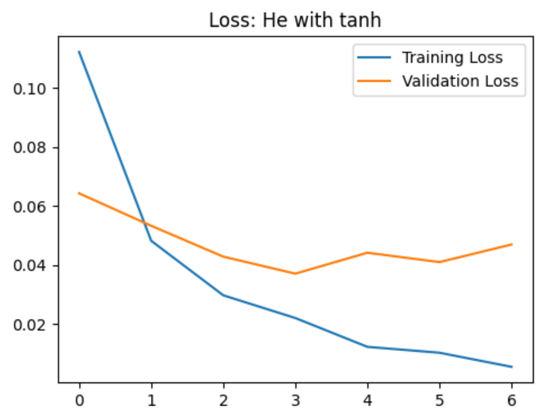




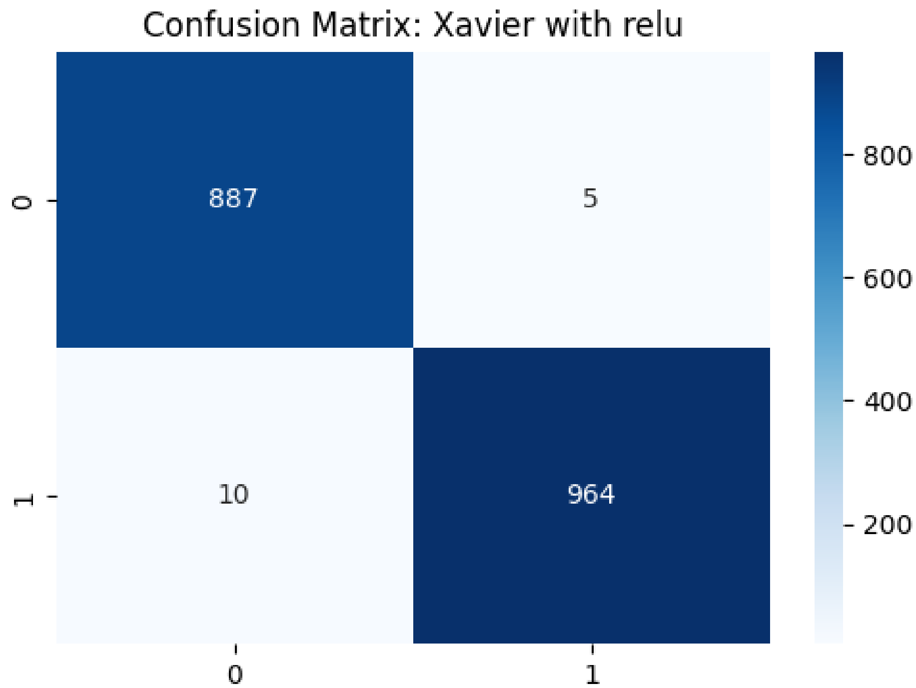

```
59/59                    0s 3ms/step
```

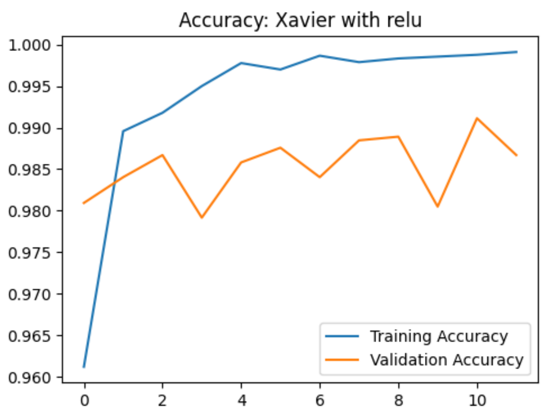


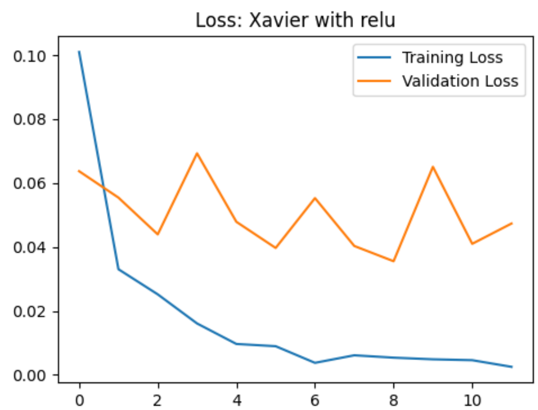




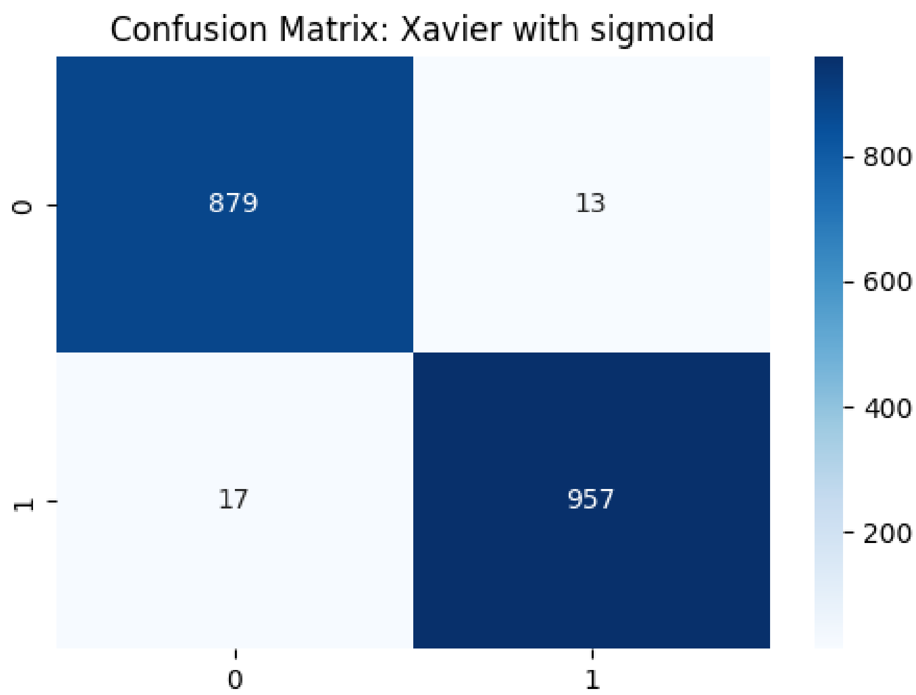

```
59/59            0s 3ms/step
```

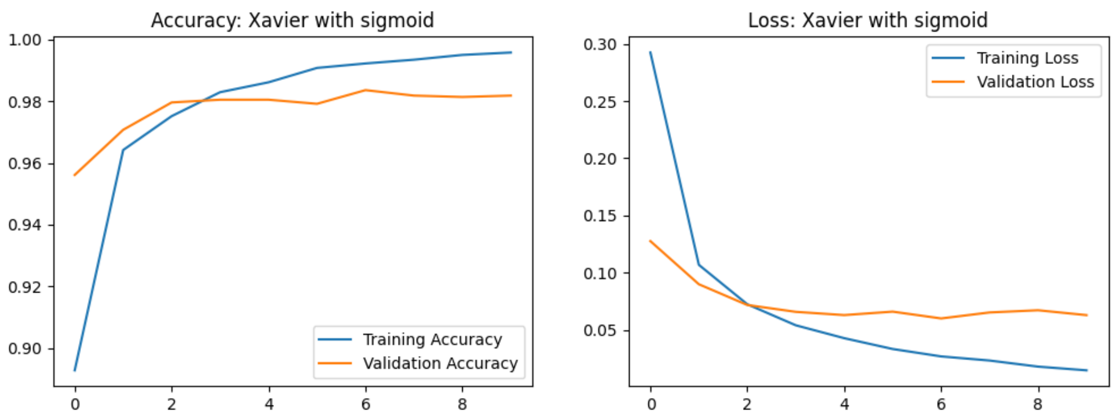




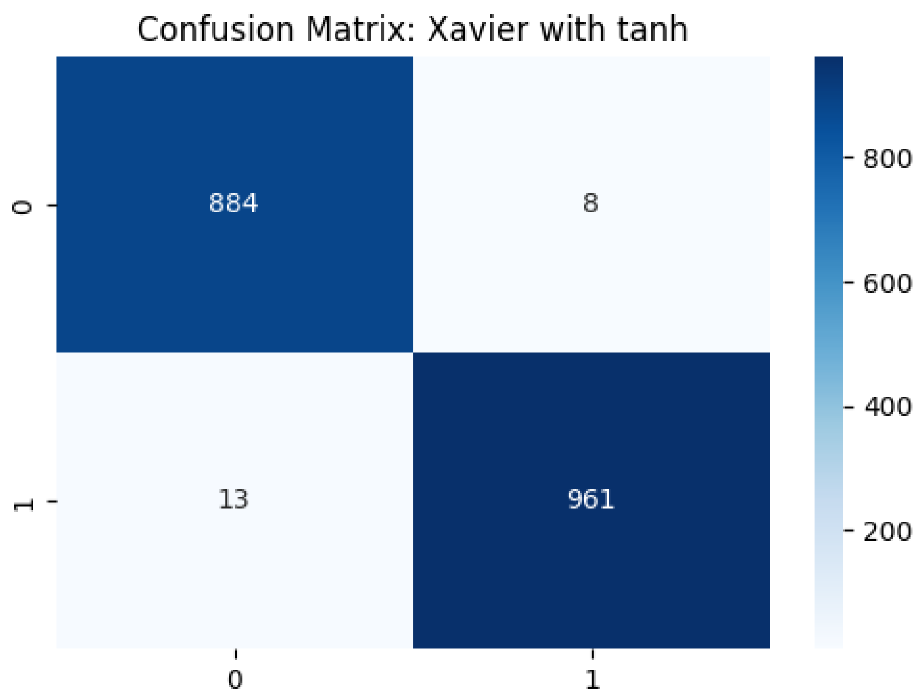


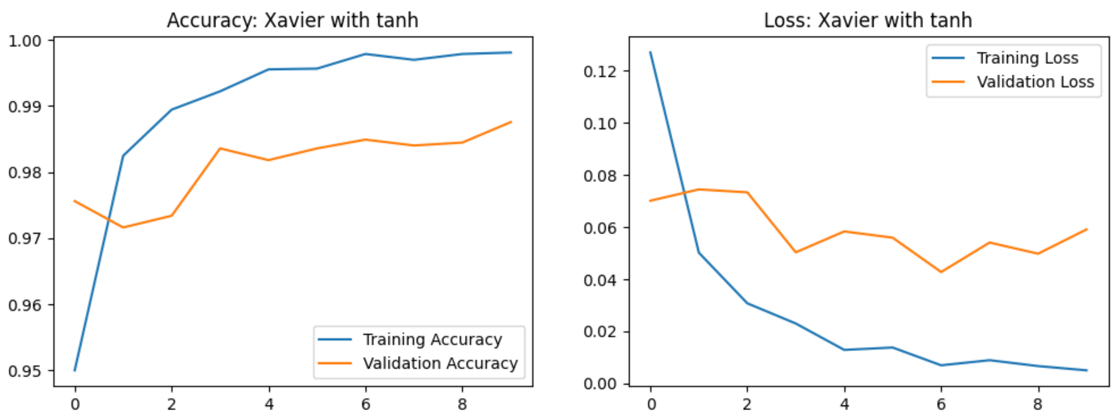

```
59/59                  0s 4ms/step
```






```
59/59                  0s 3ms/step
```





[205]:
```python
precision = precision_score(y_test, predictions)
recall = recall_score(y_test, predictions)
f1 = f1_score(y_test, predictions)

print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
```

```
Precision: 0.9917440660474717
Recall: 0.9866529774127311
```

```
F1 Score: 0.9891919711785898
```

[206]:
```python
import pandas as pd
results_df = pd.DataFrame(results)
print(results_df[['Initializer', 'Activation', 'Accuracy']])
```

```
  Initializer Activation  Accuracy
0      Normal       relu  0.993033
1      Normal    sigmoid  0.989818
2      Normal       tanh  0.993033
3          He       relu  0.988210
4          He    sigmoid  0.987138
5          He       tanh  0.988746
6      Xavier       relu  0.991961
7      Xavier    sigmoid  0.983923
8      Xavier       tanh  0.988746
```

Observations: - The Normal initializer with ReLU activation had the highest accuracy which is about 99.5% among of of the combinations indicating good efficiency. - The He initializer achieved good accuracy results too among which He_tanh has 99.30% accuracy - ReLU consistently led to quicker and more stable learning, suggesting it's highly effective at managing gradient flow. - Lower performance using Sigmoid points to its vulnerability to gradient issues, requiring careful tuning when used.

[206]:
```

```

1. Normal with Relu Combination: - Achieves nearly perfect classification with only 1 false positive and 12 false negatives, indicating strong ability in recognizing and distinguishing between the two classes. - Training and validation accuracy both reach high levels above 99%, demonstrating excellent model learning and generalization capabilities without significant overfitting. - Both training and validation loss decrease sharply and stabilize at very low levels, suggesting effective learning with minimal error at the end of training.

2. Normal with Sigmoid: - This shows a small increase in classification errors with 6 false positives and 13 false negatives slightly less effective than ReLU but still has high overall accuracy. - Exhibits a stable and consistent accuracy profile after initial learning phases, with both training and validation accuracies leveling above 98%, indicating good model reliability. - The validation loss is notably lower and shows less fluctuation compared to ReLU, pointing to better handling of overfitting and noise within the data.

3. Normal with Tanh - Displays increased sensitivity with 7 false positives and 6 false negatives, suggesting potential issues with data extremes that Tanh might not handle as efficiently as ReLU or Sigmoid. - Shows more significant fluctuations in validation accuracy, potentially indicating instability or overfitting issues within the model training process. - Experiences more pronounced spikes in validation loss, especially towards later epochs, which may reflect issues with the model's ability to generalize effectively under Tanh activation.

4. He with ReLU - Very high classification accuracy with minimal misclassifications i.e. 9 false positives, 13 false negatives, indicating effective recognition and discrimination between classes. - Has rapid learning with training accuracy quickly reaching a high level, closely mirrored by validation accuracy which indicates good model generalization without significant overfitting. - The training loss decreases sharply and stabilizes at a very low level, while validation loss also follows but with slight fluctuations, suggesting the model is well-tuned but could possibly benefit from slight adjustments to reduce overfitting.

5. He with Sigmoid - Slightly more misclassifications (11 false positives, 13 false negatives) compared to ReLU, indicating a slight reduction in model's discriminative capability. - Both training and validation accuracies show steady increase and plateau at high levels, indicative of effective learning and generalization, albeit slightly below the performance level seen with ReLU. - Training and validation loss both decrease and flatten out, demonstrating stable learning dynamics. The convergence of validation loss close to training loss suggests that the model with sigmoid is robust against overfitting.

6. He with Tanh - Has quite a few number of misclassifications to sigmoid (11 false positives, 10 false negatives), showing good performance but slightly less effective than ReLU. - Training accuracy climbs swiftly to nearly 99.5%, while validation accuracy exhibits some fluctuations but remains high, indicating some sensitivity to the model's hyperparameters or the specific characteristics of the validation data. - Both training and validation losses decrease, with training loss achieving a lower level faster. However, the validation loss shows some upward fluctuations later, potentially indicating minor overfitting issues as the model excessively adapts to the training data nuances.

7. Xavier with ReLU - This has high classification accuracy with only 5 false positives and 10 false negatives, showing robustness in distinguishing between the two classes. - Training accuracy quickly approaches near-perfect levels, while validation accuracy displays more variability, indicating potential overfitting or sensitivity to validation set nuances. - Both training and validation losses decrease rapidly initially but show notable fluctuations in later epochs, suggesting the need for further parameter tuning or regularization to stabilize learning.

8. Xavier with Sigmoid - Observes a minor increase in classification errors with 13 false positives and 17 false negatives compared to the ReLU activation, suggesting a slight decrease in model sensitivity. - Training and validation accuracies rise quickly and plateau at high levels, indicating effective learning and good generalization capabilities of the model. - Displays a smooth and steady reduction in both training and validation losses, indicating consistent learning without abrupt changes, which is beneficial for model stability and performance.

9. Xavier with Tanh - Produces a low number of false positives (8) and false negatives (13), demonstrating effective classification capabilities close to those seen with ReLU. -This achieves high training accuracy that stabilizes quickly, although validation accuracy shows some fluctuations, potentially indicating slight overfitting or model sensitivity to validation data specifics. - Shows a decreasing trend in validation loss that levels off, reflecting effective learning adaptation and robustness against overfitting compared to the ReLU and Sigmoid activations.

Overall Comaparison: - Xavier with ReLU learns quickly but can be unstable, showing ups and downs in its loss. - Sigmoid, however, learns steadily, making it more reliable over time. - Tanh is in the middle, with quick learning but more stable than ReLU. - Sigmoid does the best job in handling new, unseen data because of its consistent performance. - ReLU, while fast, might overfit and not perform as well on new data. - Tanh offers a good balance, performing well on new data without as many ups and downs as ReLU.

[206]:

```

```

Problem 3 (2 pts): Build an ANN for multi-class classification considering all the classes (10 classes) in the MNIST digit dataset. Finally, present classification report, including class-wise precision, recall, f1-score, and discuss your result.

[207]:
```
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.utils import to_categorical
from sklearn.metrics import classification_report, confusion_matrix
```

[208]:
```
# to load the MNIST dataset (28x28)
(X_train_full, Y_train_full), (X_test_full, Y_test_full) = mnist.load_data()

# shape of the data
print(f"Training data shape: {X_train_full.shape}")
print(f"Testing data shape: {X_test_full.shape}")
```

```
Training data shape: (60000, 28, 28)
Testing data shape: (10000, 28, 28)
```

[209]:
```
# Normalize the data
X_train = X_train_full / 255.0
X_test = X_test_full / 255.0

# to flatten the images from 28x28 to 784
X_train = X_train.reshape(-1, 28 * 28)
X_test = X_test.reshape(-1, 28 * 28)

# One-hot encode the labels
Y_train = to_categorical(Y_train_full, 10)
Y_test = to_categorical(Y_test_full, 10)
```

[210]:
```
# Split in 80% train and 20% validation
X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size=0.
 ↪2, random_state=42)

print(f"Training data shape: {X_train.shape}")
print(f"Validation data shape: {X_val.shape}")
```

```
Training data shape: (48000, 784)
Validation data shape: (12000, 784)
```

[211]:
```
#  model
model = Sequential()
model.add(Dense(128, input_shape=(784,), activation='relu'))  # First hidden␣
 ↪layer
model.add(Dense(64, activation='relu'))                       # Second hidden␣
 ↪layer
model.add(Dense(32, activation='relu'))                       # Third hidden␣
 ↪layer
```

```
model.add(Dense(10, activation='softmax'))                    # Multi-class␣
↪classification so used softmax

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

[212]:
```
model.summary()
```

```
Model: "sequential_60"


 Layer (type)                          Output Shape                         ␣
↪Param #

 dense_233 (Dense)                     (None, 128)                          ␣
↪100,480

 dense_234 (Dense)                     (None, 64)                             ␣
↪8,256

 dense_235 (Dense)                     (None, 32)                             ␣
↪2,080

 dense_236 (Dense)                     (None, 10)                               ␣
↪330


 Total params: 111,146 (434.16 KB)

 Trainable params: 111,146 (434.16 KB)

 Non-trainable params: 0 (0.00 B)
```

[213]:
```
# Early stopping with patience as 3
early_stopping = EarlyStopping(monitor='val_loss', patience=3,␣
↪restore_best_weights=True)
```

[214]:
```
# train the model with 100 epochs
history = model.fit(X_train, Y_train, epochs=100, batch_size=128,
                    validation_data=(X_val, Y_val),
                    callbacks=[early_stopping])
```

```
Epoch 1/100
375/375                    5s 10ms/step -
```

```
accuracy: 0.7757 - loss: 0.7713 - val_accuracy: 0.9405 - val_loss: 0.1993
Epoch 2/100
375/375             6s 16ms/step -
accuracy: 0.9521 - loss: 0.1597 - val_accuracy: 0.9630 - val_loss: 0.1301
Epoch 3/100
375/375             10s 15ms/step -
accuracy: 0.9684 - loss: 0.1061 - val_accuracy: 0.9699 - val_loss: 0.1071
Epoch 4/100
375/375             8s 8ms/step -
accuracy: 0.9762 - loss: 0.0803 - val_accuracy: 0.9707 - val_loss: 0.0993
Epoch 5/100
375/375             6s 11ms/step -
accuracy: 0.9821 - loss: 0.0596 - val_accuracy: 0.9719 - val_loss: 0.0947
Epoch 6/100
375/375             4s 8ms/step -
accuracy: 0.9863 - loss: 0.0482 - val_accuracy: 0.9730 - val_loss: 0.0925
Epoch 7/100
375/375             5s 8ms/step -
accuracy: 0.9890 - loss: 0.0386 - val_accuracy: 0.9693 - val_loss: 0.1049
Epoch 8/100
375/375             9s 18ms/step -
accuracy: 0.9905 - loss: 0.0327 - val_accuracy: 0.9698 - val_loss: 0.1049
Epoch 9/100
375/375             4s 11ms/step -
accuracy: 0.9918 - loss: 0.0286 - val_accuracy: 0.9752 - val_loss: 0.0921
Epoch 10/100
375/375             4s 9ms/step -
accuracy: 0.9945 - loss: 0.0204 - val_accuracy: 0.9755 - val_loss: 0.0982
Epoch 11/100
375/375             3s 8ms/step -
accuracy: 0.9949 - loss: 0.0176 - val_accuracy: 0.9748 - val_loss: 0.1005
Epoch 12/100
375/375             6s 11ms/step -
accuracy: 0.9955 - loss: 0.0166 - val_accuracy: 0.9684 - val_loss: 0.1325
```

[215]:
```
# Evaluate the model on the test data
test_loss, test_acc = model.evaluate(X_test, Y_test)
print(f"Test accuracy: {test_acc}")
```

```
313/313             1s 4ms/step -
accuracy: 0.9721 - loss: 0.1033
Test accuracy: 0.9746000170707703
```

[216]:
```
from sklearn.metrics import classification_report

Y_pred = model.predict(X_test)
```

```
# to convert predictions from probabilities to class labels
Y_pred_classes = np.argmax(Y_pred, axis=1)

# to convert Y_test to class labels
Y_test_classes = np.argmax(Y_test, axis=1)

# classification report
report_dict = classification_report(Y_test_classes, Y_pred_classes,␣
↪output_dict=True)
print(type(report_dict))
```

```
313/313          2s 5ms/step
<class 'dict'>
```

[217]:

```
# to convert dict to a DataFrame
report_df_result = pd.DataFrame(report_dict).transpose()

# report display
display(report_df_result)
```

	precision	recall	f1-score	support
0	0.989754	0.985714	0.987730	980.0000
1	0.989464	0.992952	0.991205	1135.0000
2	0.970221	0.978682	0.974433	1032.0000
3	0.963071	0.981188	0.972045	1010.0000
4	0.963673	0.972505	0.968069	982.0000
5	0.979381	0.958520	0.968839	892.0000
6	0.972079	0.981211	0.976623	958.0000
7	0.979331	0.967899	0.973581	1028.0000
8	0.961382	0.971253	0.966292	974.0000
9	0.976626	0.952428	0.964375	1009.0000
accuracy	0.974600	0.974600	0.974600	0.9746
macro avg	0.974498	0.974235	0.974319	10000.0000
weighted avg	0.974671	0.974600	0.974589	10000.0000

Observations: - Since this a multi class classification , we have used softmax activation function in the output layer. - Softmax is suitable because it converts the output scores from the final layer into probabilities - Here we have three hidden layers that use Relu activation funtion to introduce non-linearity into the model, making it capable of learning more complex patterns. - The classification report shows that the model performs well across all digit classes in the MNIST dataset, with an overall accuracy of 97.46%. - Class 1 and Class 7 exhibit the highest precision, indicating that these digits are predicted with the highest accuracy relative to other classes. - Class 5 has the lowest recall, suggesting that it is slightly more challenging for the model to correctly identify all instances of this class compared to others. - The F1-score, which balances precision and recall, is consistently high across all classes, but Class 9 has a slightly lower score, indicating a small trade-off between precision and recall for this class.

[217]:

```

```

[217]:

```

```

cdnvgquo1

February 24, 2025

Problem 4 (4 pts): Build ANNs for multi-class classification considering all the classes (10 classes) in the MNIST digit dataset with combinations of batch sizes and learning rates. Consider batch sizes: 4, 16, 32, and 64; and learning rate 0.01, 0.001, 0.0001, and 0.00001. Finally, create a plot of test accuracy vs. ratio of batch size to learning rate, and discuss your findings.

[1]:
```
#import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Input
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import EarlyStopping
```

[2]:
```
# Load and preprocess data
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
train_images = train_images.reshape((60000, 28, 28, 1)).astype('float32') / 255
test_images = test_images.reshape((10000, 28, 28, 1)).astype('float32') / 255
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/mnist.npz
11490434/11490434            0s
0us/step
```

[3]:
```
# model
def create_model(learning_rate):
    model = Sequential([
        Input(shape=(28, 28, 1)),
        Flatten(),
        Dense(32, activation='relu'),
        Dense(16, activation='relu'),
        Dense(10, activation='softmax')
    ])
    model.compile(optimizer=Adam(learning_rate=learning_rate),
```

```
                    loss='categorical_crossentropy',
                    metrics=['accuracy'])
    return model
```

[10]:

```
model.summary()
```

```
Model: "sequential_17"


 Layer (type)                           Output Shape                          ␣
↪Param #

 flatten_17 (Flatten)                   (None, 784)                               ␣
↪   0

 dense_51 (Dense)                       (None, 32)                            ␣
↪25,120

 dense_52 (Dense)                       (None, 16)                                ␣
↪528

 dense_53 (Dense)                       (None, 10)                                ␣
↪170



 Total params: 25,818 (100.85 KB)


 Trainable params: 25,818 (100.85 KB)


 Non-trainable params: 0 (0.00 B)
```

[4]:

```
# earlyStopping
early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=3,
    restore_best_weights=True,
    verbose=1
)

# Batch sizes and learning rates combinations
batch_sizes = [4, 16, 32, 64]
learning_rates = [0.01, 0.001, 0.0001, 0.00001]
results = []
```

[5]:
```
# to train models with different batch sizes and learning rates
for batch_size in batch_sizes:
    for learning_rate in learning_rates:
        print(f"Training with batch size {batch_size} and learning rate␣
 ↪{learning_rate}")
        model = create_model(learning_rate=learning_rate)
        model.fit(
            train_images,
            train_labels,
            epochs=100,      #epochs = 100
            batch_size=batch_size,
            validation_split=0.1,
            callbacks=[early_stopping],
            verbose=1
        )
        test_loss, test_accuracy = model.evaluate(test_images, test_labels,␣
 ↪verbose=0)
        results.append((batch_size, learning_rate, test_accuracy))
```

```
Training with batch size 4 and learning rate 0.01
Epoch 1/100
13500/13500            21s 1ms/step
- accuracy: 0.8362 - loss: 0.5546 - val_accuracy: 0.9300 - val_loss: 0.2687
Epoch 2/100
13500/13500            19s 1ms/step
- accuracy: 0.9123 - loss: 0.3418 - val_accuracy: 0.9295 - val_loss: 0.2779
Epoch 3/100
13500/13500            20s 1ms/step
- accuracy: 0.9182 - loss: 0.3273 - val_accuracy: 0.9418 - val_loss: 0.2434
Epoch 4/100
13500/13500            21s 2ms/step
- accuracy: 0.9239 - loss: 0.3116 - val_accuracy: 0.9402 - val_loss: 0.2537
Epoch 5/100
13500/13500            20s 1ms/step
- accuracy: 0.9276 - loss: 0.2937 - val_accuracy: 0.9393 - val_loss: 0.2606
Epoch 6/100
13500/13500            22s 2ms/step
- accuracy: 0.9283 - loss: 0.2981 - val_accuracy: 0.9210 - val_loss: 0.3004
Epoch 6: early stopping
Restoring model weights from the end of the best epoch: 3.
Training with batch size 4 and learning rate 0.001
Epoch 1/100
13500/13500            21s 1ms/step
- accuracy: 0.8597 - loss: 0.4568 - val_accuracy: 0.9553 - val_loss: 0.1553
Epoch 2/100
13500/13500            21s 2ms/step
- accuracy: 0.9531 - loss: 0.1598 - val_accuracy: 0.9667 - val_loss: 0.1225
```

```
Epoch 3/100
13500/13500             40s 1ms/step
- accuracy: 0.9625 - loss: 0.1223 - val_accuracy: 0.9678 - val_loss: 0.1087
Epoch 4/100
13500/13500             21s 1ms/step
- accuracy: 0.9697 - loss: 0.1005 - val_accuracy: 0.9635 - val_loss: 0.1352
Epoch 5/100
13500/13500             21s 1ms/step
- accuracy: 0.9722 - loss: 0.0899 - val_accuracy: 0.9675 - val_loss: 0.1199
Epoch 6/100
13500/13500             21s 2ms/step
- accuracy: 0.9750 - loss: 0.0809 - val_accuracy: 0.9645 - val_loss: 0.1304
Epoch 6: early stopping
Restoring model weights from the end of the best epoch: 3.
Training with batch size 4 and learning rate 0.0001
Epoch 1/100
13500/13500             21s 2ms/step
- accuracy: 0.6951 - loss: 1.0041 - val_accuracy: 0.9285 - val_loss: 0.2665
Epoch 2/100
13500/13500             40s 1ms/step
- accuracy: 0.9153 - loss: 0.3030 - val_accuracy: 0.9358 - val_loss: 0.2223
Epoch 3/100
13500/13500             22s 2ms/step
- accuracy: 0.9300 - loss: 0.2441 - val_accuracy: 0.9495 - val_loss: 0.1841
Epoch 4/100
13500/13500             21s 2ms/step
- accuracy: 0.9400 - loss: 0.2090 - val_accuracy: 0.9567 - val_loss: 0.1647
Epoch 5/100
13500/13500             41s 2ms/step
- accuracy: 0.9463 - loss: 0.1833 - val_accuracy: 0.9593 - val_loss: 0.1525
Epoch 6/100
13500/13500             20s 2ms/step
- accuracy: 0.9528 - loss: 0.1643 - val_accuracy: 0.9597 - val_loss: 0.1462
Epoch 7/100
13500/13500             21s 2ms/step
- accuracy: 0.9555 - loss: 0.1521 - val_accuracy: 0.9627 - val_loss: 0.1383
Epoch 8/100
13500/13500             43s 2ms/step
- accuracy: 0.9587 - loss: 0.1443 - val_accuracy: 0.9630 - val_loss: 0.1320
Epoch 9/100
13500/13500             40s 2ms/step
- accuracy: 0.9605 - loss: 0.1307 - val_accuracy: 0.9652 - val_loss: 0.1317
Epoch 10/100
13500/13500             21s 2ms/step
- accuracy: 0.9639 - loss: 0.1238 - val_accuracy: 0.9648 - val_loss: 0.1277
Epoch 11/100
13500/13500             40s 2ms/step
- accuracy: 0.9656 - loss: 0.1186 - val_accuracy: 0.9655 - val_loss: 0.1242
```

```
Epoch 12/100
13500/13500            21s 2ms/step
- accuracy: 0.9678 - loss: 0.1098 - val_accuracy: 0.9672 - val_loss: 0.1191
Epoch 13/100
13500/13500            21s 2ms/step
- accuracy: 0.9677 - loss: 0.1063 - val_accuracy: 0.9668 - val_loss: 0.1191
Epoch 14/100
13500/13500            41s 2ms/step
- accuracy: 0.9708 - loss: 0.1010 - val_accuracy: 0.9653 - val_loss: 0.1210
Epoch 15/100
13500/13500            21s 2ms/step
- accuracy: 0.9719 - loss: 0.0949 - val_accuracy: 0.9682 - val_loss: 0.1141
Epoch 16/100
13500/13500            21s 2ms/step
- accuracy: 0.9741 - loss: 0.0899 - val_accuracy: 0.9697 - val_loss: 0.1157
Epoch 17/100
13500/13500            41s 2ms/step
- accuracy: 0.9750 - loss: 0.0868 - val_accuracy: 0.9703 - val_loss: 0.1118
Epoch 18/100
13500/13500            41s 2ms/step
- accuracy: 0.9765 - loss: 0.0800 - val_accuracy: 0.9708 - val_loss: 0.1127
Epoch 19/100
13500/13500            41s 2ms/step
- accuracy: 0.9758 - loss: 0.0829 - val_accuracy: 0.9713 - val_loss: 0.1122
Epoch 20/100
13500/13500            41s 2ms/step
- accuracy: 0.9779 - loss: 0.0762 - val_accuracy: 0.9700 - val_loss: 0.1166
Epoch 20: early stopping
Restoring model weights from the end of the best epoch: 17.
Training with batch size 4 and learning rate 1e-05
Epoch 1/100
13500/13500            23s 2ms/step
- accuracy: 0.3344 - loss: 1.9746 - val_accuracy: 0.7358 - val_loss: 1.0314
Epoch 2/100
13500/13500            40s 2ms/step
- accuracy: 0.7638 - loss: 0.9308 - val_accuracy: 0.8717 - val_loss: 0.5906
Epoch 3/100
13500/13500            41s 2ms/step
- accuracy: 0.8535 - loss: 0.6101 - val_accuracy: 0.8998 - val_loss: 0.4452
Epoch 4/100
13500/13500            41s 2ms/step
- accuracy: 0.8771 - loss: 0.4883 - val_accuracy: 0.9113 - val_loss: 0.3757
Epoch 5/100
13500/13500            40s 2ms/step
- accuracy: 0.8903 - loss: 0.4262 - val_accuracy: 0.9167 - val_loss: 0.3356
Epoch 6/100
13500/13500            41s 2ms/step
- accuracy: 0.8959 - loss: 0.3883 - val_accuracy: 0.9228 - val_loss: 0.3083
```

```
Epoch 7/100
13500/13500              21s 2ms/step
- accuracy: 0.9042 - loss: 0.3603 - val_accuracy: 0.9257 - val_loss: 0.2902
Epoch 8/100
13500/13500              41s 2ms/step
- accuracy: 0.9073 - loss: 0.3476 - val_accuracy: 0.9280 - val_loss: 0.2758
Epoch 9/100
13500/13500              42s 2ms/step
- accuracy: 0.9124 - loss: 0.3221 - val_accuracy: 0.9292 - val_loss: 0.2650
Epoch 10/100
13500/13500              41s 2ms/step
- accuracy: 0.9142 - loss: 0.3123 - val_accuracy: 0.9317 - val_loss: 0.2557
Epoch 11/100
13500/13500              21s 2ms/step
- accuracy: 0.9172 - loss: 0.3057 - val_accuracy: 0.9330 - val_loss: 0.2481
Epoch 12/100
13500/13500              21s 2ms/step
- accuracy: 0.9193 - loss: 0.2913 - val_accuracy: 0.9350 - val_loss: 0.2425
Epoch 13/100
13500/13500              21s 2ms/step
- accuracy: 0.9207 - loss: 0.2871 - val_accuracy: 0.9355 - val_loss: 0.2365
Epoch 14/100
13500/13500              41s 2ms/step
- accuracy: 0.9224 - loss: 0.2800 - val_accuracy: 0.9370 - val_loss: 0.2319
Epoch 15/100
13500/13500              22s 2ms/step
- accuracy: 0.9249 - loss: 0.2735 - val_accuracy: 0.9377 - val_loss: 0.2259
Epoch 16/100
13500/13500              22s 2ms/step
- accuracy: 0.9241 - loss: 0.2678 - val_accuracy: 0.9390 - val_loss: 0.2218
Epoch 17/100
13500/13500              23s 2ms/step
- accuracy: 0.9263 - loss: 0.2587 - val_accuracy: 0.9383 - val_loss: 0.2187
Epoch 18/100
13500/13500              22s 2ms/step
- accuracy: 0.9281 - loss: 0.2592 - val_accuracy: 0.9400 - val_loss: 0.2151
Epoch 19/100
13500/13500              41s 2ms/step
- accuracy: 0.9281 - loss: 0.2510 - val_accuracy: 0.9415 - val_loss: 0.2120
Epoch 20/100
13500/13500              22s 2ms/step
- accuracy: 0.9305 - loss: 0.2475 - val_accuracy: 0.9425 - val_loss: 0.2081
Epoch 21/100
13500/13500              42s 2ms/step
- accuracy: 0.9315 - loss: 0.2406 - val_accuracy: 0.9432 - val_loss: 0.2055
Epoch 22/100
13500/13500              39s 2ms/step
- accuracy: 0.9315 - loss: 0.2434 - val_accuracy: 0.9448 - val_loss: 0.2029
```

```
Epoch 23/100
13500/13500              42s 2ms/step
- accuracy: 0.9309 - loss: 0.2431 - val_accuracy: 0.9438 - val_loss: 0.1996
Epoch 24/100
13500/13500              41s 2ms/step
- accuracy: 0.9334 - loss: 0.2349 - val_accuracy: 0.9463 - val_loss: 0.1970
Epoch 25/100
13500/13500              21s 2ms/step
- accuracy: 0.9369 - loss: 0.2275 - val_accuracy: 0.9468 - val_loss: 0.1941
Epoch 26/100
13500/13500              42s 2ms/step
- accuracy: 0.9361 - loss: 0.2268 - val_accuracy: 0.9483 - val_loss: 0.1919
Epoch 27/100
13500/13500              22s 2ms/step
- accuracy: 0.9365 - loss: 0.2250 - val_accuracy: 0.9482 - val_loss: 0.1894
Epoch 28/100
13500/13500              41s 2ms/step
- accuracy: 0.9373 - loss: 0.2210 - val_accuracy: 0.9480 - val_loss: 0.1875
Epoch 29/100
13500/13500              41s 2ms/step
- accuracy: 0.9390 - loss: 0.2190 - val_accuracy: 0.9485 - val_loss: 0.1853
Epoch 30/100
13500/13500              21s 2ms/step
- accuracy: 0.9390 - loss: 0.2165 - val_accuracy: 0.9495 - val_loss: 0.1838
Epoch 31/100
13500/13500              41s 2ms/step
- accuracy: 0.9387 - loss: 0.2176 - val_accuracy: 0.9502 - val_loss: 0.1842
Epoch 32/100
13500/13500              22s 2ms/step
- accuracy: 0.9408 - loss: 0.2093 - val_accuracy: 0.9502 - val_loss: 0.1798
Epoch 33/100
13500/13500              40s 2ms/step
- accuracy: 0.9407 - loss: 0.2077 - val_accuracy: 0.9505 - val_loss: 0.1791
Epoch 34/100
13500/13500              21s 2ms/step
- accuracy: 0.9422 - loss: 0.2045 - val_accuracy: 0.9513 - val_loss: 0.1765
Epoch 35/100
13500/13500              22s 2ms/step
- accuracy: 0.9427 - loss: 0.2014 - val_accuracy: 0.9525 - val_loss: 0.1750
Epoch 36/100
13500/13500              22s 2ms/step
- accuracy: 0.9436 - loss: 0.1982 - val_accuracy: 0.9517 - val_loss: 0.1737
Epoch 37/100
13500/13500              20s 2ms/step
- accuracy: 0.9433 - loss: 0.2015 - val_accuracy: 0.9528 - val_loss: 0.1722
Epoch 38/100
13500/13500              22s 2ms/step
- accuracy: 0.9450 - loss: 0.1968 - val_accuracy: 0.9543 - val_loss: 0.1710
```

```
Epoch 39/100
13500/13500              40s 2ms/step
- accuracy: 0.9457 - loss: 0.1920 - val_accuracy: 0.9533 - val_loss: 0.1695
Epoch 40/100
13500/13500              21s 2ms/step
- accuracy: 0.9430 - loss: 0.1996 - val_accuracy: 0.9540 - val_loss: 0.1682
Epoch 41/100
13500/13500              22s 2ms/step
- accuracy: 0.9470 - loss: 0.1882 - val_accuracy: 0.9547 - val_loss: 0.1676
Epoch 42/100
13500/13500              21s 2ms/step
- accuracy: 0.9474 - loss: 0.1890 - val_accuracy: 0.9543 - val_loss: 0.1659
Epoch 43/100
13500/13500              22s 2ms/step
- accuracy: 0.9472 - loss: 0.1859 - val_accuracy: 0.9548 - val_loss: 0.1651
Epoch 44/100
13500/13500              22s 2ms/step
- accuracy: 0.9474 - loss: 0.1863 - val_accuracy: 0.9552 - val_loss: 0.1637
Epoch 45/100
13500/13500              41s 2ms/step
- accuracy: 0.9492 - loss: 0.1804 - val_accuracy: 0.9545 - val_loss: 0.1628
Epoch 46/100
13500/13500              41s 2ms/step
- accuracy: 0.9486 - loss: 0.1793 - val_accuracy: 0.9565 - val_loss: 0.1616
Epoch 47/100
13500/13500              40s 2ms/step
- accuracy: 0.9500 - loss: 0.1767 - val_accuracy: 0.9562 - val_loss: 0.1609
Epoch 48/100
13500/13500              40s 2ms/step
- accuracy: 0.9488 - loss: 0.1803 - val_accuracy: 0.9565 - val_loss: 0.1609
Epoch 49/100
13500/13500              22s 2ms/step
- accuracy: 0.9500 - loss: 0.1771 - val_accuracy: 0.9580 - val_loss: 0.1592
Epoch 50/100
13500/13500              40s 2ms/step
- accuracy: 0.9502 - loss: 0.1757 - val_accuracy: 0.9565 - val_loss: 0.1583
Epoch 51/100
13500/13500              42s 2ms/step
- accuracy: 0.9524 - loss: 0.1728 - val_accuracy: 0.9573 - val_loss: 0.1568
Epoch 52/100
13500/13500              42s 2ms/step
- accuracy: 0.9503 - loss: 0.1734 - val_accuracy: 0.9582 - val_loss: 0.1563
Epoch 53/100
13500/13500              23s 2ms/step
- accuracy: 0.9511 - loss: 0.1711 - val_accuracy: 0.9580 - val_loss: 0.1552
Epoch 54/100
13500/13500              41s 2ms/step
- accuracy: 0.9540 - loss: 0.1634 - val_accuracy: 0.9585 - val_loss: 0.1545
```

```
Epoch 55/100
13500/13500             41s 2ms/step
- accuracy: 0.9507 - loss: 0.1742 - val_accuracy: 0.9598 - val_loss: 0.1535
Epoch 56/100
13500/13500             41s 2ms/step
- accuracy: 0.9518 - loss: 0.1714 - val_accuracy: 0.9588 - val_loss: 0.1529
Epoch 57/100
13500/13500             22s 2ms/step
- accuracy: 0.9529 - loss: 0.1631 - val_accuracy: 0.9598 - val_loss: 0.1523
Epoch 58/100
13500/13500             40s 2ms/step
- accuracy: 0.9532 - loss: 0.1649 - val_accuracy: 0.9600 - val_loss: 0.1517
Epoch 59/100
13500/13500             21s 2ms/step
- accuracy: 0.9530 - loss: 0.1606 - val_accuracy: 0.9597 - val_loss: 0.1505
Epoch 60/100
13500/13500             42s 2ms/step
- accuracy: 0.9544 - loss: 0.1606 - val_accuracy: 0.9612 - val_loss: 0.1492
Epoch 61/100
13500/13500             21s 2ms/step
- accuracy: 0.9547 - loss: 0.1577 - val_accuracy: 0.9605 - val_loss: 0.1491
Epoch 62/100
13500/13500             21s 2ms/step
- accuracy: 0.9549 - loss: 0.1565 - val_accuracy: 0.9612 - val_loss: 0.1477
Epoch 63/100
13500/13500             41s 2ms/step
- accuracy: 0.9561 - loss: 0.1575 - val_accuracy: 0.9607 - val_loss: 0.1474
Epoch 64/100
13500/13500             22s 2ms/step
- accuracy: 0.9552 - loss: 0.1588 - val_accuracy: 0.9605 - val_loss: 0.1473
Epoch 65/100
13500/13500             21s 2ms/step
- accuracy: 0.9573 - loss: 0.1560 - val_accuracy: 0.9610 - val_loss: 0.1464
Epoch 66/100
13500/13500             21s 2ms/step
- accuracy: 0.9575 - loss: 0.1504 - val_accuracy: 0.9617 - val_loss: 0.1460
Epoch 67/100
13500/13500             43s 2ms/step
- accuracy: 0.9581 - loss: 0.1480 - val_accuracy: 0.9603 - val_loss: 0.1451
Epoch 68/100
13500/13500             40s 2ms/step
- accuracy: 0.9569 - loss: 0.1541 - val_accuracy: 0.9607 - val_loss: 0.1447
Epoch 69/100
13500/13500             42s 2ms/step
- accuracy: 0.9582 - loss: 0.1462 - val_accuracy: 0.9612 - val_loss: 0.1439
Epoch 70/100
13500/13500             22s 2ms/step
- accuracy: 0.9586 - loss: 0.1479 - val_accuracy: 0.9613 - val_loss: 0.1437
```

```
Epoch 71/100
13500/13500              40s 2ms/step
- accuracy: 0.9589 - loss: 0.1427 - val_accuracy: 0.9617 - val_loss: 0.1430
Epoch 72/100
13500/13500              21s 2ms/step
- accuracy: 0.9583 - loss: 0.1488 - val_accuracy: 0.9617 - val_loss: 0.1427
Epoch 73/100
13500/13500              22s 2ms/step
- accuracy: 0.9605 - loss: 0.1409 - val_accuracy: 0.9622 - val_loss: 0.1420
Epoch 74/100
13500/13500              22s 2ms/step
- accuracy: 0.9598 - loss: 0.1439 - val_accuracy: 0.9620 - val_loss: 0.1423
Epoch 75/100
13500/13500              41s 2ms/step
- accuracy: 0.9590 - loss: 0.1437 - val_accuracy: 0.9620 - val_loss: 0.1413
Epoch 76/100
13500/13500              40s 2ms/step
- accuracy: 0.9593 - loss: 0.1426 - val_accuracy: 0.9618 - val_loss: 0.1412
Epoch 77/100
13500/13500              41s 2ms/step
- accuracy: 0.9594 - loss: 0.1435 - val_accuracy: 0.9627 - val_loss: 0.1404
Epoch 78/100
13500/13500              22s 2ms/step
- accuracy: 0.9622 - loss: 0.1365 - val_accuracy: 0.9617 - val_loss: 0.1404
Epoch 79/100
13500/13500              22s 2ms/step
- accuracy: 0.9608 - loss: 0.1401 - val_accuracy: 0.9622 - val_loss: 0.1398
Epoch 80/100
13500/13500              22s 2ms/step
- accuracy: 0.9617 - loss: 0.1367 - val_accuracy: 0.9617 - val_loss: 0.1391
Epoch 81/100
13500/13500              41s 2ms/step
- accuracy: 0.9608 - loss: 0.1398 - val_accuracy: 0.9612 - val_loss: 0.1394
Epoch 82/100
13500/13500              41s 2ms/step
- accuracy: 0.9608 - loss: 0.1358 - val_accuracy: 0.9620 - val_loss: 0.1386
Epoch 83/100
13500/13500              21s 2ms/step
- accuracy: 0.9612 - loss: 0.1367 - val_accuracy: 0.9625 - val_loss: 0.1383
Epoch 84/100
13500/13500              42s 2ms/step
- accuracy: 0.9622 - loss: 0.1310 - val_accuracy: 0.9625 - val_loss: 0.1384
Epoch 85/100
13500/13500              41s 2ms/step
- accuracy: 0.9626 - loss: 0.1319 - val_accuracy: 0.9630 - val_loss: 0.1370
Epoch 86/100
13500/13500              40s 2ms/step
- accuracy: 0.9619 - loss: 0.1353 - val_accuracy: 0.9625 - val_loss: 0.1369
```

```
Epoch 87/100
13500/13500            41s 2ms/step
- accuracy: 0.9633 - loss: 0.1323 - val_accuracy: 0.9628 - val_loss: 0.1363
Epoch 88/100
13500/13500            22s 2ms/step
- accuracy: 0.9620 - loss: 0.1322 - val_accuracy: 0.9627 - val_loss: 0.1356
Epoch 89/100
13500/13500            41s 2ms/step
- accuracy: 0.9640 - loss: 0.1303 - val_accuracy: 0.9623 - val_loss: 0.1356
Epoch 90/100
13500/13500            22s 2ms/step
- accuracy: 0.9629 - loss: 0.1308 - val_accuracy: 0.9637 - val_loss: 0.1349
Epoch 91/100
13500/13500            22s 2ms/step
- accuracy: 0.9627 - loss: 0.1295 - val_accuracy: 0.9640 - val_loss: 0.1353
Epoch 92/100
13500/13500            22s 2ms/step
- accuracy: 0.9653 - loss: 0.1236 - val_accuracy: 0.9635 - val_loss: 0.1343
Epoch 93/100
13500/13500            23s 2ms/step
- accuracy: 0.9639 - loss: 0.1269 - val_accuracy: 0.9627 - val_loss: 0.1348
Epoch 94/100
13500/13500            22s 2ms/step
- accuracy: 0.9633 - loss: 0.1279 - val_accuracy: 0.9640 - val_loss: 0.1343
Epoch 95/100
13500/13500            41s 2ms/step
- accuracy: 0.9647 - loss: 0.1247 - val_accuracy: 0.9638 - val_loss: 0.1332
Epoch 96/100
13500/13500            22s 2ms/step
- accuracy: 0.9654 - loss: 0.1225 - val_accuracy: 0.9647 - val_loss: 0.1334
Epoch 97/100
13500/13500            41s 2ms/step
- accuracy: 0.9644 - loss: 0.1261 - val_accuracy: 0.9638 - val_loss: 0.1331
Epoch 98/100
13500/13500            22s 2ms/step
- accuracy: 0.9642 - loss: 0.1267 - val_accuracy: 0.9638 - val_loss: 0.1330
Epoch 99/100
13500/13500            40s 2ms/step
- accuracy: 0.9652 - loss: 0.1224 - val_accuracy: 0.9635 - val_loss: 0.1323
Epoch 100/100
13500/13500            42s 2ms/step
- accuracy: 0.9660 - loss: 0.1222 - val_accuracy: 0.9640 - val_loss: 0.1322
Restoring model weights from the end of the best epoch: 100.
Training with batch size 16 and learning rate 0.01
Epoch 1/100
3375/3375            7s 2ms/step -
accuracy: 0.8560 - loss: 0.4736 - val_accuracy: 0.9420 - val_loss: 0.2078
Epoch 2/100
```

```
3375/3375            6s 2ms/step -
accuracy: 0.9285 - loss: 0.2504 - val_accuracy: 0.9467 - val_loss: 0.1920
Epoch 3/100
3375/3375            10s 2ms/step -
accuracy: 0.9375 - loss: 0.2260 - val_accuracy: 0.9487 - val_loss: 0.2034
Epoch 4/100
3375/3375            6s 2ms/step -
accuracy: 0.9423 - loss: 0.2101 - val_accuracy: 0.9495 - val_loss: 0.1917
Epoch 5/100
3375/3375            5s 2ms/step -
accuracy: 0.9446 - loss: 0.2098 - val_accuracy: 0.9562 - val_loss: 0.1916
Epoch 6/100
3375/3375            10s 2ms/step -
accuracy: 0.9486 - loss: 0.1998 - val_accuracy: 0.9567 - val_loss: 0.1709
Epoch 7/100
3375/3375            5s 2ms/step -
accuracy: 0.9519 - loss: 0.1755 - val_accuracy: 0.9505 - val_loss: 0.2025
Epoch 8/100
3375/3375            6s 2ms/step -
accuracy: 0.9521 - loss: 0.1819 - val_accuracy: 0.9600 - val_loss: 0.1771
Epoch 9/100
3375/3375            10s 2ms/step -
accuracy: 0.9515 - loss: 0.1828 - val_accuracy: 0.9538 - val_loss: 0.1932
Epoch 9: early stopping
Restoring model weights from the end of the best epoch: 6.
Training with batch size 16 and learning rate 0.001
Epoch 1/100
3375/3375            7s 2ms/step -
accuracy: 0.7944 - loss: 0.6861 - val_accuracy: 0.9552 - val_loss: 0.1680
Epoch 2/100
3375/3375            10s 2ms/step -
accuracy: 0.9397 - loss: 0.2025 - val_accuracy: 0.9588 - val_loss: 0.1385
Epoch 3/100
3375/3375            10s 2ms/step -
accuracy: 0.9540 - loss: 0.1533 - val_accuracy: 0.9647 - val_loss: 0.1232
Epoch 4/100
3375/3375            6s 2ms/step -
accuracy: 0.9627 - loss: 0.1220 - val_accuracy: 0.9680 - val_loss: 0.1139
Epoch 5/100
3375/3375            10s 2ms/step -
accuracy: 0.9680 - loss: 0.1056 - val_accuracy: 0.9673 - val_loss: 0.1120
Epoch 6/100
3375/3375            10s 2ms/step -
accuracy: 0.9712 - loss: 0.0926 - val_accuracy: 0.9730 - val_loss: 0.1033
Epoch 7/100
3375/3375            6s 2ms/step -
accuracy: 0.9739 - loss: 0.0850 - val_accuracy: 0.9687 - val_loss: 0.1048
Epoch 8/100
```

```
3375/3375 6s 2ms/step -
accuracy: 0.9774 - loss: 0.0722 - val_accuracy: 0.9723 - val_loss: 0.1005
Epoch 9/100
3375/3375 6s 2ms/step -
accuracy: 0.9784 - loss: 0.0683 - val_accuracy: 0.9705 - val_loss: 0.1116
Epoch 10/100
3375/3375 10s 2ms/step -
accuracy: 0.9798 - loss: 0.0620 - val_accuracy: 0.9698 - val_loss: 0.1012
Epoch 11/100
3375/3375 11s 2ms/step -
accuracy: 0.9810 - loss: 0.0562 - val_accuracy: 0.9708 - val_loss: 0.1100
Epoch 11: early stopping
Restoring model weights from the end of the best epoch: 8.
Training with batch size 16 and learning rate 0.0001
Epoch 1/100
3375/3375 7s 2ms/step -
accuracy: 0.5906 - loss: 1.3346 - val_accuracy: 0.9173 - val_loss: 0.3384
Epoch 2/100
3375/3375 11s 2ms/step -
accuracy: 0.8985 - loss: 0.3770 - val_accuracy: 0.9352 - val_loss: 0.2546
Epoch 3/100
3375/3375 6s 2ms/step -
accuracy: 0.9168 - loss: 0.3005 - val_accuracy: 0.9407 - val_loss: 0.2211
Epoch 4/100
3375/3375 10s 2ms/step -
accuracy: 0.9248 - loss: 0.2652 - val_accuracy: 0.9445 - val_loss: 0.2018
Epoch 5/100
3375/3375 6s 2ms/step -
accuracy: 0.9330 - loss: 0.2371 - val_accuracy: 0.9482 - val_loss: 0.1890
Epoch 6/100
3375/3375 6s 2ms/step -
accuracy: 0.9377 - loss: 0.2219 - val_accuracy: 0.9523 - val_loss: 0.1766
Epoch 7/100
3375/3375 6s 2ms/step -
accuracy: 0.9409 - loss: 0.2069 - val_accuracy: 0.9545 - val_loss: 0.1661
Epoch 8/100
3375/3375 6s 2ms/step -
accuracy: 0.9453 - loss: 0.1932 - val_accuracy: 0.9565 - val_loss: 0.1612
Epoch 9/100
3375/3375 6s 2ms/step -
accuracy: 0.9461 - loss: 0.1894 - val_accuracy: 0.9570 - val_loss: 0.1540
Epoch 10/100
3375/3375 10s 2ms/step -
accuracy: 0.9489 - loss: 0.1757 - val_accuracy: 0.9590 - val_loss: 0.1467
Epoch 11/100
3375/3375 5s 2ms/step -
accuracy: 0.9529 - loss: 0.1641 - val_accuracy: 0.9610 - val_loss: 0.1443
Epoch 12/100
```

```
3375/3375            6s 2ms/step -
accuracy: 0.9533 - loss: 0.1615 - val_accuracy: 0.9613 - val_loss: 0.1387
Epoch 13/100
3375/3375            6s 2ms/step -
accuracy: 0.9560 - loss: 0.1531 - val_accuracy: 0.9610 - val_loss: 0.1373
Epoch 14/100
3375/3375            6s 2ms/step -
accuracy: 0.9578 - loss: 0.1493 - val_accuracy: 0.9630 - val_loss: 0.1314
Epoch 15/100
3375/3375            10s 2ms/step -
accuracy: 0.9611 - loss: 0.1377 - val_accuracy: 0.9638 - val_loss: 0.1289
Epoch 16/100
3375/3375            10s 2ms/step -
accuracy: 0.9615 - loss: 0.1328 - val_accuracy: 0.9633 - val_loss: 0.1304
Epoch 17/100
3375/3375            10s 2ms/step -
accuracy: 0.9633 - loss: 0.1255 - val_accuracy: 0.9640 - val_loss: 0.1271
Epoch 18/100
3375/3375            6s 2ms/step -
accuracy: 0.9632 - loss: 0.1294 - val_accuracy: 0.9658 - val_loss: 0.1241
Epoch 19/100
3375/3375            6s 2ms/step -
accuracy: 0.9646 - loss: 0.1231 - val_accuracy: 0.9653 - val_loss: 0.1235
Epoch 20/100
3375/3375            10s 2ms/step -
accuracy: 0.9668 - loss: 0.1154 - val_accuracy: 0.9665 - val_loss: 0.1217
Epoch 21/100
3375/3375            10s 2ms/step -
accuracy: 0.9684 - loss: 0.1089 - val_accuracy: 0.9662 - val_loss: 0.1233
Epoch 22/100
3375/3375            6s 2ms/step -
accuracy: 0.9676 - loss: 0.1081 - val_accuracy: 0.9662 - val_loss: 0.1197
Epoch 23/100
3375/3375            10s 2ms/step -
accuracy: 0.9687 - loss: 0.1092 - val_accuracy: 0.9670 - val_loss: 0.1202
Epoch 24/100
3375/3375            10s 2ms/step -
accuracy: 0.9699 - loss: 0.1039 - val_accuracy: 0.9670 - val_loss: 0.1179
Epoch 25/100
3375/3375            10s 2ms/step -
accuracy: 0.9717 - loss: 0.1001 - val_accuracy: 0.9675 - val_loss: 0.1175
Epoch 26/100
3375/3375            10s 2ms/step -
accuracy: 0.9730 - loss: 0.0940 - val_accuracy: 0.9678 - val_loss: 0.1164
Epoch 27/100
3375/3375            6s 2ms/step -
accuracy: 0.9714 - loss: 0.0980 - val_accuracy: 0.9678 - val_loss: 0.1141
Epoch 28/100
```

```
3375/3375            6s 2ms/step -
accuracy: 0.9723 - loss: 0.0940 - val_accuracy: 0.9683 - val_loss: 0.1161
Epoch 29/100
3375/3375            10s 2ms/step -
accuracy: 0.9719 - loss: 0.0921 - val_accuracy: 0.9690 - val_loss: 0.1146
Epoch 30/100
3375/3375            6s 2ms/step -
accuracy: 0.9744 - loss: 0.0875 - val_accuracy: 0.9678 - val_loss: 0.1178
Epoch 30: early stopping
Restoring model weights from the end of the best epoch: 27.
Training with batch size 16 and learning rate 1e-05
Epoch 1/100
3375/3375            6s 2ms/step -
accuracy: 0.2245 - loss: 2.1659 - val_accuracy: 0.5757 - val_loss: 1.6212
Epoch 2/100
3375/3375            11s 2ms/step -
accuracy: 0.6095 - loss: 1.5118 - val_accuracy: 0.7347 - val_loss: 1.1425
Epoch 3/100
3375/3375            10s 2ms/step -
accuracy: 0.7328 - loss: 1.1096 - val_accuracy: 0.8160 - val_loss: 0.8452
Epoch 4/100
3375/3375            10s 2ms/step -
accuracy: 0.7972 - loss: 0.8527 - val_accuracy: 0.8535 - val_loss: 0.6679
Epoch 5/100
3375/3375            10s 2ms/step -
accuracy: 0.8317 - loss: 0.7000 - val_accuracy: 0.8748 - val_loss: 0.5601
Epoch 6/100
3375/3375            10s 2ms/step -
accuracy: 0.8511 - loss: 0.6081 - val_accuracy: 0.8893 - val_loss: 0.4903
Epoch 7/100
3375/3375            6s 2ms/step -
accuracy: 0.8635 - loss: 0.5415 - val_accuracy: 0.8980 - val_loss: 0.4427
Epoch 8/100
3375/3375            6s 2ms/step -
accuracy: 0.8714 - loss: 0.5019 - val_accuracy: 0.9037 - val_loss: 0.4084
Epoch 9/100
3375/3375            10s 2ms/step -
accuracy: 0.8788 - loss: 0.4670 - val_accuracy: 0.9073 - val_loss: 0.3824
Epoch 10/100
3375/3375            6s 2ms/step -
accuracy: 0.8805 - loss: 0.4465 - val_accuracy: 0.9113 - val_loss: 0.3620
Epoch 11/100
3375/3375            10s 2ms/step -
accuracy: 0.8883 - loss: 0.4181 - val_accuracy: 0.9142 - val_loss: 0.3457
Epoch 12/100
3375/3375            10s 2ms/step -
accuracy: 0.8910 - loss: 0.4051 - val_accuracy: 0.9165 - val_loss: 0.3322
Epoch 13/100
```

```
3375/3375              6s 2ms/step -
accuracy: 0.8930 - loss: 0.3923 - val_accuracy: 0.9175 - val_loss: 0.3209
Epoch 14/100
3375/3375              6s 2ms/step -
accuracy: 0.8935 - loss: 0.3835 - val_accuracy: 0.9192 - val_loss: 0.3114
Epoch 15/100
3375/3375              6s 2ms/step -
accuracy: 0.9001 - loss: 0.3677 - val_accuracy: 0.9207 - val_loss: 0.3031
Epoch 16/100
3375/3375              10s 2ms/step -
accuracy: 0.9017 - loss: 0.3568 - val_accuracy: 0.9205 - val_loss: 0.2955
Epoch 17/100
3375/3375              6s 2ms/step -
accuracy: 0.9037 - loss: 0.3500 - val_accuracy: 0.9222 - val_loss: 0.2889
Epoch 18/100
3375/3375              6s 2ms/step -
accuracy: 0.9050 - loss: 0.3417 - val_accuracy: 0.9238 - val_loss: 0.2833
Epoch 19/100
3375/3375              6s 2ms/step -
accuracy: 0.9088 - loss: 0.3298 - val_accuracy: 0.9248 - val_loss: 0.2781
Epoch 20/100
3375/3375              6s 2ms/step -
accuracy: 0.9059 - loss: 0.3340 - val_accuracy: 0.9258 - val_loss: 0.2727
Epoch 21/100
3375/3375              6s 2ms/step -
accuracy: 0.9092 - loss: 0.3242 - val_accuracy: 0.9265 - val_loss: 0.2684
Epoch 22/100
3375/3375              10s 2ms/step -
accuracy: 0.9098 - loss: 0.3206 - val_accuracy: 0.9290 - val_loss: 0.2642
Epoch 23/100
3375/3375              10s 2ms/step -
accuracy: 0.9104 - loss: 0.3171 - val_accuracy: 0.9292 - val_loss: 0.2607
Epoch 24/100
3375/3375              10s 2ms/step -
accuracy: 0.9118 - loss: 0.3108 - val_accuracy: 0.9302 - val_loss: 0.2570
Epoch 25/100
3375/3375              11s 2ms/step -
accuracy: 0.9145 - loss: 0.3069 - val_accuracy: 0.9297 - val_loss: 0.2540
Epoch 26/100
3375/3375              5s 2ms/step -
accuracy: 0.9141 - loss: 0.3030 - val_accuracy: 0.9300 - val_loss: 0.2509
Epoch 27/100
3375/3375              6s 2ms/step -
accuracy: 0.9145 - loss: 0.3000 - val_accuracy: 0.9312 - val_loss: 0.2476
Epoch 28/100
3375/3375              6s 2ms/step -
accuracy: 0.9153 - loss: 0.2969 - val_accuracy: 0.9310 - val_loss: 0.2453
Epoch 29/100
```

```
3375/3375             6s 2ms/step -
accuracy: 0.9171 - loss: 0.2917 - val_accuracy: 0.9333 - val_loss: 0.2426
Epoch 30/100
3375/3375             6s 2ms/step -
accuracy: 0.9177 - loss: 0.2892 - val_accuracy: 0.9333 - val_loss: 0.2401
Epoch 31/100
3375/3375             10s 2ms/step -
accuracy: 0.9187 - loss: 0.2937 - val_accuracy: 0.9338 - val_loss: 0.2375
Epoch 32/100
3375/3375             6s 2ms/step -
accuracy: 0.9196 - loss: 0.2842 - val_accuracy: 0.9340 - val_loss: 0.2358
Epoch 33/100
3375/3375             6s 2ms/step -
accuracy: 0.9195 - loss: 0.2821 - val_accuracy: 0.9350 - val_loss: 0.2333
Epoch 34/100
3375/3375             6s 2ms/step -
accuracy: 0.9210 - loss: 0.2758 - val_accuracy: 0.9350 - val_loss: 0.2312
Epoch 35/100
3375/3375             6s 2ms/step -
accuracy: 0.9217 - loss: 0.2777 - val_accuracy: 0.9357 - val_loss: 0.2295
Epoch 36/100
3375/3375             10s 2ms/step -
accuracy: 0.9222 - loss: 0.2755 - val_accuracy: 0.9382 - val_loss: 0.2277
Epoch 37/100
3375/3375             6s 2ms/step -
accuracy: 0.9234 - loss: 0.2687 - val_accuracy: 0.9382 - val_loss: 0.2257
Epoch 38/100
3375/3375             10s 2ms/step -
accuracy: 0.9239 - loss: 0.2709 - val_accuracy: 0.9377 - val_loss: 0.2240
Epoch 39/100
3375/3375             5s 2ms/step -
accuracy: 0.9252 - loss: 0.2664 - val_accuracy: 0.9365 - val_loss: 0.2230
Epoch 40/100
3375/3375             11s 2ms/step -
accuracy: 0.9239 - loss: 0.2642 - val_accuracy: 0.9385 - val_loss: 0.2206
Epoch 41/100
3375/3375             10s 2ms/step -
accuracy: 0.9252 - loss: 0.2606 - val_accuracy: 0.9378 - val_loss: 0.2194
Epoch 42/100
3375/3375             6s 2ms/step -
accuracy: 0.9273 - loss: 0.2538 - val_accuracy: 0.9392 - val_loss: 0.2174
Epoch 43/100
3375/3375             10s 2ms/step -
accuracy: 0.9268 - loss: 0.2596 - val_accuracy: 0.9385 - val_loss: 0.2165
Epoch 44/100
3375/3375             10s 2ms/step -
accuracy: 0.9277 - loss: 0.2505 - val_accuracy: 0.9395 - val_loss: 0.2149
Epoch 45/100
```

```
3375/3375              11s 2ms/step -
accuracy: 0.9266 - loss: 0.2586 - val_accuracy: 0.9392 - val_loss: 0.2133
Epoch 46/100
3375/3375              6s 2ms/step -
accuracy: 0.9267 - loss: 0.2541 - val_accuracy: 0.9400 - val_loss: 0.2123
Epoch 47/100
3375/3375              10s 2ms/step -
accuracy: 0.9267 - loss: 0.2596 - val_accuracy: 0.9403 - val_loss: 0.2103
Epoch 48/100
3375/3375              6s 2ms/step -
accuracy: 0.9265 - loss: 0.2528 - val_accuracy: 0.9405 - val_loss: 0.2094
Epoch 49/100
3375/3375              10s 2ms/step -
accuracy: 0.9278 - loss: 0.2530 - val_accuracy: 0.9412 - val_loss: 0.2077
Epoch 50/100
3375/3375              10s 2ms/step -
accuracy: 0.9308 - loss: 0.2431 - val_accuracy: 0.9413 - val_loss: 0.2065
Epoch 51/100
3375/3375              6s 2ms/step -
accuracy: 0.9314 - loss: 0.2454 - val_accuracy: 0.9420 - val_loss: 0.2055
Epoch 52/100
3375/3375              6s 2ms/step -
accuracy: 0.9304 - loss: 0.2458 - val_accuracy: 0.9420 - val_loss: 0.2046
Epoch 53/100
3375/3375              10s 2ms/step -
accuracy: 0.9310 - loss: 0.2450 - val_accuracy: 0.9425 - val_loss: 0.2028
Epoch 54/100
3375/3375              11s 2ms/step -
accuracy: 0.9313 - loss: 0.2452 - val_accuracy: 0.9427 - val_loss: 0.2018
Epoch 55/100
3375/3375              10s 2ms/step -
accuracy: 0.9322 - loss: 0.2398 - val_accuracy: 0.9435 - val_loss: 0.2007
Epoch 56/100
3375/3375              6s 2ms/step -
accuracy: 0.9315 - loss: 0.2359 - val_accuracy: 0.9448 - val_loss: 0.1993
Epoch 57/100
3375/3375              10s 2ms/step -
accuracy: 0.9349 - loss: 0.2355 - val_accuracy: 0.9453 - val_loss: 0.1981
Epoch 58/100
3375/3375              6s 2ms/step -
accuracy: 0.9337 - loss: 0.2337 - val_accuracy: 0.9457 - val_loss: 0.1972
Epoch 59/100
3375/3375              10s 2ms/step -
accuracy: 0.9331 - loss: 0.2327 - val_accuracy: 0.9452 - val_loss: 0.1960
Epoch 60/100
3375/3375              6s 2ms/step -
accuracy: 0.9349 - loss: 0.2303 - val_accuracy: 0.9458 - val_loss: 0.1952
Epoch 61/100
```

```
3375/3375            10s 2ms/step -
accuracy: 0.9350 - loss: 0.2290 - val_accuracy: 0.9462 - val_loss: 0.1941
Epoch 62/100
3375/3375            11s 2ms/step -
accuracy: 0.9365 - loss: 0.2210 - val_accuracy: 0.9468 - val_loss: 0.1929
Epoch 63/100
3375/3375            10s 2ms/step -
accuracy: 0.9342 - loss: 0.2288 - val_accuracy: 0.9480 - val_loss: 0.1916
Epoch 64/100
3375/3375            10s 2ms/step -
accuracy: 0.9354 - loss: 0.2258 - val_accuracy: 0.9472 - val_loss: 0.1904
Epoch 65/100
3375/3375            10s 2ms/step -
accuracy: 0.9369 - loss: 0.2200 - val_accuracy: 0.9480 - val_loss: 0.1901
Epoch 66/100
3375/3375            10s 2ms/step -
accuracy: 0.9383 - loss: 0.2156 - val_accuracy: 0.9488 - val_loss: 0.1886
Epoch 67/100
3375/3375            10s 2ms/step -
accuracy: 0.9381 - loss: 0.2176 - val_accuracy: 0.9475 - val_loss: 0.1879
Epoch 68/100
3375/3375            10s 2ms/step -
accuracy: 0.9352 - loss: 0.2255 - val_accuracy: 0.9490 - val_loss: 0.1865
Epoch 69/100
3375/3375            6s 2ms/step -
accuracy: 0.9372 - loss: 0.2173 - val_accuracy: 0.9485 - val_loss: 0.1859
Epoch 70/100
3375/3375            6s 2ms/step -
accuracy: 0.9379 - loss: 0.2160 - val_accuracy: 0.9485 - val_loss: 0.1850
Epoch 71/100
3375/3375            6s 2ms/step -
accuracy: 0.9357 - loss: 0.2186 - val_accuracy: 0.9483 - val_loss: 0.1841
Epoch 72/100
3375/3375            5s 2ms/step -
accuracy: 0.9406 - loss: 0.2072 - val_accuracy: 0.9482 - val_loss: 0.1837
Epoch 73/100
3375/3375            11s 2ms/step -
accuracy: 0.9395 - loss: 0.2100 - val_accuracy: 0.9488 - val_loss: 0.1824
Epoch 74/100
3375/3375            6s 2ms/step -
accuracy: 0.9376 - loss: 0.2134 - val_accuracy: 0.9498 - val_loss: 0.1814
Epoch 75/100
3375/3375            10s 2ms/step -
accuracy: 0.9410 - loss: 0.2050 - val_accuracy: 0.9493 - val_loss: 0.1804
Epoch 76/100
3375/3375            6s 2ms/step -
accuracy: 0.9418 - loss: 0.2040 - val_accuracy: 0.9497 - val_loss: 0.1796
Epoch 77/100
```

```
3375/3375              6s 2ms/step -
accuracy: 0.9406 - loss: 0.2064 - val_accuracy: 0.9490 - val_loss: 0.1795
Epoch 78/100
3375/3375              6s 2ms/step -
accuracy: 0.9418 - loss: 0.2050 - val_accuracy: 0.9490 - val_loss: 0.1780
Epoch 79/100
3375/3375              10s 2ms/step -
accuracy: 0.9412 - loss: 0.2032 - val_accuracy: 0.9497 - val_loss: 0.1772
Epoch 80/100
3375/3375              10s 2ms/step -
accuracy: 0.9410 - loss: 0.2052 - val_accuracy: 0.9500 - val_loss: 0.1763
Epoch 81/100
3375/3375              6s 2ms/step -
accuracy: 0.9429 - loss: 0.2002 - val_accuracy: 0.9505 - val_loss: 0.1759
Epoch 82/100
3375/3375              10s 2ms/step -
accuracy: 0.9425 - loss: 0.2023 - val_accuracy: 0.9500 - val_loss: 0.1751
Epoch 83/100
3375/3375              6s 2ms/step -
accuracy: 0.9422 - loss: 0.1977 - val_accuracy: 0.9502 - val_loss: 0.1740
Epoch 84/100
3375/3375              6s 2ms/step -
accuracy: 0.9435 - loss: 0.1961 - val_accuracy: 0.9513 - val_loss: 0.1734
Epoch 85/100
3375/3375              6s 2ms/step -
accuracy: 0.9428 - loss: 0.1994 - val_accuracy: 0.9510 - val_loss: 0.1729
Epoch 86/100
3375/3375              6s 2ms/step -
accuracy: 0.9437 - loss: 0.1949 - val_accuracy: 0.9513 - val_loss: 0.1720
Epoch 87/100
3375/3375              10s 2ms/step -
accuracy: 0.9427 - loss: 0.1960 - val_accuracy: 0.9513 - val_loss: 0.1711
Epoch 88/100
3375/3375              10s 2ms/step -
accuracy: 0.9443 - loss: 0.1969 - val_accuracy: 0.9520 - val_loss: 0.1706
Epoch 89/100
3375/3375              6s 2ms/step -
accuracy: 0.9445 - loss: 0.1926 - val_accuracy: 0.9515 - val_loss: 0.1695
Epoch 90/100
3375/3375              10s 2ms/step -
accuracy: 0.9460 - loss: 0.1884 - val_accuracy: 0.9522 - val_loss: 0.1691
Epoch 91/100
3375/3375              10s 2ms/step -
accuracy: 0.9455 - loss: 0.1897 - val_accuracy: 0.9518 - val_loss: 0.1687
Epoch 92/100
3375/3375              10s 2ms/step -
accuracy: 0.9447 - loss: 0.1943 - val_accuracy: 0.9520 - val_loss: 0.1683
Epoch 93/100
```

```
3375/3375            6s 2ms/step -
accuracy: 0.9441 - loss: 0.1941 - val_accuracy: 0.9528 - val_loss: 0.1675
Epoch 94/100
3375/3375            6s 2ms/step -
accuracy: 0.9451 - loss: 0.1901 - val_accuracy: 0.9527 - val_loss: 0.1663
Epoch 95/100
3375/3375            10s 2ms/step -
accuracy: 0.9461 - loss: 0.1876 - val_accuracy: 0.9530 - val_loss: 0.1658
Epoch 96/100
3375/3375            6s 2ms/step -
accuracy: 0.9450 - loss: 0.1904 - val_accuracy: 0.9535 - val_loss: 0.1652
Epoch 97/100
3375/3375            10s 2ms/step -
accuracy: 0.9463 - loss: 0.1879 - val_accuracy: 0.9527 - val_loss: 0.1646
Epoch 98/100
3375/3375            6s 2ms/step -
accuracy: 0.9483 - loss: 0.1843 - val_accuracy: 0.9523 - val_loss: 0.1641
Epoch 99/100
3375/3375            6s 2ms/step -
accuracy: 0.9469 - loss: 0.1865 - val_accuracy: 0.9527 - val_loss: 0.1639
Epoch 100/100
3375/3375            10s 2ms/step -
accuracy: 0.9467 - loss: 0.1844 - val_accuracy: 0.9537 - val_loss: 0.1631
Restoring model weights from the end of the best epoch: 100.
Training with batch size 32 and learning rate 0.01
Epoch 1/100
1688/1688            4s 2ms/step -
accuracy: 0.8554 - loss: 0.4671 - val_accuracy: 0.9340 - val_loss: 0.2196
Epoch 2/100
1688/1688            5s 2ms/step -
accuracy: 0.9379 - loss: 0.2108 - val_accuracy: 0.9503 - val_loss: 0.1874
Epoch 3/100
1688/1688            3s 2ms/step -
accuracy: 0.9465 - loss: 0.1868 - val_accuracy: 0.9575 - val_loss: 0.1558
Epoch 4/100
1688/1688            4s 2ms/step -
accuracy: 0.9477 - loss: 0.1812 - val_accuracy: 0.9553 - val_loss: 0.1687
Epoch 5/100
1688/1688            3s 2ms/step -
accuracy: 0.9522 - loss: 0.1627 - val_accuracy: 0.9602 - val_loss: 0.1454
Epoch 6/100
1688/1688            3s 2ms/step -
accuracy: 0.9564 - loss: 0.1545 - val_accuracy: 0.9573 - val_loss: 0.1534
Epoch 7/100
1688/1688            5s 2ms/step -
accuracy: 0.9581 - loss: 0.1487 - val_accuracy: 0.9590 - val_loss: 0.1443
Epoch 8/100
1688/1688            3s 2ms/step -
```

```
accuracy: 0.9592 - loss: 0.1462 - val_accuracy: 0.9587 - val_loss: 0.1570
Epoch 9/100
1688/1688            5s 2ms/step -
accuracy: 0.9608 - loss: 0.1402 - val_accuracy: 0.9582 - val_loss: 0.1574
Epoch 10/100
1688/1688            4s 2ms/step -
accuracy: 0.9627 - loss: 0.1354 - val_accuracy: 0.9667 - val_loss: 0.1328
Epoch 11/100
1688/1688            5s 2ms/step -
accuracy: 0.9623 - loss: 0.1331 - val_accuracy: 0.9640 - val_loss: 0.1452
Epoch 12/100
1688/1688            3s 2ms/step -
accuracy: 0.9623 - loss: 0.1326 - val_accuracy: 0.9590 - val_loss: 0.1815
Epoch 13/100
1688/1688            6s 2ms/step -
accuracy: 0.9619 - loss: 0.1385 - val_accuracy: 0.9535 - val_loss: 0.1928
Epoch 13: early stopping
Restoring model weights from the end of the best epoch: 10.
Training with batch size 32 and learning rate 0.001
Epoch 1/100
1688/1688            4s 2ms/step -
accuracy: 0.7850 - loss: 0.7054 - val_accuracy: 0.9493 - val_loss: 0.1786
Epoch 2/100
1688/1688            6s 2ms/step -
accuracy: 0.9422 - loss: 0.1968 - val_accuracy: 0.9597 - val_loss: 0.1457
Epoch 3/100
1688/1688            4s 2ms/step -
accuracy: 0.9551 - loss: 0.1534 - val_accuracy: 0.9638 - val_loss: 0.1319
Epoch 4/100
1688/1688            5s 2ms/step -
accuracy: 0.9618 - loss: 0.1287 - val_accuracy: 0.9688 - val_loss: 0.1141
Epoch 5/100
1688/1688            4s 2ms/step -
accuracy: 0.9668 - loss: 0.1095 - val_accuracy: 0.9643 - val_loss: 0.1144
Epoch 6/100
1688/1688            3s 2ms/step -
accuracy: 0.9716 - loss: 0.0893 - val_accuracy: 0.9658 - val_loss: 0.1151
Epoch 7/100
1688/1688            3s 2ms/step -
accuracy: 0.9731 - loss: 0.0860 - val_accuracy: 0.9693 - val_loss: 0.1082
Epoch 8/100
1688/1688            5s 2ms/step -
accuracy: 0.9757 - loss: 0.0780 - val_accuracy: 0.9683 - val_loss: 0.1093
Epoch 9/100
1688/1688            3s 2ms/step -
accuracy: 0.9792 - loss: 0.0671 - val_accuracy: 0.9647 - val_loss: 0.1225
Epoch 10/100
1688/1688            3s 2ms/step -
```

```
accuracy: 0.9805 - loss: 0.0649 - val_accuracy: 0.9683 - val_loss: 0.1134
Epoch 10: early stopping
Restoring model weights from the end of the best epoch: 7.
Training with batch size 32 and learning rate 0.0001
Epoch 1/100
1688/1688            4s 2ms/step -
accuracy: 0.4932 - loss: 1.6697 - val_accuracy: 0.8653 - val_loss: 0.5202
Epoch 2/100
1688/1688            5s 2ms/step -
accuracy: 0.8603 - loss: 0.5166 - val_accuracy: 0.9180 - val_loss: 0.3119
Epoch 3/100
1688/1688            4s 2ms/step -
accuracy: 0.8995 - loss: 0.3582 - val_accuracy: 0.9322 - val_loss: 0.2480
Epoch 4/100
1688/1688            5s 2ms/step -
accuracy: 0.9145 - loss: 0.2948 - val_accuracy: 0.9397 - val_loss: 0.2187
Epoch 5/100
1688/1688            3s 2ms/step -
accuracy: 0.9251 - loss: 0.2605 - val_accuracy: 0.9443 - val_loss: 0.1997
Epoch 6/100
1688/1688            4s 2ms/step -
accuracy: 0.9320 - loss: 0.2368 - val_accuracy: 0.9490 - val_loss: 0.1876
Epoch 7/100
1688/1688            3s 2ms/step -
accuracy: 0.9383 - loss: 0.2135 - val_accuracy: 0.9538 - val_loss: 0.1763
Epoch 8/100
1688/1688            3s 2ms/step -
accuracy: 0.9395 - loss: 0.2100 - val_accuracy: 0.9550 - val_loss: 0.1673
Epoch 9/100
1688/1688            4s 2ms/step -
accuracy: 0.9436 - loss: 0.1980 - val_accuracy: 0.9573 - val_loss: 0.1619
Epoch 10/100
1688/1688            3s 2ms/step -
accuracy: 0.9453 - loss: 0.1891 - val_accuracy: 0.9582 - val_loss: 0.1555
Epoch 11/100
1688/1688            5s 2ms/step -
accuracy: 0.9492 - loss: 0.1771 - val_accuracy: 0.9598 - val_loss: 0.1520
Epoch 12/100
1688/1688            4s 2ms/step -
accuracy: 0.9519 - loss: 0.1706 - val_accuracy: 0.9615 - val_loss: 0.1446
Epoch 13/100
1688/1688            5s 2ms/step -
accuracy: 0.9538 - loss: 0.1585 - val_accuracy: 0.9637 - val_loss: 0.1415
Epoch 14/100
1688/1688            5s 2ms/step -
accuracy: 0.9557 - loss: 0.1558 - val_accuracy: 0.9638 - val_loss: 0.1385
Epoch 15/100
1688/1688            5s 2ms/step -
```

```
accuracy: 0.9559 - loss: 0.1513 - val_accuracy: 0.9633 - val_loss: 0.1352
Epoch 16/100
1688/1688            3s 2ms/step -
accuracy: 0.9573 - loss: 0.1449 - val_accuracy: 0.9652 - val_loss: 0.1314
Epoch 17/100
1688/1688            5s 2ms/step -
accuracy: 0.9601 - loss: 0.1401 - val_accuracy: 0.9673 - val_loss: 0.1302
Epoch 18/100
1688/1688            5s 2ms/step -
accuracy: 0.9614 - loss: 0.1354 - val_accuracy: 0.9673 - val_loss: 0.1283
Epoch 19/100
1688/1688            6s 2ms/step -
accuracy: 0.9609 - loss: 0.1321 - val_accuracy: 0.9673 - val_loss: 0.1257
Epoch 20/100
1688/1688            3s 2ms/step -
accuracy: 0.9621 - loss: 0.1325 - val_accuracy: 0.9675 - val_loss: 0.1238
Epoch 21/100
1688/1688            5s 2ms/step -
accuracy: 0.9641 - loss: 0.1231 - val_accuracy: 0.9677 - val_loss: 0.1232
Epoch 22/100
1688/1688            6s 2ms/step -
accuracy: 0.9628 - loss: 0.1255 - val_accuracy: 0.9677 - val_loss: 0.1204
Epoch 23/100
1688/1688            3s 2ms/step -
accuracy: 0.9646 - loss: 0.1203 - val_accuracy: 0.9670 - val_loss: 0.1210
Epoch 24/100
1688/1688            3s 2ms/step -
accuracy: 0.9660 - loss: 0.1139 - val_accuracy: 0.9670 - val_loss: 0.1188
Epoch 25/100
1688/1688            4s 2ms/step -
accuracy: 0.9669 - loss: 0.1142 - val_accuracy: 0.9680 - val_loss: 0.1170
Epoch 26/100
1688/1688            4s 2ms/step -
accuracy: 0.9697 - loss: 0.1065 - val_accuracy: 0.9678 - val_loss: 0.1150
Epoch 27/100
1688/1688            6s 2ms/step -
accuracy: 0.9697 - loss: 0.1059 - val_accuracy: 0.9677 - val_loss: 0.1153
Epoch 28/100
1688/1688            5s 2ms/step -
accuracy: 0.9684 - loss: 0.1044 - val_accuracy: 0.9670 - val_loss: 0.1161
Epoch 29/100
1688/1688            3s 2ms/step -
accuracy: 0.9701 - loss: 0.1022 - val_accuracy: 0.9675 - val_loss: 0.1142
Epoch 30/100
1688/1688            4s 2ms/step -
accuracy: 0.9713 - loss: 0.0976 - val_accuracy: 0.9677 - val_loss: 0.1131
Epoch 31/100
1688/1688            3s 2ms/step -
```

```
accuracy: 0.9716 - loss: 0.0979 - val_accuracy: 0.9680 - val_loss: 0.1125
Epoch 32/100
1688/1688            5s 2ms/step -
accuracy: 0.9724 - loss: 0.0948 - val_accuracy: 0.9678 - val_loss: 0.1116
Epoch 33/100
1688/1688            6s 2ms/step -
accuracy: 0.9725 - loss: 0.0936 - val_accuracy: 0.9680 - val_loss: 0.1122
Epoch 34/100
1688/1688            4s 2ms/step -
accuracy: 0.9728 - loss: 0.0906 - val_accuracy: 0.9690 - val_loss: 0.1106
Epoch 35/100
1688/1688            6s 2ms/step -
accuracy: 0.9752 - loss: 0.0870 - val_accuracy: 0.9672 - val_loss: 0.1102
Epoch 36/100
1688/1688            3s 2ms/step -
accuracy: 0.9741 - loss: 0.0897 - val_accuracy: 0.9693 - val_loss: 0.1100
Epoch 37/100
1688/1688            3s 2ms/step -
accuracy: 0.9727 - loss: 0.0899 - val_accuracy: 0.9687 - val_loss: 0.1107
Epoch 38/100
1688/1688            3s 2ms/step -
accuracy: 0.9755 - loss: 0.0849 - val_accuracy: 0.9693 - val_loss: 0.1082
Epoch 39/100
1688/1688            4s 2ms/step -
accuracy: 0.9776 - loss: 0.0821 - val_accuracy: 0.9705 - val_loss: 0.1085
Epoch 40/100
1688/1688            4s 2ms/step -
accuracy: 0.9761 - loss: 0.0830 - val_accuracy: 0.9702 - val_loss: 0.1076
Epoch 41/100
1688/1688            6s 2ms/step -
accuracy: 0.9773 - loss: 0.0781 - val_accuracy: 0.9698 - val_loss: 0.1076
Epoch 42/100
1688/1688            3s 2ms/step -
accuracy: 0.9774 - loss: 0.0792 - val_accuracy: 0.9690 - val_loss: 0.1087
Epoch 43/100
1688/1688            3s 2ms/step -
accuracy: 0.9769 - loss: 0.0813 - val_accuracy: 0.9710 - val_loss: 0.1068
Epoch 44/100
1688/1688            3s 2ms/step -
accuracy: 0.9783 - loss: 0.0774 - val_accuracy: 0.9707 - val_loss: 0.1070
Epoch 45/100
1688/1688            5s 2ms/step -
accuracy: 0.9781 - loss: 0.0762 - val_accuracy: 0.9702 - val_loss: 0.1081
Epoch 46/100
1688/1688            5s 2ms/step -
accuracy: 0.9792 - loss: 0.0750 - val_accuracy: 0.9697 - val_loss: 0.1076
Epoch 46: early stopping
Restoring model weights from the end of the best epoch: 43.
```

```
Training with batch size 32 and learning rate 1e-05
Epoch 1/100
1688/1688            4s 2ms/step -
accuracy: 0.1859 - loss: 2.2611 - val_accuracy: 0.4627 - val_loss: 1.9556
Epoch 2/100
1688/1688            5s 2ms/step -
accuracy: 0.5143 - loss: 1.8525 - val_accuracy: 0.6697 - val_loss: 1.5229
Epoch 3/100
1688/1688            6s 2ms/step -
accuracy: 0.6635 - loss: 1.4635 - val_accuracy: 0.7660 - val_loss: 1.1992
Epoch 4/100
1688/1688            4s 2ms/step -
accuracy: 0.7390 - loss: 1.1777 - val_accuracy: 0.8185 - val_loss: 0.9495
Epoch 5/100
1688/1688            3s 2ms/step -
accuracy: 0.7882 - loss: 0.9635 - val_accuracy: 0.8507 - val_loss: 0.7750
Epoch 6/100
1688/1688            4s 2ms/step -
accuracy: 0.8151 - loss: 0.8107 - val_accuracy: 0.8718 - val_loss: 0.6566
Epoch 7/100
1688/1688            5s 2ms/step -
accuracy: 0.8385 - loss: 0.6997 - val_accuracy: 0.8858 - val_loss: 0.5730
Epoch 8/100
1688/1688            5s 2ms/step -
accuracy: 0.8525 - loss: 0.6227 - val_accuracy: 0.8935 - val_loss: 0.5128
Epoch 9/100
1688/1688            5s 2ms/step -
accuracy: 0.8577 - loss: 0.5780 - val_accuracy: 0.9003 - val_loss: 0.4674
Epoch 10/100
1688/1688            3s 2ms/step -
accuracy: 0.8697 - loss: 0.5311 - val_accuracy: 0.9058 - val_loss: 0.4322
Epoch 11/100
1688/1688            6s 2ms/step -
accuracy: 0.8773 - loss: 0.4944 - val_accuracy: 0.9087 - val_loss: 0.4041
Epoch 12/100
1688/1688            3s 2ms/step -
accuracy: 0.8841 - loss: 0.4665 - val_accuracy: 0.9138 - val_loss: 0.3814
Epoch 13/100
1688/1688            5s 2ms/step -
accuracy: 0.8870 - loss: 0.4439 - val_accuracy: 0.9165 - val_loss: 0.3623
Epoch 14/100
1688/1688            4s 2ms/step -
accuracy: 0.8918 - loss: 0.4260 - val_accuracy: 0.9197 - val_loss: 0.3462
Epoch 15/100
1688/1688            5s 2ms/step -
accuracy: 0.8965 - loss: 0.4062 - val_accuracy: 0.9220 - val_loss: 0.3323
Epoch 16/100
1688/1688            3s 2ms/step -
```

```
accuracy: 0.8976 - loss: 0.3919 - val_accuracy: 0.9232 - val_loss: 0.3205
Epoch 17/100
1688/1688              4s 2ms/step -
accuracy: 0.8992 - loss: 0.3860 - val_accuracy: 0.9245 - val_loss: 0.3098
Epoch 18/100
1688/1688              5s 2ms/step -
accuracy: 0.9030 - loss: 0.3747 - val_accuracy: 0.9273 - val_loss: 0.3005
Epoch 19/100
1688/1688              3s 2ms/step -
accuracy: 0.9034 - loss: 0.3651 - val_accuracy: 0.9280 - val_loss: 0.2922
Epoch 20/100
1688/1688              4s 2ms/step -
accuracy: 0.9079 - loss: 0.3518 - val_accuracy: 0.9293 - val_loss: 0.2845
Epoch 21/100
1688/1688              3s 2ms/step -
accuracy: 0.9083 - loss: 0.3462 - val_accuracy: 0.9288 - val_loss: 0.2777
Epoch 22/100
1688/1688              5s 2ms/step -
accuracy: 0.9095 - loss: 0.3381 - val_accuracy: 0.9305 - val_loss: 0.2715
Epoch 23/100
1688/1688              5s 2ms/step -
accuracy: 0.9106 - loss: 0.3289 - val_accuracy: 0.9310 - val_loss: 0.2656
Epoch 24/100
1688/1688              3s 2ms/step -
accuracy: 0.9104 - loss: 0.3289 - val_accuracy: 0.9315 - val_loss: 0.2606
Epoch 25/100
1688/1688              6s 2ms/step -
accuracy: 0.9132 - loss: 0.3187 - val_accuracy: 0.9335 - val_loss: 0.2557
Epoch 26/100
1688/1688              3s 2ms/step -
accuracy: 0.9151 - loss: 0.3134 - val_accuracy: 0.9348 - val_loss: 0.2513
Epoch 27/100
1688/1688              3s 2ms/step -
accuracy: 0.9147 - loss: 0.3117 - val_accuracy: 0.9355 - val_loss: 0.2470
Epoch 28/100
1688/1688              3s 2ms/step -
accuracy: 0.9179 - loss: 0.3038 - val_accuracy: 0.9363 - val_loss: 0.2430
Epoch 29/100
1688/1688              5s 2ms/step -
accuracy: 0.9182 - loss: 0.3035 - val_accuracy: 0.9363 - val_loss: 0.2394
Epoch 30/100
1688/1688              5s 2ms/step -
accuracy: 0.9198 - loss: 0.2922 - val_accuracy: 0.9383 - val_loss: 0.2359
Epoch 31/100
1688/1688              4s 2ms/step -
accuracy: 0.9204 - loss: 0.2876 - val_accuracy: 0.9390 - val_loss: 0.2325
Epoch 32/100
1688/1688              5s 2ms/step -
```

```
accuracy: 0.9221 - loss: 0.2825 - val_accuracy: 0.9398 - val_loss: 0.2292
Epoch 33/100
1688/1688            3s 2ms/step -
accuracy: 0.9237 - loss: 0.2780 - val_accuracy: 0.9403 - val_loss: 0.2261
Epoch 34/100
1688/1688            5s 2ms/step -
accuracy: 0.9245 - loss: 0.2761 - val_accuracy: 0.9413 - val_loss: 0.2234
Epoch 35/100
1688/1688            3s 2ms/step -
accuracy: 0.9215 - loss: 0.2820 - val_accuracy: 0.9422 - val_loss: 0.2209
Epoch 36/100
1688/1688            3s 2ms/step -
accuracy: 0.9259 - loss: 0.2709 - val_accuracy: 0.9435 - val_loss: 0.2182
Epoch 37/100
1688/1688            4s 2ms/step -
accuracy: 0.9238 - loss: 0.2769 - val_accuracy: 0.9438 - val_loss: 0.2156
Epoch 38/100
1688/1688            5s 2ms/step -
accuracy: 0.9267 - loss: 0.2676 - val_accuracy: 0.9435 - val_loss: 0.2132
Epoch 39/100
1688/1688            6s 2ms/step -
accuracy: 0.9278 - loss: 0.2648 - val_accuracy: 0.9443 - val_loss: 0.2112
Epoch 40/100
1688/1688            4s 2ms/step -
accuracy: 0.9285 - loss: 0.2594 - val_accuracy: 0.9453 - val_loss: 0.2087
Epoch 41/100
1688/1688            5s 2ms/step -
accuracy: 0.9294 - loss: 0.2559 - val_accuracy: 0.9458 - val_loss: 0.2067
Epoch 42/100
1688/1688            5s 2ms/step -
accuracy: 0.9276 - loss: 0.2579 - val_accuracy: 0.9467 - val_loss: 0.2047
Epoch 43/100
1688/1688            5s 2ms/step -
accuracy: 0.9296 - loss: 0.2554 - val_accuracy: 0.9465 - val_loss: 0.2028
Epoch 44/100
1688/1688            4s 2ms/step -
accuracy: 0.9307 - loss: 0.2496 - val_accuracy: 0.9460 - val_loss: 0.2011
Epoch 45/100
1688/1688            5s 2ms/step -
accuracy: 0.9326 - loss: 0.2463 - val_accuracy: 0.9468 - val_loss: 0.1993
Epoch 46/100
1688/1688            5s 2ms/step -
accuracy: 0.9316 - loss: 0.2479 - val_accuracy: 0.9477 - val_loss: 0.1975
Epoch 47/100
1688/1688            5s 2ms/step -
accuracy: 0.9320 - loss: 0.2451 - val_accuracy: 0.9480 - val_loss: 0.1961
Epoch 48/100
1688/1688            5s 2ms/step -
```

```
accuracy: 0.9341 - loss: 0.2389 - val_accuracy: 0.9480 - val_loss: 0.1943
Epoch 49/100
1688/1688            4s 2ms/step -
accuracy: 0.9345 - loss: 0.2380 - val_accuracy: 0.9485 - val_loss: 0.1928
Epoch 50/100
1688/1688            5s 2ms/step -
accuracy: 0.9341 - loss: 0.2370 - val_accuracy: 0.9492 - val_loss: 0.1914
Epoch 51/100
1688/1688            3s 2ms/step -
accuracy: 0.9335 - loss: 0.2379 - val_accuracy: 0.9490 - val_loss: 0.1902
Epoch 52/100
1688/1688            4s 2ms/step -
accuracy: 0.9351 - loss: 0.2340 - val_accuracy: 0.9495 - val_loss: 0.1889
Epoch 53/100
1688/1688            5s 2ms/step -
accuracy: 0.9361 - loss: 0.2307 - val_accuracy: 0.9503 - val_loss: 0.1872
Epoch 54/100
1688/1688            3s 2ms/step -
accuracy: 0.9358 - loss: 0.2297 - val_accuracy: 0.9508 - val_loss: 0.1860
Epoch 55/100
1688/1688            4s 2ms/step -
accuracy: 0.9364 - loss: 0.2305 - val_accuracy: 0.9510 - val_loss: 0.1846
Epoch 56/100
1688/1688            3s 2ms/step -
accuracy: 0.9380 - loss: 0.2257 - val_accuracy: 0.9517 - val_loss: 0.1832
Epoch 57/100
1688/1688            5s 2ms/step -
accuracy: 0.9372 - loss: 0.2281 - val_accuracy: 0.9517 - val_loss: 0.1820
Epoch 58/100
1688/1688            5s 2ms/step -
accuracy: 0.9390 - loss: 0.2247 - val_accuracy: 0.9517 - val_loss: 0.1810
Epoch 59/100
1688/1688            3s 2ms/step -
accuracy: 0.9401 - loss: 0.2185 - val_accuracy: 0.9523 - val_loss: 0.1797
Epoch 60/100
1688/1688            6s 2ms/step -
accuracy: 0.9383 - loss: 0.2217 - val_accuracy: 0.9525 - val_loss: 0.1784
Epoch 61/100
1688/1688            3s 2ms/step -
accuracy: 0.9393 - loss: 0.2176 - val_accuracy: 0.9518 - val_loss: 0.1776
Epoch 62/100
1688/1688            5s 2ms/step -
accuracy: 0.9405 - loss: 0.2180 - val_accuracy: 0.9528 - val_loss: 0.1764
Epoch 63/100
1688/1688            4s 2ms/step -
accuracy: 0.9428 - loss: 0.2080 - val_accuracy: 0.9520 - val_loss: 0.1755
Epoch 64/100
1688/1688            3s 2ms/step -
```

```
accuracy: 0.9436 - loss: 0.2083 - val_accuracy: 0.9528 - val_loss: 0.1744
Epoch 65/100
1688/1688             5s 2ms/step -
accuracy: 0.9406 - loss: 0.2127 - val_accuracy: 0.9532 - val_loss: 0.1732
Epoch 66/100
1688/1688             5s 2ms/step -
accuracy: 0.9422 - loss: 0.2104 - val_accuracy: 0.9525 - val_loss: 0.1724
Epoch 67/100
1688/1688             5s 2ms/step -
accuracy: 0.9419 - loss: 0.2092 - val_accuracy: 0.9532 - val_loss: 0.1714
Epoch 68/100
1688/1688             6s 2ms/step -
accuracy: 0.9424 - loss: 0.2076 - val_accuracy: 0.9543 - val_loss: 0.1704
Epoch 69/100
1688/1688             5s 2ms/step -
accuracy: 0.9442 - loss: 0.2053 - val_accuracy: 0.9552 - val_loss: 0.1696
Epoch 70/100
1688/1688             6s 2ms/step -
accuracy: 0.9432 - loss: 0.2050 - val_accuracy: 0.9540 - val_loss: 0.1688
Epoch 71/100
1688/1688             4s 2ms/step -
accuracy: 0.9454 - loss: 0.2000 - val_accuracy: 0.9550 - val_loss: 0.1678
Epoch 72/100
1688/1688             6s 2ms/step -
accuracy: 0.9439 - loss: 0.2022 - val_accuracy: 0.9538 - val_loss: 0.1674
Epoch 73/100
1688/1688             4s 2ms/step -
accuracy: 0.9431 - loss: 0.2007 - val_accuracy: 0.9542 - val_loss: 0.1663
Epoch 74/100
1688/1688             3s 2ms/step -
accuracy: 0.9427 - loss: 0.2042 - val_accuracy: 0.9558 - val_loss: 0.1654
Epoch 75/100
1688/1688             3s 2ms/step -
accuracy: 0.9437 - loss: 0.2016 - val_accuracy: 0.9562 - val_loss: 0.1643
Epoch 76/100
1688/1688             5s 2ms/step -
accuracy: 0.9432 - loss: 0.2002 - val_accuracy: 0.9558 - val_loss: 0.1639
Epoch 77/100
1688/1688             5s 2ms/step -
accuracy: 0.9453 - loss: 0.1972 - val_accuracy: 0.9567 - val_loss: 0.1630
Epoch 78/100
1688/1688             6s 2ms/step -
accuracy: 0.9479 - loss: 0.1910 - val_accuracy: 0.9567 - val_loss: 0.1624
Epoch 79/100
1688/1688             4s 2ms/step -
accuracy: 0.9467 - loss: 0.1942 - val_accuracy: 0.9570 - val_loss: 0.1617
Epoch 80/100
1688/1688             6s 2ms/step -
```

```
accuracy: 0.9450 - loss: 0.1957 - val_accuracy: 0.9575 - val_loss: 0.1606
Epoch 81/100
1688/1688            4s 2ms/step -
accuracy: 0.9460 - loss: 0.1949 - val_accuracy: 0.9573 - val_loss: 0.1599
Epoch 82/100
1688/1688            3s 2ms/step -
accuracy: 0.9435 - loss: 0.1985 - val_accuracy: 0.9572 - val_loss: 0.1595
Epoch 83/100
1688/1688            5s 2ms/step -
accuracy: 0.9451 - loss: 0.1967 - val_accuracy: 0.9577 - val_loss: 0.1586
Epoch 84/100
1688/1688            5s 2ms/step -
accuracy: 0.9471 - loss: 0.1901 - val_accuracy: 0.9587 - val_loss: 0.1579
Epoch 85/100
1688/1688            5s 2ms/step -
accuracy: 0.9472 - loss: 0.1886 - val_accuracy: 0.9573 - val_loss: 0.1575
Epoch 86/100
1688/1688            5s 2ms/step -
accuracy: 0.9478 - loss: 0.1849 - val_accuracy: 0.9582 - val_loss: 0.1569
Epoch 87/100
1688/1688            3s 2ms/step -
accuracy: 0.9478 - loss: 0.1839 - val_accuracy: 0.9583 - val_loss: 0.1563
Epoch 88/100
1688/1688            5s 2ms/step -
accuracy: 0.9486 - loss: 0.1839 - val_accuracy: 0.9585 - val_loss: 0.1554
Epoch 89/100
1688/1688            5s 2ms/step -
accuracy: 0.9469 - loss: 0.1876 - val_accuracy: 0.9587 - val_loss: 0.1550
Epoch 90/100
1688/1688            3s 2ms/step -
accuracy: 0.9481 - loss: 0.1857 - val_accuracy: 0.9582 - val_loss: 0.1545
Epoch 91/100
1688/1688            5s 2ms/step -
accuracy: 0.9487 - loss: 0.1840 - val_accuracy: 0.9582 - val_loss: 0.1539
Epoch 92/100
1688/1688            3s 2ms/step -
accuracy: 0.9499 - loss: 0.1817 - val_accuracy: 0.9578 - val_loss: 0.1536
Epoch 93/100
1688/1688            4s 2ms/step -
accuracy: 0.9501 - loss: 0.1806 - val_accuracy: 0.9585 - val_loss: 0.1528
Epoch 94/100
1688/1688            4s 2ms/step -
accuracy: 0.9497 - loss: 0.1811 - val_accuracy: 0.9587 - val_loss: 0.1521
Epoch 95/100
1688/1688            3s 2ms/step -
accuracy: 0.9499 - loss: 0.1778 - val_accuracy: 0.9597 - val_loss: 0.1516
Epoch 96/100
1688/1688            4s 2ms/step -
```

```
accuracy: 0.9491 - loss: 0.1816 - val_accuracy: 0.9588 - val_loss: 0.1513
Epoch 97/100
1688/1688            3s 2ms/step -
accuracy: 0.9496 - loss: 0.1804 - val_accuracy: 0.9590 - val_loss: 0.1505
Epoch 98/100
1688/1688            3s 2ms/step -
accuracy: 0.9503 - loss: 0.1776 - val_accuracy: 0.9595 - val_loss: 0.1499
Epoch 99/100
1688/1688            5s 2ms/step -
accuracy: 0.9503 - loss: 0.1788 - val_accuracy: 0.9588 - val_loss: 0.1499
Epoch 100/100
1688/1688            3s 2ms/step -
accuracy: 0.9506 - loss: 0.1713 - val_accuracy: 0.9593 - val_loss: 0.1488
Restoring model weights from the end of the best epoch: 100.
Training with batch size 64 and learning rate 0.01
Epoch 1/100
844/844            4s 3ms/step -
accuracy: 0.8417 - loss: 0.5082 - val_accuracy: 0.9510 - val_loss: 0.1620
Epoch 2/100
844/844            3s 3ms/step -
accuracy: 0.9440 - loss: 0.1895 - val_accuracy: 0.9568 - val_loss: 0.1460
Epoch 3/100
844/844            2s 3ms/step -
accuracy: 0.9512 - loss: 0.1617 - val_accuracy: 0.9627 - val_loss: 0.1439
Epoch 4/100
844/844            2s 3ms/step -
accuracy: 0.9586 - loss: 0.1401 - val_accuracy: 0.9638 - val_loss: 0.1254
Epoch 5/100
844/844            3s 3ms/step -
accuracy: 0.9624 - loss: 0.1276 - val_accuracy: 0.9575 - val_loss: 0.1410
Epoch 6/100
844/844            3s 3ms/step -
accuracy: 0.9641 - loss: 0.1189 - val_accuracy: 0.9565 - val_loss: 0.1493
Epoch 7/100
844/844            2s 3ms/step -
accuracy: 0.9643 - loss: 0.1154 - val_accuracy: 0.9602 - val_loss: 0.1563
Epoch 7: early stopping
Restoring model weights from the end of the best epoch: 4.
Training with batch size 64 and learning rate 0.001
Epoch 1/100
844/844            3s 3ms/step -
accuracy: 0.7561 - loss: 0.8474 - val_accuracy: 0.9447 - val_loss: 0.2059
Epoch 2/100
844/844            2s 2ms/step -
accuracy: 0.9298 - loss: 0.2476 - val_accuracy: 0.9577 - val_loss: 0.1564
Epoch 3/100
844/844            2s 2ms/step -
accuracy: 0.9471 - loss: 0.1836 - val_accuracy: 0.9625 - val_loss: 0.1329
```

```
Epoch 4/100
844/844            3s 3ms/step -
accuracy: 0.9565 - loss: 0.1515 - val_accuracy: 0.9628 - val_loss: 0.1263
Epoch 5/100
844/844            2s 2ms/step -
accuracy: 0.9627 - loss: 0.1292 - val_accuracy: 0.9620 - val_loss: 0.1265
Epoch 6/100
844/844            2s 2ms/step -
accuracy: 0.9664 - loss: 0.1130 - val_accuracy: 0.9705 - val_loss: 0.1039
Epoch 7/100
844/844            3s 2ms/step -
accuracy: 0.9717 - loss: 0.0971 - val_accuracy: 0.9675 - val_loss: 0.1064
Epoch 8/100
844/844            3s 3ms/step -
accuracy: 0.9729 - loss: 0.0878 - val_accuracy: 0.9677 - val_loss: 0.1019
Epoch 9/100
844/844            3s 3ms/step -
accuracy: 0.9776 - loss: 0.0773 - val_accuracy: 0.9677 - val_loss: 0.1034
Epoch 10/100
844/844            2s 3ms/step -
accuracy: 0.9781 - loss: 0.0723 - val_accuracy: 0.9673 - val_loss: 0.1143
Epoch 11/100
844/844            3s 3ms/step -
accuracy: 0.9791 - loss: 0.0707 - val_accuracy: 0.9713 - val_loss: 0.0970
Epoch 12/100
844/844            2s 2ms/step -
accuracy: 0.9804 - loss: 0.0666 - val_accuracy: 0.9700 - val_loss: 0.1031
Epoch 13/100
844/844            3s 3ms/step -
accuracy: 0.9811 - loss: 0.0618 - val_accuracy: 0.9725 - val_loss: 0.0946
Epoch 14/100
844/844            5s 2ms/step -
accuracy: 0.9825 - loss: 0.0573 - val_accuracy: 0.9720 - val_loss: 0.0960
Epoch 15/100
844/844            3s 2ms/step -
accuracy: 0.9842 - loss: 0.0520 - val_accuracy: 0.9702 - val_loss: 0.1044
Epoch 16/100
844/844            3s 3ms/step -
accuracy: 0.9853 - loss: 0.0492 - val_accuracy: 0.9715 - val_loss: 0.1009
Epoch 16: early stopping
Restoring model weights from the end of the best epoch: 13.
Training with batch size 64 and learning rate 0.0001
Epoch 1/100
844/844            3s 3ms/step -
accuracy: 0.4248 - loss: 1.7688 - val_accuracy: 0.8727 - val_loss: 0.5855
Epoch 2/100
844/844            2s 3ms/step -
accuracy: 0.8593 - loss: 0.5714 - val_accuracy: 0.9130 - val_loss: 0.3516
```

```
Epoch 3/100
844/844            2s 3ms/step -
accuracy: 0.8920 - loss: 0.4045 - val_accuracy: 0.9235 - val_loss: 0.2854
Epoch 4/100
844/844            3s 3ms/step -
accuracy: 0.9082 - loss: 0.3388 - val_accuracy: 0.9333 - val_loss: 0.2519
Epoch 5/100
844/844            2s 2ms/step -
accuracy: 0.9173 - loss: 0.3019 - val_accuracy: 0.9368 - val_loss: 0.2307
Epoch 6/100
844/844            2s 2ms/step -
accuracy: 0.9219 - loss: 0.2817 - val_accuracy: 0.9420 - val_loss: 0.2164
Epoch 7/100
844/844            3s 2ms/step -
accuracy: 0.9260 - loss: 0.2596 - val_accuracy: 0.9457 - val_loss: 0.2045
Epoch 8/100
844/844            2s 2ms/step -
accuracy: 0.9303 - loss: 0.2498 - val_accuracy: 0.9465 - val_loss: 0.1962
Epoch 9/100
844/844            2s 3ms/step -
accuracy: 0.9328 - loss: 0.2403 - val_accuracy: 0.9502 - val_loss: 0.1877
Epoch 10/100
844/844            2s 3ms/step -
accuracy: 0.9352 - loss: 0.2267 - val_accuracy: 0.9527 - val_loss: 0.1807
Epoch 11/100
844/844            2s 2ms/step -
accuracy: 0.9373 - loss: 0.2231 - val_accuracy: 0.9543 - val_loss: 0.1747
Epoch 12/100
844/844            2s 2ms/step -
accuracy: 0.9401 - loss: 0.2112 - val_accuracy: 0.9555 - val_loss: 0.1702
Epoch 13/100
844/844            3s 2ms/step -
accuracy: 0.9415 - loss: 0.2055 - val_accuracy: 0.9567 - val_loss: 0.1650
Epoch 14/100
844/844            3s 3ms/step -
accuracy: 0.9439 - loss: 0.2007 - val_accuracy: 0.9583 - val_loss: 0.1607
Epoch 15/100
844/844            2s 2ms/step -
accuracy: 0.9469 - loss: 0.1898 - val_accuracy: 0.9592 - val_loss: 0.1566
Epoch 16/100
844/844            2s 2ms/step -
accuracy: 0.9477 - loss: 0.1851 - val_accuracy: 0.9597 - val_loss: 0.1543
Epoch 17/100
844/844            2s 3ms/step -
accuracy: 0.9504 - loss: 0.1752 - val_accuracy: 0.9598 - val_loss: 0.1501
Epoch 18/100
844/844            3s 3ms/step -
accuracy: 0.9485 - loss: 0.1815 - val_accuracy: 0.9608 - val_loss: 0.1472
```

```
Epoch 19/100
844/844             2s 3ms/step -
accuracy: 0.9516 - loss: 0.1711 - val_accuracy: 0.9603 - val_loss: 0.1471
Epoch 20/100
844/844             2s 3ms/step -
accuracy: 0.9524 - loss: 0.1654 - val_accuracy: 0.9615 - val_loss: 0.1427
Epoch 21/100
844/844             2s 2ms/step -
accuracy: 0.9545 - loss: 0.1608 - val_accuracy: 0.9607 - val_loss: 0.1424
Epoch 22/100
844/844             2s 2ms/step -
accuracy: 0.9542 - loss: 0.1595 - val_accuracy: 0.9610 - val_loss: 0.1388
Epoch 23/100
844/844             3s 2ms/step -
accuracy: 0.9567 - loss: 0.1532 - val_accuracy: 0.9638 - val_loss: 0.1372
Epoch 24/100
844/844             2s 3ms/step -
accuracy: 0.9558 - loss: 0.1528 - val_accuracy: 0.9623 - val_loss: 0.1368
Epoch 25/100
844/844             2s 3ms/step -
accuracy: 0.9553 - loss: 0.1519 - val_accuracy: 0.9632 - val_loss: 0.1341
Epoch 26/100
844/844             2s 2ms/step -
accuracy: 0.9598 - loss: 0.1404 - val_accuracy: 0.9628 - val_loss: 0.1341
Epoch 27/100
844/844             2s 2ms/step -
accuracy: 0.9609 - loss: 0.1382 - val_accuracy: 0.9650 - val_loss: 0.1314
Epoch 28/100
844/844             3s 2ms/step -
accuracy: 0.9595 - loss: 0.1424 - val_accuracy: 0.9650 - val_loss: 0.1303
Epoch 29/100
844/844             3s 3ms/step -
accuracy: 0.9621 - loss: 0.1340 - val_accuracy: 0.9655 - val_loss: 0.1287
Epoch 30/100
844/844             5s 2ms/step -
accuracy: 0.9622 - loss: 0.1350 - val_accuracy: 0.9655 - val_loss: 0.1276
Epoch 31/100
844/844             2s 2ms/step -
accuracy: 0.9612 - loss: 0.1359 - val_accuracy: 0.9653 - val_loss: 0.1262
Epoch 32/100
844/844             2s 2ms/step -
accuracy: 0.9618 - loss: 0.1323 - val_accuracy: 0.9657 - val_loss: 0.1252
Epoch 33/100
844/844             2s 3ms/step -
accuracy: 0.9653 - loss: 0.1230 - val_accuracy: 0.9652 - val_loss: 0.1242
Epoch 34/100
844/844             2s 2ms/step -
accuracy: 0.9638 - loss: 0.1254 - val_accuracy: 0.9665 - val_loss: 0.1229
```

```
Epoch 35/100
844/844            3s 2ms/step -
accuracy: 0.9654 - loss: 0.1222 - val_accuracy: 0.9652 - val_loss: 0.1252
Epoch 36/100
844/844            2s 2ms/step -
accuracy: 0.9653 - loss: 0.1234 - val_accuracy: 0.9663 - val_loss: 0.1221
Epoch 37/100
844/844            3s 2ms/step -
accuracy: 0.9656 - loss: 0.1224 - val_accuracy: 0.9673 - val_loss: 0.1218
Epoch 38/100
844/844            3s 3ms/step -
accuracy: 0.9675 - loss: 0.1148 - val_accuracy: 0.9678 - val_loss: 0.1201
Epoch 39/100
844/844            2s 2ms/step -
accuracy: 0.9670 - loss: 0.1132 - val_accuracy: 0.9670 - val_loss: 0.1207
Epoch 40/100
844/844            2s 2ms/step -
accuracy: 0.9674 - loss: 0.1142 - val_accuracy: 0.9675 - val_loss: 0.1188
Epoch 41/100
844/844            2s 2ms/step -
accuracy: 0.9670 - loss: 0.1142 - val_accuracy: 0.9668 - val_loss: 0.1199
Epoch 42/100
844/844            3s 3ms/step -
accuracy: 0.9676 - loss: 0.1118 - val_accuracy: 0.9677 - val_loss: 0.1198
Epoch 43/100
844/844            2s 3ms/step -
accuracy: 0.9683 - loss: 0.1118 - val_accuracy: 0.9685 - val_loss: 0.1180
Epoch 44/100
844/844            2s 2ms/step -
accuracy: 0.9685 - loss: 0.1070 - val_accuracy: 0.9678 - val_loss: 0.1188
Epoch 45/100
844/844            3s 2ms/step -
accuracy: 0.9702 - loss: 0.1038 - val_accuracy: 0.9663 - val_loss: 0.1189
Epoch 46/100
844/844            2s 2ms/step -
accuracy: 0.9695 - loss: 0.1042 - val_accuracy: 0.9688 - val_loss: 0.1170
Epoch 47/100
844/844            3s 3ms/step -
accuracy: 0.9698 - loss: 0.1017 - val_accuracy: 0.9683 - val_loss: 0.1175
Epoch 48/100
844/844            2s 2ms/step -
accuracy: 0.9708 - loss: 0.1030 - val_accuracy: 0.9678 - val_loss: 0.1166
Epoch 49/100
844/844            3s 2ms/step -
accuracy: 0.9717 - loss: 0.0981 - val_accuracy: 0.9688 - val_loss: 0.1159
Epoch 50/100
844/844            2s 2ms/step -
accuracy: 0.9720 - loss: 0.0967 - val_accuracy: 0.9682 - val_loss: 0.1156
```

```
Epoch 51/100
844/844            3s 2ms/step -
accuracy: 0.9722 - loss: 0.0967 - val_accuracy: 0.9678 - val_loss: 0.1157
Epoch 52/100
844/844            3s 3ms/step -
accuracy: 0.9721 - loss: 0.0975 - val_accuracy: 0.9695 - val_loss: 0.1145
Epoch 53/100
844/844            2s 2ms/step -
accuracy: 0.9710 - loss: 0.0963 - val_accuracy: 0.9690 - val_loss: 0.1145
Epoch 54/100
844/844            2s 2ms/step -
accuracy: 0.9733 - loss: 0.0918 - val_accuracy: 0.9692 - val_loss: 0.1146
Epoch 55/100
844/844            3s 2ms/step -
accuracy: 0.9717 - loss: 0.0969 - val_accuracy: 0.9683 - val_loss: 0.1157
Epoch 55: early stopping
Restoring model weights from the end of the best epoch: 52.
Training with batch size 64 and learning rate 1e-05
Epoch 1/100
844/844            4s 3ms/step -
accuracy: 0.1760 - loss: 2.2626 - val_accuracy: 0.3185 - val_loss: 2.0967
Epoch 2/100
844/844            2s 2ms/step -
accuracy: 0.3469 - loss: 2.0500 - val_accuracy: 0.4382 - val_loss: 1.8614
Epoch 3/100
844/844            2s 2ms/step -
accuracy: 0.4417 - loss: 1.8275 - val_accuracy: 0.5177 - val_loss: 1.6438
Epoch 4/100
844/844            3s 2ms/step -
accuracy: 0.5188 - loss: 1.6206 - val_accuracy: 0.6133 - val_loss: 1.4348
Epoch 5/100
844/844            2s 2ms/step -
accuracy: 0.6034 - loss: 1.4206 - val_accuracy: 0.6767 - val_loss: 1.2486
Epoch 6/100
844/844            3s 3ms/step -
accuracy: 0.6552 - loss: 1.2633 - val_accuracy: 0.7225 - val_loss: 1.0978
Epoch 7/100
844/844            2s 2ms/step -
accuracy: 0.7011 - loss: 1.1239 - val_accuracy: 0.7602 - val_loss: 0.9729
Epoch 8/100
844/844            3s 2ms/step -
accuracy: 0.7372 - loss: 1.0060 - val_accuracy: 0.7902 - val_loss: 0.8705
Epoch 9/100
844/844            2s 2ms/step -
accuracy: 0.7596 - loss: 0.9187 - val_accuracy: 0.8158 - val_loss: 0.7864
Epoch 10/100
844/844            3s 2ms/step -
accuracy: 0.7859 - loss: 0.8275 - val_accuracy: 0.8337 - val_loss: 0.7177
```

```
Epoch 11/100
844/844               3s 3ms/step -
accuracy: 0.8011 - loss: 0.7706 - val_accuracy: 0.8475 - val_loss: 0.6605
Epoch 12/100
844/844               2s 2ms/step -
accuracy: 0.8141 - loss: 0.7188 - val_accuracy: 0.8575 - val_loss: 0.6126
Epoch 13/100
844/844               2s 2ms/step -
accuracy: 0.8281 - loss: 0.6717 - val_accuracy: 0.8667 - val_loss: 0.5726
Epoch 14/100
844/844               2s 2ms/step -
accuracy: 0.8383 - loss: 0.6334 - val_accuracy: 0.8753 - val_loss: 0.5383
Epoch 15/100
844/844               3s 2ms/step -
accuracy: 0.8454 - loss: 0.5963 - val_accuracy: 0.8827 - val_loss: 0.5089
Epoch 16/100
844/844               3s 3ms/step -
accuracy: 0.8534 - loss: 0.5696 - val_accuracy: 0.8867 - val_loss: 0.4837
Epoch 17/100
844/844               2s 2ms/step -
accuracy: 0.8578 - loss: 0.5459 - val_accuracy: 0.8913 - val_loss: 0.4613
Epoch 18/100
844/844               2s 2ms/step -
accuracy: 0.8632 - loss: 0.5262 - val_accuracy: 0.8945 - val_loss: 0.4417
Epoch 19/100
844/844               2s 2ms/step -
accuracy: 0.8700 - loss: 0.4981 - val_accuracy: 0.8983 - val_loss: 0.4245
Epoch 20/100
844/844               2s 2ms/step -
accuracy: 0.8741 - loss: 0.4826 - val_accuracy: 0.9003 - val_loss: 0.4095
Epoch 21/100
844/844               3s 3ms/step -
accuracy: 0.8782 - loss: 0.4683 - val_accuracy: 0.9043 - val_loss: 0.3957
Epoch 22/100
844/844               2s 2ms/step -
accuracy: 0.8757 - loss: 0.4574 - val_accuracy: 0.9077 - val_loss: 0.3831
Epoch 23/100
844/844               2s 2ms/step -
accuracy: 0.8801 - loss: 0.4502 - val_accuracy: 0.9082 - val_loss: 0.3718
Epoch 24/100
844/844               2s 2ms/step -
accuracy: 0.8861 - loss: 0.4287 - val_accuracy: 0.9088 - val_loss: 0.3616
Epoch 25/100
844/844               3s 2ms/step -
accuracy: 0.8864 - loss: 0.4215 - val_accuracy: 0.9128 - val_loss: 0.3521
Epoch 26/100
844/844               2s 3ms/step -
accuracy: 0.8906 - loss: 0.4078 - val_accuracy: 0.9145 - val_loss: 0.3436
```

```
Epoch 27/100
844/844              2s 2ms/step -
accuracy: 0.8922 - loss: 0.4025 - val_accuracy: 0.9158 - val_loss: 0.3357
Epoch 28/100
844/844              2s 2ms/step -
accuracy: 0.8949 - loss: 0.3920 - val_accuracy: 0.9183 - val_loss: 0.3282
Epoch 29/100
844/844              2s 2ms/step -
accuracy: 0.8980 - loss: 0.3778 - val_accuracy: 0.9192 - val_loss: 0.3216
Epoch 30/100
844/844              2s 2ms/step -
accuracy: 0.8976 - loss: 0.3780 - val_accuracy: 0.9210 - val_loss: 0.3153
Epoch 31/100
844/844              3s 3ms/step -
accuracy: 0.8994 - loss: 0.3685 - val_accuracy: 0.9220 - val_loss: 0.3095
Epoch 32/100
844/844              2s 2ms/step -
accuracy: 0.9021 - loss: 0.3646 - val_accuracy: 0.9232 - val_loss: 0.3042
Epoch 33/100
844/844              2s 2ms/step -
accuracy: 0.9042 - loss: 0.3568 - val_accuracy: 0.9242 - val_loss: 0.2993
Epoch 34/100
844/844              2s 2ms/step -
accuracy: 0.9034 - loss: 0.3568 - val_accuracy: 0.9240 - val_loss: 0.2944
Epoch 35/100
844/844              2s 2ms/step -
accuracy: 0.9034 - loss: 0.3512 - val_accuracy: 0.9257 - val_loss: 0.2900
Epoch 36/100
844/844              2s 2ms/step -
accuracy: 0.9055 - loss: 0.3441 - val_accuracy: 0.9257 - val_loss: 0.2859
Epoch 37/100
844/844              3s 3ms/step -
accuracy: 0.9065 - loss: 0.3420 - val_accuracy: 0.9270 - val_loss: 0.2820
Epoch 38/100
844/844              2s 2ms/step -
accuracy: 0.9082 - loss: 0.3348 - val_accuracy: 0.9285 - val_loss: 0.2783
Epoch 39/100
844/844              2s 2ms/step -
accuracy: 0.9077 - loss: 0.3304 - val_accuracy: 0.9290 - val_loss: 0.2750
Epoch 40/100
844/844              3s 2ms/step -
accuracy: 0.9113 - loss: 0.3219 - val_accuracy: 0.9288 - val_loss: 0.2717
Epoch 41/100
844/844              3s 2ms/step -
accuracy: 0.9109 - loss: 0.3205 - val_accuracy: 0.9300 - val_loss: 0.2686
Epoch 42/100
844/844              2s 2ms/step -
accuracy: 0.9135 - loss: 0.3154 - val_accuracy: 0.9318 - val_loss: 0.2658
```

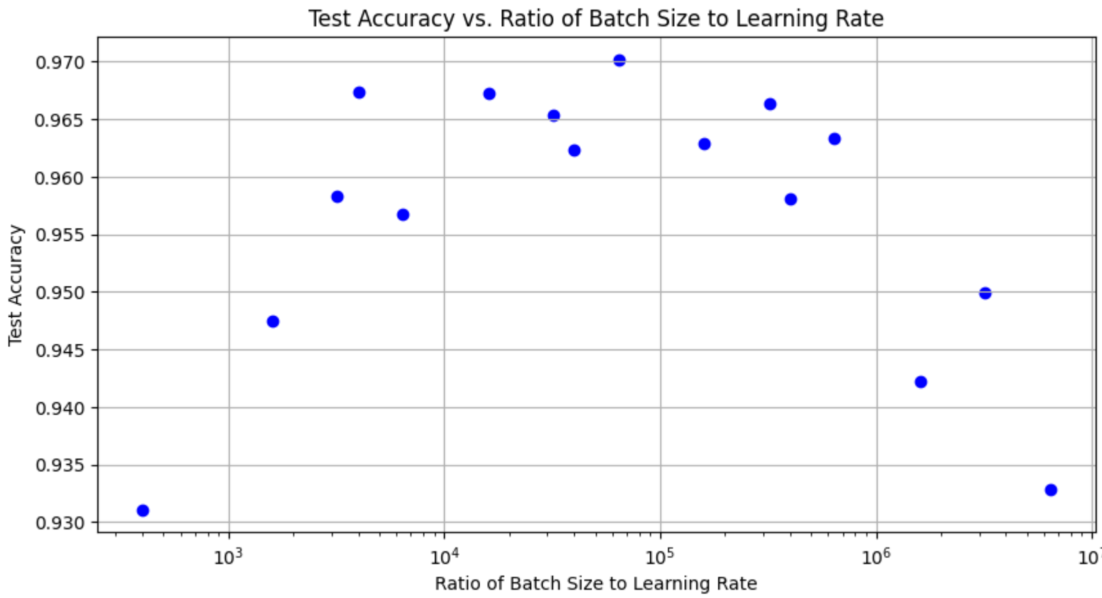
```
Epoch 43/100
844/844            2s 2ms/step -
accuracy: 0.9122 - loss: 0.3183 - val_accuracy: 0.9315 - val_loss: 0.2630
Epoch 44/100
844/844            3s 2ms/step -
accuracy: 0.9150 - loss: 0.3087 - val_accuracy: 0.9323 - val_loss: 0.2603
Epoch 45/100
844/844            2s 2ms/step -
accuracy: 0.9145 - loss: 0.3112 - val_accuracy: 0.9333 - val_loss: 0.2579
Epoch 46/100
844/844            3s 3ms/step -
accuracy: 0.9155 - loss: 0.3011 - val_accuracy: 0.9335 - val_loss: 0.2555
Epoch 47/100
844/844            2s 2ms/step -
accuracy: 0.9179 - loss: 0.2985 - val_accuracy: 0.9333 - val_loss: 0.2531
Epoch 48/100
844/844            2s 2ms/step -
accuracy: 0.9154 - loss: 0.3022 - val_accuracy: 0.9342 - val_loss: 0.2510
Epoch 49/100
844/844            3s 2ms/step -
accuracy: 0.9179 - loss: 0.2958 - val_accuracy: 0.9352 - val_loss: 0.2489
Epoch 50/100
844/844            2s 2ms/step -
accuracy: 0.9164 - loss: 0.2982 - val_accuracy: 0.9350 - val_loss: 0.2468
Epoch 51/100
844/844            3s 3ms/step -
accuracy: 0.9187 - loss: 0.2970 - val_accuracy: 0.9355 - val_loss: 0.2449
Epoch 52/100
844/844            2s 2ms/step -
accuracy: 0.9201 - loss: 0.2880 - val_accuracy: 0.9362 - val_loss: 0.2432
Epoch 53/100
844/844            3s 2ms/step -
accuracy: 0.9205 - loss: 0.2887 - val_accuracy: 0.9367 - val_loss: 0.2413
Epoch 54/100
844/844            2s 2ms/step -
accuracy: 0.9213 - loss: 0.2824 - val_accuracy: 0.9370 - val_loss: 0.2397
Epoch 55/100
844/844            2s 3ms/step -
accuracy: 0.9217 - loss: 0.2813 - val_accuracy: 0.9370 - val_loss: 0.2381
Epoch 56/100
844/844            3s 3ms/step -
accuracy: 0.9234 - loss: 0.2796 - val_accuracy: 0.9368 - val_loss: 0.2364
Epoch 57/100
844/844            5s 2ms/step -
accuracy: 0.9230 - loss: 0.2774 - val_accuracy: 0.9370 - val_loss: 0.2349
Epoch 58/100
844/844            3s 2ms/step -
accuracy: 0.9231 - loss: 0.2777 - val_accuracy: 0.9373 - val_loss: 0.2336
```

```
Epoch 59/100
844/844            2s 2ms/step -
accuracy: 0.9225 - loss: 0.2785 - val_accuracy: 0.9368 - val_loss: 0.2321
Epoch 60/100
844/844            3s 3ms/step -
accuracy: 0.9249 - loss: 0.2713 - val_accuracy: 0.9380 - val_loss: 0.2307
Epoch 61/100
844/844            2s 2ms/step -
accuracy: 0.9237 - loss: 0.2744 - val_accuracy: 0.9387 - val_loss: 0.2294
Epoch 62/100
844/844            2s 2ms/step -
accuracy: 0.9254 - loss: 0.2705 - val_accuracy: 0.9380 - val_loss: 0.2281
Epoch 63/100
844/844            3s 3ms/step -
accuracy: 0.9239 - loss: 0.2697 - val_accuracy: 0.9388 - val_loss: 0.2269
Epoch 64/100
844/844            2s 2ms/step -
accuracy: 0.9282 - loss: 0.2579 - val_accuracy: 0.9392 - val_loss: 0.2256
Epoch 65/100
844/844            3s 3ms/step -
accuracy: 0.9244 - loss: 0.2706 - val_accuracy: 0.9390 - val_loss: 0.2246
Epoch 66/100
844/844            2s 2ms/step -
accuracy: 0.9245 - loss: 0.2683 - val_accuracy: 0.9398 - val_loss: 0.2234
Epoch 67/100
844/844            2s 2ms/step -
accuracy: 0.9263 - loss: 0.2628 - val_accuracy: 0.9408 - val_loss: 0.2222
Epoch 68/100
844/844            3s 2ms/step -
accuracy: 0.9265 - loss: 0.2603 - val_accuracy: 0.9407 - val_loss: 0.2211
Epoch 69/100
844/844            3s 2ms/step -
accuracy: 0.9263 - loss: 0.2627 - val_accuracy: 0.9415 - val_loss: 0.2200
Epoch 70/100
844/844            3s 3ms/step -
accuracy: 0.9256 - loss: 0.2607 - val_accuracy: 0.9418 - val_loss: 0.2190
Epoch 71/100
844/844            2s 2ms/step -
accuracy: 0.9297 - loss: 0.2546 - val_accuracy: 0.9422 - val_loss: 0.2179
Epoch 72/100
844/844            2s 2ms/step -
accuracy: 0.9295 - loss: 0.2499 - val_accuracy: 0.9425 - val_loss: 0.2171
Epoch 73/100
844/844            3s 2ms/step -
accuracy: 0.9292 - loss: 0.2561 - val_accuracy: 0.9427 - val_loss: 0.2162
Epoch 74/100
844/844            3s 3ms/step -
accuracy: 0.9303 - loss: 0.2513 - val_accuracy: 0.9423 - val_loss: 0.2153
```

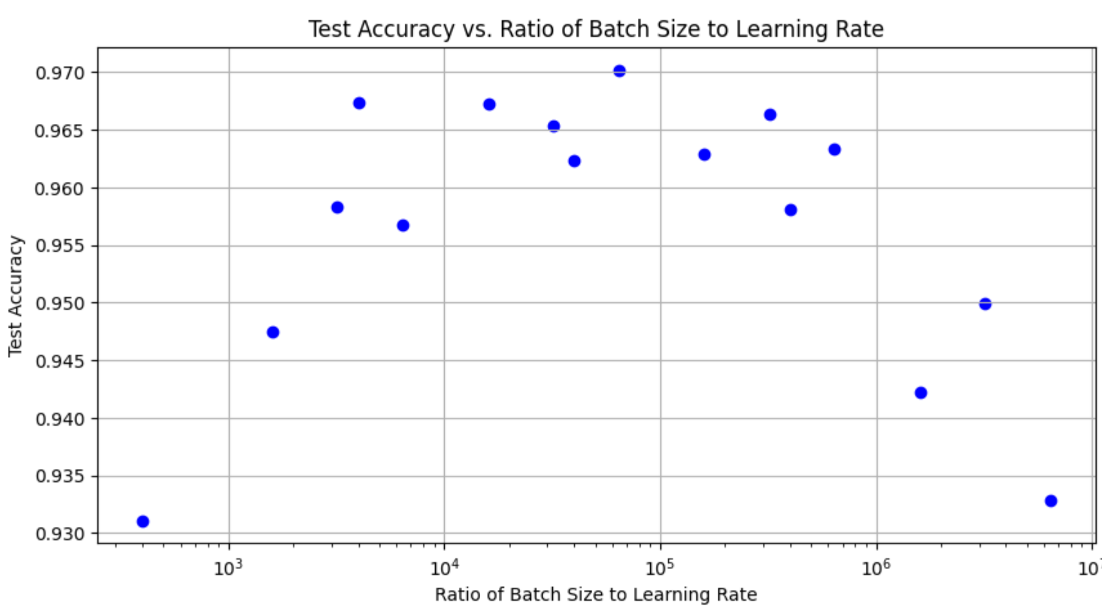
```
Epoch 75/100
844/844              4s 2ms/step -
accuracy: 0.9297 - loss: 0.2511 - val_accuracy: 0.9423 - val_loss: 0.2143
Epoch 76/100
844/844              2s 2ms/step -
accuracy: 0.9287 - loss: 0.2528 - val_accuracy: 0.9430 - val_loss: 0.2133
Epoch 77/100
844/844              3s 2ms/step -
accuracy: 0.9279 - loss: 0.2569 - val_accuracy: 0.9432 - val_loss: 0.2127
Epoch 78/100
844/844              3s 3ms/step -
accuracy: 0.9302 - loss: 0.2494 - val_accuracy: 0.9437 - val_loss: 0.2117
Epoch 79/100
844/844              2s 2ms/step -
accuracy: 0.9317 - loss: 0.2429 - val_accuracy: 0.9433 - val_loss: 0.2108
Epoch 80/100
844/844              2s 2ms/step -
accuracy: 0.9312 - loss: 0.2422 - val_accuracy: 0.9437 - val_loss: 0.2100
Epoch 81/100
844/844              3s 3ms/step -
accuracy: 0.9325 - loss: 0.2391 - val_accuracy: 0.9437 - val_loss: 0.2092
Epoch 82/100
844/844              2s 2ms/step -
accuracy: 0.9322 - loss: 0.2423 - val_accuracy: 0.9437 - val_loss: 0.2083
Epoch 83/100
844/844              3s 3ms/step -
accuracy: 0.9308 - loss: 0.2476 - val_accuracy: 0.9437 - val_loss: 0.2076
Epoch 84/100
844/844              2s 2ms/step -
accuracy: 0.9335 - loss: 0.2390 - val_accuracy: 0.9435 - val_loss: 0.2068
Epoch 85/100
844/844              3s 2ms/step -
accuracy: 0.9323 - loss: 0.2392 - val_accuracy: 0.9442 - val_loss: 0.2061
Epoch 86/100
844/844              2s 2ms/step -
accuracy: 0.9324 - loss: 0.2428 - val_accuracy: 0.9443 - val_loss: 0.2053
Epoch 87/100
844/844              3s 2ms/step -
accuracy: 0.9335 - loss: 0.2387 - val_accuracy: 0.9450 - val_loss: 0.2046
Epoch 88/100
844/844              3s 3ms/step -
accuracy: 0.9329 - loss: 0.2386 - val_accuracy: 0.9447 - val_loss: 0.2039
Epoch 89/100
844/844              2s 2ms/step -
accuracy: 0.9330 - loss: 0.2390 - val_accuracy: 0.9452 - val_loss: 0.2032
Epoch 90/100
844/844              2s 2ms/step -
accuracy: 0.9326 - loss: 0.2396 - val_accuracy: 0.9455 - val_loss: 0.2025
```

```
Epoch 91/100
844/844            2s 2ms/step -
accuracy: 0.9329 - loss: 0.2361 - val_accuracy: 0.9450 - val_loss: 0.2019
Epoch 92/100
844/844            3s 2ms/step -
accuracy: 0.9347 - loss: 0.2338 - val_accuracy: 0.9450 - val_loss: 0.2014
Epoch 93/100
844/844            3s 3ms/step -
accuracy: 0.9344 - loss: 0.2351 - val_accuracy: 0.9458 - val_loss: 0.2006
Epoch 94/100
844/844            5s 2ms/step -
accuracy: 0.9373 - loss: 0.2262 - val_accuracy: 0.9458 - val_loss: 0.2000
Epoch 95/100
844/844            3s 3ms/step -
accuracy: 0.9346 - loss: 0.2332 - val_accuracy: 0.9453 - val_loss: 0.1993
Epoch 96/100
844/844            2s 2ms/step -
accuracy: 0.9360 - loss: 0.2283 - val_accuracy: 0.9455 - val_loss: 0.1987
Epoch 97/100
844/844            3s 3ms/step -
accuracy: 0.9360 - loss: 0.2305 - val_accuracy: 0.9455 - val_loss: 0.1981
Epoch 98/100
844/844            2s 2ms/step -
accuracy: 0.9342 - loss: 0.2351 - val_accuracy: 0.9460 - val_loss: 0.1975
Epoch 99/100
844/844            2s 2ms/step -
accuracy: 0.9366 - loss: 0.2296 - val_accuracy: 0.9462 - val_loss: 0.1969
Epoch 100/100
844/844            2s 2ms/step -
accuracy: 0.9355 - loss: 0.2314 - val_accuracy: 0.9467 - val_loss: 0.1964
Restoring model weights from the end of the best epoch: 100.
```

[6]:
```python
# ratios of batch size to learning rate and plot results
ratios = [bs/lr for bs, lr, _ in results]
accuracies = [acc for _, _, acc in results]
```



[7]:
```
#plot
plt.figure(figsize=(10, 5))
plt.scatter(ratios, accuracies, color='blue')
plt.title('Test Accuracy vs. Ratio of Batch Size to Learning Rate')
plt.xlabel('Ratio of Batch Size to Learning Rate')
plt.ylabel('Test Accuracy')
plt.grid(True)
plt.xscale('log')
plt.show()
```



Observations: - The charts display how varying the ratio of batch size to learning rate affects test accuracy in training ANNs for the MNIST dataset. - Larger ratios generally correlate with higher test accuracies, indicating that either increasing batch size or decreasing the learning rate can enhance model performance. -There's a clear peak in performance at mid-range ratios, suggesting an optimal balance between batch size and learning rate, beyond which performance may plateau or even decrease, as seen in the highest ratio values.

```
[ ]:
```