Lab Workbook

Faculty Name: **Dr. Neetu Singla**

Student Name: **Piyush Gambhir**
Roll No.: **21CSU349**
Semester: **VI**
Group: **AIML-B (A3)**

Department of Computer Science and
EngineeringThe NorthCap University
Gurugram- 122017, India

Session 2023-24

INDEX

GitHub Repository Link:

ncu-lab-manual-and-end-semester-projects/NCU-CSL312 - DL - Lab Manual at main · piyush-gambhir/ncu-lab-manual-and-end-semester-projects (github.com)

Experiment No: 1

Student Name and Roll Number: Piyush Gambhir – 21CSU349	
Semester /Section: 6th Semester – AIML-B (A3)	
Link to Code: ncu-lab-manual-and-end-semester-projects/NCU-CSL312 - DL - Lab Manual/Experiment 1/Experiment 1.ipynb at main · piyush-gambhir/ncu-lab-manual-and-end-semester-projects (github.com)	
Date:	
Faculty Signature:	
Marks:	

Objective(s):
To explore the basic features of TensorFlow and Keras Package.

Experiment 1

Problem Statement:

To explore the basic features of Tensorflow and Keras packages.

Github & Google Colab Links:

GitHub Link: https://github.com/piyush-gambhir/ncu-lab-manual-and-end-semester-projects/blob/main/NCU-CSL312%20-%20DL%20-%20Lab%20Manual/Experiment%201/Experiment%201.ipynb

Google Colab Link:

Open in Colab

Installing Dependencies:

In [ ]:
```
! pip install tensorflow-cpu numpy matplotlib keras
```

Code

In [ ]:
```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras

# Constants and Variables
x = tf.constant([[1., 2., 3.], [4., 5., 6.]])
a = tf.constant([[1, 2], [3, 4]])
b = tf.constant([[1, 1], [1, 1]])
c = tf.constant([[4.0, 5.0], [10.0, 1.0]])

# Basic Tensor Operations
print(x)
print("Shape:", x.shape)
print("DType:", x.dtype)
print("Element-wise addition:", x + x)
print("Scalar multiplication:", 5 * x)

# Concatenation and Mathematical Operations
print("Concatenated:", tf.concat([x, x, x], axis=0))
print("Softmax:", tf.nn.softmax(x, axis=-1))
print("Sum:", tf.reduce_sum(x))

# Element-wise and Matrix Operations
print("Addition:\n", a + b)
print("Element-wise Multiplication:\n", a * b)
print("Matrix Multiplication:\n", tf.matmul(a, b))

# Advanced Operations
print("Max Value:", tf.reduce_max(c))
print("Argmax:", tf.math.argmax(c))
print("Softmax:\n", tf.nn.softmax(c))

# Variable operations and Gradient Computation
var = tf.Variable([0.0, 0.0, 0.0])
var.assign([1, 2, 3])
var.assign_add([1, 1, 1])

x_var = tf.Variable(1.0)
with tf.GradientTape() as tape:
    y = x_var**2 + 2 * x_var - 5
g_x = tape.gradient(y, x_var)
print("Gradient dy/dx:", g_x.numpy())

# tf.function for Graph Execution


@tf.function
def my_func(x):
    return tf.reduce_sum(x)
```

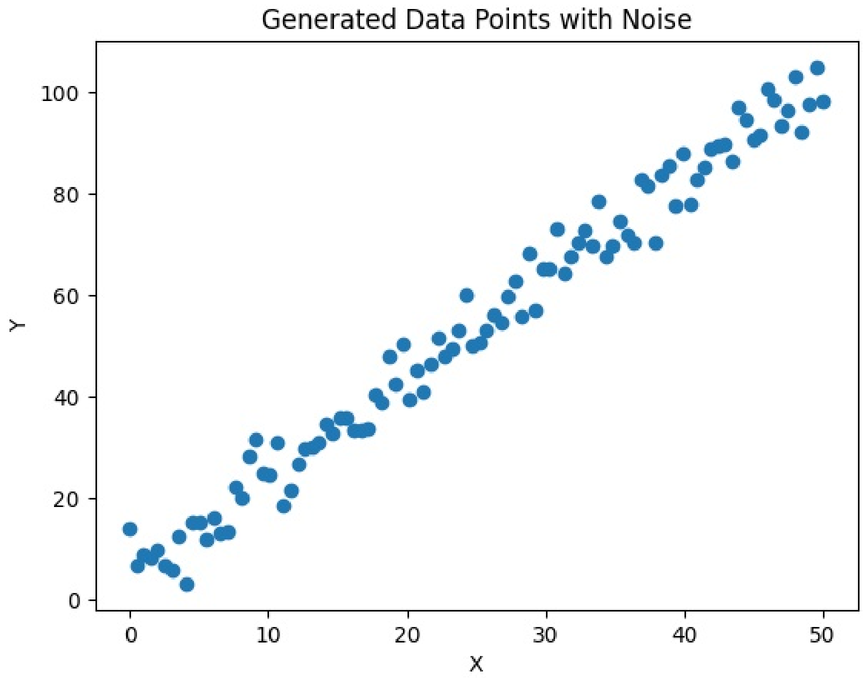
```python
print("tf.function example:", my_func(tf.constant([1, 2, 3])))

# TensorFlow Module


class MyModule(tf.Module):
    def __init__(self, value):
        super(MyModule, self).__init__()
        self.weight = tf.Variable(value)

    @tf.function
    def multiply(self, x):
        return x * self.weight


mod = MyModule(3)
print("Module example:", mod.multiply(tf.constant([1, 2, 3])))

# Simple Linear Model with Keras
model = keras.Sequential([
    keras.layers.Dense(units=1, input_shape=[1])
])
model.compile(optimizer='sgd', loss='mean_squared_error')
xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)
model.fit(xs, ys, epochs=1000, verbose=0)

# Convert the list [10.0] to a numpy array with shape (1, 1) for prediction
x_predict = np.array([10.0]).reshape(-1, 1)
predicted_value = model.predict(x_predict)
print("Model prediction for x=10.0:", predicted_value[0][0])
```

```
tf.Tensor(
[[1. 2. 3.]
 [4. 5. 6.]], shape=(2, 3), dtype=float32)
Shape: (2, 3)
DType: <dtype: 'float32'>
Element-wise addition: tf.Tensor(
[[ 2.  4.  6.]
 [ 8. 10. 12.]], shape=(2, 3), dtype=float32)
Scalar multiplication: tf.Tensor(
[[ 5. 10. 15.]
 [20. 25. 30.]], shape=(2, 3), dtype=float32)
Concatenated: tf.Tensor(
[[1. 2. 3.]
 [4. 5. 6.]
 [1. 2. 3.]
 [4. 5. 6.]
 [1. 2. 3.]
 [4. 5. 6.]], shape=(6, 3), dtype=float32)
Softmax: tf.Tensor(
[[0.09003057 0.24472848 0.6652409 ]
 [0.09003057 0.24472848 0.6652409 ]], shape=(2, 3), dtype=float32)
Sum: tf.Tensor(21.0, shape=(), dtype=float32)
Addition:
 tf.Tensor(
[[2 3]
 [4 5]], shape=(2, 2), dtype=int32)
Element-wise Multiplication:
 tf.Tensor(
[[1 2]
 [3 4]], shape=(2, 2), dtype=int32)
Matrix Multiplication:
 tf.Tensor(
[[3 3]
 [7 7]], shape=(2, 2), dtype=int32)
Max Value: tf.Tensor(10.0, shape=(), dtype=float32)
Argmax: tf.Tensor([1 0], shape=(2,), dtype=int64)
Softmax:
 tf.Tensor(
[[2.6894143e-01 7.3105854e-01]
 [9.9987662e-01 1.2339458e-04]], shape=(2, 2), dtype=float32)
Gradient dy/dx: 4.0
tf.function example: tf.Tensor(6, shape=(), dtype=int32)
Module example: tf.Tensor([3 6 9], shape=(3,), dtype=int32)
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step
Model prediction for x=10.0: 18.999922
```

In [ ]:

```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import tensorflow as tf
```

```python
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Data Generation
np.random.seed(101)
x = np.linspace(0, 50, 100)
noise = np.random.normal(loc=0.0, scale=4.0, size=len(x))
y = 2 * x + 3 + noise  # y = mx + b + noise
plt.scatter(x, y)
plt.title('Generated Data Points with Noise')
plt.xlabel('X')
plt.ylabel('Y')
plt.show()

# Neural Network Model for Regression
model = Sequential([
    Dense(4, input_dim=1, activation='relu'),
    Dense(4, activation='relu'),
    Dense(1, activation='linear')
])
model.compile(loss='mse', optimizer='adam')
model.fit(x, y, epochs=500, verbose=1)
model.summary()

# Predictions and Evaluation
x_for_predictions = np.linspace(0, 50, 1000)
y_predicted = model.predict(x_for_predictions)
predictions = model.predict(x).flatten()

mse = mean_squared_error(y, predictions)
mae = mean_absolute_error(y, predictions)
print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error: {mae}")

plt.scatter(x, y, label='Original Data')
plt.plot(x_for_predictions, y_predicted, 'r', label='Line of Best Fit')
plt.title('Original Data and Predicted Line of Best Fit')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.show()

# Data Loading and Preparation
df = pd.read_csv('fake_reg.csv')
sns.pairplot(df)
plt.show()

X = df[['feature1', 'feature2']].values
y = df['price'].values
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42)

scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Model for Predicting Prices
price_model = Sequential([
    Dense(4, input_shape=[2], activation='relu'),
    Dense(4, activation='relu'),
    Dense(1)
])
price_model.compile(optimizer='rmsprop', loss='mse')

# Fit the model and capture the history
history = price_model.fit(X_train_scaled, y_train, epochs=250, verbose=0)

# Model Evaluation
train_loss = price_model.evaluate(X_train_scaled, y_train, verbose=0)
test_loss = price_model.evaluate(X_test_scaled, y_test, verbose=0)
print(f"Training Loss: {train_loss}")
print(f"Test Loss: {test_loss}")

# Plot Training Loss
loss = history.history['loss']
plt.plot(range(len(loss)), loss)
plt.title("Training Loss per Epoch")
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.show()
```




```
c:\Users\mainp\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\core\dense.py:86: User
Warning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer usin
g an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - loss: 3816.8904
Epoch 2/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3675.2747
Epoch 3/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3616.0183
Epoch 4/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3786.6235
Epoch 5/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3614.7380
Epoch 6/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3600.3745
Epoch 7/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3731.9346
Epoch 8/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3544.4194
Epoch 9/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3867.9727
Epoch 10/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3819.0059
Epoch 11/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3638.5342
Epoch 12/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3738.2158
Epoch 13/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3679.1357
Epoch 14/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3789.3955
Epoch 15/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3729.8909
Epoch 16/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3808.5630
Epoch 17/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3720.5808
Epoch 18/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3800.4888
Epoch 19/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3922.3398
Epoch 20/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3396.6860
Epoch 21/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3751.9087
Epoch 22/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3722.5664
Epoch 23/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3788.3835
Epoch 24/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3823.5667
Epoch 25/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3814.0737
Epoch 26/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3538.2739
```

```
Epoch 27/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 4052.5713
Epoch 28/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3775.7695
Epoch 29/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3720.6047
Epoch 30/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3863.3772
Epoch 31/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3772.0359
Epoch 32/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3683.1426
Epoch 33/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3871.5901
Epoch 34/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3788.1516
Epoch 35/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3814.4290
Epoch 36/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 3506.7356
Epoch 37/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3723.1694
Epoch 38/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3822.1555
Epoch 39/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3647.8518
Epoch 40/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3642.1448
Epoch 41/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3834.0903
Epoch 42/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3767.7690
Epoch 43/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3666.2305
Epoch 44/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 3787.4944
Epoch 45/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3585.8687
Epoch 46/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3544.5134
Epoch 47/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3781.7695
Epoch 48/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3680.3552
Epoch 49/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 3668.9290
Epoch 50/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3537.4807
Epoch 51/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3689.3765
Epoch 52/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3855.9463
Epoch 53/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3758.1865
Epoch 54/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3819.4304
Epoch 55/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3773.0984
Epoch 56/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3617.3213
Epoch 57/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3531.2065
Epoch 58/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3621.2222
Epoch 59/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3594.0544
Epoch 60/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3549.4194
Epoch 61/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3401.9023
Epoch 62/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3801.1130
Epoch 63/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3791.5771
Epoch 64/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3904.9458
Epoch 65/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3530.1377
Epoch 66/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3497.6643
Epoch 67/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3849.0090
Epoch 68/500
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3727.5684
Epoch 69/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3599.6113
Epoch 70/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3535.2051
Epoch 71/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3587.6843
Epoch 72/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3509.0911
Epoch 73/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3870.8784
Epoch 74/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3798.2300
Epoch 75/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3791.4351
Epoch 76/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3647.5059
Epoch 77/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3563.2153
Epoch 78/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3664.1831
Epoch 79/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3700.1953
Epoch 80/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3486.6387
Epoch 81/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3876.1418
Epoch 82/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3756.5974
Epoch 83/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3595.6028
Epoch 84/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3649.3088
Epoch 85/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3601.9756
Epoch 86/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3534.1589
Epoch 87/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3663.2859
Epoch 88/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 3457.1731
Epoch 89/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3707.9893
Epoch 90/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3850.4265
Epoch 91/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 3573.3037
Epoch 92/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3787.0315
Epoch 93/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3530.4341
Epoch 94/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3726.6323
Epoch 95/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3707.8538
Epoch 96/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3675.9597
Epoch 97/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3565.5083
Epoch 98/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3821.0054
Epoch 99/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3659.5000
Epoch 100/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 3803.7878
Epoch 101/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3683.3623
Epoch 102/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3575.9302
Epoch 103/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3881.7678
Epoch 104/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3521.8792
Epoch 105/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3663.3721
Epoch 106/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3762.2756
Epoch 107/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3466.2732
Epoch 108/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3778.4080
Epoch 109/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3865.4961
```

```
Epoch 110/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3657.7354
Epoch 111/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3577.7549
Epoch 112/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3555.4988
Epoch 113/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3402.5037
Epoch 114/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3601.8081
Epoch 115/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3877.0283
Epoch 116/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3460.6855
Epoch 117/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3695.4744
Epoch 118/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3547.4724
Epoch 119/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3823.6931
Epoch 120/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3749.3816
Epoch 121/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3551.9087
Epoch 122/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3708.9658
Epoch 123/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3624.6333
Epoch 124/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3543.9485
Epoch 125/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3637.2446
Epoch 126/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3528.6948
Epoch 127/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3726.1416
Epoch 128/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3561.2117
Epoch 129/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3685.8352
Epoch 130/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3792.5557
Epoch 131/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3508.3396
Epoch 132/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3611.1714
Epoch 133/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3676.2795
Epoch 134/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3486.2947
Epoch 135/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3752.9761
Epoch 136/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3597.2219
Epoch 137/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3823.5085
Epoch 138/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3442.4275
Epoch 139/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 3827.8149
Epoch 140/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3535.0696
Epoch 141/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3582.6538
Epoch 142/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3740.8047
Epoch 143/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3583.7705
Epoch 144/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3692.0676
Epoch 145/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3339.5420
Epoch 146/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3575.7456
Epoch 147/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3555.1521
Epoch 148/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3795.9875
Epoch 149/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3550.7166
Epoch 150/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3612.2913
Epoch 151/500
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3668.6956
Epoch 152/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3428.4895
Epoch 153/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3717.8928
Epoch 154/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3665.3367
Epoch 155/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3798.4753
Epoch 156/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3694.0249
Epoch 157/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3592.8997
Epoch 158/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3570.4670
Epoch 159/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3610.2344
Epoch 160/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3646.8958
Epoch 161/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3790.3738
Epoch 162/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3686.1350
Epoch 163/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3595.1726
Epoch 164/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3685.2866
Epoch 165/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3610.4939
Epoch 166/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3759.0813
Epoch 167/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3503.3552
Epoch 168/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3902.9788
Epoch 169/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3749.9226
Epoch 170/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3570.8782
Epoch 171/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3676.9365
Epoch 172/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3787.3030
Epoch 173/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3285.4719
Epoch 174/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3738.2148
Epoch 175/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3722.9512
Epoch 176/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3859.4263
Epoch 177/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3531.3284
Epoch 178/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3542.4065
Epoch 179/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3951.8823
Epoch 180/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3601.0166
Epoch 181/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3665.3298
Epoch 182/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3785.4243
Epoch 183/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3615.0276
Epoch 184/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3578.2012
Epoch 185/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3547.1123
Epoch 186/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3948.3577
Epoch 187/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3787.9390
Epoch 188/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3595.8418
Epoch 189/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3734.6472
Epoch 190/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3612.0278
Epoch 191/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3693.4551
Epoch 192/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3711.5449
```

```
Epoch 193/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3581.8889
Epoch 194/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 3506.9919
Epoch 195/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3611.2136
Epoch 196/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3627.6519
Epoch 197/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3451.8752
Epoch 198/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3718.3813
Epoch 199/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3618.7676
Epoch 200/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3684.1008
Epoch 201/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3488.1860
Epoch 202/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3566.0291
Epoch 203/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3641.5952
Epoch 204/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3890.7412
Epoch 205/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3665.7502
Epoch 206/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3487.6699
Epoch 207/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3779.0593
Epoch 208/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3736.5266
Epoch 209/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3628.9646
Epoch 210/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3792.7427
Epoch 211/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3514.5664
Epoch 212/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3654.5886
Epoch 213/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3823.3696
Epoch 214/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3664.6343
Epoch 215/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3659.7388
Epoch 216/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3665.6597
Epoch 217/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3681.1626
Epoch 218/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3714.9773
Epoch 219/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3735.9863
Epoch 220/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3583.6680
Epoch 221/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3386.2151
Epoch 222/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3663.6296
Epoch 223/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3582.3069
Epoch 224/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3695.7581
Epoch 225/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3746.1292
Epoch 226/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3752.8557
Epoch 227/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3622.3643
Epoch 228/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3698.0437
Epoch 229/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3506.3298
Epoch 230/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3529.7114
Epoch 231/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3594.5698
Epoch 232/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3718.9463
Epoch 233/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3647.6880
Epoch 234/500
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3669.1826
Epoch 235/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3814.9971
Epoch 236/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3625.0571
Epoch 237/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3615.2637
Epoch 238/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3697.3149
Epoch 239/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3550.8381
Epoch 240/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3699.4607
Epoch 241/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3607.7410
Epoch 242/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3568.6294
Epoch 243/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3595.2551
Epoch 244/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3672.4287
Epoch 245/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3741.0073
Epoch 246/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3550.4480
Epoch 247/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3858.6042
Epoch 248/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3607.4082
Epoch 249/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3713.8374
Epoch 250/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3493.0554
Epoch 251/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3716.6240
Epoch 252/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3494.5198
Epoch 253/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3503.6211
Epoch 254/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3549.8018
Epoch 255/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3619.2717
Epoch 256/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3425.4307
Epoch 257/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3591.0955
Epoch 258/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3617.0510
Epoch 259/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3730.2864
Epoch 260/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3760.6926
Epoch 261/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3480.3313
Epoch 262/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3680.2073
Epoch 263/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3577.4744
Epoch 264/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3598.9878
Epoch 265/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3557.6548
Epoch 266/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3580.0815
Epoch 267/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3443.1519
Epoch 268/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3607.0271
Epoch 269/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3751.3823
Epoch 270/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3598.9592
Epoch 271/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3370.8289
Epoch 272/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3749.4119
Epoch 273/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3765.1172
Epoch 274/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3753.7019
Epoch 275/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3530.5029
```

```
Epoch 276/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3762.7134
Epoch 277/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3647.8623
Epoch 278/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3965.6323
Epoch 279/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3584.4011
Epoch 280/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3809.0718
Epoch 281/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3662.9648
Epoch 282/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3645.3472
Epoch 283/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3657.3889
Epoch 284/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3481.8865
Epoch 285/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3731.1421
Epoch 286/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3430.5081
Epoch 287/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3649.5869
Epoch 288/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3413.8455
Epoch 289/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3618.4810
Epoch 290/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3416.5496
Epoch 291/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3460.8340
Epoch 292/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3669.0669
Epoch 293/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3797.7305
Epoch 294/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3699.3115
Epoch 295/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3474.9795
Epoch 296/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3598.8396
Epoch 297/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3687.4304
Epoch 298/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3549.9939
Epoch 299/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3672.8447
Epoch 300/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3427.4580
Epoch 301/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3588.5837
Epoch 302/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3783.2983
Epoch 303/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3646.0210
Epoch 304/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3685.8931
Epoch 305/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3611.5291
Epoch 306/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3509.1682
Epoch 307/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3665.7490
Epoch 308/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3691.6858
Epoch 309/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3461.4661
Epoch 310/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3729.0508
Epoch 311/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3452.7273
Epoch 312/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3438.2395
Epoch 313/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3399.6455
Epoch 314/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3382.0073
Epoch 315/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3629.9778
Epoch 316/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3454.3887
Epoch 317/500
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3618.9810
Epoch 318/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3439.9214
Epoch 319/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3635.2361
Epoch 320/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3800.1772
Epoch 321/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3781.1365
Epoch 322/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3579.1440
Epoch 323/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3678.7109
Epoch 324/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3768.5339
Epoch 325/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3632.6880
Epoch 326/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3634.0393
Epoch 327/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3466.8225
Epoch 328/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3790.4399
Epoch 329/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3560.4646
Epoch 330/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3512.8240
Epoch 331/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3444.1353
Epoch 332/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3560.3945
Epoch 333/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3759.6860
Epoch 334/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3587.9236
Epoch 335/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3354.6216
Epoch 336/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3655.8999
Epoch 337/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3842.3271
Epoch 338/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3597.5894
Epoch 339/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3623.5762
Epoch 340/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3700.5781
Epoch 341/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3661.3533
Epoch 342/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3481.2271
Epoch 343/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3695.5081
Epoch 344/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3361.4939
Epoch 345/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3602.6375
Epoch 346/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3356.3799
Epoch 347/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3488.1899
Epoch 348/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3624.3035
Epoch 349/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3441.5093
Epoch 350/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3568.8315
Epoch 351/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3781.4521
Epoch 352/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3520.7346
Epoch 353/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3391.4167
Epoch 354/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3540.0449
Epoch 355/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3607.8835
Epoch 356/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3780.1013
Epoch 357/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3523.0718
Epoch 358/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3481.7930
```

```
Epoch 359/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3504.8594
Epoch 360/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3634.7908
Epoch 361/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3379.5002
Epoch 362/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3667.7012
Epoch 363/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3579.1216
Epoch 364/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3443.5920
Epoch 365/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3610.8706
Epoch 366/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3620.2825
Epoch 367/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3428.5823
Epoch 368/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 3645.2622
Epoch 369/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3346.5510
Epoch 370/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3419.8677
Epoch 371/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3613.5229
Epoch 372/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3432.3372
Epoch 373/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3686.5540
Epoch 374/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3697.0808
Epoch 375/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3509.4854
Epoch 376/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3482.5789
Epoch 377/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3927.1260
Epoch 378/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3488.9758
Epoch 379/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3431.9539
Epoch 380/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3522.4609
Epoch 381/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3542.3809
Epoch 382/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3497.3472
Epoch 383/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3804.5195
Epoch 384/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3419.8145
Epoch 385/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 3566.4260
Epoch 386/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3688.1892
Epoch 387/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3496.9563
Epoch 388/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3615.3118
Epoch 389/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3472.7603
Epoch 390/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3523.1501
Epoch 391/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3315.7944
Epoch 392/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3538.7412
Epoch 393/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3736.2139
Epoch 394/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3532.7183
Epoch 395/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3445.5337
Epoch 396/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3695.7170
Epoch 397/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3613.5110
Epoch 398/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3468.6104
Epoch 399/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3463.7615
Epoch 400/500
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3650.0085
Epoch 401/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3731.3623
Epoch 402/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3431.4224
Epoch 403/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3700.9255
Epoch 404/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3497.9287
Epoch 405/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3666.8196
Epoch 406/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3664.5974
Epoch 407/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3716.5508
Epoch 408/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3601.4985
Epoch 409/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3406.0740
Epoch 410/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3599.0664
Epoch 411/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3616.6870
Epoch 412/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3700.8440
Epoch 413/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3593.4841
Epoch 414/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3603.2903
Epoch 415/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3427.0139
Epoch 416/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3426.0276
Epoch 417/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3498.4465
Epoch 418/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3407.2537
Epoch 419/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3585.5059
Epoch 420/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3606.3254
Epoch 421/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3434.1121
Epoch 422/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3425.1160
Epoch 423/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3527.1877
Epoch 424/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3420.0227
Epoch 425/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3597.9490
Epoch 426/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3586.9204
Epoch 427/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3462.8303
Epoch 428/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3628.2527
Epoch 429/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3614.9878
Epoch 430/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3508.2495
Epoch 431/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3546.2881
Epoch 432/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3644.9766
Epoch 433/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3596.0020
Epoch 434/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3566.9639
Epoch 435/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3566.3191
Epoch 436/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3641.2190
Epoch 437/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3493.6919
Epoch 438/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3596.1484
Epoch 439/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3431.0552
Epoch 440/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3769.8232
Epoch 441/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3681.7207
```

```
Epoch 442/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3621.2769
Epoch 443/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3625.0840
Epoch 444/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3457.6482
Epoch 445/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3390.5500
Epoch 446/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3546.9133
Epoch 447/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3669.8533
Epoch 448/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3533.9375
Epoch 449/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3490.2947
Epoch 450/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3392.1423
Epoch 451/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3558.8081
Epoch 452/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3566.5601
Epoch 453/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3414.6689
Epoch 454/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3697.1445
Epoch 455/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3679.3684
Epoch 456/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3449.9719
Epoch 457/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3457.8250
Epoch 458/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3532.8232
Epoch 459/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3741.3855
Epoch 460/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3430.9460
Epoch 461/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3713.0288
Epoch 462/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3383.3713
Epoch 463/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3594.3408
Epoch 464/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3491.2532
Epoch 465/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 3400.2600
Epoch 466/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3360.4033
Epoch 467/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3369.3276
Epoch 468/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3486.7615
Epoch 469/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3643.2195
Epoch 470/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3667.7383
Epoch 471/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3483.6396
Epoch 472/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3817.4968
Epoch 473/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3278.1777
Epoch 474/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3666.5107
Epoch 475/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3512.7146
Epoch 476/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3425.8547
Epoch 477/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3667.4707
Epoch 478/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3402.9075
Epoch 479/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3575.8669
Epoch 480/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3597.7974
Epoch 481/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3645.1619
Epoch 482/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3564.3828
Epoch 483/500
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3410.9966
Epoch 484/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3389.3645
Epoch 485/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3746.4028
Epoch 486/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3418.7637
Epoch 487/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 3426.7507
Epoch 488/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3501.9563
Epoch 489/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3576.9556
Epoch 490/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3598.8433
Epoch 491/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3366.0596
Epoch 492/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3300.2268
Epoch 493/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3499.2441
Epoch 494/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3603.8699
Epoch 495/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3428.2437
Epoch 496/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3550.9141
Epoch 497/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3191.3435
Epoch 498/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3356.8669
Epoch 499/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3345.0784
Epoch 500/500
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3525.1396
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 4)	8
dense_1 (Dense)	(None, 4)	20
dense_2 (Dense)	(None, 1)	5

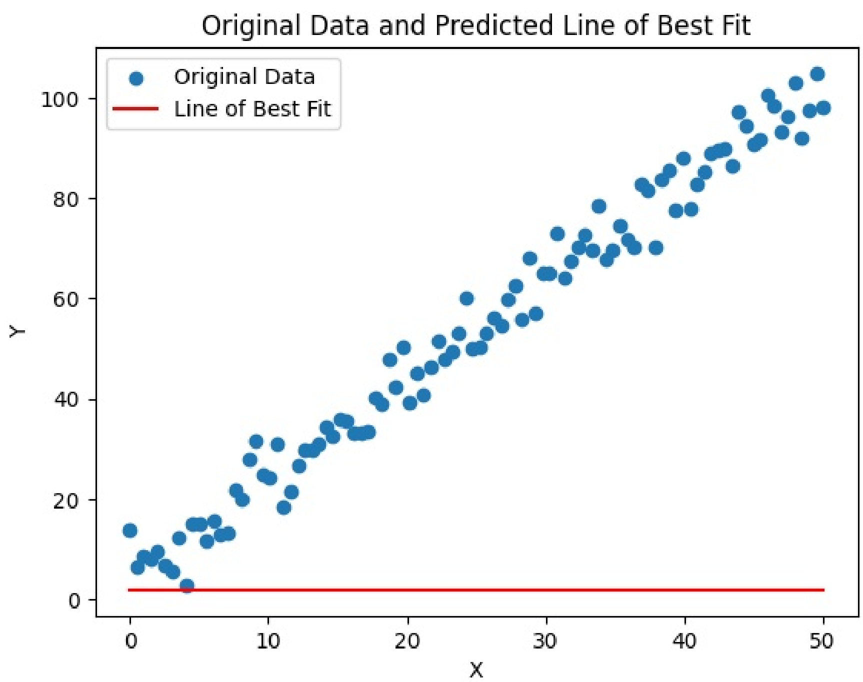
Total params: 101 (408.00 B)

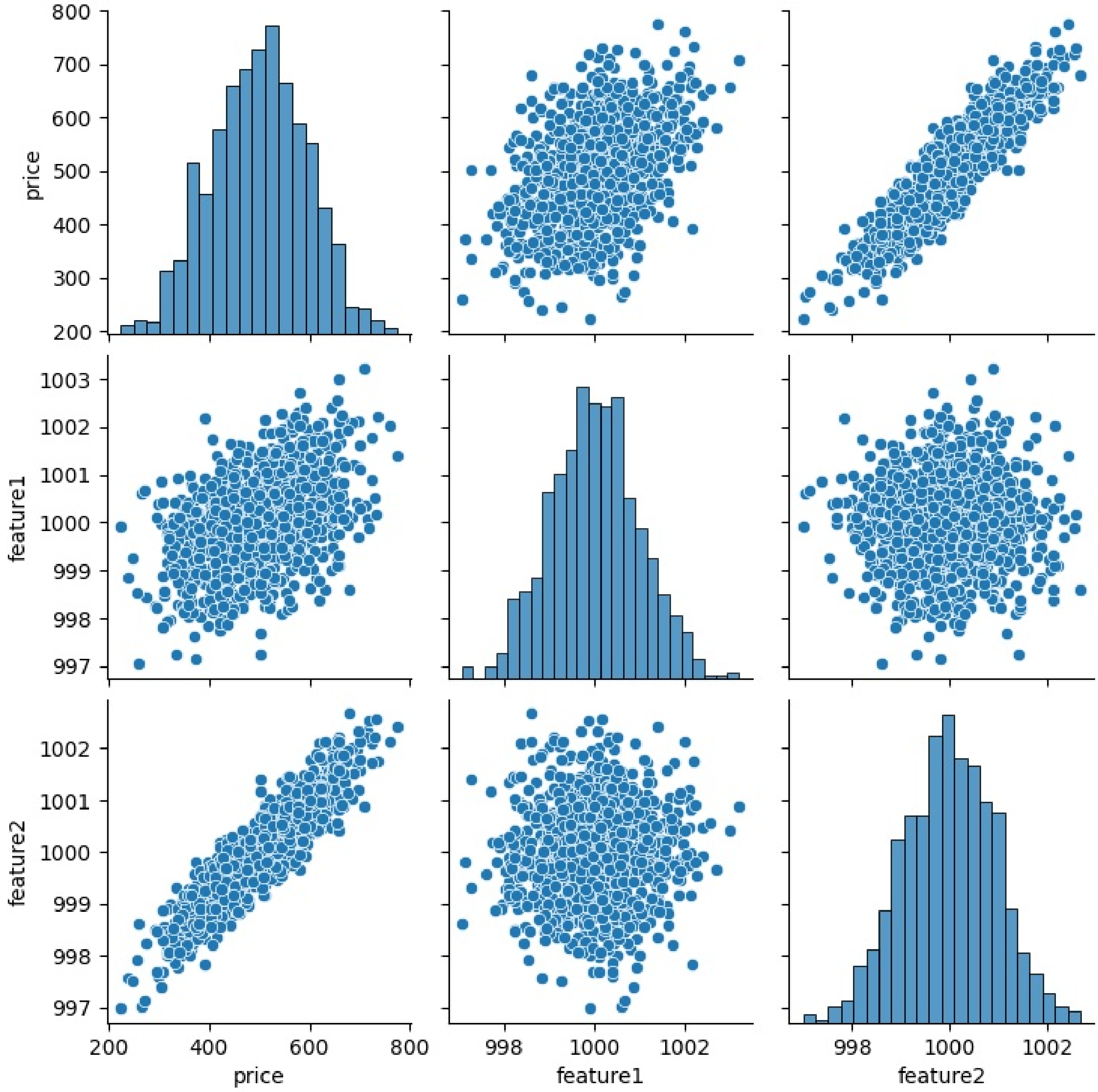
Trainable params: 33 (132.00 B)

Non-trainable params: 0 (0.00 B)

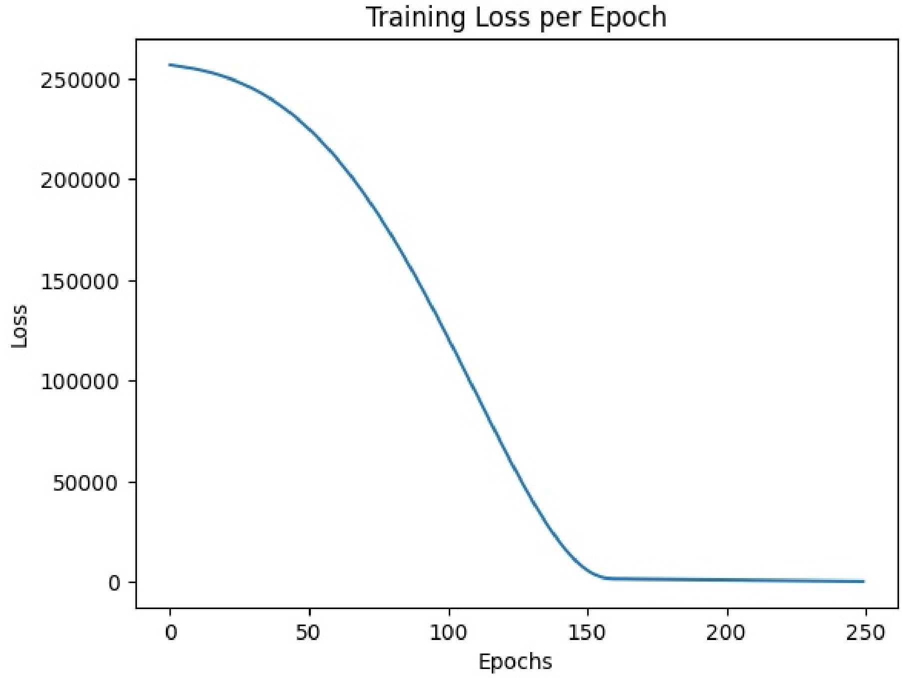
Optimizer params: 68 (276.00 B)

```
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Mean Squared Error: 3508.700145409083
Mean Absolute Error: 51.697276930750064
```






```
c:\Users\mainp\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\core\dense.py:86: User
Warning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer usin
g an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Training Loss: 419.9224548339844
Test Loss: 408.3269958496094
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Experiment No: 2

Student Name and Roll Number: Piyush Gambhir – 21CSU349	
Semester /Section: 6th Semester – AIML-B (A3)	
Link to Code: ncu-lab-manual-and-end-semester-projects/NCU-CSL312 - DL - Lab Manual/Experiment 2/Experiment 2.ipynb at main · piyush-gambhir/ncu-lab-manual-and-end-semester-projects (github.com)	
Date:	
Faculty Signature:	
Marks:	

Objective(s):
To build an ANN model to convert temperature in degree Celsius to Fahrenheit

Experiment 2

Problem Statement:

To build an ANN Model to convert temperature in degree Celsius to Fahrenheit.

GitHub & Google Collab Links:

GitHub Link: https://github.com/piyush-gambhir/ncu-lab-manual-and-end-semester-projects/blob/main/NCU-CSL312%20-%20DL%20-%20Lab%20Manual/Experiment%202/Experiment%202.ipynb

Google Collab Link:

Open in Colab

Installing Dependencies:

! pip install tensorflow numpy matplotlib scikit-learn pandas seaborn

Code

In [ ]:
```
# importing required libraries
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

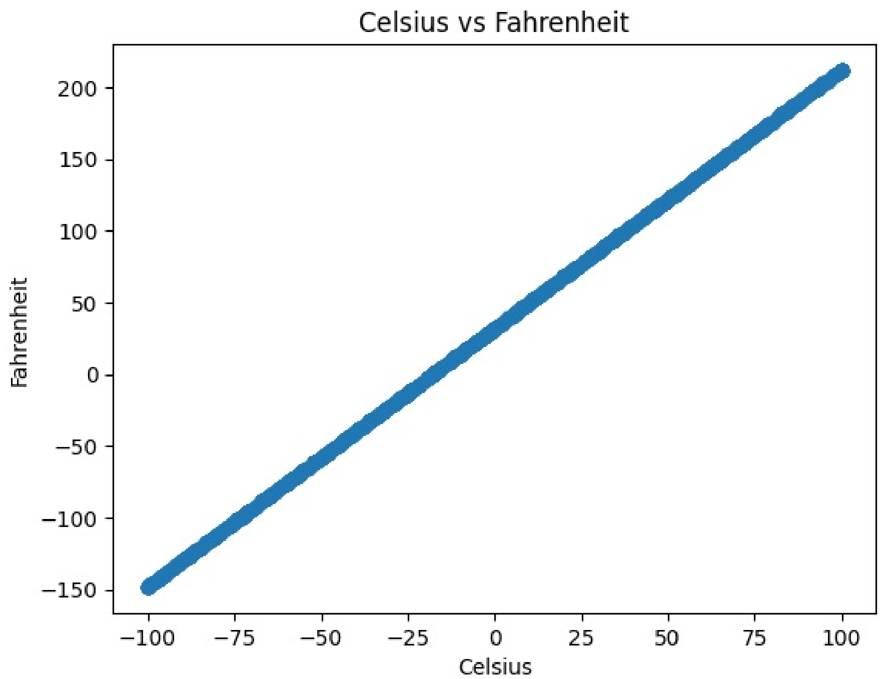
In [ ]:
```
# loading the dataset
dataset = pd.read_csv('celcius_to_fahrenheit_dataset.csv')

# printing the first 5 rows of the dataset
print("First 5 rows of the dataset:")
print(dataset.head())

# printing the last 5 rows of the dataset
print("\nLast 5 rows of the dataset:")
print(dataset.tail())
```

```
First 5 rows of the dataset:
   Celsius  Fahrenheit
0      -67       -88.6
1       40       104.0
2      -97      -142.6
3       57       134.6
4      -50       -58.0

Last 5 rows of the dataset:
     Celsius  Fahrenheit
995      -80      -112.0
996       50       122.0
997       18        64.4
998       47       116.6
999      -67       -88.6
```

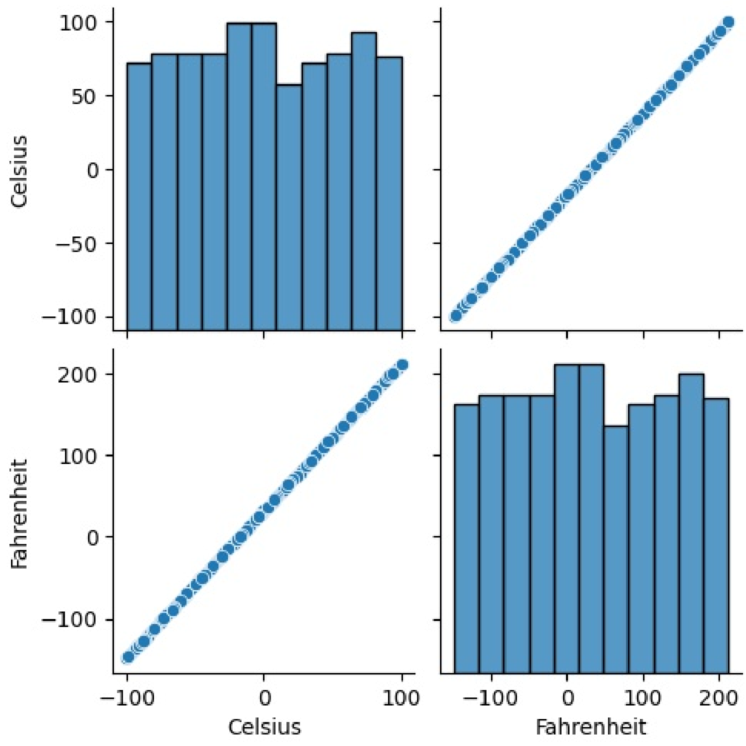
In [ ]:
```
# describing the dataset
print("\nDescription of the dataset:")
print(dataset.describe())

# checking information about the dataset
print("\nInformation about the dataset:")
print(dataset.info())
```

```
Description of the dataset:
           Celsius   Fahrenheit
count  1000.000000  1000.000000
mean     -0.029000    31.947800
std      57.334173   103.201511
min    -100.000000  -148.000000
25%     -50.000000   -58.000000
50%      -2.000000    28.400000
75%      50.000000   122.000000
max     100.000000   212.000000

Information about the dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Celsius     1000 non-null   int64  
 1   Fahrenheit  1000 non-null   float64
dtypes: float64(1), int64(1)
memory usage: 15.8 KB
None
```

In [ ]:
```
# plotting scatter plot between Celsius and Fahrenheit
plt.scatter(dataset['Celsius'], dataset['Fahrenheit'])
plt.title('Celsius vs Fahrenheit')
plt.xlabel('Celsius')
plt.ylabel('Fahrenheit')
plt.show()
```



In [ ]:
```
# plotting the pair plot of the dataset
sns.pairplot(dataset)
plt.show()
```




```
In [ ]: # creating training and testing dataset
X_train = dataset['Celsius']
y_train = dataset['Fahrenheit']

print("Shape of X_train:", X_train.shape)
print("Shape of y_train:", y_train.shape)
```

```
Shape of X_train: (1000,)
Shape of y_train: (1000,)
```

```
In [ ]: # training the model
model = tf.keras.Sequential()
model.add(tf.keras.layers.Dense(units= 32 , input_shape = (1,)))
#Dense when we have fully connected atificial neural network
# now we are adding one more layer to the network
model.add(tf.keras.layers.Dense(units = 32))
# now adding the output layer
model.add(tf.keras.layers.Dense(units = 1))
```

```
c:\Users\mainp\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\core\dense.py:86: User
Warning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer usin
g an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
In [ ]: # model summary
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 32)	64
dense_1 (Dense)	(None, 32)	1,056
dense_2 (Dense)	(None, 1)	33

Total params: 1,153 (4.50 KB)

Trainable params: 1,153 (4.50 KB)

Non-trainable params: 0 (0.00 B)

```
In [ ]: # Compiling the model with a correct learning rate format
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1.0),  # Use a float for learning rate
              loss='mean_squared_error')

# Training the model
epochs_hist = model.fit(X_train, y_train, epochs=30, validation_split=0.2)
```

```
Epoch 1/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 184185392.0000 - val_loss: 19116780.0000
Epoch 2/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 8236354.5000 - val_loss: 225760.6562
Epoch 3/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 179850.9219 - val_loss: 13941.5430
Epoch 4/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 12388.3965 - val_loss: 1299.8925
Epoch 5/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 1352.4651 - val_loss: 669.6835
Epoch 6/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 537.3693 - val_loss: 290.2394
Epoch 7/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 242.8764 - val_loss: 124.3791
Epoch 8/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 101.0022 - val_loss: 47.9577
Epoch 9/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 38.3891 - val_loss: 16.8620
Epoch 10/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 13.1718 - val_loss: 5.3163
Epoch 11/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 4.1010 - val_loss: 1.5207
Epoch 12/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 1.1502 - val_loss: 0.3860
Epoch 13/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.2851 - val_loss: 0.0883
Epoch 14/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.0644 - val_loss: 0.0180
Epoch 15/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.0129 - val_loss: 0.0033
Epoch 16/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0022 - val_loss: 5.1203e-04
Epoch 17/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3.4311e-04 - val_loss: 7.0061e-05
Epoch 18/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 4.5889e-05 - val_loss: 8.2502e-06
Epoch 19/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 5.3816e-06 - val_loss: 8.4195e-07
Epoch 20/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 5.2030e-07 - val_loss: 9.0222e-08
Epoch 21/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 6.0556e-08 - val_loss: 1.1120e-08
Epoch 22/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 9.3749e-09 - val_loss: 6.0435e-09
Epoch 23/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 5.2943e-09 - val_loss: 4.5255e-09
Epoch 24/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 4.3170e-09 - val_loss: 3.9230e-09
Epoch 25/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3.5315e-09 - val_loss: 2.9899e-09
Epoch 26/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 2.8870e-09 - val_loss: 2.6312e-09
Epoch 27/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 2.5734e-09 - val_loss: 2.1170e-09
Epoch 28/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 1.9519e-09 - val_loss: 1.8936e-09
Epoch 29/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 1.7950e-09 - val_loss: 1.6838e-09
Epoch 30/30
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 1.5256e-09 - val_loss: 1.4503e-09
```

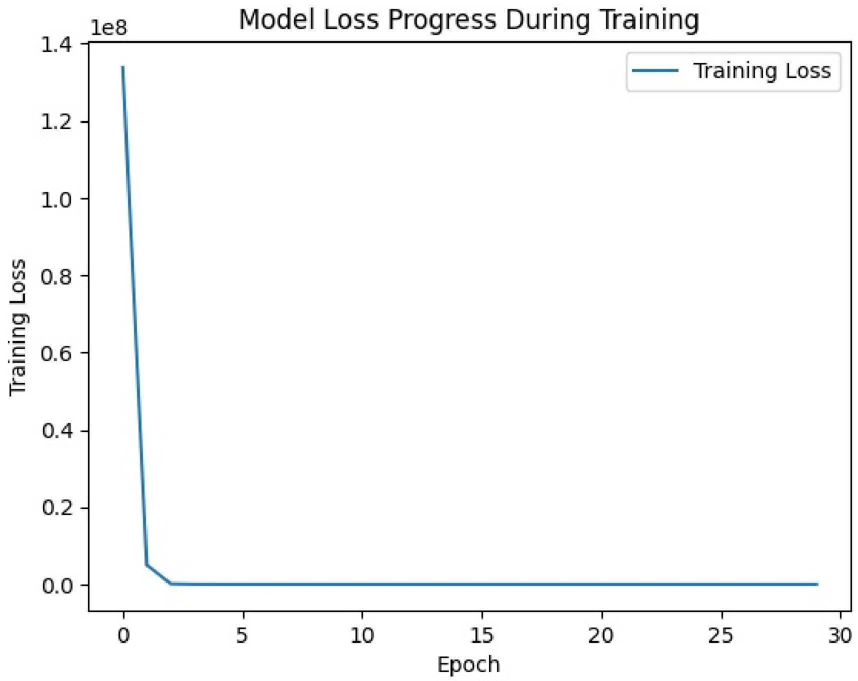
In [ ]:

```python
# evaluating the model
print("Loss of the model:", epochs_hist.history['loss'][-1])
print("Validation Loss of the model:", epochs_hist.history['val_loss'][-1])

# plotting the loss
plt.plot(epochs_hist.history['loss'])
plt.title('Model Loss Progress During Training')
plt.xlabel('Epoch')
plt.ylabel('Training Loss')
plt.legend(['Training Loss'])
```

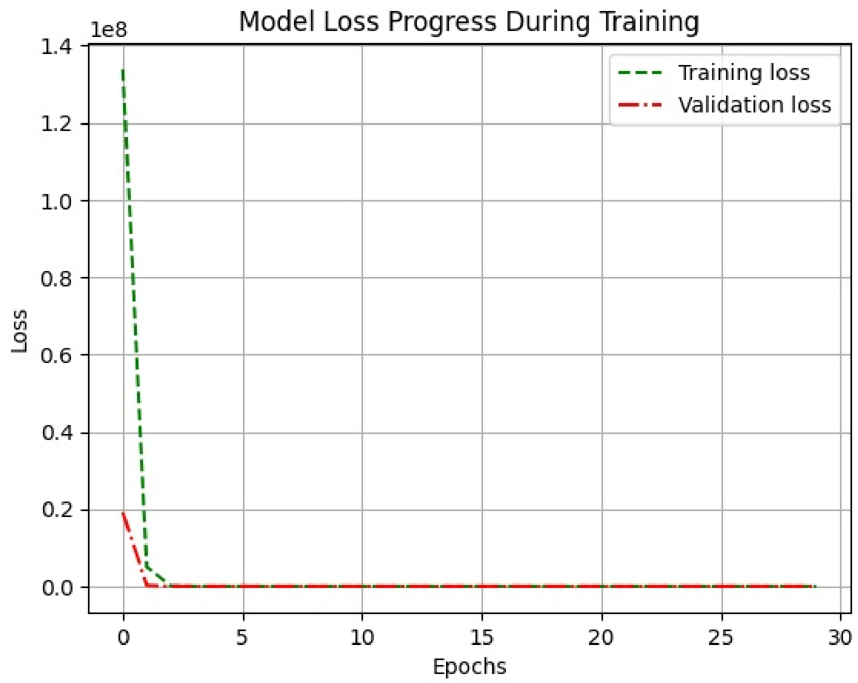
```
Loss of the model: 1.499840363017313e-09
Validation Loss of the model: 1.4503127587772724e-09
```

Out[ ]:

```
<matplotlib.legend.Legend at 0x1ef20de7250>
```



In [ ]:
```
# plotting the loss and validation loss together
plt.plot(epochs_hist.history['loss'], color='green', label='Training loss', linestyle='--')
plt.plot(epochs_hist.history['val_loss'], color='red', label='Validation loss', linestyle='-.')
plt.title('Model Loss Progress During Training')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)
plt.show()
```



In [ ]:
```
# model weights
print("Model Weights:", model.get_weights())
```

```
Model Weights: [array([[ 0.25968587,  0.22125825,  0.28224224, -0.17559958,  0.23964815,
        -0.22765678,  0.21759656,  0.35055447,  0.14478815, -0.26485002,
         0.19265151, -0.13548216, -0.2397434 , -0.19253737,  0.16309   ,
         0.30045104, -0.20057109,  0.20200449,  0.31095803,  0.13719064,
         0.22023755,  0.23763582, -0.23571041,  0.21217768, -0.19351862,
         0.17558281, -0.22309747,  0.16892143, -0.34211284,  0.06600088,
         0.316667  , -0.17547919]], dtype=float32), array([-2.8962476 ,  1.1375874 ,  1.0828029 , -4.1755123 , -
3.1734562 ,
        0.40072462,  1.2057605 , -0.25412676,  5.3564715 ,  3.0925992 ,
        3.9584725 , -1.82679   , -0.6908085 , -3.962711  ,  5.1209865 ,
       -3.2419553 , -0.99446344,  2.322351  , -0.59030426,  5.5396786 ,
       -1.2115853 ,  0.74181646, -1.8299431 ,  4.683813  ,  2.735801  ,
        5.0844874 , -0.9813589 ,  5.412247  ,  0.56680304, -2.441406  ,
       -0.47981733, -4.6937227 ], dtype=float32), array([[-1.739051  ,  2.485136  ,  2.1538115 , ..., -2.3576014
,
        -1.8810381 ,  2.283118  ],
       [-1.3970966 ,  1.282545  ,  1.1863807 , ..., -1.3349036 ,
        -1.2531143 ,  0.9784058 ],
       [ 1.0310407 , -1.3818057 , -1.2115515 , ...,  0.7019355 ,
         0.7162694 , -1.5180666 ],
       ...,
       [-1.8380595 ,  1.8653036 ,  1.6288487 , ..., -1.6006715 ,
        -1.5952858 ,  2.1657426 ],
       [ 0.9168684 , -1.1619155 , -0.6694877 , ...,  0.3581251 ,
         0.07749831, -1.3688766 ],
       [ 0.5317285 , -0.09177828, -0.7525865 , ...,  0.36294675,
         0.545534  , -0.34297764]], dtype=float32), array([ -2.4932847 ,  -9.675127  ,   0.42501694,  10.065001
,
        -0.20992652,   3.964413  ,   9.729676  ,   9.914365  ,
        -4.3035593 ,  -9.414612  ,  -2.7395303 ,   9.503131  ,
        10.006139  ,  -9.70948   ,  -3.6765878 ,   2.4854155 ,
         0.9277671 , -10.061466  ,   3.09827   ,   4.140408  ,
        -3.8939774 ,  -5.2399974 ,  -9.273331  ,  -2.40441   ,
        -9.771945  ,  -9.158293  ,  -1.0062532 ,  -3.8165867 ,
        -9.294314  ,   9.602301  ,   9.229294  ,  -9.535021  ],
      dtype=float32), array([[-0.01261254],
       [-0.16410564],
       [-0.18116668],
       [ 0.17280662],
       [-0.18493941],
       [ 0.07949521],
       [ 0.17311478],
       [ 0.07412434],
       [-0.09236651],
       [ 0.03712983],
       [ 0.07558435],
       [ 0.04765628],
       [ 0.14957748],
       [-0.16422854],
       [-0.00980624],
       [-0.09619454],
       [-0.04052752],
       [-0.06529744],
       [ 0.0081004 ],
       [ 0.12431894],
       [-0.11465567],
       [-0.18500377],
       [ 0.02729801],
       [ 0.03722989],
       [-0.27393368],
       [ 0.09166551],
       [ 0.10329478],
       [-0.04095844],
       [-0.22640787],
       [-0.03853676],
       [-0.01387837],
       [-0.23503576]], dtype=float32), array([8.492162], dtype=float32)]
```

In [ ]:

```
# Making predictions
# Convert to a numpy array and keep it as a batch of one element
Celsius_value = np.array([100])
Fahrenheit_value = model.predict(Celsius_value)
print("Fahrenheit value for Celsius value 100:", Fahrenheit_value[0])

# Calculating with formula
Fahrenheit_value_formula = 9/5 * Celsius_value[0] + 32
print("Fahrenheit value for Celsius value 100 using formula:",
      Fahrenheit_value_formula)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step
Fahrenheit value for Celsius value 100: [212.00005]
Fahrenheit value for Celsius value 100 using formula: 212.0
```

In [ ]:

```
# saving the model
```

```
model.save('celcius_to_fahrenheit_model.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. T
his file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_m
odel.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
```

Experiment No: 3

Student Name and Roll Number: Piyush Gambhir – 21CSU349	
Semester /Section: 6th Semester – AIML-B (A3)	
Link to Code: ncu-lab-manual-and-end-semester-projects/NCU-CSL312 - DL - Lab Manual/Experiment 3/Experiment 3.ipynb at main · piyush-gambhir/ncu-lab-manual-and-end-semester-projects (github.com)	
Date:	
Faculty Signature:	
Marks:	

Objective(s):
To build an ANN model for regression problem for house price predication dataset

Experiment 3: Keras Regression - Housing Prices Prediction

Problem Statement:

To build an ANN model for regression problem on house predication dataset.

GitHub & Colab Links:

GitHub Link: https://github.com/piyush-gambhir/ncu-lab-manual-and-end-semester-projects/blob/main/NCU-CSL312%20-%20DL%20-%20Lab%20Manual/Experiment%203/Experiment%203.ipynb

Google Colab Link:

Open in Colab

Dataset

Dataset Link: https://www.kaggle.com/harlfoxem/housesalesprediction

Dataset Description

This dataset contains house sale prices for King County, which includes Seattle. It includes homes sold between May 2014 and May 2015.

Feature Columns

- id - Unique ID for each home sold
- date - Date of the home sale
- price - Price of each home sold
- bedrooms - Number of bedrooms
- bathrooms - Number of bathrooms, where .5 accounts for a room with a toilet but no shower
- sqft_living - Square footage of the apartments interior living space
- sqft_lot - Square footage of the land space
- floors - Number of floors
- waterfront - A dummy variable for whether the apartment was overlooking the waterfront or not
- view - An index from 0 to 4 of how good the view of the property was
- condition - An index from 1 to 5 on the condition of the apartment,
- grade - An index from 1 to 13, where 1-3 falls short of building construction and design, 7 has an average level of construction and design, and 11-13 have a high quality level of construction and design.
- sqft_above - The square footage of the interior housing space that is above ground level
- sqft_basement - The square footage of the interior housing space that is below ground level
- yr_built - The year the house was initially built
- yr_renovated - The year of the house’s last renovation
- zipcode - What zipcode area the house is in
- lat - Lattitude
- long - Longitude
- sqft_living15 - The square footage of interior housing living space for the nearest 15 neighbors
- sqft_lot15 - The square footage of the land lots of the nearest 15 neighbors

Code

In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, explained_variance_score
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam
```

In [ ]:

```
# Load the Dataset
df = pd.read_csv('kc_house_data.csv')
```

In [ ]:
```python
# Preliminary Data Exploration
print(df.isnull().sum())  # Check for null values
print(df.describe().transpose())  # Summary statistics
```

```
id               0
date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
grade            0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
zipcode          0
lat              0
long             0
sqft_living15    0
sqft_lot15       0
dtype: int64
                 count          mean           std           min  \
id             21597.0  4.580474e+09  2.876736e+09  1.000102e+06
price          21597.0  5.402966e+05  3.673681e+05  7.800000e+04
bedrooms       21597.0  3.373200e+00  9.262989e-01  1.000000e+00
bathrooms      21597.0  2.115826e+00  7.689843e-01  5.000000e-01
sqft_living    21597.0  2.080322e+03  9.181061e+02  3.700000e+02
sqft_lot       21597.0  1.509941e+04  4.141264e+04  5.200000e+02
floors         21597.0  1.494096e+00  5.396828e-01  1.000000e+00
waterfront     21597.0  7.547345e-03  8.654900e-02  0.000000e+00
view           21597.0  2.342918e-01  7.663898e-01  0.000000e+00
condition      21597.0  3.409825e+00  6.505456e-01  1.000000e+00
grade          21597.0  7.657915e+00  1.173200e+00  3.000000e+00
sqft_above     21597.0  1.788597e+03  8.277598e+02  3.700000e+02
sqft_basement  21597.0  2.917250e+02  4.426678e+02  0.000000e+00
yr_built       21597.0  1.971000e+03  2.937523e+01  1.900000e+03
yr_renovated   21597.0  8.446479e+01  4.018214e+02  0.000000e+00
zipcode        21597.0  9.807795e+04  5.351307e+01  9.800100e+04
lat            21597.0  4.756009e+01  1.385518e-01  4.715590e+01
long           21597.0 -1.222140e+02  1.407235e-01 -1.225190e+02
sqft_living15  21597.0  1.986620e+03  6.852305e+02  3.990000e+02
sqft_lot15     21597.0  1.275828e+04  2.727444e+04  6.510000e+02

                        25%           50%           75%           max
id             2.123049e+09  3.904930e+09  7.308900e+09  9.900000e+09
price          3.220000e+05  4.500000e+05  6.450000e+05  7.700000e+06
bedrooms       3.000000e+00  3.000000e+00  4.000000e+00  3.300000e+01
bathrooms      1.750000e+00  2.250000e+00  2.500000e+00  8.000000e+00
sqft_living    1.430000e+03  1.910000e+03  2.550000e+03  1.354000e+04
sqft_lot       5.040000e+03  7.618000e+03  1.068500e+04  1.651359e+06
floors         1.000000e+00  1.500000e+00  2.000000e+00  3.500000e+00
waterfront     0.000000e+00  0.000000e+00  0.000000e+00  1.000000e+00
view           0.000000e+00  0.000000e+00  0.000000e+00  4.000000e+00
condition      3.000000e+00  3.000000e+00  4.000000e+00  5.000000e+00
grade          7.000000e+00  7.000000e+00  8.000000e+00  1.300000e+01
sqft_above     1.190000e+03  1.560000e+03  2.210000e+03  9.410000e+03
sqft_basement  0.000000e+00  0.000000e+00  5.600000e+02  4.820000e+03
yr_built       1.951000e+03  1.975000e+03  1.997000e+03  2.015000e+03
yr_renovated   0.000000e+00  0.000000e+00  0.000000e+00  2.015000e+03
zipcode        9.803300e+04  9.806500e+04  9.811800e+04  9.819900e+04
lat            4.747110e+01  4.757180e+01  4.767800e+01  4.777760e+01
long          -1.223280e+02 -1.222310e+02 -1.221250e+02 -1.213150e+02
sqft_living15  1.490000e+03  1.840000e+03  2.360000e+03  6.210000e+03
sqft_lot15     5.100000e+03  7.620000e+03  1.008300e+04  8.712000e+05
```

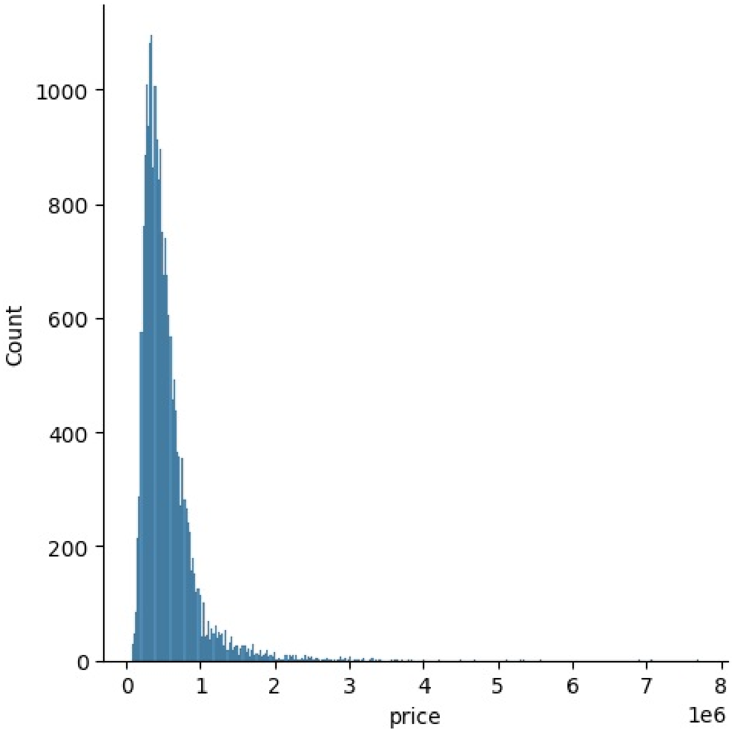
In [ ]:
```python
# Visualizing the Data
print("Visualizing the Data")

plt.figure(figsize=(12, 8))
sns.displot(df['price'])  # Distribution of house prices

plt.figure(figsize=(12, 8))
sns.countplot(df['bedrooms'])  # Count of bedrooms

plt.figure(figsize=(12, 8))
sns.scatterplot(x='price', y='sqft_living', data=df)  # Price vs. living area

plt.figure(figsize=(12, 8))
```

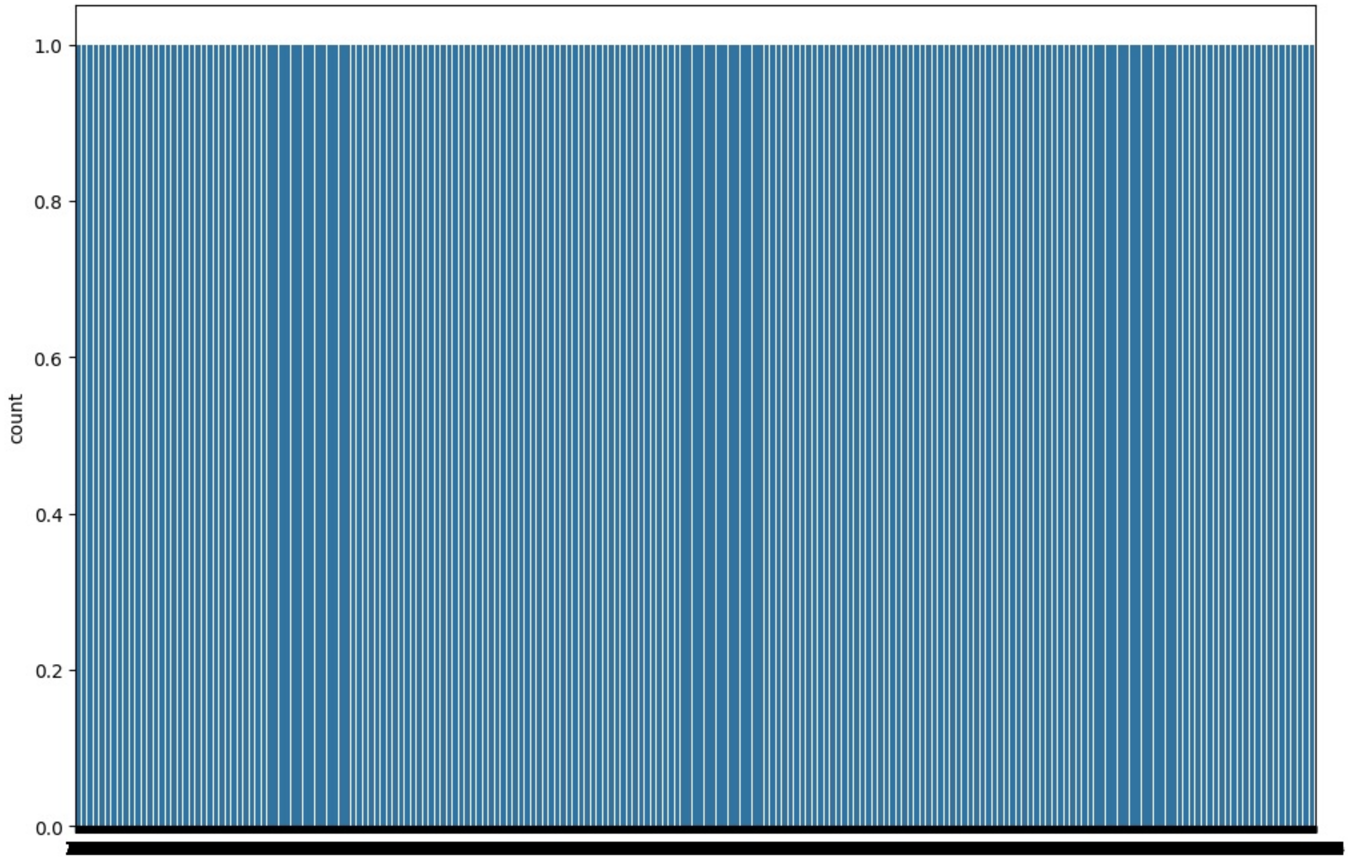
```
sns.boxplot(x='bedrooms', y='price', data=df)  # Price by bedroom count
```

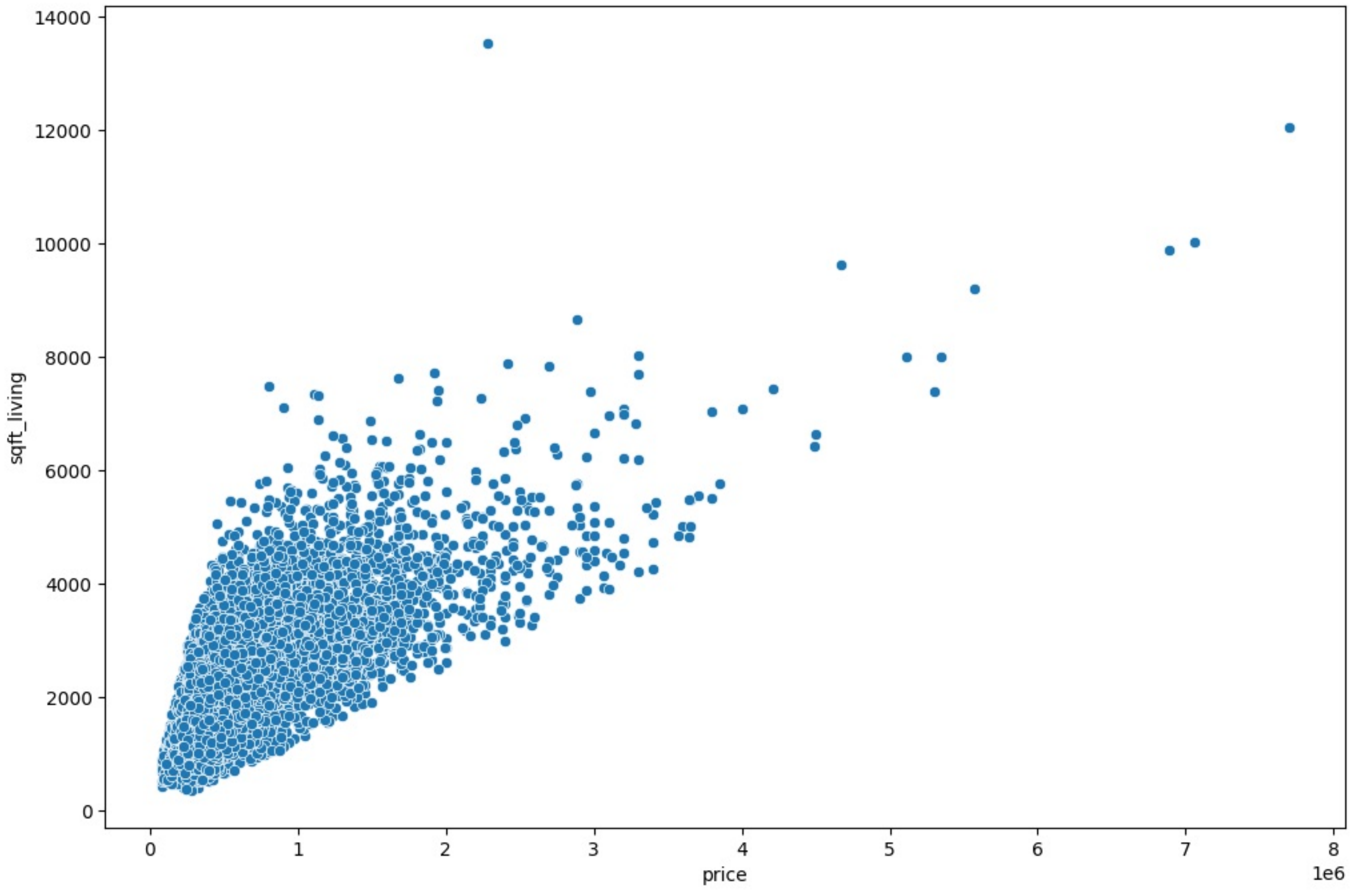
```
Visualizing the Data
```

Out[ ]:

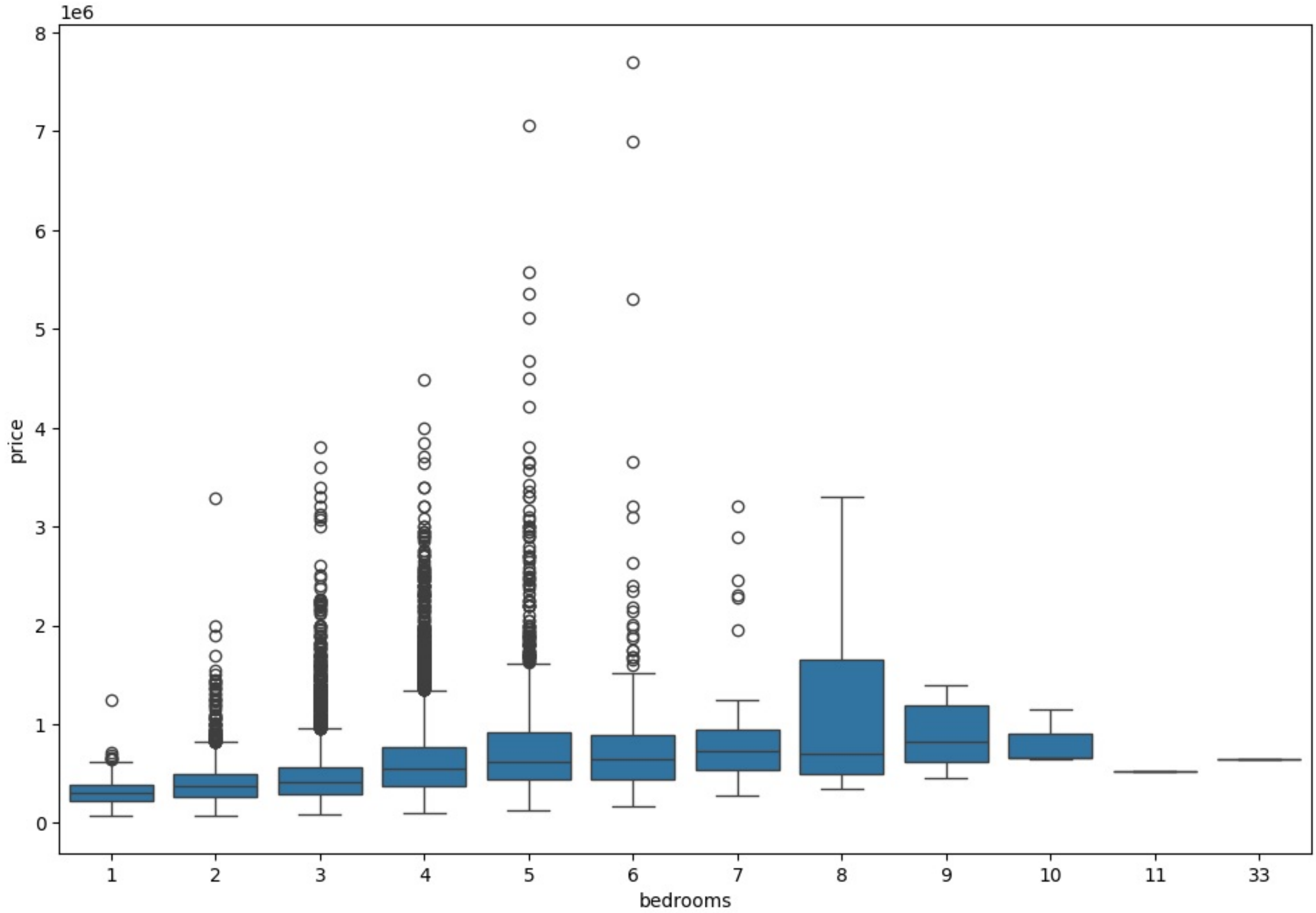
```
<Axes: xlabel='bedrooms', ylabel='price'>
<Figure size 1200x800 with 0 Axes>
```







14000
12000
10000
8000
6000
4000
2000
0
sqft_living
0
1
2
3
4
5
6
7
8
1e6
price



In [ ]:
```
# Dropping unnecessary features
df.drop(['id', 'date', 'zipcode'], axis=1, inplace=True)

# Feature Engineering from 'yr_renovated' and 'sqft_basement'
df['yr_renovated'] = df['yr_renovated'].apply(lambda x: 1 if x > 0 else 0)
df['has_basement'] = df['sqft_basement'].apply(lambda x: 1 if x > 0 else 0)
df.drop('sqft_basement', axis=1, inplace=True)

# Scaling and Train Test Split
X = df.drop('price', axis=1).values
y = df['price'].values
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=101)

scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

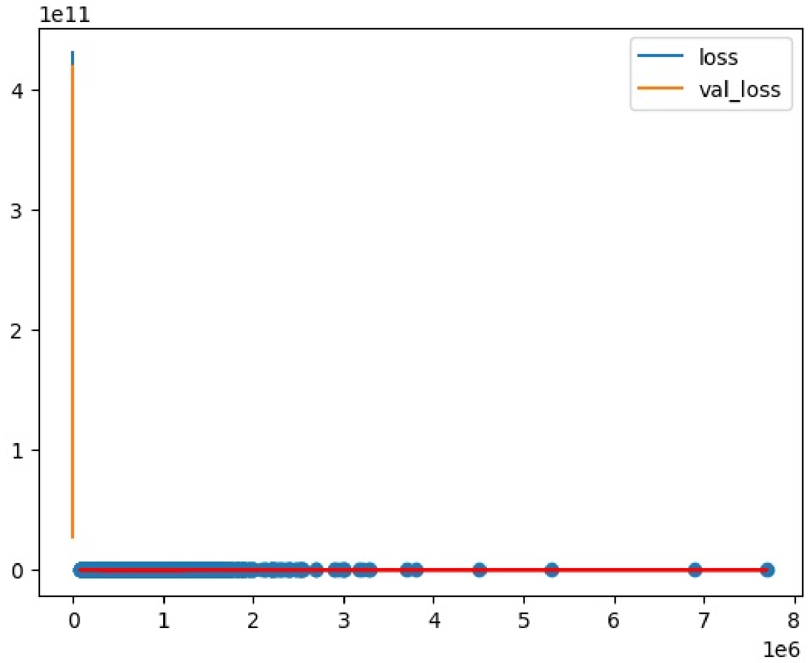
In [ ]:
```
# Creating the Neural Network Model
model = Sequential([
    Dense(19, activation='relu'),
    Dense(19, activation='relu'),
    Dense(19, activation='relu'),
    Dense(19, activation='relu'),
    Dense(1)
])

model.compile(optimizer=Adam(), loss='mse')

# Training the Model
model.fit(x=X_train, y=y_train, validation_data=(
    X_test, y_test), batch_size=128, epochs=400, verbose=0)
```

Out[ ]:
```
<keras.src.callbacks.history.History at 0x19e069d79d0>
```

In [ ]:
```
# Evaluating Model Performance
losses = pd.DataFrame(model.history.history)
losses.plot()

predictions = model.predict(X_test)
print(f"MAE: {mean_absolute_error(y_test, predictions)}")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test, predictions))}")
print(
    f"Explained Variance Score: {explained_variance_score(y_test, predictions)}")

# Plotting predictions vs actual prices
plt.scatter(y_test, predictions)
plt.plot(y_test, y_test, 'r')
```

```
203/203 ━━━━━━━━━━━━━━━━━━━━ 0s 935us/step
MAE: 100499.83408323688
RMSE: 164049.0437463553
Explained Variance Score: 0.7973987069435204
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x19e1164d050>]
```



Experiment No: 4

Student Name and Roll Number: Piyush Gambhir – 21CSU349	
Semester /Section: 6th Semester – AIML-B (A3)	
Link to Code: [ncu-lab-manual-and-end-semester-projects/NCU-CSL312 - DL - Lab Manual/Experiment 4/Experiment 4.ipynb at main · piyush-gambhir/ncu-lab-manual-and-end-semester-projects (github.com)]	
Date:	
Faculty Signature:	
Marks:	

Objective(s):
To build an ANN model for classification problem on diabetes classification to see the effect of:

a. Early Stopping

b. Dropouts

Experiment 4 - ANN Model - Breast Cancer Prediction - Early Stopping & Dropout

Problem Statement:

To build an ANN model for classification problem on breast cancer classification to see the effect of:

- a. Early Stopping
- b. Dropouts

GitHub & Google Colab Links:

GitHub Link: https://github.com/piyush-gambhir/ncu-lab-manual-and-end-semester-projects/blob/main/NCU-CSL312%20-%20DL%20-%20Lab%20Manual/Experiment%204/Experiment%204.ipynb

Google Colab Link:

Open in Colab

Installing Depenedencies

In [ ]:
```
! pip install tabulate numpy pandas matplotlib seaborn
```

```
Requirement already satisfied: tabulate in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packa
ges (0.9.0)
Requirement already satisfied: numpy in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages
(1.26.4)
Requirement already satisfied: pandas in c:\users\mainp\appdata\local\programs\python\python311\lib\site-package
s (2.2.2)
Requirement already satisfied: matplotlib in c:\users\mainp\appdata\local\programs\python\python311\lib\site-pac
kages (3.8.4)
Requirement already satisfied: seaborn in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packag
es (0.13.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\mainp\appdata\local\programs\python\python311\
lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-p
ackages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\mainp\appdata\local\programs\python\python311\lib\site
-packages (from pandas) (2024.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\si
te-packages (from matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-p
ackages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\s
ite-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\s
ite-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\sit
e-packages (from matplotlib) (24.0)
Requirement already satisfied: pillow>=8 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-pack
ages (from matplotlib) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\si
te-packages (from matplotlib) (3.1.2)
Requirement already satisfied: six>=1.5 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packa
ges (from python-dateutil>=2.8.2->pandas) (1.16.0)
```

Code

In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.metrics import classification_report, confusion_matrix
import tensorflow as tf
```

In [ ]:
```
# Loading and Initial Data Check
```

```python
df = pd.read_csv('cancer_classification.csv')
print(df.head().to_markdown())
```

```
|    |   mean radius |   mean texture |   mean perimeter |   mean area |   mean smoothness |   mean compactness
|   mean concavity |   mean concave points |   mean symmetry |   mean fractal dimension |   radius error |   tex
ture error |   perimeter error |   area error |   smoothness error |   compactness error |   concavity error |
concave points error |   symmetry error |   fractal dimension error |   worst radius |   worst texture |   worst
perimeter |   worst area |   worst smoothness |   worst compactness |   worst concavity |   worst concave points
|   worst symmetry |   worst fractal dimension |   benign_0__mal_1 |
|---:|--------------:|---------------:|-----------------:|------------:|------------------:|-------------------:
|-----------------:|----------------------:|----------------:|-------------------------:|---------------:|------
----------:|------------------:|-------------:|-------------------:|--------------------:|------------------:|--
--------------------:|-----------------:|--------------------------:|---------------:|----------------:|-------
----------:|-------------:|-------------------:|--------------------:|------------------:|---------------------
--:|-----------------:|--------------------------:|------------------:|
|  0 |         17.99 |          10.38 |           122.8  |      1001   |           0.1184  |             0.2776
|           0.3001 |               0.1471  |          0.2419 |                  0.07871 |         1.095  |
0.9053 |             8.589 |       153.4  |           0.006399 |            0.04904 |           0.05373 |
0.01587 |           0.03003 |                  0.006193 |          25.38 |           17.33 |            184.6  |
2019   |             0.1622 |              0.6656 |            0.7119 |                 0.2654 |               0.460
1 |                    0.1189  |                 0 |
|  1 |         20.57 |          17.77 |           132.9  |      1326   |           0.08474 |             0.07864
|           0.0869 |               0.07017 |          0.1812 |                  0.05667 |         0.5435 |
0.7339 |             3.398 |        74.08 |           0.005225 |            0.01308 |           0.0186  |
0.0134  |           0.01389 |                  0.003532 |          24.99 |           23.41 |            158.8  |
1956   |             0.1238 |              0.1866 |            0.2416 |                 0.186  |               0.275
|                  0.08902 |                 0 |
|  2 |         19.69 |          21.25 |           130    |      1203   |           0.1096  |             0.1599
|           0.1974 |               0.1279  |          0.2069 |                  0.05999 |         0.7456 |
0.7869 |             4.585 |        94.03 |           0.00615  |            0.04006 |           0.03832 |
0.02058 |           0.0225  |                  0.004571 |          23.57 |           25.53 |            152.5  |
1709   |             0.1444 |              0.4245 |            0.4504 |                 0.243  |               0.361
3 |                    0.08758 |                 0 |
|  3 |         11.42 |          20.38 |            77.58 |       386.1 |           0.1425  |             0.2839
|           0.2414 |               0.1052  |          0.2597 |                  0.09744 |         0.4956 |
1.156  |             3.445 |        27.23 |           0.00911  |            0.07458 |           0.05661 |
0.01867 |           0.05963 |                  0.009208 |          14.91 |           26.5  |             98.87 |
567.7 |             0.2098 |              0.8663 |            0.6869 |                 0.2575 |               0.6638
|                  0.173   |                 0 |
|  4 |         20.29 |          14.34 |           135.1  |      1297   |           0.1003  |             0.1328
|           0.198  |               0.1043  |          0.1809 |                  0.05883 |         0.7572 |
0.7813 |             5.438 |        94.44 |           0.01149  |            0.02461 |           0.05688 |
0.01885 |           0.01756 |                  0.005115 |          22.54 |           16.67 |            152.2  |
1575   |             0.1374 |              0.205  |            0.4    |                 0.1625 |               0.236
4 |                    0.07678 |                 0 |
```

In [ ]:

```python
# understanding the data
print("Data Info")
print(df.info())

print("/n")

print("Data Description")
print(df.describe().transpose().to_markdown())
```

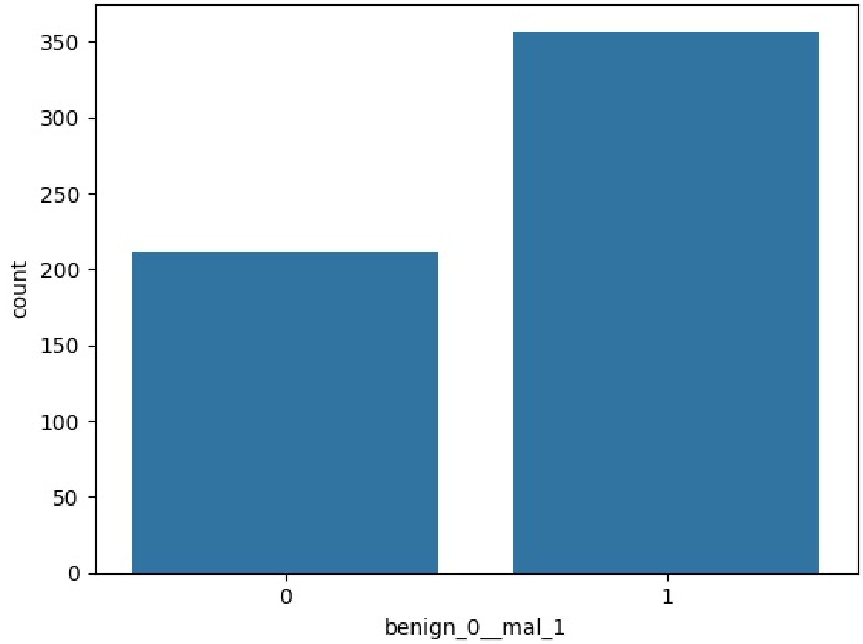
```
Data Info
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
```

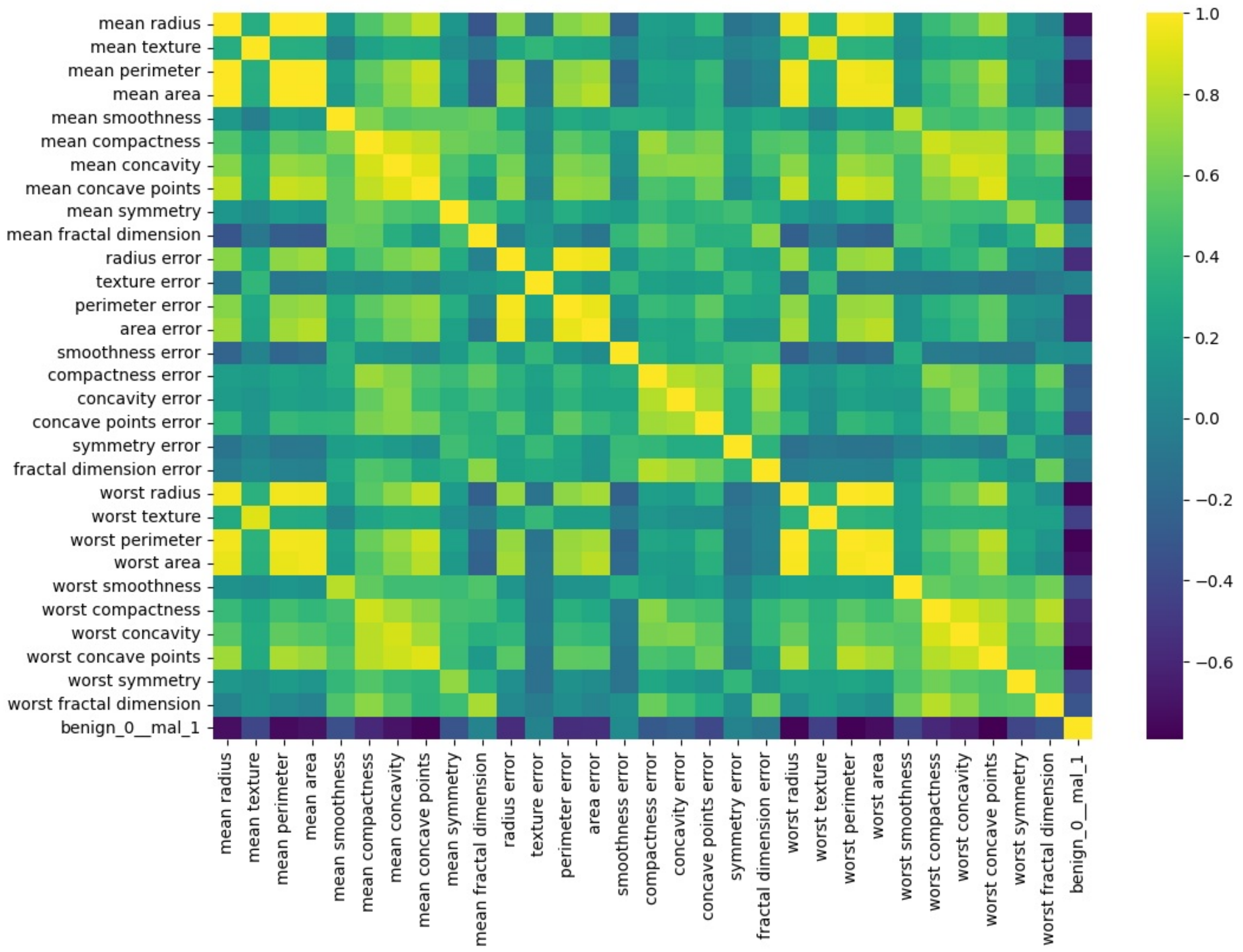
```
 23  worst area              569 non-null    float64
 24  worst smoothness        569 non-null    float64
 25  worst compactness       569 non-null    float64
 26  worst concavity         569 non-null    float64
 27  worst concave points    569 non-null    float64
 28  worst symmetry          569 non-null    float64
 29  worst fractal dimension 569 non-null    float64
 30  benign_0__mal_1         569 non-null    int64
dtypes: float64(30), int64(1)
memory usage: 137.9 KB
None
/n
Data Description
|                         |   count |         mean |          std |          min |         25% |         50% |
75% |        max |
|:------------------------|--------:|-------------:|-------------:|-------------:|------------:|------------:|-----
-------:|-----------:|
| mean radius             |     569 |  14.1273     |   3.52405    |   6.981      |  11.7       |  13.37      |   15
.78      |   28.11    |
| mean texture            |     569 |  19.2896     |   4.30104    |   9.71       |  16.17      |  18.84      |   21
.8       |   39.28    |
| mean perimeter          |     569 |  91.969      |  24.299      |  43.79       |  75.17      |  86.24      |  104
.1       |  188.5     |
| mean area               |     569 | 654.889      | 351.914      | 143.5        | 420.3       | 551.1       |  782
.7       | 2501       |
| mean smoothness         |     569 |   0.0963603  |   0.0140641  |   0.05263    |   0.08637   |   0.09587   |    0
.1053    |    0.1634  |
| mean compactness        |     569 |   0.104341   |   0.0528128  |   0.01938    |   0.06492   |   0.09263   |    0
.1304    |    0.3454  |
| mean concavity          |     569 |   0.0887993  |   0.0797198  |   0          |   0.02956   |   0.06154   |    0
.1307    |    0.4268  |
| mean concave points     |     569 |   0.0489191  |   0.0388028  |   0          |   0.02031   |   0.0335    |    0
.074     |    0.2012  |
| mean symmetry           |     569 |   0.181162   |   0.0274143  |   0.106      |   0.1619    |   0.1792    |    0
.1957    |    0.304   |
| mean fractal dimension  |     569 |   0.0627976  |   0.00706036 |   0.04996    |   0.0577    |   0.06154   |    0
.06612   |    0.09744 |
| radius error            |     569 |   0.405172   |   0.277313   |   0.1115     |   0.2324    |   0.3242    |    0
.4789    |    2.873   |
| texture error           |     569 |   1.21685    |   0.551648   |   0.3602     |   0.8339    |   1.108     |    1
.474     |    4.885   |
| perimeter error         |     569 |   2.86606    |   2.02185    |   0.757      |   1.606     |   2.287     |    3
.357     |   21.98    |
| area error              |     569 |  40.3371     |  45.491      |   6.802      |  17.85      |  24.53      |   45
.19      |  542.2     |
| smoothness error        |     569 |   0.00704098 |   0.00300252 |   0.001713   |   0.005169  |   0.00638   |    0
.008146  |    0.03113 |
| compactness error       |     569 |   0.0254781  |   0.0179082  |   0.002252   |   0.01308   |   0.02045   |    0
.03245   |    0.1354  |
| concavity error         |     569 |   0.0318937  |   0.0301861  |   0          |   0.01509   |   0.02589   |    0
.04205   |    0.396   |
| concave points error    |     569 |   0.0117961  |   0.00617029 |   0          |   0.007638  |   0.01093   |    0
.01471   |    0.05279 |
| symmetry error          |     569 |   0.0205423  |   0.00826637 |   0.007882   |   0.01516   |   0.01873   |    0
.02348   |    0.07895 |
| fractal dimension error |     569 |   0.0037949  |   0.00264607 |   0.0008948  |   0.002248  |   0.003187  |    0
.004558  |    0.02984 |
| worst radius            |     569 |  16.2692     |   4.83324    |   7.93       |  13.01      |  14.97      |   18
.79      |   36.04    |
| worst texture           |     569 |  25.6772     |   6.14626    |  12.02       |  21.08      |  25.41      |   29
.72      |   49.54    |
| worst perimeter         |     569 | 107.261      |  33.6025     |  50.41       |  84.11      |  97.66      |  125
.4       |  251.2     |
| worst area              |     569 | 880.583      | 569.357      | 185.2        | 515.3       | 686.5       | 1084
| 4254       |
| worst smoothness        |     569 |   0.132369   |   0.0228324  |   0.07117    |   0.1166    |   0.1313    |    0
.146     |    0.2226  |
| worst compactness       |     569 |   0.254265   |   0.157336   |   0.02729    |   0.1472    |   0.2119    |    0
.3391    |    1.058   |
| worst concavity         |     569 |   0.272188   |   0.208624   |   0          |   0.1145    |   0.2267    |    0
.3829    |    1.252   |
| worst concave points    |     569 |   0.114606   |   0.0657323  |   0          |   0.06493   |   0.09993   |    0
.1614    |    0.291   |
| worst symmetry          |     569 |   0.290076   |   0.0618675  |   0.1565     |   0.2504    |   0.2822    |    0
.3179    |    0.6638  |
| worst fractal dimension |     569 |   0.0839458  |   0.0180613  |   0.05504    |   0.07146   |   0.08004   |    0
.09208   |    0.2075  |
| benign_0__mal_1         |     569 |   0.627417   |   0.483918   |   0          |   0         |   1         |    1
|    1       |
```

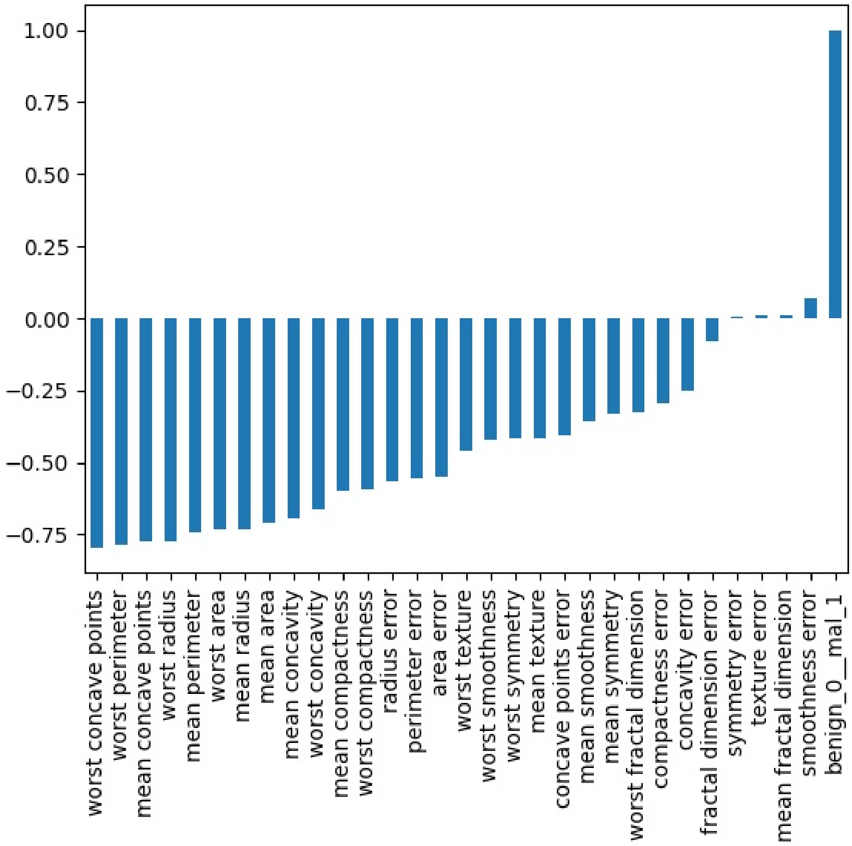
In [ ]:

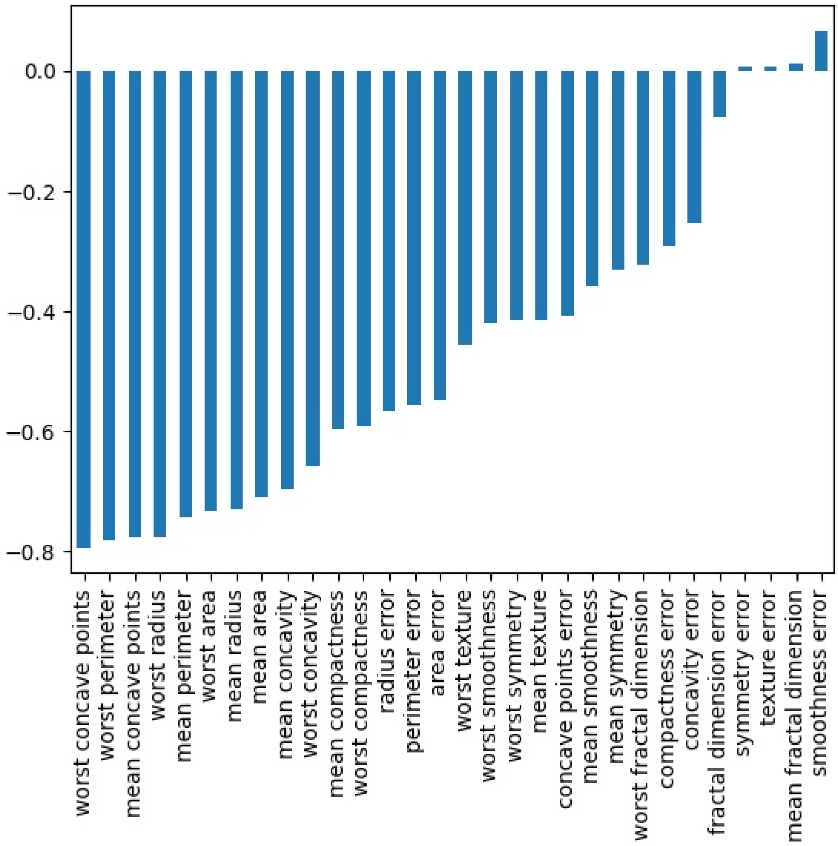
```
# Exploratory Data Analysis (EDA)
# Distribution of Target Variable
sns.countplot(x='benign_0__mal_1', data=df)
```

```
plt.show()

# Correlation Heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(), annot=False, cmap='viridis')
plt.show()

# Correlation with Target Variable
df.corr()['benign_0__mal_1'].sort_values().plot(kind='bar')
plt.show()
df.corr()['benign_0__mal_1'][:-1].sort_values().plot(kind='bar')
plt.show()
```









In [ ]:

```
# Train Test Split
X = df.drop('benign_0__mal_1', axis=1).values
y = df['benign_0__mal_1'].values
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=101)
```

In [ ]:

```
# Scaling Data
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [ ]:
```
# Creating the Model
model = Sequential([
    Dense(units=30, activation='relu'),
    Dense(units=15, activation='relu'),
    Dense(units=1, activation='sigmoid')
])
model.compile(loss='binary_crossentropy',
              optimizer='adam', metrics=['accuracy'])

# Training the Model - Example One: Overfitting
model.fit(x=X_train, y=y_train, epochs=600,
          validation_data=(X_test, y_test), verbose=1)
model_loss = pd.DataFrame(model.history.history)
model_loss.plot()
plt.show()

# Example Two: Early Stopping to prevent overfitting
# Resetting the model
model = Sequential([
    Dense(units=30, activation='relu'),
    Dense(units=15, activation='relu'),
    Dense(units=1, activation='sigmoid')
])
model.compile(loss='binary_crossentropy', optimizer='adam')
```

```
Epoch 1/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.3566 - loss: 0.7173 - val_accuracy: 0.3916 - val_loss: 0.6
914
Epoch 2/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4457 - loss: 0.6814 - val_accuracy: 0.6853 - val_loss: 0.65
86
Epoch 3/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7696 - loss: 0.6496 - val_accuracy: 0.8741 - val_loss: 0.61
98
Epoch 4/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8958 - loss: 0.6044 - val_accuracy: 0.9091 - val_loss: 0.57
14
Epoch 5/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8730 - loss: 0.5571 - val_accuracy: 0.9161 - val_loss: 0.52
08
Epoch 6/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8814 - loss: 0.5056 - val_accuracy: 0.8811 - val_loss: 0.46
89
Epoch 7/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8947 - loss: 0.4614 - val_accuracy: 0.9161 - val_loss: 0.42
29
Epoch 8/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8992 - loss: 0.4102 - val_accuracy: 0.9021 - val_loss: 0.37
29
Epoch 9/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8809 - loss: 0.3784 - val_accuracy: 0.9231 - val_loss: 0.33
41
Epoch 10/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9268 - loss: 0.3122 - val_accuracy: 0.9161 - val_loss: 0.29
66
Epoch 11/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9048 - loss: 0.2924 - val_accuracy: 0.9371 - val_loss: 0.27
00
Epoch 12/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9191 - loss: 0.2617 - val_accuracy: 0.9510 - val_loss: 0.24
35
Epoch 13/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9234 - loss: 0.2497 - val_accuracy: 0.9580 - val_loss: 0.22
19
Epoch 14/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9172 - loss: 0.2294 - val_accuracy: 0.9441 - val_loss: 0.20
64
Epoch 15/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9234 - loss: 0.2203 - val_accuracy: 0.9371 - val_loss: 0.19
45
Epoch 16/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9441 - loss: 0.1796 - val_accuracy: 0.9371 - val_loss: 0.18
42
Epoch 17/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9190 - loss: 0.2000 - val_accuracy: 0.9510 - val_loss: 0.17
43
Epoch 18/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9382 - loss: 0.1801 - val_accuracy: 0.9580 - val_loss: 0.16
54
Epoch 19/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9537 - loss: 0.1421 - val_accuracy: 0.9650 - val_loss: 0.15
84
Epoch 20/600
```

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9386 - loss: 0.1522 - val_accuracy: 0.9650 - val_loss: 0.15
24
Epoch 21/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9402 - loss: 0.1336 - val_accuracy: 0.9580 - val_loss: 0.15
06
Epoch 22/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9559 - loss: 0.1315 - val_accuracy: 0.9650 - val_loss: 0.14
35
Epoch 23/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9603 - loss: 0.1373 - val_accuracy: 0.9720 - val_loss: 0.13
85
Epoch 24/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9376 - loss: 0.1350 - val_accuracy: 0.9650 - val_loss: 0.13
67
Epoch 25/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9499 - loss: 0.1530 - val_accuracy: 0.9580 - val_loss: 0.13
42
Epoch 26/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9610 - loss: 0.1195 - val_accuracy: 0.9510 - val_loss: 0.13
19
Epoch 27/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9708 - loss: 0.1303 - val_accuracy: 0.9510 - val_loss: 0.13
01
Epoch 28/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9767 - loss: 0.0939 - val_accuracy: 0.9580 - val_loss: 0.12
87
Epoch 29/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9822 - loss: 0.1014 - val_accuracy: 0.9650 - val_loss: 0.12
60
Epoch 30/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9567 - loss: 0.1128 - val_accuracy: 0.9650 - val_loss: 0.12
94
Epoch 31/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9801 - loss: 0.1067 - val_accuracy: 0.9650 - val_loss: 0.12
26
Epoch 32/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9868 - loss: 0.0860 - val_accuracy: 0.9650 - val_loss: 0.12
32
Epoch 33/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9846 - loss: 0.0875 - val_accuracy: 0.9650 - val_loss: 0.11
96
Epoch 34/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9854 - loss: 0.1020 - val_accuracy: 0.9650 - val_loss: 0.12
61
Epoch 35/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9762 - loss: 0.0943 - val_accuracy: 0.9720 - val_loss: 0.11
67
Epoch 36/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9832 - loss: 0.0779 - val_accuracy: 0.9650 - val_loss: 0.12
10
Epoch 37/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9845 - loss: 0.0794 - val_accuracy: 0.9650 - val_loss: 0.11
89
Epoch 38/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9830 - loss: 0.0779 - val_accuracy: 0.9650 - val_loss: 0.11
87
Epoch 39/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9749 - loss: 0.0868 - val_accuracy: 0.9650 - val_loss: 0.11
62
Epoch 40/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9805 - loss: 0.0762 - val_accuracy: 0.9650 - val_loss: 0.11
74
Epoch 41/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9847 - loss: 0.0722 - val_accuracy: 0.9650 - val_loss: 0.11
86
Epoch 42/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9743 - loss: 0.0877 - val_accuracy: 0.9860 - val_loss: 0.11
22
Epoch 43/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9775 - loss: 0.0749 - val_accuracy: 0.9650 - val_loss: 0.11
59
Epoch 44/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9831 - loss: 0.0798 - val_accuracy: 0.9860 - val_loss: 0.11
04
Epoch 45/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9716 - loss: 0.0869 - val_accuracy: 0.9790 - val_loss: 0.11
07
Epoch 46/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9880 - loss: 0.0629 - val_accuracy: 0.9650 - val_loss: 0.11
76
Epoch 47/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9780 - loss: 0.0775 - val_accuracy: 0.9650 - val_loss: 0.11
32
```

```
Epoch 48/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9709 - loss: 0.0884 - val_accuracy: 0.9860 - val_loss: 0.10
96
Epoch 49/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9898 - loss: 0.0518 - val_accuracy: 0.9650 - val_loss: 0.11
62
Epoch 50/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9773 - loss: 0.0804 - val_accuracy: 0.9650 - val_loss: 0.11
27
Epoch 51/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9723 - loss: 0.0816 - val_accuracy: 0.9650 - val_loss: 0.11
14
Epoch 52/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9805 - loss: 0.0617 - val_accuracy: 0.9650 - val_loss: 0.11
97
Epoch 53/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9785 - loss: 0.0648 - val_accuracy: 0.9860 - val_loss: 0.10
83
Epoch 54/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9832 - loss: 0.0534 - val_accuracy: 0.9650 - val_loss: 0.11
41
Epoch 55/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9700 - loss: 0.0763 - val_accuracy: 0.9650 - val_loss: 0.11
31
Epoch 56/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9796 - loss: 0.0655 - val_accuracy: 0.9790 - val_loss: 0.10
87
Epoch 57/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9644 - loss: 0.0727 - val_accuracy: 0.9650 - val_loss: 0.11
31
Epoch 58/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9846 - loss: 0.0643 - val_accuracy: 0.9650 - val_loss: 0.10
95
Epoch 59/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9702 - loss: 0.0871 - val_accuracy: 0.9650 - val_loss: 0.11
17
Epoch 60/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9849 - loss: 0.0562 - val_accuracy: 0.9650 - val_loss: 0.11
08
Epoch 61/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9925 - loss: 0.0466 - val_accuracy: 0.9720 - val_loss: 0.11
28
Epoch 62/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9806 - loss: 0.0591 - val_accuracy: 0.9650 - val_loss: 0.11
08
Epoch 63/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9833 - loss: 0.0600 - val_accuracy: 0.9650 - val_loss: 0.11
47
Epoch 64/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9912 - loss: 0.0482 - val_accuracy: 0.9650 - val_loss: 0.11
26
Epoch 65/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9651 - loss: 0.0828 - val_accuracy: 0.9650 - val_loss: 0.11
24
Epoch 66/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9777 - loss: 0.0620 - val_accuracy: 0.9650 - val_loss: 0.10
98
Epoch 67/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9768 - loss: 0.0583 - val_accuracy: 0.9650 - val_loss: 0.11
50
Epoch 68/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9748 - loss: 0.0704 - val_accuracy: 0.9650 - val_loss: 0.11
12
Epoch 69/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9823 - loss: 0.0652 - val_accuracy: 0.9650 - val_loss: 0.11
39
Epoch 70/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9832 - loss: 0.0468 - val_accuracy: 0.9650 - val_loss: 0.10
80
Epoch 71/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9873 - loss: 0.0518 - val_accuracy: 0.9650 - val_loss: 0.11
34
Epoch 72/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9900 - loss: 0.0477 - val_accuracy: 0.9650 - val_loss: 0.10
83
Epoch 73/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9798 - loss: 0.0570 - val_accuracy: 0.9650 - val_loss: 0.10
89
Epoch 74/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9851 - loss: 0.0474 - val_accuracy: 0.9650 - val_loss: 0.11
53
Epoch 75/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9828 - loss: 0.0575 - val_accuracy: 0.9650 - val_loss: 0.11
```

```
01
Epoch 76/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9835 - loss: 0.0431 - val_accuracy: 0.9650 - val_loss: 0.12
46
Epoch 77/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9760 - loss: 0.0605 - val_accuracy: 0.9650 - val_loss: 0.10
82
Epoch 78/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9784 - loss: 0.0620 - val_accuracy: 0.9650 - val_loss: 0.11
40
Epoch 79/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9920 - loss: 0.0434 - val_accuracy: 0.9650 - val_loss: 0.11
02
Epoch 80/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9815 - loss: 0.0545 - val_accuracy: 0.9650 - val_loss: 0.11
21
Epoch 81/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9803 - loss: 0.0580 - val_accuracy: 0.9650 - val_loss: 0.10
83
Epoch 82/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9872 - loss: 0.0435 - val_accuracy: 0.9650 - val_loss: 0.10
96
Epoch 83/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9867 - loss: 0.0567 - val_accuracy: 0.9650 - val_loss: 0.10
92
Epoch 84/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9790 - loss: 0.0600 - val_accuracy: 0.9720 - val_loss: 0.10
75
Epoch 85/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9813 - loss: 0.0439 - val_accuracy: 0.9650 - val_loss: 0.11
04
Epoch 86/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9931 - loss: 0.0397 - val_accuracy: 0.9650 - val_loss: 0.11
24
Epoch 87/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9924 - loss: 0.0352 - val_accuracy: 0.9650 - val_loss: 0.11
00
Epoch 88/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9817 - loss: 0.0588 - val_accuracy: 0.9650 - val_loss: 0.11
02
Epoch 89/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9844 - loss: 0.0495 - val_accuracy: 0.9650 - val_loss: 0.11
04
Epoch 90/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9732 - loss: 0.0742 - val_accuracy: 0.9650 - val_loss: 0.10
97
Epoch 91/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9755 - loss: 0.0593 - val_accuracy: 0.9720 - val_loss: 0.10
87
Epoch 92/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9771 - loss: 0.0491 - val_accuracy: 0.9650 - val_loss: 0.10
95
Epoch 93/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9898 - loss: 0.0432 - val_accuracy: 0.9650 - val_loss: 0.12
08
Epoch 94/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9901 - loss: 0.0425 - val_accuracy: 0.9650 - val_loss: 0.10
84
Epoch 95/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9794 - loss: 0.0653 - val_accuracy: 0.9650 - val_loss: 0.10
88
Epoch 96/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9860 - loss: 0.0438 - val_accuracy: 0.9650 - val_loss: 0.11
40
Epoch 97/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9877 - loss: 0.0494 - val_accuracy: 0.9650 - val_loss: 0.11
07
Epoch 98/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9873 - loss: 0.0432 - val_accuracy: 0.9650 - val_loss: 0.11
93
Epoch 99/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9860 - loss: 0.0545 - val_accuracy: 0.9650 - val_loss: 0.11
10
Epoch 100/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9849 - loss: 0.0484 - val_accuracy: 0.9650 - val_loss: 0.11
02
Epoch 101/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9918 - loss: 0.0390 - val_accuracy: 0.9650 - val_loss: 0.11
28
Epoch 102/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9802 - loss: 0.0582 - val_accuracy: 0.9650 - val_loss: 0.11
02
Epoch 103/600
```

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9741 - loss: 0.0673 - val_accuracy: 0.9650 - val_loss: 0.10
95
Epoch 104/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9848 - loss: 0.0365 - val_accuracy: 0.9650 - val_loss: 0.11
42
Epoch 105/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9888 - loss: 0.0395 - val_accuracy: 0.9650 - val_loss: 0.11
21
Epoch 106/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9829 - loss: 0.0552 - val_accuracy: 0.9650 - val_loss: 0.11
41
Epoch 107/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9926 - loss: 0.0359 - val_accuracy: 0.9650 - val_loss: 0.11
23
Epoch 108/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9897 - loss: 0.0432 - val_accuracy: 0.9650 - val_loss: 0.10
78
Epoch 109/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9842 - loss: 0.0473 - val_accuracy: 0.9650 - val_loss: 0.11
25
Epoch 110/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9872 - loss: 0.0496 - val_accuracy: 0.9650 - val_loss: 0.11
50
Epoch 111/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9801 - loss: 0.0660 - val_accuracy: 0.9930 - val_loss: 0.10
42
Epoch 112/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9775 - loss: 0.0621 - val_accuracy: 0.9650 - val_loss: 0.10
98
Epoch 113/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9772 - loss: 0.0600 - val_accuracy: 0.9650 - val_loss: 0.10
88
Epoch 114/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9823 - loss: 0.0446 - val_accuracy: 0.9650 - val_loss: 0.12
06
Epoch 115/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9801 - loss: 0.0510 - val_accuracy: 0.9930 - val_loss: 0.10
38
Epoch 116/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9836 - loss: 0.0520 - val_accuracy: 0.9650 - val_loss: 0.11
62
Epoch 117/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9808 - loss: 0.0592 - val_accuracy: 0.9720 - val_loss: 0.10
42
Epoch 118/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9845 - loss: 0.0426 - val_accuracy: 0.9650 - val_loss: 0.11
06
Epoch 119/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9858 - loss: 0.0486 - val_accuracy: 0.9720 - val_loss: 0.10
65
Epoch 120/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9845 - loss: 0.0559 - val_accuracy: 0.9720 - val_loss: 0.10
66
Epoch 121/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9809 - loss: 0.0442 - val_accuracy: 0.9650 - val_loss: 0.11
72
Epoch 122/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9911 - loss: 0.0384 - val_accuracy: 0.9650 - val_loss: 0.10
90
Epoch 123/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9850 - loss: 0.0401 - val_accuracy: 0.9650 - val_loss: 0.10
85
Epoch 124/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9789 - loss: 0.0495 - val_accuracy: 0.9650 - val_loss: 0.11
05
Epoch 125/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9833 - loss: 0.0496 - val_accuracy: 0.9650 - val_loss: 0.11
20
Epoch 126/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9819 - loss: 0.0418 - val_accuracy: 0.9720 - val_loss: 0.10
86
Epoch 127/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9797 - loss: 0.0597 - val_accuracy: 0.9720 - val_loss: 0.10
85
Epoch 128/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9850 - loss: 0.0459 - val_accuracy: 0.9650 - val_loss: 0.11
07
Epoch 129/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9887 - loss: 0.0450 - val_accuracy: 0.9650 - val_loss: 0.11
42
Epoch 130/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9827 - loss: 0.0499 - val_accuracy: 0.9650 - val_loss: 0.10
94
```

```
Epoch 131/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9821 - loss: 0.0515 - val_accuracy: 0.9650 - val_loss: 0.12
39
Epoch 132/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9757 - loss: 0.0580 - val_accuracy: 0.9720 - val_loss: 0.10
71
Epoch 133/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9892 - loss: 0.0433 - val_accuracy: 0.9650 - val_loss: 0.11
10
Epoch 134/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9848 - loss: 0.0507 - val_accuracy: 0.9650 - val_loss: 0.10
96
Epoch 135/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9904 - loss: 0.0285 - val_accuracy: 0.9650 - val_loss: 0.11
42
Epoch 136/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9842 - loss: 0.0479 - val_accuracy: 0.9650 - val_loss: 0.11
42
Epoch 137/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9853 - loss: 0.0475 - val_accuracy: 0.9650 - val_loss: 0.11
39
Epoch 138/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9847 - loss: 0.0489 - val_accuracy: 0.9650 - val_loss: 0.11
16
Epoch 139/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9856 - loss: 0.0366 - val_accuracy: 0.9650 - val_loss: 0.12
02
Epoch 140/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9798 - loss: 0.0609 - val_accuracy: 0.9790 - val_loss: 0.10
61
Epoch 141/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9796 - loss: 0.0437 - val_accuracy: 0.9650 - val_loss: 0.12
33
Epoch 142/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9869 - loss: 0.0453 - val_accuracy: 0.9720 - val_loss: 0.10
74
Epoch 143/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9922 - loss: 0.0375 - val_accuracy: 0.9650 - val_loss: 0.11
85
Epoch 144/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9891 - loss: 0.0334 - val_accuracy: 0.9650 - val_loss: 0.11
72
Epoch 145/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9941 - loss: 0.0256 - val_accuracy: 0.9650 - val_loss: 0.11
19
Epoch 146/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9926 - loss: 0.0302 - val_accuracy: 0.9650 - val_loss: 0.11
73
Epoch 147/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9845 - loss: 0.0446 - val_accuracy: 0.9650 - val_loss: 0.11
20
Epoch 148/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9883 - loss: 0.0395 - val_accuracy: 0.9650 - val_loss: 0.11
04
Epoch 149/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9844 - loss: 0.0396 - val_accuracy: 0.9650 - val_loss: 0.11
52
Epoch 150/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9756 - loss: 0.0524 - val_accuracy: 0.9650 - val_loss: 0.11
03
Epoch 151/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9881 - loss: 0.0319 - val_accuracy: 0.9650 - val_loss: 0.10
88
Epoch 152/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9921 - loss: 0.0329 - val_accuracy: 0.9650 - val_loss: 0.12
30
Epoch 153/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9799 - loss: 0.0516 - val_accuracy: 0.9930 - val_loss: 0.10
55
Epoch 154/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9892 - loss: 0.0339 - val_accuracy: 0.9650 - val_loss: 0.11
67
Epoch 155/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9829 - loss: 0.0406 - val_accuracy: 0.9720 - val_loss: 0.10
72
Epoch 156/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9913 - loss: 0.0299 - val_accuracy: 0.9650 - val_loss: 0.11
62
Epoch 157/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9850 - loss: 0.0437 - val_accuracy: 0.9650 - val_loss: 0.11
20
Epoch 158/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9802 - loss: 0.0392 - val_accuracy: 0.9650 - val_loss: 0.11
```

```
23
Epoch 159/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9879 - loss: 0.0361 - val_accuracy: 0.9650 - val_loss: 0.11
63
Epoch 160/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9942 - loss: 0.0315 - val_accuracy: 0.9650 - val_loss: 0.11
19
Epoch 161/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9844 - loss: 0.0429 - val_accuracy: 0.9650 - val_loss: 0.11
67
Epoch 162/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9865 - loss: 0.0411 - val_accuracy: 0.9650 - val_loss: 0.11
64
Epoch 163/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9812 - loss: 0.0476 - val_accuracy: 0.9650 - val_loss: 0.11
48
Epoch 164/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9888 - loss: 0.0345 - val_accuracy: 0.9650 - val_loss: 0.11
73
Epoch 165/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9876 - loss: 0.0358 - val_accuracy: 0.9650 - val_loss: 0.11
36
Epoch 166/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9864 - loss: 0.0398 - val_accuracy: 0.9650 - val_loss: 0.11
76
Epoch 167/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9830 - loss: 0.0459 - val_accuracy: 0.9720 - val_loss: 0.11
33
Epoch 168/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9854 - loss: 0.0355 - val_accuracy: 0.9650 - val_loss: 0.11
93
Epoch 169/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9824 - loss: 0.0451 - val_accuracy: 0.9650 - val_loss: 0.12
30
Epoch 170/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9865 - loss: 0.0361 - val_accuracy: 0.9720 - val_loss: 0.11
10
Epoch 171/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9839 - loss: 0.0413 - val_accuracy: 0.9650 - val_loss: 0.11
99
Epoch 172/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9875 - loss: 0.0437 - val_accuracy: 0.9650 - val_loss: 0.11
60
Epoch 173/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9891 - loss: 0.0419 - val_accuracy: 0.9650 - val_loss: 0.11
54
Epoch 174/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9864 - loss: 0.0454 - val_accuracy: 0.9720 - val_loss: 0.11
43
Epoch 175/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9872 - loss: 0.0406 - val_accuracy: 0.9650 - val_loss: 0.11
84
Epoch 176/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9913 - loss: 0.0341 - val_accuracy: 0.9720 - val_loss: 0.11
51
Epoch 177/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9840 - loss: 0.0378 - val_accuracy: 0.9650 - val_loss: 0.12
52
Epoch 178/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9846 - loss: 0.0429 - val_accuracy: 0.9720 - val_loss: 0.11
47
Epoch 179/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9927 - loss: 0.0295 - val_accuracy: 0.9650 - val_loss: 0.12
44
Epoch 180/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9865 - loss: 0.0349 - val_accuracy: 0.9720 - val_loss: 0.11
81
Epoch 181/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9884 - loss: 0.0326 - val_accuracy: 0.9720 - val_loss: 0.11
64
Epoch 182/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9843 - loss: 0.0437 - val_accuracy: 0.9650 - val_loss: 0.11
77
Epoch 183/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9878 - loss: 0.0413 - val_accuracy: 0.9720 - val_loss: 0.11
35
Epoch 184/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9919 - loss: 0.0297 - val_accuracy: 0.9650 - val_loss: 0.12
02
Epoch 185/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9848 - loss: 0.0403 - val_accuracy: 0.9720 - val_loss: 0.11
65
Epoch 186/600
```

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9842 - loss: 0.0443 - val_accuracy: 0.9650 - val_loss: 0.11
84
Epoch 187/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9875 - loss: 0.0307 - val_accuracy: 0.9720 - val_loss: 0.11
30
Epoch 188/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9879 - loss: 0.0411 - val_accuracy: 0.9650 - val_loss: 0.11
68
Epoch 189/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9805 - loss: 0.0462 - val_accuracy: 0.9650 - val_loss: 0.12
09
Epoch 190/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9957 - loss: 0.0294 - val_accuracy: 0.9720 - val_loss: 0.11
26
Epoch 191/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9911 - loss: 0.0343 - val_accuracy: 0.9650 - val_loss: 0.12
17
Epoch 192/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9769 - loss: 0.0520 - val_accuracy: 0.9720 - val_loss: 0.11
18
Epoch 193/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9838 - loss: 0.0398 - val_accuracy: 0.9720 - val_loss: 0.11
33
Epoch 194/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9804 - loss: 0.0443 - val_accuracy: 0.9650 - val_loss: 0.11
71
Epoch 195/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9895 - loss: 0.0358 - val_accuracy: 0.9650 - val_loss: 0.11
99
Epoch 196/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9852 - loss: 0.0442 - val_accuracy: 0.9720 - val_loss: 0.11
41
Epoch 197/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9898 - loss: 0.0356 - val_accuracy: 0.9650 - val_loss: 0.11
87
Epoch 198/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9917 - loss: 0.0357 - val_accuracy: 0.9720 - val_loss: 0.11
72
Epoch 199/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9879 - loss: 0.0308 - val_accuracy: 0.9720 - val_loss: 0.11
80
Epoch 200/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9935 - loss: 0.0237 - val_accuracy: 0.9650 - val_loss: 0.12
64
Epoch 201/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9877 - loss: 0.0352 - val_accuracy: 0.9720 - val_loss: 0.11
29
Epoch 202/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9881 - loss: 0.0337 - val_accuracy: 0.9650 - val_loss: 0.11
96
Epoch 203/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9896 - loss: 0.0290 - val_accuracy: 0.9790 - val_loss: 0.11
22
Epoch 204/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9862 - loss: 0.0402 - val_accuracy: 0.9720 - val_loss: 0.11
57
Epoch 205/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9868 - loss: 0.0346 - val_accuracy: 0.9720 - val_loss: 0.11
36
Epoch 206/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9805 - loss: 0.0409 - val_accuracy: 0.9720 - val_loss: 0.11
62
Epoch 207/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9867 - loss: 0.0345 - val_accuracy: 0.9650 - val_loss: 0.11
98
Epoch 208/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9826 - loss: 0.0385 - val_accuracy: 0.9650 - val_loss: 0.11
90
Epoch 209/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9851 - loss: 0.0292 - val_accuracy: 0.9720 - val_loss: 0.11
70
Epoch 210/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9776 - loss: 0.0557 - val_accuracy: 0.9650 - val_loss: 0.11
80
Epoch 211/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9880 - loss: 0.0325 - val_accuracy: 0.9720 - val_loss: 0.11
29
Epoch 212/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9928 - loss: 0.0285 - val_accuracy: 0.9580 - val_loss: 0.12
76
Epoch 213/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9892 - loss: 0.0340 - val_accuracy: 0.9650 - val_loss: 0.11
72
```

```
Epoch 214/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9842 - loss: 0.0404 - val_accuracy: 0.9650 - val_loss: 0.11
61
Epoch 215/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9866 - loss: 0.0358 - val_accuracy: 0.9650 - val_loss: 0.12
25
Epoch 216/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9801 - loss: 0.0433 - val_accuracy: 0.9720 - val_loss: 0.11
53
Epoch 217/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9850 - loss: 0.0459 - val_accuracy: 0.9650 - val_loss: 0.11
89
Epoch 218/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9898 - loss: 0.0330 - val_accuracy: 0.9650 - val_loss: 0.11
88
Epoch 219/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9927 - loss: 0.0261 - val_accuracy: 0.9650 - val_loss: 0.11
87
Epoch 220/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9873 - loss: 0.0372 - val_accuracy: 0.9580 - val_loss: 0.12
61
Epoch 221/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9778 - loss: 0.0390 - val_accuracy: 0.9720 - val_loss: 0.11
67
Epoch 222/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9890 - loss: 0.0295 - val_accuracy: 0.9720 - val_loss: 0.11
88
Epoch 223/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9855 - loss: 0.0391 - val_accuracy: 0.9720 - val_loss: 0.11
66
Epoch 224/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9906 - loss: 0.0245 - val_accuracy: 0.9650 - val_loss: 0.12
48
Epoch 225/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9843 - loss: 0.0355 - val_accuracy: 0.9720 - val_loss: 0.11
88
Epoch 226/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9875 - loss: 0.0295 - val_accuracy: 0.9720 - val_loss: 0.11
86
Epoch 227/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9894 - loss: 0.0330 - val_accuracy: 0.9720 - val_loss: 0.11
84
Epoch 228/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9822 - loss: 0.0386 - val_accuracy: 0.9720 - val_loss: 0.11
99
Epoch 229/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9865 - loss: 0.0368 - val_accuracy: 0.9720 - val_loss: 0.11
65
Epoch 230/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9916 - loss: 0.0383 - val_accuracy: 0.9720 - val_loss: 0.11
69
Epoch 231/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9791 - loss: 0.0453 - val_accuracy: 0.9720 - val_loss: 0.11
61
Epoch 232/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9913 - loss: 0.0265 - val_accuracy: 0.9650 - val_loss: 0.12
23
Epoch 233/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9864 - loss: 0.0363 - val_accuracy: 0.9720 - val_loss: 0.12
04
Epoch 234/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9843 - loss: 0.0365 - val_accuracy: 0.9720 - val_loss: 0.11
83
Epoch 235/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9856 - loss: 0.0387 - val_accuracy: 0.9650 - val_loss: 0.11
91
Epoch 236/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9919 - loss: 0.0302 - val_accuracy: 0.9720 - val_loss: 0.11
71
Epoch 237/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9939 - loss: 0.0230 - val_accuracy: 0.9650 - val_loss: 0.12
05
Epoch 238/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9885 - loss: 0.0266 - val_accuracy: 0.9720 - val_loss: 0.12
12
Epoch 239/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9858 - loss: 0.0341 - val_accuracy: 0.9720 - val_loss: 0.12
01
Epoch 240/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9914 - loss: 0.0254 - val_accuracy: 0.9720 - val_loss: 0.11
84
Epoch 241/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9817 - loss: 0.0375 - val_accuracy: 0.9720 - val_loss: 0.11
```

```
98
Epoch 242/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9823 - loss: 0.0422 - val_accuracy: 0.9720 - val_loss: 0.11
71
Epoch 243/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9808 - loss: 0.0467 - val_accuracy: 0.9650 - val_loss: 0.12
27
Epoch 244/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9860 - loss: 0.0320 - val_accuracy: 0.9790 - val_loss: 0.11
49
Epoch 245/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9848 - loss: 0.0376 - val_accuracy: 0.9720 - val_loss: 0.11
76
Epoch 246/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9956 - loss: 0.0179 - val_accuracy: 0.9650 - val_loss: 0.12
23
Epoch 247/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9936 - loss: 0.0269 - val_accuracy: 0.9650 - val_loss: 0.12
39
Epoch 248/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9858 - loss: 0.0268 - val_accuracy: 0.9720 - val_loss: 0.12
22
Epoch 249/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9832 - loss: 0.0393 - val_accuracy: 0.9720 - val_loss: 0.11
79
Epoch 250/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9815 - loss: 0.0365 - val_accuracy: 0.9720 - val_loss: 0.11
96
Epoch 251/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9871 - loss: 0.0323 - val_accuracy: 0.9720 - val_loss: 0.12
04
Epoch 252/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9860 - loss: 0.0296 - val_accuracy: 0.9860 - val_loss: 0.11
20
Epoch 253/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9879 - loss: 0.0319 - val_accuracy: 0.9650 - val_loss: 0.12
29
Epoch 254/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9948 - loss: 0.0254 - val_accuracy: 0.9580 - val_loss: 0.12
84
Epoch 255/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9859 - loss: 0.0322 - val_accuracy: 0.9790 - val_loss: 0.11
53
Epoch 256/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9827 - loss: 0.0366 - val_accuracy: 0.9720 - val_loss: 0.11
90
Epoch 257/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9774 - loss: 0.0423 - val_accuracy: 0.9720 - val_loss: 0.11
87
Epoch 258/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9874 - loss: 0.0333 - val_accuracy: 0.9720 - val_loss: 0.11
75
Epoch 259/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9900 - loss: 0.0311 - val_accuracy: 0.9720 - val_loss: 0.12
08
Epoch 260/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9951 - loss: 0.0198 - val_accuracy: 0.9720 - val_loss: 0.11
91
Epoch 261/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9921 - loss: 0.0237 - val_accuracy: 0.9720 - val_loss: 0.11
59
Epoch 262/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9872 - loss: 0.0285 - val_accuracy: 0.9720 - val_loss: 0.11
61
Epoch 263/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9829 - loss: 0.0300 - val_accuracy: 0.9790 - val_loss: 0.11
30
Epoch 264/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9840 - loss: 0.0281 - val_accuracy: 0.9650 - val_loss: 0.12
22
Epoch 265/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9805 - loss: 0.0351 - val_accuracy: 0.9790 - val_loss: 0.11
37
Epoch 266/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9906 - loss: 0.0249 - val_accuracy: 0.9580 - val_loss: 0.12
47
Epoch 267/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9846 - loss: 0.0312 - val_accuracy: 0.9650 - val_loss: 0.11
93
Epoch 268/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9843 - loss: 0.0321 - val_accuracy: 0.9720 - val_loss: 0.11
72
Epoch 269/600
```

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9832 - loss: 0.0291 - val_accuracy: 0.9720 - val_loss: 0.11
35
Epoch 270/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9887 - loss: 0.0298 - val_accuracy: 0.9580 - val_loss: 0.12
18
Epoch 271/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9836 - loss: 0.0367 - val_accuracy: 0.9650 - val_loss: 0.12
29
Epoch 272/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9835 - loss: 0.0329 - val_accuracy: 0.9720 - val_loss: 0.11
96
Epoch 273/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9786 - loss: 0.0436 - val_accuracy: 0.9650 - val_loss: 0.12
15
Epoch 274/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9856 - loss: 0.0369 - val_accuracy: 0.9720 - val_loss: 0.12
02
Epoch 275/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9909 - loss: 0.0245 - val_accuracy: 0.9580 - val_loss: 0.12
43
Epoch 276/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9858 - loss: 0.0319 - val_accuracy: 0.9650 - val_loss: 0.12
18
Epoch 277/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9917 - loss: 0.0269 - val_accuracy: 0.9580 - val_loss: 0.12
15
Epoch 278/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9926 - loss: 0.0242 - val_accuracy: 0.9510 - val_loss: 0.12
85
Epoch 279/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9907 - loss: 0.0262 - val_accuracy: 0.9790 - val_loss: 0.11
61
Epoch 280/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9840 - loss: 0.0384 - val_accuracy: 0.9720 - val_loss: 0.11
94
Epoch 281/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9827 - loss: 0.0349 - val_accuracy: 0.9580 - val_loss: 0.12
53
Epoch 282/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9873 - loss: 0.0334 - val_accuracy: 0.9580 - val_loss: 0.12
10
Epoch 283/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9817 - loss: 0.0335 - val_accuracy: 0.9720 - val_loss: 0.11
93
Epoch 284/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9841 - loss: 0.0347 - val_accuracy: 0.9790 - val_loss: 0.11
61
Epoch 285/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9894 - loss: 0.0241 - val_accuracy: 0.9650 - val_loss: 0.12
04
Epoch 286/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9902 - loss: 0.0247 - val_accuracy: 0.9790 - val_loss: 0.11
38
Epoch 287/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9886 - loss: 0.0319 - val_accuracy: 0.9650 - val_loss: 0.11
97
Epoch 288/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9860 - loss: 0.0322 - val_accuracy: 0.9650 - val_loss: 0.12
20
Epoch 289/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9924 - loss: 0.0207 - val_accuracy: 0.9510 - val_loss: 0.12
96
Epoch 290/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9876 - loss: 0.0262 - val_accuracy: 0.9790 - val_loss: 0.11
80
Epoch 291/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9903 - loss: 0.0240 - val_accuracy: 0.9580 - val_loss: 0.12
43
Epoch 292/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9837 - loss: 0.0321 - val_accuracy: 0.9790 - val_loss: 0.11
55
Epoch 293/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9879 - loss: 0.0304 - val_accuracy: 0.9441 - val_loss: 0.15
73
Epoch 294/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9871 - loss: 0.0428 - val_accuracy: 0.9790 - val_loss: 0.11
29
Epoch 295/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9834 - loss: 0.0332 - val_accuracy: 0.9720 - val_loss: 0.12
05
Epoch 296/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9864 - loss: 0.0313 - val_accuracy: 0.9790 - val_loss: 0.11
84
```

```
Epoch 297/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9780 - loss: 0.0387 - val_accuracy: 0.9720 - val_loss: 0.11
96
Epoch 298/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9954 - loss: 0.0219 - val_accuracy: 0.9580 - val_loss: 0.12
34
Epoch 299/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9910 - loss: 0.0213 - val_accuracy: 0.9580 - val_loss: 0.12
46
Epoch 300/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9935 - loss: 0.0223 - val_accuracy: 0.9790 - val_loss: 0.11
75
Epoch 301/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9935 - loss: 0.0205 - val_accuracy: 0.9510 - val_loss: 0.12
69
Epoch 302/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9913 - loss: 0.0253 - val_accuracy: 0.9790 - val_loss: 0.11
75
Epoch 303/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9937 - loss: 0.0206 - val_accuracy: 0.9790 - val_loss: 0.11
86
Epoch 304/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9855 - loss: 0.0260 - val_accuracy: 0.9510 - val_loss: 0.12
68
Epoch 305/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9862 - loss: 0.0255 - val_accuracy: 0.9790 - val_loss: 0.11
78
Epoch 306/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9914 - loss: 0.0256 - val_accuracy: 0.9580 - val_loss: 0.12
19
Epoch 307/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9858 - loss: 0.0252 - val_accuracy: 0.9580 - val_loss: 0.12
43
Epoch 308/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9873 - loss: 0.0274 - val_accuracy: 0.9790 - val_loss: 0.11
66
Epoch 309/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9878 - loss: 0.0275 - val_accuracy: 0.9580 - val_loss: 0.12
13
Epoch 310/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9866 - loss: 0.0289 - val_accuracy: 0.9650 - val_loss: 0.11
85
Epoch 311/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9961 - loss: 0.0150 - val_accuracy: 0.9510 - val_loss: 0.12
74
Epoch 312/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9939 - loss: 0.0236 - val_accuracy: 0.9720 - val_loss: 0.12
12
Epoch 313/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9856 - loss: 0.0248 - val_accuracy: 0.9580 - val_loss: 0.12
12
Epoch 314/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9875 - loss: 0.0250 - val_accuracy: 0.9580 - val_loss: 0.12
10
Epoch 315/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9910 - loss: 0.0227 - val_accuracy: 0.9510 - val_loss: 0.12
40
Epoch 316/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9917 - loss: 0.0193 - val_accuracy: 0.9650 - val_loss: 0.12
15
Epoch 317/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9882 - loss: 0.0258 - val_accuracy: 0.9510 - val_loss: 0.12
37
Epoch 318/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9896 - loss: 0.0242 - val_accuracy: 0.9510 - val_loss: 0.12
57
Epoch 319/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9811 - loss: 0.0364 - val_accuracy: 0.9580 - val_loss: 0.12
44
Epoch 320/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9921 - loss: 0.0214 - val_accuracy: 0.9510 - val_loss: 0.12
74
Epoch 321/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9923 - loss: 0.0220 - val_accuracy: 0.9510 - val_loss: 0.13
12
Epoch 322/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9850 - loss: 0.0285 - val_accuracy: 0.9790 - val_loss: 0.12
11
Epoch 323/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9859 - loss: 0.0271 - val_accuracy: 0.9720 - val_loss: 0.12
22
Epoch 324/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9877 - loss: 0.0218 - val_accuracy: 0.9510 - val_loss: 0.12
```

```
84
Epoch 325/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9923 - loss: 0.0203 - val_accuracy: 0.9510 - val_loss: 0.13
23
Epoch 326/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9877 - loss: 0.0238 - val_accuracy: 0.9720 - val_loss: 0.12
35
Epoch 327/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9892 - loss: 0.0220 - val_accuracy: 0.9720 - val_loss: 0.12
34
Epoch 328/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9862 - loss: 0.0241 - val_accuracy: 0.9510 - val_loss: 0.14
08
Epoch 329/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9931 - loss: 0.0250 - val_accuracy: 0.9790 - val_loss: 0.11
58
Epoch 330/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9895 - loss: 0.0347 - val_accuracy: 0.9510 - val_loss: 0.13
89
Epoch 331/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9809 - loss: 0.0356 - val_accuracy: 0.9510 - val_loss: 0.12
96
Epoch 332/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9849 - loss: 0.0270 - val_accuracy: 0.9790 - val_loss: 0.12
43
Epoch 333/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9747 - loss: 0.0342 - val_accuracy: 0.9790 - val_loss: 0.11
99
Epoch 334/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9803 - loss: 0.0318 - val_accuracy: 0.9720 - val_loss: 0.11
97
Epoch 335/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9820 - loss: 0.0287 - val_accuracy: 0.9720 - val_loss: 0.12
79
Epoch 336/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9867 - loss: 0.0251 - val_accuracy: 0.9790 - val_loss: 0.12
13
Epoch 337/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9884 - loss: 0.0266 - val_accuracy: 0.9510 - val_loss: 0.13
36
Epoch 338/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9929 - loss: 0.0217 - val_accuracy: 0.9790 - val_loss: 0.12
14
Epoch 339/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9895 - loss: 0.0276 - val_accuracy: 0.9510 - val_loss: 0.13
59
Epoch 340/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9872 - loss: 0.0249 - val_accuracy: 0.9510 - val_loss: 0.13
55
Epoch 341/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9890 - loss: 0.0220 - val_accuracy: 0.9790 - val_loss: 0.12
17
Epoch 342/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9864 - loss: 0.0208 - val_accuracy: 0.9580 - val_loss: 0.13
16
Epoch 343/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9879 - loss: 0.0214 - val_accuracy: 0.9650 - val_loss: 0.12
87
Epoch 344/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9791 - loss: 0.0307 - val_accuracy: 0.9650 - val_loss: 0.12
66
Epoch 345/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9872 - loss: 0.0244 - val_accuracy: 0.9790 - val_loss: 0.12
15
Epoch 346/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9939 - loss: 0.0177 - val_accuracy: 0.9650 - val_loss: 0.12
74
Epoch 347/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9848 - loss: 0.0265 - val_accuracy: 0.9580 - val_loss: 0.12
98
Epoch 348/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9841 - loss: 0.0253 - val_accuracy: 0.9720 - val_loss: 0.12
73
Epoch 349/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9845 - loss: 0.0292 - val_accuracy: 0.9720 - val_loss: 0.12
95
Epoch 350/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9938 - loss: 0.0150 - val_accuracy: 0.9580 - val_loss: 0.13
02
Epoch 351/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9920 - loss: 0.0171 - val_accuracy: 0.9580 - val_loss: 0.13
05
Epoch 352/600
```

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9953 - loss: 0.0217 - val_accuracy: 0.9790 - val_loss: 0.12
36
Epoch 353/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9872 - loss: 0.0248 - val_accuracy: 0.9580 - val_loss: 0.13
03
Epoch 354/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9841 - loss: 0.0225 - val_accuracy: 0.9790 - val_loss: 0.12
27
Epoch 355/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9927 - loss: 0.0203 - val_accuracy: 0.9510 - val_loss: 0.13
38
Epoch 356/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9876 - loss: 0.0238 - val_accuracy: 0.9720 - val_loss: 0.12
68
Epoch 357/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9818 - loss: 0.0255 - val_accuracy: 0.9790 - val_loss: 0.12
88
Epoch 358/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9908 - loss: 0.0204 - val_accuracy: 0.9580 - val_loss: 0.13
18
Epoch 359/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9962 - loss: 0.0221 - val_accuracy: 0.9650 - val_loss: 0.12
84
Epoch 360/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9909 - loss: 0.0253 - val_accuracy: 0.9580 - val_loss: 0.14
00
Epoch 361/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9915 - loss: 0.0318 - val_accuracy: 0.9580 - val_loss: 0.13
30
Epoch 362/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9917 - loss: 0.0215 - val_accuracy: 0.9790 - val_loss: 0.12
43
Epoch 363/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9851 - loss: 0.0243 - val_accuracy: 0.9580 - val_loss: 0.14
09
Epoch 364/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9960 - loss: 0.0174 - val_accuracy: 0.9580 - val_loss: 0.13
50
Epoch 365/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9969 - loss: 0.0162 - val_accuracy: 0.9720 - val_loss: 0.13
00
Epoch 366/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9899 - loss: 0.0241 - val_accuracy: 0.9510 - val_loss: 0.14
04
Epoch 367/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9955 - loss: 0.0145 - val_accuracy: 0.9720 - val_loss: 0.12
51
Epoch 368/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9888 - loss: 0.0201 - val_accuracy: 0.9580 - val_loss: 0.13
65
Epoch 369/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9950 - loss: 0.0173 - val_accuracy: 0.9580 - val_loss: 0.13
20
Epoch 370/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9974 - loss: 0.0129 - val_accuracy: 0.9580 - val_loss: 0.13
98
Epoch 371/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9944 - loss: 0.0203 - val_accuracy: 0.9650 - val_loss: 0.13
03
Epoch 372/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9919 - loss: 0.0230 - val_accuracy: 0.9720 - val_loss: 0.12
55
Epoch 373/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9894 - loss: 0.0203 - val_accuracy: 0.9790 - val_loss: 0.12
92
Epoch 374/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9886 - loss: 0.0197 - val_accuracy: 0.9510 - val_loss: 0.15
31
Epoch 375/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9923 - loss: 0.0227 - val_accuracy: 0.9860 - val_loss: 0.12
16
Epoch 376/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9899 - loss: 0.0256 - val_accuracy: 0.9580 - val_loss: 0.14
24
Epoch 377/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9885 - loss: 0.0315 - val_accuracy: 0.9790 - val_loss: 0.12
48
Epoch 378/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9891 - loss: 0.0262 - val_accuracy: 0.9580 - val_loss: 0.14
30
Epoch 379/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9877 - loss: 0.0270 - val_accuracy: 0.9790 - val_loss: 0.12
54
```

```
Epoch 380/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9926 - loss: 0.0204 - val_accuracy: 0.9510 - val_loss: 0.14
81
Epoch 381/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9974 - loss: 0.0156 - val_accuracy: 0.9720 - val_loss: 0.12
36
Epoch 382/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9935 - loss: 0.0192 - val_accuracy: 0.9580 - val_loss: 0.14
37
Epoch 383/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9889 - loss: 0.0200 - val_accuracy: 0.9580 - val_loss: 0.13
63
Epoch 384/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9916 - loss: 0.0198 - val_accuracy: 0.9580 - val_loss: 0.13
42
Epoch 385/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9985 - loss: 0.0159 - val_accuracy: 0.9720 - val_loss: 0.12
84
Epoch 386/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9941 - loss: 0.0137 - val_accuracy: 0.9580 - val_loss: 0.13
94
Epoch 387/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9888 - loss: 0.0256 - val_accuracy: 0.9580 - val_loss: 0.13
41
Epoch 388/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9841 - loss: 0.0242 - val_accuracy: 0.9650 - val_loss: 0.13
45
Epoch 389/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9867 - loss: 0.0206 - val_accuracy: 0.9580 - val_loss: 0.13
79
Epoch 390/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9966 - loss: 0.0167 - val_accuracy: 0.9720 - val_loss: 0.13
30
Epoch 391/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9901 - loss: 0.0171 - val_accuracy: 0.9580 - val_loss: 0.13
63
Epoch 392/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9853 - loss: 0.0275 - val_accuracy: 0.9720 - val_loss: 0.13
16
Epoch 393/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9905 - loss: 0.0177 - val_accuracy: 0.9650 - val_loss: 0.13
72
Epoch 394/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9950 - loss: 0.0164 - val_accuracy: 0.9580 - val_loss: 0.14
42
Epoch 395/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9987 - loss: 0.0129 - val_accuracy: 0.9720 - val_loss: 0.13
38
Epoch 396/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9853 - loss: 0.0256 - val_accuracy: 0.9650 - val_loss: 0.13
80
Epoch 397/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9976 - loss: 0.0151 - val_accuracy: 0.9650 - val_loss: 0.13
84
Epoch 398/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9944 - loss: 0.0193 - val_accuracy: 0.9650 - val_loss: 0.13
91
Epoch 399/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9923 - loss: 0.0215 - val_accuracy: 0.9580 - val_loss: 0.14
29
Epoch 400/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9902 - loss: 0.0205 - val_accuracy: 0.9580 - val_loss: 0.15
27
Epoch 401/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9932 - loss: 0.0176 - val_accuracy: 0.9720 - val_loss: 0.13
47
Epoch 402/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9856 - loss: 0.0227 - val_accuracy: 0.9720 - val_loss: 0.13
55
Epoch 403/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9941 - loss: 0.0174 - val_accuracy: 0.9580 - val_loss: 0.15
06
Epoch 404/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9914 - loss: 0.0224 - val_accuracy: 0.9720 - val_loss: 0.14
07
Epoch 405/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9951 - loss: 0.0250 - val_accuracy: 0.9720 - val_loss: 0.13
57
Epoch 406/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9924 - loss: 0.0235 - val_accuracy: 0.9650 - val_loss: 0.14
50
Epoch 407/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9905 - loss: 0.0191 - val_accuracy: 0.9650 - val_loss: 0.14
```

```
71
Epoch 408/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9961 - loss: 0.0160 - val_accuracy: 0.9650 - val_loss: 0.14
48
Epoch 409/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9957 - loss: 0.0197 - val_accuracy: 0.9720 - val_loss: 0.13
82
Epoch 410/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9912 - loss: 0.0184 - val_accuracy: 0.9580 - val_loss: 0.15
09
Epoch 411/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9959 - loss: 0.0182 - val_accuracy: 0.9650 - val_loss: 0.14
48
Epoch 412/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9978 - loss: 0.0151 - val_accuracy: 0.9720 - val_loss: 0.13
68
Epoch 413/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9966 - loss: 0.0184 - val_accuracy: 0.9580 - val_loss: 0.14
73
Epoch 414/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9982 - loss: 0.0187 - val_accuracy: 0.9790 - val_loss: 0.13
92
Epoch 415/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9975 - loss: 0.0134 - val_accuracy: 0.9580 - val_loss: 0.15
23
Epoch 416/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9928 - loss: 0.0230 - val_accuracy: 0.9790 - val_loss: 0.13
42
Epoch 417/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9933 - loss: 0.0179 - val_accuracy: 0.9580 - val_loss: 0.14
95
Epoch 418/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9925 - loss: 0.0259 - val_accuracy: 0.9720 - val_loss: 0.13
30
Epoch 419/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9950 - loss: 0.0149 - val_accuracy: 0.9580 - val_loss: 0.15
50
Epoch 420/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9903 - loss: 0.0221 - val_accuracy: 0.9790 - val_loss: 0.13
61
Epoch 421/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9865 - loss: 0.0211 - val_accuracy: 0.9650 - val_loss: 0.14
68
Epoch 422/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9979 - loss: 0.0136 - val_accuracy: 0.9650 - val_loss: 0.14
28
Epoch 423/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9936 - loss: 0.0205 - val_accuracy: 0.9580 - val_loss: 0.15
26
Epoch 424/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9962 - loss: 0.0205 - val_accuracy: 0.9650 - val_loss: 0.14
27
Epoch 425/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9968 - loss: 0.0155 - val_accuracy: 0.9580 - val_loss: 0.15
09
Epoch 426/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9978 - loss: 0.0165 - val_accuracy: 0.9650 - val_loss: 0.14
30
Epoch 427/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9982 - loss: 0.0220 - val_accuracy: 0.9650 - val_loss: 0.14
79
Epoch 428/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9960 - loss: 0.0113 - val_accuracy: 0.9650 - val_loss: 0.14
62
Epoch 429/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9969 - loss: 0.0187 - val_accuracy: 0.9650 - val_loss: 0.15
11
Epoch 430/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0144 - val_accuracy: 0.9650 - val_loss: 0.14
32
Epoch 431/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9990 - loss: 0.0100 - val_accuracy: 0.9650 - val_loss: 0.14
76
Epoch 432/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9990 - loss: 0.0158 - val_accuracy: 0.9580 - val_loss: 0.16
34
Epoch 433/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9970 - loss: 0.0167 - val_accuracy: 0.9650 - val_loss: 0.14
35
Epoch 434/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9995 - loss: 0.0081 - val_accuracy: 0.9650 - val_loss: 0.14
92
Epoch 435/600
```

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9985 - loss: 0.0104 - val_accuracy: 0.9650 - val_loss: 0.14
84
Epoch 436/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9965 - loss: 0.0166 - val_accuracy: 0.9650 - val_loss: 0.15
04
Epoch 437/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9979 - loss: 0.0108 - val_accuracy: 0.9650 - val_loss: 0.14
86
Epoch 438/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9968 - loss: 0.0215 - val_accuracy: 0.9790 - val_loss: 0.13
67
Epoch 439/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9832 - loss: 0.0287 - val_accuracy: 0.9301 - val_loss: 0.21
38
Epoch 440/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9905 - loss: 0.0281 - val_accuracy: 0.9790 - val_loss: 0.14
41
Epoch 441/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9934 - loss: 0.0138 - val_accuracy: 0.9580 - val_loss: 0.16
76
Epoch 442/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9962 - loss: 0.0170 - val_accuracy: 0.9650 - val_loss: 0.15
80
Epoch 443/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.9997 - loss: 0.0134 - val_accuracy: 0.9650 - val_loss: 0.15
09
Epoch 444/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9952 - loss: 0.0179 - val_accuracy: 0.9650 - val_loss: 0.14
77
Epoch 445/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9980 - loss: 0.0087 - val_accuracy: 0.9580 - val_loss: 0.15
24
Epoch 446/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9962 - loss: 0.0168 - val_accuracy: 0.9650 - val_loss: 0.15
49
Epoch 447/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9945 - loss: 0.0148 - val_accuracy: 0.9650 - val_loss: 0.15
11
Epoch 448/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9916 - loss: 0.0223 - val_accuracy: 0.9441 - val_loss: 0.18
32
Epoch 449/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9948 - loss: 0.0148 - val_accuracy: 0.9650 - val_loss: 0.14
85
Epoch 450/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9965 - loss: 0.0178 - val_accuracy: 0.9650 - val_loss: 0.15
12
Epoch 451/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9978 - loss: 0.0115 - val_accuracy: 0.9650 - val_loss: 0.15
13
Epoch 452/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9931 - loss: 0.0176 - val_accuracy: 0.9650 - val_loss: 0.15
36
Epoch 453/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9994 - loss: 0.0130 - val_accuracy: 0.9650 - val_loss: 0.15
63
Epoch 454/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9969 - loss: 0.0148 - val_accuracy: 0.9650 - val_loss: 0.15
58
Epoch 455/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9982 - loss: 0.0147 - val_accuracy: 0.9580 - val_loss: 0.15
78
Epoch 456/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9978 - loss: 0.0135 - val_accuracy: 0.9580 - val_loss: 0.16
32
Epoch 457/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9969 - loss: 0.0161 - val_accuracy: 0.9720 - val_loss: 0.14
90
Epoch 458/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9968 - loss: 0.0112 - val_accuracy: 0.9650 - val_loss: 0.15
19
Epoch 459/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9956 - loss: 0.0117 - val_accuracy: 0.9580 - val_loss: 0.16
39
Epoch 460/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 1.0000 - loss: 0.0131 - val_accuracy: 0.9650 - val_loss: 0.15
87
Epoch 461/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9995 - loss: 0.0122 - val_accuracy: 0.9650 - val_loss: 0.14
89
Epoch 462/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9990 - loss: 0.0149 - val_accuracy: 0.9580 - val_loss: 0.15
91
```

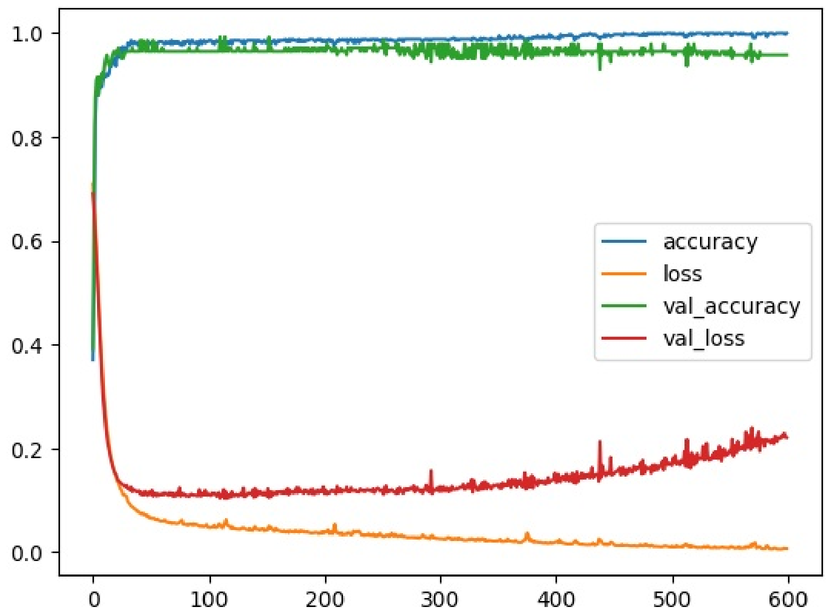
```
Epoch 463/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9974 - loss: 0.0122 - val_accuracy: 0.9650 - val_loss: 0.15
27
Epoch 464/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9974 - loss: 0.0116 - val_accuracy: 0.9650 - val_loss: 0.16
19
Epoch 465/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9995 - loss: 0.0099 - val_accuracy: 0.9650 - val_loss: 0.15
92
Epoch 466/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9995 - loss: 0.0094 - val_accuracy: 0.9650 - val_loss: 0.15
50
Epoch 467/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0134 - val_accuracy: 0.9650 - val_loss: 0.15
42
Epoch 468/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9992 - loss: 0.0120 - val_accuracy: 0.9580 - val_loss: 0.16
63
Epoch 469/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9995 - loss: 0.0143 - val_accuracy: 0.9650 - val_loss: 0.16
43
Epoch 470/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9969 - loss: 0.0136 - val_accuracy: 0.9650 - val_loss: 0.15
81
Epoch 471/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9931 - loss: 0.0182 - val_accuracy: 0.9650 - val_loss: 0.16
31
Epoch 472/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9990 - loss: 0.0087 - val_accuracy: 0.9650 - val_loss: 0.16
31
Epoch 473/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9966 - loss: 0.0115 - val_accuracy: 0.9650 - val_loss: 0.16
26
Epoch 474/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9915 - loss: 0.0178 - val_accuracy: 0.9580 - val_loss: 0.17
26
Epoch 475/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9990 - loss: 0.0139 - val_accuracy: 0.9720 - val_loss: 0.15
10
Epoch 476/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9985 - loss: 0.0078 - val_accuracy: 0.9580 - val_loss: 0.17
80
Epoch 477/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0131 - val_accuracy: 0.9720 - val_loss: 0.15
18
Epoch 478/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9925 - loss: 0.0177 - val_accuracy: 0.9650 - val_loss: 0.16
61
Epoch 479/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9972 - loss: 0.0137 - val_accuracy: 0.9650 - val_loss: 0.15
94
Epoch 480/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0129 - val_accuracy: 0.9650 - val_loss: 0.16
12
Epoch 481/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9969 - loss: 0.0149 - val_accuracy: 0.9650 - val_loss: 0.16
27
Epoch 482/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0138 - val_accuracy: 0.9650 - val_loss: 0.16
28
Epoch 483/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9982 - loss: 0.0105 - val_accuracy: 0.9790 - val_loss: 0.15
24
Epoch 484/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0150 - val_accuracy: 0.9580 - val_loss: 0.16
79
Epoch 485/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9951 - loss: 0.0168 - val_accuracy: 0.9650 - val_loss: 0.15
71
Epoch 486/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0095 - val_accuracy: 0.9650 - val_loss: 0.16
84
Epoch 487/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0125 - val_accuracy: 0.9650 - val_loss: 0.15
85
Epoch 488/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9962 - loss: 0.0150 - val_accuracy: 0.9650 - val_loss: 0.16
76
Epoch 489/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0116 - val_accuracy: 0.9650 - val_loss: 0.15
91
Epoch 490/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9931 - loss: 0.0145 - val_accuracy: 0.9650 - val_loss: 0.16
```

```
34
Epoch 491/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0093 - val_accuracy: 0.9650 - val_loss: 0.16
77
Epoch 492/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0127 - val_accuracy: 0.9650 - val_loss: 0.16
56
Epoch 493/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0119 - val_accuracy: 0.9650 - val_loss: 0.17
60
Epoch 494/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9982 - loss: 0.0117 - val_accuracy: 0.9790 - val_loss: 0.15
84
Epoch 495/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9962 - loss: 0.0186 - val_accuracy: 0.9650 - val_loss: 0.18
20
Epoch 496/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0119 - val_accuracy: 0.9650 - val_loss: 0.18
08
Epoch 497/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0103 - val_accuracy: 0.9650 - val_loss: 0.16
83
Epoch 498/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0118 - val_accuracy: 0.9650 - val_loss: 0.17
11
Epoch 499/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0139 - val_accuracy: 0.9650 - val_loss: 0.17
20
Epoch 500/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0132 - val_accuracy: 0.9650 - val_loss: 0.16
91
Epoch 501/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9994 - loss: 0.0120 - val_accuracy: 0.9650 - val_loss: 0.16
74
Epoch 502/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0077 - val_accuracy: 0.9580 - val_loss: 0.18
23
Epoch 503/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0120 - val_accuracy: 0.9650 - val_loss: 0.17
24
Epoch 504/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9987 - loss: 0.0068 - val_accuracy: 0.9580 - val_loss: 0.18
69
Epoch 505/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9969 - loss: 0.0132 - val_accuracy: 0.9650 - val_loss: 0.16
96
Epoch 506/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9969 - loss: 0.0077 - val_accuracy: 0.9650 - val_loss: 0.17
41
Epoch 507/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9994 - loss: 0.0081 - val_accuracy: 0.9650 - val_loss: 0.17
41
Epoch 508/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0114 - val_accuracy: 0.9650 - val_loss: 0.16
85
Epoch 509/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9990 - loss: 0.0088 - val_accuracy: 0.9650 - val_loss: 0.17
74
Epoch 510/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9969 - loss: 0.0129 - val_accuracy: 0.9650 - val_loss: 0.17
26
Epoch 511/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0101 - val_accuracy: 0.9650 - val_loss: 0.17
38
Epoch 512/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9950 - loss: 0.0114 - val_accuracy: 0.9580 - val_loss: 0.19
43
Epoch 513/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9961 - loss: 0.0140 - val_accuracy: 0.9720 - val_loss: 0.16
55
Epoch 514/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9919 - loss: 0.0162 - val_accuracy: 0.9371 - val_loss: 0.21
79
Epoch 515/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9951 - loss: 0.0143 - val_accuracy: 0.9790 - val_loss: 0.16
42
Epoch 516/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9974 - loss: 0.0118 - val_accuracy: 0.9580 - val_loss: 0.18
65
Epoch 517/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0101 - val_accuracy: 0.9720 - val_loss: 0.16
20
Epoch 518/600
```

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9963 - loss: 0.0090 - val_accuracy: 0.9580 - val_loss: 0.19
17
Epoch 519/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0139 - val_accuracy: 0.9650 - val_loss: 0.17
42
Epoch 520/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9987 - loss: 0.0108 - val_accuracy: 0.9790 - val_loss: 0.16
82
Epoch 521/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9978 - loss: 0.0126 - val_accuracy: 0.9650 - val_loss: 0.18
61
Epoch 522/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0103 - val_accuracy: 0.9650 - val_loss: 0.17
15
Epoch 523/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 1.0000 - loss: 0.0074 - val_accuracy: 0.9580 - val_loss: 0.19
03
Epoch 524/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0105 - val_accuracy: 0.9650 - val_loss: 0.17
95
Epoch 525/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0092 - val_accuracy: 0.9650 - val_loss: 0.18
11
Epoch 526/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0052 - val_accuracy: 0.9650 - val_loss: 0.18
07
Epoch 527/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0123 - val_accuracy: 0.9580 - val_loss: 0.18
08
Epoch 528/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9990 - loss: 0.0061 - val_accuracy: 0.9580 - val_loss: 0.20
44
Epoch 529/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9962 - loss: 0.0137 - val_accuracy: 0.9650 - val_loss: 0.17
79
Epoch 530/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 1.0000 - loss: 0.0098 - val_accuracy: 0.9650 - val_loss: 0.18
09
Epoch 531/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 1.0000 - loss: 0.0079 - val_accuracy: 0.9510 - val_loss: 0.20
99
Epoch 532/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9949 - loss: 0.0141 - val_accuracy: 0.9650 - val_loss: 0.18
25
Epoch 533/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0095 - val_accuracy: 0.9580 - val_loss: 0.17
76
Epoch 534/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 0.0080 - val_accuracy: 0.9650 - val_loss: 0.18
04
Epoch 535/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0109 - val_accuracy: 0.9650 - val_loss: 0.17
94
Epoch 536/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 0.0079 - val_accuracy: 0.9650 - val_loss: 0.18
83
Epoch 537/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0093 - val_accuracy: 0.9580 - val_loss: 0.18
13
Epoch 538/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9978 - loss: 0.0108 - val_accuracy: 0.9650 - val_loss: 0.19
37
Epoch 539/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0073 - val_accuracy: 0.9580 - val_loss: 0.18
19
Epoch 540/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9931 - loss: 0.0121 - val_accuracy: 0.9650 - val_loss: 0.19
34
Epoch 541/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0088 - val_accuracy: 0.9580 - val_loss: 0.18
07
Epoch 542/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 0.0083 - val_accuracy: 0.9580 - val_loss: 0.20
44
Epoch 543/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0058 - val_accuracy: 0.9580 - val_loss: 0.19
10
Epoch 544/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9990 - loss: 0.0055 - val_accuracy: 0.9650 - val_loss: 0.17
60
Epoch 545/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9978 - loss: 0.0096 - val_accuracy: 0.9650 - val_loss: 0.19
28
```

```
Epoch 546/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0094 - val_accuracy: 0.9650 - val_loss: 0.19
89
Epoch 547/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 1.0000 - loss: 0.0111 - val_accuracy: 0.9580 - val_loss: 0.18
59
Epoch 548/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 0.0069 - val_accuracy: 0.9650 - val_loss: 0.18
98
Epoch 549/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 1.0000 - loss: 0.0074 - val_accuracy: 0.9580 - val_loss: 0.18
82
Epoch 550/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 1.0000 - loss: 0.0094 - val_accuracy: 0.9650 - val_loss: 0.1
952
Epoch 551/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0111 - val_accuracy: 0.9580 - val_loss: 0.19
47
Epoch 552/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9983 - loss: 0.0077 - val_accuracy: 0.9720 - val_loss: 0.19
13
Epoch 553/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9952 - loss: 0.0097 - val_accuracy: 0.9510 - val_loss: 0.21
83
Epoch 554/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0141 - val_accuracy: 0.9650 - val_loss: 0.18
53
Epoch 555/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0073 - val_accuracy: 0.9580 - val_loss: 0.20
60
Epoch 556/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0068 - val_accuracy: 0.9580 - val_loss: 0.19
51
Epoch 557/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0078 - val_accuracy: 0.9580 - val_loss: 0.19
21
Epoch 558/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9987 - loss: 0.0077 - val_accuracy: 0.9580 - val_loss: 0.21
40
Epoch 559/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9982 - loss: 0.0094 - val_accuracy: 0.9580 - val_loss: 0.19
71
Epoch 560/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0074 - val_accuracy: 0.9580 - val_loss: 0.19
87
Epoch 561/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0066 - val_accuracy: 0.9580 - val_loss: 0.20
02
Epoch 562/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9992 - loss: 0.0074 - val_accuracy: 0.9580 - val_loss: 0.20
49
Epoch 563/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 0.0070 - val_accuracy: 0.9580 - val_loss: 0.19
87
Epoch 564/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9994 - loss: 0.0053 - val_accuracy: 0.9580 - val_loss: 0.19
68
Epoch 565/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9990 - loss: 0.0118 - val_accuracy: 0.9510 - val_loss: 0.23
14
Epoch 566/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9951 - loss: 0.0126 - val_accuracy: 0.9650 - val_loss: 0.20
13
Epoch 567/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9951 - loss: 0.0097 - val_accuracy: 0.9580 - val_loss: 0.22
58
Epoch 568/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9994 - loss: 0.0109 - val_accuracy: 0.9580 - val_loss: 0.21
72
Epoch 569/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9974 - loss: 0.0117 - val_accuracy: 0.9790 - val_loss: 0.19
77
Epoch 570/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0085 - val_accuracy: 0.9441 - val_loss: 0.23
98
Epoch 571/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9921 - loss: 0.0145 - val_accuracy: 0.9580 - val_loss: 0.20
23
Epoch 572/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9937 - loss: 0.0143 - val_accuracy: 0.9650 - val_loss: 0.21
51
Epoch 573/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9873 - loss: 0.0215 - val_accuracy: 0.9580 - val_loss: 0.21
```

```
91
Epoch 574/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9978 - loss: 0.0104 - val_accuracy: 0.9580 - val_loss: 0.20
38
Epoch 575/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0093 - val_accuracy: 0.9510 - val_loss: 0.22
60
Epoch 576/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0108 - val_accuracy: 0.9510 - val_loss: 0.23
20
Epoch 577/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9985 - loss: 0.0088 - val_accuracy: 0.9650 - val_loss: 0.19
50
Epoch 578/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9969 - loss: 0.0083 - val_accuracy: 0.9580 - val_loss: 0.21
87
Epoch 579/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0063 - val_accuracy: 0.9580 - val_loss: 0.21
56
Epoch 580/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0059 - val_accuracy: 0.9580 - val_loss: 0.21
67
Epoch 581/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0088 - val_accuracy: 0.9580 - val_loss: 0.20
41
Epoch 582/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0066 - val_accuracy: 0.9580 - val_loss: 0.21
78
Epoch 583/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9975 - loss: 0.0073 - val_accuracy: 0.9580 - val_loss: 0.21
78
Epoch 584/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9974 - loss: 0.0126 - val_accuracy: 0.9580 - val_loss: 0.21
46
Epoch 585/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0054 - val_accuracy: 0.9580 - val_loss: 0.21
53
Epoch 586/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0073 - val_accuracy: 0.9580 - val_loss: 0.21
02
Epoch 587/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 0.0081 - val_accuracy: 0.9580 - val_loss: 0.21
75
Epoch 588/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 1.0000 - loss: 0.0049 - val_accuracy: 0.9580 - val_loss: 0.2
152
Epoch 589/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9987 - loss: 0.0062 - val_accuracy: 0.9580 - val_loss: 0.20
91
Epoch 590/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9978 - loss: 0.0061 - val_accuracy: 0.9580 - val_loss: 0.22
39
Epoch 591/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 1.0000 - loss: 0.0080 - val_accuracy: 0.9580 - val_loss: 0.22
20
Epoch 592/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 0.0067 - val_accuracy: 0.9580 - val_loss: 0.21
33
Epoch 593/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0105 - val_accuracy: 0.9580 - val_loss: 0.22
16
Epoch 594/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0078 - val_accuracy: 0.9580 - val_loss: 0.22
04
Epoch 595/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0054 - val_accuracy: 0.9580 - val_loss: 0.21
98
Epoch 596/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0052 - val_accuracy: 0.9580 - val_loss: 0.22
52
Epoch 597/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 0.0064 - val_accuracy: 0.9580 - val_loss: 0.22
19
Epoch 598/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0060 - val_accuracy: 0.9580 - val_loss: 0.22
94
Epoch 599/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9978 - loss: 0.0065 - val_accuracy: 0.9580 - val_loss: 0.22
38
Epoch 600/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0047 - val_accuracy: 0.9580 - val_loss: 0.22
11
```

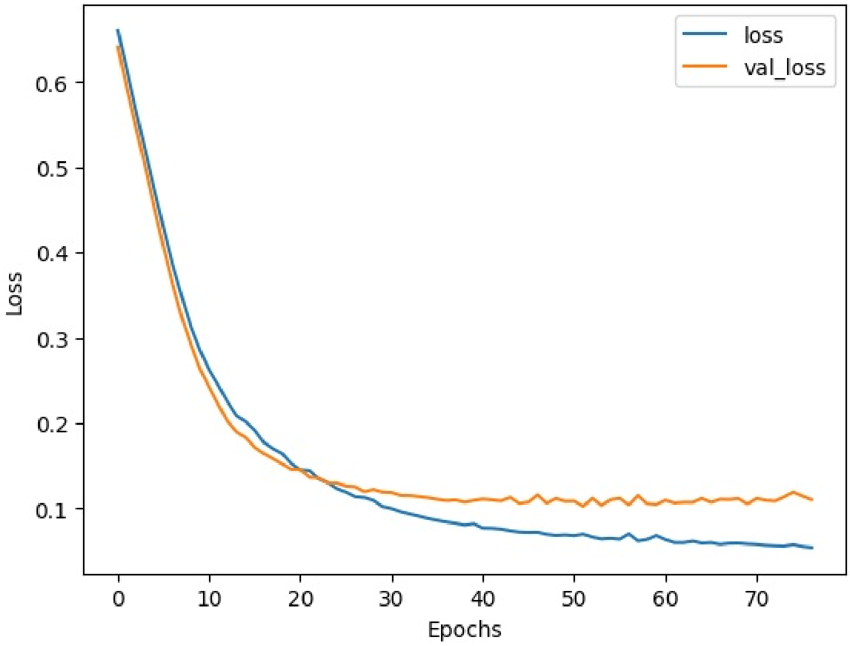


In [ ]:
```
# Early Stopping
early_stop = EarlyStopping(
    monitor='val_loss', mode='min', verbose=1, patience=25)
model.fit(x=X_train, y=y_train, epochs=600, validation_data=(
    X_test, y_test), callbacks=[early_stop], verbose=1)
model_loss = pd.DataFrame(model.history.history)
model_loss.plot()
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.show()
```

```
Epoch 1/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 3s 51ms/step - loss: 0.6708 - val_loss: 0.6407
Epoch 2/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.6177 - val_loss: 0.5912
Epoch 3/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.5778 - val_loss: 0.5454
Epoch 4/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.5323 - val_loss: 0.4998
Epoch 5/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.4802 - val_loss: 0.4518
Epoch 6/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.4237 - val_loss: 0.4060
Epoch 7/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.4111 - val_loss: 0.3628
Epoch 8/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.3476 - val_loss: 0.3239
Epoch 9/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.3148 - val_loss: 0.2922
Epoch 10/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.3020 - val_loss: 0.2632
Epoch 11/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.2732 - val_loss: 0.2420
Epoch 12/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.2440 - val_loss: 0.2217
Epoch 13/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 0.2332 - val_loss: 0.2021
Epoch 14/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.2336 - val_loss: 0.1898
Epoch 15/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.1998 - val_loss: 0.1830
Epoch 16/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.1991 - val_loss: 0.1715
Epoch 17/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.1679 - val_loss: 0.1641
Epoch 18/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.1530 - val_loss: 0.1583
Epoch 19/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.1700 - val_loss: 0.1521
Epoch 20/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.1757 - val_loss: 0.1457
Epoch 21/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.1299 - val_loss: 0.1456
Epoch 22/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.1551 - val_loss: 0.1365
Epoch 23/600
```

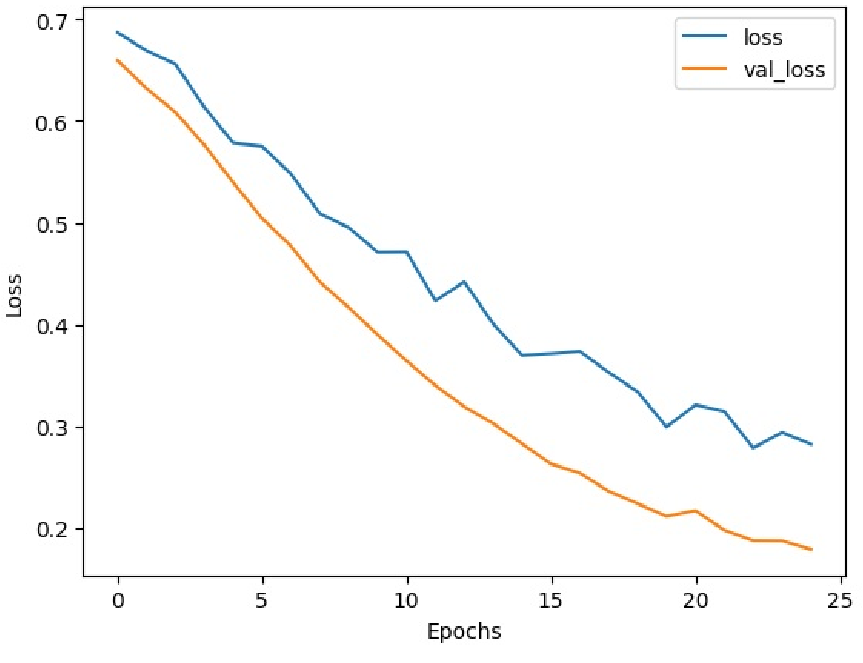
```
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 0.1493 - val_loss: 0.1358
Epoch 24/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.1191 - val_loss: 0.1298
Epoch 25/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.1128 - val_loss: 0.1297
Epoch 26/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.1237 - val_loss: 0.1261
Epoch 27/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.1109 - val_loss: 0.1252
Epoch 28/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.1173 - val_loss: 0.1197
Epoch 29/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.1120 - val_loss: 0.1220
Epoch 30/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.1137 - val_loss: 0.1191
Epoch 31/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.1051 - val_loss: 0.1186
Epoch 32/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0882 - val_loss: 0.1153
Epoch 33/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.1004 - val_loss: 0.1150
Epoch 34/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.1026 - val_loss: 0.1138
Epoch 35/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0777 - val_loss: 0.1125
Epoch 36/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0844 - val_loss: 0.1107
Epoch 37/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0835 - val_loss: 0.1092
Epoch 38/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0789 - val_loss: 0.1099
Epoch 39/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0876 - val_loss: 0.1074
Epoch 40/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0745 - val_loss: 0.1094
Epoch 41/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0697 - val_loss: 0.1110
Epoch 42/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0626 - val_loss: 0.1100
Epoch 43/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0611 - val_loss: 0.1088
Epoch 44/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0857 - val_loss: 0.1130
Epoch 45/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0696 - val_loss: 0.1057
Epoch 46/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0719 - val_loss: 0.1075
Epoch 47/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - loss: 0.0603 - val_loss: 0.1159
Epoch 48/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0582 - val_loss: 0.1062
Epoch 49/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0749 - val_loss: 0.1118
Epoch 50/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0638 - val_loss: 0.1085
Epoch 51/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0669 - val_loss: 0.1086
Epoch 52/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0684 - val_loss: 0.1024
Epoch 53/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0664 - val_loss: 0.1120
Epoch 54/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0557 - val_loss: 0.1034
Epoch 55/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0578 - val_loss: 0.1102
Epoch 56/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0613 - val_loss: 0.1120
Epoch 57/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0625 - val_loss: 0.1042
Epoch 58/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0671 - val_loss: 0.1152
Epoch 59/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 0.0608 - val_loss: 0.1055
Epoch 60/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0674 - val_loss: 0.1045
Epoch 61/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0524 - val_loss: 0.1096
Epoch 62/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0437 - val_loss: 0.1063
Epoch 63/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0545 - val_loss: 0.1072
Epoch 64/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0445 - val_loss: 0.1072
```

```
Epoch 65/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0480 - val_loss: 0.1118
Epoch 66/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0495 - val_loss: 0.1073
Epoch 67/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.0465 - val_loss: 0.1107
Epoch 68/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0516 - val_loss: 0.1105
Epoch 69/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0516 - val_loss: 0.1115
Epoch 70/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0808 - val_loss: 0.1053
Epoch 71/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0822 - val_loss: 0.1118
Epoch 72/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0530 - val_loss: 0.1095
Epoch 73/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0527 - val_loss: 0.1087
Epoch 74/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0650 - val_loss: 0.1133
Epoch 75/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0503 - val_loss: 0.1191
Epoch 76/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0512 - val_loss: 0.1146
Epoch 77/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0573 - val_loss: 0.1104
Epoch 77: early stopping
```



In [ ]:
```python
# Adding in DropOut Layers
# Resetting the model
model = Sequential([
    Dense(units=30, activation='relu'),
    Dropout(0.5),
    Dense(units=15, activation='relu'),
    Dropout(0.5),
    Dense(units=1, activation='sigmoid')
])
model.compile(loss='binary_crossentropy', optimizer='adam')

# Training with Dropout layers
model.fit(x=X_train, y=y_train, epochs=600, validation_data=(
    X_test, y_test), callbacks=[early_stop], verbose=1)
model_loss = pd.DataFrame(model.history.history)
model_loss.plot()
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.show()
```

```
Epoch 1/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - loss: 0.6901 - val_loss: 0.6594
Epoch 2/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.6685 - val_loss: 0.6319
Epoch 3/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.6607 - val_loss: 0.6085
Epoch 4/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.6163 - val_loss: 0.5764
Epoch 5/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.5886 - val_loss: 0.5395
Epoch 6/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.5830 - val_loss: 0.5042
Epoch 7/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.5545 - val_loss: 0.4770
Epoch 8/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.5122 - val_loss: 0.4419
Epoch 9/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.4922 - val_loss: 0.4167
Epoch 10/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.4831 - val_loss: 0.3899
Epoch 11/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.4709 - val_loss: 0.3644
Epoch 12/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.4227 - val_loss: 0.3401
Epoch 13/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.4342 - val_loss: 0.3193
Epoch 14/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.4118 - val_loss: 0.3028
Epoch 15/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.3903 - val_loss: 0.2830
Epoch 16/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.3738 - val_loss: 0.2631
Epoch 17/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.3880 - val_loss: 0.2542
Epoch 18/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.3598 - val_loss: 0.2362
Epoch 19/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.3263 - val_loss: 0.2241
Epoch 20/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.2939 - val_loss: 0.2116
Epoch 21/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.3107 - val_loss: 0.2172
Epoch 22/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.3118 - val_loss: 0.1981
Epoch 23/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.2791 - val_loss: 0.1878
Epoch 24/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.3027 - val_loss: 0.1875
Epoch 25/600
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.3069 - val_loss: 0.1790
Epoch 25: early stopping
```



In [ ]:
```python
# Model Evaluation using the `predict` method
predictions = model.predict(X_test)

# Convert probabilities to class labels
predictions = (predictions > 0.5).astype(int)
```

```python
# Now you can use classification_report and confusion_matrix as before
print(classification_report(y_test, predictions))
print(confusion_matrix(y_test, predictions))
```

```
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step
              precision    recall  f1-score   support

           0       0.87      0.96      0.91        55
           1       0.98      0.91      0.94        88

    accuracy                           0.93       143
   macro avg       0.92      0.94      0.93       143
weighted avg       0.93      0.93      0.93       143

[[53  2]
 [ 8 80]]
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Experiment No: 5

Student Name and Roll Number: Piyush Gambhir – 21CSU349	
Semester /Section: 6th Semester – AIML-B (A3)	
Link to Code: ncu-lab-manual-and-end-semester-projects/NCU-CSL312 - DL - Lab Manual/Experiment 5/Experiment 5.ipynb at main · piyush-gambhir/ncu-lab-manual-and-end-semester-projects (github.com)	
Date:	
Faculty Signature:	
Marks:	

Objective(s):

Experiment 5

Problem Statement:

To build an advance ANN classification model for churn modelling data with:

- a. Cross Validation
- b. Grid Search
- c. Checkpoint

GitHub & Google Colab Links:

GitHub Link: https://github.com/piyush-gambhir/ncu-lab-manual-and-end-semester-projects/blob/main/NCU-CSL312%20-%20DL%20-%20Lab%20Manual/Experiment%205/Experiment%205.ipynb

Google Colab Link:

Open in Colab

Installing Dependencies:

In [ ]:
```
! pip install tabulate numpy pandas matplotlib seaborn
```

```
Requirement already satisfied: tabulate in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packa
ges (0.9.0)
Requirement already satisfied: numpy in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages
(1.26.4)
Requirement already satisfied: pandas in c:\users\mainp\appdata\local\programs\python\python311\lib\site-package
s (2.2.2)
Requirement already satisfied: matplotlib in c:\users\mainp\appdata\local\programs\python\python311\lib\site-pac
kages (3.8.4)
Requirement already satisfied: seaborn in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packag
es (0.13.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\mainp\appdata\local\programs\python\python311\
lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-p
ackages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\mainp\appdata\local\programs\python\python311\lib\site
-packages (from pandas) (2024.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\si
te-packages (from matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-p
ackages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\s
ite-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\s
ite-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\sit
e-packages (from matplotlib) (24.0)
Requirement already satisfied: pillow>=8 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-pack
ages (from matplotlib) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\si
te-packages (from matplotlib) (3.1.2)
Requirement already satisfied: six>=1.5 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packa
ges (from python-dateutil>=2.8.2->pandas) (1.16.0)
```

Code

In [ ]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from sklearn.base import BaseEstimator, ClassifierMixin
```

In [ ]:
```
# Load the dataset
data = pd.read_csv("./churn_modelling.csv")

# Drop the columns that are not needed for modeling
```

```python
data = data.drop(['RowNumber', 'CustomerId', 'Surname'], axis=1)

# Separate features and target variable
X = data.drop('Exited', axis=1)
y = data['Exited']

# Preprocessing for numeric columns: scale numeric features
numeric_features = X.select_dtypes(
    include=['int64', 'float64']).columns.difference(['HasCrCard', 'IsActiveMember'])
numeric_transformer = StandardScaler()

# Preprocessing for categorical columns: one-hot encode categorical features
categorical_features = ['Geography', 'Gender']
categorical_transformer = OneHotEncoder(drop='first')

# Create the preprocessing pipeline
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ])
```

In [ ]:
```python
# Define the Keras Classifier Wrapper


class KerasClassifierWrapper(BaseEstimator, ClassifierMixin):
    def __init__(self, neurons=64):
        self.neurons = neurons
        self.model = None

    def fit(self, X, y, **kwargs):
        def create_model():
            model = Sequential()
            model.add(Dense(self.neurons, activation='relu',
                      input_shape=(X.shape[1],)))
            model.add(Dropout(0.2))
            model.add(Dense(self.neurons, activation='relu'))
            model.add(Dropout(0.2))
            model.add(Dense(1, activation='sigmoid'))
            model.compile(optimizer='adam',
                          loss='binary_crossentropy', metrics=['accuracy'])
            return model

        self.model = create_model()
        self.model.fit(X, y, **kwargs)
        return self

    def predict(self, X, **kwargs):
        return (self.model.predict(X, **kwargs) > 0.5).astype("int32")

    def score(self, X, y, **kwargs):
        _, accuracy = self.model.evaluate(X, y, **kwargs)
        return accuracy

    def get_params(self, deep=True):
        return {'neurons': self.neurons}

    def set_params(self, **parameters):
        for parameter, value in parameters.items():
            setattr(self, parameter, value)
        return self
```

In [ ]:
```python
# Split the data
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)

# Set up a pipeline that includes preprocessing and the estimator
pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                           ('classifier', KerasClassifierWrapper())])

# Hyperparameter grid
param_grid = {
    'classifier__neurons': [32, 64, 128],
}

# Grid search setup
grid = GridSearchCV(pipeline, param_grid, cv=3)

# Perform the grid search
grid_result = grid.fit(X_train, y_train)

# Evaluate the model
print("Best parameters found: ", grid_result.best_params_)
```

```python
print("Best accuracy found: ", grid_result.best_score_)

best_model = grid_result.best_estimator_
X_test_transformed = best_model.named_steps['preprocessor'].transform(X_test)
test_accuracy = best_model.named_steps['classifier'].score(
    X_test_transformed, y_test)
print(f"Test Accuracy: {test_accuracy:.4f}")
```

```
c:\Users\mainp\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\core\dense.py:86: User
Warning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer usin
g an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
167/167 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.7645 - loss: 0.5414
84/84 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.7920 - loss: 0.4683
c:\Users\mainp\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\core\dense.py:86: User
Warning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer usin
g an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
167/167 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.7094 - loss: 0.5776
84/84 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7811 - loss: 0.4674
c:\Users\mainp\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\core\dense.py:86: User
Warning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer usin
g an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
167/167 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.6786 - loss: 0.6017
84/84 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8058 - loss: 0.4477
c:\Users\mainp\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\core\dense.py:86: User
Warning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer usin
g an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
167/167 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.7945 - loss: 0.5081
84/84 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8012 - loss: 0.4535
c:\Users\mainp\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\core\dense.py:86: User
Warning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer usin
g an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
167/167 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.7771 - loss: 0.5325
84/84 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7934 - loss: 0.4441
c:\Users\mainp\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\core\dense.py:86: User
Warning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer usin
g an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
167/167 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.7644 - loss: 0.5186
84/84 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8148 - loss: 0.4228
c:\Users\mainp\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\core\dense.py:86: User
Warning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer usin
g an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
167/167 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.7817 - loss: 0.4978
84/84 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.8183 - loss: 0.4240
c:\Users\mainp\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\core\dense.py:86: User
Warning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer usin
g an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
167/167 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.7870 - loss: 0.4995
84/84 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.7992 - loss: 0.4358
c:\Users\mainp\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\core\dense.py:86: User
Warning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer usin
g an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
167/167 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.7589 - loss: 0.5112
84/84 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.8237 - loss: 0.4139
c:\Users\mainp\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\core\dense.py:86: User
Warning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer usin
g an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
250/250 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.7720 - loss: 0.5011
Best parameters found:  {'classifier__neurons': 128}
Best accuracy found:  0.8147505720456442
63/63 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.8328 - loss: 0.3904
Test Accuracy: 0.8415
```

Experiment No: 6

Student Name and Roll Number: Piyush Gambhir – 21CSU349	
Semester /Section: 6th Semester – AIML-B (A3)	
Link to Code: ncu-lab-manual-and-end-semester-projects/NCU-CSL312 - DL - Lab Manual/Experiment 6/Experiment 6.ipynb at main · piyush-gambhir/ncu-lab-manual-and-end-semester-projects (github.com)	
Date:	
Faculty Signature:	
Marks:	

Objective(s):

Experiment 6 - MNSIT Digit Classification Using Keras

Problem Statement:

To perform Convolutional Neural Networks for Image Classification on MNIST Dataset.

GitHub & Colab Link:

GitHub Link: https://github.com/piyush-gambhir/ncu-lab-manual-and-end-semester-projects/blob/main/NCU-CSL312%20-%20DL%20-%20Lab%20Manual/Experiment%206/Experiment%206.ipynb

Google Colab Link:

Open in Colab

Installing Dependencies:

In [ ]:
```
! pip install tabulate, numpy, pandas, matplotlib, seaborn
```

```
ERROR: Invalid requirement: 'tabulate,'
```

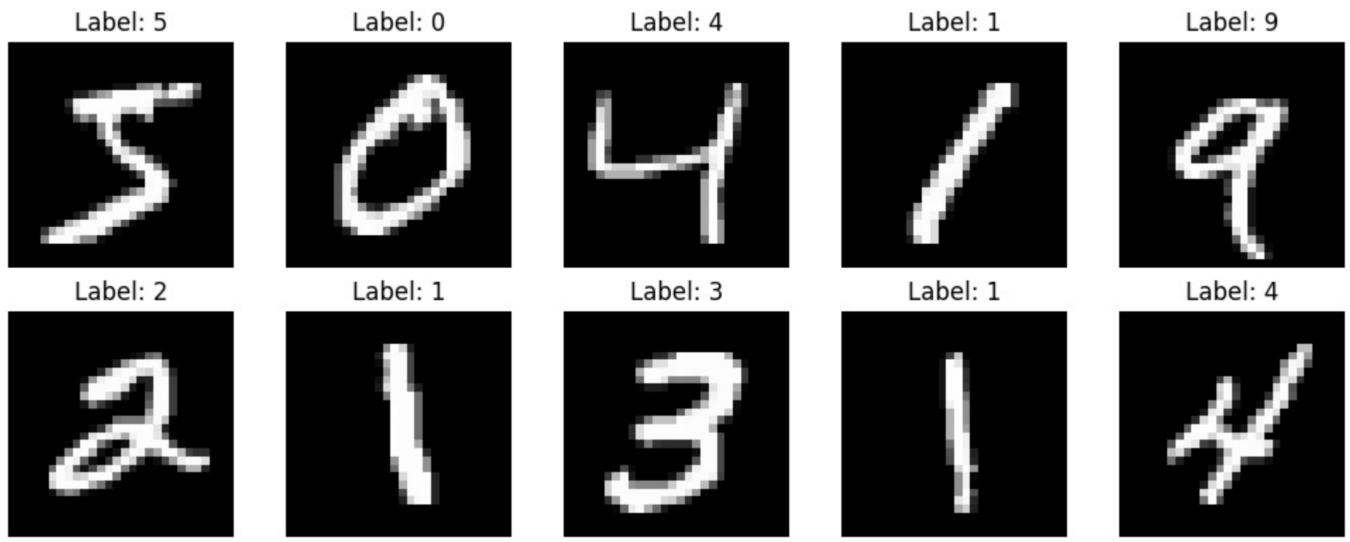
Code

In [ ]:
```
# Task 1: Import Libraries
# Import necessary libraries for data handling and visualization
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.metrics import confusion_matrix
import os
```

In [ ]:
```
# Task 2: Load and Preprocess Data
# Load MNIST data and normalize to facilitate efficient training
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = x_train.reshape(60000, 28*28).astype("float32") / 255
x_test = x_test.reshape(10000, 28*28).astype("float32") / 255
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 2s 0us/step
```

In [ ]:
```
# Task 3: Visualize the Data
# Display the first 10 images from the dataset to understand the data better
plt.figure(figsize=(10, 4))
for i in range(10):
    plt.subplot(2, 5, i + 1)
    plt.imshow(x_train[i].reshape(28, 28), cmap='gray')
    plt.title(f"Label: {y_train[i]}")
    plt.axis('off')
plt.tight_layout()
plt.show()
```



In [ ]:
```
# Task 4: Define and Compile the Model
# Set up the neural network structure and compile the model
model = tf.keras.Sequential([
    tf.keras.layers.Dense(512, activation="relu", input_shape=(28*28,)),
    tf.keras.layers.Dense(10, activation="softmax")
])
model.compile(optimizer="rmsprop", loss="sparse_categorical_crossentropy", metrics=["accuracy"])
model.summary()
```

```
c:\Users\mainp\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\core\dense.py:86: User
Warning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer usin
g an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 512)	401,920
dense_1 (Dense)	(None, 10)	5,130

Total params: 407,050 (1.55 MB)

Trainable params: 407,050 (1.55 MB)

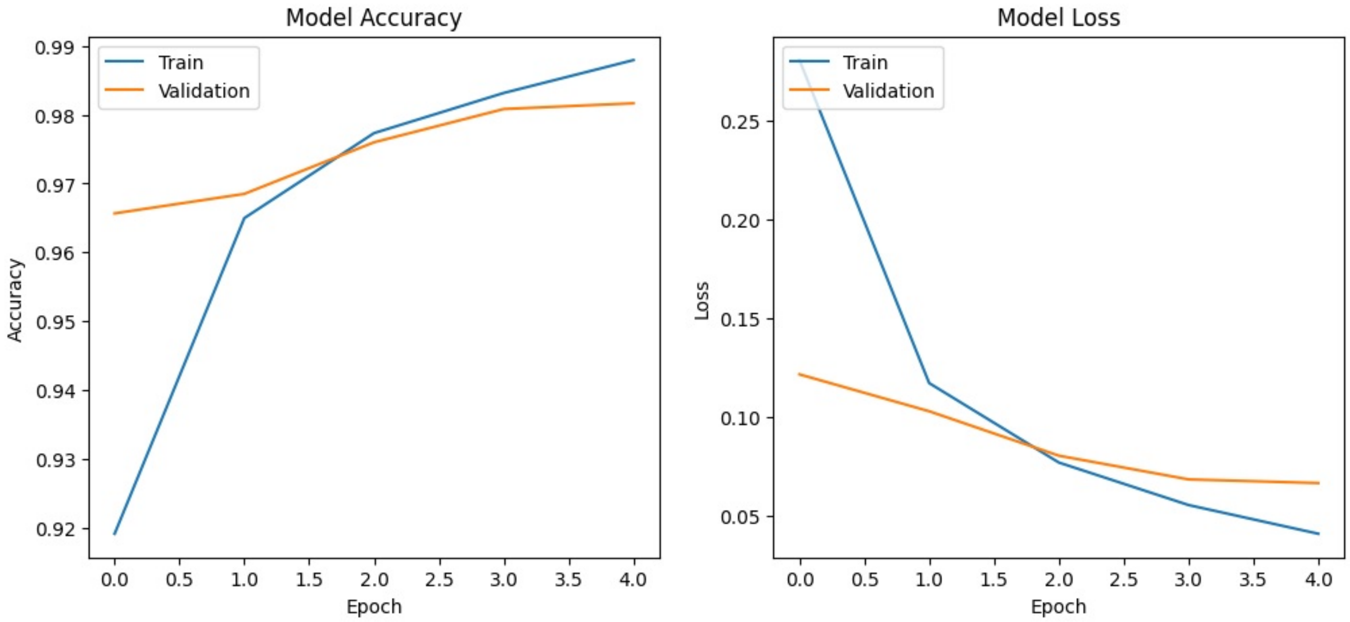
Non-trainable params: 0 (0.00 B)

In [ ]:
```
# Task 5: Train the Model
# Train the model using the training data and validate using part of it
history = model.fit(x_train, y_train, epochs=5, batch_size=128, validation_split=0.1)
```

```
Epoch 1/5
422/422 ━━━━━━━━━━━━━━━━━━━━ 4s 8ms/step - accuracy: 0.8686 - loss: 0.4597 - val_accuracy: 0.9657 - val_loss: 0.
1218
Epoch 2/5
422/422 ━━━━━━━━━━━━━━━━━━━━ 3s 8ms/step - accuracy: 0.9604 - loss: 0.1304 - val_accuracy: 0.9685 - val_loss: 0.
1031
Epoch 3/5
422/422 ━━━━━━━━━━━━━━━━━━━━ 3s 6ms/step - accuracy: 0.9778 - loss: 0.0771 - val_accuracy: 0.9760 - val_loss: 0.
0806
Epoch 4/5
422/422 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - accuracy: 0.9835 - loss: 0.0564 - val_accuracy: 0.9808 - val_loss: 0.
0686
Epoch 5/5
422/422 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - accuracy: 0.9888 - loss: 0.0390 - val_accuracy: 0.9817 - val_loss: 0.
0668
```

In [ ]:
```
# Task 6: Evaluate Model Performance
# Plot accuracy and loss graphs to review the training and validation performance
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
```

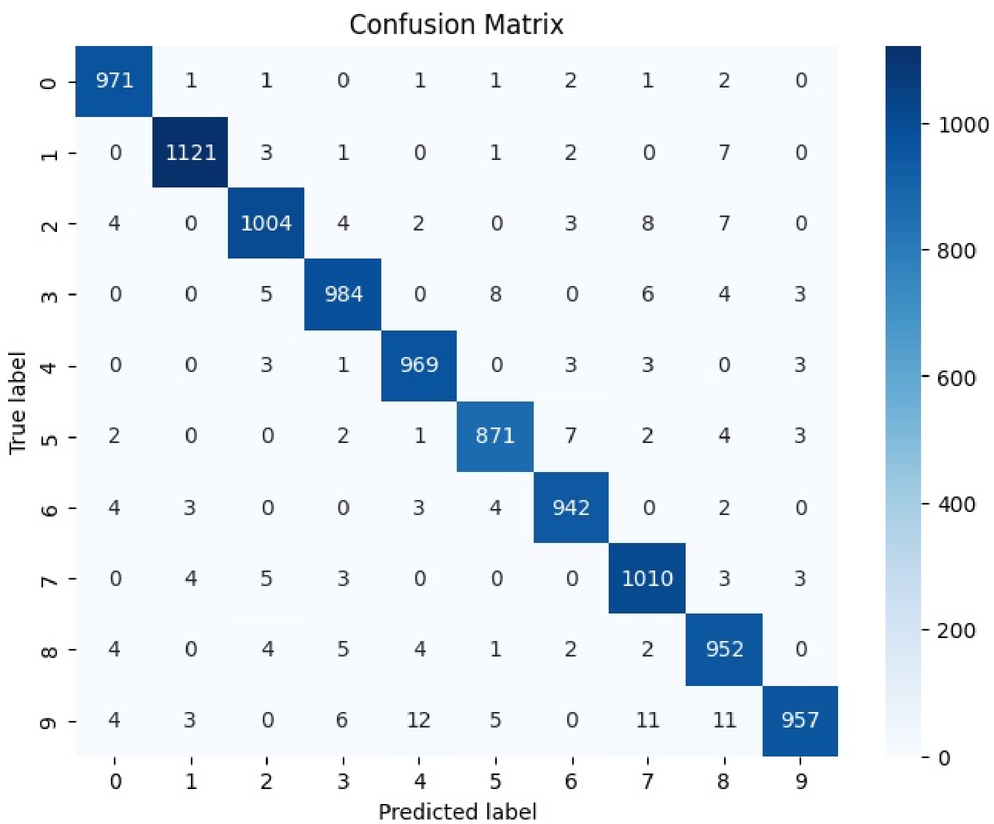
```python
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()

# Evaluate the model on test data and print the test accuracy
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f"Test accuracy: {test_acc}")
```




```
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9754 - loss: 0.0796
Test accuracy: 0.9781000018119812
```

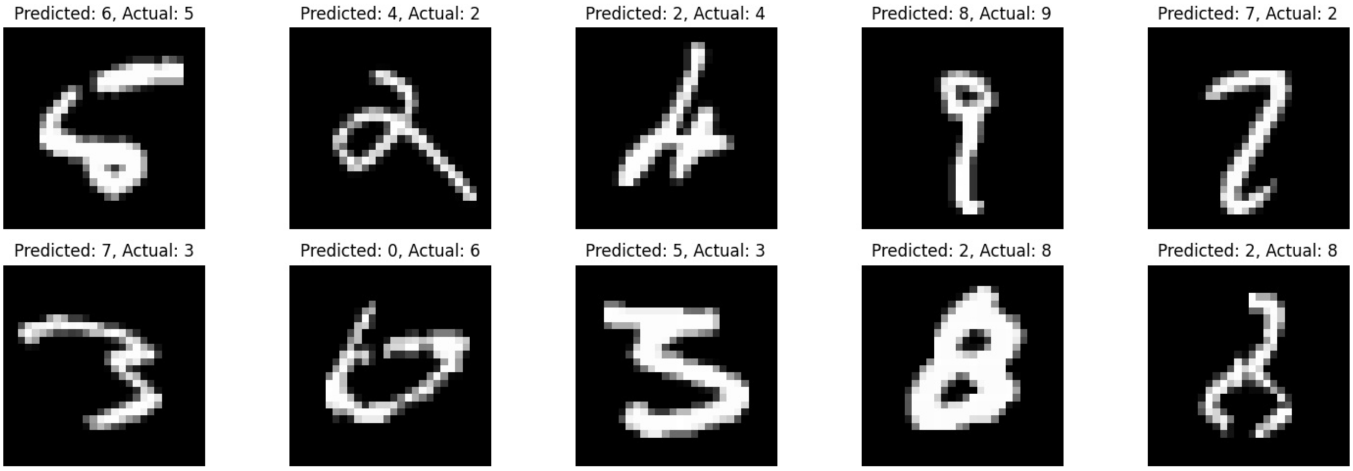
In [ ]:
```python
# Task 7: Analyze Errors with a Confusion Matrix
# Generate predictions, calculate the confusion matrix, and visualize it
preds = model.predict(x_test)
pred_classes = np.argmax(preds, axis=1)
cm = confusion_matrix(y_test, pred_classes)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap='Blues')
plt.title('Confusion Matrix')
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```



In [ ]:
```python
# Task 8: Visualize Misclassifications
```

```python
# Display images that were misclassified to analyze potential reasons
misclassified_idxs = np.where(pred_classes != y_test)[0]
plt.figure(figsize=(15, 5))
for i, mis_idx in enumerate(misclassified_idxs[:10]):
    plt.subplot(2, 5, i + 1)
    plt.imshow(x_test[mis_idx].reshape(28, 28), cmap='gray')
    plt.title(f"Predicted: {pred_classes[mis_idx]}, Actual: {y_test[mis_idx]}")
    plt.axis('off')
plt.tight_layout()
plt.show()
```



Experiment No: 7

Student Name and Roll Number: Piyush Gambhir – 21CSU349	
Semester /Section: 6th Semester – AIML-B (A3)	
Link to Code: ncu-lab-manual-and-end-semester-projects/NCU-CSL312 - DL - Lab Manual at main · piyush-gambhir/ncu-lab-manual-and-end-semester-projects (github.com)	
Date:	
Faculty Signature:	
Marks:	

Objective(s):
To create CNN model with dataset containing images of cats and dogs for image classification

Experiment 7 - CNN Model - Cats & Dogs Classification

Problem Statement:

To create CNN model with dataset containing images of cats and dogs for image classification.

GitHub & Colab Link:

GitHub Link: https://github.com/piyush-gambhir/ncu-lab-manual-and-end-semester-projects/blob/main/NCU-CSL312%20-%20DL%20-%20Lab%20Manual/Experiment%207/Experiment%207.ipynb

Google Colab Link:

Open in Colab

Dataset

Dataset Link: https://www.kaggle.com/datasets/tongpython/cat-and-dog

Installing Dependencies:

In [ ]:
```
! pip install tabulate numpy pandas matplotlib seaborn scikit-learn tensorflow keras
```

```
Requirement already satisfied: tabulate in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packa
ges (0.9.0)
Requirement already satisfied: numpy in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages
(1.26.4)
Requirement already satisfied: pandas in c:\users\mainp\appdata\local\programs\python\python311\lib\site-package
s (2.2.2)
Requirement already satisfied: matplotlib in c:\users\mainp\appdata\local\programs\python\python311\lib\site-pac
kages (3.8.4)
Requirement already satisfied: seaborn in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packag
es (0.13.2)
Requirement already satisfied: scikit-learn in c:\users\mainp\appdata\local\programs\python\python311\lib\site-p
ackages (1.4.2)
Requirement already satisfied: tensorflow in c:\users\mainp\appdata\local\programs\python\python311\lib\site-pac
kages (2.16.1)
Requirement already satisfied: keras in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages
(3.2.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\mainp\appdata\local\programs\python\python311\
lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-p
ackages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\mainp\appdata\local\programs\python\python311\lib\site
-packages (from pandas) (2024.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\si
te-packages (from matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-p
ackages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\s
ite-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\s
ite-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\sit
e-packages (from matplotlib) (24.0)
Requirement already satisfied: pillow>=8 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-pack
ages (from matplotlib) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\si
te-packages (from matplotlib) (3.1.2)
Requirement already satisfied: scipy>=1.6.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-p
ackages (from scikit-learn) (1.13.0)
Requirement already satisfied: joblib>=1.2.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-
packages (from scikit-learn) (1.4.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\mainp\appdata\local\programs\python\python311\li
b\site-packages (from scikit-learn) (3.4.0)
Requirement already satisfied: tensorflow-intel==2.16.1 in c:\users\mainp\appdata\local\programs\python\python31
1\lib\site-packages (from tensorflow) (2.16.1)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\site
-packages (from tensorflow-intel==2.16.1->tensorflow) (2.1.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\s
ite-packages (from tensorflow-intel==2.16.1->tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in c:\users\mainp\appdata\local\programs\python\python311\li
b\site-packages (from tensorflow-intel==2.16.1->tensorflow) (24.3.25)
```

```
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (3.11.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes~=0.3.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (0.3.2)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (3.3.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (4.25.3)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (2.31.0)
Requirement already satisfied: setuptools in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (65.5.0)
Requirement already satisfied: six>=1.12.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (4.11.0)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (1.16.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (1.62.2)
Requirement already satisfied: tensorboard<2.17,>=2.16 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (2.16.2)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (0.31.0)
Requirement already satisfied: rich in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from keras) (13.7.1)
Requirement already satisfied: namex in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from keras) (0.0.8)
Requirement already satisfied: optree in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from keras) (0.11.0)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from rich->keras) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from rich->keras) (2.17.2)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from astunparse>=1.6.0->tensorflow-intel==2.16.1->tensorflow) (0.43.0)
Requirement already satisfied: mdurl~=0.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from markdown-it-py>=2.2.0->rich->keras) (0.1.2)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.16.1->tensorflow) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.16.1->tensorflow) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.16.1->tensorflow) (2.2.1)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.16.1->tensorflow) (2024.2.2)
Requirement already satisfied: markdown>=2.6.8 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from tensorboard<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow) (3.6)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from tensorboard<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from tensorboard<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow) (3.0.2)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages (from werkzeug>=1.0.1->tensorboard<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow) (2.1.5)
```

Code

In [ ]:

```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [ ]:

```python
# Model definition
import warnings
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 3)),
    MaxPooling2D(2, 2),
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
```

```python
    Flatten(),
    Dense(128, activation='relu'),
    Dense(1, activation='sigmoid')
])

# Compiler settings
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Suppress warnings
warnings.filterwarnings('ignore')
```

```
c:\Users\mainp\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\convolutional\base_con
v.py:99: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models
, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(
```

In [ ]:

```python
# Image data augmentation and generators
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.1,
                                   zoom_range=0.1,
                                   horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1./255)

train_set = train_datagen.flow_from_directory('./dataset/training_set',
                                              target_size=(64, 64),
                                              batch_size=32,
                                              class_mode='binary')

test_set = test_datagen.flow_from_directory('./dataset/test_set',
                                            target_size=(64, 64),
                                            batch_size=32,
                                            class_mode='binary')
```

```
Found 8005 images belonging to 2 classes.
Found 2023 images belonging to 2 classes.
```

In [ ]:

```python
# Model training
history = model.fit(
    train_set,
    steps_per_epoch=100,  # Adjust based on your dataset size
    epochs=20,            # Training for 20 epochs
    validation_data=test_set,
    validation_steps=50   # Adjust based on your validation set size
)
```

```
Epoch 1/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 99s 937ms/step - accuracy: 0.4903 - loss: 0.7411 - val_accuracy: 0.5000 - val_loss: 0.6932
Epoch 2/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 66s 662ms/step - accuracy: 0.5265 - loss: 0.6923 - val_accuracy: 0.5579 - val_loss: 0.6862
Epoch 3/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 47s 471ms/step - accuracy: 0.5937 - loss: 0.6724 - val_accuracy: 0.6456 - val_loss: 0.6417
Epoch 4/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 95s 920ms/step - accuracy: 0.6341 - loss: 0.6418 - val_accuracy: 0.6194 - val_loss: 0.6352
Epoch 5/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 97s 976ms/step - accuracy: 0.6680 - loss: 0.6133 - val_accuracy: 0.6906 - val_loss: 0.6060
Epoch 6/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 37s 375ms/step - accuracy: 0.6771 - loss: 0.5966 - val_accuracy: 0.7234 - val_loss: 0.6035
Epoch 7/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 118s 1s/step - accuracy: 0.7049 - loss: 0.5831 - val_accuracy: 0.7138 - val_loss: 0.5685
Epoch 8/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 79s 795ms/step - accuracy: 0.7114 - loss: 0.5515 - val_accuracy: 0.7069 - val_loss: 0.5857
Epoch 9/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 51s 512ms/step - accuracy: 0.7203 - loss: 0.5640 - val_accuracy: 0.7281 - val_loss: 0.5531
Epoch 10/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 67s 634ms/step - accuracy: 0.7249 - loss: 0.5424 - val_accuracy: 0.7707 - val_loss: 0.5357
Epoch 11/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 193s 2s/step - accuracy: 0.7491 - loss: 0.5145 - val_accuracy: 0.7500 - val_loss: 0.5170
Epoch 12/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 93s 938ms/step - accuracy: 0.7325 - loss: 0.5127 - val_accuracy: 0.7163 - val_loss: 0.5456
Epoch 13/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 290s 3s/step - accuracy: 0.7429 - loss: 0.5232 - val_accuracy: 0.7231 - val_loss: 0.5598
Epoch 14/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 173s 2s/step - accuracy: 0.7521 - loss: 0.5081 - val_accuracy: 0.7423 - val_loss: 0.5143
Epoch 15/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 49s 492ms/step - accuracy: 0.7670 - loss: 0.4919 - val_accuracy: 0.7469 - val_loss: 0.5273
Epoch 16/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 73s 677ms/step - accuracy: 0.7573 - loss: 0.5033 - val_accuracy: 0.7683 - val_loss: 0.4925
Epoch 17/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 79s 794ms/step - accuracy: 0.7846 - loss: 0.4787 - val_accuracy: 0.7500 - val_loss: 0.5215
Epoch 18/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 26s 264ms/step - accuracy: 0.7586 - loss: 0.4926 - val_accuracy: 0.7825 - val_loss: 0.4505
Epoch 19/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 90s 875ms/step - accuracy: 0.7781 - loss: 0.4705 - val_accuracy: 0.7725 - val_loss: 0.4837
Epoch 20/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 72s 730ms/step - accuracy: 0.7754 - loss: 0.4696 - val_accuracy: 0.7707 - val_loss: 0.4765
```

In [ ]:

```
# Save the trained model
model.save('cat_dog_classifier_model.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
```

In [ ]:

```
# Model evaluation using the new 'evaluate' method
eval_result = model.evaluate(test_set)
print(f"Validation Accuracy: {eval_result[1]*100:.2f}%, Validation Loss: {eval_result[0]:.6f}")
```

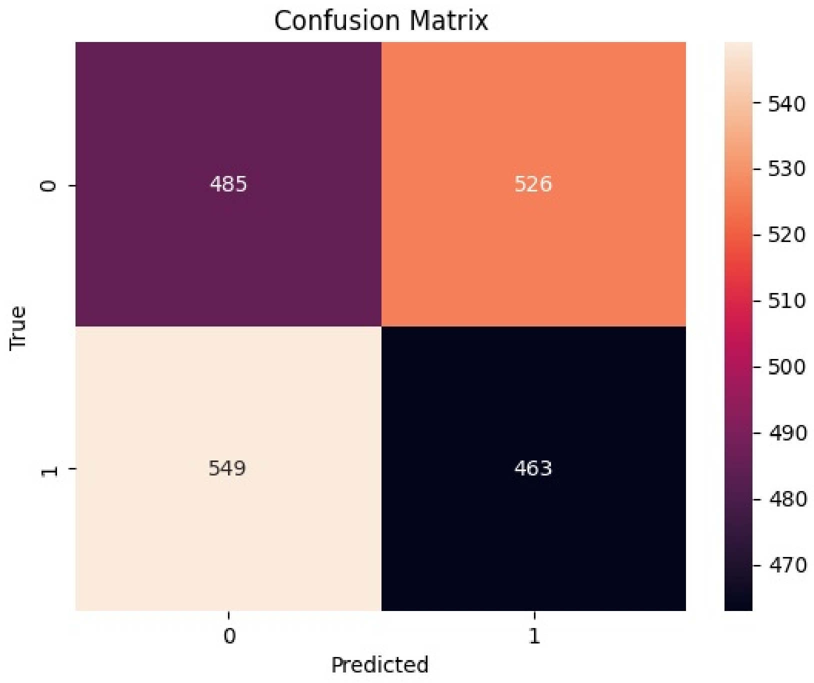
```
64/64 ━━━━━━━━━━━━━━━━━━━━ 27s 419ms/step - accuracy: 0.7738 - loss: 0.4813
Validation Accuracy: 76.52%, Validation Loss: 0.493094
```

In [ ]:

```
# Confusion matrix on the test set
y_pred = model.predict(test_set)
y_pred = (y_pred > 0.5)
y_true = test_set.classes
cm = confusion_matrix(y_true, y_pred)
sns.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
```

```
plt.show()
```

```
64/64 ━━━━━━━━━━━━━━━━━━━━ 25s 386ms/step
```



Experiment No: 8

Student Name and Roll Number: Piyush Gambhir – 21CSU349	
Semester /Section: 6th Semester – AIML-B (A3)	
Link to Code: ncu-lab-manual-and-end-semester-projects/NCU-CSL312 - DL - Lab Manual/Experiment 7/Experiment 7.ipynb at main · piyush-gambhir/ncu-lab-manual-and-end-semester-projects (github.com)	
Date:	
Faculty Signature:	
Marks:	

Objective(s):
To build an image classifier with Keras and Convolutional Neural Networks for the Fashion MNIST dataset.

Experiment 8 - MNIST Digit Classification using Keras

Problem Statement:

To build an image classifier with Keras and Convolutional Neural Networks for the Fashion MNIST dataset.

Objective:

Your task is to build an image classifier with Keras and Convolutional Neural Networks for the Fashion MNIST dataset. This data set includes 10 labels of different clothing types with 28 by 28 *grayscale* images. There is a training set of 60,000 images and 10,000 test images.

GitHub & Google Colab Link:

GitHub Link: https://github.com/piyush-gambhir/ncu-lab-manual-and-end-semester-projects/blob/main/NCU-CSL312%20-%20DL%20-%20Lab%20Manual/Experiment%208/Experiment%208.ipynb

Google Colab Link:

Open in Colab

Installing Dependencies:

In [ ]:
```
! pip install tabulate numpy pandas matplotlib seaborn torch torchvision
```

```
Requirement already satisfied: tabulate in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packa
ges (0.9.0)
Requirement already satisfied: numpy in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages
(1.26.4)
Requirement already satisfied: pandas in c:\users\mainp\appdata\local\programs\python\python311\lib\site-package
s (2.2.2)
Requirement already satisfied: matplotlib in c:\users\mainp\appdata\local\programs\python\python311\lib\site-pac
kages (3.8.4)
Requirement already satisfied: seaborn in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packag
es (0.13.2)
Requirement already satisfied: torch in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages
(2.3.0)
Requirement already satisfied: torchvision in c:\users\mainp\appdata\local\programs\python\python311\lib\site-pa
ckages (0.18.0)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\mainp\appdata\local\programs\python\python311\
lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-p
ackages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\mainp\appdata\local\programs\python\python311\lib\site
-packages (from pandas) (2024.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\si
te-packages (from matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-p
ackages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\s
ite-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\s
ite-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\sit
e-packages (from matplotlib) (24.0)
Requirement already satisfied: pillow>=8 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-pack
ages (from matplotlib) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\si
te-packages (from matplotlib) (3.1.2)
Requirement already satisfied: filelock in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packa
ges (from torch) (3.13.4)
Requirement already satisfied: typing-extensions>=4.8.0 in c:\users\mainp\appdata\local\programs\python\python31
1\lib\site-packages (from torch) (4.11.0)
Requirement already satisfied: sympy in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages
(from torch) (1.12)
Requirement already satisfied: networkx in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packa
ges (from torch) (3.3)
Requirement already satisfied: jinja2 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-package
s (from torch) (3.1.3)
Requirement already satisfied: fsspec in c:\users\mainp\appdata\local\programs\python\python311\lib\site-package
s (from torch) (2024.3.1)
Requirement already satisfied: mkl<=2021.4.0,>=2021.1.1 in c:\users\mainp\appdata\local\programs\python\python31
1\lib\site-packages (from torch) (2021.4.0)
Requirement already satisfied: intel-openmp==2021.* in c:\users\mainp\appdata\local\programs\python\python311\li
b\site-packages (from mkl<=2021.4.0,>=2021.1.1->torch) (2021.4.0)
Requirement already satisfied: tbb==2021.* in c:\users\mainp\appdata\local\programs\python\python311\lib\site-pa
ckages (from mkl<=2021.4.0,>=2021.1.1->torch) (2021.12.0)
Requirement already satisfied: six>=1.5 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packa
ges (from python-dateutil>=2.8.2->pandas) (1.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\sit
e-packages (from jinja2->torch) (2.1.5)
Requirement already satisfied: mpmath>=0.19 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-p
ackages (from sympy->torch) (1.3.0)
```

Code

In [ ]:
```python
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import datasets, transforms
import matplotlib.pyplot as plt
from torch.utils.data.sampler import SubsetRandomSampler
from torch.utils.data import DataLoader
from collections import OrderedDict
```

In [ ]:
```python
# Configuration
config = {
    'batch_size': 64,
    'n_epochs': 35,
    'lr': 0.0007,
    'dropout': 0.25,
    'input_size': 784,  # 28x28 images
    'hidden_sizes': [392, 196, 98, 49],
    'output_size': 10
}
```

In [ ]:
```
# Data Preparation
def load_data():
    transform = transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize((0.5,), (0.5,))
    ])
    train_ds = datasets.FashionMNIST('F_MNIST_data', download=True, train=True, transform=transform)
    test_ds = datasets.FashionMNIST('F_MNIST_data', download=True, train=False, transform=transform)

    # Split train set into training and validation set (80/20)
    num_train = len(train_ds)
    indices = list(range(num_train))
    np.random.shuffle(indices)
    split = int(np.floor(0.2 * num_train))
    train_idx, val_idx = indices[split:], indices[:split]

    # Creating data samplers and loaders
    train_sampler = SubsetRandomSampler(train_idx)
    val_sampler = SubsetRandomSampler(val_idx)

    train_dl = DataLoader(train_ds, batch_size=config['batch_size'], sampler=train_sampler)
    val_dl = DataLoader(train_ds, batch_size=config['batch_size'], sampler=val_sampler)
    test_dl = DataLoader(test_ds, batch_size=config['batch_size'], shuffle=True)

    return train_dl, val_dl, test_dl
```

In [ ]:
```
# Model Architecture
def build_network():
    layers = OrderedDict([
        ('fc1', nn.Linear(config['input_size'], config['hidden_sizes'][0])),
        ('relu1', nn.ReLU()),
        ('drop1', nn.Dropout(config['dropout'])),
        ('fc2', nn.Linear(config['hidden_sizes'][0], config['hidden_sizes'][1])),
        ('relu2', nn.ReLU()),
        ('drop2', nn.Dropout(config['dropout'])),
        ('fc3', nn.Linear(config['hidden_sizes'][1], config['hidden_sizes'][2])),
        ('relu3', nn.ReLU()),
        ('drop3', nn.Dropout(config['dropout'])),
        ('fc4', nn.Linear(config['hidden_sizes'][2], config['hidden_sizes'][3])),
        ('relu4', nn.ReLU()),
        ('output', nn.Linear(config['hidden_sizes'][3], config['output_size'])),
        ('logsoftmax', nn.LogSoftmax(dim=1))
    ])
    model = nn.Sequential(layers)
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    model.to(device)
    return model, device
```

In [ ]:
```
# Training and Validation
def train_validate(model, device, train_dl, val_dl, n_epochs):
    loss_fn = nn.NLLLoss()
    optimizer = optim.Adam(model.parameters(), lr=config['lr'])
    train_losses, val_losses = [], []

    for epoch in range(n_epochs):
        model.train()
        total_train_loss = 0
        for images, labels in train_dl:
            images, labels = images.to(device), labels.to(device)
            images = images.view(images.shape[0], -1)
            optimizer.zero_grad()
            outputs = model(images)
            loss = loss_fn(outputs, labels)
            loss.backward()
            optimizer.step()
            total_train_loss += loss.item()

        avg_train_loss = total_train_loss / len(train_dl)
        train_losses.append(avg_train_loss)
        val_loss, val_acc = validate(model, device, val_dl, loss_fn)
        val_losses.append(val_loss)
        print(f'Epoch {epoch}: Train Loss: {avg_train_loss:.4f}, Val Loss: {val_loss:.4f}, Val Acc: {val_acc:.2

    plot_losses(train_losses, val_losses)
```

In [ ]:
```
def validate(model, device, loader, loss_fn):
    total_loss, total_correct = 0, 0
    model.eval()
    with torch.no_grad():
        for images, labels in loader:
            images, labels = images.to(device), labels.to(device)
            images = images.view(images.shape[0], -1)
```

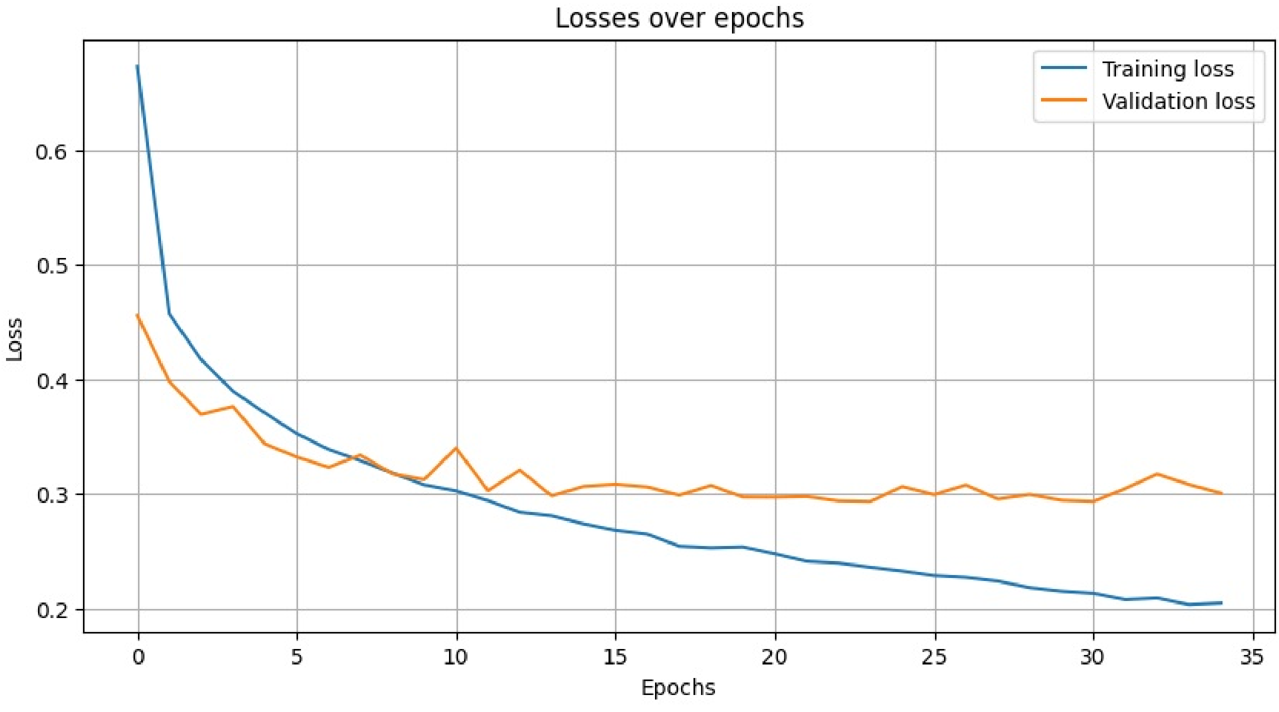
```
            outputs = model(images)
            loss = loss_fn(outputs, labels)
            total_loss += loss.item()
            total_correct += (outputs.argmax(1) == labels).type(torch.float).sum().item()

    avg_loss = total_loss / len(loader)
    accuracy = 100 * total_correct / (len(loader) * config['batch_size'])
    return avg_loss, accuracy
```

In [ ]:
```
def plot_losses(train_losses, val_losses):
    plt.figure(figsize=(10, 5))
    plt.plot(train_losses, label='Training loss')
    plt.plot(val_losses, label='Validation loss')
    plt.title('Losses over epochs')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.grid(True)
    plt.show()
```

In [ ]:
```
# Main
def main():
    train_dl, val_dl, test_dl = load_data()
    model, device = build_network()
    train_validate(model, device, train_dl, val_dl, config['n_epochs'])

if __name__ == '__main__':
    main()
```

```
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-images-idx3-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-images-idx3-ubyte.gz to F_MNIST_dat
a\FashionMNIST\raw\train-images-idx3-ubyte.gz
100.0%
Extracting F_MNIST_data\FashionMNIST\raw\train-images-idx3-ubyte.gz to F_MNIST_data\FashionMNIST\raw

Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-labels-idx1-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-labels-idx1-ubyte.gz to F_MNIST_dat
a\FashionMNIST\raw\train-labels-idx1-ubyte.gz
100.0%
Extracting F_MNIST_data\FashionMNIST\raw\train-labels-idx1-ubyte.gz to F_MNIST_data\FashionMNIST\raw

Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-images-idx3-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-images-idx3-ubyte.gz to F_MNIST_data
\FashionMNIST\raw\t10k-images-idx3-ubyte.gz
100.0%
Extracting F_MNIST_data\FashionMNIST\raw\t10k-images-idx3-ubyte.gz to F_MNIST_data\FashionMNIST\raw

Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-labels-idx1-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-labels-idx1-ubyte.gz to F_MNIST_data
\FashionMNIST\raw\t10k-labels-idx1-ubyte.gz
100.0%
```

```
Extracting F_MNIST_data\FashionMNIST\raw\t10k-labels-idx1-ubyte.gz to F_MNIST_data\FashionMNIST\raw

Epoch 0: Train Loss: 0.6731, Val Loss: 0.4556, Val Acc: 83.71%
Epoch 1: Train Loss: 0.4570, Val Loss: 0.3980, Val Acc: 85.53%
Epoch 2: Train Loss: 0.4171, Val Loss: 0.3694, Val Acc: 86.74%
Epoch 3: Train Loss: 0.3895, Val Loss: 0.3762, Val Acc: 85.88%
Epoch 4: Train Loss: 0.3706, Val Loss: 0.3435, Val Acc: 87.28%
Epoch 5: Train Loss: 0.3525, Val Loss: 0.3322, Val Acc: 87.66%
Epoch 6: Train Loss: 0.3388, Val Loss: 0.3231, Val Acc: 88.18%
Epoch 7: Train Loss: 0.3290, Val Loss: 0.3338, Val Acc: 87.33%
Epoch 8: Train Loss: 0.3182, Val Loss: 0.3175, Val Acc: 88.33%
Epoch 9: Train Loss: 0.3079, Val Loss: 0.3125, Val Acc: 88.60%
Epoch 10: Train Loss: 0.3026, Val Loss: 0.3400, Val Acc: 87.74%
Epoch 11: Train Loss: 0.2941, Val Loss: 0.3029, Val Acc: 89.05%
Epoch 12: Train Loss: 0.2840, Val Loss: 0.3207, Val Acc: 88.74%
Epoch 13: Train Loss: 0.2808, Val Loss: 0.2983, Val Acc: 88.92%
Epoch 14: Train Loss: 0.2738, Val Loss: 0.3065, Val Acc: 89.10%
Epoch 15: Train Loss: 0.2682, Val Loss: 0.3083, Val Acc: 89.01%
Epoch 16: Train Loss: 0.2648, Val Loss: 0.3060, Val Acc: 89.06%
Epoch 17: Train Loss: 0.2543, Val Loss: 0.2988, Val Acc: 89.34%
Epoch 18: Train Loss: 0.2529, Val Loss: 0.3073, Val Acc: 89.47%
Epoch 19: Train Loss: 0.2536, Val Loss: 0.2972, Val Acc: 89.54%
Epoch 20: Train Loss: 0.2477, Val Loss: 0.2971, Val Acc: 89.51%
Epoch 21: Train Loss: 0.2412, Val Loss: 0.2977, Val Acc: 89.81%
Epoch 22: Train Loss: 0.2395, Val Loss: 0.2938, Val Acc: 89.40%
Epoch 23: Train Loss: 0.2359, Val Loss: 0.2933, Val Acc: 89.70%
Epoch 24: Train Loss: 0.2326, Val Loss: 0.3064, Val Acc: 89.49%
Epoch 25: Train Loss: 0.2287, Val Loss: 0.2993, Val Acc: 89.39%
Epoch 26: Train Loss: 0.2271, Val Loss: 0.3078, Val Acc: 89.43%
Epoch 27: Train Loss: 0.2238, Val Loss: 0.2955, Val Acc: 89.46%
Epoch 28: Train Loss: 0.2181, Val Loss: 0.2995, Val Acc: 89.36%

Epoch 29: Train Loss: 0.2149, Val Loss: 0.2944, Val Acc: 89.84%
Epoch 30: Train Loss: 0.2131, Val Loss: 0.2933, Val Acc: 90.28%
Epoch 31: Train Loss: 0.2077, Val Loss: 0.3046, Val Acc: 89.83%
Epoch 32: Train Loss: 0.2091, Val Loss: 0.3172, Val Acc: 89.39%
Epoch 33: Train Loss: 0.2033, Val Loss: 0.3081, Val Acc: 89.68%
Epoch 34: Train Loss: 0.2047, Val Loss: 0.3006, Val Acc: 89.79%
```



Experiment No: 9

Student Name and Roll Number: Piyush Gambhir – 21CSU349	
Semester /Section: 6th Semester – AIML-B (A3)	
Link to Code: ncu-lab-manual-and-end-semester-projects/NCU-CSL312 - DL - Lab Manual/Experiment 9/Experiment 9.ipynb at main · piyush-gambhir/ncu-lab-manual-and-end-semester-projects (github.com)	
Date:	
Faculty Signature:	
Marks:	

Objective(s):

Experiment 9 - Image Classification - Alexnet on CIFAR-10 Dataset

Problem Statement:

To train a CNN model to classify images from the CIFAR-10 database.

GitHub & Google Colab Links:

GitHub Link: https://github.com/piyush-gambhir/ncu-lab-manual-and-end-semester-projects/blob/main/NCU-CSL312%20-%20DL%20-%20Lab%20Manual/Experiment%209/Experiment%209.ipynb

Google Colab Link:

Open in Colab

Installing Dependencies:

In [ ]:
```
! pip install tabulate numpy pandas matplotlib seaborn
```

```
Requirement already satisfied: tabulate in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packa
ges (0.9.0)
Requirement already satisfied: numpy in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages
(1.26.4)
Requirement already satisfied: pandas in c:\users\mainp\appdata\local\programs\python\python311\lib\site-package
s (2.2.2)
Requirement already satisfied: matplotlib in c:\users\mainp\appdata\local\programs\python\python311\lib\site-pac
kages (3.8.4)
Requirement already satisfied: seaborn in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packag
es (0.13.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\mainp\appdata\local\programs\python\python311\
lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-p
ackages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\mainp\appdata\local\programs\python\python311\lib\site
-packages (from pandas) (2024.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\si
te-packages (from matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-p
ackages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\s
ite-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\s
ite-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\sit
e-packages (from matplotlib) (24.0)
Requirement already satisfied: pillow>=8 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-pack
ages (from matplotlib) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\si
te-packages (from matplotlib) (3.1.2)
Requirement already satisfied: six>=1.5 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packa
ges (from python-dateutil>=2.8.2->pandas) (1.16.0)
```

Code

In [ ]:
```
import keras
from keras.datasets import cifar10
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D, ZeroPadding2D
from keras.layers import BatchNormalization

from keras.regularizers import l2
```

In [ ]:
```
# Constants
NUM_CLASSES = 10
BATCH_SIZE = 32
EPOCHS = 1
L2_REG_RATE = 0.01


def load_and_preprocess_data():
```

```python
    # Loads the CIFAR10 dataset
    (x_train, y_train), (x_test, y_test) = cifar10.load_data()

    # One hot encode outputs
    y_train = keras.utils.to_categorical(y_train, NUM_CLASSES)
    y_test = keras.utils.to_categorical(y_test, NUM_CLASSES)

    print('x_train shape:', x_train.shape)
    print(x_train.shape[0], 'train samples')
    print(x_test.shape[0], 'test samples')

    return x_train, y_train, x_test, y_test


def build_alexnet(input_shape):
    # Initialize model
    model = Sequential()

    # 1st Conv Layer
    model.add(Conv2D(96, (11, 11), input_shape=input_shape,
              padding='same', kernel_regularizer=l2(L2_REG_RATE)))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))

    # 2nd through 5th Conv Layers
    layer_configs = [(256, 5, 2), (512, 3, 2), (1024, 3, 0), (1024, 3, 2)]
    for filters, kernel_size, padding in layer_configs:
        if padding:
            model.add(ZeroPadding2D((1, 1)))
        model.add(Conv2D(filters, (kernel_size, kernel_size), padding='same'))
        model.add(BatchNormalization())
        model.add(Activation('relu'))
        if padding:
            model.add(MaxPooling2D(pool_size=(2, 2)))

    # Fully Connected Layers
    model.add(Flatten())
    model.add(Dense(3072))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(Dropout(0.5))

    model.add(Dense(4096))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(Dropout(0.5))

    model.add(Dense(NUM_CLASSES))
    model.add(BatchNormalization())
    model.add(Activation('softmax'))

    return model


def main():
    x_train, y_train, x_test, y_test = load_and_preprocess_data()
    model = build_alexnet(x_train.shape[1:])
    model.compile(loss='categorical_crossentropy',
                  optimizer=keras.optimizers.Adadelta(), metrics=['accuracy'])

    # Train the model
    model.fit(x_train, y_train, batch_size=BATCH_SIZE, epochs=EPOCHS,
              validation_data=(x_test, y_test), shuffle=True)

    # Save the model
    model.save("Trained_Models/CIFAR10_AlexNet_1_Epoch.h5")

    # Evaluate the model
    scores = model.evaluate(x_test, y_test, verbose=1)
    print('Test loss:', scores[0])
    print('Test accuracy:', scores[1])


if __name__ == '__main__':
    main()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 18s 0us/step
x_train shape: (50000, 32, 32, 3)
50000 train samples
10000 test samples
```

```
c:\Users\mainp\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\convolutional\base_con

v.py:99: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models
, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(
 103/1563 ━━━━━━━━━━━━━━━━━━━━ 1:01:58 3s/step - accuracy: 0.1258 - loss: 2.7051
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Experiment No: 10

Student Name and Roll Number: Piyush Gambhir – 21CSU349	
Semester /Section: 6th Semester – AIML-B (A3)	
Link to Code: ncu-lab-manual-and-end-semester-projects/NCU-CSL312 - DL - Lab Manual/Experiment 10/Experiment 10.ipynb at main · piyush-gambhir/ncu-lab-manual-and-end-semester-projects (github.com)	
Date:	
Faculty Signature:	
Marks:	

Objective(s):
To implement transfer 1earing using the pre-trained model (VGG16) on image dataset.

Experiment 10 - Transfer Learning - Pre Trained Model VGG16

Problem Statement:

To implement transfer learning using the pre-trained model (VGG16) on image dataset.

GitHub & Google Colab Link:

GitHub Link: https://github.com/piyush-gambhir/ncu-lab-manual-and-end-semester-projects/blob/main/NCU-CSL312%20-%20DL%20-%20Lab%20Manual/Experiment%2010/Experiment%2010.ipynb

Google Colab Link:

Open in Colab

Installing Dependencies:

In [ ]:
```
! pip install tabulate numpy pandas matplotlib seaborn
```

```
Collecting tabulate
  Downloading tabulate-0.9.0-py3-none-any.whl.metadata (34 kB)
Requirement already satisfied: numpy in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packages
(1.26.4)
Collecting pandas
  Downloading pandas-2.2.2-cp311-cp311-win_amd64.whl.metadata (19 kB)
Requirement already satisfied: matplotlib in c:\users\mainp\appdata\local\programs\python\python311\lib\site-pac
kages (3.8.4)
Collecting seaborn
  Downloading seaborn-0.13.2-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\mainp\appdata\local\programs\python\python311\
lib\site-packages (from pandas) (2.9.0.post0)
Collecting pytz>=2020.1 (from pandas)
  Downloading pytz-2024.1-py2.py3-none-any.whl.metadata (22 kB)
Collecting tzdata>=2022.7 (from pandas)
  Downloading tzdata-2024.1-py2.py3-none-any.whl.metadata (1.4 kB)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\si
te-packages (from matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-p
ackages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\s
ite-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\s
ite-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in c:\users\mainp\appdata\local\programs\python\python311\lib\sit
e-packages (from matplotlib) (24.0)
Requirement already satisfied: pillow>=8 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-pack
ages (from matplotlib) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\mainp\appdata\local\programs\python\python311\lib\si
te-packages (from matplotlib) (3.1.2)
Requirement already satisfied: six>=1.5 in c:\users\mainp\appdata\local\programs\python\python311\lib\site-packa
ges (from python-dateutil>=2.8.2->pandas) (1.16.0)
Downloading tabulate-0.9.0-py3-none-any.whl (35 kB)
Downloading pandas-2.2.2-cp311-cp311-win_amd64.whl (11.6 MB)
   ---------------------------------------- 0.0/11.6 MB ? eta -:--:--
   - -------------------------------------- 0.5/11.6 MB 14.9 MB/s eta 0:00:01
   ---- ----------------------------------- 1.2/11.6 MB 14.8 MB/s eta 0:00:01
   ---- ----------------------------------- 1.4/11.6 MB 11.0 MB/s eta 0:00:01
   ------ --------------------------------- 1.8/11.6 MB 10.2 MB/s eta 0:00:01
   --------- ------------------------------ 2.8/11.6 MB 12.9 MB/s eta 0:00:01
   ------------ --------------------------- 3.5/11.6 MB 13.9 MB/s eta 0:00:01
   ----------------- ---------------------- 5.0/11.6 MB 15.9 MB/s eta 0:00:01
   --------------------- ------------------ 6.3/11.6 MB 17.5 MB/s eta 0:00:01
   -------------------------- ------------- 7.7/11.6 MB 18.9 MB/s eta 0:00:01
   ------------------------------- -------- 9.2/11.6 MB 20.2 MB/s eta 0:00:01
   ----------------------------------- ---- 10.4/11.6 MB 21.1 MB/s eta 0:00:01
   ---------------------------------------  11.6/11.6 MB 26.2 MB/s eta 0:00:01
   ---------------------------------------- 11.6/11.6 MB 24.2 MB/s eta 0:00:00
Downloading seaborn-0.13.2-py3-none-any.whl (294 kB)
   ---------------------------------------- 0.0/294.9 kB ? eta -:--:--
   --------------------------------------- 294.9/294.9 kB 17.8 MB/s eta 0:00:00
Downloading pytz-2024.1-py2.py3-none-any.whl (505 kB)
   ---------------------------------------- 0.0/505.5 kB ? eta -:--:--
   --------------------------------------- 505.5/505.5 kB 16.0 MB/s eta 0:00:00
Downloading tzdata-2024.1-py2.py3-none-any.whl (345 kB)
   ---------------------------------------- 0.0/345.4 kB ? eta -:--:--
   --------------------------------------- 345.4/345.4 kB 20.9 MB/s eta 0:00:00
Installing collected packages: pytz, tzdata, tabulate, pandas, seaborn
Successfully installed pandas-2.2.2 pytz-2024.1 seaborn-0.13.2 tabulate-0.9.0 tzdata-2024.1
```

Code

In [ ]:

```python
import cv2
from keras.applications import vgg16
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input, decode_predictions
import numpy as np
from os import listdir
from os.path import isfile, join
```

In [ ]:

```python
# Define the path to your images
IMAGE_PATH = "images/"

# Load the VGG16 model
vgg_model = vgg16.VGG16(weights='imagenet')
```

In [ ]:

```python
def load_and_preprocess_image(img_path):
    target_size = (224, 224)  # VGG16 uses 224x224 images
    img = image.load_img(img_path, target_size=target_size)
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
```

```python
    x = preprocess_input(x)
    return x

def get_predictions(model, x):
    preds = model.predict(x)
    return decode_predictions(preds, top=3)[0]

def draw_test(name, predictions, input_im):
    BLACK = [0, 0, 0]
    # Calculate needed expansion to fit text
    extra_width = max(len(pred[1]) for pred in predictions) * 20
    expanded_image = cv2.copyMakeBorder(input_im, 0, 0, 0, input_im.shape[1] + extra_width, cv2.BORDER_CONSTANT
    img_width = input_im.shape[1]
    cv2.putText(expanded_image, str(name), (img_width + 10, 30), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 0, 255)
    y_offset = 60
    for i, prediction in enumerate(predictions):
        string = f"{prediction[1]}: {prediction[2]:.2f}"
        cv2.putText(expanded_image, string, (img_width + 10, y_offset + (i * 30)), cv2.FONT_HERSHEY_COMPLEX_SMAI
    cv2.imshow(name, expanded_image)

def process_images():
    file_names = [f for f in listdir(IMAGE_PATH) if isfile(join(IMAGE_PATH, f))]

    for file in file_names:
        img_path = join(IMAGE_PATH, file)
        x = load_and_preprocess_image(img_path)

        # Load image using opencv for display
        img_display = cv2.imread(img_path)
        img_display = cv2.resize(img_display, None, fx=0.5, fy=0.5, interpolation=cv2.INTER_CUBIC)

        # Get predictions from VGG16 model
        predictions_vgg = get_predictions(vgg_model, x)

        # Display results
        draw_test(f"VGG16 Predictions - {file}", predictions_vgg, img_display)
        cv2.waitKey(0)  # Wait for key press to continue

    cv2.destroyAllWindows()
```

In [ ]:

```python
# Main function to execute the process
if __name__ == '__main__':
    process_images()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 903ms/step
```

Output Explanation

Example Output Interpretation: When you run the script, for each image, it displays:

- Name of the image file.
- Top 3 predictions where each line shows:
  - The predicted category.
  - The model's confidence in that prediction expressed as a percentage.

For instance, if the output for an image is:

```
VGG16 Predictions - cat.jpg
Persian cat: 0.45
Tabby cat: 0.30
Siamese cat: 0.10
```

This means:

- The model is 45% confident that the image is of a Persian cat.
- The second most likely category, according to the model, is a tabby cat, with 30% confidence.
- The third guess is a Siamese cat, with 10% confidence.