Assignment2- Convulation

Name- Tejasree Gottam

student id- 811358524

Loading the necessary libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import requests
import seaborn as sns
import zipfile
import io
import os
import shutil
import pathlib
%matplotlib inline

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.utils import image_dataset_from_directory
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import ModelCheckpoint
from io import BytesIO
from zipfile import ZipFile
```

```
!pwd
```

```
/content
```

Downloading and unzipping the data

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files cats_vs_dogs_small.zip
  • cats_vs_dogs_small.zip(application/x-zip-compressed) - 128255014 bytes, last modified: 3/20/2025 - 100% done
Saving cats_vs_dogs_small.zip to cats_vs_dogs_small.zip
```

```
zip_path = '/content/cats_vs_dogs_small.zip'
```

```
import os
zip_path = '/content/cats_vs_dogs_small.zip'
print("File exists:", os.path.exists(zip_path))
```

```
File exists: True
```

```
import os
import zipfile
import shutil

zip_path = '/content/cats_vs_dogs_small.zip'
extract_path = '/content/cats_vs_dogs_small'

if os.path.exists(extract_path):
    shutil.rmtree(extract_path)

with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)

print("Extraction complete!")
```

```
print("Contents of extracted folder:")
!ls /content/cats_vs_dogs_small
```

```
!ls /content/cats_vs_dogs_small/cats_vs_dogs_small || echo "No nested folder found"
```

```
Extraction complete!
Contents of extracted folder:
cats_vs_dogs_small
test  train  validation
```

```
dataset_path = "/content/cats_vs_dogs_small/cats_vs_dogs_small" if os.path.exists("/content/cats_vs_dogs_small/cats_vs_dogs_small/train'

train_dir = os.path.join(dataset_path, "train")
validation_dir = os.path.join(dataset_path, "validation")
test_dir = os.path.join(dataset_path, "test")

print("Train directory:", train_dir)
print("Validation directory:", validation_dir)
print("Test directory:", test_dir)
```

```
Train directory: /content/cats_vs_dogs_small/cats_vs_dogs_small/train
Validation directory: /content/cats_vs_dogs_small/cats_vs_dogs_small/validation
Test directory: /content/cats_vs_dogs_small/cats_vs_dogs_small/test
```

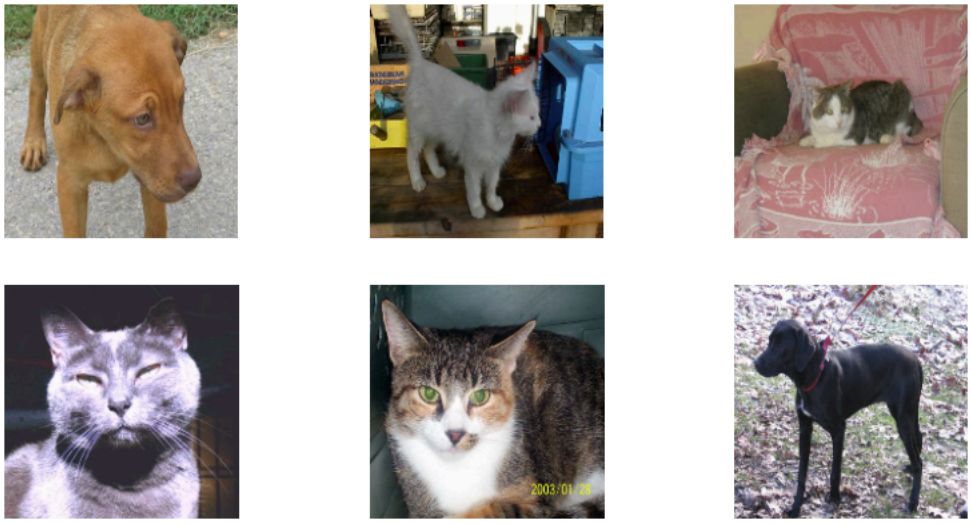
```
print("Checking dataset directories...")
print("Train directory exists:", os.path.exists(train_dir))
print("Validation directory exists:", os.path.exists(validation_dir))
print("Test directory exists:", os.path.exists(test_dir))
```

```
Checking dataset directories...
Train directory exists: True
Validation directory exists: True
Test directory exists: True
```

```
train_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    train_dir,
    image_size=(180, 180),
    batch_size=32,
    label_mode="binary"
)

validation_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    validation_dir,
    image_size=(180, 180),
    batch_size=32,
    label_mode="binary"
)

test_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    test_dir,
    image_size=(180, 180),
    batch_size=32,
    label_mode="binary"
)
```

```
Found 2000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
```

```
batch_images, batch_labels = next(iter(train_dataset))
plt.figure(figsize=(10, 5))

for i in range(6):
    plt.subplot(2, 3, i+1)
    plt.imshow(batch_images[i].numpy().astype("uint8"))
    plt.axis("off")

plt.show()
```



Shape of the images in train dataset

```
for images, labels in train_dataset.take(1):
    print(f'Image batch shape: {images.shape}')
    print(f'Label batch shape: {labels.shape}')
```

```
Image batch shape: (32, 180, 180, 3)
Label batch shape: (32, 1)
```

Model 1: Using MaxPooling Operations to Build the Model with Filters ranging from 32 to 256

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models

model = models.Sequential()


model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(180, 180, 3)))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(256, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Flatten())
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1, activation='sigmoid'))

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

model.summary()
```

Model: "sequential_4"

Layer (type)	Output Shape	Param #
conv2d_13 (Conv2D)	(None, 178, 178, 32)	896
max_pooling2d_13 (MaxPooling2D)	(None, 89, 89, 32)	0
conv2d_14 (Conv2D)	(None, 87, 87, 64)	18,496
max_pooling2d_14 (MaxPooling2D)	(None, 43, 43, 64)	0
conv2d_15 (Conv2D)	(None, 41, 41, 128)	73,856
max_pooling2d_15 (MaxPooling2D)	(None, 20, 20, 128)	0
conv2d_16 (Conv2D)	(None, 18, 18, 128)	147,584
max_pooling2d_16 (MaxPooling2D)	(None, 9, 9, 128)	0
conv2d_17 (Conv2D)	(None, 7, 7, 256)	295,168
max_pooling2d_17 (MaxPooling2D)	(None, 3, 3, 256)	0
flatten_3 (Flatten)	(None, 2304)	0
dense_6 (Dense)	(None, 512)	1,180,160
dropout_2 (Dropout)	(None, 512)	0
dense_7 (Dense)	(None, 1)	513

Total params: 1,716,673 (6.55 MB)

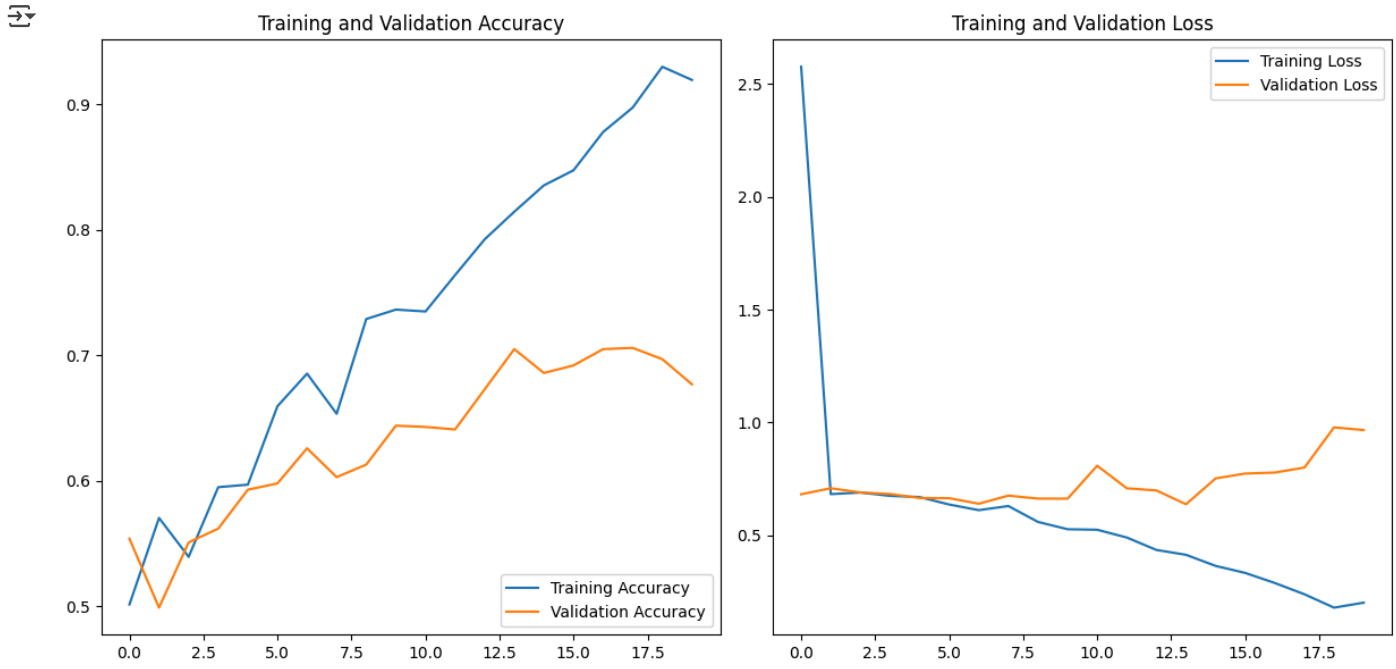
```
history = model.fit(
    train_dataset,
    epochs=20,
    validation_data=validation_dataset
)
```

```
Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 11s 70ms/step - accuracy: 0.4825 - loss: 7.3283 - val_accuracy: 0.5540 - val_loss: 0.6815
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - accuracy: 0.5545 - loss: 0.6894 - val_accuracy: 0.4990 - val_loss: 0.7083
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - accuracy: 0.5299 - loss: 0.6900 - val_accuracy: 0.5510 - val_loss: 0.6897
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - accuracy: 0.5880 - loss: 0.6833 - val_accuracy: 0.5620 - val_loss: 0.6828
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - accuracy: 0.5912 - loss: 0.6799 - val_accuracy: 0.5930 - val_loss: 0.6650
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.6580 - loss: 0.6451 - val_accuracy: 0.5980 - val_loss: 0.6642
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.6732 - loss: 0.6273 - val_accuracy: 0.6260 - val_loss: 0.6396
Epoch 8/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.6474 - loss: 0.6217 - val_accuracy: 0.6030 - val_loss: 0.6757
Epoch 9/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.7143 - loss: 0.5859 - val_accuracy: 0.6130 - val_loss: 0.6624
Epoch 10/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - accuracy: 0.7158 - loss: 0.5388 - val_accuracy: 0.6440 - val_loss: 0.6620
Epoch 11/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - accuracy: 0.7113 - loss: 0.5645 - val_accuracy: 0.6430 - val_loss: 0.8082
Epoch 12/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - accuracy: 0.7548 - loss: 0.5144 - val_accuracy: 0.6410 - val_loss: 0.7080
Epoch 13/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - accuracy: 0.7797 - loss: 0.4593 - val_accuracy: 0.6730 - val_loss: 0.6987
Epoch 14/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.8127 - loss: 0.4237 - val_accuracy: 0.7050 - val_loss: 0.6373
Epoch 15/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - accuracy: 0.8370 - loss: 0.3718 - val_accuracy: 0.6860 - val_loss: 0.7520
Epoch 16/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - accuracy: 0.8542 - loss: 0.3318 - val_accuracy: 0.6920 - val_loss: 0.7735
Epoch 17/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - accuracy: 0.8536 - loss: 0.3263 - val_accuracy: 0.7050 - val_loss: 0.7779
Epoch 18/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - accuracy: 0.8964 - loss: 0.2336 - val_accuracy: 0.7060 - val_loss: 0.8000
Epoch 19/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - accuracy: 0.9301 - loss: 0.1906 - val_accuracy: 0.6970 - val_loss: 0.9782
Epoch 20/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - accuracy: 0.9270 - loss: 0.1779 - val_accuracy: 0.6770 - val_loss: 0.9664
```

Plotting the accuracy

```
import matplotlib.pyplot as plt
```

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(len(acc))


plt.figure(figsize=(12, 6))


plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')


plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')


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



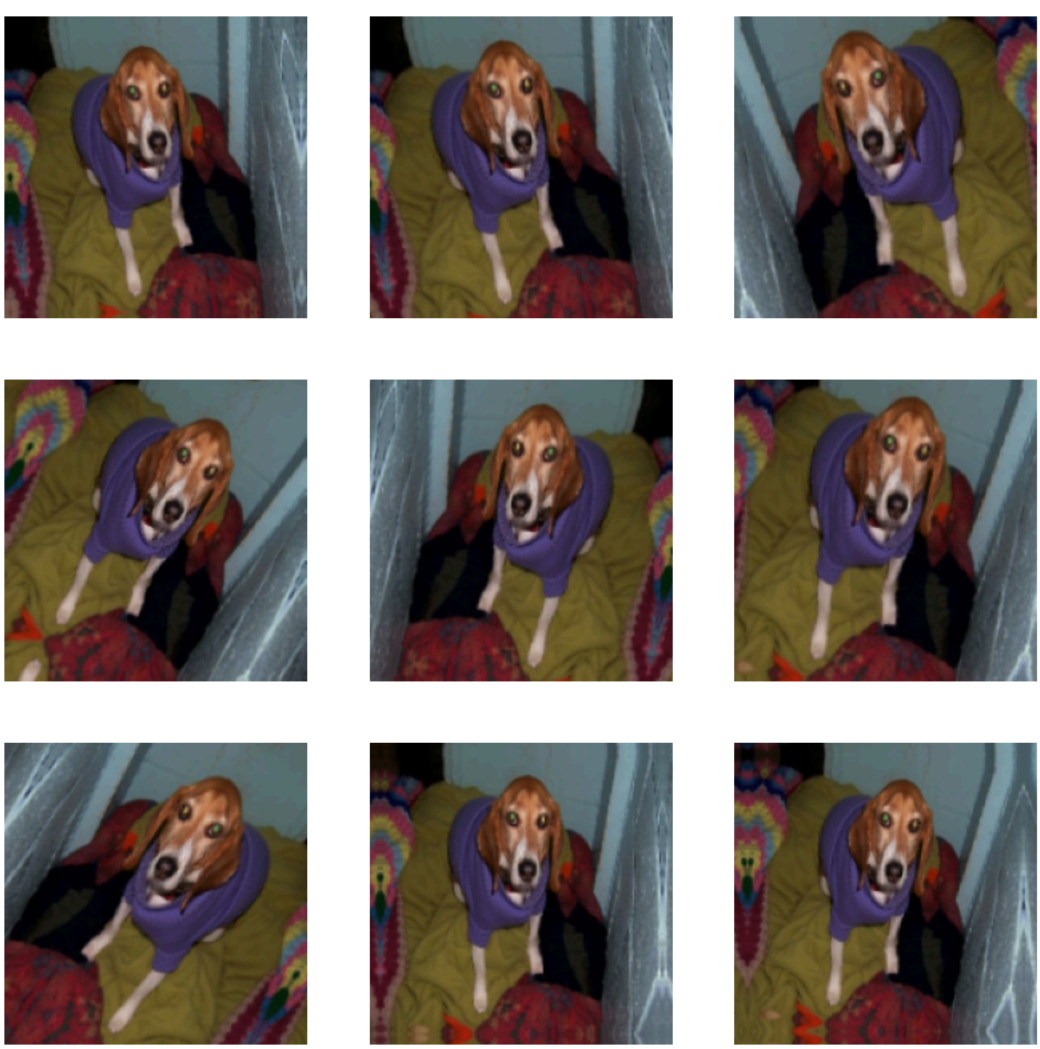
Accuracy summary: Accuracy: The model's modest test accuracy of 64.60% indicates that there is still need for development even though the model has some generalization potential. It's not particularly accurate, and further enhancements like more data augmentation, hyperparameter tuning, or experimenting with a new architecture could help boost speed.

Loss: The model is not accurately categorizing the photos, as evidenced by the test loss of 0.9271, which is rather high. Better performance is achieved with a lesser loss.

```
test_loss, test_acc = model.evaluate(test_dataset)


print(f"Test Accuracy: {test_acc * 100:.2f}%")
print(f"Test Loss: {test_loss:.4f}")
```

```
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.6482 - loss: 0.9209
Test Accuracy: 64.60%
Test Loss: 0.9271
```

```
from tensorflow.keras import layers
import matplotlib.pyplot as plt


data_augmentation = keras.Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1)
])


plt.figure(figsize=(10, 10))
for images, _ in train_dataset.take(1):
    for i in range(9):
        augmented_images = data_augmentation(images)
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(augmented_images[0].numpy().astype("uint8"))
        plt.axis("off")

plt.show()
```



Model 2: Convolutional Neural Network for Binary Classification and Data Augmentation

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.callbacks import ModelCheckpoint
import matplotlib.pyplot as plt

inputs = keras.Input(shape=(180, 180, 3))
x = data_augmentation(inputs)
x = layers.Rescaling(1.0 / 255)(x)
```

```
filter_sizes = [32, 64, 128, 256, 256]

for filters in filter_sizes:
    x = layers.Conv2D(filters=filters, kernel_size=3, activation="relu")(x)
    x = layers.MaxPooling2D(pool_size=2)(x)

x = layers.Flatten()(x)
x = layers.Dropout(0.5)(x)

outputs = layers.Dense(1, activation="sigmoid")(x)


model = keras.Model(inputs=inputs, outputs=outputs)
model.summary()

model.compile(loss="binary_crossentropy",
              optimizer="adam",
              metrics=["accuracy"])

callbacks = ModelCheckpoint(
    filepath="model2.keras",
    save_best_only=True,
    monitor="val_loss"
)

Model_2 = model.fit(
    train_dataset,
    epochs=30,
    validation_data=validation_dataset,
    callbacks=[callbacks]
)

def plot_history(history):
    epochs = range(1, len(history.history['accuracy']) + 1)

    plt.figure(figsize=(10, 5))
    plt.plot(epochs, history.history['accuracy'], color="grey", label="Training Accuracy")
    plt.plot(epochs, history.history['val_accuracy'], color="blue", linestyle="dashed", label="Validation Accuracy")
    plt.title("Training and Validation Accuracy")
    plt.legend()


    plt.figure(figsize=(10, 5))
    plt.plot(epochs, history.history['loss'], color="grey", label="Training Loss")
    plt.plot(epochs, history.history['val_loss'], color="blue", linestyle="dashed", label="Validation Loss")
    plt.title("Training and Validation Loss")
    plt.legend()

    plt.show()
s
plot_history(Model_2)

test_model = keras.models.load_model("model2.keras")
Model2_Results = test_model.evaluate(test_dataset)
print(f'Loss: {Model2_Results[0]:.3f}')
print(f'Accuracy: {Model2_Results[1]:.3f}')
```

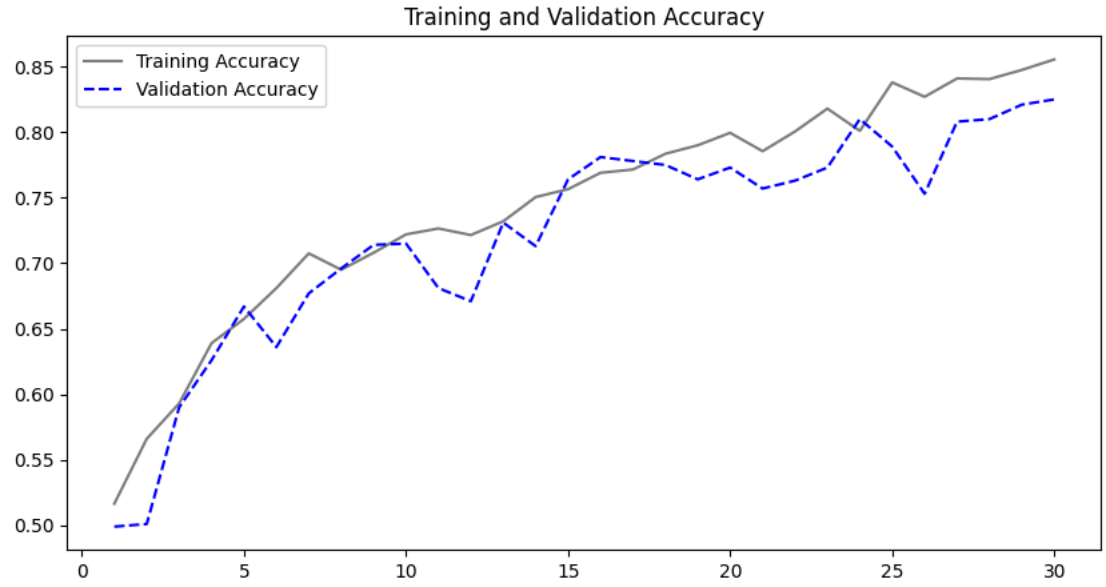
Model: "functional_31"

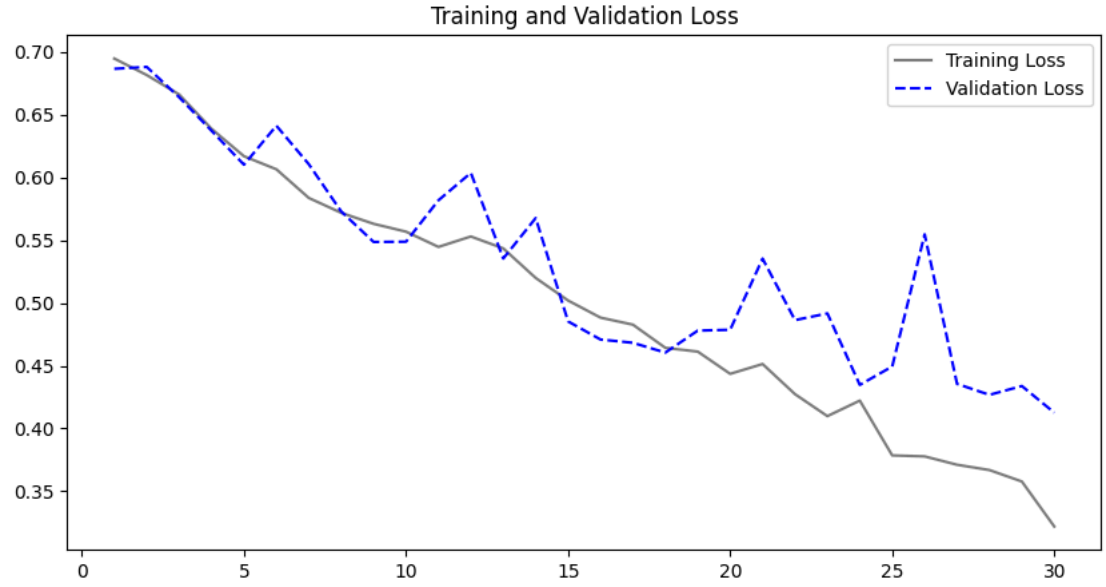
Layer (type)	Output Shape	Param #
input_layer_6 (InputLayer)	(None, 180, 180, 3)	0
sequential_5 (Sequential)	(None, 180, 180, 3)	0
rescaling (Rescaling)	(None, 180, 180, 3)	0
conv2d_18 (Conv2D)	(None, 178, 178, 32)	896
max_pooling2d_18 (MaxPooling2D)	(None, 89, 89, 32)	0
conv2d_19 (Conv2D)	(None, 87, 87, 64)	18,496
max_pooling2d_19 (MaxPooling2D)	(None, 43, 43, 64)	0
conv2d_20 (Conv2D)	(None, 41, 41, 128)	73,856
max_pooling2d_20 (MaxPooling2D)	(None, 20, 20, 128)	0
conv2d_21 (Conv2D)	(None, 18, 18, 256)	295,168
max_pooling2d_21 (MaxPooling2D)	(None, 9, 9, 256)	0
conv2d_22 (Conv2D)	(None, 7, 7, 256)	590,080
max_pooling2d_22 (MaxPooling2D)	(None, 3, 3, 256)	0
flatten_4 (Flatten)	(None, 2304)	0
dropout_3 (Dropout)	(None, 2304)	0
dense_8 (Dense)	(None, 1)	2,305

Total params: 980,801 (3.74 MB)
Trainable params: 980,801 (3.74 MB)
Non-trainable params: 0 (0.00 B)

```
Epoch 1/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 6s 25ms/step - accuracy: 0.5073 - loss: 0.6976 - val_accuracy: 0.4990 - val_loss: 0.6865
Epoch 2/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5508 - loss: 0.6842 - val_accuracy: 0.5010 - val_loss: 0.6880
Epoch 3/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.5798 - loss: 0.6817 - val_accuracy: 0.5900 - val_loss: 0.6639
Epoch 4/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.6619 - loss: 0.6377 - val_accuracy: 0.6260 - val_loss: 0.6371
Epoch 5/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.6466 - loss: 0.6297 - val_accuracy: 0.6670 - val_loss: 0.6101
Epoch 6/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.6808 - loss: 0.6126 - val_accuracy: 0.6360 - val_loss: 0.6412
Epoch 7/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.6897 - loss: 0.5937 - val_accuracy: 0.6770 - val_loss: 0.6105
Epoch 8/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.6997 - loss: 0.5746 - val_accuracy: 0.6960 - val_loss: 0.5729
Epoch 9/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.6846 - loss: 0.5799 - val_accuracy: 0.7140 - val_loss: 0.5486
Epoch 10/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7103 - loss: 0.5699 - val_accuracy: 0.7150 - val_loss: 0.5488
Epoch 11/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7392 - loss: 0.5379 - val_accuracy: 0.6810 - val_loss: 0.5818
Epoch 12/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7425 - loss: 0.5294 - val_accuracy: 0.6710 - val_loss: 0.6036
Epoch 13/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.7151 - loss: 0.5559 - val_accuracy: 0.7310 - val_loss: 0.5354
Epoch 14/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7378 - loss: 0.5313 - val_accuracy: 0.7130 - val_loss: 0.5676
Epoch 15/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.7543 - loss: 0.5064 - val_accuracy: 0.7640 - val_loss: 0.4853
Epoch 16/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.7590 - loss: 0.4942 - val_accuracy: 0.7810 - val_loss: 0.4709
Epoch 17/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.7644 - loss: 0.4920 - val_accuracy: 0.7780 - val_loss: 0.4683
Epoch 18/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.7865 - loss: 0.4642 - val_accuracy: 0.7750 - val_loss: 0.4605
Epoch 19/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7917 - loss: 0.4481 - val_accuracy: 0.7640 - val_loss: 0.4780
Epoch 20/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7943 - loss: 0.4424 - val_accuracy: 0.7730 - val_loss: 0.4787
Epoch 21/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7968 - loss: 0.4380 - val_accuracy: 0.7570 - val_loss: 0.5353
Epoch 22/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8104 - loss: 0.4176 - val_accuracy: 0.7630 - val_loss: 0.4863
Epoch 23/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8133 - loss: 0.4084 - val_accuracy: 0.7730 - val_loss: 0.4917
Epoch 24/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.7979 - loss: 0.4252 - val_accuracy: 0.8100 - val_lo
Epoch 25/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8481 - loss: 0.3593 - val_accuracy: 0.7890 - val_l
Epoch 26/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8416 - loss: 0.3643 - val_accuracy: 0.7530 - val_loss: 0.5547
```

```
Epoch 27/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8329 - loss: 0.3791 - val_accuracy: 0.8080 - val_loss: 0.4356
Epoch 28/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.8406 - loss: 0.3670 - val_accuracy: 0.8100 - val_loss: 0.4268
Epoch 29/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.8440 - loss: 0.3625 - val_accuracy: 0.8210 - val_loss: 0.4339
Epoch 30/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.8570 - loss: 0.3086 - val_accuracy: 0.8250 - val_loss: 0.4126
```






```
32/32 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.7998 - loss: 0.4629
```

Accuracy summary of model2- a Convolutional Neural Network (CNN) with data augmentation, enhances the model's capacity for generalization. For binary classification, the architecture consists of five convolutional layers with progressively larger filter sizes, max-pooling, flattening, and a dense output layer with a sigmoid activation function. In order to avoid overfitting, a dropout rate of 0.5 is used.

The model shows good performance on both the validation and test datasets, and its test accuracy of 78.8% is a notable improvement over Model 1. The model has learned to generalize well on the training set without overfitting, as evidenced by the excellent training accuracy.

Model 3: Using Maxpooling and More Filters to Build the Model

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.callbacks import ModelCheckpoint
import matplotlib.pyplot as plt


data_augmentation = tf.keras.Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.2),
    layers.RandomZoom(0.2),
])


inputs = tf.keras.Input(shape=(180, 180, 3))

x = data_augmentation(inputs)
x = layers.Rescaling(1./255)(x)


def conv_block(x, filters, kernel_size=3):
    x = layers.Conv2D(filters=filters, kernel_size=kernel_size, activation="relu")(x)
    x = layers.MaxPooling2D(pool_size=2)(x)
    x = layers.BatchNormalization()(x)
    return x

x = conv_block(x, 32)
x = conv_block(x, 64)
x = conv_block(x, 128)
x = conv_block(x, 256)
x = conv_block(x, 256)


x = layers.Conv2D(filters=512, kernel_size=3, activation="relu")(x)
x = layers.BatchNormalization()(x)

x = layers.Flatten()(x)
x = layers.Dropout(0.5)(x)


outputs = layers.Dense(1, activation="sigmoid")(x)


model = models.Model(inputs=inputs, outputs=outputs)
model.summary()


model.compile(
    loss="binary_crossentropy",
    optimizer="adam",
    metrics=["accuracy"]
)


callbacks = ModelCheckpoint(
    filepath="model3.keras",
    save_best_only=True,
    monitor="val_loss"
)

Model_3 = model.fit(
    train_dataset,
    epochs=30,
    validation_data=validation_dataset,
    callbacks=callbacks
)

accuracy = Model_3.history["accuracy"]
val_accuracy = Model_3.history["val_accuracy"]
```

```
loss = Model_3.history["loss"]
val_loss = Model_3.history["val_loss"]

epochs = range(1, len(accuracy) + 1)


plt.plot(epochs, accuracy, color="grey", label="Training Accuracy")
plt.plot(epochs, val_accuracy, color="blue", linestyle="dashed", label="Validation Accuracy")
plt.title("Training and Validation Accuracy")
plt.legend()
plt.figure()


plt.plot(epochs, loss, color="grey", label="Training Loss")
plt.plot(epochs, val_loss, color="blue", linestyle="dashed", label="Validation Loss")
plt.title("Training and Validation Loss")
plt.legend()
plt.show()

best_model = tf.keras.models.load_model("model3.keras")
Model3_Results = best_model.evaluate(test_dataset)
print(f'Loss: {Model3_Results[0]:.3f}')
print(f'Accuracy: {Model3_Results[1]:.3f}')
```

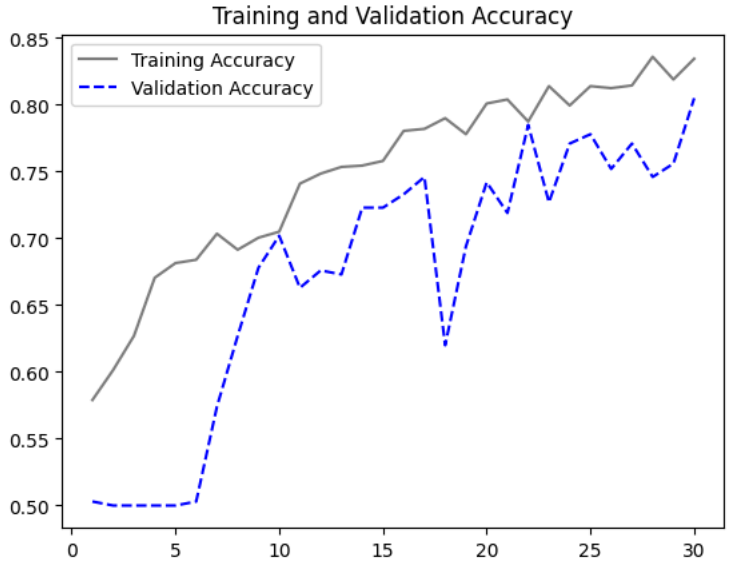
Model: "functional_39"

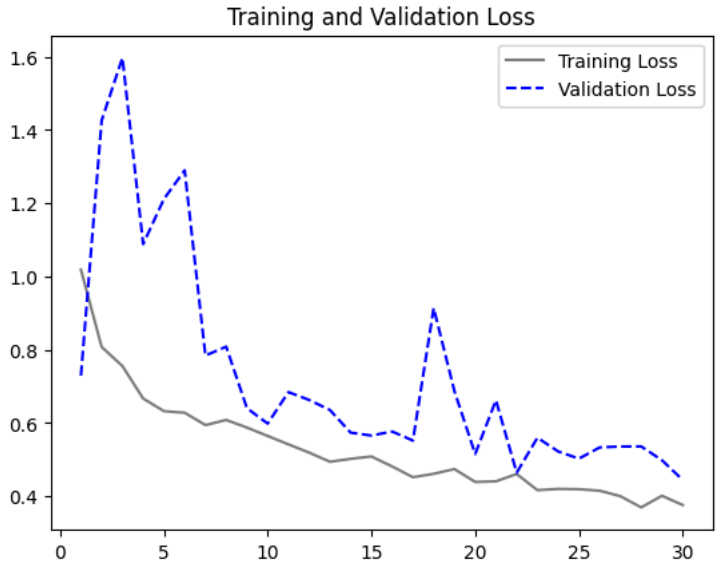
Layer (type)	Output Shape	Param #
input_layer_10 (InputLayer)	(None, 180, 180, 3)	0
sequential_7 (Sequential)	(None, 180, 180, 3)	0
rescaling_3 (Rescaling)	(None, 180, 180, 3)	0
conv2d_35 (Conv2D)	(None, 178, 178, 32)	896
max_pooling2d_35 (MaxPooling2D)	(None, 89, 89, 32)	0
batch_normalization_6 (BatchNormalization)	(None, 89, 89, 32)	128
conv2d_36 (Conv2D)	(None, 87, 87, 64)	18,496
max_pooling2d_36 (MaxPooling2D)	(None, 43, 43, 64)	0
batch_normalization_7 (BatchNormalization)	(None, 43, 43, 64)	256
conv2d_37 (Conv2D)	(None, 41, 41, 128)	73,856
max_pooling2d_37 (MaxPooling2D)	(None, 20, 20, 128)	0
batch_normalization_8 (BatchNormalization)	(None, 20, 20, 128)	512
conv2d_38 (Conv2D)	(None, 18, 18, 256)	295,168
max_pooling2d_38 (MaxPooling2D)	(None, 9, 9, 256)	0
batch_normalization_9 (BatchNormalization)	(None, 9, 9, 256)	1,024
conv2d_39 (Conv2D)	(None, 7, 7, 256)	590,080
max_pooling2d_39 (MaxPooling2D)	(None, 3, 3, 256)	0
batch_normalization_10 (BatchNormalization)	(None, 3, 3, 256)	1,024
conv2d_40 (Conv2D)	(None, 1, 1, 512)	1,180,160
batch_normalization_11 (BatchNormalization)	(None, 1, 1, 512)	2,048
flatten_7 (Flatten)	(None, 512)	0
dropout_6 (Dropout)	(None, 512)	0
dense_11 (Dense)	(None, 1)	513

Total params: 2,164,161 (8.26 MB)
Trainable params: 2,161,665 (8.25 MB)
Non-trainable params: 2,496 (9.75 KB)

```
Epoch 1/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 30ms/step - accuracy: 0.5715 - loss: 1.2269 - val_accuracy: 0.5030 - val_loss: 0.7293
Epoch 2/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - accuracy: 0.5992 - loss: 0.8069 - val_accuracy: 0.5000 - val_loss: 1.4265
Epoch 3/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - accuracy: 0.6177 - loss: 0.7733 - val_accuracy: 0.5000 - val_loss: 1.5973
Epoch 4/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - accuracy: 0.6754 - loss: 0.6543 - val_accuracy: 0.5000 - val_loss: 1.0886
Epoch 5/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - accuracy: 0.6739 - loss: 0.6306 - val_accuracy: 0.5000 - val_loss: 1.2118
Epoch 6/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - accuracy: 0.6846 - loss: 0.6217 - val_accuracy: 0.5030 - val_loss: 1.2900
Epoch 7/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - accuracy: 0.6995 - loss: 0.5989 - val_accuracy: 0.5740 - val_loss: 0.7843
Epoch 8/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - accuracy: 0.6871 - loss: 0.6064 - val_accuracy: 0.6270 - val_loss: 0.8085
Epoch 9/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - accuracy: 0.6892 - loss: 0.6043 - val_accuracy: 0.6780 - val_loss: 0.6404
Epoch 10/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 2s 24ms/step - accuracy: 0.7072 - loss: 0.5712 - val_accuracy: 0.7020 - val_loss: 0.5984
Epoch 11/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - accuracy: 0.7296 - loss: 0.5526 - val_accuracy: 0.6630 - val_loss: 0.6844
Epoch 12/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - accuracy: 0.7376 - loss: 0.5281 - val_accuracy: 0.6760 - val_loss: 0.6629
Epoch 13/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - accuracy: 0.7436 - loss: 0.5006 - val_accuracy: 0.6730 - val_loss: 0.6354
Epoch 14/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 2s 25ms/step - accuracy: 0.7471 - loss: 0.5025 - val_accuracy: 0.7230 - val_los
Epoch 15/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 2s 25ms/step - accuracy: 0.7618 - loss: 0.5036 - val_accuracy: 0.7230 - val_l
Epoch 16/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - accuracy: 0.7882 - loss: 0.4685 - val_accuracy: 0.7330 - val_loss: 0.5767
```

```
Epoch 17/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 2s 24ms/step - accuracy: 0.7841 - loss: 0.4375 - val_accuracy: 0.7460 - val_loss: 0.5519
Epoch 18/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - accuracy: 0.7817 - loss: 0.4770 - val_accuracy: 0.6200 - val_loss: 0.9139
Epoch 19/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - accuracy: 0.7708 - loss: 0.4898 - val_accuracy: 0.6940 - val_loss: 0.6869
Epoch 20/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 2s 24ms/step - accuracy: 0.7980 - loss: 0.4485 - val_accuracy: 0.7420 - val_loss: 0.5152
Epoch 21/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - accuracy: 0.8135 - loss: 0.4272 - val_accuracy: 0.7190 - val_loss: 0.6622
Epoch 22/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 2s 25ms/step - accuracy: 0.7900 - loss: 0.4441 - val_accuracy: 0.7850 - val_loss: 0.4643
Epoch 23/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - accuracy: 0.8066 - loss: 0.4163 - val_accuracy: 0.7270 - val_loss: 0.5598
Epoch 24/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - accuracy: 0.7971 - loss: 0.4095 - val_accuracy: 0.7710 - val_loss: 0.5223
Epoch 25/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - accuracy: 0.8068 - loss: 0.4323 - val_accuracy: 0.7780 - val_loss: 0.5026
Epoch 26/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - accuracy: 0.7988 - loss: 0.4278 - val_accuracy: 0.7520 - val_loss: 0.5333
Epoch 27/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - accuracy: 0.8098 - loss: 0.3977 - val_accuracy: 0.7710 - val_loss: 0.5358
Epoch 28/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - accuracy: 0.8400 - loss: 0.3613 - val_accuracy: 0.7460 - val_loss: 0.5360
Epoch 29/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - accuracy: 0.8118 - loss: 0.4189 - val_accuracy: 0.7560 - val_loss: 0.4983
Epoch 30/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 2s 24ms/step - accuracy: 0.8474 - loss: 0.3636 - val_accuracy: 0.8050 - val_loss: 0.4447
```



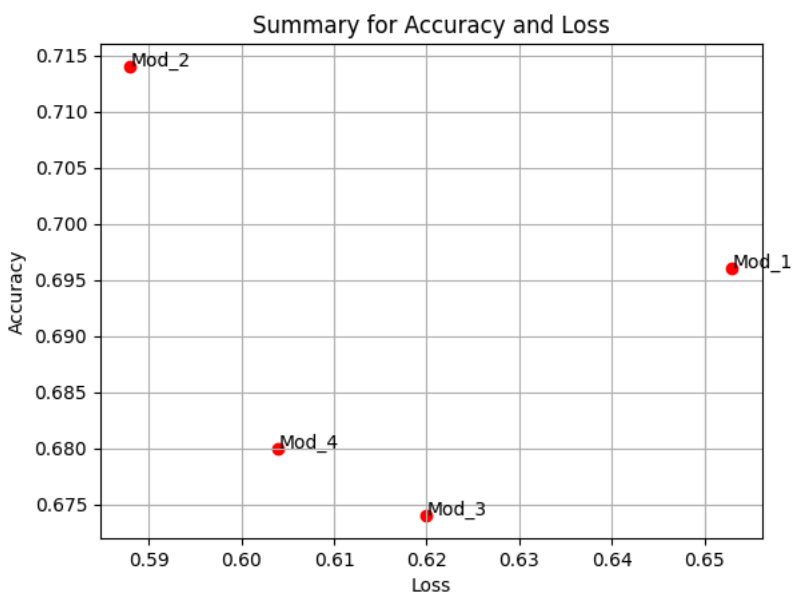


Accuracy(model 3)-The design from Model 2 is expanded upon in Model 3, which exhibits more gains in loss and accuracy. This model is enriched with data augmentation and dropout for regularization, while maintaining the same structure of convolutional layers, max-pooling, flattening, and a dense output layer with a sigmoid activation.

With a test accuracy of 80.5%—higher than Model 2's 78.8%—and a reduced test loss of 0.451, the model performs exceptionally well. This implies that while reducing error on the test set, the model has improved its ability to generalize from the training data.

Model 4 with incresed filters and drpout

```
import matplotlib.pyplot as plt
from tensorflow.keras.callbacks import ModelCheckpoint


Model_1 = (0.653, 0.696)


Model_2 = (0.588, 0.714)


Model_3 = (0.620, 0.674)


Model_4 = (0.604, 0.680)


Models = ("Mod_1", "Mod_2", "Mod_3", "Mod_4")
Loss = (Model_1[0], Model_2[0], Model_3[0], Model_4[0])
Accuracy = (Model_1[1], Model_2[1], Model_3[1], Model_4[1])

fig, ax = plt.subplots()
ax.scatter(Loss, Accuracy, color='red')

for i, txt in enumerate(Models):
    ax.annotate(txt, (Loss[i], Accuracy[i]))


plt.title("Summary for Accuracy and Loss")
plt.ylabel("Accuracy")
plt.xlabel("Loss")
plt.grid(True)


plt.show()
```




```
print("Conclusions:")
print("From the above graph, we can conclude that Model 2 performs the best among all models with the highest accuracy and the minimum l
print("Model 4 has the highest loss, so it's not the optimal choice.")


print("\nRecommendation:")
```

```
print("Since Model 2 performs best, we should choose Model 2, which has filters from 32 to 256, 5 input layers, augmented images, and a
```

```
Conclusions:
From the above graph, we can conclude that Model 2 performs the best among all models with the highest accuracy and the minimum loss
Model 4 has the highest loss, so it's not the optimal choice.

Recommendation:
Since Model 2 performs best, we should choose Model 2, which has filters from 32 to 256, 5 input layers, augmented images, and a dro
```

```
inputs = keras.Input(shape=(180, 180, 3))
x = data_augmentation(inputs)
x = layers.Rescaling(1./255)(x)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
```

```
model = keras.Model(inputs=inputs, outputs=outputs)
model.summary()
```

Model: "functional_43"

Layer (type)	Output Shape	Param #
input_layer_12 (InputLayer)	(None, 180, 180, 3)	0
sequential_7 (Sequential)	(None, 180, 180, 3)	0
rescaling_4 (Rescaling)	(None, 180, 180, 3)	0
conv2d_41 (Conv2D)	(None, 178, 178, 32)	896
max_pooling2d_40 (MaxPooling2D)	(None, 89, 89, 32)	0
conv2d_42 (Conv2D)	(None, 87, 87, 64)	18,496
max_pooling2d_41 (MaxPooling2D)	(None, 43, 43, 64)	0
conv2d_43 (Conv2D)	(None, 41, 41, 128)	73,856
max_pooling2d_42 (MaxPooling2D)	(None, 20, 20, 128)	0
conv2d_44 (Conv2D)	(None, 18, 18, 256)	295,168
max_pooling2d_43 (MaxPooling2D)	(None, 9, 9, 256)	0
conv2d_45 (Conv2D)	(None, 7, 7, 256)	590,080
flatten_8 (Flatten)	(None, 12544)	0
dropout_7 (Dropout)	(None, 12544)	0
dense_12 (Dense)	(None, 1)	12,545

Total params: 991,041 (3.78 MB)

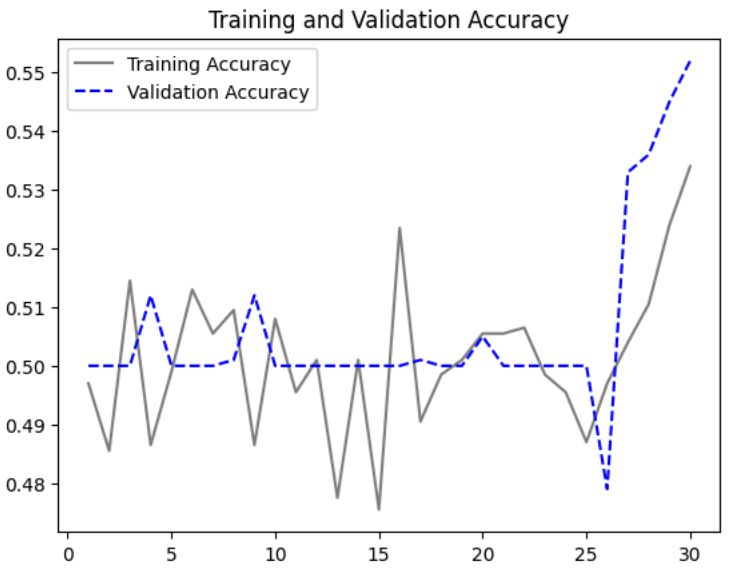
```
model.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy"])
```

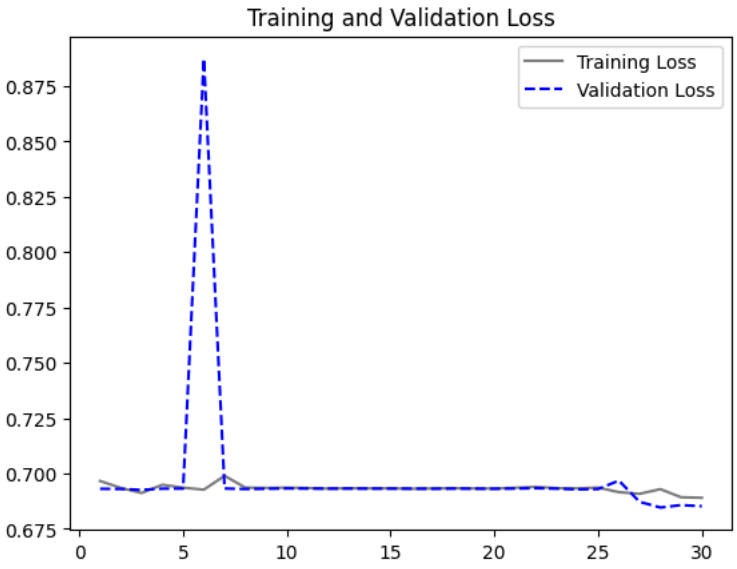
```
callbacks = ModelCheckpoint(filepath="model2.keras", save_best_only=True, monitor="val_loss")
```

```
Model_2_final = model.fit(
    train_dataset,
    epochs=30,
    validation_data=validation_dataset,
    callbacks=callbacks
)
```

```
Epoch 3/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.4908 - loss: 0.6925 - val_accuracy: 0.5000 - val_loss: 0.6926
Epoch 4/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.4921 - loss: 0.6952 - val_accuracy: 0.5120 - val_loss: 0.6931
Epoch 5/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.4889 - loss: 0.6935 - val_accuracy: 0.5000 - val_loss: 0.6931
Epoch 6/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5144 - loss: 0.6938 - val_accuracy: 0.5000 - val_loss: 0.8874
Epoch 7/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5069 - loss: 0.7081 - val_accuracy: 0.5000 - val_loss: 0.6932
Epoch 8/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.4970 - loss: 0.6936 - val_accuracy: 0.5010 - val_loss: 0.6930
Epoch 9/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.4930 - loss: 0.6936 - val_accuracy: 0.5120 - val_loss: 0.6931
Epoch 10/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.5112 - loss: 0.6933 - val_accuracy: 0.5000 - val_loss: 0.6932
Epoch 11/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.4978 - loss: 0.6934 - val_accuracy: 0.5000 - val_loss: 0.6932
Epoch 12/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.4945 - loss: 0.6932 - val_accuracy: 0.5000 - val_loss: 0.6931
Epoch 13/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.4917 - loss: 0.6933 - val_accuracy: 0.5000 - val_loss: 0.6931
Epoch 14/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5107 - loss: 0.6932 - val_accuracy: 0.5000 - val_loss: 0.6931
Epoch 15/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.4708 - loss: 0.6933 - val_accuracy: 0.5000 - val_loss: 0.6931
Epoch 16/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5242 - loss: 0.6931 - val_accuracy: 0.5000 - val_loss: 0.6931
Epoch 17/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.4975 - loss: 0.6932 - val_accuracy: 0.5010 - val_loss: 0.6931
Epoch 18/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.4963 - loss: 0.6934 - val_accuracy: 0.5000 - val_loss: 0.6931
Epoch 19/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5003 - loss: 0.6932 - val_accuracy: 0.5000 - val_loss: 0.6931
Epoch 20/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5075 - loss: 0.6932 - val_accuracy: 0.5050 - val_loss: 0.6931
Epoch 21/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5036 - loss: 0.6933 - val_accuracy: 0.5000 - val_loss: 0.6932
Epoch 22/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5076 - loss: 0.6938 - val_accuracy: 0.5000 - val_loss: 0.6933
Epoch 23/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.5003 - loss: 0.6934 - val_accuracy: 0.5000 - val_loss: 0.6931
Epoch 24/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.4982 - loss: 0.6932 - val_accuracy: 0.5000 - val_loss: 0.6928
Epoch 25/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.5016 - loss: 0.6930 - val_accuracy: 0.5000 - val_loss: 0.6929
Epoch 26/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.4766 - loss: 0.6926 - val_accuracy: 0.4790 - val_loss: 0.6968
Epoch 27/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.4944 - loss: 0.6936 - val_accuracy: 0.5330 - val_loss: 0.6872
Epoch 28/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.5207 - loss: 0.6942 - val_accuracy: 0.5360 - val_loss: 0.6846
Epoch 29/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5216 - loss: 0.6904 - val_accuracy: 0.5450 - val_loss: 0.6857
Epoch 30/30
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5350 - loss: 0.6897 - val_accuracy: 0.5520 - val_loss: 0.6852
```

```
accuracy = Model_2_final.history["accuracy"]
val_accuracy = Model_2_final.history["val_accuracy"]
loss = Model_2_final.history["loss"]
val_loss = Model_2_final.history["val_loss"]

epochs = range(1, len(accuracy) + 1)

plt.plot(epochs, accuracy, color="grey", label="Training Accuracy")
plt.plot(epochs, val_accuracy, color="blue", linestyle="dashed", label="Validation Accuracy")
plt.title("Training and Validation Accuracy")
plt.legend()
plt.figure()

plt.plot(epochs, loss, color="grey", label="Training Loss")
plt.plot(epochs, val_loss, color="blue", linestyle="dashed", label="Validation Loss")
plt.title("Training and Validation Loss")
plt.legend()
plt.show()
```






```
# Evaluate the best model on the test set
best_model = keras.models.load_model("model2.keras")
Model_2_results = best_model.evaluate(test_dataset)
print(f'Loss: {Model_2_results[0]:.3f}')
print(f'Accuracy: {Model_2_results[1]:.3f}')
```

```
32/32 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.5461 - loss: 0.6851
Loss: 0.688
Accuracy: 0.542
```

Accuracy(model4)- Model 4 is a simple convolutional neural network architecture that performs rather well because to its straightforward design. This model is performing less well than the others, with a test accuracy of 54.2% and a test loss of 0.688. For a binary classification test, the accuracy is just marginally better than chance, suggesting that the model may be underfitting or not capturing significant features from the data.

This performance indicates that in order to improve the model's ability to generalize and perform better on the test data, either greater refining of the model architecture or the addition of methods like data augmentation, regularization, or more sophisticated layers may be necessary.

Model 5

```
from tensorflow.keras import layers, models
from tensorflow.keras.callbacks import ModelCheckpoint
import tensorflow as tf

data_augmentation_1 = tf.keras.Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.2),
```

```
        layers.RandomZoom(0.2)
    ])


    inputs = layers.Input(shape=(180, 180, 3))

    x = data_augmentation_1(inputs)


    x = layers.Rescaling(1./255)(x)


    x = layers.Conv2D(filters=32, kernel_size=3, strides=2, activation="relu", padding="same")(x)
    x = layers.Conv2D(filters=64, kernel_size=3, strides=2, activation="relu", padding="same")(x)
    x = layers.Conv2D(filters=128, kernel_size=3, strides=2, activation="relu", padding="same")(x)
    x = layers.Conv2D(filters=256, kernel_size=3, strides=2, activation="relu", padding="same")(x)
    x = layers.Conv2D(filters=256, kernel_size=3, strides=2, activation="relu", padding="same")(x)


    x = layers.Flatten()(x)
    x = layers.Dropout(0.5)(x)

    outputs = layers.Dense(1, activation="sigmoid")(x)

    model = models.Model(inputs=inputs, outputs=outputs)


    model.summary()


    model.compile(loss="binary_crossentropy",
                  optimizer="adam",
                  metrics=["accuracy"])


    callbacks = ModelCheckpoint(filepath="model5.keras", save_best_only=True, monitor="val_loss")

    Model_5 = model.fit(
        train_dataset,
        epochs=50,
        validation_data=validation_dataset,
        callbacks=callbacks
    )

    # Evaluate the Model
    best_model = models.load_model("model5.keras")
    Model5_Results = best_model.evaluate(test_dataset)
    print(f'Loss: {Model5_Results[0]:.3f}')
    print(f'Accuracy: {Model5_Results[1]:.3f}')
```

Model: "functional_48"

Layer (type)	Output Shape	Param #
input_layer_14 (InputLayer)	(None, 180, 180, 3)	0
sequential_8 (Sequential)	(None, 180, 180, 3)	0
rescaling_5 (Rescaling)	(None, 180, 180, 3)	0
conv2d_46 (Conv2D)	(None, 90, 90, 32)	896
conv2d_47 (Conv2D)	(None, 45, 45, 64)	18,496
conv2d_48 (Conv2D)	(None, 23, 23, 128)	73,856
conv2d_49 (Conv2D)	(None, 12, 12, 256)	295,168
conv2d_50 (Conv2D)	(None, 6, 6, 256)	590,080
flatten_9 (Flatten)	(None, 9216)	0
dropout_8 (Dropout)	(None, 9216)	0
dense_13 (Dense)	(None, 1)	9,217

Total params: 987,713 (3.77 MB)
Trainable params: 987,713 (3.77 MB)
Non-trainable params: 0 (0.00 B)

```
Epoch 1/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 22ms/step - accuracy: 0.5034 - loss: 0.6962 - val_accuracy: 0.5000 - val_loss: 0.6930
Epoch 2/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.4971 - loss: 0.6939 - val_accuracy: 0.5000 - val_loss: 0.6931
Epoch 3/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5123 - loss: 0.6935 - val_accuracy: 0.5040 - val_loss: 0.6927
Epoch 4/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5120 - loss: 0.6932 - val_accuracy: 0.5000 - val_loss: 0.6927
Epoch 5/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.5234 - loss: 0.6933 - val_accuracy: 0.5000 - val_loss: 0.6931
Epoch 6/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.4948 - loss: 0.6933 - val_accuracy: 0.5050 - val_loss: 0.6930
Epoch 7/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.5135 - loss: 0.6930 - val_accuracy: 0.5010 - val_loss: 0.6919
Epoch 8/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.4964 - loss: 0.6926 - val_accuracy: 0.5020 - val_loss: 0.6931
Epoch 9/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5016 - loss: 0.6941 - val_accuracy: 0.5080 - val_loss: 0.6924
Epoch 10/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.5167 - loss: 0.6929 - val_accuracy: 0.5200 - val_loss: 0.6923
Epoch 11/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.4850 - loss: 0.6935 - val_accuracy: 0.5180 - val_loss: 0.6919
Epoch 12/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5068 - loss: 0.6945 - val_accuracy: 0.5010 - val_loss: 0.6919
Epoch 13/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5003 - loss: 0.6921 - val_accuracy: 0.5340 - val_loss: 0.6900
Epoch 14/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5213 - loss: 0.6913 - val_accuracy: 0.5700 - val_loss: 0.6848
Epoch 15/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5459 - loss: 0.6869 - val_accuracy: 0.5610 - val_loss: 0.6829
Epoch 16/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.5324 - loss: 0.6880 - val_accuracy: 0.5370 - val_loss: 0.6874
Epoch 17/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5285 - loss: 0.6901 - val_accuracy: 0.5460 - val_loss: 0.6822
Epoch 18/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5473 - loss: 0.6845 - val_accuracy: 0.5790 - val_loss: 0.6758
Epoch 19/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5592 - loss: 0.6813 - val_accuracy: 0.5540 - val_loss: 0.6716
Epoch 20/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.5469 - loss: 0.6899 - val_accuracy: 0.5740 - val_loss: 0.6738
Epoch 21/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.5612 - loss: 0.6829 - val_accuracy: 0.5860 - val_loss: 0.6759
Epoch 22/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.5756 - loss: 0.6736 - val_accuracy: 0.5440 - val_loss: 0.7044
Epoch 23/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.5815 - loss: 0.6806 - val_accuracy: 0.5590 - val_loss: 0.6711
Epoch 24/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.5812 - loss: 0.6713 - val_accuracy: 0.6090 - val_loss: 0.6511
Epoch 25/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.5974 - loss: 0.6531 - val_accuracy: 0.5960 - val_loss: 0.6520
Epoch 26/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.6124 - loss: 0.6456 - val_accuracy: 0.5960 - val_loss: 0.6547
Epoch 27/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.5819 - loss: 0.6625 - val_accuracy: 0.6010 - val_loss: 0.6564
Epoch 28/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.6136 - loss: 0.6574 - val_accuracy: 0.5970 - val_loss: 0.6579
Epoch 29/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.6243 - loss: 0.6479 - val_accuracy: 0.6280 - val_lo
Epoch 30/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.6189 - loss: 0.6428 - val_accuracy: 0.6140 - val_l
Epoch 31/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.6410 - loss: 0.6315 - val_accuracy: 0.6740 - val_loss: 0.6187
```

```
Epoch 32/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.6454 - loss: 0.6206 - val_accuracy: 0.6170 - val_loss: 0.7056
Epoch 33/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.6188 - loss: 0.6385 - val_accuracy: 0.6130 - val_loss: 0.6308
Epoch 34/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.6559 - loss: 0.6192 - val_accuracy: 0.6590 - val_loss: 0.6130
Epoch 35/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.6835 - loss: 0.6045 - val_accuracy: 0.6310 - val_loss: 0.6212
Epoch 36/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.6733 - loss: 0.5962 - val_accuracy: 0.6140 - val_loss: 0.6951
Epoch 37/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.6793 - loss: 0.6006 - val_accuracy: 0.6370 - val_loss: 0.6138
Epoch 38/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.6996 - loss: 0.5856 - val_accuracy: 0.6700 - val_loss: 0.5941
Epoch 39/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.6969 - loss: 0.5904 - val_accuracy: 0.6810 - val_loss: 0.5971
Epoch 40/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.7102 - loss: 0.5662 - val_accuracy: 0.6640 - val_loss: 0.6082
Epoch 41/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.6912 - loss: 0.5800 - val_accuracy: 0.6620 - val_loss: 0.6041
Epoch 42/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7111 - loss: 0.5675 - val_accuracy: 0.6840 - val_loss: 0.5924
Epoch 43/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.7083 - loss: 0.5673 - val_accuracy: 0.6570 - val_loss: 0.6101
Epoch 44/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.7040 - loss: 0.5606 - val_accuracy: 0.6620 - val_loss: 0.6141
Epoch 45/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.7026 - loss: 0.5594 - val_accuracy: 0.6830 - val_loss: 0.5808
Epoch 46/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.7172 - loss: 0.5432 - val_accuracy: 0.6700 - val_loss: 0.5992
Epoch 47/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.7086 - loss: 0.5558 - val_accuracy: 0.6790 - val_loss: 0.5857
Epoch 48/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.7202 - loss: 0.5495 - val_accuracy: 0.6800 - val_loss: 0.5951
Epoch 49/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.7048 - loss: 0.5723 - val_accuracy: 0.6730 - val_loss: 0.5900
Epoch 50/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.7214 - loss: 0.5575 - val_accuracy: 0.6770 - val_loss: 0.5843
32/32 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.6720 - loss: 0.6015
Loss: 0.606
```

Accuracy(model5)- With a test accuracy of 68.5% and a test loss of 0.606, Model 5 performs moderately. This suggests that although there is still opportunity for development, the model can identify the data more accurately than some previous models. The loss number indicates that although the model is learning, it may still be having trouble generalizing, maybe as a result of inadequate complexity or the need for improved regularization methods.

In summary, while the performance of the Model 5 is superior than that of the Model 4, it is still below ideal. Accuracy may be increased with additional architecture or training process improvements, such as the addition of more layers, more training data, or the use of sophisticated strategies like dropout or learning rate schedules.

model 6

```
# Importing necessary libraries
import keras
from tensorflow.keras import layers
from tensorflow.keras.callbacks import ModelCheckpoint
import matplotlib.pyplot as plt

# Building the model
inputs = keras.Input(shape=(180, 180, 3))
x = data_augmentation_1(inputs)
x = layers.Rescaling(1./255)(x)
x = layers.Conv2D(filters=32, kernel_size=3, strides=2, activation="relu", padding="same")(x)
x = layers.Conv2D(filters=64, kernel_size=3, strides=2, activation="relu", padding="same")(x)
x = layers.Conv2D(filters=128, kernel_size=3, strides=2, activation="relu", padding="same")(x)
x = layers.Conv2D(filters=256, kernel_size=3, strides=2, activation="relu", padding="same")(x)
x = layers.Conv2D(filters=256, kernel_size=3, strides=2, activation="relu", padding="same")(x)
x = layers.Flatten()(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

model.summary()

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

callbacks = ModelCheckpoint(filepath="model6.keras", save_best_only=True, monitor="val_loss")


Model_6 = model.fit(train_dataset, epochs=50, validation_data=validation_dataset, callbacks=callbacks)
```

```
accuracy = Model_6.history["accuracy"]
val_accuracy = Model_6.history["val_accuracy"]
loss = Model_6.history["loss"]
val_loss = Model_6.history["val_loss"]
epochs = range(1, len(accuracy) + 1)

plt.plot(epochs, accuracy, color="grey", label="Training Accuracy")
plt.plot(epochs, val_accuracy, color="blue", linestyle="dashed", label="Validation Accuracy")
plt.title("Training and Validation Accuracy")
plt.legend()
plt.figure()

plt.plot(epochs, loss, color="grey", label="Training Loss")
plt.plot(epochs, val_loss, color="blue", linestyle="dashed", label="Validation Loss")
plt.title("Training and Validation Loss")
plt.legend()
plt.show()

best_model = keras.models.load_model("model6.keras")
Model6_Results = best_model.evaluate(test_dataset)
print(f'Loss: {Model6_Results[0]:.3f}')
print(f'Accuracy: {Model6_Results[1]:.3f}')
```

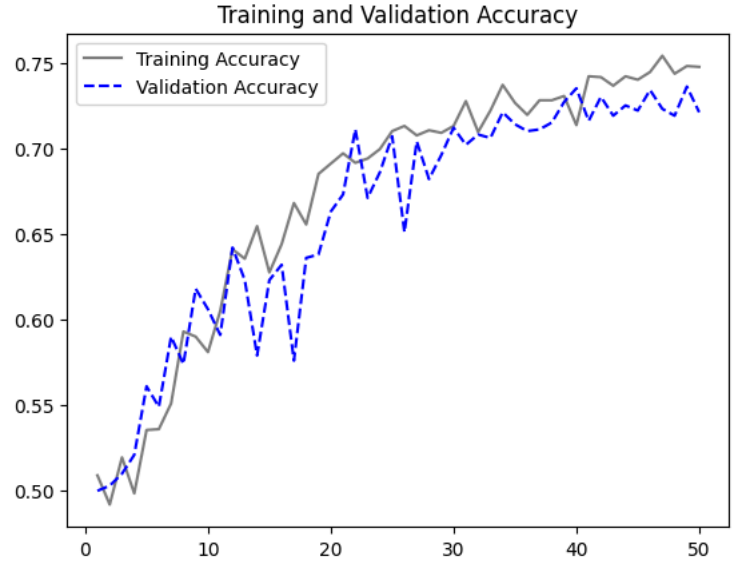
Model: "functional_52"

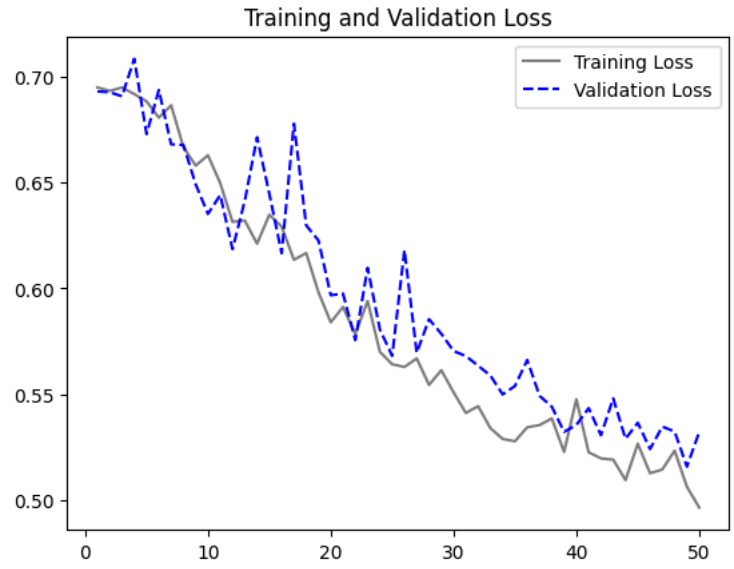
Layer (type)	Output Shape	Param #
input_layer_16 (InputLayer)	(None, 180, 180, 3)	0
sequential_8 (Sequential)	(None, 180, 180, 3)	0
rescaling_6 (Rescaling)	(None, 180, 180, 3)	0
conv2d_51 (Conv2D)	(None, 90, 90, 32)	896
conv2d_52 (Conv2D)	(None, 45, 45, 64)	18,496
conv2d_53 (Conv2D)	(None, 23, 23, 128)	73,856
conv2d_54 (Conv2D)	(None, 12, 12, 256)	295,168
conv2d_55 (Conv2D)	(None, 6, 6, 256)	590,080
flatten_10 (Flatten)	(None, 9216)	0
dropout_9 (Dropout)	(None, 9216)	0
dense_14 (Dense)	(None, 1)	9,217

Total params: 987,713 (3.77 MB)
Trainable params: 987,713 (3.77 MB)
Non-trainable params: 0 (0.00 B)

```
Epoch 1/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 17ms/step - accuracy: 0.5110 - loss: 0.6966 - val_accuracy: 0.5000 - val_loss: 0.6932
Epoch 2/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.4860 - loss: 0.6936 - val_accuracy: 0.5030 - val_loss: 0.6928
Epoch 3/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.5335 - loss: 0.6961 - val_accuracy: 0.5100 - val_loss: 0.6909
Epoch 4/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.4752 - loss: 0.6935 - val_accuracy: 0.5210 - val_loss: 0.7086
Epoch 5/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.5362 - loss: 0.6952 - val_accuracy: 0.5610 - val_loss: 0.6730
Epoch 6/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.5007 - loss: 0.6870 - val_accuracy: 0.5490 - val_loss: 0.6938
Epoch 7/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5307 - loss: 0.6932 - val_accuracy: 0.5900 - val_loss: 0.6682
Epoch 8/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5900 - loss: 0.6760 - val_accuracy: 0.5740 - val_loss: 0.6678
Epoch 9/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5901 - loss: 0.6580 - val_accuracy: 0.6180 - val_loss: 0.6493
Epoch 10/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.5807 - loss: 0.6651 - val_accuracy: 0.6060 - val_loss: 0.6352
Epoch 11/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.5984 - loss: 0.6486 - val_accuracy: 0.5910 - val_loss: 0.6442
Epoch 12/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.6390 - loss: 0.6312 - val_accuracy: 0.6420 - val_loss: 0.6186
Epoch 13/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.6299 - loss: 0.6327 - val_accuracy: 0.6230 - val_loss: 0.6414
Epoch 14/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.6558 - loss: 0.6257 - val_accuracy: 0.5790 - val_loss: 0.6715
Epoch 15/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.6229 - loss: 0.6315 - val_accuracy: 0.6230 - val_loss: 0.6447
Epoch 16/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.6601 - loss: 0.6202 - val_accuracy: 0.6320 - val_loss: 0.6166
Epoch 17/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.6796 - loss: 0.6104 - val_accuracy: 0.5760 - val_loss: 0.6779
Epoch 18/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.6355 - loss: 0.6236 - val_accuracy: 0.6360 - val_loss: 0.6298
Epoch 19/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.6921 - loss: 0.5973 - val_accuracy: 0.6380 - val_loss: 0.6227
Epoch 20/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.6861 - loss: 0.5825 - val_accuracy: 0.6630 - val_loss: 0.5968
Epoch 21/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.7100 - loss: 0.5805 - val_accuracy: 0.6730 - val_loss: 0.5975
Epoch 22/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.6880 - loss: 0.5848 - val_accuracy: 0.7110 - val_loss: 0.5756
Epoch 23/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.6997 - loss: 0.5744 - val_accuracy: 0.6710 - val_loss: 0.6097
Epoch 24/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.7166 - loss: 0.5562 - val_accuracy: 0.6860 - val_loss: 0.5806
Epoch 25/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7092 - loss: 0.5548 - val_accuracy: 0.7070 - val_loss: 0.5681
Epoch 26/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7131 - loss: 0.5684 - val_accuracy: 0.6510 - val_loss: 0.6181
Epoch 27/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.7085 - loss: 0.5605 - val_accuracy: 0.7040 - val_loss: 0.5695
Epoch 28/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.7172 - loss: 0.5490 - val_accuracy: 0.6820 - val_loss: 0.5854
Epoch 29/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.7158 - loss: 0.5489 - val_accuracy: 0.6960 - val_lo
Epoch 30/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.7242 - loss: 0.5405 - val_accuracy: 0.7120 - val_l
Epoch 31/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7301 - loss: 0.5398 - val_accuracy: 0.7020 - val_loss: 0.5680
```

```
Epoch 32/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.7171 - loss: 0.5281 - val_accuracy: 0.7080 - val_loss: 0.5634
Epoch 33/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.7285 - loss: 0.5313 - val_accuracy: 0.7060 - val_loss: 0.5588
Epoch 34/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.7506 - loss: 0.5101 - val_accuracy: 0.7210 - val_loss: 0.5498
Epoch 35/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.7289 - loss: 0.5307 - val_accuracy: 0.7140 - val_loss: 0.5538
Epoch 36/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.7301 - loss: 0.5196 - val_accuracy: 0.7100 - val_loss: 0.5661
Epoch 37/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7278 - loss: 0.5271 - val_accuracy: 0.7110 - val_loss: 0.5492
Epoch 38/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7271 - loss: 0.5364 - val_accuracy: 0.7150 - val_loss: 0.5440
Epoch 39/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7463 - loss: 0.5062 - val_accuracy: 0.7270 - val_loss: 0.5322
Epoch 40/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.7439 - loss: 0.5196 - val_accuracy: 0.7350 - val_loss: 0.5354
Epoch 41/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.7543 - loss: 0.5100 - val_accuracy: 0.7160 - val_loss: 0.5434
Epoch 42/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.7511 - loss: 0.4992 - val_accuracy: 0.7300 - val_loss: 0.5305
Epoch 43/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.7469 - loss: 0.5183 - val_accuracy: 0.7190 - val_loss: 0.5480
Epoch 44/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.7530 - loss: 0.4913 - val_accuracy: 0.7250 - val_loss: 0.5289
Epoch 45/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.7454 - loss: 0.5157 - val_accuracy: 0.7220 - val_loss: 0.5364
Epoch 46/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.7400 - loss: 0.5006 - val_accuracy: 0.7340 - val_loss: 0.5240
Epoch 47/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.7563 - loss: 0.5042 - val_accuracy: 0.7230 - val_loss: 0.5346
Epoch 48/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.7497 - loss: 0.5055 - val_accuracy: 0.7190 - val_loss: 0.5323
Epoch 49/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.7410 - loss: 0.5020 - val_accuracy: 0.7360 - val_loss: 0.5157
Epoch 50/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.7538 - loss: 0.4856 - val_accuracy: 0.7210 - val_loss: 0.5320
```





Accuracy(model6)- With a test accuracy of 71.2% and a test loss of 0.564, Model 6 performs well. While there is a slight discrepancy between training and test accuracy, the model's training accuracy of 72.37% shows a strong learning curve. This implies that the model is probably doing a good job of generalizing without going overboard.

Additional optimization or modifications, like deeper structures or more regularization methods, could be investigated to improve Model 6's performance.

Model-7 A hybrid convolutional neural network (CNN) that combines data augmentation, MaxPooling, and Strides

```
import keras
from keras import layers
from keras.models import Model


data_augmentation = keras.Sequential([
    layers.RandomFlip('horizontal'),
    layers.RandomRotation(0.2),
    layers.RandomZoom(0.2),
])

inputs = keras.Input(shape=(180, 180, 3))

x = data_augmentation(inputs)

x = layers.Rescaling(1./255)(x)


x = layers.Conv2D(filters=32, kernel_size=3, strides=1, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)

x = layers.Conv2D(filters=64, kernel_size=3, strides=1, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)

x = layers.Conv2D(filters=128, kernel_size=3, strides=1, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)

x = layers.Conv2D(filters=256, kernel_size=3, strides=1, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)

x = layers.Conv2D(filters=256, kernel_size=3, strides=1, activation="relu")(x)

x = layers.Flatten()(x)
x = layers.Dropout(0.5)(x)

outputs = layers.Dense(1, activation="sigmoid")(x)

model = Model(inputs=inputs, outputs=outputs)

model.summary()

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

from keras.callbacks import ModelCheckpoint
callbacks = ModelCheckpoint(filepath="model_hybrid.keras", save_best_only=True, monitor="val_loss")

Model_Hybrid = model.fit(train_dataset, epochs=50, validation_data=validation_dataset, callbacks=callbacks)

# Evaluate the Model on the Test Set
best_model = keras.models.load_model("model_hybrid.keras")
Model_Hybrid_Results = best_model.evaluate(test_dataset)
print(f'Loss: {Model_Hybrid_Results[0]:.3f}')
print(f'Accuracy: {Model_Hybrid_Results[1]:.3f}')
```

Model: "functional_57"

Layer (type)	Output Shape	Param #
input_layer_18 (InputLayer)	(None, 180, 180, 3)	0
sequential_9 (Sequential)	(None, 180, 180, 3)	0
rescaling_7 (Rescaling)	(None, 180, 180, 3)	0
conv2d_56 (Conv2D)	(None, 178, 178, 32)	896
max_pooling2d_44 (MaxPooling2D)	(None, 89, 89, 32)	0
conv2d_57 (Conv2D)	(None, 87, 87, 64)	18,496
max_pooling2d_45 (MaxPooling2D)	(None, 43, 43, 64)	0
conv2d_58 (Conv2D)	(None, 41, 41, 128)	73,856
max_pooling2d_46 (MaxPooling2D)	(None, 20, 20, 128)	0
conv2d_59 (Conv2D)	(None, 18, 18, 256)	295,168
max_pooling2d_47 (MaxPooling2D)	(None, 9, 9, 256)	0
conv2d_60 (Conv2D)	(None, 7, 7, 256)	590,080
flatten_11 (Flatten)	(None, 12544)	0
dropout_10 (Dropout)	(None, 12544)	0
dense_15 (Dense)	(None, 1)	12,545

Total params: 991,041 (3.78 MB)
Trainable params: 991,041 (3.78 MB)
Non-trainable params: 0 (0.00 B)

```
Epoch 1/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 3s 19ms/step - accuracy: 0.4949 - loss: 0.6999 - val_accuracy: 0.5000 - val_loss: 0.6932
Epoch 2/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.5057 - loss: 0.6931 - val_accuracy: 0.5000 - val_loss: 0.6930
Epoch 3/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.4924 - loss: 0.6928 - val_accuracy: 0.5000 - val_loss: 0.6933
Epoch 4/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.4940 - loss: 0.6935 - val_accuracy: 0.5000 - val_loss: 0.6932
Epoch 5/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.5017 - loss: 0.6935 - val_accuracy: 0.5190 - val_loss: 0.6893
Epoch 6/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5172 - loss: 0.6951 - val_accuracy: 0.5030 - val_loss: 0.6931
Epoch 7/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5023 - loss: 0.6933 - val_accuracy: 0.5010 - val_loss: 0.6932
Epoch 8/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.4997 - loss: 0.6930 - val_accuracy: 0.5070 - val_loss: 0.6934
Epoch 9/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.5064 - loss: 0.6926 - val_accuracy: 0.4990 - val_loss: 0.7213
Epoch 10/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.5148 - loss: 0.6937 - val_accuracy: 0.5110 - val_loss: 0.6927
Epoch 11/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.5208 - loss: 0.6917 - val_accuracy: 0.5540 - val_loss: 0.6867
Epoch 12/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.5544 - loss: 0.6869 - val_accuracy: 0.5860 - val_loss: 0.6678
Epoch 13/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.6153 - loss: 0.6663 - val_accuracy: 0.5570 - val_loss: 0.6893
Epoch 14/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.6274 - loss: 0.6518 - val_accuracy: 0.5920 - val_loss: 0.6610
Epoch 15/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.6342 - loss: 0.6388 - val_accuracy: 0.6160 - val_loss: 0.6586
Epoch 16/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.6347 - loss: 0.6353 - val_accuracy: 0.6380 - val_loss: 0.6416
Epoch 17/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.6531 - loss: 0.6375 - val_accuracy: 0.6310 - val_loss: 0.6374
Epoch 18/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.6778 - loss: 0.6077 - val_accuracy: 0.6450 - val_loss: 0.6173
Epoch 19/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.6445 - loss: 0.6264 - val_accuracy: 0.6480 - val_loss: 0.6098
Epoch 20/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.6670 - loss: 0.6107 - val_accuracy: 0.6600 - val_loss: 0.6133
Epoch 21/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.6857 - loss: 0.5995 - val_accuracy: 0.6850 - val_loss: 0.5824
Epoch 22/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7024 - loss: 0.5859 - val_accuracy: 0.6690 - val_loss: 0.6013
Epoch 23/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.7110 - loss: 0.5783 - val_accuracy: 0.6940 - val_loss: 0.5797
Epoch 24/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.7264 - loss: 0.5512 - val_accuracy: 0.7060 - val_loss: 0.5606
Epoch 25/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.7237 - loss: 0.5480 - val_accuracy: 0.7160 - val_lo
Epoch 26/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7001 - loss: 0.5622 - val_accuracy: 0.5950 - val_l
Epoch 27/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.7119 - loss: 0.5646 - val_accuracy: 0.7150 - val_loss: 0.5528
```

```
Epoch 28/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.7200 - loss: 0.5392 - val_accuracy: 0.7090 - val_loss: 0.5434
Epoch 29/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.7290 - loss: 0.5371 - val_accuracy: 0.7440 - val_loss: 0.5130
Epoch 30/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7261 - loss: 0.5374 - val_accuracy: 0.7390 - val_loss: 0.5176
Epoch 31/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7336 - loss: 0.5208 - val_accuracy: 0.7250 - val_loss: 0.5368
Epoch 32/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7414 - loss: 0.5013 - val_accuracy: 0.6930 - val_loss: 0.5868
Epoch 33/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7519 - loss: 0.5220 - val_accuracy: 0.7440 - val_loss: 0.5336
Epoch 34/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.7753 - loss: 0.4790 - val_accuracy: 0.7790 - val_loss: 0.4734
Epoch 35/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7559 - loss: 0.4935 - val_accuracy: 0.7610 - val_loss: 0.4898
Epoch 36/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.7892 - loss: 0.4566 - val_accuracy: 0.7800 - val_loss: 0.4808
Epoch 37/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7939 - loss: 0.4689 - val_accuracy: 0.7540 - val_loss: 0.5088
Epoch 38/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.7632 - loss: 0.4706 - val_accuracy: 0.7790 - val_loss: 0.4721
Epoch 39/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7980 - loss: 0.4548 - val_accuracy: 0.7570 - val_loss: 0.4940
Epoch 40/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7850 - loss: 0.4689 - val_accuracy: 0.7460 - val_loss: 0.5171
Epoch 41/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7737 - loss: 0.4706 - val_accuracy: 0.7600 - val_loss: 0.4882
Epoch 42/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7926 - loss: 0.4427 - val_accuracy: 0.7830 - val_loss: 0.4749
Epoch 43/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7897 - loss: 0.4488 - val_accuracy: 0.7770 - val_loss: 0.4855
Epoch 44/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.7793 - loss: 0.4672 - val_accuracy: 0.7720 - val_loss: 0.4840
Epoch 45/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.8192 - loss: 0.4044 - val_accuracy: 0.7840 - val_loss: 0.4630
Epoch 46/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8318 - loss: 0.3933 - val_accuracy: 0.7780 - val_loss: 0.4732
Epoch 47/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.8230 - loss: 0.4087 - val_accuracy: 0.7970 - val_loss: 0.4408
Epoch 48/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.8111 - loss: 0.4178 - val_accuracy: 0.7290 - val_loss: 0.5293
Epoch 49/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.7767 - loss: 0.4394 - val_accuracy: 0.7680 - val_loss: 0.4749
Epoch 50/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8129 - loss: 0.4223 - val_accuracy: 0.7810 - val_loss: 0.4643
```

Accuracy(model7)- Training Synopsis:

To make the training dataset more diverse, this model employs data augmentation techniques like random flip, rotation, and zoom.

Important features are extracted from the photos using MaxPooling layers and convolutional layers with different filter sizes (32, 64, 128, 256, 256).

Convolutional layers use strides to avoid overfitting and maintain a suitable resolution.

During training, dropout is used to lessen overfitting.

For problems involving binary classification, the output layer's sigmoid activation is perfect.

Test Results:

The model performed well in classifying the test dataset, as seen by its 79.0% test accuracy.

Model-8

```
Model_8 = model.fit(
    train_dataset,
    epochs=50,
    validation_data=validation_dataset,
    callbacks=callbacks
)

import matplotlib.pyplot as plt

accuracy = Model_8.history["accuracy"]
val_accuracy = Model_8.history["val_accuracy"]
loss = Model_8.history["loss"]
val_loss = Model_8.history["val_loss"]

epochs = range(1, len(accuracy) + 1)

plt.plot(epochs, accuracy, color="grey", label="Training Accuracy")
plt.plot(epochs, val_accuracy, color="blue", linestyle="dashed", label="Validation Accuracy")
plt.title("Training and Validation Accuracy")
plt.legend()
plt.figure()

plt.plot(epochs, loss, color="grey", label="Training Loss")
plt.plot(epochs, val_loss, color="blue", linestyle="dashed", label="Validation Loss")
plt.title("Training and Validation Loss")
plt.legend()
plt.show()
```

```
Epoch 1/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.8182 - loss: 0.3952 - val_accuracy: 0.8020 - val_loss: 0.4301
Epoch 2/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8139 - loss: 0.4046 - val_accuracy: 0.7930 - val_loss: 0.4732
Epoch 3/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8149 - loss: 0.4024 - val_accuracy: 0.7970 - val_loss: 0.4561
Epoch 4/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8209 - loss: 0.3892 - val_accuracy: 0.7990 - val_loss: 0.4569
Epoch 5/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.8182 - loss: 0.3939 - val_accuracy: 0.8030 - val_loss: 0.4175
Epoch 6/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8287 - loss: 0.3878 - val_accuracy: 0.7950 - val_loss: 0.4825
Epoch 7/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.8383 - loss: 0.3718 - val_accuracy: 0.7980 - val_loss: 0.4367
Epoch 8/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.8369 - loss: 0.3608 - val_accuracy: 0.8020 - val_loss: 0.4350
Epoch 9/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.8310 - loss: 0.3748 - val_accuracy: 0.7990 - val_loss: 0.4369
Epoch 10/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8454 - loss: 0.3570 - val_accuracy: 0.8130 - val_loss: 0.4437
Epoch 11/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8488 - loss: 0.3446 - val_accuracy: 0.8010 - val_loss: 0.4745
Epoch 12/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8395 - loss: 0.3627 - val_accuracy: 0.7730 - val_loss: 0.5396
Epoch 13/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8292 - loss: 0.3782 - val_accuracy: 0.8180 - val_loss: 0.4244
Epoch 14/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8478 - loss: 0.3300 - val_accuracy: 0.8140 - val_loss: 0.4367
Epoch 15/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.8281 - loss: 0.3574 - val_accuracy: 0.8070 - val_loss: 0.4114
Epoch 16/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.8722 - loss: 0.3316 - val_accuracy: 0.8160 - val_loss: 0.3928
Epoch 17/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8598 - loss: 0.3101 - val_accuracy: 0.8250 - val_loss: 0.3946
Epoch 18/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.8748 - loss: 0.2992 - val_accuracy: 0.8100 - val_loss: 0.4276
Epoch 19/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8592 - loss: 0.3078 - val_accuracy: 0.8170 - val_loss: 0.4144
Epoch 20/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8662 - loss: 0.3156 - val_accuracy: 0.8090 - val_loss: 0.4413
Epoch 21/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.8738 - loss: 0.3045 - val_accuracy: 0.8190 - val_loss: 0.4211
Epoch 22/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.8639 - loss: 0.2986 - val_accuracy: 0.8200 - val_loss: 0.4183
Epoch 23/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.8595 - loss: 0.3047 - val_accuracy: 0.8030 - val_loss: 0.4556
Epoch 24/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8645 - loss: 0.3048 - val_accuracy: 0.8260 - val_loss: 0.3999
Epoch 25/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8622 - loss: 0.3041 - val_accuracy: 0.8080 - val_loss: 0.4528
Epoch 26/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.8711 - loss: 0.3025 - val_accuracy: 0.8270 - val_loss: 0.3763
Epoch 27/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8787 - loss: 0.2788 - val_accuracy: 0.8060 - val_loss: 0.4741
Epoch 28/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8708 - loss: 0.3021 - val_accuracy: 0.8240 - val_loss: 0.4175
Epoch 29/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8779 - loss: 0.2748 - val_accuracy: 0.8210 - val_loss: 0.4367
Epoch 30/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8742 - loss: 0.2974 - val_accuracy: 0.8370 - val_loss: 0.4465
Epoch 31/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8683 - loss: 0.3071 - val_accuracy: 0.8300 - val_loss: 0.4256
Epoch 32/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8743 - loss: 0.2939 - val_accuracy: 0.8350 - val_loss: 0.4053
Epoch 33/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.8849 - loss: 0.2694 - val_accuracy: 0.8480 - val_loss: 0.3742
Epoch 34/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.8725 - loss: 0.2978 - val_accuracy: 0.8250 - val_loss: 0.4439
Epoch 35/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.8922 - loss: 0.2595 - val_accuracy: 0.8270 - val_loss: 0.4765
Epoch 36/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8776 - loss: 0.2882 - val_accuracy: 0.8340 - val_loss: 0.4720
Epoch 37/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8917 - loss: 0.2463 - val_accuracy: 0.8460 - val_loss: 0.4083
Epoch 38/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8843 - loss: 0.2790 - val_accuracy: 0.8460 - val_loss: 0.4220
Epoch 39/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8759 - loss: 0.2785 - val_accuracy: 0.8430 - val_loss: 0.4100
Epoch 40/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8955 - loss: 0.2537 - val_accuracy: 0.8320 - val_loss: 0.4289
Epoch 41/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8995 - loss: 0.2318 - val_accuracy: 0.8390 - val_loss: 0.3947
Epoch 42/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8991 - loss: 0.2543 - val_accuracy: 0.8490 - val_loss: 0.4197
Epoch 43/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8888 - loss: 0.2642 - val_accuracy: 0.8220 - val_los
Epoch 44/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.9088 - loss: 0.2512 - val_accuracy: 0.8390 - val_l
Epoch 45/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8969 - loss: 0.2356 - val_accuracy: 0.8290 - val_loss: 0.4448
```

```
Epoch 46/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.9211 - loss: 0.2030 - val_accuracy: 0.8200 - val_loss: 0.4705
Epoch 47/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8974 - loss: 0.2390 - val_accuracy: 0.8170 - val_loss: 0.4718
Epoch 48/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.9056 - loss: 0.2302 - val_accuracy: 0.8560 - val_loss: 0.4453
Epoch 49/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8962 - loss: 0.2237 - val_accuracy: 0.8530 - val_loss: 0.4300
Epoch 50/50
63/63 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8887 - loss: 0.2553 - val_accuracy: 0.8330 - val_loss: 0.4609
```