afoegovsc

April 4, 2024

ASSIGNMENT - 1

CONVOLUTION

```
[ ]: GROUP-13 (PHANI VARSHITHA, DURGA CHOWDARY)
```

```
[ ]: Training from scratch model
```

```
[1]: # Creating directories and assiging images to training, validation and test␣
      ↪directories
     import os, shutil, pathlib

     directory = pathlib.Path("C:/Users/varshitha/Downloads/dogs-vs-cats/train/
      ↪train")
     small_directory = pathlib.Path("C:/Users/varshitha/Downloads/dogs-vs-cats/train/
      ↪small")

     def make_subset(subset_name, start_index, end_index):
         for category in ("cat", "dog"):
             dir = small_directory / subset_name / category
             os.makedirs(dir)
             fnames = [f"{category}.{i}.jpg" for i in range(start_index, end_index)]
             for fname in fnames:
                 shutil.copyfile(src=directory / fname,
                                 dst= dir / fname)

     make_subset("train", start_index=0, end_index=1000)
     make_subset("validation", start_index=1000, end_index=1500)
     make_subset("test", start_index=1500, end_index=2500)
```

```
---------------------------------------------------------------------------
FileExistsError                           Traceback (most recent call last)
Cell In[1], line 16
     12         for fname in fnames:
     13             shutil.copyfile(src=directory / fname,
     14                             dst= dir / fname)
---> 16 make_subset("train", start_index=0, end_index=1000)
     17 make_subset("validation", start_index=1000, end_index=1500)
     18 make_subset("test", start_index=1500, end_index=2500)

Cell In[1], line 10, in make_subset(subset_name, start_index, end_index)
      8 for category in ("cat", "dog"):
      9     dir = small_directory / subset_name / category
---> 10     os.makedirs(dir)
     11     fnames = [f"{category}.{i}.jpg" for i in range(start_index,␣
  ↪end_index)]
     12     for fname in fnames:

File <frozen os>:225, in makedirs(name, mode, exist_ok)

FileExistsError: [WinError 183] Cannot create a file when that file already␣
  ↪exists: 'C:
  ↪\\Users\\varshitha\\Downloads\\dogs-vs-cats\\train\\small\\train\\cat'
```

[2]:
```python
#Building the model and running the model summary
from tensorflow import keras
from tensorflow.keras import layers

inputs = keras.Input(shape=(180, 180, 3))
x = layers.Rescaling(1./255)(inputs)
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)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

model.summary()
```

```
Model: "functional_1"


 Layer (type)                         Output Shape                          ␣
↪Param #

 input_layer (InputLayer)             (None, 180, 180, 3)                       ␣
↪   0
```

```
rescaling (Rescaling)                   (None, 180, 180, 3)                   ␣
↪  0

conv2d (Conv2D)                         (None, 178, 178, 32)                  ␣
↪896

max_pooling2d (MaxPooling2D)            (None, 89, 89, 32)                    ␣
↪  0

conv2d_1 (Conv2D)                       (None, 87, 87, 64)                  ␣
↪18,496

max_pooling2d_1 (MaxPooling2D)          (None, 43, 43, 64)                    ␣
↪  0

conv2d_2 (Conv2D)                       (None, 41, 41, 128)                 ␣
↪73,856

max_pooling2d_2 (MaxPooling2D)          (None, 20, 20, 128)                   ␣
↪  0

conv2d_3 (Conv2D)                       (None, 18, 18, 256)                ␣
↪295,168

max_pooling2d_3 (MaxPooling2D)          (None, 9, 9, 256)                     ␣
↪  0

conv2d_4 (Conv2D)                       (None, 7, 7, 256)                  ␣
↪590,080

flatten (Flatten)                       (None, 12544)                         ␣
↪  0

dense (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)
```

```
[3]: # Configuration of the model
model.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

```
[4]: # Declaring the image size and batch size to read the images from train.␣
 ↪validation and test directories
from tensorflow.keras.utils import image_dataset_from_directory

train_dataset = image_dataset_from_directory(
    small_directory / "train",
    image_size=(180, 180),
    batch_size=32)
validation_dataset = image_dataset_from_directory(
    small_directory / "validation",
    image_size=(180, 180),
    batch_size=32)
test_dataset = image_dataset_from_directory(
    small_directory / "test",
    image_size=(180, 180),
    batch_size=32)
```

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

```
[5]: # Using the callbacks function to monitor validation loss and running the model
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=30,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/30
63/63                 81s 1s/step -
accuracy: 0.5002 - loss: 0.7058 - val_accuracy: 0.5990 - val_loss: 0.6926
Epoch 2/30
63/63                 58s 916ms/step -
accuracy: 0.5165 - loss: 0.6933 - val_accuracy: 0.5130 - val_loss: 0.6886
Epoch 3/30
63/63                 57s 909ms/step -
```

```
accuracy: 0.5520 - loss: 0.6898 - val_accuracy: 0.5490 - val_loss: 0.6627
Epoch 4/30
63/63            56s 886ms/step -
accuracy: 0.5926 - loss: 0.6731 - val_accuracy: 0.6610 - val_loss: 0.6203
Epoch 5/30
63/63            56s 894ms/step -
accuracy: 0.6448 - loss: 0.6277 - val_accuracy: 0.6930 - val_loss: 0.6049
Epoch 6/30
63/63            58s 923ms/step -
accuracy: 0.6929 - loss: 0.5952 - val_accuracy: 0.7040 - val_loss: 0.5664
Epoch 7/30
63/63            57s 898ms/step -
accuracy: 0.6924 - loss: 0.5744 - val_accuracy: 0.6850 - val_loss: 0.5952
Epoch 8/30
63/63            55s 871ms/step -
accuracy: 0.7239 - loss: 0.5527 - val_accuracy: 0.6930 - val_loss: 0.5870
Epoch 9/30
63/63            55s 877ms/step -
accuracy: 0.7412 - loss: 0.5175 - val_accuracy: 0.7300 - val_loss: 0.5400
Epoch 10/30
63/63            56s 890ms/step -
accuracy: 0.7771 - loss: 0.4801 - val_accuracy: 0.7340 - val_loss: 0.5512
Epoch 11/30
63/63            56s 889ms/step -
accuracy: 0.7913 - loss: 0.4400 - val_accuracy: 0.7090 - val_loss: 0.5850
Epoch 12/30
63/63            56s 882ms/step -
accuracy: 0.8171 - loss: 0.4112 - val_accuracy: 0.7550 - val_loss: 0.5626
Epoch 13/30
63/63            56s 891ms/step -
accuracy: 0.8616 - loss: 0.3263 - val_accuracy: 0.7270 - val_loss: 0.6649
Epoch 14/30
63/63            57s 899ms/step -
accuracy: 0.8799 - loss: 0.3065 - val_accuracy: 0.7400 - val_loss: 0.8033
Epoch 15/30
63/63            470s 8s/step -
accuracy: 0.8910 - loss: 0.2618 - val_accuracy: 0.7420 - val_loss: 0.8554
Epoch 16/30
63/63            56s 891ms/step -
accuracy: 0.9165 - loss: 0.2079 - val_accuracy: 0.7640 - val_loss: 0.7255
Epoch 17/30
63/63            56s 896ms/step -
accuracy: 0.9534 - loss: 0.1371 - val_accuracy: 0.7550 - val_loss: 0.8336
Epoch 18/30
63/63            56s 890ms/step -
accuracy: 0.9568 - loss: 0.1108 - val_accuracy: 0.7380 - val_loss: 0.8604
Epoch 19/30
63/63            56s 890ms/step -
```

```
accuracy: 0.9645 - loss: 0.0807 - val_accuracy: 0.7480 - val_loss: 0.9386
Epoch 20/30
63/63               56s 895ms/step -
accuracy: 0.9747 - loss: 0.0646 - val_accuracy: 0.7560 - val_loss: 1.0551
Epoch 21/30
63/63               56s 893ms/step -
accuracy: 0.9744 - loss: 0.0680 - val_accuracy: 0.7550 - val_loss: 1.0564
Epoch 22/30
63/63               56s 894ms/step -
accuracy: 0.9869 - loss: 0.0565 - val_accuracy: 0.7660 - val_loss: 1.0919
Epoch 23/30
63/63               56s 889ms/step -
accuracy: 0.9874 - loss: 0.0452 - val_accuracy: 0.7650 - val_loss: 1.2689
Epoch 24/30
63/63               56s 892ms/step -
accuracy: 0.9836 - loss: 0.0684 - val_accuracy: 0.7520 - val_loss: 1.2833
Epoch 25/30
63/63               56s 893ms/step -
accuracy: 0.9880 - loss: 0.0290 - val_accuracy: 0.7630 - val_loss: 1.4795
Epoch 26/30
63/63               56s 894ms/step -
accuracy: 0.9856 - loss: 0.0456 - val_accuracy: 0.7470 - val_loss: 1.5276
Epoch 27/30
63/63               56s 891ms/step -
accuracy: 0.9912 - loss: 0.0203 - val_accuracy: 0.6960 - val_loss: 2.7429
Epoch 28/30
63/63               57s 897ms/step -
accuracy: 0.9730 - loss: 0.0811 - val_accuracy: 0.7390 - val_loss: 1.9263
Epoch 29/30
63/63               56s 895ms/step -
accuracy: 0.9899 - loss: 0.0212 - val_accuracy: 0.7560 - val_loss: 1.6912
Epoch 30/30
63/63               56s 890ms/step -
accuracy: 0.9876 - loss: 0.0341 - val_accuracy: 0.7660 - val_loss: 1.8089
```

[6]:
```
# Testing the model
model.evaluate(test_dataset)
```

```
63/63               18s 272ms/step -
accuracy: 0.7235 - loss: 2.1129
```

[6]:
```
[2.0441811084747314, 0.7360000014305115]
```

[7]:
```
# Declaring Data Augumentation
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)
```

[8]:
```
# Building the model and configuing it
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.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

[9]:
```
# Using the callbacks function to monitor validation loss and running the model
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_with_augmentation.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=60,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/60
63/63                  63s 942ms/step -
accuracy: 0.4804 - loss: 0.7004 - val_accuracy: 0.5000 - val_loss: 0.6927
Epoch 2/60
63/63                  57s 897ms/step -
accuracy: 0.4926 - loss: 0.6944 - val_accuracy: 0.5000 - val_loss: 0.6963
Epoch 3/60
63/63                  71s 1s/step -
```

```
accuracy: 0.5115 - loss: 0.6943 - val_accuracy: 0.5400 - val_loss: 0.6903
Epoch 4/60
63/63            62s 955ms/step -
accuracy: 0.5329 - loss: 0.6927 - val_accuracy: 0.6380 - val_loss: 0.6724
Epoch 5/60
63/63            59s 938ms/step -
accuracy: 0.6021 - loss: 0.7005 - val_accuracy: 0.5480 - val_loss: 0.6696
Epoch 6/60
63/63            58s 921ms/step -
accuracy: 0.6351 - loss: 0.6420 - val_accuracy: 0.6340 - val_loss: 0.6398
Epoch 7/60
63/63            57s 909ms/step -
accuracy: 0.6340 - loss: 0.6425 - val_accuracy: 0.5760 - val_loss: 0.6966
Epoch 8/60
63/63            164s 3s/step -
accuracy: 0.6476 - loss: 0.6195 - val_accuracy: 0.6270 - val_loss: 0.6296
Epoch 9/60
63/63            179s 3s/step -
accuracy: 0.6665 - loss: 0.6153 - val_accuracy: 0.6840 - val_loss: 0.5875
Epoch 10/60
63/63            219s 3s/step -
accuracy: 0.6965 - loss: 0.5873 - val_accuracy: 0.6670 - val_loss: 0.5999
Epoch 11/60
63/63            190s 3s/step -
accuracy: 0.7099 - loss: 0.5888 - val_accuracy: 0.6070 - val_loss: 0.6398
Epoch 12/60
63/63            200s 3s/step -
accuracy: 0.6854 - loss: 0.5787 - val_accuracy: 0.7130 - val_loss: 0.5644
Epoch 13/60
63/63            204s 3s/step -
accuracy: 0.7188 - loss: 0.5686 - val_accuracy: 0.6990 - val_loss: 0.5782
Epoch 14/60
63/63            213s 3s/step -
accuracy: 0.7204 - loss: 0.5524 - val_accuracy: 0.6640 - val_loss: 0.6425
Epoch 15/60
63/63            234s 3s/step -
accuracy: 0.7289 - loss: 0.5645 - val_accuracy: 0.7370 - val_loss: 0.5365
Epoch 16/60
63/63            199s 3s/step -
accuracy: 0.7300 - loss: 0.5385 - val_accuracy: 0.6810 - val_loss: 0.6535
Epoch 17/60
63/63            195s 3s/step -
accuracy: 0.7470 - loss: 0.5236 - val_accuracy: 0.7420 - val_loss: 0.5129
Epoch 18/60
63/63            189s 3s/step -
accuracy: 0.7638 - loss: 0.5047 - val_accuracy: 0.7420 - val_loss: 0.5405
Epoch 19/60
63/63            197s 3s/step -
```

```
accuracy: 0.7659 - loss: 0.4946 - val_accuracy: 0.7060 - val_loss: 0.6174
Epoch 20/60
63/63            167s 3s/step -
accuracy: 0.7561 - loss: 0.5125 - val_accuracy: 0.7060 - val_loss: 0.5646
Epoch 21/60
63/63            170s 3s/step -
accuracy: 0.7857 - loss: 0.5083 - val_accuracy: 0.7190 - val_loss: 0.5310
Epoch 22/60
63/63            174s 3s/step -
accuracy: 0.7654 - loss: 0.4956 - val_accuracy: 0.7280 - val_loss: 0.5402
Epoch 23/60
63/63            169s 3s/step -
accuracy: 0.7655 - loss: 0.4804 - val_accuracy: 0.7950 - val_loss: 0.4669
Epoch 24/60
63/63            236s 3s/step -
accuracy: 0.7902 - loss: 0.4446 - val_accuracy: 0.7890 - val_loss: 0.4665
Epoch 25/60
63/63            168s 3s/step -
accuracy: 0.7901 - loss: 0.4448 - val_accuracy: 0.7680 - val_loss: 0.4742
Epoch 26/60
63/63            205s 3s/step -
accuracy: 0.8030 - loss: 0.4458 - val_accuracy: 0.7930 - val_loss: 0.4646
Epoch 27/60
63/63            171s 3s/step -
accuracy: 0.7904 - loss: 0.4375 - val_accuracy: 0.6740 - val_loss: 0.7323
Epoch 28/60
63/63            200s 3s/step -
accuracy: 0.8112 - loss: 0.4096 - val_accuracy: 0.7840 - val_loss: 0.4775
Epoch 29/60
63/63            172s 3s/step -
accuracy: 0.8209 - loss: 0.3876 - val_accuracy: 0.7960 - val_loss: 0.4577
Epoch 30/60
63/63            189s 3s/step -
accuracy: 0.8118 - loss: 0.4145 - val_accuracy: 0.7360 - val_loss: 0.5865
Epoch 31/60
63/63            201s 3s/step -
accuracy: 0.8033 - loss: 0.4146 - val_accuracy: 0.7860 - val_loss: 0.4503
Epoch 32/60
63/63            183s 3s/step -
accuracy: 0.8251 - loss: 0.4164 - val_accuracy: 0.8130 - val_loss: 0.4192
Epoch 33/60
63/63            171s 3s/step -
accuracy: 0.8177 - loss: 0.4075 - val_accuracy: 0.8040 - val_loss: 0.4415
Epoch 34/60
63/63            220s 3s/step -
accuracy: 0.8314 - loss: 0.3659 - val_accuracy: 0.7890 - val_loss: 0.4775
Epoch 35/60
63/63            186s 3s/step -
```

```
accuracy: 0.8358 - loss: 0.3601 - val_accuracy: 0.8170 - val_loss: 0.4768
Epoch 36/60
63/63            188s 3s/step -
accuracy: 0.8372 - loss: 0.3436 - val_accuracy: 0.8090 - val_loss: 0.4365
Epoch 37/60
63/63            180s 3s/step -
accuracy: 0.8565 - loss: 0.3399 - val_accuracy: 0.7880 - val_loss: 0.5268
Epoch 38/60
63/63            193s 3s/step -
accuracy: 0.8480 - loss: 0.3441 - val_accuracy: 0.8470 - val_loss: 0.3858
Epoch 39/60
63/63            189s 3s/step -
accuracy: 0.8496 - loss: 0.3377 - val_accuracy: 0.8010 - val_loss: 0.4915
Epoch 40/60
63/63            169s 3s/step -
accuracy: 0.8604 - loss: 0.3385 - val_accuracy: 0.8090 - val_loss: 0.4505
Epoch 41/60
63/63            182s 3s/step -
accuracy: 0.8681 - loss: 0.2986 - val_accuracy: 0.7940 - val_loss: 0.4880
Epoch 42/60
63/63            176s 3s/step -
accuracy: 0.8685 - loss: 0.3160 - val_accuracy: 0.8300 - val_loss: 0.4454
Epoch 43/60
63/63            214s 3s/step -
accuracy: 0.8883 - loss: 0.2917 - val_accuracy: 0.8130 - val_loss: 0.4780
Epoch 44/60
63/63            124s 2s/step -
accuracy: 0.8663 - loss: 0.3035 - val_accuracy: 0.7820 - val_loss: 0.6767
Epoch 45/60
63/63            57s 899ms/step -
accuracy: 0.8743 - loss: 0.3097 - val_accuracy: 0.8520 - val_loss: 0.4004
Epoch 46/60
63/63            59s 943ms/step -
accuracy: 0.8921 - loss: 0.2842 - val_accuracy: 0.8470 - val_loss: 0.4384
Epoch 47/60
63/63            57s 909ms/step -
accuracy: 0.8826 - loss: 0.2939 - val_accuracy: 0.8290 - val_loss: 0.7008
Epoch 48/60
63/63            59s 940ms/step -
accuracy: 0.8744 - loss: 0.2991 - val_accuracy: 0.8280 - val_loss: 0.4454
Epoch 49/60
63/63            62s 986ms/step -
accuracy: 0.8857 - loss: 0.2634 - val_accuracy: 0.8500 - val_loss: 0.4342
Epoch 50/60
63/63            57s 909ms/step -
accuracy: 0.8836 - loss: 0.2762 - val_accuracy: 0.8320 - val_loss: 0.5525
Epoch 51/60
63/63            56s 891ms/step -
```

```
accuracy: 0.9002 - loss: 0.2703 - val_accuracy: 0.8220 - val_loss: 0.5321
Epoch 52/60
63/63              56s 890ms/step -
accuracy: 0.8685 - loss: 0.2824 - val_accuracy: 0.8200 - val_loss: 0.6358
Epoch 53/60
63/63              56s 886ms/step -
accuracy: 0.8815 - loss: 0.3173 - val_accuracy: 0.8350 - val_loss: 0.5284
Epoch 54/60
63/63              57s 906ms/step -
accuracy: 0.8994 - loss: 0.2394 - val_accuracy: 0.8500 - val_loss: 0.4843
Epoch 55/60
63/63              61s 971ms/step -
accuracy: 0.8941 - loss: 0.2527 - val_accuracy: 0.8500 - val_loss: 0.4278
Epoch 56/60
63/63              56s 890ms/step -
accuracy: 0.8957 - loss: 0.2528 - val_accuracy: 0.8560 - val_loss: 0.4838
Epoch 57/60
63/63              59s 931ms/step -
accuracy: 0.9065 - loss: 0.2169 - val_accuracy: 0.8340 - val_loss: 0.5528
Epoch 58/60
63/63              56s 883ms/step -
accuracy: 0.9040 - loss: 0.2497 - val_accuracy: 0.8500 - val_loss: 0.5641
Epoch 59/60
63/63              59s 930ms/step -
accuracy: 0.9050 - loss: 0.2441 - val_accuracy: 0.8230 - val_loss: 0.5185
Epoch 60/60
63/63              0s 3s/step -
accuracy: 0.9136 - loss: 0.2195
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[9], line 8
      1 # Using the callbacks function to monitor validation loss and running␣
 ↪the model
      2 callbacks = [
      3     keras.callbacks.ModelCheckpoint(
      4         filepath="convnet_from_scratch_with_augmentation.keras",
      5         save_best_only=True,
      6         monitor="val_loss")
      7 ]
----> 8 history = model.fit(
      9     train_dataset,
     10     epochs=60,
     11     validation_data=validation_dataset,
     12     callbacks=callbacks)
```

```
File ~\anaconda3\Lib\site-packages\keras\src\utils\traceback_utils.py:117, in␣
 ↪filter_traceback.<locals>.error_handler(*args, **kwargs)
    115 filtered_tb = None
    116 try:
--> 117     return fn(*args, **kwargs)
    118 except Exception as e:
    119     filtered_tb = _process_traceback_frames(e.__traceback__)

File ~\anaconda3\Lib\site-packages\keras\src\backend\tensorflow\trainer.py:351,␣
 ↪in TensorFlowTrainer.fit(self, x, y, batch_size, epochs, verbose, callbacks,␣
 ↪validation_split, validation_data, shuffle, class_weight, sample_weight,␣
 ↪initial_epoch, steps_per_epoch, validation_steps, validation_batch_size,␣
 ↪validation_freq)
    340 if getattr(self, "_eval_epoch_iterator", None) is None:
    341     self._eval_epoch_iterator = TFEpochIterator(
    342         x=val_x,
    343         y=val_y,
   (…)
    349         shuffle=False,
    350     )
--> 351 val_logs = self.evaluate(
    352     x=val_x,
    353     y=val_y,
    354     sample_weight=val_sample_weight,
    355     batch_size=validation_batch_size or batch_size,
    356     steps=validation_steps,
    357     callbacks=callbacks,
    358     return_dict=True,
    359     _use_cached_eval_dataset=True,
    360 )
    361 val_logs = {
    362     "val_" + name: val for name, val in val_logs.items()
    363 }
    364 epoch_logs.update(val_logs)

File ~\anaconda3\Lib\site-packages\keras\src\utils\traceback_utils.py:117, in␣
 ↪filter_traceback.<locals>.error_handler(*args, **kwargs)
    115 filtered_tb = None
    116 try:
--> 117     return fn(*args, **kwargs)
    118 except Exception as e:
    119     filtered_tb = _process_traceback_frames(e.__traceback__)

File ~\anaconda3\Lib\site-packages\keras\src\backend\tensorflow\trainer.py:437,␣
 ↪in TensorFlowTrainer.evaluate(self, x, y, batch_size, verbose, sample_weight,␣
 ↪steps, callbacks, return_dict, **kwargs)
    435 for step, iterator in epoch_iterator.enumerate_epoch():
    436     callbacks.on_test_batch_begin(step)
--> 437     logs = self.test_function(iterator)
```

```
    438     callbacks.on_test_batch_end(step, self._pythonify_logs(logs))
    439     if self.stop_evaluating:

File ~\anaconda3\Lib\site-packages\tensorflow\python\util\traceback_utils.py:
 ↪150, in filter_traceback.<locals>.error_handler(*args, **kwargs)
    148 filtered_tb = None
    149 try:
--> 150   return fn(*args, **kwargs)
    151 except Exception as e:
    152   filtered_tb = _process_traceback_frames(e.__traceback__)

File␣
 ↪~\anaconda3\Lib\site-packages\tensorflow\python\eager\polymorphic_function\polymorphic_func
 ↪py:833, in Function.__call__(self, *args, **kwds)
    830 compiler = "xla" if self._jit_compile else "nonXla"
    832 with OptionalXlaContext(self._jit_compile):
--> 833   result = self._call(*args, **kwds)
    835 new_tracing_count = self.experimental_get_tracing_count()
    836 without_tracing = (tracing_count == new_tracing_count)

File␣
 ↪~\anaconda3\Lib\site-packages\tensorflow\python\eager\polymorphic_function\polymorphic_func
 ↪py:878, in Function._call(self, *args, **kwds)
    875 self._lock.release()
    876 # In this case we have not created variables on the first call. So we can
    877 # run the first trace but we should fail if variables are created.
--> 878 results = tracing_compilation.call_function(
    879     args, kwds, self._variable_creation_config
    880 )
    881 if self._created_variables:
    882   raise ValueError("Creating variables on a non-first call to a function"
    883                    " decorated with tf.function.")

File␣
 ↪~\anaconda3\Lib\site-packages\tensorflow\python\eager\polymorphic_function\tracing_compilat
 ↪py:139, in call_function(args, kwargs, tracing_options)
    137 bound_args = function.function_type.bind(*args, **kwargs)
    138 flat_inputs = function.function_type.unpack_inputs(bound_args)
--> 139 return function._call_flat(  # pylint: disable=protected-access
    140     flat_inputs, captured_inputs=function.captured_inputs
    141 )

File␣
 ↪~\anaconda3\Lib\site-packages\tensorflow\python\eager\polymorphic_function\concrete_functio
 ↪py:1322, in ConcreteFunction._call_flat(self, tensor_inputs, captured_inputs)
   1318 possible_gradient_type = gradients_util.PossibleTapeGradientTypes(args)
   1319 if (possible_gradient_type == gradients_util.POSSIBLE_GRADIENT_TYPES_NONE
   1320     and executing_eagerly):
   1321   # No tape is watching; skip to running the function.
```

```
-> 1322   return self._inference_function.call_preflattened(args)
   1323 forward_backward = self._select_forward_and_backward_functions(
   1324     args,
   1325     possible_gradient_type,
   1326     executing_eagerly)
   1327 forward_function, args_with_tangents = forward_backward.forward()

File␣
 ↪~\anaconda3\Lib\site-packages\tensorflow\python\eager\polymorphic_function\at mic_function
 ↪py:216, in AtomicFunction.call_preflattened(self, args)
    214 def call_preflattened(self, args: Sequence[core.Tensor]) -> Any:
    215   """Calls with flattened tensor inputs and returns the structured␣
 ↪output."""
--> 216   flat_outputs = self.call_flat(*args)
    217   return self.function_type.pack_output(flat_outputs)

File␣
 ↪~\anaconda3\Lib\site-packages\tensorflow\python\eager\polymorphic_function\at mic_function
 ↪py:251, in AtomicFunction.call_flat(self, *args)
    249 with record.stop_recording():
    250   if self._bound_context.executing_eagerly():
--> 251     outputs = self._bound_context.call_function(
    252         self.name,
    253         list(args),
    254         len(self.function_type.flat_outputs),
    255     )
    256   else:
    257     outputs = make_call_op_in_graph(
    258         self,
    259         list(args),
    260         self._bound_context.function_call_options.as_attrs(),
    261     )

File ~\anaconda3\Lib\site-packages\tensorflow\python\eager\context.py:1500, in␣
 ↪Context.call_function(self, name, tensor_inputs, num_outputs)
   1498 cancellation_context = cancellation.context()
   1499 if cancellation_context is None:
-> 1500   outputs = execute.execute(
   1501       name.decode("utf-8"),
   1502       num_outputs=num_outputs,
   1503       inputs=tensor_inputs,
   1504       attrs=attrs,
   1505       ctx=self,
   1506   )
   1507 else:
   1508   outputs = execute.execute_with_cancellation(
   1509       name.decode("utf-8"),
   1510       num_outputs=num_outputs,
   (…)
   1514       cancellation_manager=cancellation_context,
   1515   )

File ~\anaconda3\Lib\site-packages\tensorflow\python\eager\execute.py:53, in␣
 ↪quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
     51 try:
     52   ctx.ensure_initialized()
---> 53   tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name
     54                                       inputs, attrs, num_outputs)
     55 except core._NotOkStatusException as e:
     56   if name is not None:

KeyboardInterrupt:
```

[11]:
```python
test_model = keras.models.load_model(
    "convnet_from_scratch_with_augmentation.keras")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
63/63                47s 717ms/step -
accuracy: 0.8393 - loss: 0.3790
Test accuracy: 0.827
```

Question 2

[12]:
```python
# Training has 2000 samples, test has 1000 samples, and validation has 500␣
 ↪samples
make_subset("train2", start_index=0, end_index=2000)
make_subset("validation2", start_index=2000, end_index=2500)
make_subset("test2", start_index=2500, end_index=3000)
```

[13]:
```python
train_dataset = image_dataset_from_directory(
    small_directory / "train2",
    image_size=(180, 180),
    batch_size=32)
validation_dataset = image_dataset_from_directory(
    small_directory / "validation2",
    image_size=(180, 180),
    batch_size=32)
test_dataset = image_dataset_from_directory(
    small_directory / "test2",
    image_size=(180, 180),
    batch_size=32)
```

```
Found 4000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
```

[14]:
```
# Configuring the model
model.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

[15]:
```
# Using the callbacks function to monitor validation loss and running the model
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=30,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/30
125/125             290s 2s/step -
accuracy: 0.8485 - loss: 0.3906 - val_accuracy: 0.6920 - val_loss: 0.9860
Epoch 2/30
125/125             107s 855ms/step -
accuracy: 0.8672 - loss: 0.3447 - val_accuracy: 0.8570 - val_loss: 0.3621
Epoch 3/30
125/125             108s 867ms/step -
accuracy: 0.8608 - loss: 0.3257 - val_accuracy: 0.8520 - val_loss: 0.3622
Epoch 4/30
125/125             104s 830ms/step -
accuracy: 0.8634 - loss: 0.3053 - val_accuracy: 0.8620 - val_loss: 0.3278
Epoch 5/30
125/125             104s 833ms/step -
accuracy: 0.8819 - loss: 0.2899 - val_accuracy: 0.8620 - val_loss: 0.4097
Epoch 6/30
125/125             104s 830ms/step -
accuracy: 0.8732 - loss: 0.3138 - val_accuracy: 0.8710 - val_loss: 0.3442
Epoch 7/30
125/125             105s 838ms/step -
accuracy: 0.8708 - loss: 0.2861 - val_accuracy: 0.8590 - val_loss: 0.3934
Epoch 8/30
125/125             105s 841ms/step -
accuracy: 0.8796 - loss: 0.2916 - val_accuracy: 0.8800 - val_loss: 0.3442
Epoch 9/30
125/125             111s 888ms/step -
accuracy: 0.8892 - loss: 0.2845 - val_accuracy: 0.8730 - val_loss: 0.3880
Epoch 10/30
125/125             104s 832ms/step -
```

```
accuracy: 0.8902 - loss: 0.2769 - val_accuracy: 0.8910 - val_loss: 0.3395
Epoch 11/30
125/125            102s 819ms/step -
accuracy: 0.8880 - loss: 0.2597 - val_accuracy: 0.8800 - val_loss: 0.3649
Epoch 12/30
125/125            103s 820ms/step -
accuracy: 0.9042 - loss: 0.2343 - val_accuracy: 0.8850 - val_loss: 0.3779
Epoch 13/30
125/125            102s 819ms/step -
accuracy: 0.8937 - loss: 0.2849 - val_accuracy: 0.8200 - val_loss: 0.5321
Epoch 14/30
125/125            102s 819ms/step -
accuracy: 0.8889 - loss: 0.2672 - val_accuracy: 0.8970 - val_loss: 0.3671
Epoch 15/30
125/125            102s 819ms/step -
accuracy: 0.9015 - loss: 0.2589 - val_accuracy: 0.8650 - val_loss: 0.4217
Epoch 16/30
125/125            102s 817ms/step -
accuracy: 0.9055 - loss: 0.2465 - val_accuracy: 0.8750 - val_loss: 0.3371
Epoch 17/30
125/125            102s 817ms/step -
accuracy: 0.8983 - loss: 0.2513 - val_accuracy: 0.8870 - val_loss: 0.3750
Epoch 18/30
125/125            104s 828ms/step -
accuracy: 0.9001 - loss: 0.2427 - val_accuracy: 0.8880 - val_loss: 0.3410
Epoch 19/30
125/125            105s 838ms/step -
accuracy: 0.9062 - loss: 0.2372 - val_accuracy: 0.8770 - val_loss: 0.3063
Epoch 20/30
125/125            102s 816ms/step -
accuracy: 0.9105 - loss: 0.2287 - val_accuracy: 0.9040 - val_loss: 0.3134
Epoch 21/30
125/125            103s 822ms/step -
accuracy: 0.9016 - loss: 0.2346 - val_accuracy: 0.8930 - val_loss: 0.3410
Epoch 22/30
125/125            103s 821ms/step -
accuracy: 0.9138 - loss: 0.2285 - val_accuracy: 0.8940 - val_loss: 0.3157
Epoch 23/30
125/125            103s 823ms/step -
accuracy: 0.9208 - loss: 0.2052 - val_accuracy: 0.8510 - val_loss: 0.4893
Epoch 24/30
125/125            104s 829ms/step -
accuracy: 0.9002 - loss: 0.2468 - val_accuracy: 0.8800 - val_loss: 0.3389
Epoch 25/30
125/125            103s 821ms/step -
accuracy: 0.9083 - loss: 0.2358 - val_accuracy: 0.9050 - val_loss: 0.3922
Epoch 26/30
125/125            104s 832ms/step -
```

```
accuracy: 0.8971 - loss: 0.2426 - val_accuracy: 0.8860 - val_loss: 0.5145
Epoch 27/30
125/125              103s 826ms/step -
accuracy: 0.9153 - loss: 0.2185 - val_accuracy: 0.8680 - val_loss: 0.4142
Epoch 28/30
125/125              104s 835ms/step -
accuracy: 0.9132 - loss: 0.2388 - val_accuracy: 0.8880 - val_loss: 0.4453
Epoch 29/30
125/125              108s 860ms/step -
accuracy: 0.9262 - loss: 0.2279 - val_accuracy: 0.8790 - val_loss: 0.3628
Epoch 30/30
125/125              103s 824ms/step -
accuracy: 0.9238 - loss: 0.2028 - val_accuracy: 0.8630 - val_loss: 0.6498
```

[16]:
```
# Testing the model
test_model = keras.models.load_model("convnet_from_scratch.keras")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32              9s 255ms/step -
accuracy: 0.8667 - loss: 0.3663
Test accuracy: 0.866
```

Question 3

[17]:
```
from tensorflow.keras.utils import image_dataset_from_directory
# Set up the training subset
make_subset("train3", start_index=0, end_index=2500)
make_subset("validation3", start_index=2500, end_index=3000)
make_subset("test3", start_index=3000, end_index=3500)


train_dataset = image_dataset_from_directory(
    small_directory / "train3",
    image_size=(180, 180),
    batch_size=32)
validation_dataset = image_dataset_from_directory(
    small_directory / "validation3",
    image_size=(180, 180),
    batch_size=32)
test_dataset = image_dataset_from_directory(
    small_directory / "test3",
    image_size=(180, 180),
    batch_size=32)


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

```
Found 5000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Epoch 1/30
157/157            132s 838ms/step -
accuracy: 0.8934 - loss: 0.2738 - val_accuracy: 0.6230 - val_loss: 2.9534
Epoch 2/30
157/157            128s 814ms/step -
accuracy: 0.8931 - loss: 0.3533 - val_accuracy: 0.8530 - val_loss: 0.4656
Epoch 3/30
157/157            129s 822ms/step -
accuracy: 0.9015 - loss: 0.2459 - val_accuracy: 0.8520 - val_loss: 0.4885
Epoch 4/30
157/157            128s 816ms/step -
accuracy: 0.9034 - loss: 0.2462 - val_accuracy: 0.8750 - val_loss: 0.3949
Epoch 5/30
157/157            323s 2s/step -
accuracy: 0.9003 - loss: 0.2509 - val_accuracy: 0.8380 - val_loss: 0.5518
Epoch 6/30
157/157            399s 3s/step -
accuracy: 0.9035 - loss: 0.2469 - val_accuracy: 0.8820 - val_loss: 0.3874
Epoch 7/30
157/157            501s 3s/step -
accuracy: 0.8992 - loss: 0.2571 - val_accuracy: 0.8690 - val_loss: 0.4359
Epoch 8/30
157/157            430s 2s/step -
accuracy: 0.9109 - loss: 0.2341 - val_accuracy: 0.8860 - val_loss: 0.4402
Epoch 9/30
157/157            388s 2s/step -
accuracy: 0.8928 - loss: 0.2562 - val_accuracy: 0.8680 - val_loss: 0.3408
Epoch 10/30
157/157            385s 2s/step -
accuracy: 0.9009 - loss: 0.2694 - val_accuracy: 0.8750 - val_loss: 0.4640
Epoch 11/30
157/157            446s 2s/step -
accuracy: 0.9036 - loss: 0.2511 - val_accuracy: 0.8790 - val_loss: 0.4207
Epoch 12/30
157/157            442s 2s/step -
accuracy: 0.9039 - loss: 0.2524 - val_accuracy: 0.8820 - val_loss: 0.4373
Epoch 13/30
157/157            438s 2s/step -
accuracy: 0.9013 - loss: 0.2446 - val_accuracy: 0.8840 - val_loss: 0.4204
Epoch 14/30
```

```
157/157              390s 2s/step -
accuracy: 0.9096 - loss: 0.2353 - val_accuracy: 0.8850 - val_loss: 0.3154
Epoch 15/30
157/157              383s 2s/step -
accuracy: 0.9168 - loss: 0.2117 - val_accuracy: 0.8910 - val_loss: 0.4458
Epoch 16/30
157/157              446s 2s/step -
accuracy: 0.9101 - loss: 0.2541 - val_accuracy: 0.8020 - val_loss: 0.5602
Epoch 17/30
157/157              442s 2s/step -
accuracy: 0.9106 - loss: 0.2361 - val_accuracy: 0.8930 - val_loss: 0.3960
Epoch 18/30
157/157              441s 2s/step -
accuracy: 0.9107 - loss: 0.2374 - val_accuracy: 0.8540 - val_loss: 0.5323
Epoch 19/30
157/157              390s 2s/step -
accuracy: 0.9117 - loss: 0.2392 - val_accuracy: 0.8370 - val_loss: 0.7038
Epoch 20/30
157/157              476s 3s/step -
accuracy: 0.9113 - loss: 0.2397 - val_accuracy: 0.8750 - val_loss: 0.3703
Epoch 21/30
157/157              413s 3s/step -
accuracy: 0.9140 - loss: 0.2366 - val_accuracy: 0.8990 - val_loss: 0.3755
Epoch 22/30
157/157              390s 2s/step -
accuracy: 0.9069 - loss: 0.2449 - val_accuracy: 0.8900 - val_loss: 0.4252
Epoch 23/30
157/157              385s 2s/step -
accuracy: 0.9052 - loss: 0.2772 - val_accuracy: 0.8680 - val_loss: 0.6309
Epoch 24/30
157/157              389s 2s/step -
accuracy: 0.9042 - loss: 0.2536 - val_accuracy: 0.8920 - val_loss: 0.5278
Epoch 25/30
157/157              383s 2s/step -
accuracy: 0.9122 - loss: 0.2299 - val_accuracy: 0.8470 - val_loss: 0.6819
Epoch 26/30
157/157              387s 2s/step -
accuracy: 0.9158 - loss: 0.2686 - val_accuracy: 0.8610 - val_loss: 0.6989
Epoch 27/30
157/157              388s 2s/step -
accuracy: 0.9041 - loss: 0.2532 - val_accuracy: 0.8740 - val_loss: 0.5317
Epoch 28/30
157/157              386s 2s/step -
accuracy: 0.8943 - loss: 0.2934 - val_accuracy: 0.8430 - val_loss: 0.6614
Epoch 29/30
157/157              387s 2s/step -
accuracy: 0.9136 - loss: 0.2355 - val_accuracy: 0.8780 - val_loss: 0.4405
Epoch 30/30
```

```
157/157              387s 2s/step -
accuracy: 0.9171 - loss: 0.2485 - val_accuracy: 0.8810 - val_loss: 0.5315
32/32               25s 714ms/step -
accuracy: 0.8955 - loss: 0.2341
```

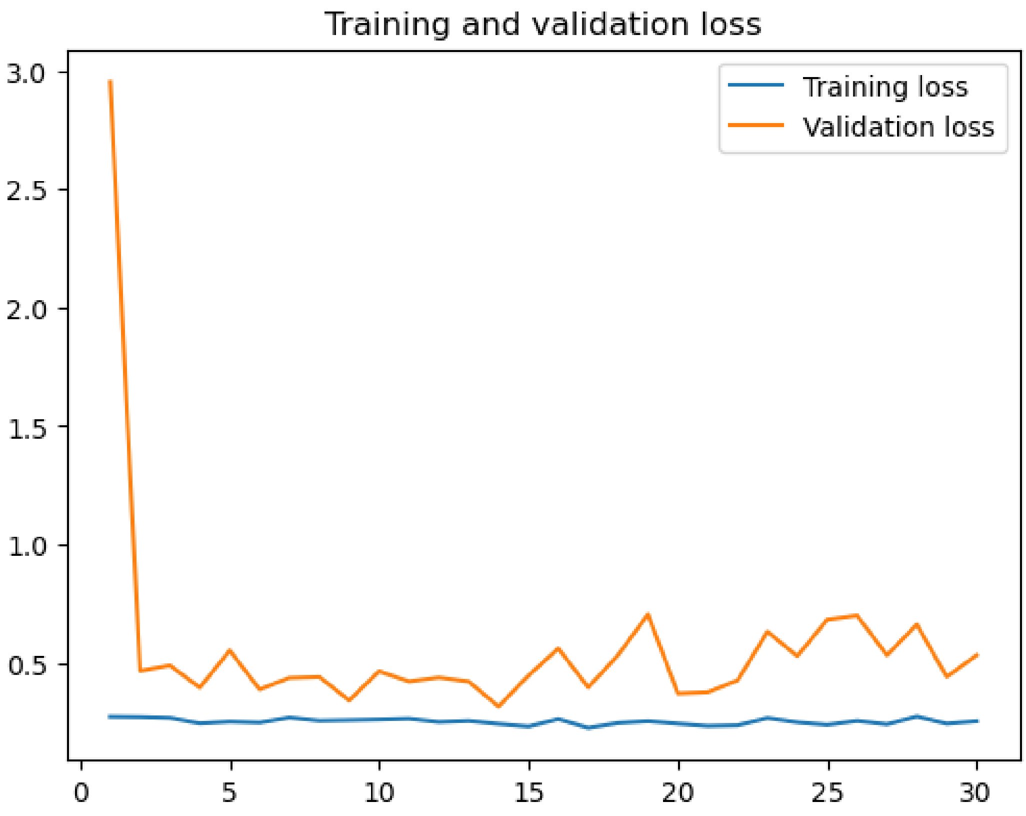
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[17], line 30
     28 test_model = keras.models.load_model("convnet_from_scratch.keras")
     29 test_loss, test_acc = test_model.evaluate(test_dataset)
---> 30 history_dict.append(test_acc)
     31 print(f"Test accuracy: {test_acc:.3f}")

NameError: name 'history_dict' is not defined
```

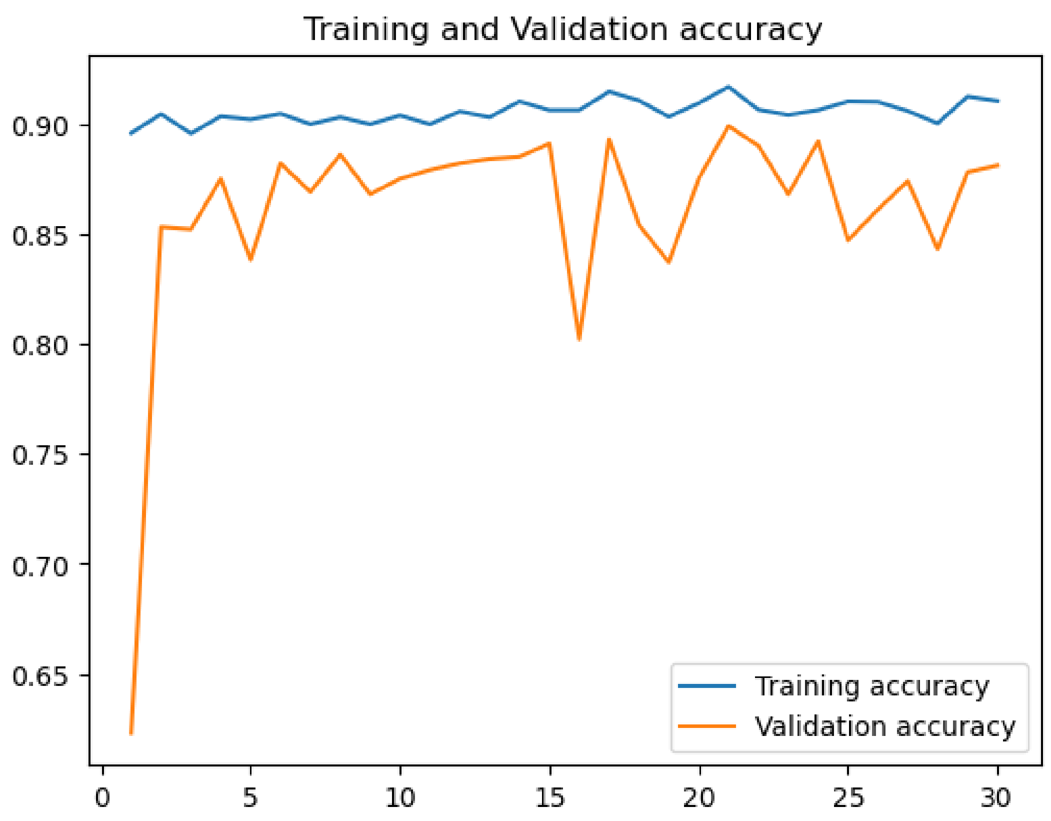
[23]:
```
test_model = keras.models.load_model("convnet_from_scratch.keras")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
63/63               45s 689ms/step -
accuracy: 0.9170 - loss: 0.2241
Test accuracy: 0.909
```

[24]:
```
# Storing accuracies and losses in variables
accuracy = history.history["accuracy"]
val_accuracy = history.history["val_accuracy"]
loss = history.history["loss"]
val_loss = history.history["val_loss"]
epochs = range(1,len(accuracy)+1)
# Plotting losses
plt.plot(epochs,loss,label="Training loss")
plt.plot(epochs,val_loss,label="Validation loss")
plt.title("Training and validation loss")
plt.legend()
plt.figure()
# Plotting accuracies
plt.plot(epochs,accuracy,label="Training accuracy")
plt.plot(epochs, val_accuracy,label="Validation accuracy")
plt.title("Training and Validation accuracy")
plt.legend()
```

[24]:
```
<matplotlib.legend.Legend at 0x207a42764d0>
```


Training and validation loss
Training loss
Validation loss
3.0
2.5
2.0
1.5
1.0
0.5
0
5
10
15
20
25
30



Question 4

Step - 1 data

[25]:
```
# Training has 1500 samples, test has 1000 samples, and validation has 500␣
 ↪samples


train_dataset = image_dataset_from_directory(
    small_directory / "train",
    image_size=(180, 180),
    batch_size=32)
validation_dataset = image_dataset_from_directory(
    small_directory / "validation",
    image_size=(180, 180),
    batch_size=32)
test_dataset = image_dataset_from_directory(
    small_directory / "test",
    image_size=(180, 180),
    batch_size=32)
```

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

[26]:
```python
# Loading pre-trained weights to VGG16 model
conv_base = keras.applications.vgg16.VGG16(
    weights="imagenet",
    include_top=False,
    input_shape=(180, 180, 3))
conv_base.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256              11s
0us/step
```

Model: "vgg16"

```
 Layer (type)                          Output Shape                            ␣
↪Param #

 input_layer_4 (InputLayer)            (None, 180, 180, 3)                         ␣
↪   0

 block1_conv1 (Conv2D)                 (None, 180, 180, 64)                      ␣
↪1,792

 block1_conv2 (Conv2D)                 (None, 180, 180, 64)                    ␣
↪36,928

 block1_pool (MaxPooling2D)            (None, 90, 90, 64)                          ␣
↪   0

 block2_conv1 (Conv2D)                 (None, 90, 90, 128)                     ␣
↪73,856

 block2_conv2 (Conv2D)                 (None, 90, 90, 128)                    ␣
↪147,584

 block2_pool (MaxPooling2D)            (None, 45, 45, 128)                         ␣
↪   0

 block3_conv1 (Conv2D)                 (None, 45, 45, 256)                    ␣
↪295,168

 block3_conv2 (Conv2D)                 (None, 45, 45, 256)                    ␣
↪590,080
```

```
 block3_conv3 (Conv2D)               (None, 45, 45, 256)                 ␣
↪590,080

 block3_pool (MaxPooling2D)          (None, 22, 22, 256)                     ␣
↪   0

 block4_conv1 (Conv2D)               (None, 22, 22, 512)               ␣
↪1,180,160

 block4_conv2 (Conv2D)               (None, 22, 22, 512)               ␣
↪2,359,808

 block4_conv3 (Conv2D)               (None, 22, 22, 512)               ␣
↪2,359,808

 block4_pool (MaxPooling2D)          (None, 11, 11, 512)                     ␣
↪   0

 block5_conv1 (Conv2D)               (None, 11, 11, 512)               ␣
↪2,359,808

 block5_conv2 (Conv2D)               (None, 11, 11, 512)               ␣
↪2,359,808

 block5_conv3 (Conv2D)               (None, 11, 11, 512)               ␣
↪2,359,808

 block5_pool (MaxPooling2D)          (None, 5, 5, 512)                       ␣
↪   0
```

```
Total params: 14,714,688 (56.13 MB)

Trainable params: 14,714,688 (56.13 MB)

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

[33]:
```
# Defining function to extract features and labels
import numpy as np

def get_features_and_labels(dataset):
    all_features = []
    all_labels = []
    for images, labels in dataset:
        preprocessed_images = keras.applications.vgg16.preprocess_input(images)
        features = conv_base.predict(preprocessed_images)
        all_features.append(features)
        all_labels.append(labels)
    return np.concatenate(all_features), np.concatenate(all_labels)
# Extracting the features and labels from datasets
train_features, train_labels =  get_features_and_labels(train_dataset)
val_features, val_labels =  get_features_and_labels(validation_dataset)
test_features, test_labels =  get_features_and_labels(test_dataset)
train_features.shape
```

```
1/1               14s 14s/step
1/1               13s 13s/step
1/1               13s 13s/step
1/1               13s 13s/step
1/1               4s 4s/step
1/1               5s 5s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
1/1               4s 4s/step
```

```
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 6s 6s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 2s 2s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 6s 6s/step
1/1                 6s 6s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
1/1                 4s 4s/step
```

```
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                1s 1s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                5s 5s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                5s 5s/step
1/1                4s 4s/step
1/1                4s 4s/step
1/1                5s 5s/step
1/1                4s 4s/step
1/1                4s 4s/step
```

```
1/1              4s 4s/step
1/1              4s 4s/step
1/1              4s 4s/step
1/1              4s 4s/step
1/1              4s 4s/step
1/1              4s 4s/step
1/1              4s 4s/step
1/1              4s 4s/step
1/1              4s 4s/step
1/1              4s 4s/step
1/1              4s 4s/step
1/1              4s 4s/step
1/1              4s 4s/step
1/1              4s 4s/step
1/1              4s 4s/step
1/1              4s 4s/step
1/1              4s 4s/step
1/1              4s 4s/step
1/1              4s 4s/step
1/1              4s 4s/step
1/1              4s 4s/step
1/1              4s 4s/step
1/1              4s 4s/step
1/1              4s 4s/step
1/1              4s 4s/step
1/1              4s 4s/step
1/1              2s 2s/step
```

[33]:
```
(2000, 5, 5, 512)
```

[34]:
```python
# Building the model
inputs = keras.Input(shape=(5, 5, 512))
x = layers.Flatten()(inputs)
x = layers.Dense(256)(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs, outputs)
model.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])

# Running the callback function to monitor validation loss
callbacks = [
    keras.callbacks.ModelCheckpoint(
      filepath="feature_extraction.keras",
      save_best_only=True,
      monitor="val_loss")
]

# Training the model
history = model.fit(
    train_features, train_labels,
    epochs=30,
    validation_data=(val_features, val_labels),
    callbacks=callbacks)
```

```
Epoch 1/30
63/63                 5s 37ms/step -
accuracy: 0.8611 - loss: 26.4721 - val_accuracy: 0.9730 - val_loss: 3.3998
Epoch 2/30
63/63                 2s 28ms/step -
accuracy: 0.9725 - loss: 3.7335 - val_accuracy: 0.9500 - val_loss: 9.8640
Epoch 3/30
63/63                 2s 27ms/step -
accuracy: 0.9793 - loss: 2.3200 - val_accuracy: 0.9730 - val_loss: 5.0514
Epoch 4/30
63/63                 3s 29ms/step -
accuracy: 0.9899 - loss: 1.1943 - val_accuracy: 0.9790 - val_loss: 3.5139
Epoch 5/30
63/63                 2s 27ms/step -
accuracy: 0.9945 - loss: 0.7297 - val_accuracy: 0.9790 - val_loss: 4.0251
Epoch 6/30
63/63                 2s 28ms/step -
accuracy: 0.9970 - loss: 0.2611 - val_accuracy: 0.9760 - val_loss: 3.8740
Epoch 7/30
63/63                 2s 28ms/step -
accuracy: 0.9927 - loss: 0.9179 - val_accuracy: 0.9790 - val_loss: 3.9487
Epoch 8/30
63/63                 2s 28ms/step -
accuracy: 0.9999 - loss: 0.0097 - val_accuracy: 0.9740 - val_loss: 4.7128
Epoch 9/30
63/63                 2s 27ms/step -
accuracy: 0.9959 - loss: 0.3810 - val_accuracy: 0.9810 - val_loss: 3.8231
Epoch 10/30
63/63                 2s 28ms/step -
accuracy: 0.9989 - loss: 0.0801 - val_accuracy: 0.9810 - val_loss: 3.7257
Epoch 11/30
63/63                 2s 28ms/step -
accuracy: 0.9988 - loss: 0.0869 - val_accuracy: 0.9760 - val_loss: 4.9615
Epoch 12/30
63/63                 2s 27ms/step -
accuracy: 0.9988 - loss: 0.2072 - val_accuracy: 0.9810 - val_loss: 4.6418
Epoch 13/30
63/63                 2s 28ms/step -
```

```
accuracy: 0.9970 - loss: 0.2032 - val_accuracy: 0.9830 - val_loss: 4.0999
Epoch 14/30
63/63            2s 28ms/step -
accuracy: 0.9963 - loss: 0.1376 - val_accuracy: 0.9820 - val_loss: 4.5909
Epoch 15/30
63/63            2s 27ms/step -
accuracy: 1.0000 - loss: 1.2925e-08 - val_accuracy: 0.9820 - val_loss: 4.5743
Epoch 16/30
63/63            3s 29ms/step -
accuracy: 0.9985 - loss: 0.0787 - val_accuracy: 0.9810 - val_loss: 4.7422
Epoch 17/30
63/63            2s 27ms/step -
accuracy: 0.9989 - loss: 0.0508 - val_accuracy: 0.9840 - val_loss: 4.1882
Epoch 18/30
63/63            2s 26ms/step -
accuracy: 1.0000 - loss: 1.2407e-11 - val_accuracy: 0.9840 - val_loss: 4.1882
Epoch 19/30
63/63            2s 28ms/step -
accuracy: 0.9980 - loss: 0.1779 - val_accuracy: 0.9790 - val_loss: 4.2030
Epoch 20/30
63/63            2s 28ms/step -
accuracy: 1.0000 - loss: 1.4191e-08 - val_accuracy: 0.9790 - val_loss: 4.2015
Epoch 21/30
63/63            2s 28ms/step -
accuracy: 1.0000 - loss: 3.0198e-09 - val_accuracy: 0.9790 - val_loss: 4.1996
Epoch 22/30
63/63            2s 28ms/step -
accuracy: 0.9991 - loss: 0.0723 - val_accuracy: 0.9740 - val_loss: 4.9745
Epoch 23/30
63/63            3s 29ms/step -
accuracy: 0.9982 - loss: 0.4865 - val_accuracy: 0.9740 - val_loss: 5.1009
Epoch 24/30
63/63            2s 27ms/step -
accuracy: 1.0000 - loss: 1.2341e-13 - val_accuracy: 0.9740 - val_loss: 5.1009
Epoch 25/30
63/63            2s 28ms/step -
accuracy: 0.9972 - loss: 0.2697 - val_accuracy: 0.9770 - val_loss: 5.4381
Epoch 26/30
63/63            2s 27ms/step -
accuracy: 0.9991 - loss: 0.0472 - val_accuracy: 0.9770 - val_loss: 6.0637
Epoch 27/30
63/63            2s 27ms/step -
accuracy: 1.0000 - loss: 7.6638e-08 - val_accuracy: 0.9790 - val_loss: 6.0050
Epoch 28/30
63/63            3s 28ms/step -
accuracy: 0.9991 - loss: 0.0790 - val_accuracy: 0.9850 - val_loss: 4.1412
Epoch 29/30
63/63            2s 27ms/step -
```

```
accuracy: 1.0000 - loss: 2.7671e-32 - val_accuracy: 0.9850 - val_loss: 4.1412
Epoch 30/30
63/63              2s 25ms/step -
accuracy: 0.9996 - loss: 0.0435 - val_accuracy: 0.9820 - val_loss: 4.9244
```

[35]:
```
# Testing the model
test_model = keras.models.load_model("feature_extraction.keras")
test_loss, test_acc = test_model.evaluate(test_features, test_labels)
print(f"Test accuracy: {test_acc:.3f}")
```

```
63/63              1s 11ms/step -
accuracy: 0.9691 - loss: 4.1440
Test accuracy: 0.965
```

[62]:
```
# Loading pre-trained weights to the VGG16 model
conv_base  = keras.applications.vgg16.VGG16(
    weights="imagenet",
    include_top=False)

# UnFreezing the layers of the pretrained CNN
conv_base.trainable = True
for layer in conv_base.layers[:-4]:
    layer.trainable = False
```

[63]:
```
# Declaring Data Augumentation
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)
# Building the model and configuring it
inputs = keras.Input(shape=(180, 180, 3))
x = data_augmentation(inputs)
x = keras.applications.vgg16.preprocess_input(x)
x = conv_base(x)
x = layers.Flatten()(x)
x = layers.Dense(256)(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs, outputs)

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

[64]:
```
# Using the callbacks function to monitor validation loss and running the model
history = model.fit(
    train_dataset,
    epochs=20,
    validation_data=validation_dataset)
```

```
Epoch 1/20
63/63                 1465s 23s/step -
accuracy: 0.4998 - loss: 846.0219 - val_accuracy: 0.5000 - val_loss: 0.6931
Epoch 2/20
63/63                 1237s 19s/step -
accuracy: 0.5015 - loss: 0.6936 - val_accuracy: 0.5000 - val_loss: 0.6932
Epoch 3/20
63/63                 435s 7s/step -
accuracy: 0.4603 - loss: 0.6939 - val_accuracy: 0.5000 - val_loss: 0.6931
Epoch 4/20
63/63                 432s 7s/step -
accuracy: 0.5067 - loss: 0.6933 - val_accuracy: 0.5000 - val_loss: 0.6931
Epoch 5/20
63/63                 433s 7s/step -
accuracy: 0.4910 - loss: 0.6937 - val_accuracy: 0.5000 - val_loss: 0.6933
Epoch 6/20
63/63                 434s 7s/step -
accuracy: 0.5060 - loss: 0.6933 - val_accuracy: 0.5000 - val_loss: 0.6932
Epoch 7/20
63/63                 435s 7s/step -
accuracy: 0.4950 - loss: 0.6937 - val_accuracy: 0.5000 - val_loss: 0.6934
Epoch 8/20
63/63                 435s 7s/step -
accuracy: 0.4871 - loss: 0.6941 - val_accuracy: 0.5000 - val_loss: 0.6931
Epoch 9/20
63/63                 434s 7s/step -
accuracy: 0.4938 - loss: 0.6936 - val_accuracy: 0.5000 - val_loss: 0.6934
Epoch 10/20
63/63                 436s 7s/step -
accuracy: 0.4829 - loss: 0.6936 - val_accuracy: 0.5000 - val_loss: 0.6932
Epoch 11/20
63/63                 436s 7s/step -
accuracy: 0.4951 - loss: 0.6932 - val_accuracy: 0.5000 - val_loss: 0.6931
Epoch 12/20
63/63                 436s 7s/step -
accuracy: 0.4900 - loss: 0.6935 - val_accuracy: 0.5000 - val_loss: 0.6932
Epoch 13/20
63/63                 435s 7s/step -
accuracy: 0.4940 - loss: 0.6938 - val_accuracy: 0.5000 - val_loss: 0.6932
Epoch 14/20
63/63                 435s 7s/step -
```

```
accuracy: 0.5234 - loss: 0.6931 - val_accuracy: 0.5000 - val_loss: 0.6933
Epoch 15/20
63/63                 436s 7s/step -
accuracy: 0.4777 - loss: 0.6943 - val_accuracy: 0.5000 - val_loss: 0.6931
Epoch 16/20
63/63                 437s 7s/step -
accuracy: 0.5020 - loss: 0.6935 - val_accuracy: 0.5000 - val_loss: 0.6932
Epoch 17/20
63/63                 436s 7s/step -
accuracy: 0.4935 - loss: 0.6936 - val_accuracy: 0.5000 - val_loss: 0.6933
Epoch 18/20
63/63                 434s 7s/step -
accuracy: 0.4968 - loss: 0.6931 - val_accuracy: 0.5000 - val_loss: 0.6933
Epoch 19/20
63/63                 435s 7s/step -
accuracy: 0.4854 - loss: 0.6933 - val_accuracy: 0.5000 - val_loss: 0.6933
Epoch 20/20
63/63                 436s 7s/step -
accuracy: 0.4775 - loss: 0.6946 - val_accuracy: 0.5000 - val_loss: 0.6931
```

[78]:
```
# Testing the model
test_model = model.evaluate(test_dataset)
```

```
32/32                 133s 4s/step -
accuracy: 0.4837 - loss: 0.6939
```

[73]:
```
# Training the model on last sample as well:

train_dataset = image_dataset_from_directory(
    small_directory / "train3",
    image_size=(180, 180),
    batch_size=32)
validation_dataset = image_dataset_from_directory(
    small_directory / "validation3",
    image_size=(180, 180),
    batch_size=32)
test_dataset = image_dataset_from_directory(
    small_directory / "test3",
    image_size=(180, 180),
    batch_size=32)

    # Training the model
history = model.fit(
      train_dataset,
      epochs=5,
      validation_data=validation_dataset)
```

```
Found 5000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Epoch 1/5
157/157              1053s 7s/step -
accuracy: 0.5134 - loss: 0.6928 - val_accuracy: 0.5000 - val_loss: 0.6932
Epoch 2/5
157/157              919s 6s/step -
accuracy: 0.4990 - loss: 0.6935 - val_accuracy: 0.5000 - val_loss: 0.6931
Epoch 3/5
157/157              917s 6s/step -
accuracy: 0.4973 - loss: 0.6936 - val_accuracy: 0.5000 - val_loss: 0.6933
Epoch 4/5
157/157              916s 6s/step -
accuracy: 0.4953 - loss: 0.6940 - val_accuracy: 0.5000 - val_loss: 0.6933
Epoch 5/5
157/157              917s 6s/step -
accuracy: 0.4989 - loss: 0.6939 - val_accuracy: 0.5000 - val_loss: 0.6934
```

[75]:
```
# Testing the model
test_model = keras.models.load_model("feature_extraction.keras")
test_loss, test_acc = test_model.evaluate(test_features, test_labels)
print(f"Test accuracy: {test_acc:.3f}")
```

```
63/63              1s 7ms/step -
accuracy: 0.9691 - loss: 4.1440
Test accuracy: 0.965
```