aml-assignment-2

March 31, 2024

Assignment 2: Convolution.

SHIVA CHIATANYA GOUD GADILA - 811252042

MALLENAHALLI VIJAYA PRANEETH SIMHA - 811252718

Retrieving the data

[1]:
```
!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

[2]:
```
!kaggle competitions download -c dogs-vs-cats
```

```
Downloading dogs-vs-cats.zip to /content
100% 811M/812M [00:38<00:00, 24.0MB/s]
100% 812M/812M [00:39<00:00, 21.8MB/s]
```

[3]:
```
!unzip -qq dogs-vs-cats.zip
!unzip -qq train.zip
```

Transferring the images to the training, validation, and test directories.

[4]:
```
import os, shutil, pathlib

original_dir = pathlib.Path("train")
new_base_dir = pathlib.Path("cats_vs_dogs_small")

def make_subset(subset_name, start_index, end_index):
    for category in ("cat", "dog"):
        dir = new_base_dir / 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=original_dir / fname,
                            dst=dir / fname)
```

TRAINING THE CONVENT NETWORK FROM SCRATCH:

MODEL 1: TRAINING SAMPLE OF 1000, VALIDATION SAMPLE OF 500 AND TEST SAMPLE OF 500

[5]:
```
make_subset("test", start_index=0, end_index=500)
make_subset("validation", start_index=500, end_index=1000)
make_subset("train", start_index=1000, end_index=2000)
```

[6]:
```
from tensorflow import keras
from tensorflow.keras import layers
```

[7]:
```
#Instantiating a small convnet for dogs vs. cats classification:

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_1 = keras.Model(inputs=inputs, outputs=outputs)
```

[8]:
```
Model_1.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 180, 180, 3)]     0

 rescaling (Rescaling)       (None, 180, 180, 3)       0

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

 max_pooling2d (MaxPooling2  (None, 89, 89, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 87, 87, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 43, 43, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 41, 41, 128)       73856

 max_pooling2d_2 (MaxPoolin  (None, 20, 20, 128)       0
 g2D)

 conv2d_3 (Conv2D)           (None, 18, 18, 256)       295168

 max_pooling2d_3 (MaxPoolin  (None, 9, 9, 256)         0
 g2D)

 conv2d_4 (Conv2D)           (None, 7, 7, 256)         590080

 flatten (Flatten)           (None, 12544)             0

 dense (Dense)               (None, 1)                 12545

=================================================================
Total params: 991041 (3.78 MB)
Trainable params: 991041 (3.78 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[9]:

```
#Configuring the model for training:

Model_1.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

DATA PREPROCESSING:

[10]:

```
#Using image_dataset_from_directory to read images

from tensorflow.keras.utils import image_dataset_from_directory

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

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

[11]:

```
import numpy as np
import tensorflow as tf
random_numbers = np.random.normal(size=(1000, 16))
dataset = tf.data.Dataset.from_tensor_slices(random_numbers)
```

[12]:

```
for i, element in enumerate(dataset):
    print(element.shape)
    if i >= 2:
        break
```

```
(16,)
(16,)
(16,)
```

[13]:

```
batched_dataset = dataset.batch(32)
for i, element in enumerate(batched_dataset):
    print(element.shape)
    if i >= 2:
        break
```

```
(32, 16)
(32, 16)
(32, 16)
```

[14]:

```
reshaped_dataset = dataset.map(lambda x: tf.reshape(x, (4, 4)))
for i, element in enumerate(reshaped_dataset):
    print(element.shape)
    if i >= 2:
        break
```

```
(4, 4)
(4, 4)
(4, 4)
```

[15]:

```
#Displaying the shapes of the data and labels yielded by the Dataset:

for data_batch, labels_batch in train_dataset:
    print("data batch shape:", data_batch.shape)
    print("labels batch shape:", labels_batch.shape)
    break
```

```
data batch shape: (32, 180, 180, 3)
labels batch shape: (32,)
```

[16]:

```
#Fitting the model using a Dataset

callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = Model_1.fit(
    train_dataset,
    epochs=30,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

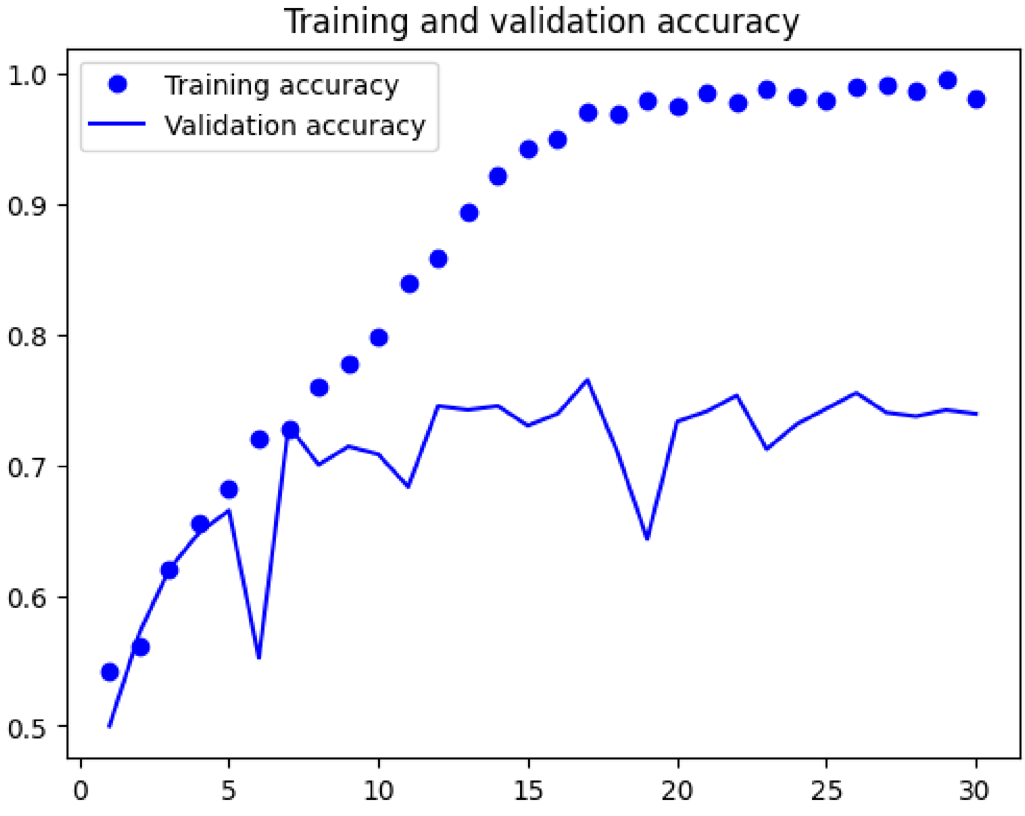
```
Epoch 1/30
63/63 [==============================] - 14s 133ms/step - loss: 0.6924 -
accuracy: 0.5425 - val_loss: 0.7394 - val_accuracy: 0.5000
Epoch 2/30
63/63 [==============================] - 5s 75ms/step - loss: 0.6884 - accuracy:
0.5610 - val_loss: 0.6769 - val_accuracy: 0.5710
Epoch 3/30
63/63 [==============================] - 4s 63ms/step - loss: 0.6539 - accuracy:
0.6200 - val_loss: 0.6455 - val_accuracy: 0.6200
Epoch 4/30
63/63 [==============================] - 6s 95ms/step - loss: 0.6224 - accuracy:
0.6550 - val_loss: 0.6343 - val_accuracy: 0.6480
Epoch 5/30
63/63 [==============================] - 5s 74ms/step - loss: 0.5940 - accuracy:
0.6820 - val_loss: 0.6235 - val_accuracy: 0.6650
Epoch 6/30
63/63 [==============================] - 4s 58ms/step - loss: 0.5722 - accuracy:
0.7195 - val_loss: 0.7598 - val_accuracy: 0.5520
Epoch 7/30
63/63 [==============================] - 4s 64ms/step - loss: 0.5337 - accuracy:
0.7280 - val_loss: 0.5439 - val_accuracy: 0.7300
Epoch 8/30
63/63 [==============================] - 7s 111ms/step - loss: 0.5034 -
accuracy: 0.7595 - val_loss: 0.5992 - val_accuracy: 0.7000
Epoch 9/30
63/63 [==============================] - 4s 58ms/step - loss: 0.4716 - accuracy:
0.7775 - val_loss: 0.6081 - val_accuracy: 0.7140
Epoch 10/30
63/63 [==============================] - 5s 81ms/step - loss: 0.4234 - accuracy:
0.7980 - val_loss: 0.6847 - val_accuracy: 0.7080
Epoch 11/30
63/63 [==============================] - 4s 61ms/step - loss: 0.3738 - accuracy:
0.8390 - val_loss: 0.6834 - val_accuracy: 0.6830
Epoch 12/30
63/63 [==============================] - 5s 76ms/step - loss: 0.3177 - accuracy:
0.8590 - val_loss: 0.6105 - val_accuracy: 0.7450
Epoch 13/30
```

```
63/63 [==============================] - 6s 87ms/step - loss: 0.2444 - accuracy:
0.8935 - val_loss: 0.6725 - val_accuracy: 0.7420
Epoch 14/30
63/63 [==============================] - 4s 65ms/step - loss: 0.2137 - accuracy:
0.9210 - val_loss: 0.8268 - val_accuracy: 0.7450
Epoch 15/30
63/63 [==============================] - 7s 111ms/step - loss: 0.1625 -
accuracy: 0.9415 - val_loss: 0.9790 - val_accuracy: 0.7300
Epoch 16/30
63/63 [==============================] - 4s 61ms/step - loss: 0.1498 - accuracy:
0.9500 - val_loss: 0.8154 - val_accuracy: 0.7390
Epoch 17/30
63/63 [==============================] - 5s 82ms/step - loss: 0.0886 - accuracy:
0.9695 - val_loss: 1.0547 - val_accuracy: 0.7650
Epoch 18/30
63/63 [==============================] - 6s 88ms/step - loss: 0.0947 - accuracy:
0.9685 - val_loss: 1.0872 - val_accuracy: 0.7100
Epoch 19/30
63/63 [==============================] - 4s 59ms/step - loss: 0.0571 - accuracy:
0.9790 - val_loss: 1.7071 - val_accuracy: 0.6430
Epoch 20/30
63/63 [==============================] - 4s 64ms/step - loss: 0.0766 - accuracy:
0.9745 - val_loss: 1.3385 - val_accuracy: 0.7330
Epoch 21/30
63/63 [==============================] - 7s 97ms/step - loss: 0.0426 - accuracy:
0.9845 - val_loss: 1.3004 - val_accuracy: 0.7410
Epoch 22/30
63/63 [==============================] - 4s 57ms/step - loss: 0.0671 - accuracy:
0.9775 - val_loss: 1.4239 - val_accuracy: 0.7530
Epoch 23/30
63/63 [==============================] - 6s 95ms/step - loss: 0.0356 - accuracy:
0.9875 - val_loss: 2.4992 - val_accuracy: 0.7120
Epoch 24/30
63/63 [==============================] - 5s 71ms/step - loss: 0.0698 - accuracy:
0.9820 - val_loss: 1.4507 - val_accuracy: 0.7310
Epoch 25/30
63/63 [==============================] - 4s 57ms/step - loss: 0.0862 - accuracy:
0.9790 - val_loss: 1.4374 - val_accuracy: 0.7430
Epoch 26/30
63/63 [==============================] - 5s 74ms/step - loss: 0.0278 - accuracy:
0.9890 - val_loss: 1.5221 - val_accuracy: 0.7550
Epoch 27/30
63/63 [==============================] - 6s 94ms/step - loss: 0.0313 - accuracy:
0.9905 - val_loss: 1.9706 - val_accuracy: 0.7400
Epoch 28/30
63/63 [==============================] - 4s 58ms/step - loss: 0.0444 - accuracy:
0.9860 - val_loss: 1.9662 - val_accuracy: 0.7370
Epoch 29/30
```

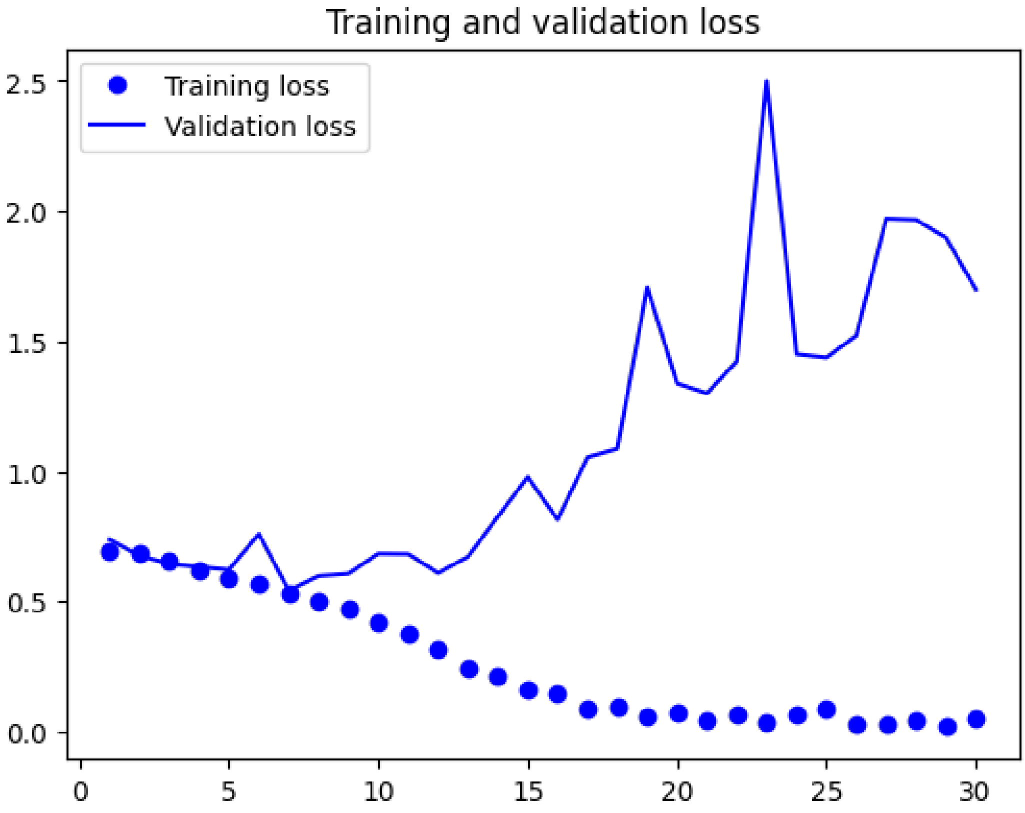
```
63/63 [==============================] - 4s 61ms/step - loss: 0.0191 - accuracy:
0.9945 - val_loss: 1.8982 - val_accuracy: 0.7420
Epoch 30/30
63/63 [==============================] - 6s 98ms/step - loss: 0.0498 - accuracy:
0.9810 - val_loss: 1.6993 - val_accuracy: 0.7390
```

[17]:

```
#Displaying curves of loss and accuracy during training:

import matplotlib.pyplot as plt
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)
plt.plot(epochs, accuracy, "bo", label="Training accuracy")
plt.plot(epochs, val_accuracy, "b", label="Validation accuracy")
plt.title("Training and validation accuracy")
plt.legend()
plt.figure()
plt.plot(epochs, loss, "bo", label="Training loss")
plt.plot(epochs, val_loss, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.legend()
plt.show()#
```



Training and validation accuracy



[18]:
```
#Evaluating the model on the test set:

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 [==============================] - 1s 29ms/step - loss: 0.5348 - accuracy:
0.7370
Test accuracy: 0.737
```

REDUCING THE OVERFITTING MODELS

MODEL 2- USING DATA AUGMENTATION:

[19]:
```
from tensorflow import keras
from tensorflow.keras import layers
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)
```

[20]:

```
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)
outputs = layers.Dense(1, activation="sigmoid")(x)
Model_2 = keras.Model(inputs=inputs, outputs=outputs)

Model_2.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

[21]:

```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_with_augmentation.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = Model_2.fit(
    train_dataset,
    epochs=30,
    validation_data=validation_dataset,
     callbacks=callbacks)
```

```
Epoch 1/30
63/63 [==============================] - 10s 103ms/step - loss: 0.7007 -
accuracy: 0.5030 - val_loss: 0.6927 - val_accuracy: 0.5000
Epoch 2/30
63/63 [==============================] - 4s 64ms/step - loss: 0.6943 - accuracy:
0.5000 - val_loss: 0.6909 - val_accuracy: 0.5640
Epoch 3/30
63/63 [==============================] - 4s 60ms/step - loss: 0.6893 - accuracy:
0.5365 - val_loss: 0.6719 - val_accuracy: 0.6260
Epoch 4/30
63/63 [==============================] - 6s 98ms/step - loss: 0.6765 - accuracy:
```

```
0.5900 - val_loss: 0.6565 - val_accuracy: 0.6170
Epoch 5/30
63/63 [==============================] - 4s 60ms/step - loss: 0.6600 - accuracy:
0.6355 - val_loss: 0.6616 - val_accuracy: 0.6000
Epoch 6/30
63/63 [==============================] - 6s 86ms/step - loss: 0.6358 - accuracy:
0.6430 - val_loss: 0.6384 - val_accuracy: 0.6330
Epoch 7/30
63/63 [==============================] - 6s 89ms/step - loss: 0.6434 - accuracy:
0.6580 - val_loss: 0.6226 - val_accuracy: 0.6540
Epoch 8/30
63/63 [==============================] - 4s 61ms/step - loss: 0.6025 - accuracy:
0.6725 - val_loss: 0.5940 - val_accuracy: 0.6950
Epoch 9/30
63/63 [==============================] - 4s 58ms/step - loss: 0.6046 - accuracy:
0.6850 - val_loss: 0.6440 - val_accuracy: 0.6660
Epoch 10/30
63/63 [==============================] - 7s 98ms/step - loss: 0.5990 - accuracy:
0.6790 - val_loss: 0.6212 - val_accuracy: 0.7000
Epoch 11/30
63/63 [==============================] - 4s 60ms/step - loss: 0.5847 - accuracy:
0.6875 - val_loss: 0.5835 - val_accuracy: 0.6870
Epoch 12/30
63/63 [==============================] - 7s 100ms/step - loss: 0.5803 -
accuracy: 0.6965 - val_loss: 0.5810 - val_accuracy: 0.7010
Epoch 13/30
63/63 [==============================] - 5s 69ms/step - loss: 0.5737 - accuracy:
0.7075 - val_loss: 0.5667 - val_accuracy: 0.7170
Epoch 14/30
63/63 [==============================] - 4s 61ms/step - loss: 0.5404 - accuracy:
0.7310 - val_loss: 0.5244 - val_accuracy: 0.7410
Epoch 15/30
63/63 [==============================] - 6s 86ms/step - loss: 0.5473 - accuracy:
0.7315 - val_loss: 0.5199 - val_accuracy: 0.7560
Epoch 16/30
63/63 [==============================] - 6s 87ms/step - loss: 0.5290 - accuracy:
0.7295 - val_loss: 0.5523 - val_accuracy: 0.7380
Epoch 17/30
63/63 [==============================] - 5s 76ms/step - loss: 0.5272 - accuracy:
0.7415 - val_loss: 0.5913 - val_accuracy: 0.7260
Epoch 18/30
63/63 [==============================] - 7s 101ms/step - loss: 0.5089 -
accuracy: 0.7435 - val_loss: 0.7099 - val_accuracy: 0.6080
Epoch 19/30
63/63 [==============================] - 4s 60ms/step - loss: 0.5016 - accuracy:
0.7475 - val_loss: 0.4918 - val_accuracy: 0.7700
Epoch 20/30
63/63 [==============================] - 4s 60ms/step - loss: 0.4940 - accuracy:
```

```
0.7620 - val_loss: 0.9062 - val_accuracy: 0.6680
Epoch 21/30
63/63 [==============================] - 6s 87ms/step - loss: 0.4851 - accuracy:
0.7630 - val_loss: 0.4852 - val_accuracy: 0.7840
Epoch 22/30
63/63 [==============================] - 6s 81ms/step - loss: 0.4713 - accuracy:
0.7765 - val_loss: 0.6031 - val_accuracy: 0.7510
Epoch 23/30
63/63 [==============================] - 4s 58ms/step - loss: 0.4630 - accuracy:
0.7785 - val_loss: 0.5476 - val_accuracy: 0.7550
Epoch 24/30
63/63 [==============================] - 4s 61ms/step - loss: 0.4627 - accuracy:
0.7815 - val_loss: 0.4785 - val_accuracy: 0.7820
Epoch 25/30
63/63 [==============================] - 7s 102ms/step - loss: 0.4402 -
accuracy: 0.8010 - val_loss: 0.5763 - val_accuracy: 0.7380
Epoch 26/30
63/63 [==============================] - 4s 63ms/step - loss: 0.4411 - accuracy:
0.8010 - val_loss: 0.5013 - val_accuracy: 0.7800
Epoch 27/30
63/63 [==============================] - 4s 60ms/step - loss: 0.4409 - accuracy:
0.7935 - val_loss: 0.4463 - val_accuracy: 0.8140
Epoch 28/30
63/63 [==============================] - 5s 73ms/step - loss: 0.4281 - accuracy:
0.7980 - val_loss: 0.4518 - val_accuracy: 0.8010
Epoch 29/30
63/63 [==============================] - 6s 87ms/step - loss: 0.4113 - accuracy:
0.8160 - val_loss: 0.4863 - val_accuracy: 0.7820
Epoch 30/30
63/63 [==============================] - 4s 58ms/step - loss: 0.4127 - accuracy:
0.8185 - val_loss: 0.5258 - val_accuracy: 0.7560
```

[22]:

```
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}")
```

```
32/32 [==============================] - 2s 32ms/step - loss: 0.4246 - accuracy:
0.8190
Test accuracy: 0.819
```

[ ]:

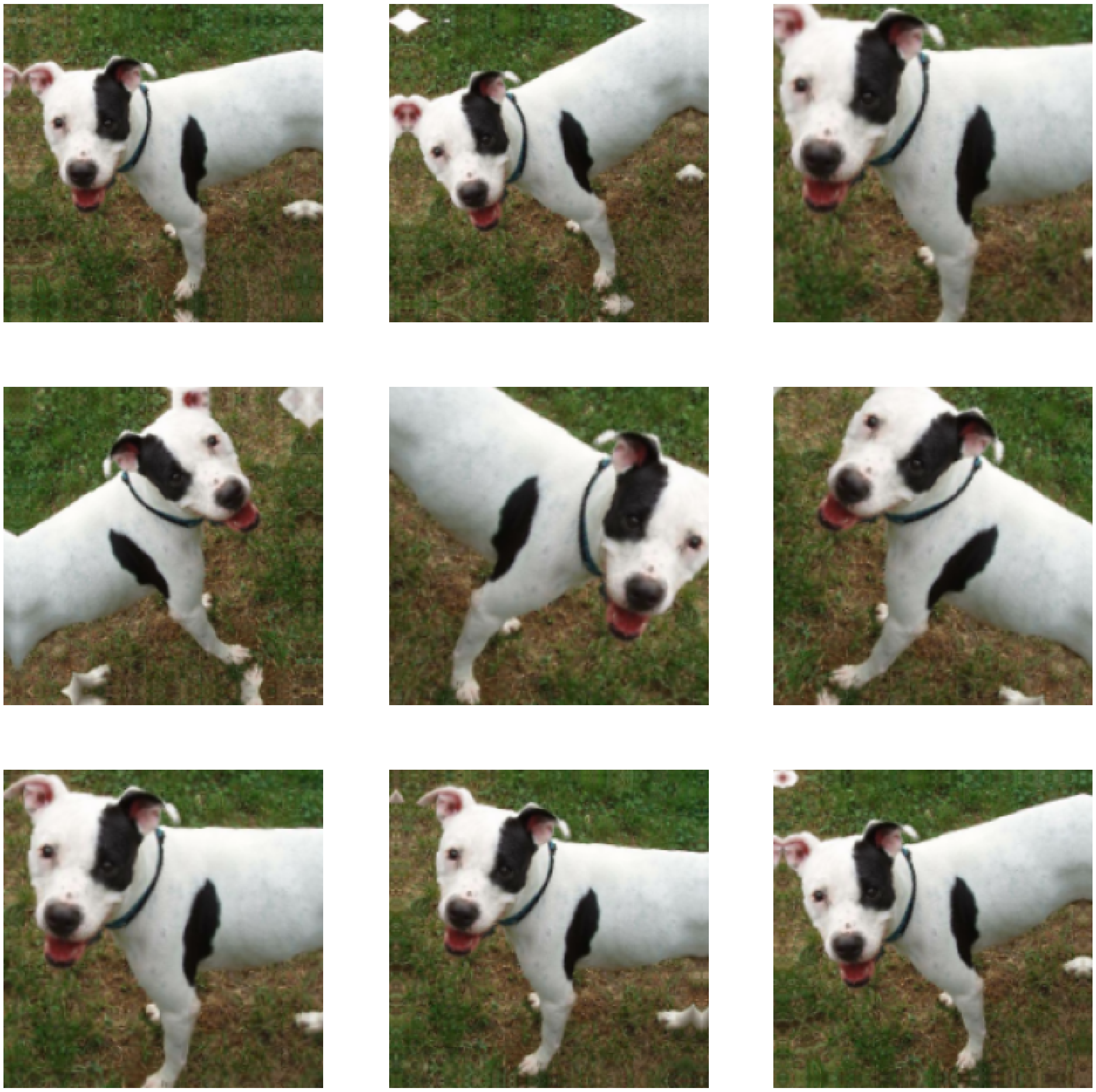
```

```

[23]:

```
#Defining a data augmentation stage to add to an image model:

data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)
```

[24]:

```
#Displaying some randomly augmented training images

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")
```



MODEL 3 - DROPOUT METHOD

[25]:

```
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)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
Model_3 = keras.Model(inputs=inputs, outputs=outputs)

Model_3.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

[26]:

```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_with_dropout.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = Model_3.fit(
    train_dataset,
    epochs=30,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/30
63/63 [==============================] - 10s 102ms/step - loss: 0.7047 -
accuracy: 0.5145 - val_loss: 0.6921 - val_accuracy: 0.5000
Epoch 2/30
63/63 [==============================] - 4s 60ms/step - loss: 0.6967 - accuracy:
0.5205 - val_loss: 0.6866 - val_accuracy: 0.5730
Epoch 3/30
63/63 [==============================] - 4s 59ms/step - loss: 0.6908 - accuracy:
0.5410 - val_loss: 0.6922 - val_accuracy: 0.5010
Epoch 4/30
63/63 [==============================] - 5s 79ms/step - loss: 0.6767 - accuracy:
```

```
0.6070 - val_loss: 0.6622 - val_accuracy: 0.6150
Epoch 5/30
63/63 [==============================] - 6s 89ms/step - loss: 0.6476 - accuracy:
0.6345 - val_loss: 0.6564 - val_accuracy: 0.6210
Epoch 6/30
63/63 [==============================] - 4s 59ms/step - loss: 0.6248 - accuracy:
0.6610 - val_loss: 0.6433 - val_accuracy: 0.6300
Epoch 7/30
63/63 [==============================] - 7s 110ms/step - loss: 0.5890 -
accuracy: 0.6930 - val_loss: 0.6144 - val_accuracy: 0.6590
Epoch 8/30
63/63 [==============================] - 4s 58ms/step - loss: 0.5901 - accuracy:
0.6825 - val_loss: 0.7780 - val_accuracy: 0.6200
Epoch 9/30
63/63 [==============================] - 4s 60ms/step - loss: 0.5602 - accuracy:
0.7085 - val_loss: 0.6061 - val_accuracy: 0.6770
Epoch 10/30
63/63 [==============================] - 6s 87ms/step - loss: 0.5277 - accuracy:
0.7325 - val_loss: 0.5595 - val_accuracy: 0.7180
Epoch 11/30
63/63 [==============================] - 6s 83ms/step - loss: 0.5016 - accuracy:
0.7645 - val_loss: 0.6106 - val_accuracy: 0.6950
Epoch 12/30
63/63 [==============================] - 4s 57ms/step - loss: 0.4562 - accuracy:
0.7770 - val_loss: 0.5892 - val_accuracy: 0.6910
Epoch 13/30
63/63 [==============================] - 4s 61ms/step - loss: 0.4259 - accuracy:
0.7995 - val_loss: 0.5687 - val_accuracy: 0.7450
Epoch 14/30
63/63 [==============================] - 7s 111ms/step - loss: 0.3854 -
accuracy: 0.8265 - val_loss: 0.5532 - val_accuracy: 0.7380
Epoch 15/30
63/63 [==============================] - 4s 61ms/step - loss: 0.3366 - accuracy:
0.8520 - val_loss: 0.6629 - val_accuracy: 0.7450
Epoch 16/30
63/63 [==============================] - 5s 68ms/step - loss: 0.2951 - accuracy:
0.8720 - val_loss: 0.7394 - val_accuracy: 0.7280
Epoch 17/30
63/63 [==============================] - 7s 99ms/step - loss: 0.2670 - accuracy:
0.8845 - val_loss: 0.6280 - val_accuracy: 0.7300
Epoch 18/30
63/63 [==============================] - 4s 59ms/step - loss: 0.2183 - accuracy:
0.9125 - val_loss: 0.7169 - val_accuracy: 0.7580
Epoch 19/30
63/63 [==============================] - 4s 61ms/step - loss: 0.1816 - accuracy:
0.9245 - val_loss: 1.0737 - val_accuracy: 0.6700
Epoch 20/30
63/63 [==============================] - 7s 101ms/step - loss: 0.1563 -
```

```
accuracy: 0.9420 - val_loss: 0.8744 - val_accuracy: 0.7420
Epoch 21/30
63/63 [==============================] - 4s 62ms/step - loss: 0.1253 - accuracy:
0.9565 - val_loss: 1.1023 - val_accuracy: 0.7280
Epoch 22/30
63/63 [==============================] - 4s 65ms/step - loss: 0.1093 - accuracy:
0.9605 - val_loss: 1.0383 - val_accuracy: 0.7550
Epoch 23/30
63/63 [==============================] - 6s 90ms/step - loss: 0.1114 - accuracy:
0.9585 - val_loss: 0.9574 - val_accuracy: 0.7510
Epoch 24/30
63/63 [==============================] - 4s 58ms/step - loss: 0.0992 - accuracy:
0.9680 - val_loss: 1.1496 - val_accuracy: 0.7740
Epoch 25/30
63/63 [==============================] - 7s 103ms/step - loss: 0.0758 -
accuracy: 0.9710 - val_loss: 1.2123 - val_accuracy: 0.7410
Epoch 26/30
63/63 [==============================] - 5s 68ms/step - loss: 0.0550 - accuracy:
0.9785 - val_loss: 1.4881 - val_accuracy: 0.7410
Epoch 27/30
63/63 [==============================] - 4s 58ms/step - loss: 0.0526 - accuracy:
0.9805 - val_loss: 1.3756 - val_accuracy: 0.7450
Epoch 28/30
63/63 [==============================] - 4s 68ms/step - loss: 0.0676 - accuracy:
0.9760 - val_loss: 1.3157 - val_accuracy: 0.7570
Epoch 29/30
63/63 [==============================] - 6s 93ms/step - loss: 0.0743 - accuracy:
0.9740 - val_loss: 1.7772 - val_accuracy: 0.7400
Epoch 30/30
63/63 [==============================] - 4s 59ms/step - loss: 0.0666 - accuracy:
0.9770 - val_loss: 1.4717 - val_accuracy: 0.7550
```

[27]:

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

```
32/32 [==============================] - 1s 28ms/step - loss: 0.5331 - accuracy:
0.7580
Test accuracy: 0.758
```

MODEL 4 - DATA AUGMENTATION AND DROPOUT METHOD:

[28]:

```python
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)
```

[29]:
```
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_4 = keras.Model(inputs=inputs, outputs=outputs)

Model_4.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

[30]:
```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_with_augmentation_dropout.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = Model_4.fit(
    train_dataset,
    epochs=30,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/30
63/63 [==============================] - 9s 88ms/step - loss: 0.6996 - accuracy:
0.4925 - val_loss: 0.6928 - val_accuracy: 0.5790
Epoch 2/30
63/63 [==============================] - 4s 59ms/step - loss: 0.6991 - accuracy:
0.5060 - val_loss: 0.6949 - val_accuracy: 0.5000
Epoch 3/30
63/63 [==============================] - 7s 105ms/step - loss: 0.6927 -
accuracy: 0.5185 - val_loss: 0.6902 - val_accuracy: 0.5250
Epoch 4/30
63/63 [==============================] - 4s 62ms/step - loss: 0.6862 - accuracy:
```

```
0.5890 - val_loss: 0.6873 - val_accuracy: 0.5150
Epoch 5/30
63/63 [==============================] - 4s 64ms/step - loss: 0.6687 - accuracy:
0.5905 - val_loss: 0.6276 - val_accuracy: 0.6630
Epoch 6/30
63/63 [==============================] - 7s 105ms/step - loss: 0.6337 -
accuracy: 0.6445 - val_loss: 0.6243 - val_accuracy: 0.6660
Epoch 7/30
63/63 [==============================] - 4s 62ms/step - loss: 0.6330 - accuracy:
0.6560 - val_loss: 0.6132 - val_accuracy: 0.6790
Epoch 8/30
63/63 [==============================] - 6s 88ms/step - loss: 0.6110 - accuracy:
0.6720 - val_loss: 0.7306 - val_accuracy: 0.6030
Epoch 9/30
63/63 [==============================] - 6s 87ms/step - loss: 0.5976 - accuracy:
0.6820 - val_loss: 0.5970 - val_accuracy: 0.6800
Epoch 10/30
63/63 [==============================] - 4s 59ms/step - loss: 0.5941 - accuracy:
0.6825 - val_loss: 0.6032 - val_accuracy: 0.6760
Epoch 11/30
63/63 [==============================] - 4s 65ms/step - loss: 0.5952 - accuracy:
0.6895 - val_loss: 0.6653 - val_accuracy: 0.6330
Epoch 12/30
63/63 [==============================] - 8s 114ms/step - loss: 0.5765 -
accuracy: 0.7080 - val_loss: 0.6308 - val_accuracy: 0.6560
Epoch 13/30
63/63 [==============================] - 4s 62ms/step - loss: 0.5711 - accuracy:
0.7115 - val_loss: 0.6257 - val_accuracy: 0.6680
Epoch 14/30
63/63 [==============================] - 6s 98ms/step - loss: 0.5556 - accuracy:
0.7300 - val_loss: 0.5580 - val_accuracy: 0.7120
Epoch 15/30
63/63 [==============================] - 5s 71ms/step - loss: 0.5391 - accuracy:
0.7280 - val_loss: 0.5752 - val_accuracy: 0.7050
Epoch 16/30
63/63 [==============================] - 5s 74ms/step - loss: 0.5421 - accuracy:
0.7310 - val_loss: 0.5564 - val_accuracy: 0.7320
Epoch 17/30
63/63 [==============================] - 7s 100ms/step - loss: 0.5264 -
accuracy: 0.7375 - val_loss: 0.5501 - val_accuracy: 0.7270
Epoch 18/30
63/63 [==============================] - 4s 61ms/step - loss: 0.5257 - accuracy:
0.7425 - val_loss: 0.5098 - val_accuracy: 0.7470
Epoch 19/30
63/63 [==============================] - 4s 61ms/step - loss: 0.5178 - accuracy:
0.7450 - val_loss: 0.5056 - val_accuracy: 0.7590
Epoch 20/30
63/63 [==============================] - 6s 95ms/step - loss: 0.5108 - accuracy:
```

```
0.7535 - val_loss: 0.5252 - val_accuracy: 0.7390
Epoch 21/30
63/63 [==============================] - 4s 59ms/step - loss: 0.5104 - accuracy:
0.7530 - val_loss: 0.5707 - val_accuracy: 0.6960
Epoch 22/30
63/63 [==============================] - 5s 84ms/step - loss: 0.5002 - accuracy:
0.7585 - val_loss: 0.6116 - val_accuracy: 0.6720
Epoch 23/30
63/63 [==============================] - 6s 90ms/step - loss: 0.4796 - accuracy:
0.7755 - val_loss: 0.5508 - val_accuracy: 0.7300
Epoch 24/30
63/63 [==============================] - 4s 61ms/step - loss: 0.5021 - accuracy:
0.7625 - val_loss: 0.4857 - val_accuracy: 0.7620
Epoch 25/30
63/63 [==============================] - 4s 62ms/step - loss: 0.4656 - accuracy:
0.7850 - val_loss: 0.5184 - val_accuracy: 0.7560
Epoch 26/30
63/63 [==============================] - 7s 102ms/step - loss: 0.4456 -
accuracy: 0.7945 - val_loss: 0.4910 - val_accuracy: 0.7600
Epoch 27/30
63/63 [==============================] - 4s 60ms/step - loss: 0.4371 - accuracy:
0.8055 - val_loss: 0.5462 - val_accuracy: 0.7620
Epoch 28/30
63/63 [==============================] - 4s 61ms/step - loss: 0.4358 - accuracy:
0.8010 - val_loss: 0.4393 - val_accuracy: 0.8010
Epoch 29/30
63/63 [==============================] - 7s 103ms/step - loss: 0.4318 -
accuracy: 0.7990 - val_loss: 0.4653 - val_accuracy: 0.7780
Epoch 30/30
63/63 [==============================] - 5s 68ms/step - loss: 0.4278 - accuracy:
0.8030 - val_loss: 0.4994 - val_accuracy: 0.7780
```

[31]:

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

```
32/32 [==============================] - 1s 32ms/step - loss: 0.4004 - accuracy:
0.8180
Test accuracy: 0.818
```

MODEL 5 - INCREASING THE TRAINING SAMPLE SIZE TO 5000, INCLUDING MAXPOOLING, DATA AUGMENTATION AND DROPOUT TECHNIQUE (DROPOUT RATE = 0.05)

[32]:

```python
from tensorflow.keras.utils import image_dataset_from_directory

make_subset("train_1", start_index=0, end_index=5000)
make_subset("validation_1", start_index=5000, end_index=5500)
make_subset("test_1", start_index=5500, end_index=6000)

train_dataset_1 = image_dataset_from_directory(
    new_base_dir / "train_1",
    image_size=(180, 180),
    batch_size=32)
validation_dataset_1 = image_dataset_from_directory(
    new_base_dir / "validation_1",
    image_size=(180, 180),
    batch_size=32)
test_dataset_1 = image_dataset_from_directory(
    new_base_dir / "test_1",
    image_size=(180, 180),
    batch_size=32)
```

```
Found 10000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
```

[33]:

```
#Defining a new convnet that includes image augmentation and dropout

inputs = keras.Input(shape=(180, 180, 3))
x = data_augmentation(inputs)
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)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
Model_5 = keras.Model(inputs=inputs, outputs=outputs)

Model_5.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

[34]:

```
from tensorflow import keras
from tensorflow.keras import layers
import matplotlib.pyplot as plt
from keras.callbacks import EarlyStopping
from keras import regularizers
```

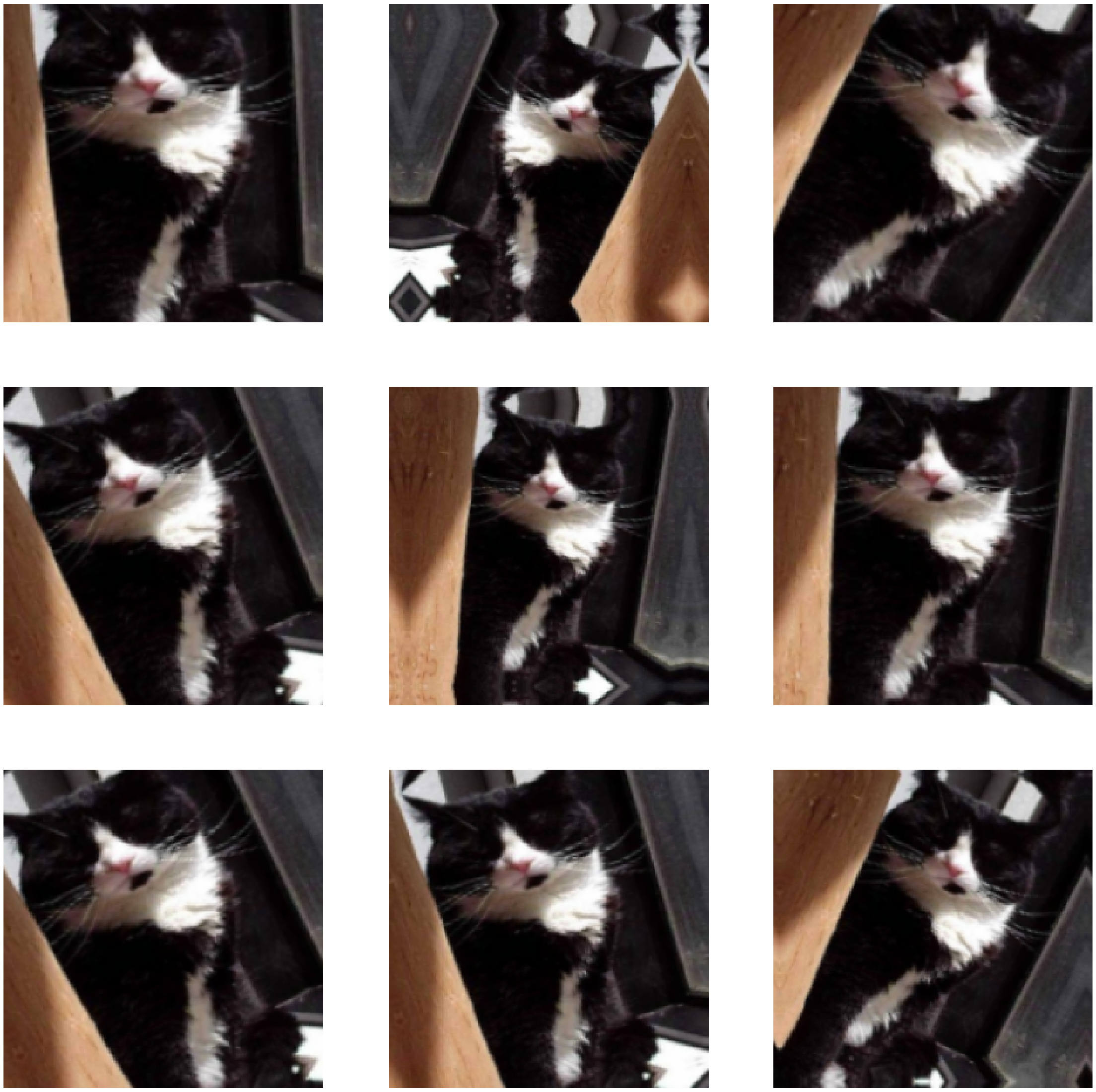
```
early_stopping_monitor = EarlyStopping(patience=10)
```

[35]:

```
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)
```

[36]:

```
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")
```



[37]:
```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch.keras",
        save_best_only=True,
        monitor="val_loss"), early_stopping_monitor
]
history = Model_5.fit(
    train_dataset_1,
    epochs=30,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/30
```

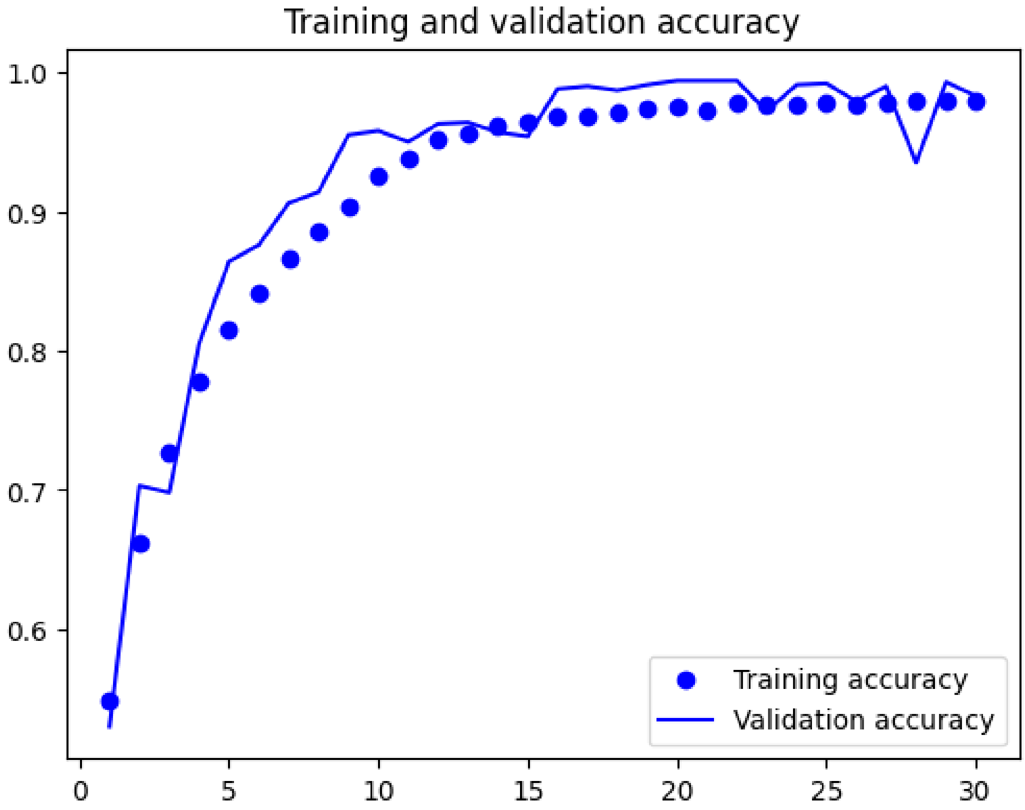
```
313/313 [==============================] - 24s 69ms/step - loss: 0.6958 -
accuracy: 0.5484 - val_loss: 0.6781 - val_accuracy: 0.5300
Epoch 2/30
313/313 [==============================] - 20s 61ms/step - loss: 0.6191 -
accuracy: 0.6621 - val_loss: 0.5799 - val_accuracy: 0.7030
Epoch 3/30
313/313 [==============================] - 17s 54ms/step - loss: 0.5420 -
accuracy: 0.7266 - val_loss: 0.5537 - val_accuracy: 0.6980
Epoch 4/30
313/313 [==============================] - 18s 56ms/step - loss: 0.4751 -
accuracy: 0.7779 - val_loss: 0.4254 - val_accuracy: 0.8050
Epoch 5/30
313/313 [==============================] - 18s 56ms/step - loss: 0.4124 -
accuracy: 0.8146 - val_loss: 0.3236 - val_accuracy: 0.8640
Epoch 6/30
313/313 [==============================] - 18s 57ms/step - loss: 0.3629 -
accuracy: 0.8415 - val_loss: 0.3336 - val_accuracy: 0.8760
Epoch 7/30
313/313 [==============================] - 18s 56ms/step - loss: 0.3144 -
accuracy: 0.8658 - val_loss: 0.2220 - val_accuracy: 0.9060
Epoch 8/30
313/313 [==============================] - 18s 57ms/step - loss: 0.2650 -
accuracy: 0.8862 - val_loss: 0.2131 - val_accuracy: 0.9140
Epoch 9/30
313/313 [==============================] - 18s 56ms/step - loss: 0.2324 -
accuracy: 0.9041 - val_loss: 0.1184 - val_accuracy: 0.9550
Epoch 10/30
313/313 [==============================] - 18s 55ms/step - loss: 0.1816 -
accuracy: 0.9260 - val_loss: 0.1209 - val_accuracy: 0.9580
Epoch 11/30
313/313 [==============================] - 18s 57ms/step - loss: 0.1597 -
accuracy: 0.9377 - val_loss: 0.1316 - val_accuracy: 0.9500
Epoch 12/30
313/313 [==============================] - 19s 59ms/step - loss: 0.1248 -
accuracy: 0.9522 - val_loss: 0.0904 - val_accuracy: 0.9630
Epoch 13/30
313/313 [==============================] - 18s 57ms/step - loss: 0.1196 -
accuracy: 0.9560 - val_loss: 0.0893 - val_accuracy: 0.9640
Epoch 14/30
313/313 [==============================] - 18s 58ms/step - loss: 0.1066 -
accuracy: 0.9618 - val_loss: 0.1361 - val_accuracy: 0.9570
Epoch 15/30
313/313 [==============================] - 18s 57ms/step - loss: 0.1022 -
accuracy: 0.9640 - val_loss: 0.1278 - val_accuracy: 0.9540
Epoch 16/30
313/313 [==============================] - 18s 57ms/step - loss: 0.0942 -
accuracy: 0.9685 - val_loss: 0.0360 - val_accuracy: 0.9880
Epoch 17/30
```

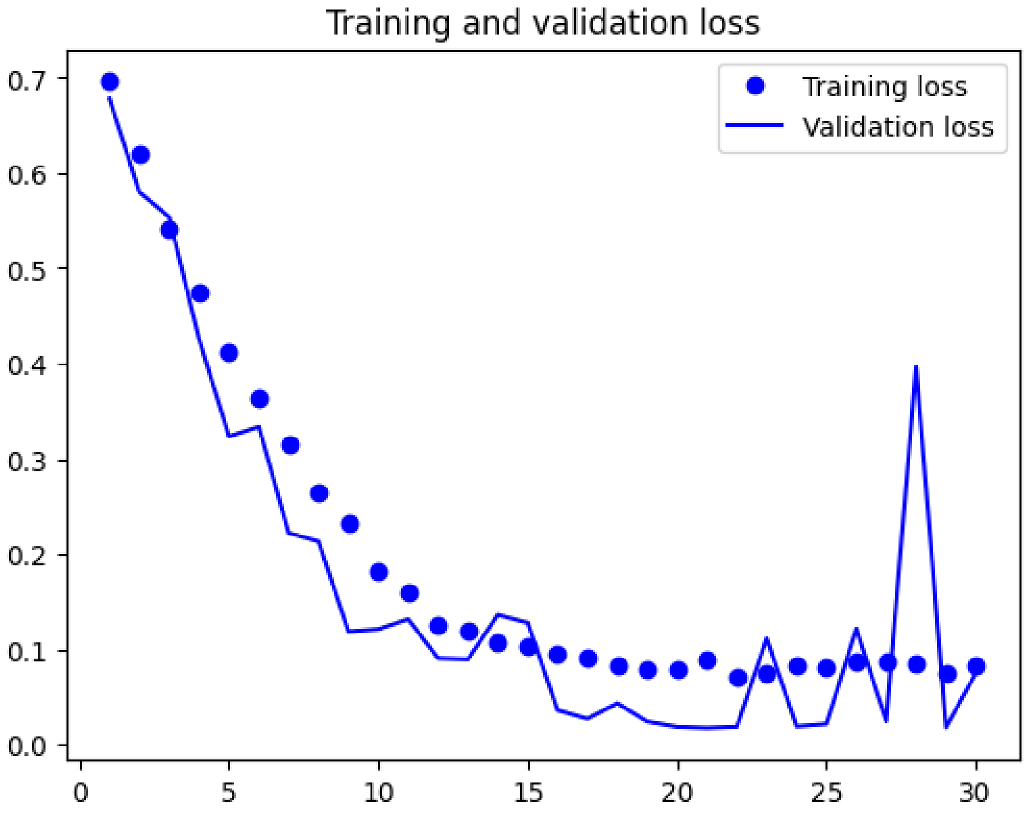
```
313/313 [==============================] - 18s 57ms/step - loss: 0.0903 -
accuracy: 0.9687 - val_loss: 0.0272 - val_accuracy: 0.9900
Epoch 18/30
313/313 [==============================] - 18s 57ms/step - loss: 0.0834 -
accuracy: 0.9712 - val_loss: 0.0432 - val_accuracy: 0.9870
Epoch 19/30
313/313 [==============================] - 18s 57ms/step - loss: 0.0787 -
accuracy: 0.9743 - val_loss: 0.0241 - val_accuracy: 0.9910
Epoch 20/30
313/313 [==============================] - 18s 57ms/step - loss: 0.0792 -
accuracy: 0.9747 - val_loss: 0.0185 - val_accuracy: 0.9940
Epoch 21/30
313/313 [==============================] - 18s 57ms/step - loss: 0.0882 -
accuracy: 0.9730 - val_loss: 0.0174 - val_accuracy: 0.9940
Epoch 22/30
313/313 [==============================] - 18s 56ms/step - loss: 0.0717 -
accuracy: 0.9781 - val_loss: 0.0186 - val_accuracy: 0.9940
Epoch 23/30
313/313 [==============================] - 18s 56ms/step - loss: 0.0753 -
accuracy: 0.9767 - val_loss: 0.1112 - val_accuracy: 0.9720
Epoch 24/30
313/313 [==============================] - 18s 57ms/step - loss: 0.0829 -
accuracy: 0.9769 - val_loss: 0.0190 - val_accuracy: 0.9910
Epoch 25/30
313/313 [==============================] - 18s 57ms/step - loss: 0.0810 -
accuracy: 0.9783 - val_loss: 0.0216 - val_accuracy: 0.9920
Epoch 26/30
313/313 [==============================] - 19s 58ms/step - loss: 0.0862 -
accuracy: 0.9761 - val_loss: 0.1217 - val_accuracy: 0.9790
Epoch 27/30
313/313 [==============================] - 17s 54ms/step - loss: 0.0878 -
accuracy: 0.9776 - val_loss: 0.0245 - val_accuracy: 0.9900
Epoch 28/30
313/313 [==============================] - 18s 57ms/step - loss: 0.0847 -
accuracy: 0.9791 - val_loss: 0.3962 - val_accuracy: 0.9350
Epoch 29/30
313/313 [==============================] - 18s 57ms/step - loss: 0.0756 -
accuracy: 0.9794 - val_loss: 0.0177 - val_accuracy: 0.9930
Epoch 30/30
313/313 [==============================] - 18s 56ms/step - loss: 0.0826 -
accuracy: 0.9796 - val_loss: 0.0750 - val_accuracy: 0.9830
```

[38]:

```
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)
```

```
plt.plot(epochs, accuracy, "bo", label="Training accuracy")
plt.plot(epochs, val_accuracy, "b", label="Validation accuracy")
plt.title("Training and validation accuracy")
plt.legend()
plt.figure()
plt.plot(epochs, loss, "bo", label="Training loss")
plt.plot(epochs, val_loss, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.legend()
plt.show()
```





[39]:

```
test_model = keras.models.load_model("convnet_from_scratch.keras")
test_loss, test_acc = test_model.evaluate(test_dataset_1)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 2s 30ms/step - loss: 0.6666 - accuracy:
0.8870
Test accuracy: 0.887
```

MODEL 6 - INCREASING THE TRAINING SAMPLE TO 10000

[40]:

```
from tensorflow.keras.utils import image_dataset_from_directory


make_subset("train_4", start_index=0, end_index=10000)
make_subset("validation_4", start_index=10000, end_index=10500)
make_subset("test_4", start_index=10500, end_index=11000)

train_dataset_4 = image_dataset_from_directory(
    new_base_dir / "train_4",
    image_size=(180, 180),
    batch_size=32)
validation_dataset_4 = image_dataset_from_directory(
    new_base_dir / "validation_4",
    image_size=(180, 180),
    batch_size=32)
test_dataset_4 = image_dataset_from_directory(
    new_base_dir / "test_4",
    image_size=(180, 180),
    batch_size=32)
```

```
Found 20000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
```

[41]:

```
inputs = keras.Input(shape=(180, 180, 3))
x = data_augmentation(inputs)
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)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
Model_6 = keras.Model(inputs=inputs, outputs=outputs)

Model_6.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

[42]:

```
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)
```

[43]:

```
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")
```

[44]:

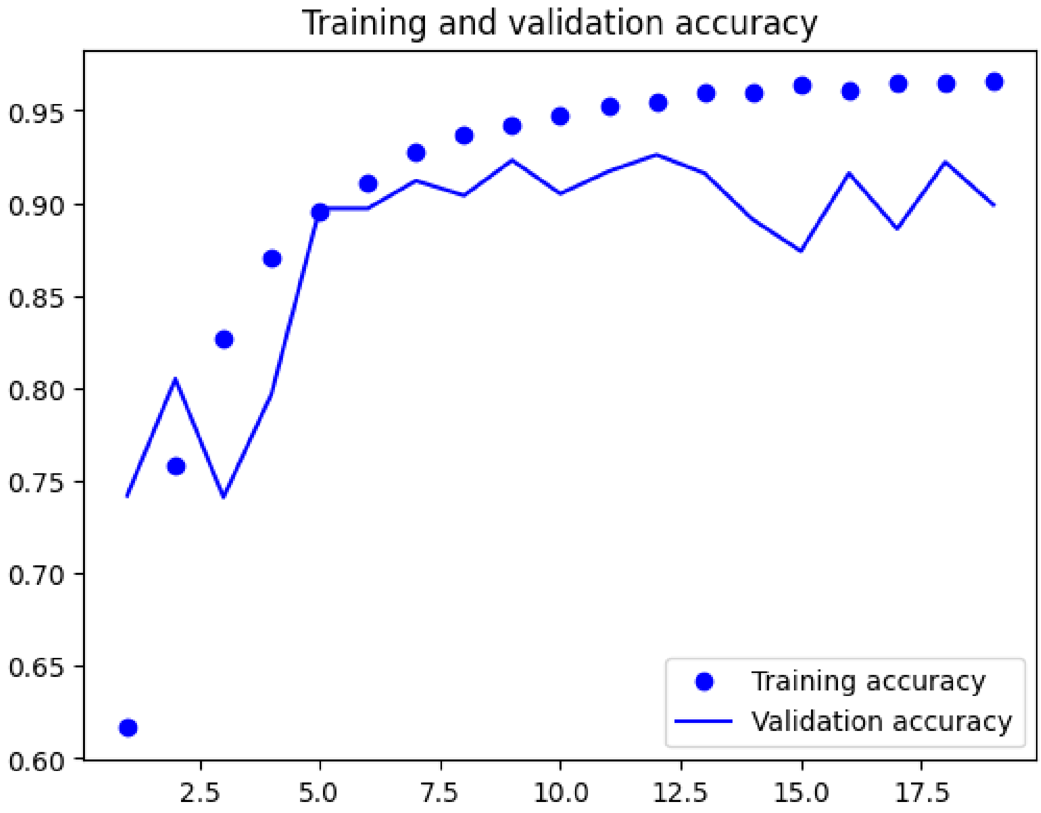
```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch.keras",
        save_best_only=True,
        monitor="val_loss"), early_stopping_monitor
]
history = Model_6.fit(
    train_dataset_4,
    epochs=30,
    validation_data=validation_dataset_4,
    callbacks=callbacks)
```

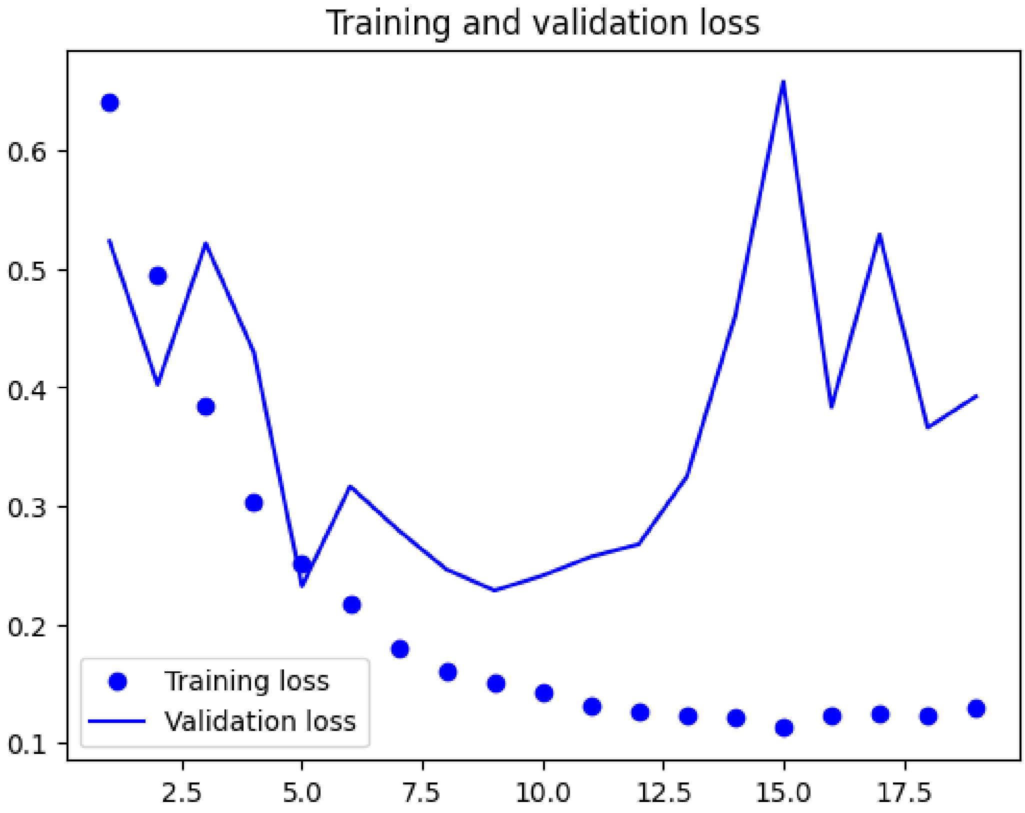
```
Epoch 1/30
625/625 [==============================] - 36s 52ms/step - loss: 0.6409 -
accuracy: 0.6166 - val_loss: 0.5234 - val_accuracy: 0.7420
Epoch 2/30
625/625 [==============================] - 34s 54ms/step - loss: 0.4952 -
accuracy: 0.7580 - val_loss: 0.4023 - val_accuracy: 0.8050
Epoch 3/30
625/625 [==============================] - 34s 54ms/step - loss: 0.3844 -
accuracy: 0.8268 - val_loss: 0.5215 - val_accuracy: 0.7410
Epoch 4/30
625/625 [==============================] - 34s 55ms/step - loss: 0.3033 -
accuracy: 0.8701 - val_loss: 0.4296 - val_accuracy: 0.7970
Epoch 5/30
625/625 [==============================] - 35s 55ms/step - loss: 0.2507 -
accuracy: 0.8957 - val_loss: 0.2320 - val_accuracy: 0.8970
Epoch 6/30
625/625 [==============================] - 34s 55ms/step - loss: 0.2168 -
accuracy: 0.9107 - val_loss: 0.3164 - val_accuracy: 0.8970
Epoch 7/30
625/625 [==============================] - 35s 55ms/step - loss: 0.1805 -
accuracy: 0.9277 - val_loss: 0.2794 - val_accuracy: 0.9120
Epoch 8/30
625/625 [==============================] - 34s 55ms/step - loss: 0.1611 -
accuracy: 0.9373 - val_loss: 0.2463 - val_accuracy: 0.9040
Epoch 9/30
625/625 [==============================] - 34s 55ms/step - loss: 0.1506 -
accuracy: 0.9417 - val_loss: 0.2284 - val_accuracy: 0.9230
Epoch 10/30
625/625 [==============================] - 35s 55ms/step - loss: 0.1425 -
accuracy: 0.9473 - val_loss: 0.2410 - val_accuracy: 0.9050
Epoch 11/30
625/625 [==============================] - 33s 53ms/step - loss: 0.1309 -
accuracy: 0.9523 - val_loss: 0.2569 - val_accuracy: 0.9170
Epoch 12/30
625/625 [==============================] - 34s 53ms/step - loss: 0.1258 -
accuracy: 0.9549 - val_loss: 0.2676 - val_accuracy: 0.9260
Epoch 13/30
625/625 [==============================] - 34s 55ms/step - loss: 0.1228 -
accuracy: 0.9595 - val_loss: 0.3248 - val_accuracy: 0.9160
Epoch 14/30
625/625 [==============================] - 34s 53ms/step - loss: 0.1209 -
accuracy: 0.9602 - val_loss: 0.4604 - val_accuracy: 0.8910
Epoch 15/30
625/625 [==============================] - 34s 55ms/step - loss: 0.1124 -
accuracy: 0.9642 - val_loss: 0.6580 - val_accuracy: 0.8740
Epoch 16/30
625/625 [==============================] - 37s 58ms/step - loss: 0.1231 -
accuracy: 0.9612 - val_loss: 0.3833 - val_accuracy: 0.9160
```

```
Epoch 17/30
625/625 [==============================] - 36s 58ms/step - loss: 0.1246 -
accuracy: 0.9650 - val_loss: 0.5290 - val_accuracy: 0.8860
Epoch 18/30
625/625 [==============================] - 37s 59ms/step - loss: 0.1229 -
accuracy: 0.9649 - val_loss: 0.3659 - val_accuracy: 0.9220
Epoch 19/30
625/625 [==============================] - 35s 56ms/step - loss: 0.1294 -
accuracy: 0.9654 - val_loss: 0.3923 - val_accuracy: 0.8990
```

[45]:

```
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)
plt.plot(epochs, accuracy, "bo", label="Training accuracy")
plt.plot(epochs, val_accuracy, "b", label="Validation accuracy")
plt.title("Training and validation accuracy")
plt.legend()
plt.figure()
plt.plot(epochs, loss, "bo", label="Training loss")
plt.plot(epochs, val_loss, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.legend()
plt.show()
```


Training and validation accuracy
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
2.5
5.0
7.5
10.0
12.5
15.0
17.5
● Training accuracy
— Validation accuracy



[46]:
```
test_model = keras.models.load_model("convnet_from_scratch.keras")
test_loss, test_acc = test_model.evaluate(test_dataset_4)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 1s 32ms/step - loss: 0.3237 - accuracy:
0.8830
Test accuracy: 0.883
```

INSTANTIATING AND FREEZING THE VGG16 CONVOLUTIONAL BASE

PRE-TRAINED MODEL

MODEL 7 - SAMPLE SIZE OF 1000

[79]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

# Load pre-trained VGG16 model without the top layer
conv_base = keras.applications.VGG16(
    weights="imagenet",
    include_top=False,
    input_shape=(180, 180, 3))
```

```python
# Freeze convolutional base layers
conv_base.trainable = True
for layer in conv_base.layers[:-4]:
    layer.trainable = False

# Define data augmentation pipeline
data_augmentation = keras.Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.2),
])

# Define your model
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, activation='relu')(x)  # Add activation function
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs, outputs)

# Compile the model
model.compile(loss="binary_crossentropy",
              optimizer=keras.optimizers.RMSprop(learning_rate=1e-5),
              metrics=["accuracy"])

# Define callbacks
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="fine_tuning.h5",
        save_best_only=True,
        monitor="val_loss")
]

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

```
Epoch 1/30
63/63 [==============================] - ETA: 0s - loss: 3.1088 - accuracy:
```

```
0.6780
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103:
UserWarning: You are saving your model as an HDF5 file via `model.save()`. This
file format is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

```
63/63 [==============================] - 14s 187ms/step - loss: 3.1088 -
accuracy: 0.6780 - val_loss: 0.5563 - val_accuracy: 0.8880
Epoch 2/30
63/63 [==============================] - 14s 216ms/step - loss: 0.9146 -
accuracy: 0.8310 - val_loss: 0.3115 - val_accuracy: 0.9330
Epoch 3/30
63/63 [==============================] - 12s 180ms/step - loss: 0.5377 -
accuracy: 0.8710 - val_loss: 0.2285 - val_accuracy: 0.9380
Epoch 4/30
63/63 [==============================] - 11s 179ms/step - loss: 0.3763 -
accuracy: 0.9050 - val_loss: 0.1930 - val_accuracy: 0.9510
Epoch 5/30
63/63 [==============================] - 11s 169ms/step - loss: 0.2471 -
accuracy: 0.9260 - val_loss: 0.1460 - val_accuracy: 0.9540
Epoch 6/30
63/63 [==============================] - 11s 173ms/step - loss: 0.2122 -
accuracy: 0.9320 - val_loss: 0.1474 - val_accuracy: 0.9600
Epoch 7/30
63/63 [==============================] - 13s 198ms/step - loss: 0.1988 -
accuracy: 0.9390 - val_loss: 0.1725 - val_accuracy: 0.9610
Epoch 8/30
63/63 [==============================] - 12s 176ms/step - loss: 0.1541 -
accuracy: 0.9495 - val_loss: 0.1392 - val_accuracy: 0.9660
Epoch 9/30
63/63 [==============================] - 11s 166ms/step - loss: 0.1194 -
accuracy: 0.9560 - val_loss: 0.1691 - val_accuracy: 0.9700
Epoch 10/30
63/63 [==============================] - 11s 167ms/step - loss: 0.0983 -
accuracy: 0.9685 - val_loss: 0.1752 - val_accuracy: 0.9690
Epoch 11/30
63/63 [==============================] - 11s 175ms/step - loss: 0.0949 -
accuracy: 0.9565 - val_loss: 0.1763 - val_accuracy: 0.9720
Epoch 12/30
63/63 [==============================] - 11s 165ms/step - loss: 0.0917 -
accuracy: 0.9680 - val_loss: 0.1689 - val_accuracy: 0.9730
Epoch 13/30
63/63 [==============================] - 11s 170ms/step - loss: 0.0908 -
accuracy: 0.9660 - val_loss: 0.1488 - val_accuracy: 0.9750
Epoch 14/30
63/63 [==============================] - 11s 166ms/step - loss: 0.0644 -
accuracy: 0.9760 - val_loss: 0.1711 - val_accuracy: 0.9750
```

```
Epoch 15/30
63/63 [==============================] - 11s 172ms/step - loss: 0.0634 -
accuracy: 0.9750 - val_loss: 0.1955 - val_accuracy: 0.9720
Epoch 16/30
63/63 [==============================] - 11s 172ms/step - loss: 0.0730 -
accuracy: 0.9750 - val_loss: 0.1890 - val_accuracy: 0.9760
Epoch 17/30
63/63 [==============================] - 11s 168ms/step - loss: 0.0620 -
accuracy: 0.9785 - val_loss: 0.2087 - val_accuracy: 0.9740
Epoch 18/30
63/63 [==============================] - 11s 167ms/step - loss: 0.0516 -
accuracy: 0.9775 - val_loss: 0.1962 - val_accuracy: 0.9750
Epoch 19/30
63/63 [==============================] - 11s 174ms/step - loss: 0.0712 -
accuracy: 0.9800 - val_loss: 0.1875 - val_accuracy: 0.9780
Epoch 20/30
63/63 [==============================] - 13s 196ms/step - loss: 0.0341 -
accuracy: 0.9865 - val_loss: 0.2172 - val_accuracy: 0.9770
Epoch 21/30
63/63 [==============================] - 11s 169ms/step - loss: 0.0416 -
accuracy: 0.9860 - val_loss: 0.2199 - val_accuracy: 0.9750
Epoch 22/30
63/63 [==============================] - 11s 171ms/step - loss: 0.0322 -
accuracy: 0.9890 - val_loss: 0.2049 - val_accuracy: 0.9750
Epoch 23/30
63/63 [==============================] - 11s 167ms/step - loss: 0.0355 -
accuracy: 0.9860 - val_loss: 0.2074 - val_accuracy: 0.9780
Epoch 24/30
63/63 [==============================] - 11s 165ms/step - loss: 0.0279 -
accuracy: 0.9905 - val_loss: 0.1995 - val_accuracy: 0.9770
Epoch 25/30
63/63 [==============================] - 11s 173ms/step - loss: 0.0182 -
accuracy: 0.9930 - val_loss: 0.2368 - val_accuracy: 0.9780
Epoch 26/30
63/63 [==============================] - 12s 194ms/step - loss: 0.0263 -
accuracy: 0.9910 - val_loss: 0.2054 - val_accuracy: 0.9800
Epoch 27/30
63/63 [==============================] - 11s 171ms/step - loss: 0.0320 -
accuracy: 0.9900 - val_loss: 0.1906 - val_accuracy: 0.9780
Epoch 28/30
63/63 [==============================] - 13s 197ms/step - loss: 0.0170 -
accuracy: 0.9950 - val_loss: 0.2366 - val_accuracy: 0.9770
Epoch 29/30
63/63 [==============================] - 11s 170ms/step - loss: 0.0337 -
accuracy: 0.9880 - val_loss: 0.2344 - val_accuracy: 0.9750
Epoch 30/30
63/63 [==============================] - 11s 167ms/step - loss: 0.0243 -
accuracy: 0.9930 - val_loss: 0.2212 - val_accuracy: 0.9800
```

[80]:
```
# Load the saved model
test_model = keras.models.load_model("fine_tuning.h5")

# Evaluate the model on the test dataset
test_loss, test_acc = test_model.evaluate(test_dataset)

# Print the test accuracy
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 4s 94ms/step - loss: 0.1539 - accuracy:
0.9710
Test accuracy: 0.971
```

MODEL 8 - SAMPLE SIZE OF 5000

[81]:
```
conv_base  = keras.applications.vgg16.VGG16(
    weights="imagenet",
    include_top=False,
    input_shape=(180, 180, 3))
```

[82]:
```
conv_base  = keras.applications.vgg16.VGG16(
    weights="imagenet",
    include_top=False)

conv_base.trainable = True
for layer in conv_base.layers[:-4]:
    layer.trainable = False
```

[83]:
```
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)
```

[84]:
```
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_8 = keras.Model(inputs, outputs)
Model_8.compile(loss="binary_crossentropy",
              optimizer=keras.optimizers.RMSprop(learning_rate=1e-5),
              metrics=["accuracy"])
```

[86]:

```python
# Define the callbacks
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="fine_tuning2.h5",
        save_best_only=True,
        monitor="val_loss")
]

# Train Model_8
history = Model_8.fit(
    train_dataset_1,
    epochs=10,
    validation_data=validation_dataset_1,
    callbacks=callbacks
)
```

```
Epoch 1/10
313/313 [==============================] - 42s 132ms/step - loss: 0.1885 -
accuracy: 0.9370 - val_loss: 0.1056 - val_accuracy: 0.9700
Epoch 2/10
313/313 [==============================] - 39s 125ms/step - loss: 0.1466 -
accuracy: 0.9528 - val_loss: 0.1224 - val_accuracy: 0.9700
Epoch 3/10
313/313 [==============================] - 40s 127ms/step - loss: 0.1133 -
accuracy: 0.9598 - val_loss: 0.1275 - val_accuracy: 0.9760
Epoch 4/10
313/313 [==============================] - 41s 128ms/step - loss: 0.0931 -
accuracy: 0.9695 - val_loss: 0.0996 - val_accuracy: 0.9760
Epoch 5/10
313/313 [==============================] - 40s 127ms/step - loss: 0.0935 -
accuracy: 0.9679 - val_loss: 0.1162 - val_accuracy: 0.9770
Epoch 6/10
313/313 [==============================] - 42s 132ms/step - loss: 0.0842 -
accuracy: 0.9699 - val_loss: 0.1113 - val_accuracy: 0.9730
Epoch 7/10
313/313 [==============================] - 42s 131ms/step - loss: 0.0725 -
accuracy: 0.9754 - val_loss: 0.0903 - val_accuracy: 0.9790
Epoch 8/10
313/313 [==============================] - 41s 129ms/step - loss: 0.0651 -
accuracy: 0.9759 - val_loss: 0.1203 - val_accuracy: 0.9770
Epoch 9/10
313/313 [==============================] - 39s 125ms/step - loss: 0.0578 -
accuracy: 0.9811 - val_loss: 0.1987 - val_accuracy: 0.9780
Epoch 10/10
313/313 [==============================] - 40s 126ms/step - loss: 0.0652 -
accuracy: 0.9801 - val_loss: 0.1305 - val_accuracy: 0.9770
```

[87]:

```python
# Load the saved model
test_model = keras.models.load_model("fine_tuning2.h5")

# Evaluate the model on the test dataset
test_loss, test_acc = test_model.evaluate(test_dataset_1)

# Print the test accuracy
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 4s 96ms/step - loss: 0.0723 - accuracy:
0.9810
Test accuracy: 0.981
```

MODEL 9 - SAMPLE SIZE OF 10000

[89]:

```python
conv_base  = keras.applications.vgg16.VGG16(
    weights="imagenet",
    include_top=False,
    input_shape=(180, 180, 3))
```

[90]:

```python
conv_base  = keras.applications.vgg16.VGG16(
    weights="imagenet",
    include_top=False)

conv_base.trainable = True
for layer in conv_base.layers[:-4]:
    layer.trainable = False
```

[91]:

```python
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)
```

[92]:

```python
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_9 = keras.Model(inputs, outputs)
Model_9.compile(loss="binary_crossentropy",
              optimizer=keras.optimizers.RMSprop(learning_rate=1e-5),
              metrics=["accuracy"])
```

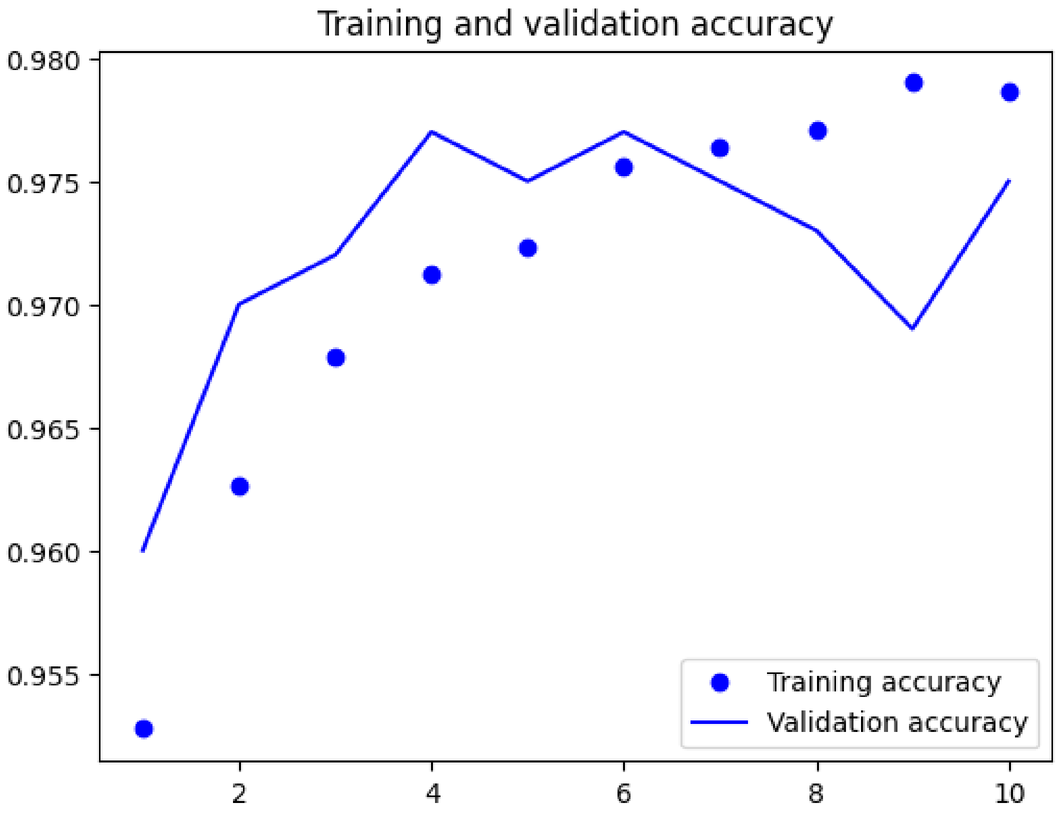
[94]:

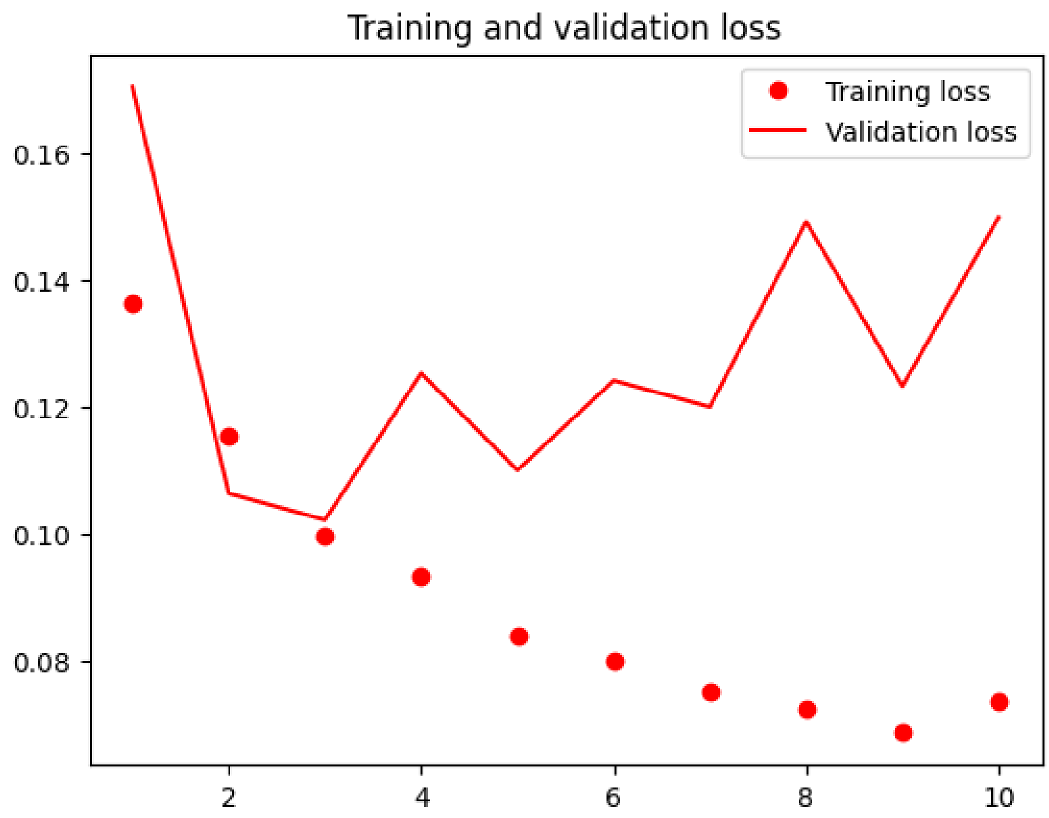
```
# Define the callbacks
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="fine_tuning3.h5",
        save_best_only=True,
        monitor="val_loss")
]

# Train Model_9
history = Model_9.fit(
    train_dataset_4,
    epochs=10,
    validation_data=validation_dataset_4,
    callbacks=callbacks
)
```

```
Epoch 1/10
625/625 [==============================] - 77s 123ms/step - loss: 0.1364 -
accuracy: 0.9528 - val_loss: 0.1705 - val_accuracy: 0.9600
Epoch 2/10
625/625 [==============================] - 77s 123ms/step - loss: 0.1156 -
accuracy: 0.9626 - val_loss: 0.1063 - val_accuracy: 0.9700
Epoch 3/10
625/625 [==============================] - 76s 121ms/step - loss: 0.0998 -
accuracy: 0.9679 - val_loss: 0.1021 - val_accuracy: 0.9720
Epoch 4/10
625/625 [==============================] - 76s 121ms/step - loss: 0.0933 -
accuracy: 0.9712 - val_loss: 0.1253 - val_accuracy: 0.9770
Epoch 5/10
625/625 [==============================] - 76s 121ms/step - loss: 0.0838 -
accuracy: 0.9723 - val_loss: 0.1100 - val_accuracy: 0.9750
Epoch 6/10
625/625 [==============================] - 76s 121ms/step - loss: 0.0800 -
accuracy: 0.9756 - val_loss: 0.1241 - val_accuracy: 0.9770
Epoch 7/10
625/625 [==============================] - 78s 124ms/step - loss: 0.0750 -
accuracy: 0.9764 - val_loss: 0.1200 - val_accuracy: 0.9750
Epoch 8/10
625/625 [==============================] - 76s 121ms/step - loss: 0.0723 -
accuracy: 0.9771 - val_loss: 0.1492 - val_accuracy: 0.9730
Epoch 9/10
625/625 [==============================] - 76s 121ms/step - loss: 0.0685 -
accuracy: 0.9790 - val_loss: 0.1232 - val_accuracy: 0.9690
Epoch 10/10
625/625 [==============================] - 78s 124ms/step - loss: 0.0735 -
accuracy: 0.9786 - val_loss: 0.1499 - val_accuracy: 0.9750
```

[95]:

```
import matplotlib.pyplot as plt
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)
plt.plot(epochs, accuracy, "bo", label="Training accuracy")
plt.plot(epochs, val_accuracy, "b", label="Validation accuracy")
plt.title("Training and validation accuracy")
plt.legend()
plt.figure()
plt.plot(epochs, loss, "ro", label="Training loss")
plt.plot(epochs, val_loss, "r", label="Validation loss")
plt.title("Training and validation loss")
plt.legend()
plt.show()
```





[97]:

```
# Load the saved model
test_model_9 = keras.models.load_model("fine_tuning3.h5")

# Evaluate the model on the test dataset
test_loss, test_acc = test_model_9.evaluate(test_dataset_4)

# Print the test accuracy
print(f"Test accuracy: {test_acc:.3f}")
```

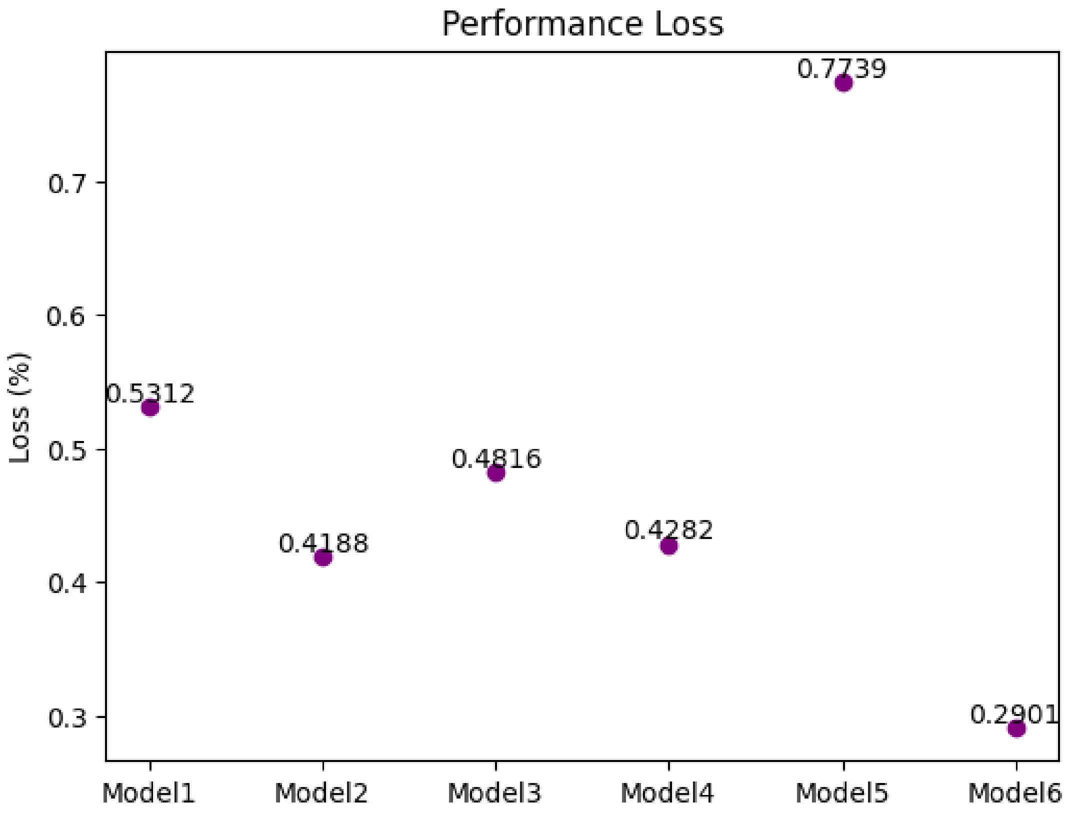
```
32/32 [==============================] - 3s 94ms/step - loss: 0.1100 - accuracy:
0.9750
Test accuracy: 0.975
```

SCRATCH MODELS:

[98]:

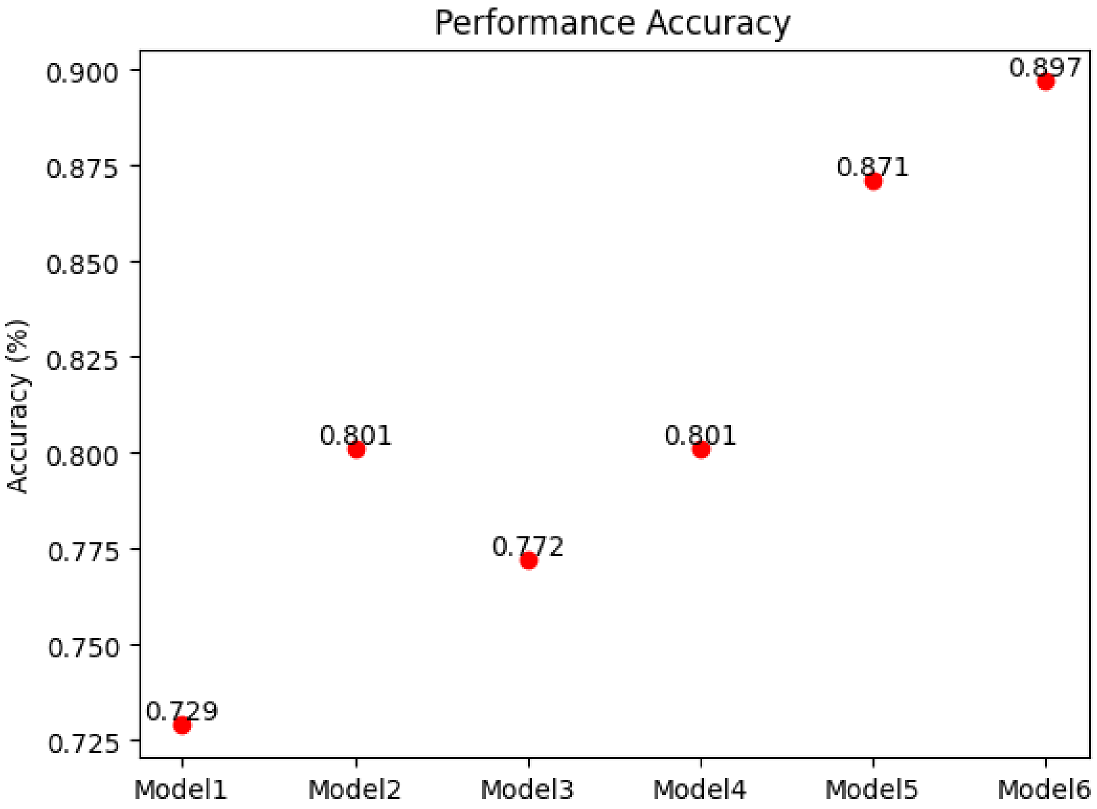
```
loss_dict = {'Model1': 0.5312, 'Model2': 0.4188, 'Model3': 0.4816, 'Model4': 0.
 ↪4282,
             'Model5': 0.7739, 'Model6': 0.2901}

# Get model names and loss values as separate lists
models = list(loss_dict.keys())
losses = list(loss_dict.values())
```

```
# Plot the scatter plot with labels
plt.scatter(models, losses, color='purple')
plt.title('Performance Loss')
plt.ylabel('Loss (%)')

for (xi, yi) in zip(models, losses):
    plt.text(xi, yi, str(yi), va='bottom', ha='center')
```



[99]:

```
import matplotlib.pyplot as plt

# Create a dictionary with model names as keys and accuracy values as values
acc_dict = {'Model1': 0.729, 'Model2': 0.801, 'Model3': 0.772, 'Model4': 0.801,
            'Model5': 0.871, 'Model6': 0.897}

# Get model names and accuracy values as separate lists
models = list(acc_dict.keys())
accuracy = list(acc_dict.values())

# Plot the scatter plot with labels
plt.scatter(models, accuracy, color='red')
plt.title('Performance Accuracy')
plt.ylabel('Accuracy (%)')

for (xi, yi) in zip(models, accuracy):
  plt.text(xi, yi, str(yi), va='bottom', ha='center')

plt.show()
```

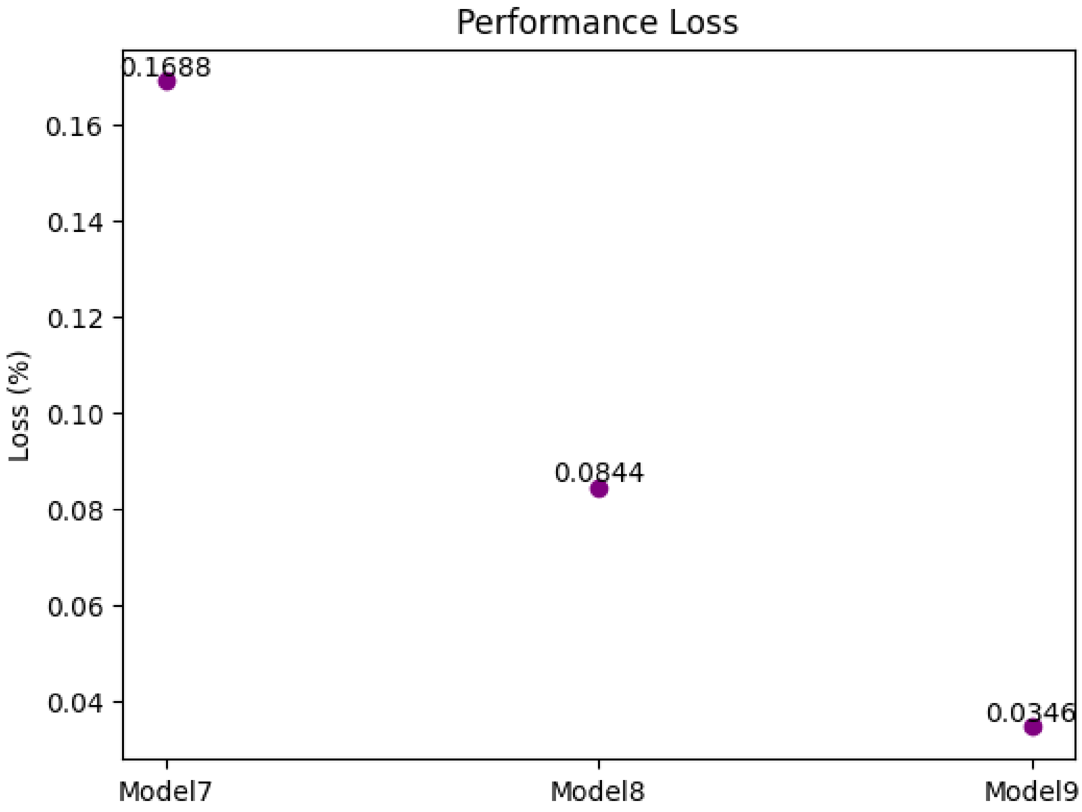


PRE-TRAINED MODELS:

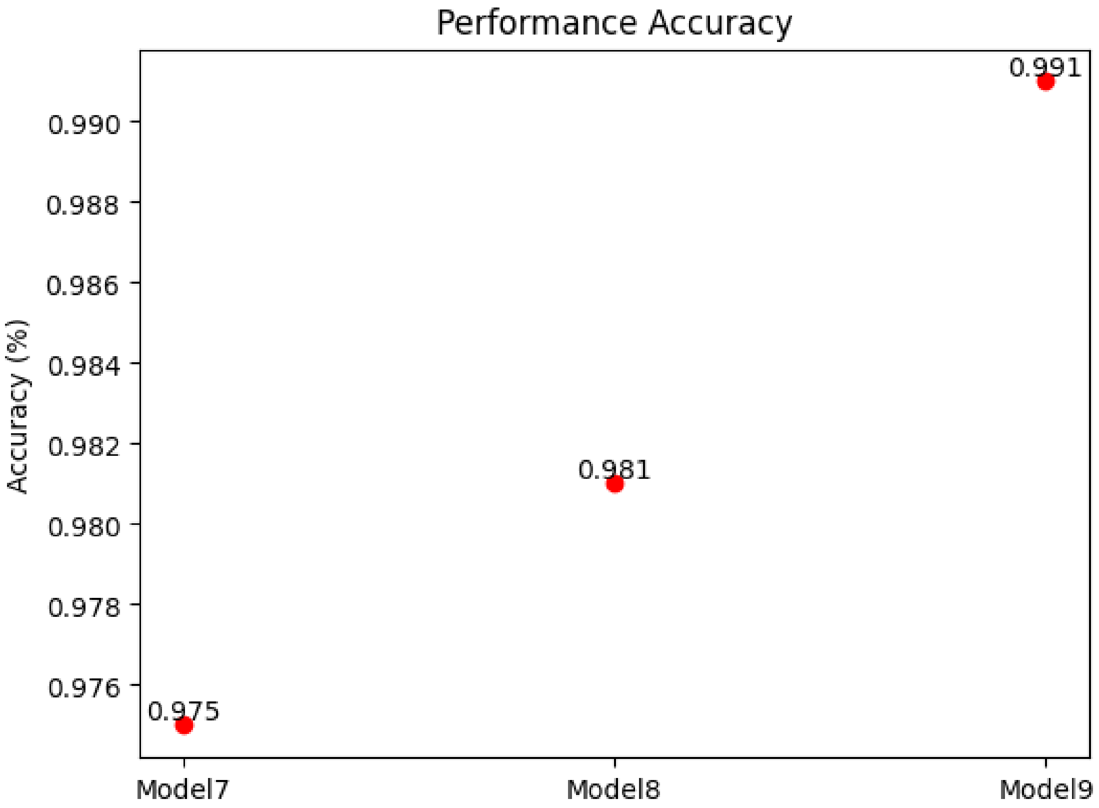
[100]:

```
loss_dict = {'Model7': 0.1688, 'Model8': 0.0844, 'Model9': 0.0346}

# Get model names and loss values as separate lists
models = list(loss_dict.keys())
losses = list(loss_dict.values())

# Plot the scatter plot with labels
plt.scatter(models, losses, color='purple')
plt.title('Performance Loss')
plt.ylabel('Loss (%)')
```

```
for (xi, yi) in zip(models, losses):
    plt.text(xi, yi, str(yi), va='bottom', ha='center')

plt.show()
```



[101]:
```
# Create a dictionary with model names as keys and accuracy values as values
acc_dict = {'Model7': 0.975, 'Model8': 0.981, 'Model9': 0.991}

# Get model names and accuracy values as separate lists
models = list(acc_dict.keys())
accuracy = list(acc_dict.values())

# Plot the scatter plot with labels
plt.scatter(models, accuracy, color='red')
plt.title('Performance Accuracy')
plt.ylabel('Accuracy (%)')

for (xi, yi) in zip(models, accuracy):
  plt.text(xi, yi, str(yi), va='bottom', ha='center')
```


Performance Accuracy
Accuracy (%)
0.990
0.988
0.986
0.984
0.982
0.980
0.978
0.976
0.975
0.981
0.991
Model7
Model8
Model9

[ ]: