Base_model

October 22, 2023

1 Assignment 2: Convolution

Yanxi Li, yli130@kent.edu

1.1 Preparation and the base model

1.1.1 Download the data

[1]:
```
#!unzip -qq '/fs/ess/PGS0333/BA_64061_KSU/data/dogs-vs-cats.zip'  # download␣
↪the data
```

[2]:
```
#!unzip -qq train.zip  # unzip the data
```

1.1.2 Load the library and split the data

[4]:
```
# import the library
import os, shutil, pathlib
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras import layers
```

[5]:
```
# copy and split to train, validation and test data
import os, shutil, pathlib

original_dir = pathlib.Path("train")
new_base_dir = pathlib.Path("cats_vs_dogs_small")  # store the smaller new␣
↪dataset

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)

# we create a new dataset containing three subsets
```

```
make_subset("train", start_index=0, end_index=1000) # training set with 1000␣
↪samples of each class
make_subset("validation", start_index=1000, end_index=1500) # validation set␣
↪with 500 samples of each class
make_subset("test", start_index=1500, end_index=2000)  # test set with 500␣
↪samples of each class
```

1.1.3 Build the basic model from scratch

[6]:
```
inputs = keras.Input(shape=(180, 180, 3))
x = layers.Rescaling(1./255)(inputs)  # rescale the input
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)   # use dropout
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)
```

[7]:
```
model.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 (MaxPooling2D) (None, 89, 89, 32)        0

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

_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 43, 43, 64)        0

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

_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 20, 20, 128)       0

_________________________________________________________________
```

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

_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 9, 9, 256)         0

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

_________________________________________________________________
flatten (Flatten)            (None, 12544)             0

_________________________________________________________________
dense (Dense)                (None, 1)                 12545
=================================================================
Total params: 991,041
Trainable params: 991,041
Non-trainable params: 0

_________________________________________________________________
```

Configuring the model

[8]:
```
model.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

Data preprocessing

[9]:
```
# use image_dataset_from_derectory to read the image
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.
```

Display shape of data and labels in batch

[10]:
```
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,)
```

Fit the model

[11]:

```
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 [==============================] - 8s 42ms/step - loss: 0.7373 - accuracy:
0.5215 - val_loss: 0.6916 - val_accuracy: 0.5120
Epoch 2/30
63/63 [==============================] - 2s 34ms/step - loss: 0.7132 - accuracy:
0.5200 - val_loss: 0.6861 - val_accuracy: 0.5220
Epoch 3/30
63/63 [==============================] - 2s 31ms/step - loss: 0.6920 - accuracy:
0.5635 - val_loss: 0.6627 - val_accuracy: 0.6050
Epoch 4/30
63/63 [==============================] - 2s 31ms/step - loss: 0.6565 - accuracy:
0.6240 - val_loss: 0.6462 - val_accuracy: 0.6180
Epoch 5/30
63/63 [==============================] - 2s 31ms/step - loss: 0.6225 - accuracy:
0.6705 - val_loss: 0.6257 - val_accuracy: 0.6390
Epoch 6/30
63/63 [==============================] - 2s 32ms/step - loss: 0.5859 - accuracy:
0.7040 - val_loss: 0.5532 - val_accuracy: 0.7070
Epoch 7/30
63/63 [==============================] - 2s 37ms/step - loss: 0.5570 - accuracy:
0.7320 - val_loss: 0.5845 - val_accuracy: 0.6980
Epoch 8/30
63/63 [==============================] - 2s 35ms/step - loss: 0.4993 - accuracy:
0.7550 - val_loss: 0.6331 - val_accuracy: 0.6790
Epoch 9/30
63/63 [==============================] - 2s 36ms/step - loss: 0.4571 - accuracy:
0.7770 - val_loss: 0.7906 - val_accuracy: 0.6440
Epoch 10/30
63/63 [==============================] - 2s 32ms/step - loss: 0.4029 - accuracy:
0.8035 - val_loss: 0.5844 - val_accuracy: 0.7130
Epoch 11/30
```

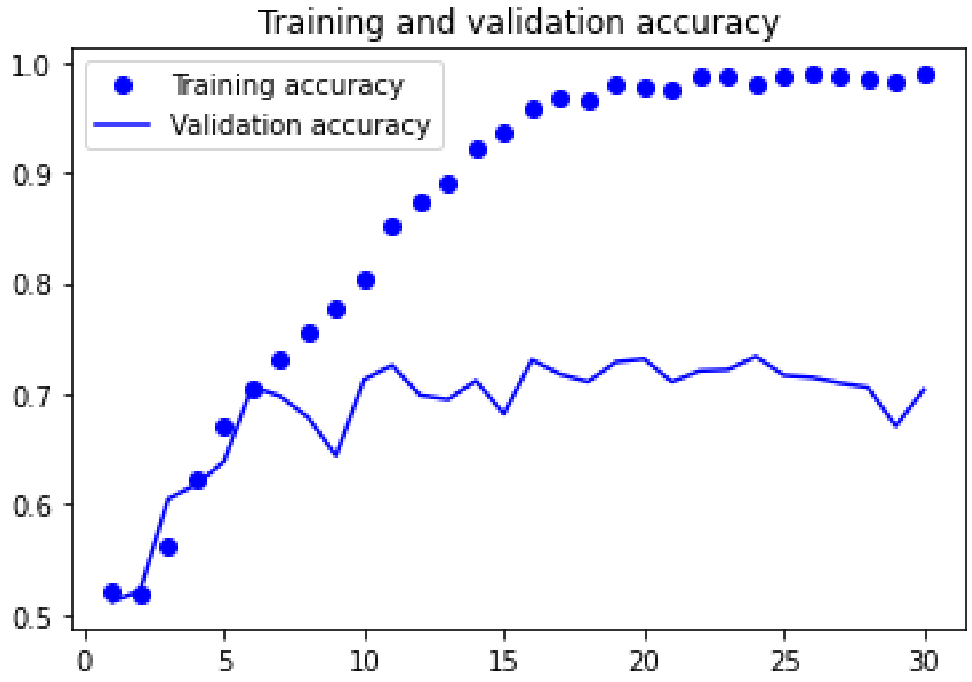
```
63/63 [==============================] - 2s 32ms/step - loss: 0.3493 - accuracy:
0.8510 - val_loss: 0.6550 - val_accuracy: 0.7260
Epoch 12/30
63/63 [==============================] - 2s 32ms/step - loss: 0.2881 - accuracy:
0.8730 - val_loss: 0.9093 - val_accuracy: 0.6990
Epoch 13/30
63/63 [==============================] - 2s 31ms/step - loss: 0.2541 - accuracy:
0.8895 - val_loss: 0.8314 - val_accuracy: 0.6950
Epoch 14/30
63/63 [==============================] - 2s 31ms/step - loss: 0.1918 - accuracy:
0.9210 - val_loss: 0.9504 - val_accuracy: 0.7120
Epoch 15/30
63/63 [==============================] - 2s 31ms/step - loss: 0.1685 - accuracy:
0.9365 - val_loss: 1.0503 - val_accuracy: 0.6820
Epoch 16/30
63/63 [==============================] - 2s 31ms/step - loss: 0.1123 - accuracy:
0.9585 - val_loss: 1.2045 - val_accuracy: 0.7310
Epoch 17/30
63/63 [==============================] - 2s 31ms/step - loss: 0.0949 - accuracy:
0.9685 - val_loss: 1.3385 - val_accuracy: 0.7180
Epoch 18/30
63/63 [==============================] - 2s 30ms/step - loss: 0.0908 - accuracy:
0.9645 - val_loss: 1.5259 - val_accuracy: 0.7110
Epoch 19/30
63/63 [==============================] - 2s 31ms/step - loss: 0.0910 - accuracy:
0.9800 - val_loss: 1.5021 - val_accuracy: 0.7290
Epoch 20/30
63/63 [==============================] - 2s 31ms/step - loss: 0.0691 - accuracy:
0.9780 - val_loss: 1.5248 - val_accuracy: 0.7320
Epoch 21/30
63/63 [==============================] - 2s 33ms/step - loss: 0.0798 - accuracy:
0.9760 - val_loss: 1.4735 - val_accuracy: 0.7110
Epoch 22/30
63/63 [==============================] - 2s 35ms/step - loss: 0.0528 - accuracy:
0.9865 - val_loss: 1.7064 - val_accuracy: 0.7210
Epoch 23/30
63/63 [==============================] - 2s 35ms/step - loss: 0.0404 - accuracy:
0.9870 - val_loss: 1.7444 - val_accuracy: 0.7220
Epoch 24/30
63/63 [==============================] - 2s 33ms/step - loss: 0.0597 - accuracy:
0.9795 - val_loss: 1.6660 - val_accuracy: 0.7340
Epoch 25/30
63/63 [==============================] - 2s 31ms/step - loss: 0.0578 - accuracy:
0.9870 - val_loss: 2.1822 - val_accuracy: 0.7170
Epoch 26/30
63/63 [==============================] - 2s 32ms/step - loss: 0.0306 - accuracy:
0.9890 - val_loss: 2.1361 - val_accuracy: 0.7150
Epoch 27/30
```

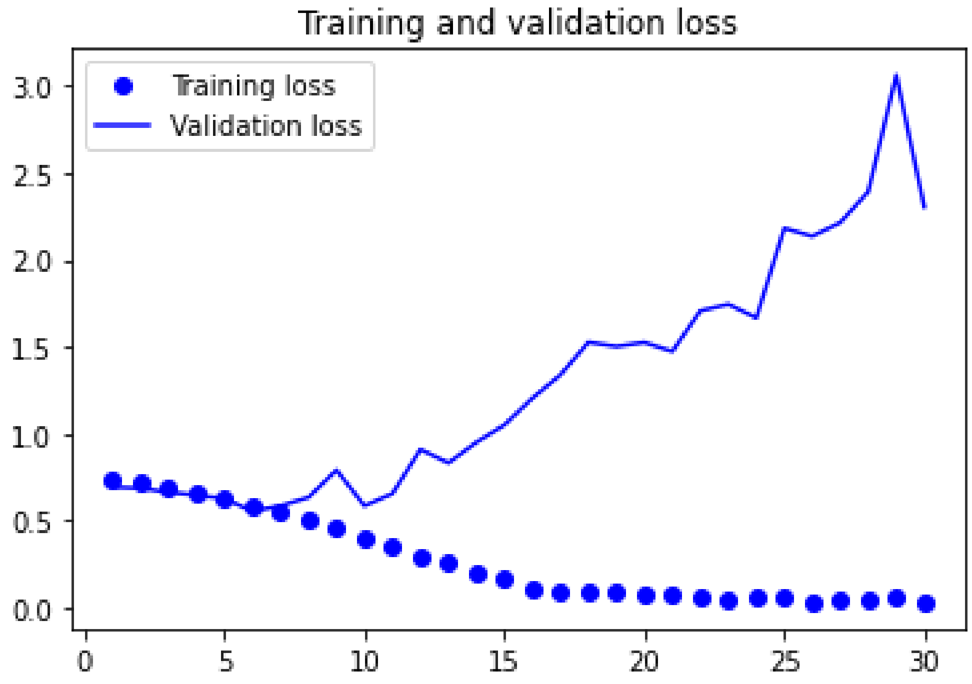
```
63/63 [==============================] - 2s 31ms/step - loss: 0.0501 - accuracy:
0.9870 - val_loss: 2.2148 - val_accuracy: 0.7100
Epoch 28/30
63/63 [==============================] - 2s 30ms/step - loss: 0.0508 - accuracy:
0.9850 - val_loss: 2.3925 - val_accuracy: 0.7060
Epoch 29/30
63/63 [==============================] - 2s 32ms/step - loss: 0.0546 - accuracy:
0.9830 - val_loss: 3.0640 - val_accuracy: 0.6710
Epoch 30/30
63/63 [==============================] - 2s 32ms/step - loss: 0.0341 - accuracy:
0.9890 - val_loss: 2.3057 - val_accuracy: 0.7040
```

Display curves of loss and accuracy in training

[12]:

```
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()
```





These plots show the overfit of the model. The training accuracy is increasing over time and almost reach 100% at end.

Evaluate model on test set

[13]:
```
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 16ms/step - loss: 0.6210 - accuracy:
0.6830
Test accuracy: 0.683
```

1.1.4 Build the basic model with dropout

[14]:
```
inputs = keras.Input(shape=(180, 180, 3))
x = layers.Rescaling(1./255)(inputs)  # rescale the input
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)   # use dropout
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

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

[15]:
```
# use image_dataset_from_derectory to read the image
#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.
```

[16]:

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

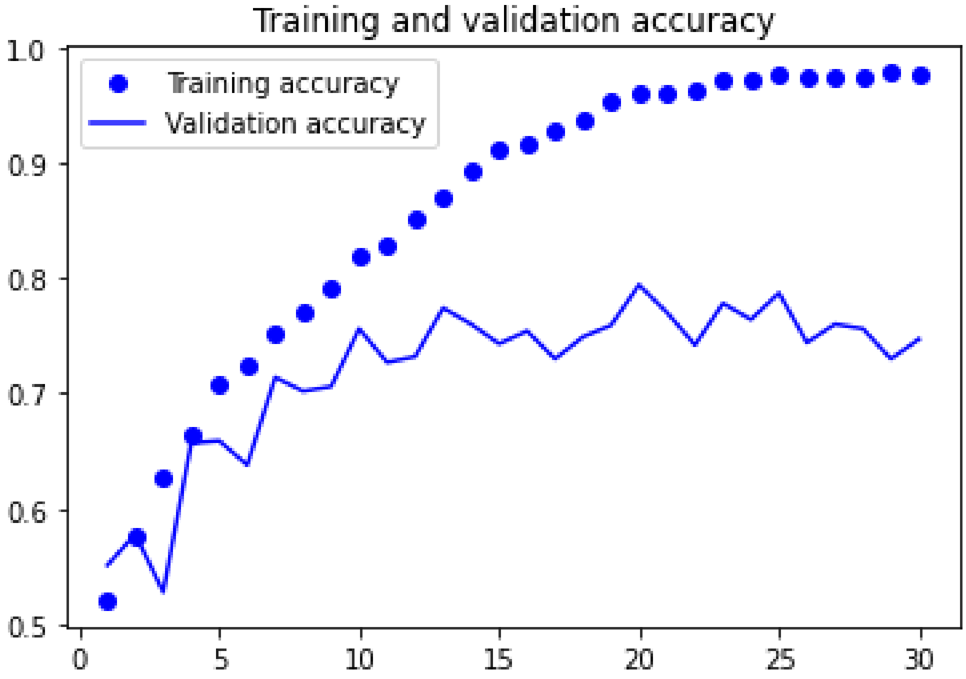
```
Epoch 1/30
63/63 [==============================] - 3s 34ms/step - loss: 0.7086 - accuracy:
0.5210 - val_loss: 0.6860 - val_accuracy: 0.5510
Epoch 2/30
63/63 [==============================] - 2s 31ms/step - loss: 0.6862 - accuracy:
0.5760 - val_loss: 0.6771 - val_accuracy: 0.5780
Epoch 3/30
63/63 [==============================] - 2s 31ms/step - loss: 0.6536 - accuracy:
0.6275 - val_loss: 0.8273 - val_accuracy: 0.5280
Epoch 4/30
63/63 [==============================] - 2s 35ms/step - loss: 0.6164 - accuracy:
0.6650 - val_loss: 0.6161 - val_accuracy: 0.6570
Epoch 5/30
63/63 [==============================] - 2s 34ms/step - loss: 0.5776 - accuracy:
0.7075 - val_loss: 0.7362 - val_accuracy: 0.6590
Epoch 6/30
63/63 [==============================] - 2s 35ms/step - loss: 0.5598 - accuracy:
0.7250 - val_loss: 0.6701 - val_accuracy: 0.6380
Epoch 7/30
63/63 [==============================] - 2s 36ms/step - loss: 0.5207 - accuracy:
0.7525 - val_loss: 0.5844 - val_accuracy: 0.7140
Epoch 8/30
63/63 [==============================] - 2s 33ms/step - loss: 0.4880 - accuracy:
0.7715 - val_loss: 0.6226 - val_accuracy: 0.7020
Epoch 9/30
63/63 [==============================] - 2s 31ms/step - loss: 0.4307 - accuracy:
0.7910 - val_loss: 0.7311 - val_accuracy: 0.7060
Epoch 10/30
63/63 [==============================] - 2s 32ms/step - loss: 0.4277 - accuracy:
0.8190 - val_loss: 0.5493 - val_accuracy: 0.7560
Epoch 11/30
63/63 [==============================] - 2s 32ms/step - loss: 0.3878 - accuracy:
0.8290 - val_loss: 0.6207 - val_accuracy: 0.7270
```

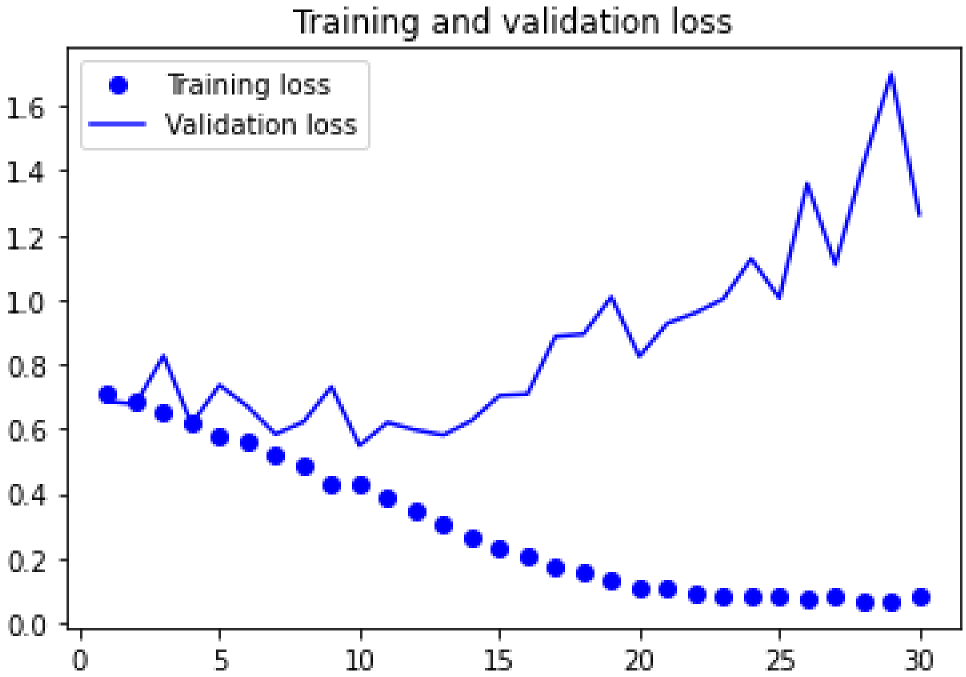
```
Epoch 12/30
63/63 [==============================] - 2s 31ms/step - loss: 0.3447 - accuracy:
0.8525 - val_loss: 0.5973 - val_accuracy: 0.7320
Epoch 13/30
63/63 [==============================] - 2s 31ms/step - loss: 0.3049 - accuracy:
0.8710 - val_loss: 0.5817 - val_accuracy: 0.7740
Epoch 14/30
63/63 [==============================] - 2s 31ms/step - loss: 0.2626 - accuracy:
0.8925 - val_loss: 0.6262 - val_accuracy: 0.7600
Epoch 15/30
63/63 [==============================] - 2s 31ms/step - loss: 0.2291 - accuracy:
0.9110 - val_loss: 0.7037 - val_accuracy: 0.7430
Epoch 16/30
63/63 [==============================] - 2s 30ms/step - loss: 0.2053 - accuracy:
0.9160 - val_loss: 0.7085 - val_accuracy: 0.7540
Epoch 17/30
63/63 [==============================] - 2s 31ms/step - loss: 0.1767 - accuracy:
0.9275 - val_loss: 0.8866 - val_accuracy: 0.7300
Epoch 18/30
63/63 [==============================] - 2s 34ms/step - loss: 0.1585 - accuracy:
0.9365 - val_loss: 0.8940 - val_accuracy: 0.7490
Epoch 19/30
63/63 [==============================] - 2s 34ms/step - loss: 0.1314 - accuracy:
0.9520 - val_loss: 1.0091 - val_accuracy: 0.7590
Epoch 20/30
63/63 [==============================] - 2s 37ms/step - loss: 0.1051 - accuracy:
0.9590 - val_loss: 0.8249 - val_accuracy: 0.7940
Epoch 21/30
63/63 [==============================] - 2s 36ms/step - loss: 0.1083 - accuracy:
0.9605 - val_loss: 0.9275 - val_accuracy: 0.7700
Epoch 22/30
63/63 [==============================] - 2s 33ms/step - loss: 0.0925 - accuracy:
0.9635 - val_loss: 0.9598 - val_accuracy: 0.7420
Epoch 23/30
63/63 [==============================] - 2s 33ms/step - loss: 0.0866 - accuracy:
0.9710 - val_loss: 1.0029 - val_accuracy: 0.7780
Epoch 24/30
63/63 [==============================] - 2s 32ms/step - loss: 0.0802 - accuracy:
0.9705 - val_loss: 1.1273 - val_accuracy: 0.7640
Epoch 25/30
63/63 [==============================] - 2s 32ms/step - loss: 0.0842 - accuracy:
0.9755 - val_loss: 1.0059 - val_accuracy: 0.7870
Epoch 26/30
63/63 [==============================] - 2s 32ms/step - loss: 0.0767 - accuracy:
0.9750 - val_loss: 1.3593 - val_accuracy: 0.7440
Epoch 27/30
63/63 [==============================] - 2s 31ms/step - loss: 0.0820 - accuracy:
0.9735 - val_loss: 1.1100 - val_accuracy: 0.7600
```

```
Epoch 28/30
63/63 [==============================] - 2s 31ms/step - loss: 0.0680 - accuracy:
0.9740 - val_loss: 1.4193 - val_accuracy: 0.7560
Epoch 29/30
63/63 [==============================] - 2s 32ms/step - loss: 0.0677 - accuracy:
0.9780 - val_loss: 1.6990 - val_accuracy: 0.7300
Epoch 30/30
63/63 [==============================] - 2s 32ms/step - loss: 0.0857 - accuracy:
0.9765 - val_loss: 1.2649 - val_accuracy: 0.7470
```

Display curves of loss and accuracy in training

[17]:
```
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()
```





Evaluate model on test set

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

```
32/32 [==============================] - 1s 16ms/step - loss: 0.6097 - accuracy:
0.7140
Test accuracy: 0.714
```

Question_1

October 22, 2023

0.1 Question 1: Use data augmentation to reduce overfit

[1]:

```
# import the library
import os, shutil, pathlib
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras import layers
```

[2]:

```
'''# copy and split to train, validation and test data
import os, shutil, pathlib

original_dir = pathlib.Path("train")
new_base_dir = pathlib.Path("cats_vs_dogs_small")  # store the smaller new␣
 ↪dataset

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)

# we create a new dataset containing three subsets
make_subset("train", start_index=0, end_index=1000) # training set with 1000␣
 ↪samples of each class
make_subset("validation", start_index=1000, end_index=1500) # validation set␣
 ↪with 500 samples of each class
make_subset("test", start_index=1500, end_index=2000)  # test set with 500␣
 ↪samples of each class'''
```

[2]:

```
'# copy and split to train, validation and test data\nimport os, shutil,
pathlib\n\noriginal_dir = pathlib.Path("train")\nnew_base_dir =
pathlib.Path("cats_vs_dogs_small")  # store the smaller new dataset\n\ndef
make_subset(subset_name, start_index, end_index):\n    for category in ("cat",
"dog"):\n        dir = new_base_dir / subset_name / category\n
os.makedirs(dir)\n        fnames = [f"{category}.{i}.jpg" for i in
```

```
range(start_index, end_index)]\n        for fname in fnames:\n
shutil.copyfile(src=original_dir / fname,\n                            dst=dir /
fname)\n\n# we create a new dataset containing three
subsets\nmake_subset("train", start_index=0, end_index=1000) # training set with
1000 samples of each class\nmake_subset("validation", start_index=1000,
end_index=1500) # validation set with 500 samples of each
class\nmake_subset("test", start_index=1500, end_index=2000)  # test set with
500 samples of each class'
```

[3]:
```
new_base_dir = pathlib.Path("cats_vs_dogs_small")
# use image_dataset_from_derectory to read the image
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.
```

0.1.1 Use data augmentation

[4]:
```
# define a data augmentation stage
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)
```

[5]:
```
# display some 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")
```

0.1.2 Build the model with data augmentation

[6]:

```
inputs = keras.Input(shape=(180, 180, 3))
x = data_augmentation(inputs)    # use data augmentation
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)    # use dropout
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

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

[7]:

```
# train the regularized convnet
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_with_augmentation.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=100,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/100
63/63 [==============================] - 6s 43ms/step - loss: 0.7978 - accuracy:
0.5070 - val_loss: 0.7002 - val_accuracy: 0.5000
Epoch 2/100
63/63 [==============================] - 2s 31ms/step - loss: 0.7017 - accuracy:
0.5255 - val_loss: 0.6912 - val_accuracy: 0.5290
Epoch 3/100
63/63 [==============================] - 2s 33ms/step - loss: 0.6948 - accuracy:
0.5520 - val_loss: 0.6854 - val_accuracy: 0.5190
Epoch 4/100
63/63 [==============================] - 2s 32ms/step - loss: 0.6767 - accuracy:
0.5785 - val_loss: 0.6445 - val_accuracy: 0.6510
Epoch 5/100
63/63 [==============================] - 2s 34ms/step - loss: 0.6621 - accuracy:
0.6305 - val_loss: 0.6460 - val_accuracy: 0.6160
Epoch 6/100
63/63 [==============================] - 2s 35ms/step - loss: 0.6478 - accuracy:
0.6515 - val_loss: 0.6287 - val_accuracy: 0.6360
Epoch 7/100
63/63 [==============================] - 2s 32ms/step - loss: 0.6238 - accuracy:
0.6695 - val_loss: 0.7161 - val_accuracy: 0.5560
```

```
Epoch 8/100
63/63 [==============================] - 2s 32ms/step - loss: 0.6378 - accuracy:
0.6680 - val_loss: 0.6234 - val_accuracy: 0.6310
Epoch 9/100
63/63 [==============================] - 2s 31ms/step - loss: 0.6083 - accuracy:
0.6780 - val_loss: 0.6181 - val_accuracy: 0.6520
Epoch 10/100
63/63 [==============================] - 2s 31ms/step - loss: 0.6198 - accuracy:
0.6865 - val_loss: 0.6321 - val_accuracy: 0.6190
Epoch 11/100
63/63 [==============================] - 2s 31ms/step - loss: 0.5821 - accuracy:
0.6960 - val_loss: 0.6873 - val_accuracy: 0.6280
Epoch 12/100
63/63 [==============================] - 2s 31ms/step - loss: 0.5686 - accuracy:
0.7105 - val_loss: 0.6852 - val_accuracy: 0.6530
Epoch 13/100
63/63 [==============================] - 2s 31ms/step - loss: 0.5623 - accuracy:
0.7140 - val_loss: 0.6492 - val_accuracy: 0.6610
Epoch 14/100
63/63 [==============================] - 2s 31ms/step - loss: 0.5508 - accuracy:
0.7290 - val_loss: 0.5610 - val_accuracy: 0.7150
Epoch 15/100
63/63 [==============================] - 2s 31ms/step - loss: 0.5465 - accuracy:
0.7335 - val_loss: 0.5574 - val_accuracy: 0.6970
Epoch 16/100
63/63 [==============================] - 2s 31ms/step - loss: 0.5440 - accuracy:
0.7315 - val_loss: 0.5127 - val_accuracy: 0.7530
Epoch 17/100
63/63 [==============================] - 2s 31ms/step - loss: 0.5310 - accuracy:
0.7430 - val_loss: 0.6575 - val_accuracy: 0.6860
Epoch 18/100
63/63 [==============================] - 2s 30ms/step - loss: 0.5069 - accuracy:
0.7590 - val_loss: 0.5824 - val_accuracy: 0.7170
Epoch 19/100
63/63 [==============================] - 2s 30ms/step - loss: 0.5028 - accuracy:
0.7570 - val_loss: 0.5013 - val_accuracy: 0.7450
Epoch 20/100
63/63 [==============================] - 2s 30ms/step - loss: 0.4930 - accuracy:
0.7625 - val_loss: 0.5137 - val_accuracy: 0.7530
Epoch 21/100
63/63 [==============================] - 2s 30ms/step - loss: 0.4903 - accuracy:
0.7680 - val_loss: 0.5223 - val_accuracy: 0.7300
Epoch 22/100
63/63 [==============================] - 2s 30ms/step - loss: 0.4684 - accuracy:
0.7815 - val_loss: 0.4735 - val_accuracy: 0.7730
Epoch 23/100
63/63 [==============================] - 2s 31ms/step - loss: 0.4679 - accuracy:
0.7870 - val_loss: 0.5275 - val_accuracy: 0.7620
```

```
Epoch 24/100
63/63 [==============================] - 2s 30ms/step - loss: 0.4616 - accuracy:
0.7835 - val_loss: 0.4615 - val_accuracy: 0.7910
Epoch 25/100
63/63 [==============================] - 2s 30ms/step - loss: 0.4743 - accuracy:
0.7705 - val_loss: 0.4971 - val_accuracy: 0.7710
Epoch 26/100
63/63 [==============================] - 2s 31ms/step - loss: 0.4326 - accuracy:
0.8095 - val_loss: 0.4645 - val_accuracy: 0.7940
Epoch 27/100
63/63 [==============================] - 2s 31ms/step - loss: 0.4313 - accuracy:
0.8040 - val_loss: 0.4392 - val_accuracy: 0.8000
Epoch 28/100
63/63 [==============================] - 2s 31ms/step - loss: 0.4264 - accuracy:
0.8055 - val_loss: 0.7133 - val_accuracy: 0.7040
Epoch 29/100
63/63 [==============================] - 2s 30ms/step - loss: 0.4185 - accuracy:
0.8090 - val_loss: 0.4207 - val_accuracy: 0.8230
Epoch 30/100
63/63 [==============================] - 2s 31ms/step - loss: 0.4065 - accuracy:
0.8185 - val_loss: 0.4185 - val_accuracy: 0.8210
Epoch 31/100
63/63 [==============================] - 2s 31ms/step - loss: 0.3985 - accuracy:
0.8125 - val_loss: 0.9788 - val_accuracy: 0.6780
Epoch 32/100
63/63 [==============================] - 2s 31ms/step - loss: 0.3997 - accuracy:
0.8155 - val_loss: 0.4682 - val_accuracy: 0.7900
Epoch 33/100
63/63 [==============================] - 2s 31ms/step - loss: 0.3794 - accuracy:
0.8320 - val_loss: 0.4133 - val_accuracy: 0.8180
Epoch 34/100
63/63 [==============================] - 2s 30ms/step - loss: 0.3588 - accuracy:
0.8415 - val_loss: 0.4766 - val_accuracy: 0.8000
Epoch 35/100
63/63 [==============================] - 2s 31ms/step - loss: 0.3560 - accuracy:
0.8430 - val_loss: 0.4595 - val_accuracy: 0.8160
Epoch 36/100
63/63 [==============================] - 2s 31ms/step - loss: 0.3575 - accuracy:
0.8450 - val_loss: 0.6186 - val_accuracy: 0.7700
Epoch 37/100
63/63 [==============================] - 2s 31ms/step - loss: 0.3725 - accuracy:
0.8415 - val_loss: 0.4238 - val_accuracy: 0.8200
Epoch 38/100
63/63 [==============================] - 2s 30ms/step - loss: 0.3747 - accuracy:
0.8380 - val_loss: 0.3889 - val_accuracy: 0.8300
Epoch 39/100
63/63 [==============================] - 2s 31ms/step - loss: 0.3357 - accuracy:
0.8600 - val_loss: 0.5724 - val_accuracy: 0.7730
```

```
Epoch 40/100
63/63 [==============================] - 2s 30ms/step - loss: 0.3261 - accuracy:
0.8600 - val_loss: 0.6193 - val_accuracy: 0.7970
Epoch 41/100
63/63 [==============================] - 2s 31ms/step - loss: 0.3443 - accuracy:
0.8545 - val_loss: 0.4352 - val_accuracy: 0.8180
Epoch 42/100
63/63 [==============================] - 2s 32ms/step - loss: 0.3119 - accuracy:
0.8585 - val_loss: 0.3971 - val_accuracy: 0.8310
Epoch 43/100
63/63 [==============================] - 2s 31ms/step - loss: 0.3553 - accuracy:
0.8500 - val_loss: 0.4091 - val_accuracy: 0.8280
Epoch 44/100
63/63 [==============================] - 2s 31ms/step - loss: 0.2975 - accuracy:
0.8785 - val_loss: 0.5352 - val_accuracy: 0.8150
Epoch 45/100
63/63 [==============================] - 2s 31ms/step - loss: 0.3118 - accuracy:
0.8755 - val_loss: 0.4800 - val_accuracy: 0.8250
Epoch 46/100
63/63 [==============================] - 2s 31ms/step - loss: 0.2982 - accuracy:
0.8645 - val_loss: 0.4459 - val_accuracy: 0.8240
Epoch 47/100
63/63 [==============================] - 2s 31ms/step - loss: 0.2826 - accuracy:
0.8680 - val_loss: 0.6733 - val_accuracy: 0.7900
Epoch 48/100
63/63 [==============================] - 2s 31ms/step - loss: 0.2935 - accuracy:
0.8810 - val_loss: 0.4218 - val_accuracy: 0.8430
Epoch 49/100
63/63 [==============================] - 2s 31ms/step - loss: 0.2792 - accuracy:
0.8855 - val_loss: 0.4418 - val_accuracy: 0.8010
Epoch 50/100
63/63 [==============================] - 2s 31ms/step - loss: 0.2908 - accuracy:
0.8715 - val_loss: 0.3958 - val_accuracy: 0.8540
Epoch 51/100
63/63 [==============================] - 2s 32ms/step - loss: 0.2724 - accuracy:
0.8800 - val_loss: 0.4704 - val_accuracy: 0.8130
Epoch 52/100
63/63 [==============================] - 2s 32ms/step - loss: 0.2841 - accuracy:
0.8785 - val_loss: 0.4588 - val_accuracy: 0.8420
Epoch 53/100
63/63 [==============================] - 2s 31ms/step - loss: 0.2997 - accuracy:
0.8755 - val_loss: 0.5019 - val_accuracy: 0.7880
Epoch 54/100
63/63 [==============================] - 2s 31ms/step - loss: 0.2477 - accuracy:
0.9060 - val_loss: 0.5825 - val_accuracy: 0.7670
Epoch 55/100
63/63 [==============================] - 2s 32ms/step - loss: 0.2583 - accuracy:
0.8980 - val_loss: 0.4427 - val_accuracy: 0.8310
```

```
Epoch 56/100
63/63 [==============================] - 2s 31ms/step - loss: 0.2583 - accuracy:
0.8985 - val_loss: 0.4171 - val_accuracy: 0.8530
Epoch 57/100
63/63 [==============================] - 2s 31ms/step - loss: 0.2668 - accuracy:
0.8910 - val_loss: 0.5839 - val_accuracy: 0.8410
Epoch 58/100
63/63 [==============================] - 2s 31ms/step - loss: 0.2395 - accuracy:
0.9055 - val_loss: 0.6619 - val_accuracy: 0.8330
Epoch 59/100
63/63 [==============================] - 2s 31ms/step - loss: 0.2277 - accuracy:
0.9075 - val_loss: 0.8641 - val_accuracy: 0.7990
Epoch 60/100
63/63 [==============================] - 2s 31ms/step - loss: 0.2488 - accuracy:
0.9045 - val_loss: 0.6031 - val_accuracy: 0.8110
Epoch 61/100
63/63 [==============================] - 2s 31ms/step - loss: 0.2263 - accuracy:
0.9050 - val_loss: 0.4859 - val_accuracy: 0.8380
Epoch 62/100
63/63 [==============================] - 2s 31ms/step - loss: 0.2293 - accuracy:
0.9090 - val_loss: 0.5076 - val_accuracy: 0.8360
Epoch 63/100
63/63 [==============================] - 2s 32ms/step - loss: 0.2404 - accuracy:
0.9035 - val_loss: 0.9342 - val_accuracy: 0.7640
Epoch 64/100
63/63 [==============================] - 2s 31ms/step - loss: 0.2202 - accuracy:
0.9115 - val_loss: 0.7011 - val_accuracy: 0.8200
Epoch 65/100
63/63 [==============================] - 2s 31ms/step - loss: 0.2350 - accuracy:
0.9135 - val_loss: 1.1186 - val_accuracy: 0.7660
Epoch 66/100
63/63 [==============================] - 2s 32ms/step - loss: 0.2301 - accuracy:
0.9060 - val_loss: 0.6011 - val_accuracy: 0.8310
Epoch 67/100
63/63 [==============================] - 2s 32ms/step - loss: 0.2161 - accuracy:
0.9155 - val_loss: 0.8790 - val_accuracy: 0.7940
Epoch 68/100
63/63 [==============================] - 2s 32ms/step - loss: 0.2021 - accuracy:
0.9250 - val_loss: 0.5953 - val_accuracy: 0.8250
Epoch 69/100
63/63 [==============================] - 2s 32ms/step - loss: 0.2192 - accuracy:
0.9080 - val_loss: 0.5215 - val_accuracy: 0.8410
Epoch 70/100
63/63 [==============================] - 2s 32ms/step - loss: 0.2309 - accuracy:
0.9150 - val_loss: 0.8904 - val_accuracy: 0.8050
Epoch 71/100
63/63 [==============================] - 2s 32ms/step - loss: 0.2185 - accuracy:
0.9185 - val_loss: 0.4998 - val_accuracy: 0.8370
```

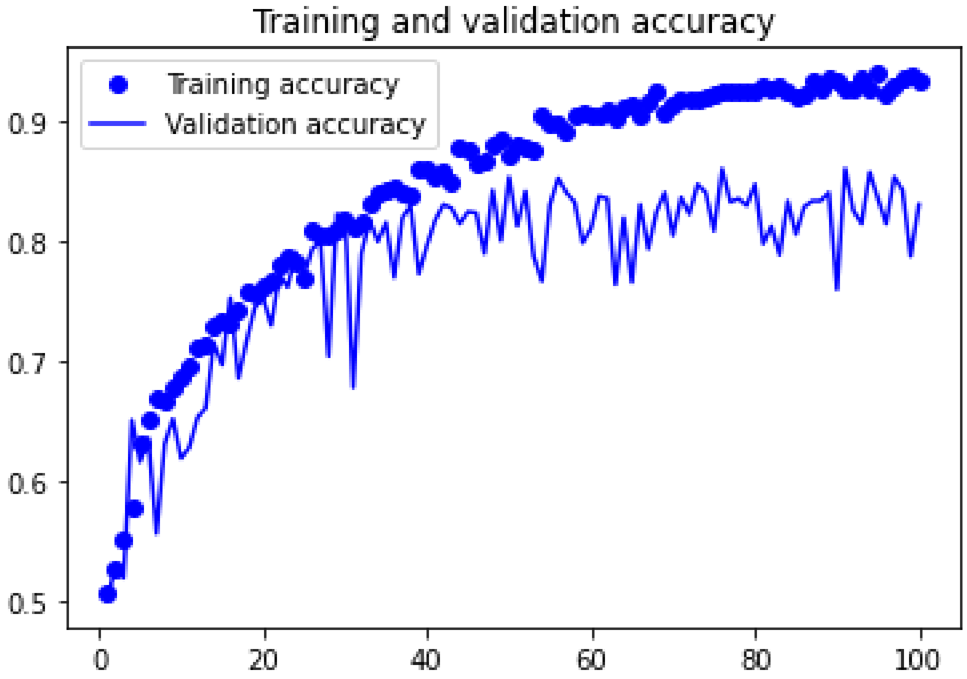
```
Epoch 72/100
63/63 [==============================] - 2s 32ms/step - loss: 0.2127 - accuracy:
0.9190 - val_loss: 0.5497 - val_accuracy: 0.8230
Epoch 73/100
63/63 [==============================] - 2s 33ms/step - loss: 0.2134 - accuracy:
0.9180 - val_loss: 0.6487 - val_accuracy: 0.8480
Epoch 74/100
63/63 [==============================] - 2s 32ms/step - loss: 0.1910 - accuracy:
0.9215 - val_loss: 0.6032 - val_accuracy: 0.8410
Epoch 75/100
63/63 [==============================] - 2s 32ms/step - loss: 0.1957 - accuracy:
0.9225 - val_loss: 0.6630 - val_accuracy: 0.8090
Epoch 76/100
63/63 [==============================] - 2s 32ms/step - loss: 0.1951 - accuracy:
0.9255 - val_loss: 0.5095 - val_accuracy: 0.8610
Epoch 77/100
63/63 [==============================] - 2s 31ms/step - loss: 0.1937 - accuracy:
0.9260 - val_loss: 0.7902 - val_accuracy: 0.8330
Epoch 78/100
63/63 [==============================] - 2s 33ms/step - loss: 0.1949 - accuracy:
0.9255 - val_loss: 0.6182 - val_accuracy: 0.8360
Epoch 79/100
63/63 [==============================] - 2s 34ms/step - loss: 0.1902 - accuracy:
0.9250 - val_loss: 0.7397 - val_accuracy: 0.8300
Epoch 80/100
63/63 [==============================] - 2s 33ms/step - loss: 0.1957 - accuracy:
0.9260 - val_loss: 0.6013 - val_accuracy: 0.8480
Epoch 81/100
63/63 [==============================] - 2s 33ms/step - loss: 0.2003 - accuracy:
0.9290 - val_loss: 1.0275 - val_accuracy: 0.7980
Epoch 82/100
63/63 [==============================] - 2s 32ms/step - loss: 0.2065 - accuracy:
0.9285 - val_loss: 0.6700 - val_accuracy: 0.8130
Epoch 83/100
63/63 [==============================] - 2s 32ms/step - loss: 0.1910 - accuracy:
0.9290 - val_loss: 0.7211 - val_accuracy: 0.7890
Epoch 84/100
63/63 [==============================] - 2s 33ms/step - loss: 0.2060 - accuracy:
0.9265 - val_loss: 0.7134 - val_accuracy: 0.8340
Epoch 85/100
63/63 [==============================] - 2s 33ms/step - loss: 0.2152 - accuracy:
0.9220 - val_loss: 0.8923 - val_accuracy: 0.8060
Epoch 86/100
63/63 [==============================] - 2s 33ms/step - loss: 0.2063 - accuracy:
0.9225 - val_loss: 0.6472 - val_accuracy: 0.8290
Epoch 87/100
63/63 [==============================] - 2s 33ms/step - loss: 0.1848 - accuracy:
0.9340 - val_loss: 0.5930 - val_accuracy: 0.8340
```

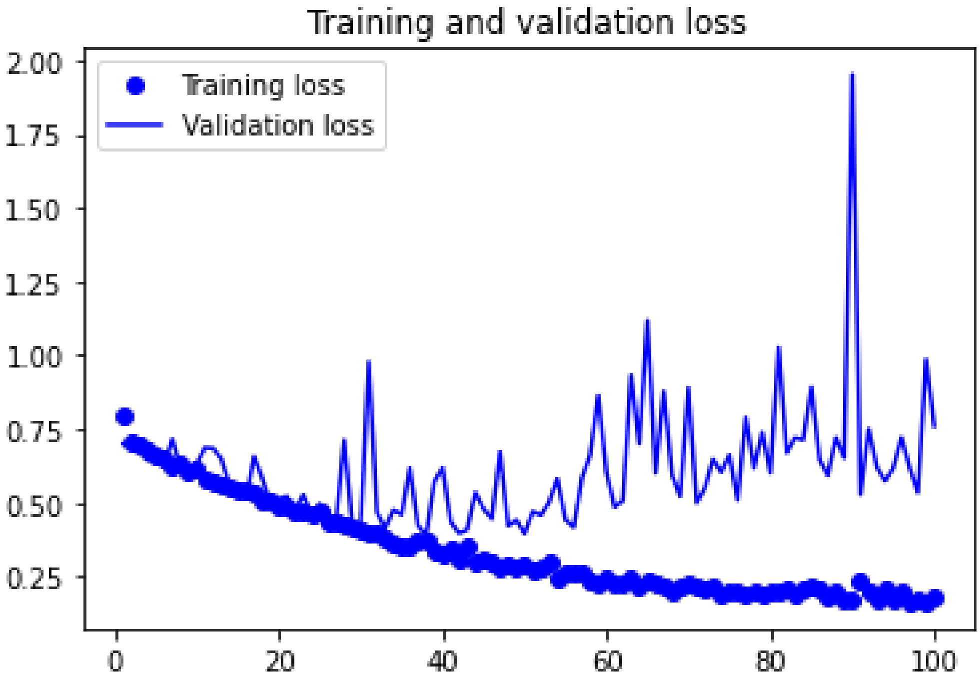
```
Epoch 88/100
63/63 [==============================] - 2s 32ms/step - loss: 0.1984 - accuracy:
0.9270 - val_loss: 0.7216 - val_accuracy: 0.8340
Epoch 89/100
63/63 [==============================] - 2s 33ms/step - loss: 0.1687 - accuracy:
0.9365 - val_loss: 0.6529 - val_accuracy: 0.8410
Epoch 90/100
63/63 [==============================] - 2s 32ms/step - loss: 0.1766 - accuracy:
0.9340 - val_loss: 1.9519 - val_accuracy: 0.7600
Epoch 91/100
63/63 [==============================] - 2s 32ms/step - loss: 0.2313 - accuracy:
0.9285 - val_loss: 0.5285 - val_accuracy: 0.8610
Epoch 92/100
63/63 [==============================] - 2s 32ms/step - loss: 0.1984 - accuracy:
0.9280 - val_loss: 0.7506 - val_accuracy: 0.8260
Epoch 93/100
63/63 [==============================] - 2s 32ms/step - loss: 0.1690 - accuracy:
0.9355 - val_loss: 0.6209 - val_accuracy: 0.8150
Epoch 94/100
63/63 [==============================] - 2s 32ms/step - loss: 0.2045 - accuracy:
0.9270 - val_loss: 0.5741 - val_accuracy: 0.8580
Epoch 95/100
63/63 [==============================] - 2s 34ms/step - loss: 0.1690 - accuracy:
0.9405 - val_loss: 0.6183 - val_accuracy: 0.8360
Epoch 96/100
63/63 [==============================] - 2s 33ms/step - loss: 0.2020 - accuracy:
0.9235 - val_loss: 0.7223 - val_accuracy: 0.8150
Epoch 97/100
63/63 [==============================] - 2s 33ms/step - loss: 0.1674 - accuracy:
0.9310 - val_loss: 0.6190 - val_accuracy: 0.8540
Epoch 98/100
63/63 [==============================] - 2s 32ms/step - loss: 0.1710 - accuracy:
0.9360 - val_loss: 0.5351 - val_accuracy: 0.8430
Epoch 99/100
63/63 [==============================] - 2s 33ms/step - loss: 0.1640 - accuracy:
0.9380 - val_loss: 0.9874 - val_accuracy: 0.7880
Epoch 100/100
63/63 [==============================] - 2s 32ms/step - loss: 0.1776 - accuracy:
0.9350 - val_loss: 0.7607 - val_accuracy: 0.8310
```

Display curves of loss and accuracy in training

[8]:
```
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()
```





Evaluate the model on test set

[9]:
```
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 [==============================] - 1s 16ms/step - loss: 0.4547 - accuracy:
0.8080
Test accuracy: 0.808
```

Question_2

October 22, 2023

0.1 Question 2: Increasing the training sample size

[1]:
```
# import the library
import os, shutil, pathlib
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras import layers

new_base_dir = pathlib.Path("cats_vs_dogs_large")
```

[2]:
```
'''# copy and split to train, validation and test data
original_dir = pathlib.Path("train")
new_base_dir = pathlib.Path("cats_vs_dogs_large")  # store the smaller new␣
↪dataset

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)

# we create a new dataset containing three subsets
make_subset("train", start_index=0, end_index=3000) # training set with 3000␣
↪samples of each class
make_subset("validation", start_index=3000, end_index=3500) # validation set␣
↪with 500 samples of each class
make_subset("test", start_index=3500, end_index=4000)  # test set with 500␣
↪samples of each class'''
```

[2]:
```
'# copy and split to train, validation and test data\noriginal_dir =
pathlib.Path("train")\nnew_base_dir = pathlib.Path("cats_vs_dogs_large")  #
store the smaller new dataset\n\ndef make_subset(subset_name, start_index,
end_index):\n    for category in ("cat", "dog"):\n        dir = new_base_dir /
subset_name / category\n        os.makedirs(dir)\n        fnames =
[f"{category}.{i}.jpg" for i in range(start_index, end_index)]\n        for
```

```
fname in fnames:\n            shutil.copyfile(src=original_dir / fname,\n
dst=dir / fname)\n\n# we create a new dataset containing three
subsets\nmake_subset("train", start_index=0, end_index=3000) # training set with
3000 samples of each class\nmake_subset("validation", start_index=3000,
end_index=3500) # validation set with 500 samples of each
class\nmake_subset("test", start_index=3500, end_index=4000)  # test set with
500 samples of each class'
```

0.1.1 Build the model with large training sample without data augmentation

[3]:
```python
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=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.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"])
```

[4]:
```python
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 6000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
```

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

[5]:
```python
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,)
```

[6]:
```python
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_big_sample.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=50,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/50
188/188 [==============================] - 24s 106ms/step - loss: 0.6887 -
accuracy: 0.5505 - val_loss: 0.6416 - val_accuracy: 0.6100
Epoch 2/50
188/188 [==============================] - 5s 25ms/step - loss: 0.6280 -
accuracy: 0.6512 - val_loss: 0.6340 - val_accuracy: 0.6120
Epoch 3/50
188/188 [==============================] - 5s 24ms/step - loss: 0.5826 -
accuracy: 0.6943 - val_loss: 0.7560 - val_accuracy: 0.6620
Epoch 4/50
188/188 [==============================] - 5s 24ms/step - loss: 0.5387 -
accuracy: 0.7353 - val_loss: 0.5183 - val_accuracy: 0.7520
Epoch 5/50
188/188 [==============================] - 5s 24ms/step - loss: 0.4911 -
accuracy: 0.7652 - val_loss: 0.5658 - val_accuracy: 0.7230
Epoch 6/50
188/188 [==============================] - 5s 25ms/step - loss: 0.4458 -
accuracy: 0.7975 - val_loss: 0.4445 - val_accuracy: 0.8010
Epoch 7/50
188/188 [==============================] - 5s 24ms/step - loss: 0.3939 -
accuracy: 0.8193 - val_loss: 0.4767 - val_accuracy: 0.7720
Epoch 8/50
188/188 [==============================] - 5s 25ms/step - loss: 0.3504 -
accuracy: 0.8468 - val_loss: 0.4084 - val_accuracy: 0.8070
Epoch 9/50
188/188 [==============================] - 5s 24ms/step - loss: 0.3171 -
```

```
accuracy: 0.8633 - val_loss: 0.4487 - val_accuracy: 0.7960
Epoch 10/50
188/188 [==============================] - 5s 24ms/step - loss: 0.2734 -
accuracy: 0.8847 - val_loss: 0.4697 - val_accuracy: 0.8100
Epoch 11/50
188/188 [==============================] - 5s 24ms/step - loss: 0.2391 -
accuracy: 0.9020 - val_loss: 0.4376 - val_accuracy: 0.8260
Epoch 12/50
188/188 [==============================] - 5s 24ms/step - loss: 0.2073 -
accuracy: 0.9143 - val_loss: 0.4877 - val_accuracy: 0.8160
Epoch 13/50
188/188 [==============================] - 5s 23ms/step - loss: 0.1729 -
accuracy: 0.9300 - val_loss: 0.4605 - val_accuracy: 0.8460
Epoch 14/50
188/188 [==============================] - 5s 24ms/step - loss: 0.1511 -
accuracy: 0.9418 - val_loss: 0.6108 - val_accuracy: 0.8100
Epoch 15/50
188/188 [==============================] - 5s 23ms/step - loss: 0.1290 -
accuracy: 0.9498 - val_loss: 0.5390 - val_accuracy: 0.8260
Epoch 16/50
188/188 [==============================] - 5s 23ms/step - loss: 0.1240 -
accuracy: 0.9493 - val_loss: 0.5320 - val_accuracy: 0.8370
Epoch 17/50
188/188 [==============================] - 4s 23ms/step - loss: 0.1044 -
accuracy: 0.9610 - val_loss: 0.5514 - val_accuracy: 0.8250
Epoch 18/50
188/188 [==============================] - 4s 23ms/step - loss: 0.0982 -
accuracy: 0.9663 - val_loss: 0.6529 - val_accuracy: 0.8320
Epoch 19/50
188/188 [==============================] - 5s 24ms/step - loss: 0.0887 -
accuracy: 0.9653 - val_loss: 0.6559 - val_accuracy: 0.8170
Epoch 20/50
188/188 [==============================] - 4s 23ms/step - loss: 0.0884 -
accuracy: 0.9692 - val_loss: 0.7840 - val_accuracy: 0.8240
Epoch 21/50
188/188 [==============================] - 4s 23ms/step - loss: 0.0872 -
accuracy: 0.9703 - val_loss: 0.6808 - val_accuracy: 0.8370
Epoch 22/50
188/188 [==============================] - 4s 23ms/step - loss: 0.0862 -
accuracy: 0.9718 - val_loss: 0.6493 - val_accuracy: 0.8320
Epoch 23/50
188/188 [==============================] - 4s 23ms/step - loss: 0.0682 -
accuracy: 0.9773 - val_loss: 0.8469 - val_accuracy: 0.8430
Epoch 24/50
188/188 [==============================] - 5s 24ms/step - loss: 0.0781 -
accuracy: 0.9730 - val_loss: 0.7578 - val_accuracy: 0.8430
Epoch 25/50
188/188 [==============================] - 4s 23ms/step - loss: 0.0787 -
```

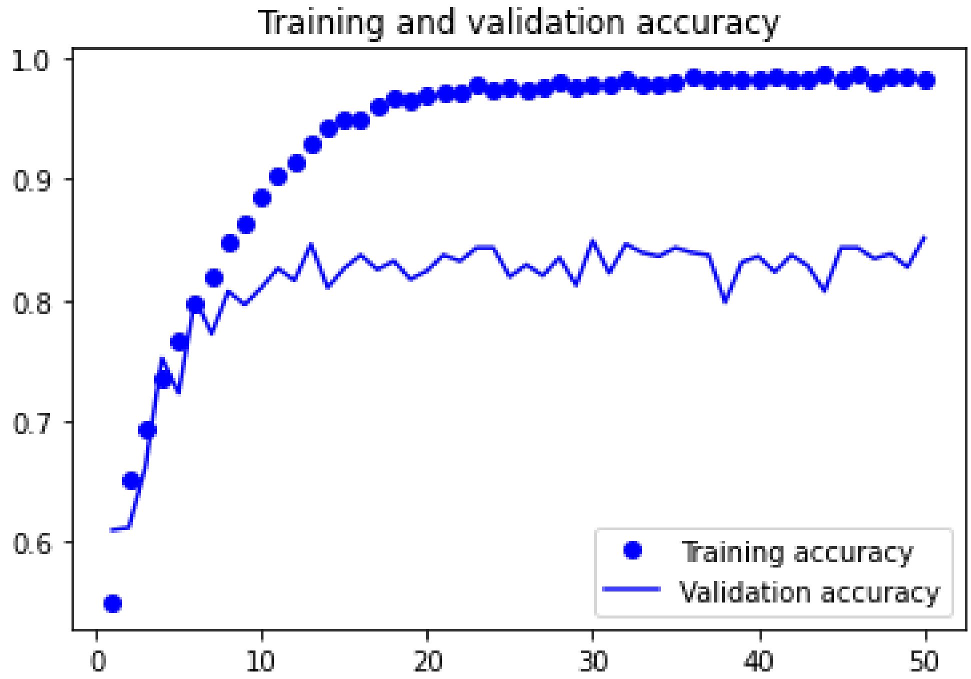
```
accuracy: 0.9758 - val_loss: 0.7780 - val_accuracy: 0.8190
Epoch 26/50
188/188 [==============================] - 5s 23ms/step - loss: 0.0752 -
accuracy: 0.9743 - val_loss: 0.8358 - val_accuracy: 0.8290
Epoch 27/50
188/188 [==============================] - 5s 24ms/step - loss: 0.0761 -
accuracy: 0.9763 - val_loss: 0.9785 - val_accuracy: 0.8200
Epoch 28/50
188/188 [==============================] - 5s 24ms/step - loss: 0.0669 -
accuracy: 0.9793 - val_loss: 0.9829 - val_accuracy: 0.8350
Epoch 29/50
188/188 [==============================] - 5s 23ms/step - loss: 0.0918 -
accuracy: 0.9757 - val_loss: 0.9013 - val_accuracy: 0.8120
Epoch 30/50
188/188 [==============================] - 5s 23ms/step - loss: 0.0645 -
accuracy: 0.9780 - val_loss: 0.9504 - val_accuracy: 0.8490
Epoch 31/50
188/188 [==============================] - 5s 24ms/step - loss: 0.0803 -
accuracy: 0.9767 - val_loss: 1.4901 - val_accuracy: 0.8220
Epoch 32/50
188/188 [==============================] - 4s 23ms/step - loss: 0.0690 -
accuracy: 0.9823 - val_loss: 0.8569 - val_accuracy: 0.8460
Epoch 33/50
188/188 [==============================] - 5s 24ms/step - loss: 0.0687 -
accuracy: 0.9770 - val_loss: 1.0201 - val_accuracy: 0.8390
Epoch 34/50
188/188 [==============================] - 5s 23ms/step - loss: 0.0674 -
accuracy: 0.9787 - val_loss: 1.0352 - val_accuracy: 0.8360
Epoch 35/50
188/188 [==============================] - 5s 24ms/step - loss: 0.0790 -
accuracy: 0.9797 - val_loss: 1.2013 - val_accuracy: 0.8430
Epoch 36/50
188/188 [==============================] - 5s 24ms/step - loss: 0.0651 -
accuracy: 0.9837 - val_loss: 1.0712 - val_accuracy: 0.8390
Epoch 37/50
188/188 [==============================] - 4s 23ms/step - loss: 0.0712 -
accuracy: 0.9813 - val_loss: 1.3066 - val_accuracy: 0.8370
Epoch 38/50
188/188 [==============================] - 5s 24ms/step - loss: 0.0648 -
accuracy: 0.9828 - val_loss: 1.9001 - val_accuracy: 0.7980
Epoch 39/50
188/188 [==============================] - 5s 24ms/step - loss: 0.0774 -
accuracy: 0.9817 - val_loss: 1.4017 - val_accuracy: 0.8310
Epoch 40/50
188/188 [==============================] - 5s 24ms/step - loss: 0.0628 -
accuracy: 0.9832 - val_loss: 1.2043 - val_accuracy: 0.8360
Epoch 41/50
188/188 [==============================] - 5s 24ms/step - loss: 0.0633 -
```

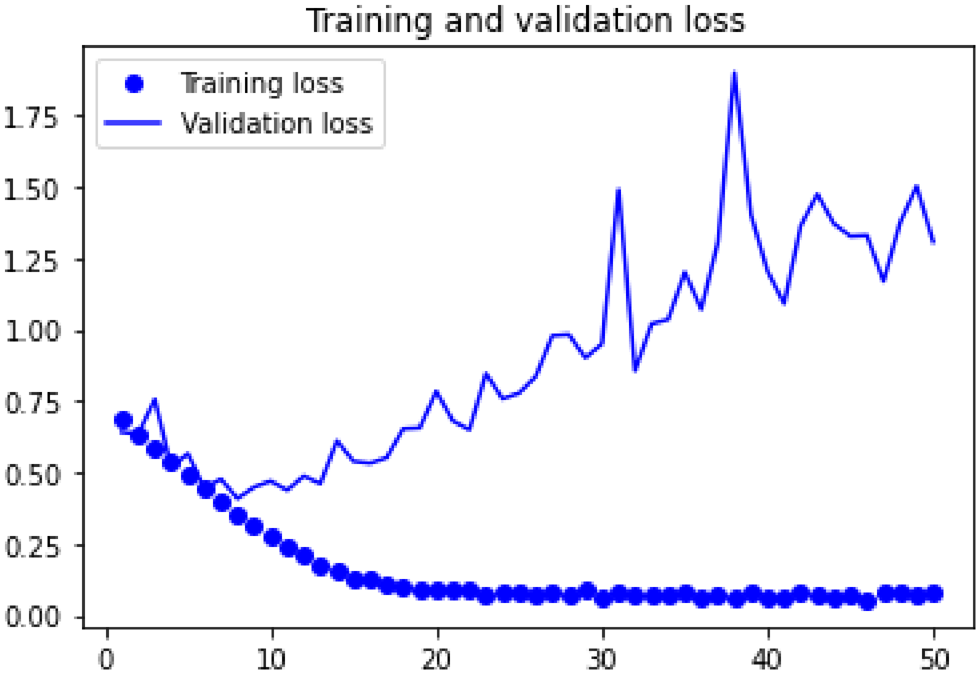
```
accuracy: 0.9850 - val_loss: 1.0905 - val_accuracy: 0.8230
Epoch 42/50
188/188 [==============================] - 5s 24ms/step - loss: 0.0774 -
accuracy: 0.9820 - val_loss: 1.3632 - val_accuracy: 0.8370
Epoch 43/50
188/188 [==============================] - 5s 24ms/step - loss: 0.0695 -
accuracy: 0.9828 - val_loss: 1.4746 - val_accuracy: 0.8280
Epoch 44/50
188/188 [==============================] - 5s 24ms/step - loss: 0.0661 -
accuracy: 0.9858 - val_loss: 1.3710 - val_accuracy: 0.8070
Epoch 45/50
188/188 [==============================] - 5s 24ms/step - loss: 0.0691 -
accuracy: 0.9820 - val_loss: 1.3272 - val_accuracy: 0.8430
Epoch 46/50
188/188 [==============================] - 5s 23ms/step - loss: 0.0531 -
accuracy: 0.9860 - val_loss: 1.3299 - val_accuracy: 0.8430
Epoch 47/50
188/188 [==============================] - 5s 24ms/step - loss: 0.0807 -
accuracy: 0.9810 - val_loss: 1.1691 - val_accuracy: 0.8340
Epoch 48/50
188/188 [==============================] - 5s 24ms/step - loss: 0.0772 -
accuracy: 0.9833 - val_loss: 1.3745 - val_accuracy: 0.8380
Epoch 49/50
188/188 [==============================] - 5s 25ms/step - loss: 0.0713 -
accuracy: 0.9842 - val_loss: 1.5021 - val_accuracy: 0.8270
Epoch 50/50
188/188 [==============================] - 5s 24ms/step - loss: 0.0796 -
accuracy: 0.9818 - val_loss: 1.3072 - val_accuracy: 0.8510
```

Display the curve of accuracy and loss during training

[7]:

```
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()
```





Evaluate model on test set

[8]:
```python
test_model = keras.models.load_model("convnet_from_scratch_big_sample.keras")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 3s 48ms/step - loss: 0.4710 - accuracy:
0.7970
Test accuracy: 0.797
```

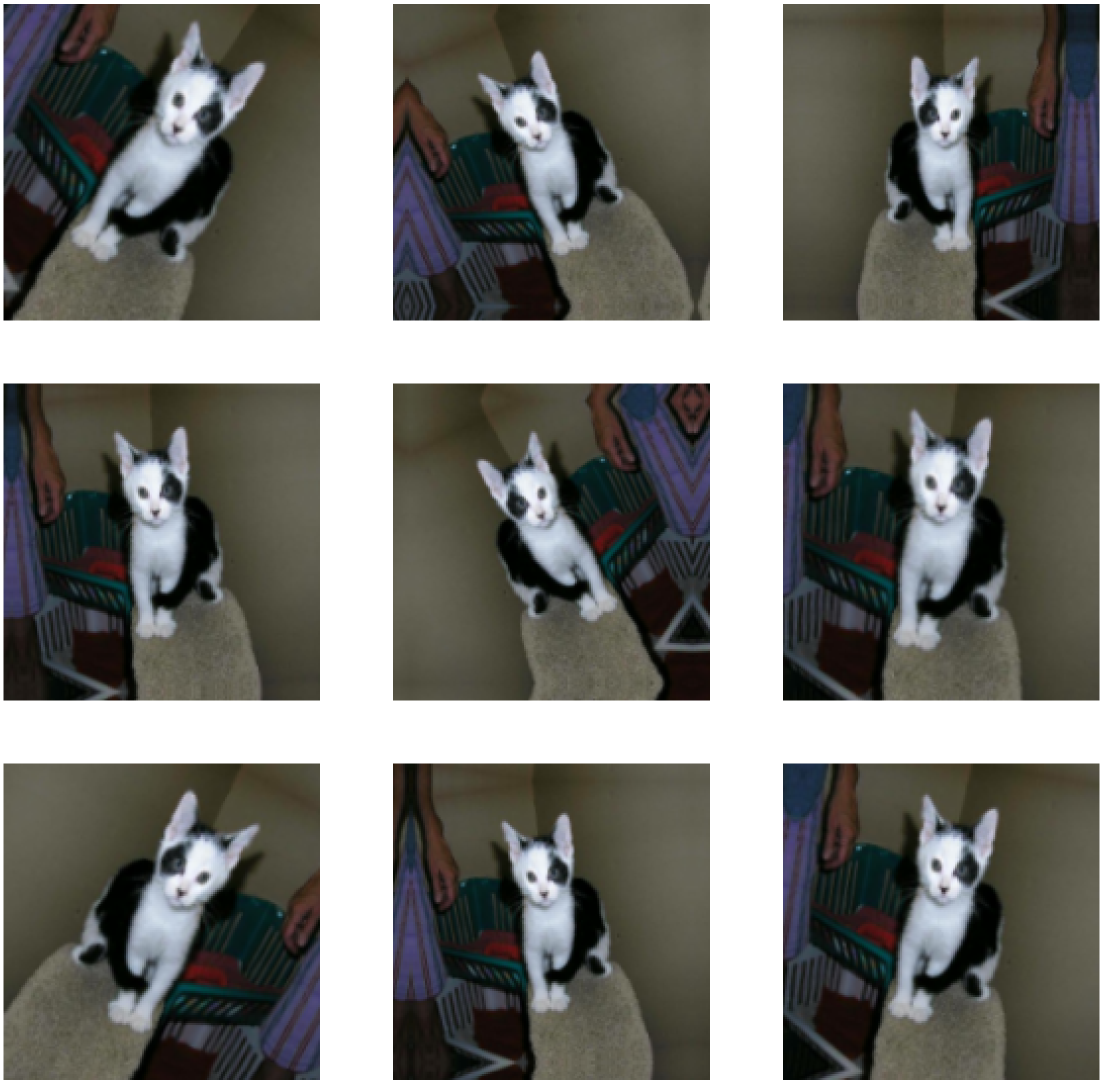
We increased the size of the training set from 1000 to 3000 for each classification, trained the original model, and found that the degree of overfitting was reduced. And the prediction accuracy has been greatly improved.

0.1.2 Build the new model with large training sample and data augmentation

Data augmentation

[9]:
```python
# define a data augmentation stage
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)
```

[10]:
```python
# display some 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")
```



[11]:

```
inputs = keras.Input(shape=(180, 180, 3))
x = data_augmentation(inputs)   # use data augmentation
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)    # use dropout
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

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

```
[12]: #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 6000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
```

```
[13]: callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_big_sample_with_augmentation.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=100,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/100
188/188 [==============================] - 6s 27ms/step - loss: 0.7068 -
accuracy: 0.5345 - val_loss: 0.6748 - val_accuracy: 0.5710
Epoch 2/100
188/188 [==============================] - 5s 25ms/step - loss: 0.6732 -
accuracy: 0.5987 - val_loss: 0.6290 - val_accuracy: 0.6270
Epoch 3/100
188/188 [==============================] - 5s 24ms/step - loss: 0.6416 -
```

```
accuracy: 0.6485 - val_loss: 0.5754 - val_accuracy: 0.6990
Epoch 4/100
188/188 [==============================] - 4s 23ms/step - loss: 0.5933 -
accuracy: 0.6925 - val_loss: 0.5859 - val_accuracy: 0.6980
Epoch 5/100
188/188 [==============================] - 5s 24ms/step - loss: 0.5621 -
accuracy: 0.7138 - val_loss: 0.5665 - val_accuracy: 0.6980
Epoch 6/100
188/188 [==============================] - 5s 23ms/step - loss: 0.5268 -
accuracy: 0.7468 - val_loss: 0.4822 - val_accuracy: 0.7690
Epoch 7/100
188/188 [==============================] - 4s 23ms/step - loss: 0.5046 -
accuracy: 0.7578 - val_loss: 0.5170 - val_accuracy: 0.7570
Epoch 8/100
188/188 [==============================] - 4s 23ms/step - loss: 0.5004 -
accuracy: 0.7627 - val_loss: 0.4459 - val_accuracy: 0.8020
Epoch 9/100
188/188 [==============================] - 5s 24ms/step - loss: 0.4766 -
accuracy: 0.7788 - val_loss: 0.4916 - val_accuracy: 0.7630
Epoch 10/100
188/188 [==============================] - 5s 25ms/step - loss: 0.4630 -
accuracy: 0.7795 - val_loss: 0.6910 - val_accuracy: 0.7150
Epoch 11/100
188/188 [==============================] - 5s 25ms/step - loss: 0.4440 -
accuracy: 0.7958 - val_loss: 0.4070 - val_accuracy: 0.8240
Epoch 12/100
188/188 [==============================] - 5s 24ms/step - loss: 0.4235 -
accuracy: 0.8100 - val_loss: 0.5303 - val_accuracy: 0.7590
Epoch 13/100
188/188 [==============================] - 5s 24ms/step - loss: 0.4199 -
accuracy: 0.8142 - val_loss: 0.3606 - val_accuracy: 0.8430
Epoch 14/100
188/188 [==============================] - 5s 25ms/step - loss: 0.3956 -
accuracy: 0.8267 - val_loss: 0.6295 - val_accuracy: 0.7490
Epoch 15/100
188/188 [==============================] - 5s 25ms/step - loss: 0.3888 -
accuracy: 0.8270 - val_loss: 0.3313 - val_accuracy: 0.8710
Epoch 16/100
188/188 [==============================] - 5s 25ms/step - loss: 0.3703 -
accuracy: 0.8385 - val_loss: 0.3225 - val_accuracy: 0.8710
Epoch 17/100
188/188 [==============================] - 5s 26ms/step - loss: 0.3745 -
accuracy: 0.8343 - val_loss: 0.3929 - val_accuracy: 0.8550
Epoch 18/100
188/188 [==============================] - 5s 26ms/step - loss: 0.3675 -
accuracy: 0.8403 - val_loss: 0.2962 - val_accuracy: 0.8900
Epoch 19/100
188/188 [==============================] - 5s 25ms/step - loss: 0.3510 -
```

```
accuracy: 0.8490 - val_loss: 0.3509 - val_accuracy: 0.8530
Epoch 20/100
188/188 [==============================] - 5s 26ms/step - loss: 0.3516 -
accuracy: 0.8475 - val_loss: 0.3913 - val_accuracy: 0.8440
Epoch 21/100
188/188 [==============================] - 5s 25ms/step - loss: 0.3527 -
accuracy: 0.8512 - val_loss: 0.3554 - val_accuracy: 0.8780
Epoch 22/100
188/188 [==============================] - 5s 25ms/step - loss: 0.3307 -
accuracy: 0.8588 - val_loss: 0.2837 - val_accuracy: 0.8930
Epoch 23/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3346 -
accuracy: 0.8610 - val_loss: 0.3812 - val_accuracy: 0.8190
Epoch 24/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3465 -
accuracy: 0.8607 - val_loss: 0.3135 - val_accuracy: 0.8920
Epoch 25/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3149 -
accuracy: 0.8642 - val_loss: 0.2706 - val_accuracy: 0.9020
Epoch 26/100
188/188 [==============================] - 5s 25ms/step - loss: 0.3274 -
accuracy: 0.8622 - val_loss: 0.3916 - val_accuracy: 0.8950
Epoch 27/100
188/188 [==============================] - 5s 25ms/step - loss: 0.3158 -
accuracy: 0.8740 - val_loss: 0.2715 - val_accuracy: 0.8980
Epoch 28/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3217 -
accuracy: 0.8662 - val_loss: 0.2951 - val_accuracy: 0.9010
Epoch 29/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3160 -
accuracy: 0.8678 - val_loss: 0.3324 - val_accuracy: 0.8890
Epoch 30/100
188/188 [==============================] - 5s 25ms/step - loss: 0.3218 -
accuracy: 0.8697 - val_loss: 0.2680 - val_accuracy: 0.8940
Epoch 31/100
188/188 [==============================] - 5s 24ms/step - loss: 0.2944 -
accuracy: 0.8798 - val_loss: 0.3707 - val_accuracy: 0.8630
Epoch 32/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3078 -
accuracy: 0.8730 - val_loss: 0.3748 - val_accuracy: 0.8710
Epoch 33/100
188/188 [==============================] - 5s 24ms/step - loss: 0.2970 -
accuracy: 0.8745 - val_loss: 0.2830 - val_accuracy: 0.8890
Epoch 34/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3191 -
accuracy: 0.8755 - val_loss: 0.3664 - val_accuracy: 0.8790
Epoch 35/100
188/188 [==============================] - 5s 25ms/step - loss: 0.2918 -
```

```
accuracy: 0.8807 - val_loss: 0.5897 - val_accuracy: 0.8150
Epoch 36/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3131 -
accuracy: 0.8698 - val_loss: 0.2511 - val_accuracy: 0.9110
Epoch 37/100
188/188 [==============================] - 5s 25ms/step - loss: 0.3033 -
accuracy: 0.8733 - val_loss: 0.2998 - val_accuracy: 0.9050
Epoch 38/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3081 -
accuracy: 0.8737 - val_loss: 0.3922 - val_accuracy: 0.8590
Epoch 39/100
188/188 [==============================] - 5s 25ms/step - loss: 0.2981 -
accuracy: 0.8732 - val_loss: 0.3542 - val_accuracy: 0.8550
Epoch 40/100
188/188 [==============================] - 5s 24ms/step - loss: 0.2989 -
accuracy: 0.8740 - val_loss: 0.2998 - val_accuracy: 0.9000
Epoch 41/100
188/188 [==============================] - 5s 25ms/step - loss: 0.2963 -
accuracy: 0.8815 - val_loss: 0.6574 - val_accuracy: 0.8670
Epoch 42/100
188/188 [==============================] - 5s 25ms/step - loss: 0.3174 -
accuracy: 0.8748 - val_loss: 0.4455 - val_accuracy: 0.8880
Epoch 43/100
188/188 [==============================] - 5s 25ms/step - loss: 0.3334 -
accuracy: 0.8723 - val_loss: 0.3546 - val_accuracy: 0.8650
Epoch 44/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3158 -
accuracy: 0.8745 - val_loss: 0.3068 - val_accuracy: 0.8980
Epoch 45/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3330 -
accuracy: 0.8723 - val_loss: 0.2762 - val_accuracy: 0.9020
Epoch 46/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3071 -
accuracy: 0.8747 - val_loss: 0.2585 - val_accuracy: 0.9020
Epoch 47/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3139 -
accuracy: 0.8743 - val_loss: 0.3289 - val_accuracy: 0.9030
Epoch 48/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3037 -
accuracy: 0.8750 - val_loss: 0.4795 - val_accuracy: 0.7940
Epoch 49/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3276 -
accuracy: 0.8662 - val_loss: 0.2599 - val_accuracy: 0.9080
Epoch 50/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3174 -
accuracy: 0.8762 - val_loss: 0.4917 - val_accuracy: 0.8880
Epoch 51/100
188/188 [==============================] - 5s 25ms/step - loss: 0.3226 -
```

```
accuracy: 0.8755 - val_loss: 0.2596 - val_accuracy: 0.9060
Epoch 52/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3535 -
accuracy: 0.8758 - val_loss: 0.3642 - val_accuracy: 0.8940
Epoch 53/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3563 -
accuracy: 0.8563 - val_loss: 0.5994 - val_accuracy: 0.8460
Epoch 54/100
188/188 [==============================] - 5s 27ms/step - loss: 0.3626 -
accuracy: 0.8615 - val_loss: 0.4274 - val_accuracy: 0.8730
Epoch 55/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3481 -
accuracy: 0.8595 - val_loss: 0.3944 - val_accuracy: 0.9050
Epoch 56/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3809 -
accuracy: 0.8595 - val_loss: 0.3301 - val_accuracy: 0.8850
Epoch 57/100
188/188 [==============================] - 5s 25ms/step - loss: 0.3549 -
accuracy: 0.8627 - val_loss: 0.5240 - val_accuracy: 0.8330
Epoch 58/100
188/188 [==============================] - 5s 26ms/step - loss: 0.3515 -
accuracy: 0.8633 - val_loss: 0.2896 - val_accuracy: 0.8930
Epoch 59/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3537 -
accuracy: 0.8593 - val_loss: 0.3723 - val_accuracy: 0.9020
Epoch 60/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3729 -
accuracy: 0.8488 - val_loss: 0.3423 - val_accuracy: 0.9130
Epoch 61/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3837 -
accuracy: 0.8428 - val_loss: 0.2808 - val_accuracy: 0.8960
Epoch 62/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3811 -
accuracy: 0.8478 - val_loss: 0.3233 - val_accuracy: 0.8610
Epoch 63/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3687 -
accuracy: 0.8450 - val_loss: 0.7174 - val_accuracy: 0.8730
Epoch 64/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3904 -
accuracy: 0.8490 - val_loss: 0.3290 - val_accuracy: 0.8990
Epoch 65/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3971 -
accuracy: 0.8433 - val_loss: 0.4285 - val_accuracy: 0.8830
Epoch 66/100
188/188 [==============================] - 5s 24ms/step - loss: 0.4162 -
accuracy: 0.8333 - val_loss: 0.3273 - val_accuracy: 0.8890
Epoch 67/100
188/188 [==============================] - 5s 24ms/step - loss: 0.3758 -
```

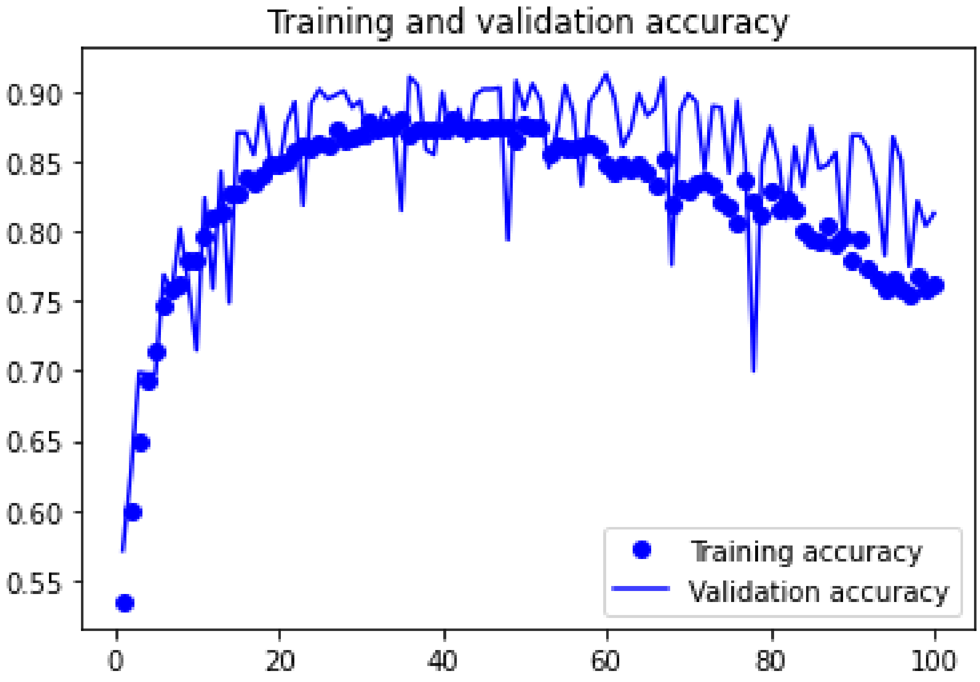
```
accuracy: 0.8523 - val_loss: 0.4767 - val_accuracy: 0.9100
Epoch 68/100
188/188 [==============================] - 5s 25ms/step - loss: 0.4731 -
accuracy: 0.8197 - val_loss: 0.4694 - val_accuracy: 0.7760
Epoch 69/100
188/188 [==============================] - 5s 24ms/step - loss: 0.4512 -
accuracy: 0.8308 - val_loss: 0.3268 - val_accuracy: 0.8860
Epoch 70/100
188/188 [==============================] - 5s 25ms/step - loss: 0.4264 -
accuracy: 0.8302 - val_loss: 0.4251 - val_accuracy: 0.8990
Epoch 71/100
188/188 [==============================] - 5s 25ms/step - loss: 0.4292 -
accuracy: 0.8345 - val_loss: 0.2989 - val_accuracy: 0.8930
Epoch 72/100
188/188 [==============================] - 5s 24ms/step - loss: 0.4247 -
accuracy: 0.8365 - val_loss: 0.4221 - val_accuracy: 0.8440
Epoch 73/100
188/188 [==============================] - 5s 24ms/step - loss: 0.4238 -
accuracy: 0.8322 - val_loss: 0.5305 - val_accuracy: 0.8900
Epoch 74/100
188/188 [==============================] - 5s 25ms/step - loss: 0.4485 -
accuracy: 0.8223 - val_loss: 0.4971 - val_accuracy: 0.8890
Epoch 75/100
188/188 [==============================] - 5s 25ms/step - loss: 0.4338 -
accuracy: 0.8180 - val_loss: 0.4373 - val_accuracy: 0.8420
Epoch 76/100
188/188 [==============================] - 5s 25ms/step - loss: 0.4834 -
accuracy: 0.8067 - val_loss: 0.2999 - val_accuracy: 0.8940
Epoch 77/100
188/188 [==============================] - 5s 25ms/step - loss: 0.4318 -
accuracy: 0.8367 - val_loss: 0.4074 - val_accuracy: 0.8500
Epoch 78/100
188/188 [==============================] - 5s 25ms/step - loss: 0.4577 -
accuracy: 0.8220 - val_loss: 1.2151 - val_accuracy: 0.7000
Epoch 79/100
188/188 [==============================] - 5s 24ms/step - loss: 0.4828 -
accuracy: 0.8118 - val_loss: 0.5328 - val_accuracy: 0.8480
Epoch 80/100
188/188 [==============================] - 5s 25ms/step - loss: 0.4772 -
accuracy: 0.8285 - val_loss: 0.4710 - val_accuracy: 0.8750
Epoch 81/100
188/188 [==============================] - 5s 25ms/step - loss: 0.4858 -
accuracy: 0.8153 - val_loss: 0.3951 - val_accuracy: 0.8510
Epoch 82/100
188/188 [==============================] - 5s 24ms/step - loss: 0.4565 -
accuracy: 0.8240 - val_loss: 0.5807 - val_accuracy: 0.8090
Epoch 83/100
188/188 [==============================] - 5s 24ms/step - loss: 0.5008 -
```

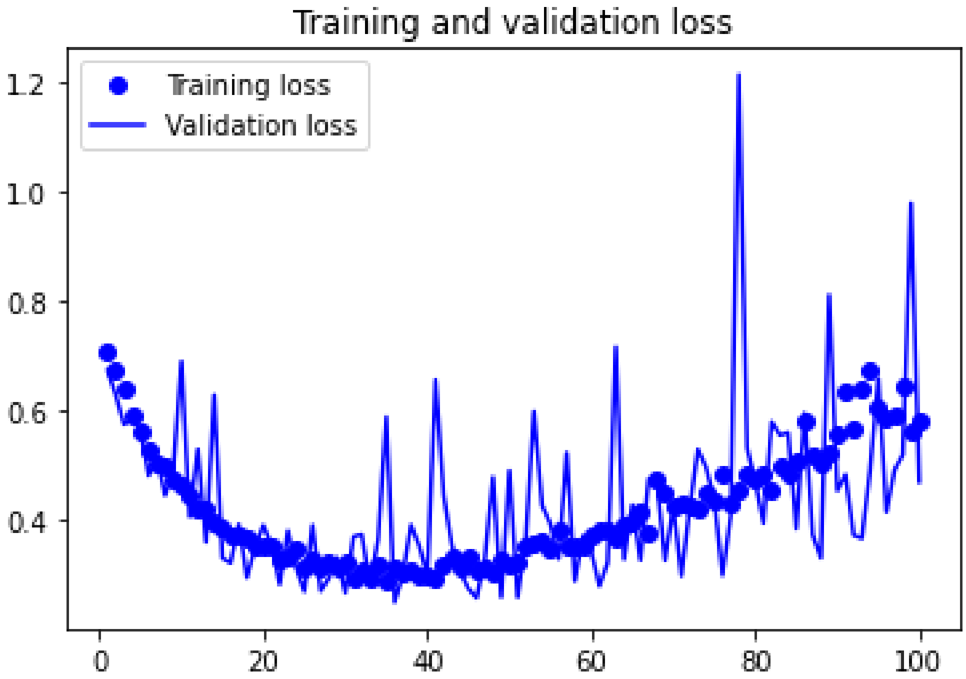
```
accuracy: 0.8160 - val_loss: 0.5563 - val_accuracy: 0.8610
Epoch 84/100
188/188 [==============================] - 5s 25ms/step - loss: 0.4830 -
accuracy: 0.8012 - val_loss: 0.5609 - val_accuracy: 0.8320
Epoch 85/100
188/188 [==============================] - 5s 24ms/step - loss: 0.5089 -
accuracy: 0.7948 - val_loss: 0.3848 - val_accuracy: 0.8750
Epoch 86/100
188/188 [==============================] - 5s 24ms/step - loss: 0.5799 -
accuracy: 0.7928 - val_loss: 0.5965 - val_accuracy: 0.8450
Epoch 87/100
188/188 [==============================] - 5s 23ms/step - loss: 0.5191 -
accuracy: 0.8037 - val_loss: 0.3742 - val_accuracy: 0.8480
Epoch 88/100
188/188 [==============================] - 5s 24ms/step - loss: 0.5022 -
accuracy: 0.7917 - val_loss: 0.3315 - val_accuracy: 0.8570
Epoch 89/100
188/188 [==============================] - 5s 24ms/step - loss: 0.5249 -
accuracy: 0.7968 - val_loss: 0.8121 - val_accuracy: 0.7930
Epoch 90/100
188/188 [==============================] - 5s 23ms/step - loss: 0.5569 -
accuracy: 0.7803 - val_loss: 0.4548 - val_accuracy: 0.8690
Epoch 91/100
188/188 [==============================] - 5s 25ms/step - loss: 0.6345 -
accuracy: 0.7943 - val_loss: 0.4839 - val_accuracy: 0.8690
Epoch 92/100
188/188 [==============================] - 5s 25ms/step - loss: 0.5692 -
accuracy: 0.7738 - val_loss: 0.3732 - val_accuracy: 0.8590
Epoch 93/100
188/188 [==============================] - 5s 25ms/step - loss: 0.6399 -
accuracy: 0.7663 - val_loss: 0.3674 - val_accuracy: 0.8320
Epoch 94/100
188/188 [==============================] - 5s 25ms/step - loss: 0.6764 -
accuracy: 0.7590 - val_loss: 0.4922 - val_accuracy: 0.7830
Epoch 95/100
188/188 [==============================] - 5s 25ms/step - loss: 0.6047 -
accuracy: 0.7653 - val_loss: 0.6592 - val_accuracy: 0.8680
Epoch 96/100
188/188 [==============================] - 5s 26ms/step - loss: 0.5853 -
accuracy: 0.7580 - val_loss: 0.4156 - val_accuracy: 0.8510
Epoch 97/100
188/188 [==============================] - 5s 25ms/step - loss: 0.5905 -
accuracy: 0.7543 - val_loss: 0.4929 - val_accuracy: 0.7750
Epoch 98/100
188/188 [==============================] - 5s 24ms/step - loss: 0.6440 -
accuracy: 0.7672 - val_loss: 0.5203 - val_accuracy: 0.8220
Epoch 99/100
188/188 [==============================] - 5s 24ms/step - loss: 0.5611 -
```

```
accuracy: 0.7587 - val_loss: 0.9802 - val_accuracy: 0.8040
Epoch 100/100
188/188 [==============================] - 5s 25ms/step - loss: 0.5820 -
accuracy: 0.7628 - val_loss: 0.4709 - val_accuracy: 0.8130
```

Display the curve of accuracy and loss during training

[14]:
```
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()
```





Evaluate model on test set

[15]:
```
test_model = keras.models.load_model(
    "convnet_from_scratch_big_sample_with_augmentation.keras")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 1s 15ms/step - loss: 0.3352 - accuracy:
0.8980
Test accuracy: 0.898
```

Question_3

October 22, 2023

0.1 Question 3: Changing the training sample to get the better performance

The beginning model use the training sample 1000 for each classification(cats and dogs), and in question_2, I increase the training sample from 1000 to 3000 in each dataset. The result shows that the test set prediction accuracy is improved.
This time, I decide to further increase the training sample to 5000 for each classification.

[1]:
```
# import the library
import os, shutil, pathlib
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras import layers
```

0.1.1 Resplit the train, valition and test data

[2]:
```
#original_dir = pathlib.Path("train")
new_base_dir = pathlib.Path("cats_vs_dogs_large")
'''
def make_subset(subset_name, start_index, end_index):
    for category in ("cat", "dog"):
        dir = new_base_dir / subset_name / category
        if os.path.exists(dir):
         shutil.rmtree(dir)
         os.mkdir(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)

make_subset("train", start_index=0, end_index=5000)  # increase training sample␣
 ↪set to 5000 for each class
make_subset("validation", start_index=5000, end_index=5500)
make_subset("test", start_index=5500, end_index=6000)'''
```

[2]:
```
'\ndef make_subset(subset_name, start_index, end_index):\n    for category in
("cat", "dog"):\n        dir = new_base_dir / subset_name / category\n        if
os.path.exists(dir):\n         shutil.rmtree(dir)\n         os.mkdir(dir)\n
```

```
fnames = [f"{category}.{i}.jpg" for i in range(start_index, end_index)]\n
for fname in fnames:\n            shutil.copyfile(src=original_dir / fname,\n
dst=dir / fname)\n\nmake_subset("train", start_index=0, end_index=5000)  #
increase training sample set to 5000 for each class\nmake_subset("validation",
start_index=5000, end_index=5500)\nmake_subset("test", start_index=5500,
end_index=6000)'
```

0.1.2 Build the model with more larger trainig set without data augmentation

[3]:
```
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=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.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"])
```

[4]:
```
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 10000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
```

[5]:

```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_extra_big_sample.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=50,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/50
313/313 [==============================] - 13s 26ms/step - loss: 0.6900 -
accuracy: 0.5586 - val_loss: 0.6523 - val_accuracy: 0.6180
Epoch 2/50
313/313 [==============================] - 7s 22ms/step - loss: 0.6187 -
accuracy: 0.6700 - val_loss: 0.5617 - val_accuracy: 0.7010
Epoch 3/50
313/313 [==============================] - 7s 22ms/step - loss: 0.5497 -
accuracy: 0.7300 - val_loss: 0.5109 - val_accuracy: 0.7500
Epoch 4/50
313/313 [==============================] - 7s 22ms/step - loss: 0.4861 -
accuracy: 0.7649 - val_loss: 0.4791 - val_accuracy: 0.7730
Epoch 5/50
313/313 [==============================] - 7s 22ms/step - loss: 0.4366 -
accuracy: 0.7957 - val_loss: 0.4166 - val_accuracy: 0.8140
Epoch 6/50
313/313 [==============================] - 7s 22ms/step - loss: 0.3893 -
accuracy: 0.8289 - val_loss: 0.5775 - val_accuracy: 0.7700
Epoch 7/50
313/313 [==============================] - 7s 23ms/step - loss: 0.3426 -
accuracy: 0.8511 - val_loss: 0.3994 - val_accuracy: 0.8170
Epoch 8/50
313/313 [==============================] - 7s 23ms/step - loss: 0.2982 -
accuracy: 0.8788 - val_loss: 0.7488 - val_accuracy: 0.7130
Epoch 9/50
313/313 [==============================] - 7s 23ms/step - loss: 0.2661 -
accuracy: 0.8914 - val_loss: 0.5911 - val_accuracy: 0.7880
Epoch 10/50
313/313 [==============================] - 7s 22ms/step - loss: 0.2275 -
accuracy: 0.9056 - val_loss: 0.4244 - val_accuracy: 0.8280
Epoch 11/50
313/313 [==============================] - 7s 22ms/step - loss: 0.1927 -
accuracy: 0.9216 - val_loss: 0.5551 - val_accuracy: 0.8250
Epoch 12/50
313/313 [==============================] - 7s 23ms/step - loss: 0.1729 -
```

```
accuracy: 0.9312 - val_loss: 0.5017 - val_accuracy: 0.8410
Epoch 13/50
313/313 [==============================] - 7s 22ms/step - loss: 0.1466 -
accuracy: 0.9431 - val_loss: 0.5648 - val_accuracy: 0.8460
Epoch 14/50
313/313 [==============================] - 7s 22ms/step - loss: 0.1321 -
accuracy: 0.9482 - val_loss: 0.4806 - val_accuracy: 0.8490
Epoch 15/50
313/313 [==============================] - 7s 22ms/step - loss: 0.1241 -
accuracy: 0.9540 - val_loss: 0.4791 - val_accuracy: 0.8520
Epoch 16/50
313/313 [==============================] - 7s 22ms/step - loss: 0.1158 -
accuracy: 0.9582 - val_loss: 0.5621 - val_accuracy: 0.8480
Epoch 17/50
313/313 [==============================] - 7s 22ms/step - loss: 0.1051 -
accuracy: 0.9610 - val_loss: 0.6024 - val_accuracy: 0.8470
Epoch 18/50
313/313 [==============================] - 7s 22ms/step - loss: 0.0988 -
accuracy: 0.9642 - val_loss: 0.5075 - val_accuracy: 0.8690
Epoch 19/50
313/313 [==============================] - 7s 23ms/step - loss: 0.1029 -
accuracy: 0.9650 - val_loss: 0.7051 - val_accuracy: 0.8250
Epoch 20/50
313/313 [==============================] - 7s 22ms/step - loss: 0.0880 -
accuracy: 0.9700 - val_loss: 0.5229 - val_accuracy: 0.8560
Epoch 21/50
313/313 [==============================] - 7s 22ms/step - loss: 0.0878 -
accuracy: 0.9707 - val_loss: 0.9779 - val_accuracy: 0.8470
Epoch 22/50
313/313 [==============================] - 7s 22ms/step - loss: 0.0958 -
accuracy: 0.9689 - val_loss: 0.5461 - val_accuracy: 0.8720
Epoch 23/50
313/313 [==============================] - 7s 22ms/step - loss: 0.0933 -
accuracy: 0.9692 - val_loss: 0.6228 - val_accuracy: 0.8710
Epoch 24/50
313/313 [==============================] - 7s 22ms/step - loss: 0.0915 -
accuracy: 0.9715 - val_loss: 0.6926 - val_accuracy: 0.8460
Epoch 25/50
313/313 [==============================] - 7s 22ms/step - loss: 0.0844 -
accuracy: 0.9727 - val_loss: 0.8928 - val_accuracy: 0.8570
Epoch 26/50
313/313 [==============================] - 7s 22ms/step - loss: 0.0863 -
accuracy: 0.9725 - val_loss: 0.9245 - val_accuracy: 0.8550
Epoch 27/50
313/313 [==============================] - 7s 22ms/step - loss: 0.0857 -
accuracy: 0.9754 - val_loss: 0.9017 - val_accuracy: 0.8720
Epoch 28/50
313/313 [==============================] - 7s 22ms/step - loss: 0.0930 -
```

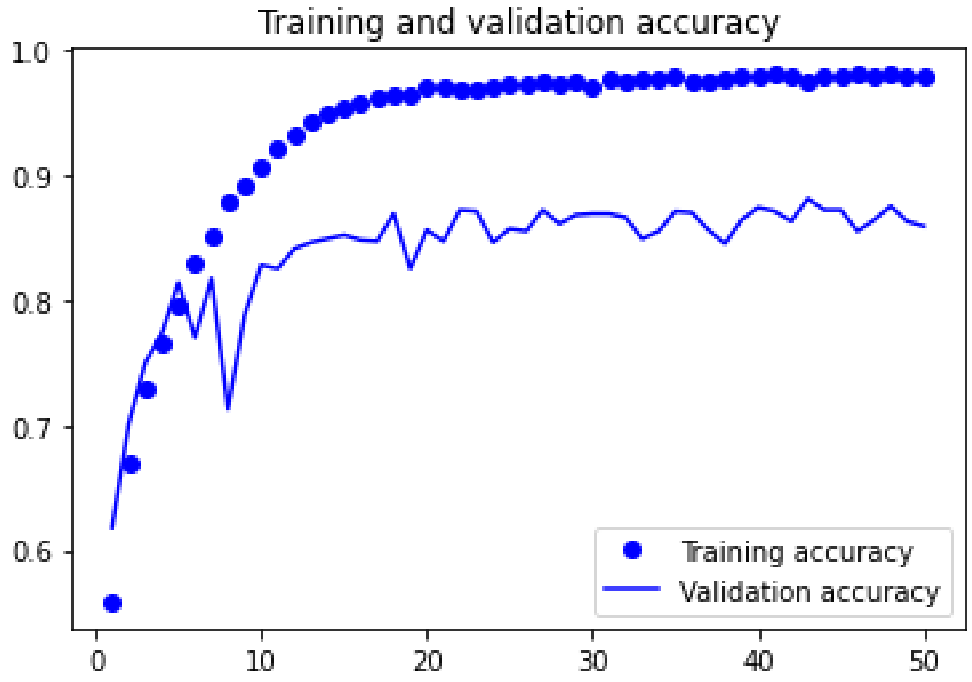
```
accuracy: 0.9730 - val_loss: 0.9246 - val_accuracy: 0.8610
Epoch 29/50
313/313 [==============================] - 7s 22ms/step - loss: 0.0992 -
accuracy: 0.9738 - val_loss: 0.7015 - val_accuracy: 0.8680
Epoch 30/50
313/313 [==============================] - 7s 22ms/step - loss: 0.1048 -
accuracy: 0.9701 - val_loss: 0.8366 - val_accuracy: 0.8690
Epoch 31/50
313/313 [==============================] - 7s 23ms/step - loss: 0.0949 -
accuracy: 0.9767 - val_loss: 0.9252 - val_accuracy: 0.8690
Epoch 32/50
313/313 [==============================] - 7s 22ms/step - loss: 0.1041 -
accuracy: 0.9741 - val_loss: 0.9140 - val_accuracy: 0.8660
Epoch 33/50
313/313 [==============================] - 7s 22ms/step - loss: 0.0851 -
accuracy: 0.9778 - val_loss: 1.3089 - val_accuracy: 0.8490
Epoch 34/50
313/313 [==============================] - 7s 22ms/step - loss: 0.0951 -
accuracy: 0.9766 - val_loss: 1.0647 - val_accuracy: 0.8550
Epoch 35/50
313/313 [==============================] - 7s 22ms/step - loss: 0.0791 -
accuracy: 0.9782 - val_loss: 1.1245 - val_accuracy: 0.8710
Epoch 36/50
313/313 [==============================] - 7s 22ms/step - loss: 0.1105 -
accuracy: 0.9739 - val_loss: 1.1627 - val_accuracy: 0.8700
Epoch 37/50
313/313 [==============================] - 7s 22ms/step - loss: 0.1133 -
accuracy: 0.9746 - val_loss: 1.0289 - val_accuracy: 0.8560
Epoch 38/50
313/313 [==============================] - 7s 23ms/step - loss: 0.1011 -
accuracy: 0.9764 - val_loss: 1.2074 - val_accuracy: 0.8450
Epoch 39/50
313/313 [==============================] - 7s 22ms/step - loss: 0.0909 -
accuracy: 0.9785 - val_loss: 1.0358 - val_accuracy: 0.8640
Epoch 40/50
313/313 [==============================] - 7s 22ms/step - loss: 0.1075 -
accuracy: 0.9784 - val_loss: 1.0881 - val_accuracy: 0.8740
Epoch 41/50
313/313 [==============================] - 7s 22ms/step - loss: 0.0948 -
accuracy: 0.9806 - val_loss: 1.1682 - val_accuracy: 0.8710
Epoch 42/50
313/313 [==============================] - 7s 23ms/step - loss: 0.0944 -
accuracy: 0.9790 - val_loss: 1.0997 - val_accuracy: 0.8630
Epoch 43/50
313/313 [==============================] - 7s 22ms/step - loss: 0.1071 -
accuracy: 0.9751 - val_loss: 0.8841 - val_accuracy: 0.8810
Epoch 44/50
313/313 [==============================] - 7s 22ms/step - loss: 0.1143 -
```

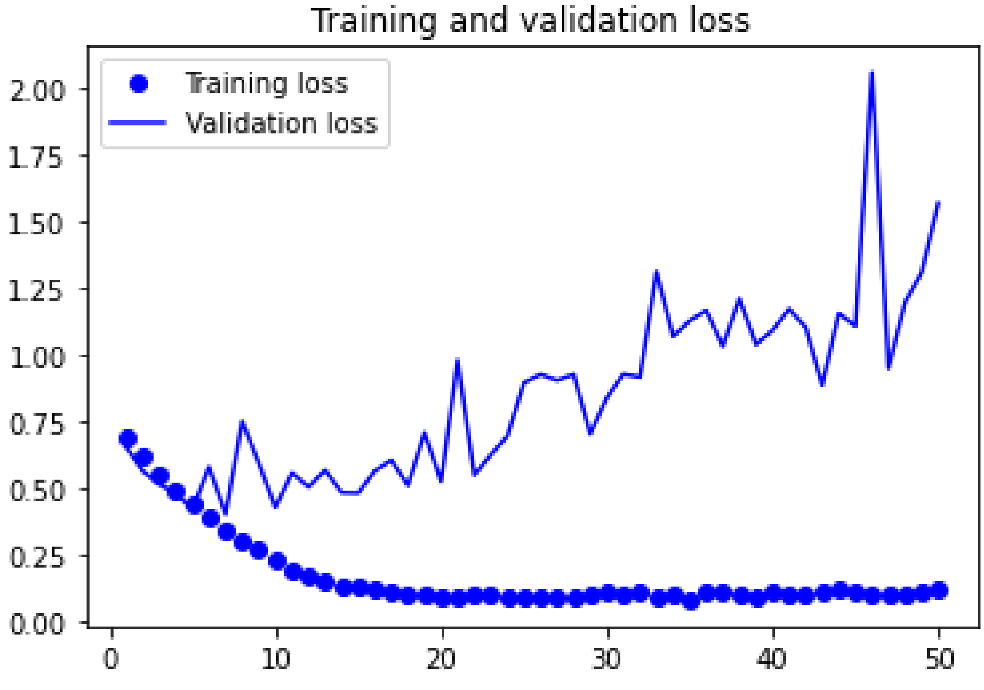
```
accuracy: 0.9780 - val_loss: 1.1524 - val_accuracy: 0.8720
Epoch 45/50
313/313 [==============================] - 7s 22ms/step - loss: 0.1111 -
accuracy: 0.9794 - val_loss: 1.1043 - val_accuracy: 0.8720
Epoch 46/50
313/313 [==============================] - 7s 23ms/step - loss: 0.0943 -
accuracy: 0.9802 - val_loss: 2.0557 - val_accuracy: 0.8550
Epoch 47/50
313/313 [==============================] - 7s 22ms/step - loss: 0.1017 -
accuracy: 0.9785 - val_loss: 0.9479 - val_accuracy: 0.8640
Epoch 48/50
313/313 [==============================] - 7s 23ms/step - loss: 0.0969 -
accuracy: 0.9801 - val_loss: 1.1941 - val_accuracy: 0.8750
Epoch 49/50
313/313 [==============================] - 7s 23ms/step - loss: 0.1043 -
accuracy: 0.9794 - val_loss: 1.3047 - val_accuracy: 0.8630
Epoch 50/50
313/313 [==============================] - 7s 23ms/step - loss: 0.1166 -
accuracy: 0.9789 - val_loss: 1.5654 - val_accuracy: 0.8590
```

Display the curve of accuracy and loss during training

[6]:

```python
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()
```





Evaluate on test set

[7]:
```
test_model = keras.models.load_model(
    "convnet_from_scratch_extra_big_sample.keras")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 2s 43ms/step - loss: 0.4017 - accuracy:
0.8430
Test accuracy: 0.843
```

0.1.3 Build the model with more larger training set and data augmentation

Data augmentation

[8]:
```
# define a data augmentation stage
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)
```

[9]:
```
# display some 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")
```

[10]:

```
inputs = keras.Input(shape=(180, 180, 3))
x = data_augmentation(inputs)   # use data augmentation
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)    # use dropout
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

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

[11]:
```
new_base_dir = pathlib.Path("cats_vs_dogs_large")
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 10000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
```

[12]:
```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_extra_big_sample_augmentation.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=100,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/100
313/313 [==============================] - 9s 24ms/step - loss: 0.6911 -
accuracy: 0.5501 - val_loss: 0.6381 - val_accuracy: 0.6510
Epoch 2/100
313/313 [==============================] - 8s 24ms/step - loss: 0.6283 -
accuracy: 0.6551 - val_loss: 1.7110 - val_accuracy: 0.5110
Epoch 3/100
313/313 [==============================] - 8s 24ms/step - loss: 0.5855 -
accuracy: 0.7006 - val_loss: 0.6168 - val_accuracy: 0.6580
```

```
Epoch 4/100
313/313 [==============================] - 8s 24ms/step - loss: 0.5296 -
accuracy: 0.7384 - val_loss: 0.7948 - val_accuracy: 0.6560
Epoch 5/100
313/313 [==============================] - 8s 24ms/step - loss: 0.4958 -
accuracy: 0.7658 - val_loss: 0.4487 - val_accuracy: 0.7730
Epoch 6/100
313/313 [==============================] - 7s 23ms/step - loss: 0.4626 -
accuracy: 0.7858 - val_loss: 0.4081 - val_accuracy: 0.8150
Epoch 7/100
313/313 [==============================] - 8s 24ms/step - loss: 0.4381 -
accuracy: 0.7995 - val_loss: 0.3678 - val_accuracy: 0.8270
Epoch 8/100
313/313 [==============================] - 7s 23ms/step - loss: 0.4273 -
accuracy: 0.8102 - val_loss: 0.3786 - val_accuracy: 0.8210
Epoch 9/100
313/313 [==============================] - 8s 24ms/step - loss: 0.4028 -
accuracy: 0.8194 - val_loss: 0.4059 - val_accuracy: 0.8010
Epoch 10/100
313/313 [==============================] - 7s 24ms/step - loss: 0.3809 -
accuracy: 0.8275 - val_loss: 0.3208 - val_accuracy: 0.8450
Epoch 11/100
313/313 [==============================] - 8s 24ms/step - loss: 0.3623 -
accuracy: 0.8397 - val_loss: 0.2671 - val_accuracy: 0.8810
Epoch 12/100
313/313 [==============================] - 8s 24ms/step - loss: 0.3446 -
accuracy: 0.8490 - val_loss: 0.3491 - val_accuracy: 0.8600
Epoch 13/100
313/313 [==============================] - 8s 24ms/step - loss: 0.3387 -
accuracy: 0.8581 - val_loss: 0.3265 - val_accuracy: 0.8640
Epoch 14/100
313/313 [==============================] - 8s 24ms/step - loss: 0.3268 -
accuracy: 0.8621 - val_loss: 0.2670 - val_accuracy: 0.8830
Epoch 15/100
313/313 [==============================] - 8s 24ms/step - loss: 0.3144 -
accuracy: 0.8683 - val_loss: 0.2740 - val_accuracy: 0.8880
Epoch 16/100
313/313 [==============================] - 8s 24ms/step - loss: 0.3211 -
accuracy: 0.8681 - val_loss: 0.2529 - val_accuracy: 0.8880
Epoch 17/100
313/313 [==============================] - 8s 24ms/step - loss: 0.3035 -
accuracy: 0.8710 - val_loss: 0.3580 - val_accuracy: 0.8710
Epoch 18/100
313/313 [==============================] - 8s 24ms/step - loss: 0.3073 -
accuracy: 0.8788 - val_loss: 0.3264 - val_accuracy: 0.8650
Epoch 19/100
313/313 [==============================] - 8s 24ms/step - loss: 0.3068 -
accuracy: 0.8733 - val_loss: 0.3118 - val_accuracy: 0.8630
```

```
Epoch 20/100
313/313 [==============================] - 7s 24ms/step - loss: 0.3216 -
accuracy: 0.8694 - val_loss: 0.4431 - val_accuracy: 0.8220
Epoch 21/100
313/313 [==============================] - 8s 24ms/step - loss: 0.3123 -
accuracy: 0.8682 - val_loss: 0.4056 - val_accuracy: 0.8620
Epoch 22/100
313/313 [==============================] - 8s 24ms/step - loss: 0.3265 -
accuracy: 0.8659 - val_loss: 2.8525 - val_accuracy: 0.6220
Epoch 23/100
313/313 [==============================] - 8s 25ms/step - loss: 0.3337 -
accuracy: 0.8675 - val_loss: 0.2719 - val_accuracy: 0.9050
Epoch 24/100
313/313 [==============================] - 8s 24ms/step - loss: 0.3668 -
accuracy: 0.8648 - val_loss: 0.2726 - val_accuracy: 0.8910
Epoch 25/100
313/313 [==============================] - 8s 24ms/step - loss: 0.3510 -
accuracy: 0.8528 - val_loss: 0.3550 - val_accuracy: 0.8560
Epoch 26/100
313/313 [==============================] - 8s 24ms/step - loss: 0.3396 -
accuracy: 0.8578 - val_loss: 0.2933 - val_accuracy: 0.8870
Epoch 27/100
313/313 [==============================] - 8s 24ms/step - loss: 0.3517 -
accuracy: 0.8548 - val_loss: 0.4418 - val_accuracy: 0.8910
Epoch 28/100
313/313 [==============================] - 8s 24ms/step - loss: 0.3750 -
accuracy: 0.8503 - val_loss: 0.6342 - val_accuracy: 0.7880
Epoch 29/100
313/313 [==============================] - 8s 24ms/step - loss: 0.4009 -
accuracy: 0.8440 - val_loss: 0.3187 - val_accuracy: 0.8820
Epoch 30/100
313/313 [==============================] - 8s 24ms/step - loss: 0.3756 -
accuracy: 0.8567 - val_loss: 0.2920 - val_accuracy: 0.8980
Epoch 31/100
313/313 [==============================] - 8s 24ms/step - loss: 0.3892 -
accuracy: 0.8435 - val_loss: 1.7900 - val_accuracy: 0.7560
Epoch 32/100
313/313 [==============================] - 8s 24ms/step - loss: 0.4068 -
accuracy: 0.8373 - val_loss: 2.1837 - val_accuracy: 0.6220
Epoch 33/100
313/313 [==============================] - 8s 24ms/step - loss: 0.4161 -
accuracy: 0.8330 - val_loss: 0.3196 - val_accuracy: 0.8660
Epoch 34/100
313/313 [==============================] - 8s 24ms/step - loss: 0.4143 -
accuracy: 0.8288 - val_loss: 0.3347 - val_accuracy: 0.8820
Epoch 35/100
313/313 [==============================] - 8s 24ms/step - loss: 0.4605 -
accuracy: 0.8223 - val_loss: 0.3712 - val_accuracy: 0.8250
```

```
Epoch 36/100
313/313 [==============================] - 8s 24ms/step - loss: 0.4534 -
accuracy: 0.8114 - val_loss: 0.3456 - val_accuracy: 0.8510
Epoch 37/100
313/313 [==============================] - 7s 23ms/step - loss: 0.4772 -
accuracy: 0.8147 - val_loss: 0.4180 - val_accuracy: 0.8170
Epoch 38/100
313/313 [==============================] - 7s 24ms/step - loss: 0.5468 -
accuracy: 0.8051 - val_loss: 0.3710 - val_accuracy: 0.8470
Epoch 39/100
313/313 [==============================] - 8s 24ms/step - loss: 0.4705 -
accuracy: 0.8111 - val_loss: 0.3639 - val_accuracy: 0.8440
Epoch 40/100
313/313 [==============================] - 8s 24ms/step - loss: 0.5597 -
accuracy: 0.8030 - val_loss: 0.3101 - val_accuracy: 0.8600
Epoch 41/100
313/313 [==============================] - 8s 24ms/step - loss: 0.5111 -
accuracy: 0.7913 - val_loss: 0.5411 - val_accuracy: 0.8060
Epoch 42/100
313/313 [==============================] - 7s 23ms/step - loss: 0.5971 -
accuracy: 0.7739 - val_loss: 0.3814 - val_accuracy: 0.8560
Epoch 43/100
313/313 [==============================] - 8s 24ms/step - loss: 0.5601 -
accuracy: 0.7793 - val_loss: 0.3024 - val_accuracy: 0.8780
Epoch 44/100
313/313 [==============================] - 8s 24ms/step - loss: 0.5200 -
accuracy: 0.7900 - val_loss: 0.5317 - val_accuracy: 0.8540
Epoch 45/100
313/313 [==============================] - 8s 24ms/step - loss: 0.5518 -
accuracy: 0.7911 - val_loss: 0.4125 - val_accuracy: 0.8130
Epoch 46/100
313/313 [==============================] - 8s 24ms/step - loss: 0.5575 -
accuracy: 0.7782 - val_loss: 0.5128 - val_accuracy: 0.8600
Epoch 47/100
313/313 [==============================] - 7s 23ms/step - loss: 0.5801 -
accuracy: 0.7665 - val_loss: 0.3169 - val_accuracy: 0.8610
Epoch 48/100
313/313 [==============================] - 7s 23ms/step - loss: 0.6898 -
accuracy: 0.7673 - val_loss: 0.3931 - val_accuracy: 0.8240
Epoch 49/100
313/313 [==============================] - 8s 24ms/step - loss: 0.6063 -
accuracy: 0.7722 - val_loss: 0.5211 - val_accuracy: 0.7800
Epoch 50/100
313/313 [==============================] - 8s 24ms/step - loss: 0.5793 -
accuracy: 0.7743 - val_loss: 0.3911 - val_accuracy: 0.8360
Epoch 51/100
313/313 [==============================] - 8s 24ms/step - loss: 0.6774 -
accuracy: 0.7764 - val_loss: 0.4827 - val_accuracy: 0.8320
```

```
Epoch 52/100
313/313 [==============================] - 8s 24ms/step - loss: 0.5514 -
accuracy: 0.7849 - val_loss: 0.9400 - val_accuracy: 0.8020
Epoch 53/100
313/313 [==============================] - 8s 24ms/step - loss: 0.6579 -
accuracy: 0.7593 - val_loss: 0.3682 - val_accuracy: 0.8160
Epoch 54/100
313/313 [==============================] - 8s 25ms/step - loss: 0.6196 -
accuracy: 0.7702 - val_loss: 1.4859 - val_accuracy: 0.7750
Epoch 55/100
313/313 [==============================] - 8s 24ms/step - loss: 0.7325 -
accuracy: 0.7811 - val_loss: 0.4734 - val_accuracy: 0.8310
Epoch 56/100
313/313 [==============================] - 8s 24ms/step - loss: 0.6196 -
accuracy: 0.7583 - val_loss: 0.5175 - val_accuracy: 0.7920
Epoch 57/100
313/313 [==============================] - 8s 24ms/step - loss: 0.7773 -
accuracy: 0.7569 - val_loss: 0.3619 - val_accuracy: 0.8580
Epoch 58/100
313/313 [==============================] - 8s 24ms/step - loss: 0.6234 -
accuracy: 0.7679 - val_loss: 0.4620 - val_accuracy: 0.8280
Epoch 59/100
313/313 [==============================] - 7s 23ms/step - loss: 0.6546 -
accuracy: 0.7458 - val_loss: 0.4088 - val_accuracy: 0.8150
Epoch 60/100
313/313 [==============================] - 7s 24ms/step - loss: 0.6850 -
accuracy: 0.7198 - val_loss: 0.3993 - val_accuracy: 0.8530
Epoch 61/100
313/313 [==============================] - 8s 24ms/step - loss: 1.0199 -
accuracy: 0.7395 - val_loss: 0.4642 - val_accuracy: 0.8340
Epoch 62/100
313/313 [==============================] - 7s 23ms/step - loss: 0.6997 -
accuracy: 0.7303 - val_loss: 0.4602 - val_accuracy: 0.8130
Epoch 63/100
313/313 [==============================] - 7s 23ms/step - loss: 0.7121 -
accuracy: 0.7331 - val_loss: 0.4869 - val_accuracy: 0.7890
Epoch 64/100
313/313 [==============================] - 7s 23ms/step - loss: 0.6996 -
accuracy: 0.7230 - val_loss: 0.3862 - val_accuracy: 0.8260
Epoch 65/100
313/313 [==============================] - 8s 24ms/step - loss: 0.6526 -
accuracy: 0.7404 - val_loss: 0.6430 - val_accuracy: 0.7480
Epoch 66/100
313/313 [==============================] - 8s 24ms/step - loss: 0.5881 -
accuracy: 0.7301 - val_loss: 0.4858 - val_accuracy: 0.7810
Epoch 67/100
313/313 [==============================] - 7s 23ms/step - loss: 0.7283 -
accuracy: 0.7255 - val_loss: 0.4399 - val_accuracy: 0.7950
```

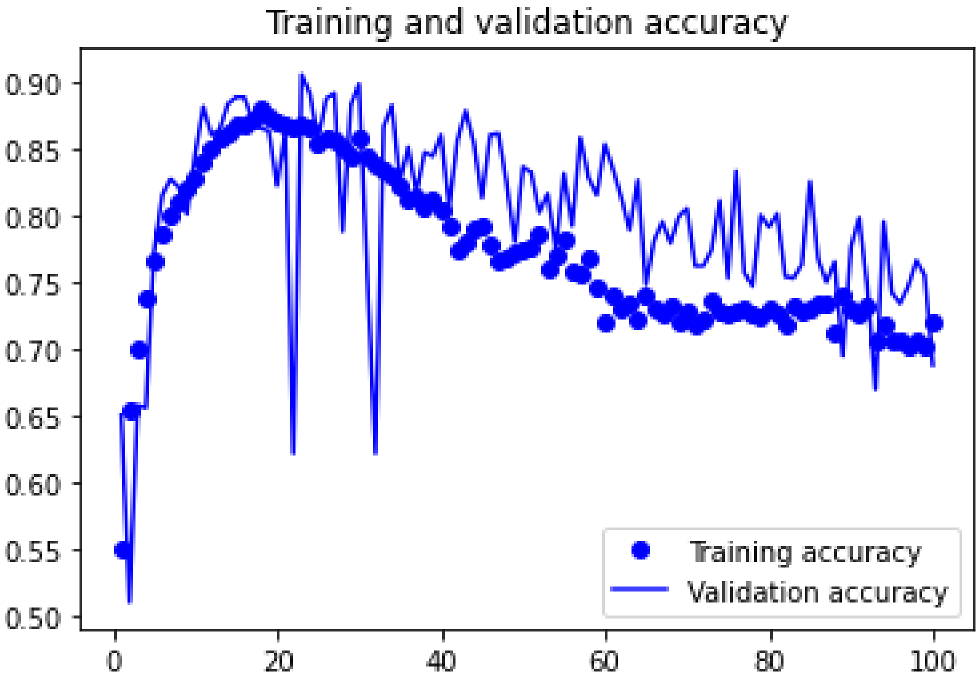
```
Epoch 68/100
313/313 [==============================] - 8s 24ms/step - loss: 0.6103 -
accuracy: 0.7323 - val_loss: 0.5108 - val_accuracy: 0.7790
Epoch 69/100
313/313 [==============================] - 8s 24ms/step - loss: 0.6621 -
accuracy: 0.7208 - val_loss: 0.4455 - val_accuracy: 0.7990
Epoch 70/100
313/313 [==============================] - 8s 24ms/step - loss: 0.6876 -
accuracy: 0.7271 - val_loss: 0.4700 - val_accuracy: 0.8050
Epoch 71/100
313/313 [==============================] - 7s 24ms/step - loss: 0.8256 -
accuracy: 0.7190 - val_loss: 0.5905 - val_accuracy: 0.7620
Epoch 72/100
313/313 [==============================] - 8s 24ms/step - loss: 0.8129 -
accuracy: 0.7225 - val_loss: 0.4713 - val_accuracy: 0.7620
Epoch 73/100
313/313 [==============================] - 8s 24ms/step - loss: 0.6715 -
accuracy: 0.7361 - val_loss: 0.4793 - val_accuracy: 0.7740
Epoch 74/100
313/313 [==============================] - 8s 24ms/step - loss: 0.6954 -
accuracy: 0.7275 - val_loss: 0.4871 - val_accuracy: 0.8110
Epoch 75/100
313/313 [==============================] - 8s 25ms/step - loss: 0.9392 -
accuracy: 0.7254 - val_loss: 0.6903 - val_accuracy: 0.7530
Epoch 76/100
313/313 [==============================] - 8s 24ms/step - loss: 0.6709 -
accuracy: 0.7273 - val_loss: 0.6284 - val_accuracy: 0.8330
Epoch 77/100
313/313 [==============================] - 8s 24ms/step - loss: 0.6899 -
accuracy: 0.7304 - val_loss: 0.5768 - val_accuracy: 0.7570
Epoch 78/100
313/313 [==============================] - 8s 24ms/step - loss: 0.8521 -
accuracy: 0.7269 - val_loss: 2.5417 - val_accuracy: 0.7470
Epoch 79/100
313/313 [==============================] - 7s 23ms/step - loss: 0.6282 -
accuracy: 0.7245 - val_loss: 0.4343 - val_accuracy: 0.8000
Epoch 80/100
313/313 [==============================] - 7s 24ms/step - loss: 0.7263 -
accuracy: 0.7297 - val_loss: 0.5238 - val_accuracy: 0.7910
Epoch 81/100
313/313 [==============================] - 7s 24ms/step - loss: 0.6971 -
accuracy: 0.7264 - val_loss: 0.4669 - val_accuracy: 0.8010
Epoch 82/100
313/313 [==============================] - 7s 23ms/step - loss: 0.6804 -
accuracy: 0.7175 - val_loss: 0.5627 - val_accuracy: 0.7540
Epoch 83/100
313/313 [==============================] - 8s 24ms/step - loss: 0.6460 -
accuracy: 0.7330 - val_loss: 0.7470 - val_accuracy: 0.7530
```

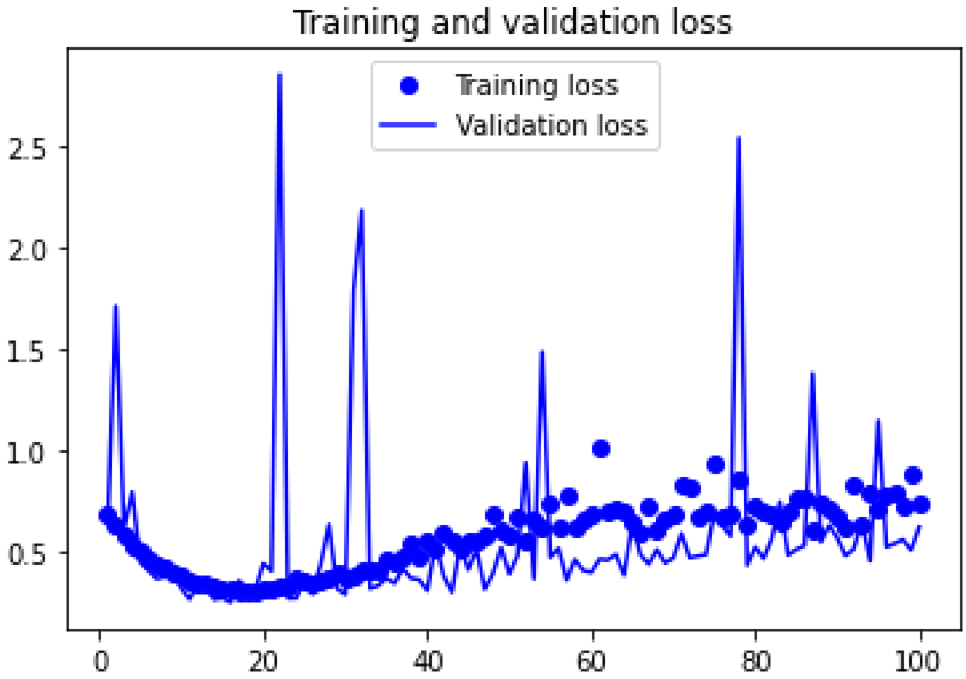
```
Epoch 84/100
313/313 [==============================] - 8s 24ms/step - loss: 0.6983 -
accuracy: 0.7274 - val_loss: 0.4844 - val_accuracy: 0.7630
Epoch 85/100
313/313 [==============================] - 8s 24ms/step - loss: 0.7670 -
accuracy: 0.7302 - val_loss: 0.5112 - val_accuracy: 0.8250
Epoch 86/100
313/313 [==============================] - 8s 25ms/step - loss: 0.7656 -
accuracy: 0.7336 - val_loss: 0.5315 - val_accuracy: 0.7670
Epoch 87/100
313/313 [==============================] - 7s 23ms/step - loss: 0.6101 -
accuracy: 0.7338 - val_loss: 1.3804 - val_accuracy: 0.7500
Epoch 88/100
313/313 [==============================] - 8s 24ms/step - loss: 0.7403 -
accuracy: 0.7125 - val_loss: 0.5481 - val_accuracy: 0.7650
Epoch 89/100
313/313 [==============================] - 8s 24ms/step - loss: 0.7103 -
accuracy: 0.7398 - val_loss: 0.6298 - val_accuracy: 0.6950
Epoch 90/100
313/313 [==============================] - 7s 23ms/step - loss: 0.6685 -
accuracy: 0.7302 - val_loss: 0.5691 - val_accuracy: 0.7760
Epoch 91/100
313/313 [==============================] - 8s 24ms/step - loss: 0.6205 -
accuracy: 0.7266 - val_loss: 0.4812 - val_accuracy: 0.7980
Epoch 92/100
313/313 [==============================] - 8s 24ms/step - loss: 0.8319 -
accuracy: 0.7324 - val_loss: 0.5130 - val_accuracy: 0.7460
Epoch 93/100
313/313 [==============================] - 7s 24ms/step - loss: 0.6367 -
accuracy: 0.7063 - val_loss: 0.6478 - val_accuracy: 0.6700
Epoch 94/100
313/313 [==============================] - 8s 24ms/step - loss: 0.7931 -
accuracy: 0.7173 - val_loss: 0.4557 - val_accuracy: 0.7950
Epoch 95/100
313/313 [==============================] - 7s 23ms/step - loss: 0.7061 -
accuracy: 0.7059 - val_loss: 1.1462 - val_accuracy: 0.7420
Epoch 96/100
313/313 [==============================] - 8s 24ms/step - loss: 0.7727 -
accuracy: 0.7068 - val_loss: 0.5241 - val_accuracy: 0.7340
Epoch 97/100
313/313 [==============================] - 8s 24ms/step - loss: 0.7951 -
accuracy: 0.7023 - val_loss: 0.5410 - val_accuracy: 0.7470
Epoch 98/100
313/313 [==============================] - 7s 23ms/step - loss: 0.7221 -
accuracy: 0.7064 - val_loss: 0.5605 - val_accuracy: 0.7660
Epoch 99/100
313/313 [==============================] - 7s 23ms/step - loss: 0.8866 -
accuracy: 0.7027 - val_loss: 0.5087 - val_accuracy: 0.7550
```

```
Epoch 100/100
313/313 [==============================] - 8s 24ms/step - loss: 0.7326 -
accuracy: 0.7194 - val_loss: 0.6234 - val_accuracy: 0.6880
```

Display the curve of accuracy and loss during training

[13]:
```
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()
```





Evaluate model on test set

[14]:
```
test_model = keras.models.load_model(
    "convnet_from_scratch_extra_big_sample_augmentation.keras")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 1s 16ms/step - loss: 0.2351 - accuracy:
0.9040
Test accuracy: 0.904
```

Further increase training set does not improve the model much, advanced techniques need to be included.

Question_4_1

October 22, 2023

0.1 Question 4_1: Using the pretrained network

0.1.1 Model with fast feature extraction without data augmentation

[1]:
```
# import the library
import os, shutil, pathlib
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras import layers
# import the dataset
new_base_dir = pathlib.Path("cats_vs_dogs_small")
```

[2]:
```
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.
```

Initiate the VGG16 convolutional base

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

[4]:
```
conv_base.summary()
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 180, 180, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 180, 180, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 180, 180, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 90, 90, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 90, 90, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 90, 90, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 45, 45, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 45, 45, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 45, 45, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 45, 45, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 22, 22, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 22, 22, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 22, 22, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 22, 22, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 11, 11, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 11, 11, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 11, 11, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 11, 11, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 5, 5, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

Extract VGG16 features and corresponding labels

[5]:

```python
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)

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)
```

[6]:

```python
# shape of the extracted features
train_features.shape
```

[6]:

```
(2000, 5, 5, 512)
```

Define and train with the densely connected classifier

[7]:

```python
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"])

callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="feature_extraction_without_augmentation.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_features, train_labels,
    epochs=20,
    validation_data=(val_features, val_labels),
    callbacks=callbacks)
```

```
Epoch 1/20
63/63 [==============================] - 1s 7ms/step - loss: 15.0293 - accuracy:
```

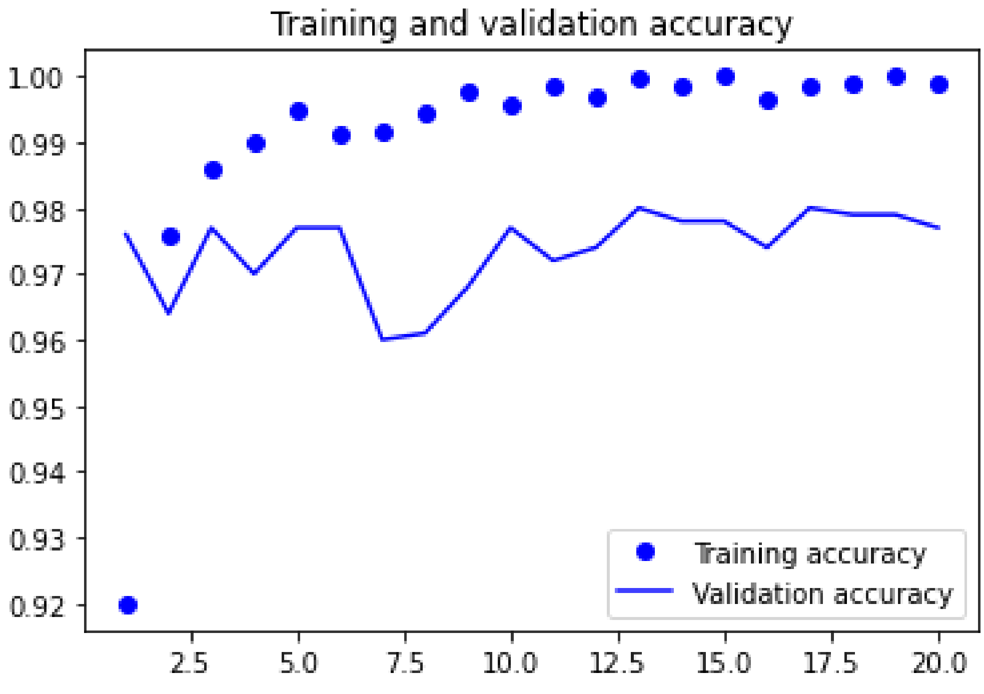
```
0.9200 - val_loss: 2.3054 - val_accuracy: 0.9760
Epoch 2/20
63/63 [==============================] - 0s 4ms/step - loss: 3.6233 - accuracy:
0.9760 - val_loss: 6.8050 - val_accuracy: 0.9640
Epoch 3/20
63/63 [==============================] - 0s 3ms/step - loss: 1.5489 - accuracy:
0.9860 - val_loss: 3.9630 - val_accuracy: 0.9770
Epoch 4/20
63/63 [==============================] - 0s 3ms/step - loss: 1.1687 - accuracy:
0.9900 - val_loss: 4.5758 - val_accuracy: 0.9700
Epoch 5/20
63/63 [==============================] - 0s 3ms/step - loss: 0.6211 - accuracy:
0.9950 - val_loss: 3.2959 - val_accuracy: 0.9770
Epoch 6/20
63/63 [==============================] - 0s 3ms/step - loss: 0.6381 - accuracy:
0.9910 - val_loss: 3.8667 - val_accuracy: 0.9770
Epoch 7/20
63/63 [==============================] - 0s 3ms/step - loss: 1.2350 - accuracy:
0.9915 - val_loss: 7.9491 - val_accuracy: 0.9600
Epoch 8/20
63/63 [==============================] - 0s 3ms/step - loss: 0.2576 - accuracy:
0.9945 - val_loss: 7.8166 - val_accuracy: 0.9610
Epoch 9/20
63/63 [==============================] - 0s 3ms/step - loss: 0.2636 - accuracy:
0.9975 - val_loss: 5.7002 - val_accuracy: 0.9680
Epoch 10/20
63/63 [==============================] - 0s 3ms/step - loss: 0.3717 - accuracy:
0.9955 - val_loss: 4.7412 - val_accuracy: 0.9770
Epoch 11/20
63/63 [==============================] - 0s 3ms/step - loss: 0.2124 - accuracy:
0.9985 - val_loss: 4.2118 - val_accuracy: 0.9720
Epoch 12/20
63/63 [==============================] - 0s 3ms/step - loss: 0.2208 - accuracy:
0.9970 - val_loss: 6.1981 - val_accuracy: 0.9740
Epoch 13/20
63/63 [==============================] - 0s 3ms/step - loss: 0.0571 - accuracy:
0.9995 - val_loss: 3.9218 - val_accuracy: 0.9800
Epoch 14/20
63/63 [==============================] - 0s 3ms/step - loss: 0.1088 - accuracy:
0.9985 - val_loss: 3.9260 - val_accuracy: 0.9780
Epoch 15/20
63/63 [==============================] - 0s 3ms/step - loss: 2.8149e-32 -
accuracy: 1.0000 - val_loss: 3.9260 - val_accuracy: 0.9780
Epoch 16/20
63/63 [==============================] - 0s 3ms/step - loss: 0.2586 - accuracy:
0.9965 - val_loss: 5.3130 - val_accuracy: 0.9740
Epoch 17/20
63/63 [==============================] - 0s 3ms/step - loss: 0.0578 - accuracy:
```

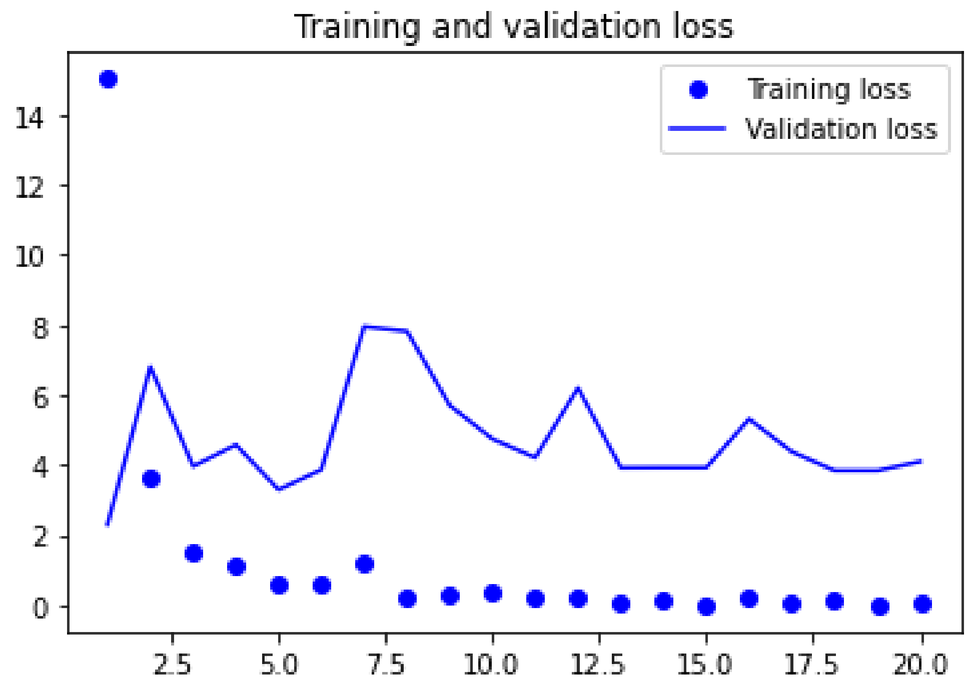
```
0.9985 - val_loss: 4.3786 - val_accuracy: 0.9800
Epoch 18/20
63/63 [==============================] - 0s 3ms/step - loss: 0.1304 - accuracy:
0.9990 - val_loss: 3.8411 - val_accuracy: 0.9790
Epoch 19/20
63/63 [==============================] - 0s 3ms/step - loss: 2.1655e-18 -
accuracy: 1.0000 - val_loss: 3.8411 - val_accuracy: 0.9790
Epoch 20/20
63/63 [==============================] - 0s 3ms/step - loss: 0.0374 - accuracy:
0.9990 - val_loss: 4.0988 - val_accuracy: 0.9770
```

Display the curve of accuracy and loss during training

[8]:

```python
acc = history.history["accuracy"]
val_acc = history.history["val_accuracy"]
loss = history.history["loss"]
val_loss = history.history["val_loss"]
epochs = range(1, len(acc) + 1)
plt.plot(epochs, acc, "bo", label="Training accuracy")
plt.plot(epochs, val_acc, "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()
```





The plot shows that the model is almost overfitting from the start because here we did not use the data augmentation which is important to prevent overfitting.

0.1.2 Model with feature extraction and data augmentation

Freeze the VGG16 convolutional base

[9]:
```
conv_base  = keras.applications.vgg16.VGG16(
    weights="imagenet",
    include_top=False)
conv_base.trainable = False  # empty trainable weights of the layer
```

[10]:
```
len(conv_base.trainable_weights) # double check the number of trainable weights
```

[10]:
```
0
```

Add the data augmentation stage and densely classifier to the conv base

[11]:
```
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
```

```
)

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"])
```

[12]:

```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="feature_extraction_with_data_augmentation.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=50,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/50
63/63 [==============================] - 5s 55ms/step - loss: 15.7831 -
accuracy: 0.8940 - val_loss: 5.9588 - val_accuracy: 0.9550
Epoch 2/50
63/63 [==============================] - 3s 52ms/step - loss: 8.4034 - accuracy:
0.9425 - val_loss: 4.7212 - val_accuracy: 0.9760
Epoch 3/50
63/63 [==============================] - 4s 56ms/step - loss: 6.4884 - accuracy:
0.9540 - val_loss: 3.1435 - val_accuracy: 0.9740
Epoch 4/50
63/63 [==============================] - 4s 53ms/step - loss: 5.1314 - accuracy:
0.9595 - val_loss: 3.9398 - val_accuracy: 0.9760
Epoch 5/50
63/63 [==============================] - 3s 52ms/step - loss: 4.6748 - accuracy:
0.9640 - val_loss: 3.6222 - val_accuracy: 0.9810
Epoch 6/50
63/63 [==============================] - 4s 55ms/step - loss: 3.1866 - accuracy:
0.9695 - val_loss: 4.8407 - val_accuracy: 0.9730
Epoch 7/50
63/63 [==============================] - 4s 56ms/step - loss: 3.2521 - accuracy:
```

```
0.9710 - val_loss: 3.3618 - val_accuracy: 0.9770
Epoch 8/50
63/63 [==============================] - 4s 53ms/step - loss: 3.1981 - accuracy:
0.9690 - val_loss: 4.4487 - val_accuracy: 0.9730
Epoch 9/50
63/63 [==============================] - 3s 52ms/step - loss: 2.6173 - accuracy:
0.9775 - val_loss: 3.9148 - val_accuracy: 0.9790
Epoch 10/50
63/63 [==============================] - 3s 52ms/step - loss: 3.2629 - accuracy:
0.9725 - val_loss: 3.1466 - val_accuracy: 0.9780
Epoch 11/50
63/63 [==============================] - 3s 52ms/step - loss: 2.0958 - accuracy:
0.9790 - val_loss: 4.7782 - val_accuracy: 0.9760
Epoch 12/50
63/63 [==============================] - 3s 52ms/step - loss: 2.3109 - accuracy:
0.9800 - val_loss: 2.6361 - val_accuracy: 0.9830
Epoch 13/50
63/63 [==============================] - 4s 56ms/step - loss: 2.0647 - accuracy:
0.9800 - val_loss: 8.4578 - val_accuracy: 0.9550
Epoch 14/50
63/63 [==============================] - 3s 52ms/step - loss: 2.5306 - accuracy:
0.9740 - val_loss: 2.3561 - val_accuracy: 0.9790
Epoch 15/50
63/63 [==============================] - 3s 52ms/step - loss: 2.0231 - accuracy:
0.9825 - val_loss: 3.2229 - val_accuracy: 0.9750
Epoch 16/50
63/63 [==============================] - 3s 52ms/step - loss: 1.1701 - accuracy:
0.9835 - val_loss: 3.0757 - val_accuracy: 0.9750
Epoch 17/50
63/63 [==============================] - 4s 56ms/step - loss: 1.6125 - accuracy:
0.9775 - val_loss: 2.9145 - val_accuracy: 0.9760
Epoch 18/50
63/63 [==============================] - 4s 52ms/step - loss: 1.3923 - accuracy:
0.9850 - val_loss: 4.0492 - val_accuracy: 0.9680
Epoch 19/50
63/63 [==============================] - 3s 53ms/step - loss: 1.5445 - accuracy:
0.9840 - val_loss: 3.4254 - val_accuracy: 0.9750
Epoch 20/50
63/63 [==============================] - 4s 60ms/step - loss: 1.0196 - accuracy:
0.9860 - val_loss: 1.4886 - val_accuracy: 0.9840
Epoch 21/50
63/63 [==============================] - 3s 52ms/step - loss: 1.5959 - accuracy:
0.9815 - val_loss: 2.6445 - val_accuracy: 0.9770
Epoch 22/50
63/63 [==============================] - 3s 52ms/step - loss: 1.5999 - accuracy:
0.9805 - val_loss: 2.0415 - val_accuracy: 0.9830
Epoch 23/50
63/63 [==============================] - 3s 52ms/step - loss: 1.8479 - accuracy:
```

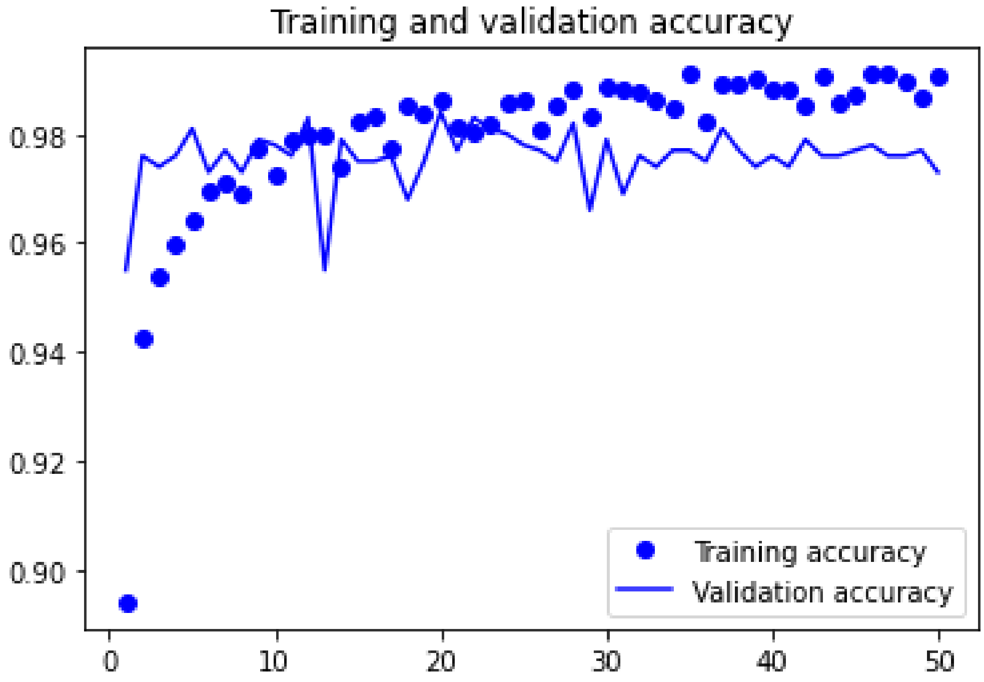
```
0.9820 - val_loss: 1.9757 - val_accuracy: 0.9810
Epoch 24/50
63/63 [==============================] - 3s 52ms/step - loss: 0.8278 - accuracy:
0.9855 - val_loss: 2.5043 - val_accuracy: 0.9800
Epoch 25/50
63/63 [==============================] - 3s 51ms/step - loss: 1.0837 - accuracy:
0.9860 - val_loss: 3.2428 - val_accuracy: 0.9780
Epoch 26/50
63/63 [==============================] - 3s 52ms/step - loss: 1.5470 - accuracy:
0.9810 - val_loss: 3.7802 - val_accuracy: 0.9770
Epoch 27/50
63/63 [==============================] - 4s 60ms/step - loss: 0.8201 - accuracy:
0.9850 - val_loss: 4.4545 - val_accuracy: 0.9750
Epoch 28/50
63/63 [==============================] - 3s 52ms/step - loss: 0.6363 - accuracy:
0.9880 - val_loss: 2.4569 - val_accuracy: 0.9820
Epoch 29/50
63/63 [==============================] - 4s 53ms/step - loss: 1.1598 - accuracy:
0.9835 - val_loss: 5.2461 - val_accuracy: 0.9660
Epoch 30/50
63/63 [==============================] - 3s 52ms/step - loss: 0.8538 - accuracy:
0.9885 - val_loss: 2.8245 - val_accuracy: 0.9790
Epoch 31/50
63/63 [==============================] - 4s 59ms/step - loss: 0.8028 - accuracy:
0.9880 - val_loss: 3.4184 - val_accuracy: 0.9690
Epoch 32/50
63/63 [==============================] - 4s 56ms/step - loss: 0.8688 - accuracy:
0.9875 - val_loss: 3.1664 - val_accuracy: 0.9760
Epoch 33/50
63/63 [==============================] - 3s 52ms/step - loss: 1.1711 - accuracy:
0.9860 - val_loss: 3.4489 - val_accuracy: 0.9740
Epoch 34/50
63/63 [==============================] - 3s 52ms/step - loss: 0.9261 - accuracy:
0.9845 - val_loss: 2.5860 - val_accuracy: 0.9770
Epoch 35/50
63/63 [==============================] - 3s 52ms/step - loss: 0.5101 - accuracy:
0.9910 - val_loss: 2.2665 - val_accuracy: 0.9770
Epoch 36/50
63/63 [==============================] - 4s 56ms/step - loss: 0.8975 - accuracy:
0.9825 - val_loss: 2.1496 - val_accuracy: 0.9750
Epoch 37/50
63/63 [==============================] - 3s 52ms/step - loss: 0.6774 - accuracy:
0.9890 - val_loss: 2.0931 - val_accuracy: 0.9810
Epoch 38/50
63/63 [==============================] - 4s 56ms/step - loss: 0.6366 - accuracy:
0.9890 - val_loss: 2.8960 - val_accuracy: 0.9770
Epoch 39/50
63/63 [==============================] - 3s 52ms/step - loss: 0.7178 - accuracy:
```

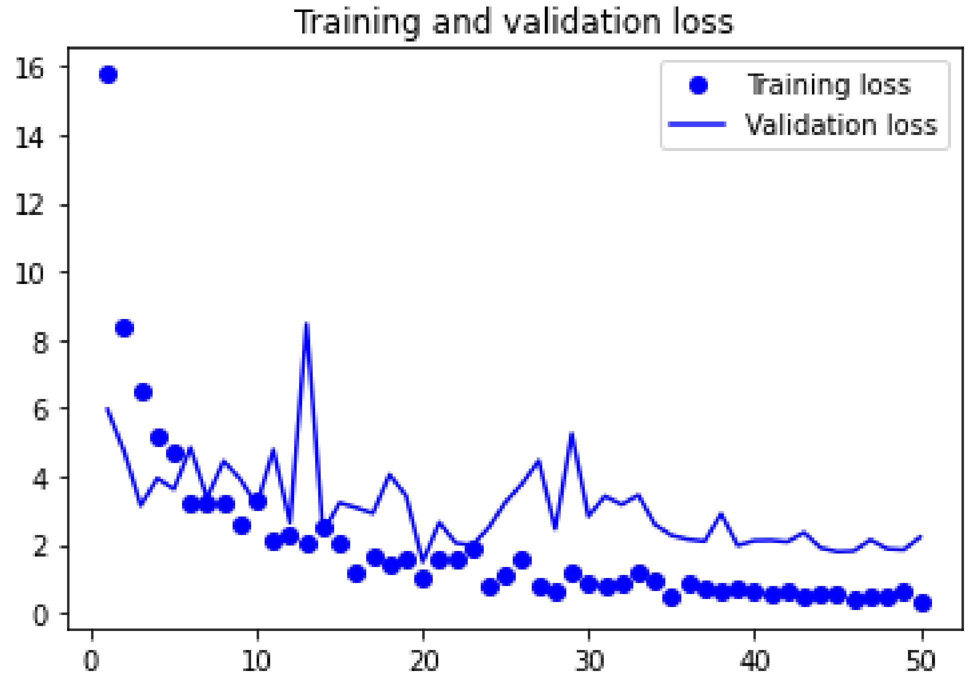
```
0.9900 - val_loss: 1.9683 - val_accuracy: 0.9740
Epoch 40/50
63/63 [==============================] - 4s 55ms/step - loss: 0.6636 - accuracy:
0.9880 - val_loss: 2.1129 - val_accuracy: 0.9760
Epoch 41/50
63/63 [==============================] - 3s 52ms/step - loss: 0.5388 - accuracy:
0.9880 - val_loss: 2.1225 - val_accuracy: 0.9740
Epoch 42/50
63/63 [==============================] - 3s 52ms/step - loss: 0.6593 - accuracy:
0.9850 - val_loss: 2.0755 - val_accuracy: 0.9790
Epoch 43/50
63/63 [==============================] - 3s 52ms/step - loss: 0.4833 - accuracy:
0.9905 - val_loss: 2.3506 - val_accuracy: 0.9760
Epoch 44/50
63/63 [==============================] - 3s 52ms/step - loss: 0.5945 - accuracy:
0.9855 - val_loss: 1.8998 - val_accuracy: 0.9760
Epoch 45/50
63/63 [==============================] - 3s 52ms/step - loss: 0.5284 - accuracy:
0.9870 - val_loss: 1.7880 - val_accuracy: 0.9770
Epoch 46/50
63/63 [==============================] - 4s 51ms/step - loss: 0.3932 - accuracy:
0.9910 - val_loss: 1.8064 - val_accuracy: 0.9780
Epoch 47/50
63/63 [==============================] - 3s 52ms/step - loss: 0.4414 - accuracy:
0.9910 - val_loss: 2.1360 - val_accuracy: 0.9760
Epoch 48/50
63/63 [==============================] - 4s 56ms/step - loss: 0.4818 - accuracy:
0.9895 - val_loss: 1.8735 - val_accuracy: 0.9760
Epoch 49/50
63/63 [==============================] - 3s 53ms/step - loss: 0.6704 - accuracy:
0.9865 - val_loss: 1.8425 - val_accuracy: 0.9770
Epoch 50/50
63/63 [==============================] - 3s 52ms/step - loss: 0.3286 - accuracy:
0.9905 - val_loss: 2.2160 - val_accuracy: 0.9730
```

Display the curve of accuracy and loss during training

[13]:
```
acc = history.history["accuracy"]
val_acc = history.history["val_accuracy"]
loss = history.history["loss"]
val_loss = history.history["val_loss"]
epochs = range(1, len(acc) + 1)
plt.plot(epochs, acc, "bo", label="Training accuracy")
plt.plot(epochs, val_acc, "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()
```





Evaluate model on test set

[14]:

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

```
32/32 [==============================] - 1s 32ms/step - loss: 3.7545 - accuracy:
0.9770
Test accuracy: 0.977
```

After the data augmentaion, the model overfitting problem is reduced and the model shows a strong improvement.

0.1.3 Model with fine-tuning to further improve the model

Freezing all layers except the fourth(last) conv layer

[15]:

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

Fine-tuning the model

[16]:

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

callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="fine_tuning.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 [==============================] - 6s 64ms/step - loss: 0.8203 - accuracy:
0.9840 - val_loss: 1.8919 - val_accuracy: 0.9780
Epoch 2/30
63/63 [==============================] - 4s 56ms/step - loss: 0.5354 - accuracy:
0.9900 - val_loss: 1.1257 - val_accuracy: 0.9780
Epoch 3/30
```

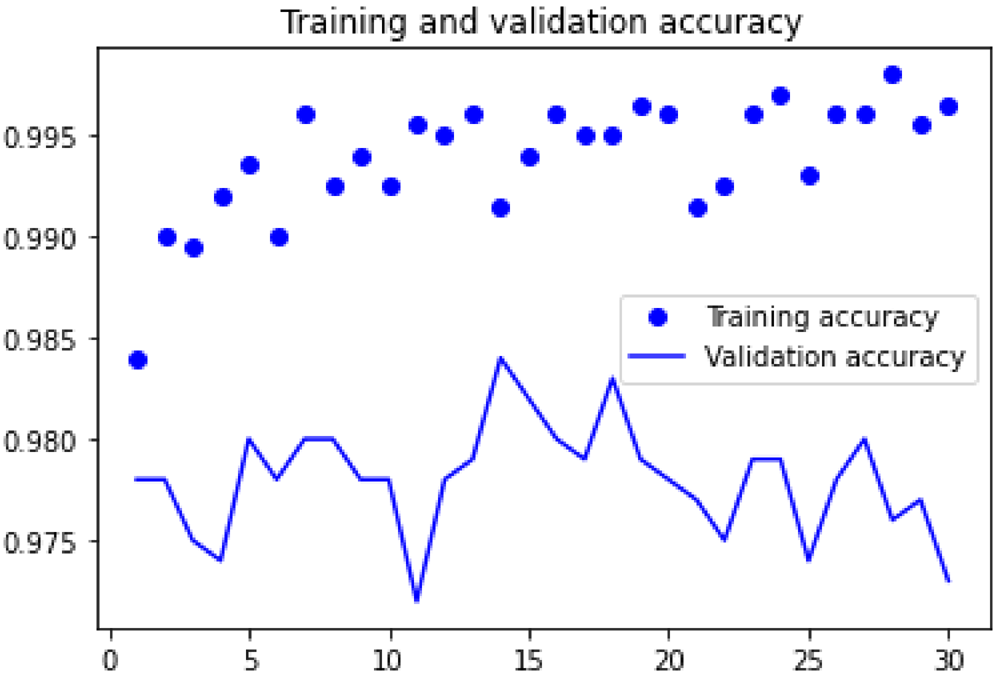
```
63/63 [==============================] - 4s 60ms/step - loss: 0.4262 - accuracy:
0.9895 - val_loss: 1.8802 - val_accuracy: 0.9750
Epoch 4/30
63/63 [==============================] - 4s 56ms/step - loss: 0.2004 - accuracy:
0.9920 - val_loss: 1.8672 - val_accuracy: 0.9740
Epoch 5/30
63/63 [==============================] - 4s 59ms/step - loss: 0.1941 - accuracy:
0.9935 - val_loss: 1.1593 - val_accuracy: 0.9800
Epoch 6/30
63/63 [==============================] - 4s 56ms/step - loss: 0.4592 - accuracy:
0.9900 - val_loss: 1.7764 - val_accuracy: 0.9780
Epoch 7/30
63/63 [==============================] - 4s 56ms/step - loss: 0.0796 - accuracy:
0.9960 - val_loss: 1.5191 - val_accuracy: 0.9800
Epoch 8/30
63/63 [==============================] - 4s 57ms/step - loss: 0.3456 - accuracy:
0.9925 - val_loss: 1.8870 - val_accuracy: 0.9800
Epoch 9/30
63/63 [==============================] - 4s 56ms/step - loss: 0.2386 - accuracy:
0.9940 - val_loss: 1.6848 - val_accuracy: 0.9780
Epoch 10/30
63/63 [==============================] - 4s 58ms/step - loss: 0.2783 - accuracy:
0.9925 - val_loss: 1.7178 - val_accuracy: 0.9780
Epoch 11/30
63/63 [==============================] - 4s 56ms/step - loss: 0.0904 - accuracy:
0.9955 - val_loss: 2.1791 - val_accuracy: 0.9720
Epoch 12/30
63/63 [==============================] - 4s 60ms/step - loss: 0.1961 - accuracy:
0.9950 - val_loss: 1.6213 - val_accuracy: 0.9780
Epoch 13/30
63/63 [==============================] - 4s 56ms/step - loss: 0.0644 - accuracy:
0.9960 - val_loss: 1.4284 - val_accuracy: 0.9790
Epoch 14/30
63/63 [==============================] - 4s 60ms/step - loss: 0.1808 - accuracy:
0.9915 - val_loss: 1.4207 - val_accuracy: 0.9840
Epoch 15/30
63/63 [==============================] - 4s 60ms/step - loss: 0.2266 - accuracy:
0.9940 - val_loss: 1.2127 - val_accuracy: 0.9820
Epoch 16/30
63/63 [==============================] - 4s 64ms/step - loss: 0.1200 - accuracy:
0.9960 - val_loss: 1.4560 - val_accuracy: 0.9800
Epoch 17/30
63/63 [==============================] - 4s 56ms/step - loss: 0.1203 - accuracy:
0.9950 - val_loss: 1.9027 - val_accuracy: 0.9790
Epoch 18/30
63/63 [==============================] - 4s 60ms/step - loss: 0.0744 - accuracy:
0.9950 - val_loss: 1.4723 - val_accuracy: 0.9830
Epoch 19/30
```

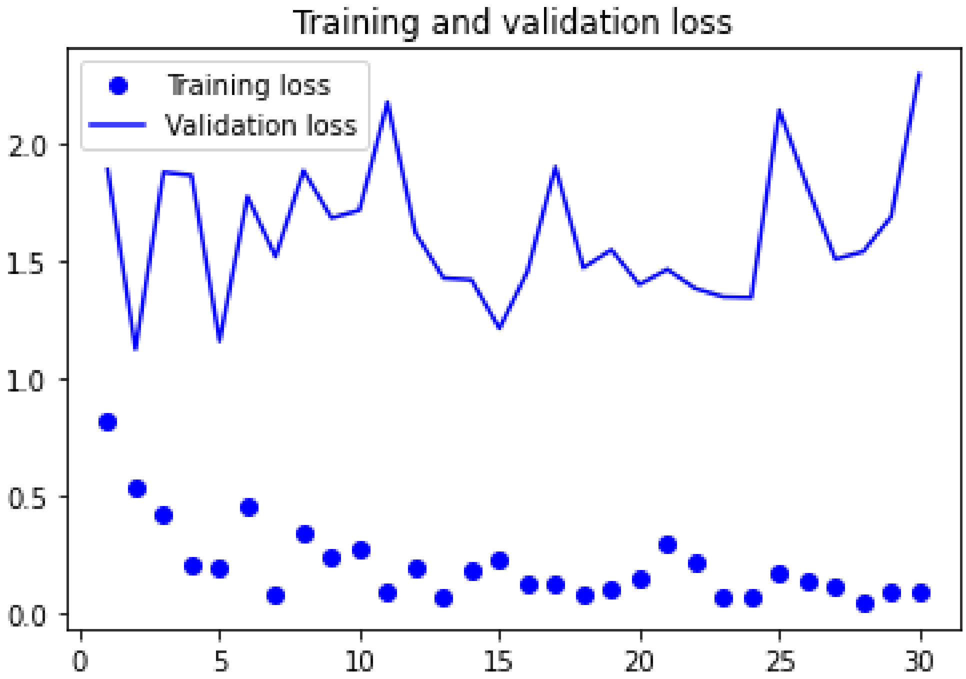
```
63/63 [==============================] - 4s 57ms/step - loss: 0.1075 - accuracy:
0.9965 - val_loss: 1.5490 - val_accuracy: 0.9790
Epoch 20/30
63/63 [==============================] - 4s 56ms/step - loss: 0.1420 - accuracy:
0.9960 - val_loss: 1.4008 - val_accuracy: 0.9780
Epoch 21/30
63/63 [==============================] - 4s 60ms/step - loss: 0.2953 - accuracy:
0.9915 - val_loss: 1.4651 - val_accuracy: 0.9770
Epoch 22/30
63/63 [==============================] - 4s 56ms/step - loss: 0.2129 - accuracy:
0.9925 - val_loss: 1.3837 - val_accuracy: 0.9750
Epoch 23/30
63/63 [==============================] - 5s 78ms/step - loss: 0.0714 - accuracy:
0.9960 - val_loss: 1.3481 - val_accuracy: 0.9790
Epoch 24/30
63/63 [==============================] - 4s 56ms/step - loss: 0.0683 - accuracy:
0.9970 - val_loss: 1.3450 - val_accuracy: 0.9790
Epoch 25/30
63/63 [==============================] - 4s 60ms/step - loss: 0.1687 - accuracy:
0.9930 - val_loss: 2.1449 - val_accuracy: 0.9740
Epoch 26/30
63/63 [==============================] - 4s 56ms/step - loss: 0.1396 - accuracy:
0.9960 - val_loss: 1.8180 - val_accuracy: 0.9780
Epoch 27/30
63/63 [==============================] - 4s 56ms/step - loss: 0.1174 - accuracy:
0.9960 - val_loss: 1.5088 - val_accuracy: 0.9800
Epoch 28/30
63/63 [==============================] - 4s 56ms/step - loss: 0.0460 - accuracy:
0.9980 - val_loss: 1.5415 - val_accuracy: 0.9760
Epoch 29/30
63/63 [==============================] - 4s 57ms/step - loss: 0.0953 - accuracy:
0.9955 - val_loss: 1.6906 - val_accuracy: 0.9770
Epoch 30/30
63/63 [==============================] - 4s 56ms/step - loss: 0.0942 - accuracy:
0.9965 - val_loss: 2.2966 - val_accuracy: 0.9730
```

Display the curve of accuracy and loss during training

[17]:
```python
acc = history.history["accuracy"]
val_acc = history.history["val_accuracy"]
loss = history.history["loss"]
val_loss = history.history["val_loss"]
epochs = range(1, len(acc) + 1)
plt.plot(epochs, acc, "bo", label="Training accuracy")
plt.plot(epochs, val_acc, "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()
```





Evaluate model on test set

[18]:
```
model = keras.models.load_model("fine_tuning.keras")
test_loss, test_acc = model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 1s 33ms/step - loss: 1.6147 - accuracy:
0.9790
Test accuracy: 0.979
```

The fine-tuning push the performance a bit further.

Question_4_2

October 22, 2023

0.1 Question 4_2: Using the pretrained network with large training sample

[1]:
```
# import the library
import os, shutil, pathlib
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras import layers
# import the dataset
new_base_dir = pathlib.Path("cats_vs_dogs_large")
```

[2]:
```
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 10000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
```

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

[4]:
```
conv_base.summary()
```

```
Model: "vgg16"
_________________________________________________________________
```

```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 180, 180, 3)]     0

_________________________________________________________________
block1_conv1 (Conv2D)        (None, 180, 180, 64)      1792

_________________________________________________________________
block1_conv2 (Conv2D)        (None, 180, 180, 64)      36928

_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 90, 90, 64)        0

_________________________________________________________________
block2_conv1 (Conv2D)        (None, 90, 90, 128)       73856

_________________________________________________________________
block2_conv2 (Conv2D)        (None, 90, 90, 128)       147584

_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 45, 45, 128)       0

_________________________________________________________________
block3_conv1 (Conv2D)        (None, 45, 45, 256)       295168

_________________________________________________________________
block3_conv2 (Conv2D)        (None, 45, 45, 256)       590080

_________________________________________________________________
block3_conv3 (Conv2D)        (None, 45, 45, 256)       590080

_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 22, 22, 256)       0

_________________________________________________________________
block4_conv1 (Conv2D)        (None, 22, 22, 512)       1180160

_________________________________________________________________
block4_conv2 (Conv2D)        (None, 22, 22, 512)       2359808

_________________________________________________________________
block4_conv3 (Conv2D)        (None, 22, 22, 512)       2359808

_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 11, 11, 512)       0

_________________________________________________________________
block5_conv1 (Conv2D)        (None, 11, 11, 512)       2359808

_________________________________________________________________
block5_conv2 (Conv2D)        (None, 11, 11, 512)       2359808

_________________________________________________________________
block5_conv3 (Conv2D)        (None, 11, 11, 512)       2359808

_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 5, 5, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0

_________________________________________________________________
```

[5]:

```
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)

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)
```

[6]:

```
train_features.shape
```

[6]:

```
(10000, 5, 5, 512)
```

[7]:

```
'''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"])

callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="feature_extraction_without_augmentation.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_features, train_labels,
    epochs=20,
    validation_data=(val_features, val_labels),
    callbacks=callbacks)'''
```

[7]:

```
'inputs = keras.Input(shape=(5, 5, 512))\nx = layers.Flatten()(inputs)\nx =
layers.Dense(256)(x)\nx = layers.Dropout(0.5)(x)\noutputs = layers.Dense(1,
activation="sigmoid")(x)\nmodel = keras.Model(inputs,
outputs)\nmodel.compile(loss="binary_crossentropy",\n
optimizer="rmsprop",\n              metrics=["accuracy"])\n\ncallbacks = [\n
keras.callbacks.ModelCheckpoint(\n
filepath="feature_extraction_without_augmentation.keras",\n
```

```
save_best_only=True,\n        monitor="val_loss")\n]\nhistory = model.fit(\n
train_features, train_labels,\n    epochs=20,\n
validation_data=(val_features, val_labels),\n    callbacks=callbacks)'
```

```
[8]: conv_base  = keras.applications.vgg16.VGG16(
         weights="imagenet",
         include_top=False)
     conv_base.trainable = False  # empty trainable weights of the layer
```

```
[9]: len(conv_base.trainable_weights) # double check the number of trainable weights
```

```
[9]: 0
```

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

      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"])
```

```
[11]: callbacks = [
          keras.callbacks.ModelCheckpoint(
              filepath="large_sample_feature_extraction_with_data_augmentation.keras",
              save_best_only=True,
              monitor="val_loss")
      ]
      history = model.fit(
          train_dataset,
          epochs=50,
          validation_data=validation_dataset,
          callbacks=callbacks)
```

```
Epoch 1/50
```

```
313/313 [==============================] - 12s 35ms/step - loss: 9.9548 -
accuracy: 0.9338 - val_loss: 10.8579 - val_accuracy: 0.9470
Epoch 2/50
313/313 [==============================] - 11s 35ms/step - loss: 5.4353 -
accuracy: 0.9504 - val_loss: 4.3261 - val_accuracy: 0.9680
Epoch 3/50
313/313 [==============================] - 11s 34ms/step - loss: 2.9153 -
accuracy: 0.9591 - val_loss: 2.4201 - val_accuracy: 0.9710
Epoch 4/50
313/313 [==============================] - 11s 35ms/step - loss: 1.4689 -
accuracy: 0.9613 - val_loss: 1.1375 - val_accuracy: 0.9710
Epoch 5/50
313/313 [==============================] - 11s 35ms/step - loss: 0.9186 -
accuracy: 0.9598 - val_loss: 0.6390 - val_accuracy: 0.9710
Epoch 6/50
313/313 [==============================] - 11s 35ms/step - loss: 0.7098 -
accuracy: 0.9622 - val_loss: 0.7237 - val_accuracy: 0.9750
Epoch 7/50
313/313 [==============================] - 11s 35ms/step - loss: 0.6647 -
accuracy: 0.9637 - val_loss: 0.9109 - val_accuracy: 0.9680
Epoch 8/50
313/313 [==============================] - 11s 35ms/step - loss: 0.6756 -
accuracy: 0.9660 - val_loss: 0.8537 - val_accuracy: 0.9710
Epoch 9/50
313/313 [==============================] - 11s 35ms/step - loss: 0.6770 -
accuracy: 0.9643 - val_loss: 1.0447 - val_accuracy: 0.9630
Epoch 10/50
313/313 [==============================] - 11s 35ms/step - loss: 0.6723 -
accuracy: 0.9667 - val_loss: 0.9784 - val_accuracy: 0.9710
Epoch 11/50
313/313 [==============================] - 11s 35ms/step - loss: 0.7050 -
accuracy: 0.9685 - val_loss: 1.1275 - val_accuracy: 0.9710
Epoch 12/50
313/313 [==============================] - 11s 35ms/step - loss: 0.7184 -
accuracy: 0.9667 - val_loss: 1.1155 - val_accuracy: 0.9750
Epoch 13/50
313/313 [==============================] - 11s 35ms/step - loss: 0.7012 -
accuracy: 0.9711 - val_loss: 0.7348 - val_accuracy: 0.9740
Epoch 14/50
313/313 [==============================] - 11s 35ms/step - loss: 0.6956 -
accuracy: 0.9688 - val_loss: 0.8136 - val_accuracy: 0.9750
Epoch 15/50
313/313 [==============================] - 11s 35ms/step - loss: 0.7067 -
accuracy: 0.9696 - val_loss: 2.7637 - val_accuracy: 0.9470
Epoch 16/50
313/313 [==============================] - 11s 35ms/step - loss: 0.7294 -
accuracy: 0.9698 - val_loss: 1.2325 - val_accuracy: 0.9710
Epoch 17/50
```

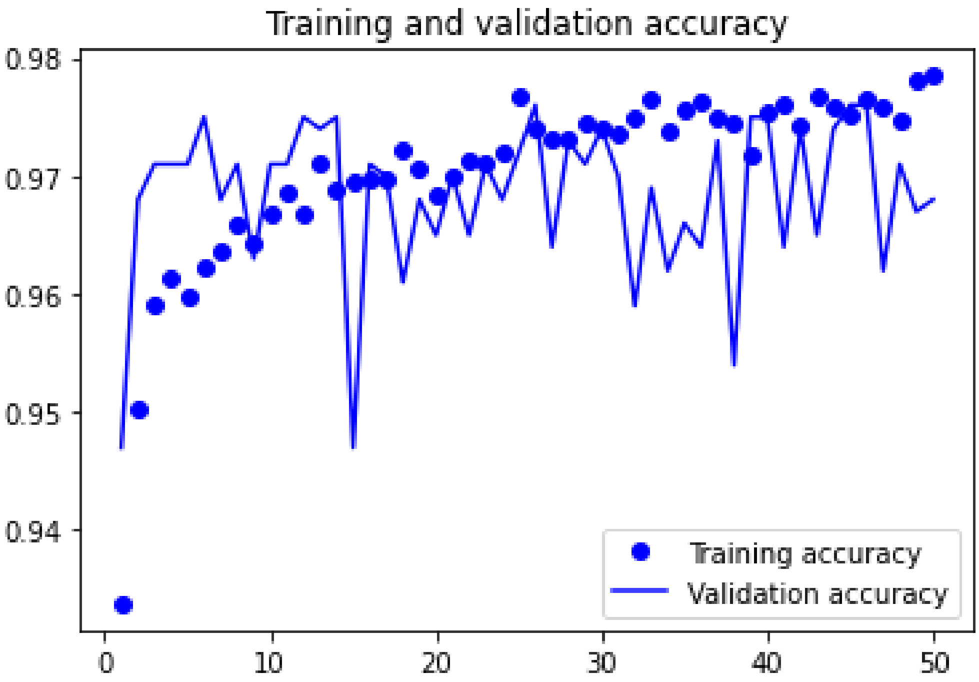
```
313/313 [==============================] - 11s 35ms/step - loss: 0.7902 -
accuracy: 0.9697 - val_loss: 1.4503 - val_accuracy: 0.9700
Epoch 18/50
313/313 [==============================] - 11s 35ms/step - loss: 0.8003 -
accuracy: 0.9722 - val_loss: 2.2824 - val_accuracy: 0.9610
Epoch 19/50
313/313 [==============================] - 11s 35ms/step - loss: 0.8226 -
accuracy: 0.9706 - val_loss: 1.5911 - val_accuracy: 0.9680
Epoch 20/50
313/313 [==============================] - 11s 35ms/step - loss: 0.8641 -
accuracy: 0.9683 - val_loss: 1.8121 - val_accuracy: 0.9650
Epoch 21/50
313/313 [==============================] - 11s 35ms/step - loss: 0.8303 -
accuracy: 0.9700 - val_loss: 1.5761 - val_accuracy: 0.9700
Epoch 22/50
313/313 [==============================] - 11s 35ms/step - loss: 0.8514 -
accuracy: 0.9714 - val_loss: 1.6323 - val_accuracy: 0.9650
Epoch 23/50
313/313 [==============================] - 11s 35ms/step - loss: 0.7226 -
accuracy: 0.9711 - val_loss: 1.6437 - val_accuracy: 0.9710
Epoch 24/50
313/313 [==============================] - 12s 37ms/step - loss: 0.7945 -
accuracy: 0.9721 - val_loss: 1.7327 - val_accuracy: 0.9680
Epoch 25/50
313/313 [==============================] - 11s 35ms/step - loss: 0.7455 -
accuracy: 0.9767 - val_loss: 1.8003 - val_accuracy: 0.9720
Epoch 26/50
313/313 [==============================] - 11s 35ms/step - loss: 0.8087 -
accuracy: 0.9740 - val_loss: 1.3962 - val_accuracy: 0.9760
Epoch 27/50
313/313 [==============================] - 11s 35ms/step - loss: 0.8226 -
accuracy: 0.9732 - val_loss: 1.6483 - val_accuracy: 0.9640
Epoch 28/50
313/313 [==============================] - 11s 35ms/step - loss: 0.8237 -
accuracy: 0.9732 - val_loss: 1.4034 - val_accuracy: 0.9730
Epoch 29/50
313/313 [==============================] - 11s 35ms/step - loss: 0.7573 -
accuracy: 0.9744 - val_loss: 2.1802 - val_accuracy: 0.9710
Epoch 30/50
313/313 [==============================] - 11s 35ms/step - loss: 0.8248 -
accuracy: 0.9740 - val_loss: 1.8767 - val_accuracy: 0.9740
Epoch 31/50
313/313 [==============================] - 11s 35ms/step - loss: 0.8757 -
accuracy: 0.9736 - val_loss: 2.3764 - val_accuracy: 0.9700
Epoch 32/50
313/313 [==============================] - 11s 35ms/step - loss: 0.8132 -
accuracy: 0.9750 - val_loss: 2.7868 - val_accuracy: 0.9590
Epoch 33/50
```

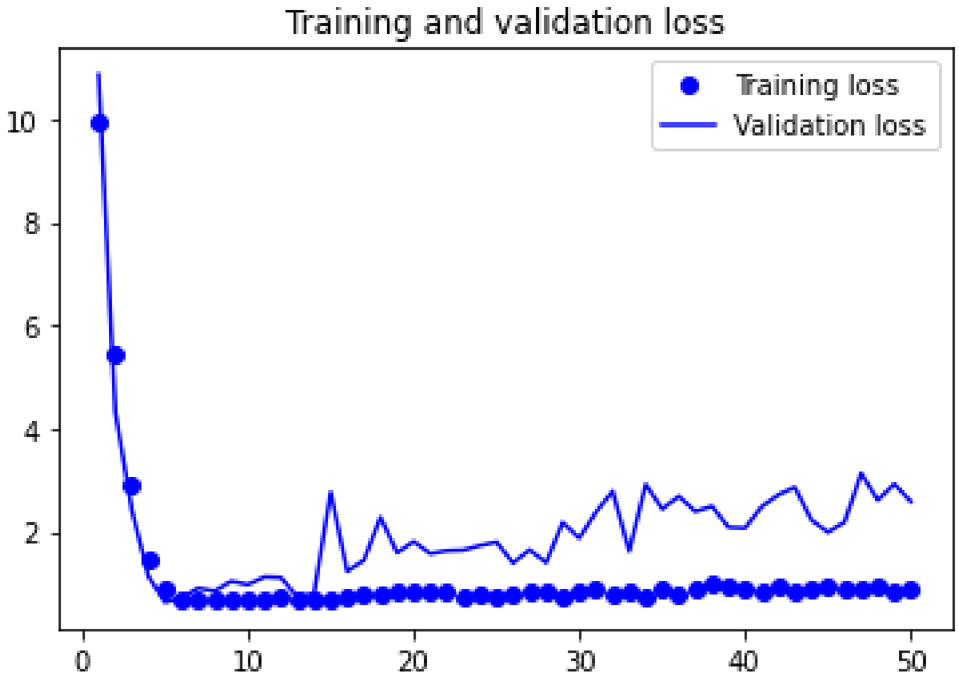
```
313/313 [==============================] - 11s 35ms/step - loss: 0.8663 -
accuracy: 0.9765 - val_loss: 1.6230 - val_accuracy: 0.9690
Epoch 34/50
313/313 [==============================] - 11s 35ms/step - loss: 0.7486 -
accuracy: 0.9739 - val_loss: 2.9162 - val_accuracy: 0.9620
Epoch 35/50
313/313 [==============================] - 11s 35ms/step - loss: 0.9120 -
accuracy: 0.9756 - val_loss: 2.4417 - val_accuracy: 0.9660
Epoch 36/50
313/313 [==============================] - 11s 35ms/step - loss: 0.7855 -
accuracy: 0.9764 - val_loss: 2.6882 - val_accuracy: 0.9640
Epoch 37/50
313/313 [==============================] - 11s 35ms/step - loss: 0.8787 -
accuracy: 0.9750 - val_loss: 2.3924 - val_accuracy: 0.9730
Epoch 38/50
313/313 [==============================] - 11s 35ms/step - loss: 1.0153 -
accuracy: 0.9744 - val_loss: 2.4948 - val_accuracy: 0.9540
Epoch 39/50
313/313 [==============================] - 11s 35ms/step - loss: 0.9633 -
accuracy: 0.9718 - val_loss: 2.0854 - val_accuracy: 0.9750
Epoch 40/50
313/313 [==============================] - 11s 35ms/step - loss: 0.8862 -
accuracy: 0.9754 - val_loss: 2.0734 - val_accuracy: 0.9750
Epoch 41/50
313/313 [==============================] - 11s 35ms/step - loss: 0.8198 -
accuracy: 0.9760 - val_loss: 2.4859 - val_accuracy: 0.9640
Epoch 42/50
313/313 [==============================] - 11s 35ms/step - loss: 0.9520 -
accuracy: 0.9743 - val_loss: 2.7150 - val_accuracy: 0.9740
Epoch 43/50
313/313 [==============================] - 11s 35ms/step - loss: 0.8242 -
accuracy: 0.9767 - val_loss: 2.8649 - val_accuracy: 0.9650
Epoch 44/50
313/313 [==============================] - 11s 35ms/step - loss: 0.8780 -
accuracy: 0.9758 - val_loss: 2.2286 - val_accuracy: 0.9740
Epoch 45/50
313/313 [==============================] - 11s 35ms/step - loss: 0.9379 -
accuracy: 0.9751 - val_loss: 1.9907 - val_accuracy: 0.9760
Epoch 46/50
313/313 [==============================] - 11s 35ms/step - loss: 0.8988 -
accuracy: 0.9765 - val_loss: 2.1868 - val_accuracy: 0.9760
Epoch 47/50
313/313 [==============================] - 11s 35ms/step - loss: 0.8864 -
accuracy: 0.9758 - val_loss: 3.1339 - val_accuracy: 0.9620
Epoch 48/50
313/313 [==============================] - 11s 35ms/step - loss: 0.9293 -
accuracy: 0.9747 - val_loss: 2.6128 - val_accuracy: 0.9710
Epoch 49/50
```

```
313/313 [==============================] - 11s 35ms/step - loss: 0.8373 -
accuracy: 0.9780 - val_loss: 2.9261 - val_accuracy: 0.9670
Epoch 50/50
313/313 [==============================] - 11s 35ms/step - loss: 0.8765 -
accuracy: 0.9785 - val_loss: 2.5842 - val_accuracy: 0.9680
```

[12]:

```
acc = history.history["accuracy"]
val_acc = history.history["val_accuracy"]
loss = history.history["loss"]
val_loss = history.history["val_loss"]
epochs = range(1, len(acc) + 1)
plt.plot(epochs, acc, "bo", label="Training accuracy")
plt.plot(epochs, val_acc, "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()
```





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

```
32/32 [==============================] - 1s 30ms/step - loss: 0.6662 - accuracy:
0.9760
Test accuracy: 0.976
```

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

```
[15]: model.compile(loss="binary_crossentropy",
                    optimizer=keras.optimizers.RMSprop(learning_rate=1e-5),
                    metrics=["accuracy"])

      callbacks = [
          keras.callbacks.ModelCheckpoint(
              filepath="fine_tuning.keras",
              save_best_only=True,
              monitor="val_loss")
      ]
      history = model.fit(
          train_dataset,
```

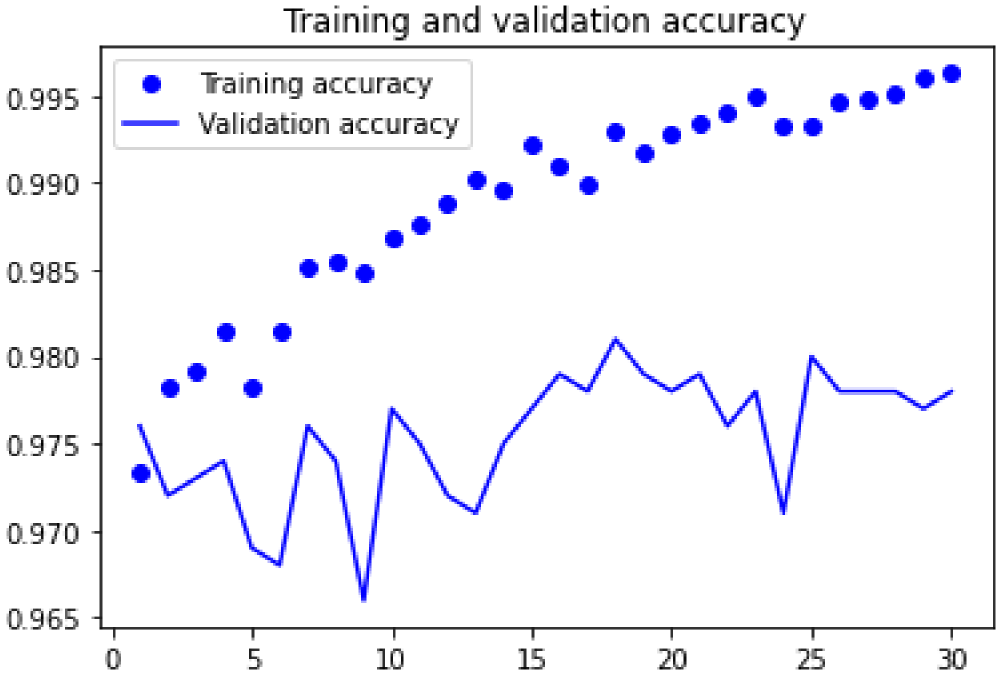
```
    epochs=30,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

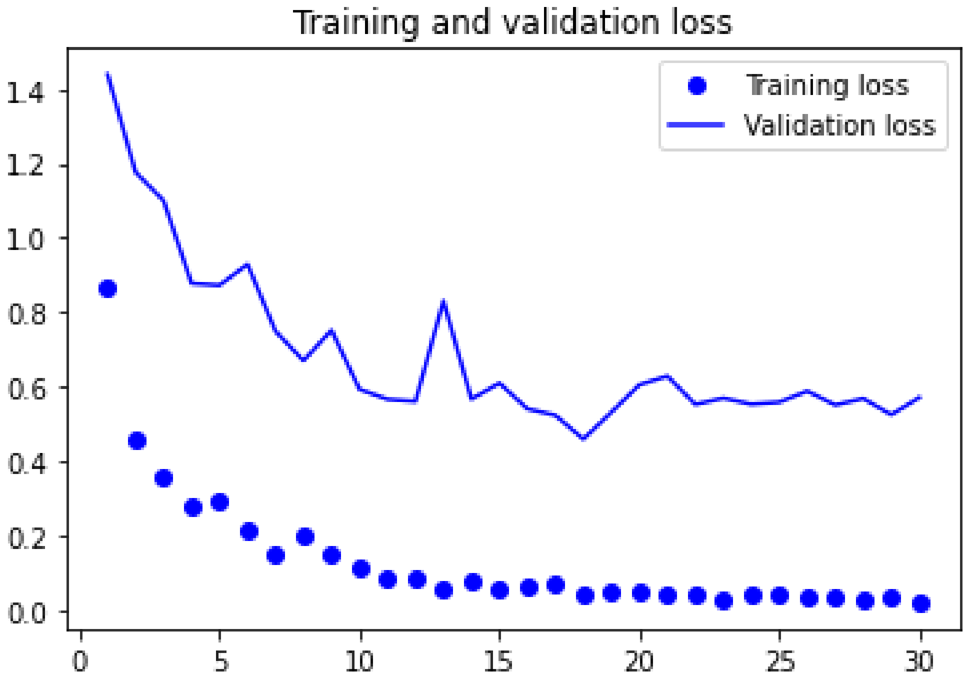
```
Epoch 1/30
313/313 [==============================] - 14s 40ms/step - loss: 0.8696 -
accuracy: 0.9734 - val_loss: 1.4392 - val_accuracy: 0.9760
Epoch 2/30
313/313 [==============================] - 12s 39ms/step - loss: 0.4602 -
accuracy: 0.9782 - val_loss: 1.1769 - val_accuracy: 0.9720
Epoch 3/30
313/313 [==============================] - 12s 39ms/step - loss: 0.3563 -
accuracy: 0.9792 - val_loss: 1.1005 - val_accuracy: 0.9730
Epoch 4/30
313/313 [==============================] - 12s 39ms/step - loss: 0.2767 -
accuracy: 0.9814 - val_loss: 0.8779 - val_accuracy: 0.9740
Epoch 5/30
313/313 [==============================] - 12s 39ms/step - loss: 0.2950 -
accuracy: 0.9782 - val_loss: 0.8730 - val_accuracy: 0.9690
Epoch 6/30
313/313 [==============================] - 12s 39ms/step - loss: 0.2163 -
accuracy: 0.9814 - val_loss: 0.9293 - val_accuracy: 0.9680
Epoch 7/30
313/313 [==============================] - 12s 39ms/step - loss: 0.1484 -
accuracy: 0.9852 - val_loss: 0.7493 - val_accuracy: 0.9760
Epoch 8/30
313/313 [==============================] - 12s 39ms/step - loss: 0.1982 -
accuracy: 0.9854 - val_loss: 0.6707 - val_accuracy: 0.9740
Epoch 9/30
313/313 [==============================] - 12s 39ms/step - loss: 0.1502 -
accuracy: 0.9848 - val_loss: 0.7521 - val_accuracy: 0.9660
Epoch 10/30
313/313 [==============================] - 12s 39ms/step - loss: 0.1157 -
accuracy: 0.9869 - val_loss: 0.5938 - val_accuracy: 0.9770
Epoch 11/30
313/313 [==============================] - 12s 39ms/step - loss: 0.0857 -
accuracy: 0.9876 - val_loss: 0.5667 - val_accuracy: 0.9750
Epoch 12/30
313/313 [==============================] - 12s 39ms/step - loss: 0.0824 -
accuracy: 0.9889 - val_loss: 0.5613 - val_accuracy: 0.9720
Epoch 13/30
313/313 [==============================] - 12s 39ms/step - loss: 0.0589 -
accuracy: 0.9902 - val_loss: 0.8309 - val_accuracy: 0.9710
Epoch 14/30
313/313 [==============================] - 12s 39ms/step - loss: 0.0755 -
accuracy: 0.9896 - val_loss: 0.5672 - val_accuracy: 0.9750
Epoch 15/30
```

```
313/313 [==============================] - 12s 39ms/step - loss: 0.0590 -
accuracy: 0.9922 - val_loss: 0.6104 - val_accuracy: 0.9770
Epoch 16/30
313/313 [==============================] - 12s 39ms/step - loss: 0.0657 -
accuracy: 0.9909 - val_loss: 0.5411 - val_accuracy: 0.9790
Epoch 17/30
313/313 [==============================] - 12s 39ms/step - loss: 0.0697 -
accuracy: 0.9899 - val_loss: 0.5239 - val_accuracy: 0.9780
Epoch 18/30
313/313 [==============================] - 12s 39ms/step - loss: 0.0430 -
accuracy: 0.9929 - val_loss: 0.4597 - val_accuracy: 0.9810
Epoch 19/30
313/313 [==============================] - 12s 39ms/step - loss: 0.0524 -
accuracy: 0.9918 - val_loss: 0.5321 - val_accuracy: 0.9790
Epoch 20/30
313/313 [==============================] - 12s 39ms/step - loss: 0.0490 -
accuracy: 0.9928 - val_loss: 0.6058 - val_accuracy: 0.9780
Epoch 21/30
313/313 [==============================] - 12s 39ms/step - loss: 0.0419 -
accuracy: 0.9934 - val_loss: 0.6292 - val_accuracy: 0.9790
Epoch 22/30
313/313 [==============================] - 12s 39ms/step - loss: 0.0393 -
accuracy: 0.9941 - val_loss: 0.5531 - val_accuracy: 0.9760
Epoch 23/30
313/313 [==============================] - 12s 39ms/step - loss: 0.0297 -
accuracy: 0.9949 - val_loss: 0.5699 - val_accuracy: 0.9780
Epoch 24/30
313/313 [==============================] - 12s 39ms/step - loss: 0.0446 -
accuracy: 0.9933 - val_loss: 0.5538 - val_accuracy: 0.9710
Epoch 25/30
313/313 [==============================] - 12s 39ms/step - loss: 0.0436 -
accuracy: 0.9933 - val_loss: 0.5591 - val_accuracy: 0.9800
Epoch 26/30
313/313 [==============================] - 12s 39ms/step - loss: 0.0353 -
accuracy: 0.9946 - val_loss: 0.5888 - val_accuracy: 0.9780
Epoch 27/30
313/313 [==============================] - 12s 39ms/step - loss: 0.0390 -
accuracy: 0.9948 - val_loss: 0.5515 - val_accuracy: 0.9780
Epoch 28/30
313/313 [==============================] - 12s 39ms/step - loss: 0.0298 -
accuracy: 0.9951 - val_loss: 0.5690 - val_accuracy: 0.9780
Epoch 29/30
313/313 [==============================] - 12s 39ms/step - loss: 0.0325 -
accuracy: 0.9960 - val_loss: 0.5253 - val_accuracy: 0.9770
Epoch 30/30
313/313 [==============================] - 12s 39ms/step - loss: 0.0219 -
accuracy: 0.9963 - val_loss: 0.5714 - val_accuracy: 0.9780
```

[16]:

```
acc = history.history["accuracy"]
val_acc = history.history["val_accuracy"]
loss = history.history["loss"]
val_loss = history.history["val_loss"]
epochs = range(1, len(acc) + 1)
plt.plot(epochs, acc, "bo", label="Training accuracy")
plt.plot(epochs, val_acc, "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()
```





[17]:

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

```
32/32 [==============================] - 1s 30ms/step - loss: 0.2811 - accuracy:
0.9820
Test accuracy: 0.982
```