In [1]:
```
from google.colab import files
files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Out[1]:
```
Saving kaggle.json to kaggle.json
{'kaggle.json': b'{"username":"meghanagitay","key":"cb44c5ff4c1c20e2caa85474f118115
b"}'}
```

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

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

```
Downloading dogs-vs-cats.zip to /content
100% 810M/812M [00:21<00:00, 42.9MB/s]
100% 812M/812M [00:21<00:00, 40.1MB/s]
```

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

Copying Images to Train, Test and Validation Folders

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

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

def make_subset(subset_name, start_index, end_index):
    for category in ("cat", "dog"):
        dir = new_base_dir / subset_name / category
        os.makedirs(dir)
        fnames = [f"{category}.{i}.jpg" for i in range(start_index, end_index)]
        for fname in fnames:
            shutil.copyfile(src=original_dir / fname,
                            dst=dir / fname)
make_subset("validation", start_index=0, end_index=500)
make_subset("test", start_index=500, end_index=1000)
make_subset("train", start_index=1000, end_index=2000)
```

Data Preprocessing

In [ ]:
```
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.
```

Model Building

In [ ]:

```
from tensorflow import keras
from tensorflow.keras import layers
from keras import regularizers
inputs = keras.Input(shape=(180, 180, 3))
x = layers.Rescaling(1./255)(inputs)
x = layers.Conv2D(filters=32,kernel_size=3,activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128,kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256,kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

model.summary()
```

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

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

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

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

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

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

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

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

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

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

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

 flatten (Flatten)           (None, 12544)             0

 dense (Dense)               (None, 1)                 12545

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

In [ ]:
```
from keras.optimizers import Adam
model.compile(loss="binary_crossentropy",
              optimizer="adam",
               metrics=["accuracy"])
```

In [ ]:
```
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,)
```

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

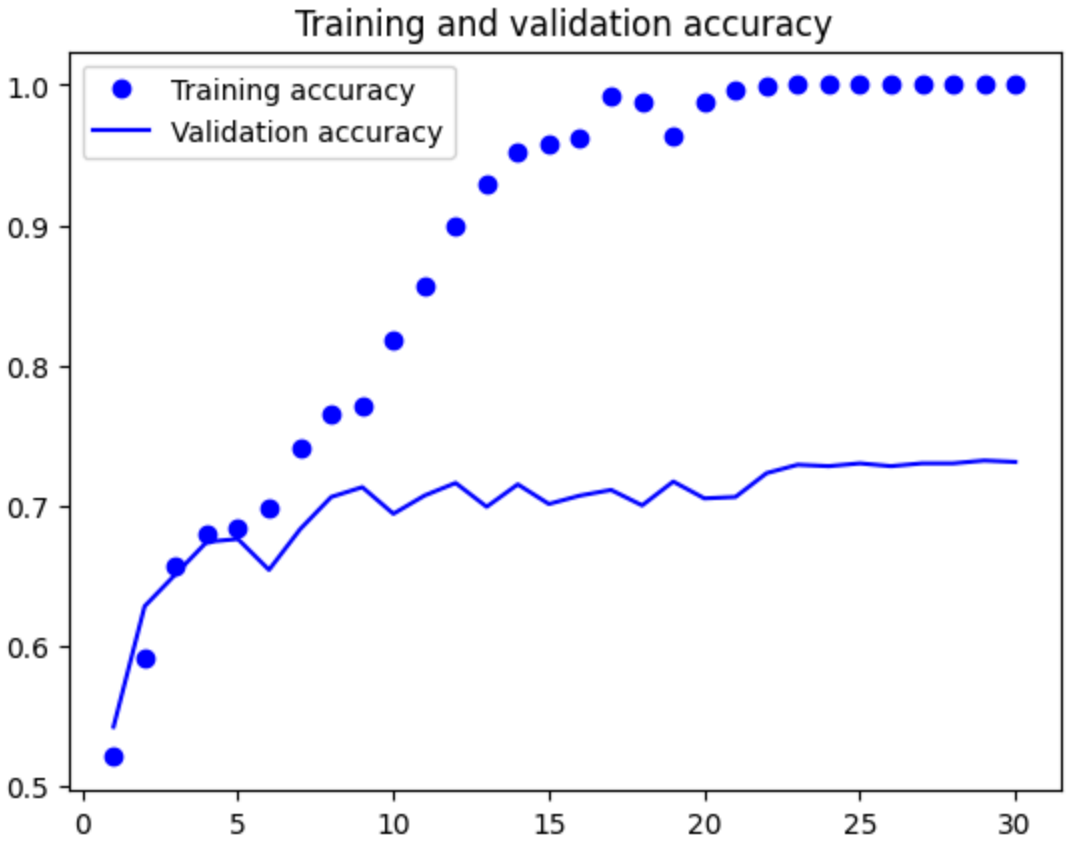
```
    train_dataset,
    epochs=20,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

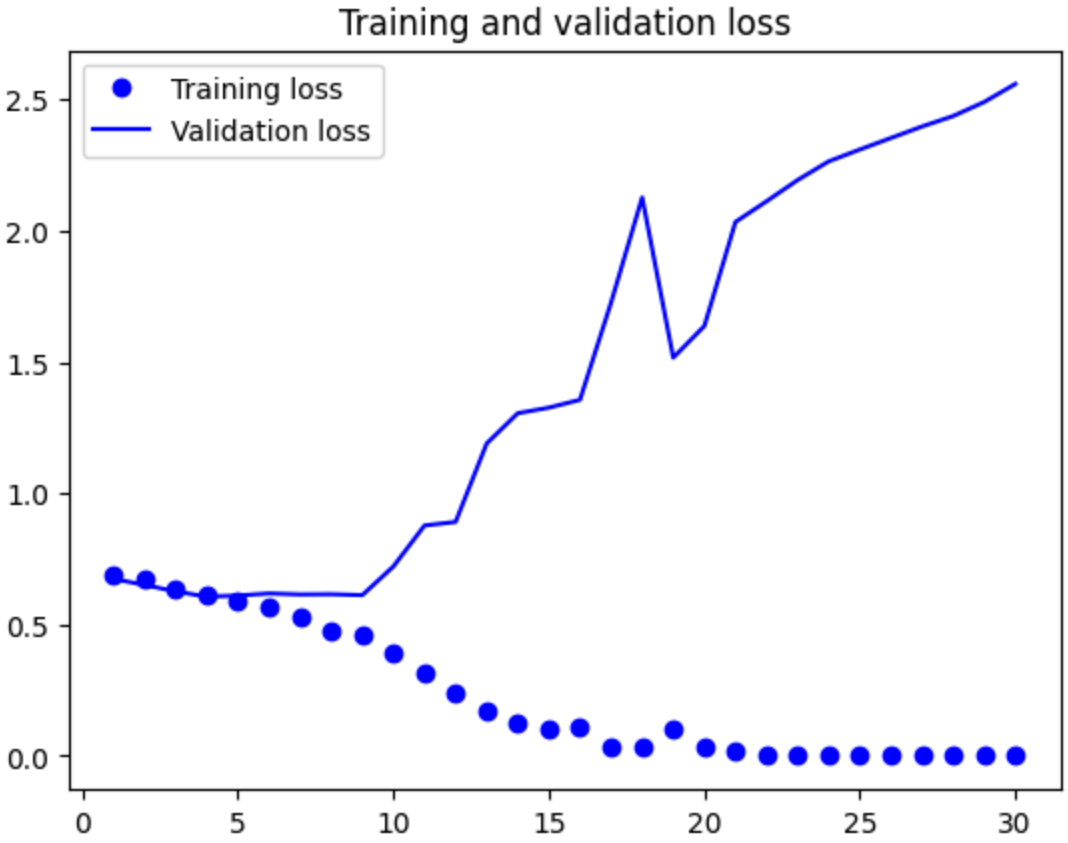
```
Epoch 1/30
63/63 [==============================] - 7s 34ms/step - loss: 0.6906 - accuracy: 0.52
05 - val_loss: 0.6742 - val_accuracy: 0.5420
Epoch 2/30
63/63 [==============================] - 2s 23ms/step - loss: 0.6704 - accuracy: 0.59
05 - val_loss: 0.6512 - val_accuracy: 0.6280
Epoch 3/30
63/63 [==============================] - 1s 23ms/step - loss: 0.6331 - accuracy: 0.65
65 - val_loss: 0.6264 - val_accuracy: 0.6510
Epoch 4/30
63/63 [==============================] - 2s 23ms/step - loss: 0.6097 - accuracy: 0.67
90 - val_loss: 0.6065 - val_accuracy: 0.6740
Epoch 5/30
63/63 [==============================] - 2s 23ms/step - loss: 0.5914 - accuracy: 0.68
45 - val_loss: 0.6108 - val_accuracy: 0.6760
Epoch 6/30
63/63 [==============================] - 1s 22ms/step - loss: 0.5667 - accuracy: 0.69
80 - val_loss: 0.6188 - val_accuracy: 0.6540
Epoch 7/30
63/63 [==============================] - 1s 22ms/step - loss: 0.5297 - accuracy: 0.74
15 - val_loss: 0.6151 - val_accuracy: 0.6830
Epoch 8/30
63/63 [==============================] - 1s 22ms/step - loss: 0.4780 - accuracy: 0.76
55 - val_loss: 0.6159 - val_accuracy: 0.7060
Epoch 9/30
63/63 [==============================] - 1s 21ms/step - loss: 0.4626 - accuracy: 0.77
15 - val_loss: 0.6124 - val_accuracy: 0.7130
Epoch 10/30
63/63 [==============================] - 1s 21ms/step - loss: 0.3951 - accuracy: 0.81
85 - val_loss: 0.7217 - val_accuracy: 0.6940
Epoch 11/30
63/63 [==============================] - 1s 21ms/step - loss: 0.3181 - accuracy: 0.85
60 - val_loss: 0.8775 - val_accuracy: 0.7070
Epoch 12/30
63/63 [==============================] - 1s 21ms/step - loss: 0.2431 - accuracy: 0.89
90 - val_loss: 0.8911 - val_accuracy: 0.7160
Epoch 13/30
63/63 [==============================] - 1s 21ms/step - loss: 0.1739 - accuracy: 0.92
95 - val_loss: 1.1898 - val_accuracy: 0.6990
Epoch 14/30
63/63 [==============================] - 1s 21ms/step - loss: 0.1225 - accuracy: 0.95
15 - val_loss: 1.3050 - val_accuracy: 0.7150
Epoch 15/30
63/63 [==============================] - 1s 21ms/step - loss: 0.1052 - accuracy: 0.95
85 - val_loss: 1.3263 - val_accuracy: 0.7010
Epoch 16/30
63/63 [==============================] - 1s 22ms/step - loss: 0.1075 - accuracy: 0.96
15 - val_loss: 1.3561 - val_accuracy: 0.7070
Epoch 17/30
63/63 [==============================] - 1s 22ms/step - loss: 0.0360 - accuracy: 0.99
20 - val_loss: 1.7265 - val_accuracy: 0.7110
Epoch 18/30
63/63 [==============================] - 1s 22ms/step - loss: 0.0325 - accuracy: 0.98
80 - val_loss: 2.1266 - val_accuracy: 0.7000
Epoch 19/30
63/63 [==============================] - 1s 22ms/step - loss: 0.0997 - accuracy: 0.96
40 - val_loss: 1.5165 - val_accuracy: 0.7170
Epoch 20/30
63/63 [==============================] - 1s 22ms/step - loss: 0.0373 - accuracy: 0.98
75 - val_loss: 1.6372 - val_accuracy: 0.7050
```

```
Epoch 21/30
63/63 [==============================] - 1s 21ms/step - loss: 0.0192 - accuracy: 0.99
60 - val_loss: 2.0332 - val_accuracy: 0.7060
Epoch 22/30
63/63 [==============================] - 1s 21ms/step - loss: 0.0055 - accuracy: 0.99
95 - val_loss: 2.1120 - val_accuracy: 0.7230
Epoch 23/30
63/63 [==============================] - 1s 21ms/step - loss: 0.0020 - accuracy: 1.00
00 - val_loss: 2.1930 - val_accuracy: 0.7290
Epoch 24/30
63/63 [==============================] - 1s 21ms/step - loss: 6.9427e-04 - accuracy:
1.0000 - val_loss: 2.2640 - val_accuracy: 0.7280
Epoch 25/30
63/63 [==============================] - 1s 21ms/step - loss: 4.0110e-04 - accuracy:
1.0000 - val_loss: 2.3091 - val_accuracy: 0.7300
Epoch 26/30
63/63 [==============================] - 1s 21ms/step - loss: 3.1333e-04 - accuracy:
1.0000 - val_loss: 2.3529 - val_accuracy: 0.7280
Epoch 27/30
63/63 [==============================] - 1s 21ms/step - loss: 2.5498e-04 - accuracy:
1.0000 - val_loss: 2.3967 - val_accuracy: 0.7300
Epoch 28/30
63/63 [==============================] - 1s 22ms/step - loss: 2.1239e-04 - accuracy:
1.0000 - val_loss: 2.4364 - val_accuracy: 0.7300
Epoch 29/30
63/63 [==============================] - 1s 22ms/step - loss: 1.7774e-04 - accuracy:
1.0000 - val_loss: 2.4901 - val_accuracy: 0.7320
Epoch 30/30
63/63 [==============================] - 1s 22ms/step - loss: 1.4106e-04 - accuracy:
1.0000 - val_loss: 2.5587 - val_accuracy: 0.7310
```

In [ ]:

```
import matplotlib.pyplot as plt
accuracy = history.history["accuracy"]
val_accuracy = history.history["val_accuracy"]
loss = history.history["loss"]
val_loss = history.history["val_loss"]
epochs = range(1, len(accuracy) + 1)
plt.plot(epochs, accuracy, "bo", label="Training accuracy")
plt.plot(epochs, val_accuracy, "b", label="Validation accuracy")
plt.title("Training and validation accuracy")
plt.legend()
plt.figure()
plt.plot(epochs, loss, "bo", label="Training loss")
plt.plot(epochs, val_loss, "b", label="Validation loss")
plt.title("Training and validation loss")
plt.legend()
plt.show()
```





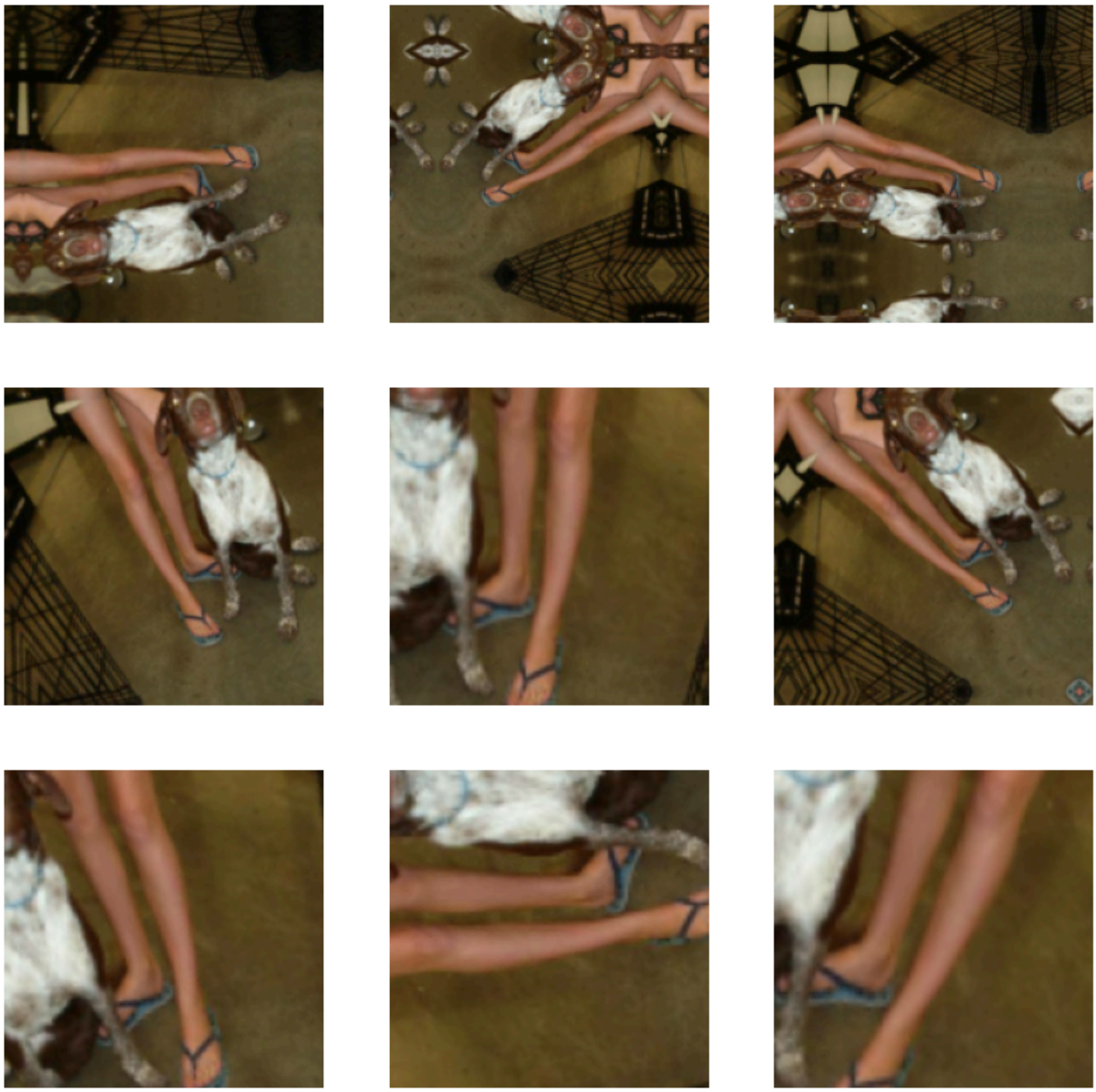
In [ ]:
```
test_model = keras.models.load_model("convnet_from_scratch.keras")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 1s 10ms/step - loss: 0.6243 - accuracy: 0.65
10
Test accuracy: 0.651
```

Adding Data Augmentation

In [ ]:
```python
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.3),
        layers.RandomZoom(0.6),
    ]
)
```

In [ ]:
```python
import warnings
warnings.filterwarnings("ignore")
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")
```



In [ ]:
```
inputs = keras.Input(shape=(180, 180, 3))
x = data_augmentation(inputs)
x = layers.Rescaling(1./255)(inputs)
x = layers.Conv2D(filters=32,kernel_size=3,activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128,kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256,kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
#x = layers.Conv2D(filters=512,strides=2, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
x = layers.Dense(512, activation='relu', kernel_regularizer=regularizers.l2(0.0001))(x
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

model.summary()
from keras.optimizers import Adam
```

```
model.compile(loss="binary_crossentropy",
              optimizer=Adam(learning_rate=0.0001),
              metrics=["accuracy"])
```

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 180, 180, 3)]     0

 rescaling_1 (Rescaling)     (None, 180, 180, 3)       0

 conv2d_5 (Conv2D)           (None, 178, 178, 32)      896

 max_pooling2d_4 (MaxPoolin  (None, 89, 89, 32)        0
 g2D)

 conv2d_6 (Conv2D)           (None, 87, 87, 64)        18496

 max_pooling2d_5 (MaxPoolin  (None, 43, 43, 64)        0
 g2D)

 conv2d_7 (Conv2D)           (None, 41, 41, 128)       73856

 max_pooling2d_6 (MaxPoolin  (None, 20, 20, 128)       0
 g2D)

 conv2d_8 (Conv2D)           (None, 18, 18, 256)       295168

 max_pooling2d_7 (MaxPoolin  (None, 9, 9, 256)         0
 g2D)

 conv2d_9 (Conv2D)           (None, 7, 7, 256)         590080

 flatten_1 (Flatten)         (None, 12544)             0

 dense_1 (Dense)             (None, 512)               6423040

 dropout (Dropout)           (None, 512)               0

 dense_2 (Dense)             (None, 1)                 513

=================================================================
Total params: 7402049 (28.24 MB)
Trainable params: 7402049 (28.24 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [ ]:
```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_with_augmentation.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 [==============================] - 5s 31ms/step - loss: 0.7761 - accuracy: 0.50
35 - val_loss: 0.7564 - val_accuracy: 0.5000
Epoch 2/30
63/63 [==============================] - 2s 29ms/step - loss: 0.7297 - accuracy: 0.58
20 - val_loss: 0.7165 - val_accuracy: 0.5720
Epoch 3/30
63/63 [==============================] - 2s 30ms/step - loss: 0.7017 - accuracy: 0.62
05 - val_loss: 0.6932 - val_accuracy: 0.6610
Epoch 4/30
63/63 [==============================] - 2s 29ms/step - loss: 0.6748 - accuracy: 0.64
05 - val_loss: 0.6517 - val_accuracy: 0.6600
Epoch 5/30
63/63 [==============================] - 2s 31ms/step - loss: 0.6407 - accuracy: 0.66
25 - val_loss: 0.6266 - val_accuracy: 0.6810
Epoch 6/30
63/63 [==============================] - 2s 29ms/step - loss: 0.6090 - accuracy: 0.70
30 - val_loss: 0.5905 - val_accuracy: 0.7150
Epoch 7/30
63/63 [==============================] - 1s 22ms/step - loss: 0.5872 - accuracy: 0.72
20 - val_loss: 0.5940 - val_accuracy: 0.7160
Epoch 8/30
63/63 [==============================] - 2s 29ms/step - loss: 0.5497 - accuracy: 0.73
90 - val_loss: 0.5710 - val_accuracy: 0.7130
Epoch 9/30
63/63 [==============================] - 1s 22ms/step - loss: 0.5278 - accuracy: 0.75
70 - val_loss: 0.5740 - val_accuracy: 0.7070
Epoch 10/30
63/63 [==============================] - 2s 29ms/step - loss: 0.4969 - accuracy: 0.77
35 - val_loss: 0.5583 - val_accuracy: 0.7270
Epoch 11/30
63/63 [==============================] - 2s 29ms/step - loss: 0.4733 - accuracy: 0.79
80 - val_loss: 0.5515 - val_accuracy: 0.7410
Epoch 12/30
63/63 [==============================] - 2s 23ms/step - loss: 0.4506 - accuracy: 0.80
50 - val_loss: 0.6010 - val_accuracy: 0.7390
Epoch 13/30
63/63 [==============================] - 2s 29ms/step - loss: 0.4281 - accuracy: 0.81
80 - val_loss: 0.5505 - val_accuracy: 0.7550
Epoch 14/30
63/63 [==============================] - 1s 23ms/step - loss: 0.3798 - accuracy: 0.84
65 - val_loss: 0.5605 - val_accuracy: 0.7460
Epoch 15/30
63/63 [==============================] - 1s 22ms/step - loss: 0.3649 - accuracy: 0.85
35 - val_loss: 0.5726 - val_accuracy: 0.7580
Epoch 16/30
63/63 [==============================] - 1s 22ms/step - loss: 0.3540 - accuracy: 0.85
40 - val_loss: 0.5677 - val_accuracy: 0.7630
Epoch 17/30
63/63 [==============================] - 1s 22ms/step - loss: 0.2915 - accuracy: 0.89
10 - val_loss: 0.5734 - val_accuracy: 0.7590
Epoch 18/30
63/63 [==============================] - 1s 22ms/step - loss: 0.2627 - accuracy: 0.90
70 - val_loss: 0.5719 - val_accuracy: 0.7740
Epoch 19/30
63/63 [==============================] - 1s 22ms/step - loss: 0.2228 - accuracy: 0.92
95 - val_loss: 0.5632 - val_accuracy: 0.7850
Epoch 20/30
63/63 [==============================] - 1s 22ms/step - loss: 0.2048 - accuracy: 0.93
90 - val_loss: 0.5720 - val_accuracy: 0.7730
```

```
Epoch 21/30
63/63 [==============================] - 1s 22ms/step - loss: 0.1636 - accuracy: 0.95
05 - val_loss: 0.6330 - val_accuracy: 0.7830
Epoch 22/30
63/63 [==============================] - 2s 23ms/step - loss: 0.1818 - accuracy: 0.94
30 - val_loss: 0.6799 - val_accuracy: 0.7770
Epoch 23/30
63/63 [==============================] - 2s 23ms/step - loss: 0.1473 - accuracy: 0.95
50 - val_loss: 0.7129 - val_accuracy: 0.7740
Epoch 24/30
63/63 [==============================] - 2s 23ms/step - loss: 0.1113 - accuracy: 0.97
65 - val_loss: 0.7209 - val_accuracy: 0.7780
Epoch 25/30
63/63 [==============================] - 1s 22ms/step - loss: 0.0941 - accuracy: 0.98
20 - val_loss: 0.9464 - val_accuracy: 0.7350
Epoch 26/30
63/63 [==============================] - 1s 22ms/step - loss: 0.1007 - accuracy: 0.97
90 - val_loss: 0.7943 - val_accuracy: 0.7700
Epoch 27/30
63/63 [==============================] - 1s 22ms/step - loss: 0.0723 - accuracy: 0.99
10 - val_loss: 0.8400 - val_accuracy: 0.7620
Epoch 28/30
63/63 [==============================] - 1s 22ms/step - loss: 0.0689 - accuracy: 0.99
20 - val_loss: 0.8382 - val_accuracy: 0.7700
Epoch 29/30
63/63 [==============================] - 1s 22ms/step - loss: 0.0574 - accuracy: 0.99
50 - val_loss: 0.8519 - val_accuracy: 0.7810
Epoch 30/30
63/63 [==============================] - 1s 22ms/step - loss: 0.0511 - accuracy: 0.99
70 - val_loss: 0.9228 - val_accuracy: 0.7710
```

In [ ]:
```
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 9ms/step - loss: 0.5681 - accuracy: 0.741
0
Test accuracy: 0.741
```

Choosing a Random train sample size

Train Sample 2000

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

original_dir = pathlib.Path("train")
new_base_dir = pathlib.Path("cats_vs_dogs_small_1")

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)
make_subset("validation", start_index=0, end_index=500)
```

```
make_subset("test", start_index=500, end_index=1000)
make_subset("train", start_index=1000, end_index=3000)
```

In [ ]:
```
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 4000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
```

In [ ]:
```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_with_augmentation_2000.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=30,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/30
125/125 [==============================] - 3s 23ms/step - loss: 0.4282 - accuracy: 0.
8455 - val_loss: 0.5073 - val_accuracy: 0.7910
Epoch 2/30
125/125 [==============================] - 3s 23ms/step - loss: 0.3188 - accuracy: 0.
8817 - val_loss: 0.5004 - val_accuracy: 0.7980
Epoch 3/30
125/125 [==============================] - 3s 19ms/step - loss: 0.2707 - accuracy: 0.
9115 - val_loss: 0.5062 - val_accuracy: 0.8030
Epoch 4/30
125/125 [==============================] - 2s 19ms/step - loss: 0.2262 - accuracy: 0.
9252 - val_loss: 0.5214 - val_accuracy: 0.7930
Epoch 5/30
125/125 [==============================] - 3s 20ms/step - loss: 0.1878 - accuracy: 0.
9452 - val_loss: 0.5477 - val_accuracy: 0.8050
Epoch 6/30
125/125 [==============================] - 3s 20ms/step - loss: 0.1445 - accuracy: 0.
9595 - val_loss: 0.5830 - val_accuracy: 0.8150
Epoch 7/30
125/125 [==============================] - 3s 20ms/step - loss: 0.1350 - accuracy: 0.
9615 - val_loss: 0.5807 - val_accuracy: 0.8040
Epoch 8/30
125/125 [==============================] - 3s 20ms/step - loss: 0.1042 - accuracy: 0.
9765 - val_loss: 0.6959 - val_accuracy: 0.7950
Epoch 9/30
125/125 [==============================] - 3s 20ms/step - loss: 0.0856 - accuracy: 0.
9858 - val_loss: 0.6770 - val_accuracy: 0.8170
Epoch 10/30
125/125 [==============================] - 3s 20ms/step - loss: 0.0755 - accuracy: 0.
9875 - val_loss: 0.7055 - val_accuracy: 0.7980
Epoch 11/30
125/125 [==============================] - 3s 20ms/step - loss: 0.0584 - accuracy: 0.
9948 - val_loss: 0.7215 - val_accuracy: 0.8170
Epoch 12/30
125/125 [==============================] - 3s 20ms/step - loss: 0.0500 - accuracy: 0.
9970 - val_loss: 0.7941 - val_accuracy: 0.8040
Epoch 13/30
125/125 [==============================] - 3s 20ms/step - loss: 0.0574 - accuracy: 0.
9942 - val_loss: 0.8104 - val_accuracy: 0.7900
Epoch 14/30
125/125 [==============================] - 3s 20ms/step - loss: 0.0727 - accuracy: 0.
9872 - val_loss: 0.8395 - val_accuracy: 0.8000
Epoch 15/30
125/125 [==============================] - 2s 19ms/step - loss: 0.0543 - accuracy: 0.
9937 - val_loss: 0.7983 - val_accuracy: 0.8040
Epoch 16/30
125/125 [==============================] - 3s 20ms/step - loss: 0.0414 - accuracy: 0.
9983 - val_loss: 0.8633 - val_accuracy: 0.8020
Epoch 17/30
125/125 [==============================] - 2s 19ms/step - loss: 0.0405 - accuracy: 0.
9983 - val_loss: 0.9108 - val_accuracy: 0.8080
Epoch 18/30
125/125 [==============================] - 3s 20ms/step - loss: 0.0579 - accuracy: 0.
9912 - val_loss: 0.8319 - val_accuracy: 0.7970
Epoch 19/30
125/125 [==============================] - 3s 20ms/step - loss: 0.0440 - accuracy: 0.
9977 - val_loss: 0.8391 - val_accuracy: 0.7990
Epoch 20/30
125/125 [==============================] - 3s 20ms/step - loss: 0.0445 - accuracy: 0.
9967 - val_loss: 0.9177 - val_accuracy: 0.8100
```

```
Epoch 21/30
125/125 [==============================] - 2s 19ms/step - loss: 0.0385 - accuracy: 0.
9975 - val_loss: 1.3359 - val_accuracy: 0.7650
Epoch 22/30
125/125 [==============================] - 3s 19ms/step - loss: 0.0657 - accuracy: 0.
9872 - val_loss: 0.8737 - val_accuracy: 0.7900
Epoch 23/30
125/125 [==============================] - 3s 20ms/step - loss: 0.0407 - accuracy: 0.
9965 - val_loss: 0.9912 - val_accuracy: 0.7900
Epoch 24/30
125/125 [==============================] - 3s 20ms/step - loss: 0.0362 - accuracy: 0.
9992 - val_loss: 0.9905 - val_accuracy: 0.8000
Epoch 25/30
125/125 [==============================] - 3s 20ms/step - loss: 0.0305 - accuracy: 1.
0000 - val_loss: 1.0272 - val_accuracy: 0.8030
Epoch 26/30
125/125 [==============================] - 3s 20ms/step - loss: 0.0333 - accuracy: 0.
9990 - val_loss: 0.9589 - val_accuracy: 0.8030
Epoch 27/30
125/125 [==============================] - 3s 20ms/step - loss: 0.0381 - accuracy: 0.
9975 - val_loss: 1.1412 - val_accuracy: 0.7770
Epoch 28/30
125/125 [==============================] - 3s 20ms/step - loss: 0.0493 - accuracy: 0.
9930 - val_loss: 1.0767 - val_accuracy: 0.7930
Epoch 29/30
125/125 [==============================] - 3s 19ms/step - loss: 0.0610 - accuracy: 0.
9880 - val_loss: 0.9004 - val_accuracy: 0.8050
Epoch 30/30
125/125 [==============================] - 3s 20ms/step - loss: 0.0362 - accuracy: 0.
9977 - val_loss: 0.9505 - val_accuracy: 0.7920
```

In [ ]:
```
test_model = keras.models.load_model(
    "convnet_from_scratch_with_augmentation_2000.keras")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 1s 12ms/step - loss: 0.5336 - accuracy: 0.78
10
Test accuracy: 0.781
```

Train Sample: 3000

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

original_dir = pathlib.Path("train")
new_base_dir = pathlib.Path("cats_vs_dogs_small_5")

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)
make_subset("validation", start_index=0, end_index=500)
make_subset("test", start_index=500, end_index=1000)
make_subset("train", start_index=2000, end_index=4000)
```

In [ ]:
```
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 4000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
```

In [ ]:
```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_with_augmentation_30000.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=20,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/20
125/125 [==============================] - 3s 24ms/step - loss: 0.4014 - accuracy: 0.
8595 - val_loss: 0.5851 - val_accuracy: 0.7700
Epoch 2/20
125/125 [==============================] - 3s 23ms/step - loss: 0.2295 - accuracy: 0.
9243 - val_loss: 0.5646 - val_accuracy: 0.8050
Epoch 3/20
125/125 [==============================] - 3s 20ms/step - loss: 0.1613 - accuracy: 0.
9532 - val_loss: 0.6713 - val_accuracy: 0.8070
Epoch 4/20
125/125 [==============================] - 3s 20ms/step - loss: 0.1125 - accuracy: 0.
9730 - val_loss: 0.7232 - val_accuracy: 0.8080
Epoch 5/20
125/125 [==============================] - 3s 20ms/step - loss: 0.0809 - accuracy: 0.
9847 - val_loss: 0.7073 - val_accuracy: 0.8110
Epoch 6/20
125/125 [==============================] - 3s 20ms/step - loss: 0.0596 - accuracy: 0.
9927 - val_loss: 0.8641 - val_accuracy: 0.8060
Epoch 7/20
125/125 [==============================] - 3s 19ms/step - loss: 0.0488 - accuracy: 0.
9960 - val_loss: 0.8803 - val_accuracy: 0.8060
Epoch 8/20
125/125 [==============================] - 3s 20ms/step - loss: 0.0506 - accuracy: 0.
9933 - val_loss: 0.8813 - val_accuracy: 0.8090
Epoch 9/20
125/125 [==============================] - 3s 20ms/step - loss: 0.0458 - accuracy: 0.
9965 - val_loss: 0.8989 - val_accuracy: 0.8060
Epoch 10/20
125/125 [==============================] - 3s 20ms/step - loss: 0.0413 - accuracy: 0.
9977 - val_loss: 0.9687 - val_accuracy: 0.8120
Epoch 11/20
125/125 [==============================] - 3s 20ms/step - loss: 0.0523 - accuracy: 0.
9927 - val_loss: 0.9135 - val_accuracy: 0.8040
Epoch 12/20
125/125 [==============================] - 3s 20ms/step - loss: 0.0414 - accuracy: 0.
9973 - val_loss: 0.8961 - val_accuracy: 0.8080
Epoch 13/20
125/125 [==============================] - 3s 19ms/step - loss: 0.0322 - accuracy: 0.
9995 - val_loss: 0.9677 - val_accuracy: 0.8130
Epoch 14/20
125/125 [==============================] - 3s 20ms/step - loss: 0.0292 - accuracy: 1.
0000 - val_loss: 0.9682 - val_accuracy: 0.8210
Epoch 15/20
125/125 [==============================] - 3s 20ms/step - loss: 0.0289 - accuracy: 0.
9998 - val_loss: 1.0560 - val_accuracy: 0.7950
Epoch 16/20
125/125 [==============================] - 3s 20ms/step - loss: 0.0292 - accuracy: 1.
0000 - val_loss: 1.0084 - val_accuracy: 0.8040
Epoch 17/20
125/125 [==============================] - 3s 20ms/step - loss: 0.0267 - accuracy: 1.
0000 - val_loss: 1.0253 - val_accuracy: 0.8090
Epoch 18/20
125/125 [==============================] - 3s 20ms/step - loss: 0.0257 - accuracy: 1.
0000 - val_loss: 1.0311 - val_accuracy: 0.8130
Epoch 19/20
125/125 [==============================] - 2s 19ms/step - loss: 0.0272 - accuracy: 0.
9992 - val_loss: 1.0240 - val_accuracy: 0.8000
Epoch 20/20
125/125 [==============================] - 3s 20ms/step - loss: 0.0254 - accuracy: 1.
0000 - val_loss: 1.0623 - val_accuracy: 0.8050
```

In [ ]:
```
test_model = keras.models.load_model(
    "convnet_from_scratch_with_augmentation_30000.keras")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 1s 9ms/step - loss: 0.5876 - accuracy: 0.788
0
Test accuracy: 0.788
```

Train Sample: 4000

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

original_dir = pathlib.Path("train")
new_base_dir = pathlib.Path("cats_vs_dogs_small_3")

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)
make_subset("validation", start_index=0, end_index=500)
make_subset("test", start_index=500, end_index=1000)
make_subset("train", start_index=1000, end_index=5000)
```

In [ ]:
```
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 8000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
```

In [ ]:
```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_with_augmentation_4000.keras",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=20,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/20
250/250 [==============================] - 5s 21ms/step - loss: 0.2585 - accuracy: 0.
9166 - val_loss: 0.4791 - val_accuracy: 0.8240
Epoch 2/20
250/250 [==============================] - 5s 18ms/step - loss: 0.1347 - accuracy: 0.
9644 - val_loss: 0.5674 - val_accuracy: 0.8210
Epoch 3/20
250/250 [==============================] - 5s 18ms/step - loss: 0.0995 - accuracy: 0.
9746 - val_loss: 0.5733 - val_accuracy: 0.8140
Epoch 4/20
250/250 [==============================] - 5s 19ms/step - loss: 0.0594 - accuracy: 0.
9909 - val_loss: 0.7475 - val_accuracy: 0.8150
Epoch 5/20
250/250 [==============================] - 5s 18ms/step - loss: 0.0514 - accuracy: 0.
9934 - val_loss: 0.7044 - val_accuracy: 0.8260
Epoch 6/20
250/250 [==============================] - 5s 18ms/step - loss: 0.0526 - accuracy: 0.
9930 - val_loss: 0.7537 - val_accuracy: 0.8200
Epoch 7/20
250/250 [==============================] - 5s 18ms/step - loss: 0.0362 - accuracy: 0.
9979 - val_loss: 0.8301 - val_accuracy: 0.8090
Epoch 8/20
250/250 [==============================] - 5s 19ms/step - loss: 0.0577 - accuracy: 0.
9893 - val_loss: 0.7506 - val_accuracy: 0.8290
Epoch 9/20
250/250 [==============================] - 5s 18ms/step - loss: 0.0381 - accuracy: 0.
9970 - val_loss: 0.8698 - val_accuracy: 0.8100
Epoch 10/20
250/250 [==============================] - 5s 18ms/step - loss: 0.0386 - accuracy: 0.
9967 - val_loss: 0.8518 - val_accuracy: 0.8190
Epoch 11/20
250/250 [==============================] - 5s 19ms/step - loss: 0.0290 - accuracy: 0.
9998 - val_loss: 0.9072 - val_accuracy: 0.8200
Epoch 12/20
250/250 [==============================] - 5s 18ms/step - loss: 0.0339 - accuracy: 0.
9971 - val_loss: 1.0367 - val_accuracy: 0.7930
Epoch 13/20
250/250 [==============================] - 5s 18ms/step - loss: 0.0351 - accuracy: 0.
9971 - val_loss: 0.9628 - val_accuracy: 0.8230
Epoch 14/20
250/250 [==============================] - 5s 19ms/step - loss: 0.0404 - accuracy: 0.
9948 - val_loss: 0.8865 - val_accuracy: 0.8010
Epoch 15/20
250/250 [==============================] - 5s 18ms/step - loss: 0.0431 - accuracy: 0.
9945 - val_loss: 0.8942 - val_accuracy: 0.8270
Epoch 16/20
250/250 [==============================] - 5s 18ms/step - loss: 0.0460 - accuracy: 0.
9933 - val_loss: 0.7595 - val_accuracy: 0.8390
Epoch 17/20
250/250 [==============================] - 5s 18ms/step - loss: 0.0303 - accuracy: 0.
9987 - val_loss: 0.9124 - val_accuracy: 0.8140
Epoch 18/20
250/250 [==============================] - 5s 19ms/step - loss: 0.0382 - accuracy: 0.
9960 - val_loss: 0.9227 - val_accuracy: 0.8180
Epoch 19/20
250/250 [==============================] - 5s 18ms/step - loss: 0.0416 - accuracy: 0.
9951 - val_loss: 0.9651 - val_accuracy: 0.8100
Epoch 20/20
250/250 [==============================] - 5s 18ms/step - loss: 0.0363 - accuracy: 0.
9969 - val_loss: 0.9699 - val_accuracy: 0.8250
```

In [ ]:
```
test_model = keras.models.load_model(
    "convnet_from_scratch_with_augmentation_4000.keras")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 1s 10ms/step - loss: 0.5068 - accuracy: 0.82
50
Test accuracy: 0.825
```

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

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vg
g16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [==============================] - 2s 0us/step
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (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: 14714688 (56.13 MB)
Trainable params: 14714688 (56.13 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [ ]:

```
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.3),
        layers.RandomZoom(0.6)
    ]
)

conv_base.trainable = True
```

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

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)
from keras.optimizers import Adam
model.compile(loss="binary_crossentropy",
              optimizer=Adam(learning_rate=0.0001),
               metrics=["accuracy"])
```

In [ ]:

```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="vgg16_with_data_augmentation.h5",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=20,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/20
250/250 [==============================] - 11s 43ms/step - loss: 0.2420 - accuracy:
0.9001 - val_loss: 0.0966 - val_accuracy: 0.9630
Epoch 2/20
250/250 [==============================] - 11s 43ms/step - loss: 0.2154 - accuracy:
0.9118 - val_loss: 0.0861 - val_accuracy: 0.9650
Epoch 3/20
250/250 [==============================] - 11s 43ms/step - loss: 0.1925 - accuracy:
0.9212 - val_loss: 0.0774 - val_accuracy: 0.9750
Epoch 4/20
250/250 [==============================] - 11s 43ms/step - loss: 0.2105 - accuracy:
0.9144 - val_loss: 0.0551 - val_accuracy: 0.9790
Epoch 5/20
250/250 [==============================] - 10s 41ms/step - loss: 0.1836 - accuracy:
0.9243 - val_loss: 0.0692 - val_accuracy: 0.9670
Epoch 6/20
250/250 [==============================] - 10s 41ms/step - loss: 0.1704 - accuracy:
0.9289 - val_loss: 0.1031 - val_accuracy: 0.9590
Epoch 7/20
250/250 [==============================] - 10s 41ms/step - loss: 0.1643 - accuracy:
0.9365 - val_loss: 0.0618 - val_accuracy: 0.9770
Epoch 8/20
250/250 [==============================] - 10s 41ms/step - loss: 0.1529 - accuracy:
0.9381 - val_loss: 0.0627 - val_accuracy: 0.9820
Epoch 9/20
250/250 [==============================] - 10s 41ms/step - loss: 0.1346 - accuracy:
0.9460 - val_loss: 0.0615 - val_accuracy: 0.9780
Epoch 10/20
250/250 [==============================] - 10s 41ms/step - loss: 0.1260 - accuracy:
0.9457 - val_loss: 0.0783 - val_accuracy: 0.9770
Epoch 11/20
250/250 [==============================] - 10s 41ms/step - loss: 0.1308 - accuracy:
0.9490 - val_loss: 0.0629 - val_accuracy: 0.9780
Epoch 12/20
250/250 [==============================] - 10s 42ms/step - loss: 0.1327 - accuracy:
0.9481 - val_loss: 0.1300 - val_accuracy: 0.9590
Epoch 13/20
250/250 [==============================] - 11s 43ms/step - loss: 0.1284 - accuracy:
0.9491 - val_loss: 0.0510 - val_accuracy: 0.9820
Epoch 14/20
250/250 [==============================] - 11s 43ms/step - loss: 0.1289 - accuracy:
0.9500 - val_loss: 0.0448 - val_accuracy: 0.9790
Epoch 15/20
250/250 [==============================] - 10s 41ms/step - loss: 0.1220 - accuracy:
0.9536 - val_loss: 0.0606 - val_accuracy: 0.9740
Epoch 16/20
250/250 [==============================] - 10s 42ms/step - loss: 0.1103 - accuracy:
0.9569 - val_loss: 0.0638 - val_accuracy: 0.9750
Epoch 17/20
250/250 [==============================] - 10s 41ms/step - loss: 0.1114 - accuracy:
0.9578 - val_loss: 0.0497 - val_accuracy: 0.9820
Epoch 18/20
250/250 [==============================] - 10s 42ms/step - loss: 0.1060 - accuracy:
0.9556 - val_loss: 0.0604 - val_accuracy: 0.9790
Epoch 19/20
250/250 [==============================] - 11s 42ms/step - loss: 0.1119 - accuracy:
0.9544 - val_loss: 0.0543 - val_accuracy: 0.9780
Epoch 20/20
250/250 [==============================] - 10s 41ms/step - loss: 0.1048 - accuracy:
0.9620 - val_loss: 0.0604 - val_accuracy: 0.9760
```

In [ ]:
```
test_model = keras.models.load_model(
    "vgg16_with_data_augmentation.h5")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 1s 31ms/step - loss: 0.0650 - accuracy: 0.98
30
Test accuracy: 0.983
```