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":"telugunikhitha","key":"f97e0d2d911ad423b70e76692207
cdb6"}'}
```

In [2]:
```
!mkdir ~/.kaggle/
```

In [3]:
```
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

In [4]:
```
!kaggle competitions download -c dogs-vs-cats
```

```
Downloading dogs-vs-cats.zip to /content
 99% 801M/812M [00:06<00:00, 120MB/s]
100% 812M/812M [00:06<00:00, 132MB/s]
```

In [5]:
```
!unzip -qq dogs-vs-cats.zip
```

In [6]:
```
!unzip -qq train.zip
```

In [7]:
```
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("train", start_index=0, end_index=1000)
make_subset("validation", start_index=1000, end_index=1500)
make_subset("test", start_index=1500, end_index=2000)
```

In [8]:
```
from tensorflow import keras
from tensorflow.keras import layers

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

In [9]:
```
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 [10]:
```
model.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

We use the regularization strategy in the DATA PREPROCESSING stage because we are concerned that the model might overfit. Each image is converted into a tensor in this instance.

In [11]:
```
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.
```

Callback can be used to store the model's weights at the conclusion of each epoch or to stop training the model early if it is not improving. Additionally, callbacks can be used to schedule learning rate adjustments, record metrics, and show the model's success.

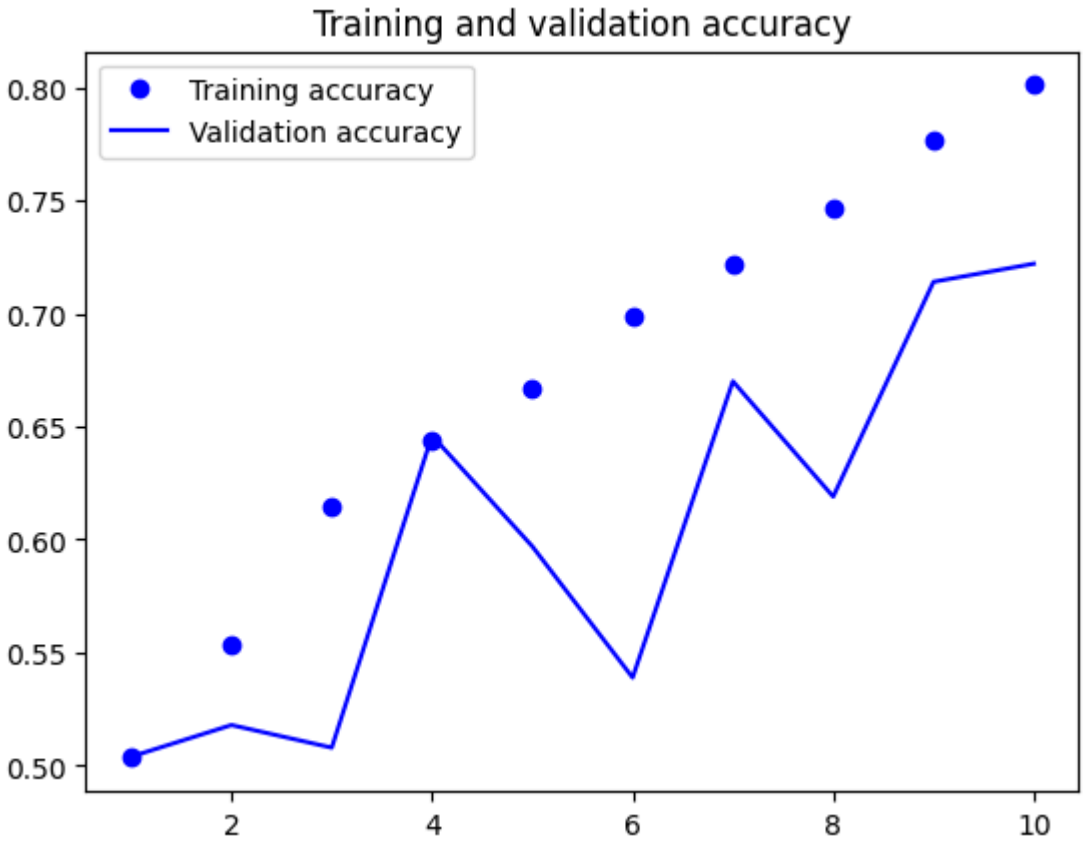
In [12]:
```
from tensorflow import keras
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch1.x",
        save_best_only=True,
        monitor="val_loss"
    )
]
history = model.fit(
    train_dataset,
    epochs=10,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

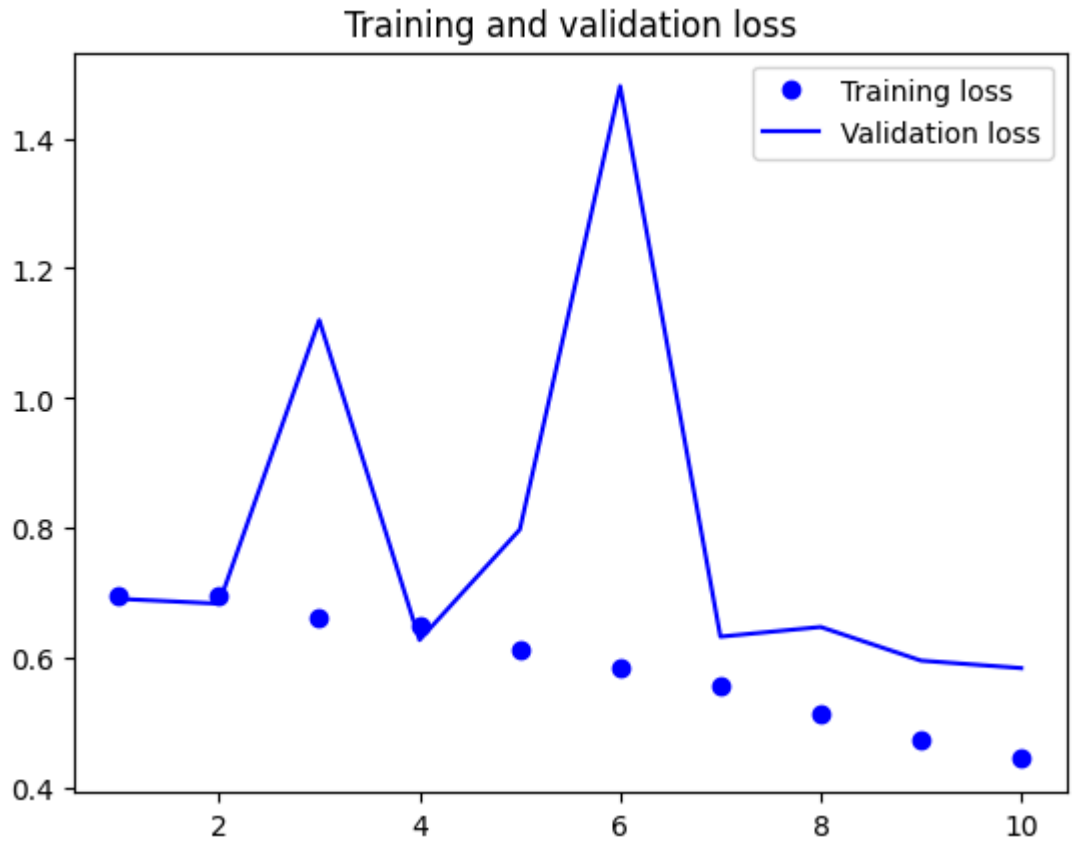
```
Epoch 1/10
63/63 [==============================] - 7s 48ms/step - loss: 0.6948 - accuracy:
0.5035 - val_loss: 0.6914 - val_accuracy: 0.5040
Epoch 2/10
63/63 [==============================] - 2s 38ms/step - loss: 0.6977 - accuracy:
0.5535 - val_loss: 0.6840 - val_accuracy: 0.5180
Epoch 3/10
63/63 [==============================] - 1s 21ms/step - loss: 0.6617 - accuracy:
0.6145 - val_loss: 1.1203 - val_accuracy: 0.5080
Epoch 4/10
63/63 [==============================] - 2s 37ms/step - loss: 0.6489 - accuracy:
0.6440 - val_loss: 0.6281 - val_accuracy: 0.6460
Epoch 5/10
63/63 [==============================] - 1s 22ms/step - loss: 0.6120 - accuracy:
0.6670 - val_loss: 0.7980 - val_accuracy: 0.5970
Epoch 6/10
63/63 [==============================] - 1s 21ms/step - loss: 0.5850 - accuracy:
0.6990 - val_loss: 1.4802 - val_accuracy: 0.5390
Epoch 7/10
63/63 [==============================] - 1s 21ms/step - loss: 0.5564 - accuracy:
0.7215 - val_loss: 0.6332 - val_accuracy: 0.6700
Epoch 8/10
63/63 [==============================] - 1s 21ms/step - loss: 0.5131 - accuracy:
0.7465 - val_loss: 0.6481 - val_accuracy: 0.6190
Epoch 9/10
63/63 [==============================] - 2s 34ms/step - loss: 0.4754 - accuracy:
0.7765 - val_loss: 0.5963 - val_accuracy: 0.7140
Epoch 10/10
63/63 [==============================] - 2s 35ms/step - loss: 0.4453 - accuracy:
0.8010 - val_loss: 0.5848 - val_accuracy: 0.7220
```

It has been observed that accuracy rises with the number of epochs. Accuracy = 72.20% Val_acc = 80.10%

In [13]:
```
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 [14]:
```
test_model = keras.models.load_model("convnet_from_scratch1.x")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 1s 9ms/step - loss: 0.6169 - accuracy: 0.
7090
Test accuracy: 0.709
```

Test accuracy with no data augmentation = 70.90%

Data Augmentation

A method called "data augmentation" makes new, altered versions of the original data in order to expand the size of a training set. This enhances the model's capacity for generalization and lessens overfitting.

In [15]:
```
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)
```

In [16]:
```
inputs = keras.Input(shape=(180, 180, 3))
x = data_augmentation(inputs)
x = layers.Rescaling(1./255)(x)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
```

```
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

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

In [17]:
```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_with_augmentation1.x",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=10,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/10
63/63 [==============================] - 6s 54ms/step - loss: 0.6949 - accuracy:
0.5105 - val_loss: 0.6982 - val_accuracy: 0.5000
Epoch 2/10
63/63 [==============================] - 3s 48ms/step - loss: 0.6903 - accuracy:
0.5500 - val_loss: 0.6758 - val_accuracy: 0.6130
Epoch 3/10
63/63 [==============================] - 2s 23ms/step - loss: 0.6772 - accuracy:
0.5850 - val_loss: 0.6942 - val_accuracy: 0.5050
Epoch 4/10
63/63 [==============================] - 3s 46ms/step - loss: 0.6675 - accuracy:
0.6005 - val_loss: 0.6582 - val_accuracy: 0.5820
Epoch 5/10
63/63 [==============================] - 1s 22ms/step - loss: 0.6502 - accuracy:
0.6365 - val_loss: 0.6969 - val_accuracy: 0.5730
Epoch 6/10
63/63 [==============================] - 3s 51ms/step - loss: 0.6492 - accuracy:
0.6405 - val_loss: 0.6332 - val_accuracy: 0.6240
Epoch 7/10
63/63 [==============================] - 3s 47ms/step - loss: 0.6211 - accuracy:
0.6515 - val_loss: 0.6020 - val_accuracy: 0.6680
Epoch 8/10
63/63 [==============================] - 2s 23ms/step - loss: 0.6068 - accuracy:
0.6700 - val_loss: 0.6287 - val_accuracy: 0.6560
Epoch 9/10
63/63 [==============================] - 2s 23ms/step - loss: 0.5909 - accuracy:
0.6845 - val_loss: 0.6023 - val_accuracy: 0.6690
Epoch 10/10
63/63 [==============================] - 3s 47ms/step - loss: 0.5820 - accuracy:
0.6980 - val_loss: 0.5807 - val_accuracy: 0.6780
```

Although doing data augmentation on the model did not result in improved results, it is still possible to verify this by trying data augmentation on a larger training sample.

accuracy=69.80 val_acc=67.80

In [18]:
```
test_model = keras.models.load_model(
    "convnet_from_scratch_with_augmentation1.x")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 0s 9ms/step - loss: 0.6186 - accuracy: 0.
6820
Test accuracy: 0.682
```

Test accuracy was not improved

2) Increase training sample size

Attempted to increase training sample size from 1000 to 1500.

In [19]:
```
import os, shutil, pathlib

original_dir = pathlib.Path("train")
new_base_dir = pathlib.Path("cats_vs_dogs_small_2")

def make_subset(subset_name, start_index, end_index):
    for category in ("cat", "dog"):
        dir = new_base_dir / subset_name / category
        os.makedirs(dir, exist_ok=True)
        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=1500)
make_subset("validation", start_index=1500, end_index=2000)
make_subset("test", start_index=2000, end_index=2500)
```

In [20]:
```
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 3000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
```

In [21]:
```
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.compile(loss="binary_crossentropy",
```

```
              optimizer="rmsprop",
              metrics=["accuracy"])
```

In [22]:
```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch2.x",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=10,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/10
94/94 [==============================] - 5s 40ms/step - loss: 0.7179 - accuracy:
0.5283 - val_loss: 0.6927 - val_accuracy: 0.5000
Epoch 2/10
94/94 [==============================] - 3s 28ms/step - loss: 0.6880 - accuracy:
0.5520 - val_loss: 0.6765 - val_accuracy: 0.6160
Epoch 3/10
94/94 [==============================] - 3s 28ms/step - loss: 0.6570 - accuracy:
0.6217 - val_loss: 0.6378 - val_accuracy: 0.6580
Epoch 4/10
94/94 [==============================] - 2s 19ms/step - loss: 0.6183 - accuracy:
0.6640 - val_loss: 0.7658 - val_accuracy: 0.6180
Epoch 5/10
94/94 [==============================] - 3s 28ms/step - loss: 0.5934 - accuracy:
0.6980 - val_loss: 0.6099 - val_accuracy: 0.6710
Epoch 6/10
94/94 [==============================] - 3s 29ms/step - loss: 0.5358 - accuracy:
0.7333 - val_loss: 0.5623 - val_accuracy: 0.7090
Epoch 7/10
94/94 [==============================] - 2s 20ms/step - loss: 0.4998 - accuracy:
0.7580 - val_loss: 0.5882 - val_accuracy: 0.7010
Epoch 8/10
94/94 [==============================] - 2s 19ms/step - loss: 0.4404 - accuracy:
0.7927 - val_loss: 0.5811 - val_accuracy: 0.7190
Epoch 9/10
94/94 [==============================] - 2s 19ms/step - loss: 0.4177 - accuracy:
0.8043 - val_loss: 0.6489 - val_accuracy: 0.7060
Epoch 10/10
94/94 [==============================] - 2s 19ms/step - loss: 0.3563 - accuracy:
0.8373 - val_loss: 0.8225 - val_accuracy: 0.6990
```

In [23]:
```
test_model = keras.models.load_model(
    "convnet_from_scratch2.x")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 1s 12ms/step - loss: 0.5265 - accuracy:
0.7340
Test accuracy: 0.734
```

Accuracy= 52.65% val_acc=73.4% test_acc = 73.40

Using data augmentation

In [24]:
```
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
```

```
        layers.RandomZoom(0.2),
    ]
)
```

In [25]:
```
inputs = keras.Input(shape=(180, 180, 3))
x = data_augmentation(inputs)
x = layers.Rescaling(1./255)(x)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

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

In [26]:
```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_with_augmentation2.x",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=10,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/10
94/94 [==============================] - 7s 40ms/step - loss: 0.6971 - accuracy:
0.5037 - val_loss: 0.6934 - val_accuracy: 0.5000
Epoch 2/10
94/94 [==============================] - 2s 22ms/step - loss: 0.6922 - accuracy:
0.5177 - val_loss: 0.7300 - val_accuracy: 0.5000
Epoch 3/10
94/94 [==============================] - 4s 39ms/step - loss: 0.6940 - accuracy:
0.5003 - val_loss: 0.6924 - val_accuracy: 0.5000
Epoch 4/10
94/94 [==============================] - 2s 21ms/step - loss: 0.6936 - accuracy:
0.5280 - val_loss: 0.6927 - val_accuracy: 0.5000
Epoch 5/10
94/94 [==============================] - 4s 38ms/step - loss: 0.6908 - accuracy:
0.5253 - val_loss: 0.6804 - val_accuracy: 0.5830
Epoch 6/10
94/94 [==============================] - 2s 21ms/step - loss: 0.6849 - accuracy:
0.5603 - val_loss: 0.6841 - val_accuracy: 0.5600
Epoch 7/10
94/94 [==============================] - 2s 21ms/step - loss: 0.6784 - accuracy:
0.5667 - val_loss: 0.6830 - val_accuracy: 0.5560
Epoch 8/10
94/94 [==============================] - 4s 43ms/step - loss: 0.6646 - accuracy:
0.5910 - val_loss: 0.6790 - val_accuracy: 0.5830
Epoch 9/10
94/94 [==============================] - 4s 39ms/step - loss: 0.6445 - accuracy:
0.6387 - val_loss: 0.6440 - val_accuracy: 0.6270
Epoch 10/10
94/94 [==============================] - 4s 37ms/step - loss: 0.6398 - accuracy:
0.6357 - val_loss: 0.6424 - val_accuracy: 0.6400
```

In [27]:

```
test_model = keras.models.load_model(
    "convnet_from_scratch_with_augmentation2.x")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 0s 9ms/step - loss: 0.6292 - accuracy: 0.
6520
Test accuracy: 0.652
```

Accuracy=62.92% val_acc=65.20% test_acc=65.20%

3. Finding the ideal training sample size

We set the training, validation, and test set sizes, respectively, to 1500, 1000, and 500.

In [28]:
```
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, exist_ok=True)
        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=1500)
make_subset("validation", start_index=1500, end_index=2500)
make_subset("test", start_index=2500, end_index=3000)
```

In [29]:
```
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 3000 files belonging to 2 classes.
Found 2000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
```

In [30]:
```
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.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

In [31]:
```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch3.x",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
```

```
    train_dataset,
    epochs=10,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/10
94/94 [==============================] - 5s 35ms/step - loss: 0.6986 - accuracy:
0.5307 - val_loss: 0.6893 - val_accuracy: 0.5530
Epoch 2/10
94/94 [==============================] - 2s 23ms/step - loss: 0.6937 - accuracy:
0.5703 - val_loss: 0.7016 - val_accuracy: 0.5295
Epoch 3/10
94/94 [==============================] - 3s 32ms/step - loss: 0.6558 - accuracy:
0.6197 - val_loss: 0.6321 - val_accuracy: 0.6460
Epoch 4/10
94/94 [==============================] - 3s 31ms/step - loss: 0.6005 - accuracy:
0.6737 - val_loss: 0.6207 - val_accuracy: 0.6335
Epoch 5/10
94/94 [==============================] - 3s 31ms/step - loss: 0.5617 - accuracy:
0.7017 - val_loss: 0.5346 - val_accuracy: 0.7445
Epoch 6/10
94/94 [==============================] - 2s 22ms/step - loss: 0.5130 - accuracy:
0.7467 - val_loss: 0.6552 - val_accuracy: 0.6575
Epoch 7/10
94/94 [==============================] - 2s 23ms/step - loss: 0.4910 - accuracy:
0.7710 - val_loss: 0.5562 - val_accuracy: 0.7290
Epoch 8/10
94/94 [==============================] - 3s 33ms/step - loss: 0.4420 - accuracy:
0.7930 - val_loss: 0.5205 - val_accuracy: 0.7265
Epoch 9/10
94/94 [==============================] - 2s 22ms/step - loss: 0.3883 - accuracy:
0.8230 - val_loss: 0.6120 - val_accuracy: 0.7395
Epoch 10/10
94/94 [==============================] - 2s 22ms/step - loss: 0.3534 - accuracy:
0.8423 - val_loss: 0.6198 - val_accuracy: 0.6895
```

In [32]:
```
test_model = keras.models.load_model(
    "convnet_from_scratch3.x")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 0s 9ms/step - loss: 0.5311 - accuracy: 0.
7340
Test accuracy: 0.734
```

Accuracy= 53.11% val_Acc= 73.40% test_Acc= 73.40%

Using Data augmentation

In [33]:
```
data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.2),
    ]
)
```

In [34]:
```
inputs = keras.Input(shape=(180, 180, 3))
x = data_augmentation(inputs)
x = layers.Rescaling(1./255)(x)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
```

```
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)

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

In [35]:

```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="convnet_from_scratch_with_augmentation3.x",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=10,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/10
94/94 [==============================] - 6s 43ms/step - loss: 0.6961 - accuracy:
0.5030 - val_loss: 0.6904 - val_accuracy: 0.6195
Epoch 2/10
94/94 [==============================] - 4s 42ms/step - loss: 0.6922 - accuracy:
0.5280 - val_loss: 0.6883 - val_accuracy: 0.5000
Epoch 3/10
94/94 [==============================] - 4s 43ms/step - loss: 0.6900 - accuracy:
0.5217 - val_loss: 0.6857 - val_accuracy: 0.5385
Epoch 4/10
94/94 [==============================] - 4s 44ms/step - loss: 0.6845 - accuracy:
0.5363 - val_loss: 0.6793 - val_accuracy: 0.6335
Epoch 5/10
94/94 [==============================] - 4s 41ms/step - loss: 0.6751 - accuracy:
0.5740 - val_loss: 0.6471 - val_accuracy: 0.6130
Epoch 6/10
94/94 [==============================] - 2s 24ms/step - loss: 0.6641 - accuracy:
0.5930 - val_loss: 0.6846 - val_accuracy: 0.5155
Epoch 7/10
94/94 [==============================] - 2s 25ms/step - loss: 0.6726 - accuracy:
0.5787 - val_loss: 0.6718 - val_accuracy: 0.5760
Epoch 8/10
94/94 [==============================] - 4s 41ms/step - loss: 0.6609 - accuracy:
0.6127 - val_loss: 0.6417 - val_accuracy: 0.6490
Epoch 9/10
94/94 [==============================] - 4s 45ms/step - loss: 0.6389 - accuracy:
0.6310 - val_loss: 0.6406 - val_accuracy: 0.6545
Epoch 10/10
94/94 [==============================] - 4s 42ms/step - loss: 0.6206 - accuracy:
0.6607 - val_loss: 0.6147 - val_accuracy: 0.6730
```

In [36]:

```
test_model = keras.models.load_model(
    "convnet_from_scratch_with_augmentation3.x")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 1s 9ms/step - loss: 0.6329 - accuracy: 0.
6480
Test accuracy: 0.648
```

Accuracy= 64.80% val_acc=63.9% test_acc= 64.80%

4.Using a pre-trained network

VGG16 is the architecture of this pre-trained network.

Feature extraction - Instantiating the VGG16 convolutional base

In [37]:
```
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-application
s/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [==============================] - 0s 0us/step
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_7 (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)
_________________________________________________________________
```

Feature extraction - Extracting features and corresponding labels

In [38]:

```
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)

train_features.shape
```

```
1/1 [==============================] - 2s 2s/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
```

```
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 24ms/step
```

```
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 1s 863ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 1s 585ms/step
```

Out[38]: (3000, 5, 5, 512)

Feature extraction - Defining and training the densely connected classifier

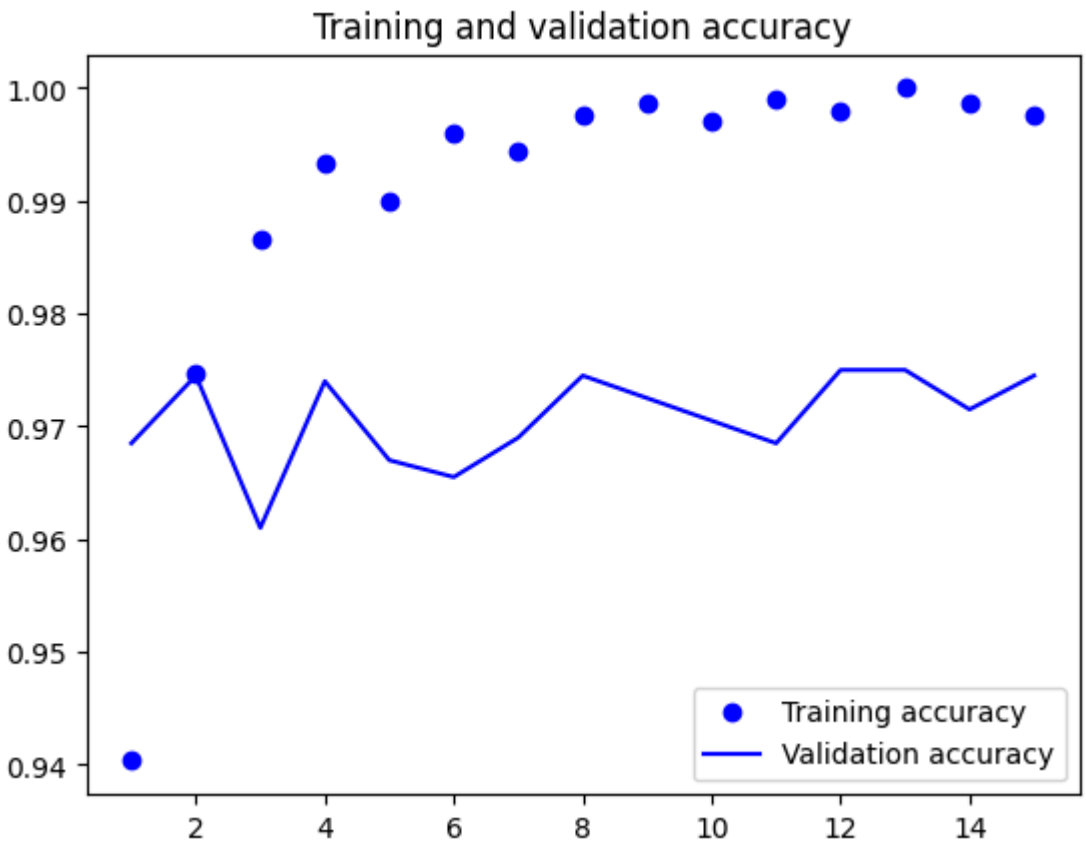
In [39]:
```
inputs = keras.Input(shape=(5, 5, 512))
x = layers.Flatten()(inputs)
x = layers.Dense(256)(x)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs, outputs)
model.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])

callbacks = [
    keras.callbacks.ModelCheckpoint(
      filepath="feature_extractionPT1.x",
      save_best_only=True,
      monitor="val_loss")
]
history = model.fit(
    train_features, train_labels,
    epochs=15,
    validation_data=(val_features, val_labels),
    callbacks=callbacks)
```

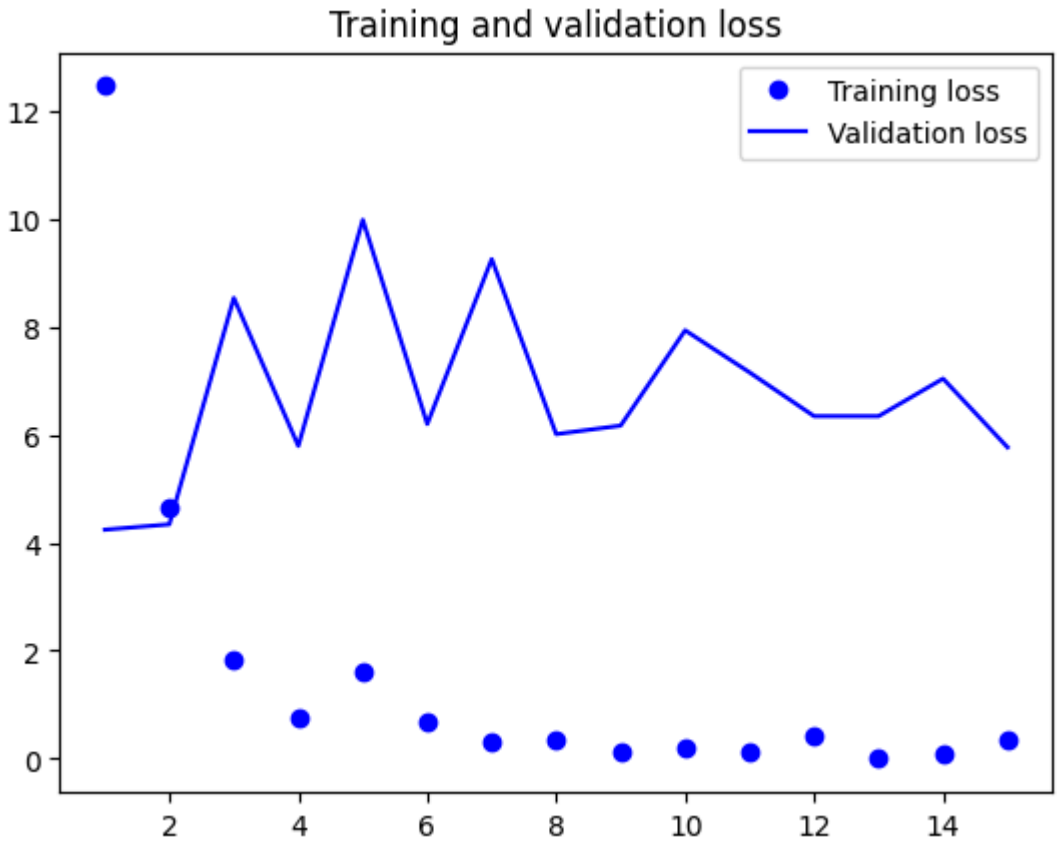
```
Epoch 1/15
94/94 [==============================] - 2s 13ms/step - loss: 12.4482 - accuracy:
0.9403 - val_loss: 4.2397 - val_accuracy: 0.9685
Epoch 2/15
94/94 [==============================] - 0s 5ms/step - loss: 4.6501 - accuracy: 0.
9747 - val_loss: 4.3389 - val_accuracy: 0.9745
Epoch 3/15
94/94 [==============================] - 0s 5ms/step - loss: 1.8296 - accuracy: 0.
9867 - val_loss: 8.5284 - val_accuracy: 0.9610
Epoch 4/15
94/94 [==============================] - 0s 5ms/step - loss: 0.7543 - accuracy: 0.
9933 - val_loss: 5.7894 - val_accuracy: 0.9740
Epoch 5/15
94/94 [==============================] - 0s 5ms/step - loss: 1.6027 - accuracy: 0.
9900 - val_loss: 9.9731 - val_accuracy: 0.9670
Epoch 6/15
94/94 [==============================] - 0s 5ms/step - loss: 0.6772 - accuracy: 0.
9960 - val_loss: 6.1955 - val_accuracy: 0.9655
Epoch 7/15
94/94 [==============================] - 0s 5ms/step - loss: 0.3023 - accuracy: 0.
9943 - val_loss: 9.2452 - val_accuracy: 0.9690
Epoch 8/15
94/94 [==============================] - 0s 5ms/step - loss: 0.3689 - accuracy: 0.
9977 - val_loss: 6.0078 - val_accuracy: 0.9745
Epoch 9/15
94/94 [==============================] - 0s 5ms/step - loss: 0.1236 - accuracy: 0.
9987 - val_loss: 6.1654 - val_accuracy: 0.9725
Epoch 10/15
94/94 [==============================] - 0s 5ms/step - loss: 0.2136 - accuracy: 0.
9970 - val_loss: 7.9288 - val_accuracy: 0.9705
Epoch 11/15
94/94 [==============================] - 0s 5ms/step - loss: 0.1140 - accuracy: 0.
9990 - val_loss: 7.1493 - val_accuracy: 0.9685
Epoch 12/15
94/94 [==============================] - 0s 5ms/step - loss: 0.4274 - accuracy: 0.
9980 - val_loss: 6.3401 - val_accuracy: 0.9750
Epoch 13/15
94/94 [==============================] - 0s 5ms/step - loss: 9.6420e-20 - accurac
y: 1.0000 - val_loss: 6.3401 - val_accuracy: 0.9750
Epoch 14/15
94/94 [==============================] - 0s 5ms/step - loss: 0.0987 - accuracy: 0.
9987 - val_loss: 7.0345 - val_accuracy: 0.9715
Epoch 15/15
94/94 [==============================] - 0s 5ms/step - loss: 0.3620 - accuracy: 0.
9977 - val_loss: 5.7638 - val_accuracy: 0.9745
```

accuracy=99.8% val_acc=97.45%

In [40]:

```
import matplotlib.pyplot as plt
acc = history.history["accuracy"]
val_acc = history.history["val_accuracy"]
loss = history.history["loss"]
val_loss = history.history["val_loss"]
epochs = range(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()
```





In [41]:
```
conv_base  = keras.applications.vgg16.VGG16(
    weights="imagenet",
    include_top=False)
conv_base.trainable = False


conv_base.trainable = True
print("This is the number of trainable weights "
```

```
      "before freezing the conv base:", len(conv_base.trainable_weights))


conv_base.trainable = False
print("This is the number of trainable weights "
      "after freezing the conv base:", len(conv_base.trainable_weights))
```

```
This is the number of trainable weights before freezing the conv base: 26
This is the number of trainable weights after freezing the conv base: 0
```

Feature extraction with Data Augmentation

In [42]:
```
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"])
```

In [43]:
```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="feature_extraction_with_data_augmentationPT2.x",
        save_best_only=True,
        monitor="val_loss")
]
history = model.fit(
    train_dataset,
    epochs=5,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/5
94/94 [==============================] - 9s 87ms/step - loss: 15.3378 - accuracy:
0.9080 - val_loss: 3.9828 - val_accuracy: 0.9700
Epoch 2/5
94/94 [==============================] - 5s 55ms/step - loss: 6.2680 - accuracy:
0.9483 - val_loss: 6.1716 - val_accuracy: 0.9655
Epoch 3/5
94/94 [==============================] - 8s 81ms/step - loss: 5.8436 - accuracy:
0.9513 - val_loss: 2.5762 - val_accuracy: 0.9795
Epoch 4/5
94/94 [==============================] - 5s 54ms/step - loss: 4.6681 - accuracy:
0.9653 - val_loss: 3.9659 - val_accuracy: 0.9735
Epoch 5/5
94/94 [==============================] - 5s 55ms/step - loss: 4.2427 - accuracy:
0.9663 - val_loss: 3.8034 - val_accuracy: 0.9755
```

In [44]:
```
test_model = keras.models.load_model(
    "feature_extraction_with_data_augmentationPT2.x")
```

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

```
32/32 [==============================] - 1s 31ms/step - loss: 6.6940 - accuracy:
0.9680
Test accuracy: 0.968
```

Accuracy=66.94% val_Acc=96.80% test_acc=96.80%