Step 1

```
from google.colab import drive
drive.mount('/content/gdrive')
!unzip -qq /content/gdrive/MyDrive/kagglecatsanddogs_5340.zip
```

```
Mounted at /content/gdrive
```

```
import os, shutil, pathlib

original_dir = pathlib.Path("PetImages")
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"{i}.jpg" for i in range(start_index, end_index)]
       for fname in fnames:
           shutil.copyfile(src=original_dir/category / fname,
                           dst=dir / fname)
```

```
make_subset("train", start_index=667, end_index=1667)
make_subset("validation", start_index=1668, end_index=2168)
make_subset("test", start_index=2169, end_index=2669)
```

```
import tensorflow as tf
from tensorflow.keras import layers, models, optimizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Building the model

```
img_size = (150, 150)
batch_size = 64
```

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
validation_datagen = ImageDataGenerator(rescale=1./255)
```

```
# Load and augment data
train_generator = train_datagen.flow_from_directory(
    new_base_dir / "train",
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary'
)
validation_generator = validation_datagen.flow_from_directory(
    new_base_dir / "validation",
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary'
)
```

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

```
test_generator = validation_datagen.flow_from_directory(
    new_base_dir / "test",
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary'
)
```

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

```
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Flatten(),
    layers.Dense(512, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])
```

```
model.compile(loss='binary_crossentropy',
              optimizer=optimizers.RMSprop(learning_rate=1e-4),
              metrics=['accuracy'])
```

Train and fit the data to the model

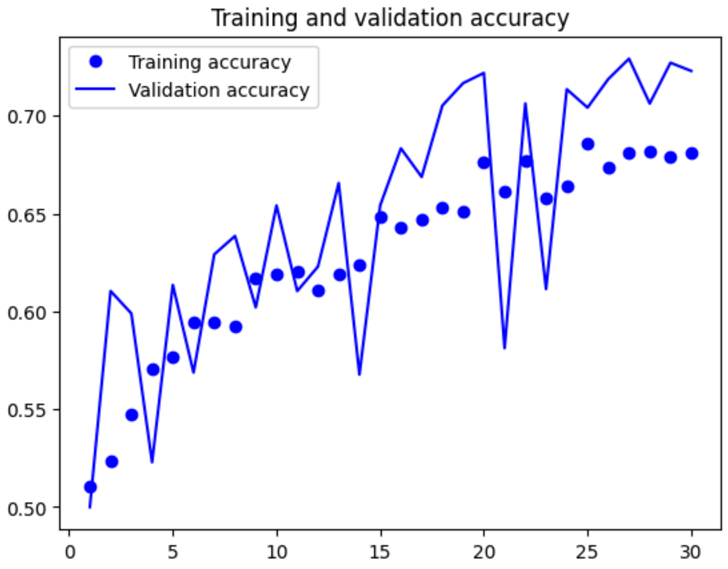
```
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    epochs=30,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // batch_size
)
```

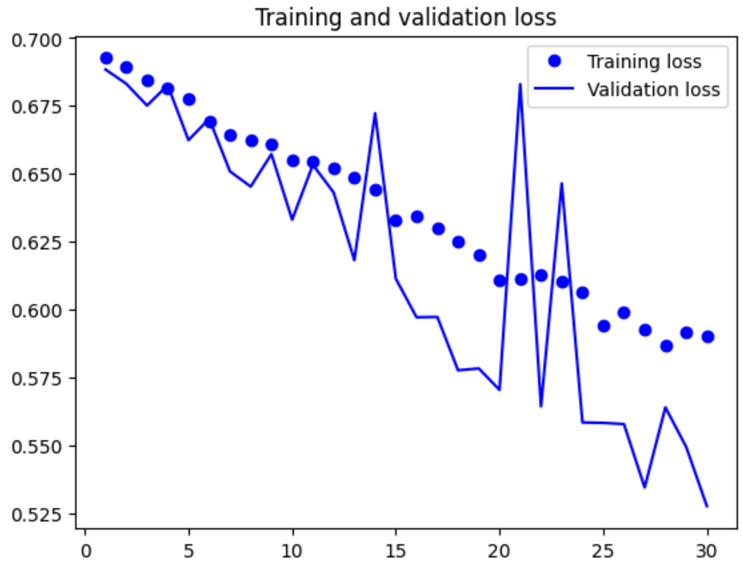
```
31/31 [==============================] - 15s 408ms/step - loss: 0.6926 - accuracy: 0.5108 - val_loss: 0.6884 - val_accuracy: 0.5000
Epoch 2/30
31/31 [==============================] - 14s 439ms/step - loss: 0.6893 - accuracy: 0.5237 - val_loss: 0.6831 - val_accuracy: 0.6104
Epoch 3/30
31/31 [==============================] - 13s 424ms/step - loss: 0.6846 - accuracy: 0.5475 - val_loss: 0.6751 - val_accuracy: 0.5990
Epoch 4/30
31/31 [==============================] - 14s 437ms/step - loss: 0.6813 - accuracy: 0.5708 - val_loss: 0.6826 - val_accuracy: 0.5229
Epoch 5/30
31/31 [==============================] - 13s 421ms/step - loss: 0.6778 - accuracy: 0.5764 - val_loss: 0.6624 - val_accuracy: 0.6135
```

```
Epoch 7/30
31/31 [==============================] - 14s 447ms/step - loss: 0.6641 - accuracy: 0.5945 - val_loss: 0.6509 - val_accuracy: 0.6292
Epoch 8/30
31/31 [==============================] - 14s 441ms/step - loss: 0.6623 - accuracy: 0.5922 - val_loss: 0.6453 - val_accuracy: 0.6385
Epoch 9/30
31/31 [==============================] - 13s 435ms/step - loss: 0.6609 - accuracy: 0.6173 - val_loss: 0.6572 - val_accuracy: 0.6021
Epoch 10/30
31/31 [==============================] - 13s 431ms/step - loss: 0.6551 - accuracy: 0.6188 - val_loss: 0.6331 - val_accuracy: 0.6542
Epoch 11/30
31/31 [==============================] - 13s 422ms/step - loss: 0.6544 - accuracy: 0.6204 - val_loss: 0.6533 - val_accuracy: 0.6104
Epoch 12/30
31/31 [==============================] - 13s 420ms/step - loss: 0.6521 - accuracy: 0.6111 - val_loss: 0.6431 - val_accuracy: 0.6229
Epoch 13/30
31/31 [==============================] - 13s 425ms/step - loss: 0.6485 - accuracy: 0.6193 - val_loss: 0.6182 - val_accuracy: 0.6656
Epoch 14/30
31/31 [==============================] - 14s 437ms/step - loss: 0.6440 - accuracy: 0.6240 - val_loss: 0.6723 - val_accuracy: 0.5677
Epoch 15/30
31/31 [==============================] - 13s 420ms/step - loss: 0.6329 - accuracy: 0.6482 - val_loss: 0.6113 - val_accuracy: 0.6542
Epoch 16/30
31/31 [==============================] - 13s 423ms/step - loss: 0.6342 - accuracy: 0.6431 - val_loss: 0.5971 - val_accuracy: 0.6833
Epoch 17/30
31/31 [==============================] - 13s 423ms/step - loss: 0.6299 - accuracy: 0.6467 - val_loss: 0.5973 - val_accuracy: 0.6687
Epoch 18/30
31/31 [==============================] - 13s 435ms/step - loss: 0.6248 - accuracy: 0.6529 - val_loss: 0.5776 - val_accuracy: 0.7052
Epoch 19/30
31/31 [==============================] - 13s 432ms/step - loss: 0.6202 - accuracy: 0.6513 - val_loss: 0.5783 - val_accuracy: 0.7167
Epoch 20/30
31/31 [==============================] - 13s 435ms/step - loss: 0.6106 - accuracy: 0.6761 - val_loss: 0.5704 - val_accuracy: 0.7219
Epoch 21/30
31/31 [==============================] - 14s 436ms/step - loss: 0.6113 - accuracy: 0.6612 - val_loss: 0.6830 - val_accuracy: 0.5813
Epoch 22/30
31/31 [==============================] - 13s 431ms/step - loss: 0.6129 - accuracy: 0.6767 - val_loss: 0.5643 - val_accuracy: 0.7063
Epoch 23/30
31/31 [==============================] - 13s 418ms/step - loss: 0.6104 - accuracy: 0.6575 - val_loss: 0.6464 - val_accuracy: 0.6115
Epoch 24/30
31/31 [==============================] - 13s 434ms/step - loss: 0.6064 - accuracy: 0.6643 - val_loss: 0.5584 - val_accuracy: 0.7135
Epoch 25/30
31/31 [==============================] - 13s 429ms/step - loss: 0.5943 - accuracy: 0.6860 - val_loss: 0.5582 - val_accuracy: 0.7042
Epoch 26/30
31/31 [==============================] - 13s 432ms/step - loss: 0.5991 - accuracy: 0.6736 - val_loss: 0.5578 - val_accuracy: 0.7188
Epoch 27/30
31/31 [==============================] - 13s 426ms/step - loss: 0.5926 - accuracy: 0.6808 - val_loss: 0.5344 - val_accuracy: 0.7292
Epoch 28/30
31/31 [==============================] - 13s 425ms/step - loss: 0.5867 - accuracy: 0.6818 - val_loss: 0.5639 - val_accuracy: 0.7063
Epoch 29/30
31/31 [==============================] - 13s 421ms/step - loss: 0.5917 - accuracy: 0.6787 - val_loss: 0.5494 - val_accuracy: 0.7271
Epoch 30/30
31/31 [==============================] - 13s 424ms/step - loss: 0.5901 - accuracy: 0.6808 - val_loss: 0.5276 - val_accuracy: 0.7229
```

Displaying curves of loss and accuracy during training

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





Test Model

```
# Evaluate the model on test data
test_datagen = ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_directory(
    new_base_dir / "test",
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary'
)
test_loss, test_acc = model.evaluate(test_generator)
print('Test accuracy:', test_acc)
```

```
Found 1000 images belonging to 2 classes.
16/16 [==============================] - 2s 111ms/step - loss: 0.5479 - accuracy: 0.7090
Test accuracy: 0.7089999914169312
```

Step 2

```
import os, shutil, pathlib

original_dir = pathlib.Path("PetImages")
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
       os.makedirs(dir)
       fnames = [f"{i}.jpg" for i in range(start_index, end_index)]
       for fname in fnames:
           shutil.copyfile(src=original_dir/category / fname,
                           dst=dir / fname)
```

```
make_subset("train", start_index=667, end_index=2667)
make_subset("validation", start_index=2668, end_index=3168)
make_subset("test", start_index=3169, end_index=3669)
```

```
import tensorflow as tf
from tensorflow.keras import layers, models, optimizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Building the model

```
img_size = (150, 150)
batch_size = 64
```

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
validation_datagen = ImageDataGenerator(rescale=1./255)
```

```
# Load and augment data
train_generator = train_datagen.flow_from_directory(
    new_base_dir / "train",
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary'
)
validation_generator = validation_datagen.flow_from_directory(
    new_base_dir / "validation",
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary'
)
```

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

```
test_generator = validation_datagen.flow_from_directory(
    new_base_dir / "test",
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary'
)
```

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

```
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Flatten(),
    layers.Dense(512, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])
```

```
model.compile(loss='binary_crossentropy',
              optimizer=optimizers.RMSprop(learning_rate=1e-4),
              metrics=['accuracy'])
```

Train and fit the data to the model

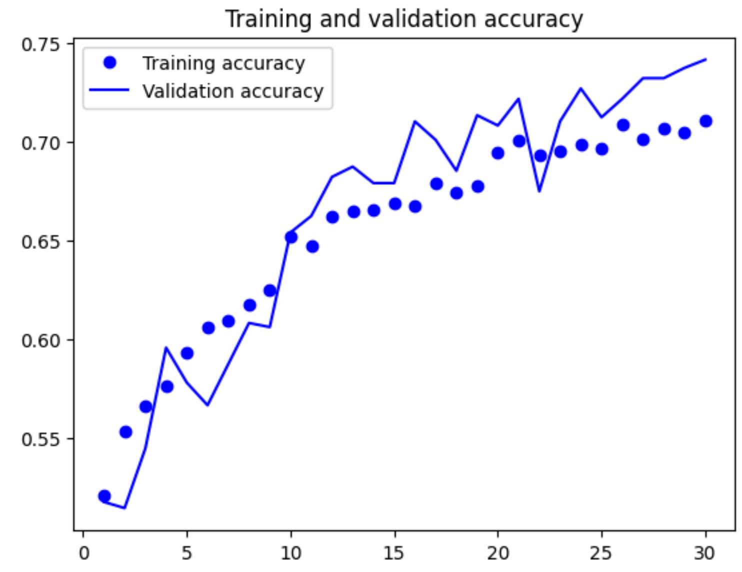
```
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    epochs=30,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // batch_size
)
```

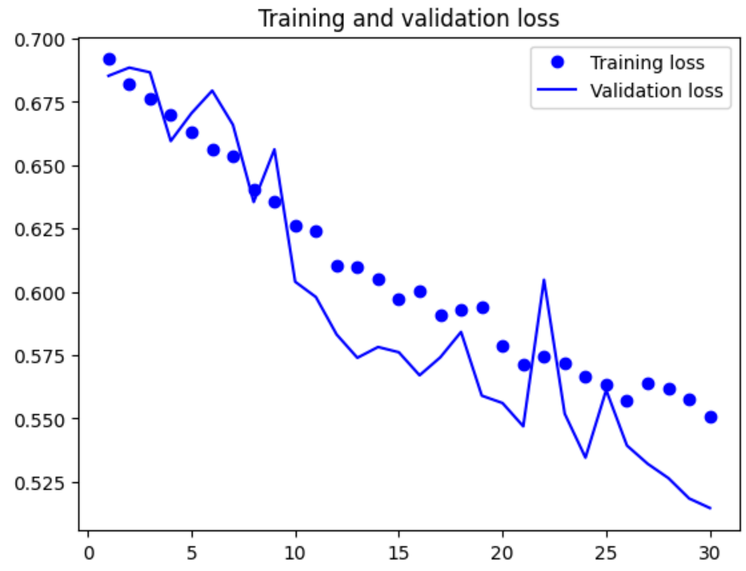
```
Epoch 1/30
62/62 [==============================] - 26s 406ms/step - loss: 0.6918 - accuracy: 0.5211 - val_loss: 0.6853 - val_accuracy: 0.5177
```

```
Epoch 2/30
62/62 [==============================] - 25s 403ms/step - loss: 0.6822 - accuracy: 0.5534 - val_loss: 0.6885 - val_accuracy: 0.5146
Epoch 3/30
62/62 [==============================] - 25s 404ms/step - loss: 0.6762 - accuracy: 0.5663 - val_loss: 0.6866 - val_accuracy: 0.5448
Epoch 4/30
62/62 [==============================] - 25s 404ms/step - loss: 0.6700 - accuracy: 0.5767 - val_loss: 0.6595 - val_accuracy: 0.5958
Epoch 5/30
62/62 [==============================] - 25s 405ms/step - loss: 0.6628 - accuracy: 0.5932 - val_loss: 0.6705 - val_accuracy: 0.5781
Epoch 6/30
62/62 [==============================] - 25s 406ms/step - loss: 0.6560 - accuracy: 0.6065 - val_loss: 0.6794 - val_accuracy: 0.5667
Epoch 7/30
62/62 [==============================] - 25s 405ms/step - loss: 0.6535 - accuracy: 0.6096 - val_loss: 0.6659 - val_accuracy: 0.5875
Epoch 8/30
62/62 [==============================] - 25s 403ms/step - loss: 0.6403 - accuracy: 0.6176 - val_loss: 0.6355 - val_accuracy: 0.6083
Epoch 9/30
62/62 [==============================] - 26s 412ms/step - loss: 0.6356 - accuracy: 0.6253 - val_loss: 0.6562 - val_accuracy: 0.6062
Epoch 10/30
62/62 [==============================] - 25s 410ms/step - loss: 0.6261 - accuracy: 0.6519 - val_loss: 0.6040 - val_accuracy: 0.6542
Epoch 11/30
62/62 [==============================] - 26s 412ms/step - loss: 0.6239 - accuracy: 0.6474 - val_loss: 0.5979 - val_accuracy: 0.6625
Epoch 12/30
62/62 [==============================] - 26s 412ms/step - loss: 0.6104 - accuracy: 0.6623 - val_loss: 0.5831 - val_accuracy: 0.6823
Epoch 13/30
62/62 [==============================] - 25s 405ms/step - loss: 0.6096 - accuracy: 0.6651 - val_loss: 0.5739 - val_accuracy: 0.6875
Epoch 14/30
62/62 [==============================] - 26s 412ms/step - loss: 0.6050 - accuracy: 0.6657 - val_loss: 0.5782 - val_accuracy: 0.6792
Epoch 15/30
62/62 [==============================] - 24s 393ms/step - loss: 0.5971 - accuracy: 0.6687 - val_loss: 0.5761 - val_accuracy: 0.6792
Epoch 16/30
62/62 [==============================] - 25s 398ms/step - loss: 0.6005 - accuracy: 0.6679 - val_loss: 0.5670 - val_accuracy: 0.7104
Epoch 17/30
62/62 [==============================] - 25s 404ms/step - loss: 0.5908 - accuracy: 0.6789 - val_loss: 0.5741 - val_accuracy: 0.7010
Epoch 18/30
62/62 [==============================] - 26s 413ms/step - loss: 0.5931 - accuracy: 0.6745 - val_loss: 0.5841 - val_accuracy: 0.6854
Epoch 19/30
62/62 [==============================] - 25s 405ms/step - loss: 0.5941 - accuracy: 0.6776 - val_loss: 0.5590 - val_accuracy: 0.7135
Epoch 20/30
62/62 [==============================] - 26s 414ms/step - loss: 0.5787 - accuracy: 0.6944 - val_loss: 0.5560 - val_accuracy: 0.7083
Epoch 21/30
62/62 [==============================] - 25s 407ms/step - loss: 0.5710 - accuracy: 0.7010 - val_loss: 0.5469 - val_accuracy: 0.7219
Epoch 22/30
62/62 [==============================] - 25s 401ms/step - loss: 0.5744 - accuracy: 0.6936 - val_loss: 0.6047 - val_accuracy: 0.6750
Epoch 23/30
62/62 [==============================] - 25s 403ms/step - loss: 0.5718 - accuracy: 0.6956 - val_loss: 0.5518 - val_accuracy: 0.7104
Epoch 24/30
62/62 [==============================] - 26s 412ms/step - loss: 0.5668 - accuracy: 0.6989 - val_loss: 0.5345 - val_accuracy: 0.7271
Epoch 25/30
62/62 [==============================] - 25s 402ms/step - loss: 0.5636 - accuracy: 0.6966 - val_loss: 0.5612 - val_accuracy: 0.7125
Epoch 26/30
62/62 [==============================] - 25s 404ms/step - loss: 0.5571 - accuracy: 0.7091 - val_loss: 0.5393 - val_accuracy: 0.7219
Epoch 27/30
62/62 [==============================] - 25s 403ms/step - loss: 0.5640 - accuracy: 0.7015 - val_loss: 0.5321 - val_accuracy: 0.7323
Epoch 28/30
62/62 [==============================] - 25s 405ms/step - loss: 0.5619 - accuracy: 0.7068 - val_loss: 0.5264 - val_accuracy: 0.7323
Epoch 29/30
62/62 [==============================] - 25s 403ms/step - loss: 0.5574 - accuracy: 0.7050 - val_loss: 0.5184 - val_accuracy: 0.7375
```

Displaying curves of loss and accuracy during training

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





Test model

```
# Evaluate the model on test data
test_datagen = ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_directory(
    new_base_dir / "test",
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary'
)
test_loss, test_acc = model.evaluate(test_generator)
print('Test accuracy:', test_acc)
```

```
Found 1000 images belonging to 2 classes.
16/16 [==============================] - 2s 98ms/step - loss: 0.5108 - accuracy: 0.7440
Test accuracy: 0.7440000176429749
```

Step 3

```
import os, shutil, pathlib

original_dir = pathlib.Path("PetImages")
new_base_dir = pathlib.Path("cats_vs_dogs_large2")
```

```
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"{i}.jpg" for i in range(start_index, end_index)]
       for fname in fnames:
           shutil.copyfile(src=original_dir/category / fname,
                           dst=dir / fname)
```

```
make_subset("train", start_index=667, end_index=6667)
make_subset("validation", start_index=6668, end_index=7668)
make_subset("test", start_index=7669, end_index=8669)
```

Building the model

```
img_size = (150, 150)
batch_size = 64
```

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
validation_datagen = ImageDataGenerator(rescale=1./255)
```

```
# Load and augment data
train_generator = train_datagen.flow_from_directory(
    new_base_dir / "train",
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary'
)
validation_generator = validation_datagen.flow_from_directory(
    new_base_dir / "validation",
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary'
)
```

```
Found 12000 images belonging to 2 classes.
Found 2000 images belonging to 2 classes.
```

```
test_generator = validation_datagen.flow_from_directory(
    new_base_dir / "test",
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary'
)
```

```
Found 2000 images belonging to 2 classes.
```

```
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),
    layers.Flatten(),
    layers.Dense(512, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])
```

```
model.compile(loss='binary_crossentropy',
              optimizer=optimizers.RMSprop(learning_rate=1e-4),
              metrics=['accuracy'])
```

Train and fit the data to the model

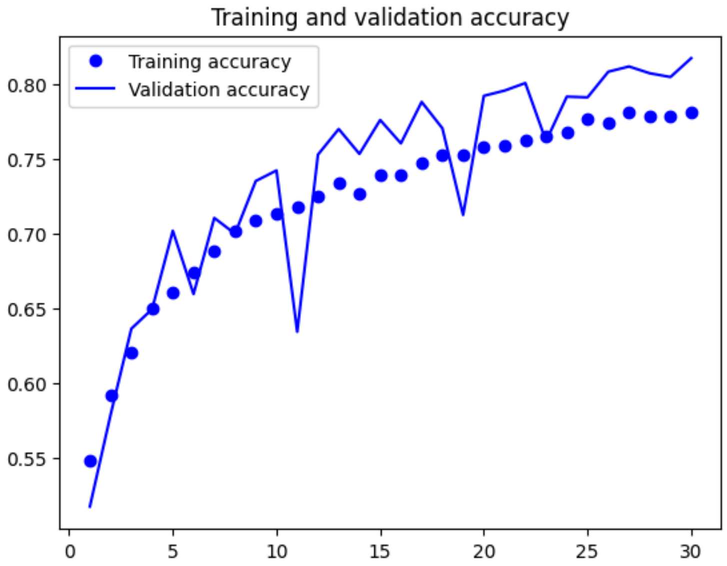
```
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    epochs=30,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // batch_size
)
```

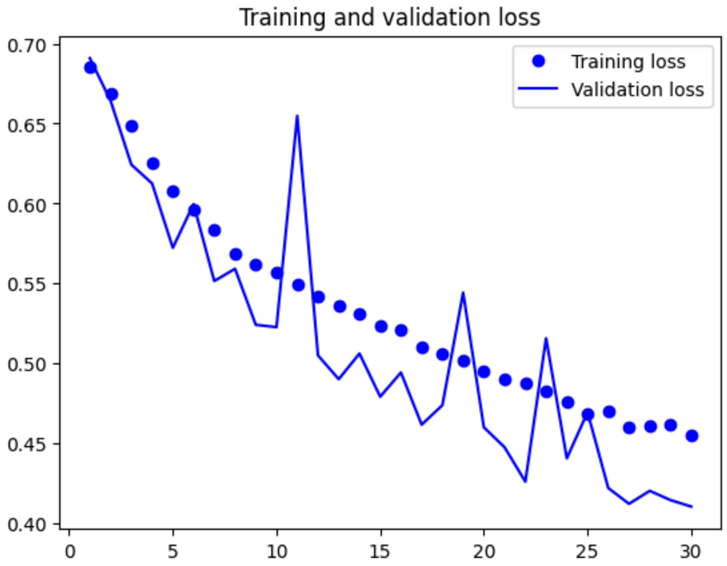
```
187/187 [==============================] - 76s 398ms/step - loss: 0.6858 - accuracy: 0.5478 - val_loss: 0.6909 - val_accuracy: 0.5171
Epoch 2/30
187/187 [==============================] - 74s 398ms/step - loss: 0.6683 - accuracy: 0.5914 - val_loss: 0.6642 - val_accuracy: 0.5786
Epoch 3/30
187/187 [==============================] - 73s 392ms/step - loss: 0.6490 - accuracy: 0.6203 - val_loss: 0.6243 - val_accuracy: 0.6361
Epoch 4/30
```

```
187/187 [==============================] - 74s 394ms/step - loss: 0.6082 - accuracy: 0.6603 - val_loss: 0.5723 - val_accuracy: 0.7016
Epoch 6/30
187/187 [==============================] - 73s 392ms/step - loss: 0.5965 - accuracy: 0.6733 - val_loss: 0.5996 - val_accuracy: 0.6593
Epoch 7/30
187/187 [==============================] - 74s 397ms/step - loss: 0.5832 - accuracy: 0.6880 - val_loss: 0.5515 - val_accuracy: 0.7102
Epoch 8/30
187/187 [==============================] - 75s 400ms/step - loss: 0.5689 - accuracy: 0.7018 - val_loss: 0.5592 - val_accuracy: 0.6996
Epoch 9/30
187/187 [==============================] - 75s 400ms/step - loss: 0.5619 - accuracy: 0.7082 - val_loss: 0.5241 - val_accuracy: 0.7349
Epoch 10/30
187/187 [==============================] - 74s 395ms/step - loss: 0.5566 - accuracy: 0.7132 - val_loss: 0.5227 - val_accuracy: 0.7419
Epoch 11/30
187/187 [==============================] - 74s 397ms/step - loss: 0.5491 - accuracy: 0.7172 - val_loss: 0.6549 - val_accuracy: 0.6341
Epoch 12/30
187/187 [==============================] - 75s 403ms/step - loss: 0.5420 - accuracy: 0.7250 - val_loss: 0.5050 - val_accuracy: 0.7525
Epoch 13/30
187/187 [==============================] - 73s 391ms/step - loss: 0.5361 - accuracy: 0.7333 - val_loss: 0.4901 - val_accuracy: 0.7697
Epoch 14/30
187/187 [==============================] - 74s 395ms/step - loss: 0.5314 - accuracy: 0.7268 - val_loss: 0.5062 - val_accuracy: 0.7530
Epoch 15/30
187/187 [==============================] - 73s 390ms/step - loss: 0.5237 - accuracy: 0.7388 - val_loss: 0.4792 - val_accuracy: 0.7757
Epoch 16/30
187/187 [==============================] - 74s 395ms/step - loss: 0.5210 - accuracy: 0.7385 - val_loss: 0.4943 - val_accuracy: 0.7601
Epoch 17/30
187/187 [==============================] - 75s 399ms/step - loss: 0.5102 - accuracy: 0.7468 - val_loss: 0.4617 - val_accuracy: 0.7878
Epoch 18/30
187/187 [==============================] - 74s 394ms/step - loss: 0.5056 - accuracy: 0.7525 - val_loss: 0.4739 - val_accuracy: 0.7702
Epoch 19/30
187/187 [==============================] - 74s 395ms/step - loss: 0.5017 - accuracy: 0.7523 - val_loss: 0.5443 - val_accuracy: 0.7122
Epoch 20/30
187/187 [==============================] - 76s 407ms/step - loss: 0.4951 - accuracy: 0.7574 - val_loss: 0.4600 - val_accuracy: 0.7918
Epoch 21/30
187/187 [==============================] - 74s 396ms/step - loss: 0.4898 - accuracy: 0.7581 - val_loss: 0.4475 - val_accuracy: 0.7954
Epoch 22/30
187/187 [==============================] - 76s 406ms/step - loss: 0.4876 - accuracy: 0.7620 - val_loss: 0.4261 - val_accuracy: 0.8004
Epoch 23/30
187/187 [==============================] - 75s 400ms/step - loss: 0.4829 - accuracy: 0.7650 - val_loss: 0.5157 - val_accuracy: 0.7616
Epoch 24/30
187/187 [==============================] - 74s 396ms/step - loss: 0.4759 - accuracy: 0.7676 - val_loss: 0.4406 - val_accuracy: 0.7913
Epoch 25/30
187/187 [==============================] - 73s 388ms/step - loss: 0.4688 - accuracy: 0.7760 - val_loss: 0.4694 - val_accuracy: 0.7908
Epoch 26/30
187/187 [==============================] - 71s 378ms/step - loss: 0.4698 - accuracy: 0.7740 - val_loss: 0.4220 - val_accuracy: 0.8080
Epoch 27/30
187/187 [==============================] - 71s 378ms/step - loss: 0.4605 - accuracy: 0.7806 - val_loss: 0.4122 - val_accuracy: 0.8115
Epoch 28/30
187/187 [==============================] - 71s 380ms/step - loss: 0.4613 - accuracy: 0.7782 - val_loss: 0.4203 - val_accuracy: 0.8070
Epoch 29/30
187/187 [==============================] - 72s 386ms/step - loss: 0.4619 - accuracy: 0.7785 - val_loss: 0.4145 - val_accuracy: 0.8044
Epoch 30/30
187/187 [==============================] - 73s 391ms/step - loss: 0.4547 - accuracy: 0.7808 - val_loss: 0.4105 - val_accuracy: 0.8170
```

Displaying curves of loss and accuracy during training

```python
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()
```





Test the model

```python
# Evaluate the model on test data
test_datagen = ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_directory(
    new_base_dir / "test",
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary'
)
test_loss, test_acc = model.evaluate(test_generator)
print('Test accuracy:', test_acc)
```

```
Found 2000 images belonging to 2 classes.
32/32 [==============================] - 3s 94ms/step - loss: 0.3977 - accuracy: 0.8185
Test accuracy: 0.8184999823570251
```

Step 4

```
import tensorflow as tf
from tensorflow.keras import layers, models, optimizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
import os, shutil, pathlib

original_dir = pathlib.Path("PetImages")
new_base_dir = pathlib.Path("cats_vs_dogs_pretrained")
```

```
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"{i}.jpg" for i in range(start_index, end_index)]
       for fname in fnames:
           shutil.copyfile(src=original_dir/category / fname,
                           dst=dir / fname)
```

```
make_subset("train", start_index=667, end_index=6667)
make_subset("validation", start_index=6668, end_index=7668)
make_subset("test", start_index=7669, end_index=8669)
```

Building the model

```
img_size = (150, 150)
batch_size = 64
```

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
validation_datagen = ImageDataGenerator(rescale=1./255)
```

```
pretrained_model = tf.keras.applications.VGG16(
    weights='imagenet',
    include_top=False,
    input_shape=(150, 150, 3)
)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [==============================] - 2s 0us/step
```

```
pretrained_model.trainable = False
```

```
model_pretrained = models.Sequential([
    pretrained_model,
    layers.Flatten(),
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(1, activation='sigmoid')
])
```

```
model_pretrained.compile(loss='binary_crossentropy',
                         optimizer=optimizers.RMSprop(learning_rate=2e-5),
                         metrics=['accuracy'])
```

```
train_generator = train_datagen.flow_from_directory(
    new_base_dir / "train",
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary'
)

validation_generator = validation_datagen.flow_from_directory(
    new_base_dir / "validation",
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary'
)
```

```
Found 12000 images belonging to 2 classes.
Found 2000 images belonging to 2 classes.
```

```
test_generator = validation_datagen.flow_from_directory(
    new_base_dir / "test",
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary'
)
```

```
Found 2000 images belonging to 2 classes.
```

```
history_pretrained = model_pretrained.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    epochs=30,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // batch_size
)
```

```
187/187 [==============================] - 77s 380ms/step - loss: 0.5742 - accuracy: 0.6976 - val_loss: 0.3795 - val_accuracy: 0.8518
Epoch 2/30
```

```
187/187 [==============================] - 72s 383ms/step - loss: 0.4114 - accuracy: 0.8111 - val_loss: 0.3067 - val_accuracy: 0.8664
Epoch 4/30
187/187 [==============================] - 72s 383ms/step - loss: 0.3839 - accuracy: 0.8264 - val_loss: 0.3044 - val_accuracy: 0.8710
Epoch 5/30
187/187 [==============================] - 72s 384ms/step - loss: 0.3724 - accuracy: 0.8282 - val_loss: 0.2889 - val_accuracy: 0.8730
Epoch 6/30
187/187 [==============================] - 73s 389ms/step - loss: 0.3659 - accuracy: 0.8327 - val_loss: 0.2791 - val_accuracy: 0.8816
Epoch 7/30
187/187 [==============================] - 72s 386ms/step - loss: 0.3571 - accuracy: 0.8364 - val_loss: 0.2761 - val_accuracy: 0.8831
Epoch 8/30
187/187 [==============================] - 73s 389ms/step - loss: 0.3449 - accuracy: 0.8459 - val_loss: 0.2753 - val_accuracy: 0.8826
Epoch 9/30
187/187 [==============================] - 72s 383ms/step - loss: 0.3509 - accuracy: 0.8425 - val_loss: 0.2675 - val_accuracy: 0.8866
Epoch 10/30
187/187 [==============================] - 71s 380ms/step - loss: 0.3353 - accuracy: 0.8534 - val_loss: 0.2723 - val_accuracy: 0.8881
Epoch 11/30
187/187 [==============================] - 71s 381ms/step - loss: 0.3360 - accuracy: 0.8512 - val_loss: 0.2618 - val_accuracy: 0.8891
Epoch 12/30
187/187 [==============================] - 74s 393ms/step - loss: 0.3298 - accuracy: 0.8564 - val_loss: 0.2624 - val_accuracy: 0.8916
Epoch 13/30
187/187 [==============================] - 72s 386ms/step - loss: 0.3289 - accuracy: 0.8560 - val_loss: 0.2649 - val_accuracy: 0.8891
Epoch 14/30
187/187 [==============================] - 72s 387ms/step - loss: 0.3218 - accuracy: 0.8620 - val_loss: 0.2621 - val_accuracy: 0.8896
Epoch 15/30
187/187 [==============================] - 71s 379ms/step - loss: 0.3239 - accuracy: 0.8553 - val_loss: 0.2590 - val_accuracy: 0.8876
Epoch 16/30
187/187 [==============================] - 72s 385ms/step - loss: 0.3150 - accuracy: 0.8610 - val_loss: 0.2619 - val_accuracy: 0.8901
Epoch 17/30
187/187 [==============================] - 71s 377ms/step - loss: 0.3187 - accuracy: 0.8577 - val_loss: 0.2616 - val_accuracy: 0.8911
Epoch 18/30
187/187 [==============================] - 72s 383ms/step - loss: 0.3249 - accuracy: 0.8565 - val_loss: 0.2562 - val_accuracy: 0.8891
Epoch 19/30
187/187 [==============================] - 72s 383ms/step - loss: 0.3099 - accuracy: 0.8622 - val_loss: 0.2647 - val_accuracy: 0.8906
Epoch 20/30
187/187 [==============================] - 71s 378ms/step - loss: 0.3088 - accuracy: 0.8675 - val_loss: 0.2543 - val_accuracy: 0.8921
Epoch 21/30
187/187 [==============================] - 72s 383ms/step - loss: 0.3022 - accuracy: 0.8651 - val_loss: 0.2541 - val_accuracy: 0.8926
Epoch 22/30
187/187 [==============================] - 72s 385ms/step - loss: 0.3090 - accuracy: 0.8642 - val_loss: 0.2522 - val_accuracy: 0.8972
Epoch 23/30
187/187 [==============================] - 71s 378ms/step - loss: 0.3084 - accuracy: 0.8657 - val_loss: 0.2478 - val_accuracy: 0.8967
Epoch 24/30
187/187 [==============================] - 71s 378ms/step - loss: 0.3025 - accuracy: 0.8684 - val_loss: 0.2543 - val_accuracy: 0.8962
Epoch 25/30
187/187 [==============================] - 71s 378ms/step - loss: 0.2975 - accuracy: 0.8710 - val_loss: 0.2532 - val_accuracy: 0.8967
Epoch 26/30
187/187 [==============================] - 72s 385ms/step - loss: 0.2994 - accuracy: 0.8699 - val_loss: 0.2525 - val_accuracy: 0.8957
Epoch 27/30
187/187 [==============================] - 72s 385ms/step - loss: 0.2962 - accuracy: 0.8704 - val_loss: 0.2458 - val_accuracy: 0.8977
Epoch 28/30
187/187 [==============================] - 72s 385ms/step - loss: 0.2944 - accuracy: 0.8682 - val_loss: 0.2479 - val_accuracy: 0.8987
Epoch 29/30
187/187 [==============================] - 71s 381ms/step - loss: 0.2948 - accuracy: 0.8701 - val_loss: 0.2488 - val_accuracy: 0.8987
Epoch 30/30
```

Test the model

```python
test_generator = validation_datagen.flow_from_directory(
    new_base_dir / "test",
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary',
    shuffle=False
)
```

```
Found 2000 images belonging to 2 classes.
```

```python
test_loss_pretrained, test_acc_pretrained = model_pretrained.evaluate(test_generator)
print('Pretrained model test accuracy:', test_acc_pretrained)
```

```
32/32 [==============================] - 4s 127ms/step - loss: 0.2077 - accuracy: 0.9140
Pretrained model test accuracy: 0.9139999747276306
```