```
!wget --no-check-certificate \
    https://storage.googleapis.com/mledu-datasets/cats_and_dogs_filtered.zip \
    -O /tmp/cats_and_dogs_filtered.zip
```

```
--2021-12-06 14:04:28--  https://storage.googleapis.com/mledu-datasets/cats_and_dogs_
Resolving storage.googleapis.com (storage.googleapis.com)... 142.250.81.208, 142.250
Connecting to storage.googleapis.com (storage.googleapis.com)|142.250.81.208|:443...
HTTP request sent, awaiting response... 200 OK
Length: 68606236 (65M) [application/zip]
Saving to: ‘/tmp/cats_and_dogs_filtered.zip’

/tmp/cats_and_dogs_ 100%[===================>]  65.43M   113MB/s    in 0.6s

2021-12-06 14:04:29 (113 MB/s) - ‘/tmp/cats_and_dogs_filtered.zip’ saved [68606236/68
```

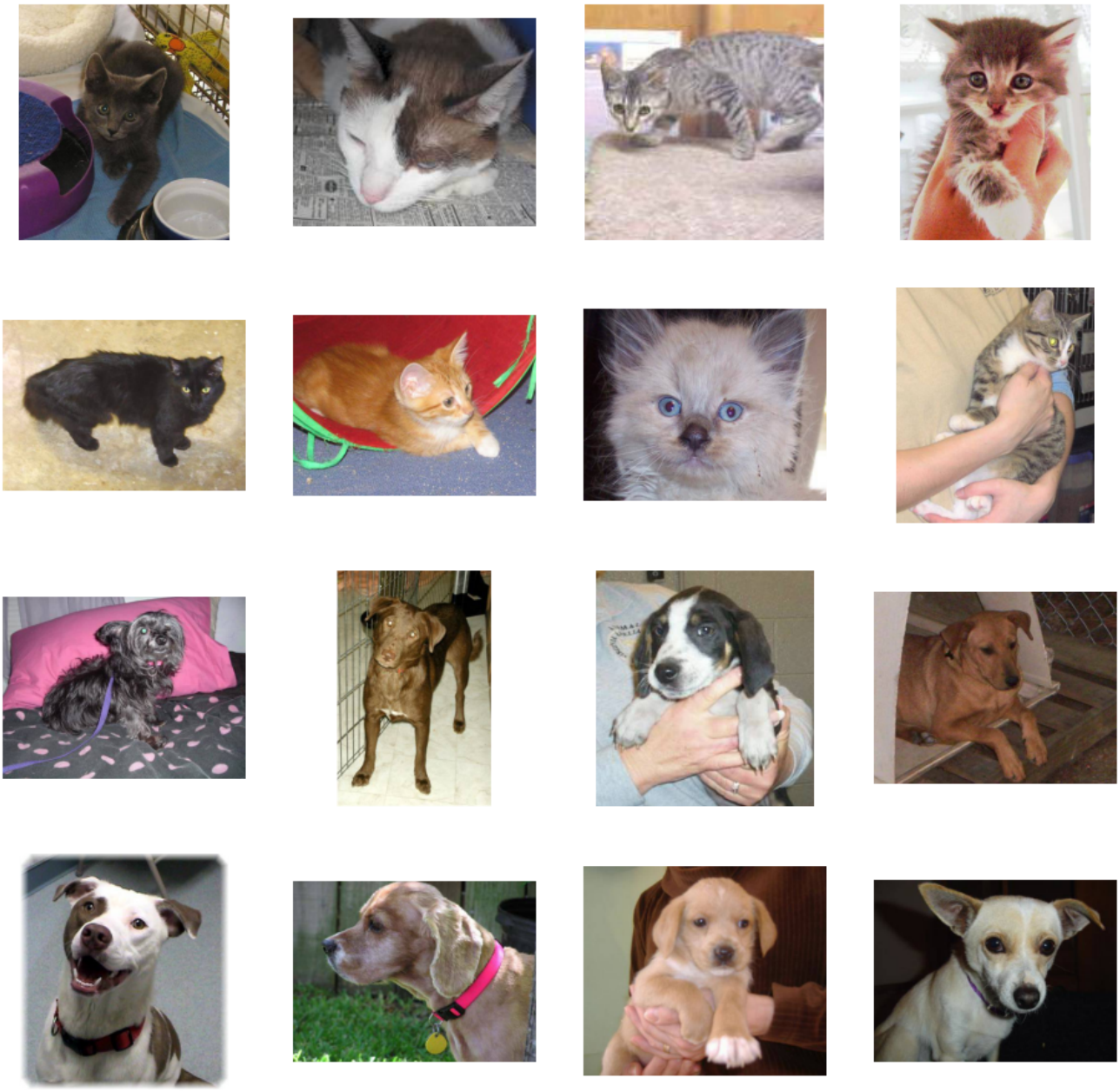
```
import os
import zipfile

local_zip = '/tmp/cats_and_dogs_filtered.zip'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('/tmp')
zip_ref.close()


base_dir = '/tmp/cats_and_dogs_filtered'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'validation')

# Directory with our training cat pictures
train_cats_dir = os.path.join(train_dir, 'cats')

# Directory with our training dog pictures
train_dogs_dir = os.path.join(train_dir, 'dogs')

# Directory with our validation cat pictures
validation_cats_dir = os.path.join(validation_dir, 'cats')

# Directory with our validation dog pictures
validation_dogs_dir = os.path.join(validation_dir, 'dogs')


train_cat_fnames = os.listdir(train_cats_dir)
print(train_cat_fnames[:10])

train_dog_fnames = os.listdir(train_dogs_dir)
train_dog_fnames.sort()
print(train_dog_fnames[:10])
```

```
['cat.767.jpg', 'cat.735.jpg', 'cat.261.jpg', 'cat.100.jpg', 'cat.359.jpg', 'cat.574
['dog.0.jpg', 'dog.1.jpg', 'dog.10.jpg', 'dog.100.jpg', 'dog.101.jpg', 'dog.102.jpg',
```

```
print('total training cat images:', len(os.listdir(train_cats_dir)))
print('total training dog images:', len(os.listdir(train_dogs_dir)))
print('total validation cat images:', len(os.listdir(validation_cats_dir)))
print('total validation dog images:', len(os.listdir(validation_dogs_dir)))
```

```
total training cat images: 1000
total training dog images: 1000
total validation cat images: 500
total validation dog images: 500
```

```
%matplotlib inline

import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# Parameters for our graph; we'll output images in a 4x4 configuration
nrows = 4
ncols = 4

# Index for iterating over images
pic_index = 0


# Set up matplotlib fig, and size it to fit 4x4 pics
fig = plt.gcf()
fig.set_size_inches(ncols * 4, nrows * 4)

pic_index += 8
next_cat_pix = [os.path.join(train_cats_dir, fname)
                for fname in train_cat_fnames[pic_index-8:pic_index]]
next_dog_pix = [os.path.join(train_dogs_dir, fname)
                for fname in train_dog_fnames[pic_index-8:pic_index]]

for i, img_path in enumerate(next_cat_pix+next_dog_pix):
  # Set up subplot; subplot indices start at 1
  sp = plt.subplot(nrows, ncols, i + 1)
  sp.axis('Off') # Don't show axes (or gridlines)

  img = mpimg.imread(img_path)
  plt.imshow(img)

plt.show()
```



```
from tensorflow.keras import layers
from tensorflow.keras import Model


# Our input feature map is 150x150x3: 150x150 for the image pixels, and 3 for
# the three color channels: R, G, and B
img_input = layers.Input(shape=(150, 150, 3))

# First convolution extracts 16 filters that are 3x3
# Convolution is followed by max-pooling layer with a 2x2 window
x = layers.Conv2D(16, 3, activation='relu')(img_input)
x = layers.MaxPooling2D(2)(x)

# Second convolution extracts 32 filters that are 3x3
# Convolution is followed by max-pooling layer with a 2x2 window
x = layers.Conv2D(32, 3, activation='relu')(x)
x = layers.MaxPooling2D(2)(x)

# Third convolution extracts 64 filters that are 3x3
# Convolution is followed by max-pooling layer with a 2x2 window
```

```
x = layers.Conv2D(64, 3, activation='relu')(x)
x = layers.MaxPooling2D(2)(x)


# Flatten feature map to a 1-dim tensor so we can add fully connected layers
x = layers.Flatten()(x)

# Create a fully connected layer with ReLU activation and 512 hidden units
x = layers.Dense(512, activation='relu')(x)

# Create output layer with a single node and sigmoid activation
output = layers.Dense(1, activation='sigmoid')(x)

# Create model:
# input = input feature map
# output = input feature map + stacked convolution/maxpooling layers + fully
# connected layer + sigmoid output layer
model = Model(img_input, output)

print(model)
```

```
<tensorflow.python.keras.engine.functional.Functional object at 0x7f7a4cd820d0>
```

```
!pip install tensorflow-quantum
```

```
Requirement already satisfied: tensorflow-quantum in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: google-auth==1.18.0 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: google-api-core==1.21.0 in /usr/local/lib/python3.7/di
Requirement already satisfied: sympy==1.5 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: cirq==0.11.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: protobuf==3.13.0 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: googleapis-common-protos==1.52.0 in /usr/local/lib/pyt
Requirement already satisfied: cirq-google==0.11.0 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: cirq-core==0.11.0 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: networkx~=2.4 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: numpy~=1.16 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: matplotlib~=3.0 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: sortedcontainers~=2.0 in /usr/local/lib/python3.7/dist
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from c
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: requests~=2.18 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: pytz in /usr/local/lib/python3.7/dist-packages (from g
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: setuptools>=34.0.0 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.7/dis
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.7/dist
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: grpcio<2.0dev,>=1.29.0 in /usr/local/lib/python3.7/dis
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local
```

```
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-pa
```

```
!pip install -q cirq
!pip install tensorflow==2.4.1
```

```
Collecting tensorflow==2.4.1
  Downloading tensorflow-2.4.1-cp37-cp37m-manylinux2010_x86_64.whl (394.3 MB)
     |████████████████████████████████| 394.3 MB 14 kB/s
Requirement already satisfied: opt-einsum~=3.3.0 in /usr/local/lib/python3.7/dist-
Collecting h5py~=2.10.0
  Downloading h5py-2.10.0-cp37-cp37m-manylinux1_x86_64.whl (2.9 MB)
     |████████████████████████████████| 2.9 MB 67.9 MB/s
Requirement already satisfied: google-pasta~=0.2 in /usr/local/lib/python3.7/dist-
```

```
import tensorflow as tf
import tensorflow_quantum as tfq

import cirq
import sympy
import numpy as np

import seaborn as sns
import collections

# visualization tools
%matplotlib inline
import matplotlib.pyplot as plt
from cirq.contrib.svg import SVGCircuit

from sklearn.model_selection import train_test_split
```

```
Collecting gast==0.3.3
```

```
def binary_encode(x,threshold=0.5):
  """
  Encodes the given datset to use binary encoding

  Parameters:
  X(array) : Image data to be processed for encoding
  threshold(float): Threshold for binary encoding, 0.5 by default

  Returns:
  encoded_images(array): Binary encoded Image Data

  """
  encoded_images = list()
  for image in x:
    # pixel value is 1 if it's greater than threshold or else zero
    encoded_image = [1 if j>threshold else 0 for j in image[0]]
    encoded_images.append(encoded_image)
  return np.array(encoded_images)
```

```
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-
```

```
def create_circuit_from_image(encoded_image):
  """
  Returns a circuit for given encoded image

  Parameters:
  encoded_image (array): Encoded Image

  Returns:
  circuit (cirq.Circuit object): cirq circuit
```

```
    """
    qubits = cirq.GridQubit.rect(2,2)
    circuit = cirq.Circuit()
    for i, pixel in enumerate(encoded_image):
      if pixel:
        circuit.append(cirq.X(qubits[i]))
    return circuit


model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 150, 150, 3)]     0

conv2d (Conv2D)              (None, 148, 148, 16)      448

max_pooling2d (MaxPooling2D  (None, 74, 74, 16)       0
)

conv2d_1 (Conv2D)            (None, 72, 72, 32)        4640

max_pooling2d_1 (MaxPooling  (None, 36, 36, 32)       0
2D)

conv2d_2 (Conv2D)            (None, 34, 34, 64)        18496

max_pooling2d_2 (MaxPooling  (None, 17, 17, 64)       0
2D)

flatten (Flatten)            (None, 18496)             0

dense (Dense)                (None, 512)               9470464

dense_1 (Dense)              (None, 1)                 513

=================================================================
Total params: 9,494,561
Trainable params: 9,494,561
Non-trainable params: 0
_________________________________________________________________
```

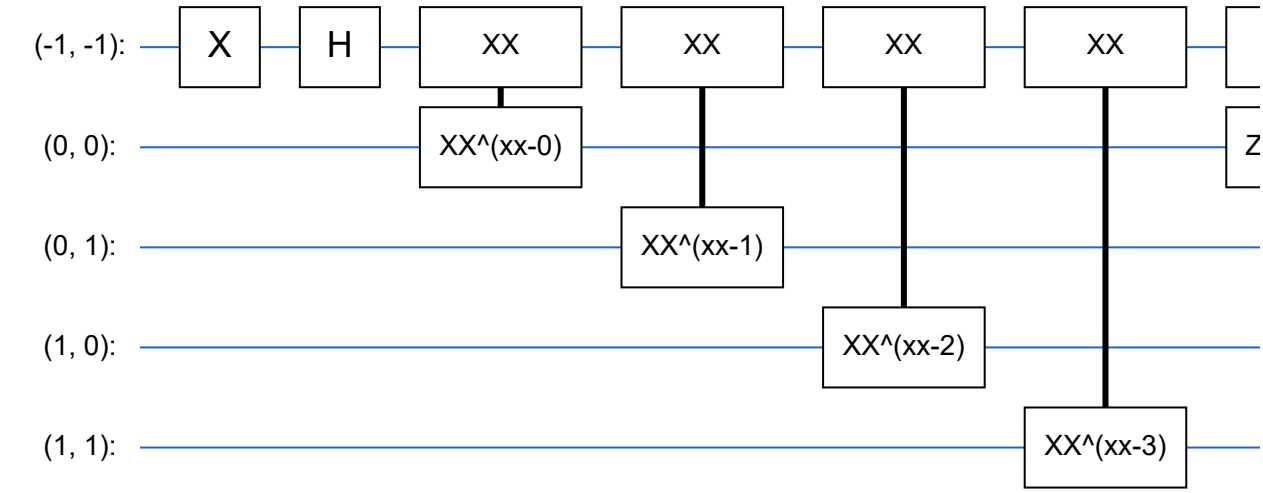
```
class QNN():
    def __init__(self, data_qubits, readout):
      self.data_qubits = data_qubits
      self.readout = readout

    def add_singleQubit_gate(self,circuit, gate, qubit_index):
      """
      Adds single qubit gate to the circuit
      Parameters:
      circuit(cirq.Circuit object): Cirq circuit
      gate(cirq gate): gate to append to the circuit
      qubits(list): index of qubits to apply the gate
      Returns:
```

```
        None
        """
        for index in qubit_index:
          circuit.append(gate(self.data_qubits[index]))

    def add_twoQubit_gate(self,circuit, gate, qubit_index):
        """
        Adds two qubit gate to the circuit
        Parameters:
        circuit(cirq.Circuit object): Cirq circuit
        gate(cirq gate): gate to append to the circuit
        qubits(list): index of qubits to apply the gate
        Returns:
        None
        """
        if len(qubit_index)!=2:
          raise Exception("The length of the list of indices passed for two qubit \
          gate operations must be equal to two")
        circuit.append(gate(self.data_qubits[qubit_index[0]], self.data_qubits[qubit_index[1

    def add_layer(self, circuit, gate, symbol_gate):
        """
        Adds New Gates/Layers to the Circuit
        Parameters:
        circuit(cirq.Circuit object): Cirq circuit
        gate(cirq gate): gate to append to the circuit
        symbol_gate(string): symbol for the gate
        Returns:
        None
        """
        for i, qubit in enumerate(self.data_qubits):
          symbol = sympy.Symbol(symbol_gate+ '-' + str(i))
          circuit.append(gate(qubit, self.readout)**symbol)


def create_qnn():
    """Create a QNN model circuit and readout operation to go along with it."""
    data_qubits = cirq.GridQubit.rect(2,2)  # a 4x4 grid.
    readout = cirq.GridQubit(-1, -1)         # a single qubit at [-1,-1]
    circuit = cirq.Circuit()

    # Prepare the readout qubit.
    circuit.append(cirq.X(readout))
    circuit.append(cirq.H(readout))

    qnn = QNN(
        data_qubits = data_qubits,
        readout=readout)

    """
    # Though we don't use single and double Qubit Gates in our Circuit, we provide
    # the methods "add_singleQubit_gate" and "add_twoQubit_gate" for our Class QNN
    # that can be used to add Single and Double Qubit Gates respectively.
    # An exmaple is shown below:
```

```
    #Add Hadamard Gates
    qnn.add_singleQubit_gate(circuit, cirq.H, [0,1,2,3])

    #Add CNOT gates
    qnn.add_twoQubit_gate(circuit, cirq.CNOT, [0, 1])
    qnn.add_twoQubit_gate(circuit, cirq.CNOT, [2, 3])
    """

    # Add the ising coupling XX gate
    qnn.add_layer(circuit, cirq.XX, "xx")
    qnn.add_layer(circuit, cirq.ZZ, "zz")

    # Finally, prepare the readout qubit.
    circuit.append(cirq.H(readout))

    return circuit, cirq.Z(readout)

qmodel, model_readout = create_qnn()
```

```
SVGCircuit(qmodel)
```

```
findfont: Font family ['Arial'] not found. Falling back to DejaVu Sans.
```




```
from tensorflow.keras.optimizers import RMSprop

model.compile(loss='binary_crossentropy',
              optimizer=RMSprop(lr=0.001),
              metrics=['acc'])
```

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/rmsprop.py:130: UserWarning
  super(RMSprop, self).__init__(name, **kwargs)
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# All images will be rescaled by 1./255
train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)
```

```
# Flow training images in batches of 20 using train_datagen generator
train_generator = train_datagen.flow_from_directory(
        train_dir,  # This is the source directory for training images
        target_size=(150, 150),  # All images will be resized to 150x150
        batch_size=20,
        # Since we use binary_crossentropy loss, we need binary labels
        class_mode='binary')

# Flow validation images in batches of 20 using val_datagen generator
validation_generator = val_datagen.flow_from_directory(
        validation_dir,
        target_size=(150, 150),
        batch_size=20,
        class_mode='binary')
```

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

```
history = model.fit_generator(
      train_generator,
      steps_per_epoch=100,  # 2000 images = batch_size * steps
      epochs=15,
      validation_data=validation_generator,
      validation_steps=50,  # 1000 images = batch_size * steps
      verbose=2)
```

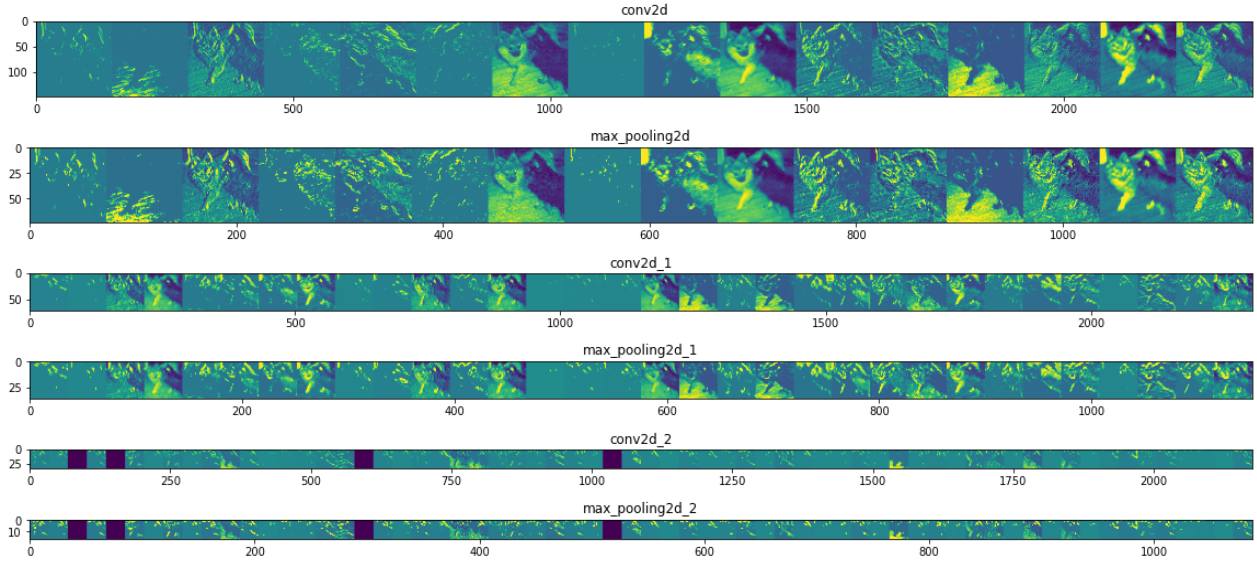
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: UserWarning: `Model.f
  import sys
Epoch 1/15
100/100 - 56s - loss: 0.8516 - acc: 0.5685 - val_loss: 0.6684 - val_acc: 0.5690 - 56s
Epoch 2/15
100/100 - 55s - loss: 0.6446 - acc: 0.6300 - val_loss: 0.6015 - val_acc: 0.6600 - 55s
Epoch 3/15
100/100 - 55s - loss: 0.5455 - acc: 0.7260 - val_loss: 0.5659 - val_acc: 0.7350 - 55s
Epoch 4/15
100/100 - 55s - loss: 0.4716 - acc: 0.7755 - val_loss: 0.5666 - val_acc: 0.7050 - 55s
Epoch 5/15
100/100 - 55s - loss: 0.3815 - acc: 0.8310 - val_loss: 0.5847 - val_acc: 0.7280 - 55s
Epoch 6/15
100/100 - 55s - loss: 0.3148 - acc: 0.8720 - val_loss: 0.7921 - val_acc: 0.7100 - 55s
Epoch 7/15
100/100 - 57s - loss: 0.2100 - acc: 0.9105 - val_loss: 0.9534 - val_acc: 0.6570 - 57s
Epoch 8/15
100/100 - 55s - loss: 0.1460 - acc: 0.9405 - val_loss: 1.6490 - val_acc: 0.6540 - 55s
Epoch 9/15
100/100 - 55s - loss: 0.1189 - acc: 0.9545 - val_loss: 1.1396 - val_acc: 0.7070 - 55s
Epoch 10/15
100/100 - 55s - loss: 0.0991 - acc: 0.9715 - val_loss: 1.2568 - val_acc: 0.7300 - 55s
Epoch 11/15
100/100 - 55s - loss: 0.0631 - acc: 0.9840 - val_loss: 1.4959 - val_acc: 0.7160 - 55s
Epoch 12/15
100/100 - 55s - loss: 0.0466 - acc: 0.9855 - val_loss: 1.2664 - val_acc: 0.7070 - 55s
Epoch 13/15
100/100 - 55s - loss: 0.0476 - acc: 0.9895 - val_loss: 1.9454 - val_acc: 0.7100 - 55s
Epoch 14/15
100/100 - 55s - loss: 0.0830 - acc: 0.9800 - val_loss: 1.9980 - val_acc: 0.7050 - 55s
```

```
Epoch 15/15
100/100 - 55s - loss: 0.0348 - acc: 0.9910 - val_loss: 2.2995 - val_acc: 0.7110 - 55s
```

```
import numpy as np
import random
from tensorflow.keras.preprocessing.image import img_to_array, load_img

# Let's define a new Model that will take an image as input, and will output
# intermediate representations for all layers in the previous model after
# the first.
successive_outputs = [layer.output for layer in model.layers[1:]]
visualization_model = Model(img_input, successive_outputs)

# Let's prepare a random input image of a cat or dog from the training set.
cat_img_files = [os.path.join(train_cats_dir, f) for f in train_cat_fnames]
dog_img_files = [os.path.join(train_dogs_dir, f) for f in train_dog_fnames]
img_path = random.choice(cat_img_files + dog_img_files)

img = load_img(img_path, target_size=(150, 150))  # this is a PIL image
x = img_to_array(img)  # Numpy array with shape (150, 150, 3)
x = x.reshape((1,) + x.shape)  # Numpy array with shape (1, 150, 150, 3)

# Rescale by 1/255
x /= 255

# Let's run our image through our network, thus obtaining all
# intermediate representations for this image.
successive_feature_maps = visualization_model.predict(x)

# These are the names of the layers, so can have them as part of our plot
layer_names = [layer.name for layer in model.layers[1:]]

# Now let's display our representations
for layer_name, feature_map in zip(layer_names, successive_feature_maps):
  if len(feature_map.shape) == 4:
    # Just do this for the conv / maxpool layers, not the fully-connected layers
    n_features = feature_map.shape[-1]  # number of features in feature map
    # The feature map has shape (1, size, size, n_features)
    size = feature_map.shape[1]
    # We will tile our images in this matrix
    display_grid = np.zeros((size, size * n_features))
    for i in range(n_features):
      # Postprocess the feature to make it visually palatable
      x = feature_map[0, :, :, i]
      x -= x.mean()
      x /= x.std()
      x *= 64
      x += 128
      x = np.clip(x, 0, 255).astype('uint8')
      # We'll tile each filter into this big horizontal grid
      display_grid[:, i * size : (i + 1) * size] = x
    # Display the grid
    scale = 20. / n_features
```

```
    plt.figure(figsize=(scale * n_features, scale))
    plt.title(layer_name)
    plt.grid(False)
    plt.imshow(display_grid, aspect='auto', cmap='viridis')
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:43: RuntimeWarning: i
```




```
# Retrieve a list of accuracy results on training and validation data
# sets for each training epoch
acc = history.history['acc']
val_acc = history.history['val_acc']

# Retrieve a list of list results on training and validation data
# sets for each training epoch
loss = history.history['loss']
val_loss = history.history['val_loss']

# Get number of epochs
epochs = range(len(acc))

# Plot training and validation accuracy per epoch
plt.plot(epochs, acc)
plt.plot(epochs, val_acc)
plt.title('Training and validation accuracy')

plt.figure()
```

```
# Plot training and validation loss per epoch
plt.plot(epochs, loss)
plt.plot(epochs, val_loss)
plt.title('Training and validation loss')
```

```
Text(0.5, 1.0, 'Training and validation loss')
```