import matplotlib.pyplot as plt

import numpy as np

import os

import tensorflow as tf

\_URL = 'https://storage.googleapis.com/mledu-datasets/cats\_and\_dogs\_filtered.zip'

path\_to\_zip = tf.keras.utils.get\_file('cats\_and\_dogs.zip', origin=\_URL, extract=True)

PATH = os.path.join(os.path.dirname(path\_to\_zip), 'cats\_and\_dogs\_filtered')

train\_dir = os.path.join(PATH, 'train')

validation\_dir = os.path.join(PATH, 'validation')

BATCH\_SIZE = 32

IMG\_SIZE = (160, 160)

train\_dataset = tf.keras.utils.image\_dataset\_from\_directory(train\_dir,shuffle=True,batch\_size=BATCH\_SIZE,image\_size=IMG\_SIZE)

validation\_dataset = tf.keras.utils.image\_dataset\_from\_directory(validation\_dir,

                                                                 shuffle=True,

                                                                 batch\_size=BATCH\_SIZE,

                                                                 image\_size=IMG\_SIZE)

class\_names = train\_dataset.class\_names

plt.figure(figsize=(10, 10))

for images, labels in train\_dataset.take(1):

  for i in range(9):

    ax = plt.subplot(3, 3, i + 1)

    plt.imshow(images[i].numpy().astype("uint8"))

    plt.title(class\_names[labels[i]])

    plt.axis("off")

val\_batches = tf.data.experimental.cardinality(validation\_dataset)

test\_dataset = validation\_dataset.take(val\_batches // 5)

validation\_dataset = validation\_dataset.skip(val\_batches // 5)

print('Number of validation batches: %d' % tf.data.experimental.cardinality(validation\_dataset))

print('Number of test batches: %d' % tf.data.experimental.cardinality(test\_dataset))

AUTOTUNE = tf.data.AUTOTUNE

train\_dataset = train\_dataset.prefetch(buffer\_size=AUTOTUNE)

validation\_dataset = validation\_dataset.prefetch(buffer\_size=AUTOTUNE)

test\_dataset = test\_dataset.prefetch(buffer\_size=AUTOTUNE)

data\_augmentation = tf.keras.Sequential([

  tf.keras.layers.RandomFlip('horizontal'),

  tf.keras.layers.RandomRotation(0.2),

])

for image, \_ in train\_dataset.take(1):

  plt.figure(figsize=(10, 10))

  first\_image = image[0]

  for i in range(9):

    ax = plt.subplot(3, 3, i + 1)

    augmented\_image = data\_augmentation(tf.expand\_dims(first\_image, 0))

    plt.imshow(augmented\_image[0] / 255)

    plt.axis('off')

for image, \_ in train\_dataset.take(1):

  plt.figure(figsize=(10, 10))

  first\_image = image[0]

  for i in range(9):

    ax = plt.subplot(3, 3, i + 1)

    augmented\_image = data\_augmentation(tf.expand\_dims(first\_image, 0))

    plt.imshow(augmented\_image[0] / 255)

    plt.axis('off')

# Create the base model from the pre-trained model

IMG\_SHAPE = IMG\_SIZE + (3,)

base\_model = tf.keras.applications.MobileNetV2(input\_shape=IMG\_SHAPE,

                                               include\_top=False,

                                               weights='imagenet')

image\_batch, label\_batch = next(iter(train\_dataset))

feature\_batch = base\_model(image\_batch)

print(feature\_batch.shape)

(32, 5, 5, 1280)

base\_model.trainable = False

Model: "mobilenetv2\_1.00\_160"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param # Connected to

==================================================================================================

input\_1 (InputLayer) [(None, 160, 160, 3 0 []

)]

Conv1 (Conv2D) (None, 80, 80, 32) 864 ['input\_1[0][0]']

bn\_Conv1 (BatchNormalization) (None, 80, 80, 32) 128 ['Conv1[0][0]']

Conv1\_relu (ReLU) (None, 80, 80, 32) 0 ['bn\_Conv1[0][0]']

expanded\_conv\_depthwise (Depth (None, 80, 80, 32) 288 ['Conv1\_relu[0][0]']

wiseConv2D)

expanded\_conv\_depthwise\_BN (Ba (None, 80, 80, 32) 128 ['expanded\_conv\_depthwise[0][0]']

tchNormalization)

expanded\_conv\_depthwise\_relu ( (None, 80, 80, 32) 0 ['expanded\_conv\_depthwise\_BN[0][0

ReLU) ]']

expanded\_conv\_project (Conv2D) (None, 80, 80, 16) 512 ['expanded\_conv\_depthwise\_relu[0]

[0]']

...

Total params: 2,257,984

Trainable params: 0

Non-trainable params: 2,257,984

inputs = tf.keras.Input(shape=(160, 160, 3))

x = data\_augmentation(inputs)

x = preprocess\_input(x)

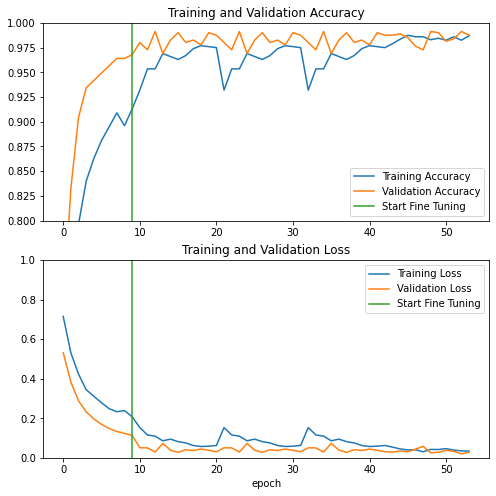
x = base\_model(x, training=False)

x = global\_average\_layer(x)

x = tf.keras.layers.Dropout(0.2)(x)

outputs = prediction\_layer(x)

model = tf.keras.Model(inputs, outputs)



# Retrieve a batch of images from the test set

image\_batch, label\_batch = test\_dataset.as\_numpy\_iterator().next()

predictions = model.predict\_on\_batch(image\_batch).flatten()

# Apply a sigmoid since our model returns logits

predictions = tf.nn.sigmoid(predictions)

predictions = tf.where(predictions < 0.5, 0, 1)

print('Predictions:\n', predictions.numpy())

print('Labels:\n', label\_batch)

plt.figure(figsize=(10, 10))

for i in range(9):

  ax = plt.subplot(3, 3, i + 1)

  plt.imshow(image\_batch[i].astype("uint8"))

  plt.title(class\_names[predictions[i]])

  plt.axis("off")

