Dogs&Cats-classifier

February 23, 2021

0.0.1 Cats and Dogs Classifier using Convolutional Neural Networks

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[2]:
```
! ls drive/MyDrive/CATS_DOGS.zip
```

```
drive/MyDrive/CATS_DOGS.zip
```

[4]:
```
# libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('classic')
```

0.0.2 Paths

[5]:
```
# archive = '../Data/CATS_DOGS.zip' # local
archive = 'drive/MyDrive/CATS_DOGS.zip' # colab
```

[6]:
```
# unzipping
import zipfile
zip_obj = zipfile.ZipFile(file=archive)
```

[7]:
```
import os
if os.path.exists('drive/MyDrive/cd_data/'):
    print(True)
else:
    print(False)
    print(os.mkdir('drive/MyDrive/cd_data/'))
    print('created!')
```

```
False
None
created!
```

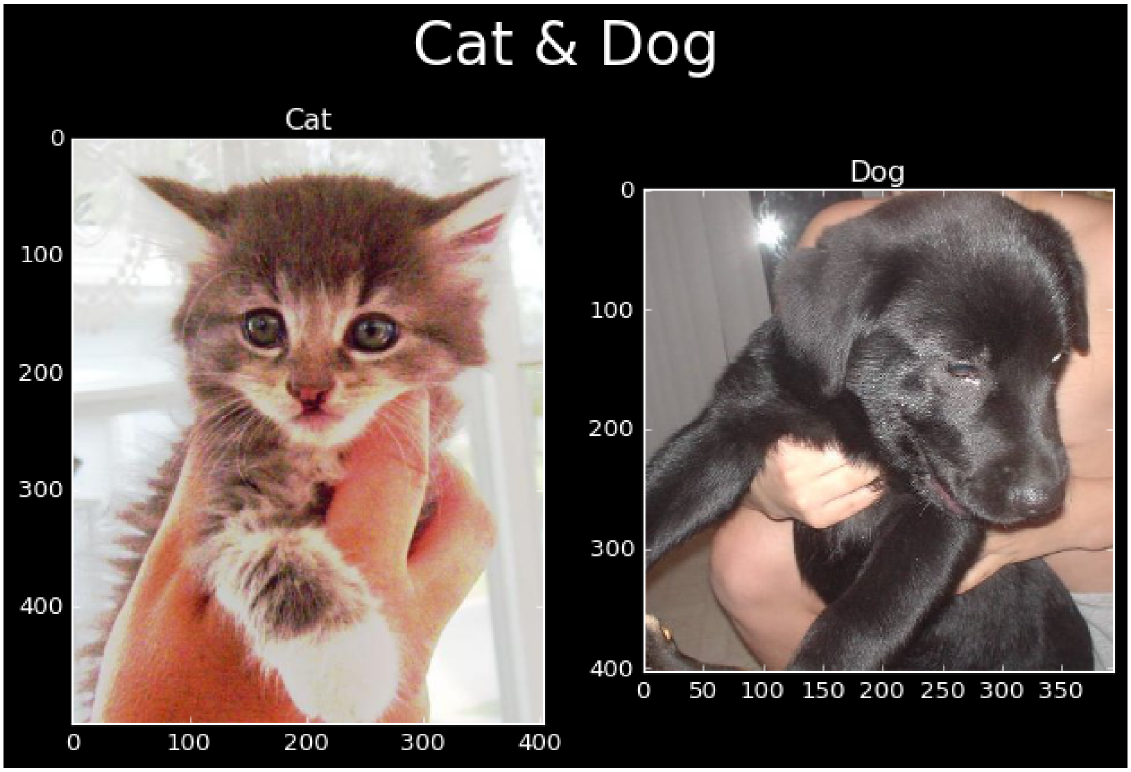
[8]:
```
# # extract to the location

# # zip_obj.extractall('../Data/') # local
zip_obj.extractall('drive/MyDrive/cd_data/') # colab
```

[9]:
```
# train and validation paths

# train_dir = '../Data/CATS_DOGS/train/'
# valid_dir = '../Data/CATS_DOGS/test/'

train_dir = 'drive/MyDrive/cd_data/CATS_DOGS/train/'
valid_dir = 'drive/MyDrive/cd_data/CATS_DOGS/test/'
```

0.0.3 Raw Data

[10]:
```
# random



# cat = '../Data/CATS_DOGS/train/CAT/100.jpg'
# dog = '../Data/CATS_DOGS/train/DOG/1050.jpg'

cat = 'drive/MyDrive/cd_data/CATS_DOGS/train/CAT/100.jpg'
dog = 'drive/MyDrive/cd_data/CATS_DOGS/train/DOG/1050.jpg'
```

[11]:
```
plt.style.use('dark_background')
from PIL import Image
img_cat = Image.open(cat)
img_dog = Image.open(dog)
fig, axes = plt.subplots(1,2)
axes[0].imshow(np.array(img_cat),label='cat')
axes[0].set_title('Cat')
axes[1].imshow(np.array(img_dog),label='dog')
axes[1].set_title('Dog')
fig.suptitle('Cat & Dog',size=30)
fig.tight_layout()
```



[12]:
```
'''
Through this random image selection and obseravtion,
it is evident that shapes are not homogeneous;
'''
```

[12]: '\nThrough this random image selection and obseravtion, \nit is evident that shapes are not homogeneous;\n'

0.0.4 Generators

[13]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

[14]:
```
help(ImageDataGenerator)
```

```
Help on class ImageDataGenerator in module
tensorflow.python.keras.preprocessing.image:

class ImageDataGenerator(keras_preprocessing.image.image_data_generator.ImageDat
aGenerator)
 |  Generate batches of tensor image data with real-time data augmentation.
 |
 |   The data will be looped over (in batches).
 |
```

```
|  Arguments:
|      featurewise_center: Boolean.
|          Set input mean to 0 over the dataset, feature-wise.
|      samplewise_center: Boolean. Set each sample mean to 0.
|      featurewise_std_normalization: Boolean.
|          Divide inputs by std of the dataset, feature-wise.
|      samplewise_std_normalization: Boolean. Divide each input by its std.
|      zca_epsilon: epsilon for ZCA whitening. Default is 1e-6.
|      zca_whitening: Boolean. Apply ZCA whitening.
|      rotation_range: Int. Degree range for random rotations.
|      width_shift_range: Float, 1-D array-like or int
|          - float: fraction of total width, if < 1, or pixels if >= 1.
|          - 1-D array-like: random elements from the array.
|          - int: integer number of pixels from interval
|              `(-width_shift_range, +width_shift_range)`
|          - With `width_shift_range=2` possible values
|              are integers `[-1, 0, +1]`,
|              same as with `width_shift_range=[-1, 0, +1]`,
|              while with `width_shift_range=1.0` possible values are floats
|              in the interval [-1.0, +1.0).
|      height_shift_range: Float, 1-D array-like or int
|          - float: fraction of total height, if < 1, or pixels if >= 1.
|          - 1-D array-like: random elements from the array.
|          - int: integer number of pixels from interval
|              `(-height_shift_range, +height_shift_range)`
|          - With `height_shift_range=2` possible values
|              are integers `[-1, 0, +1]`,
|              same as with `height_shift_range=[-1, 0, +1]`,
|              while with `height_shift_range=1.0` possible values are floats
|              in the interval [-1.0, +1.0).
|      brightness_range: Tuple or list of two floats. Range for picking
|          a brightness shift value from.
|      shear_range: Float. Shear Intensity
|          (Shear angle in counter-clockwise direction in degrees)
|      zoom_range: Float or [lower, upper]. Range for random zoom.
|          If a float, `[lower, upper] = [1-zoom_range, 1+zoom_range]`.
|      channel_shift_range: Float. Range for random channel shifts.
|      fill_mode: One of {"constant", "nearest", "reflect" or "wrap"}.
|          Default is 'nearest'.
|          Points outside the boundaries of the input are filled
|          according to the given mode:
|          - 'constant': kkkkkkkk|abcd|kkkkkkkk (cval=k)
|          - 'nearest':  aaaaaaaa|abcd|dddddddd
|          - 'reflect':  abcddcba|abcd|dcbaabcd
|          - 'wrap':  abcdabcd|abcd|abcdabcd
|      cval: Float or Int.
|          Value used for points outside the boundaries
|          when `fill_mode = "constant"`.
```

````
 |      horizontal_flip: Boolean. Randomly flip inputs horizontally.
 |      vertical_flip: Boolean. Randomly flip inputs vertically.
 |      rescale: rescaling factor. Defaults to None.
 |          If None or 0, no rescaling is applied,
 |          otherwise we multiply the data by the value provided
 |          (after applying all other transformations).
 |      preprocessing_function: function that will be applied on each input.
 |          The function will run after the image is resized and augmented.
 |          The function should take one argument:
 |          one image (Numpy tensor with rank 3),
 |          and should output a Numpy tensor with the same shape.
 |      data_format: Image data format,
 |          either "channels_first" or "channels_last".
 |          "channels_last" mode means that the images should have shape
 |          `(samples, height, width, channels)`,
 |          "channels_first" mode means that the images should have shape
 |          `(samples, channels, height, width)`.
 |          It defaults to the `image_data_format` value found in your
 |          Keras config file at `~/.keras/keras.json`.
 |          If you never set it, then it will be "channels_last".
 |      validation_split: Float. Fraction of images reserved for validation
 |          (strictly between 0 and 1).
 |      dtype: Dtype to use for the generated arrays.
 |
 |  Examples:
 |
 |  Example of using `.flow(x, y)`:
 |
 |  ```python
 |  (x_train, y_train), (x_test, y_test) = cifar10.load_data()
 |  y_train = np_utils.to_categorical(y_train, num_classes)
 |  y_test = np_utils.to_categorical(y_test, num_classes)
 |  datagen = ImageDataGenerator(
 |      featurewise_center=True,
 |      featurewise_std_normalization=True,
 |      rotation_range=20,
 |      width_shift_range=0.2,
 |      height_shift_range=0.2,
 |      horizontal_flip=True)
 |  # compute quantities required for featurewise normalization
 |  # (std, mean, and principal components if ZCA whitening is applied)
 |  datagen.fit(x_train)
 |  # fits the model on batches with real-time data augmentation:
 |  model.fit(datagen.flow(x_train, y_train, batch_size=32),
 |            steps_per_epoch=len(x_train) / 32, epochs=epochs)
 |  # here's a more "manual" example
 |  for e in range(epochs):
 |      print('Epoch', e)
````

````
|      batches = 0
|      for x_batch, y_batch in datagen.flow(x_train, y_train, batch_size=32):
|          model.fit(x_batch, y_batch)
|          batches += 1
|          if batches >= len(x_train) / 32:
|              # we need to break the loop by hand because
|              # the generator loops indefinitely
|              break
|  ```
|
|  Example of using `.flow_from_directory(directory)`:
|
|  ```python
|  train_datagen = ImageDataGenerator(
|          rescale=1./255,
|          shear_range=0.2,
|          zoom_range=0.2,
|          horizontal_flip=True)
|  test_datagen = ImageDataGenerator(rescale=1./255)
|  train_generator = train_datagen.flow_from_directory(
|          'data/train',
|          target_size=(150, 150),
|          batch_size=32,
|          class_mode='binary')
|  validation_generator = test_datagen.flow_from_directory(
|          'data/validation',
|          target_size=(150, 150),
|          batch_size=32,
|          class_mode='binary')
|  model.fit(
|          train_generator,
|          steps_per_epoch=2000,
|          epochs=50,
|          validation_data=validation_generator,
|          validation_steps=800)
|  ```
|
|  Example of transforming images and masks together.
|
|  ```python
|  # we create two instances with the same arguments
|  data_gen_args = dict(featurewise_center=True,
|                       featurewise_std_normalization=True,
|                       rotation_range=90,
|                       width_shift_range=0.1,
|                       height_shift_range=0.1,
|                       zoom_range=0.2)
|  image_datagen = ImageDataGenerator(**data_gen_args)
````

```
 |  mask_datagen = ImageDataGenerator(**data_gen_args)
 |  # Provide the same seed and keyword arguments to the fit and flow methods
 |  seed = 1
 |  image_datagen.fit(images, augment=True, seed=seed)
 |  mask_datagen.fit(masks, augment=True, seed=seed)
 |  image_generator = image_datagen.flow_from_directory(
 |      'data/images',
 |      class_mode=None,
 |      seed=seed)
 |  mask_generator = mask_datagen.flow_from_directory(
 |      'data/masks',
 |      class_mode=None,
 |      seed=seed)
 |  # combine generators into one which yields image and masks
 |  train_generator = zip(image_generator, mask_generator)
 |  model.fit(
 |      train_generator,
 |      steps_per_epoch=2000,
 |      epochs=50)
 |  ```
 |
 |  Method resolution order:
 |      ImageDataGenerator
 |      keras_preprocessing.image.image_data_generator.ImageDataGenerator
 |      builtins.object
 |
 |  Methods defined here:
 |
 |  __init__(self, featurewise_center=False, samplewise_center=False,
featurewise_std_normalization=False, samplewise_std_normalization=False,
zca_whitening=False, zca_epsilon=1e-06, rotation_range=0, width_shift_range=0.0,
height_shift_range=0.0, brightness_range=None, shear_range=0.0, zoom_range=0.0,
channel_shift_range=0.0, fill_mode='nearest', cval=0.0, horizontal_flip=False,
vertical_flip=False, rescale=None, preprocessing_function=None,
data_format=None, validation_split=0.0, dtype=None)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  flow(self, x, y=None, batch_size=32, shuffle=True, sample_weight=None,
seed=None, save_to_dir=None, save_prefix='', save_format='png', subset=None)
 |      Takes data & label arrays, generates batches of augmented data.
 |
 |      Arguments:
 |          x: Input data. Numpy array of rank 4 or a tuple. If tuple, the first
 |            element should contain the images and the second element another
numpy
 |            array or a list of numpy arrays that gets passed to the output
without
 |            any modifications. Can be used to feed the model miscellaneous
```

```
data
 |              along with the images. In case of grayscale data, the channels
axis of
 |              the image array should have value 1, in case of RGB data, it
should
 |              have value 3, and in case of RGBA data, it should have value 4.
 |            y: Labels.
 |            batch_size: Int (default: 32).
 |            shuffle: Boolean (default: True).
 |            sample_weight: Sample weights.
 |            seed: Int (default: None).
 |            save_to_dir: None or str (default: None). This allows you to
optionally
 |              specify a directory to which to save the augmented pictures being
 |              generated (useful for visualizing what you are doing).
 |            save_prefix: Str (default: `''`). Prefix to use for filenames of
saved
 |              pictures (only relevant if `save_to_dir` is set).
 |            save_format: one of "png", "jpeg"
 |                (only relevant if `save_to_dir` is set). Default: "png".
 |            subset: Subset of data (`"training"` or `"validation"`) if
 |              `validation_split` is set in `ImageDataGenerator`.
 |
 |        Returns:
 |            An `Iterator` yielding tuples of `(x, y)`
 |                where `x` is a numpy array of image data
 |                (in the case of a single image input) or a list
 |                of numpy arrays (in the case with
 |                additional inputs) and `y` is a numpy array
 |                of corresponding labels. If 'sample_weight' is not None,
 |                the yielded tuples are of the form `(x, y, sample_weight)`.
 |                If `y` is None, only the numpy array `x` is returned.
 |
 |  flow_from_dataframe(self, dataframe, directory=None, x_col='filename',
y_col='class', weight_col=None, target_size=(256, 256), color_mode='rgb',
classes=None, class_mode='categorical', batch_size=32, shuffle=True, seed=None,
save_to_dir=None, save_prefix='', save_format='png', subset=None,
interpolation='nearest', validate_filenames=True, **kwargs)
 |        Takes the dataframe and the path to a directory + generates batches.
 |
 |         The generated batches contain augmented/normalized data.
 |
 |        **A simple tutorial can be found **[here](
 |                                    http://bit.ly/keras_flow_from_dataframe).
 |
 |        Arguments:
 |            dataframe: Pandas dataframe containing the filepaths relative to
 |              `directory` (or absolute paths if `directory` is None) of the
```

```
images
 |              in a string column. It should include other column/s
 |                depending on the `class_mode`: - if `class_mode` is
`"categorical"`
 |                  (default value) it must include the `y_col` column with the
 |                  class/es of each image. Values in column can be
string/list/tuple
 |                  if a single class or list/tuple if multiple classes. - if
 |                  `class_mode` is `"binary"` or `"sparse"` it must include the
given
 |                  `y_col` column with class values as strings. - if `class_mode`
is
 |                  `"raw"` or `"multi_output"` it should contain the columns
 |                  specified in `y_col`. - if `class_mode` is `"input"` or `None`
no
 |                  extra column is needed.
 |            directory: string, path to the directory to read images from. If
`None`,
 |              data in `x_col` column should be absolute paths.
 |            x_col: string, column in `dataframe` that contains the filenames (or
 |              absolute paths if `directory` is `None`).
 |            y_col: string or list, column/s in `dataframe` that has the target
data.
 |            weight_col: string, column in `dataframe` that contains the sample
 |                weights. Default: `None`.
 |            target_size: tuple of integers `(height, width)`, default: `(256,
256)`.
 |              The dimensions to which all images found will be resized.
 |            color_mode: one of "grayscale", "rgb", "rgba". Default: "rgb".
Whether
 |              the images will be converted to have 1 or 3 color channels.
 |            classes: optional list of classes (e.g. `['dogs', 'cats']`). Default
is
 |              None. If not provided, the list of classes will be automatically
 |              inferred from the `y_col`, which will map to the label indices,
will
 |              be alphanumeric). The dictionary containing the mapping from class
 |              names to class indices can be obtained via the attribute
 |              `class_indices`.
 |            class_mode: one of "binary", "categorical", "input", "multi_output",
 |                "raw", sparse" or None. Default: "categorical".
 |                Mode for yielding the targets:
 |                - `"binary"`: 1D numpy array of binary labels,
 |                - `"categorical"`: 2D numpy array of one-hot encoded labels.
 |                  Supports multi-label output.
 |                - `"input"`: images identical to input images (mainly used to
work
 |                  with autoencoders),
```

```
 |              - `"multi_output"`: list with the values of the different
columns,
 |              - `"raw"`: numpy array of values in `y_col` column(s),
 |              - `"sparse"`: 1D numpy array of integer labels, - `None`, no
targets
 |                are returned (the generator will only yield batches of image
data,
 |                which is useful to use in `model.predict()`).
 |          batch_size: size of the batches of data (default: 32).
 |          shuffle: whether to shuffle the data (default: True)
 |          seed: optional random seed for shuffling and transformations.
 |          save_to_dir: None or str (default: None). This allows you to
optionally
 |            specify a directory to which to save the augmented pictures being
 |            generated (useful for visualizing what you are doing).
 |          save_prefix: str. Prefix to use for filenames of saved pictures
(only
 |            relevant if `save_to_dir` is set).
 |          save_format: one of "png", "jpeg"
 |              (only relevant if `save_to_dir` is set). Default: "png".
 |          subset: Subset of data (`"training"` or `"validation"`) if
 |            `validation_split` is set in `ImageDataGenerator`.
 |          interpolation: Interpolation method used to resample the image if
the
 |            target size is different from that of the loaded image. Supported
 |            methods are `"nearest"`, `"bilinear"`, and `"bicubic"`. If PIL
version
 |            1.1.3 or newer is installed, `"lanczos"` is also supported. If PIL
 |            version 3.4.0 or newer is installed, `"box"` and `"hamming"` are
also
 |            supported. By default, `"nearest"` is used.
 |          validate_filenames: Boolean, whether to validate image filenames in
 |            `x_col`. If `True`, invalid images will be ignored. Disabling this
 |            option can lead to speed-up in the execution of this function.
 |            Defaults to `True`.
 |          **kwargs: legacy arguments for raising deprecation warnings.
 |
 |      Returns:
 |          A `DataFrameIterator` yielding tuples of `(x, y)`
 |          where `x` is a numpy array containing a batch
 |          of images with shape `(batch_size, *target_size, channels)`
 |          and `y` is a numpy array of corresponding labels.
 |
 |  flow_from_directory(self, directory, target_size=(256, 256),
color_mode='rgb', classes=None, class_mode='categorical', batch_size=32,
shuffle=True, seed=None, save_to_dir=None, save_prefix='', save_format='png',
follow_links=False, subset=None, interpolation='nearest')
 |      Takes the path to a directory & generates batches of augmented data.
```

```
 |
 |      Arguments:
 |          directory: string, path to the target directory. It should contain
one
 |            subdirectory per class. Any PNG, JPG, BMP, PPM or TIF images
inside
 |            each of the subdirectories directory tree will be included in the
 |            generator. See [this script](
 |
https://gist.github.com/fchollet/0830affa1f7f19fd47b06d4cf89ed44d)
 |                for more details.
 |          target_size: Tuple of integers `(height, width)`, defaults to `(256,
 |            256)`. The dimensions to which all images found will be resized.
 |          color_mode: One of "grayscale", "rgb", "rgba". Default: "rgb".
Whether
 |            the images will be converted to have 1, 3, or 4 channels.
 |          classes: Optional list of class subdirectories
 |              (e.g. `['dogs', 'cats']`). Default: None. If not provided, the
list
 |                of classes will be automatically inferred from the
subdirectory
 |                names/structure under `directory`, where each subdirectory
will be
 |                treated as a different class (and the order of the classes,
which
 |                will map to the label indices, will be alphanumeric). The
 |                dictionary containing the mapping from class names to class
 |                indices can be obtained via the attribute `class_indices`.
 |          class_mode: One of "categorical", "binary", "sparse",
 |              "input", or None. Default: "categorical".
 |              Determines the type of label arrays that are returned: -
 |                "categorical" will be 2D one-hot encoded labels, - "binary"
will
 |                be 1D binary labels, "sparse" will be 1D integer labels, -
"input"
 |                will be images identical to input images (mainly used to work
with
 |                autoencoders). - If None, no labels are returned (the
generator
 |                will only yield batches of image data, which is useful to use
with
 |                `model.predict()`). Please note that in case of
 |                class_mode None, the data still needs to reside in a
subdirectory
 |                of `directory` for it to work correctly.
 |          batch_size: Size of the batches of data (default: 32).
 |          shuffle: Whether to shuffle the data (default: True) If set to
False,
```

```
 |           sorts the data in alphanumeric order.
 |         seed: Optional random seed for shuffling and transformations.
 |         save_to_dir: None or str (default: None). This allows you to
optionally
 |           specify a directory to which to save the augmented pictures being
 |           generated (useful for visualizing what you are doing).
 |         save_prefix: Str. Prefix to use for filenames of saved pictures
(only
 |           relevant if `save_to_dir` is set).
 |         save_format: One of "png", "jpeg"
 |             (only relevant if `save_to_dir` is set). Default: "png".
 |         follow_links: Whether to follow symlinks inside
 |             class subdirectories (default: False).
 |         subset: Subset of data (`"training"` or `"validation"`) if
 |           `validation_split` is set in `ImageDataGenerator`.
 |         interpolation: Interpolation method used to resample the image if
the
 |           target size is different from that of the loaded image. Supported
 |           methods are `"nearest"`, `"bilinear"`, and `"bicubic"`. If PIL
version
 |           1.1.3 or newer is installed, `"lanczos"` is also supported. If PIL
 |           version 3.4.0 or newer is installed, `"box"` and `"hamming"` are
also
 |           supported. By default, `"nearest"` is used.
 |
 |     Returns:
 |         A `DirectoryIterator` yielding tuples of `(x, y)`
 |             where `x` is a numpy array containing a batch
 |             of images with shape `(batch_size, *target_size, channels)`
 |             and `y` is a numpy array of corresponding labels.
 |
 |  ----------------------------------------------------------------------
 |  Methods inherited from
keras_preprocessing.image.image_data_generator.ImageDataGenerator:
 |
 |  apply_transform(self, x, transform_parameters)
 |      Applies a transformation to an image according to given parameters.
 |
 |      # Arguments
 |          x: 3D tensor, single image.
 |          transform_parameters: Dictionary with string - parameter pairs
 |              describing the transformation.
 |              Currently, the following parameters
 |              from the dictionary are used:
 |              - `'theta'`: Float. Rotation angle in degrees.
 |              - `'tx'`: Float. Shift in the x direction.
 |              - `'ty'`: Float. Shift in the y direction.
 |              - `'shear'`: Float. Shear angle in degrees.
```

```
 |              - `'zx'`: Float. Zoom in the x direction.
 |              - `'zy'`: Float. Zoom in the y direction.
 |              - `'flip_horizontal'`: Boolean. Horizontal flip.
 |              - `'flip_vertical'`: Boolean. Vertical flip.
 |              - `'channel_shift_intensity'`: Float. Channel shift intensity.
 |              - `'brightness'`: Float. Brightness shift intensity.
 |
 |      # Returns
 |          A transformed version of the input (same shape).
 |
 |  fit(self, x, augment=False, rounds=1, seed=None)
 |      Fits the data generator to some sample data.
 |
 |      This computes the internal data stats related to the
 |      data-dependent transformations, based on an array of sample data.
 |
 |      Only required if `featurewise_center` or
 |      `featurewise_std_normalization` or `zca_whitening` are set to True.
 |
 |      When `rescale` is set to a value, rescaling is applied to
 |      sample data before computing the internal data stats.
 |
 |      # Arguments
 |          x: Sample data. Should have rank 4.
 |           In case of grayscale data,
 |           the channels axis should have value 1, in case
 |           of RGB data, it should have value 3, and in case
 |           of RGBA data, it should have value 4.
 |          augment: Boolean (default: False).
 |              Whether to fit on randomly augmented samples.
 |          rounds: Int (default: 1).
 |              If using data augmentation (`augment=True`),
 |              this is how many augmentation passes over the data to use.
 |          seed: Int (default: None). Random seed.
 |
 |  get_random_transform(self, img_shape, seed=None)
 |      Generates random parameters for a transformation.
 |
 |      # Arguments
 |          seed: Random seed.
 |          img_shape: Tuple of integers.
 |              Shape of the image that is transformed.
 |
 |      # Returns
 |          A dictionary containing randomly chosen parameters describing the
 |          transformation.
 |
 |  random_transform(self, x, seed=None)
```

```
|      Applies a random transformation to an image.
|
|      # Arguments
|          x: 3D tensor, single image.
|          seed: Random seed.
|
|      # Returns
|          A randomly transformed version of the input (same shape).
|
|  standardize(self, x)
|      Applies the normalization configuration in-place to a batch of inputs.
|
|      `x` is changed in-place since the function is mainly used internally
|      to standardize images and feed them to your network. If a copy of `x`
|      would be created instead it would have a significant performance cost.
|      If you want to apply this method without changing the input in-place
|      you can call the method creating a copy before:
|
|      standardize(np.copy(x))
|
|      # Arguments
|          x: Batch of inputs to be normalized.
|
|      # Returns
|          The inputs, normalized.
|
|  ----------------------------------------------------------------------
|  Data descriptors inherited from
keras_preprocessing.image.image_data_generator.ImageDataGenerator:
|
|  __dict__
|      dictionary for instance variables (if defined)
|
|  __weakref__
|      list of weak references to the object (if defined)
```

[15]:

```
train_gen = ImageDataGenerator(rescale=1/255.)
val_gen = ImageDataGenerator(rescale=1/255.)
```

[16]:

```
# both cats and dogs combined
train_generator = train_gen.
 ↪flow_from_directory(train_dir,target_size=(128,128),batch_size=64,shuffle=True,class_mode='
validation_generator = val_gen.
 ↪flow_from_directory(valid_dir,target_size=(128,128),batch_size=64,shuffle=False,class_mode=
```

```
Found 18743 images belonging to 2 classes.
Found 6251 images belonging to 2 classes.
```

Generators does almost everything that we need in a robust way by just parameterizing what we need. in this case because the image shapes where Heterogeneous in nature, we took a target size of 128,128 to transform every image into that size.

[17]:
```
total_data = train_generator.samples + validation_generator.samples
print('Classes Include : ',train_generator.class_indices)
print('Train Samples : ', train_generator.samples)
print('Validation/Test Samples : ', validation_generator.samples)
print('Total Samples present (cats+dogs) : ', total_data)
print('Transformed Image shape : ',train_generator.image_shape)
print('Color Channels : (RGB)', train_generator.image_shape[2])
```

```
Classes Include :  {'CAT': 0, 'DOG': 1}
Train Samples :  18743
Validation/Test Samples :  6251
Total Samples present (cats+dogs) :  24994
Transformed Image shape :  (128, 128, 3)
Color Channels : (RGB) 3
```

0.0.5 Ordinary Convolutional Neural Network Method

[18]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Flatten,␣
 ↪BatchNormalization, Activation, Dropout
```

[34]:
```
input_shape = train_generator.image_shape

net = Sequential()

#convnet

net.add(Conv2D(32, kernel_size=3,input_shape = train_generator.image_shape,␣
 ↪activation='relu'))
net.add(MaxPooling2D(2,2))

net.add(Conv2D(64,(3,3), activation='relu'))
net.add(MaxPooling2D(2,2))

net.add(Conv2D(128, (3,3), activation='relu'))
net.add(MaxPooling2D(2,2))

net.add(Flatten())

net.add(Dense(512, activation='relu'))
net.add(Dense(1,activation='sigmoid'))
```

[35]:
```
net.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 126, 126, 32)      896
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 63, 63, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 61, 61, 64)        18496
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 30, 30, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 28, 28, 128)       73856
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 14, 14, 128)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               12845568
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 513
=================================================================
Total params: 12,939,329
Trainable params: 12,939,329
Non-trainable params: 0
_________________________________________________________________
```

[36]:
```
from tensorflow.keras.callbacks import EarlyStopping
adam = tf.keras.optimizers.Adam(lr=0.001)
```

[37]:
```
net.compile(optimizer=adam, loss='binary_crossentropy', metrics=['accuracy'])
```

[38]:
```
early_stop = EarlyStopping(patience=3)
```

[39]:
```
import time

start = time.perf_counter()

perf = net.fit_generator(train_generator, epochs=15,␣
 ↪callbacks=[early_stop],validation_data=validation_generator)

elapsed = time.perf_counter()- start

print('Elapsed {}'.format(elapsed/60))
```

```
/usr/local/lib/python3.6/dist-
packages/tensorflow/python/keras/engine/training.py:1844: UserWarning:
`Model.fit_generator` is deprecated and will be removed in a future version.
Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '

Epoch 1/15
  2/293 […] - ETA: 1:07 - loss: 1.2549 - accuracy:
0.4570

/usr/local/lib/python3.6/dist-packages/PIL/TiffImagePlugin.py:770: UserWarning:
Possibly corrupt EXIF data.  Expecting to read 32 bytes but only got 0. Skipping
tag 270
  " Skipping tag %s" % (size, len(data), tag)
/usr/local/lib/python3.6/dist-packages/PIL/TiffImagePlugin.py:770: UserWarning:
Possibly corrupt EXIF data.  Expecting to read 5 bytes but only got 0. Skipping
tag 271
  " Skipping tag %s" % (size, len(data), tag)
/usr/local/lib/python3.6/dist-packages/PIL/TiffImagePlugin.py:770: UserWarning:
Possibly corrupt EXIF data.  Expecting to read 8 bytes but only got 0. Skipping
tag 272
  " Skipping tag %s" % (size, len(data), tag)
/usr/local/lib/python3.6/dist-packages/PIL/TiffImagePlugin.py:770: UserWarning:
Possibly corrupt EXIF data.  Expecting to read 8 bytes but only got 0. Skipping
tag 282
  " Skipping tag %s" % (size, len(data), tag)
/usr/local/lib/python3.6/dist-packages/PIL/TiffImagePlugin.py:770: UserWarning:
Possibly corrupt EXIF data.  Expecting to read 8 bytes but only got 0. Skipping
tag 283
  " Skipping tag %s" % (size, len(data), tag)
/usr/local/lib/python3.6/dist-packages/PIL/TiffImagePlugin.py:770: UserWarning:
Possibly corrupt EXIF data.  Expecting to read 20 bytes but only got 0. Skipping
tag 306
  " Skipping tag %s" % (size, len(data), tag)
/usr/local/lib/python3.6/dist-packages/PIL/TiffImagePlugin.py:770: UserWarning:
Possibly corrupt EXIF data.  Expecting to read 48 bytes but only got 0. Skipping
tag 532
  " Skipping tag %s" % (size, len(data), tag)
/usr/local/lib/python3.6/dist-packages/PIL/TiffImagePlugin.py:788: UserWarning:
Corrupt EXIF data.  Expecting to read 2 bytes but only got 0.
  warnings.warn(str(msg))

293/293 [==============================] - 92s 312ms/step - loss: 0.6959 -
accuracy: 0.5951 - val_loss: 0.5447 - val_accuracy: 0.7207
Epoch 2/15
293/293 [==============================] - 91s 310ms/step - loss: 0.5172 -
accuracy: 0.7472 - val_loss: 0.4482 - val_accuracy: 0.7906
Epoch 3/15
293/293 [==============================] - 92s 313ms/step - loss: 0.4282 -
```

```
accuracy: 0.8010 - val_loss: 0.4345 - val_accuracy: 0.7903
Epoch 4/15
293/293 [==============================] - 91s 310ms/step - loss: 0.3586 -
accuracy: 0.8409 - val_loss: 0.3913 - val_accuracy: 0.8219
Epoch 5/15
293/293 [==============================] - 92s 313ms/step - loss: 0.3031 -
accuracy: 0.8719 - val_loss: 0.3582 - val_accuracy: 0.8416
Epoch 6/15
293/293 [==============================] - 92s 314ms/step - loss: 0.2391 -
accuracy: 0.9005 - val_loss: 0.4140 - val_accuracy: 0.8322
Epoch 7/15
293/293 [==============================] - 91s 310ms/step - loss: 0.1694 -
accuracy: 0.9329 - val_loss: 0.3971 - val_accuracy: 0.8440
Epoch 8/15
293/293 [==============================] - 91s 311ms/step - loss: 0.1156 -
accuracy: 0.9560 - val_loss: 0.4748 - val_accuracy: 0.8375
Elapsed 12.185325518533329
```

[40]:
```
loss, acc = net.evaluate_generator(validation_generator)
```

```
/usr/local/lib/python3.6/dist-
packages/tensorflow/python/keras/engine/training.py:1877: UserWarning:
`Model.evaluate_generator` is deprecated and will be removed in a future
version. Please use `Model.evaluate`, which supports generators.
  warnings.warn('`Model.evaluate_generator` is deprecated and '
```

[44]:
```
# validation Accuracy
print('Validation Accuracy - {:.2f}%'.format(acc*100))
print('validation loss - {:.2f}'.format(loss))
```

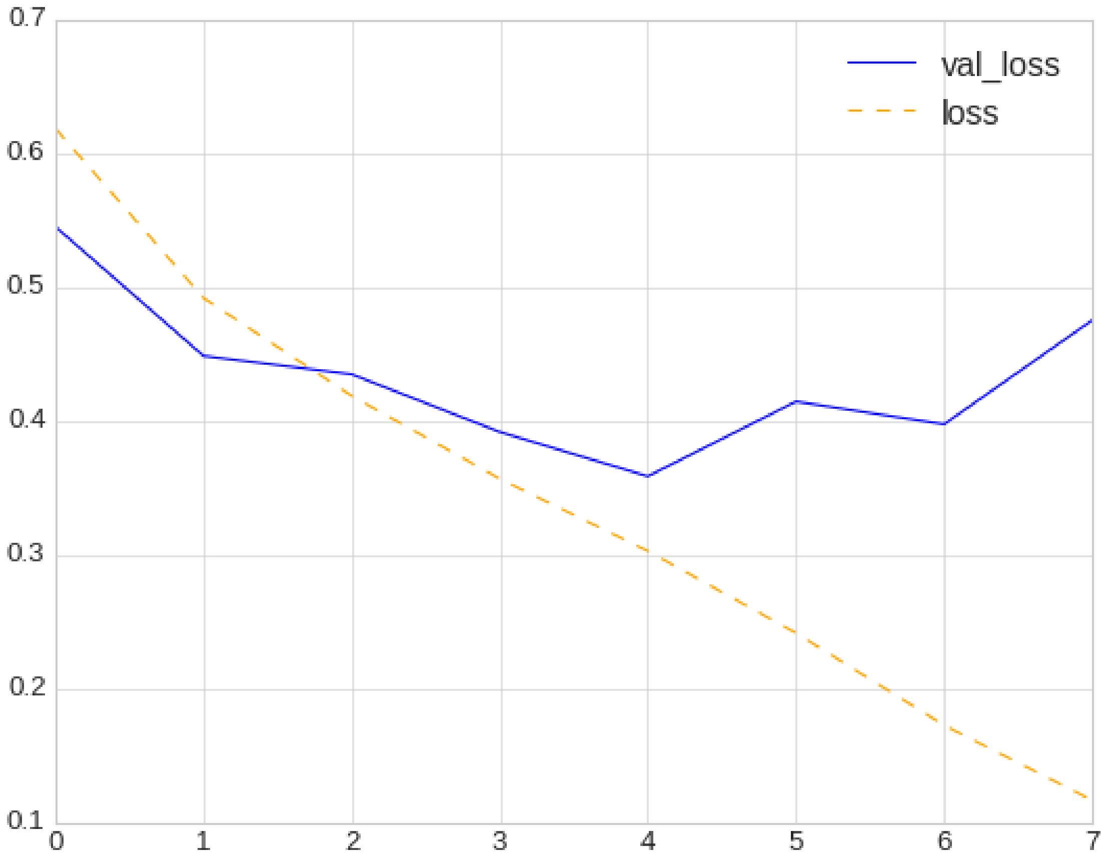
```
Validation Accuracy - 83.75%
validation loss - 0.47
```

[43]:
```
history = pd.DataFrame(perf.history)
```

[43]:
```

```

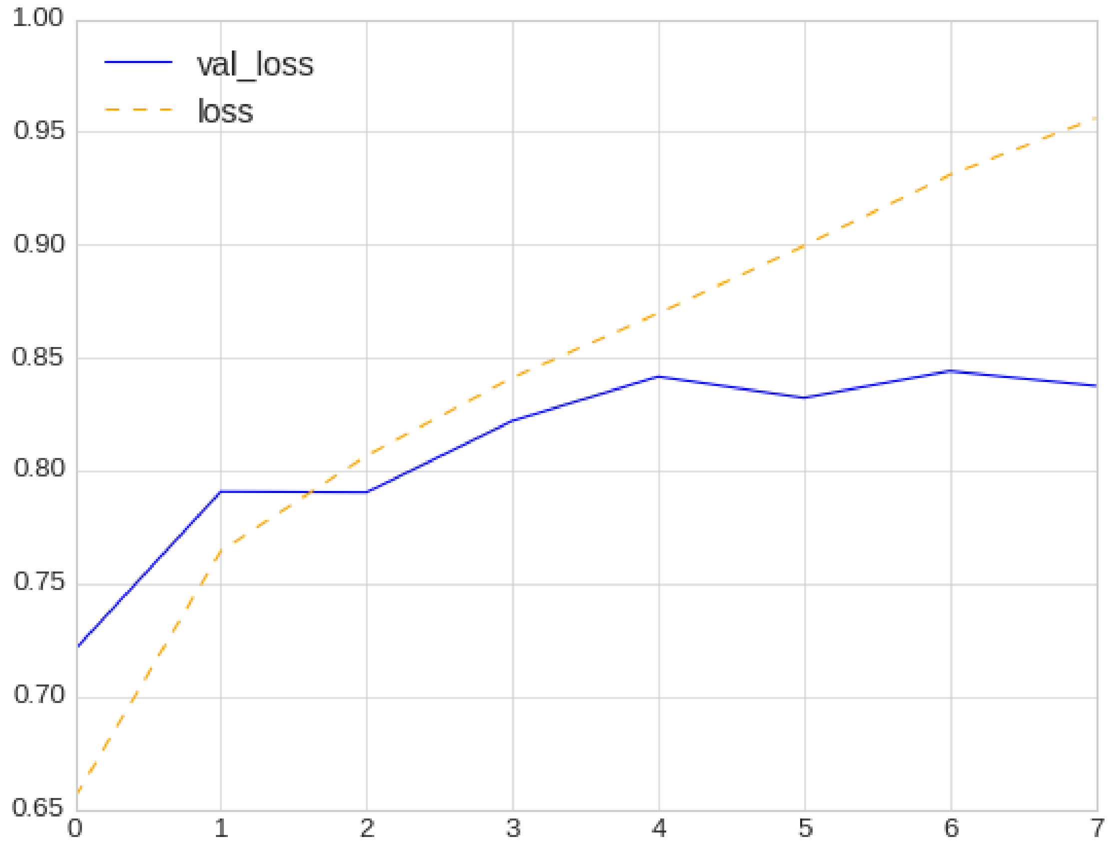
[45]:
```
plt.style.use('seaborn-whitegrid')
plt.plot(history['val_loss'], ls='-',color='blue',label='val_loss')
plt.plot(history['loss'],ls='--',color='orange',label='loss' )
plt.legend()
```

[45]: <matplotlib.legend.Legend at 0x7f234a9e0198>



[50]:
```
plt.style.use('seaborn-whitegrid')
plt.plot(history['val_accuracy'], ls='-',color='blue',label='val_loss')
plt.plot(history['accuracy'],ls='--',color='orange',label='loss' )
plt.legend(loc='upper left')
```

[50]: `<matplotlib.legend.Legend at 0x7f234a6bb4a8>`



[ ]:
```
# Classwise Evaluation
```

[52]:
```
pred_probs = net.predict_generator(validation_generator)
preds = pred_probs > 0.5
```

```
/usr/local/lib/python3.6/dist-
packages/tensorflow/python/keras/engine/training.py:1905: UserWarning:
`Model.predict_generator` is deprecated and will be removed in a future version.
Please use `Model.predict`, which supports generators.
  warnings.warn('`Model.predict_generator` is deprecated and '
```

[53]:
```
preds
```

[53]:
```
array([[ True],
       [False],
       [False],
       …,
       [ True],
       [ True],
       [ True]])
```

[54]:
```python
from sklearn.metrics import classification_report, confusion_matrix
```

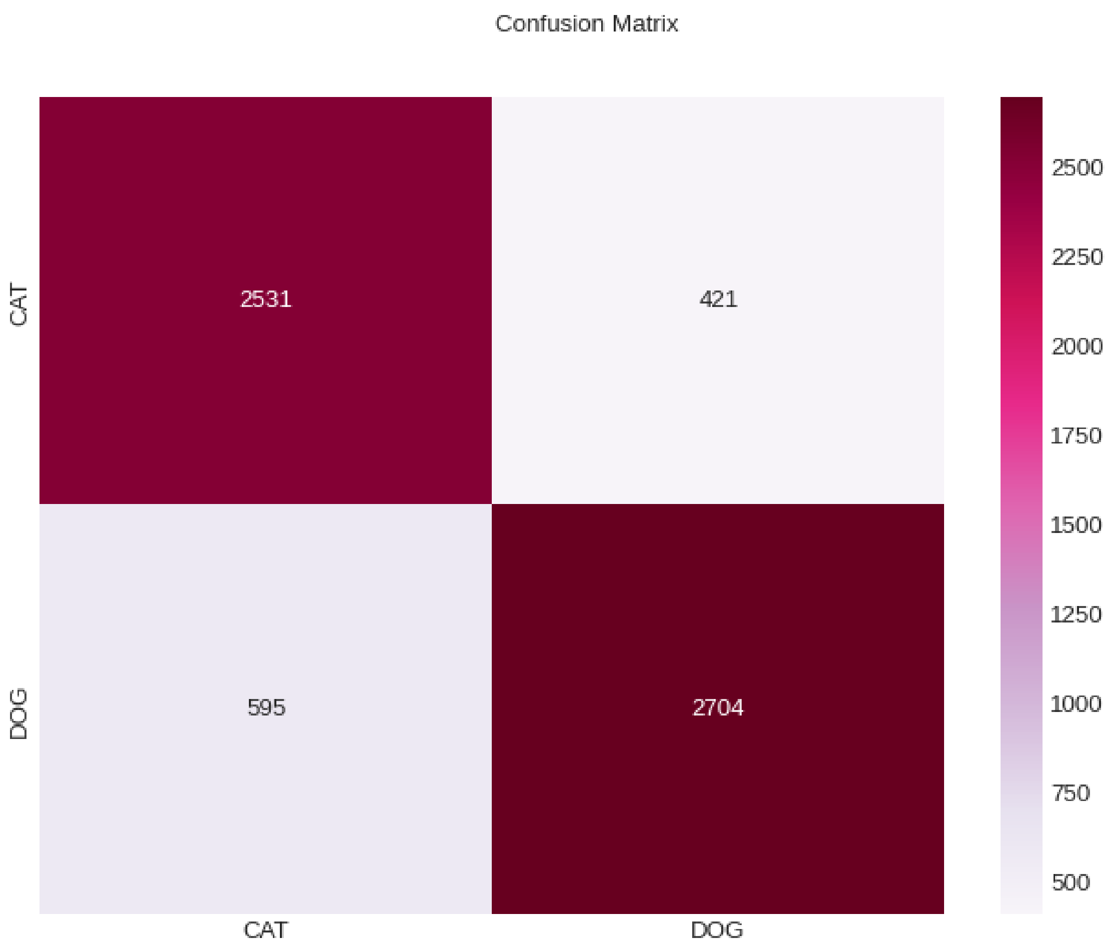
[55]:
```python
print(classification_report(preds, validation_generator.classes))
```

```
              precision    recall  f1-score   support

       False       0.81      0.86      0.83      2952
        True       0.87      0.82      0.84      3299

    accuracy                           0.84      6251
   macro avg       0.84      0.84      0.84      6251
weighted avg       0.84      0.84      0.84      6251
```

[ ]:
```python

```

[56]:
```python
plt.figure(figsize=(10,7))
plt.suptitle('Confusion Matrix')
confmat = pd.DataFrame(confusion_matrix(preds, validation_generator.classes),␣
 ↪columns=validation_generator.class_indices.keys())
confmat.index = validation_generator.class_indices.keys()
sns.heatmap(confmat, annot=True, fmt='d', cmap='PuRd')
```

[56]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f23ac2deba8>
```



The Classic case of Misclassifying dogs as cats and cats as dogs. the latter one is mostly happening i.cats as dogs classification

[94]:
```
net.save('CovnNet(cats&dogs).h5')
```

0.0.6 Inference

While the training Set Accuracy steadily increases till it reaches 90 the validation accuracy is stall at 80's, it is clearly the case of overfitting or high variance scenario.

[83]:
```
from tensorflow.keras.applications import MobileNetV2
```

[84]:
```
train_generator.image_shape
```

[84]:
```
(128, 128, 3)
```

[85]:
```
conv_base = MobileNetV2(input_shape=train_generator.image_shape,␣
 ↪include_top=False, weights='imagenet')
```

[86]:
```
conv_base.trainable
```

[86]:
```
True
```

[87]:
```
# freezing layers
conv_base.trainable= False
```

[88]:
```
conv_base.summary()
```

```
Model: "mobilenetv2_1.00_128"

________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
==================
input_2 (InputLayer)            [(None, 128, 128, 3) 0

________________________________________________________________________________
__________________
Conv1 (Conv2D)                  (None, 64, 64, 32)   864         input_2[0][0]

________________________________________________________________________________
__________________
bn_Conv1 (BatchNormalization)   (None, 64, 64, 32)   128         Conv1[0][0]

________________________________________________________________________________
__________________
Conv1_relu (ReLU)               (None, 64, 64, 32)   0           bn_Conv1[0][0]

________________________________________________________________________________
__________________
expanded_conv_depthwise (Depthw (None, 64, 64, 32)   288
Conv1_relu[0][0]

________________________________________________________________________________
__________________
expanded_conv_depthwise_BN (Bat (None, 64, 64, 32)   128
expanded_conv_depthwise[0][0]

________________________________________________________________________________
__________________
expanded_conv_depthwise_relu (R (None, 64, 64, 32)   0
expanded_conv_depthwise_BN[0][0]

________________________________________________________________________________
__________________
expanded_conv_project (Conv2D)  (None, 64, 64, 16)   512
expanded_conv_depthwise_relu[0][0

________________________________________________________________________________
__________________
expanded_conv_project_BN (Batch (None, 64, 64, 16)   64
expanded_conv_project[0][0]

________________________________________________________________________________
__________________
```

```
block_1_expand (Conv2D)         (None, 64, 64, 96)   1536
expanded_conv_project_BN[0][0]
__________________________________________________________________________________________________
block_1_expand_BN (BatchNormali (None, 64, 64, 96)   384
block_1_expand[0][0]
__________________________________________________________________________________________________
block_1_expand_relu (ReLU)      (None, 64, 64, 96)   0
block_1_expand_BN[0][0]
__________________________________________________________________________________________________
block_1_pad (ZeroPadding2D)     (None, 65, 65, 96)   0
block_1_expand_relu[0][0]
__________________________________________________________________________________________________
block_1_depthwise (DepthwiseCon (None, 32, 32, 96)   864
block_1_pad[0][0]
__________________________________________________________________________________________________
block_1_depthwise_BN (BatchNorm (None, 32, 32, 96)   384
block_1_depthwise[0][0]
__________________________________________________________________________________________________
block_1_depthwise_relu (ReLU)   (None, 32, 32, 96)   0
block_1_depthwise_BN[0][0]
__________________________________________________________________________________________________
block_1_project (Conv2D)        (None, 32, 32, 24)   2304
block_1_depthwise_relu[0][0]
__________________________________________________________________________________________________
block_1_project_BN (BatchNormal (None, 32, 32, 24)   96
block_1_project[0][0]
__________________________________________________________________________________________________
block_2_expand (Conv2D)         (None, 32, 32, 144)  3456
block_1_project_BN[0][0]
__________________________________________________________________________________________________
block_2_expand_BN (BatchNormali (None, 32, 32, 144)  576
block_2_expand[0][0]
__________________________________________________________________________________________________
block_2_expand_relu (ReLU)      (None, 32, 32, 144)  0
block_2_expand_BN[0][0]
__________________________________________________________________________________________________
```

```
block_2_depthwise (DepthwiseCon (None, 32, 32, 144)  1296
block_2_expand_relu[0][0]
________________________________________________________________________________
__________________
block_2_depthwise_BN (BatchNorm (None, 32, 32, 144)  576
block_2_depthwise[0][0]
________________________________________________________________________________
__________________
block_2_depthwise_relu (ReLU)   (None, 32, 32, 144)  0
block_2_depthwise_BN[0][0]
________________________________________________________________________________
__________________
block_2_project (Conv2D)        (None, 32, 32, 24)   3456
block_2_depthwise_relu[0][0]
________________________________________________________________________________
__________________
block_2_project_BN (BatchNormal (None, 32, 32, 24)   96
block_2_project[0][0]
________________________________________________________________________________
__________________
block_2_add (Add)               (None, 32, 32, 24)   0
block_1_project_BN[0][0]
block_2_project_BN[0][0]
________________________________________________________________________________
__________________
block_3_expand (Conv2D)         (None, 32, 32, 144)  3456
block_2_add[0][0]
________________________________________________________________________________
__________________
block_3_expand_BN (BatchNormali (None, 32, 32, 144)  576
block_3_expand[0][0]
________________________________________________________________________________
__________________
block_3_expand_relu (ReLU)      (None, 32, 32, 144)  0
block_3_expand_BN[0][0]
________________________________________________________________________________
__________________
block_3_pad (ZeroPadding2D)     (None, 33, 33, 144)  0
block_3_expand_relu[0][0]
________________________________________________________________________________
__________________
block_3_depthwise (DepthwiseCon (None, 16, 16, 144)  1296
block_3_pad[0][0]
________________________________________________________________________________
__________________
block_3_depthwise_BN (BatchNorm (None, 16, 16, 144)  576
block_3_depthwise[0][0]
________________________________________________________________________________
```

```
__________________
block_3_depthwise_relu (ReLU)   (None, 16, 16, 144)  0
block_3_depthwise_BN[0][0]
________________________________________________________________________________
__________________
block_3_project (Conv2D)        (None, 16, 16, 32)   4608
block_3_depthwise_relu[0][0]
________________________________________________________________________________
__________________
block_3_project_BN (BatchNormal (None, 16, 16, 32)   128
block_3_project[0][0]
________________________________________________________________________________
__________________
block_4_expand (Conv2D)         (None, 16, 16, 192)  6144
block_3_project_BN[0][0]
________________________________________________________________________________
__________________
block_4_expand_BN (BatchNormali (None, 16, 16, 192)  768
block_4_expand[0][0]
________________________________________________________________________________
__________________
block_4_expand_relu (ReLU)      (None, 16, 16, 192)  0
block_4_expand_BN[0][0]
________________________________________________________________________________
__________________
block_4_depthwise (DepthwiseCon (None, 16, 16, 192)  1728
block_4_expand_relu[0][0]
________________________________________________________________________________
__________________
block_4_depthwise_BN (BatchNorm (None, 16, 16, 192)  768
block_4_depthwise[0][0]
________________________________________________________________________________
__________________
block_4_depthwise_relu (ReLU)   (None, 16, 16, 192)  0
block_4_depthwise_BN[0][0]
________________________________________________________________________________
__________________
block_4_project (Conv2D)        (None, 16, 16, 32)   6144
block_4_depthwise_relu[0][0]
________________________________________________________________________________
__________________
block_4_project_BN (BatchNormal (None, 16, 16, 32)   128
block_4_project[0][0]
________________________________________________________________________________
__________________
block_4_add (Add)               (None, 16, 16, 32)   0
block_3_project_BN[0][0]
block_4_project_BN[0][0]
```

```
__________________________________________________________________________________
__________________
block_5_expand (Conv2D)         (None, 16, 16, 192)  6144
block_4_add[0][0]
__________________________________________________________________________________
__________________
block_5_expand_BN (BatchNormali (None, 16, 16, 192)  768
block_5_expand[0][0]
__________________________________________________________________________________
__________________
block_5_expand_relu (ReLU)      (None, 16, 16, 192)  0
block_5_expand_BN[0][0]
__________________________________________________________________________________
__________________
block_5_depthwise (DepthwiseCon (None, 16, 16, 192)  1728
block_5_expand_relu[0][0]
__________________________________________________________________________________
__________________
block_5_depthwise_BN (BatchNorm (None, 16, 16, 192)  768
block_5_depthwise[0][0]
__________________________________________________________________________________
__________________
block_5_depthwise_relu (ReLU)   (None, 16, 16, 192)  0
block_5_depthwise_BN[0][0]
__________________________________________________________________________________
__________________
block_5_project (Conv2D)        (None, 16, 16, 32)   6144
block_5_depthwise_relu[0][0]
__________________________________________________________________________________
__________________
block_5_project_BN (BatchNormal (None, 16, 16, 32)   128
block_5_project[0][0]
__________________________________________________________________________________
__________________
block_5_add (Add)               (None, 16, 16, 32)   0
block_4_add[0][0]
block_5_project_BN[0][0]
__________________________________________________________________________________
__________________
block_6_expand (Conv2D)         (None, 16, 16, 192)  6144
block_5_add[0][0]
__________________________________________________________________________________
__________________
block_6_expand_BN (BatchNormali (None, 16, 16, 192)  768
block_6_expand[0][0]
__________________________________________________________________________________
__________________
block_6_expand_relu (ReLU)      (None, 16, 16, 192)  0
```

```
block_6_expand_BN[0][0]
__________________________________________________________________________________________________
block_6_pad (ZeroPadding2D)     (None, 17, 17, 192)  0
block_6_expand_relu[0][0]
__________________________________________________________________________________________________
block_6_depthwise (DepthwiseCon (None, 8, 8, 192)    1728
block_6_pad[0][0]
__________________________________________________________________________________________________
block_6_depthwise_BN (BatchNorm (None, 8, 8, 192)    768
block_6_depthwise[0][0]
__________________________________________________________________________________________________
block_6_depthwise_relu (ReLU)   (None, 8, 8, 192)    0
block_6_depthwise_BN[0][0]
__________________________________________________________________________________________________
block_6_project (Conv2D)        (None, 8, 8, 64)     12288
block_6_depthwise_relu[0][0]
__________________________________________________________________________________________________
block_6_project_BN (BatchNormal (None, 8, 8, 64)     256
block_6_project[0][0]
__________________________________________________________________________________________________
block_7_expand (Conv2D)         (None, 8, 8, 384)    24576
block_6_project_BN[0][0]
__________________________________________________________________________________________________
block_7_expand_BN (BatchNormali (None, 8, 8, 384)    1536
block_7_expand[0][0]
__________________________________________________________________________________________________
block_7_expand_relu (ReLU)      (None, 8, 8, 384)    0
block_7_expand_BN[0][0]
__________________________________________________________________________________________________
block_7_depthwise (DepthwiseCon (None, 8, 8, 384)    3456
block_7_expand_relu[0][0]
__________________________________________________________________________________________________
block_7_depthwise_BN (BatchNorm (None, 8, 8, 384)    1536
block_7_depthwise[0][0]
__________________________________________________________________________________________________
block_7_depthwise_relu (ReLU)   (None, 8, 8, 384)    0
```

```
block_7_depthwise_BN[0][0]

__________________________________________________________________________________________________
block_7_project (Conv2D)        (None, 8, 8, 64)     24576
block_7_depthwise_relu[0][0]

__________________________________________________________________________________________________
block_7_project_BN (BatchNormal (None, 8, 8, 64)     256
block_7_project[0][0]

__________________________________________________________________________________________________
block_7_add (Add)               (None, 8, 8, 64)     0
block_6_project_BN[0][0]
block_7_project_BN[0][0]

__________________________________________________________________________________________________
block_8_expand (Conv2D)         (None, 8, 8, 384)    24576
block_7_add[0][0]

__________________________________________________________________________________________________
block_8_expand_BN (BatchNormali (None, 8, 8, 384)    1536
block_8_expand[0][0]

__________________________________________________________________________________________________
block_8_expand_relu (ReLU)      (None, 8, 8, 384)    0
block_8_expand_BN[0][0]

__________________________________________________________________________________________________
block_8_depthwise (DepthwiseCon (None, 8, 8, 384)    3456
block_8_expand_relu[0][0]

__________________________________________________________________________________________________
block_8_depthwise_BN (BatchNorm (None, 8, 8, 384)    1536
block_8_depthwise[0][0]

__________________________________________________________________________________________________
block_8_depthwise_relu (ReLU)   (None, 8, 8, 384)    0
block_8_depthwise_BN[0][0]

__________________________________________________________________________________________________
block_8_project (Conv2D)        (None, 8, 8, 64)     24576
block_8_depthwise_relu[0][0]

__________________________________________________________________________________________________
block_8_project_BN (BatchNormal (None, 8, 8, 64)     256
block_8_project[0][0]

__________________________________________________________________________________________________
```

```
block_8_add (Add)              (None, 8, 8, 64)     0
block_7_add[0][0]
block_8_project_BN[0][0]
--------------------------------------------------------------------------------
------------------
block_9_expand (Conv2D)        (None, 8, 8, 384)    24576
block_8_add[0][0]
--------------------------------------------------------------------------------
------------------
block_9_expand_BN (BatchNormali (None, 8, 8, 384)   1536
block_9_expand[0][0]
--------------------------------------------------------------------------------
------------------
block_9_expand_relu (ReLU)     (None, 8, 8, 384)    0
block_9_expand_BN[0][0]
--------------------------------------------------------------------------------
------------------
block_9_depthwise (DepthwiseCon (None, 8, 8, 384)   3456
block_9_expand_relu[0][0]
--------------------------------------------------------------------------------
------------------
block_9_depthwise_BN (BatchNorm (None, 8, 8, 384)   1536
block_9_depthwise[0][0]
--------------------------------------------------------------------------------
------------------
block_9_depthwise_relu (ReLU)  (None, 8, 8, 384)    0
block_9_depthwise_BN[0][0]
--------------------------------------------------------------------------------
------------------
block_9_project (Conv2D)       (None, 8, 8, 64)     24576
block_9_depthwise_relu[0][0]
--------------------------------------------------------------------------------
------------------
block_9_project_BN (BatchNormal (None, 8, 8, 64)    256
block_9_project[0][0]
--------------------------------------------------------------------------------
------------------
block_9_add (Add)              (None, 8, 8, 64)     0
block_8_add[0][0]
block_9_project_BN[0][0]
--------------------------------------------------------------------------------
------------------
block_10_expand (Conv2D)       (None, 8, 8, 384)    24576
block_9_add[0][0]
--------------------------------------------------------------------------------
------------------
block_10_expand_BN (BatchNormal (None, 8, 8, 384)   1536
block_10_expand[0][0]
```

```
==================================================================================
==================
block_10_expand_relu (ReLU)     (None, 8, 8, 384)    0
block_10_expand_BN[0][0]
==================================================================================
==================
block_10_depthwise (DepthwiseCo (None, 8, 8, 384)    3456
block_10_expand_relu[0][0]
==================================================================================
==================
block_10_depthwise_BN (BatchNor (None, 8, 8, 384)    1536
block_10_depthwise[0][0]
==================================================================================
==================
block_10_depthwise_relu (ReLU)  (None, 8, 8, 384)    0
block_10_depthwise_BN[0][0]
==================================================================================
==================
block_10_project (Conv2D)       (None, 8, 8, 96)     36864
block_10_depthwise_relu[0][0]
==================================================================================
==================
block_10_project_BN (BatchNorma (None, 8, 8, 96)     384
block_10_project[0][0]
==================================================================================
==================
block_11_expand (Conv2D)        (None, 8, 8, 576)    55296
block_10_project_BN[0][0]
==================================================================================
==================
block_11_expand_BN (BatchNormal (None, 8, 8, 576)    2304
block_11_expand[0][0]
==================================================================================
==================
block_11_expand_relu (ReLU)     (None, 8, 8, 576)    0
block_11_expand_BN[0][0]
==================================================================================
==================
block_11_depthwise (DepthwiseCo (None, 8, 8, 576)    5184
block_11_expand_relu[0][0]
==================================================================================
==================
block_11_depthwise_BN (BatchNor (None, 8, 8, 576)    2304
block_11_depthwise[0][0]
==================================================================================
==================
block_11_depthwise_relu (ReLU)  (None, 8, 8, 576)    0
block_11_depthwise_BN[0][0]
```

```
--------------------------------------------------------------------------------
------------------
block_11_project (Conv2D)       (None, 8, 8, 96)     55296
block_11_depthwise_relu[0][0]

--------------------------------------------------------------------------------
------------------
block_11_project_BN (BatchNorma (None, 8, 8, 96)     384
block_11_project[0][0]

--------------------------------------------------------------------------------
------------------
block_11_add (Add)              (None, 8, 8, 96)     0
block_10_project_BN[0][0]
block_11_project_BN[0][0]

--------------------------------------------------------------------------------
------------------
block_12_expand (Conv2D)        (None, 8, 8, 576)    55296
block_11_add[0][0]

--------------------------------------------------------------------------------
------------------
block_12_expand_BN (BatchNormal (None, 8, 8, 576)    2304
block_12_expand[0][0]

--------------------------------------------------------------------------------
------------------
block_12_expand_relu (ReLU)     (None, 8, 8, 576)    0
block_12_expand_BN[0][0]

--------------------------------------------------------------------------------
------------------
block_12_depthwise (DepthwiseCo (None, 8, 8, 576)    5184
block_12_expand_relu[0][0]

--------------------------------------------------------------------------------
------------------
block_12_depthwise_BN (BatchNor (None, 8, 8, 576)    2304
block_12_depthwise[0][0]

--------------------------------------------------------------------------------
------------------
block_12_depthwise_relu (ReLU)  (None, 8, 8, 576)    0
block_12_depthwise_BN[0][0]

--------------------------------------------------------------------------------
------------------
block_12_project (Conv2D)       (None, 8, 8, 96)     55296
block_12_depthwise_relu[0][0]

--------------------------------------------------------------------------------
------------------
block_12_project_BN (BatchNorma (None, 8, 8, 96)     384
block_12_project[0][0]

--------------------------------------------------------------------------------
------------------
block_12_add (Add)              (None, 8, 8, 96)     0
```

```
block_11_add[0][0]
block_12_project_BN[0][0]
--------------------------------------------------------------------------------
------------------
block_13_expand (Conv2D)        (None, 8, 8, 576)    55296
block_12_add[0][0]
--------------------------------------------------------------------------------
------------------
block_13_expand_BN (BatchNormal (None, 8, 8, 576)    2304
block_13_expand[0][0]
--------------------------------------------------------------------------------
------------------
block_13_expand_relu (ReLU)     (None, 8, 8, 576)    0
block_13_expand_BN[0][0]
--------------------------------------------------------------------------------
------------------
block_13_pad (ZeroPadding2D)    (None, 9, 9, 576)    0
block_13_expand_relu[0][0]
--------------------------------------------------------------------------------
------------------
block_13_depthwise (DepthwiseCo (None, 4, 4, 576)    5184
block_13_pad[0][0]
--------------------------------------------------------------------------------
------------------
block_13_depthwise_BN (BatchNor (None, 4, 4, 576)    2304
block_13_depthwise[0][0]
--------------------------------------------------------------------------------
------------------
block_13_depthwise_relu (ReLU)  (None, 4, 4, 576)    0
block_13_depthwise_BN[0][0]
--------------------------------------------------------------------------------
------------------
block_13_project (Conv2D)       (None, 4, 4, 160)    92160
block_13_depthwise_relu[0][0]
--------------------------------------------------------------------------------
------------------
block_13_project_BN (BatchNorma (None, 4, 4, 160)    640
block_13_project[0][0]
--------------------------------------------------------------------------------
------------------
block_14_expand (Conv2D)        (None, 4, 4, 960)    153600
block_13_project_BN[0][0]
--------------------------------------------------------------------------------
------------------
block_14_expand_BN (BatchNormal (None, 4, 4, 960)    3840
block_14_expand[0][0]
--------------------------------------------------------------------------------
------------------
```

```
block_14_expand_relu (ReLU)     (None, 4, 4, 960)    0
block_14_expand_BN[0][0]

--------------------------------------------------------------------------------
------------------
block_14_depthwise (DepthwiseCo (None, 4, 4, 960)    8640
block_14_expand_relu[0][0]

--------------------------------------------------------------------------------
------------------
block_14_depthwise_BN (BatchNor (None, 4, 4, 960)    3840
block_14_depthwise[0][0]

--------------------------------------------------------------------------------
------------------
block_14_depthwise_relu (ReLU)  (None, 4, 4, 960)    0
block_14_depthwise_BN[0][0]

--------------------------------------------------------------------------------
------------------
block_14_project (Conv2D)       (None, 4, 4, 160)    153600
block_14_depthwise_relu[0][0]

--------------------------------------------------------------------------------
------------------
block_14_project_BN (BatchNorma (None, 4, 4, 160)    640
block_14_project[0][0]

--------------------------------------------------------------------------------
------------------
block_14_add (Add)              (None, 4, 4, 160)    0
block_13_project_BN[0][0]
block_14_project_BN[0][0]

--------------------------------------------------------------------------------
------------------
block_15_expand (Conv2D)        (None, 4, 4, 960)    153600
block_14_add[0][0]

--------------------------------------------------------------------------------
------------------
block_15_expand_BN (BatchNormal (None, 4, 4, 960)    3840
block_15_expand[0][0]

--------------------------------------------------------------------------------
------------------
block_15_expand_relu (ReLU)     (None, 4, 4, 960)    0
block_15_expand_BN[0][0]

--------------------------------------------------------------------------------
------------------
block_15_depthwise (DepthwiseCo (None, 4, 4, 960)    8640
block_15_expand_relu[0][0]

--------------------------------------------------------------------------------
------------------
block_15_depthwise_BN (BatchNor (None, 4, 4, 960)    3840
block_15_depthwise[0][0]

--------------------------------------------------------------------------------
```

```
------------------
block_15_depthwise_relu (ReLU)  (None, 4, 4, 960)    0
block_15_depthwise_BN[0][0]
________________________________________________________________________________
------------------
block_15_project (Conv2D)       (None, 4, 4, 160)    153600
block_15_depthwise_relu[0][0]
________________________________________________________________________________
------------------
block_15_project_BN (BatchNorma (None, 4, 4, 160)    640
block_15_project[0][0]
________________________________________________________________________________
------------------
block_15_add (Add)              (None, 4, 4, 160)    0
block_14_add[0][0]
block_15_project_BN[0][0]
________________________________________________________________________________
------------------
block_16_expand (Conv2D)        (None, 4, 4, 960)    153600
block_15_add[0][0]
________________________________________________________________________________
------------------
block_16_expand_BN (BatchNormal (None, 4, 4, 960)    3840
block_16_expand[0][0]
________________________________________________________________________________
------------------
block_16_expand_relu (ReLU)     (None, 4, 4, 960)    0
block_16_expand_BN[0][0]
________________________________________________________________________________
------------------
block_16_depthwise (DepthwiseCo (None, 4, 4, 960)    8640
block_16_expand_relu[0][0]
________________________________________________________________________________
------------------
block_16_depthwise_BN (BatchNor (None, 4, 4, 960)    3840
block_16_depthwise[0][0]
________________________________________________________________________________
------------------
block_16_depthwise_relu (ReLU)  (None, 4, 4, 960)    0
block_16_depthwise_BN[0][0]
________________________________________________________________________________
------------------
block_16_project (Conv2D)       (None, 4, 4, 320)    307200
block_16_depthwise_relu[0][0]
________________________________________________________________________________
------------------
block_16_project_BN (BatchNorma (None, 4, 4, 320)    1280
block_16_project[0][0]
```

```
--------------------------------------------------------------------------------
------------------
Conv_1 (Conv2D)                (None, 4, 4, 1280)   409600
block_16_project_BN[0][0]

--------------------------------------------------------------------------------
------------------
Conv_1_bn (BatchNormalization)  (None, 4, 4, 1280)   5120        Conv_1[0][0]

--------------------------------------------------------------------------------
------------------
out_relu (ReLU)                (None, 4, 4, 1280)   0           Conv_1_bn[0][0]
================================================================================
==================
Total params: 2,257,984
Trainable params: 0
Non-trainable params: 2,257,984
--------------------------------------------------------------------------------
------------------
```

[90]:
```
model = Sequential()

model.add(conv_base)

model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dense(1,activation='sigmoid'))
```

[92]:
```
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenetv2_1.00_128 (Functi (None, 4, 4, 1280)        2257984
_________________________________________________________________
flatten_6 (Flatten)          (None, 20480)             0
_________________________________________________________________
dense_12 (Dense)             (None, 128)               2621568
_________________________________________________________________
dense_13 (Dense)             (None, 1)                 129
=================================================================
Total params: 4,879,681
Trainable params: 2,621,697
Non-trainable params: 2,257,984
_________________________________________________________________
```

[95]:
```
model.compile(optimizer='adam', loss='binary_crossentropy',metrics=['accuracy'])
```

[96]:
```
import time

start = time.perf_counter()

perf = model.fit_generator(train_generator, epochs=15,␣
 ↪callbacks=[early_stop],validation_data=validation_generator)

elapsed = time.perf_counter()- start

print('Elapsed {}'.format(elapsed/60))
```

```
/usr/local/lib/python3.6/dist-
packages/tensorflow/python/keras/engine/training.py:1844: UserWarning:
`Model.fit_generator` is deprecated and will be removed in a future version.
Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
```

```
Epoch 1/15
271/293 [==========================>…] - ETA: 5s - loss: 0.2902 - accuracy:
0.9396
```

```
/usr/local/lib/python3.6/dist-packages/PIL/TiffImagePlugin.py:770: UserWarning:
Possibly corrupt EXIF data.  Expecting to read 32 bytes but only got 0. Skipping
tag 270
  " Skipping tag %s" % (size, len(data), tag)
/usr/local/lib/python3.6/dist-packages/PIL/TiffImagePlugin.py:770: UserWarning:
Possibly corrupt EXIF data.  Expecting to read 5 bytes but only got 0. Skipping
tag 271
  " Skipping tag %s" % (size, len(data), tag)
/usr/local/lib/python3.6/dist-packages/PIL/TiffImagePlugin.py:770: UserWarning:
Possibly corrupt EXIF data.  Expecting to read 8 bytes but only got 0. Skipping
tag 272
  " Skipping tag %s" % (size, len(data), tag)
/usr/local/lib/python3.6/dist-packages/PIL/TiffImagePlugin.py:770: UserWarning:
Possibly corrupt EXIF data.  Expecting to read 8 bytes but only got 0. Skipping
tag 282
  " Skipping tag %s" % (size, len(data), tag)
/usr/local/lib/python3.6/dist-packages/PIL/TiffImagePlugin.py:770: UserWarning:
Possibly corrupt EXIF data.  Expecting to read 8 bytes but only got 0. Skipping
tag 283
  " Skipping tag %s" % (size, len(data), tag)
/usr/local/lib/python3.6/dist-packages/PIL/TiffImagePlugin.py:770: UserWarning:
Possibly corrupt EXIF data.  Expecting to read 20 bytes but only got 0. Skipping
tag 306
  " Skipping tag %s" % (size, len(data), tag)
/usr/local/lib/python3.6/dist-packages/PIL/TiffImagePlugin.py:770: UserWarning:
```

```
Possibly corrupt EXIF data.  Expecting to read 48 bytes but only got 0. Skipping
tag 532
  " Skipping tag %s" % (size, len(data), tag)
/usr/local/lib/python3.6/dist-packages/PIL/TiffImagePlugin.py:788: UserWarning:
Corrupt EXIF data.  Expecting to read 2 bytes but only got 0.
  warnings.warn(str(msg))
```

```
293/293 [==============================] - 93s 309ms/step - loss: 0.2795 -
accuracy: 0.9410 - val_loss: 0.0867 - val_accuracy: 0.9635
Epoch 2/15
293/293 [==============================] - 89s 303ms/step - loss: 0.0524 -
accuracy: 0.9800 - val_loss: 0.0967 - val_accuracy: 0.9651
Epoch 3/15
293/293 [==============================] - 89s 303ms/step - loss: 0.0266 -
accuracy: 0.9907 - val_loss: 0.1207 - val_accuracy: 0.9610
Epoch 4/15
293/293 [==============================] - 88s 302ms/step - loss: 0.0168 -
accuracy: 0.9939 - val_loss: 0.1493 - val_accuracy: 0.9619
Elapsed 5.979793827750003
```

[97]:

```
new_loss, new_acc = model.evaluate_generator(validation_generator)
```

```
/usr/local/lib/python3.6/dist-
packages/tensorflow/python/keras/engine/training.py:1877: UserWarning:
`Model.evaluate_generator` is deprecated and will be removed in a future
version. Please use `Model.evaluate`, which supports generators.
  warnings.warn('`Model.evaluate_generator` is deprecated and '
```

[98]:

```
print('Validation Accuracy - {:.2f}%'.format(new_acc*100))
print('validation loss - {:.2f}'.format(new_loss))
```

```
Validation Accuracy - 96.19%
validation loss - 0.15
```

[99]:

```
model_history = pd.DataFrame(perf.history)
```

[100]:

```
model_history
```

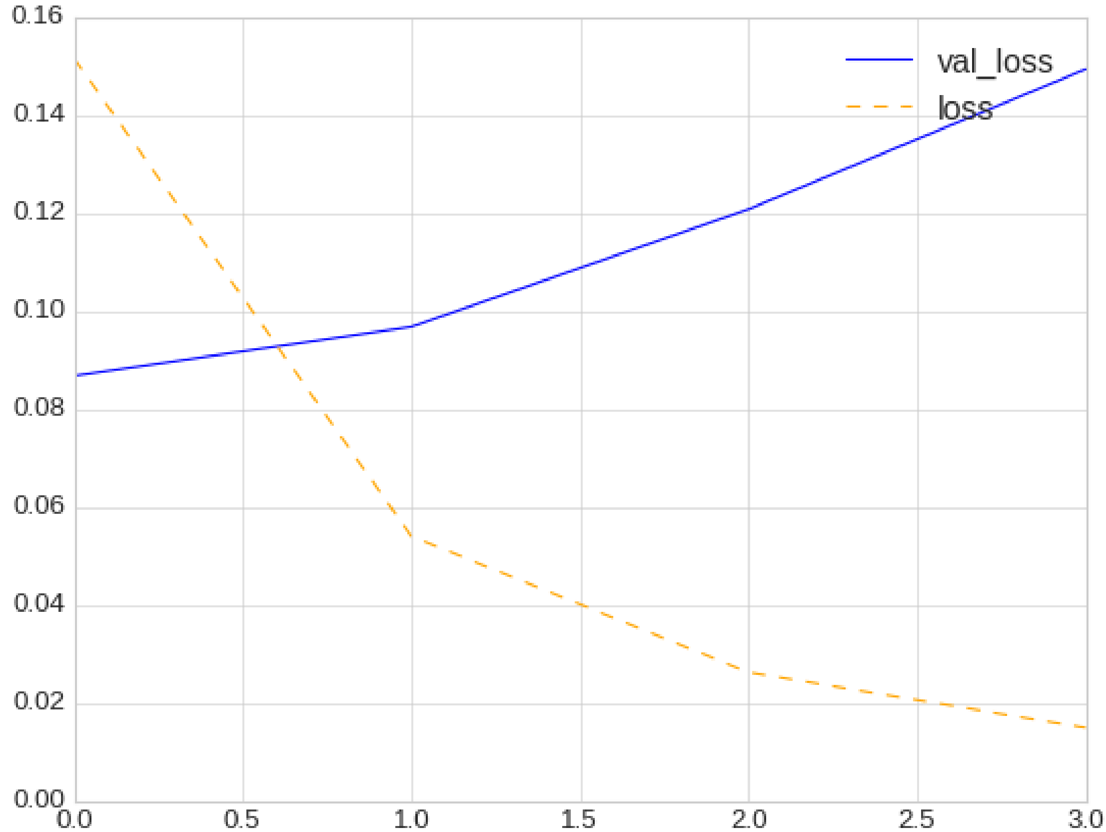
[100]:

	loss	accuracy	val_loss	val_accuracy
0	0.151495	0.958011	0.086692	0.963526
1	0.053979	0.979619	0.096691	0.965126
2	0.026196	0.990236	0.120692	0.960966
3	0.014898	0.994505	0.149257	0.961926

[101]:

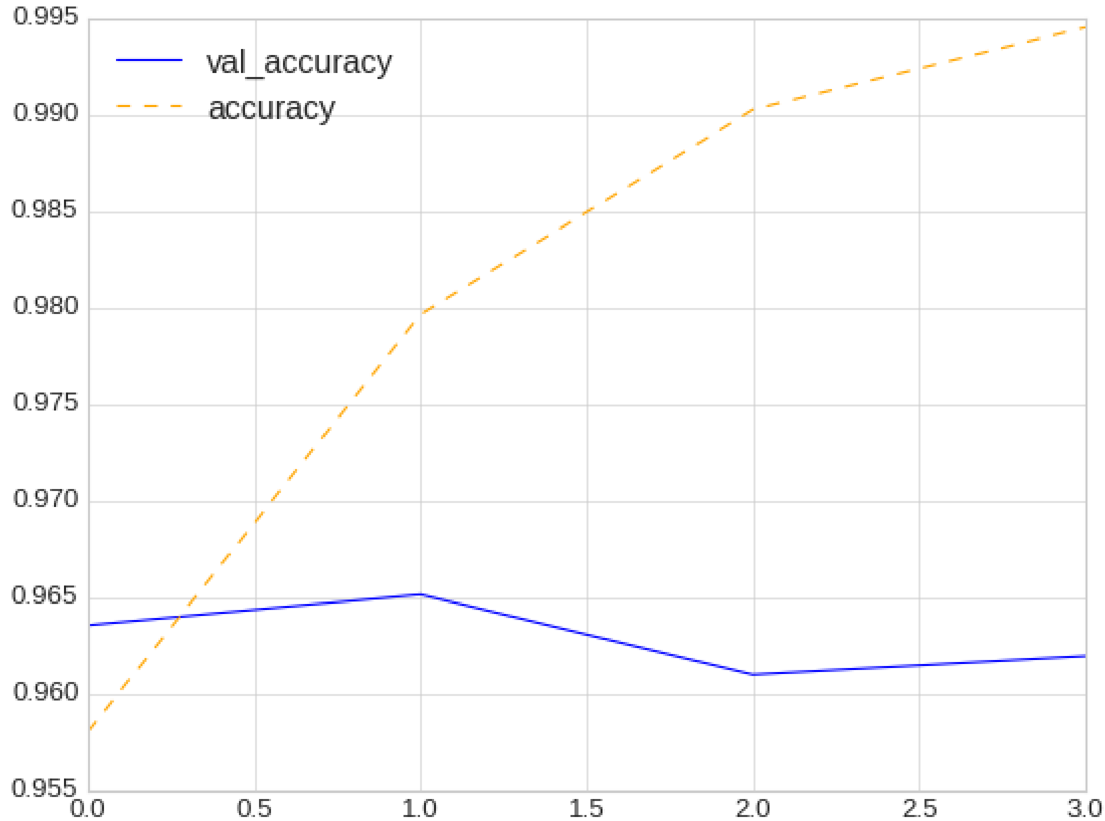
```
plt.style.use('seaborn-whitegrid')
plt.plot(model_history['val_loss'], ls='-',color='blue',label='val_loss')
plt.plot(model_history['loss'],ls='--',color='orange',label='loss' )
plt.legend()
```

[101]: <matplotlib.legend.Legend at 0x7f22163a81d0>



[103]:
```
plt.style.use('seaborn-whitegrid')
plt.plot(model_history['val_accuracy'],␣
↪ls='-',color='blue',label='val_accuracy')
plt.plot(model_history['accuracy'],ls='--',color='orange',label='accuracy' )
plt.legend(loc='upper left')
```

[103]: <matplotlib.legend.Legend at 0x7f22149be6d8>



[104]:
```
pred_probs = model.predict_generator(validation_generator)
preds = pred_probs > 0.5
```

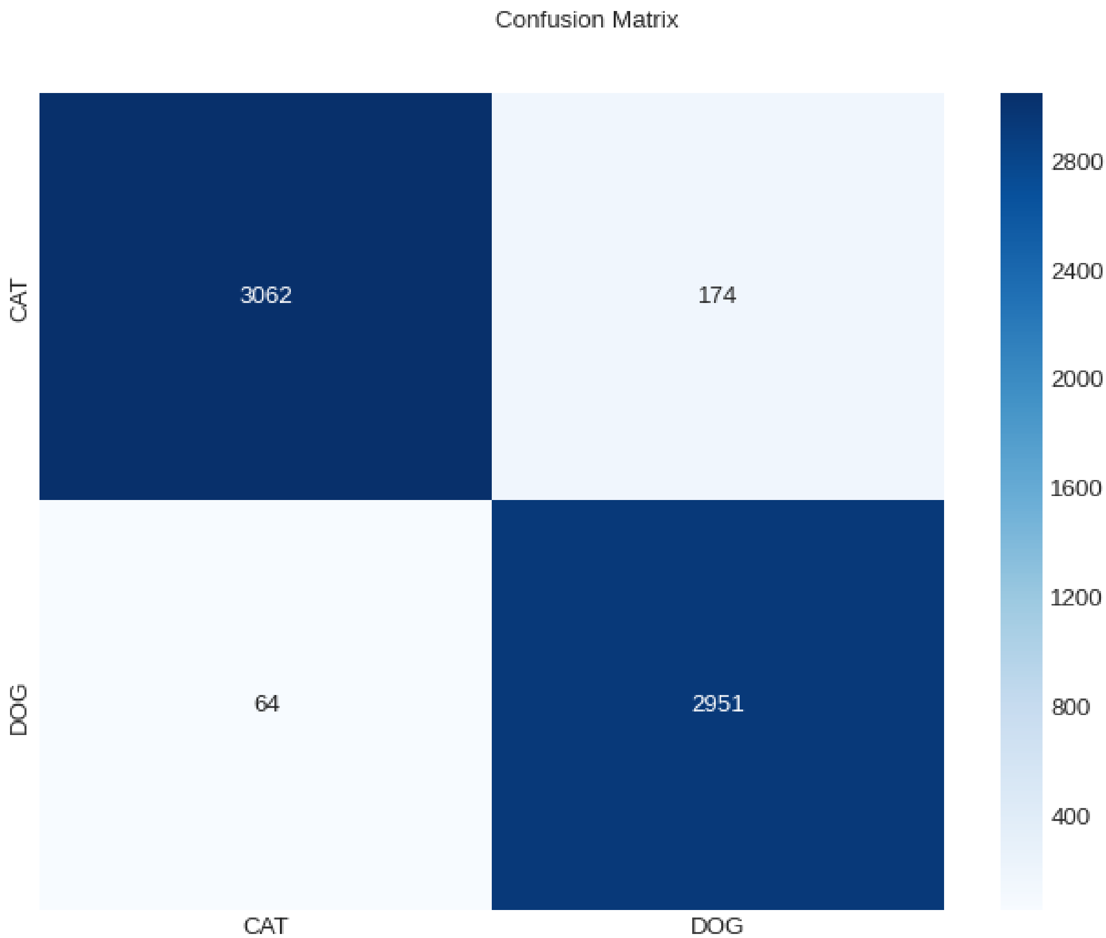
```
/usr/local/lib/python3.6/dist-
packages/tensorflow/python/keras/engine/training.py:1905: UserWarning:
`Model.predict_generator` is deprecated and will be removed in a future version.
Please use `Model.predict`, which supports generators.
  warnings.warn('`Model.predict_generator` is deprecated and '
```

[105]:
```
print(classification_report(preds, validation_generator.classes))
```

```
              precision    recall  f1-score   support

       False       0.98      0.95      0.96      3236
        True       0.94      0.98      0.96      3015

    accuracy                           0.96      6251
   macro avg       0.96      0.96      0.96      6251
weighted avg       0.96      0.96      0.96      6251
```

[106]:
```
plt.figure(figsize=(10,7))
plt.suptitle('Confusion Matrix')
confmat = pd.DataFrame(confusion_matrix(preds, validation_generator.classes),␣
 ↪columns=validation_generator.class_indices.keys())
confmat.index = validation_generator.class_indices.keys()
sns.heatmap(confmat, annot=True, fmt='d', cmap='Blues')
```

[106]: <matplotlib.axes._subplots.AxesSubplot at 0x7f22163e8d68>



1 Inference-2

With **MobileNetV2** + custom output layer, the model achieved 99% on training set and 96% on the Validation Set, this difference indicates that the model is overfitting on the training set, but this is not prominent to finetune futher generally. But to create a state of the art cats & dogs classifier we can futhur fine-tune it to reduce that 3% of variance with techniques like Dropout and L2 Regularisation.

[107]:
```
model.save('Cats&Dogs-MovbileNetv2.h5')
```

1.1 Miscellaneous

[108]:
```
import os
path = './samples/'
overview_path = './samples/overview.txt'
eval_path  = './samples/evaluate.txt'

if os.path.exists(path):

    print('samples dir, exists..checking for dictionaries existence..')

    if os.path.exists(overview_path) and os.path.exists(eval_path):
        print('Data exists. no need of overwritting.')
    else:
        print("overview and eval doesn't exist, proceed to step-2")

else:
    print("samples/ dir is non-existent, Establishing one..")
    os.mkdir(path) # samples directory
```

```
samples/ dir is non-existent, Establishing one..
```

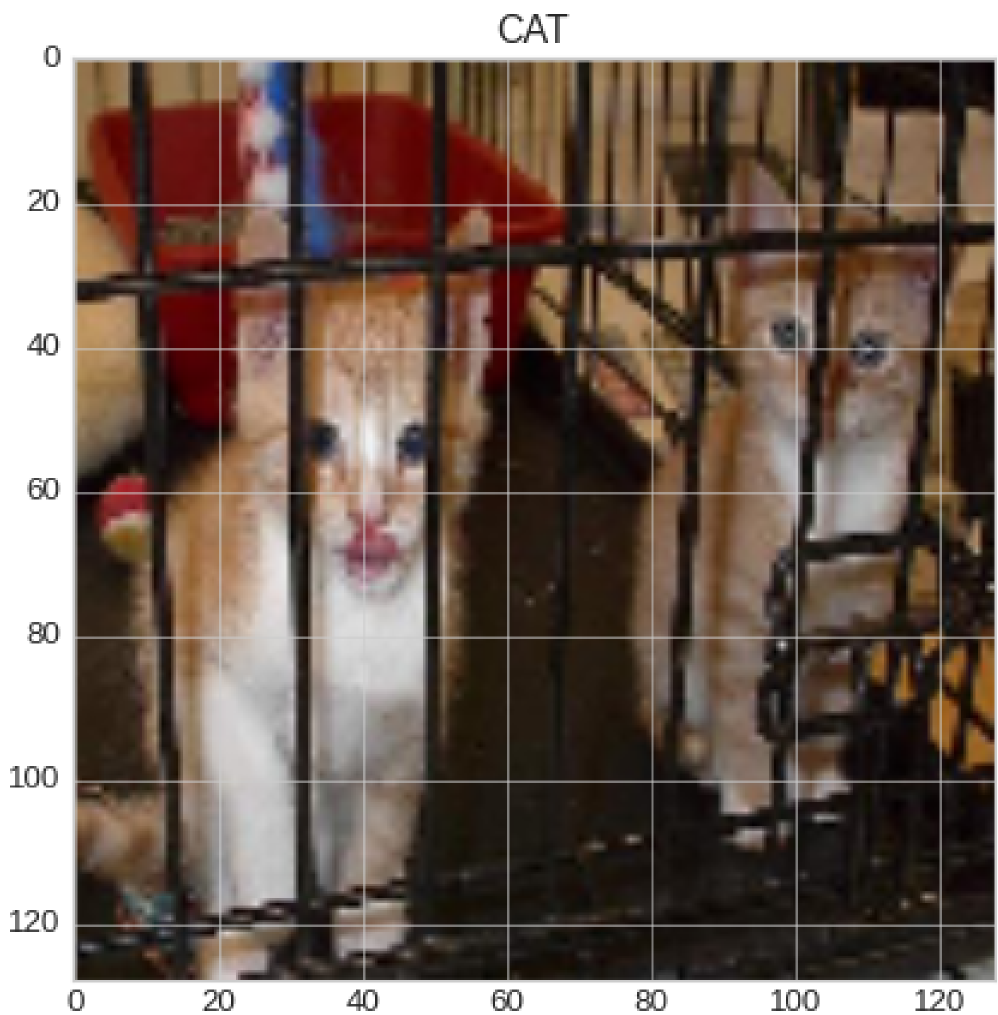
[110]:
```
x_train, y_train = next(train_generator) # one batch
x_test, y_test = next(validation_generator) # one batch
```

[136]:
```
classes = np.array(list(validation_generator.class_indices))
x = np.array((x_train[0],x_train[7], x_train[10]))
y = np.array((y_train[0],y_train[7], y_train[10]))
```

[147]:
```
y = ['CAT', 'DOG', 'CAT']
```

[148]:
```
plt.imshow(x[2])
plt.title(y[2])
```

[148]:
```
Text(0.5, 1.0, 'CAT')
```



[155]:
```

```

[155]:
```
array([0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 0., 1., 0., 0., 0., 0., 0.,
       1., 1., 1., 1., 0., 0., 0., 0., 0., 0., 1., 0., 0., 1., 1., 1., 1.,
       1., 0., 0., 0., 0., 0., 1., 1., 1., 1., 0., 1., 0., 1., 0., 1., 0.,
       1., 0., 0., 1., 0., 1., 0., 0., 1., 0., 1., 0., 1.], dtype=float32)
```

[157]:
```
overview_dict = {}
eval_dict = {}

# fill the following -
# for overview
#string
kind = 'Image Data'
#tuple
```

```
dimensions = x_train.shape
#labels : str(list of unique target values)
targets = list(validation_generator.class_indices.values())
#nd.array
data = x
#nd.array or class_names
labels = y

vars0 = ['kind','dimensions', 'targets', 'data', 'labels']

# filling overview_dict
for x in vars0:
    try:
        overview_dict[x] = eval(x)
    except:
        overview_dict[x] = x

# evaluate_dict

eval_dict = {'test_cases' : x_train, 'true': y_train, 'class_names':␣
 ↪['CAT','DOG'] ,'model':'/model.h5'}
```

[159]:
```
import pickle
# dump 1
with open(overview_path,'wb') as f:
    pickle.dump(overview_dict,f)

# dump 2
with open(eval_path,'wb') as f:
    pickle.dump(eval_dict,f)
```

[160]:
```
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenetv2_1.00_128 (Functi (None, 4, 4, 1280)        2257984
_________________________________________________________________
flatten_6 (Flatten)          (None, 20480)             0
_________________________________________________________________
dense_12 (Dense)             (None, 128)               2621568
_________________________________________________________________
dense_13 (Dense)             (None, 1)                 129
=================================================================
Total params: 4,879,681
Trainable params: 2,621,697
```

```
Non-trainable params: 2,257,984
_________________________________________________________________
```

[161]:
```
s = '''
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenetv2_1.00_128 (Functi (None, 4, 4, 1280)        2257984
_________________________________________________________________
flatten_6 (Flatten)          (None, 20480)             0
_________________________________________________________________
dense_12 (Dense)             (None, 128)               2621568
_________________________________________________________________
dense_13 (Dense)             (None, 1)                 129
=================================================================
Total params: 4,879,681
Trainable params: 2,621,697
Non-trainable params: 2,257,984
_________________________________________________________________



'''
```

[162]:
```
report = '''

              precision    recall  f1-score   support

       False       0.98      0.95      0.96      3236
        True       0.94      0.98      0.96      3015

    accuracy                           0.96      6251
   macro avg       0.96      0.96      0.96      6251
weighted avg       0.96      0.96      0.96      6251


'''
```

[ ]:
```

```

[166]:
```
synopsis = '''

For this problem two Implementations were used with Two inferences at the end␣
 ↪of each implementation

exe-1 = ConvNet from Scratch
```

```
* A ConvNet was built from scratch with 3 Conv2D, 3 MaxPool, 1 Flatten and 2␣
↪Dense Layers with Output function as Sigmoid ( you can find the␣
↪implementation in the ipynb-pdf ).
* This ConvNet was set for 15 epochs with EarlyStopping Callback of patience 3.␣
↪The Elapsed Training time was 12.18 Mins on Cloud GPU instance of only 8␣
↪epochs of the stipulated 15.
* Attained Training Accuracy - 95.60% and Validation Accuracy - 83.75% with␣
↪approx. 8% difference, This network was overfitting with an average f1-score␣
↪of 84%.
* This Network had the classic case of misclassifying cats as dogs.

exe-2 = MobileNetv2 ( Frozen )

* Convbase of MobileNetV2 with ImageNet weights and Frozen layers was␣
↪constructed.
* Custom Output layers of 1 Flatten, 2 Dense was attached at the end.
* Same as the previous implementation the model was set to 15 epochs with same␣
↪EarlyStopping Setup. Elapsed Training time is 5 Mins with 3 Epochs of␣
↪Training.
* This Time the Model Performed well enough with the pre-trained ImageNet␣
↪Weights. The misclassification problem from the first case was smoothened.
* With **MobileNetV2** + custom output layer, the model achieved 99% on␣
↪training set and 96% on the Validation Set, this difference indicates that␣
↪the model is overfitting on the training set, but this is not prominent to␣
↪fine tune further generally. But to create a state of the art cats & dogs␣
↪classifier we can futhur fine-tune it to reduce that 3% of variance with␣
↪techniques like Dropout and L2 Regularisation.


'''
```

[167]:
```
print(synopsis)
```

```
For this problem two Implementations were used with Two inferences at the end of
each implementation

exe-1 = ConvNet from Scratch

* A ConvNet was built from scratch with 3 Conv2D, 3 MaxPool, 1 Flatten and 2
Dense Layers with Output function as Sigmoid ( you can find the implementation
in the ipynb-pdf ).
* This ConvNet was set for 15 epochs with EarlyStopping Callback of patience 3.
The Elapsed Training time was 12.18 Mins on Cloud GPU instance of only 8 epochs
of the stipulated 15.
```

* Attained Training Accuracy - 95.60% and Validation Accuracy - 83.75% with approx. 8% difference, This network was overfitting with an average f1-score of 84%.
* This Network had the classic case of misclassifying cats as dogs.

exe-2 = MobileNetv2 (Frozen)

* Convbase of MobileNetV2 with ImageNet weights and Frozen layers was constructed.
* Custom Output layers of 1 Flatten, 2 Dense was attached at the end.
* Same as the previous implementation the model was set to 15 epochs with same EarlyStopping Setup. Elapsed Training time is 5 Mins with 3 Epochs of Training.
* This Time the Model Performed well enough with the pre-trained ImageNet Weights. The misclassification problem from the first case was smoothened.
* With **MobileNetV2** + custom output layer, the model achieved 99% on training set and 96% on the Validation Set, this difference indicates that the model is overfitting on the training set, but this is not prominent to fine tune further generally. But to create a state of the art cats & dogs classifier we can futhur fine-tune it to reduce that 3% of variance with techniques like Dropout and L2 Regularisation.

[170]:
```
desc = '''A simple Cats and Dogs Image Dataset with 24994 samples of both cats␣
 ↪and dog pictures combined.'''
project_name = 'Cats & Dogs Classifier'
framework = 'Keras'
prediction_type = 'Classification of 2 Targets'
network_type = 'MobileNetV2'
architecture = s
layers = '19 Residual Units + 3 Custom Output Layers'
hidden_units = 2
activations = "['relu', 'sigmoid']"
epochs = "Set of 15, Trained for 3 ( EarlyStopping ) "
metrics = "Accuracy"
loss = "Binary Cross-Entropy"
optimiser = 'Adam'
learning_rate = 0.001
batch_size = 64
train_performance = '99.45%'
test_performance = '96.14%'
classification_report = report
elapsed = "5Min, runtime : Colab Cloud GPU"
summary = synopsis
ipynb = './Projects/Transfer-Learning/Dogs&Cats/Dogs&Cats-classifier.pdf'
plots = './Projects/Transfer-Learning/Dogs&Cats/Plots'
```

[184]:
```python
p = {}
```

[194]:
```python
var = ['desc','project_name',
 ↪'framework','prediction_type','network_type','architecture','layers','hidden_units','activa

param = {}
for val in var:

  try:
      param[val] = eval(val)

  except:
      param[val] = val

    # check if anything is missing
```

[195]:
```python
param
```

[195]:
```
{'activations': "['relu', 'sigmoid']",
 'architecture': '\nModel: "sequential_4"\n________________________________________
____________________________\nLayer (type)                 Output Shape
Param #   \n=================================================================\nm
obilenetv2_1.00_128 (Functi (None, 4, 4, 1280)       2257984
\n_________________________________________________________________\nflatten_6
(Flatten)           (None, 20480)            0
\n_________________________________________________________________\ndense_12
(Dense)              (None, 128)               2621568
\n_________________________________________________________________\ndense_13
(Dense)              (None, 1)                 129
\n=================================================================\nTotal
params: 4,879,681\nTrainable params: 2,621,697\nNon-trainable params: 2,257,984\
n_________________________________________________________________\n\n\n\n',
 'batch_size': 64,
 'classification_report': '\n              \n              precision    recall
f1-score   support\n\n       False       0.98      0.95      0.96      3236\n
True       0.94      0.98      0.96      3015\n\n    accuracy
0.96      6251\n   macro avg       0.96      0.96      0.96      6251\nweighted
avg       0.96      0.96      0.96      6251\n\n\n',
 'desc': 'A simple Cats and Dogs Image Dataset with 24994 samples of both cats
and dog pictures combined.',
 'elapsed': '5Min, runtime : Colab Cloud GPU',
 'epochs': 'Set of 15, Trained for 3 ( EarlyStopping ) ',
 'framework': 'Keras',
 'hidden_units': 2,
 'ipynb': './Projects/Transfer-Learning/Dogs&Cats/Dogs&Cats-classifier.pdf',
 'layers': '19 Residual Units + 3 Custom Output Layers',
 'learning_rate': 0.001,
```

```
 'loss': 'Binary Cross-Entropy',
 'metrics': 'Accuracy',
 'network_type': 'MobileNetV2',
 'optimiser': 'Adam',
 'plots': './Projects/Transfer-Learning/Dogs&Cats/Plots',
 'prediction_type': 'Classification of 2 Targets',
 'project_name': 'Cats & Dogs Classifier',
 'summary': '\n\nFor this problem two Implementations were used with Two
inferences at the end of each implementation\n\nexe-1 = ConvNet from
Scratch\n\n* A ConvNet was built from scratch with 3 Conv2D, 3 MaxPool, 1
Flatten and 2 Dense Layers with Output function as Sigmoid ( you can find the
implementation in the ipynb-pdf ). \n* This ConvNet was set for 15 epochs with
EarlyStopping Callback of patience 3. The Elapsed Training time was 12.18 Mins
on Cloud GPU instance of only 8 epochs of the stipulated 15.\n* Attained
Training Accuracy - 95.60% and Validation Accuracy - 83.75% with approx. 8%
difference, This network was overfitting with an average f1-score of 84%.\n*
This Network had the classic case of misclassifying cats as dogs.\n\nexe-2 =
MobileNetv2 ( Frozen )\n\n* Convbase of MobileNetV2 with ImageNet weights and
Frozen layers was constructed.\n* Custom Output layers of 1 Flatten, 2 Dense was
attached at the end.\n* Same as the previous implementation the model was set to
15 epochs with same EarlyStopping Setup. Elapsed Training time is 5 Mins with 3
Epochs of Training.\n* This Time the Model Performed well enough with the pre-
trained ImageNet Weights. The misclassification problem from the first case was
smoothened. \n* With **MobileNetV2** + custom output layer, the model achieved
99% on training set and 96% on the Validation Set, this difference indicates
that the model is overfitting on the training set, but this is not prominent to
fine tune further generally. But to create a state of the art cats & dogs
classifier we can futhur fine-tune it to reduce that 3% of variance with
techniques like Dropout and L2 Regularisation.\n\n\n',
 'test_performance': '96.14%',
 'train_performance': '99.45%'}
```

[196]:
```python
import pickle
file = open("artefacts.txt", "wb")
dictionary = param
pickle.dump(dictionary, file)
file.close()
```

[ ]: