Homework 3: Convolutional Neural Networks

Due Wednesday 11/24 at 11:59 pm EST

Download the dataset `cats-notcats` from github (given as a part of the assignment). This dataset has images of cats and images that are not cats (in separate folders). The task is to train a convolutional neural network (CNN) to build a classifier that can classify a new image as either `cat` or `not cat`

1. Load the dataset and create three stratified splits - train/validation/test in the ratio of 70/10/20.

In [4]:
```
import tensorflow as tf
from tensorflow import keras

batch_size=32
image_generator = tf.keras.preprocessing.image.ImageDataGenerator(validatio
all_data_batches = image_generator.flow_from_directory(directory='/Users/Gr
                                                       class_mode='binary',

val_gen = image_generator.flow_from_directory(directory='/Users/Griffin/rep
                                              class_mode='binary', target_s




#print(batches)
```

```
Found 5668 images belonging to 2 classes.
Found 1133 images belonging to 2 classes.
```

In [5]:
```
#for batch in all_data_batches:
#    print(batch)
```

2. Create a CNN that has the following hidden layers:

   a. 2D convolution layer with a 3x3 kernel size, has 128 filters, stride of 1 and padded to yield the same size as input, followed by a ReLU activation layer

   b. Max pooling layer of 2x2

   c. Dense layer with 128 dimensions and ReLU as the activation layer

In [6]:
```
    model = tf.keras.Sequential(
        [
            tf.keras.layers.Conv2D(128, kernel_size=(3,3), activation='relu
            tf.keras.layers.MaxPooling2D(
                pool_size=(2,2)),
            tf.keras.layers.Flatten(),
            tf.keras.layers.Dense(128, activation="relu", name="c"),
        ]
    )
model.compile(loss='binary_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

3. Train the classifier for 20 epochs with 100 steps per epoch. Also use the validation data during training the estimator.

In [7]:
```
model.fit_generator(all_data_batches, epochs=20, steps_per_epoch=100, valid
model.save_weights(first_try.h5)
```

```
<ipython-input-7-13f639a6401b>:1: UserWarning: `Model.fit_generator` is d
eprecated and will be removed in a future version. Please use `Model.fit
`, which supports generators.
  model.fit_generator(all_data_batches, epochs=20, steps_per_epoch=100, v
alidation_data=val_gen)

---------------------------------------------------------------------------
--
ValueError                                Traceback (most recent call las
t)
<ipython-input-7-13f639a6401b> in <module>
----> 1 model.fit_generator(all_data_batches, epochs=20, steps_per_epoch=
100, validation_data=val_gen)
      2 model.save_weights(first_try.h5)

~/opt/anaconda3/lib/python3.8/site-packages/keras/engine/training.py in f
it_generator(self, generator, steps_per_epoch, epochs, verbose, callback
s, validation_data, validation_steps, validation_freq, class_weight, max_
queue_size, workers, use_multiprocessing, shuffle, initial_epoch)
   2014         'Please use `Model.fit`, which supports generators.',
   2015         stacklevel=2)
-> 2016     return self.fit(
   2017         generator,
   2018         steps_per_epoch=steps_per_epoch,

~/opt/anaconda3/lib/python3.8/site-packages/keras/utils/traceback_utils.p
y in error_handler(*args, **kwargs)
     65     except Exception as e:  # pylint: disable=broad-except
     66       filtered_tb = _process_traceback_frames(e.__traceback__)
---> 67       raise e.with_traceback(filtered_tb) from None
     68     finally:
     69       del filtered_tb

~/opt/anaconda3/lib/python3.8/site-packages/keras/engine/input_spec.py in
assert_input_compatibility(input_spec, inputs, layer_name)
    225       ndim = x.shape.rank
    226       if ndim is not None and ndim < spec.min_ndim:
--> 227         raise ValueError(f'Input {input_index} of layer "{layer_n
ame}" '
    228                          'is incompatible with the layer: '
    229                          f'expected min_ndim={spec.min_ndim}, '

ValueError: Exception encountered when calling layer "sequential" (type S
equential).

Input 0 of layer "conv2d" is incompatible with the layer: expected min_nd
im=4, found ndim=2. Full shape received: (16, 196608)

Call arguments received:
  • inputs=tf.Tensor(shape=(16, 256, 256, 3), dtype=float32)
  • training=False
  • mask=None
```

4. Plot the accuracy and the loss over epochs for train & validation sets

In [ ]:
```
```

5. Add the following layers to (2) before the dense layer:

   a. 2D convolution layer with a 3x3 kernel size, has 64 filters, stride of 1 and padded to yield the same size as input, followed by a ReLU activation layer

   b. Max pooling layer of 2x2

   c. 2D convolution layer with a 3x3 kernel size, has 32 filters, stride of 1 and padded to yield the same size as input, followed by a ReLU activation layer

   d. Max pooling layer of 2x2

   e. Dense layer with 256 dimensions and ReLU as the activation layer

In [17]:
```
model2 = tf.keras.Sequential(
    [
        tf.keras.layers.Conv2D(kernel_size=(3,3), activation='relu', pa
                               strides=1, input_shape=(256,256,
        tf.keras.layers.MaxPooling2D(
            pool_size=(2,2)),
        tf.keras.layers.Conv2D(kernel_size=(3,3), activation='relu', pa
                               strides=1, input_shape=(256,256,
        tf.keras.layers.MaxPooling2D(
            pool_size=(2,2)),
        tf.keras.layers.Dense(256, activation="relu")
        tf.keras.layers.Dense(128, activation="relu"),
    ]
)
model2.compile(tf.keras.optimizers.Adam(lr=0.01), loss="categorical_cro
```

6. Train the classifier again for 20 epochs with 100 steps per epoch. Also use the validation data during training the estimator.

In [19]:
```
model.fit(INPUT, OUTPUT, epochs=20, steps_per_epoch=100)
```

7. Plot the accuracy and the loss over epochs for train & validation sets

In [21]:
```
#code here
```