Week#6_assignment

March 5, 2021

Assignment 6.1 Using section 5.1 in Deep Learning with Python as a guide (listing 5.3 in particular), create a ConvNet model that classifies images in the MNIST digit dataset. Save the model, predictions, metrics, and validation plots in the dsc650/assignments/assignment06/results directory. If you are using JupyterHub, you can include those plots in your Jupyter notebook.

[43]:

```
from keras.datasets import mnist
from keras.utils import to_categorical
from keras import layers
from keras import models
import tensorflow as tf
```

[6]:

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

[8]:

```
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

[9]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 3, 3, 64)          36928
_________________________________________________________________
```

```
flatten (Flatten)            (None, 576)               0
_________________________________________________________________
dense (Dense)                (None, 64)                36928
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
_________________________________________________________________
```

[10]:

```
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
train_images = train_images.reshape((60000, 28, 28, 1))
train_images = train_images.astype('float32') / 255
test_images = test_images.reshape((10000, 28, 28, 1))
test_images = test_images.astype('float32') / 255
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5, batch_size=64)
```

```
Epoch 1/5
938/938 [==============================] - 13s 14ms/step - loss: 0.3693 -
accuracy: 0.8800
Epoch 2/5
938/938 [==============================] - 12s 12ms/step - loss: 0.0507 -
accuracy: 0.9838
Epoch 3/5
938/938 [==============================] - 12s 12ms/step - loss: 0.0332 -
accuracy: 0.9902
Epoch 4/5
938/938 [==============================] - 12s 12ms/step - loss: 0.0240 -
accuracy: 0.9924
Epoch 5/5
938/938 [==============================] - 12s 12ms/step - loss: 0.0192 -
accuracy: 0.9942
```

[10]:

```
<tensorflow.python.keras.callbacks.History at 0x7f1a2c8dc760>
```

[11]:

```
test_loss, test_acc = model.evaluate(test_images, test_labels)
test_acc
```

```
313/313 [==============================] - 1s 4ms/step - loss: 0.0252 -
accuracy: 0.9928
```

[11]:

```
0.9927999973297119
```

[12]:

```
model.save('Convnet_MNIST_digit.h5')
```

Using section 5.2 in Deep Learning with Python as a guide, create a ConvNet model that classifies images CIFAR10 small images classification dataset. Do not use dropout or data-augmentation in this part. Save the model, predictions, metrics, and validation plots in the dsc650/assignments/assignment06/results directory. If you are using JupyterHub, you can include those plots in your Jupyter notebook.

[44]:

```
import os
import keras
from keras.preprocessing import image
#(x_train, y_train), (x_val, y_val) = keras.datasets.cifar10.load_data()
```

from keras import optimizers model.compile(loss='binary_crossentropy', optimizer=optimizers.RMSprop(lr=1e-4),metrics=['acc'])

[45]:

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

[46]:

```
model.summary()
```

```
Model: "sequential_1"

-----------------------------------------------------------------
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 148, 148, 32)      896

-----------------------------------------------------------------
max_pooling2d_4 (MaxPooling2 (None, 74, 74, 32)        0

-----------------------------------------------------------------
conv2d_5 (Conv2D)            (None, 72, 72, 64)        18496

-----------------------------------------------------------------
max_pooling2d_5 (MaxPooling2 (None, 36, 36, 64)        0

-----------------------------------------------------------------
conv2d_6 (Conv2D)            (None, 34, 34, 128)       73856

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

```
max_pooling2d_6 (MaxPooling2 (None, 17, 17, 128)       0
-----------------------------------------------------------------
conv2d_7 (Conv2D)            (None, 15, 15, 128)       147584
-----------------------------------------------------------------
max_pooling2d_7 (MaxPooling2 (None, 7, 7, 128)         0
-----------------------------------------------------------------
flatten_1 (Flatten)          (None, 6272)              0
-----------------------------------------------------------------
dense_2 (Dense)              (None, 512)               3211776
-----------------------------------------------------------------
dense_3 (Dense)              (None, 1)                 513
=================================================================
Total params: 3,453,121
Trainable params: 3,453,121
Non-trainable params: 0
-----------------------------------------------------------------
```

[47]:

```
from keras import optimizers
model.compile(loss='binary_crossentropy',
                      optimizer=optimizers.RMSprop(lr=1e-4),
                      metrics=['acc'])
```

[48]:

```
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)
```

[49]:

```
from PIL import Image
import numpy as np
img_data = np.random.random(size=(100, 100, 3))
img = tf.keras.preprocessing.image.array_to_img(img_data)
array = tf.keras.preprocessing.image.img_to_array(img)
```

[50]:

```
num_classes = 10
epochs = 5
(x_train, y_train), (x_test, y_test) = keras.datasets.cifar10.load_data()
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
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)
```

```
Epoch 1/5
```

```
---------------------------------------------------------------------------
InvalidArgumentError                      Traceback (most recent call last)
<ipython-input-50-dd3cffd2b1a2> in <module>
     15 datagen.fit(x_train)
     16 # fits the model on batches with real-time data augmentation:
---> 17 model.fit(datagen.flow(x_train, y_train, batch_size=32),

     18           steps_per_epoch=len(x_train) / 32, epochs=epochs)

/opt/conda/lib/python3.8/site-packages/tensorflow/python/keras/engine/training.
 ↪py in fit(self, x, y, batch_size, epochs, verbose, callbacks,␣
 ↪validation_split, validation_data, shuffle, class_weight, sample_weight,␣
 ↪initial_epoch, steps_per_epoch, validation_steps, validation_batch_size,␣
 ↪validation_freq, max_queue_size, workers, use_multiprocessing)
   1098                 _r=1):
   1099               callbacks.on_train_batch_begin(step)
-> 1100               tmp_logs = self.train_function(iterator)
   1101               if data_handler.should_sync:
   1102                 context.async_wait()

/opt/conda/lib/python3.8/site-packages/tensorflow/python/eager/def_function.py␣
 ↪in __call__(self, *args, **kwds)
    826     tracing_count = self.experimental_get_tracing_count()
    827     with trace.Trace(self._name) as tm:
--> 828       result = self._call(*args, **kwds)
    829       compiler = "xla" if self._experimental_compile else "nonXla"
    830       new_tracing_count = self.experimental_get_tracing_count()

/opt/conda/lib/python3.8/site-packages/tensorflow/python/eager/def_function.py␣
 ↪in _call(self, *args, **kwds)
    886         # Lifting succeeded, so variables are initialized and we can ru␣
 ↪the
    887         # stateless function.
--> 888         return self._stateless_fn(*args, **kwds)
    889     else:
    890       _, _, _, filtered_flat_args = \

/opt/conda/lib/python3.8/site-packages/tensorflow/python/eager/function.py in␣
 ↪__call__(self, *args, **kwargs)
   2940       (graph_function,
   2941        filtered_flat_args) = self._maybe_define_function(args, kwargs)
```

```
-> 2942     return graph_function._call_flat(
   2943         filtered_flat_args, captured_inputs=graph_function.
 ↪captured_inputs)  # pylint: disable=protected-access
   2944

/opt/conda/lib/python3.8/site-packages/tensorflow/python/eager/function.py in␣
 ↪_call_flat(self, args, captured_inputs, cancellation_manager)
   1916         and executing_eagerly):
   1917       # No tape is watching; skip to running the function.
-> 1918       return self._build_call_outputs(self._inference_function.call(
   1919           ctx, args, cancellation_manager=cancellation_manager))
   1920     forward_backward = self._select_forward_and_backward_functions(

/opt/conda/lib/python3.8/site-packages/tensorflow/python/eager/function.py in␣
 ↪call(self, ctx, args, cancellation_manager)
    553       with _InterpolateFunctionError(self):
    554         if cancellation_manager is None:
--> 555           outputs = execute.execute(
    556               str(self.signature.name),
    557               num_outputs=self._num_outputs,

/opt/conda/lib/python3.8/site-packages/tensorflow/python/eager/execute.py in␣
 ↪quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
     57   try:
     58     ctx.ensure_initialized()
---> 59     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name,␣
 ↪op_name,
     60                                         inputs, attrs, num_outputs)
     61   except core._NotOkStatusException as e:

InvalidArgumentError:  Incompatible shapes: [0,1] vs. [32,10]
         [[node gradient_tape/binary_crossentropy/logistic_loss/mul/
 ↪BroadcastGradientArgs (defined at <ipython-input-50-dd3cffd2b1a2>:17) ]] [Op:
 ↪__inference_train_function_2517]

Function call stack:
train_function
```

[14]:

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

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-14-3a487bb94be1> in <module>
      1 import matplotlib.pyplot as plt
----> 2 acc = model.history['acc']
      3 val_acc = model.history['val_acc']
      4 loss = model.history['loss']
      5 val_loss = model.history['val_loss']

TypeError: 'NoneType' object is not subscriptable
```

[ ]:

[ ]:

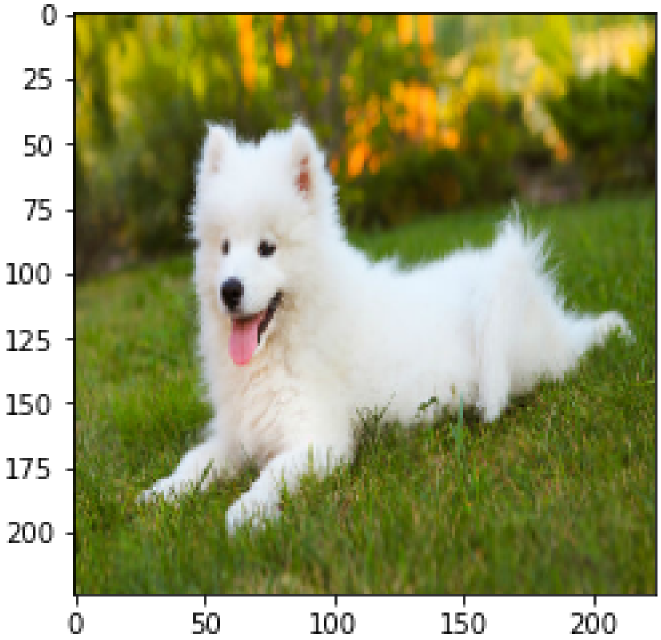
Load the ResNet50 model. Perform image classification on five to ten images of your choice. They can be personal images or publically available images. Include the images in dsc650/assignments/assignment06/images/. Save the predictions dsc650/assignments/assignment06/results/predictions/resnet50 directory. If you are using JupyterHub, you can include those plots in your Jupyter notebook.

[52]:

```
import numpy as np
from keras.preprocessing import image
from keras.preprocessing.image import img_to_array
from keras.applications.resnet50 import preprocess_input
from keras.applications.imagenet_utils import decode_predictions
import matplotlib.pyplot as plt
```

[56]:

```
img = image.load_img('samoyed_puppy_dog_pictures.jpg', target_size = (224, 224))
plt.imshow(img)
img = image.img_to_array(img)
img = np.expand_dims(img, axis=0)
img = preprocess_input(img)
```



[63]:

```
model = keras.applications.ResNet50(weights='imagenet')
preds = model.predict(img)
print('Predicted:', decode_predictions(preds, top=1)[0])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels.h5
102973440/102967424 [==============================] - 5s 0us/step
Downloading data from https://storage.googleapis.com/download.tensorflow.org/dat
a/imagenet_class_index.json
40960/35363 [==================================] - 0s 1us/step
Predicted: [('n02111889', 'Samoyed', 0.9526178)]
```

[ ]: