C1_W3_Lab_3_custom-layer-activation

December 23, 2020

1 Ungraded Lab: Activation in Custom Layers

In this lab, we extend our knowledge of building custom layers by adding an activation parameter. The implementation is pretty straightforward as you'll see below.

1.1 Imports

[1]:
```
try:
    # %tensorflow_version only exists in Colab.
    %tensorflow_version 2.x
except Exception:
    pass

import tensorflow as tf
from tensorflow.keras.layers import Layer
```

1.2 Adding an activation layer

To use the built-in activations in Keras, we can specify an `activation` parameter in the `__init__()` method of our custom layer class. From there, we can initialize it by using the `tf.keras.activations.get()` method. This takes in a string identifier that corresponds to one of the available activations in Keras. Next, you can now pass in the forward computation to this activation in the `call()` method.

[30]:
```
import numpy as np
class SimpleDense(Layer):

    # add an activation parameter
    def __init__(self, units=32, activation=None):
        super(SimpleDense, self).__init__()
        self.units = units

        # define the activation to get from the built-in activation layers in␣
↪Keras
        self.activation = tf.keras.activations.get(activation)
```

```
    def build(self, input_shape):
        w_init = tf.random_normal_initializer()
        self.w = tf.Variable(name="kernel",
            initial_value=w_init(shape=(input_shape[-1], self.units),
                                 dtype='float32'),
            trainable=True)
        b_init = tf.zeros_initializer()
        self.b = tf.Variable(name="bias",
            initial_value=b_init(shape=(self.units,), dtype='float32'),
            trainable=True)
        super().build(input_shape)


    def call(self, inputs):

        # pass the computation to the activation layer
        return self.activation(tf.matmul(inputs, self.w) +
                               np.sum(((np.unique(self.
↪w*10,return_counts=True)[1]/(self.w.shape[0]*self.w.shape[1])))**2))
```

[31]:
```
mnist = tf.keras.datasets.mnist
(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    SimpleDense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

```
      ␣
↪---------------------------------------------------------------------------

       NotImplementedError                       Traceback (most recent call␣
↪last)
```

```
      <ipython-input-31-6e1e31853915> in <module>
        7     SimpleDense(128, activation='relu'),
        8     tf.keras.layers.Dropout(0.2),
  ----> 9     tf.keras.layers.Dense(10, activation='softmax')
       10 ])
       11


      /opt/conda/lib/python3.7/site-packages/tensorflow_core/python/training/
↪tracking/base.py in _method_wrapper(self, *args, **kwargs)
      455     self._self_setattr_tracking = False  # pylint:␣
↪disable=protected-access
      456     try:
  --> 457       result = method(self, *args, **kwargs)
      458     finally:
      459       self._self_setattr_tracking = previous_value  # pylint:␣
↪disable=protected-access


      /opt/conda/lib/python3.7/site-packages/tensorflow_core/python/keras/
↪engine/sequential.py in __init__(self, layers, name)
      114       tf_utils.assert_no_legacy_layers(layers)
      115       for layer in layers:
  --> 116         self.add(layer)
      117
      118   @property


      /opt/conda/lib/python3.7/site-packages/tensorflow_core/python/training/
↪tracking/base.py in _method_wrapper(self, *args, **kwargs)
      455     self._self_setattr_tracking = False  # pylint:␣
↪disable=protected-access
      456     try:
  --> 457       result = method(self, *args, **kwargs)
      458     finally:
      459       self._self_setattr_tracking = previous_value  # pylint:␣
↪disable=protected-access


      /opt/conda/lib/python3.7/site-packages/tensorflow_core/python/keras/
↪engine/sequential.py in add(self, layer)
      201       # If the model is being built continuously on top of an input␣
↪layer:
      202       # refresh its output.
  --> 203       output_tensor = layer(self.outputs[0])
      204       if len(nest.flatten(output_tensor)) != 1:
      205         raise TypeError('All layers in a Sequential model '
```

```
       /opt/conda/lib/python3.7/site-packages/tensorflow_core/python/keras/
↪engine/base_layer.py in __call__(self, inputs, *args, **kwargs)
       771                     not base_layer_utils.
↪is_in_eager_or_tf_function()):
       772                   with auto_control_deps.
↪AutomaticControlDependencies() as acd:
   --> 773                     outputs = call_fn(cast_inputs, *args, **kwargs)
       774                     # Wrap Tensors in `outputs` in `tf.identity` to␣
↪avoid
       775                     # circular dependencies.


       /opt/conda/lib/python3.7/site-packages/tensorflow_core/python/autograph/
↪impl/api.py in wrapper(*args, **kwargs)
       235       except Exception as e:  # pylint:disable=broad-except
       236         if hasattr(e, 'ag_error_metadata'):
   --> 237           raise e.ag_error_metadata.to_exception(e)
       238         else:
       239           raise


       NotImplementedError: in converted code:

       <ipython-input-30-797fc29cdc60>:29 call  *
           return self.activation(tf.matmul(inputs, self.w) +
       <__array_function__ internals>:6 unique

       /opt/conda/lib/python3.7/site-packages/numpy/lib/arraysetops.py:261␣
↪unique
           ar = np.asanyarray(ar)
       /opt/conda/lib/python3.7/site-packages/numpy/core/_asarray.py:138␣
↪asanyarray
           return array(a, dtype, copy=False, order=order, subok=True)
       /opt/conda/lib/python3.7/site-packages/tensorflow_core/python/framework/
↪ops.py:728 __array__
           " array.".format(self.name))

       NotImplementedError: Cannot convert a symbolic Tensor (simple_dense_11/
↪mul:0) to a numpy array.
```

[ ]:
```
[0.07434838510355912, 0.9772]
```

[ ]:
```

```