# pylint: disable=no-member, arguments-differ, attribute-defined-outside-init, unused-argument

"""

Implementing Full Sparse Layer, allow specify use\_binary\_value in call() to

overide default action.

"""

from twml.layers import FullSparse as defaultFullSparse

from twml.layers.full\_sparse import sparse\_dense\_matmul

import tensorflow.compat.v1 as tf

class FullSparse(defaultFullSparse):

def call(self, inputs, use\_binary\_values=None, \*\*kwargs): # pylint: disable=unused-argument

"""The logic of the layer lives here.

Arguments:

inputs:

A SparseTensor or a list of SparseTensors.

If `inputs` is a list, all tensors must have same `dense\_shape`.

Returns:

- If `inputs` is `SparseTensor`, then returns `bias + inputs \* dense\_b`.

- If `inputs` is a `list[SparseTensor`, then returns

`bias + add\_n([sp\_a \* dense\_b for sp\_a in inputs])`.

"""

if use\_binary\_values is not None:

default\_use\_binary\_values = use\_binary\_values

else:

default\_use\_binary\_values = self.use\_binary\_values

if isinstance(default\_use\_binary\_values, (list, tuple)):

raise ValueError(

"use\_binary\_values can not be %s when inputs is %s"

% (type(default\_use\_binary\_values), type(inputs))

)

outputs = sparse\_dense\_matmul(

inputs,

self.weight,

self.use\_sparse\_grads,

default\_use\_binary\_values,

name="sparse\_mm",

partition\_axis=self.partition\_axis,

num\_partitions=self.num\_partitions,

compress\_ids=self.\_use\_compression,

cast\_indices\_dtype=self.\_cast\_indices\_dtype,

)

if self.bias is not None:

outputs = tf.nn.bias\_add(outputs, self.bias)

if self.activation is not None:

return self.activation(outputs) # pylint: disable=not-callable

return outputs