"""

Implementing partition Layer

"""

from .layer import Layer

import tensorflow.compat.v1 as tf

class Partition(Layer):

"""

This layer implements:

.. code-block:: python

tf.dynamic\_partition(input\_vals, partition\_ids, self.partitions)

Input:

partitions:

the number of partitions which we will divide the hashmap keys/bvalues

Output:

A layer that performs partitioning

"""

def \_\_init\_\_(self, partitions=2, \*\*kwargs):

self.partitions = partitions

super(Partition, self).\_\_init\_\_(\*\*kwargs)

def compute\_output\_shape(self, input\_shape):

"""Computes the output shape of the layer given the input shape.

Args:

input\_shape: A (possibly nested tuple of) `TensorShape`. It need not

be fully defined (e.g. the batch size may be unknown).

Raises NotImplementedError.

"""

raise NotImplementedError

def call(self, partition\_ids, input\_vals, input\_keys, \*\*kwargs):

"""This layer is responsible for partitioning the values/keys of a hashmap

Arguments:

partition\_ids:

Tensor that is equivalent to boolean (int32).

input\_vals:

Tensor that represents the values of the hashmap(float).

input\_keys:

Tensor that represents the keys of the hashmap(float)

Returns:

The output of the partition layer, which is a list of lists which looks

something like:

.. code-block:: python

[[vals\_0, vals\_1], [keys\_0, keys\_1], [indices\_0, indices\_1]]

where:

vals\_x:

values of the hashmap for partition x

keys\_x:

keys of the hashmap for partition x

indices\_x:

indices of the hashmap for partition x

"""

partioned\_val = tf.dynamic\_partition(input\_vals, partition\_ids, self.partitions)

partioned\_keys = tf.dynamic\_partition(input\_keys, partition\_ids, self.partitions)

partioned\_indices = tf.dynamic\_partition(tf.range(tf.shape(partition\_ids)[0]),

tf.cast(partition\_ids, tf.int32), self.partitions)

return [partioned\_val, partioned\_keys, partioned\_indices]