import numpy as np

import tensorflow.compat.v1 as tf

TWML\_INIT\_FEED\_KEY = "TWML\_INIT\_FEED\_COLLECTION"

class PartitionConstant(tf.keras.initializers.Constant):

"""A constant initializer that supports partitions"""

def \_\_call\_\_(self, shape, dtype=None, partition\_info=None):

if partition\_info is not None:

if not isinstance(self.value, np.ndarray):

raise ValueError(

"Currently, PartitionConstant only supports "

"partitioning on np.ndarrays. Got {}".format(type(self.value).\_\_name\_\_))

offsets = partition\_info.var\_offset

indices = tuple([slice(offset, offset + size) for offset, size in zip(offsets, shape)])

subset = self.value[indices]

return subset

else:

return self.value

partition\_constant\_initializer = PartitionConstant

class PlaceholderInitializer(tf.keras.initializers.Initializer):

"""A placeholder initializer that supports partitions"""

def \_\_init\_\_(self, shape, dtype):

self.dtype = dtype

self.value = tf.placeholder(dtype=dtype, shape=shape)

def \_\_call\_\_(self, shape, dtype=None, partition\_info=None):

if partition\_info is not None:

if self.dtype != dtype:

raise ValueError("dtype does not match placeholder dtype")

offsets = partition\_info.var\_offset

indices = tuple([slice(offset, offset + size) for offset, size in zip(offsets, shape)])

subset = self.value[indices]

return subset

else:

return self.value

def get\_init\_feed\_dict():

"""Get the init feed dictionary to be used when running the init op."""

# Get the reference to the collection.

init\_feed\_collection = tf.get\_collection(TWML\_INIT\_FEED\_KEY)

init\_feed\_dict = {}

for d in init\_feed\_collection:

init\_feed\_dict.update(d)

return init\_feed\_dict

def clear\_init\_feed\_collection():

"""Clear the init feed collection."""

init\_feed\_collection = tf.get\_collection\_ref(TWML\_INIT\_FEED\_KEY)

while init\_feed\_collection:

init\_feed\_collection.pop()