import tensorflow.compat.v1 as tf

def diag\_mask(n\_data, pairwise\_label\_scores):

"""

This is so far only used in pariwise learning-to-rank

Args:

n\_data: a int `Tensor`.

pairwise\_label\_scores: a dense `Tensor` of shape [n\_data, n\_data].

Returns:

values in pairwise\_label\_scores except the diagonal

each cell contains a paiwise score difference

only selfs/diags are 0s

"""

mask = tf.ones([n\_data, n\_data]) - tf.diag(tf.ones([n\_data]))

mask = tf.cast(mask, dtype=tf.float32)

pair\_count = tf.to\_float(n\_data) \* (tf.to\_float(n\_data) - 1)

pair\_count = tf.cast(pair\_count, dtype=tf.float32)

return mask, pair\_count

def full\_mask(n\_data, pairwise\_label\_scores):

"""

This is so far only used in pariwise learning-to-rank

Args:

n\_data: a int `Tensor`.

pairwise\_label\_scores: a dense `Tensor` of shape [n\_data, n\_data].

Returns:

values in pairwise\_label\_scores except pairs that have the same labels

each cell contains a paiwise score difference

all pairwise\_label\_scores = 0.5: selfs and same labels are 0s

"""

not\_consider = tf.equal(pairwise\_label\_scores, 0.5)

mask = tf.ones([n\_data, n\_data]) - tf.cast(not\_consider, dtype=tf.float32)

mask = tf.cast(mask, dtype=tf.float32)

pair\_count = tf.reduce\_sum(mask)

pair\_count = tf.cast(pair\_count, dtype=tf.float32)

return mask, pair\_count