from typing import Callable

import tensorflow as tf

class WarmUp(tf.keras.optimizers.schedules.LearningRateSchedule):

def \_\_init\_\_(

self,

initial\_learning\_rate: float,

decay\_schedule\_fn: Callable,

warmup\_steps: int,

power: float = 1.0,

name: str = "",

):

super().\_\_init\_\_()

self.initial\_learning\_rate = initial\_learning\_rate

self.warmup\_steps = warmup\_steps

self.power = power

self.decay\_schedule\_fn = decay\_schedule\_fn

self.name = name

def \_\_call\_\_(self, step):

with tf.name\_scope(self.name or "WarmUp") as name:

global\_step\_float = tf.cast(step, tf.float32)

warmup\_steps\_float = tf.cast(self.warmup\_steps, tf.float32)

warmup\_percent\_done = global\_step\_float / warmup\_steps\_float

warmup\_learning\_rate = self.initial\_learning\_rate \* tf.math.pow(

warmup\_percent\_done, self.power

)

return tf.cond(

global\_step\_float < warmup\_steps\_float,

lambda: warmup\_learning\_rate,

lambda: self.decay\_schedule\_fn(step - self.warmup\_steps),

name=name,

)

def get\_config(self):

return {

"initial\_learning\_rate": self.initial\_learning\_rate,

"decay\_schedule\_fn": self.decay\_schedule\_fn,

"warmup\_steps": self.warmup\_steps,

"power": self.power,

"name": self.name,

}