# slim.batch\_norm使用注意事项

1. 添加依赖：

update\_ops = tf.get\_collection(tf.GraphKeys.UPDATE\_OPS)  
#使用依赖,保证每次训练之前要更新batch norm的均值和方差，不然后面会导致训练时精度高,但是测试时精度极低  
with tf.control\_dependencies([tf.group(\*update\_ops)]):  
 train\_step = optimizer.minimize(loss, global\_step)

1. 均值方差warm up

one\_moving\_meaning=sess.graph.get\_tensor\_by\_name(conv\_1/batch\_norm/moving\_mean:0')  
# 大坑2！！！！  
#除了使用依赖更新均值和方差,还要等待均值和方差warm up到稳定的程度才能算ok  
#这里可以使用某个均值张量的均值来查看是否已经稳定(即均值不会一直增长,而是在某个固定值附近震荡)  
#同时为了加速warm up,可以设定batch norm的decay=0.95(默认0.999)  
print("one\_moving\_meaning:",np.mean(one\_moving\_meaning.eval()))

# 解决方案:

1. **无placeholder下的训练:**

训练过程中，模型中slim.batch\_norm的is\_training参数直接置为True

-->测试:

1. CKPT模型: ----精度正常

再一次通过代码加载模型,此时将模型中is\_training参数直接置为False,并加载之前训练好的ckpt模型和参数,直接进行测试。

1. PB模型: ----精度异常

因为PB模型是被冻结的,所以无论通过直接获得pb中模型中节点进行测试,还是像ckpt一样通过重新加载模型并将is\_training置为False进行测试,都没办法从根本上将is\_training置为False,进而会导致测试精度和训练精度相差极大。

1. **有placeholder下的训练 (推荐)：**

将is\_training使用tf.placeholder\_with\_default进行指定,并设置默认值为False。

is\_training\_placeholder=tf.placeholder\_with\_default(False,shape=(),name="is\_training")

-->测试:

1. CKPT模型： ----精度正常

无需重新通过代码定义和加载模型并将模型中is\_training参数直接置为False，直接在sess.run函数中将is\_training\_placeholder置为False即可。

如下：

\_inputs = tf.get\_default\_graph().get\_tensor\_by\_name('inputs:0')  
\_is\_training = tf.get\_default\_graph().get\_tensor\_by\_name('is\_training:0')  
# \_classes = tf.get\_default\_graph().get\_tensor\_by\_name('classes:0')  
\_softmax\_output = tf.get\_default\_graph().get\_tensor\_by\_name('softmax\_output:0')

softmax\_out,pred\_class=sess.run([\_softmax\_output,\_classes],feed\_dict=  
 {\_inputs:image\_data,\_is\_training:False})

1. PB模型： ----精度正常

同”有placeholder下的训练->CKPT模型测试”

PS:同时带有tf.placeholder\_with\_default的训练方式，默认is\_training=False,所以在使用openVino的模型优化器转换pb模型时也不会报错(FusedBatchNorm doesn't support is\_training=True)。