1. 对**保存模型**进行**冻结操作**的注意事项：

要想对保存模型进行冻结操作，建议使用 [结构保存(model.to\_json()) + 权值保存(model.save\_weights)] 这种方法保存模型。

原因：

采用 model.save() 和 load\_model()的方法得到的模型，在做冻结操作时候会发生权重错误。

采用 [ 结构保存(model.to\_json()) + 权值保存(model.save\_weights) ]，在做冻结操作时候就不会发生权重错误

1. 如果网络层的定义部分：

y = Dense(units=128, activation='relu', kernel\_initializer='he\_normal', trainable=False)(y)

定义了trainable=False，那么就不能通过model.trainable = True 来改变这一层的'冻结状态';

但可以通过model.layers[i].trainable=True 的方法来改变：

1. 查看可训练(trainable)和不可训练(non\_trainable)的权值方法：

方法：model.trainable\_weights (可训练权值)

print('参与训练的权值名称：')

for x in model.trainable\_weights:

print(x.name)

print('\n')

方法：model.non\_trainable\_weights (不可训练权值)

print('不参与训练的权值名称：')

for x in model.non\_trainable\_weights:

print(x.name)

print('\n')

参考文献来自于CSDN blog:

<https://blog.csdn.net/jdzwanghao/article/details/80697104>

解决transfer learning 过程中对BN 层的冻结问题：

参考文献来自于：

<https://zhuanlan.zhihu.com/p/56225304>

显示设置：

x = BatchNormalization()(y, training=False)

隐式设置：

# Set up inference-mode base

K.set\_learning\_phase(0)

inputs = Input(...)

x = layer1(...)(inputs)

x = layer2(...)(x)

...

x = layerN(...)(x)

# Add training-mode layers

K.set\_learning\_phase(1)

x = layerNp1(...)(x)

x = layerNp2(...)(x)

Keras 2.2.x learning\_phase 机制源码级解读： <https://www.jianshu.com/p/a7edf79aff0e> 这个参考文献可以更好的理解隐式设置的运行机制，建议一并阅读。