1. tf.concat()
2. tf.gather()类似于数组的索引，可以把向量中某些索引值提取出来，得到新的向量，适用于提取不连续的内容，如果用于提取二维数组，表示提取索引对应的行向量。
3. tf.gather\_nd()允许在多维上进行索引。如果方括号中有一个数，表示提取整行，如果方括号中有两个数，表示对应的元素。若对应tensor为三维，则分别对应第几个矩阵，该矩阵中的第几行，该矩阵中的第几列。
4. tf.greater()和tf.greater\_equal()逐元素进行判断
5. tf.cast()转换数据类型
6. tf.expand\_dims()和tf.squeeze()增加、压缩张量的维度：一维数组转二维矩阵，二维矩阵转一维数组。
7. 如果在创建Session时没有指定Graph，则该Session会加载默认Graph，如果在一个进程中创建了多个Graph，则需要创建不同的Session来加载每个Graph， 而每个Graph则可以加载在多个Session中计算。
8. 调用Operation.run()或则Tensor.eval()方法： 这两个方法都接收参数session，用于指定在哪个session中计算。但该参数是可选的，默认为None，此时表示在进程默认session中计算。
9. 如何设置一个Session为默认的Session：在with语句中调用Session.as\_default()方法
10. 如果我们需要定义多个Graph，则需要在with语句中调用Graph.as\_default()方法将某个graph设置成默认Graph，于是with语句块中调用的Operation或Tensor将会添加到该Graph中。
11. Operation对象的创建是通过直接调用Python operation方法（例如tf.matmul()）或者Graph.create\_op()。
12. 当一个Graph加载到一个Session中，则可以调用Session.run(op)来执行op，或者调用op.run()来执行（op.run()是tf.get\_default\_session().run()的缩写）。
13. Tensor表示的是Operation的输出结果。不过，Tensor只是一个符号句柄，其并没有保存Operation输出结果的值。通过调用Session.run(tensor)或者tensor.eval()方可获取该Tensor的值。
14. 当Session加载Graph的时候，Graph里面的计算节点都不会被触发执行。当运行sess.run(output)的时候，会沿着指定的Tensor output来进图路径往回触发相对应的节点进行计算。所以在计算Graph时，并不一定是Graph中的所有节点都被计算了，而是指定的计算节点或者该节点的输出结果被需要时。
15. 当我们基于checkpoint文件(ckpt)加载参数时，实际上我们使用Saver.restore取代了initializer的初始化，checkpoint文件会记录保存信息，通过它可以定位最新保存的模型：.meta文件保存了当前图结构。.index文件保存了当前参数名。.data文件保存了当前参数值。
16. tf.sign(x, name=None)
17. name\_scope和variable\_scope
18. name\_scope:为了更好地管理变量的命名空间而提出的。
19. variable\_scope:跟tf.get\_variable()配合使用，实现变量共享的功能
20. tf.name\_scope() 并不会对 tf.get\_variable() 创建的变量有任何影响。
21. tf.name\_scope() 主要是用来管理命名空间的，这样子让我们的整个模型更加有条理。而 tf.variable\_scope() 的作用是为了实现变量共享，它和 tf.get\_variable() 来完成变量共享的功能。
22. tf.trainable\_variables()获取所有的可训练变量
23. tf.constant\_initializer()
24. 使用tf.get\_variable()定义的变量不会被tf.name\_scope()当中的名字所影响.
25. 如果想要达到重复利用变量的效果, 我们就要使用 tf.variable\_scope(), 并搭配 tf.get\_variable() 这种方式产生和提取变量. tf.get\_variable() 如果遇到了同样名字的变量时, 它会单纯的提取这个同样名字的变量(避免产生新变量). 而在重复使用的时候, 一定要在代码中强调 scope.reuse\_variables(), 否则系统将会报错.
26. Tensorflow中name\_scope和variable\_scope的理解:之所以会出现这两种类型的scope，主要是后者（variable scope）为了实现TensorFlow中的变量共享机制：即为了使得在代码的任何部分可以使用某一个已经创建的变量，tf引入了变量共享机制，使得可以轻松地共享变量，而不用传一个变量的引用。
27. 命名域(name scope)：通过tf.name\_scope()来实现
28. 变量域（variable scope）：通过tf.variable\_scope()来实现；可以通过设置reuse 标志以及初始化方式来影响域下的变量。
29. 这两种作用域都会给tf.Variable()创建的变量加上词头，而tf.name\_scope对tf.get\_variable()创建的变量没有词头影响
30. 当 reuse 设置为 True 或者 tf.AUTO\_REUSE 时，表示这个scope下的变量是重用的或者共享的，也说明这个变量以前就已经创建好了。但如果这个变量以前没有被创建过，则在tf.variable\_scope下调用tf.get\_variable创建这个变量会报错。
31. 如果使用tf.get\_variable()创建变量，且没有设置共享变量，重名时会报错
32. tf.Variable.init(initial\_value, trainable=True, collections=None, validate\_shape=True, name=None)
33. tf.train.get\_checkpoint\_state(checkpoint\_dir, latest\_filename = None) 该函数返回的是checkpoint文件CheckpointState proto类型的内容，其中有model\_checkpoint\_path和all\_model\_checkpoint\_paths两个属性。其中model\_checkpoint\_path保存了最新的tensorflow模型文件的文件名，all\_model\_checkpoint\_paths则有未被删除的所有tensorflow模型文件的文件名。
34. tf.Summary所有需要在TensorBoard上展示的统计结果。tf.name\_scope()层次化展示；tf.summary.scalar()；tf.summary.histogram()；tf.summary.merge\_all()；tf.summary.FileWrite
35. tf.trainable\_variables()返回的是需要训练的变量列表

tf.all\_variables()返回的是所有变量的列表

1. tf.train.batch是按顺序读取数据，队列中的数据始终是一个有序的队列。tf.train.shuffle\_batch是将队列中数据打乱后，再读取出来，因此队列中剩下的数据也是乱序的，队尾也是一直在补充。capacity是队列的长度。min\_after\_dequeue是出队后，队列至少剩下min\_after\_dequeue个数据
2. tf.train.slice\_input\_producer([image,label],num\_epochs=10),随机产生一个图片和标签,num\_epochs=10,则表示把所有的数据过10遍。这个函数只可以加载图片的路径，放入图片的path，path必须是一个list或者tensorlist
3. 使用tf.train.slice\_input\_producer([path]),也可以批量读取图片，得到每个图片的路径后，我们可以加载图片并解码成三维数组的形式（图像的深度必须是3通道或者4通道，笔者实验灰度图像，一直不成功）。当使用tf.train.slice\_input\_producer()时，加载图片数据的reader使用tf.read\_file(filename),直接读取。
4. tf.train.string\_input\_producer(path),传入路径时，不需要放入list中。然后加载图片的reader是tf.WholeFileReader(),其他地方和tf.train.slice\_input\_producer()函数用法基本类似。