Fine-tune on pre-trained models(caffe)

在一些特定任务中我们很难拿到大量的数据，在这种情况下重新训练一个新的效果好的网络是不容易的，而且参数不好调整，这时候使用fine tuning是一个明智的选择。另外，即使有大量的数据如果使用finetuning技术往往也会得到比不用finetuning更好的效果。

所谓finetuning就是用别人在大含有大量数据的数据集上训练好的模型（比如使用ImageNet数据训练的大量Model-Zoo）的基础上，加上自己的数据，来训练新的模型。finetuning的好处在于不用完全重新训练模型，从而提高效率，因为一般新训练模型准确率都会从很低的值开始慢慢上升，但是finetuning能够让我们在比较少的迭代次数之后就能得到一个比较好的效果，甚至在做第一次test的时候精度就比较高。

在进行finetuning时的trick：如果我们的数据和预训练模型的数据非常类似，那么只需要微调后面的少数层；如果我们的数据和预训练模型的数据有较大不同，那么需要调整很多层甚至是整个网络。更详细的介绍请戳图片的链接。

	very similar dataset	very different dataset
very little data	Use linear classifier on top layer	You're in trouble... Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a large number of layers

在进行finetuning时需要修改的内容：

1. train_val.prototxt
    a. 对于直接使用预训练模型的参数的层，将其学习率调为0；
    b. 对于需要微调的层，将其学习率调小；
    c. 删除除了分类层（很多时候是最后一个全连接层）之外其它层的初始化（ weight_filler {type: "xavier" } bias_filler {type: "constant" }）；
    d. 对于BatchNorm层需要将use_global_stats的值置为true（BatchNorm层的use_global_stats参数，只有在使用原始数据从零开始训练的时候值是false，在进行finetuning的时候值是true，在test的时候值也是true）；
    e. 将分类层的输出类别数目改为我们任务所需要的类别数目；
    f. 将分类层的名字改掉（预训练模型是根据网络层的名字对参数进行赋值的，改名字之后预训练模型在赋值的时候就会因为名字不匹配而重新训练，也就达到了重新训练分类层的目的，从而适应我们的任务）。
2. solver.prototxt(主要是相对于预训练模型的solver进行的修改)
    a. test_iter改小，因为数据量减少了；
    b. base_lr改小，进行finetuning的时候需要调小学习率。

举例说明：

train_val prototxt compare report(SqueezeNet-Finetuning).html（SqueezeNet的预训练模型）（注意：看比较报告的时候不需要关注data层，只是预训练模型使用的数据格式是lmdb我们使用的是原始图片、同时我们使用了数据增强而已。）

solver.prototxt compare report(SqueezeNet-Finetuning).html（注意：该solver.prototxt不是相对于预训练模型的solver.prototxt改的，因为所用的优化方法不同，这里只体现出降低了学习率）。