Week 5 Report

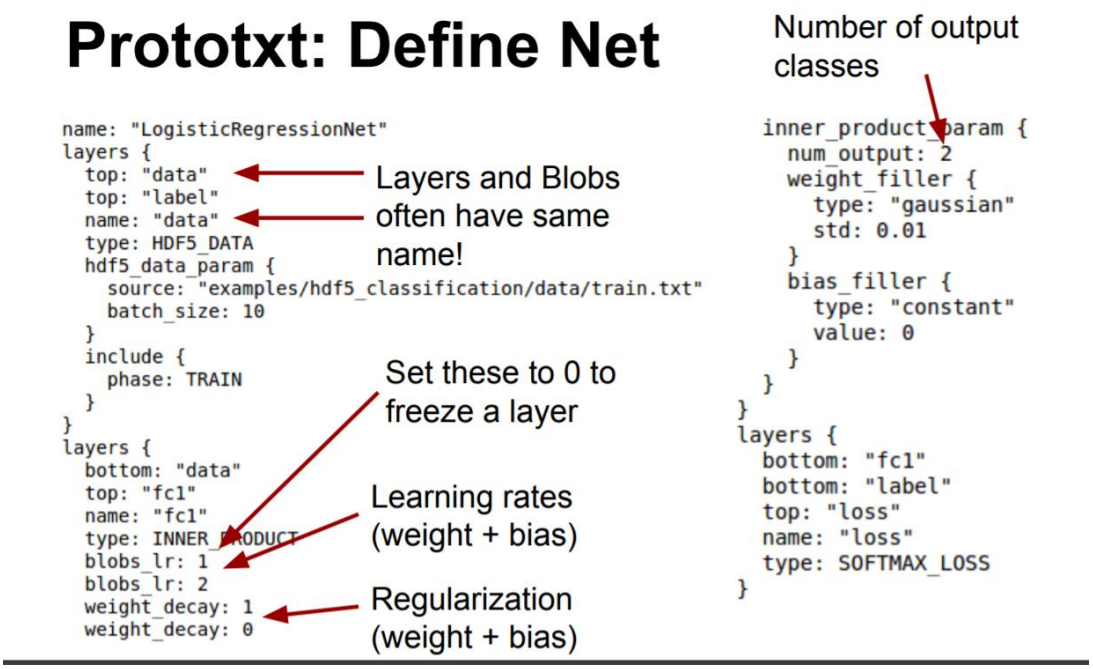
Qian Jiang

1. Notes about caffe
   a) 特点
      - Pure C++ / CUDA architecture for deep learning o command line, Python, MATLAB interfaces
      - Fast, well-tested code
      - Tools, reference models, demos, and recipes
      - Seamless switch between CPU and GPU
        - Caffe::set_mode(Caffe::GPU);
   b) 基本组成：
      - Blob: Stores data and derivatives
      - Layer: Transforms bottom blobs to top blobs
      - Net: Many layers; computes gradients via forward / backward
      - Solver: Uses gradients to update weights
   c) Protocol Buffer：一种类似于 XML 的用于序列化数据的自动机制。用来对数据进行序列化和反序列化。
      （序列化（Serialization）：将数据结构或对象转换成二进制串的过程。反序列化（Deserialization）：将在序列化过程中所生成的二进制串转换成数据结构或者对象的过程。）
      Define message types in .proto file
      Define messages in .prototxt or .binaryproto files (Caffe also uses .caffemodel)
   d) 数据格式：Lmdb 和 Leveldb
      （LMDB 采用键值对<key-value>的存储格式，key 就是字符形式的 ID，value 是 Caffe 中 Datum 类的序列化形式。）
   e) 基本步骤：
      1. Convert data
      2. Define net (as prototxt)
      3. Define solver (as prototxt)
      4. Train (with pretrained weights)

Reference:

http://caffe.berkeleyvision.org/
http://vision.stanford.edu/teaching/cs231n/slides/2015/caffe_tutorial.pdf
http://graphics.cs.cmu.edu/courses/16-824/2016_spring/slide

s/caffe_tutorial.pdf
https://www.csdn.net/article/2015-01-22/2823663
https://github.com/BVLC/caffe/blob/85bb397acfd383a676c125c75d877642d6b39ff6/src/caffe/proto/caffe.proto



每一层前四行层属性，后 xxx_param 为层参数。

2. HWGQ Codes tf version

```
def
get_hwgq(bitA)
:
```

```
    def quantize(x, k):
        # in order of
        #不同阶数
        assert k in [2,3,4,5], 'Does not support %d
bits' % k
        code_book={
        '2':[0.5380, 0., 0.5380*(2**2-1)],
        '3':[0.3218, 0., 0.3218*(2**3-1)],
        '4':[0.1813, 0., 0.1813*(2**4-1)],
        '5':[0.1029, 0., 0.1029*(2**5-1)]
        }
```

```
            delta, minv, maxv = code_book[str(k)]
            #print(delta,minv,maxv)
            @tf.custom_gradient
            def _quantize(x):
                return
#quantize 且限制为 float32 类型,对应论文中
```

$$Q(x) = \begin{cases} q_i, & \text{if } x \in (t_i, t_{i+1}], \\ 0, & x \leq 0, \end{cases}$$

```
tf.to_float(x>0.)*(tf.clip_by_value((tf.floor(x/del
ta + 0.5)+tf.to_float(x<0.5*delta))*delta, minv,
maxv)),
#计算 dy
lambda dy:
dy*tf.to_float(x>minv)*tf.to_float(x<maxv)

            return _quantize(x)

        def fa(x):
            if bitA == 32:
                return x

            return quantize(x, bitA)
        return fa
```

3. Batch Normalization

   Key:输入先做归一化处理，然后再进入网络的下一层，从而避免数据分布不均的问题。(归一化：即均值为 0，方差为 1)

Problem: Internal Covariate Shift→1). lower learning rates and 2).careful parameter initialization 3).hard to train models with saturating nonlinearities.

Solution: BN for each training mini-batch

Pros:1) higher learning rates 2) less careful about initialization 3)eliminating the need for Dropout