Weekly Report

July 20, 2017

1 2017.6.16

1.1 用LeNet训练基元草图

重新训练网络
1.六边形画成五边形1个，基本不像两个。
2.图片缩小时，采用区域差值方式。区域插值是opencv中提供的5种resize差值方式实验效果最好的差值方式。

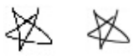

Figure 1: 左图为其他插值方式，右图为区域差值

重新训练网络，得到的识别准确率为95%。**UI Demo**

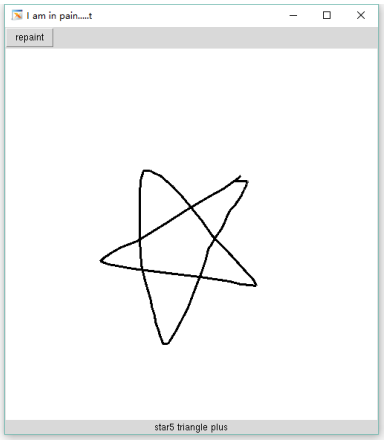

Figure 2: UI

2 2017.6.18

由于基元存在旋转的情况，不适合把基元做average。
故找出当前画的基元的fc1层feature vector 和训练集中基元的fc1 feature vector最相近的前3个作为提示给出。下图是一个demo

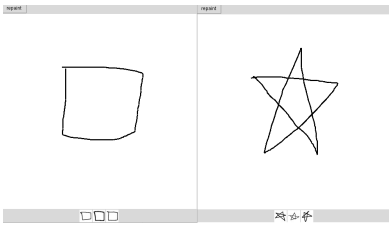

Figure 3: 主区域是当前画的基元，底下三个从训练集找出最相近的三个基元

3 2017.6.23

在网络上加dropout层，识别的准确率稳定在95.4%。
将圆形，五边形，三角形，梯形，矩形，五角星，六边形，平行四边形旋转90，180，270度，其余基元样本直接复制成原来的四倍。得到总共20000个样本，训练迭代30000次，识别准确率为94.9%。
UI demo
画一个基元，从训练样本找出和该基元在fc2 特征最像的10个样本，并且按照特征距离远近，获得10个样本的权值，再将10个图片相加，获得shadow图。

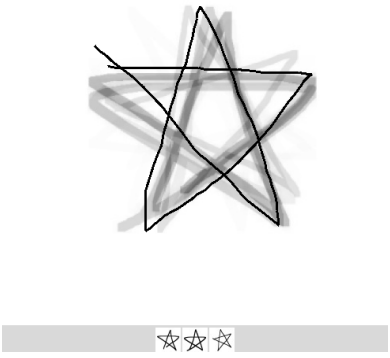

Figure 4: 五角星

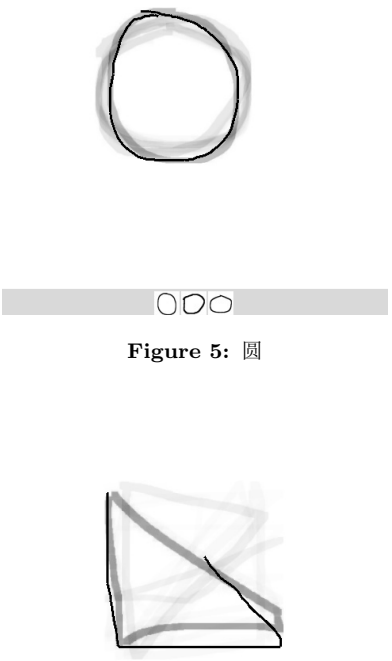

Figure 5: 圆

Figure 6: 三角形

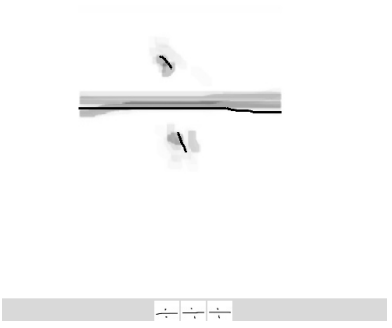

Figure 7: 除

To Do:
从3340训练样本中找到距离当前画的基元10个最近的样本比较耗时，约2到3秒时间，考虑构建树结构或其他数据机构去加快搜索。

4 2017.6.30

调研数据检索的一些方法，一部分类似于KD-Tree对数据空间进行划分，但这类方法一般超过10维，计算效率就比较低。还有一类是以lsh(Locality-Sensitive Hashing)为基础的方法，检索速度较快，对于空间距离越近，被检索到的概率就越大，越远则越小。

实验一：
实验lsh的检索方法，非常的不稳定，部分图像检索不到最近邻，部分图像检索的最近邻为10个以下。

实验二：
shadow图片大部分来自识别分数最高的前3类，故只从识别分数最高的前3类选取10张最近邻(计算fc2特征间的欧式距离)。feature vector虽然20维，但是能量分布主要集中在识别分数最高的3个维度。找到10个最近邻，10个最近邻的权值按照fc2特征间的欧式距离来算，搜索时间为8ms-12ms, 优化选取10 个近邻的过程(原来是遍历整个训练集10遍，找10个最近邻，而现在实现是维护10个最近邻有序序列，只对训练集遍历一遍)。

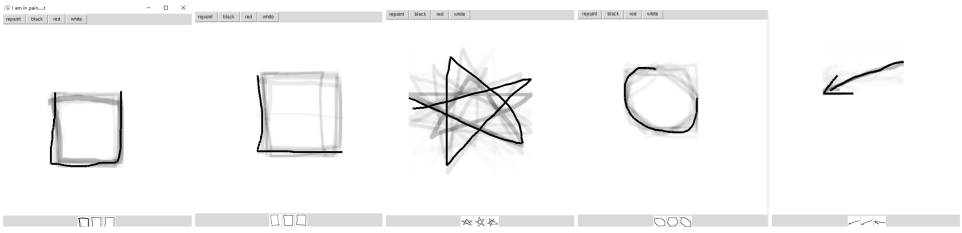

Figure 8: 实验结果

实验三：
在识别分数最高的三个类别中计算fc2特征的欧式距离，找出10个最近邻，10个最近邻权值的计算是按照fc2特征间的欧式距离来算，整个计算时间在40ms左右

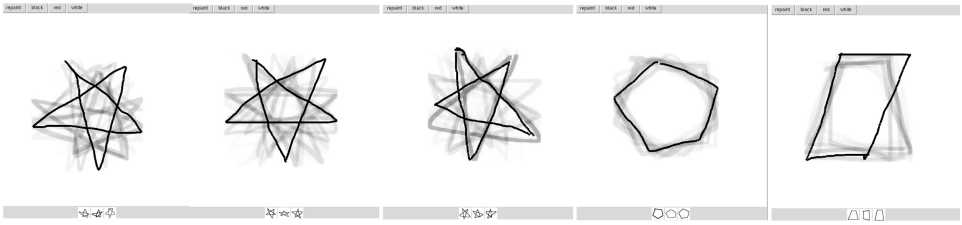

Figure 9: 实验结果

实验四：
在识别分数最高的三个类别中计算fc2特征的欧式距离，找出10个最近邻，10个最近邻权值的计算是按照fc1特征间的欧式距离来算，fc1的特征更加lowlevel,整个计算时间在70ms左右

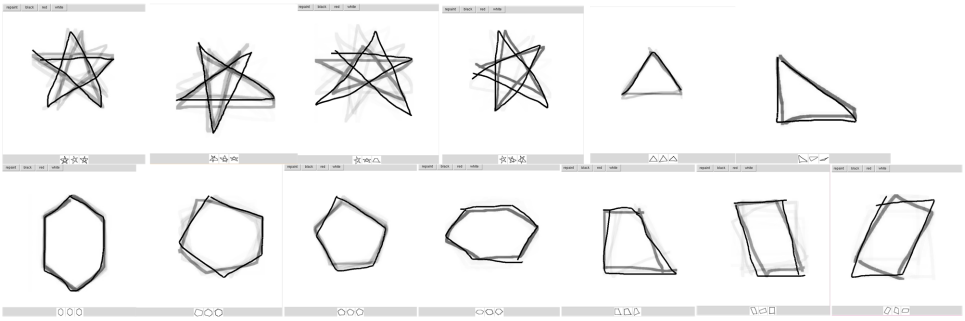

Figure 10: 实验结果

5 2017.7.7

ShortStraw 找角点
没有语义信息的知道，ShortStraw算法往往很难找出我们需要的角点。如下图。

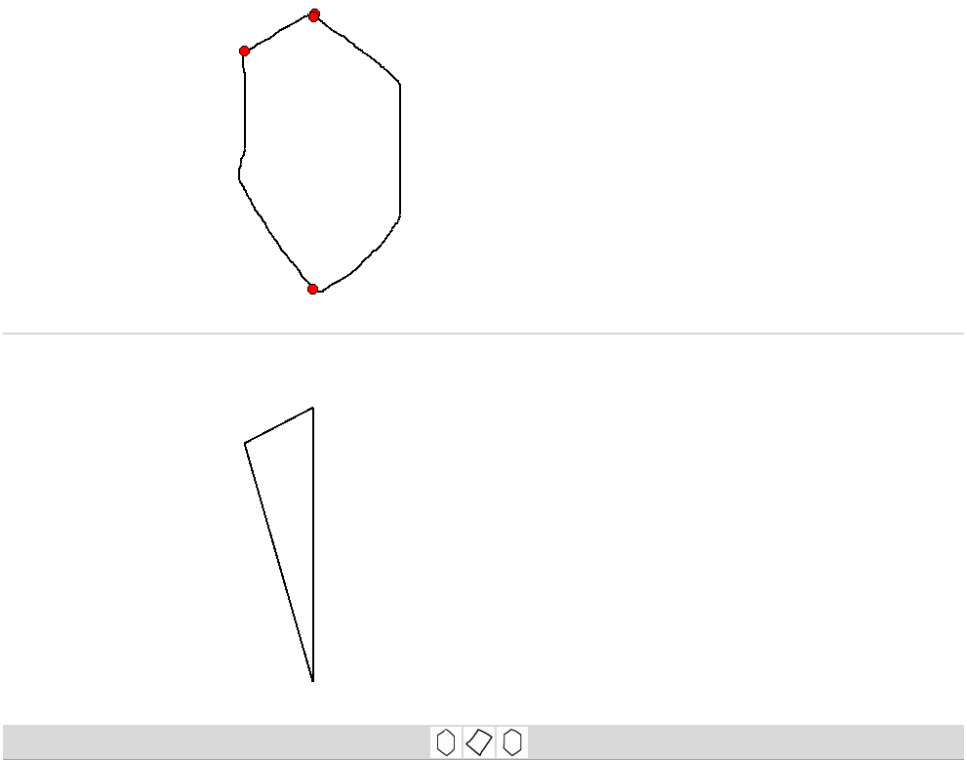

Figure 11: 实验结果

通过对基元的识别。能够从high level 上ShortStraw要找出多少个角点，如下图。

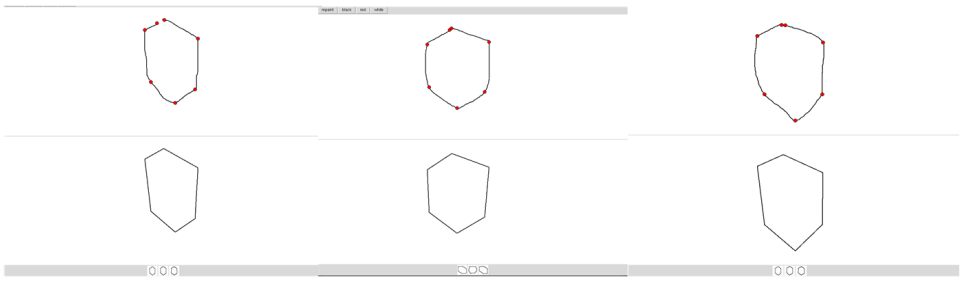

Figure 12: 实验结果

构建流程图。

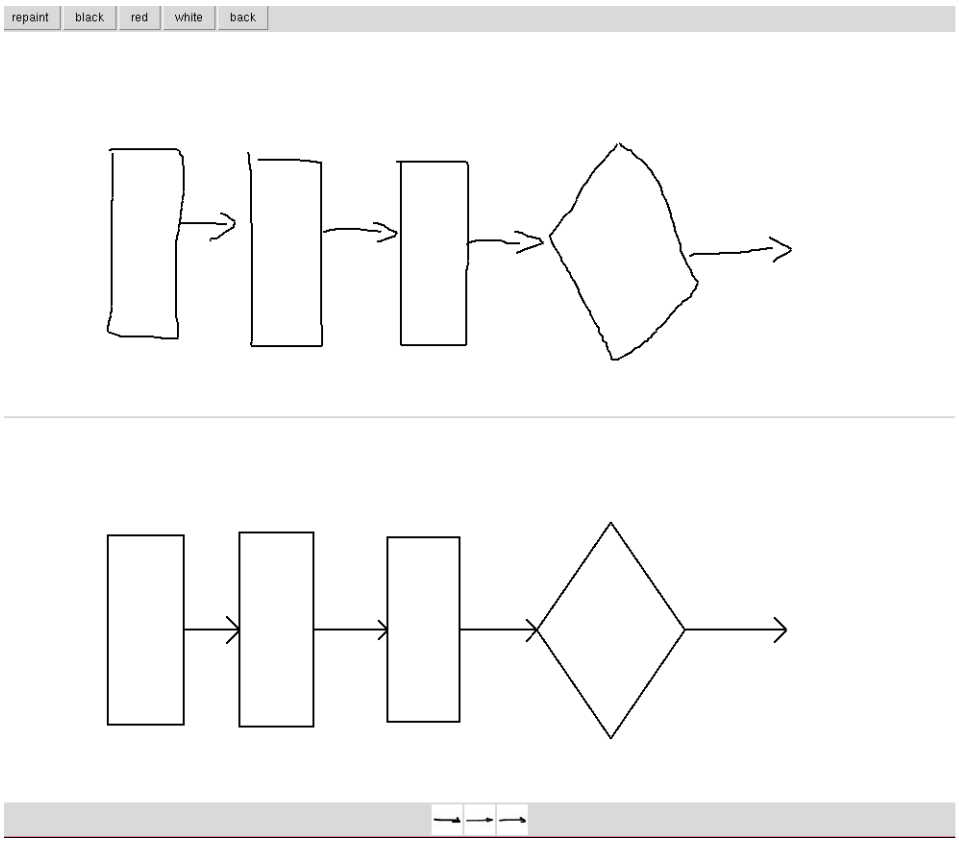

Figure 13: 实验结果

6 2017.7.20

重新整理数据14类，分别如下：1.圆形2.菱形3.下箭头4.上箭头5.上线双箭头6.左箭头7. 右箭头8.左右双箭头9.水平线10.竖直线11.平行四边形12.梯形13.三角形14.矩形。
前数据部分基元情形覆盖不全，如矩形类中，竖长形的矩形比较少，平行四边形中某些类型也比较缺乏。重新加入情形，丰富类内基元的情形。

References