毕业设计记录

## 目标：

借鉴RCNN算法的思路实现图像内物体Bounding Box的确定，网络结构采用AlexNet或者VGGNet，使用的数据集为Pascal Voc，并制作一个网页前端用于展示。

## 环境：

* 1. GPU：GTX 960M 4G
  2. CPU：i5-6300HQ
  3. 内存：8G

系统：Ubuntu 18.04.1 LTS

CUDA：V9.0.176

NVIDIA驱动：390.87

Python：2.7

数据集：Pascal VOC 2012

MXNet：1.3.1

没有设置cudNN。

## 关于数据集：

* 1. 一共含有17125张图片。

ImageSets文件夹中：

其中Action下存放的是人的动作（例如running、jumping等等）

Layout下存放的是具有人体部位的数据（人的head、hand、feet等等）

Main下存放的是图像物体识别的数据，总共分为20类。

Segmentation下存放的是可用于分割的数据。

主要使用Main文件夹中的用来训练。

## 实现的大致思路：

* 1. 对于一张图片，使用选择性搜索（Selective Search）会产生一些候选框，再对这些候选框进行简单的筛选后（除去长或宽小于32的框），把每个候选框都重新resize后输入到训练好的AlexNet中得到输出分类以及得分，再进行非极大值抑制（NMS）之后把该候选框在图片上画出，并展示在网页前端。
  2. 对于训练，根据Annotations文件夹中的xml文件，把JPEGImages中的每张图片中的每个object都裁剪出来并resize到AlexNet的标准输入尺寸（227x227），把这些图片制作成RecordIO数据对AlexNet进行训练。

## 生成训练用数据集：

首先，根据Annotations文件夹中的xml文件，把JPEGImages中的图片中的object都裁剪出来并resize到AlexNet的标准输入尺寸（227x227）保存在ResizedObjects文件夹中，这部分工作由DataSetObjectExtraction.py完成。

然后，使用产生随机数的方式将这些图片以7:2:1的方式划分为训练集、验证集和测试集，这部分工作由GenerateImgList.py完成，生成的lst文件存放在Data/RecordIO中。

再然后，用mxnet自带的im2rec.py生成rec文件，在控制台中输入如下命令即可：sudo python utils/im2rec.py --num-thread=4 Data/RecordIO/train.lst .。

至此，就完成了数据集的制作。

## 搭建网络并训练：

参考AlexNet标准网络结构，并根据Pascal VOC数据集一共有20个类别的特性，将第三个全连接层的单元数设置为20。

迭代器ImageRecordIter设置的参数有：

data\_shape=(3, 227, 227),

batch\_size=50,

mean\_r = 128,

mean\_g = 128,

mean\_b = 128

其他为默认。

训练使用的学习策略为MXNet中默认的FactorScheduler，参数为每500次迭代学习率衰减为原来的0.95。

训练总epoch为40，每10个epoch保存一次模型。

搭建的网络结构图为AlexNet.gv.pdf。

## 迷你数据集训练：

在进行大数据集训练之前，应准备一个迷你数据集先测试一下，因此取了训练集中的前30个图片和验证集的前20个图片作为迷你的训练集和验证集。主要是为了验证网络结构的正确性，对于如此小的数据集，应当能在没有正则化的情况下轻易达到过拟合。

## 训练记录：

1.训练总用时5835.39844489s，Train-accuracy一直保持在0.43左右，Train-cross-entropy一直保持在2.27–2.29之间，Train-mse一直保持在75.3多，Validation的也差不多，也是基本保持不变。

## 测试和评估：

接下来要对训练过后的模型进行测试和评估。

首先是测试，这里使用之前分出来的test测试集中的数据进行测试，载入的参数为训练时保存下来的最后一组参数，也就是编号为35的params文件，使用的指标为accuracy，即准确率。

## 错误记录：

1.利用list文件产生rec文件时发生的错误：

hongyigeng@hongyigeng-GL552VW:~/PycharmProjects/ImageSearch$ sudo ~/mxnet/bin/im2rec train.lst Data/ train.rec

/home/hongyigeng/mxnet/bin/im2rec: error while loading shared libraries: libcudart.so.9.1: cannot open shared object file: No such file or directory

检查过环境变量没有问题。

尝试以下3条命令：

$ sudo cp /usr/local/cuda/lib64/libcudart.so.9.0 /usr/local/lib/libcudart.so.9.0 && sudo ldconfig

$ sudo cp /usr/local/cuda/lib64/libcublas.so.9.0 /usr/local/lib/libcublas.so.9.0 && sudo ldconfig

$ sudo cp /usr/local/cuda/lib64/libcurand.so.9.0 /usr/local/lib/libcurand.so.9.0 && sudo ldconfig

无效

解决办法：

使用mxnet/tools文件夹内的im2rec.py即可

sudo python ~/mxnet/tools/im2rec.py --num-thread=4 train.lst .

2.运行训练脚本时发生错误：

RuntimeError: simple\_bind error. Arguments:

data: (50, 3L, 227L, 227L)

softmax\_label: (50,)

[19:36:47] src/storage/storage.cc:119: Check failed: e == cudaSuccess || e == cudaErrorCudartUnloading CUDA: unknown error

尝试重装CUDA9.1以替换CUDA9.0：

首先卸载：sudo /usr/local/cuda/bin/uninstall\_cuda\_9.0.pl

运行安装程序：sudo sh cuda\_9.1.85\_387.26\_linux.run –override

安装之后试图运行CUDA-examples中的deviceQuery出错，显示：

./deviceQuery Starting...

CUDA Device Query (Runtime API) version (CUDART static linking)

cudaGetDeviceCount returned 30

-> unknown error

Result = FAIL

解决办法：

重启。。。

3.import mxnet的时候出现了OSError libcudart 9.0什么的错误，直接卸载原来的mxnet装上mxnet的GPU版本mxnet-cu91

4.运行训练脚本的时候出现如下错误：

Check failed: static\_cast<index\_t>(res.rows) >= param\_.data\_shape[1] && static\_cast<index\_t>(res.cols) >= param\_.data\_shape[2] input image size smaller than input shape

应该是数据集中的有些图片尺寸小于AlexNet的输入图片尺寸要求277\*277。

解决办法就是要把数据集中的图片全部都resize一下。

## 问题记录：

1.训练出来的模型的准确率太低，测试了测试集和验证集，准确率都只有40%。查看了Annotations中的xml文件，发现很多图片标注出来的物体都不止一个，而我做lst文件时只是取了第一个物体做为该图片所属的类别，而第一个物体未必是该图片中最明显的，可能这个会是导致准确率低的原因吧。

试着去找单标签的数据集，即图片只有一个标签，图片中只有一个明显的主要物体，有CIFAR-10和CIFAR-100，但是这个数据集中的图片分辨率都太低，只有32x32，很不行。因此考虑还是使用Pascal VOC数据集，但是要带bounding box作为输出，考虑实现一个RCNN。

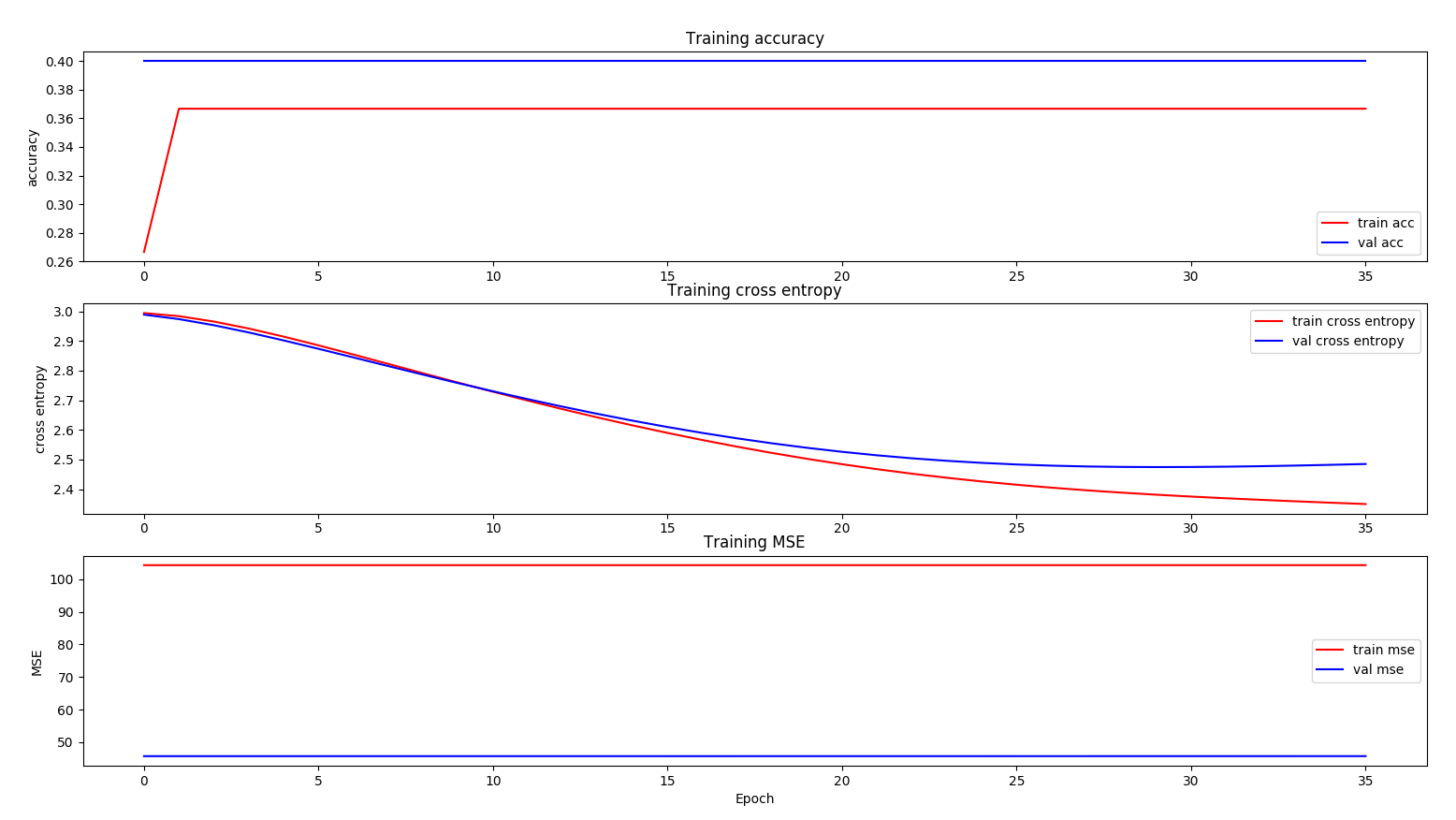
2.在尝试RCNN的算法思想之后，用AlexNet训练框中的物体效果也不好。

测试之后发现准确率还是只有42.65%.。。。。

测试发现模型的预测全部都是person，而person占了测试集中的40%多。

查看训练日志发现训练时的loss根本就没有降。。。

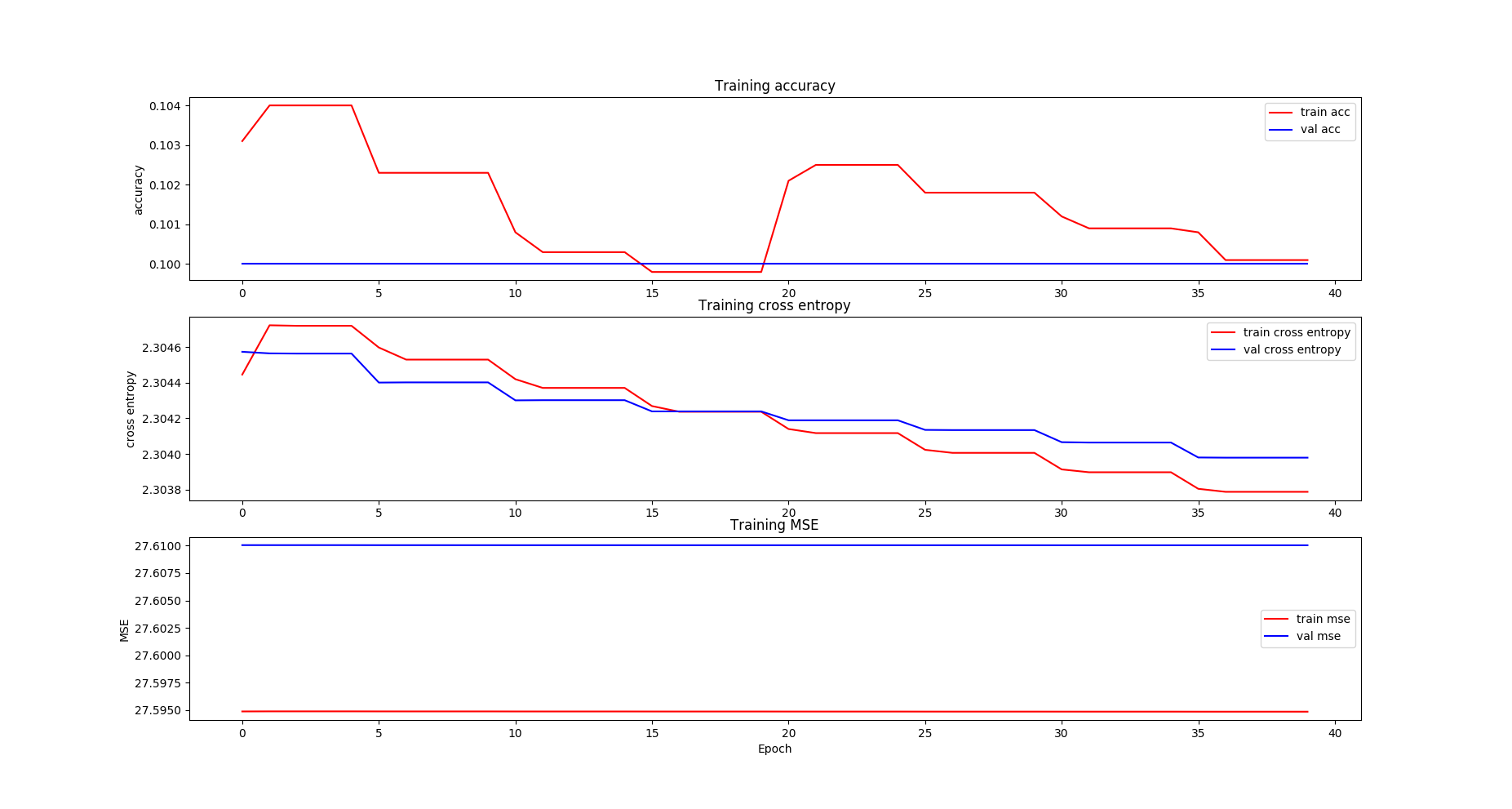
3.在迷你数据集上的训练发现训练的效果居然也是一样！甚至还更差劲，Train和val的acc都超过40%。



在把dropout层去掉之后效果依旧没有更好。

先尝试使用AlexNet训练CIFAR-10数据集，看是否正常，这部分内容写在test/testCifar10.py中：

除了optimizer\_params为自己原来使用的之外，其他部分包括数据的载入以及网络结构参考网上https://www.cnblogs.com/Mu001999/p/6221100.html，训练的效果如下：



网络结构其实和AlexNet差别也不大，是3个conv+act+pool+dropout然后再接两个全连接层，现在考虑照搬那个博客写的，再看看效果如何：

直接用博客中的代码居然验证集的准确率一直都是0.1！loss也一直不降！那这样的话会是哪里出了问题呢？