整个交通手势识别的流程主如下：

# 数据采集

使用手机、相机、摄像头等视频采集设备进行交通手势视频的采集，将视频按一定时间间隔进行采样，获得图片并调整图片的大小送入行人检测网络的输入。

# 行人检测

## 模型训练

行人检测网络使用行人和非行人的数据进行模型的训练，从输入图片中裁剪出[64 32]的滑动窗口，从热度图(heat map)中获得滑动窗口的维度（热度图表示数据集中行人的分布，表明图像中行人各种尺度和位置的表示）。最后，对获得的候选框应用非最大值抑制（NMS），将他们合并在一起并检测完整的行人。

行人检测网络总共有12层，包含卷积层、全连接层和分类输出层。

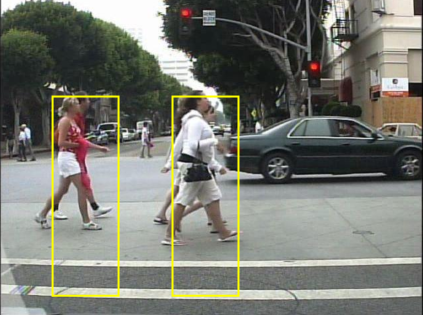
|  |  |  |  |
| --- | --- | --- | --- |
| 层号 | 变量名 | 层名 | 说明 |
| 1 | imageinput | 图片输入 | 图片尺寸为64×32×3，并进行零中心归一化 |
| 2 | conv\_1 | 卷积层 | 20个5×5×3的卷积，步幅[1 1]，填充[0 0 0 0] |
| 3 | relu\_1 | 整流线性单元 | 整流线性单元 |
| 4 | max\_pool\_1 | 最大池化层 | 2×2最大池化，步幅[2 2]，填充[0 0 0 0] |
| 5 | crossnorm | 归一化层 | 每个元素对应5个通道进行交叉通道归一化 |
| 6 | conv\_2 | 卷积层 | 20个5×5×20的卷积，步幅[1 1]，填充[0 0 0 0] |
| 7 | relu\_2 | 整流线性单元 | 整流线性单元 |
| 8 | maxpool\_2 | 最大池化层 | 2×2最大池化，步幅[2 2]，填充[0 0 0 0] |
| 9 | fc\_1 | 全连接层 | 512全连接层 |
| 10 | fc\_2 | 全连接层 | 2全连接层 |
| 11 | softmax | 软最大层 | 软最大层 |
| 12 | classoutput | 分类输出 | “非行人”和“行人”类别的交叉熵 |

## 模型预测

1. Region proposal
2. 感兴趣区域分类

检测框的精度应该大于0.9，否则过滤掉。

1. 使用非最大值抑制合并相似的候选框



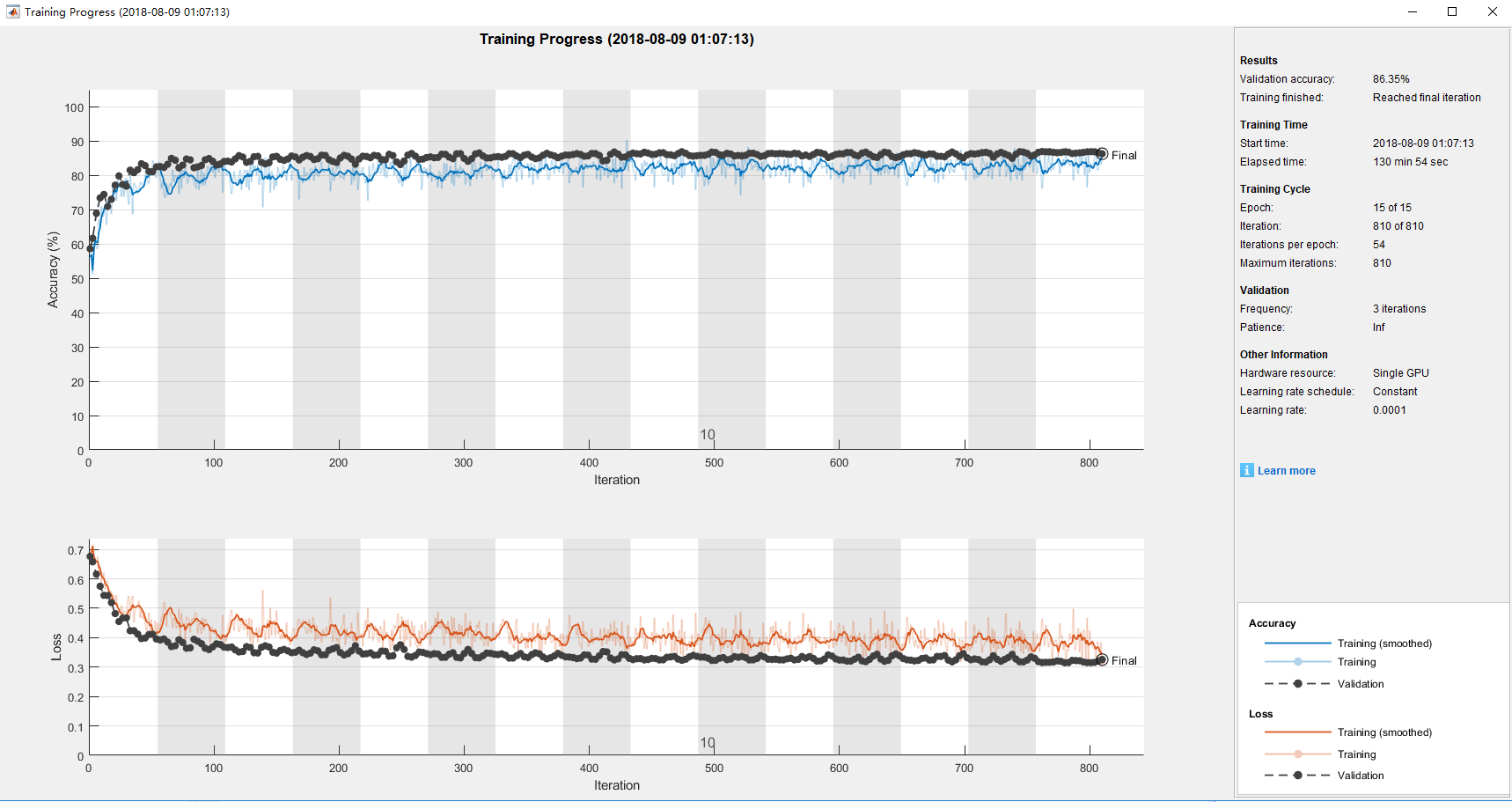
# 警察行人分类

将识别出来的行人进行二分类：警察或者行人。只有当分类结果为警察时候，进行交通手势的分类，否则不进行后续的动作分类。

警察行人的分类方法和模型结构采用[第四章](#_动作分类模型训练)的方法，训练集和验证集的数据为从网上爬取的警察、行人的图片数据，训练集17,994张（行人9,119张、警察8,875张），验证集2,010张（行人1,014张，警察996张）。

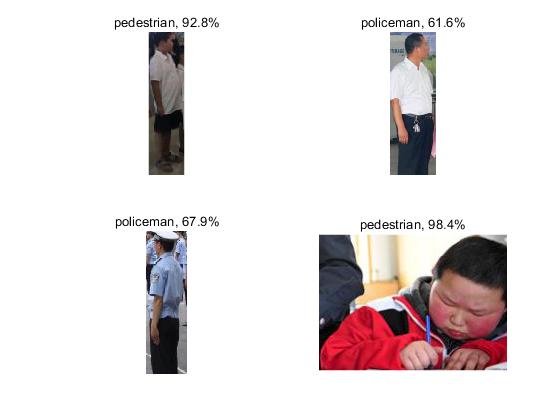
D:\workspace\traffic-gesture-recognition\PoliceRecognition\is_policeman_dataset\train\policeman\41_1.pngD:\workspace\traffic-gesture-recognition\PoliceRecognition\is_policeman_dataset\train\pedestrian\1490_1.png

模型训练的过程



最终在验证集上的准确率为86.35%。

分类验证



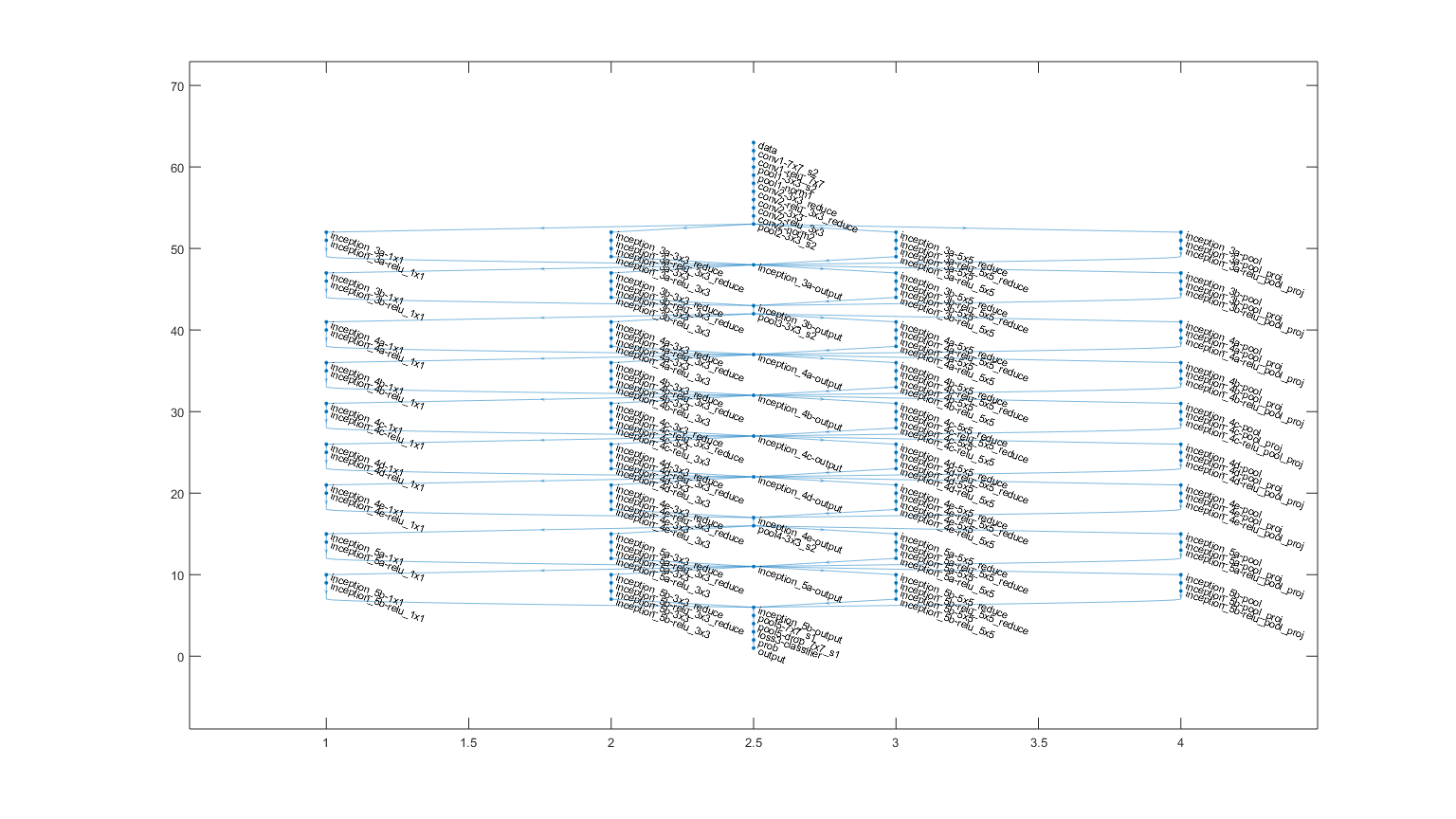
# 交通手势动作识别

## 数据加载

配置数据源的路径（里面包含文件夹go\_straight（46,421张）、park\_right、stop、turn\_right等）。按70%的训练集和30%的验证集对标签数据进行随机采样训练。

## 加载预训练模型

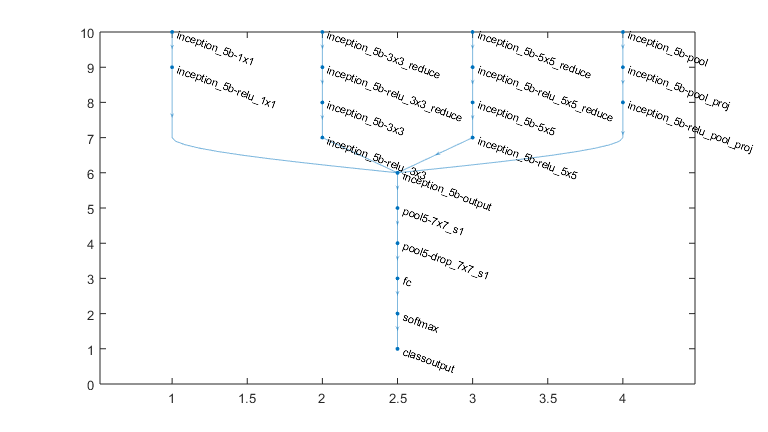
加载在ImageNet数据集上预训练的[GoogleNet](https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet)模型（已下载到model\vgg16.model），网络结构图为



网络中第一个元素是图片的输入层，要求为224×224×3的输入图片，其中3为色彩通道的数目。

## 替换最后三层

为了再次训练GoogleNet用来分类交通手势图片，我们替换了网络的最后三层（loos3-classifier、prob和output），它们包含了网络提取的类别概率和标签的特征，用新的三层结构来进行替换：全连接层、softmax层和分类输出层），并将最后的全连接的大小和需要分类的动作类别的数目相同（这里为4个手势动作）。为了使迁移后网络学习地更快，增加全连接层的学习率因子为10（权重学习率因子和偏置学习率因子都为10）。替换以后最后几层的网络结构图为



## 冻结初始层

到目前为止，在手势分类数据上的网络已经准备好。（可选）为了极大的加快网络的训练，我们可以冻结初始许多层的权重（因为冻结层的梯度不再需要结算），通过设置这些层的学习率为0，在学习的时候就不会更新冻结层的参数。在GoogLeNet中，开始的110层包含inception\_5a模块，我们设置开始的110层的学习率为0。

## 训练网络

### 输入图片处理

网络需要的输入图片大小是224×224×3，但是采集得到的图片大小不一。我们调整需要自动调整图片的大小，并使用在垂直方向对训练图片进行随机裁剪的方法对训练数据进行增强（随机变化的范围为水平和垂直方向30个像素）。数据增强能防止网络的过拟合并记住训练图片的准确细节。

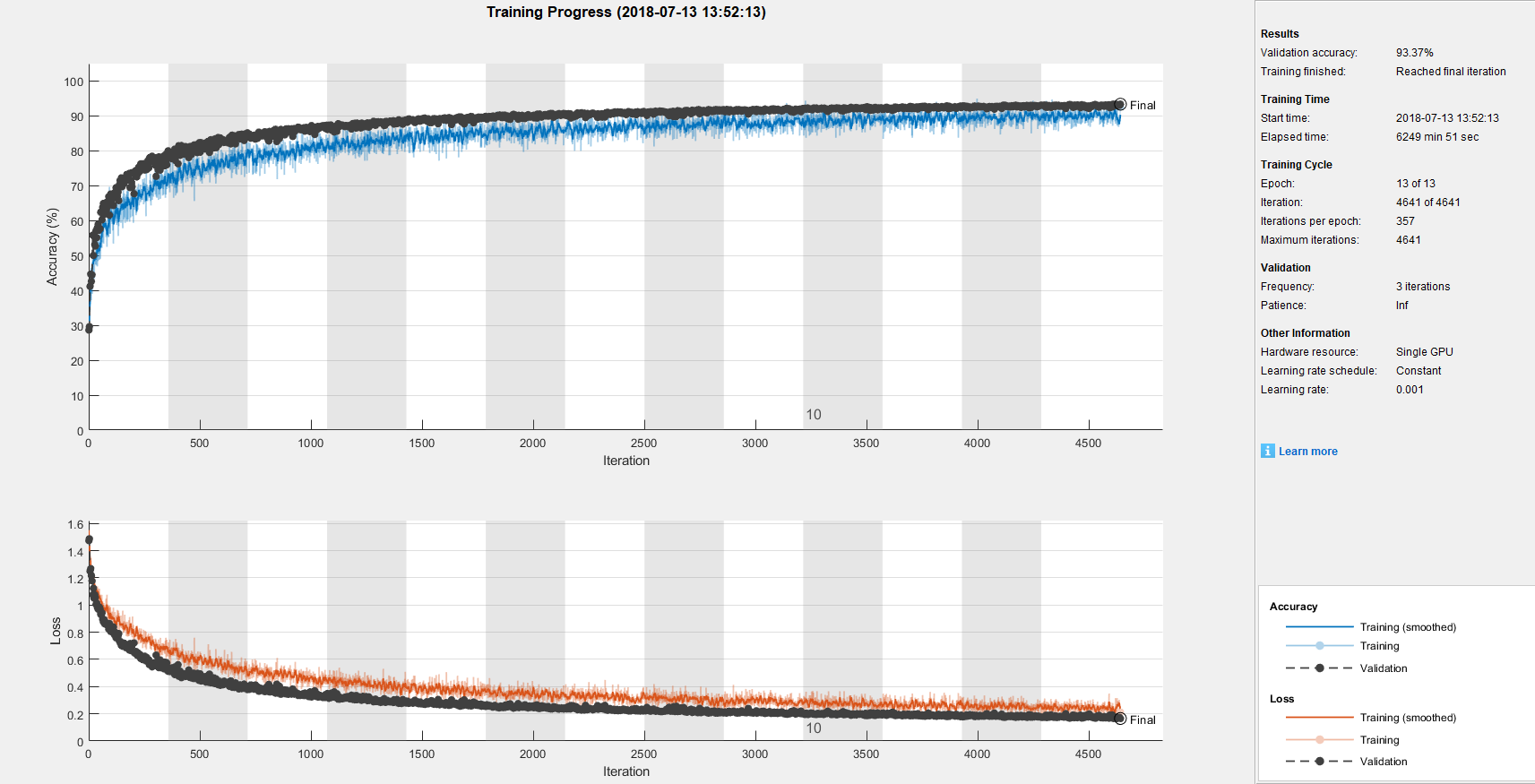
### 设置训练参数

为了加快训练，在最后全连接层设置较高的学习率，在中间层设置较低的学习率，在冻结层将学习率设置为0。

指定训练的epoch为13（执行迁移学习，训练的时候不需要很多epoch），每个epoch都会遍历整个训练数据集。指定最小批次的大小（根据显存设置，这里使用机器的显存为6G，设置的最小批次大小为256）。

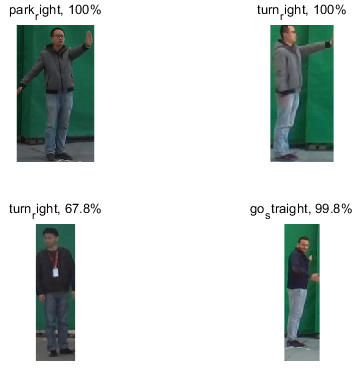
### 训练

当存在计算能力在3.0以上的GPU时，选择GPU进行训练，否则使用CPU进行训练。训练的过程图为



## 分类验证

使用精调后的网络分类验证集的图片，并计算网络的准确率。下图为验证集图片预测动作的标签和概率。



## 模型使用

加载已训练好的模型(model/gesture.mat)和待分类的图片，使用predict方法进行交通手势的识别，返回预测的标签和概率，验证集的准确率为93.37%，自己录制视频测试手势分类的准确率目前为91.12%。作为一个视频序列看待时，采取投票的方式，将预测最多的手势类别作为整个视频的交通手势动作类别，准确率93.75%。



# 时间效率

在415帧视频上进行测试，CPU表示在CPU i5上进行测试，GPU表示在GTX1060上进行测试，其中不包括模型的加载时间。

|  |  |  |
| --- | --- | --- |
|  | CPU | GPU |
| 行人检测 | 0.232s/帧 | 0.189s/帧 |
| 警察行人分类 | 0.352s/帧 | 0.089s/帧 |
| 交通手势分类 | 0.181s/帧 | 0.008s/帧 |

# 后续工作

## 提高行人识别的准确率

基于已有的数据训练的行人识别模型不能很好的框住警察手臂张开时的完整身子，调整识别框的宽度同时会引入识别的非感兴趣区域，后续可以从原始标注数据的角度或者提出新的方法来提高行人（主要是警察）识别的准确率。

## 提高动作分类模型的精度和泛化能力

为了提高模型的识别精度和泛化能力，我们可以采集更多不同场景下、不同人的交通手势数据，加入到模型的训练过程当中。

## 增加识别动作的类别

目前该模型支持四种交通手势的识别，为了使模型识别更多的交通手势，我们可以采集其他4种交通手势数据，比如：左转弯、左转弯待转、变道、减速，作为单独的类别，将处理好的图片数据按类别放入到和go\_straight、park\_right、stop和turn\_right同级目录中，文件名即为类别名，进行模型的训练。