

项目报告

项目报告

1.项目背景

2.问题描述

3.数据或输入

 3.1数据描述

 3.2数据特点

 3.3数据探索

 3.4图片分析

 3.4.1随机抽取一类图片

 3.4.2测试集数据

 3.5数据总结

4.解决方法描述

 4.1问题分析及解决方案

 4.2卷积神经网络

 4.3评估标准

 4.4基准模型

5.项目设计

6.项目实施

 6.1算法选择

 6.2Loss函数选择

 6.3优化函数选择

 6.4模型搭建

 6.4.1模型介绍

 VGG Net

 Google Net:

 Residual Net:

 实验模型:

 迁移学习

 Fine-tune

 多模型融合

 6.5具体步骤

 6.5.1总述

 6.5.2训练结果

 ResNet50模型

 InceptionV3模型

 Xception模型

 最终模型

 6.6模型总结

7.实际应用

 7.1实际应用概述

 7.2总体框架描述

 7.3技术系统架构

 7.4界面设计

 7.5执行流程图

 7.6技术路线

 7.7规范性设计

8.总结

 Reference

1.项目背景

我们经常遇到这样的场景：一盏灯变成绿色，你面前的车不走。另外，在没有任何意外发生的情况下，前面的车辆突然减速，或者转弯变道。等等这些现象，给道路安全带来了很大的影响。那么造成这样现象的原因是什么，主要有因为司机疲劳驾驶，或者走神去做其他事情，想象身边的例子，开车时候犯困，开始时候打电话，发短信，喝水，拿后面东西，整理化妆的都有。这对道路安全和行车效率形成了极大的影响。该项目的主要目的是通过摄像头采集数据实现驾驶员的行为检测，通过该项目可以实时对图像数据进行分析得到驾驶员行为状态。主要所使用的技术为目前最流行的深度学习中卷积神经网络进行实现。随着LeCun在1989年通过使用卷积神经网络实现手写数字识别，到2012年Alexnet在Imagenet竞赛中夺得图像分类冠军，深度学习的浪潮就此掀起，鉴于此。本问题使用卷积神经网络完全可以实现。



img_6.jpg



img_115.jpg



img_149.jpg



img_357.jpg



img_448.jpg



img_462.jpg



img_557.jpg



img_623.jpg

2.问题描述

驾驶员可能存在的走神的行为，大概有如下几种：

左右手用手机打字，左右手用手持方式打电话，调收音机（玩车机），喝饮料，拿后面的东西，整理头发和化妆，和其他乘客说话。为此，分类的准确率 accuracy 就是衡量解决这个问题好坏的重要指标

该问题简单来说就是一个分类问题，使用深度学习方法进行分类识别。输入为一张彩色图片，输出为十种指定状态的概率。由此可见，如果使用深度学习方法，多分类问题的输出层为softmax，通过该层可以预测每一个类别的概率，最终求得最大概率。

3.数据或输入

3.1数据描述

输入数据集来自Kaggle 下载地址如下：

<https://www.kaggle.com/c/state-farm-distracted-driver-detection/data>

下载下来解压后有3个文件

- driver_imgs_list.csv.zip (92.89K)

- imgs.zip (4G) 所有的图片数据, 解压后
 - train (训练集数据)
 - c0 ~ c9 分别表示不同状态的训练集
 - test (测试集数据, 用于提交Kaggle比赛的测试集)
- sample_submission.csv.zip (206.25K) Kaggle比赛需要提交的样本

其中 driver_imgs_list.csv.zip 的是对分类标号和人分类编号的csv文件。这个csv表格有三列

文件夹名	文件名	描述
subject	人的ID	不同的人值不同
classname	状态	c0 ~ c9
img	img_*.jpg	司机图片

3.2 数据特点

数据有以下特点：

- 1.训练集的图片数量是 22424, 测试集的图片是 79726, 测试集的数量远大于训练集。
- 2.训练集司机完全不是测试集司机。

由于数据有这些特点, 那么过拟合可能是要遇到的最大的问题。

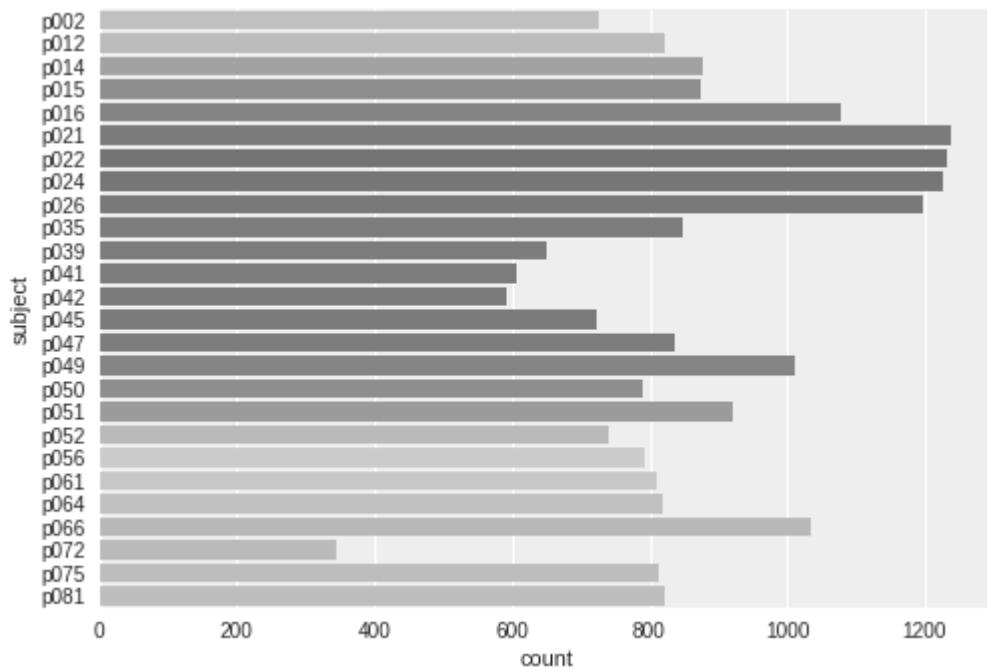
该数据分为10类, 每类约有1000张图片, 训练集平衡, 整体约20000张图片, 可以使用迁移学习把后面几层改为可训练层进行训练。该项目难度中上等, 主要难度在于大量的数据训练和模型调试搭建。

3.3 数据探索

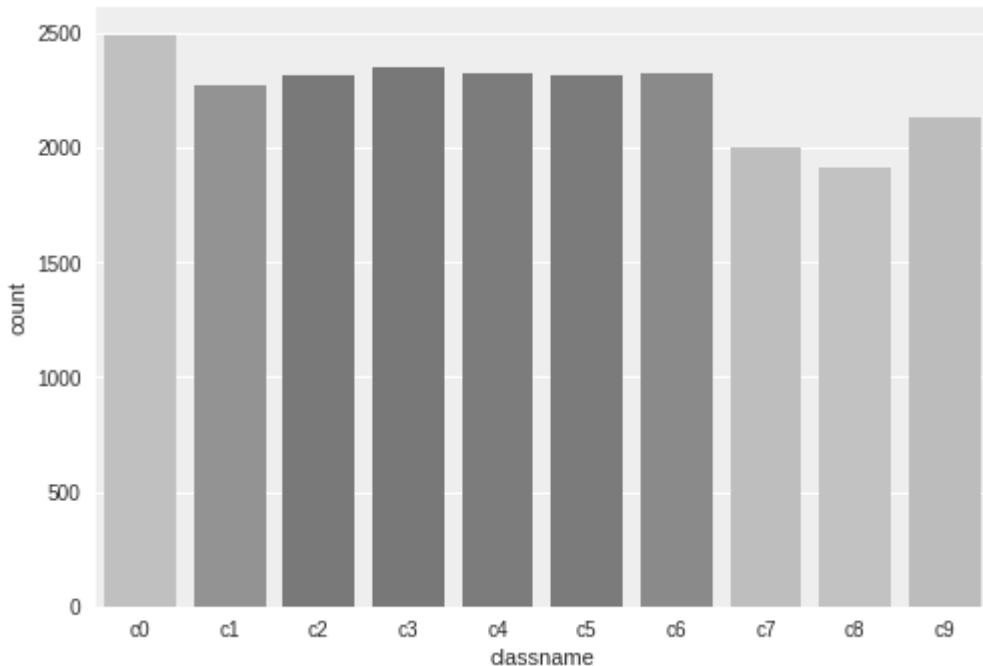
训练数据集中, 共有图片 22424张, 测试数据集中, 共有图片79726张

图片大小都是 640x480。

训练数据集一共有26个司机, 测试数据集有司机若干, 且和训练司机是不同的。每个司机对应的图片如下



一共有10种状态，且每个状态下面的图片数量如下：



由此可见数据基本为平衡状态，可以直接用来训练，不需要进行数据平衡处理。

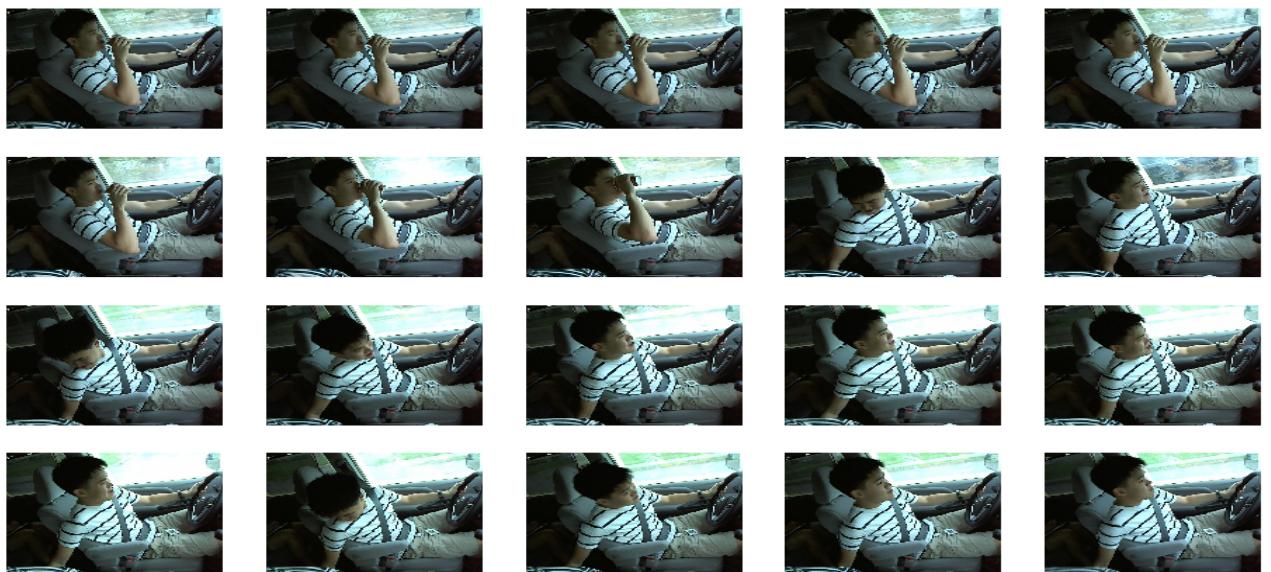
3.4 图片分析

3.4.1 随机抽取一类图片

看看司机喝水的图片(随机抽取了一类状态并列出)：



每个训练数据集的司机的图片是连续的:



3.4.2 测试集数据

测试集数据的司机和训练集的司机完全不同:



3.5 数据总结

基于以上数据的可视化，可以看出以下特点：

1. 训练数据和测试数据司机不同，但是车应该是一个车，不过摄像头拜访的位置，角度，差不多，但是是有差异的。所有后面要做预处理防止过拟合。
2. 车旁边玻璃上外的画面，是循环类似的，也就是每个人的都是依次操作响应的步骤来获取数据，车的行驶路径差不多，同一个动作的时候，窗外的画面是差不多，这一点可能会导致过拟合。
3. 训练数据的标注存在极为少数错误的，但是我没有好的办法批量筛选出错误的，所以，只能简单肉眼基本看一遍，找到部分错误的，并且放到正确的分类下。
4. 训练数据就26个司机，测试数据的图片更多，司机更多，场景更多。这样看来训练数据是非常宝贵的。

4. 解决方法描述

4.1 问题分析及解决方案

这是典型的分类问题，其目的是将图片归类为C0-C9十类。评估指标采用精度 **accuracy** 来评估结果好坏。Logloss 的评估方式，这也是 kaggle 比赛的评估方式 主要解决方法是使用卷积神经网络模型。

4.2 卷积神经网络

整体上采用卷积神经网络CNN来解决问题。卷积神经网络是神经网络的一种，把图片的部分相邻像素变成长宽更小，高度越深的单元，从而提炼出局部的特征，一层一层将特征往后映射，之后提取出最终的特征。现在世界上大部分图片处理和计算机视觉问题都是用卷积神经网络CNN来解决的。基于最早的Lenet，Alexnet，GoogLe net或者Resnet 进行分类识别。

4.3 评估标准

这是典型的分类问题，一般的评估指标采用 **Accuracy** 来评估结果好坏。

而 **logloss** 的评估方式是 kaggle 比赛的评估方式：

$$\text{logloss} = -1/N \sum_1^N \sum_1^M y_{ij} \log(p_{ij})$$

相对比这两种方案。**Accuracy**对于判断正确和错误的比重是一样的，也就是对了就多一个，错了就少一个，最终看正确的百分比，也就是判断正确的占总体测试数据的比例。由于**Accuracy**无法显示出函数是否具有显著性，所以选择**logloss**方法进行评判。

4.4 基准模型

基准模型使用在vgg,该模型为较为经典的一个神经网络模型，之前用来实现手写数字识别，可以作为baseline进行测试比较。

5.项目设计

项目设计分为以下几个阶段：

1. 进行基准模型搭建，搭建vgg算法模型，得到基准的准确率。
2. 进行深度学习模型构建，学习GoogLe net 和 Res net进行分类识别。计算准确率。
3. 通过阅读文献对模型结构修改，使用数据增强方法进行进一步微调，提升其准确率。

6.项目实施

6.1 算法选择

我整体上采用卷积神经网络 CNN 来解决问题。卷积神经网络是神经网络的一种，把图片的部分相邻像素变成长宽更小，高度越深的单元，从而提炼出局部的特征，一层一层将特征往后映射，之后提取出最终的特征。现在世界上大部分图片处理和计算机视觉问题都是用卷积神经网络 CNN 来解决的。

6.2 Loss 函数选择

由于是多分类问题，所以采用 categorical_crossentropy 的计算原理来使用 loss，下面简单介绍一下 categorical_crossentropy 的数学公式：

$$\text{logloss} = -1/N \sum_1^N \sum_1^M y_{ij} \log(p_{ij})$$

该函数具有一个显著特点，下面举例简述下：

1. 该函数是对每个分类的概率分别求和。
2. **logloss**的评估方式对判断有明显性，如果正确了， $P_{ij}=1 \Rightarrow \log(P_{ij})=0$,而 $P_{ij}=0.999 \Rightarrow \log(P_{ij})=-0.001$ 。最后增加的 log 差不多。但如果判断错误，如 $P_{ij}=0 \Rightarrow \log(P_{ij}) = -\infty$ 。 $P_{ih}=0.001 \Rightarrow \log(P_{ij})=-6.9$ 也就是判断错误一个，对得分的影响会非常大，所以用 logloss 评估对比 accuracy，更能反映模型和算法的能力。

6.3 优化函数选择

我最终选择 Adam 来做主优化器，然后再用学习率极低的 RMSprop 来补充优化。

下面简单介绍一些优化器的由来：

RMSprop 优化器，可以算作 Adadelta 的一个特例：

$$\rho=0.5 \quad E|g^2|_t = \rho * E|g^2|_{t-1} + (1 - \rho) * g_t^2$$

就变为了求梯度平方和的平均数。

如果再求根的话，就变成了 RMS(均方根)：

$$RMS|g|_t = \sqrt{E|g^2|_t + \epsilon}$$

此时，这个 RMS 就可以作为学习率 η 的一个约束：

$$\Delta x_t = -\frac{\eta}{RMS|g|_t} * g_t$$

RMSprop 的特点是：

- 1.RMSprop 依然依赖于全局学习率。
- 2.RMSprop 算是 Adagrad 的一种发展，和 Adadelta 的变体，效果趋于二者之间适合处理非平稳目标。

Adam:

Adam 优化器(Adaptive Moment Estimation) 本质上是带有动量项的 RMSprop，它利用梯度的一阶矩估计和二阶矩估计动态调整每个参数的学习率。 Adam 的优点主要在于经过偏置校正后，每一次迭代学习率都有个确定范围，使得参数比较平稳。公式如下：

$$\begin{aligned} m_t &= u * m_{t-1} + (1 - u) * g_t \\ n_t &= v * n_{t-1} + (1 - v) * g_t^2 \end{aligned}$$

Adam 的特点是：

1. 结合了 Adagrad 善于处理稀疏梯度和 RMSprop 善于处理非平稳目标的优点
2. 对内存需求较小
3. 为不同的参数计算不同的自适应学习率
4. 也适用于大多非凸优化 - 适用于大数据集和高维空间

由于 Adam 的这些特点，我在做 Fine-tune 的时候选择 Adam 作为主优化器，在优化到差不多的时候，再采用 RMSprop 进行超低学习率的调优。由于单模型的特征已经学习过了，所以在混合模型的训练的时候，我直接 Adam 用超低学习率来进行学习。

试验后发现，在单模型 Fine-tune 的时候，使用 RMSprop 比起 Adam 优化完成的，要好 0.03~0.05 个 accuracy，LogLoss 要好 0.05~0.2 个点。但是如果全部使用 RMSprop，收敛得比较慢，且也达到开始用 Adam 优化器训练的精度。

6.4 模型搭建

基于借鉴 **lenet** 和 **vggnet** 等网络模型，我自行搭建了两个神经网络模型。先简单介绍下前人的模型。

6.4.1 模型介绍

VGG Net

整体使用的卷积核都比较小（3x3），3x3是可以表示「左右」、「上下」、「中心」这些模式的最小单元了。

还有比较特殊的1x1的卷积核（Inception-v1也有这个东西），可看做是空间的线性映射。

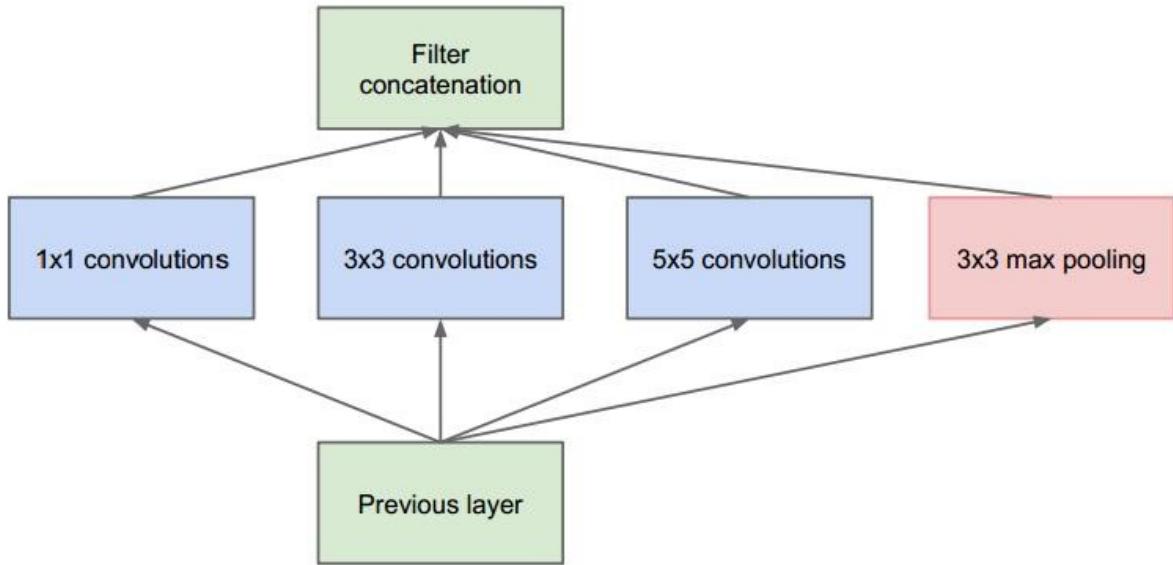
前面几层是卷积层的堆叠，后面几层是全连接层，最后是softmax层。所有隐层的激活单元都是ReLU，论文中会介绍好几个网络结构，只有其中一个应用了局部响应归一化层（Local Response Normalisation）。

使用多个较小卷积核的卷积层代替一个卷积核较大的卷积层，一方面可以减少参数，另一方面作者认为相当于进行了更多的非线性映射，可以增加网络的拟合/表达能力。

Google Net:

该网络主要又有一个Inception的结构：

如图所示



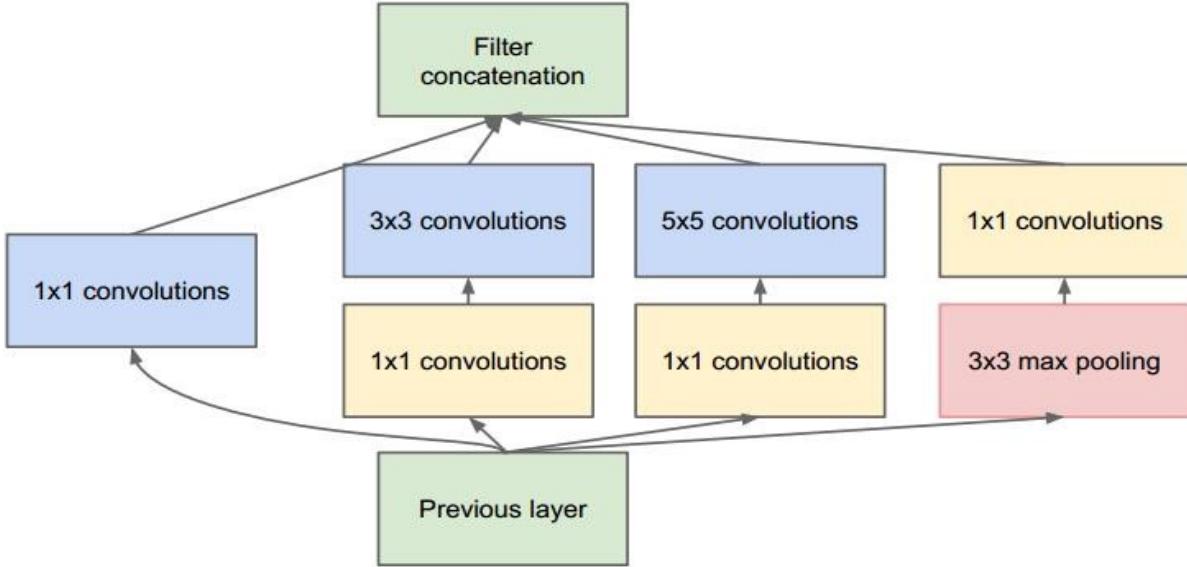
该结构特点：

1. 采用不同大小的卷积核拥有不同大小的感受野，最后拼接意味着不同尺度特征的融合；
2. 之所以卷积核大小采用1、3和5，主要是为了方便对齐。设定卷积步长stride=1之后，只要分别设定pad=0、1、2，那么卷积之后便可以得到相同维度的特征，然后这些特征就可以直接拼接在一起了；
3. 文章说很多地方都表明pooling挺有效，所以Inception里面也嵌入了。
4. 网络越到后面，特征越抽象，而且每个特征所涉及的感受野也更大了，因此随着层数的增加，3x3和5x5卷积的比例也要增加。

但是，使用5x5的卷积核仍然会带来巨大的计算量。为此，文章借鉴Network in Network，采用1x1卷积核来进行降维。

例如：上一层的输出为 $100 \times 100 \times 128$ ，经过具有256个输出的5x5卷积层之后(stride=1, pad=2)，输出数据为 $100 \times 100 \times 256$ 。其中，卷积层的参数为 $128 \times 5 \times 5 \times 256$ 。假如上一层输出先经过具有32个输出的1x1卷积层，再经过具有256个输出的5x5卷积层，那么最终的输出数据仍为 $100 \times 100 \times 256$ ，但卷积参数量已经减少为 $128 \times 1 \times 1 \times 32 + 32 \times 5 \times 5 \times 256$ ，大约减少了4倍。

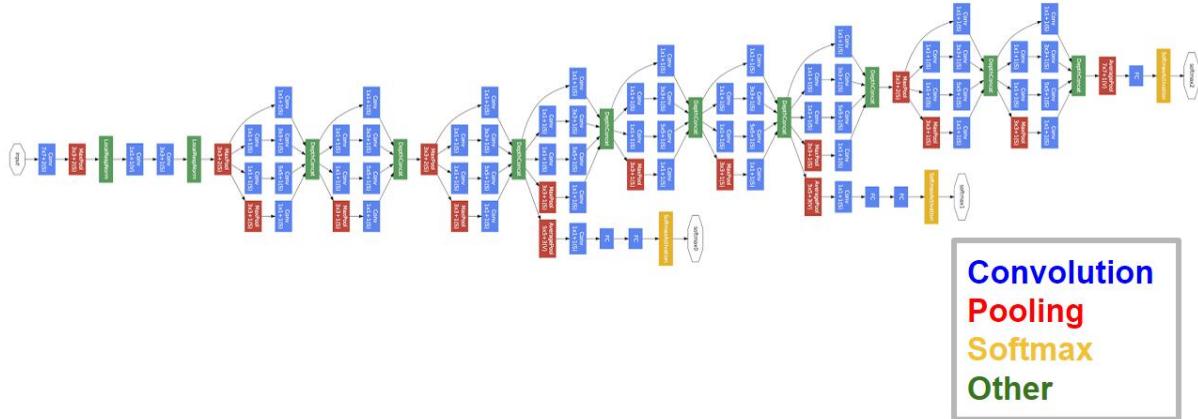
结构如图所示：



1.第一个模型使用输入为128*128大小，深度为三通道的图片作为输入，搭建8层卷积层，没有添加全连接层，其目的是减少权重大小，但是经过多次训练结果显示效果并不好，偶尔一次会出现正确率达到90左右的情况，可想而知初始化权重是一件非常重要的事情。

该模型整体结构如图所示：

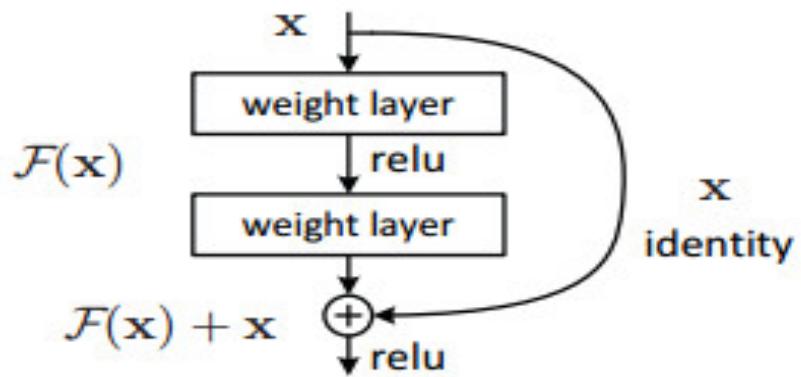
GoogLeNet



Residual Net:

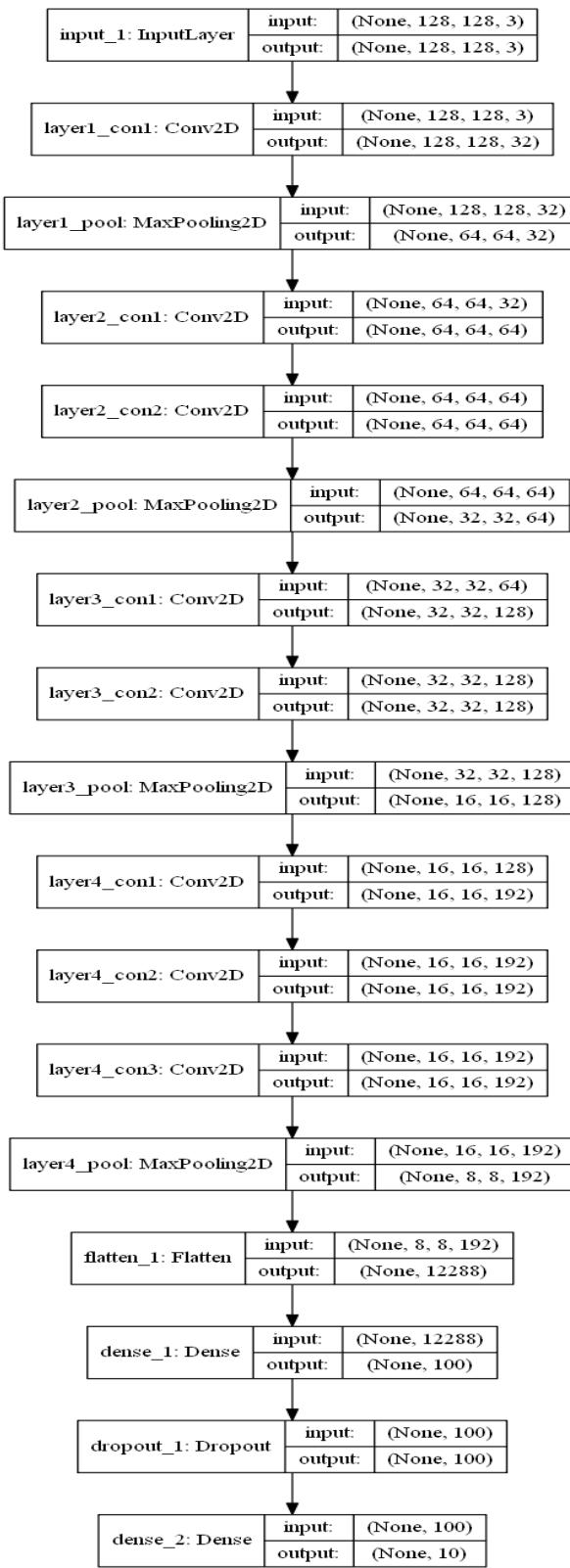
核心思想是出了一种减轻网络训练负担的残差学习框架，这种网络比以前使用过的网络本质上层次更深。明确地将这层作为输入层相关的学习残差函数，而不是学习未知的函数。解决了该问题的一大障碍是臭名昭著的梯度爆发与消失问题，它从一开始就阻碍了收敛。然而，这个问题很大程度上被归一的初始化和中心归一层解决了，它确保几十层的网络开始用反向传播收敛随机梯度下降（SGD）。主要介绍了一个深层次的残差学习框架来解决精准度下降问题。明确地让这些层适合残差映射，而不是寄希望于每一个堆叠层直接适合一个所需的底层映射。形式上，把 $H(\mathbf{x})$ 作为所需的基本映射，让堆叠的非线性层适合另一个映射 $F(\mathbf{x}) = H(\mathbf{x}) - \mathbf{x}$ 。那么原映射便转化成： $F(\mathbf{x}) + \mathbf{x}$ 。假设优化剩余的映射，比优化原来未引用的映射更容易。如果身份映射是最佳的，那么将剩余的映射推为零，就比用一堆非线性层来适应身份映射更容易。

具体结构见下图：

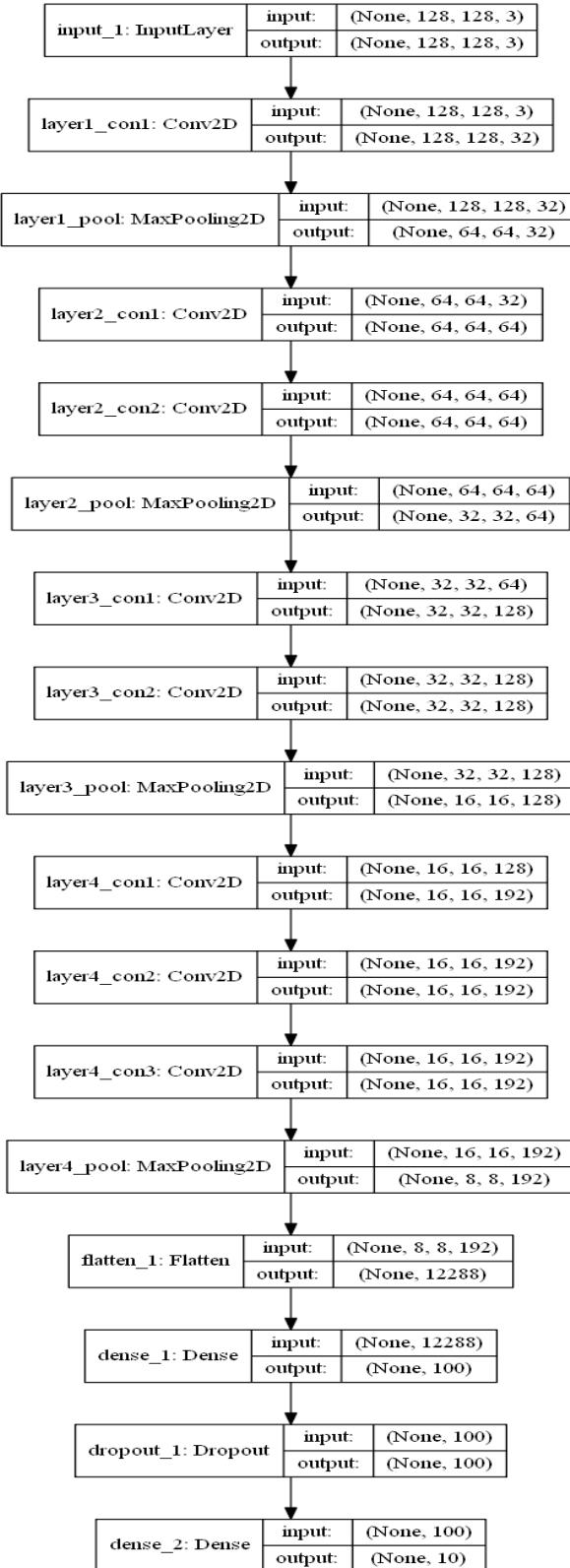


实验模型：

1. 模型1网络结构见附件model.png所示,主要是将前人的一些优点进行了融合, 尽量减少了模型的大小。



2.模型二主要是在模型一的基础上末端添加了一层全连接，使得具有更好的分类效果，经过测试发现效果果不其然十分显著，在validation数据集上的**Accuracy**达到98%，效果十分好。



迁移学习

我选择使用迁移学习，迁移学习就是在直接用海量数据下已经学习好的成熟模型结果，在从基础上做些小的改造再根据实际问题重新训练从而完成学习的过程。

我选择迁移学习主要是两点原因考虑，首先，题目的数据集比较小，如果从头开始学习很容易学习不该学习的特征，其次就是学习的时间的考虑，我要节约使用GPU的时间，因为价格太贵了。

由于迁移学习是基于已经学习好的成熟模型的基础之上再次训练，神经网络大部分层次的特征都已经提取了，用较少的迭代次数就能够完成学习。所以迁移学习能够节省不少时间。

Fine-tune

我选择使用Fine-tune，就是在迁移学习的基础上，不锁定ImageNet模型的部分weights的情况下训练，也就是ImageNet的weights部分层次锁定，部分层次重新训练。之所以这样考虑，是因为本项目的数据集的特征和ImageNet数据集的特征相当大，如果只训练全连接层，很难找到特征，所以要推到全连接层之前的层次去训练。

多模型融合

我选择采用多模型，因为不同模型的设计思路不同，提取图片特征的原理也不同，理论上使用多模型融合的方式能够提高精度，而我采用的融合方式不是求个平均值，而是把各个单模型Fine-tune的weights去掉全连接前的输出，串联起来，再通过神经网络训练来解决各个模型的权重。

6.5具体步骤

6.5.1 总述

第一步：

首先，将train数据，划分为训练和验证集，我随机选取了百分之20图片作为验证集进行训练。另外由于数据的特性，需要做一下数据增强，如做一下旋转，平移，放缩等来增加数据的多样性，从而有效防止过拟合。还需要进行图像预处理，预处理主要包括归一化以及减去图像均值的操作，这样才能保证模型不会看到一个有很大RGB偏移的图像，避免后面在神经网络学习的时候，学到不必要的特征，从而产生过拟合的现象。

第二步

测试基准模型的结果，再一次我选择VGG16的imagenet迁移学习模型来作为基准模型，然后只重新训练全连接层，从而得到一个基准模型的结果。

第三步

依次尝试自己依据前人的模型配置模型，进行参数调节。

第四步

模型优化完成后，开始做多模型融合，并优化融合模型，得出最终结果。然后生成pred文件，提交到kaggle上看最后的得分。

6.5.2 训练结果

基准模型：

使用vgg网络进行fine-tune结果如下：

```
loss: 1.4377 - acc: 0.5305  
val_loss: 1.2111 - val_acc: 0.6449
```

我这里尝试了对Vgg16, Vgg19, ResNet50, InceptionV3, Xception做Fine-tune并优化，最后结果ResNet50, InceptionV3, Xception都成功的，即训练集和测试集的accuracy都做到0.9左右；但是Vgg16和Vgg19都没有成功，也就是很长时间的训练，accuracy最多只能到0.6左右，且尝试过不同的层次做fine-tune，均没有成功。

ResNet50模型

经过各种尝试和优化，发现以下几点是最优参数

1. 图片缩小到(240, 320, 3)，可能因为很多场景动作很小，缩太小不利于提出特征。且反复尝试，保持图片比例不变进入神经网络的结果却是最好。
2. 现用Adam先训练6轮，再用RMSprop用极小的学习率0.00001训练6轮。

3.多次实验，发现在第152层的时候开始tune效果最好，就是锁定0-151层的权重，从152层起，权重是可以训练的。

ResNet的输入模型时，图放缩到大小为(240, 320, 3)
其去掉全连接层之后的输出是长度为2048的向量。

在最后迭代后的结果是：

loss: 0.0038 - acc: 0.9987
val_loss: 0.7260 - val_acc: 0.8596

InceptionV3模型

经过各种尝试和优化，发现以下几点是最优参数

1. 图片缩小到(360, 480, 3)，可能因为很多场景动作很小，缩太小不利于提出特征。且反复尝试，保持图片比例不变进入神经网络的结果却是最好。
2. 现用Adam先训练4轮，再用RMSprop用极小的学习率0.00001训练6轮。
3. 多次实验，发现在第172层的时候开始tune效果最好，就是锁定0-171层的权重，从172层起，权重是可以训练的。

InceptionV3的输入模型时，图放缩到大小为(360, 480, 3)
其去掉全连接层之后的输出是长度为2048的向量。

在最后迭代后的结果是：

loss: 0.0016 - acc: 0.9995
val_loss: 0.3046 - val_acc: 0.9376

Xception模型

经过各种尝试和优化，发现以下几点是最优参数

1. 图片缩小到(360, 480, 3)，可能因为很多场景动作很小，缩太小不利于提出特征。且反复尝试，保持图片比例不变进入神经网络的结果却是最好。
2. 现用Adam先训练4轮，再用RMSprop用极小的学习率0.00001训练6轮。
3. 多次实验，发现在第172层的时候开始tune效果最好，就是锁定0-171层的权重，从172层起，权重是可以训练的。

Xception的输入模型时，图放缩到大小为(360, 480, 3)
其去掉全连接层之后的输出是长度为2048的向量。

在最后迭代后的结果是：

loss: 0.0095 - acc: 0.9973
val_loss: 0.4296 - val_acc: 0.9036

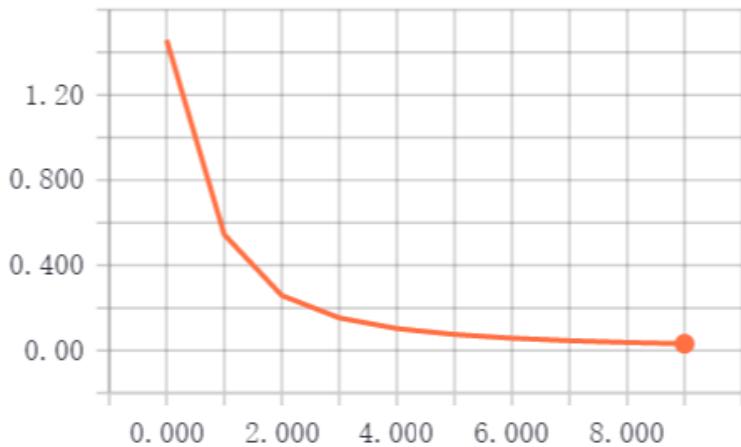
模型	训练 Accuracy	验证 Accuracy	训练 Loss	验证 Loss
基准模型 VGG16	0.5305	0.6449	1.4377	1.2111
Resnet	0.9987	0.8596	0.0038	0.7260
InceptionV3	0.9995	0.9376	0.0016	0.3046
Xception	0.9973	0.9036	0.0095	0.4296

最终模型

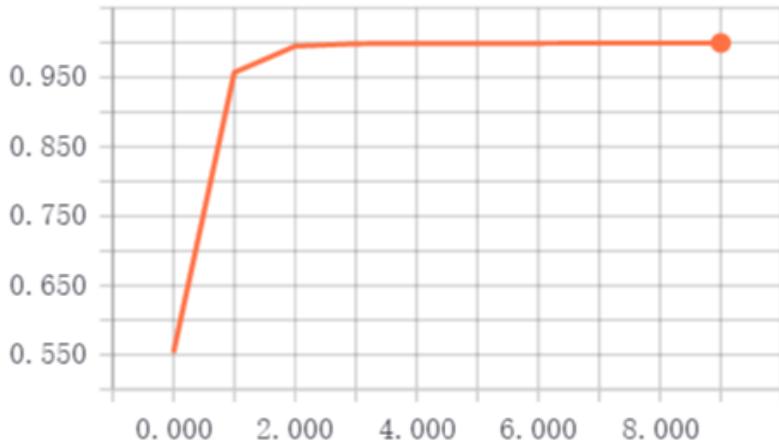
新的模型把ResNet, InceptionV3 和 Xception 混合起来做，从而完成最终模型

新的模型把ResNet, InceptionV3 和 Xception的去掉全链接层的结果混合起来，然后重新用神经网络训练全链接层。最后的在Adam优化器下迭代10轮之后效果如下

loss



acc



最终结果如下：

loss : 0.0317 - acc : 0.9995

valloss : 0.2900 - valacc : 0.9345

将其结果提交到Kaggle后，得到的分数如下：

0.2577

6.6模型总结

相比于网络上的其他网络模型，如*Alexnet* *VGGnet* *GoogleNet*等网络，该网络具有比较小的运算量，整体权重导出后仅有50MB，最后只有一层全连接层，运算速度快，可以有较高的实时性，且分类效果正确率达到98%，具有较高的现实意义。为此我出于兴趣搭建了一个APP来实现这个功能。

7.实际应用

7.1实际应用概述

机动车出行是当今社会最普遍的出行方式，国家统计局的最新数据显示，全国机动车驾驶员人数已经高达3亿余人。与此庞大数量相对应的是同样持高的交通事故发生数。而引起交通事故发生的原因中，驾驶人员的不良驾驶行为占到了相当大的比例。需要有管理人员对其监管，并且驾驶人员对自身的驾驶行为进行矫正，这也就是基于深度学习技术的驾驶行为评测系统诞生的最初构想。

基于深度学习的车内危险驾驶行为的识别系统的主要功能是将视频或静态图像格式的数据上传到云端，云端服务器通过深度学习算法进行图像识别，分析驾驶人员的驾驶行为，并对不良行为分类、记录。

本项目总体以方便驾驶人员回看、矫正自身驾驶行为，以及为管理者对驾驶人员有效管理途径为中心进行需求导向，解决了驾驶人员对自身驾驶行为进行矫正，或管理者对驾驶人员进行监管约束时缺乏依据的痛点。

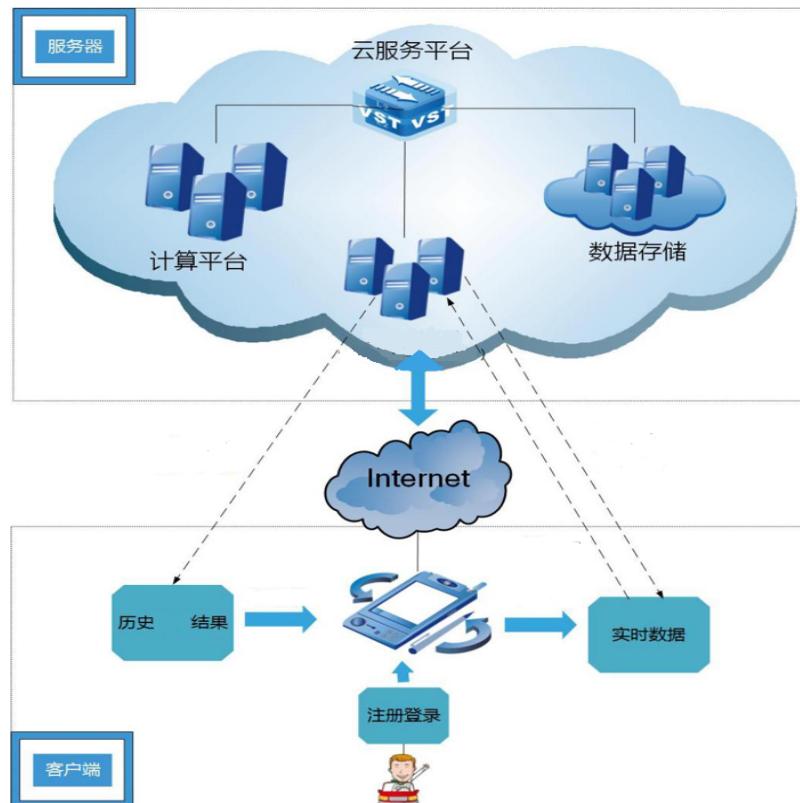
7.2总体框架描述

主要功能有以下三点：

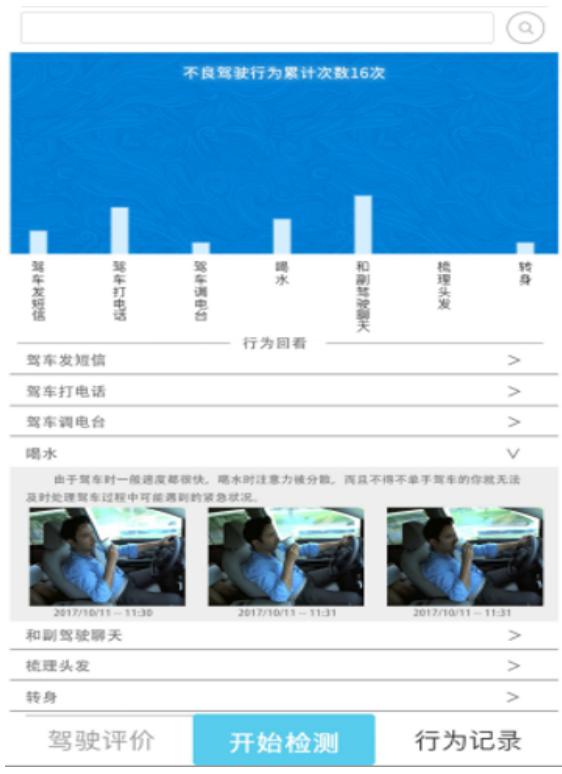
- (1)建立驾驶人员信息系统，实现驾驶人员与管理者凭借工作编号登陆、并按照相应权限使用系统。
- (2)系统以视频或静态图像的格式获取驾驶人员的驾驶行为信息，并对内容进行分析，识别并记录信息中的不良行为
- (3)可以查看指定时间的驾驶行为记录，和系统评价，并且可以将不良行为按时间、频率分布，以图表的形式进行查看。

7.3技术系统架构

实现前端移动平台与后端算法处理数据库平台分离，使用REST服务框架，前端主要负责司机登陆上传视频图像数据到云端，后台接受数据后进行算法分析得到分析结果存入数据库。



7.4界面设计



7.5 执行流程图



7.6 技术路线

前端

前端APP端通过Http和REST的api与后台进行交互，通过IL2CPP方法直接编译为Native app，并且使用.Net技术跨平台，实现移动端和web端互联互通。与传统应用相比，拥有更优秀的性能，信息也更为安全。

后台

后台使用基于python的flask框架，使用nginx工具管理静态资源，gzip对数据进行压缩从而提高传输率，利用sqlalchemy和mysql提供数据库储存，并使用redis缓存任务队列，利用supervisor管理和部署项目，提高稳定性和并发性。

算法

主要使用Keras框架进行卷积神经网络模型的搭建，参考牛津大学视觉图像组开发的VGG16模型进行超参数调节，通过搭载Nvidia GTX 1060显卡，使用网络开源数据集和手工标注数据集结合进行迁移学习训练，数据集约30000张左右图片，经过训练调节参数其在验证集上的准确率达到98%以上，达到项目使用要求。

7.7 规范性设计

(1) 后端API接口规范性

后台API接口整体规范采用RESTful方式来实施。API与用户的通信协议，总是使用HTTP协议。应该将API的版本号放入URL。

(2) 数据库设计规范性

表中应该避免可为空的列。表不应该有重复的值或者列。表中记录应该有一个唯一的标识符。数据库对象要有统一的前缀名。尽量只存储单一实体类型的数据。

(3) 代码规范性

基本的缩进和tab的使用，运算符之间空格的使用，避免某行过于长的问题，注释的使用，用变量去代替某个表达式，变量和函数的取名问题，不要利用书上说的优先级来省略括号，不要使用特别复杂的表达式，不要使用太多层次的循环和判断，避免废弃代码的使用，代码的结构化，模块化，函数的使用以及函数功能的单一性

(4) UI界面规范性

遵循一致的准则，确立标准并遵循

8. 总结

最终模型的结果达到了我之前设定的 Loss < 1.0，在前 1/5 的目标。

但是由于我自身机器性能问题，和业务学习不多没有足够时间训练数据，没有做得更好。不过我会继续改进算法，后续改进思路大概如下。

(1)全面彻底清理训练数据异常数据

(2)训练数据集中有些错误标记，对结果有些影响，要想办法清理掉这些。

(3)可以参考前后几帧图片来判断我现有的方式，是把每一帧数据当作完全独立的图片来处理的，但是如果参考前后每帧的画面内容，想办法用起来，对结果应该是有些帮助的。而且如果最终在车上实地部署的化，新数据预测的时候也可以采用关联前后帧的方式来做，这样应该会有更高的精度。

后续我有更多时间，还会继续优化和尝试，把分数做得更高。

Reference

[1]Y. LeCun, B. boser, J.Denker, D.Henderson, R. Howard, W. Hubbard, and L. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Comp.* ,1989. 1

[2]Karen Simonyan, Andrew Zisserman Very Deep Convolutional Networks for Large-Scale Image Recognition 2015

[3]Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma. Provable bounds for learning some deep representations. 2013

[4] Wei Liu, Yangqing Jia Going Deeper with Convolutions Christian Szegedy 2014

[5]Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. 2013

[6]Christian Szegedy, Vincent Vanhoucke Rethinking the Inception Architecture for Computer Vision

[7]手把手教你如何在 Kaggle 猫狗大战冲到 Top2% [杨培文] <https://ypw.io/dogs-vs-cats-2/>

[8]使用 Keras 来破解 captcha 验证码 [杨培文] <https://ypw.io/captcha/>

[9]Kaggle 求生：亚马逊热带雨林篇 [刘思聪]https://zhuanlan.zhihu.com/p/28084438?utm_source=itdadao&utm_medium=referral

[10] Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

[11]Very Deep Convolutional Networks for Large-Scale Visual Recognition