

# 机器学习纳米学位

走神司机 潘超 优达学城  
2018年6月6日

## I. 问题的定义

### 项目概述

项目将从一些车载视频摄像头中截取的静态图像识别驾驶员是否处于安全驾驶的状态。

生活中许多驾驶员喜欢一边开车一边做别的事情，如：打电话、发微信、吃东西、聊天、疲劳驾驶等等，安全隐患非常大。特别是一些大巴车司机，关乎到整个大巴车上几十个人的人身安全。该项目的数据来源方Kaggle比赛平台中的资料也指出：

根据美国疾病防控中心机动车安全部门的数据，五分之一的车祸是由分心的司机造成的。可悲的是，这意味着每年有42.5万人受伤，3000人因分心驾驶而死亡。

早期驾驶员状态检测方法主要是基于车辆运行状态的检测方法，包括车道偏离报警、转向盘检测等，对驾驶员本身的特征敏感度不高，容易因环境因素误判，也不能从根本上解决驾驶员状态检测的问题，而近年的基于深度学习的图像识别技术则提供了不错的解决办法，通过对视频图像进行分析检测驾驶员当前的状态并给予提醒，甚至在出现更严重的危险情况时通过车辆控制信号及时主动刹停汽车。

项目使用的相关数据及评分规则均来于二年前的Kaggle比赛，当年一共有1440名参赛队伍参于该赛事。项目地址：[state-farm-distracted-driver-detection](#)

### 问题陈述

处理通过车载摄像头记录到的驾驶员状态图像，对图像进行识别处理，分析图像中驾驶员当前所处的状态，以满足对安全驾驶提醒的需求。需要从图像中识别包括如下的驾驶员状态：

0. 安全驾驶
1. 右手打字
2. 右手打电话
3. 左手打字
4. 左手打电话
5. 调收音机
6. 喝饮料
7. 拿后面的东西
8. 整理头发和化妆

## 9. 和其他乘客说话

每一张图片识别出的结果应该是该图片分别在十种状态中的概率值，如安全驾驶的图片的理想识别结果应该为c0类别的概率为1，其他9种类别的概率为0。

项目将使用卷积神经网络来识别这些图像属于哪种状态，卷积神经网络是从2012年开始迅速成长起来的新型图像识别算法和架构，至今已发展出许多不同的版本，在图像识别方面取得了越来越高的准确率。

## 评价指标

评估指标使用kaggle中该项目的评估方式，即multi-class logarithmic loss，损失值计算公式：

$$\text{logloss} = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^M y_{ij} \log(p_{ij})$$

公式中 $N$ 为图像的数量，用于训练集时为当前训练集的数量，用于验证集时为验证集的数量，测试集同理。 $M$ 表示图像标记的数量，在该项目中 $M$ 为10。 $y_{ij}$ 为第*i*个图像在第*j*分类中的标记概率，如果图像为该类，则该值为1，否则为0。 $\log$ 为自然对数， $p_{ij}$ 为第*i*个图像在第*j*分类中标记的预测概率。将每一张图像每个分类的预测概率的自然对数与分类目标标记的积相加再取负均值，最终即为多分类损失值。

## II. 分析

### 数据的探索

数据集来源于往年的Kaggle竞赛。压缩包中文件结构如下：

- driver\_imgs\_list.csv
- sample\_submission.csv
- imgs.zip
  - test
    - img\_1.jpg
    - img\_2.jpg
    - .....
  - train
    - c0
      - img\_34.jpg
      - img\_105.jpg
      - .....
    - c1

- c2
- .....

数据集中包含大量车载摄像头对驾驶员位置的摄影截图，可清楚看到驾驶员的各种行为，包括打电话、喝饮料、拿后面的东西、打字等。数据集中将图片数据分为了训练集和测试集，训练集可用于该项目中训练模型，测试集可在模型训练完毕后检验预测效果，可提交至Kaggle中计算已训练模型的最终得分。训练集中已将图像标记分类，分为c0到c9一共十个文件夹存放，共22424张图片。测试集中有79729张未标记分类的图片。

数据集中每一张图片大小为640\*480像素。图片中的驾驶员各种各样，有胖有瘦，有高有矮，有男有女、甚至还有不同肤色的驾驶员，有的驾驶员手臂上还有纹身。图片的光线有明，也有暗，甚至还有些有点爆光过度，导致难以发现手中的透明杯子。

### 1. 因光照原因看不见喝饮料的杯子



### 2. 胖驾驶员



3. 图像模糊

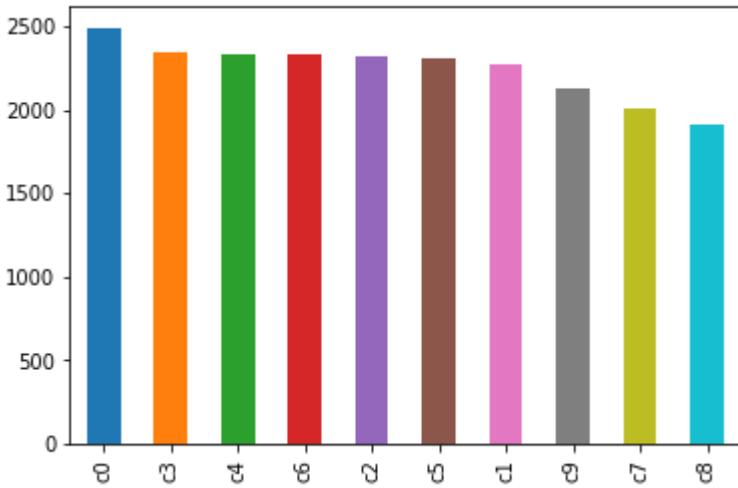


探索性可视化

从数据集中的driver\_imgs\_list.csv文件中可得处一些关于该数据集的信息：

- 总共有22424张已标记图像
- 总共有26名驾驶员的不同状态的图像，每名驾驶员的图像数量并不完全相同，最多的驾驶员有1237张图像，最少的驾驶员只有346张图像
- 总共有10种不同的驾驶状态，且每种状态的图片数量差别不大，即它们呈均匀分布

训练数据中司机状态分类呈均匀分布：



每个司机拥有的图像数量：



## 算法和技术

### Tensorflow

Tensorflow是一个Google公司开发的基于数据流图计算的人工智能神经网络处理框架，可广泛应用于如语音识别，自然语言理解，计算机视觉，广告等。

### Keras

Keras是一个高层神经网络API，基于tensorflow、Theano以及CNTK后端框架的上层框架，让tensorflow等基础神经网络框架更易于使用。

## 数据增强

由于图像有明有暗，司机在图像中所处的位置也不同，甚至有些图片比较模糊，决定在训练之前对分割出的训练图像和验证图像做数据增强处理。

数据增强是将图片变换处理出多式多样的不同副本，图像中的主要内容不变，但经过数据增强预处理的图像会被挤压、缩小、旋转角度、添加噪点、Dropout、模糊处理、提升亮度、降低亮度等，将图像这样预处理后模型将有机会认识到图像的各种形态，增加模型的鲁棒性和泛化能力。

项目将尝试使用Keras中的ImageDataGenerator对训练的图像数据做增强处理。

## InceptionV3

InceptionNet模型架构首次出现在ILSVRC2014的比赛中，由Google开发，以较大优势取得了当年的第一名，当前为InceptionV1，而InceptionV3则为它的后期改进版本。InceptionV3拥有比AlexNet和VGG19更大的网络，但其计算量只有15亿次浮点运算，同时只有500万的参数量，准确率却大大胜于AlexNet和VGG19。拥有着计算量小、模型训练快、分类准确的特点。

InceptionNet内部多处使用InceptionModule，总共拥有3种，共10个InceptionModule，多个小的网络反复使用一起堆叠成了一个大的神经网络。

type	patch size/stride or remarks	input size
conv	$3 \times 3 / 2$	$299 \times 299 \times 3$
conv	$3 \times 3 / 1$	$149 \times 149 \times 32$
conv padded	$3 \times 3 / 1$	$147 \times 147 \times 32$
pool	$3 \times 3 / 2$	$147 \times 147 \times 64$
conv	$3 \times 3 / 1$	$73 \times 73 \times 64$
conv	$3 \times 3 / 2$	$71 \times 71 \times 80$
conv	$3 \times 3 / 1$	$35 \times 35 \times 192$
$3 \times$ Inception	As in figure 5	$35 \times 35 \times 288$
$5 \times$ Inception	As in figure 6	$17 \times 17 \times 768$
$2 \times$ Inception	As in figure 7	$8 \times 8 \times 1280$
pool	$8 \times 8$	$8 \times 8 \times 2048$
linear	logits	$1 \times 1 \times 2048$
softmax	classifier	$1 \times 1 \times 1000$

第一种Module使用了两个 $3 \times 3$ 卷积替代了之前版本的 $5 \times 5$ 卷积，减小了计算量，但可以达到同样的效果。

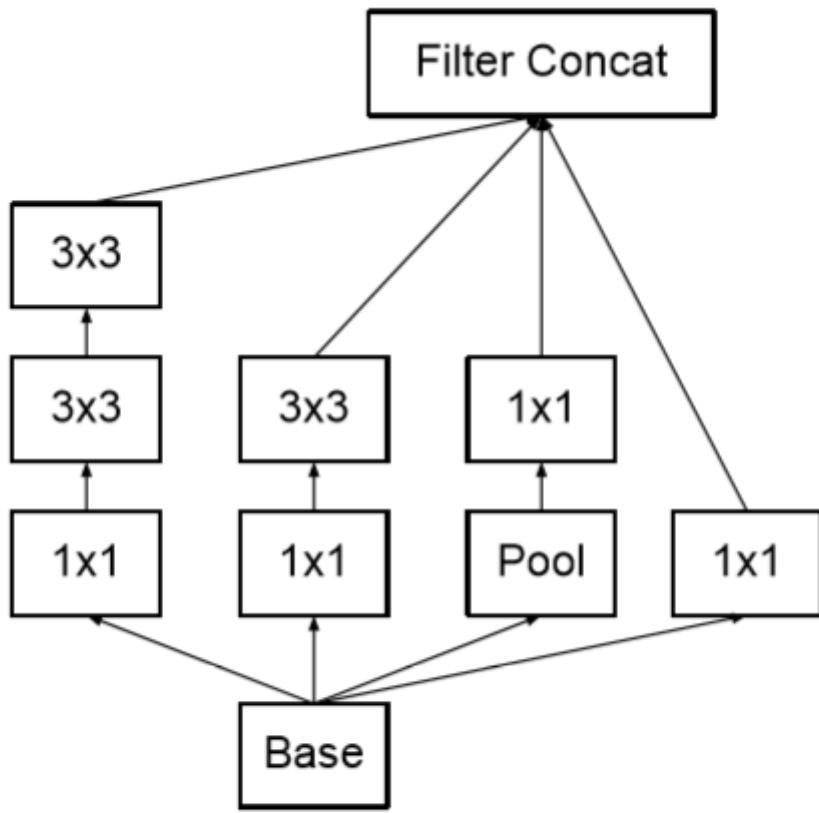
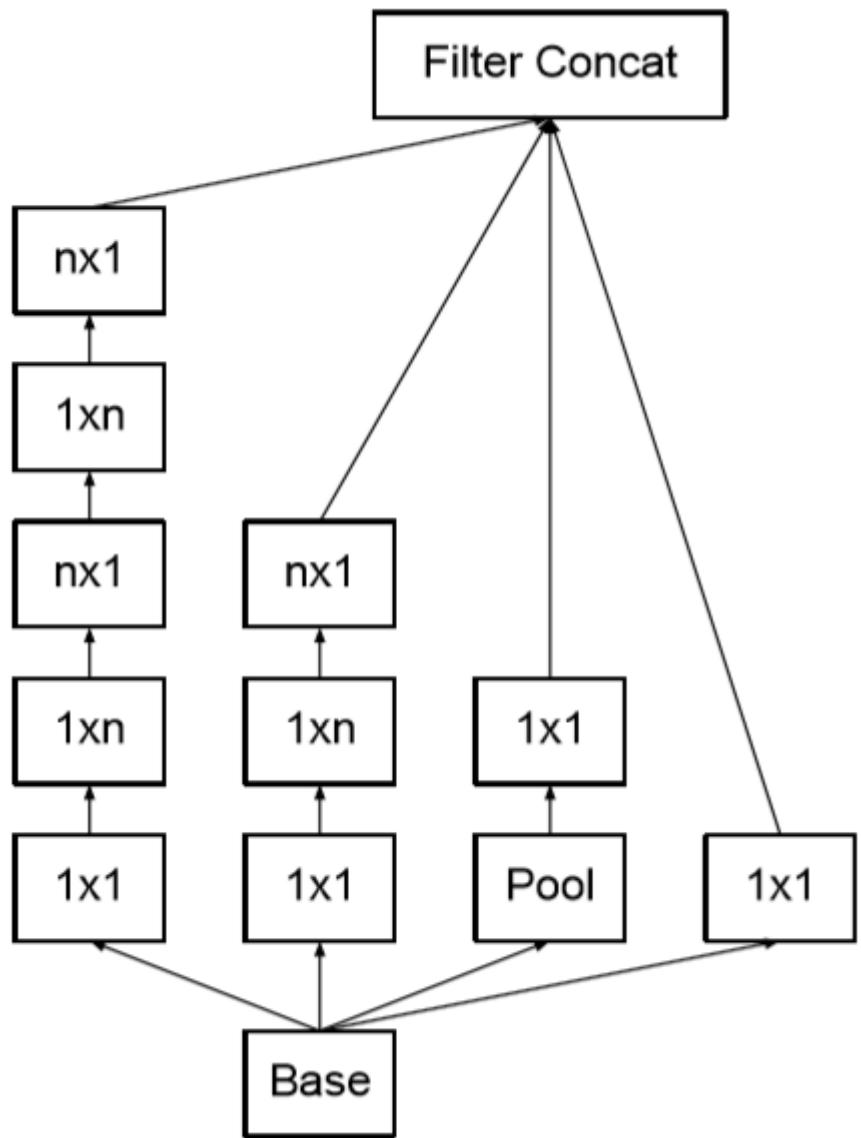
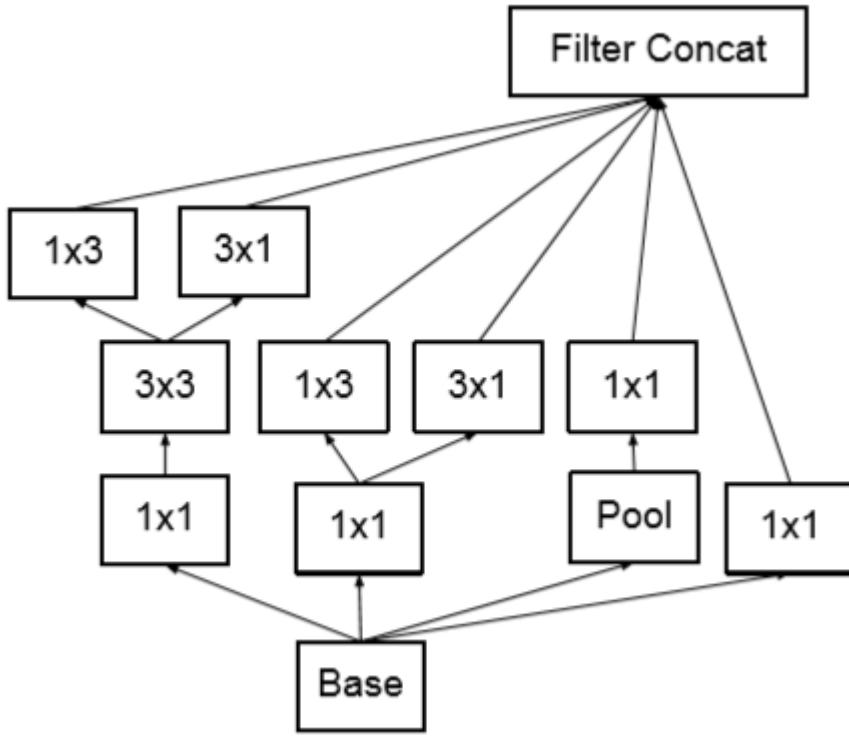


Figure 5. Inception modules where each  $5 \times 5$  convolution is replaced by two  $3 \times 3$  convolution, as suggested by principle 3 of Section 2.

第二种Module



第三种Module



每一个InceptionModule都会分为4个或更多的分支来进行不同的卷积处理，以便学习数据的更多形态，再进行融合。

InceptionNet在V2版本时还提出了Batch Normalization方法，是一个非常有效的正则化方法，可以让大型卷积网络的训练速度加快很多倍，同时收敛后的分类准确率也可以得到大幅提高。

BN在用于神经网络某层时，会对每一个mini-batch数据的内部进行标准化（normalization）处理，使输出规范化到 $N(0,1)$ 的正态分布。传统的深度神经网络在训练时，每一层的输入的分布都在变化，导致训练变得困难，我们只能使用一个很小的学习速率解决这个问题。而对每一层使用BN之后，我们就可以有效地解决这个问题，学习速率可以增大很多倍，达到之前的准确率所需要的迭代次数只有1/14，训练时间大大缩短。

## ResNet50

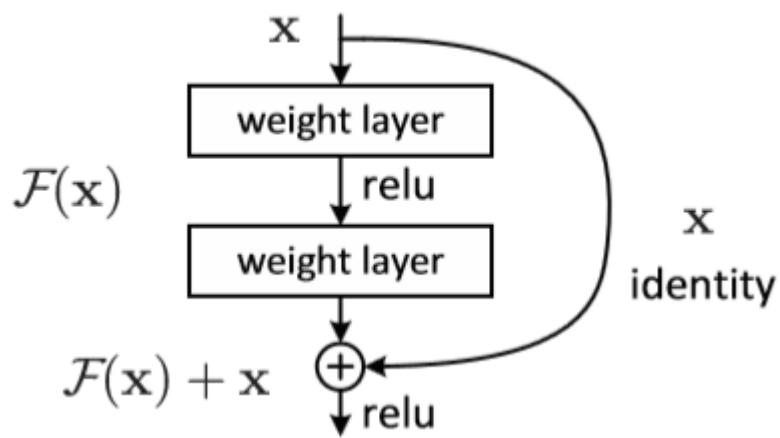
ResNet (Residual Neural Network) 由微软研究院的Kaiming He等4名华人提出，通过使用Residual Unit成功训练152层深的神经网络，在ILSVRC 2015比赛中获得了冠军，取得3.57%的top-5错误率，同时参数量却比VGGNet低，效果非常突出。

ResNet的结构可以极快地加速超深神经网络的训练，模型的准确率也有非常大的提升。

ResNet最初的灵感出自这个问题：在不断加神经网络的深度时，会出现一个Degradation的问题，即准确率会先上升然后达到饱和，再持续增加深度则会导致准确率下降。

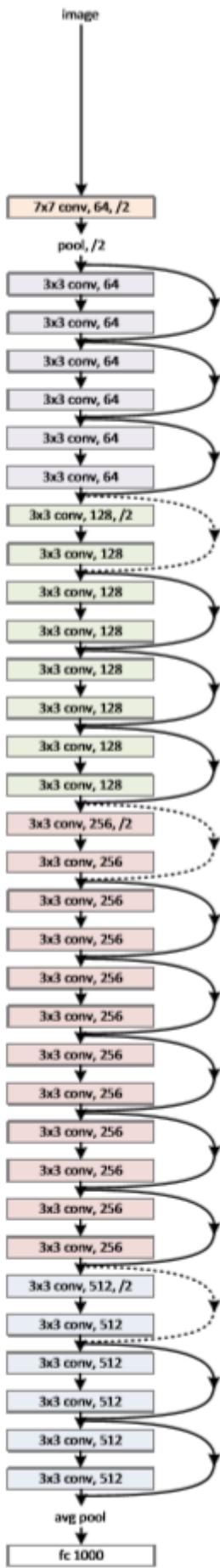
这并不是过拟合的问题，因为不光在测试集上误差增大，训练集本身误差也会增大。假设有一个比较浅的网络达到了饱和的准确率，那么后面再加上几个的全等映射层，起码误差不会增加，即更深的网络不应该带来训练集上误差上升。

而这里提到的使用全等映射直接将前一层输出传到后面的思想，就是ResNet的灵感来源。假定某段神经网络的输入是 $x$ ，期望输出是 $H(x)$ ，如果我们直接把输入 $x$ 传到输出作为初始结果，那么此时我们需要学习的目标就是 $F(x)=H(x)-x$ 。



这就是一个ResNet的残差学习单元 (Residual Unit)，ResNet相当于将学习目标改变了，不再是学习一个完整的输出 $H(x)$ ，只是输出和输入的差别 $H(x)-x$ ，即残差。

### 34-layer residual

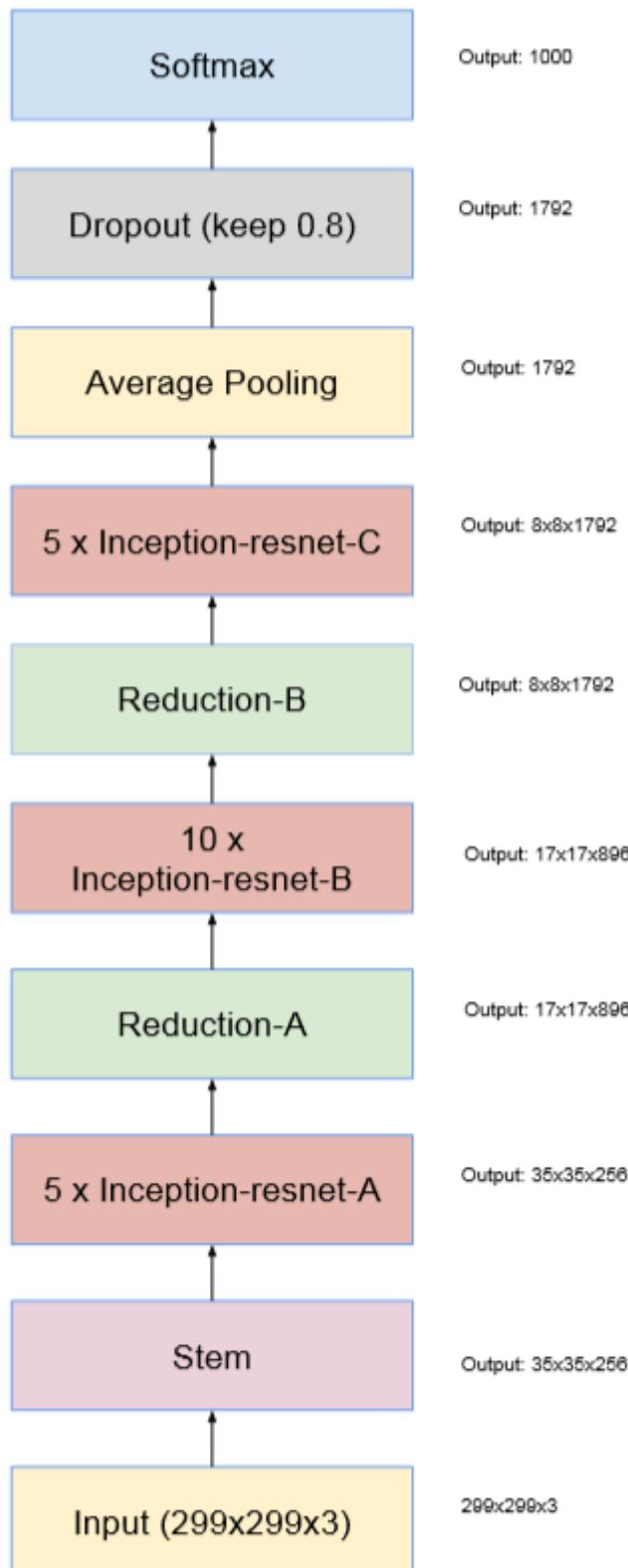


ResNet有很多旁路的支线将输入直接连到后面的层，使得后面的层可以直接学习残差

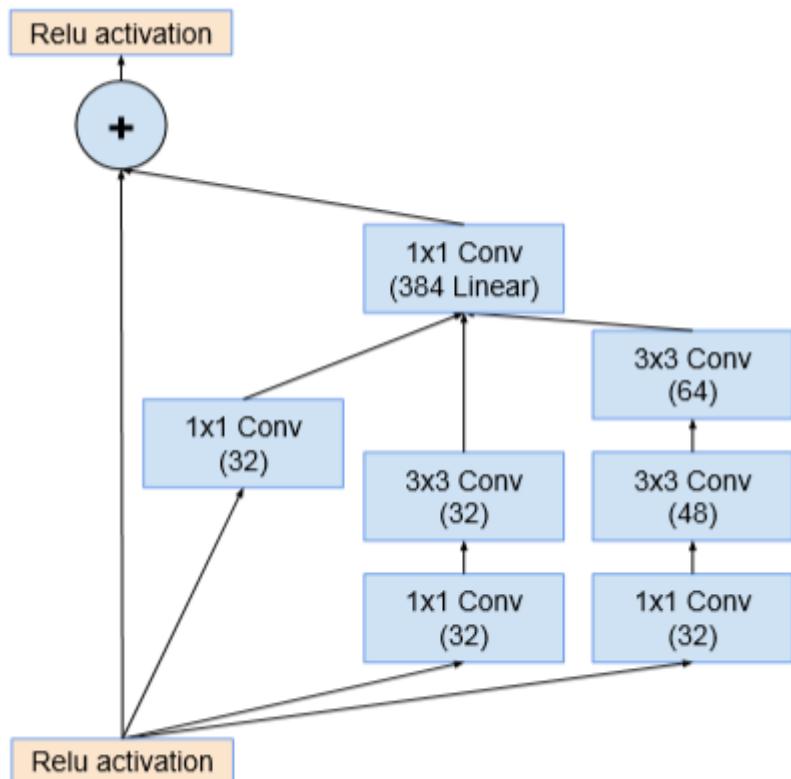
传统的卷积层或全连接层在信息传递时，或多或少会存在信息丢失、损耗等问题。ResNet在某种程度上解决了这个问题，通过直接将输入信息绕道传到输出，保护信息的完整性，整个网络则只需要学习输入、输出差别的那一部分，简化学习目标和难度。

## InceptionResNet

InceptionResNet（也可以叫InceptionV4）网络也是由Google发布，由InceptionV3借助ResNet残差网络思想演变而来。



以下为InceptionResNetV2网络中使用的其中一个模块，可以看到有一个从模块开始直接到结束的线路，应用了ResNet中的残差网络思想



这是另一个使用了残差网络思想的模块，该模块为上图中的Inception-resnet-B模块

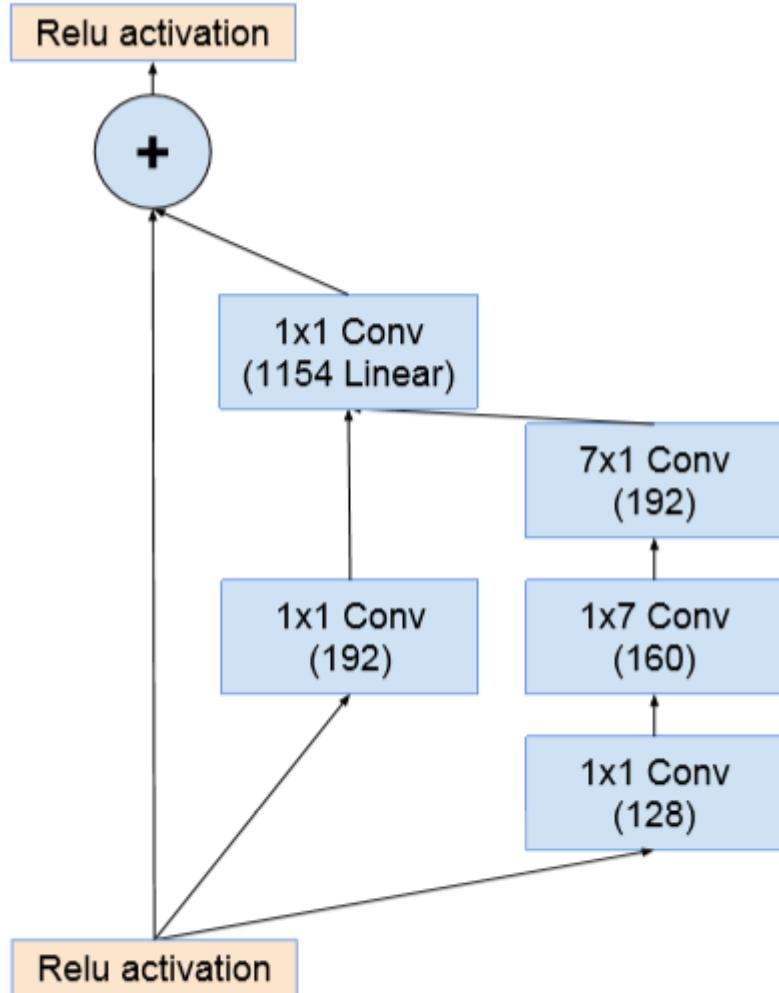


Figure 17. The schema for  $17 \times 17$  grid (Inception-ResNet-B) module of the Inception-ResNet-v2 network.

## Xception

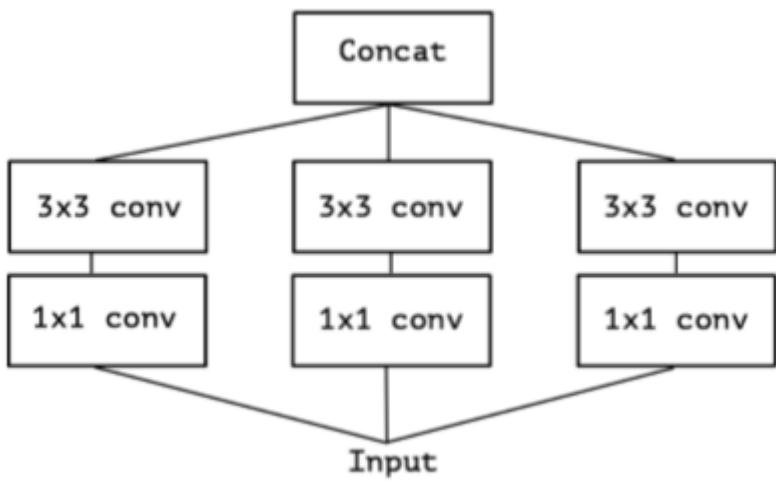
Xception也是由Google发布的深度神经网络架构。公开于2017年4月

Inception模块是一大类在ImageNet上取得顶尖结果的模型的基本模块，例如GoogLeNet、Inception V2/V3和Inception-ResNet。有别于VGG等传统的网络通过堆叠简单的 $3 \times 3$ 卷积实现特征提取，Inception模块通过组合 $1 \times 1$ 、 $3 \times 3$ 、 $5 \times 5$ 和pooling等结构，用更少的参数和更少的计算开销可以学习到更丰富的特征表示。

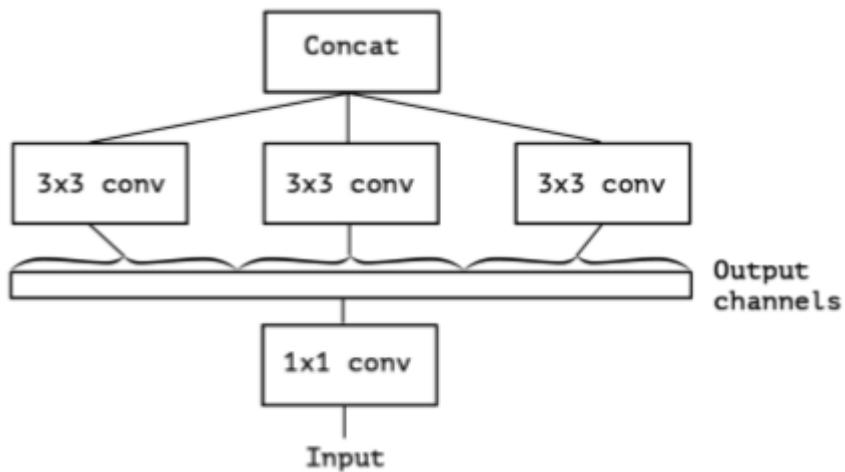
通常，在一组特征图上进行卷积需要三维的卷积核，也即卷积核需要同时学习空间上的相关性和通道间的相关性。将这两种相关性显式地分离开来，是Inception模块的思想之一：Inception模块首先使用 $1 \times 1$ 的卷积核将特征图的各个通道映射到一个新的空间，在这一过程中学习通道间的相关性；再通过常规的 $3 \times 3$ 或 $5 \times 5$ 的卷积核进行卷积，以同时学习空间上的相关性和通道间的相关性。

但此时，通道间的相关性和空间相关性仍旧没有完全分离，也即 $3 \times 3$ 或 $5 \times 5$ 的卷积核仍然是多通道输入的，那么是否可以假设它们可以被完全分离？显然，当所有 $3 \times 3$ 或 $5 \times 5$ 的卷积都作用在只有一个通道的特征图上时，通道间的相关性和空间上的相关性即达到了完全分离的效果。

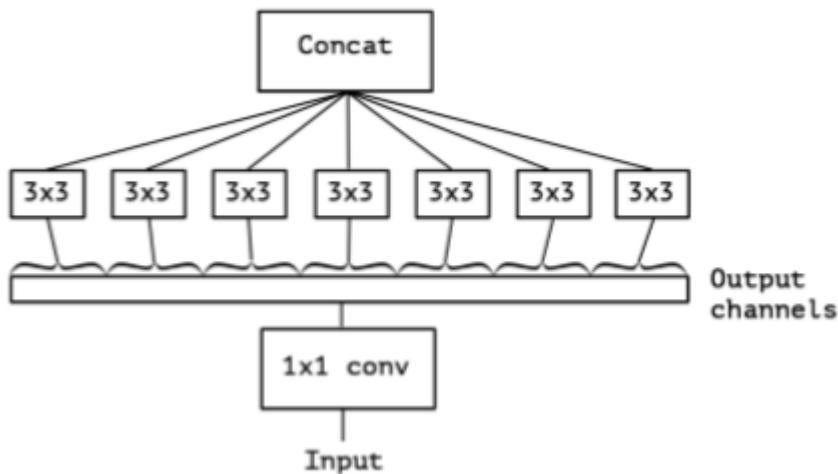
若将Inception模块简化，仅保留包含 $3 \times 3$ 的卷积的分支：



再将所有 $1 \times 1$ 的卷积进行拼接：

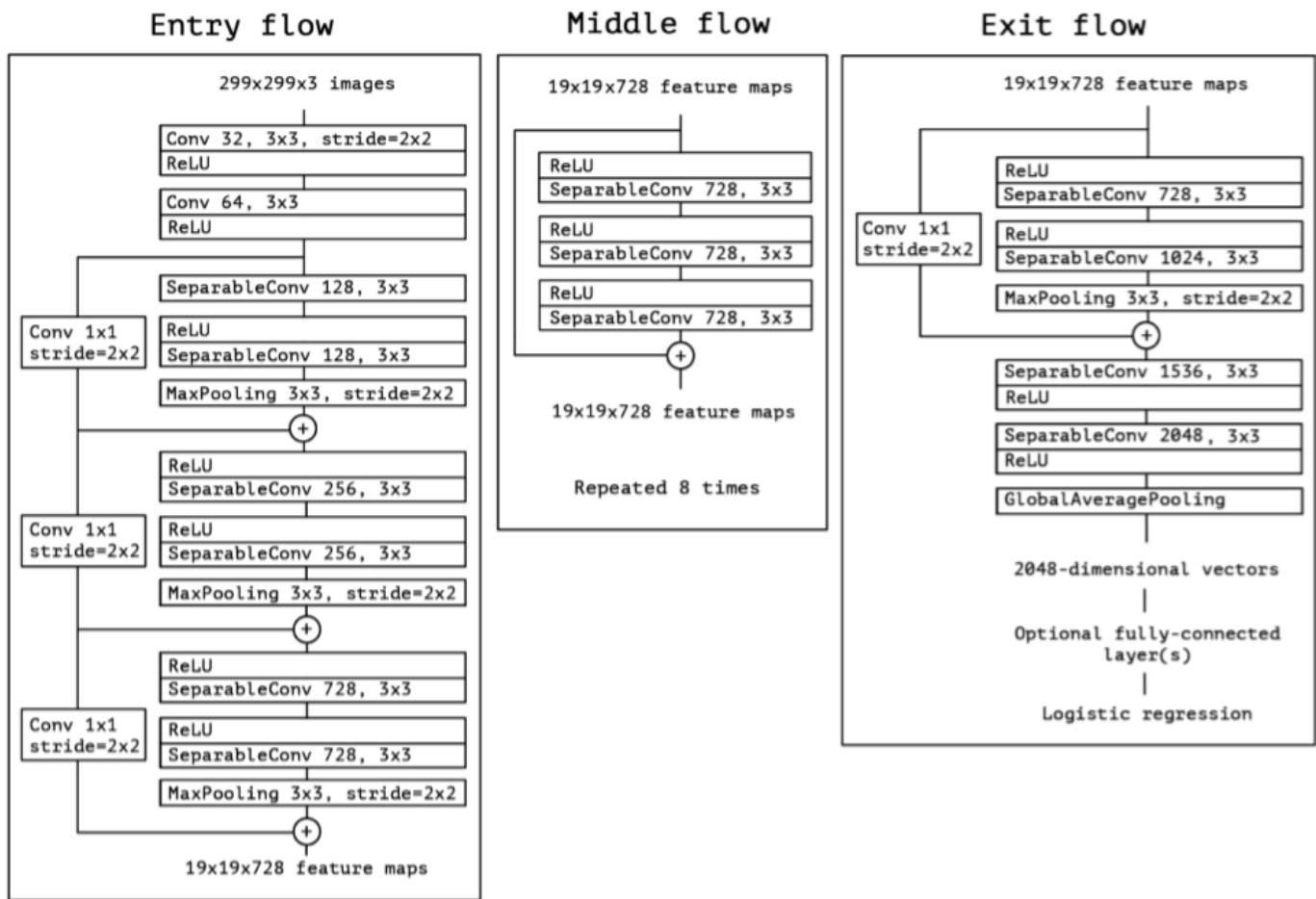


进一步增多 $3 \times 3$ 的卷积的分支的数量，使它与 $1 \times 1$ 的卷积的输出通道数相等：



此时每个 $3 \times 3$ 的卷积即作用于仅包含一个通道的特征图上，作者称之为“极致的Inception（Extream Inception）”模块，这就是Xception的基本模块。事实上，调节每个 $3 \times 3$ 的卷积作用的特征图的通道数，即调节 $3 \times 3$ 的卷积的分支的数量与 $1 \times 1$ 的卷积的输出通道数的比例，可以实现一系列处于传统Inception模块和“极致的Inception”模块之间的状态。

Xception网络则由这些“极致的Inception”模块构成，以下是结构图：



数据将先经过Entry flow区，再经过Middel flow区并在该区域重复执行8次，最后经过Exit flow。其中所有的卷积层及分离卷积层后面都使用了Batch Normalization方法。

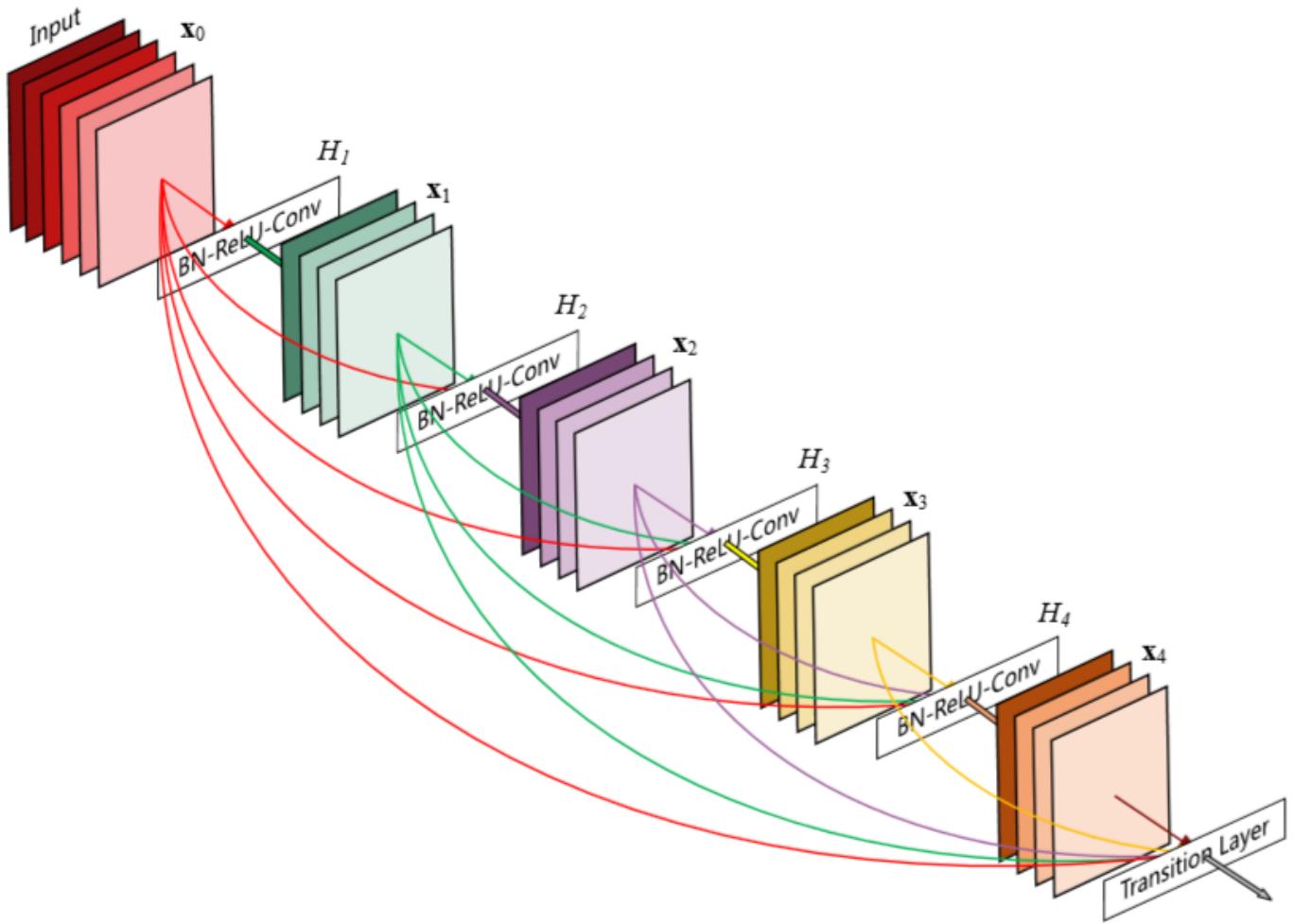
## DenseNet

DenseNet发布于2017年，是CVPR 2017 Best Paper。DenseNet中的DenseBlock的一个重要思想就是对前每一层都增加一个单独的shortcut，使得任意两层网络都可以直接“沟通”，也是参考了ResNet的残差网络思想，但使用的更多、更广，不过由此带来的代价就是训练时占用的显存更多。

DenseNet和ResNet的一个明显区别是，ResNet是求和，而DenseNet是做一个拼接，每一层网络的输入包括前面所有层网络的输出。第L层的输入等于 $k \times (L - 1) + k_0$ ，其中k是生长率，表示每一层的通道数，比如下图网络的通道数为4。

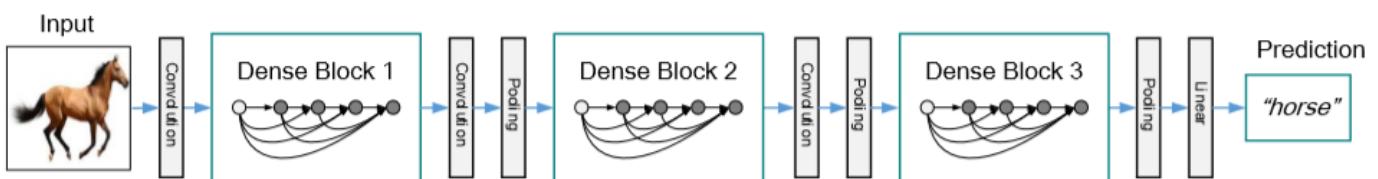
DenseNet提升了信息和梯度在网络中的传输效率，每层都能直接从损失函数拿到梯度，并且直接得到输入信号，这样就能训练更深的网络，这种网络结构还有正则化的效果。其他网络致力于从深度和宽度来提升网络性能，

DenseNet致力于从特征重用的角度来提升网络性能



这种结构的好处是可以缓解梯度消失，省参数省计算，特征重用可以起到抗过拟合的作用。达到相同的精度，dense net只需要res net一半的参数和一半的计算量。

下图是一个完整的DenseNet网络结构，其中有3个DenseBlock：

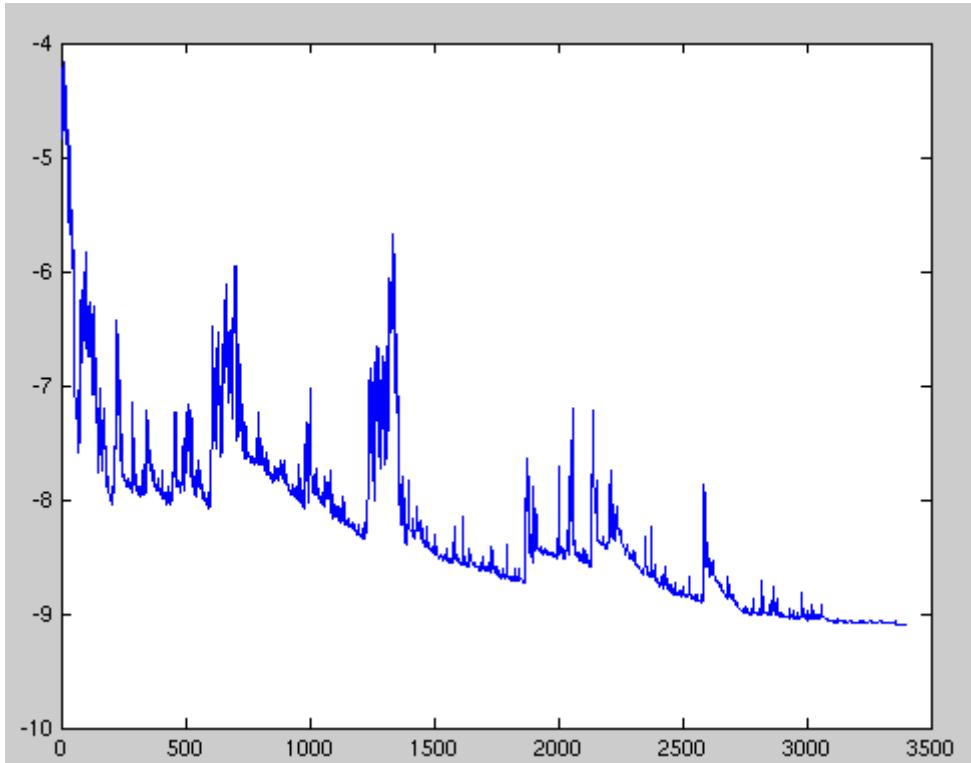


## SGD

随机梯度下降优化器。和 BGD 的一次用所有数据计算梯度相比，SGD 每次更新时对每个样本进行梯度更新。对于很大的数据集来说，可能会有相似的样本，这样 BGD 在计算梯度时会出现冗余，而 SGD 一次只进行一次更新，就没有冗余，而且比较快，并且可以新增样本。

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta; x^{(i)}; y^{(i)})$$

但是 SGD 因为更新比较频繁，会造成 cost function 有严重的震荡。



## Adam

这个算法是另一种计算每个参数的自适应学习率的方法。存储了过去梯度的平方 $v_t$ 的指数衰减平均值，也保持了过去梯度 $m_t$ 的指数衰减平均值：

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1 - \beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \end{aligned}$$

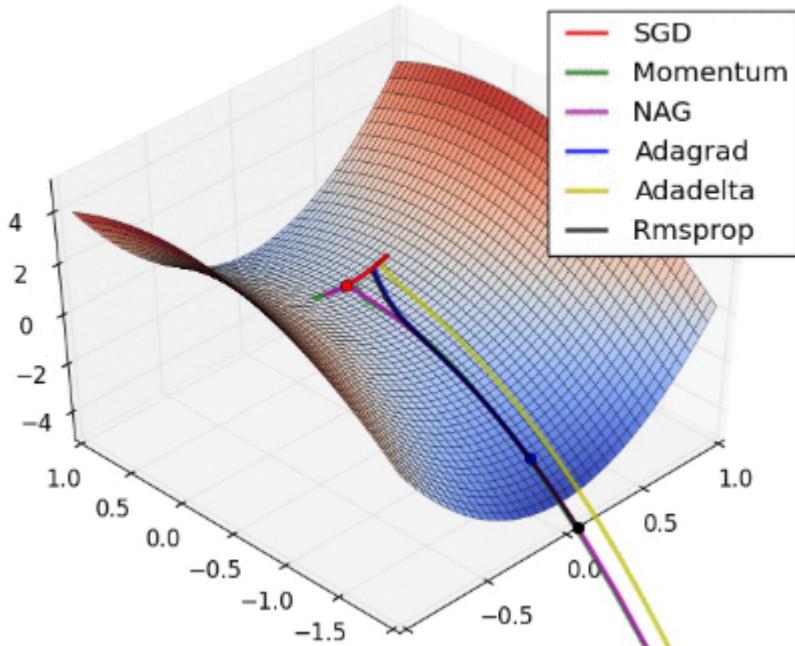
如果 $m_t$ 和 $v_t$ 被初始化为0向量，那它们就会向0偏置，所以做了偏差校正，通过计算偏差校正后的 $\hat{m}_t$ 和 $\hat{v}_t$ 来抵消这些偏差：

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1 - \beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1 - \beta_2^t} \end{aligned}$$

然后使用这些变量来更新学习参数：

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

实践表明，Adam比SGD优化方法效果更好，学习速度更快，也不像SGD容易受鞍点影响导致停留在局部极小值。



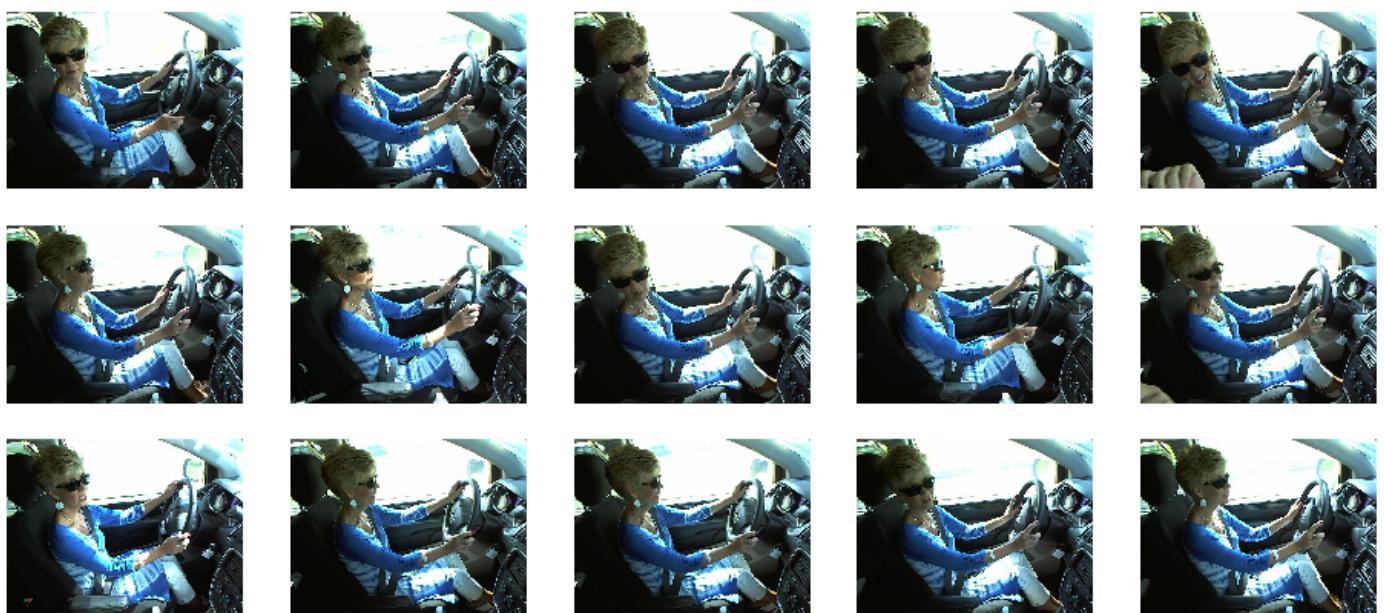
## 基准模型

使用Kaggle中该项目的排名分数做为基准模型。使用前10%的分数作为基准，第144名，最小损失值为0.25634。

## III. 方法

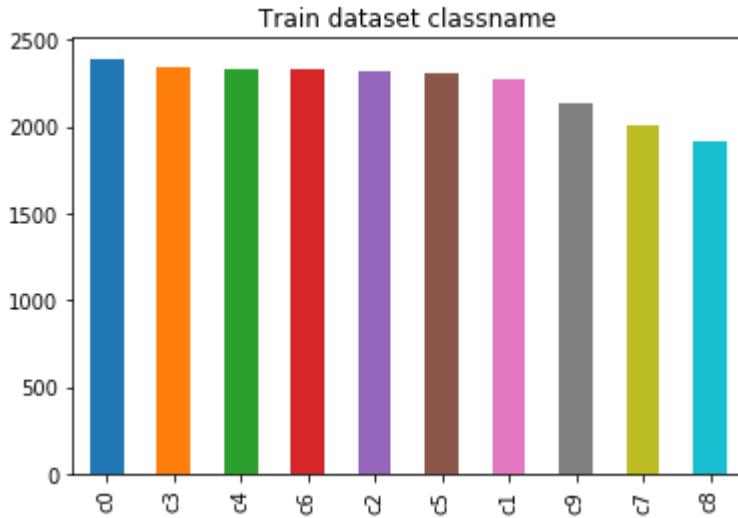
### 数据预处理

经过抽察数据集发现，ID为p081的这名司机在c0安全驾驶类别中存在多张图像能判断为与其他乘客聊天的状态，属于异常数据，决定弃用p081在c0类别中的所有数据。如：img\_6002.jpg、img\_28578.jpg、img\_45571.jpg、img\_47447.jpg、img\_49150.jpg、img\_51985.jpg、img\_5963.jpg、img\_3349.jpg



`driver_imgs_list.csv` 数据表就像一个训练图像集的索引文件，通过它可引用到对应的图像文件。使用该文件中的司机ID的指引，排除掉ID为p081的司机在c0分类中的图像数据（并不删除p081司机在c0文件夹中的图像，仅仅在该csv文件中删除p081在c0分类的文件索引），排除后生成一份新的csv文件 `drivers_img_nop081_list.csv`，此后的数据处理和训练都将基于该数据索引文件来处理。

清除p081的司机在c0分类中的图像索引后，新的csv数据表中各类数据分布仍然呈平均分布：



排除p081的数据后，总共有22342张图像数据可用。

查看原始图像文件，这里选取数据表中同一个司机ID的前10张图片，发现几乎完全一样，连肉眼都不能清楚分辨，图片的重复程度非常高。如下图片展示：



参考了原比赛中前几名选手的方案，决定使用图像拼接的方式对其进行数据增强处理，目的是为了让模型在训练时更多地关注司机的行为，如：双手握盘安全驾驶、拿手机打电话、拿手机发短信、转身拿后方东西等，而不让模型去记住这些司机的样貌、行态、穿着。图像拼接后的效果：



使用新的 `driver_imgs_nop081_list.csv` 数据索引文件处理每一份图像，拼接时将每张图像的右半块替换为与该图像相同分类下的随机另一张图数据的右半块，并保存拼接后的新图像到新的 `data/imgs/train_aug` 文件夹中，同样按c0-c9文件夹分类存放好，并记录在新的数据索引文件 `drivers_img_aug_list.csv` 中，记录时同样存放司机ID、所属分类、图像文件名这些信息，便于后期分割验证集和训练时使用。执行图像拼接数据增强后，其中可用的数据集增加到了44648张图像，大大丰富了训练集。

## 执行过程

该项目执行时将使用Keras框架中各带有imagenet预训练权重的深度学习模型，如：InceptionV3、Xception、InceptionResNetV2、ResNet50、DenseNet201。使用这些模型进行训练时都将排除掉原模型中的顶层全连接层，加入全局平均池化层对模型的高层特征向量进行降维，并加入Dropout层，Dropout层将在训练过程中按一定的概率将网络中的神经元暂时丢弃，减少模型的训练参数，防止过拟合。为了在学习过拟合、欠拟合之间找到一个平衡，Dropout层的参数将设置为0.5。Dropout参数大，丢弃率高，学习速度慢，同样的精度需要的训练代数更多，容易欠拟合；Dropout参数小，丢弃率低，学习速度快，模型更早进入过拟合状态。

再在模型的最顶部加入一个10分类的全连接层（因项目有10种数据类别），使用 `use_bias=False, kernel_regularizer=l2(0.01)` 参数去掉全连接层的偏移项，并加入L2正则对训练参数进行惩罚，以此来防止过拟合。

因数据集较多，在向模型中fit训练数据时使用`ImageDataGenerator`来分批fit图像，并对图像做旋转、缩放、翻转等数据增强处理

## 验证集分割

这里因为数据集中的司机图像是从视频中截取出来的，可能存在两张甚至多种几乎一样的图像分别位于训练集和验证集中。训练后做验证时因为验证集存在几乎相同的图像，会导致验证分数被提高，但实际上模型仅仅是记住了该图片，因此分割验证集里需要采用一些策略：使用司机ID来做验证集分割。

因使用图像拼接的方法生成了更多的图像文件及新的数据索引表 `drivers_img_aug_list.csv`，因此分割验证集将更麻烦一些，需要同时处理源始图像及拼接图像。这里编写了一个 `split_valid.py` 模块，其

中中 `split` 方法专门用于处理验证集分割。传入需要作为验证集的司机ID、训练集目录、验证集目录等参数后，将分割验证集图像到新的目录中，便于模型训练时使用。

这里采用软链接的方式将分割后的验证集图像链接到新的目录 `data/imgs/val2` 中，这样减少COPY图像文件的时间，迅速分割，也减少磁盘占用。因为我们验证集的目的主要是验证模型在原图像集中的精确度和泛化能力，所以并没有将拼接后的图像提取出来也作为验证集。

训练集也同样采用软链接的方式将原始图像和拼接后的图像都链接到新的一个训练目录 `data/imgs/train2` 中，并按分类文件夹存放。

分割的效果如下：

```
split_valid.split(choice_ids=choices,
                   train_pd_path="data/drivers_img_nop081_list.csv",
                   train_aug_pd_path="data/drivers_img_aug_list.csv",
                   train_dir=train_dir,
                   val_dir=val_dir,
                   test_dir=test_dir,
                   origin_test_dir="data/imgs/test",
                   saved_weights_dir="saved_weights")
```

output:

```
选择作为验证集的司机ID: ['p024', 'p002']
分割的训练集数量: 20373 , 验证集数量: 1951
从增强数据集中分割的训练集数量: 20373
link images to directory data/imgs/train2
link augmented images to directory data/imgs/train2
link images to directory data/imgs/val2
split valid data done!
```

## 使用clip来防止logloss分数过高

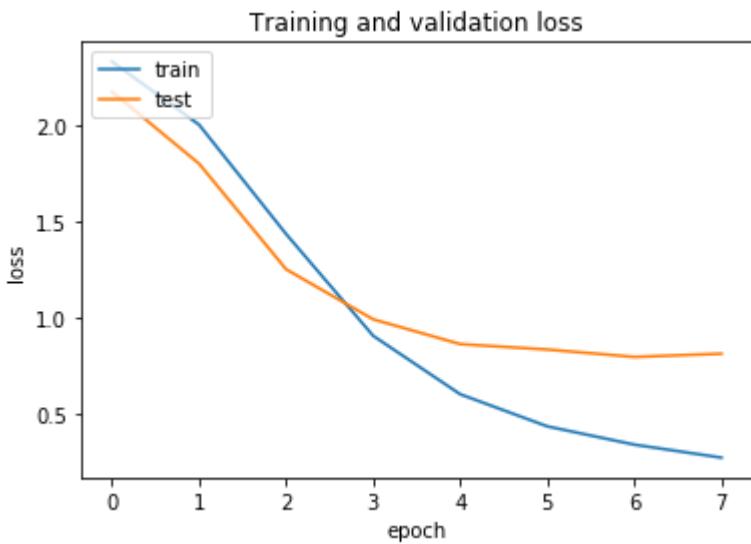
这里用到了一个小技巧，将每个预测结果的值限制到[0.005, 0.995]这个区间，因为kaggle的评估标准是LogLoss，当值很小时 $\log$ 的结果相差非常大， $\log(0) \approx -\infty$ ， $\log(0.005) = -5.30$ ，而 $\log(0.995) = -0.005$ 和 $\log(1) = 0$ 相差也不大，处理后对kaggle评分有一定好处。

## 使用InceptionV3模型训练

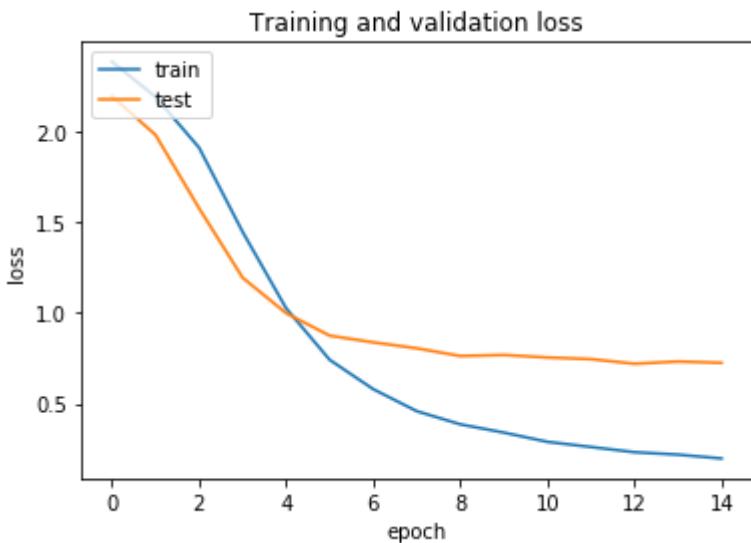
这里使用Keras框架中的InceptionV3模型训练，并使用imagenet的预训练权重。具体实现代码：  
`inceptionv3.ipynb`

### 1. 未使用拼接图像的模型训练

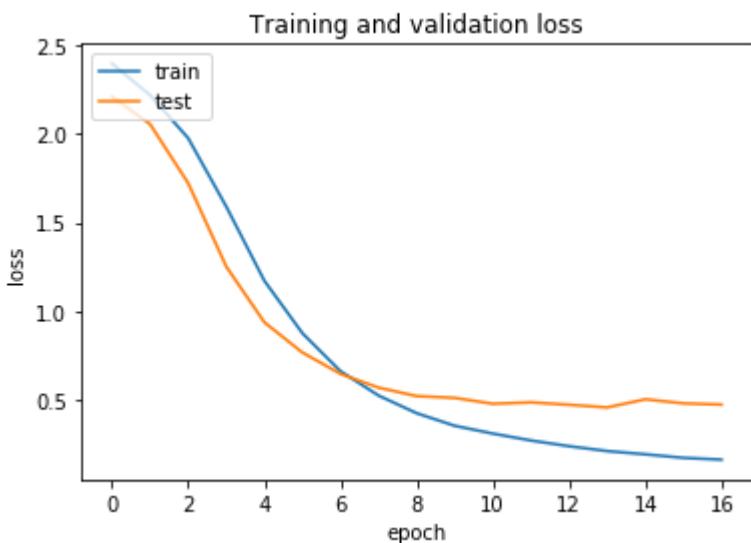
使用SGD优化器，学习率为0.0003，衰减为9e-8，未排除全连接层的偏置项，没有使用L2正则化，可以看到模型`val_loss`在下降到0.8几的时候就出现了过拟合现象



SGD优化器学习率降为0.0002，衰减为9e-8，未排除全连接层的偏置项，没有使用L2正则化，模型过拟合现象有减少，val\_loss在下降到大概0.7几时才出现过拟合

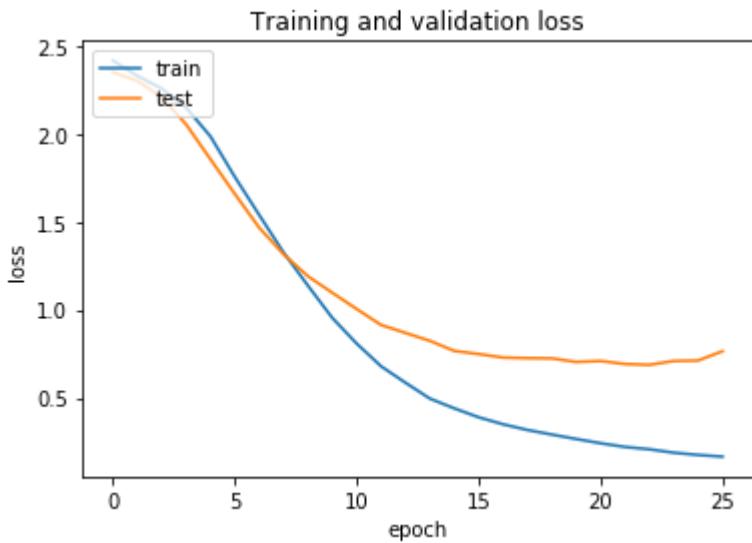


SGD优化器学习率降为0.0002，衰减为20e-8，排除了全连接层的偏置项，没有使用L2正则化，发现模型过拟合现象有明显减少，val\_loss在下降到大概0.5左右才出现过拟合。



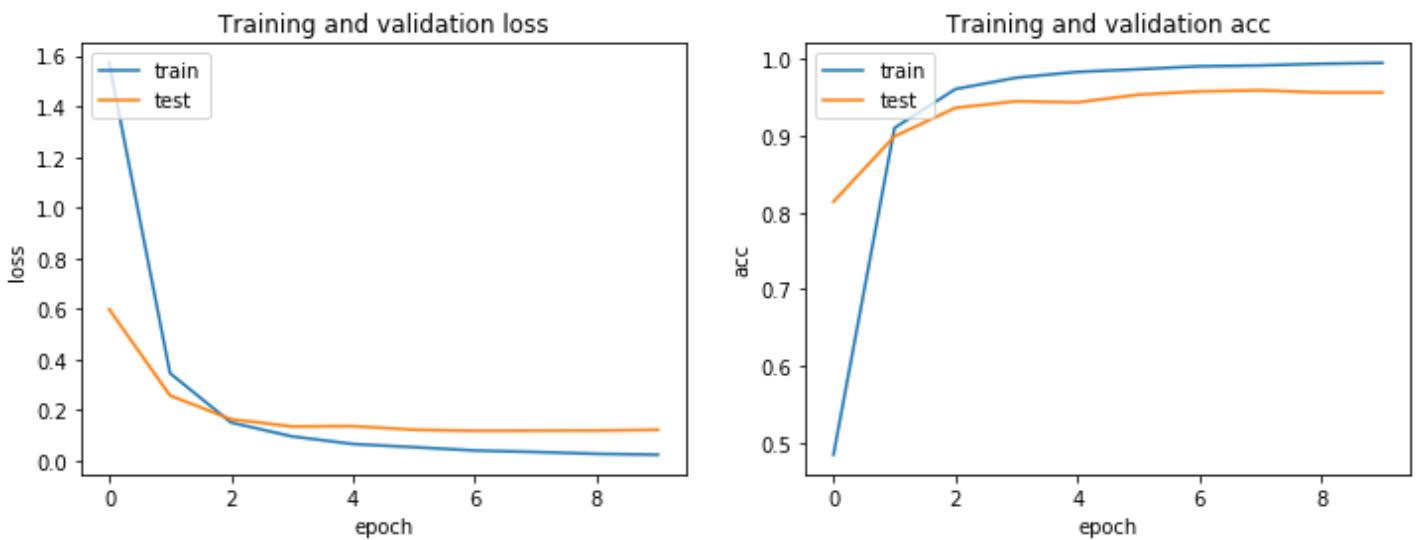
由上可见降低学习率，提高衰减，排除全连接层偏置项是可以防止过拟合的。

再次尝试通过锁层的方式来防止过拟合，锁定InceptionV3模型中的第229-310层，也即不训练最后3个块，使用SGD优化器，学习率为0.0003，衰减为30e-8，排除全连接层的偏置项，发现模型训练非常缓慢，处于欠拟合状态，不适合使用该方法来防止过拟合



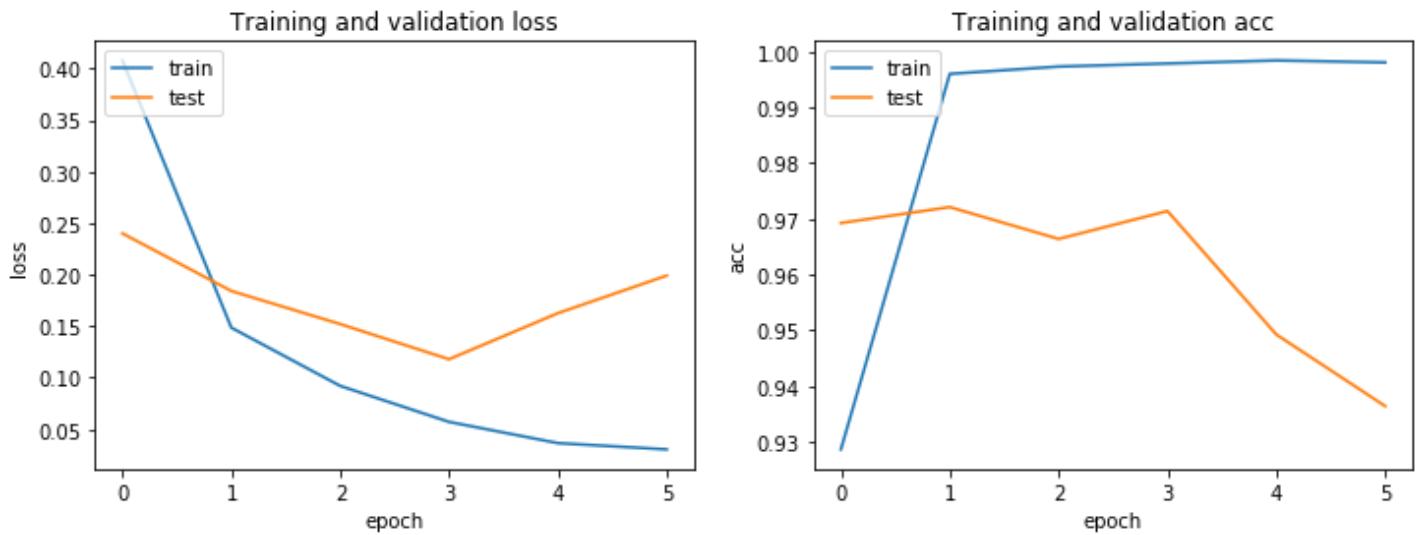
## 2. 使用拼接图像的模型训练

使用SGD优化器，学习率为0.0003，衰减为30e-8，未排除全连接层的偏置项，没有使用L2正则化，实际使用的训练集是原来的2倍，可发现模型的训练效果明显提升，val\_loss已能下降到0.11左右：



使用该模型对测试集进行预测后得到分数：private 0.41971, public 0.63264。看来我们离设定的目标大大近了一步。

因SGD优化器收敛速度较慢，且可能下降到一个局部最优值后无法再次下降，因此决定使用Adam优化器，并继续排除全连接层的偏置项，加入L2正则化来更进一步防止过拟合。学习速度为0.0001，衰减为6e-8。可看到Adam优化器可快速使loss分数下降，迅速达到极小值附近，但可能这里设定的学习率参数较大，模型训练中出现了较大的波动。



使用该模型对测试集进行预测后得到分数：private: 0.31309, public: 0.35854。更加接近设定的目标。

再次修改Adam优化器参数，学习率降低为0.00005，衰减设为2e-8，并使用Keras框架中的 ModelCheckpoint 和 EarlyStopping 使模型出现过拟合时保存最优训练模型，多次调整尝试后得到的最优结果为：

```
loss: 0.0148 - acc: 0.9993 - val_loss: 0.0248 - val_acc: 0.9959
```

使用该模型对测试集进行预测后得到分数：private: 0.28068, public: 0.31967。经过多次尝试后发现该参数训练出的模型得出的分数是最优的，因此最终采用了该模型。

预测效果查看：

c5 |operation radio| 99.94%



c5 |operation radio| 99.92%



c0 |safe driving| 99.63%



c8 |hair and makeup| 99.64%



c3 |texting - left| 99.88%



c3 |texting - left| 99.92%



c8 |hair and makeup| 85.52%

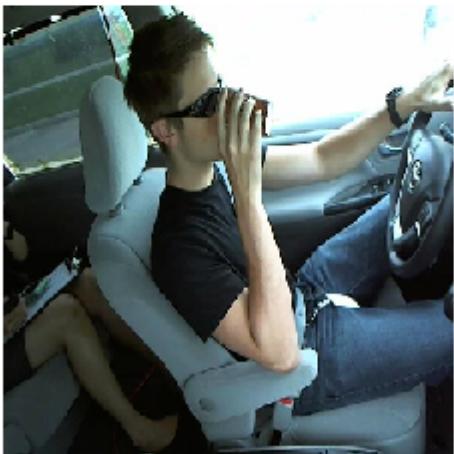


c9 |talking| 52.85%





c6 |drinking| 99.83%



c7 |reaching behind| 99.95%



## 使用Xception模型训练

这里使用Keras框架中的Xception模型训练，并使用Imagenet的预训练权重。具体实现代码：  
xception.ipynb

因为在训练InceptionV3模型时已尝试不使用拼接图像进行训练，发现效果不，因此该模型及后续模型的训练都将使用原始图像+拼接图像的方式训练。多次调整尝试后，使用的最终训练参数如下：

- Adam优化器
- 学习率：0.00003
- 衰减：1e-8
- 排除全连接层偏置项
- 加入L2正则化

最优训练结果：loss: 0.0254 - acc: 0.9990 - val\_loss: 0.0410 - val\_acc: 0.9931

提交到kaggle中后得到成绩：private: 0.30469, public: 0.35846

预测效果查看：

c5 |operation radio| 99.73%



c5 |operation radio| 99.68%



c0 |safe driving| 97.56%



c8 |hair and makeup| 99.90%



c3 |texting - left| 99.90%



c3 |texting - left| 99.88%



c8 |hair and makeup| 96.85%

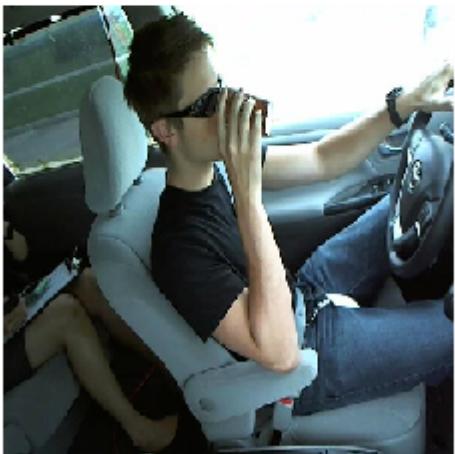


c0 |safe driving| 80.56%





c6 |drinking| 99.61%



c7 |reaching behind| 99.92%



## 使用InceptionResNetV2模型训练

这里使用Keras框架中的InceptionResNetV2模型训练，并使用imagenet的预训练权重。具体实现代码：InceptionResNetV2.ipynb

多次调整尝试后，使用的最终训练参数如下：

- Adam优化器
- 学习率：0.00003
- 衰减： $1e-8$
- 排除全连接层偏置项
- 加入L2正则化

最优训练结果：loss: 0.0267 - acc: 0.9986 - val\_loss: 0.1077 - val\_acc: 0.9731

提交到kaggle中后得到成绩：private: 0.32819, public: 0.43092

预测效果查看：

c5 |operation radio| 99.77%



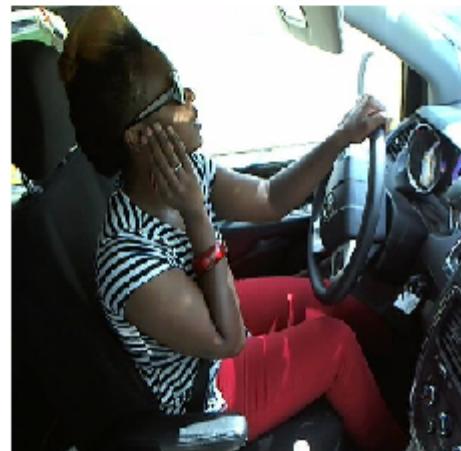
c5 |operation radio| 99.95%



c0 |safe driving| 99.44%



c8 |hair and makeup| 99.98%



c3 |texting - left| 99.93%



c3 |texting - left| 99.83%



c8 |hair and makeup| 99.26%



c0 |safe driving| 98.80%





## 使用ResNet50模型训练

这里使用Keras框架中的ResNet50模型训练，并使用imagenet的预训练权重。具体实现代码：  
resnet50.ipynb

多次调整尝试后，使用的最终训练参数如下：

- Adam优化器
- 学习率：0.00005
- 衰减：2e-8
- 排除全连接层偏置项
- 加入L2正则化

最优训练结果：loss: 0.2102 - acc: 0.9933 - val\_loss: 0.2963 - val\_acc: 0.9575

提交到kaggle中后得到成绩：private: 0.37406, public: 0.42121

预测效果查看：

c5 |operation radio| 99.90%



c5 |operation radio| 99.96%



c0 |safe driving| 98.87%



c8 |hair and makeup| 99.92%



c3 |texting - left| 99.95%



c3 |texting - left| 99.98%



c8 |hair and makeup| 79.45%

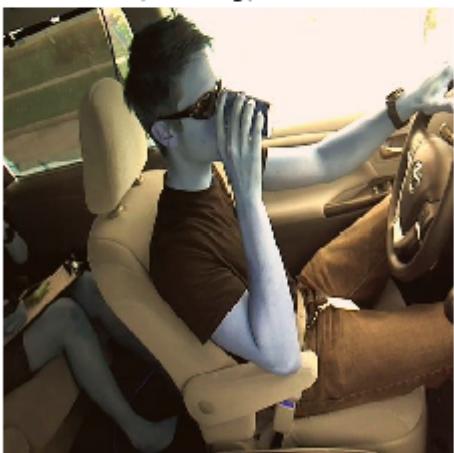


c0 |safe driving| 44.26%





c6 |drinking| 99.90%



c7 |reaching behind| 100.00%



## 使用DenseNet201

这里使用Keras框架中的DenseNet201模型训练，并使用imagenet的预训练权重。具体实现代码：  
densenet201.ipynb

多次调整尝试后，使用的最终训练参数如下：

- Adam优化器
- 学习率：0.00003
- 衰减： $1e-8$
- 排除全连接层偏置项
- 加入L2正则化

最优训练结果：loss: 0.0206 - acc: 0.9991 - val\_loss: 0.0540 - val\_acc: 0.9923

提交到kaggle中后得到成绩：private: 0.28420, public: 0.30792

预测效果查看：

c5 |operation radio| 99.86%



c5 |operation radio| 99.84%



c0 |safe driving| 99.73%



c8 |hair and makeup| 99.67%



c3 |texting - left| 99.90%



c3 |texting - left| 99.50%



c8 |hair and makeup| 93.23%

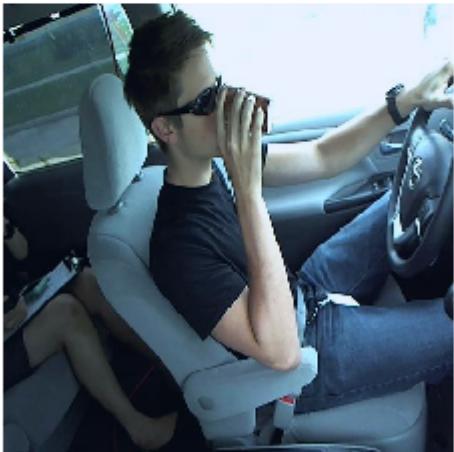


c0 |safe driving| 78.64%





c6 |drinking| 98.70%



c7 |reaching behind| 99.97%



## 模型融合

以上每一个模型在kaggle中的分数都在0.28以上，离我们设定的0.25的目标还有一段距离。决定使用模型融合的技术来继续降低该loss分数。模型融合类似于多个模型对它们的预测结果进行投票选举，通过该选举选出测试集中每张图片的最优预测结果，达到进一步提升模型准确度的目的。

以下是选取的各模型的训练比较：

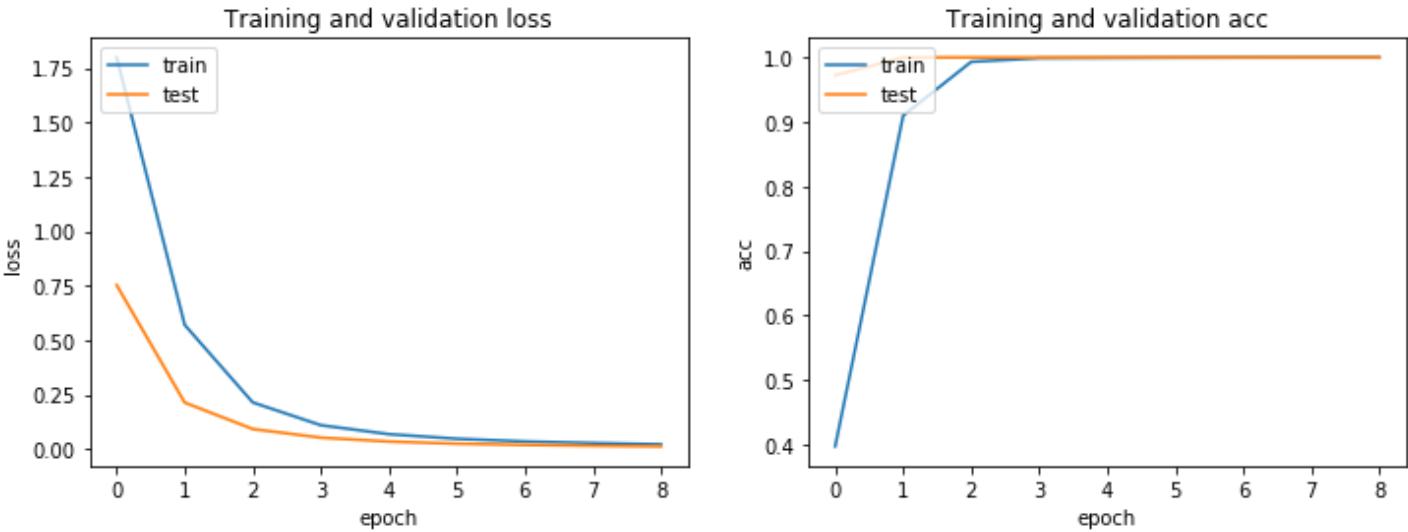
模型	训练acc	验证acc	训练loss	验证loss	Kaggle public	Kaggle private
InceptionV3	0.9993	0.9959	0.0148	0.0248	0.31967	0.28068
Xception	0.9990	0.9931	0.0254	0.0410	0.35846	0.30469
InceptionResNetV2	0.9986	0.9731	0.0267	0.1077	0.43092	0.32819
ResNet50	0.9933	0.9575	0.2102	0.2963	0.42121	0.37406
DenseNet201	0.9991	0.9923	0.0206	0.0540	0.30792	0.28420

从该比较中将选取最优的4个模型作为融合训练的模型，选取的模型为：InceptionV3、Xception、InceptionResNetV2、DenseNet201。

加载每一个需要融合的模型，去掉最后的全连接分类层，对训练集、验证集、测试集提取特征向量，保存到bottleneck文件中，具体实现代码：write\_bottleneck.ipynb。这里只提取原始图像的特征向量。

构建一个只有Dropout层和全连接分类层的最简模型，加载bottleneck文件中的特征向量进行训练。因为只对各模型提取出的高层次特征向量进行训练，因此训练速度将非常快：

loss: 0.0089 - acc: 0.9999 - val\_loss: 0.0050 - val\_acc: 1.0000



融合训练后就可以对测试集的特征向量进行预测了，最终预测成绩：

[mix\\_pred.csv](#)  
a few seconds ago by 竹子  
mixed+inceptionv3+inceptionresnetv2+xception+densenet201+epochs10

0.22399 0.24394



kaggle的private分数达到了0.22399的好成绩，在kaggle的排名表中可以排在第95名，也达到了最初设定的基准。

## IV. 项目结论

### 对项目的思考

由于该项目的数据源是来自于视频截图，存在许多大致重复的图像，且测试集中的图像数据大大超过训练集中的图像数据，因此该项目的主要难度就是防止过拟合。我在项目的初期常试了许多防止过拟合的手段，包括：降低学习率、增加Dropout丢弃率、去掉全连接层偏置参数、L2正则化、图像旋转翻转等数据增强，都能获得一些效果，但始终离期望的效果太远。后来吸取了原参赛者分享的经验，使用了图像拼接的数据增强手段后，效果大幅提升，成绩终于有所突破。最终提交到kaggle的分数结果是令人满意的，甚至有点点超出原来的期望的目标。

且其中训练集是由26个司机的图像组成的，存在一些特殊性，如果没有观察到该特殊性，则在划分验证集时将会出现问题，导致验证分数异常地高，而提交到kaggle的分数却会很低。

由上综述认为，在对模型进行训练时对数据集的观察和分析非常重要。数据集的大小、特性、处理方式都对模型训练产生着很大的影响。

### 需要作出的改进

后期我认为还可以继续使用图像拼接的方式丰富训练集，还可以对训练集的图像进行更细致的分析和数据增强，但这都需要耗费更多的时间。

另外也可以使用更多的模型来训练，这样可以有更多的模型做模型融合，俗话说人多力量大嘛，模型也是如此，有更多的优秀模型来参于“投票”，融合模型的预测结果也将更准确。

## 参考文献

- [1]Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna. [Rethinking the Inception Architecture for Computer Vision](#). arXiv:1512.00567, 2015.
- [2]Fran ois Chollet. [Xception: Deep Learning with Depthwise Separable Convolutions](#). arXiv prepr int arXiv:1610.02357, 2016.
- [3]Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi. [Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning](#). arXiv prepr int arXiv:1602.07261, 2016.
- [4]Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. [Deep Residual Learning for Image Recognition](#). arXiv prepr int arXiv:1512.03385, 2015.
- [5]Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger. [Densely Connected Convolutional Networks](#). arXiv prepr int arXiv:1608.06993v5. 2016.
- [6]黄文坚. [CNN浅析和历年ImageNet冠军模型解析](#). 发表时间: 2017年5月22日.
- [7]Kaggle. [State Farm Distracted Driver Detection](#). 2016.
- [8]Sebastian Ruder. [An overview of gradient descent optimization algorithms](#). 19 Jan 2016.