

# 机器学习纳米学位毕业设计

## 基于深度学习的走神司机识别

Magicyang 2018 年 5 月 8 日

## 1 定义

### 1.1 项目概述

根据 CDC 部门的数据，五分之一的车祸是由一名走神司机引起的。可悲的是，这意味着每年有 425,000 人受伤，3000 人因驾驶走神而死亡。如果能有效的区分驾驶员是否在专心驾驶，并制定相关的一系列规范，可以从一定程度上降低驾驶事故。

这是 KAGGLE 上两年前的竞赛题目。对应网页地址：

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

训练数据一共有 22424 张图片，26 个司机的数据供训练。一共有 10 个分类：

- c0: 安全驾驶
- c1: 右手打字
- c2: 右手打电话
- c3: 左手打字
- c4: 左手打电话
- c5: 调收音机
- c6: 喝饮料
- c7: 拿后面的东西
- c8: 整理头发和化妆
- c9: 和其他乘客说话

希望能尽可能的识别图片中司机的驾驶状态。

## 1.2 问题陈诉

该项目的数据集来自真实的驾驶司机图片。这个项目是一个监督学习图像多分类的问题，需要从驾驶司机的图片中分析出当前司机的驾驶行为。

基于图像分类的主要算法模型有：

- [VGGNet](#) 14.09 <sup>1</sup>
- [ResNet](#) 15.12 <sup>2</sup>
- [Inception v3](#) 15.12 <sup>3</sup>
- [InceptionResNetV2](#) 16.02 <sup>4</sup>
- [Xception](#) 16.10 <sup>5</sup>
- [NASNet](#) 17.07 <sup>6</sup>

从中选择一个算法模型作为基准算法模型进行相关的改造和训练，使得图片根据算法分类得出的结果与司机实际的状态一致。

## 1. 评价指标

KAGGLE 不同的项目评价标准也会有不同。本项目的评判是使用了 Multi class log loss 多分类的对数损失函数。这也是多分类问题通常使用的评价标准。

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

其中 n 是测试集中图像的数目，M 是图像类标签的数目。如果观测对象 i 归属于 j 分类， $y_{ij}$  为 1，否则为 0。 $p_{ij}$  表示预测观测对象到 j 分类的概率。为了避免对数函数的极值，用  $\max(\min(p, 1-10^{-15}), 10^{-15})$  代替。logloss 越小，预测错误率越低，模型也就越好。

KERAS 有对应的损失参数：

**categorical\_crossentropy**：亦称作多类的对数损失，注意使用该目标函数时，需要将标签转化为形如 (nb\_samples, nb\_classes) 的二值序列。

目标是将损失函数尽量的降低。

具体可以参看：<https://www.kaggle.com/c/state-farm-distracted-driver-detection#evaluation>

本项目需要保证 LOGLOSS 在 Private Leaderboard 排名 10%以内，即：LOGLOSS 小于 0.25634。

## 2 分析

### 2.1 数据的探索

KAGGLE 项目提供了对应的数据集。数据集的下载地址：

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

先从训练集开始分析。driver\_imgs\_list.csv 是训练集的基础数据。

subject	classname	img
p002	c0	img_44733.jpg
p002	c0	img_72999.jpg
p002	c0	img_25094.jpg
p002	c0	img_69092.jpg

Subject、classname、img 分别代表：司机编号、司机状态分类、图片名称。

从该表中可以获取到训练集共有 22424 张图片，26 个司机，10 个分类。

下图 1 是训练集各个分类的数目：

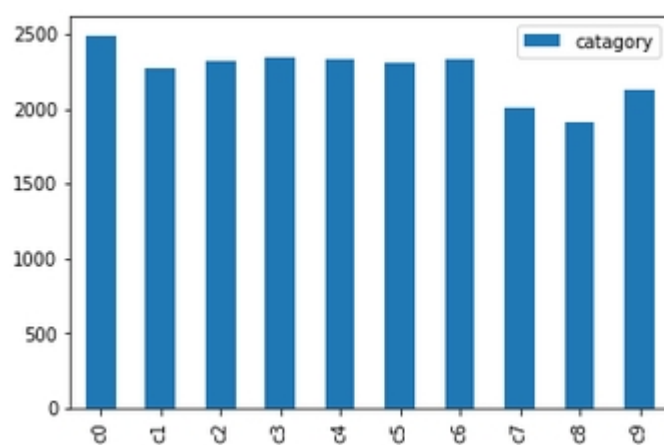


图 1

从图 1 可以看出各个分类的训练图数目基本一致。

图 2 是采样后同一个司机的 10 种状态原始数据展示。



图 2

- c0: 安全驾驶
- c1: 右手打字
- c2: 右手打电话
- c3: 左手打字
- c4: 左手打电话
- c5: 调收音机
- c6: 喝饮料
- c7: 拿后面的东西
- c8: 整理头发和化妆
- c9: 和其他乘客说话

从图二上我们大概可以看出，司机状态不同主要是手部和脸部动作决定的。也希望在最后的验证过程中，可以看到对应的结果。

竞赛提交到 KAGGLE 的测试集共有 79726 张图片。最终需要提交测试集的结果到 KAGGLE 上，并获得验证分数。

## 2.2 算法和技术

自从 Alex 在 2012 年提出的 alexnet 网络结构模型引爆了神经网络的应用热潮，并赢得了 2012 届图像识别大赛的冠军之后，深度学习成为图像识别，尤其是图像分类的主要算法。

最近今年陆续发表了如下的图像分类算法：

- [VGGNet](#) 14.09 <sup>1</sup>
- [ResNet](#) 15.12 <sup>2</sup>
- [Inception v3](#) 15.12 <sup>3</sup>
- [InceptionResNetV2](#) 16.02 <sup>4</sup>
- [Xception](#) 16.10 <sup>5</sup>

- [NASNet](#) 17.07<sup>6</sup>

	Top-1 accuracy	Top-5 accuracy
<b>VGG-16</b>	<b>0.715</b>	<b>0.901</b>
<b>ResNet-152</b>	<b>0.770</b>	<b>0.933</b>
<b>Inception V3</b>	<b>0.782</b>	<b>0.941</b>
<b>Xception</b>	<b>0.790</b>	<b>0.945</b>

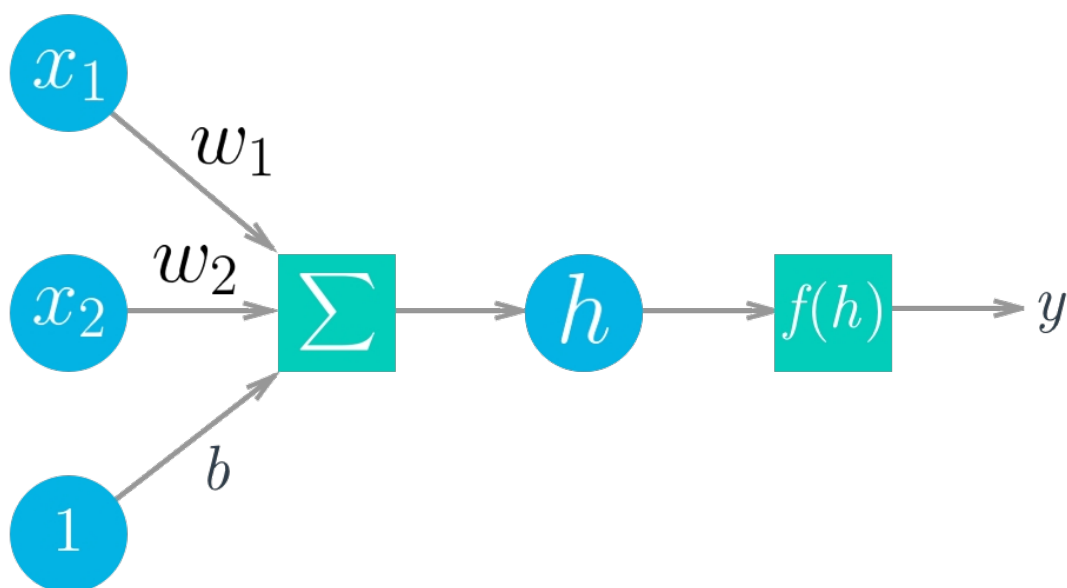
图 3

从上图 3 可以看到目前的模型分类准确率越来越高。

本项目主要使用了 VGG16 和 Xception 的模型做迁移学习。

### 2.2.1 神经网络

从 ALEXNET 开始图像分类的主流算法都是卷积神经网络。神经网络的最基础组成部分是神经元，先借用下图来描述一下神经元的基本结构：



$$h = \sum_i w_i * x_i + b$$

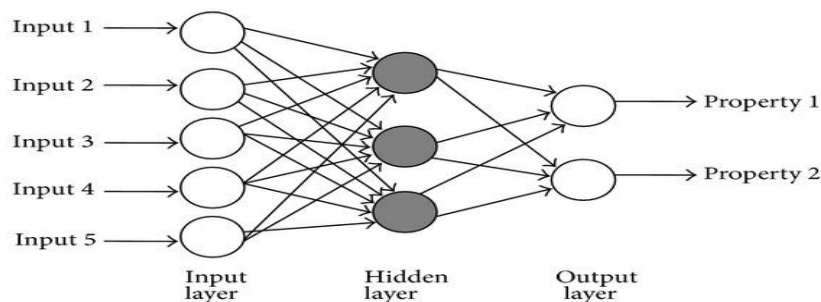
$f(h)$  是  $h$  的激活函数。 $f(h)$  可以是  $f(h)=h$ ， $y$  直接等于  $h$ 。也可以是 ReLU，SigMoid 类似的函数。

神经网络根据神经元的组织结构不同，总体上可以分为两类：前向神经网络和反馈神经网络。

区别在于前向网络主要是后层的网络只接收前层的输入数据。典型的前向网络包含了 **CNN**（卷积神经网络）。而主流的反馈网络包括了 **RNN**（循环神经网络）。

本文主要还是基于 **CNN** 实现的，因此这里主要介绍一下前向神经网络。

提到前向神经网络，必须要提到的就是 **MLP**(多层感知机)。



多层感知机一般包含：输入层、隐层和输出层，从上图可以看到 **MLP** 是一个全连接的神经网络。通过不同神经元的组合，可以实现输入层到输出层的非线性预测。

前向网络的核心训练方式是反向传播算法（**Backpropagation**），通过 **BP** 计算出的梯度变化，结合梯度下降算法，达到训练神经网络的目标。

对于全连接神经网络，容易过拟合。避免过拟合的方案主要有：

- （1）提前终止训练，当验证集的损失函数变大的时候。
- （2）加入正则化，一般是 **L2** 正则。
- （3）加入 **DROPOUT**，**DROPOUT** 的选择推荐是在 **0.2-0.5** 之间。

## 2.2.2 卷积神经网络

从 **ALEXNET** 开始图像分类的主流算法都是卷积神经网络。相对于前面提到的 **MLP** 来说，卷积神经网络有如下的特点：

1. 局部感知，对于图片来说，如 **VGG16** 的输入层有 **224\*224\*3**，如果使用 **MLP**，每层之间，会有数以亿的参数需要调整。在 **CNN** 中，使用局部感知来降低运算量。然后通过多层的网络，再把这些局部特征整合在一起。
2. 参数共享，在局部感知的处理下，需要训练的参数还是太多了。主要思想就是所有的局部感知处理在每一层卷积的时候使用统一的卷积核，比如 **VGG16** 的第一层只需要 **3\*3\*3** 的卷积核，只要 **28** 个参数就可以对图像进行卷积。
3. 多卷积核，前两个特征主要是降低参数，方便训练。但是这样也带来一个问题，输入可以有很多特征，单卷积核的特征提取是不充分的。因此使用多卷积核分别进行卷积处理，从而得到不同的特征。

从卷积神经网络的层次来看，卷积神经网络包括如下的主要层次：

1. 卷积层，主要用来提取特征。
2. 池化层，两个作用：

- (1) 降低过拟合
- (2) 减少输出的规模。卷积层使用 **STRIDE** 也可以做到，不过一般还是使用池化池来实现。可以兼顾降低过拟合。
3. 全连接层，最终将卷积层、池化层、激活函数等运算出的特征整合起来，应用于最后的数据预测。

在实际的模型中，我们还会看到 **Flatten** 层等。

这里提一下归一化（**BatchNormalization**），在近期的网络中，发现使用归一化能非常有效的提升性能。如在 **Xception** 模型中就加入了不少 **BN** 层。为什 **BN** 层性能良好？

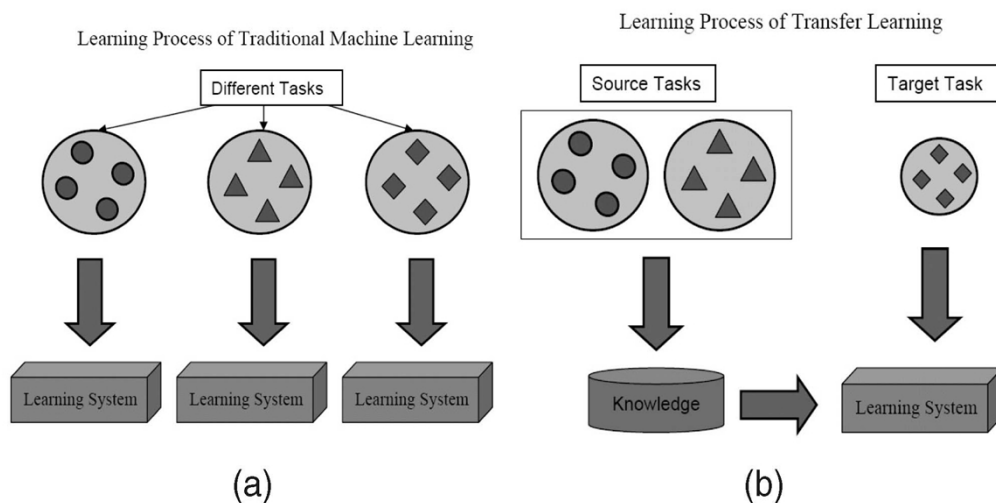
其核心思想是在深度网络的训练中，每一层网络的输入都会因为前一层网络参数的变化导致其分布发生改变，这就要求我们必须使用一个很小的学习率和对参数很好的初始化，但是这么做会让训练过程变得慢而且复杂。通过 **Batch Normalization** 可以很好的解决这个问题。

对于深度学习网络来说归一化的梯度比较容易收敛，是最容易训练的。可以参考如下的知乎文章：

<https://www.zhihu.com/question/38102762>

## 2.2.3 迁移学习

什么是迁移学习，我们先看一张图：



背景：

传统的机器学习方法包括有监督学习、无监督学习和半监督学习，针对不完全数据，有监督和半监督学习还有相应的变形，如处理噪声分类标签问题以及代价敏感学习。然而，这些方法的大多数都假设已标注数据与未标注数据的分布是相同的。与之相反的是，迁移学习允许源空间、任务空间，并且在测试集和训练集中的分布是不同的。

而在 2005 年，the Broad Agency Announcement (BAA) 05-29 of Defense Advanced Research Projects Agency (DARPA)'s Information Processing Technology Office (IPTO)重新定义了迁移学习：迁移学习的目的是从一个或多个源任务中提取知识，并将知识应用于目标任务。<sup>9</sup>



在《A Survey on Transfer Learning》<sup>9</sup>中，使用了域和任务来描述迁移学习。

数据域（domain）由两个部分组成：特征空间  $X$  和边缘概率分布

$P(x)$ ，这里  $x=x_1, \dots, x_n \in X$ 。领域表示成  $D=\{X, P(x)\}$ 。

任务组成：给定一个领域  $D=\{X, P(x)\}$  的情况下，一个任务也包含两个部分：标签空间  $Y$  和一个目标预测函数  $f(\cdot)$ 。一个任务表示为： $T=\{Y, f(\cdot)\}$ 。

论文中给出了迁移学习的定义：给定一个源域  $D_s$  和一个学习任务  $T_s$ ，一个目标域  $D_t$  和一个学习任务  $T_t$ ，迁移学习的目的是使用在  $D_s$  和  $T_s$  上的知识帮助提高在目标域  $D_t$  上的预测函数  $f_t(x)$  的学习，其中  $D_s \neq D_t$  或者  $T_s \neq T_t$ 。

学习方法		$D_s$ & $D_t$	$T_s$ & $T_t$
传统机器学习		相同	相同
迁移学习	归纳式迁移学习/ 无监督迁移学习	相同	不同但是相关
	直推式迁移学习	不同但是相关	不同但是相关
			相同

本例中，IMAGENET 和我们的司机分类都是图像分类，对于域和任务来说都是具有一定的相关性。因此本例中适用归纳式迁移学习。

然后我们知道 IMAGENET 有大量的数据源标签，走神司机项目只有较少的标签数据。有标注的归纳式迁移学习和多任务学习有一定的关联性。两者的相似性在于：两者都有特征共享的思想在其中。不同点在于：迁移学习只是希望通过从源任务迁移知识以在目标任务中获得好的表现，而多任务学习试图将源与目标任务同时学习。

迁移学习的方法有如下四种：

迁移学习方法	简单描述
实例迁移	重新对源域中的一些数据进行重加权，以便在目标域中使用
特征表示迁移	找到好的特征表示以便减少源和目标域中的差别以及分类和回归模型的误差
参数迁移	在源和目标域中发现共享参数和先验知识
关系型知识迁移	在源和目标域中建立相关知识的映射。两个域是相关域，并且在各个域中独立同分布的假设是松弛的

归纳式迁移学习可以同时使用这四种迁移方法。

本项目中，使用了特征表示迁移和参数迁移。同时使用了 IMAGENET 的训练框架和预加载了 IMAGENET 的参数。

关于迁移学习的内容可以参考：

<https://www.zhihu.com/question/41979241>



## 2.2.4 VGG16

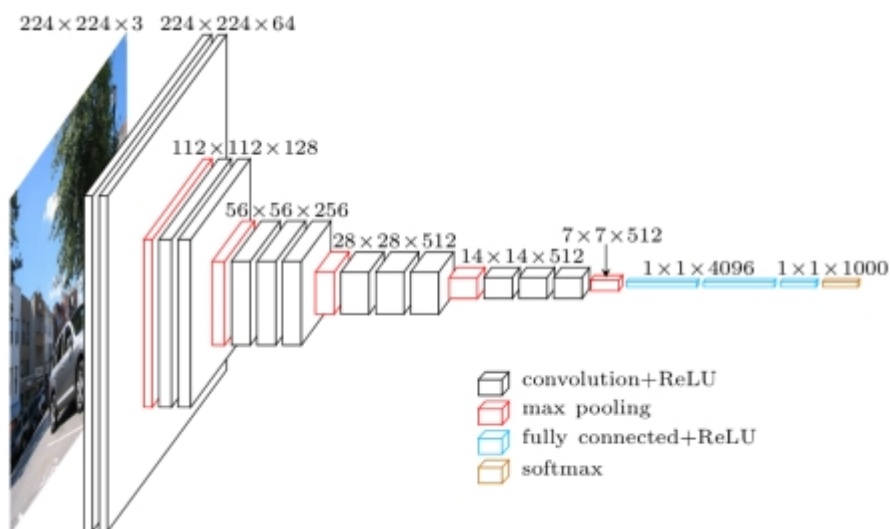


图 4

VGG16 的模型见上图 4。

VGG16 是 2014 年 9 月提出的图像分类模型，从上图 4 可以看到他的模型相对来说比较简单，只有 4 组卷积层，大量的参数运算都集中在全连接层。

下面介绍一下 ReLU 激活函数，目前 CNN 的卷积层都是使用 ReLU 的激活函数，SIGMOID 的激活函数会造成梯度消散。ReLU 的公式如下：

$$\text{Relu}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

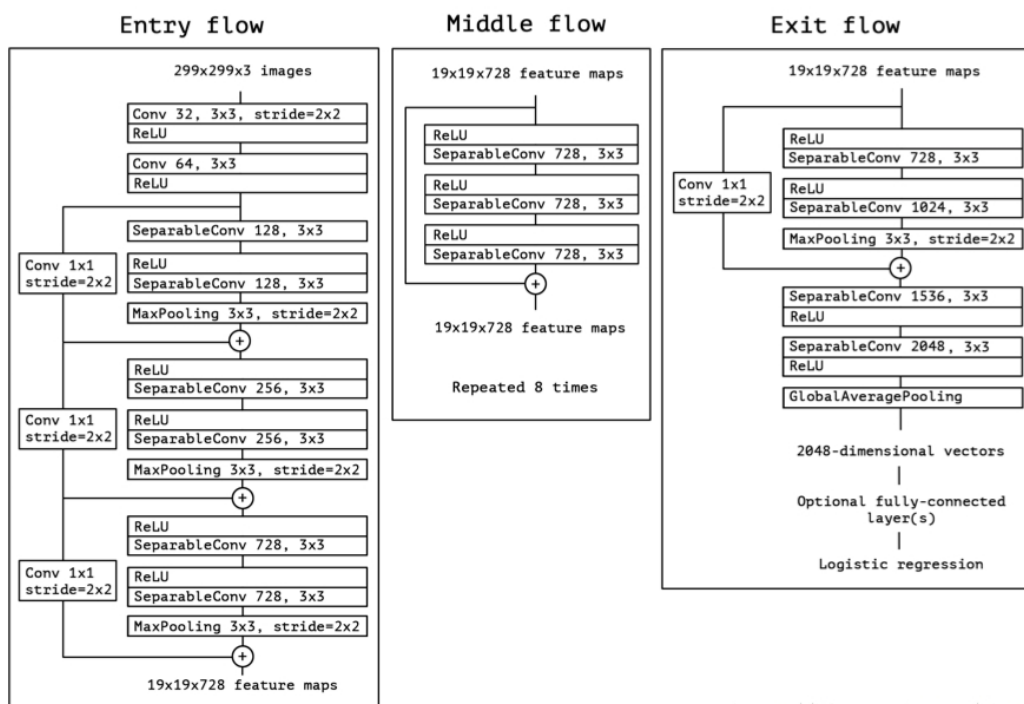
ReLU 有以下优势：对于线性函数而言，ReLU 的表达能力更强，尤其体现在深度网络中；而对于非线性函数而言，ReLU 由于非负区间的梯度为常数，因此不存在梯度消失问题，使得模型的收敛速度维持在一个稳定状态。同时相对于 Sigmoid 来说，没有指数级的运算，只有 MAX 运算，前向运算速度上也会快很多。

ReLU 的另一个优势是在生物学上的合理性，它是单边的，更符合生物学上神经元的特征。

ReLU 也有缺点，大的梯度流经过 ReLU 单元时可能导致神经不会在以后任何数据节点再被激活。当这发生时，经过此单元的梯度将永远为零。所以神经网络训练时需要选择合适的学习率，尽量避免这种情况的出现。

## 2.2.5 XCEPTION

Figure 5. The Xception architecture: the data first goes through the entry flow, then through the middle flow which is repeated eight times, and finally through the exit flow. Note that all Convolution and SeparableConvolution layers are followed by batch normalization [7] (not included in the diagram). All SeparableConvolution layers use a depth multiplier of 1 (no depth expansion).



<http://blog.csdn.net/whz1861>

图 5

XCEPTION 的模型见上图 5。

XCEPTION 的模型相对于前面的 VGG16 来说就复杂了很多，它的核心思想是将 Inception modul 拆分成一系列操作，独立处理 spatial-correlations 和 cross-channel correlations，网络处理起来更加简单有效。

从上图 5 的模型中可以看到在卷积块之间是使用了 1\*1 卷积相加的操作，类似于 RESNET 的操作，主要用于空间数据整合。而在卷积块中间而是使用了 SeparableConv 进行 CHANNEL 卷积。

关于 SeparableConv 的结构可以看下面两张图：

Figure 3. A strictly equivalent reformulation of the simplified Inception module.

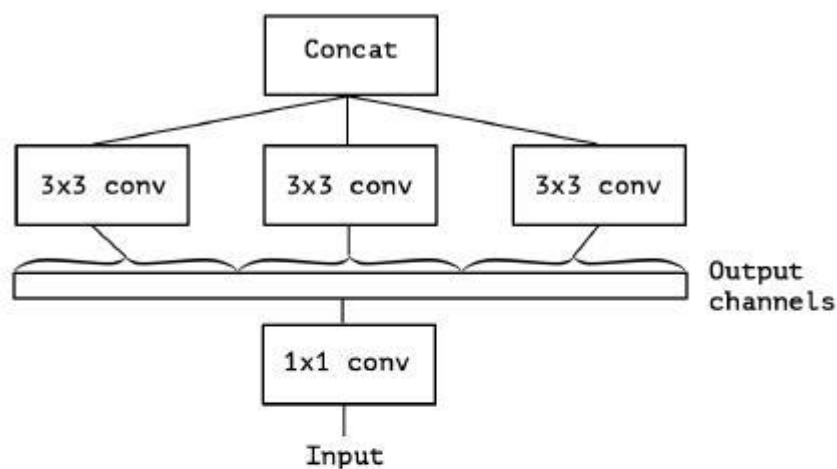


Figure 4. An “extreme” version of our Inception module, with one spatial convolution per output channel of the 1x1 convolution.

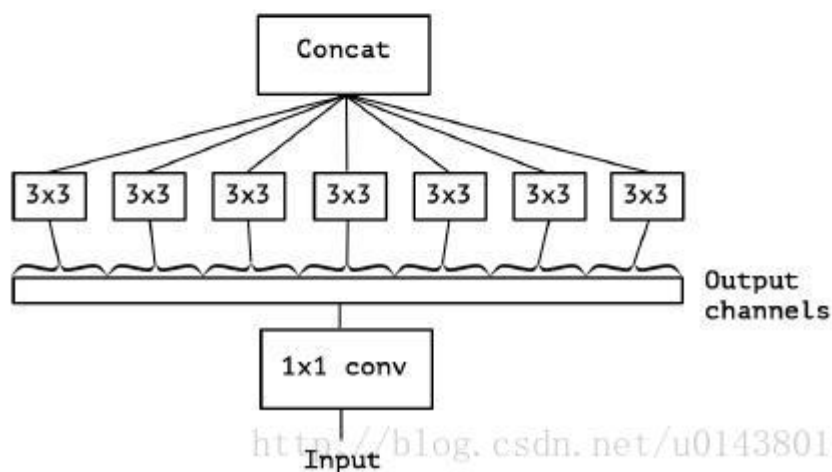
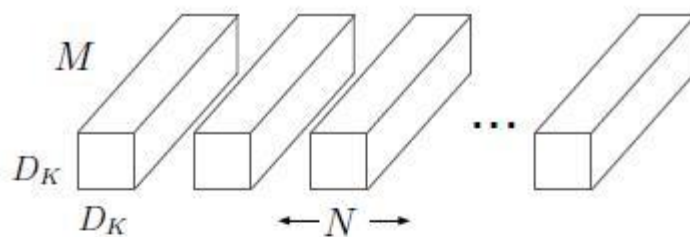
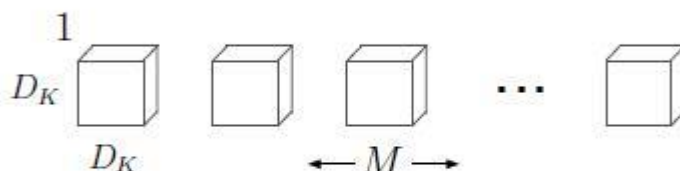


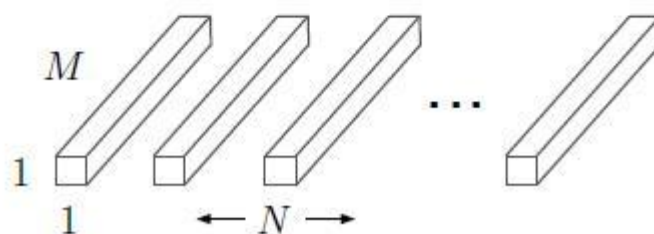
Figure4 是 INCEPTION 的终极模型，先进行 1\*1 卷积，接着在 CHANNEL 的维度进行分离，接着对分离结果分别做 3\*3 卷积，最后整合成和原 CHANNEL 维度相同的结果。



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters

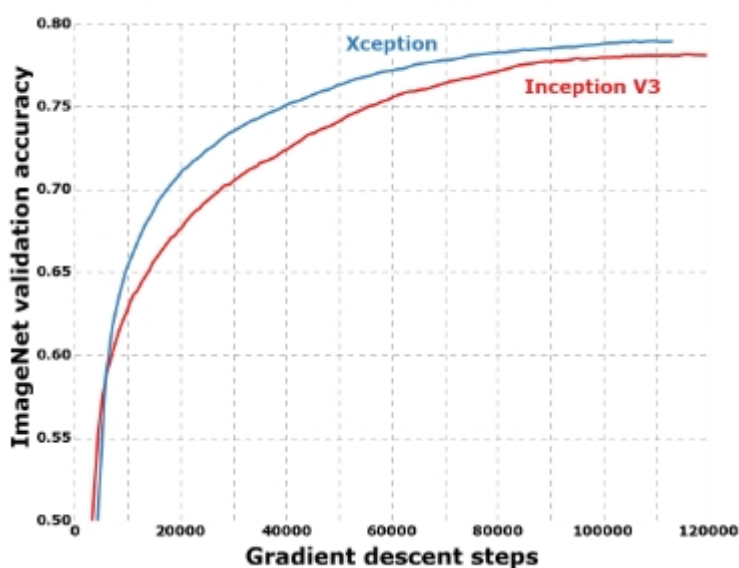


(c)  $1 \times 1$  Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

其实 depthwise separable convolution 和上面的 Figure4 是很像的。差别有两个：1、顺序不一样，在 depthwise separable convolution 中是先进行一个 channel-wise 的 spatial convolution，也就是上图的 (b)，然后是  $1 \times 1$  的卷积。而在 Figure4 中是先进行  $1 \times 1$  的卷积，再进行 channel-wise 的 spatial convolution，最后 concat。

从整体模型来看，XCEPTION 一方面同时结合了 INCEPTION 和 RESNET 的优点，另一方面模型同时从 CHANNEL 和空间两个方向去做卷积去提取特征，在论文<sup>5</sup>中也可以看到其测试结果优于其他 CNN 卷积网络。

Figure 6. Training profile on ImageNet



从图 3 可以看到 Xception 在主流算法里 IMAGENET 验证集里的准确率是最高的，同时对比上图可以发现对比 INCEPTIONV3，XCEPTION 有更快的收敛速度。

项目中使用 KERAS 工具构建模型，KERAS 已经集成了 VGG16 和 XCEPTION 的模型，大量降低了我们构造模型错误的可能性。

## 2.3 基准模型

本项目需要保证 LOGLOSS 在 Private Leaderboard 排名 10%以内，即：LOGLOSS 小于 0.25634。

在训练过程中，希望验证集的 LOGLOSS 尽可能的低。

VGG 的模型相对简单，可以用来先做相应的测试。而 XCEPTION 模型相对性能更好，在需要时候替换模型。

# 3 方法

## 3.1 数据预处理

### 3.1.1 数据分割

该项目中一共有 20000 多张图片，需要正确的分割训练集和验证集。从输入文件可以看到这 20000 张图片分属于 26 个司机。从数据探索可以看到相同司机的图片差异很小，相同状态下不同司机的图片差异更大。所以

不能直接对整个训练集进行数据划分，需要对司机进行训练集和验证集的划分，即训练某些司机的图片，在其他司机上进行验证。

### 3.1.2 特征提取

对于不同的基准模型，输入的像素是不一样的。如 VGG16 是  $224*224*3$ ，而 XCEPTION 是  $299*299*3$ 。项目统一使用 OPENCV 读取图片特征。

另外需要注意的是我们都是使用的 IMAGENET 的训练集作为迁移学习的基础，对于不同的模型，特征预处理是不一样的。KERAS 已经封装好了预处理方法。可以详见：

[https://github.com/yangchengtest/keras/blob/master/keras/applications/imagenet\\_utils.py](https://github.com/yangchengtest/keras/blob/master/keras/applications/imagenet_utils.py)

如果需要一次处理特征，需要关注一下内存的使用情况。

## 3.2 执行过程

### 3.2.1 读取特征的内存分析

前面提到需要考虑特征读取的内存。例如使用 XCEPTION 模型，使用 UINT8 来记录特征所需要的内存是： $224*224*299*299*3$  约 6G 内存。在实际处理过程中，我把预处理统一提前在 CPU 一次处理，不用每个 EPOCH 处理一次，使用了 FLOAT16 来记录特征这就是需要 12G 内存。如果开始阶段就把所有训练数据的特征提取出来，再分割成训练集特征和验证集特征，就需要 2 倍的内存开销，对于 64G 以下的用户来说，可能出现 OOM。因此我个人是在切分完验证集和训练集之后再获取图片特征。还有一个办法就是使用 UINT8 记录特征，在 KERAS 训练模型里做预处理，也可以节约内存，各有利弊。

### 3.2.2 模型的选择和是否冻结的问题

前面提到图片比较相似，需要全部训练模型，在实际的操作中，个人还是尝试了先通过卷积层先训练出图片卷积层的特征，然后只训练全连接层。实际尝试后发现，验证集的 logloss 无法降低。

接着开始使用 VGG16 的模型来进行训练，在实际训练中，开始忘记在模型中加入 IMAGENET 的权重，导致训练集的 logloss 一直无法降低。我个人是这么理解的：如果不加初始权重实际上就不是迁移学习了，是整个识别算法全部重训练，这个需要大量的训练数据集和大量的训练次数来迭代完成，而我们的训练集只有 20000 多张图片，很多分类特征根本训练不出来。如果看到 logloss 一直很高，看一下是否忘记加模型权重了。

最后还是选择了 XCEPTION 作为基准模型，这部分在后面的优化里去描述。

### 3.2.3 优化器的参数选择

CNN 的反向传播本质上都是梯度下降，随着深度学习的发展，优化器也在不断的发展。

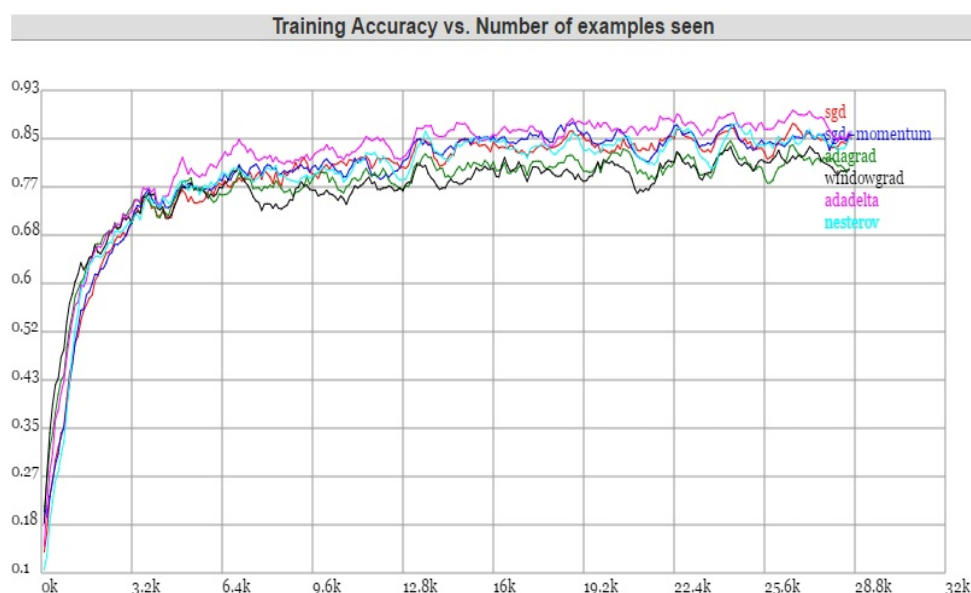


图 6

从图 6 可以看到各个算法的准确率标识。

各个优化器的优劣可以参考知乎的文章：

<https://zhuanlan.zhihu.com/p/22252270>

目前主流的优化器选择是 ADAM<sup>7</sup> 和 SGD-Nesterov。

提到优化器，就不得不提学习率。当学习率过低的时候，可能陷入局部极值，而当学习率过大的时候，又可能导致反向传播过快，出现梯度爆炸。因此在选择优化器后，选择合适的学习率比较重要。例如在本项目中，使用了 ADAM 优化器，ADAM 的学习率推荐是  $10^{-3}$  到  $10^{-5}$ 。选择  $10^{-3}$  时就会出现梯度爆炸的情况。

## 3.3 完善


### 3.3.1 使用 KFOLD


这里需要提到的是交叉验证法。它的基本思想就是将原始数据（dataset）进行分组，一部分做为训练集来训练模型，另一部分做为测试集来评价模型。交叉验证用于评估模型的预测性能，尤其是训练好的模型



在新数据上的表现，可以在一定程度上减小过拟合。还可以从有限的数据中获取尽可能多的有效信息。

项目的模型并不大，这里使用的 KFOLD(K 折交叉验证)，通过 K 折后的预测平均可以有效的提高准确率，降低单训练的过拟合。

<b>submission_10.csv</b> a minute ago by <a href="#">magicyang</a> xception-10	0.36330	0.31445	<input type="checkbox"/>
<b>submission_9.csv</b> 33 minutes ago by <a href="#">magicyang</a> xception-9	0.41243	0.48640	<input type="checkbox"/>
<b>submission_8.csv</b> 34 minutes ago by <a href="#">magicyang</a> xception-8 	0.33843	0.30654	<input type="checkbox"/>
<b>submission_7.csv</b> an hour ago by <a href="#">magicyang</a> xception-7	0.43228	0.39570	<input type="checkbox"/>
<b>submission_6.csv</b> an hour ago by <a href="#">magicyang</a> xception-6	0.34553	0.39912	<input type="checkbox"/>
<b>submission_5.csv</b> an hour ago by <a href="#">magicyang</a> xception-5	0.32300	0.35430	<input type="checkbox"/>

<b>submission_4.csv</b> an hour ago by <a href="#">magicyang</a> xception-4	0.38370	0.35981	<input type="checkbox"/>
<b>submission_3.csv</b> 3 hours ago by <a href="#">magicyang</a> xception-3 	0.39483	0.33210	<input type="checkbox"/>
<b>submission_2.csv</b> 3 hours ago by <a href="#">magicyang</a> xception-2	0.35226	0.37786	<input type="checkbox"/>
<b>submission_1.csv</b> 3 hours ago by <a href="#">magicyang</a> xception-1	0.37598	0.37278	<input type="checkbox"/>
<b>submission.csv</b> 6 days ago by <a href="#">magicyang</a> XCEPTION KFOLD	0.22823	0.23925	<input type="checkbox"/>

上图记录了 10 个 XCEPTION 提交 KAGGLE 的分数和最终使用 KFOLD 模型的分。单模型的 PRIVATE SCORE 都在 0.3-0.5 之间，当使用 KFOLD 后 PRIVATE SCORE 下降到了 0.2223。使用交叉验证法可以显著提升模型效果。

### 3.3.2 模型的改进

Submission and Description	Private Score	Public Score	Use for Final Score
<a href="#">submission.csv</a> 3 days ago by <a href="#">magicyang</a> XCEPTION KFOLD	0.22823	0.23925	<input type="checkbox"/>
<a href="#">submission.csv</a> 5 days ago by <a href="#">magicyang</a> <a href="#">add submission details</a>	0.29202	0.30417	<input type="checkbox"/>

图 7

先上图，图 7 是 kaggle 提交的最终结果，在使用 VGG16 的模型情况下，使用 KFOLD 运算，logloss 并没有进入 10%，而使用 XCEPTION 则相对来说比较容易的进入了 10%。个人分析原因是相对来说，XCEPTION 相对于 VGG16 来说，VGG16 的参数主要集中在全连接层，卷积层的参数并不多，越多的卷积层参数表示底层可描述的细节越多。这就导致在细节特征提取上，XCEPTION 相对于 VGG16 有天然的优势。

下面就开始选择优化器，ADAM 有更快的收敛速度，本项目最终还是选择 ADAM 作为优化器。主要还是调整学习率，合适的学习率可以看到 VAL\_LOSS 先是慢慢下降，后续过拟合再升高的变化。

最后为了降低训练时间和防止过拟合，使用 EASY STOPPING 来结束模型训练。

## 4 结果

### 4.1 模型的评价

最终模型简述为如下的步骤：

- (1) 将 XCEPTION 作为基础模型，在卷积层后加入一层 AveragePooling 和一层激活函数为 SOTFMAX 10 的全连接层。使用 IMAGENET 数据集的卷积层权重作为新模型的卷积层权重。
- (2) 对数据进行预训练，使用 KFOLD 方法，折线数 10，将司机 ID 分解。从 KFOLD 分解出的训练集司机 ID 和验证集司机 ID 中分别读取图片特征。由于训练数据读取是有顺序的，在提交训练之前需要打乱。
- (3) 选择学习率为  $10^{-4}$  的 ADAM 优化器作为优化器进行训练。主要参数还有 batch\_size=32,epoch=10,early\_stop 的 patience=2。

下面说一下训练参数的选择。

学习率前面已经提过，过高过低都会有问题，目前的学习率是试验出来的。

**BATCH\_SIZE** 由于我们都是使用 GPU 进行训练，2 进制的 **BATCH\_SIZE** 更方便机器学习，在 GPU 内存允许的条件下可以提升一下 **BATCH\_SIZE** 数目，**BATCH\_SIZE** 过低，**VAL\_LOSS** 也会无法收敛。

**early\_stop** 的选择则相对无奈，这个跟 GPU 的性能是强相关的，我自己的 GTX1070 的 8G 显卡，**BATCH\_SIZE** 只能支持 16，一个代次需要 18 分钟。后续使用 AWS 的 P3.2XLARGE，一个代次只需要 4 分钟。由于使用了 **KFOLD** 需要训练 10 个模型，时间规模上是巨大的。因此使用了较小的 **early\_stop**。其实这跟优化算法也有关系，如果使用 SGD-Nesterov，训练代次会增加，收敛速度会变慢一点，也可能会得到更加低的 **VAL\_LOSS**。时间和金钱成本需要考虑。

在模型训练中使用了司机 ID 分离训练集和验证集，以及使用 **KFOLD** 降低单模型的过拟合。使用 **XCEPTION** 替代 **VGG16** 增加卷积层的描述性，我认为项目的最终模型是优于初始模型的。

## 4.2 模型的验证

本项目是 KAGGLE 的竞赛项目，最终是需要提交到 KAGGLE 上进行验证的。测试集中一共有 79726 张图片。基于内存的考虑，分解成了 2 个测试集进行模型预测。

前面提到训练过程中使用了 **KFOLD**，在验证的时候，需要对同一个输入图片，需要分别使用 10 个训练后的模型对输入进行预测，然后对 10 个预测进行平均，最后得到最终的训练结果。

使用 **DataFrame** 记录训练后的结果，最终提交到 KAGGLE 上，获得分数，评价是否满足项目的要求。

Submission and Description	Private Score	Public Score	Use for Final Score
<a href="#">submission.csv</a> 3 days ago by <a href="#">magicyang</a> XCEPTION KFOLD	0.22823	0.23925	<input type="checkbox"/>
<a href="#">submission.csv</a> 5 days ago by <a href="#">magicyang</a> <a href="#">add submission details</a>	0.29202	0.30417	<input type="checkbox"/>

最终 Private Score 为 0.22823 低于目标 0.25634，项目达成。

## 4.3 CAM 可视化

最后来看一下 CAM<sup>8</sup> 可视化，可以参考附录的论文。对于实现来说就是用卷积层的结果点乘全连接层分类，对于 **XCEPTION** 来说， $10 \times 10 \times 2048$  点乘  $2048 \times 1$  会得到一个  $10 \times 10$  的图，放大到输入特征图大小( $299 \times 299$ )，用热度图的方式展现出来。

可以用下图 8 对比图 2，在 CAM 图例，司机的特征区域都比较明显的。模型训练结果满足需求。

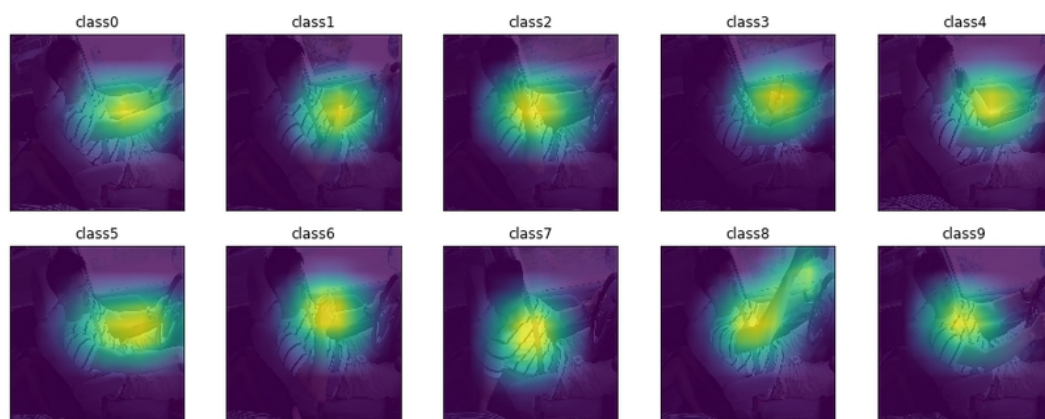


图 8

## 5 项目结论

### 5.1 对项目的思考

总体来看这是一个非常好的项目，项目中需要掌握以下的知识：

- (1) 数据预处理，司机项目需要考虑内存，考虑特征规模。
- (2) 图像处理，OPENCV 的基础函数。
- (3) 评价标准，司机项目使用 **logloss** 评价体系，对该评价体系有了深入了解。
- (4) 模型使用，对于简单模型，可能无法达到标准，这就需要我们更多的了解新模型，理解新模型，使用新模型
- (5) 如果不使用 **KFOLD** 结果也不会优秀，教会我们使用交叉验证法。
- (6) 走神司机的项目模型不算小，从该项目中大概了解了模型的计算规模，学会了使用 **AWS** 的 **P3.2XLARGE** 进行编程。以后遇到相应的问题，可以更快的评价计算量和对项目进行整体规划。
- (7) 学会了 **CAM** 可视化，看懂论文也花了很长的时间，对可视化有了一定的探索。
- (8) 调参很重要，大概试错了一些，还需继续加强。

从开始做毕业项目到完成这个毕业项目，总共花了 **12** 个工作日的时间，如果需要做到更好需要更多的时间，很遗憾给自己定了一些小目标，后面接着要去做新的项目，很感谢这个项目给我个人的锻炼。

### 5.2 后续改进

这里还有一些做的不够完美的内容，个人觉得项目可以在以下方面继续加强：

- (1) 调参不是特别充分，**ADAM** 收敛太快，换 **SGD** 可能结果更好。
- (2) 可以尝试融合模型，相对单模型，融合模型结果应该会更好。

(3) 没有针对训练集做数据增强，做数据增强的化结果应该会更好。

## 参考文献：

[1] Karen Simonyan, Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition.(arXiv:1409.1556,2014)

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun.Deep Residual Learning for Image Recognition.(arxiv:1512.03385,2015)

[3] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna .Rethinking the Inception Architecture for Computer Vision. (arXiv:1512:00567,2015)

[4] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. (arXiv:1602.07261,2016)

[5] François Chollet .Xception: Deep Learning with Depthwise Separable Convolutions. (arXiv:1610.02357,2016)

[6] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, Quoc V. Le.Learning Transferable Architectures for Scalable Image Recognition. (arXiv:1707.07012,2017)

[7] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. CVPR'16 (arXiv:1512.04150, 2015).

[8] Diederik P. Kingma, Jimmy Ba.Adam: A Method for Stochastic Optimization (arXiv:1412.6980,2014).

[9] Sinno Jialin Pan and Qiang Yang, Fellow, IEEE:A Survey on Transfer Learning(IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 22, NO. 10, OCTOBER 2010)