Distracted driver detection

1.定义

一、项目概述

该项目是 Kaggle 上 State Farm 公司举办的一个竞赛项目，其目的是得到一个可检测司机驾驶行为的模型

司机在驾驶过程中，由于长时间连续行车，很容易产生视觉、心理上的疲劳、放松，进而也就容易出现走神的情况，客观上表现为东张西望，打电话，与他人随意交流的情况。根据《中华人民共和国道路交通安全法实施条例》，打电话等行为会分散驾驶员注意力，为行车驾驶带来不必要的安全隐患。或许对于行车多年的老驾驶员而言，驾轻就熟的开车技术能够让他并不需要太多注意力就能处理好各种开车情况。但对于其他驾驶员而言，这是对他们自身乃至对乘客的安全不负责的态度。既然如此，一个能够实时监测驾驶员动作，辅助驾驶员实时注意力集中开车，对于保障行车安全有很大的帮助。

State Farm 在很多汽车仪表盘上安防摄像头，jiequ1 司机驾驶过程中的 2D 图片，希望 Kaggle 参赛者能基于这些驾驶图片，训练一个可检测驾驶行为的模型。

本项目使用近几年在 LSVRC 大赛上取得突破性进展的深度卷积神经网络模型，在这些预训练模型的基础上，利用迁移学习对本项目的数据集进行训练，最终获得不错效果。

二、问题描述

该项目本质上是一个监督分类学习的问题。训练数据集已经做好了分类标注。模型的目标是学习不同驾驶行为的特征，以致能正确识别一个未见过驾驶图片。

本项目所用数据集来自 State Farm 收集的驾驶图片数据库。可从 Kaggle 的 Districted_Driver_detection 项目中下载数据集，已做好初步分类，得到按 c0~c9 分类好的 train set（c0~c9 分别对应不同的驾驶行为，即作为同文件夹的图片标签）以及未分类的 test set。

本项目中，为了减少过拟合的可能性，首先按驾驶员 ID 对训练集按照一定比例分割为训练集和验证集，根据标签分类，使用 keras 的生成器数据增强的同时，shuffle 序列重排。数据准备好之后，构造一系列 CNN 模型，利用交叉验证对数据进行训练，记录每次迭代的 Loss 和 Accuracy。训练数据处理完后，模型被用于预测测试数据，随后将预测结果保存成 csv 件，csv 每一行记录一个被测图片的 10 种行为分类的概率。最后将该 csv 文件上传到 Kaggle，根据该项目的判断指标来对预测结果进行评分。

2.分析

三、数据或输入

数据集来自 State Farm 提供的训练和测试数据集，所有数据集均是从车载摄像头录像中截取下来的静态图片。其中训练集已经做了基于 label-class 的分类，分类为[c0,...,c9]，其含义依次是：-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。

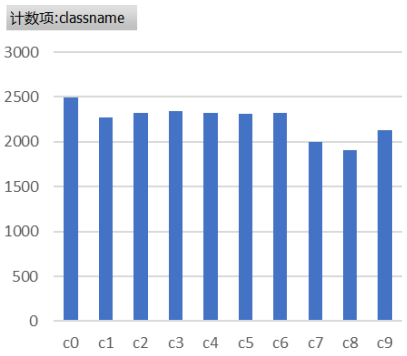


图 2：训练数据集的分类分布

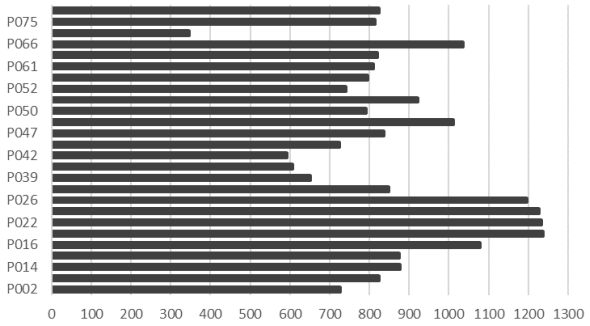


图 3：不同司机图片数量

训练集中相同分类的图片在同一个文件夹中，各个分类的图片数量分布见图 2。测试集没有做分类，一共包含 79721 张 2D 图片。数据集的图片尺寸均是 640x480。除了数据集，该项目还提供了司机 id 列表，该列表包含了数据集中图片对应的司机 id 及对应的分类，从图 3 可以看出不同司机的图片分类的分布。

下载连接：https://www.kaggle.com/c/state-farm-distracted-driver-detection/data

此外，分析图片数据，不难发现项目难点所在。下面随机抽样图片展示如下。

（a） （b） （c）

（d） （e） （f）

图 4　随机图片展示

可以发现，图片都是从同一个角度拍摄，且图片环境相似度极高，有一定可能性导致过拟合；不同司机图片数量不同，驾驶员特征各异，很容易成为模型学习的主要特征，这是我们所不希望看到的；图片明暗程度较差，可能导致部分特征缺失；此外个别图片还存在干扰的人物因素。

从数据集的分布我们可知对于每项分类的学习数据基本充足均等，以上提出的问题都可能对我们的训练造成一定影响，必须针对的采取措施解决。

对于机器学习算法来说，其本身有很多 hypter-parameters 需要验证和调整，其目的是降低对现有数据的过度依赖，未来能泛化（generalize）到未见过的数据。

本项目中我们希望模型学到的是检测图片当中的司机行为，而不是司机的衣着或是汽车内装饰灯。并且从 driver_id 列表可知，训练集一共有 26 个不同的司机，但我们无法知道测试集有多少个司机（这也与现实世界的假设一致，因为算法不可能一直应用于模型训练过的司机）。为了降低司机或汽车的属性影响到训练效果，我们有必要排除模型对相同司机的依赖。即模型训练和验证使用完全不同的司机/汽车 id 图片。

具体做法是，首先根据司机的 id 来将训练图片进行分类，然后对司机 id index 进行 K_Fold 处理，这样得到的 train set 和 validation set 分别是不同的司机 id 对应的图片，例如 trainset 的司机 id 有[p015,p022,p051]，validationset 的司机 id 有[p002,p081]。对这些不同组合的训练集进行训练，得到不同的模型参数。测试阶段将这些不同的模型分别对测试集进行预测，得到多个测试结果，最后将其进行合并，这样往往能得到更好的综合结果。

四、解决方法描述

该项目的最终目的是根据训练集，对测试集的图片进行分类，也就是说是一个基于监督学习的分类识别问题。虽然监督学习分类算法有很多，但图片分类的许多复杂的特性使得许多传统的监督分类学习算法的并不适用，例如状态空间巨大、图像含义在不同位置的平移不变性（translation invariance）等。

目前应用于图片分类、检测及分割最广泛的算法即是卷积神经网络，以下用 CNN 来代指卷积神经网络。与普通的前馈神经网络不同，CNN 受到动物的视觉皮层实验的启发，不同层之间的神经元并不是全连接，后一层神经元的刺激（输入）只与前一层部分神经元的输出有关，这一部分起作用的神经元也被称作感受野（receptive field）。感受野作用于神经元的计算方式类似于数学上的卷积操作，这也就是卷积神经网络名字的来源。

CNN 突出的特点有，层与层之间局部连接导致网络 parameters 大大减少，作用于感受野的过滤器在整个图片上的 parameters 共享导致 CNN 有很好的物体平移不变性，网络结构简单清晰有利于对网络内部结构进行可视化分析等等[1]。

近几年来涌现很多非常有效的卷积神经网络模型，自从 LSVRC-12 大赛上 Krizhevsky 提出 AlexNet[2]深度卷积神经网络模型，每年的 LSVRC 大赛都被卷积

神经网络统治，例如 LSVRC-14 提出的 VGG16 模型[3]和 GoogeLenet 模型[4]、LSVRC-15 的冠军 ResNet 模型[5]等。

由于这些模型都已经在 LSVRC 数据集上进行了充分训练，并得到了很好的分类和检测的效果，很多研究者发现利用这些训练好的模型（pretrained_model），在其他图片分类的任务上也能得到很好的效果，这也就是「迁移学习」的利用。[6][7]

该项目的实现使用 Numpy 来做数据的提取和转换；使用 Scikit-learn 来做计算结果指标以及交叉验证；使用 Tensorflow 来做深度学习的模型构建和训练。该项目使用的计算资源是 AWSEC2p2.xlarge 实例。

五、评估标准

该项目是多分类的问题，该问题的评价指标主要有两个，预测结果准确率以及模型的训练和预测时长。

Log Loss: Kaggle 对预测结果的评分使用 multi-class logarithmic loss，也称 cross-entropy，计算公式为：

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

Ps:N 为测试条目数；M 为标签数；y_{ij} 为某条目的某标签编号；p_{ij} 某条目是某标签的概率。

Time:一个预测算法能应用到实际产品中，训练和预测时长也很关键。走神检测对实时性要求高，否则没有意义，目前计划准确率在 90%以上，预测时间越短更好。

六、基准模型

虽然该项目在 Kaggle 上已经结束了，但我们仍可以查看该项目上所有的参赛者的 Leaderboard。该项目的 Leaderboard 一共有 1440 组参赛者，Public Leaderboard 的评价标准使用 31%的测试集来计算其 logloss，而 Private Leaderboard 的评价标准使用剩下的 69%测试集来计算最终的 logloss。指数越小的团队排名越高，最终的排名以 Private Leaderboard 为准。本项目的目标是测试集的分类结果能排在 Private Leaderboard Top10%，争取能进入前 100 名，根据查看，score<=0.25634。

3.方法

七、数据预处理

深度学习无法直接对图片数据进行处理，我们必须对图片进行预处理。本项目主要基于 Keras 架构进行代码编写。

根据前文，我们确认下载的数据集还需要进行进一步归类整理。这边基于源

数据集，司机 ID 共 26 条，按照比例是 4：22，将 train_set 分成训练集和验证集，各自建立文件夹，并用 c0~c9 作为子文件夹命名，测试集不变动，完成数据集的分类整理。

其次，运用 ImageDatagenerator()作为数据提取器，提取图片的同时也提取了该图片对应的分类，保证图片和分类总是一一对应。

图片还需要进行数据转换和处理，为减少数据量，现将图片进行缩放，为使用预训练模型，我们将图片尺寸缩放成预训练模型能接受的尺寸。这需要根据所选择的预训练模型进行辨别，使用 VGG、ResNet 模型，需要将图片缩放为 224x224；使用 InceptionV3、InceptionResnetV2、Xception，需要将图片缩放为 299x299，然后将图片转换为 Numpy array，最后对图像数据进行预处理。

根据预训练模型的不同，其数据预处理方法也不同。VGG、Resnet 进行零均值化；InceptionV3、InceptionResnetV2、Xception 数据预处理，把数据值都约束在[0,1)之间。

第三，将图片的分类进行数据转换，由于我们训练使用的指标是 cross entropy，所以我们将图片分类的数值转换成二进制的分类矩阵。最后，我们需要将提取出的数据进行乱序排列，以免受到原始数据顺序的干扰。

八、实现

该项目的实现尝试多种方法，包括自己构建简单卷积模型，InceptionV3、InceptionResnetV2、Xception 预训练模型，其他预训练模型也尝试过，最终选择这三个表现最好的模型，作为项目使用模型。在训练过程中最关键的就是调参，确保自己的模型能够取到最好结果，主要通过试错法，选取合适的参数范围，多次试验，观察其验证集训练效果，确认最合适的 dropout、learning rate。当然，其中调参遵循的准则如下：

1.如果 val_loss 还在下降，那么应该多训练几代，或者增加模型复杂度；

2.如果 val_loss 开始上升，则可能是过拟合问题，需要正则化（dropout 等）；

3.如果 val_loss 振荡，那就是学习率太大，需要减小学习率；

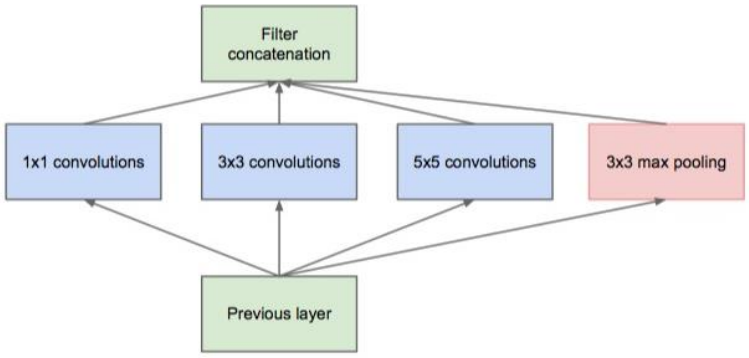
4.如果 val_loss 已经完全稳定，那么应该减小训练代数。

其主要试验思路，即通过训练三个模型，并提出最佳模型权重 hdf5 文件，再利用 sgd 优化器进行精调，从而得到单模型的 test 结果。再结合三个预测结果进行均值化，取得最好的预测结果。

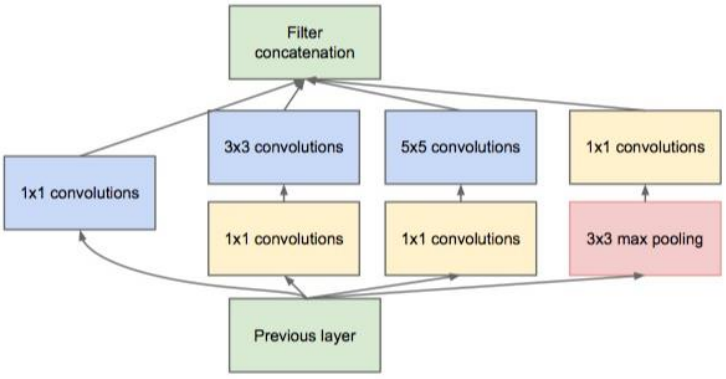
使用 Pretrained InceptionV3

模型的深度对于特征提取非常重要，自己构建的模型深度和学习效果都无法与预训练模型相比，根据本项目的要求，直接使用合适的预训练模型，虽然模型层数增加直接使所需训练参数增加，且容易出现 overfitting，提高了训练难度，但模型训练效果得到提升。另外可以了解的是，卷积神经网络学习过程中，通过底层的权重提取通用的表层特性，如物体形状，线条、曲率、大小等特征，再通

过过滤器，不断提取得到高级特征，最终确认分类依据的深层抽象特征。我们利用一个已经过大数据集充分训练的模型，保留该模型的底层 weights，来微调该模型的高层 weights。在微调高层 weights 后，我们也可以微调一些低层 weights 来让模型收敛到更佳的位置。



(a) Inception module, naïve version



(b) Inception module with dimensionality reduction

图 5：Inception Module

GoogLeNet 模型是 ILSVRC2014 的冠军，不同于 VGG 模型，GoogLeNet 打破了以往串行叠加卷积层的策略。对于一个卷积层，使用不同大小的 filter 能得到不同特性，filer 的选择很多是基于经验和实验结果来判断的。GoogLeNet 创新性地并行使用了多个不同的 filter，综合所有 filter 结果得到输出，该模块也被称为 Inception Module。

如图所示，module 中并行使用了不同的 filters，每种 filter 都可能提取数据的不同特征信息，例如 1x1 conv 可能提取数据的细微特征，5x5 conv 有更大的感受野，可能提取其他的特征。另外，每组 filters 之前都有一层 NiN 1x1 convs，其目的是减少输入给尺寸较大的 convs 的维度，有效减少模型训练的计算量。

在这个方案中，我们使用 Keras 自带的 InceptionV3 模型。模型权重使用预训练权重'imagenet'，利用 GlobalAveragePooling2D()进行均值化，并用 softmax-10 替代原模型的分类器 softmax-1000。

该模型的训练阶段，直接开放所有层进行训练，并提取较优的权重模型进行微调优化，得到收敛最佳的权重模型。

在编译阶段，优化器使用 Adam，loss_function 使用 categorical_cross_entropy，

metrics 使用 accuracy，设定 lr=0.0003，dropout=0.3，迭代次数 15，batchsize：32。

微调阶段，optimizer 为 sgd，lr=0.0001，decay=1e-6，momentum=0.9，nesterov=True，dropout=0.4。

使用 Pretrained InceptionResNetV2

除了 InceptionV3 模型，本项目还使用了 InceptionResNetV2 模型对图片进行了训练。

为了进一步推进这个领域的进步，2016 年 Google 发布 Inception-ResNet-v2，它在 ILSVRC 图像分类基准测试中实现了当时最好的成绩。Inception-ResNet-v2 是早期 Inception V3 模型变化而来，从残差网络 ResNet 进行借鉴并发展得到的。

模型	大小	Top1 准确率	Top5 准确率	参数数目	深度
Xception	88MB	0.79	0.945	22,910,480	126
VGG16	528MB	0.715	0.901	138,357,544	23
VGG19	549MB	0.727	0.91	143,667,240	26
ResNet50	99MB	0.759	0.929	25,636,712	168
InceptionV3	92MB	0.788	0.944	23,851,784	159
InceptionResNetV2	215MB	0.804	0.953	55,873,736	572
MobileNet	17MB	0.665	0.871	4,253,864	88

表格 |：各模型准确率、大小等模型参数表

与之前模型的实现方案类似，我们使用了 Keras 的 InceptionResNetV2 模型，模型权重使用预训练权重'imagenet'，利用 GlobalAveragePooling2D()进行均值化，并用 softmax-10 替代原模型的分类器 softmax-1000。

该模型的训练阶段，直接开放所有层进行训练，并提取较优的权重模型进行微调优化，得到收敛最佳的权重模型。

在编译阶段，优化器使用 Adam，loss_function 使用 categorical_cross_entropy，metrics 使用 accuracy，设定 lr=0.0001，dropout=0.4，迭代次数 15，batchsize：32。

微调阶段，optimizer 为 sgd，lr=0.0001，decay=1e-6，momentum=0.9，nesterov=True，dropout=0.5。

使用 Pretrained Xception

Xception 表示 extreme inception，正如其名字表达的，它将 Inception 的原理推向了极致。它的假设是：跨通道的相关性和空间相关性是完全可分离的，最好不要联合映射它们。

在传统的卷积网络中，卷积层会同时寻找跨空间和跨深度的相关性。过滤器同时考虑了一个空间维度和一个跨通道或深度。在 Inception 中，我们将两者稍微分开。我们使用 1×1 卷积将原始输入投射到多个分开的更小输入空间，并对其中的每个输入空间，使用一种不同类型的过滤器来对这些数据进行更小的模块变换。

Xception 更进一步，不再只将输入数据分割成几个压缩的数据块，而是为每个输出通道单独映射空间相关性，然后再执行 1×1 的深度方向卷积来获取跨通道的相关性。这本质上时借鉴了深度方面可分的卷积运算（depthwise separable

convolution），它包含一个深度方向的卷积（为各通道单独执行的空间卷积）和一个逐点的卷积（一个跨通道的 1×1 卷积）。

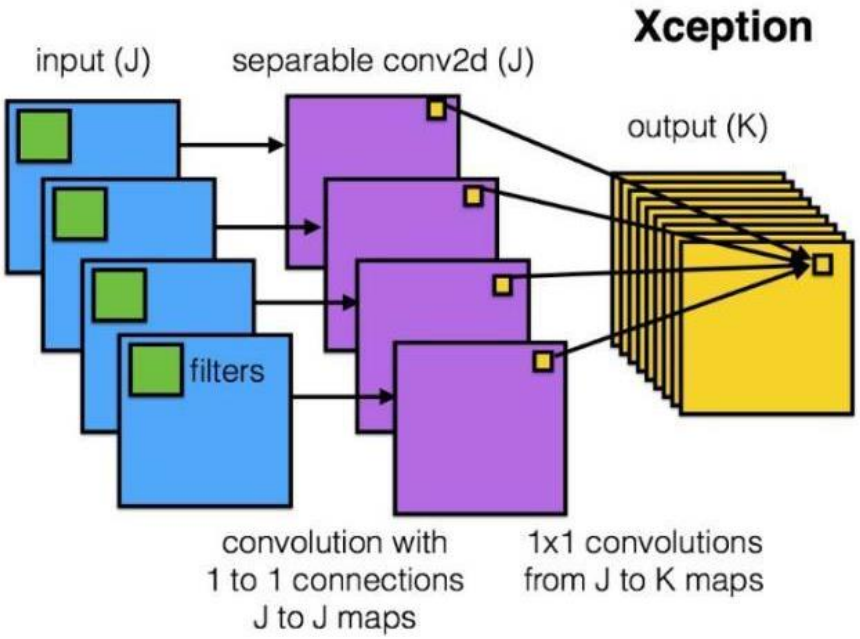


图 6：Xception module

我们使用了 Keras 的 Xception 模型，模型权重使用预训练权重'imagenet'，利用 GlobalAveragePooling2D()进行均值化，并用 softmax-10 替代原模型的分类器 softmax-1000。

该模型的训练阶段，直接开放所有层进行训练，并提取较优的权重模型进行微调优化，得到收敛最佳的权重模型。

在编译阶段，优化器使用 Adam，loss_function 使用 categorical_cross_entropy，metrics 使用 accuracy，设定 lr=0.0001，dropout=0.5，迭代次数 15，batchsize：32。

微调阶段，optimizer 为 sgd，lr=0.00005，decay=1e-6，momentum=0.9，nesterov=True，dropout=0.5。

使用 CAM 技术可视化训练好的 CNNs 模型

通常而言，图片分类的任务往往也需要物体定位（localization），利用物体定位我们可以看出我们的分类模型是否能正确检测到目标分类的图像。本项目我们将使用 Bolei Zhou 提出的 Class_Activation_Mapping[8]来对训练的模型进行物体定位。

CAM 对物体定位的核心思想是，CNNs 的高层卷积层往往「记忆」了目标分类的位置信息，通过将对目标分类敏感的 filters 按权重累加，即可得到目标分类的映射 mapping。

CAM 具体实现大致是，将深层的某卷积层的输出，与最后某目标分类的权重做数量积，再映射回 image space，这样就得到目标分类的可视化图像。该图像颜色深的部位就代表该目标分类的定位。如下图 7 所示。

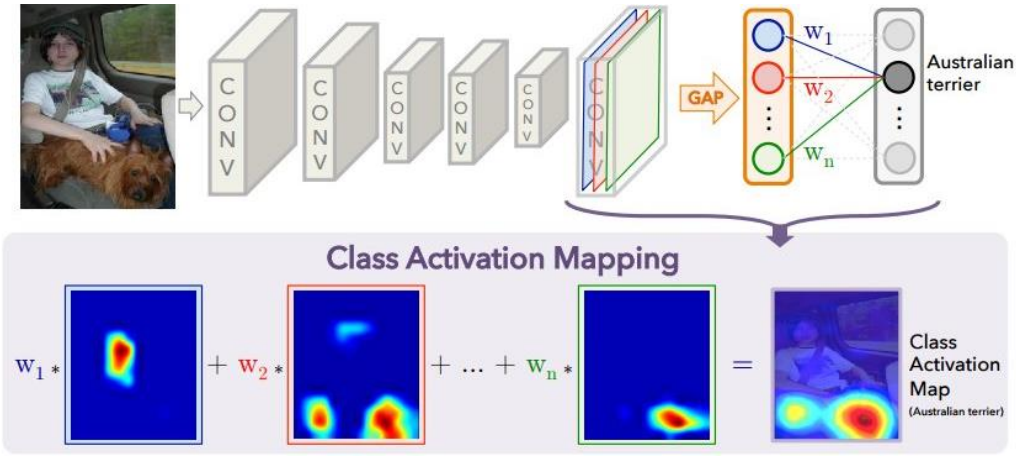


图 7：Class Activation Mapping 算法步骤[10]\

本项目利用 Keras 将训练好的 InceptionV3,InceptionResNetV2、Xception 模型的输出结果 outputs 和 weights 提取输入到函数 heatmap 最后得到模型的 CAM 可视化结果。

优化

在实现和训练 InceptionV3 模型的过程中，发现了很多问题：

- 预测结果并不是很好。
- 模型并没有防止 overfitting 的措施。
- 训练速度较慢。
- 模型训练不充分。

根据这些问题本项目主要采用了如下的优化方案：

- 训练多个模型参数，综合预测结果。
- 使用 Adam 优化器结合 sgd 优化器来挑选和微调模型效果。
- 参数调整优化，针对 Adam 的 lr 可以考虑设置 callback，根据一定规律变化，提高模型训练效果和速度，防止掉入极小值。

在确定这些优化方案之后，InceptionV3、InceptionResNetV2、Xception 全部都使用这些优化方案。这样的好处是，由于预训练模型的 parameters 都非常庞大，所以如果针对不同的优化方案去训练预训练模型的话，太耗时间了，所以一个取巧的办法是确定好优化方案后，直接将这些优化方案实施到预训练模型上。当然，如果有不确定的优化方案，还是应该在预训练模型上进行训练后，在判断该优化方案是否可行。

Adam 粗调，sgd 精调

为了加快模型权重的训练，在编译阶段，所有模型首先使用 adam 进行优化，选择出最优权重，并用 sgd 进一步微调，使模型收敛效果更好。

Early Stopping

在使用深度 CNN 时，往往需要很多 epoch 模型才能收敛。但需要多少 epoch 模型才收敛，这是一个经验问题，如果 epoch 过少则训练不完备，如果 epoch 过多则训练时间过长，并且可能会导致 overfitting。

这里我们使用 Earlystopping 的方法来解决这个经验问题。具体而言，我们使用两种 earlystopping 方法：设定‘val_loss=1e-5’，如果验证集的损失小于这个值时则训练停止；监控‘val_loss’，如果验证集损失有 5 次迭代都没有提高，则停止训练。

4.结果

九、模型评价与验证

根据上一章的实现，我们一次得到如下几个测试结果。

使用 Pretrained InceptionV3

Fine-tune InceptionV3 最后得到的训练集结果如下图 8 所示。上传该模型的测试结果，得到 private-score 为 0.25093，kaggle 排名 138/1440。

比较以上结果，可以看出，预训练模型，这些深度增大的模型学习效果优于我们构建的简单卷积模型，也说明预训练模型用于迁移学习非常有效。

使用 Pretrained InceptionResNetV2

该 Fine-tune InceptionResNetV2 得到的结果如图 9 所示。上传该模型的测试结果，得到 private-score 为 0.28072，排名 173/1440。

使用 Pretrained Xception

该 Fine-tune Xception 得到的结果如图 10 所示。上传该模型的测试结果，得到 private-score 为 0.29234，排名 185/1440。

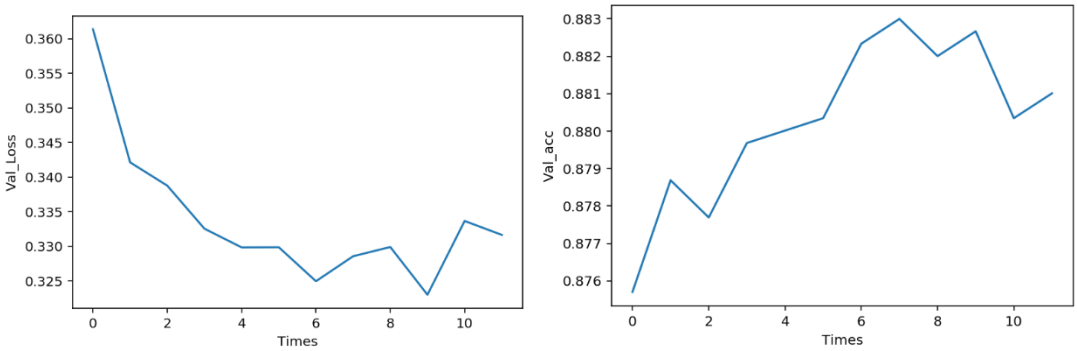


图 8：Pretrained InceptionV3 模型训练结果

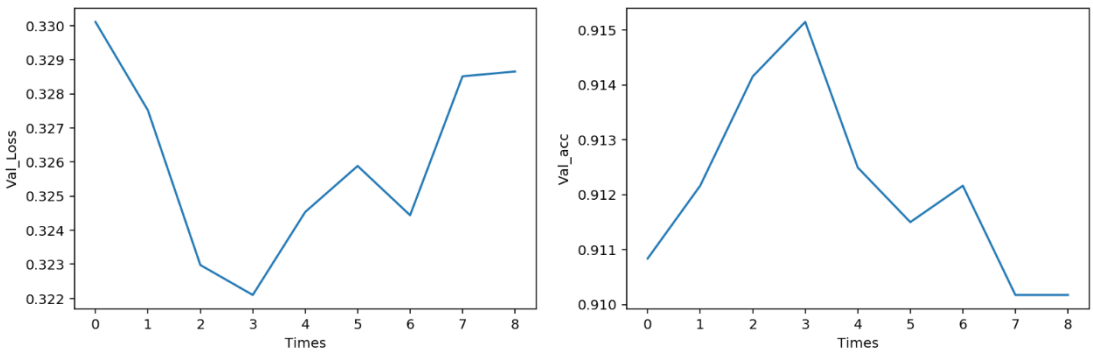


图 9：Pretrained InceptionResNetV2 模型训练结果

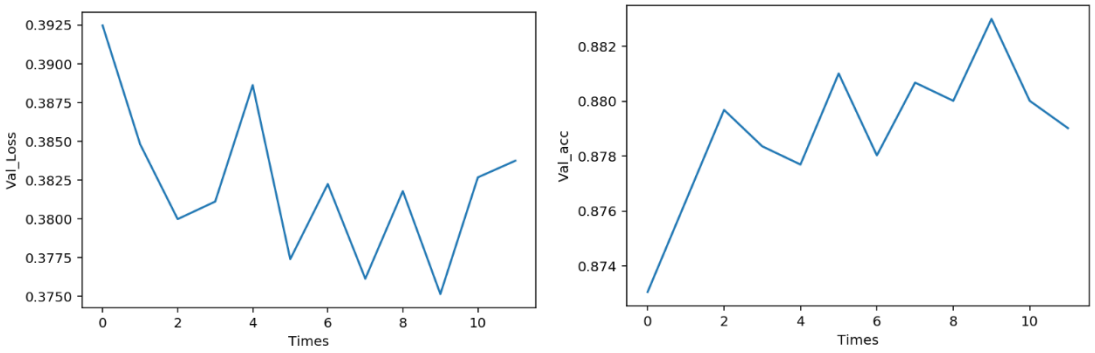


图 10：Pretrained Xception 模型训练结果

比较三种模型，从训练数据（见导出.html 文件）可见，对比 val_acc，InceptionResnetV2 远高于其他模型，但与 Xception 相同存在抖动、容易过拟合的现象，不易得到最优解，从稳定性、准确率等综合考虑，InceptionV3 表现最好。

Merge Xception, InceptionV3 and InceptionResNetV2

通过比较以上三种模型的预测结果，我们发现对于有的图片 Xception 能准确预测，但其他模型却存在预测失败的情况；另外一些图片则情况相反，Xception 得到错误预测，InceptionV3、InceptionResNetV2 却得到了正确预测。

这些预测失败的图片，往往是模型对该图片的结果归类不明确，预测结果概率集中在两个或三个 label 上。但相对别的模型，可能预测确定、训练完备。从以上分析可得出，如果我们综合不同模型的预测结果，可能得到更好的预测结果。

我们综合了 Xception、InceptionV3 和 InceptionResNetV2 的预测结果，得到 private-score 为 0.21150，排名 81/1440。从得分上来看，综合多个模型的表现最好，得分已经达到了 Leaderboard 的 Top-6%，达到了本项目的预期目标。

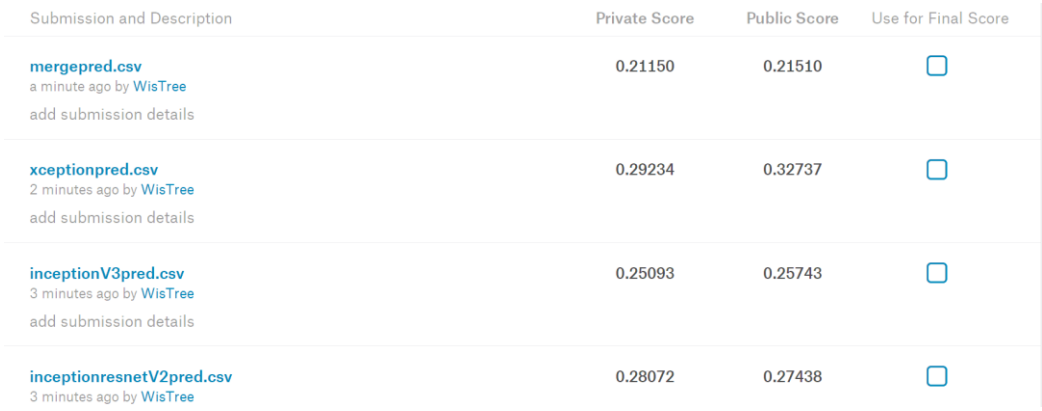


图 10：各训练模型在 Kaggle Leaderboard 上的表现。

模型	training time/epoch (adam)	training time/epoch (sgd)	prediction time/pic
Xception	490s	478s	0.0071s
InceptionV3	415s	410s	0.0058s
InceptionResnetV2	451s	428s	0.0071s

表格 II：各模型训练和预测时间

测试机器为 Google Cloud Tesla P100。从训练效果和时间综合来看 InceptionV3 都是非常不错的选择。当然也有可能，其他模型可以表现更好，但是并没有被我训练完善，但是不管怎么样，在实际使用中，对于响应时间和预测准确度必须有一个权衡，必须在保证较好准确度的情况下，使响应时间尽量短。

Adam&SGD 分步优化对比 Adam 单优化

为了对分步优化和 Adam 单优化有一个直观的认识，这里直接抽取三个模型的训练数据作为比较数据。为了节省篇幅，同时说清关系，这里通过提取模型的最好结果作为比较依据（即验证集效果最好）。

优化方案	Model	Loss	Acc	Time
Adam&SGD	InceptionV3	0.3230	0.8827	825s
Adam		0.3863	0.8886	415s
Adam&SGD	InceptionResnetV2	0.3221	0.9151	878s
Adam		0.3994	0.8936	451s
Adam&SGD	Xception	0.3751	0.8830	490s
Adam		0.4065	0.8787	969s

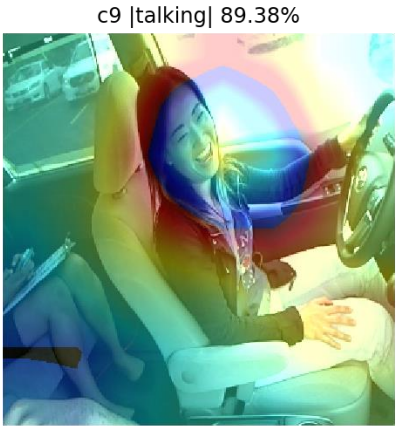
表格 III：各模型训练和预测时间

从上表，我们可以直观发现，Adam 单步的得分普遍要略差于 Adam&SGD 分步优化，特别是 val_loss,当低于 0.35 时，一点点的提升都是至关重要的，而单 Adam 优化很难做到。

当然，选择分步优化的好处，类似于锁层分步优化，保证部分效果较优的基础上进行进一步优化，但其缺点就在于时间必然会延长。（锁层分步优化：锁住模型部分层的权重，重点优化某些层，保证从表层特征识别到抽象特征的学习等各层效果最好。）

十、结果分析

当然在模型的训练过程中，可视化的作用是必不可少的，特别是对特征权重效果的可视化，可以让我们直观明确模型的学习情况。针对以上模型，进行可视化处理，这里通过随机抽取几张图片，展现各模型的学习检测效果，下图所示为部分挑选出来各模型识别有歧义的图片。



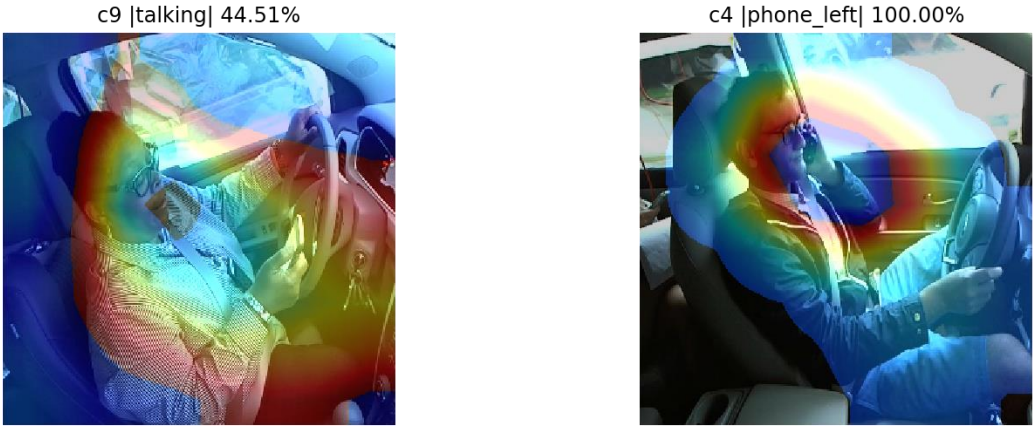


图 11：InceptionV3 可视化结果

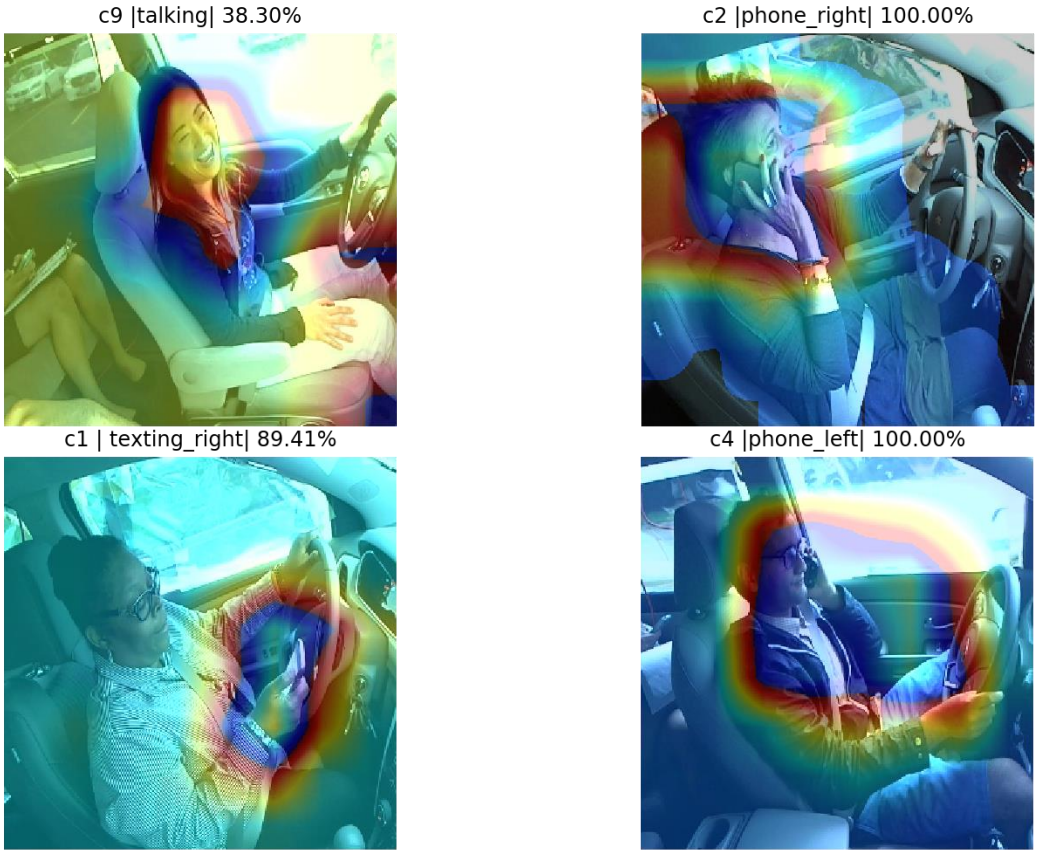
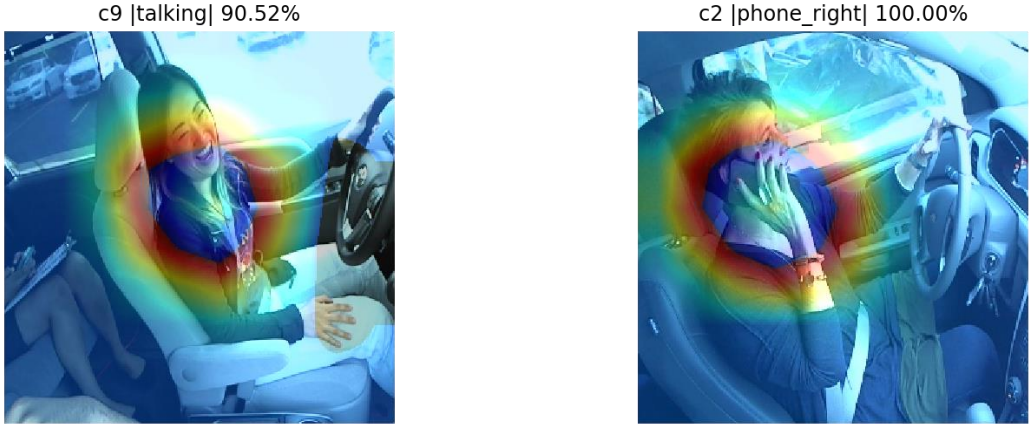
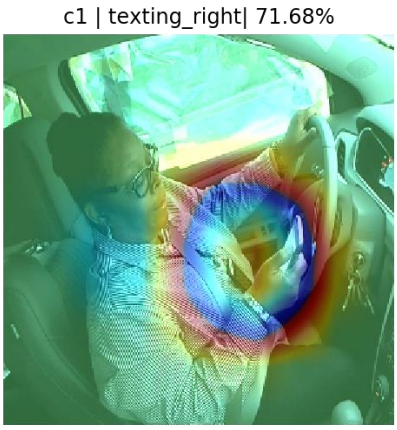


图 12：InceptionResnetV2 可视化结果





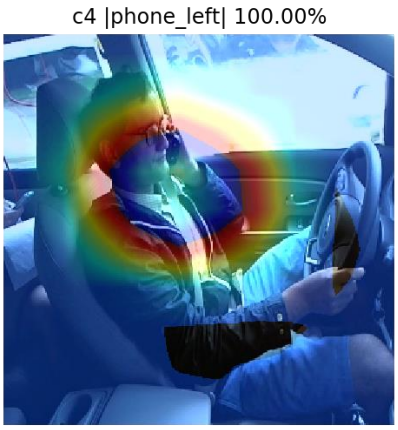
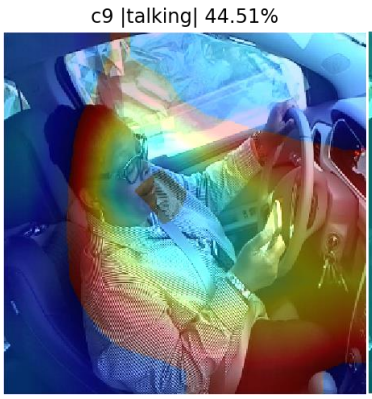
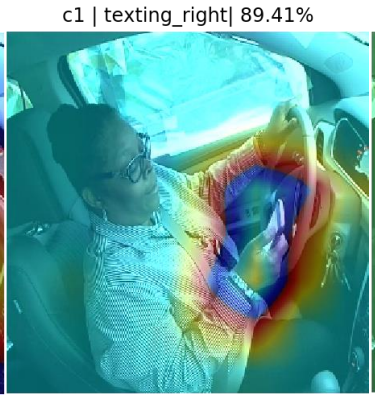
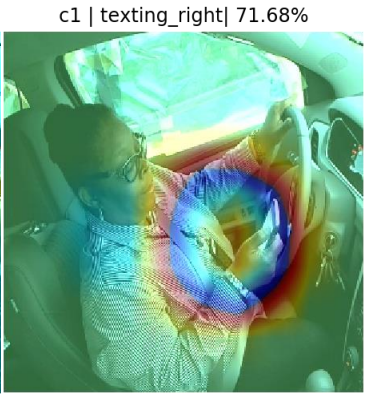


图 13：Xception 可视化结果

下面提出其中一张图片作为说明。







InceptionV3 预测结果　　InceptionResnetV2 预测结果　　Xception 预测结果

图 14：模型预测错误的图片

这张图片最终预测结果为 c1|texting_right|。从 heatmap 我们可以明显看到，InceptionV3 并没有正确学习到特征，或者说对于特征的识别出现了错误。这原因可能是因为图中人物的肤色对图像边缘等形状做了遮挡，导致特征错误识别；手机的位置在方向盘车标处附近，导致手机特征被忽略；InceptionResnetV2 和 Xception 正确根据手机的识别，确认分类为 texting_right。那么把三个模型的预测结果进行均值处理，就可以消除 InceptionV3 的错误结果，实现正确分类。

从上面几组图片可以充分证明前述所说，三种模型的预测结果可能存在差异，甚至截然不同的项目，那么通过综合多种模型的预测结果可以提高分类准确率。

5.结论

十一、总结

了解过简单 CNN 模型无法胜任这类较大数据库的多分类问题，本项目直接使用 InceptionV3，InceptionResnetV2，Xception 等预训练模型进行训练和数据测试，并确定优化方案，再对三种预训练模型结果进行比较和合并结果，选出最优解决方案。

本项目最后取得 Kaggle Leaderboard Top6%的结果，主要得益于两点原因，1.迁移训练的使用，利用各种预训练模型让我们在进行数据训练的时候事半功倍；2.多模型的数据融合，极大的提高了模型训练效果；3.模型微调对于最终提高模型效果有非常直观的作用；4.在数据集的处理中根据司机 id 分训练集和测试集，减少了模型学习到不期望的特征。

十二、后续改进

在模型的训练过程中，并没有对图像进行裁切，事实上很多图像元素是不需要的，我们只需要关注驾驶员的肢体及面部等特征，并不需要去过多关注车身等部分特征，一定程度上可提高训练效果。

通过基于人体骨骼点识别，根据人体骨骼点的完整程度，识别司机肢体，再进行凸包形态学处理，从而把人体相关的部分图片提取出来，完成剪切操作；把图片用白色填充成一个方形图片，此时，图片大小、特征减少，从而使图片可以正常用于训练。

十三、参考文献

[1] Convolutional Neural Networks: Architectures, Convolution / Pooling Layers, http://cs231n.github.io/convolutional-networks/

[2] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

[3] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In CVPR, 2015.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015.

[6] Pan SJ, Yang Q. A survey on transfer learning. IEEE Trans Knowl Data Eng. 2010;22(10):1345–59.

[7] Transfer Learning and Fine-tuning Convolutional Neural Networks, http://cs231n.github.io/transfer-learning/

[8] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In In Advances in Neural Information Processing Systems, 2014.