MLND 毕业项目: 探测走神司机的走神行为

赖天涤

2017 年 9 月 24 日

1 定义

1.1 项目描述

State Farm Distracted Driver Detection 是 2016 年 Kaggle 上的一个数据科学比赛, 比赛提供了 22424 张司机驾驶汽车时的行为动作照片, 并提供对应图片的动作标记, 标记司机是处于正常, 左手输入, 右手输入, 左手打电话, 右手打电话, 化妆整理头发, 调收音机, 喝水, 和其它乘客谈话, 拿后面的东西等十个状态. 比赛参与者需要构建自己的模型, 并运用提供的图片标签训练模型, 然后对比赛提供的无标签的司机行为图片进行分类, 并使分类结果尽可能准确. 参与者把分类结果提交给 Kaggle, 大赛方对提交的数据进行评估, 并根据评分进行排名.

1.2 解决方案

图片分类问题上, 现在用得最广泛的是**卷积神经网络** (Convolutional Neural Network,**CNN**) 模型, 在历年的 IamgeNet 大规模视觉识别挑战赛上, 获奖的方案基本是卷积神经网络以及各种 CNN 的变体. 比如 2014 年的 VGGNet 就是 6-16 层 CNN, 卷积神经网络的层次越深, 模型的抽象能力越强, 就越能准确地对图片分类,2016 年的 ResNeXt 就多达 156 层. 在这个项目里面我采用 VGG-16, 以及 Network in Network 去识别司机的走神行为, 同时根据模型能解释原因的需求, 对 VGG 采用了 **CAM**(Class Activation Mapping), 以生成热图.

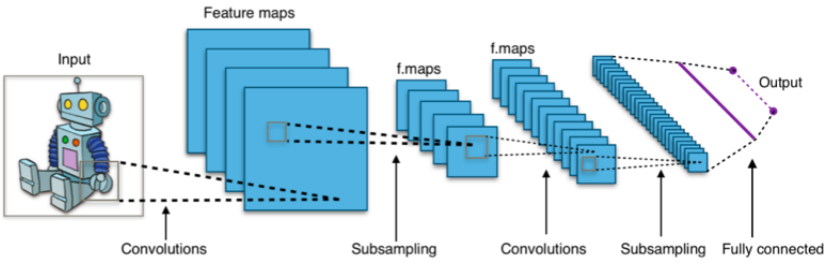

图 1: 卷积神经网络

1.3 模型性能评估

Kaggle 上提供的评估方法是:

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

其中 N 是上面提到的无标签测试照片的总数,M 是司机行为的总数, 也就是照片待归类的分类总数, 在这里 $M = 10$, log 是自然对数, 如果图片 i 属于分类 j ,y_{ij} 就等于 1, 否则等于 0, p_{ij} 是我们模型预测的图片 i 属于分类 j 的概率.Kaggle 会对提交的数据进行缩放, 使每一张图片在各个分类上的概率的总和为 1, 并把一些过于大或者过于小的特异点用公式 $max(min(p, 1 - 10^{-}15), 10^{-}15)$ 代替.

2 分析

2.1 卷积神经网络

这个项目用到的卷积神经网络, 通过感受野扫描图像, 把图像感受野内的像素与过滤器的参数进行点积运算生成新的特征层, 然后在新生成的特征层再进行新的过滤器扫描. 图 1 所示, 是两层卷积网络, 两层池化层, 再加上一层全连接层的 CNN.

2.2 VGGNet

VGGNet[1] 是牛津大学计算机视觉 (VisualGeometry Group) 和 DeepMind 发现的深度卷积网络, 网络通过反复堆叠卷积核大小为 $3 * 3$ 的卷

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

图 2: VGGNet

积层和 $2*2$ 的池化层, 构筑 11-19 层的卷积神经网络.VGGNet 获得了 ILSVRC2014 年的定位项目冠军分类项目亚军.VGGNet 的参数非常简单都是 $3*3$ 的卷积核心和 $2*2$ 的池化核, 非常方便实现. 在这个项目里, 采用了图二所示的 16 层网络结构, 把最后三层全连接层去掉用 GAP 代替.

2.3 CAM

这个项目提出了需要模型能解释分类的原因, 这个项目使用了 MIT 的人工智能实验室发现的 CAM[2](Class activation mapping). 图 3 所示, 把卷积神经网络的全连接层用全局均值池化层 (Global average pooling,GAP) 代替, 把池化后的矩阵通过全连接层进行分类. 把进入 GAP 的特征层与全

图 3: Class Activation Mapping

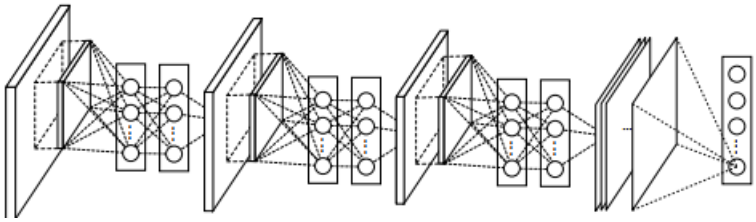

图 4: Network in Network

连接层的参数相乘求和可以得到当前模型的注意点.

2.4 NIN

在这个项目了, 还探索了 Network in Network[3], 在 VGGNet 的 16layers 模型里, 最后的全连接层的参数规模是 $7*7*512*4096$,$4096*4096$, 参数非常多, 如此大规模的全连接层不仅使全连接层的参数占了模型的大部分, 还很容易导致模型的过拟合.Network in Network 提出了一种解决办法, 如图 4: 把传统卷积神经网络后的全连接层用 GAP 代替, 然后用一层全连接层分类, 在卷积层的卷积核后面加入多层感知网络, 如图所示的是三层 MLPlayers, 每层 MLP 由一层普通卷积层和两层卷积核大小为 $1*1$ 的卷积层 (实现多层感知网络) 堆叠而成.Network in Network 相对 VGGNet 等模型大大减小了参数规模, 而且也由于每层卷积网络的抽象能力得到了提高, 模型的性能比 VGGNet 有了小幅度的提升.

2.5 dropout

为了防止过拟合, 项目里使用了 Dropout[4],Dropout 是 Hinton 在 2012 年提出的用于改善神经网络过拟合的 trick.Dropout 通过模型在进行误差反向传播的时候对一部分隐藏层的参数不更新, 使模型的隐藏节点能学习到更多能单独工作的特征, 而不是所有节点联合工作的特征. 使用了 droptout 的模型在泛化时能获得更好的效果, 如图 5 所示, 模型在对测试集分类时获得了更好的 loss.

2.6 目标

在图片归类问题上, 我们参考 CIFAR-10 这个 kaggle 比赛,CIFAR-10 是把图片归类为飞机, 鸟, 猫, 狗等 10 个分类,CIFAR-10 比侦测走神司机的走神动作更困难, 原因在同类图片间有很大的差别. 在 CIFAR-10 上, 最好的成绩为 95.53% 的准确率. 因此, 这个项目的目标是对司机的走神行为侦测的准确率最终能达到 95% 以上.

3 实现

3.1 数据预处理

比赛提供的训练图片 (图 6 所示) 是分辨率为 $640 * 480$ 的 JPG 格式, 因为 VGGNet 对输入的规定为 $224 * 224$, 而增大输入会导致模型参数过大, 用于训练的机器无法运行, 所以这里把比赛图片 resize 为 $224 * 224$ 分辨率. 项目还对图片输入标准化,3 个 channel 的每个元素都除以 255, 并减去 0.5, 使输入数据都落入 (-0.5,0.5) 的区间. 对大赛提供的"driverimagslist.csv" 有三列数据, 项目提取了"classname","img" 两个 column,"img" 列描述的是图片的路径,"classname" 是对应"img" 图片的归类, 项目用 pandas 的 dummy 方法对"classname" 进行 onehot 处理.

3.2 模型结构描述

项目里面实现了三个模型 (表 1), 第一个命名为 vgg13_cam, 是由 13 层的卷积层, 一层全局池化层, 一层全连接层堆叠而成, 在每层卷积层里, 把激活函数 relu 改用 leaky relu. leaky relu 的公式为:$max(0.1 * x, x)$, relu 在输入为负数的时候会在误差反向传播时候对更新进行阻断, 如果网络很大部

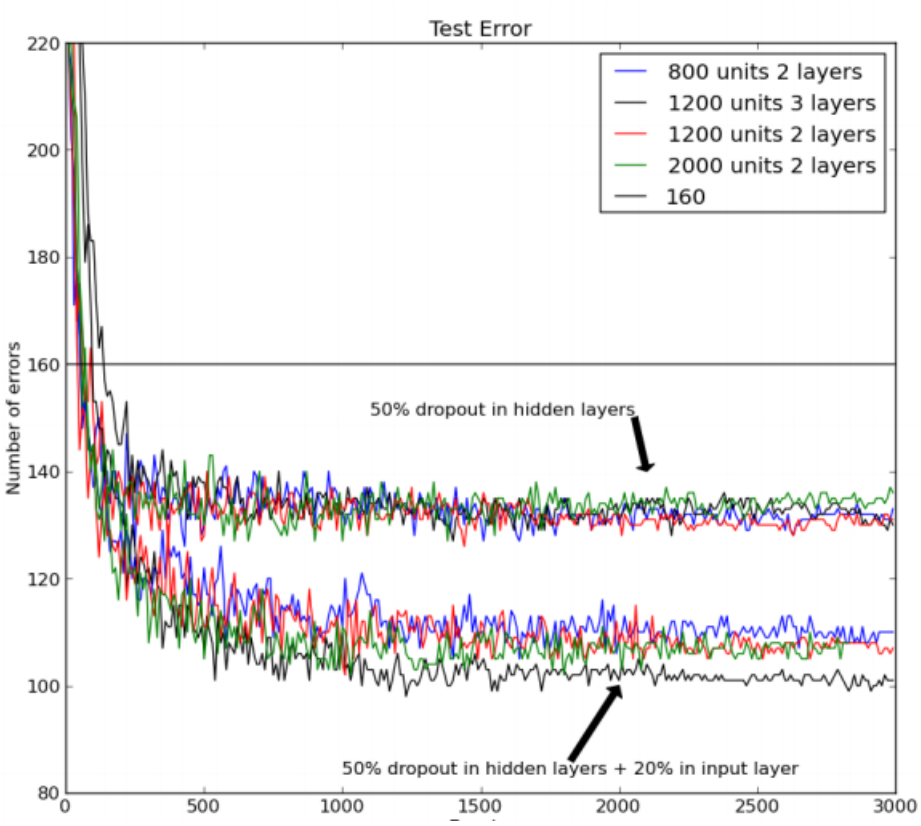

图 5: Network in Network

图 6: Drink

图 7: 生成热图

分输入都处于负数状态, 会导致网络陷入不可训练的状态,leaky relu 可以让网络输入为负的时候, 依然进行小幅度的更新参数. 另外, 为了提高网络训练速度和降低调参难度, 在每个卷积层都引入了 batch normalization. 第二个模型是如图 4 所示的 Network in Network(nin), 在每层 conv 里面, 依然使用了 batch normalization, 激活函数是 relu, 而每层 MLP 都是通过使用 kernel size 为 $1*1$, 步进间隔为 1 的卷积层实现. 第三个模型 deep_nin 是在第二个 NIN 模型的基础上添加更多的卷积层和 MLP, 尝试验证增加 NIN 的深度是否能提高模型的性能,deep_nin 在 nin 的基础上增加了 3 层卷积层和 3 层 MLP 层 (表 1). 三个模型的损失 (loss) 都是用交叉熵求得:

$$-\sum_{x\in X} p(x)logq(x)$$

3.3 热图的生成

三种结构模型的末段都是以卷积层接上 GAP 再接上全连接层, 为了制作热图, 首先需要把 GAP 层的输入提取出来, 也就是在 forward 过程中保存最后卷积层的运算结果, 保存出来的运算结果的形状为:$14*14*512$(vgg_gam),$28*28*512$(nin),$28*28*512$(deep_nin). 在从已训练好的模型中提取最后一层全连接层的参数, 用被激活的参数与前面保存的卷积运算结果相乘并求和, 用 threshold 过滤掉小于 0.1 的值, 用 0 代替 (避免最终的合成整张图片偏蓝). 然后转换成 jet 形式的 heatmap(图 7 的左 1), 把转换后的热图与原图相加, 得到了图 7 右 1 的合成结果. 合成图表示模型注意到了当前图片里司机右手在靠近中控的位置, 所以模型认为司机在调收音机.

表 1: 实现的三个模型

vgg13_gam	**nin**	**deep_nin**
conv1+relu+bn	conv1+relu+bn	conv1+relu+bn
conv2+relu+bn	MLP	conv2+relu+bn
max_pool	MLP	MLP
conv3+relu+bn	max_pool	MLP
conv4+relu+bn	conv2+relu+bn	MLP
max_pool	MLP	max_pool
conv5+relu+bn	MLP	conv3+relu+bn
conv6+relu+bn	max_pool	conv4+relu+bn
conv7+relu+bn	conv3+relu+bn	MLP
max_pool	MLP	MLP
conv8+relu+bn	MLP	MLP
conv9+relu+bn	max_pool	max_pool
conv10+relu+bn	conv+relu	conv5+relu+bn
max_pool	conv+relu	conv6+relu+bn
conv11+relu+bn	conv+relu	MLP
conv12+relu+bn	gap	MLP
conv13+relu+bn	fully_connect	MLP
max_pool		max_pool
gap		conv7+relu+bn
fully_connect		conv8+relu+bn
		conv9+relu+bn
		gap
		fully_connect

表 2: 系统配置

操作系统	windows10
CPU	i7 6770
RAM	16G
GPU	8G GTX1080

3.4 硬件环境和软件依赖

模型在一台安装了 CUDA 的 windows 操作系统电脑上训练和预测, 使用 tensorflow 实现模型,pandas,numpy 进行数据预处理,pillow 和 matplotpy 用于图片的读取打印和制作热图. 电脑的 CPU 型号为 i7 6770, 内存 16G, 显卡是 8G 的 GTX1080(表 2).

4 过程与改进

(1) 在项目刚开始, 先实现了一个 6 层的 VGGNet, 模型在 train set 上训练一个 epoch, 训练时候的准确率达到了 95%, 但生成的 csv 提交到 kaggle,Private Score:14.4,Public Score:15.1, 比大赛提供的 sample_submission(一个不做任何预测的结果) 还差. 最后分析可能是 kaggle 的评估函数对错误预测有很高的惩罚, 而且 6 层的 VGG 因为抽象能力很低泛化能力低, 在测试集上的预测效果不好. 最终更复杂的模型运算结果的确比 6 层的 VGGNet 好了很多.(2) 在最开始的数据预处理, 一次性读入了所有的图片并转换成 numpy 的 array, 然而发现这种操作会占用极大的内存, 训练集约 1GB 的图片转换后占用了差不多 20GB 的内存, 这样不仅导致了机器的性能降低, 模型每次真正开始训练前还花费大量时间读取图片到内存, 最后改成了用 yield 生成器, 每次只读取一个 batch 的图片.(3) 在训练 16 层 VGG 模型的时候, 模型对 learning rate 特别敏感, 尝试了 0.01,0.001,0.002,0.005 的 learning rate, 模型都没办法有很好的训练效果, 最终在每层卷积网络后面加入了 batch normalization 后, 情况得到了很大改善, 模型的训练速度也变快了, 一个 epoch 就到了 95% 的准确率.(4) 在热图生成函数里,jet heat map 和原图相加, 刚开始热图分配的权值是 127, 原图分配的是 128, 发现合成结果原图会很暗, 热区域完全覆盖了原图的信息, 最后把权重调整到 95,159, 很好地兼顾了原图信息和热区域的显示.(5) 在刚开始训练模型时没有保存

checkpoint, 到后面每个 epoch 都保存一次, 在验证模型的时候得到了极大的方便.

5 结果

对 vgg13_gam 模型进行了 10 折 KFold,1 epoch 的训练, 最后模型在训练上的准确率为 99.85%, kaggle 最终对结果的评分为 Private Score:1.467, Public Score:1.453. 模型的热图如图 8, 我们把此模型在未参与训练的测试集合上跑 20 个样本, 得到图 11 的结果, 我们发现除了第二行第二个预测错误外, 其它都正确, 而这个预测错误的司机是在双手离开方向盘, 头偏向副驾驶作笑脸, 我们人类很容易判别出司机是在和副驾驶谈话, 可是对于模型来说模型见过的司机在和其它乘客说话的样本都是双手握着方向盘的, 所以模型出现了错误的判断.

对 nin 模型进行了 10 折 KFold,3 epoch 的训练, 最后模型在训练上的准确率为 99.9%,kaggle 最终对预测结果的评分为 Private Score:1.374,Public Score:1.291. 模型的热图如图 9, 我们把此模型在未参与训练的测试集合上跑 20 个样本, 得到图 12 的结果, 我们发现除了第二行第三个预测结果与图 11(vgg_gam) 不同外, 其它结果都一样. 对这个不同的预测分析发现, 这个司机头偏向右边, 但是嘴型是闭合的, 双手握在方向盘上, 在人类看来, 司机处在和其它乘客说话 (恰好在说话间隔, 嘴巴刚好闭上) 和正常驾驶 (看右边的后视镜) 都是有可能的, 所以模型预测出不同的结果也是可以理解的.

对 deep_nin 模型进行了 10 折 KFold,3 epoch 的训练, 最后模型在训练上的准确率为 99.2%, kaggle 最终对预测结果的评分为 Private Score:1.544, Public Score:1.518. 模型的热图如图 10, 我们把此模型在未参与训练的测试集合上跑 20 个样本, 得到图 13 的结果. 我们发现在预测里跟上面讨论的一样, 第二行第二个和第三个都预测得跟我们人类分类不一样, 这到底是模型对还是我的判断对都有争议.

对比三个模型的热图, 在驾驶员喝水情景下, 模型关注的是驾驶员右手拿着杯子放在嘴边, 调收音机情景下, 模型关注驾驶员的右手是否靠近中控的位置, 而左手右手打电话, 左右手发短信, 模型会同时关注左右手的动作姿势, 联合起来判断; 而驾驶员拿后面座位的东西这个情景, 模型会观察驾驶员全身的姿态, 而不仅仅是身体姿态的具体一小部分.

三个模型的成绩对比 (表 3), 成绩并不是很好, 但考虑到最近两年出现

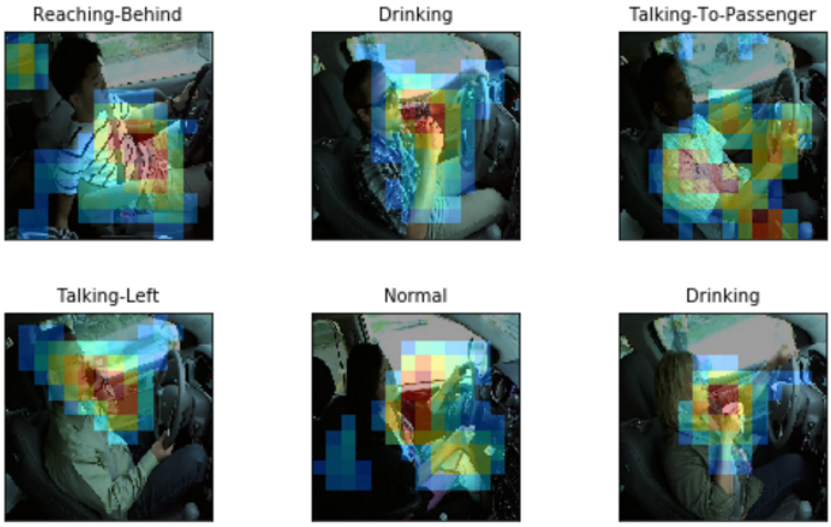

图 8: vgg heat map

表 3: 模型成绩

	train_acc	**public_score**	**private_score**	**rank**
vgg_gam	99.85%	1.467	1.453	645/1440
nin	99.90%	1.374	1.291	610/1440
deep_nin	99.27%	1.518	1.518	665/1440

了很多表现优秀的模型可以实现更好的分类准确度, 尤其是其特别优秀的 pre-train 参数, 可以获得更加好的泛化性能, 对比之下, 成绩还在接受范围.

6 总结

这个项目大大提高了自己使用 tensorflow 实现模型的能力, 加深了对卷积神经网络的了解, 同时也熟悉了卷积网络一些加速训练防止过拟合和调参上的 trick. 但回顾同时发现很多存在的问题:

(1) 图片在进行预处理时候, 尺寸从 $640 * 480$ resize 到 $224 * 224$, 这个过程丢失了大量的信息, 如果这些信息能保存下来, 可能可以获得更好的成绩.

(2) 为了充分利用训练数据, 并没有从数据划分独立的测试集合, 没能有

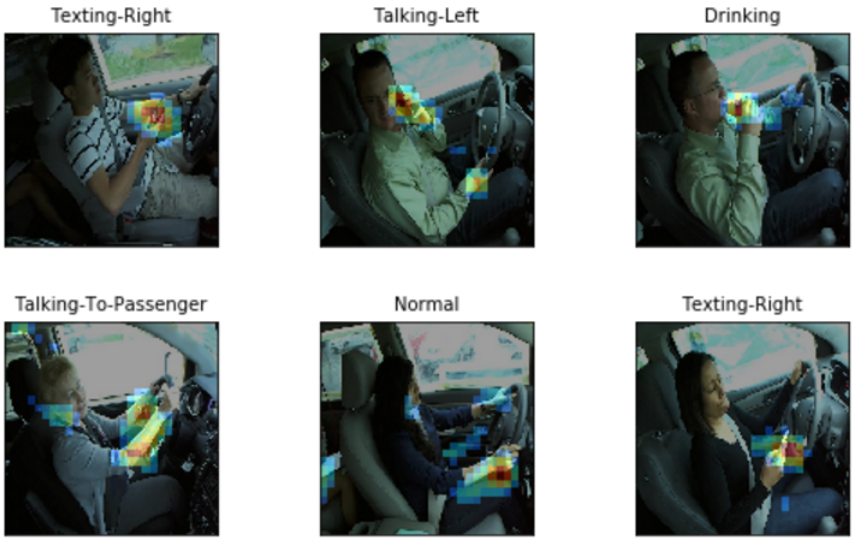

图 9: nin heat map

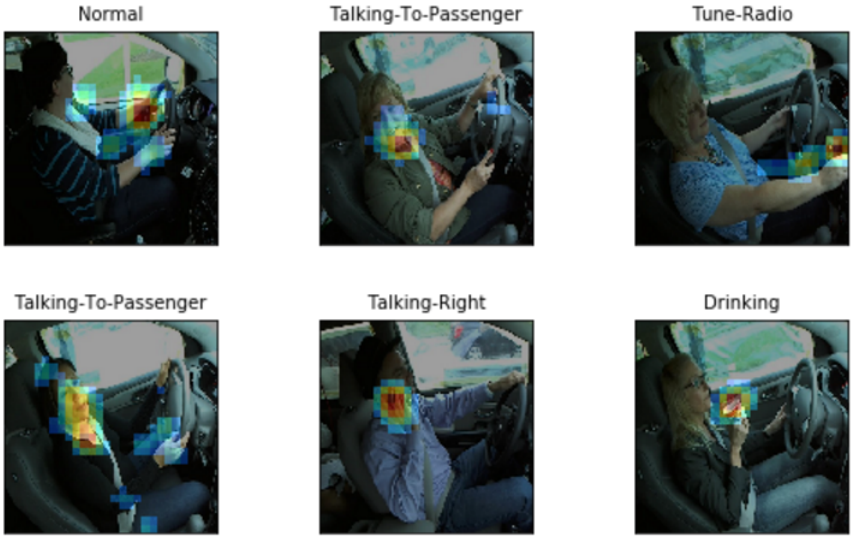

图 10: deep_nin heat map

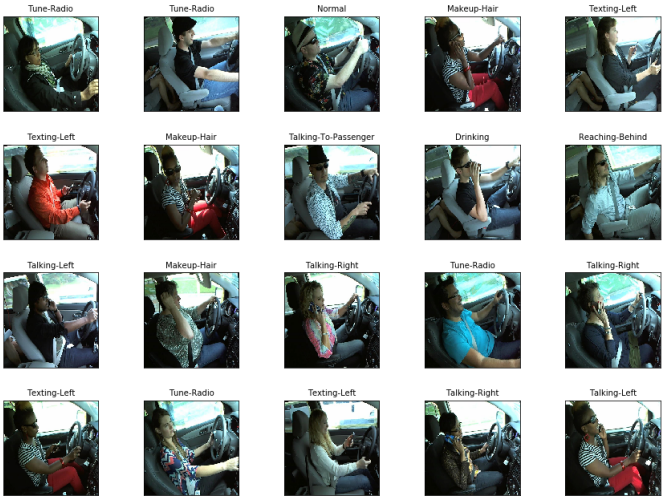

图 11: vgg run on test set

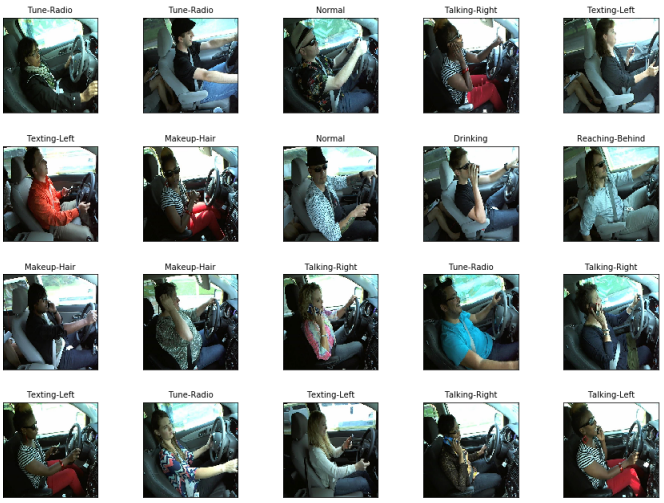

图 12: nin run on test set

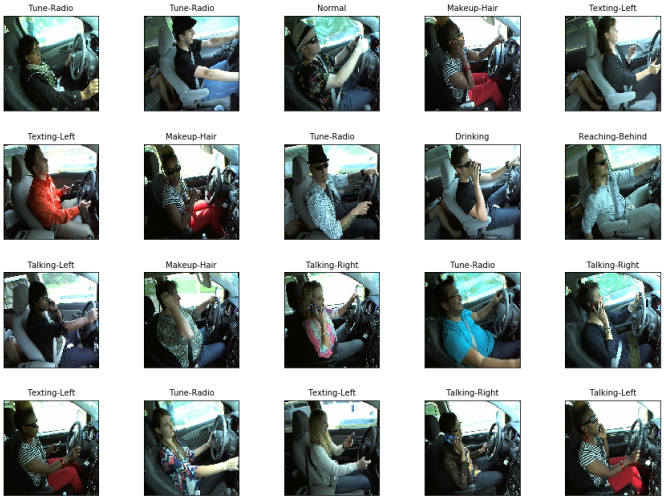

图 13: deep_nin run on test set

效地观察模型是否过拟合, 只能通过提交的每个 epoch 生成的 csv 获得成绩去猜测模型训练情况.

(3) 在训练时候保存的 checkpoint 太少, 有时候错过了, 又要重新进行训练.

(4) 在制作热力图的时候, 发现了一个很有趣的现象, 用 VGG 模型出来的热力图更自然更具有连续性, 具体表现为在关注焦点与非关注区域之间有大片的连续的梯度下降. 而对比起来,NIN 的热力图显得非常之集中, 关注焦点和非关注区域之间几乎没有多少连续变化的梯度, 可以认为 VGG_gam 模型是从关注点与周围的联系与结构关系对图片分类, 而 NIN 是从关注点是否出现某特征对图片进行分类. 同时, 我还发现,NIN 最后面卷积网络 (MLP 之后接上的卷积网络) 的层数对热力图有相当大的影响, 层数是一层的时候, 热力图非常混乱, 关注点散落在图片各处, 看不出模型当前的关注点, 在把卷积网络增加到 3 层后, 热力图就变得非常有条理 (图 7).

参考文献

[1] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[2] B. Zhou, A. Khosla, L. A., A. Oliva, and A. Torralba, “Learning Deep Features for Discriminative Localization.” *CVPR*, 2016.

[3] M. Lin, Q. Chen, and S. Yan, “Network in network,” *arXiv preprint arXiv:1312.4400*, 2013.

[4] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” *arXiv preprint arXiv:1207.0580*, 2012.