合包项目阶段性总结

1. Summary

The goal of this project is to combine different package names that belong to the same app together. The goal also has an emphasis on the high accuracy and recall on top 10 thousand apps (based on the number of device coverage).

2. 背景

合包需求来源于公司内部。公司需要对手机上来的 applist 做标准化。标准化数据主要用于观象台的 app 排行榜及海外业务和 MC。应用兴趣的标签页也主要依赖于 app 标准化。

先前公司针对合包有过两次尝试：

第一次：历史的合包程序

历史合包程序使用的是海明距离算法，经过验证，这套算法的正确率召回率均比较低，且海明距离比较适合长文本的相似度计算，不适合我们的合包场景。之前在人工的帮助下，合包正确率约 93%

第二次：tamr 的合包算法

tamr 使用的是 cosin 相似度算法，输入样本，自己设置相似度规则，然后根据规则出来的配对数据，人工进行标注是否配对，标注的结果供机器学习。

	总样本	tamr 合包数量	验证数量	有效样本	正确数量	正确率
第一次结果(相似度阈值=70%)	14020	1516	702	667	489	73%
第二次结果(相似度阈值=50%)	14020	806	806	766	687	89.6%

注：有效样本为验证数据去掉不确定的数据

即 4 月 10 日 调整后（针对包名中的后缀进行了规则处理，如.BAIDU，.HUAWEI）tamr 最终结果正确率 89.6%，预估召回率 68.1%。

总结: 机器学习可以解决一部分问题，但是不够灵活，我们对数据结果没有掌控度，如果机器学习能够加上一些人为制定的规则，应该会更理想些。

3. 目标

正确率 90%以上，召回率 80%以上

4. 数据来源

数据来源于张晓宇。是基于应用名称从多个主流应用商店爬取的包括如下字段信息：应用商店名称，应用包名，应用描述，应用名称，应用商店 URL，开发者信息，应用下载次数，应用 hash 值，应用图标 URL，应用主分类（来源于应用商店），应用子标签（来源于应用商店），应用版本号。

（1） 有标签数据

基于人工标注的方式制造了样本量为 19289（不同包名），包含不重复 metaid 15502 个的标签数据集。该数据集覆盖了设备覆盖量 top 1 万的应用（也是我们希望能极大程度保证准确率的应用）

HDFS 路径：/datalab/user/kefu/packageNameCombine/data/packageName.csv

（2） 无标签数据

目前从应用商店爬取的数据

Android 数据：/datascience/datacloud/datagroup/data/zhuaqu/andriod

IOS 数据：/datascience/datacloud/datagroup/data/zhuaqu/ios

5. 特征

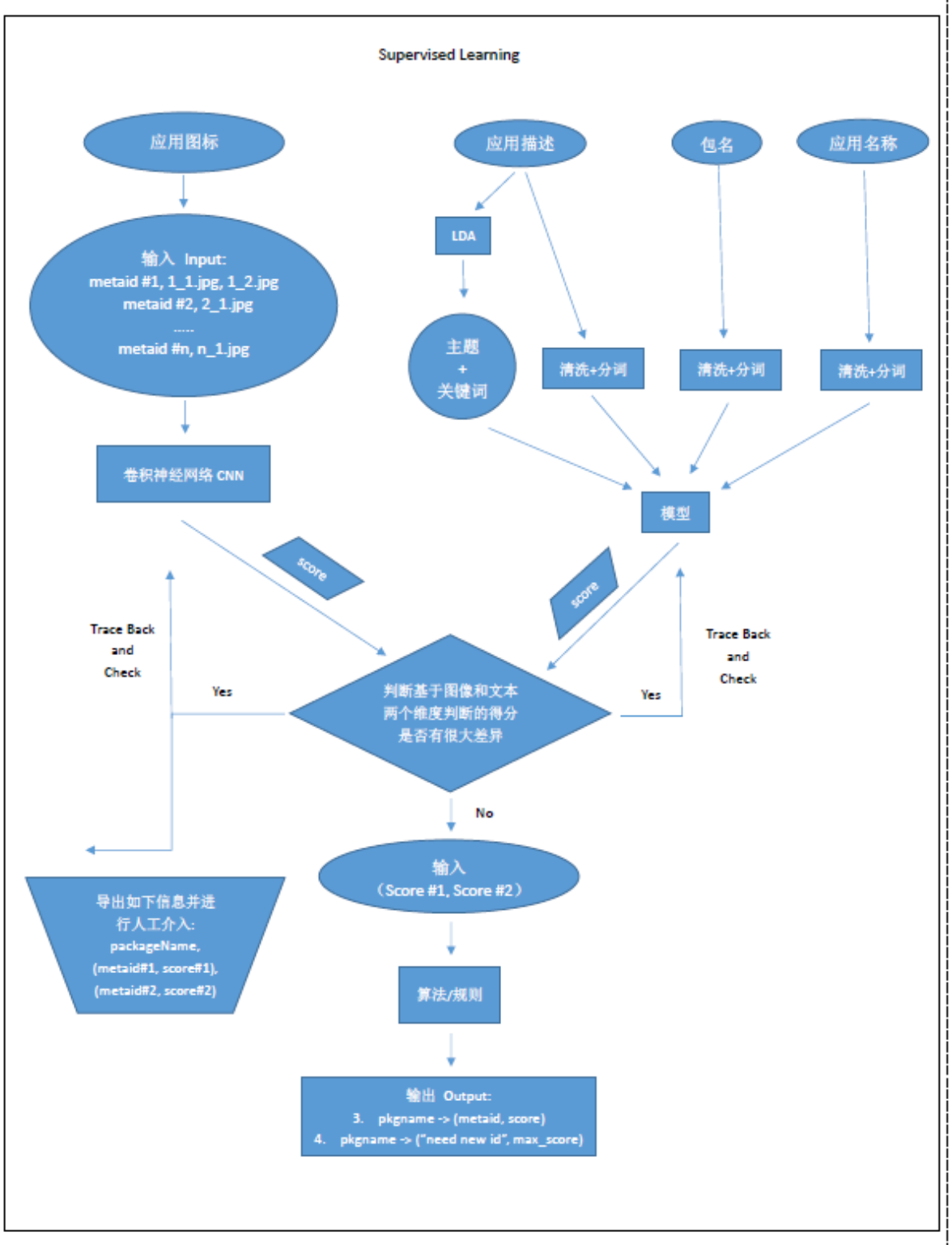
目前考虑会用到的特征包括图标，描述，包名，名称和开发者信息。其中在描述和名称的使用上会采用一些分词或者主题模型的方法提取出一些关键词或是抽象属性（游戏/社交/工具等）

6. 思路方案

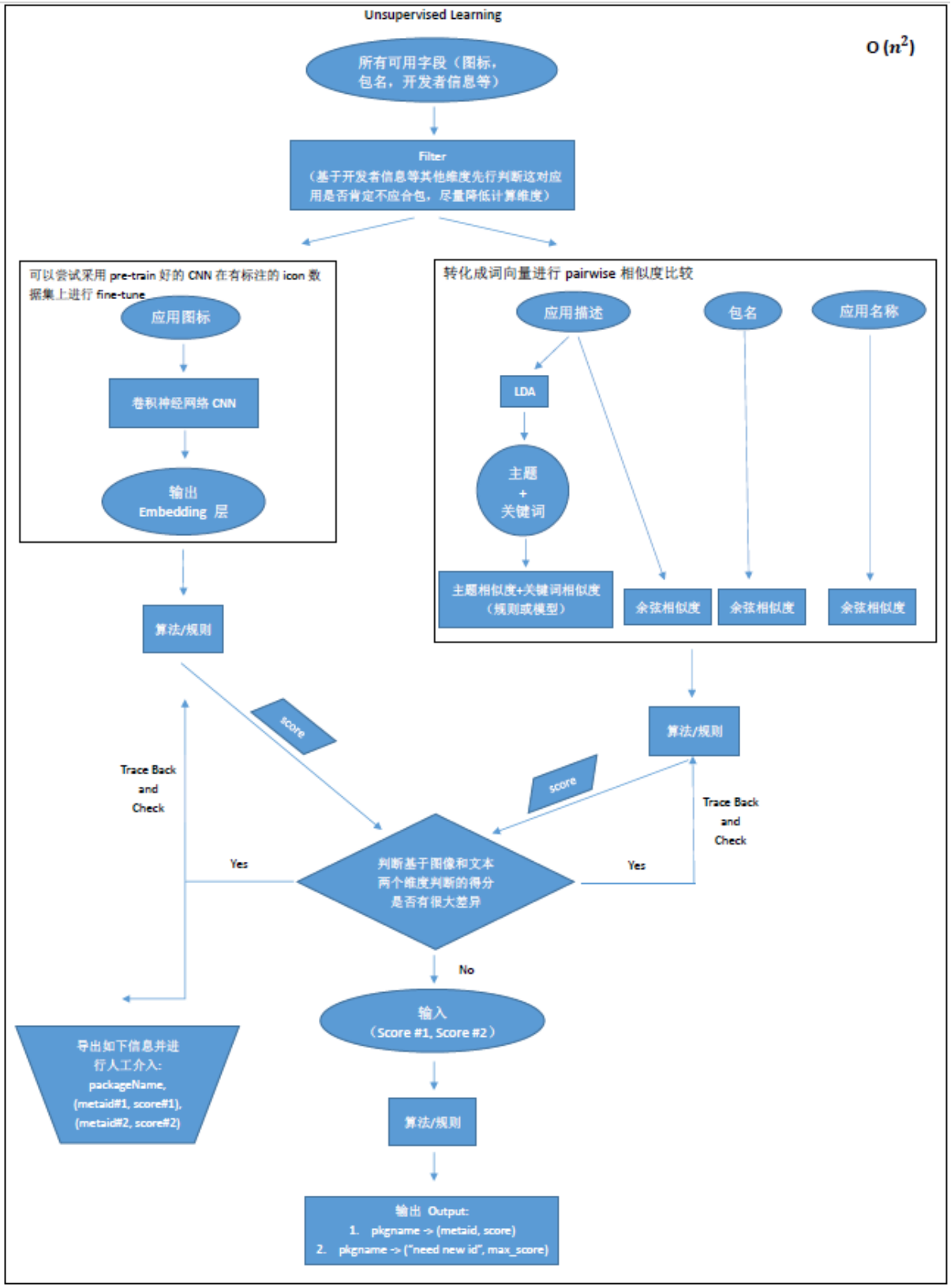
a) 监督性学习（主要针对有标签的 Top 1 万应用）

每日监控：

1. 通过自动化脚本从主流应用商店爬取 Top 1w 应用名称对应的图标 icon 并返回 (metaid，应用名称，应用图标 icon) 字段
2. 利用已有的 CNN 预测所有当天爬取的应用图标 icon，并将预测结果与实际的 metaid 进行比较
3. 如果有任何应用预测错误，发出报警并人工介入查找原因（很大可能是某个应用的图标进行了大的改变）
   a) 如果某些应用有了全新的图标，更新训练数据集并重新训练 CNN 模型



b) 非监督性学习


Unsupervised Learning
O (n^2)
所有可用字段（图标，包名，开发者信息等）
Filter
（基于开发者信息等其他维度先行判断这对应用是否肯定不应合包，尽量降低计算维度）
可以尝试采用 pre-train 好的 CNN 在有标注的 icon 数据集上进行 fine-tune
应用图标
卷积神经网络 CNN
输出 Embedding 层
转化成词向量进行 pairwise 相似度比较
应用描述
包名
应用名称
LDA
主题 + 关键词
主题相似度+关键词相似度（规则或模型）
余弦相似度
余弦相似度
余弦相似度
算法/规则
score
Trace Back and Check
判断基于图像和文本两个维度判断的得分是否有很大差异
Yes
No
score
算法/规则
Trace Back and Check
Yes
导出如下信息并进行人工介入:
packageName,
(metaid#1, score#1),
(metaid#2, score#2)
输入
（Score #1, Score #2）
算法/规则
输出 Output:
1. pkgname -> (metaid, score)
2. pkgname -> ("need new id", max_score)

7. 实验

a) 实验过程

i. 准备数据：

从 19289 张标签应用图标图片（包含 15502 个不同 metaid）中随机抽取 70%的数据作为学习样本（11347 张图片）。每张图片转化成 RGB 格式，去除 alpha 层（图片透明层）。如图片本身为灰度图片，则复制其像素数值 3 次构建成 3 维矩阵与其他 RGB 图片维度保持一致。每次图片读取 batch size 为 10 张图片，读取时对图片进行下列操作增加噪声：

- 随机对图片进行水平旋转（50%概率）
- 对图片进行一定角度的旋转（+/-10 度）
- 将图片 resize 到 50x50 像素

ii. 构建卷积神经网络：

4 层卷积，1 层全连接。每层卷积神经元数目依次为 6,16,32。每个卷积层直接采用 Rectified Linear Unit (Relu)作为激活函数。在全连接层前采用 Global Average Pooling 的方式提取特征。每次卷积操作时使用的 filter 为 2x2 的窗口。最终输出采用 softmax 函数输出的对数结果

```
def GlobalAvgPool(x):
    x = torch.mean(x.view(x.size(0), x.size(1), -1), dim=2)
    return x
```

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 2)
        self.conv2 = nn.Conv2d(6, 16, 2)
        self.conv3 = nn.Conv2d(16, 32, 2)
        self.conv4 = nn.Conv2d(32, 64, 2)
        self.fc1 = nn.Linear(64, 10826+1)

    def forward(self, x):
        in_size = x.size(0)
        x = F.relu(self.conv1(x))
        x = F.relu(self.conv2(x))
        x = F.relu(self.conv3(x))
        x = F.relu(self.conv4(x))
        x = GlobalAvgPool(x)
        x = x.view(in_size, -1)
        x = self.fc1(x)
        return F.log_softmax(x)
```

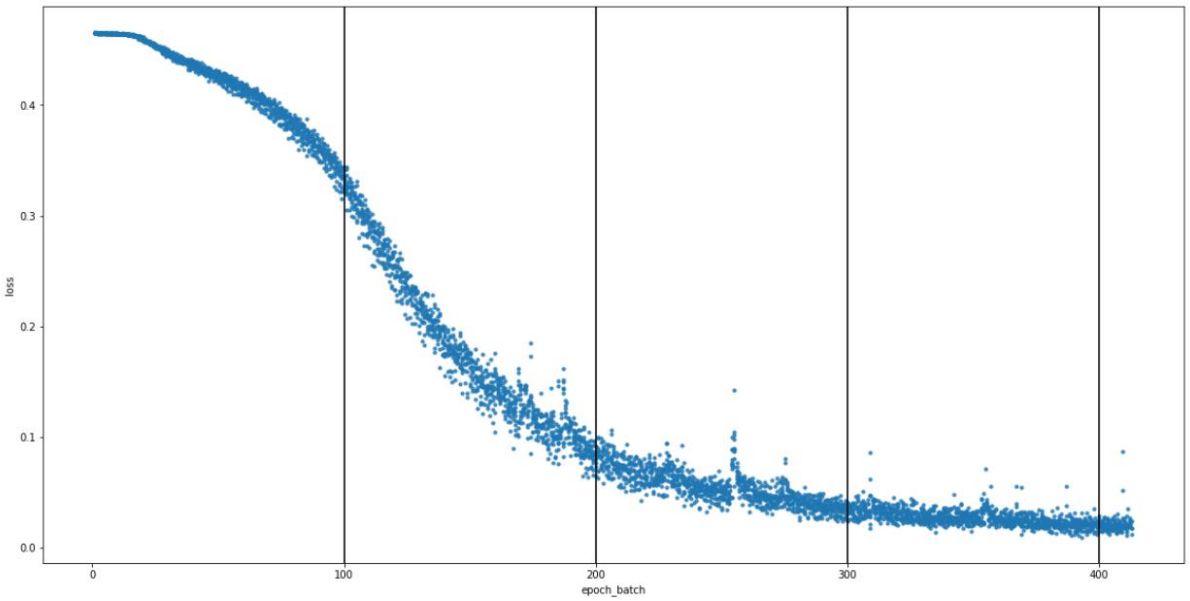
iii. 训练模型：

单机 Intel i5 4 核，learning_rate = 0.001, momentum = 0.9。迭代 412 次。总训练时长大约在 15 个小时左右。

b) 实验结论

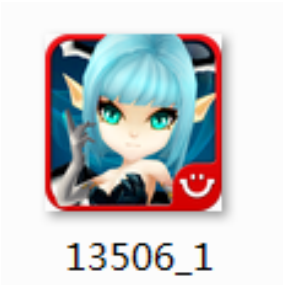
i. 训练数据效果检验

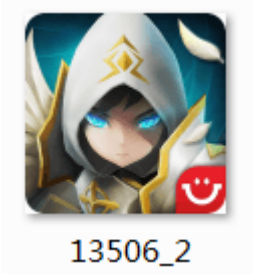
训练完成的模型在原数据上进行预测。完成全部图片的预测耗时 72 秒，准确率 93.05%。模型训练过程中 loss 的变化过程如下图：

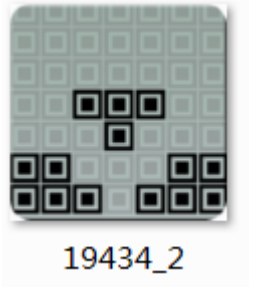


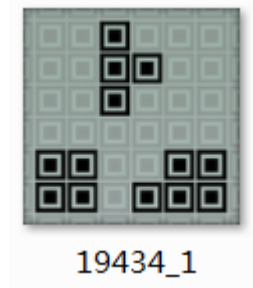
对于一些图标有所调整的应用，模型仍能正确识别

训练数据图像　　预测数据图像





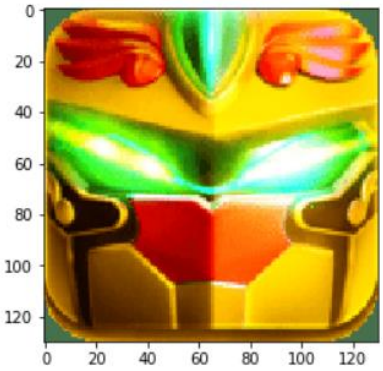




针对预测错误的样本进行观察。通过人的肉眼还是能明显看出有些图片并非同款应用的图标，但是从图片本身的色彩构成以及结构构成来说确实有一定相似度。例如下面的两个图片，模型认为左边的图片和右边的图片属于同一款应用的图标：

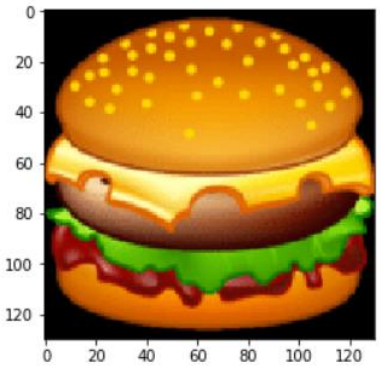
Original metaid: 12248

<matplotlib.image.AxesImage at 0x19c2e9b0>



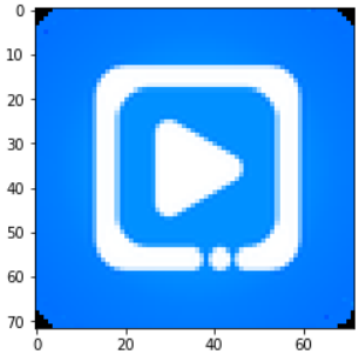
Predicted metaid: 120376

<matplotlib.image.AxesImage at 0x19e30e48>



Original metaid: 3628

<matplotlib.image.AxesImage at 0xebfa390>



Predicted metaid: 118289

<matplotlib.image.AxesImage at 0xed5f7b8>

也有一些是明显不像的图片，模型会认为他们很像。从这些图片应该可以推断出是模型提取特征的时候还不够完善

Original metaid: 120420

<matplotlib.image.AxesImage at 0xeccbda0>

Predicted metaid: 61429

<matplotlib.image.AxesImage at 0xeac6fd0>

除此之外，也有图标近似于一摸一样但是应属于不同 metaid 的应用

metaid	appName_orig	name_en	name_ch	appName_standard	packageName
255	球球大作战	NaN	qiuqiudazuozhan	NaN	com.game.battleball.kuwo
255	球球大作战	NaN	qiuqiudazuozhan	NaN	com.ztgame.bob
255	球球大作战	NaN	qiuqiudazuozhan	NaN	com.pdragon.qiuqiu
115992	球球大作战秘籍	NaN	qiuqiudazuozhanmiji	球球大作战秘籍	cc.koler.guide.qiuqiu

255_2

115990_1

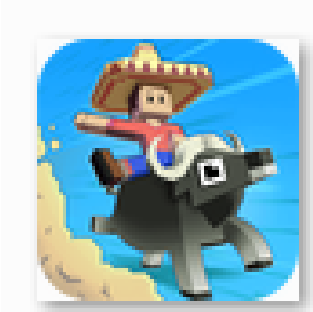
metaid	appName_orig	name_en	name_ch	appName_standard	packageName
825	美团外卖	NaN	meituanwaimai	美团外卖	com.sankuai.meituan.takeoutnew
825	美团外卖	NaN	meituanwaimai	NaN	com.sankuai.meituan.waimai.sunflower
6105	美团骑手	NaN	meituanqishou	美团骑手	com.sankuai.meituan.dispatch.homebrew

825_1

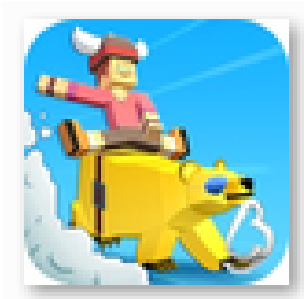
6105_1

同时，也有恶意抄袭 App 的现象，无论是从图标还是从包名观察都很相似

metaid	appName_orig	name_en	name_ch	appName_standard	packageName
101487	Stampede	Stampede	Stampede	Stampede	com.yodo1.rodeo.cmcc
115499	疯狂动物园	NaN	fengkuangdongwuyuan	疯狂动物园	com.yodo1.rodeo.mi
115499	疯狂动物园	NaN	fengkuangdongwuyuan	NaN	com.yodo1.rodeo.leshi
115499	疯狂动物园	NaN	fengkuangdongwuyuan	NaN	com.yodo1.rodeo.JINLI
120988	Rodeo Stampede	Rodeo Stampede	Rodeo Stampede	Rodeo Stampede	com.featherweightgames.stampede



115499_2



101487_1

120988_1

metaid	appName_orig	name_en	name_ch	appName_standard	packageName
292	迅雷	NaN	xunlei	迅雷	com.xunlei.downloadprovider
292	迅雷	NaN	xunlei	NaN	com.xunlei.downloadprovider.pad
120263	Traveloka	Traveloka	Traveloka	Traveloka	com.traveloka.android

292_2

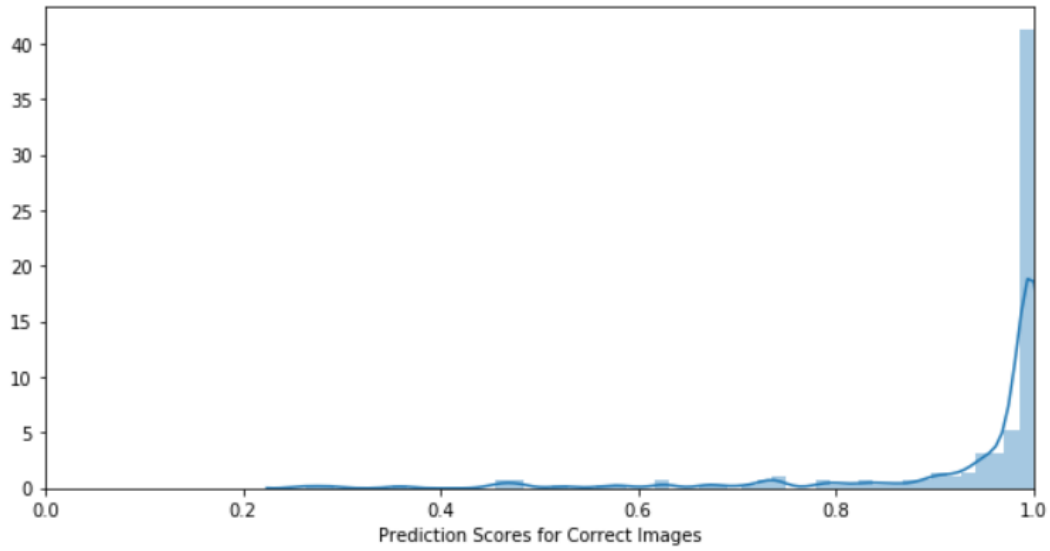
120263_1

ii. 新数据效果检验

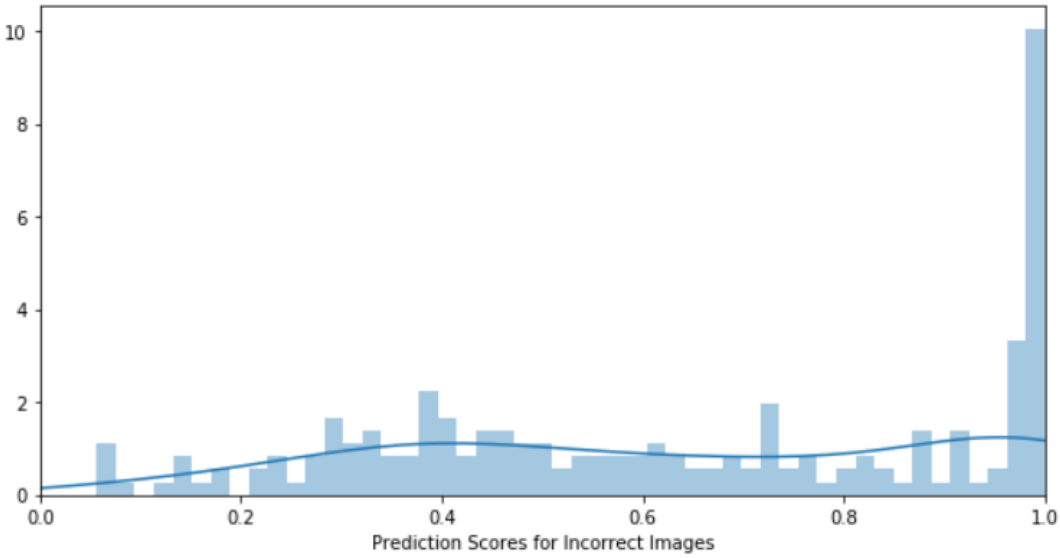
1. 已在训练数据中见过的 App 的新图标图片（有可能新图标和原来一致，也有可能新图标相较于老图标有所改动）

测试图片 384 张，正确识别 metaid 的准确率仅为 50%左右。

预测正确的图片所得的模型预测置信度分布如下图。可见预测正确的图片绝大部分的置信度数值在 0.9 以上。



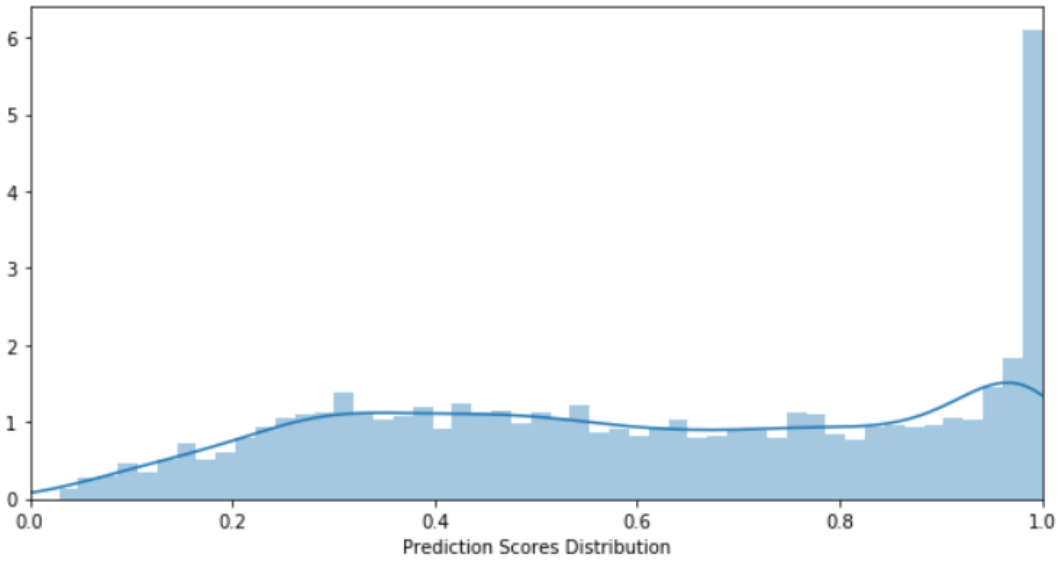
预测错误的图片所得的模型预测置信度分布如下图。



通过肉眼观察错误预测图片，发现有很多人工标注错误的图片（即图片完全相同，但是标注的 metaid 不同。模型本身并没有预测错）。详情参考 GitLab 上 label_error.docx 文件。

2. 在训练数据中未见过的 App 的图标

下图为模型针对每一个未见过的 App 的最大置信度分值（模型认为他与已经见过的 App 里最像的相似度程度）。可见如果以目前模型的水准，针对不少新 App，模型会错误的把那些 App 分到已有的 metaid 上。而不是告诉我们没有任何一个已知的 App 与其相似，应该打上新 metaid（最高置信度的分值也很低）。



8. 总结

基于目前的实验结果，卷积神经网络还是能很好的记住什么样的图标是哪款应用的。但是仍然有不足。

9. 优化建议

a) 增加卷积层的数量（4 层只是初步尝试，可以考虑增加层数提取更有效的特征）

b) 增加迭代次数（从 loss 的变化来看还可以继续收敛）

c) 考虑如何将图片合理的灰度化

i. 同款应用会有不同颜色的图标：QQ（蓝色和白色两种背景）

ii. 不同应用之间的唯一差异是颜色：男性助手，女性助手（蓝色和红色之分）

d) 对图片进行更多的转换（垂直旋转等）

e) 将设计方案中文字字段也加入模型预测中

f) 去除或者修正人工标签中出现错误的图片，详情参考 GitLab 上 label_error.docx 文件