

# Kaggle 猫狗大战实验报告

曹小虎

2020 年 2 月 16 日

## 目录

<b>1</b>	<b>定义</b>	<b>2</b>
1.1	项目概览	2
1.2	问题说明	2
1.3	二分类任务对数损失函数	2
<b>2</b>	<b>分析</b>	<b>3</b>
2.1	数据研究	3
2.2	训练集探索性可视化	3
2.3	训练集猫狗图片数量对比可视化	3
2.4	算法与方法	6
2.5	基准测试	8
<b>3</b>	<b>方法</b>	<b>8</b>
3.1	数据预处理一：排除异常图片	8
3.2	数据预处理二：准备目录结构	10
3.3	编译模型	10
3.4	测试	12
<b>4</b>	<b>结论</b>	<b>13</b>
4.1	思考	14
4.2	改进	14

摘要

摘要

# 1 定义

## 1.1 项目概览

本项目是 Kaggle 数据大赛的一个竞赛题目，解决一个计算机视觉领域的图片二分类问题。

## 1.2 问题说明

大赛网站上提供有训练集图片和测试集图片的下载链接。

我们需要下载两个数据文件包并解压缩到同一个工作目录下。在训练集的目录中一共包含 25000 张文件名包含了或猫或狗单词标记的图片，图片的内容与文件名称对应包含至少一只猫或者狗。

我们的目标是构建一个二分类的分类器。分类器接受一张图片作为输入，根据图片内容包含的特征输出判断结果：图片呢绒是猫或者是狗。

为了达成目标，我们以深度学习方法为工具，复用预训练模型的顶部分类器以下卷积层的结构和训练权重，连接自定义分类器；并且在训练图片分类器的过程中，尝试解冻预训练模型卷积层。

## 1.3 二分类任务对数损失函数

对数损失，即对数似然损失 (Log-likelihood Loss)，也称逻辑斯谛回归损失 (Logistic Loss) 或交叉熵损失 (cross-entropy Loss)，是在概率估计上定义的。它常用于 (multi-nominal, 多项) 逻辑斯谛回归和神经网络，以及一些期望极大算法的变体。可用于评估分类器的概率输出。

对数损失通过惩罚错误的分类，实现对分类器的准确度 (Accuracy) 的量化。最小化对数损失基本等价于最大化分类器的准确度。为了计算对数损失，分类器必须提供对输入的所属的每个类别的概率值，不只是最可能的类别。对数损失函数的计算公式如下：

$$L(Y, P(Y|X)) = -\log P(Y|X) = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^M y_{ij} \log(p_{ij})$$

其中，Y 为输出变量，X 为输入变量，L 为损失函数。N 为输入样本量，M 为可能的类别数， $y_{ij}$  是一个二值指标，表示类别 j 是否是输入实例  $x_i$  的真实类别。 $p_{ij}$  为模型或分类器预测输入实例  $x_i$  属于类别 j 的概率。

在处理二分类问题的时候, 则对数损失函数的公式简化为:

$$\text{LogLoss} = -\frac{1}{n} \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

在上面的表达式中,  $n$  为测试集中图片的数量,  $y_i$  为预测图中动物为狗的概率。如果实际图中的动物是狗, 则  $y_i=1$ , 否则  $y_i=0$ ;  $y_i$  为输入实例  $x_i$  的真实类别,  $\hat{y}_i$  为预测输入实例  $x_i$  属于类别 1 的概率。

## 2 分析

大赛的数据集下载链接是<https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/data>。在提交的毕业项目的 README.md 文档中, 我们采用了 Kaggle 官方提供的命令行工具下载数据集。

### 2.1 数据研究

在工作目录解压数据集 train.zip 之后, 我们会看到一个名称为 train 的目录, 里面一共有 25000 张 jpg 格式的彩色图片。

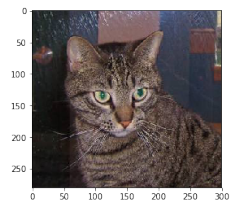
### 2.2 训练集探索性可视化

随机浏览其中几个训练图片, 我们可以发现如下几个特征:

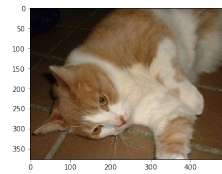
- 所有图片都是彩色图片
- 图片尺寸大小略有不同
- 每张图片的中间部位至少包含一只猫或者狗
- 图片背景所占比例一般在 50% 以下

### 2.3 训练集猫狗图片数量对比可视化

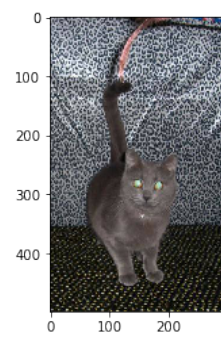
训练集中标签为猫和狗的图片数量分布对比图像如下:



Cat



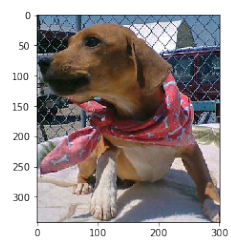
Cat



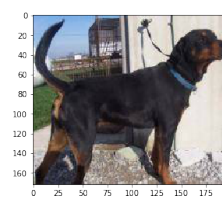
Cat



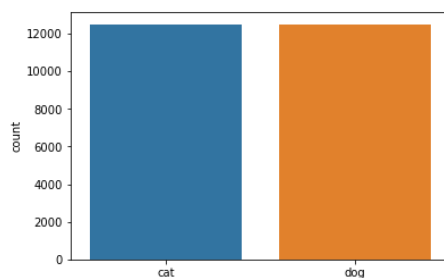
Dog



Dog



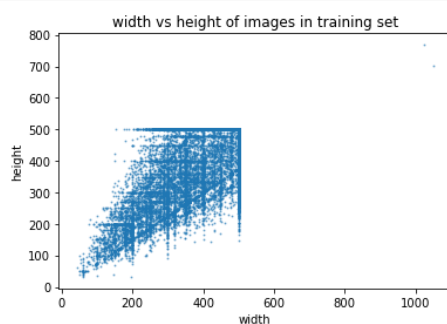
Dog



训练集中标签为全部图片宽高分布散点图如下:

```
In [80]: plt.title("width vs height of images in training set")
plt.xlabel("width")
plt.ylabel("height")

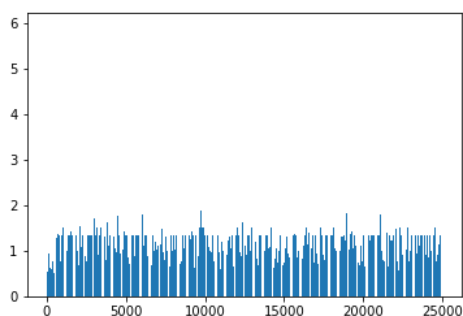
plt.scatter(width_data, height_data, s = 1, alpha = 0.5)
plt.show()
```



从上面图像右上角的两个点我们可以推断：训练集中存在少数几个宽接近 1000 像素，高接近 800 像素的超大尺寸图片。

训练集中全部图片宽高比例分布图如下：

```
In [100]: plt.bar(range(0,25000), ratio_data)
plt.show()
```



从上面图像可以看出，训练集图片的宽高比例全部在 2 以下，多数图片的这个比例分布在 0.5 1.5 之间。

## 2.4 算法与方法

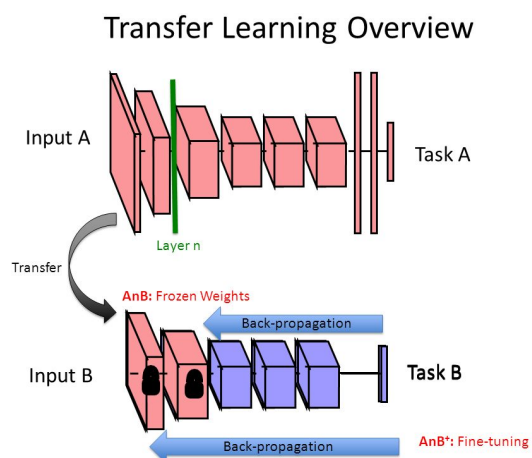
Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.715	0.901	138,357,544	23
VGG19	549 MB	0.727	0.910	143,667,240	26
ResNet50	99 MB	0.759	0.929	25,636,712	168
InceptionV3	92 MB	0.788	0.944	23,851,784	159
InceptionResNetV2	215 MB	0.804	0.953	55,873,736	572
MobileNet	17 MB	0.665	0.871	4,253,864	88

卷积神经网络已经能够在知名图片数据集 ImageNet 上非常出色的完成分类任务。这些预先训练的卷积神经网络的特定结构已经训练权重文件已经包含了如何从图片中提取不同图像特征，并且运用这些特征完成分类任务的知识与信息。

ImageNet 数据集中就包含了 108 种狗和 7 种猫的图片，与我们现有的训练集图片相似度极高。

因此我们希望能把这些知识和信息迁移应用到猫狗图片二分类任务当中，这也是迁移学习的基本原理。

我们选用在 Imagenet 数据集上表现最好的预训练模型，采用迁移学习的方法搭建执行猫狗图片分类任务的深度神经网络。



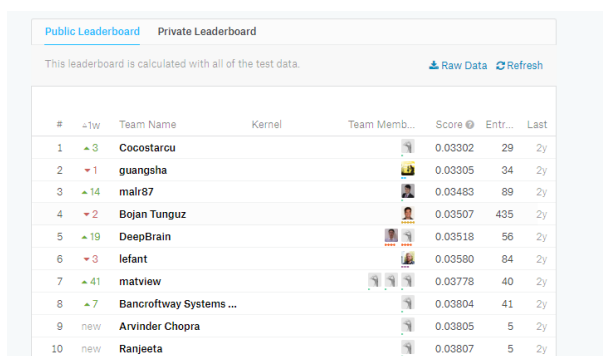
具体步骤如下：

- 引入在 Imagenet 数据集上表现最好的预训练模型，Inception-ResNet V2，但是不引入预训练模型顶部的分类器
- 复用预训练模型的输入层，需要对训练集图片进行适当缩放以适应预训练模型的输入要求
- 复用预训练模型的卷积层的结构和权重
- 连接自定义分类器：专用于完成本次猫狗二分类任务
- 自定义分类器中使用 Dropout 技术：训练过程中随机忽略前一层每个神经元到分类器的输入权重值
- 为分类器选择合适的优化器：探索优化器的种类和学习率参数，追求模型最佳表现
- 解冻预训练模型中的卷积层：重新训练这些层的权重值，追求模型最佳表现
- 使用图片增强技术：训练过程中，在拉伸、翻转、变形、扭曲训练集图片的基础上动态生成新训练实例，帮助模型对抗可能出现的过拟合

## 2.5 基准测试

Kaggle 的 LeaderBoard 上显示了之前很多大赛参赛者所取得的 LogLoss 比分，其中总共有 1314 个排名，我的计划是让自己的比分冲入前 5%，也就是大概是排名在第 65 名之前。

图 1: 明显错误的图片



#	Δ1w	Team Name	Kernel	Team Memb...	Score @	Entr...	Last
1	▲ 3	Cocostarcu			0.03302	29	2y
2	▼ 1	guangsha			0.03305	34	2y
3	▲ 14	malr87			0.03483	89	2y
4	▼ 2	Bojan Tunguz			0.03507	435	2y
5	▲ 19	DeepBrain			0.03518	56	2y
6	▼ 3	lefant			0.03580	84	2y
7	▲ 41	matview			0.03778	40	2y
8	▲ 7	Bancroftway Systems ...			0.03804	41	2y
9	new	Arvinder Chopra			0.03805	5	2y
10	new	Ranjeeta			0.03807	5	2y

## 3 方法

### 3.1 数据预处理一：排除异常图片

数据预处理包括训练开始之前的预处理和训练过程中的预处理，本小节主要讨论训练开始之前的图片预处理。

我们需要在训练集中检测是否存在非猫非狗，或者由于其他原因质量很差的图片，然后把他们从训练集中删除掉。

以下类型图片不应该作为输入数据参与模型训练，否则会对模型的训练效果造成负面影响。



图 2: 明显错误的图片

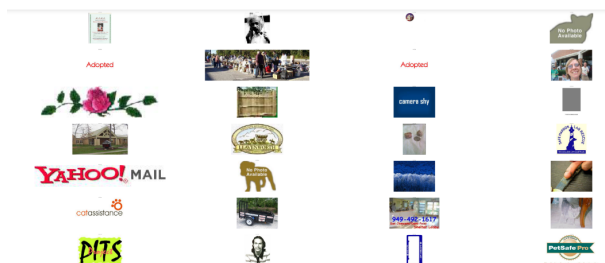
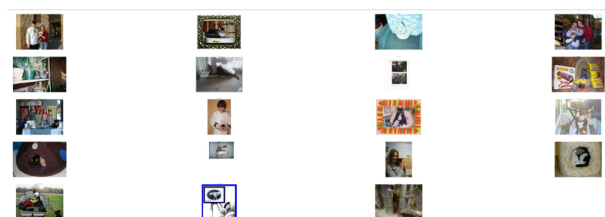


图 3: 猫或者狗的卡通图片



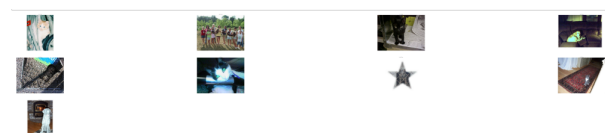
图 4: 背景复杂但仍包含正常单个猫或狗图片



背景复杂的猫或者狗的训练图片会有到模型过度关注图片背景中的图像特征，因而也会对模型训练效果发生负面影响。

此外，经过慎重考虑，我决定仍然把下面的图片保留在训练集中。这类图片的主要内容都包含猫或者狗，但是仍然被预处理过程使用的一组模型认为是错误图片。

图 5: 预处理过程中被误认为是质量较低的图片



### 3.2 数据预处理二：准备目录结构

使用命令行工具直接下载全部训练图片以后，解压压缩包，全部图片被包含在一个名称为 train 的文件目录中。

基于此，我们要新建一组目录结构：

图 6: 为模型训练输入图片准备的目录结构

```
mlnd-cat-vs-dog-cnn tree ./
./
├── train
│   ├── train1
│   │   ├── cat
│   │   └── dog
│   └── valid
│       ├── cat
│       └── dog
```

经过上一步图片筛选，train 目录下一共剩下 24944 张训练图片。

我从中挑选 2500 张猫和狗的图片移动到 valid/cat 和 valid/dog 目录下，留作模型训练工作过程中的验证集数据。

其余的猫狗图片分别移动到 train1/cat 和 train1/dog 目录下，模型会从这里读入训练数据。

### 3.3 编译模型

参照算法与方法部分描述组装并且编译模型。

在代码中引入了预先训练好的 InceptionResNetV2 模型。

图 7: 引入基准模型

```
batch_size = 64
target_size = (299, 299)

# 不包含原有模型的全连接层
base_model = InceptionResNetV2(include_top=False,
                               weights='imagenet',
                               input_shape=(299, 299, 3),
                               pooling='avg')

x = base_model.output

# x = Dense(units = 1024, activation = 'relu', name='fc1')(x)
x = Dropout(0.8, name='dropout')(x)

# Classifier
predictions = Dense(units = 1, activation='sigmoid', name='predictions')(x)

model = Model(inputs=base_model.input, outputs=predictions)

print("模型组装完毕!")
```

InceptionResNetV2 构造函数中传入的各参数表明 (按照从上到下的顺序):

- 不引入预训练模型顶部的分类器

- 使用模型在 ImageNet 大赛上的预训练权重
- 输入图片的长、宽和颜色纬度
- 为模型输出应用 global average pooling

在以上模型输出之后连接 Dropout 层。

在 Dropout 层后面连接名称为 predictions 的全连阶层。输出空间维度为 1，激活函数 sigmoid。

组装模型的最后一步是在变量 model 中保存用 Model 类型定义的最终模型。

此外，借助于可视化工具，我冻结了从输入方向看模型前 680 层的训练权重。这样在图片训练过程中，模型就会重用 ImageNet 的训练权重。在后面训练过程中，可以根据模型训练结果来调整。冻结的层数越多，复用 ImageNet 权重的层数越多，训练成本越低。有可能影响模型在猫狗图片训练集上提取图像特征，降低模型分类准确度；反之，冻结的层数越少，训练成本越高，可能导致模型在当前分类任务中表现出过拟合，与过拟合有关的就是较差的泛化预测能力。

训练模型的过程就是使用编译好的模型，从训练集图片目录加载图片拟合模型，从验证集目录加载图片验证模型训练结果。当模型在训练集上表现出过拟合，在验证集的验证误差开始上升之后立即停止训练，保存表现最优的模型权重文件。

图 8: 模型 Accuracy 度走势

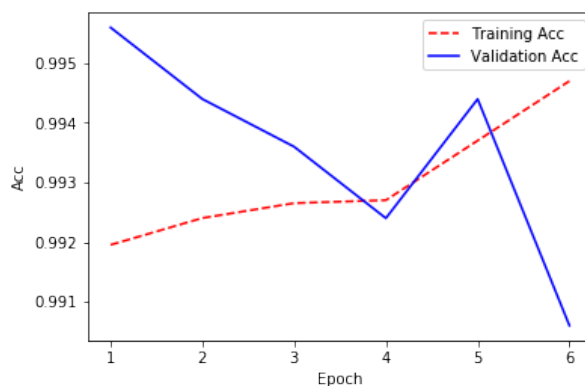
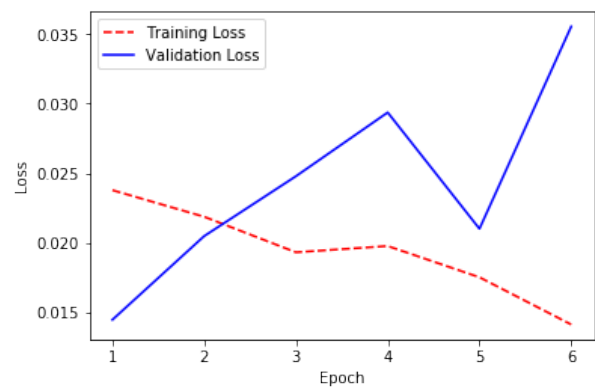


图 9: 模型 Loss



3.4 测试

训练过程保存了第五轮训练的模型权重。  
准备测试目录。  
加载模型权重文件，执行预测。

图 10: 用训练好的模型执行测试

```
In [34]: move_test_images()
print("开始导入测试集图片:")
test_datagen = ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_directory(test_folder,
                                                    target_size=target_size,
                                                    batch_size=batch_size,
                                                    shuffle = False, class_mode=None)

print("测试目录准备完成! ")

开始导入测试集图片:
Found 12500 images belonging to 1 classes.
测试目录准备完成!

加载模型结构和权重

In [35]: # 使用新的模型结构加载模型权重
json_file = open("best_model.json", "r")
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
print("loaded_model重新加载模型权重! ")
loaded_model.load_weights('final_model_weights.h5')
print("loaded_model重新加载模型权重完成! ")

loaded_model重新加载模型权重!
loaded_model重新加载模型权重完成!

In [36]: print("预测执行开始! ")
pred_result = loaded_model.predict_generator(test_generator, verbose=1)
print("预测执行完毕! ")

预测执行开始!
196/196 [=====] - 61s 313ms/step
预测执行完毕!
```

导出测试结果为 CSV 文件。

图 11: 预测结果导出为 csv 文件

导出预测结果

```
In [40]: import pandas as pd
from keras.preprocessing.image import *

solution = pd.read_csv("sample_submission.csv")

for i, fname in enumerate(test_generator_filenames):
    index = int(fname[fname.rfind('/')+1:fname.rfind('.')])
    solution.set_value(index-1, 'label', pred_result[i])

print("导出结果完成!")

导出结果完成!
/home/ec2-user/anaconda3/envs/tensorflow_p36/lib/python3.6/site-packages/
FutureWarning: set_value is deprecated and will be removed in a future re:
r .iat[] accessors instead

In [41]: solution.to_csv("pred-30.csv", index = False)
solution.head(10)

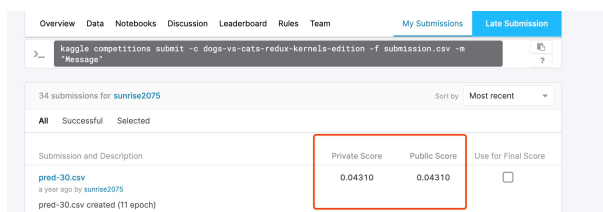
Out[41]:
```

	id	label
0	1	0.995
1	2	0.995
2	3	0.995
3	4	0.995
4	5	0.005
5	6	0.005
6	7	0.005
7	8	0.005
8	9	0.005
9	10	0.005

## 4 结论

上传预测结果的 csv 文件到 Kaggle, 得分如下:

图 12: 实际得分
















The screenshot shows the Kaggle competition submission page for 'sunrise2075'. It displays 34 submissions, sorted by 'Most recent'. A red box highlights the 'Private Score' and 'Public Score' for the submission 'pred-30.csv', both of which are 0.04310. The 'Use for Final Score' checkbox is unchecked.

Submission and Description	Private Score	Public Score	Use for Final Score
pred-30.csv a year ago by sunrise2075 pred-30.csv created (11 epoch)	0.04310	0.04310	<input type="checkbox"/>

根据 Public LiveBoard 的记录, 这分数可以排名 24。

图 13: 公开排名

14	—	HMen		0.03928	12	3y
15	—	a.ewais		0.03994	50	3y
16	—	yangpelwen		0.04008	27	3y
17	—	Anjith George		0.04077	46	3y
18	—	jilist12345678		0.04127	13	3y
19	—	NK255		0.04134	6	3y
20	—	nash		0.04183	5	3y
21	—	travall		0.04214	61	3y
22	—	Damodar		0.04280	24	3y
23	—	Bharat		0.04280	11	3y
24	—	Rongcheng Lin		0.04337	39	3y
25	—	Anish Shah		0.04354	24	3y
26	—	lystdo		0.04357	19	3y

4.1 思考

4.2 改进