机器学习纳⽶学位毕业项⽬ **--** 猫狗⼤战

苗沛

2018.09.15

# 问题的定义

## 项⽬概述

本项⽬来源于 kaggle 竞赛项⽬， 最终⽬的是训练⼀个机器学习模型， 输⼊⼀张图⽚来分辨图像中的猫和狗，也可是猫或狗的⾯部坐标甚⾄是⾝体的⼀部分。是⼀个典型的图像⼆分类问题。

本项⽬使⽤的卷积神经⽹络(Convolutional Neural Network, CNN)，卷积神经⽹络是深度学习技术中极具代表的

⽹络结构之⼀，在图像处理领域取得了很⼤的成功，在国际标准的 ImageNet 数据集上，许多成功的模型都是基于CNN 的。CNN 相较于传统的图像处理算法的优点之⼀在于，避免了对图像复杂的前期预处理过程（提取⼈⼯特征等），可以直接输⼊原始图像。CNN ⽹络对图⽚进⾏多次卷基层和池化层处理，在输出层给出两个节点并进⾏softmax 计算得到两个类别各⾃的概率。

本项⽬最终需要训练基于 CNN 的机器学习模型，对测试样本进⾏分类，并将最终结果上传 kaggle 进⾏最终评判。

本项⽬同时也实现了使⽤ Keras 和 Flask 搭建部署⼀个简单易⽤的深度学习图像⽹⻚应⽤，可以通过⽹⻚导⼊⼀张彩⾊猫或者狗的图⽚预测是猫或者狗的概率。

## 问题称述

数据集中⼤部分图⽚是正常的，有少部分异常图⽚和低分辨率图⽚，对于训练集来说这些异常数据是要剔除 掉的。

数据集中的⽂件名是以 type.num.jpg ⽅式命名的，⽐如 cat.0.jpg。使⽤ Keras 的 ImageDataGenerator 需要将不同种类的图⽚分在不同的⽂件夹中。

数据集中的图像⼤⼩是不固定的，但是神经⽹络输⼊节点的个数是固定的。所以在将图像的像素作为输⼊之 前，需要将图像的⼤⼩进⾏ resize。

## 评价指标

对数损失（Log loss）亦被称为逻辑回归损失（Logistic regression loss）或交叉熵损失（Cross-entropy loss）。交叉熵是常⽤的评价⽅式之⼀，它实际上刻画的是两个概率分布之间的距离，是分类问题中使⽤⼴泛的⼀种损失函 数。

本⽂实际上是⼆分类问题， 因此可以采⽤ logloss 损失函数作为评价指标， 计算公式如下：





其中：

n 是测试集中图⽚数量

 是图⽚预测为狗的概率

如果图像是狗，则为1，如果是猫，则为0 是⾃然（基数 ）对数

采⽤交叉熵作为损失函数可以有效的解决梯度消失和梯度爆炸的问题。

交叉熵损失越⼩，代表模型的性能越好。上述评估指标可⽤于评估该项⽬的解决⽅案以及基准模型。

# 分析

## 数据的探索

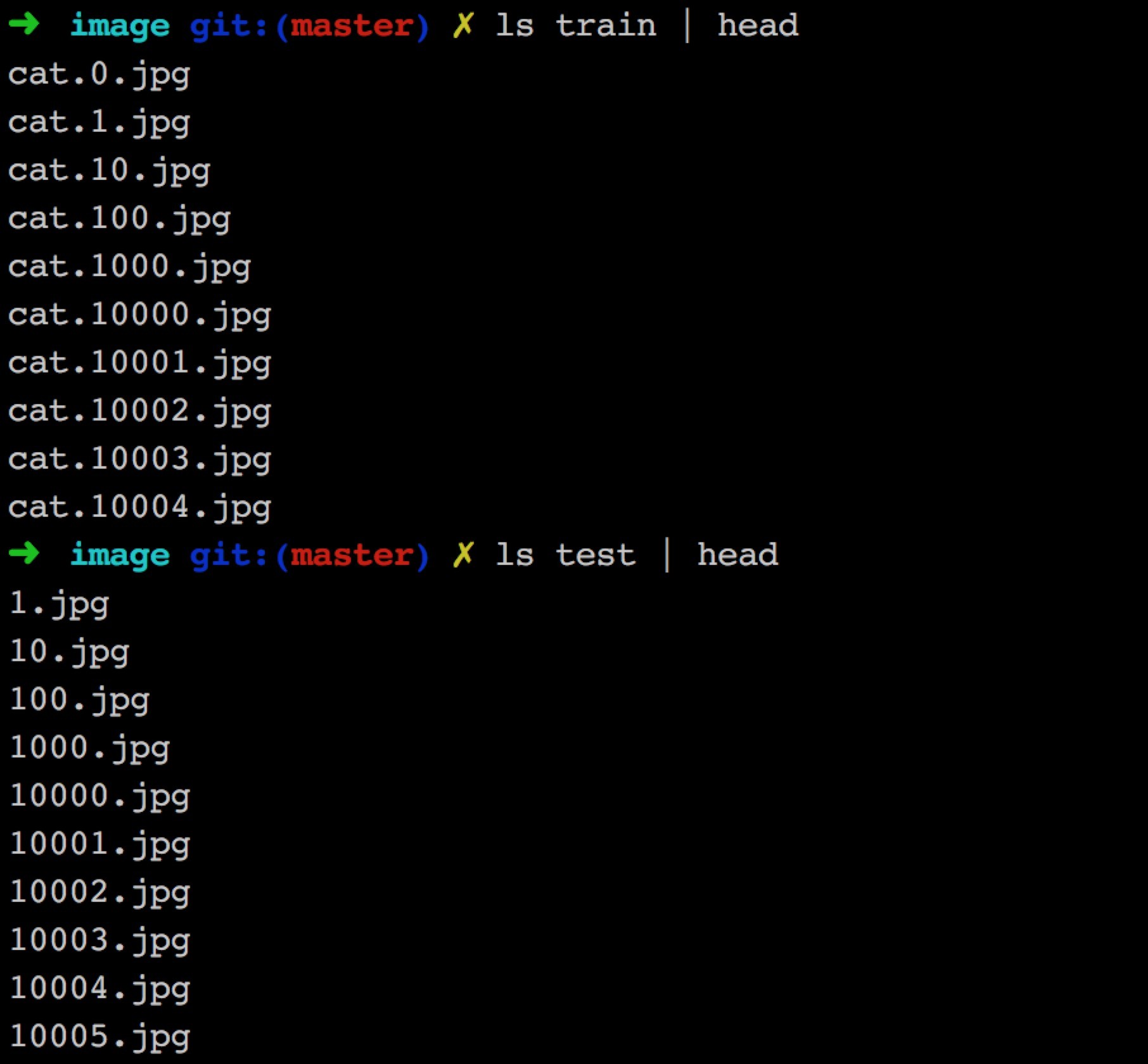
下载 kaggle 猫狗数据集解压后分为 3 个⽂件 train.zip、 test.zip 和 sample\_submission.csv。

train 训练集包含了 25000 张猫狗的图⽚，猫狗各⼀半，每张图⽚包含图⽚本⾝和图⽚名。命名规则根据

“type.num.jpg” ⽅式命名。

test 测试集包含了 12500 张猫狗的图⽚，没有标定是猫还是狗，每张图⽚命名规则根据 “num.jpg”，需要注意的是测试集编号从 1 开始，⽽训练集的编号从 0 开始。

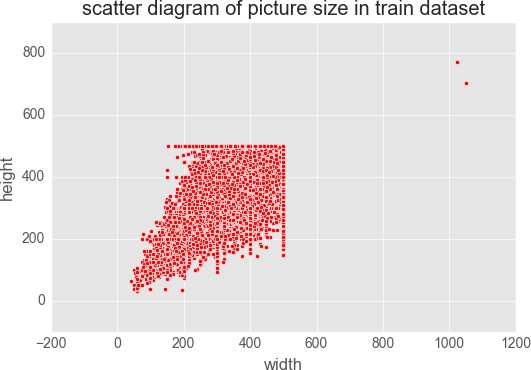
sample\_submission.csv 需要将最终测试集的测试结果写⼊.csv ⽂件中，上传⾄ kaggle 进⾏打分。



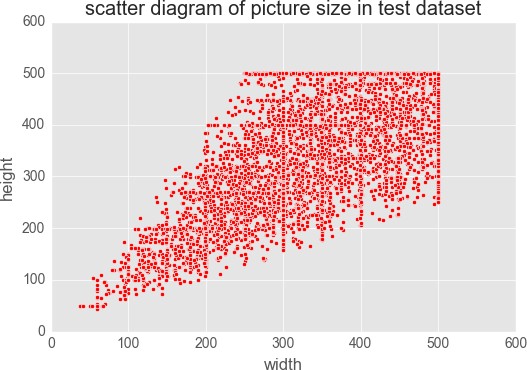
从训练集中随机提取图⽚可视化如下：



训练集中图⽚的尺⼨散点分布图:



测试集中图⽚的尺⼨散点分布图：



通过对图⽚中的⾊彩-像素⽐进⾏ IQR 分析，可以发现很多分辨率低、⽆关的图⽚，下⾯是其中⼀些不合格的图

⽚：



经过观察数据，数据集中⼤部分图⽚是正常的，有少部分异常图⽚，对于训练集来说这些异常数据是要剔除掉的。

## 算法和技术

### 问题分析

在给定⼀张图⽚，系统需要预测出图像属于预先定义类别中的哪⼀类。在计算机视觉领域，⽬前解决这类问题的核

⼼技术框架是深度学习（Deep Learning），特别地，针对图像类型的数据，是深度学习中的卷积神经⽹络

（Convolutional Neural Networks, ConvNets）架构。常⻅的卷积神经⽹络架构如下：



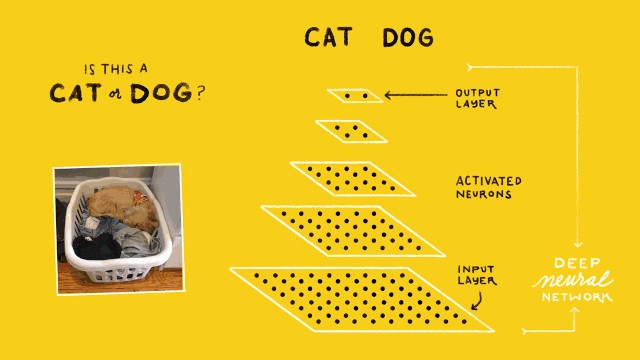
卷积神经⽹络中卷积层和池化层主要是对图⽚的⼏何特征进⾏抽取，⽐如浅层的卷积池化层可以抽取出⼀些直线，

⻆点等简单的抽象信息，深层的卷积池化层可以抽取⼈脸等复杂的抽象信息，最后的全连接层才是对图⽚分类的关

键处理。

因此可以利⽤已经训练好的卷积神经⽹络提取图⽚中复杂的⼏何特征，即将原始图⽚⽤已经训练好的卷积神经⽹络 处理之后的输出，作为新的输⼊，然后加上⾃⼰的全连接层，去进⾏分类。在模型训练的过程中，只改变新加的全 连接层的权重。

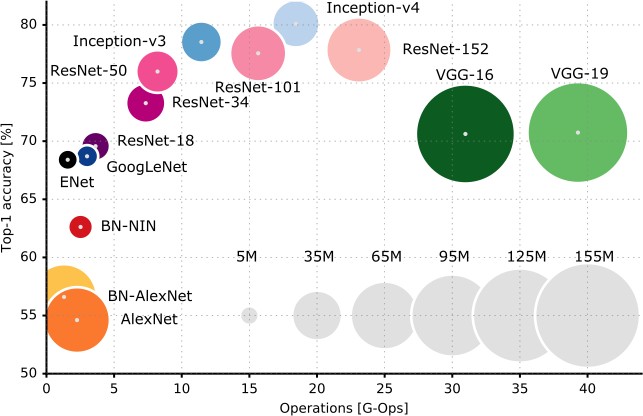
总的来说，卷积神经⽹络是⼀种特殊的神经⽹络结构，即通过卷积操作可以实现对图像特征的⾃动学习，选取那些 有⽤的视觉特征以最⼤化图像分类的准确率。



上图给出了⼀个简单的猫狗识别的卷积神经⽹络结构，在最底下（同时也是最⼤的）的点块表⽰的是⽹络的输⼊层

（Input Layer），通常这⼀层作⽤是读⼊图像作为⽹络的数据输⼊。在最上⾯的点块是⽹络的输出层（Output Layer），其作⽤是预测并输出读⼊图像的类别，在这⾥由于只需要区分猫和狗，因此输出层只有 2 个神经计算单元。⽽位于输⼊和输出层的，都称之为隐含层（Hidden Layer），图中有 3 个隐含层，图像分类的隐含层都是由卷积操作完成的，因此这样的隐含层也成为卷积层（Convolutional Layer）。因此，输⼊层、卷积层、输出层的结构及其对应的参数就构成了⼀个典型的卷积神经⽹络。

当然，在实际中使⽤的卷积神经⽹络要⽐这个⽰例的结构更加复杂，⾃ 2012 年的 ImageNet ⽐赛起，⼏乎每⼀年都会有新的⽹络结构诞⽣，已经被⼤家认可的常⻅⽹络有 AlexNet, VGG-Net, GoogLeNet, Inception V2-V4, ResNet 等等。这些卷积神经⽹络都是在 ImageNet 数据集上表现⾮常优异的神经⽹络，具体准确率和模型⼤⼩如下图所⽰。

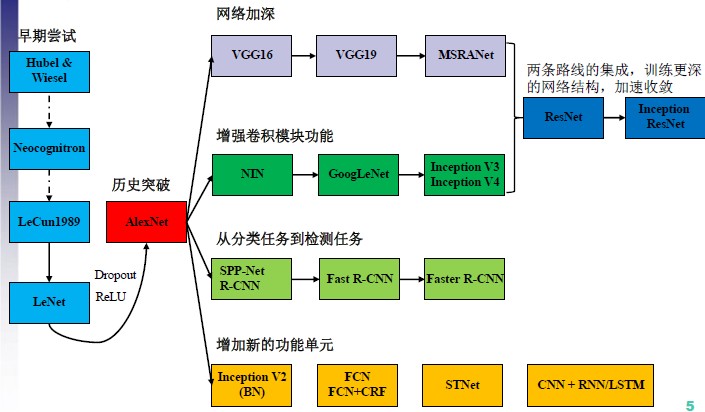


### 模型选择及技术

由于每⼀种神经⽹络提取的特征都不⼀样，因此本项⽬将多个神经⽹络处理的结果拼接，作为最后⼀层全连接层的 输⼊，这样做可以有效地降低⽅差。

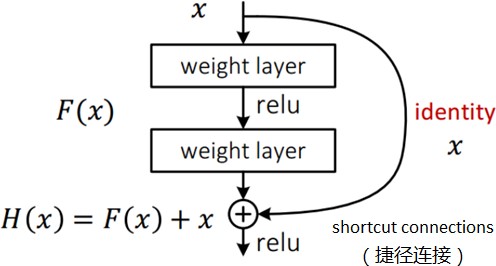
本项⽬迁移学习部分使⽤ Keras 实现，⽽ Keras 中可以导⼊的模型有 Xception，VGG16，VGG19，ResNet50， InceptionV3，InceptionResNet -V2，MobileNet. 综合考虑模型的分类准确率和⼤⼩，选⽤迁移学习的基础模型为 ResNet50，InceptionV3 和 Xception。

卷积神经⽹络结构演化图：



**ResNet**：

ResNet引⼊了残差⽹络结构（residual network），通过这种残差⽹络结构，可以把⽹络层弄的很深（据说⽬前可以达到 1000 多层），并且最终的分类效果也⾮常好，残差⽹络的基本结构如下图所⽰，很明显，该图是带有跳跃结构的：



残差⽹络借鉴了⾼速⽹络（Highway Network）的跨层链接思想，但对其进⾏改进（残差项原本是带权值的，但ResNet ⽤恒等映射代替之）。

假定某段神经⽹络的输⼊是 ，期望输出是  ，即  是期望的复杂潜在映射，如果是要学习这样的模型， 则训练难度会⽐较⼤；回想前⾯的假设，如果已经学习到较饱和的准确率（或者当发现下层的误差变⼤时），那么 接下来的学习⽬标就转变为恒等映射的学习，也就是使输⼊ 近似于输出 ，以保持在后⾯的层次中不会造成精度下降。

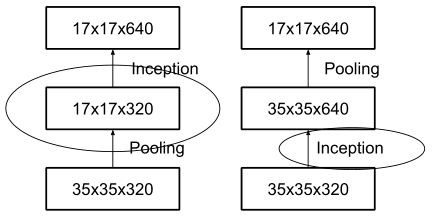
在上图的残差⽹络结构图中，通过“shortcut connections（捷径连接）”的⽅式，直接把输⼊ 传到输出作为初始结果，输出结果为 ，当  时，那么 ，也就是上⾯所提到的恒等映射。于是， ResNet 相当于将学习⽬标改变了，不再是学习⼀个完整的输出，⽽是⽬标值  和 的差值，也就是所谓的残差  ，因此，后⾯的训练⽬标就是要将残差结果逼近于 ，使到随着⽹络加深，准确率不下降。

这种残差跳跃式的结构，打破了传统的神经⽹络 层的输出只能给 层作为输⼊的惯例，使某⼀层的输出可以直接跨过⼏层作为后⾯某⼀层的输⼊，其意义在于为叠加多层⽹络⽽使得整个学习模型的错误率不降反升的难题 提供了新的⽅向。

在此之前，深度神经⽹络常常会有梯度消失问题的困扰，即来⾃误差函数的梯度信号会在反向传播回更早的层时指 数级地下降。本质上讲，在误差信号反向回到更早的层时，它们会变得⾮常⼩以⾄于⽹络⽆法学习。但是，因为ResNet 的梯度信号可以直接通过捷径连接回到更早的层，所以⼀下⼦就可以构建 50 层、101 层、152 层甚⾄1000 层以上的⽹络了，⽽且它们的表现依然良好。那时候，这在当时最佳的基础上实现了巨⼤的⻜跃——这个 22 层的⽹络赢得了 ILSVRC 2014 挑战赛。

**Inception V3**：

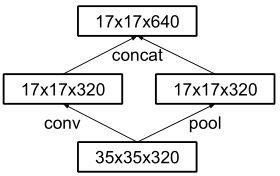
Inception 模块之间特征图的缩⼩，主要有下⾯两种⽅式：



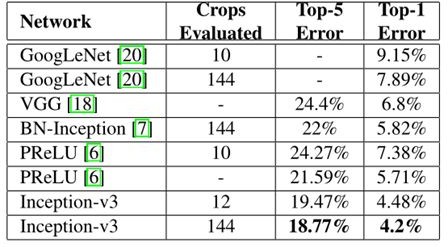
右图是先进⾏ Inception 操作，再进⾏池化来下采样，但是这样参数量明显多于左图(⽐较⽅式同前⽂的降维后Inception 模块)，因此 v2 采⽤的是左图的⽅式，即在不同的 Inception 之间（35/17/8 的梯度）采⽤池化来进⾏下采样。

但是，左图这种操作会造成表达瓶颈问题，也就是说特征图的⼤⼩不应该出现急剧的衰减(只经过⼀层就骤降)。如 果出现急剧缩减，将会丢失⼤量的信息，对模型的训练造成困难。

因此，在2015年12⽉提出的 Inception V3结构借鉴 Inception 的结构设计了采⽤⼀种并⾏的降维结构，如下图：



经过优化后的inception v3⽹络与其他⽹络识别误差率对⽐如表所⽰：

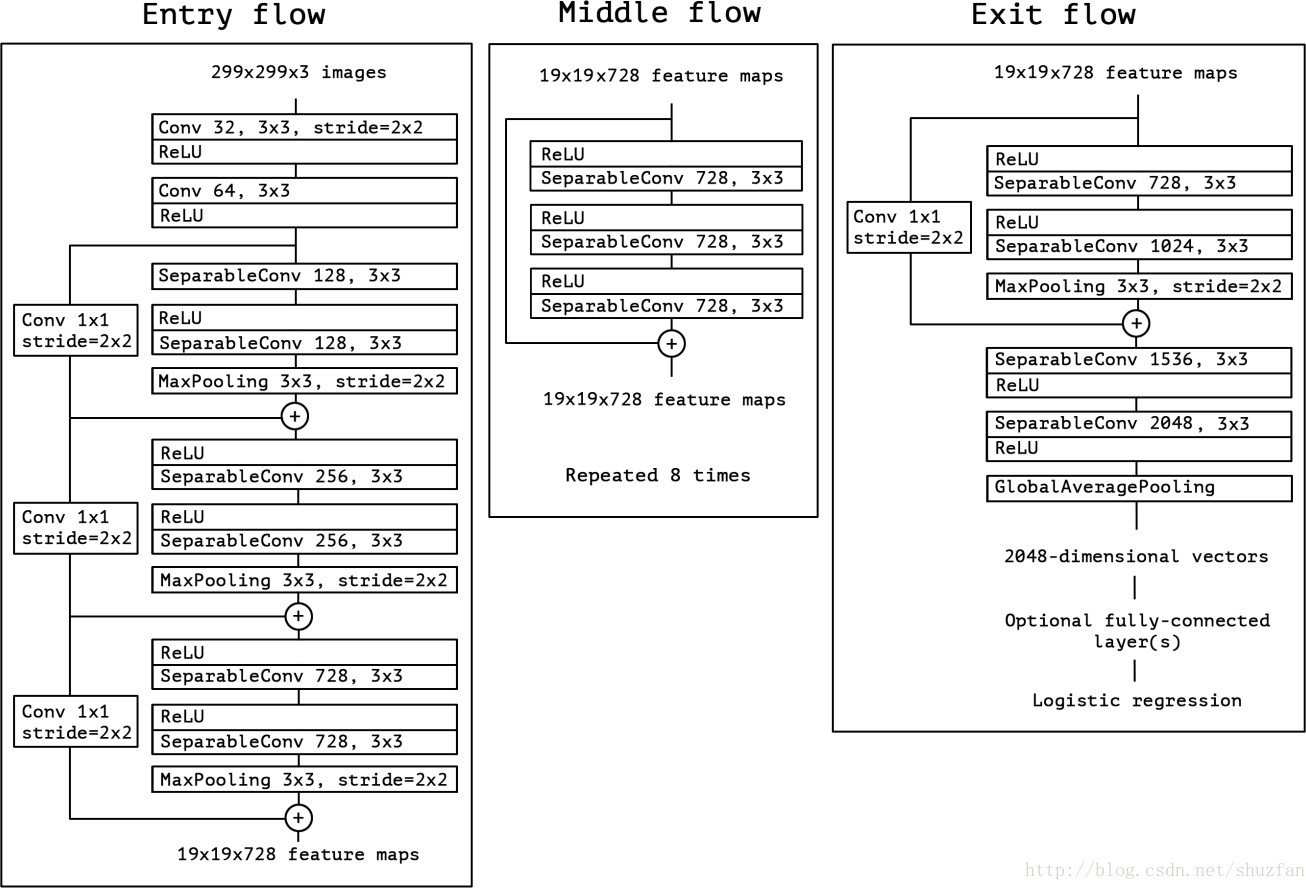


Inception V3 ⼀个最重要的改进是分解（Factorization），将 分解成两个⼀维的卷积（  ） ， 也是

⼀样（  ） ，这样的好处，既可以加速计算，⼜可以将 1 个卷积拆成 2 个卷积，使得⽹络深度进⼀步增加， 增加了⽹络的⾮线性（每增加⼀层都要进⾏ ReLU）。 另外，⽹络输⼊从 变为了 。

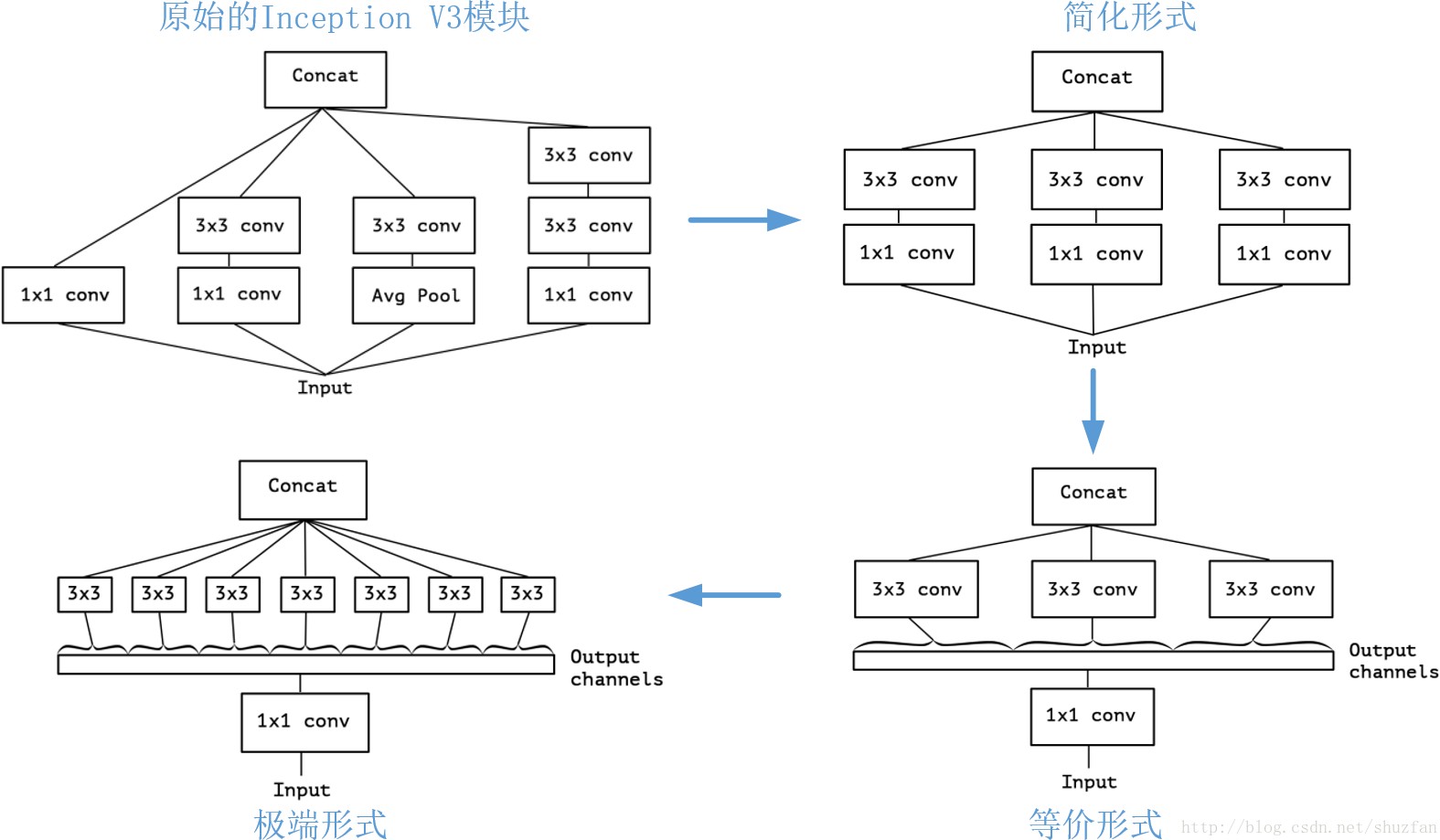
**Xception**：

Xception 实际上采⽤了类似于 ResNet 的⽹络结构，主体部分采⽤了模块化设计。如下图所⽰：



Xception 是 google 继 Inception 后提出的对 Inception v3 的另⼀种改进，主要是采⽤ depthwise separable convolution 来替换原来Inception v3中的卷积操作。

Xception 取名的由来是 "Extreme Inception"，Inception V3 的演进过程：



"极端形式"同 SeparableConv 的区别主要有两点：

卷积和 卷积的先后顺序。 原来的 Inception 结构是先 卷积，后 卷积。作者认为这⾥的区别并不重要。

两个卷积层之间是否有激活函数。 原来的 Inception 中间是有 ReLU 激活的。 但实验结果证明不加激活效果更好。

## 基准模型

本项⽬的最低要求是 kaggle Public Leaderboard 前 10%。在 kaggle 上，总共有 1314 只队伍参加了⽐赛，所以需要最终的结果排在 131 位之前，131 位的得分是 0.06127，所以⽬标是模型预测结果要⼩于 0.06127。

# ⽅法

## 数据预处理

对于异常数据的清理采⽤ [优达学习笔记](https://zhuanlan.zhihu.com/youdanote) 提供的“预处理模型”⽅法实现异常数据清洗。评价 ImageNet 有指标 Top-1 和 Top-5：

本项⽬使⽤ InceptionV3 top-10 训练train训练集。训练过程中将图⽚的名称和预测 top-10 的结果保存到字典⾥，

训练结束后保存字典为

train\_decode\_predictions.csv

常数据排查。具体实现代码参考

。将模型预测结果与

。

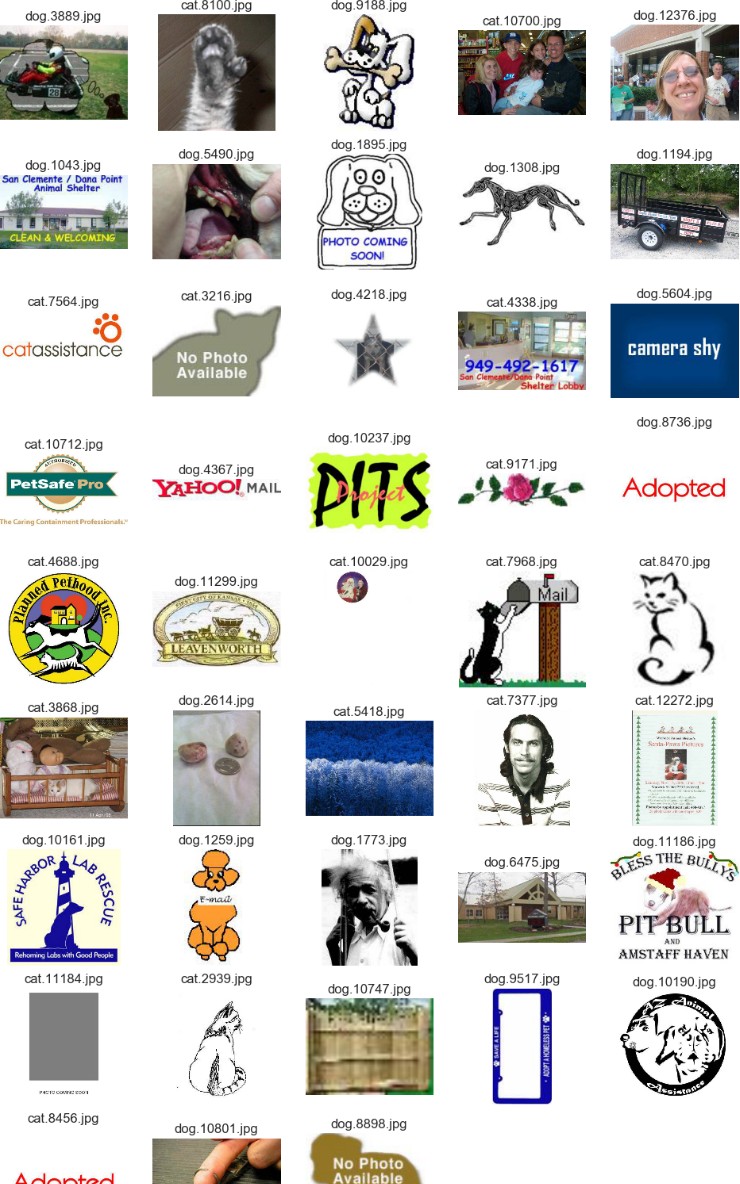
outlier\_detection.ipynb

进⾏异

使⽤ InceptionV3 模型排查出的异常图⽚总数为：131张，其中有些图⽚是正常的，经过筛选后选出 43 张异常图

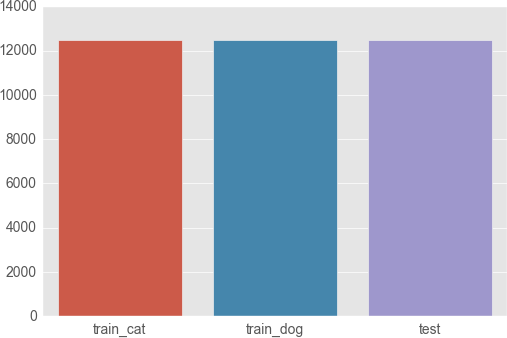
ImageNetClasses.csv

⽚作为本试验要清理的异常图⽚。这些异常图⽚如下图：





去除异常数据后数据分布如下：



数据集清洗后，猫的数量：12482，狗的数量：12475，测试集图⽚数量：12500。

由于我们的数据集的⽂件名是以 type.num.jpg 这样的⽅式命名的，⽐如 cat.0.jpg，但是使⽤ Keras 的

需要将不同种类的图⽚分在不同的⽂件夹中，因此我们需要对数据集进⾏预处理。这⾥我 们采取的思路是创建符号链接(symbol link)，这样的好处是不⽤复制⼀遍图⽚，占⽤不必要的空间。

ImageDataGenerator

⽂件⽬录结构如下：

image/

├── test

├── img\_test

│ ├── test -> ../test/

├── train

├── img\_train

│ ├── cat

│ └── dog

## 执⾏过程

### ⽣成迁移学习特征向量

Xception，InceptionV3 和 ResNet50 这三个模型对于输⼊数据都有各⾃的默认值，⽐如在输⼊图⽚⼤⼩维度上， Xception 和 InceptionV3 默认输⼊图⽚⼤⼩是 ，ResNet50 默认输⼊图⽚⼤⼩是 ；在输⼊数值维度上，Xception 和 InceptionV3 默认输⼊数值在 (-1, 1) 范围内。当要输⼊与默认图⽚⼤⼩不同的图⽚时，只需传⼊当前图⽚⼤⼩即可。ResNet50 需要对图⽚进⾏中⼼化处理，由于载⼊的 ResNet50 模型是在 ImageNet 数据上训练出来的，所以在预处理中每个像素点都要减去 ImageNet 均值。当要输⼊与默认图⽚⼤⼩不同的图⽚时，

只需传⼊当前图⽚⼤⼩即可。当输⼊数值不符合默认要求时，使⽤每个模型的预处理函数 即可将输⼊图⽚处理成该模型的标准输⼊。

preprocess\_input

常⻅的卷积神经⽹络结构在前⾯的若⼲层都是卷积池化层及其各种变种，后⾯⼏层都是全连接层，这些全连接层之 前的⽹络层被称为瓶颈层 (bottleneck). 将新的图⽚通过训练好的卷积神经⽹络直到瓶颈层的过程可以看做是对图像进⾏特征提取的过程。⼀般情况下，为了减少内存的消耗， 加快计算的过程，再将瓶颈层的结果输⼊全连接层之前，做⼀次全局平均 池化，⽐如 ResNet50 瓶颈层输出结果是 ，如果直接输⼊到全连接层，参数会⾮常多，所以进⾏⼀次全局平均池化，将输出矩阵调整为 ，这么做还有⼀个好处，那就是可以降低过拟 合的程度。

在 Keras 中载⼊模型并进⾏全局平均池化，只需要在载⼊模型的时候，设置 ,

include\_top=False

. 每个模型都将图⽚处理成⼀个 的⾏向量，将这三个⾏向量进⾏拼接，得到⼀个的⾏向量，作为数据预处理的结果。

pooling='avg'

def write\_gap(MODEL, image\_size, lambda\_func=None): width = image\_size[0]

height = image\_size[1]

input\_tensor = Input((height, width, 3)) x = input\_tensor

if lambda\_func:

x = Lambda(lambda\_func)(x)

base\_model = MODEL(input\_tensor=x, weights='imagenet', include\_top=False, pooling='avg')

model = Model(base\_model.input, GlobalAveragePooling2D()(base\_model.output))

gen = ImageDataGenerator()

train\_generator = gen.flow\_from\_directory("img\_train", image\_size, shuffle=False,

batch\_size=16)

test\_generator = gen.flow\_from\_directory("img\_test", image\_size, shuffle=False,

batch\_size=16, class\_mode=None)

train = model.predict\_generator(train\_generator, train\_generator.nb\_sample) test = model.predict\_generator(test\_generator, test\_generator.nb\_sample)

with h5py.File("gap\_%s.h5"%MODEL. name ) as h: h.create\_dataset("train", data=train) h.create\_dataset("test", data=test) h.create\_dataset("label", data=train\_generator.classes)

### 载⼊特征向量

经过上⾯的代码以后，我们获得了三个特征向量⽂件，分别是：

gap\_ResNet50.h5 gap\_InceptionV3.h5 gap\_Xception.h5

这⾥需要载⼊这些特征向量，并且将它们合成⼀条特征向量，然后记得把 X 和 y 打乱，不然之后设置的时候会出问题。这⾥设置了 numpy 的随机数种⼦为 2018。

validation\_split

np.random.seed(2018)

X\_train = [] X\_test = []

for filename in ["gap\_ResNet50.h5", "gap\_Xception.h5", "gap\_InceptionV3.h5"]: with h5py.File(filename, 'r') as h:

X\_train.append(np.array(h['train'])) X\_test.append(np.array(h['test'])) y\_train = np.array(h['label'])

X\_train = np.concatenate(X\_train, axis=1) X\_test = np.concatenate(X\_test, axis=1)

X\_train, y\_train = shuffle(X\_train, y\_train)

### 构建模型

载⼊预处理的数据之后，先进⾏⼀次概率为 0.5 的 dropout，然后直接连接输出层，激活函数为 Sigmoid，优化器为 Adadelta，输出⼀个零维张量，表⽰某张图⽚中有狗的概率。

input\_tensor = Input(X\_train.shape[1:]) x = input\_tensor

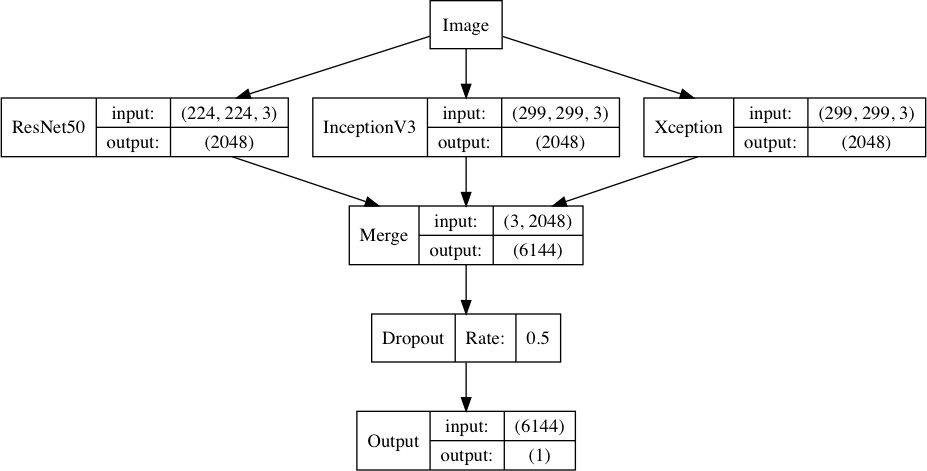
x = Dropout(0.5)(x)

x = Dense(1, activation='sigmoid')(x) model = Model(input\_tensor, x)

model.compile(optimizer='adadelta',

loss='binary\_crossentropy', metrics=['accuracy'])

整个迁移学习的神经⽹格结构如下所⽰:



### 训练模型

模型构件好了以后，我们就可以进⾏训练了，这⾥我们设置验证集⼤⼩为 20% ，也就是说训练集是 19964 张图，

验证集是 4991 张图。

model\_history = model.fit(X\_train,

y\_train, batch\_size=128, nb\_epoch=8, verbose=1, validation\_split=0.2,

callbacks = [TensorBoard(log\_dir='./Graph')])

model.save('model.h5')

保存模型为：model.h5

## 完善

本项⽬通过对图⽚中的⾊彩-像素⽐进⾏ IQR 分析，经⼈⼯筛选后剔除了分辨率低、⽆关的图⽚，这种⽅法并不能把所有的异常图⽚都剔除掉。⽐如还有⼀些图⽚被别的物体遮挡，对训练造成⼲扰等等。后期可以考虑 多⽅⾯对图⽚做筛选处理

图⽚中猫狗的拍摄⻆度不尽相同，⽽且猫狗占整张图⽚的⽐例也有所差别。为了让模型尽量不受这些因素的

⼲扰，增强模型的泛化能⼒，需要对原始图⽚进⾏⼀些随机操作，⽐如旋转、剪切变换、缩放、⽔平翻转

ImageDataGenerator

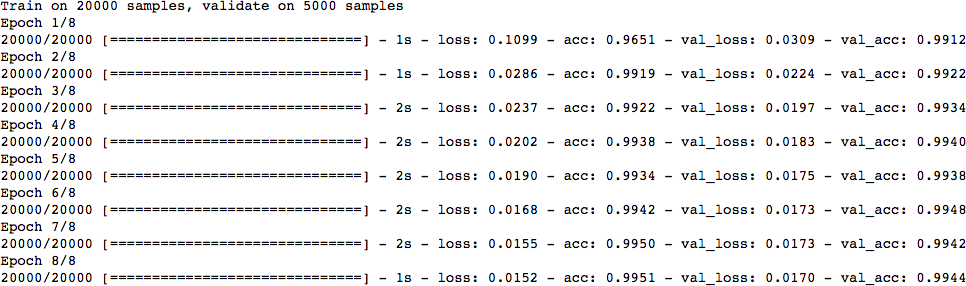
等。Keras 提供的图⽚⽣成器

可以很⽅便地对图⽚进⾏提升

对模型参数的优化，通过不同的优化器的尝试得到更好的优化结果

# 结果

## 模型的评价与验证

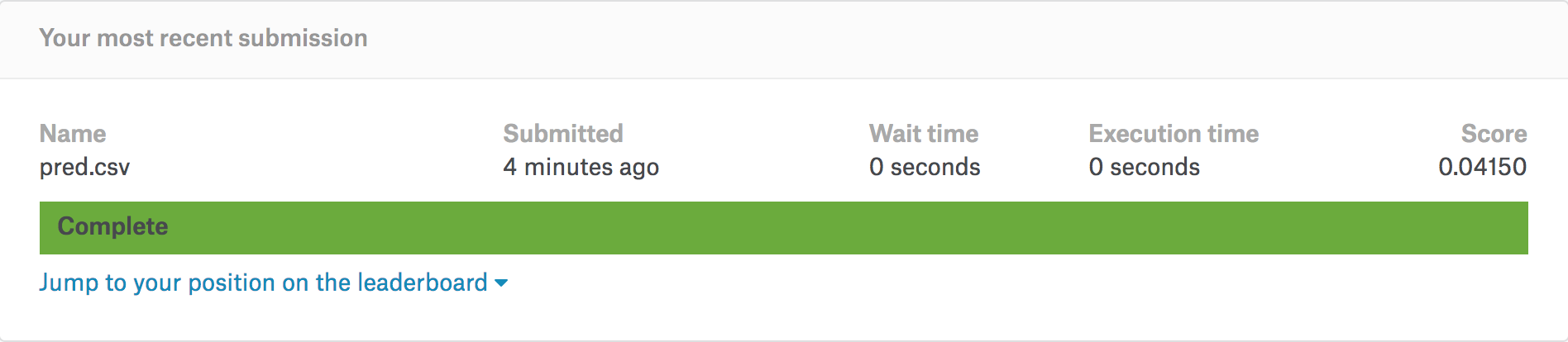


我们可以看到，训练的过程很快，⼗秒以内就能训练完，准确率也很⾼，在验证集上最⾼达到了99.4%的准确率， 这相当于⼀千张图只错了6张，可以说⽐我还厉害。

训练过中的 loss 和 accuracy 如下：



将测试集的处理结果提交到kaggle上，loss为0.04150，和验证集的loss近似。



## 合理性分析

单个 ResNet50 模型 5 次迭代训练结果：

训练集loss：0.0755，验证集loss：0.0419。

使⽤ Xception，InceptionV3 和 ResNet50 这三个模型进⾏迁移学习 5 次迭代训练结果： 训练集loss：0.0152，验证集loss：0.0170。

原始的 Xception 有 126 层，原始的 InceptionV3 有 159 层，原始的 ResNet50 有 168 层，更多的层数，不同的卷积核各种各样的的组合，可以更好的抽取图⽚中的泛化特征，这样既可以提⾼分类的准确率，⼜可以降低模型的 过拟合⻛险，所以现在在各种⽐赛中斩头露⻆的神经⽹络层数都⾮常的多，深度很深，这也是类似的神经⽹络被称 为深度学习的⼀个主要原因。

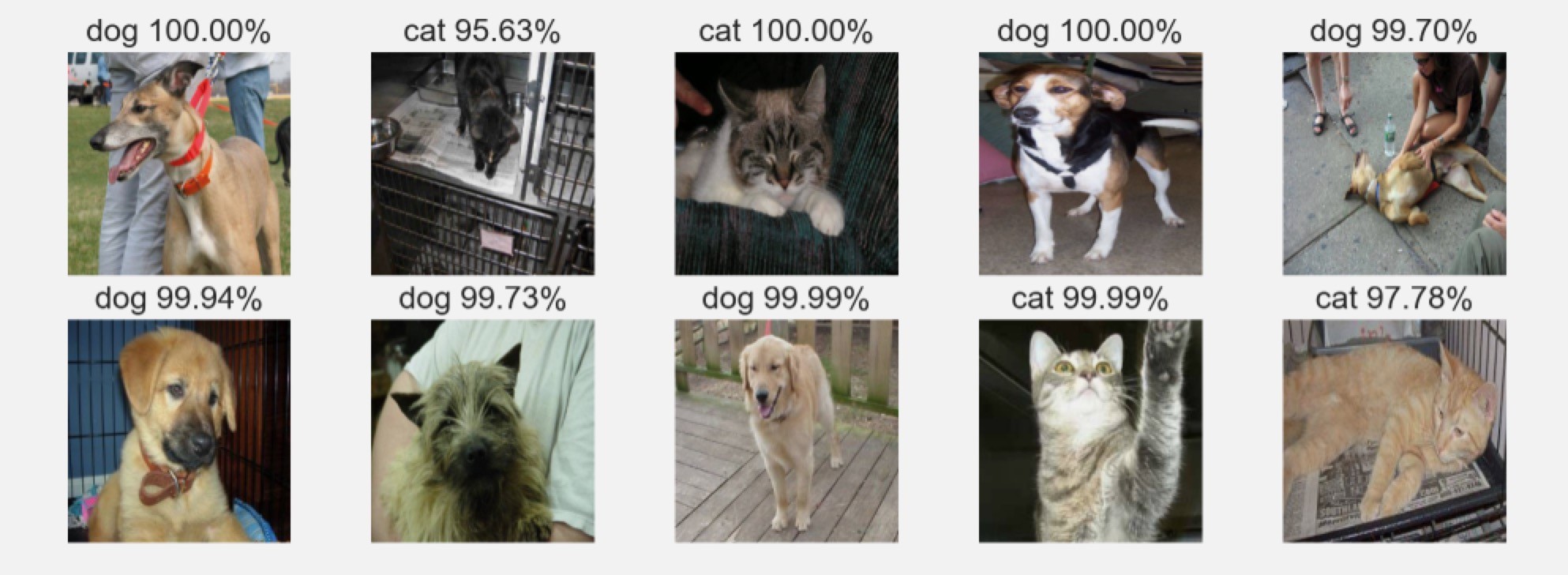
Xception，InceptionV3 和 ResNet50 这三个模型进⾏组合迁移学习，效果⽐先单个神经⽹络模型效果好。这⾥利

⽤了 bagging 的思想，通过多个模型处理数据并进⾏组合，可以有效降低模型的⽅差，减少过拟合程度，提⾼分类准确率。

# 项⽬结论

## 结果可视化

下图是使⽤优化之前的 ResNet50 模型进⾏预测的结果展⽰，可以看到都是正确的结果，只是精度达不到项⽬要求。使⽤了知乎上优达学城官⽅⽅法进⾏了实现，完全能达到项⽬的要求。



## 对项⽬的思考

深度学习毫⽆疑问是处理图像问题最佳的机器模型，近年来各⼤赛的前⼏名均是通过深度学习获取了前⼏名的好成 绩。但是相⽐于传统的机器学习模型，深度学习需要更多的数据，更强⼤的算⼒和资源。本项⽬在训练迁移学习模 型使⽤的是⾃⼰的 Mac，⾜⾜跑了三天训练完，泪崩。建议在使⽤的时候⽤云计算平台去训练。

kaggle 上猫狗⼤战前⼏名的 loss 达到了 0.0330，相⽐于本⽂中的 0.0415，绝对值减少了 0.0085，说明还是有较

⼤的改进空间。本⽂只是使⽤的 Xception，InceptionV3 和 ResNet50 这三个模型进⾏了提取特征向量，然后将特征向量直接拼接，忽略了特征之间的位置关系。除了这三个模型，还可以增加更多新的模型，或者使⽤stacking的

⽅法进⾏模型融合，进⼀步降低⽅差，提⾼分类的准确率。还可以从原始样本上⼊⼿，有些图⽚的分类概率正确， 但是不够确定，可以先做⼀部分处理，然后让模型更加确定。

## 需要作出的改进

相⽐ Keras，TensorFlow 真的真的太⿇烦了，但是 Google 为什么要把 TensorFlow 做的这么⿇烦呢？个⼈认为是为了运⾏的⾼效率和极⼤的灵活性做出了让步。TensorFlow 是经过⼯业界实际产品考验过的框架，⽣态及其丰

富，想实现⼀个功能，有多种写法，学起来有⼀种 C++ 的感觉，Keras 更像是为了快速出活的框架，如果想做⼀些改动⼗分的⿇烦，Keras 的默认 backend 就是 TensorFlow，所以 Keras 的执⾏效率是⽐ TensorFlow 慢很多的。

tf.contrib.keras

tf.keras

TensorFlow1.4 ⾥⾯已经把

更改为核⼼模块

，所以 TensorFlow 以后⽤起来应该

也会⽅便很多。最后还想说的是 PyTorch，好多⼈都推荐，说不仅仅有 TensorFlow 的⾼效率，⽽且很 pythonic， 可以在任意层和 numpy 数组进⾏转化。

# 参考⽂献

1. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015.
2. Christian Szegedy, Wei Liu, Yangqing Jia. Going Deeper with Convolutions arXiv:1409.4842, 2014
3. François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions arXiv:1610.02357, 2016

[4][⼿把⼿教你如何在Kaggle猫狗⼤战冲到Top2%](<https://zhuanlan.zhihu.com/p/25978105>)

1. Karen Simonyan and Andrew Zisserman. VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE- SCALE IMAGE RECOGNITION. At ICLR,2015.
2. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.

[[7][Building powerful image classiﬁcation models using very little data.](https://blog.keras.io/building-power ful-image-classiﬁcation-models-using-very-little-data.html)](https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html)

[[8][Dogs vs. Cats: Image Classiﬁcation with Deep Learning using TensorFlow in Python.]( https://www.datasci encecentral.com/proﬁles/blogs/dogs-vs-cats-image-classiﬁcation-with-deep-learning-using)](https://www.datasciencecentral.com/profiles/blogs/dogs-vs-cats-image-classification-with-deep-learning-using)

[[9][ImageNet: VGGNet, ResNet, Inception, and Xception with Keras.](https://www.pyimagesearch.com/2017/ 03/20/imagenet-vggnet-resnet-inception-xception-keras/)](https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/)

1. The residual module in ResNet as originally proposed by He et al. in 2015.
2. An Analysis of Deep Neural Network Models for Practical Applications. arXiv:1605.07678, 2017.