Dogs vs. Cats Redux: Kernels Edition

刘 诚

（一）项目概述

本项目是为完成 Kaggle 上的图片分类任务 Dogs vs. Cats Redux: Kernels Edition 而作，该任务属于计算机视觉 （Computer Vision）领域的问题。

计算机视觉（简称 CV）是人工智能领域的重要核心技术之一，是研究如何使人工系统从图像或多维数据中提取信息的科学。伴随着技术成熟度的提高，人脸识别、物体识别等分类、分割算法精度的不断提升，计算机视觉技术广泛应用于安防、金融、互联网、物流、零售、医疗、制造业、农业等领域。

其中深度学习技术中的卷积神经网络（CNN）在计算机视觉领域取得了很大的成功，在 ImageNe 数据集上，许多成功的模型都是基于 CNN 的。CNN 网络对图片进行多次卷积和池化操作，提取图片特征，可以以此来判断图片样本类别。

本作是深度学习技术的一个简单应用，以学习、研究、探讨 CV 技术为目的而展开，正文将具体介绍数据处理、模型搭建、算法、调参等详细过程。

实验过程使用了带 GPU 的云服务计算，可大幅提高训练速度。

（二）项目介绍

1.问题分析

Dogs vs. Cats Redux: Kernels Edition 是 kaggle 上的一个竞赛项目，该项目向参与者提供训练集和测试集，要求完成测试集分类。其中训练集为 25000 张带标签的图片，测试集为 12500 张带 id 的无标签图片。题目具体要求如下：

Distinguish images of dogs from cats.

For each image in the test set, predict a probability that the image is a dog (1 = dog, 0 = cat).

预测图片是狗的概率，即区分测试集中的猫和狗。可以确定该题是一个二分类问题，测试集中的图片一定是猫或者狗。

CNN 网络非常适合解决图片识别类问题，当前主流的 CNN 模型包括 VGG、ResNet、DenseNet、Inception 等系列。

主流 CNN 模型 Top-1 error 比较

Network	Top-1 error	Top-5 error
AlexNet	43.45	20.91
VGG-11	30.98	11.37
VGG-13	30.07	10.75
VGG-16	28.41	9.62
VGG-19	27.62	9.12
VGG-11 with batch normalization	29.62	10.19
VGG-13 with batch normalization	28.45	9.63
VGG-16 with batch normalization	26.63	8.50
VGG-19 with batch normalization	25.76	8.15
ResNet-18	30.24	10.92
ResNet-34	26.70	8.58
ResNet-50	23.85	7.13
ResNet-101	22.63	6.44
ResNet-152	21.69	5.94
SqueezeNet 1.0	41.90	19.58
SqueezeNet 1.1	41.81	19.38
Densenet-121	25.35	7.83
Densenet-169	24.00	7.00
Densenet-201	22.80	6.43
Densenet-161	22.35	6.20
Inception v3	22.55	6.44
GoogleNet	30.22	10.47
ShuffleNet V2	30.64	11.68
MobileNet V2	28.12	9.71
ResNeXt-50-32x4d	22.38	6.30

Network	Top-1 error	Top-5 error
ResNeXt-101-32x8d	20.69	5.47
Wide ResNet-50-2	21.49	5.91
Wide ResNet-101-2	21.16	5.72
MNASNet 1.0	26.49	8.456

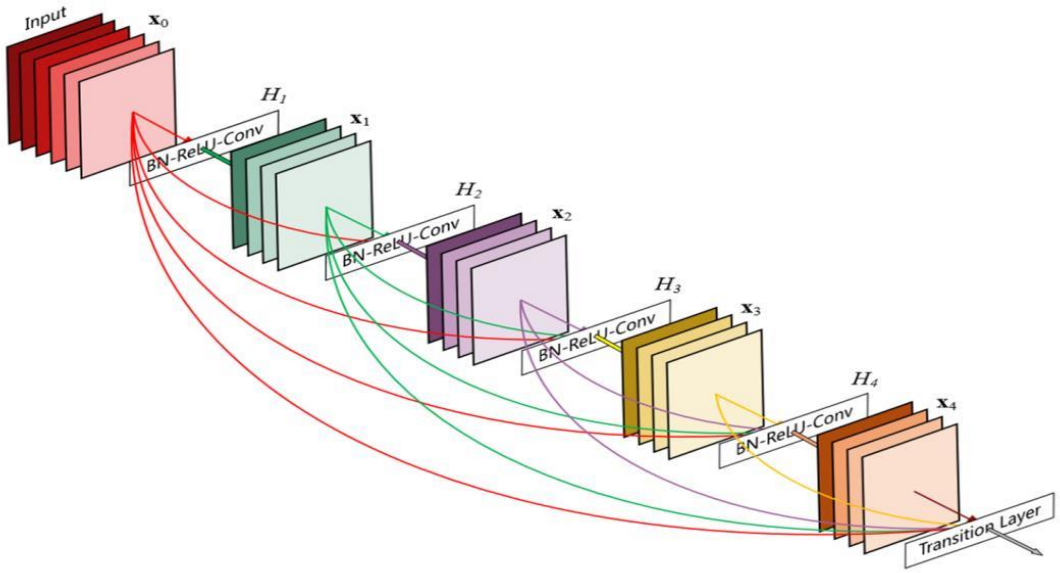
Top-1 error =（正确标记 与 模型输出的最佳标记不同的样本数）/ 总样本数

Top-1 error 越小，模型的正确率越高。从上述列表可以看到 Densenet-161 的 Top-1 error 仅 22.35，相对其他 CNN 结构有着更优秀的表现，它还有另外一个优势就是综合计算量相对其他模型要小得多。因此我确定 Densenet-161 作为本项目基准模型。

2.模型的结构和组成

densenet-161 属于 DenseNet 系列中的一个版本。

在介绍 DenseNet 之前不得不先讲一下 ResNet，ResNet 模型的核心是通过建立前面层与后面层之间的“短路连接”（shortcuts，skip connection），有助于训练过程中梯度的反向传播，从而能训练出更深的 CNN 网络。DenseNet 模型的基本思路与 ResNet 一致，但是它建立的是前面所有层与后面层的密集连接（dense connection），它的名称也是由此而来。DenseNet 的另一大特色是通过特征在 channel 上的连接来实现特征重用（feature reuse）。这些特点让 DenseNet 在参数和计算成本更少的情形下实现比 ResNet 更优的性能。



相比 ResNet，DenseNet 提出了一个更激进的密集连接机制：即互相连接所有的层，具体来说就是每个层都会接受其前面所有层作为其额外的输入。下图 1 为 ResNet 网络的连接机制，作为对比，图 2 为 DenseNet 的密集连接机制。可以看到，ResNet 是每个层与前面的某层（一般是 2~3 层）短路连接在一起，连接方式是通过元素级相加。而在 DenseNet 中，每个层都会与前面所有层在 channel 维度上连接（concat）在一起，并作为下一层的输入。对于一个 L 层的网络，DenseNet 共包含 L(L+1)/2 个连接，相比 ResNet，这是一种密集连接。

而且 DenseNet 是直接 concat 来自不同层的特征图，这可以实现特征重用，提升效率。

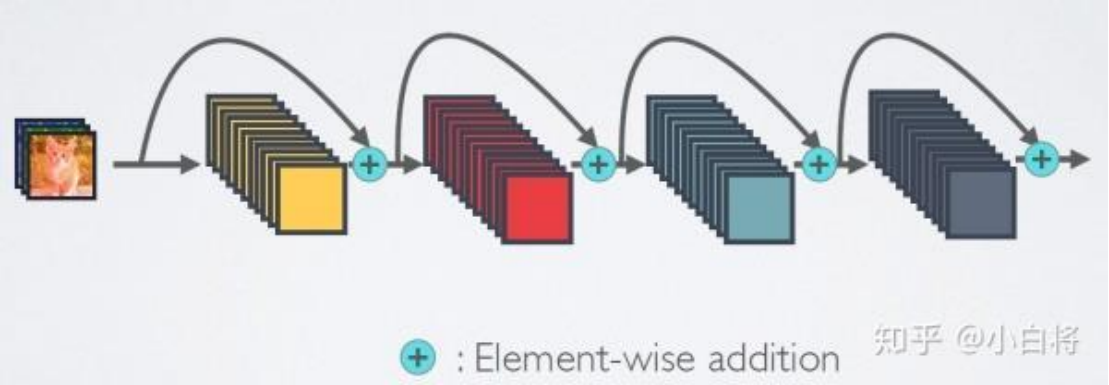


图 1 ResNet 网络的短路连接机制（其中+代表的是元素级相加操作）

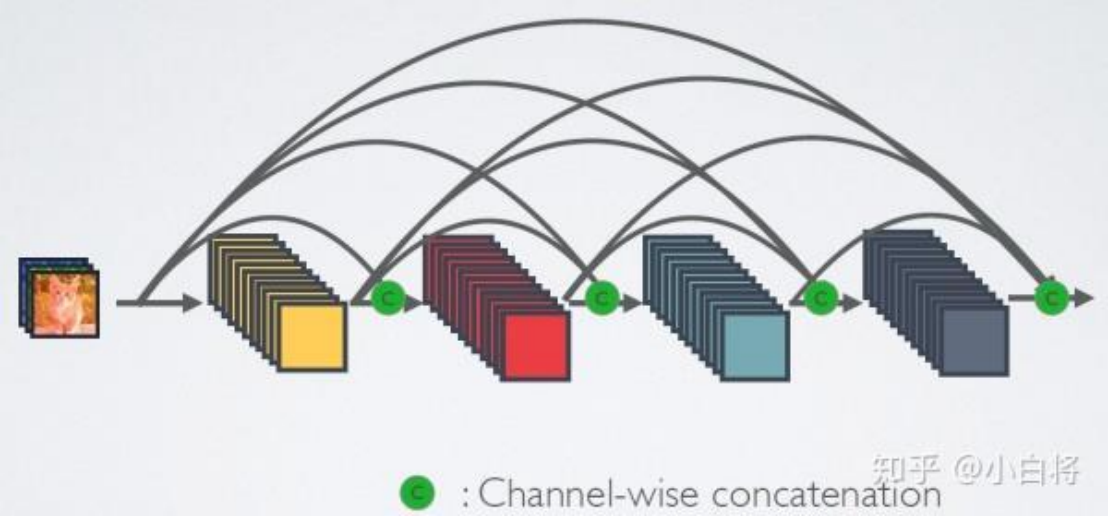
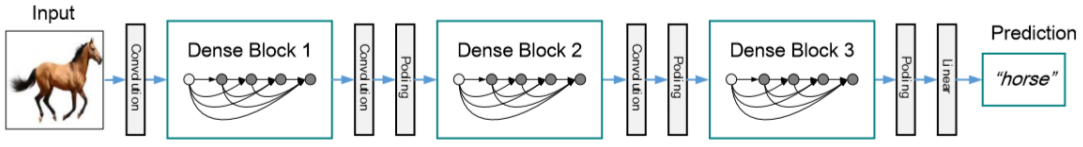


图 2 DenseNet 网络的密集连接机制（其中 c 代表的是 channel 级连接操作）

DenseNet 在 L 层的输出，会连接前面所有层作为输入：

$$\mathbf{x}_\ell = H_\ell([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}]),$$

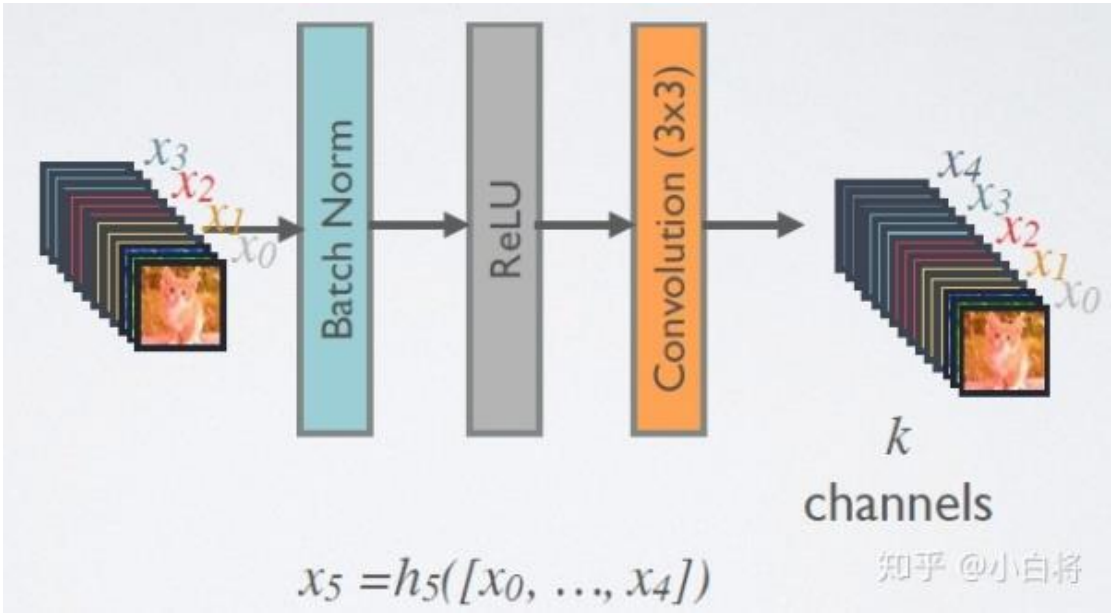
H 代表非线性转化函数（non-liear transformation），
其可能包括一系列的 BN(Batch Normalization)，ReLU，Pooling 及 Conv 操作。



DenseNet 网络结构

CNN 网络一般要经过 Pooling 或者 stride > 1 的 Conv 来降低特征图的大小，而 DenseNet 的密集连接方式需要特征图大小保持一致。为了解决这个问题，DenseNet 网络中使用 DenseBlock+Transition 的结构，其中 DenseBlock

是包含很多层的模块，每个层的特征图大小相同，层与层之间采用密集连接方式。而 Transition 模块是连接两个相邻的 DenseBlock，并且通过 Pooling 使特征图大小降低。

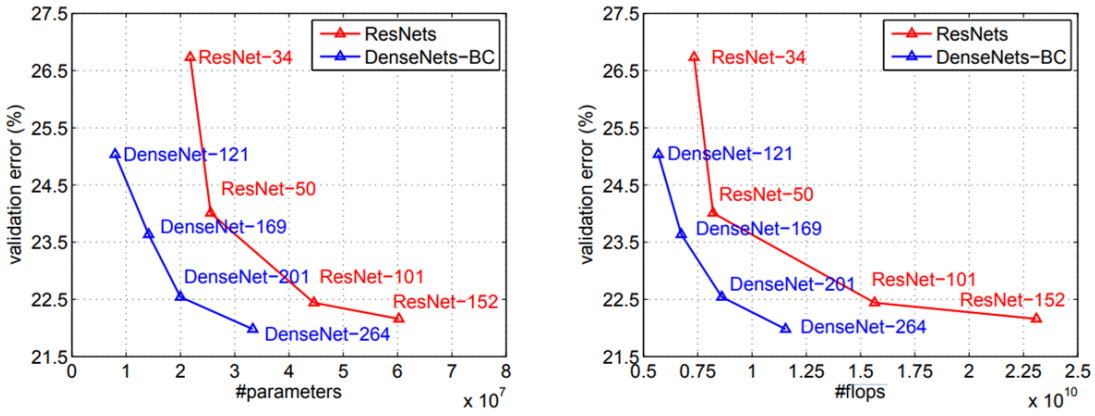


与 ResNet 不同，所有 DenseBlock 中各个层卷积之后均输出 k 个特征图，即得到的特征图的 channel 数为 k ，或者说采用 k 个卷积核。k 在 DenseNet 称为 growth rate，这是一个超参数。一般情况下使用较小的 k（比如 12），就可以得到较佳的性能。假定输入层的特征图的 channel 数为 k0 ，那么 l 层输入的 channel 数为 k0+k(l-1) 。本案例所使用的 densenet-161 是 k = 48 的结构。

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

DenseNet architectures for ImageNet

DenseNet 具有缓解梯度消失问题，加强特征传播，鼓励特征重用，并大大减少参数的数量等优点。在 top-1 error 相当的情况下 DenseNet 的参数量和计算量几乎都是 Resnet 的一半。



Comparison of the DenseNets and ResNets top-1 error rates

3. 评价指标

3.1 Cross Entropy Loss

kaggle 官方的评分函数 log loss 实际为 ***binary cross entropy loss***（二元交叉熵损失函数）：

Submissions are scored on the log loss:

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

- n is the number of images in the test set
- $\hat{y}_i$ is the predicted probability of the image being a dog
- y_i is 1 if the image is a dog, 0 if cat
- *log*() is the natural (base e) logarithm

该函数常用于二分类问题，是一种度量概率距离的方式，loss 值越小说明模型拟合的分布越接近真实分布，模型表现越好，反之模型表现越差。

而 torch 库中 CrossEntropyLoss( )函数与其有相同的计算意义，是评估本项目模型的最佳损失函数：

$$\text{loss}(x, \text{class}) = -\log\left(\frac{\exp(x[\text{class}])}{\sum_j \exp(x[j])}\right)$$

x 为预测结果， class 为标签对应的类别

该公式的意思是，对模型输出 x 计算 softmax 后，用 class 对应类别的概率：$\text{Probability}(\text{class}) = \frac{\exp(x[\text{class}])}{\sum_j \exp(x[j])}$，计算负对数似然函数(negative log-likelihood Loss, NLLLoss)，这与 LogLoss 具有相同的计算意义。

3.2 Accuracy

从应用角度考虑，我们还应评估模型的准确率——Accuracy。一般来说，loss 越小准确率越高，但也有时会存在偏差，所以实际的准确率 Accuracy 也应作为对模型的重要评估指标之一。

计算方法：Accuracy = 预测正确的样本数 / 样本总数

用预测结果与样本标签作比较，预测结果正确的样本占整体样本的比率即为准确率。

4. 优化器

4.1 Adam 优化算法

Adam：adaptive moment estimation（自适应矩估计），Adam 是一种对随机目标函数执行一阶梯度优化的算法，该算法基于适应性低阶矩估计，通过计算梯度的一阶矩估计和二阶矩估计而为不同的参数设计独立的自适应性学习率，是需要资源更少、令模型收敛更快的最优化算法。

4.2 随机梯度下降(SGD)及学习率策略

SGD：Stochastic Gradient Descent，随机梯度下降，每次迭代使用一个样本来对参数进行更新。

优点：

由于不是在全部训练数据上的 loss，而是在每轮迭代中，随机优化某一条训练数据上的 loss，这样每一轮参数的更新速度大大加快。

缺点：

① SGD 无法做到线性收敛，单样本更新参数使得模型准确率下降。

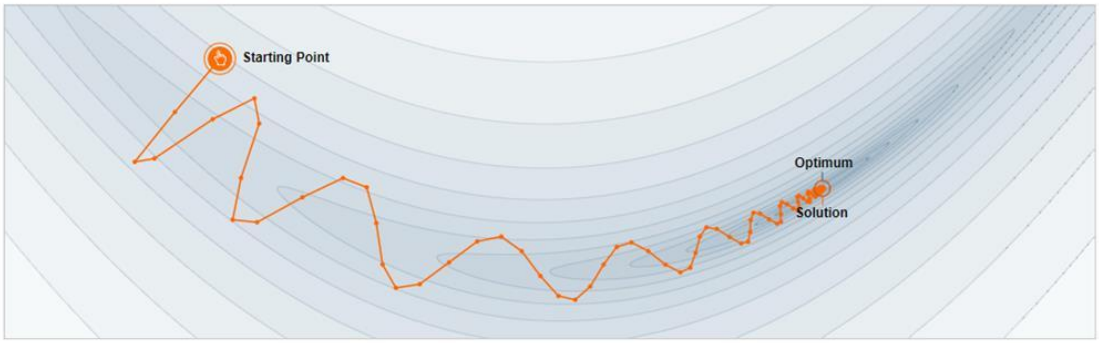
② 可能会收敛到局部最优解，因为单个样本并不能代表全体样本的梯度。

引入 Momentum（动量）的 SGD：

$$v_{t+1}=\mu*v_t+g_{t+1}$$

$$p_{t+1}=p_t-lr*v_{t+1}$$

p：parameters，g：gradient，v：velocity，μ：momentum



Momentum 梯度下降法，计算了梯度的指数加权平均数，当前权值的改变会受到上一次权值改变的影响。引入 Momentum 一方面可以使 SGD 加速，另一方面可以逃过局部最优解。

SGD 的学习率：

由于 SGD 的学习率 lr 是固定不变的，在训练后期不利于模型找到最优解，我们需要用学习率调度函数 lr_scheduler.ReduceLROnPlateau 来调整 lr。ReduceLROnPlateau 将读取一个参考指标，如果在 patience 期间没有看到该指标改善，则降低学习率。

5.超参数 batch_size

一般来说 batch_size 越大，其确定的下降方向越准，引起训练震荡越小，同时内存利用率越高，大矩阵乘法的并行效率越高。大的 batch_size 所需的迭代次数减少，但是对参数的修正也显得更加缓慢，batch_size 增大到一定程度后，梯度方向基本不再变化，过大的 batch_size 也可能会超出内存容量。而太小的 batch_size 类似于单样本更新，收敛相对困难，可能会得到局部最优解。所以我们在选择 batch_size 时需要根据实际情况权衡。

6.神经网络中的函数

6.1 Dropout

nn.Dropout(p=0.2)：在训练期间，以概率 p 随机置零张量的特征元素，这是一种可以防止神经元过拟合的正则化技术。

6.2 ReLU

nn.ReLU()：Rectified Linear Unit 激活函数。在每个线性层后面添加 Relu 激活函数，可以更有效率地梯度下降以及反向传播，避免梯度爆炸和梯度消失问题，同时使神经网络整体计算成本下降。

7.最终目标

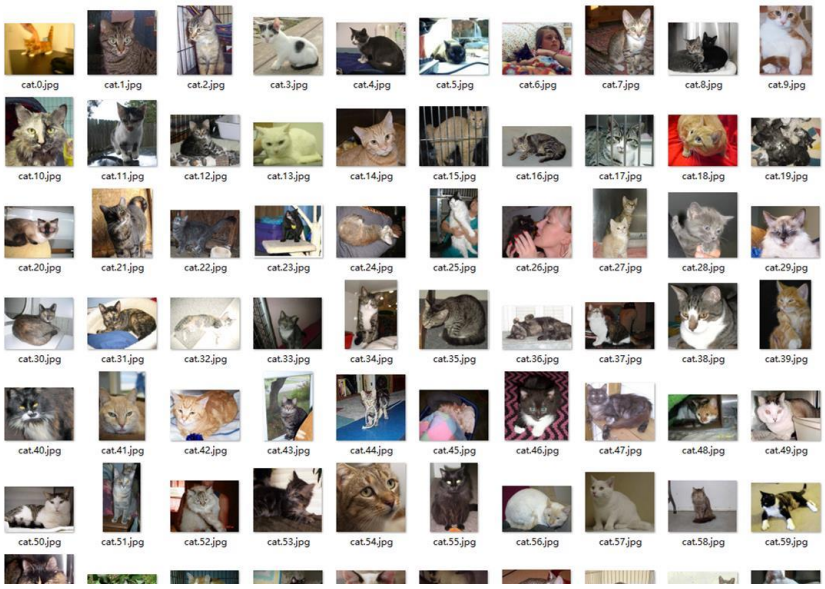
训练模型，实现预测准确率 99%以上，LogLoss 评分小于 0.06127（Public Leaderboard 前 10%）。

（三）实验过程

1. 数据探索

1.1 人工观察

训练集大部分都是猫狗图片，图片的文件名即为标签，文件名有序且分类清晰。

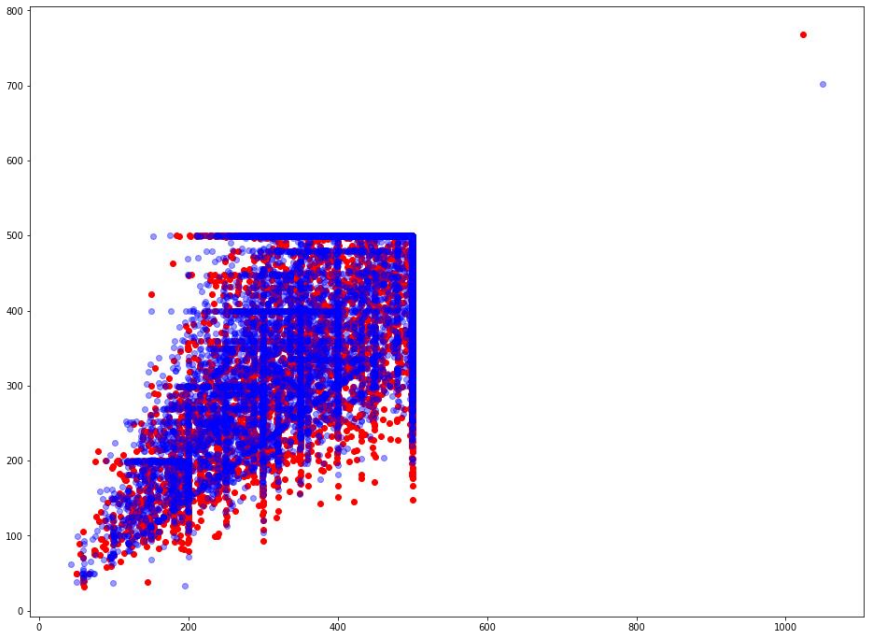


1.2 获取训练集标签信息

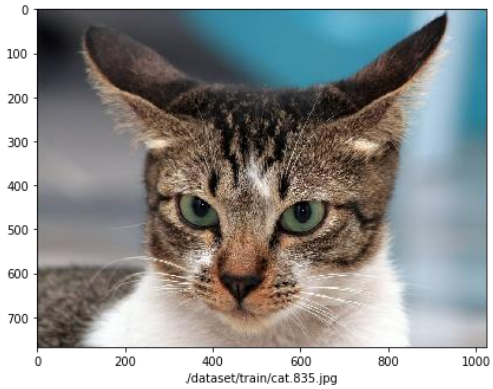
读取训练集图片文件名，按文件名判断标签种类，分别获得 12500 个 cat 和 12500 个 dog

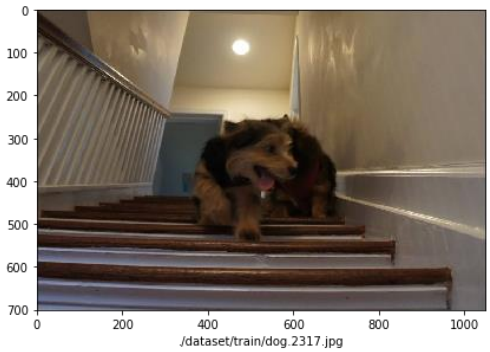
1.3 可视化数据分布

读取图片尺寸，并用散点图可视化尺寸分布，排查是否存在异常尺寸。由散点图可以观察到在尺寸分布图右上角，猫狗各有一个异常点。



根据尺寸信息筛选出这两张异常的图片，分别为 cat.835.jpg 和 dog.2317.jpg





读取图片，发现这两张其实都是正常的猫和狗，只要对其 Resize 后仍然可以作为正常训练样本使用。

2. 数据预处理

2.1 分割训练集和验证集

为保证数据分布平衡，正式训练集将由数量相等的猫和狗构成。从当前 training set 中各取猫狗样本的 80%作为正式训练集，剩下 20%作为验证集。

分割后：训练集包含 10000 张猫和 10000 张狗，验证集包含 2500 张猫和 2500 张狗。

2.2 图片变换

使用 torchvision.transforms.Compose( )来组合图片变换函数。

训练集使用缩放，随机裁剪、翻转等转化操作，其中随机性可以增加训练数据量，还可以减轻过拟合。

测试集则直接使用缩放和中心裁剪，少了随机因素，因为测试集不会用来训练模型。

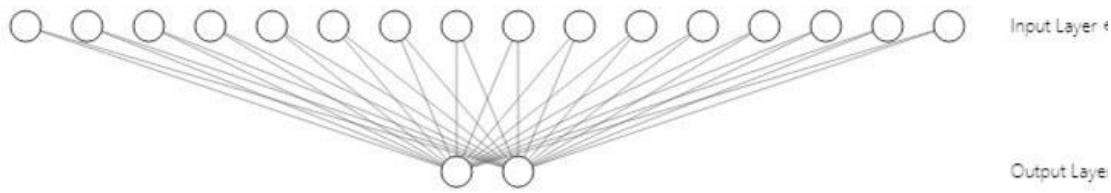
2.3 批次变换

由于 torchvision.transforms.Compose( )一次只能转化一张图，而模型要求维度为(batch,channel,height,width) 的输入，我需要自定义一个批次变换函数 batch_transform( )，该函数完成从读取文件到 transform 变换、图片张量拼接等操作，以便最终能返回符合模型要求维度的张量。

3. 第一次实验

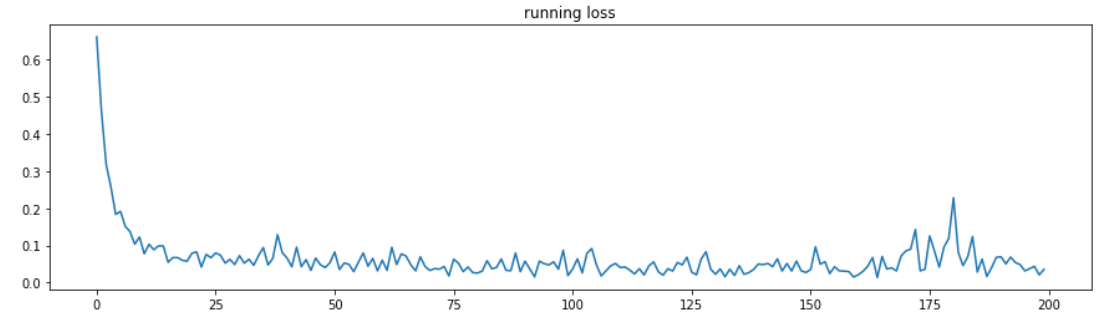
3.1 准备模型

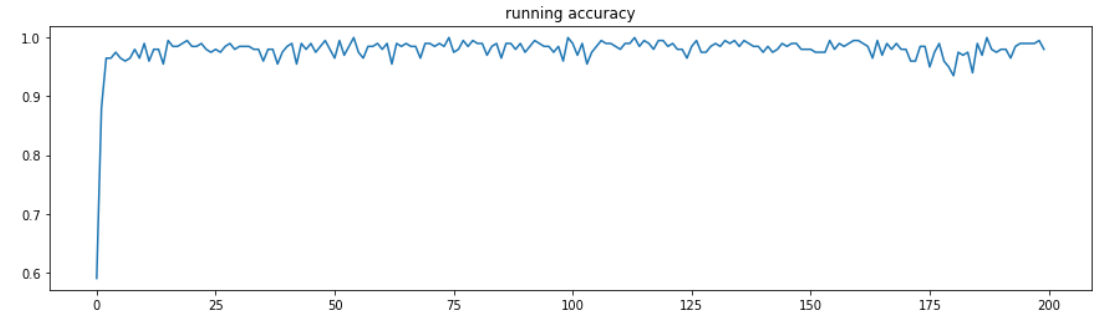
我打算使用迁移学习的方式学习数据集。首先从 torchvision 下载预训练的 densenet161，冻结除 classifier 以外其他所有层的参数，修改 classifier 为最简单的两层全连接结构 nn.Linear(in_features,2)，输出维度为 2 的向量，其两个维度分别对应两个预测类别。由于预训练模型是用 ImageNet 的图片训练的，而 ImageNet 的 1000 个分类中本来就有猫和狗，这里可直接使用 classifier 之前的预训练权重提取训练集样本特征作为 classifier 的输入。

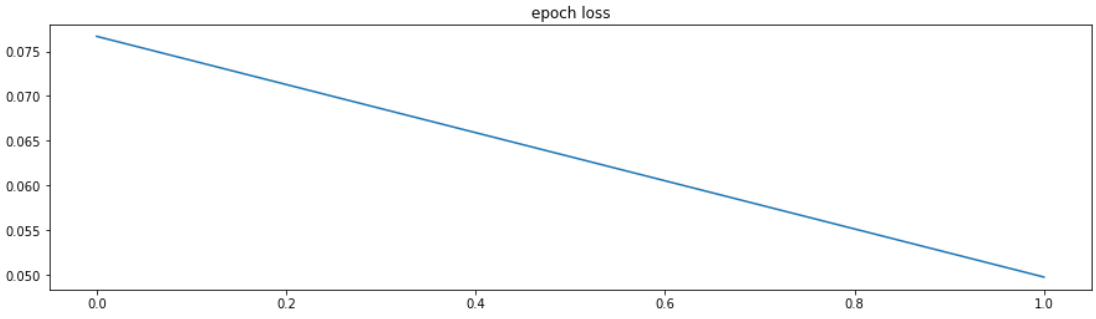


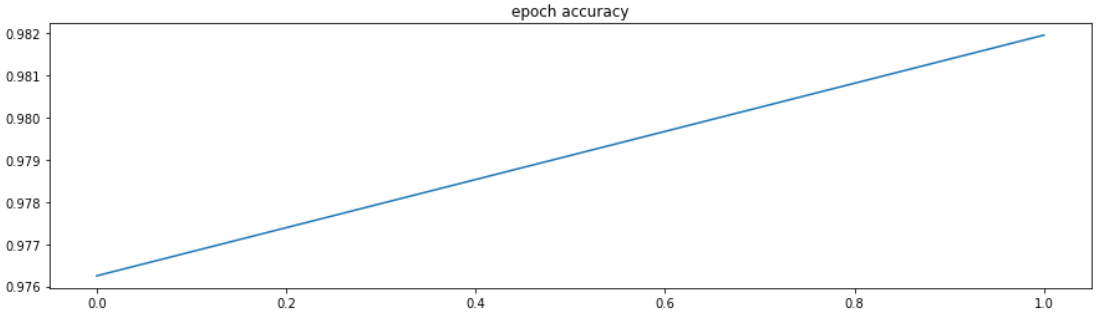
3.2 训练模型

模型：densenet161，其中 classifier=nn.Linear(in_features,2)
损失函数：CrossEntropyLoss
优化器：Adam
创建 DataLoader，将训练集随机打乱，并设置 batch_size=50
训练次数 epoch=2
训练过程指标如下：









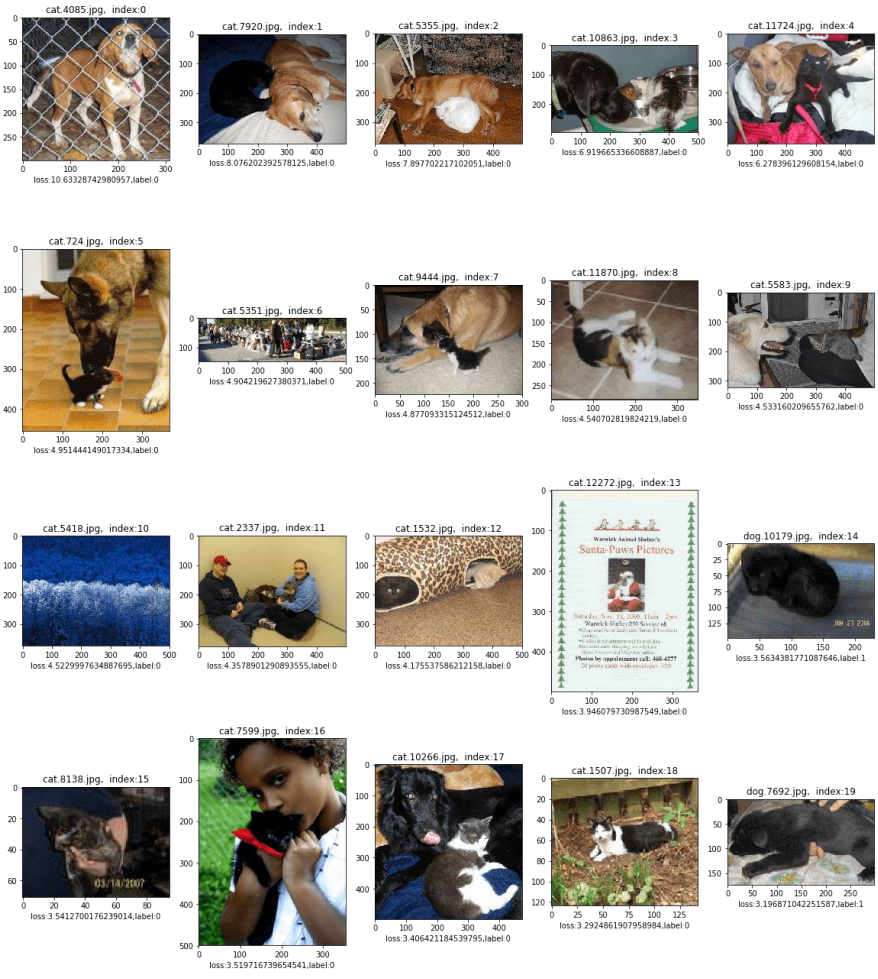
如上图，running loss 最后波动较大，我很好奇是什么样本导致 loss 出现巨大波动，于是回头检查训练集。

3.3 人工筛查

经查看，我发现训练集中存在一些构图异常或者标签与图片不符的样本，这些异常图片并非模型需要学习的样本，我们训练模型的目的只是为了区分猫和狗，学习其他种类的样本就失去了训练这个模型的意义。异常图片带着 cat、dog 标签混杂在训练集中，会让模型难以拟合样本，或者可能学习到异常样本特征，从而做出错误的判断。

3.4 模型筛查

我用训练过的 densenet161 再预测一次训练样本，这次我单独计算了每个样本的 loss，并设定一个阈值 **1.2040**（预测正确的概率不超过 30%，由负对数似然函数逆推而来），筛出 loss 大于 1.2040 的样本，看看是些什么样的图片。



如上图，筛出的图片中不乏一些标签错误、非猫非狗、像素模糊、颜色深暗、背景复杂、角度刁钻，或者同时存在猫和狗的图片。

3.5 对筛选出来的图片再进行一遍人工处理

第一张图 cat.4085.jpg 实际为 dog 而被错误标记为 cat，所以它的 loss 异常大，我修改标签后将其移回正常训练集。

另外还有很多黑猫黑狗的图片，这些图片相对难以辨认，但如果图片是清晰且正常的，应该移回正常训练集，因为测试集可能也包含这些品种。

非猫非狗的图片直接剔除，这不是模型需要学习的样本，模型的最初目标是学习识别猫和狗，如果学习其他类型的样本就失去了模型最初的意义。

同时存在猫和狗的图片属于歧义样本也应该剔除。

剩下的如果连人工都难以辨认的图片也应该剔除。

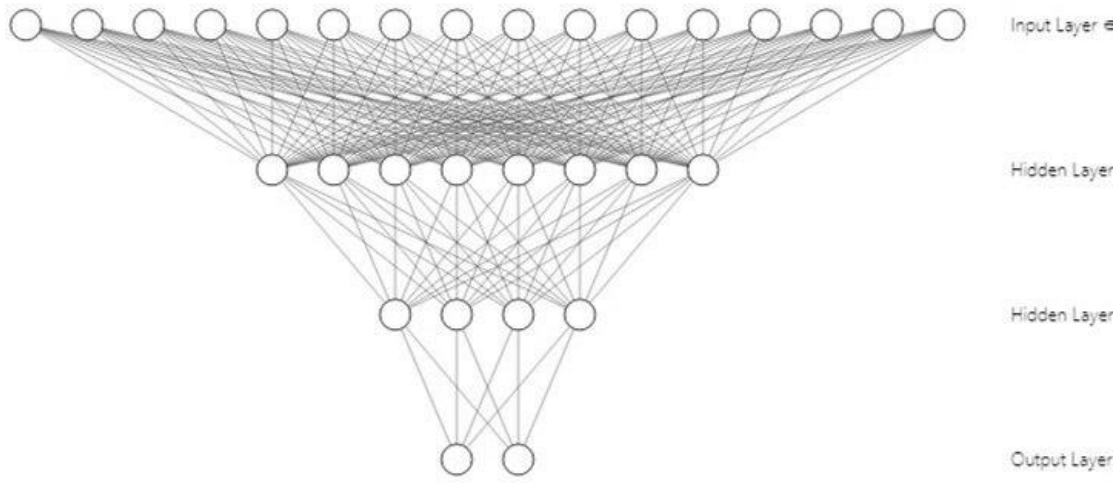
最终经过人工调整，训练集剩下 19906 个样本，共删除 94 个样本，我将用筛选后的 19906 个样本进行第二次实验。

4. 第二次实验

4.1 准备模型

修改模型：

第二次实验仍然使用预训练的 densenet161 作为基准模型，这一次我希望模型学习到更多样化的特征，所以修改其 classifier 为更复杂的四层全连接网络：



Classifier=Sequential ((2048, 1024),(1024,512),(512,256),(256,2))

每层的输入特征需先经过 Dropout 函数随机置零特征元素，前三层的输出特征需经过 ReLU()函数激活。

调整模型参数：

冻结 denseblock4 之前的所有参数，保留之后几层参数的梯度状态。

对于 CNN 深度学习模型来说，前面的特征层泛化能力较强，越往后面的层，越专注于原始数据(由于预训练模型是用 ImageNet 数据训练过的，这里指 ImageNet 数据)，这里我打开模型后面几层(classifier, features.norm5, features.denseblock4)参数训练状态，来学习当前的数据集。

4.2 优化器及学习率策略

学习率设置：

为了得到最优解，这次使用随机梯度下降算法 SGD 来训练模型，并针对不同层使用不同的学习率策略，其中 classifier 起步学习率为 0.01，而 features.norm5 和 features.denseblock4 起步学习率为 0.001，因为 classifier 之前的参数是训练过的，较接近最优解，使用了较小的学习率。

学习率控制：

另外使用 ReduceLROnPlateau 根据每个 epoch 的平均 validation loss 控制学习率衰减，这样在最后几个 epoch 可以使用更小的步伐，慢慢逼近最优解。

4.3 训练模型

模型：densenet161，修改 classifier

损失函数：CrossEntropyLoss

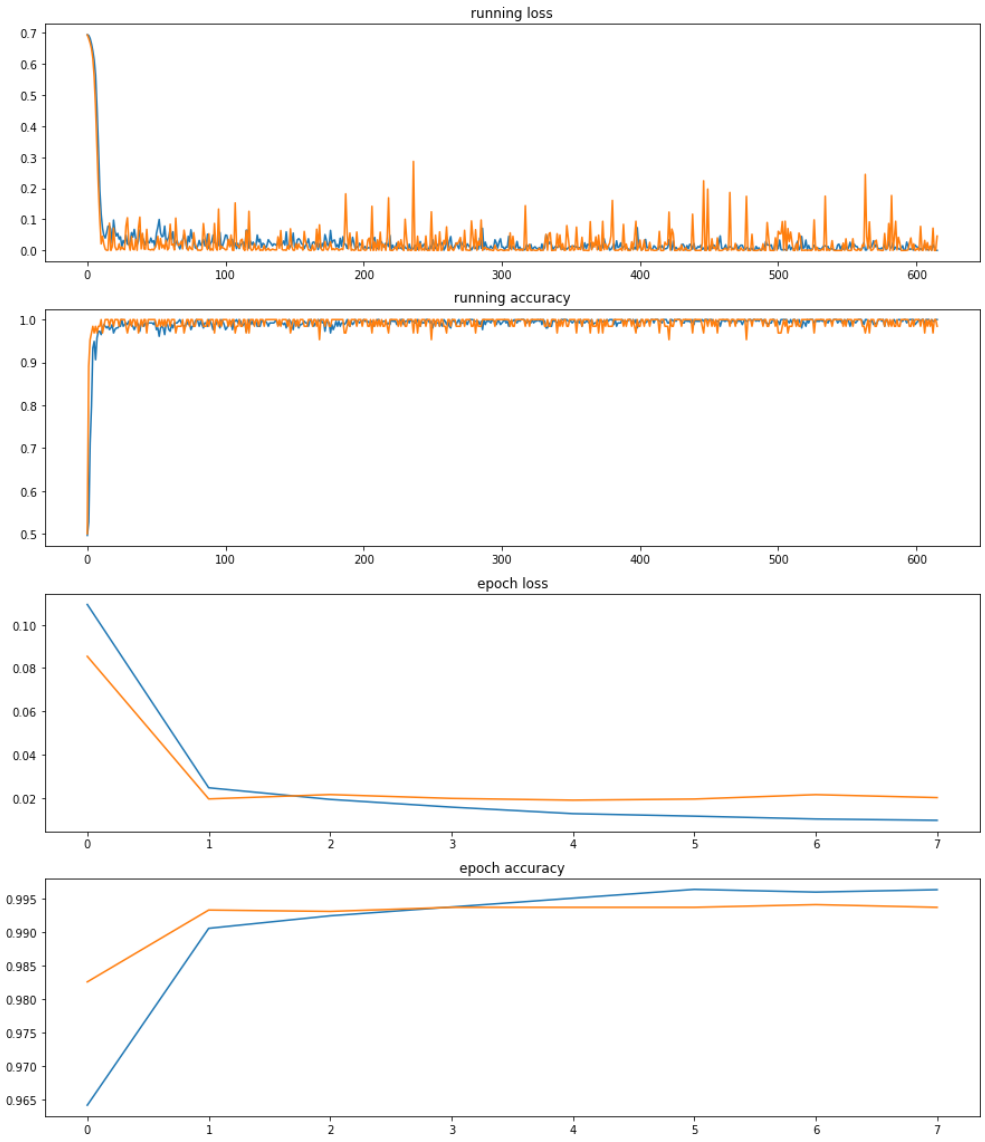
优化器：SGD，并设置 momentum=0.91

创建 DataLoader，将训练集随机打乱，并设置 batch_size=64

训练次数 epoch=8

训练指标如下：

Blue color：training， Orange color：validation



如上图，使用 SGD 算法训练模型，loss 收敛很快，可能是我使用了较大的学习率(0.01)起步的缘故，第一个 epoch 还没训练完，准确率就达到了 95%以上，另外也得益于 DesNet 的优秀性能以及使用预训练权重初始化模型。

但是 95%的准确率还未达到我们的目标，想要得到更小的 loss 和更高的 accuracy，需要降低学习率继续逼近最优解。这也是我使用 ReduceLROnPlateau 控制学习率的原因。

第二次实验我一口气训练了 8 个 epoch，每个 epoch 完成时都保存了模型权重，用于模型备选或者回溯。

在第 8 个 epoch 完成时，训练集 loss：0.009714， accuracy：99.63%；验证集 loss：0.020199， accuracy：99.37%。此时模型其实已经过拟合了。

5. 预测并挑选最佳模型

5.1 测试集数据

读取测试集 12500 个文件，保存成（File_Path, ID）格式。

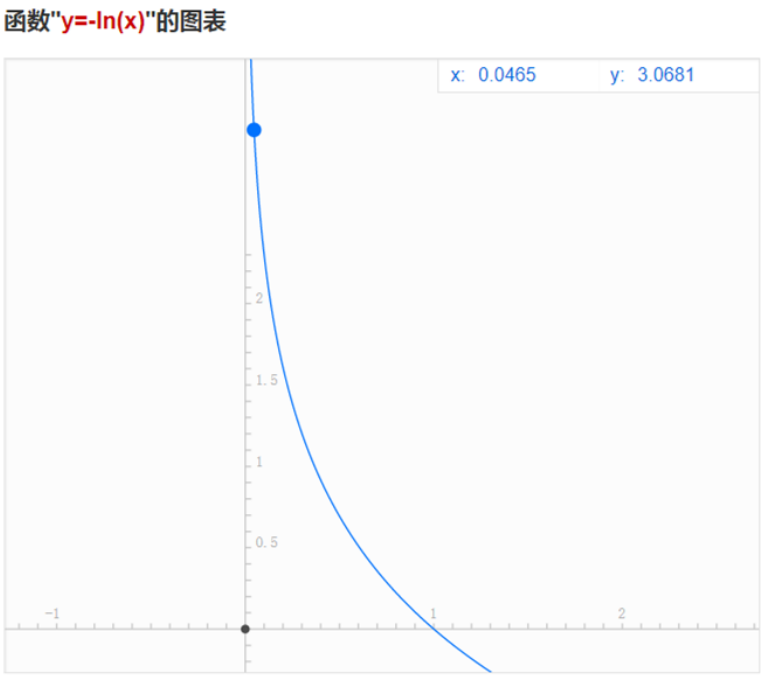
创建 DataLoader，并设置 batch_size=100

5.2 选择最佳模型

现在我有 8 个训练过的模型，由于不确定哪一个是最优解，我用 8 个模型分别对测试集进行了预测，最后取结果最优的模型作为最终解。

5.3 将结果 clip

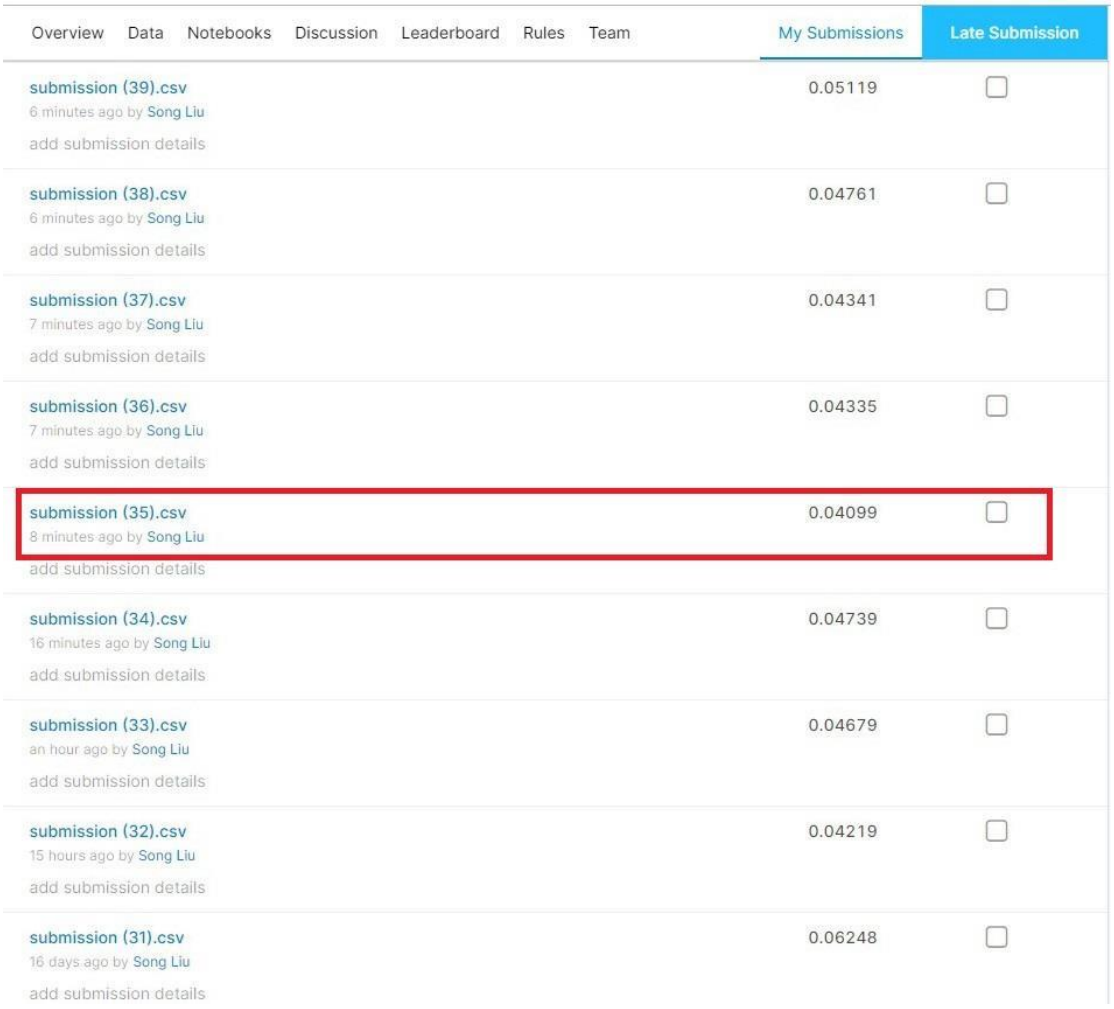
Log Loss 外层实际上是负对数函数，我们可以从 y = -ln(x)图形观察其规律，这是一个单调递减函数，在(0, 0.05)区间的变化率远大于在(0.995,1]区间的变化率，如果将结果 clip 到[0.005,0.995]，对于预测正确的样本 loss 不会升高很多，但对于预测错误的样本 loss 却可以降低很多，压缩变量 x 的区间，可以带来 y 区间平均值降低的效果。



5.4 提交结果

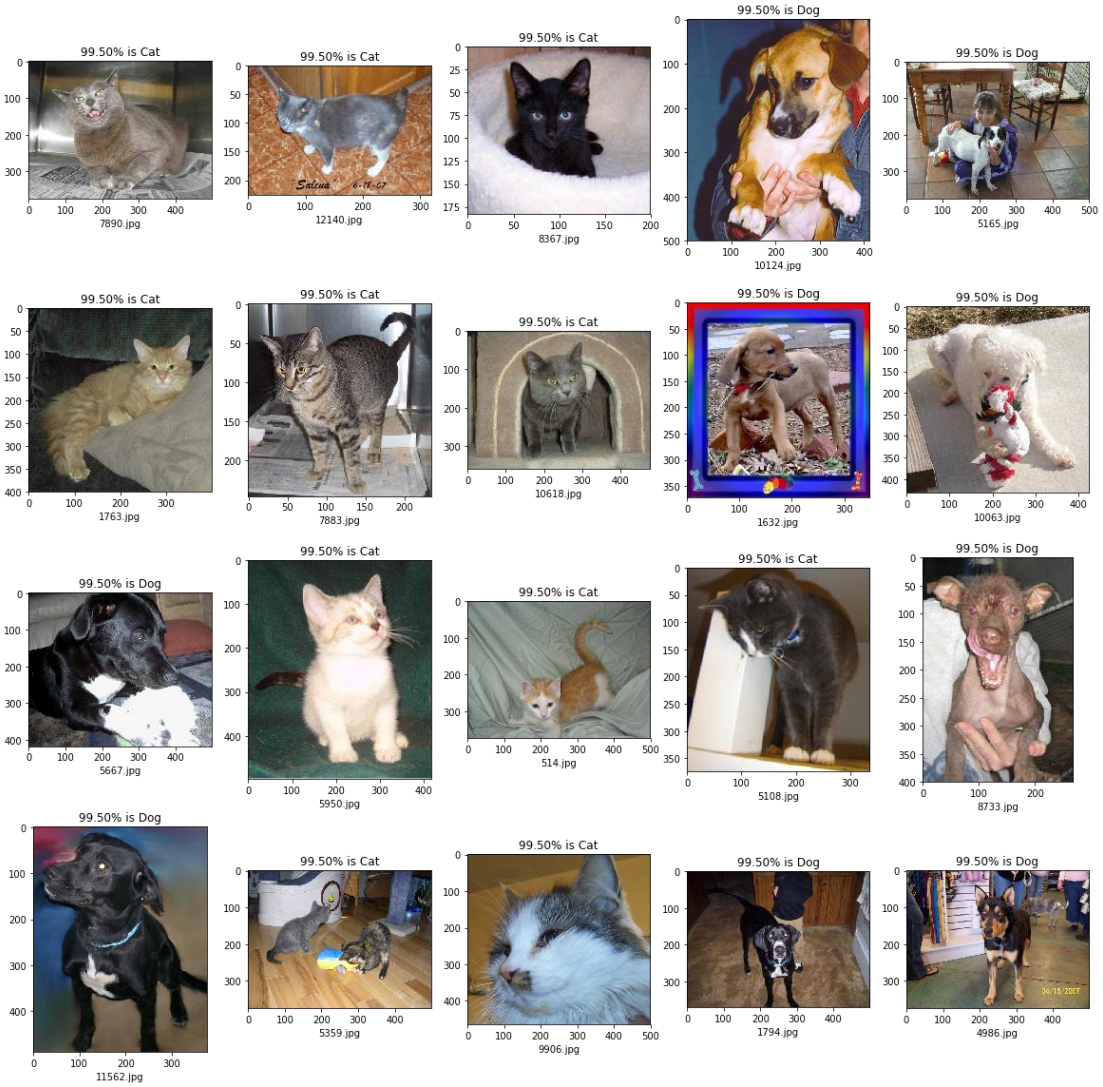
将预测结果按要求格式保存为 cvs，并提交至 https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/submit，取最小分数作为最优分数。

经验证，八个模型中第五个取得了最好成绩 0.04099，可以确定第五个模型是我们需要的最佳模型。



6. 可视化预测结果

使用第五个模型预测，并可视化预测结果，经过随机验证，模型可以很准确地预测测试图片，即使一些不常见的品种也没有问题。



（四）项目总结

1.训练数据中存在异常值，清洗数据是机器学习的第一步。

2.超参数 learning rate 和 batch_size 会对结果产生影响，如何选择合适的超参数，需要综合数据分布及特点、模型结构和优化器等因素考虑。

3.精确调整的 SGD 优于 Adam 算法。

4.如果损失函数无法很好收敛，适当增加网络深度不失为一种好方法。

参考文献：

1. Gao Huang/Cornell University, Zhuang Liu/Tsinghua University, Laurens van der Maaten/Facebook AI Research,《Densely Connected Convolutional Networks》

2. ADAM 算法：Diederik P. Kingma, Jimmy Lei Ba,《A METHOD FOR STOCHASTIC OPTIMIZATION》

3. momentum：Ilya Sutskever, James Martens, George Dahl, Geoffrey Hinton,《On the importance of initialization and momentum in deep learning》

4. Dropout：G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov,《Improving neural networks by preventing co-adaptation of feature detectors》

5. 博客：https://bigquant.com/community/t/topic/127342

6. 知乎：https://zhuanlan.zhihu.com/p/34435247