猫狗大战

机器学习工程师 纳米学位 毕业项目

张洪阳 2018年4月15日

目录

1. 问题定义

1.1 项目概述

猫狗大战源自 Kaggle 于 2013 年举办的一个娱乐竞赛项目，要求分辨给定测试图片是猫还是狗，属于计算机视觉领域图像分类问题，适合使用以卷积神经网络（Convolutional Neural Network，CNN）构建模型、最小化对数损失函数作为策略、梯度下降与反向传播作为算法的深度学习方法。

卷积神经网络根据生物视觉神经中感受野（Receptive Field）概念的启发，使用卷积核与上层的稀疏连接取代传统 DNN（Deep Neural Network）层与层之间的全连接权重，极大降低了网络参数数量，而理论上相比全连接网络表示能力只是略微降低，并且更容易训练。由于计算机视觉相关问题数据结构的特殊性（图像、视频等），往往每个像素只与附近区域的像素高度相关，非常适合使用 CNN 模型。

1.2 问题陈述

Kaggle 竞赛提供的数据包含了一个训练集和一个测试集，训练集有 25000 张图片，猫狗各一半，测试集有 12500 张图片。这些从真实世界中采集而来的猫狗图片，图像分辨率不同，背景环境复杂，猫狗的品种繁多，这些都为我们的分类增加了难度。我们需要搭建模型，用训练集里面得 25000 张图片来训练模型，最后再使用测试集来验证我们的模型，看看对于猫狗类型的判断准确率。

1.3 评价指标

模型输出单元使用 Sigmoid 激活函数，损失函数采用对数损失函数，即伯努利分布下的交叉熵：

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

这是一个二分类问题的通用评价指标，

n 代表测试集中的图片数量；

$\hat{y}_i$为测试图片是狗的概率；

y_i如果图片是狗为 1，是猫为 0。

2 分析

2.1 数据的探索

训练数据集包括 25000 张图片，并在文件名中标注了图片为猫还是狗，猫狗数量各占一半，测试数据集包括 12500 张图片，没有标注类别，文件以数值 ID 命名。

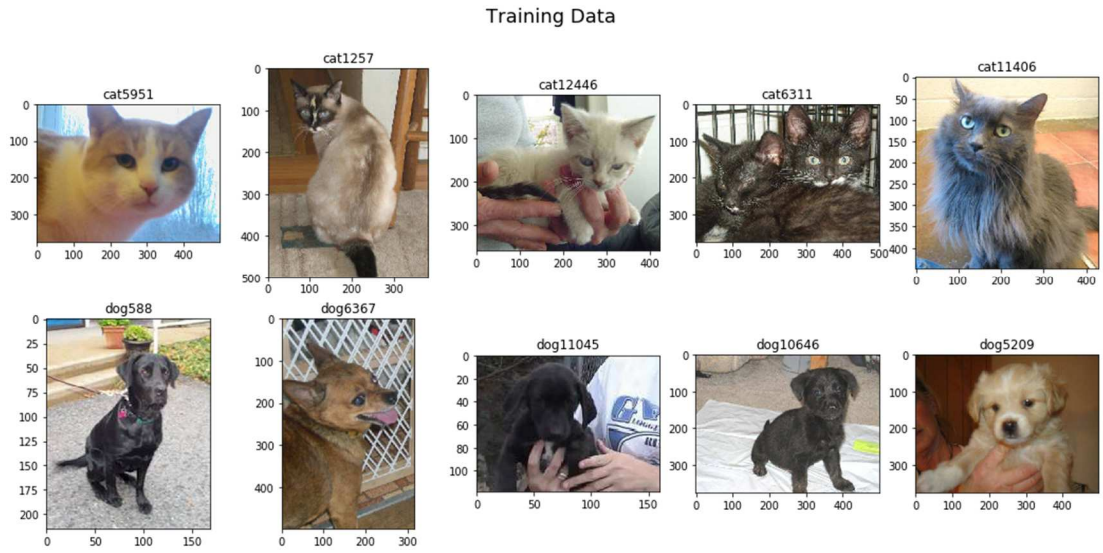


图 1. 训练数据按类别随机抽样

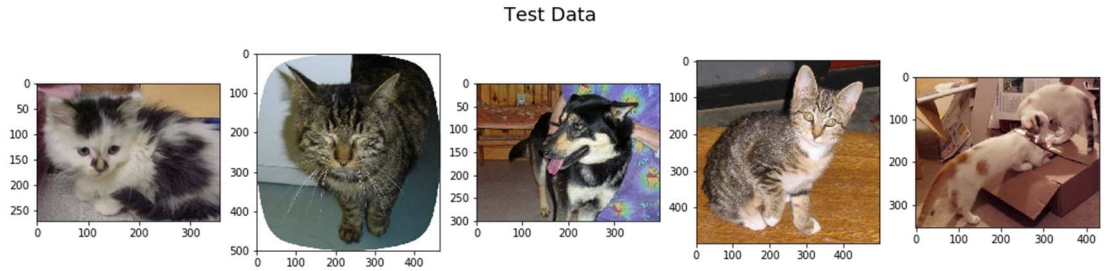


图 2. 训练数据随机抽样

如上图所示，图 1 为从训练数据集中随机抽样猫狗各 5 个样本，图 2 为从测试数据集中随机抽样 5 个样本，可见数据内容具备多样性与复杂性。如：图片尺寸参差不齐，不同图片包括了猫/狗身体不同角度，不同的猫/狗数量，有的图片没有包含全身，甚至没有正面脸部，图片背景元素也具备充分的干扰性。

图片样本特征十分抽象，很难使用数据统计方法剔除异常值，由于数量太大，逐个进行裁剪、填充需要大量时间，并且深度学习对于此类异常样本具有较强鲁棒性。所以，需要使用预训练的分类模型对数据集进行初步分类，识别非猫狗异常样本，然后再对数量较少的异常样本集合进行人为判断，对内容正确但尺寸异常的数

据进行裁剪、填充。本项目使用了 Xception 预训练模型进行数据处理,并手动进行筛选.

2.2 算法和技术

根据分析可知，猫狗识别本是上是一个二分类问题。适用于分类的机器学习算法很多，比如支持向量机 SVM，随机森林 RF, 决策树 DT， 然而这些方法在注明数据及上 Imagenet 数据集上取得的效果一直不是很好，并不能有效的解决图像识别问题，人们急需要一种新的方法来克服这一计算机视觉难题。直到 2013 年 Alex 发明 Alexnet 深度学习神经网络模型取得 Imagenet 比赛第一名，图像识别问题才算解决了。深度学习其实并不是一个全新的概念，早在上世纪 40 年代，就已经提出，但是由于当初的计算级的计算能力小，数据集稀缺，深度学习一度处于低谷。

卷积神经网络根据生物视觉神经中感受野（Receptive Field）概念的启发，使用卷积核与上层的稀疏连接取代普通 DNN 层与层之间的全连接权重，极大降低了网络参数数量，而理论上相比全连接网络表示能力只是略微降低，并且更容易训练。由于计算机视觉相关问题数据结构的特殊性（图像、视频等），往往每个像素只与附近区域的像素高度相关，非常适合使用 CNN 模型。

卷积是一种数学运算，其标准定义为：

$$x(t) * w(t) = \int_{-\infty}^{\infty} x(a) \cdot w(t-a) da$$

卷积运算的实际意义需要应用到不同场景，若将 t 理解为时间，令 a<0 时 w=0，可以理解为从过去到时刻 t，x 以 w 为权重的累加和。扩展到离散场景时，可定义为：

$$x(t)*w(t)=\sum_{a=-\infty}^{\infty}x(a)\cdot w(t-a)$$

在 CNN 图像应用场景中，卷积可以扩展为更为直观的二维形式：

$$I(i,j)*K(i,j)=\sum_{m=1}^{W}\sum_{n=1}^{H}I(m,n)\cdot K(i-m,j-n)$$

其中 I 表示输入图像，i、j 为图像像素坐标，K 表示卷积核，W、H 为卷积核尺寸，即对输入图像中每个像素值（暂不考虑步长）及其附近 W、H 区域的像素值进行加权累加和，以得到一副新的“图像”。

通过卷积计算得到的新“图像”，可以表示出原图像的某些特征，比如传统的边缘检测算子就是一种人为设计权重的卷积核，将如 3×3 大小的卷积核分别对原图像每个像素进行卷积运算，从而得到如下图所示的边缘检测效果，而 CNN 模型则是根据代价函数和梯度下降让卷积核自己学习需要的权重，让模型自己学习需要提取的图像特征。并且，随着模型深度的增加，卷积核的视野逐渐增大，从而可以提取更为高级、抽象的特征，如从边缘到毛发，最终到身体轮廓。

边缘检测算子

卷积神经网络通常包含以下几种层：

卷积层（Convolutional layer），卷积神经网路中每层卷积层由若干卷积单元组成，每个卷积单元的参数都是通过反向传播算法优化得到的。卷积运算的目的是

提取输入的不同特征，第一层卷积层可能只能提取一些低级的特征如边缘、线条和角等层级，更多层的网络能从低级特征中迭代提取更复杂的特征。卷积过程采用了稀疏交互，参数共享，等变表示三大思想。稀疏交互是利用了局部感受野，限制了空间的大小，参数共享就是权值共享不但能减少参数数量，还能控制模型规模，增强模型的泛化能力。

线性整流层（Rectified Linear Units layer, ReLU layer），这一层神经的活性化函数(Activation function)使用线性整流(Rectified Linear Units, ReLU) f(x)=max(0,x)。

池化层（Pooling layer），通常在卷积层之后会得到维度很大的特征，将特征切成几个区域，取其最大值或平均值，得到新的、维度较小的特征。池化输出的是邻近区域的概括统计量，一般是矩形区域。池化有最大池化、平均池化、滑动平均池化、L2 范数池化等。池化能使特征获得平移不变性。卷积也会产生平移不变性，卷积对输入平移是不变的，池化对特征平移是不变的。池化能显著地减少参数，能解决不同规格的输入的问题。

全连接层（ Fully-Connected layer），把所有局部特征结合变成全局特征，用来计算最后每一类的得分。

对于图像数据的维度规模，大型 DNN 结构，随着网络变大和参数增多，过拟合情况会逐渐严重，表现为训练集上升， 验证集下降。但是随着规模继续变大，网络变成了"难以训练"的问题，此时的表现是训练集下降，验证集下降。相对的，CNN 的缺点是增加了额外的需要人为调整的超参数，如卷积核大小、步长等。

项目使用 Python3.5 开发，通过对 Tensorflow 进一步封装的 Keras API 进行模型实现与训练。Tensorflow 是 Google 提供的深度学习开源框架，其主要功能是自

动完成了计算图的梯度计算与反向传播过程，使开发者可以专注于模型搭建与训练，并且将前馈计算、反向传播等密集浮点运算交由效率更高的语言完成，支持 GPU 加速计算。由于 GPU 的构架特点，其浮点运算性能远超 CPU，使用 NVIDIA GPU 加速计算，还需要安装 CUDA 以及 cuDNN。Keras 对 Tensorflow 进行了进一步封装，实现了常用的层次结构，使调用接口更加人性化，并且提供了一些搭建好的预训练模型，以提高开发效率，避免重复造轮子。

项目主要使用 Keras 提供的 ImageNet 数据集预训练模型进行迁移学习，通过多模型拼接等方法提高模型性能。

需要注意，由于 Kaggle 评估指标为损失函数，进行多模型拼接时，不宜使用投票表决法、均值法等提高准确度指标的方法；同时，选择解决过拟合方法时，也不宜使用 L1/L2 Regularization，前者会导致权重稀疏，后者会导致权重衰减，两者都会使预测时输出层 Sigmoid 函数接收的激活绝对值变小，从而输出概率不够自信。Dropout 方法在每次训练时随机丢掉一部分特征，使当前某些强特征随时会被在下次训练时丢弃掉，从而让模型不会过分依赖某几个强特征，虽然也会降低训练时的输出层激活绝对值，但预测过程中将不会对特征进行丢弃，非常适用于此项目；图像增强方法是通过让模型见到更多人造数据以提高泛化性能，但会相应增加每代训练时间，需要根据实验结果进行取舍。

又由于 loss 对数函数的性质，当样本预测分类错误时，若模型输出过于自信，使对数函数输入接近 0，将导致对数函数值接近负无穷，而样本分类正确时取值于对数函数饱和端，略微降低自信度影响不大，所以在提交 Kaggle 评估得分时，应当对输出概率进行截断。令预测错误率为 a，输出截断范围[x, 1-x]，该优化问题可近似为：

$$\arg\max_{x}[a\log x+(1-a)\log(1-x)]$$

求导得到 x=a，由于模型在验证集正确率约为 0.996，这里采取对输出截断至范围[0.005, 0.995]。

2.3 基准模型

ImageNet 的分类评价指标一般为 top-5 test error，即根据概率从大到小排名 前 5 的分类结果中没有包括正确分类的百分比错误率。

GoogLeNet 即第一版 Inception 模型，增加网络宽度，减少了选择卷积核尺寸等参数的人为因素，直接将不同尺寸卷积核以及池化层放在同一层作为一个 block，并拼接为一个输出层，然后为了降低计算参数，使用 1×1 卷积核进行降 维，达到单模型 7.9%，组合模型 6.7%的 top-5 error[1]。Inceptionv3 在 GoogLeNet 的基础上，将大型卷积核分解为多个小型甚至一维卷积核，并加上非线性激活单元，在降低参数数量的基础上提高了模型表现， 组合模型最好表现达到 3.58%的 top-5error[2]。

ResNet 则着重解决网络深度增加难以训练的问题。以解决输入到输出的直接映射为核心思想，将输出函数改为源目标函数 H(x)与输入的残差（Residual） F(x)，再将输入层无参数连接（或维度映射）到输出层，使最终输出保持为 H(x)=F(x)+x，使网络能够达到极高的深度而几乎不增加额外参数。ResNet50 与 ResNet152 分别达到 6.7%与 5.7%，组合模型甚至达到 3.7%的 top-5error[3]。

Xception 将图像空间信息与 channel 信息分开处理，使用 depthwiseseparable convolution 方法对每个 channel 分开卷积，然后使用 1×1

卷积核进行合并，进一 步降低参数数量，提高模型表现能力，在 ImageNet 上单模型表现接近 Inception v4，作者将此归为 Inception 的复杂设计对 ImageNet 存在过拟合[4]。

利用 ResNet50、InceptionV3、Xception 模型进行迁移学习，使用输出的特征进行模型融合训练，可以在验证集上达到 99.6%的正确率，在测试集的损失函数值达到 0.4141，提交 Kaggle 排名为 20/1314，要求项目结果不低于此标准。

3 方法

3.1 数据预处理

由于 ImageNet 分类中已包含 118 种狗类别和 7 种猫类别，使用预训练模型分别对各 12500 个猫狗训练样本进行预测，并输

出前 n 个预测结果分类，若猫样本的 n 个预测结果中没有 7 种猫类别，或狗样本的 n 个预测结果中没有 118 种狗类别，即判定该样本为异常样本，n 值的大小经过反复实验设定为猫样本 60，狗样本 20。最后使用 Xception 模型预测，共得到共 62 个异常样本。

部分异常样本

可以发现，其中大部分图片如“dog.10161.jpg”、“cat.4688.jpg”判定为异常样本毫无疑问，必须剔除。为了进一步筛选异常样本，在将此 62 个异常样本剔除后，本次项目从中进一步人为判断筛选出 17 个轻度异常样本，并将其裁剪、填充至模型需要的尺寸，项目选用的模型输入尺寸为 299×299.

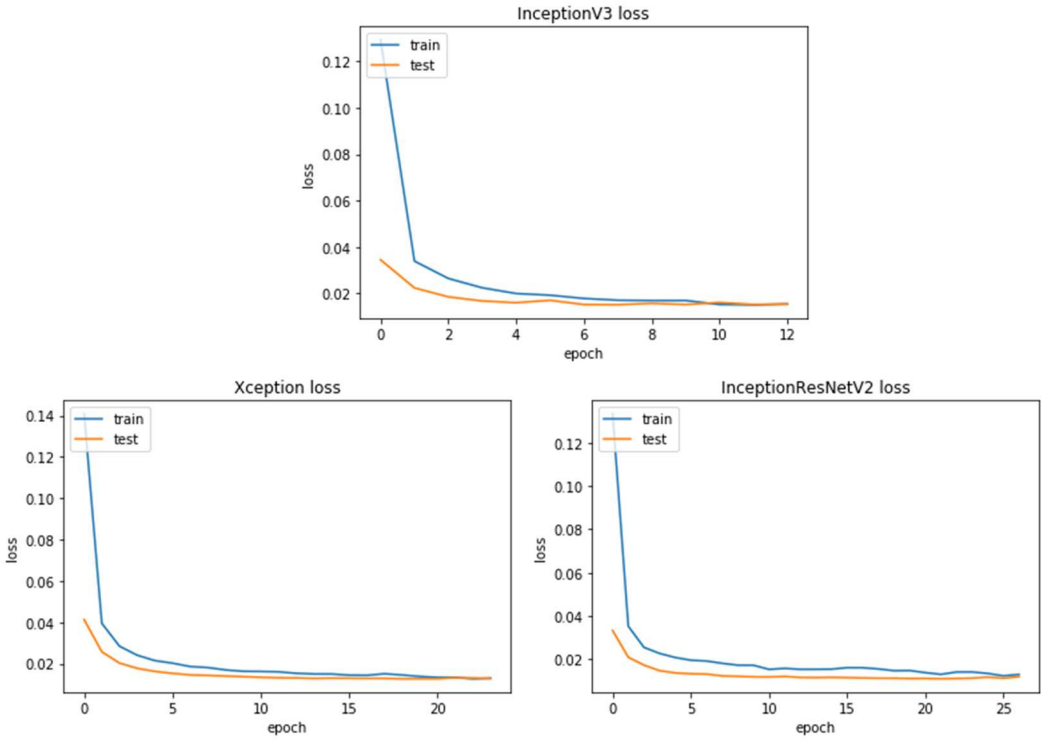
3.2 实现

3.2.1 实验结果

训练使用 64 个样本作为 batch size，Adam 作为优化方法，学习率1^{-3}，以及使用衰减率1^{-6}的 Keras 默认衰减函数：

$$lr = \frac{lr}{1 + decay\_rate * batch\_num}$$

将 Early Stop 设置为验证集 loss 连续 5 代没有降低即停止，保留 loss 最低的模型，学习曲线如下：



学习曲线

将预测结果提交 Kaggle 之后，InceptionResNetV2 表现最佳，最好能得到 0.03929 的 loss，InceptionV3 则最好得到 0.04121 的 loss，Xception 模型虽然在验证集表现强于 InceptionV3，但在测试集只得到了最好 0.04188 的 loss，说明 Xception 对验证集的过拟合更严重。

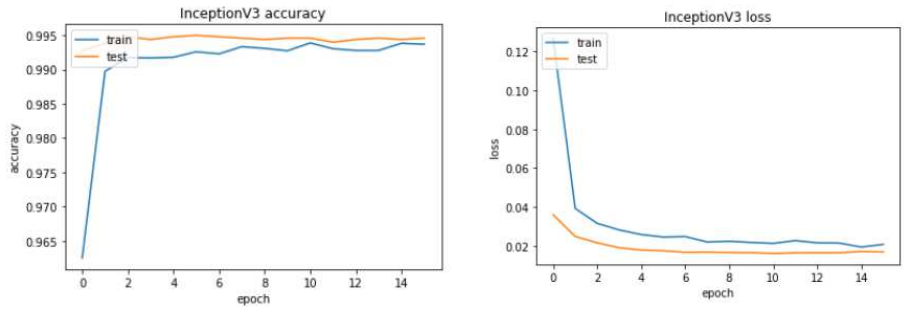
3.2.2 多模型融合

本项目中，直接应用了 4 个已经预训练好的模型，ResNet50，InceptionResNetV2，InceptionV3，Xception。这 4 个模型预测时关注的特征相似但不完全相同，在进行多模型融合时，可以将从不同模型提取的特征向量拼接合并为一个特征向量，以增加模型性能。用于提取特征向量的模型结构，之后构建模型

对特征向量拼接。

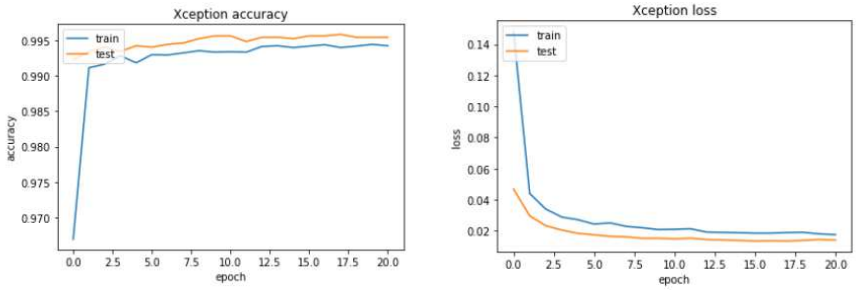
先查看单个模型的训练效果:

InceptionV3 的 acc 为 0.9946, loss 为 0.0168:



InceptionV3 单个模型学习曲线

Xception 的 acc 为 0.9954, loss 为 0.0142

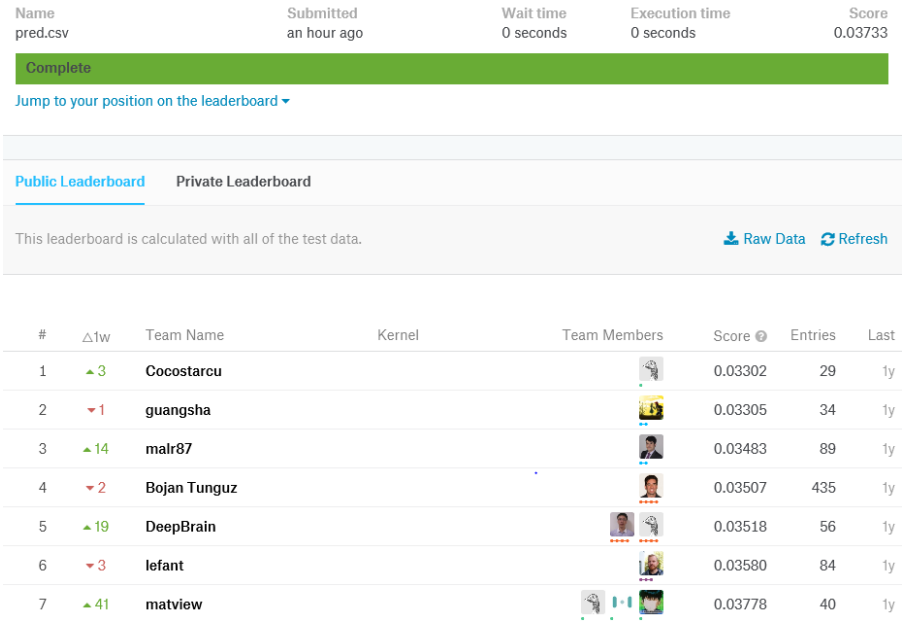


Xception 单个模型学习曲线

4 结果

4.1 模型的评价与验证

可以看到单个模型的准确率不能达到我们的目标,其他两个模型也如是,不再列举. 之后选择不同模型组合,从实验结果来看,四个模型融合取得了较好的效果,训练集上实现 acc 为 0.9981, loss 为 0.0055,验证集上 acc 为 0.9962, loss 为 0.0099. Kaggle 上 score 为 0.03773,超过了第 7 名的成绩,达到了我们预期的效果.



Name	Submitted	Wait time	Execution time	Score
pred.csv	an hour ago	0 seconds	0 seconds	0.03733

Complete

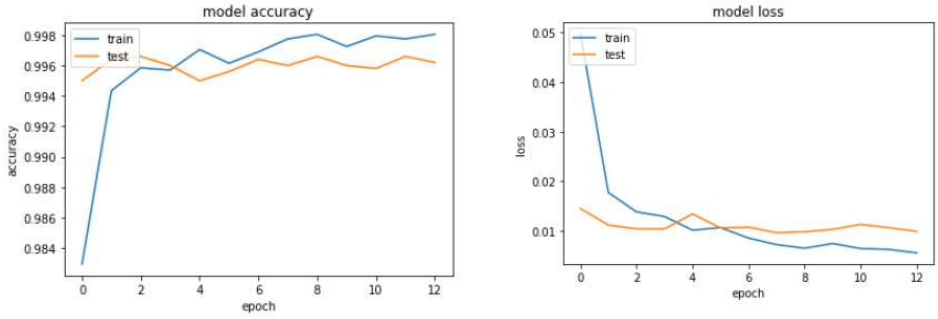
Jump to your position on the leaderboard

Public Leaderboard | Private Leaderboard

This leaderboard is calculated with all of the test data. Raw Data Refresh

#	△1w	Team Name	Kernel	Team Members	Score	Entries	Last
1	▲3	Cocostarcu			0.03302	29	1y
2	▼1	guangsha			0.03305	34	1y
3	▲14	malr87			0.03483	89	1y
4	▼2	Bojan Tunguz			0.03507	435	1y
5	▲19	DeepBrain			0.03518	56	1y
6	▼3	lefant			0.03580	84	1y
7	▲41	matview			0.03778	40	1y

Kaggle 上成绩排名



多模型融合学习曲线

4.2 合理性分析

最后可以看到，经过结合四种不同模型，合并它们的特征向量，我得到较好的识别结果，其效果大于单独使用其中的一种模型。因为这里的每一个模型都是久经检验的，可以说更有所长，综合它们到一起，可以高度概括出图片当中的内容，所以最后得出的结果较好。

5 结论

5.1 思考

整个工作的流程可以概括为：数据预处理，提取特征向量，综合不同的模型载入特征向量，构建模型并训练，测试集预测最终结果。决定最后结果的关键点在于第二步，如何提取特征向量，我直接使用了已训练的，久经考验的几个模型，综合使用它们，这样可以节约我自己的训练时间，且可得到较好的结果。我还考虑过这几种模型不同的组合使用，发现四种全用时，效果最好。我还考虑过 dropout 数值对于结果的影响。我们知道 dropout 技巧实际上是一种模型平均，就是把来自不同模型的估计或者说预测通过一定的权重平均起来。每次以一定的概率忽略一些隐层节点，这样每次训练的网络就是不一样的，每次训练都可以看做一个新的模型；此外，隐层节点以一定的概率出现，不能保证哪两个节点同时出现，这样就阻止了某些特征关联与其他特征的情况。最后发现，dropout 在 0.5 的时候，效果最好。总的来说，目前的结果还是不错的，高效的识别率，Kaggle 上 Score 为 0.03773,超过了第 7 名的成绩,较好解决了猫狗识别问题。

不同模型学习到的特征图会有差别，使用多模型特征融合相当于从多个不同的角度综合看待问题，性能提升很大。

5.2 改进

想要进一步提高识别率，似乎就需要深入钻研一下神经网络基本结构，不断练习调节参数，积累对于调参的心得。众所周知，深度神经网络的参数非常之多，而且内部具体是如何工作的，目前还是一个黑箱状态，人们了解的并不够透彻，所以

不同参数之间对结果有着不同的影响，这个需要更多的时间和经验来掌握。

参考文献

[1] Christian Szegedy, Wei Liu, Yangqing Jia. Going Deeper with Convolutions, 2014.

[2] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe. Rethinking the Inception Architecture for Computer Vision, 2015.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren. Deep Residual Learning for Image Recognition, 2015.

[4] François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions, 2017.