机器学习工程师纳米学位毕业项目
基于深度学习的猫狗识别

丁元虎
2018年04月14日

I. 问题的定义

1.1 项目概述

Dogs vs. Cats 项目来源于2013年Kaggle举办的一场娱乐性质的图形分类识别比赛。时隔三、四年，基于深度学习的图像识别技术取得了突飞猛进的发展，先后出现了VGGNet (14.09), GoogLeNet (14.09), ResNet (15.12), Inception v3 (15.12), Inception v4 (16.02), Xception (16.10)等等很有名的模型，其中许多模型的识别准确率已经超过人类。两年前，Kaggle又重新把这个有名的项目作为 'playground competition' 带回来了。本项目使用的数据集即该竞赛使用的数据集，数据集包括训练数据集和测试数据集，其中训练数据集包括25000张标签好的猫或狗的图片，测试集包括12500张未标识的猫狗图片。

在图形识别领域，卷积神经网络（Convolutional Neural Network)变得越来越流行。卷积神经网络是近年发展起来，并引起广泛重视的一种高效识别方法。20世纪60年代，Hubel和Wiesel在研究猫脑皮层中用于局部敏感和方向选择的神经元时发现其独特的网络结构可以有效地降低反馈神经网络的复杂性，继而提出了卷积神经网络（Convolutional Neural Networks-简称CNN）。现在，CNN已经成为众多科学领域的研究热点之一，特别是在模式分类领域，由于该网络避免了对图像的复杂前期预处理，可以直接输入原始图像，因而得到了更为广泛的应用。参加该竞赛的选手很多即采用了卷积神经网络取得很好的排名，在本项目中本人也选择使用卷积神经网络技术。

1.2 问题描述

猫狗识别项目本质上是一个监督学习二分类项目。根据每张图片的特征，图片可以被分为猫和狗。该项目的目标即建立一个合理的模型，能根据图片的特征准确地识别出图片中的动物是属于猫还是狗。
监督式学习（英语：Supervised learning），是一个机器学习中的方法，可以由训练资料中学到或建立一个模式（函数 / learning model），并依此模式推测新的实例。训练资料是由输入物件（通常是向量）和预期输出所组成。函数的输出可以是一个连续的值（称为回归分

析），或是预测一个分类标签（称作分类）。二分类问题即根据模型的输入特征把它分为两个类别，这个模型被称作分类器。
目前最广泛被使用的分类器有人工神经网路、支持向量机、最近邻居法、高斯混合模型、朴素贝叶斯方法、决策树和径向基函数分类。分类器的表现很大程度上地跟要被分类的资料特性有关。并没有某一单一分类器可以在所有给定的问题上都表现最好，这被称为'天下没有白吃的午餐理论'。在机器视觉领域，近年来大热的深度学习网络即人工神经网络的一种，取得了很大的成功。本项目作为图像识别的项目，可以用深度学习技术来解决。

1.3 评价指标

该项目作为一个二分类项目，其评价指标有很多，比如可以用准确度（Accuracy）- 即正确分类的样本数除以总样本数得出；也可以利用混淆矩阵画出 'Precision-Recall curve' 来评判模型；其他还有F-score等等指标都可以对二分类模型进行评价。
在猫狗识别项目中，Kaggle官方指定的评价标准是Log Loss：

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

其中，

- n 代表数据集中图片的数量
- $\hat{y}_i$ 代表当前图片被预测为dog的可能性
- y_i 为1当图片为dog的时候，当图片为cat的时候y_i为0
- log() 是底数为e的自然对数

Log Loss可以用来评估分类器的精度。当模型为每个类输出一个概率，而不是仅有可能的类时，就可以使用它。Log Loss是一种 "软" 的精度测量，包含了概率置信的概念。它与信息理论密切相关：Log Loss其实就是真实标签的分布与预测之间的交叉熵。简单来讲，熵即事物的不确定性。交叉熵融合了真实分布的熵，其用来衡量在给定的真实分布下，使用非真实分布所指定的策略消除系统的不确定性所需要付出的努力的大小。交叉熵越低，这个策略就越好，最低的交叉熵也就是使用了真实分布所计算出来的信息熵，因为此时 ，交叉熵 = 信息熵。这也是为什么在机器学习中的分类算法中，我们总是最小化交叉熵，因为交叉熵越低，就证明由算法所产生的策略最接近最优策略，也间接证明我们算法所算出的非真实分布越接近真实分布。通过最小化交叉熵，可以最大化一个分类器的精度。该数值越小越好。那么在该项目中我们也使用LogLoss作为评价指标。

II. 分析

2.1 数据的探索

图1. 训练集图片示例

本项目使用的数据集即Kaggle竞赛的'Dogs vs. Cats'数据集，数据集包括训练数据集和测试数据集，其中训练数据集包括25000张标签好的猫或狗的图片，测试集包括12500张未标识的猫狗图片。测试集25000张图片中，有12500张标注为cat，另外一半标注为dog，训练集的分布较为平衡。通过随机浏览方式查看了1000张标注为cat的照片和1000标注为dog的图片，未发现标注错误的情况，根据统计学原理可以大致估计训练集中标记错误的样本比例不会超过0.1%。图片质量参差不齐，文件大小大至60k小至3k的都有，有很多照片甚至是模糊的，图片尺寸也千差万别。猫狗的品种很多，在图片中的位置、数量及大小变化很大。

2.2 数据可视化

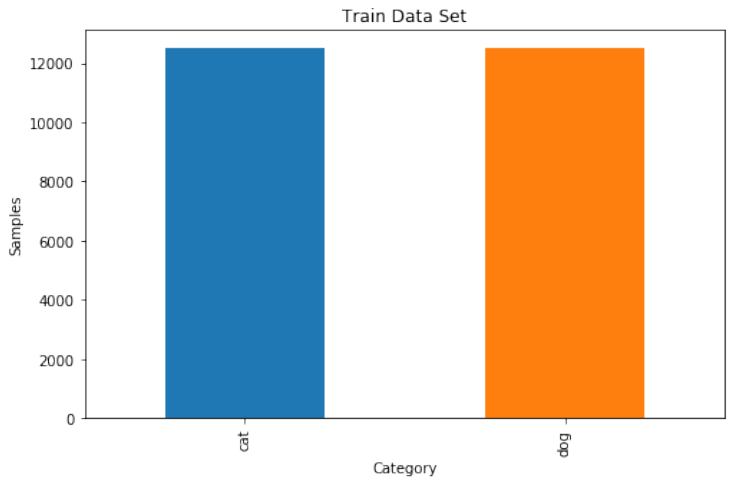

图2. 训练集中cat和dog图片的分布

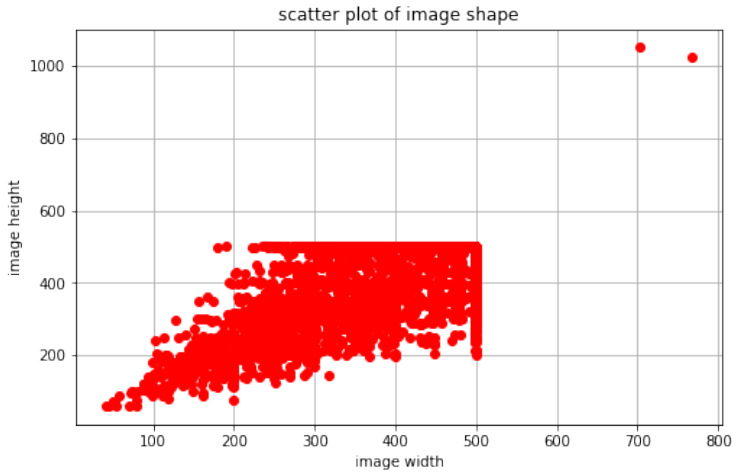

图3. 训练集中前5000张图片的shape散点图

图2中显示了训练集中cat和dog的分布，从图中可以看出，训练数据集分布很平衡，cat和dog的样本数量相等。图3是取了训练集中的5000张图片，其中包括2500张cat和2500张dog图片，图片shape分布的散点图。从图中可以看出，99%的图片在500*500以内，图像大小跨度较大，最小的图片可能才30*30，最大的图甚至可以达到1000*1000。

2.3 算法和技术

在该项目中，我们可能需要用到的工具有开源的图像处理库opencv及开源的深度学习库TensorFlow、Keras。除此之外还会用到很多前辈的发现和提出的深度学习模型，比如ResNet50、Inception、Xception等。

人工神经网络（ANNs）

人工神经网络（ANNs）是受生物神经网络启发而设计的机器学习技术。这样的系统通过样本来学习（逐步提高他们的能力）完成任务，通常没有任务特定的编程。例如，在图像识别中，他们可能会通过分析手动标记为“猫”或“无猫”的示例图像并使用分析结果识别其他图像中的猫，从而学习识别包含猫的图像。研究者发现，在使用基于规则编程的传统计算机算法难以解决的问题中，他们效果显著。人工神经网络是基于一组称为人造神经元的连接单元（类似于生物脑中的轴突）。神经元之间的每个连接（突触）都可以将信号传送给另一个神经元。接收（突触后）神经元可以处理信号，然后发出与其连接的下游神经元的信号。神经元可能具有状态，通常由实数表示，通常在0和1之间。神经元和突触也可能具有随着学习进行而变化的权重，这可以增加或减少其向下游发送的信号的强度。

神经网络方法的最初目标是以与人类大脑相同的方式解决问题。随着时间的推移，注意力集中在匹配特定的心理能力上，导致与生物学的偏离，如反向传播，或者反向传递信息，并调整网络以反映这些信息。当前人工神经网络已经用于各种任务，包括计算机视觉，语音识别，机器翻译，社交网络过滤，游戏板和视频游戏以及医疗诊断。截至2017年，神经网络通常

具有几千至几百万单位以及数百万的连接。尽管这个数量比人类大脑上的神经元数量少了几个数量级，但这些网络已经可以在很多任务上超出人类的水平。

深度学习（Deep Learning）

神经网络根据神经元连接方式的不同分为两种，前向神经网络和递归神经网络。项目中使用的卷积神经网络具体来说一种前向神经网络。前向神经网络的神经元组成一种分层的结构，前一层的神经元只与后一层的神经元相连，每一层之中的神经元没有连接。神经网络的层数又叫做深度，多层的神经网络就是深度学习名字的来源。图4为历史上几个有名的深度学习模型在ILSVRC比赛上的成绩。

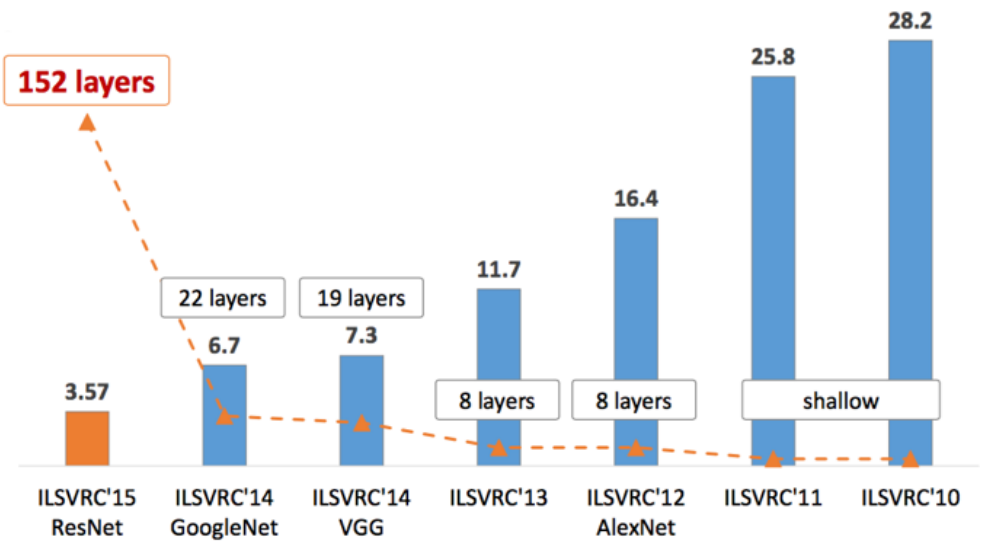

图4. ILSVRC历年的Top-5错误率

Keras

Keras是基于Python的深度学习库。它是一个高层神经网络API，由纯Python编写而成，它的后端可以是Tensorflow、Theano以及CNTK。它为支持快速实验而生，能够把你的idea迅速转换为结果，可以简易和快速地实现原型的设计，同时支持CNN和RNN，可以实现CPU和GPU的无缝切换。

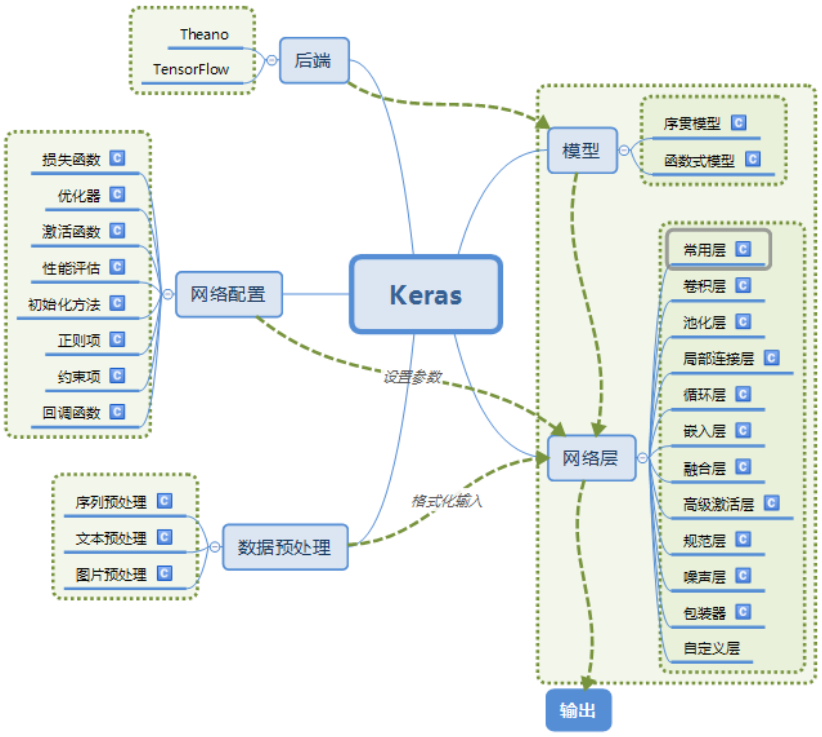

图5. Keras模块图

截止现在，keras已经在其库中集成了如下7种预训练模型。其中VGG网络遵循现在的基本卷积网络的原型布局：一系列卷积层、最大池化层和激活层，最后还有一些全连接的分类层。

MobileNet 本质上是为移动应用优化后的 Xception 架构的流线型（streamline）版本。InceptionResNetV2顾名思义就是Inception和ResNet的结合。剩下的三个可以说真正重新定义了我们看待神经网络的方式。从下表中我们可以看出这三个模型的表现，可以说是出类拔萃。在本项目中，将会使用这三类模型作为基础模型，原因有二。第一，这三个模型本身在图像识别上就表现优秀；第二，在keras能方便地调用在imagenet上训练过的预训练模型。

模型	大小	Top1准确率	Top5准确率	参数数目	深度
Xception	88MB	0.790	0.945	22,910,480	126
VGG16	528MB	0.715	0.901	138,357,544	23
VGG19	549MB	0.727	0.910	143,667,240	26
ResNet50	99MB	0.759	0.929	25,636,712	168
InceptionV3	92MB	0.788	0.944	23,851,784	159
IncetionResNetV2	215MB	0.804	0.953	55,873,736	572
MobileNet	17MB	0.665	0.871	4,253,864	88

图6. Keras库中自带预训练模型的列表

ResNet50

ResNet是2015年ILSVRC的冠军，其论文为Deep Residual Learning for Image Recognition。ResNet也是创新了网络的结构形式，引入了残差网络（residual net）。ResNet的残差结构如下：

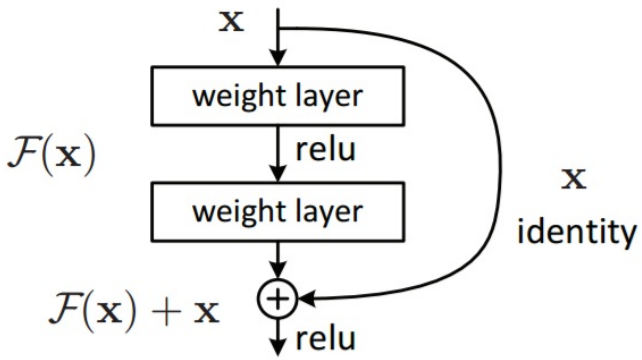

图7. ResNet的残差结构

未加残差结构时，学习映射为H(x)，但是H(x)不容易学；加上参加结构后，学习映射变为F(x)=H(x)−x，学习F(x)比学习H(x)容易，那么通过学习F(x)来得到H(x)=F(x)+x，这就是residual结构。

ResNet主要创新：

- 发现degradation problem，更深的网络准确率未必更好。
- 引入残差结构，是深层网络优化变容易，使网络更深。

InceptionV3

该模型的名字Inception来源于电影《盗梦空间（Inception）》，因为那句”we need to go deeper”的台词很契合深度学习。Inception的作者对训练更大型网络的计算效率尤其感兴趣。

在传统的卷积网络中，每一层都会从之前的层提取信息，以便将输入数据转换成更有用的表征。但是，不同类型的层会提取不同种类的信息。5×5 卷积核的输出中的信息就和 3×3 卷积核的输出不同，又不同于最大池化核的输出……在任意给定层，我们怎么知道什么样的变换能提供最有用的信息呢?

Inception 最早的论文关注的是一种用于深度网络的新型构建模块，现在这一模块被称为「Inception module」。这个模块主要的创新点如下：

提供多种信息，让模型自身来决定使用哪种信息，但这会带来计算量的陡增。

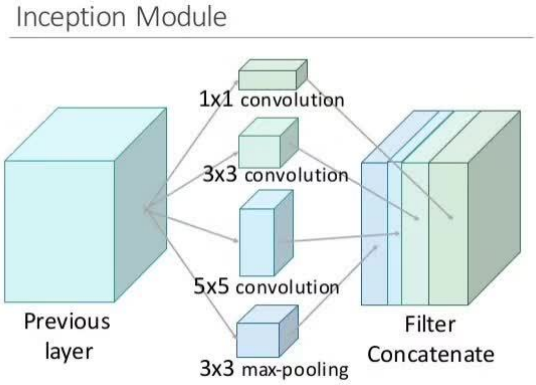

图8 Inception Module结构

使用 1×1 卷积来执行降维。为了解决上述计算瓶颈，Inception 的作者使用了 1×1 卷积来过滤输出的深度。

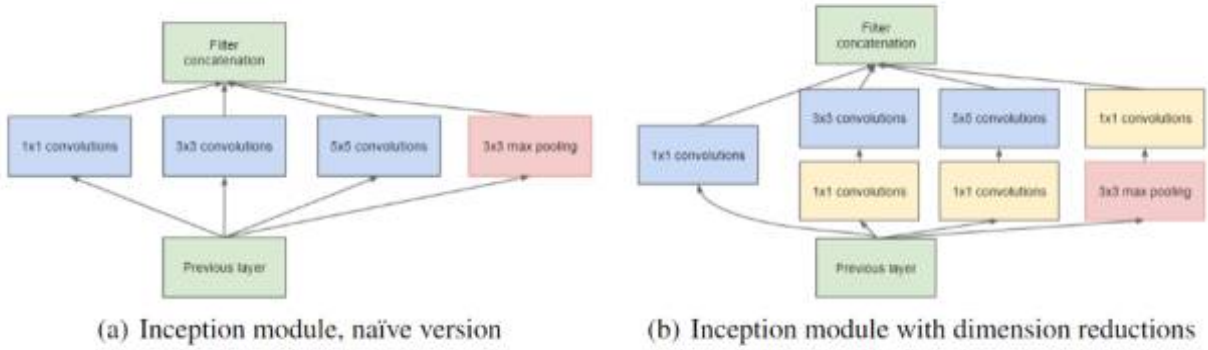

图9. 降维前后Inception Module结构对比

Inception 的第一个版本是 GoogLeNet，也就是前面提及的赢得了 ILSVRC 2014 比赛的 22 层网络。一年之后，研究者在第二篇论文中发展出了 Inception v2 和 v3，并在原始版本上实现了多种改进——其中最值得一提的是将更大的卷积重构成了连续的更小的卷积，让学习变得更轻松。比如在 v3 中，5×5 卷积被替换成了两个 连续的 3×3 卷积。

Inception 很快就变成了一种具有决定性意义的模型架构。最新的版本 Inception v4 甚至将残差连接放进了每一个模组中，创造出了一种 Inception-ResNet 混合结构。但更重要的是，Inception 展现了经过良好设计的「网中有网」架构的能力，让神经网络的表征能力又更上了一层楼。

Xception

Xception 表示「extreme inception」,它的作者也是 Keras 的作者 - Francois Chollet。该模型在2017 年 4 月才公开，而且正如其名字表达的那样，它将 Inception 的原理推向了极致。

它的假设是：跨通道的相关性和空间相关性是完全可分离的，最好不要联合映射它们。

在传统的卷积网络中，卷积层会同时寻找跨空间和跨深度的相关性。让我们再看一下标准的卷积层：

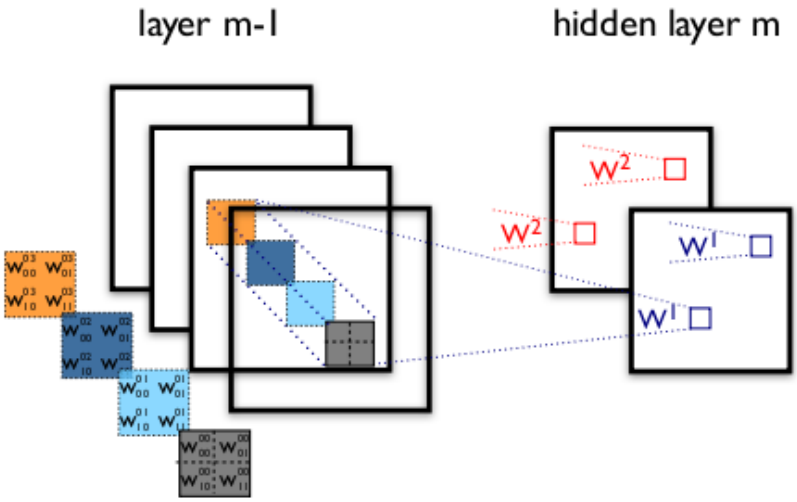

图10. 标准卷积层示意图

在上图中，过滤器同时考虑了一个空间维度（每个 2×2 的彩色方块）和一个跨通道或「深度」维度（4 个方块的堆叠）。在输入图像的输入层，这就相当于一个在所有 3 个 RGB 通道上查看一个 2×2 像素块的卷积过滤器。那问题来了：我们有什么理由去同时考虑图像区域和通道?

在 Inception 中，我们开始将两者稍微分开。我们使用 1×1 的卷积将原始输入投射到多个分开的更小的输入空间，而且对于其中的每个输入空间，我们都使用一种不同类型的过滤器来对这些数据的更小的 3D 模块执行变换。Xception 更进一步。不再只是将输入数据分割成几个压缩的数据块，而是为每个输出通道单独映射空间相关性，然后再执行 1×1 的深度方面的卷积来获取跨通道的相关性。

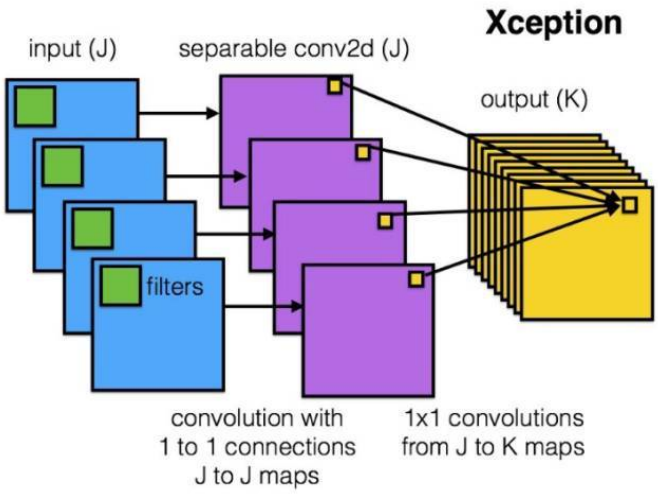

图 11. 空间维度和通道分离的卷积层

其作者指出这本质上相当于一种已有的被称为深度方面可分的卷积（depthwise separable convolution）的运算，它包含一个深度方面的卷积（一个为每个通道单独执行的空间卷积），后面跟着一个逐点的卷积（一个跨通道的 1×1 卷积）。我们可以将其看作是首先求跨一个 2D 空间的相关性，然后再求跨一个 1D 空间的相关性。可以看出，这种 2D+1D 映射学起来比全 3D 映射更加简单。

而且这种做法是有效的，在 ImageNet 数据集上，Xception 的表现稍稍优于 Inception v3，而且在一个有 17000 类的更大规模的图像分类数据集上的表现更是好得多。最重要的是，它的模型参数的数量和Inception一样多，说明它的计算效率也更高。

迁移学习（Transfer Learning）

在面对某一领域的具体问题时，通常可能无法得到构建模型所需规模的数据。然而在一个模型训练任务中针对某种类型数据获得的关系也可以轻松地应用于同一领域的不同问题。这种技术也叫做迁移学习（Transfer Learning）。

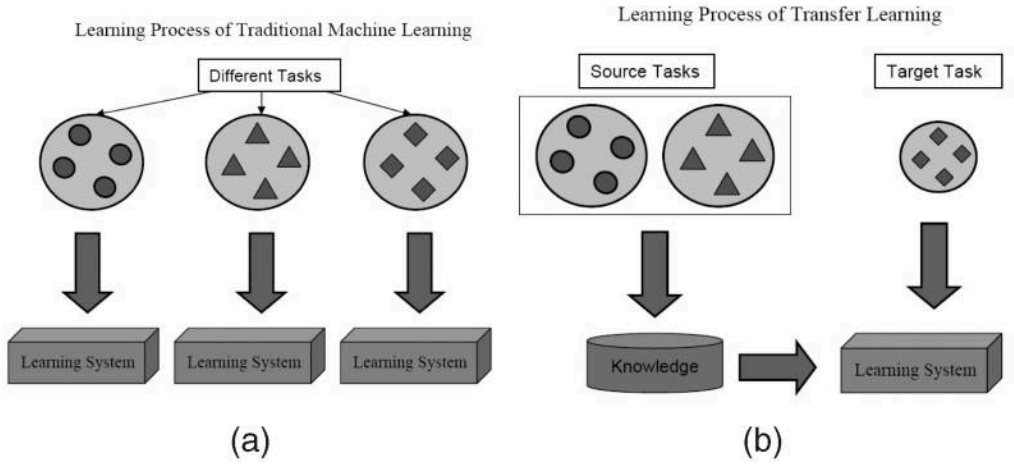

图12. 迁移学习示意图

借助迁移学习技术，我们可以直接使用预训练过的模型，这种模型已经通过大量容易获得的数据集进行过训练（虽然是针对完全不同的任务进行训练的，但输入的内容完全相同，只不过输出的结果不同）。随后从中找出输出结果可重用的层。我们可以使用这些层的输出结果充当输入，进而训练出一个所需参数的数量更少，规模也更小的网络。这个小规模网络只需要了解特定问题的内部关系，同时已经通过预培训模型学习过数据中蕴含的模式。通过这种方式，即可将经过训练检测猫咪的模型重新用于再现梵高的画作。

模型融合

模型融合是kaggle等比赛中经常使用到的一个利器，它通常可以在各种不同的机器学习任务中使结果获得提升。顾名思义，模型融合就是综合考虑不同模型的情况，并将它们的结果融合到一起。模型融合主要通过几部分来实现：从提交结果文件中融合、stacking和blending。

模型微调（fine tune）

所谓fine tune就是用别人训练好的模型，加上我们自己的数据，来训练新的模型。fine tune相当于使用别人的模型的前几层，来提取浅层特征，然后在最后再落入我们自己的分类中。fine tune的好处在于不用完全重新训练模型，从而提高效率，因为一般新训练模型准确率都会从很低的值开始慢慢上升，但是fine tune能够让我们在比较少的迭代次数之后得到一个比较好的效果。在数据量不是很大的情况下，fine tune会是一个比较好的选择。但是如果你希望定义自己的网络结构的话，就需要从头开始了。因此fine tune常常跟迁移学习搭配使用。

2.3 基准指标

该项目在Kaggle上目前有1314组参赛者，其中成绩最好者的成绩是0.03302，最差的成绩是19.45806，Top10%的划分线是0.06127，Top5%的划分线是0.05038。在该项目上定的目标是保证进入Top10%，争取进入Top5%。

III. 方法

3.1 数据预处理

观察该项目的数据集可以发现，数据是一堆包括猫狗的图片，图片质量参差不齐，图片内容也千差万别，甚至有不包含猫狗的图片混在其中。图片以其类别命名，命名具体方式为type.num.jpg，比如cat.1001.jpg。

那么在数据准备阶段要对数据进行预处理，包括如下工作：

首先，要从图片的名称里提取出训练集的标签，其次要要把图片读取进来，考虑到数据图片大小不一，要把读取进来的图片resize成统一的大小，大小根据后面的模型来决定。

训练集需要经过随机打乱，分成两部分，一部分用来训练模型，另外一部分用来验证模型的分类性能。本项目中，选择了20%的图片作为验证集，即5000张图片的验证集。使用软连接方法，设置了文件夹validation和文件夹train2。软连接的优势在于不用复制图片，节省存储空间。

项目数据集有25000张图片用于训练，把这25000张图片读入系统需要内存太大，不太现实。所以在该项目中，我们会使用生成器（generator）来给模型喂数据。Keras 自带有ImageDataGenerator，可以很好地实现我们的目的。ImageDataGenerator可以设置数据源和读入图片的大小，以适应不同模型的需求，大大减小了工作量，这就是Keras的优势。

3.2 模型实现

在模型实现过程中，使用了迁移学习技术，把Keras库里自带的预训练模型中的ResNet50、InceptionV3以及Xception迁移过来了。迁移过来的不仅仅是模型，更重要的是模型的参数，这样就不用去重新去训练模型的参数了。
不过，上面提到的几个模型都是在ImageNet上训练的过的多分类模型。因此，模型拿过来我们还不能直接使用，需要对顶部的几层重新构造，并经过训练，得到最终模型。模型训练结果如下表。

模型	训练集Loss	训练集Acc	验证集Loss	验证集Acc
Xception	0.0608	0.9799	**0.0389**	**0.9880**
ResNet50	0.0903	0.9657	0.0744	0.9734
InceptionV3	0.0988	0.9666	0.0739	0.9782

表1. 模型初始的训练和验证表现

通过表1我们可以看到单独训练的三个模型的表现。就单个模型而言，Xception的表现最佳，其验证集的loss下降到了0.0389，准确率也达到0.988。ResNet50与InceptionV3表现相当，loss降到0.074左右，准确率也都达到0.97。可见这些在ImageNet数据集上预训练过的模型泛化能力还是很强大的。

接下来，我们对以上三个模型按不同组合进行模型融合，融合后的模型训练结果如下表。

模型	训练集Loss	训练集Acc	验证集Loss	验证集Acc
Xception+ResNet50	0.0874	0.9691	0.0507	0.9816
ResNet50+InceptionV3	0.1329	0.9598	0.0788	0.9738
InceptionV3+Xception	2.3762	0.6191	1.1359	0.7064
Xception+ResNet50+InceptionV3	0.1670	0.9585	0.0881	0.9768

表2. 初始模型融合后的训练和验证表现

观察模型融合后的结果表2可以说有点让人大跌眼镜，融合后的模型表现反而不如单个模型的表现。表现最佳的融合组合是 ResNet50+Xception ，验证集loss降到了0.0507，准确率达到0.9816。即使是最好的组合，其效果也还不如单个Xception。表现最差的组合是InceptionV3+Xception，验证集loss在1.1359，验证集准确率竟然只有0.7064，组合后的模型与单个模型比较，效果出现断崖式下降。同时融合三个模型也没有效果。从以上结果可以得出一个结论，那就是模型融合并不一定会提高模型的能力。

3.3 模型改进

在上一节中，使用迁移学习（transfer learning）技术，把Keras库中自带的三个在ImageNet上预训练过的模型迁移过来。以上三个模型中单个模型的表现都较好，但融合的效果不尽人意。但是即使是单个模型中表现最好的Xception的效果，跟项目的目标相比还是差的较远。因此，在此节主要使用模型微调（fine tune）技术对模型进行微调。

首先是ResNet50模型，分别尝试对ResNet50放开5层、8层、11层、15层进行训练，发现ResNet50能达到的最好效果就是验证集loss在0.06上下，准确率在0.982左右，在模型放开8层或11层的时候能达到。

其次是Xception模型，也对Xception尝试放开各种层，最后发现竟然是把所有层全放开进行训练能达到最好效果。验证集loss下降到了0.0296，准确率达到0.9942。把所有层都放开后，由于参数很多，训练时间较长每轮的训练时长达到30分钟，整个训练下来用了2.5小时。

最后是InceptionV3模型，最后发现InceptionV3在放开22层后进行训练，验证集loss下降到了0.0390，准确率达到0.9860。

综合看表3三个模型的微调结果，ResNet50、Xception、InceptionV3的验证集分类准确率分别获得了0.9、0.54、0.78的提高。提高效果显著，比上节的模型融合效果要好很多。

模型	训练集Loss	训练集Acc	验证集Loss	验证集Acc
Xception_fine_tune	0.0014	0.9996	0.0296	0.9942
ResNet50_fine_tune	0.0109	0.9963	0.0650	0.9824
InceptionV3_fine_tune	0.0219	0.9928	0.0390	0.9860

表3. 模型经过微调后的训练和验证表现

经过fine tune后的Xception其实表现已经不凡，接下来我尝试了用微调后的模型来进行模型融合，集把微调技术和模型融合技术结合起来使用。表*显示了不同组合模型融合后的训练和验证表现。

观察表4可以看出，经过微调之后的模型融合效果非同一般。Xception+ResNet50组合验证集loss竟然降到了0.0032，Xception+ResNet50+InceptionV3组合的验证集准确率达到0.9988。模型在融合后有了质的提升。

模型	训练集Loss	训练集Acc	验证集Loss	验证集Acc
Xception+ResNet50	0.0038	0.9990	**0.0032**	0.9984
ResNet50+InceptionV3	0.0218	0.9956	0.0248	0.9948
InceptionV3+Xception	0.0071	0.9986	0.0133	0.9980
Xception+ResNet50+InceptionV3	0.0078	0.9987	0.0082	**0.9988**

表4. 微调模型融合后的训练和验证表现

IV. 结果

4.1 模型评价与验证

经过几十次不断的尝试，最后终于锁定了最终的模型的范围。从验证集的数据结果来看微调后的Xception+ResNet50组合及微调后的Xception+ResNet50+InceptionV3组合都有可能取得最好成绩。因此，分别使用两个模型对测试集进行预测，预测的结果提交到Kaggle得到最终得分。最后Xception+ResNet50+InceptionV3组合的得分是0.04961，这个成绩在Kaggle排行榜上的排名在58位，已经在Top5%以内；Xception+ResNet50组合的得分是0.04278，在Kaggle上的排名在22位，相当于已经进入了前2%，该成绩已经远远超出一开始定的目标。图13为模型在Kaggle上的评分。

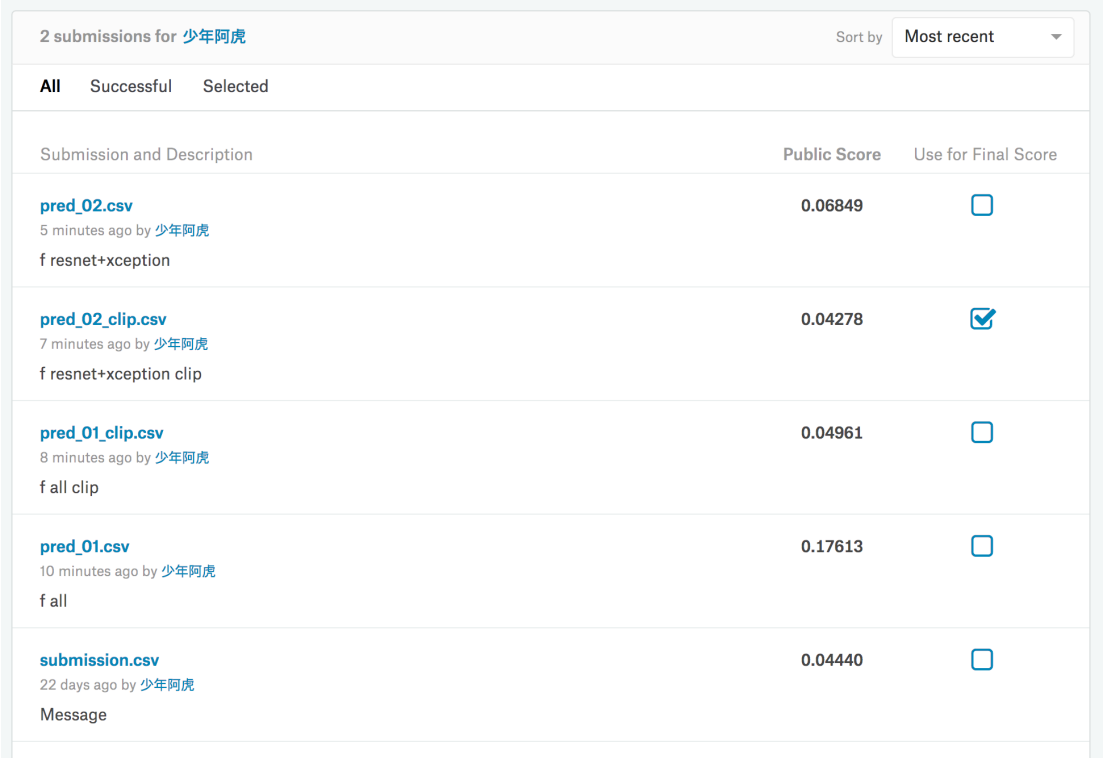

图13. Kaggle Public Score

4.2 结果分析

在训练完成得到最优的模型之后，我从测试集中挑选了24张比较具有代表性的图片，将其和预测的结果绘制成图表，如图*所示，图片上方是图片名称加模型输出的概率值，显性表示为狗的概率，比如第一张图为狗的概率是0.5%，那么为猫的概率为99.5%。在图*中大部分是模型分类容易混淆的图片，也包括人类很容易识别但模型分类错误的图片。

首先看126.jpg、144.jpg、489.jpg、1479.jpg、1496.jpg及2903.jpg，这几张图片有个共同特点就是图片特征不完全或者模糊，我认为机器学习比较难识别的图片，但是在识别的结果里这几张图片的预测得分是很高的。可见模型的鲁棒性还是很好的。

下列图片中有几张图片是分类错误的图片，比如32.jpg很明显是条狗，但模型给出是狗的概率是0.5%，这是一个特例，我猜测是因为这只狗的姿势摆得太想猫了，模型提取到了猫的特征，因此给出是猫的概率为99.5%。460.jpg、675.jpg及1016.jpg人眼很明显能识别出来，但模型没能识别，尚不知道原因。

图片571.jpg和2420.jpg代表了一类图片，那就是图片中同时有猫和狗，我不知道这类图片测试集中的标签是什么，但是很显然无论模型预测是猫还是狗，都是准确的。但是当模型同时提取到猫和狗的特征的时候，它只能输出模棱两可的概率，比如2420.jpg。我觉得这种测试图片会影响模型的最后得分。

图14. 从测试集中出的24张比较特殊的图

2085.jpg、3086.jpg和3735.jpg代表的是另一类图片，这类图片的特点是有一堆目标物堆在一起，给模型提取特征应该造成了一些困难。模型给出的概率都不是很确切，说明模型对这些图片的识别还是存在一些困难的，但对人类来说，这些图片的识别其实还是比较简单的。模型的提高可能可以尝试从这些图片上下点功夫。

V. 项目结论

5.1 对项目的思考

在猫狗识别项目中，大致实施过程可以分成如下五步：

第一步，数据的准备和预处理。在该过程中，我人工浏览了上千张训练图片和测试图片，观察了样本的特点，并用可视化的方式探索了样本的分布和样本shape的分布。项目给定的数据都是图片文件，其中分为训练集和测试集，训练集是有标签的，标签即其文件名。考虑到数据集的大小，一次载入会导致内存溢出，项目中选择使用keras自带的图片生成器（ImageDataGenerator）来分批给模型输入数据。利用系统的软连接，按照图片生成器的规范设置好数据文件夹，方便使用 '.flow_from_directory(directory)' 来分批生成输入数据。

首先把图片和标签转化成模型训练和测试所需要的array和list。其次，把训练集随机分成训练集和验证集，验证集用来验证模型的分类效果；

第二步，利用TensorFlow、Keras 载入预训练模型。项目中选择了ResNet50、InceptionV3、Xception三个在“ImageNet”上训练过的预训练模型。因为这三个模型本身都是多分类模型，因此对模型的TOP几层稍作修改，变成二分类模型。利用图片生成器给模型输入数据，训练了修改后的几层。从模型的训练效果看，成绩最好的是Xception，其次是InceptionV3，最后是ResNet50。但总体来说，这三个模型的效果都不错，可见该类模型的泛化能力很好，毕竟是在大数据集上训练过的，较适合用于迁移学习（Transfer Learning）。

第三步，使用模型融合技术。尝试利用三种模型的不同组合方式来进行模型融合，得出各种组合的表现。

第四步，使用fine tune技术来微调模型，在微调后的模型基础上再重复第二和第三步，比较模型表现，找到最佳效果组合，以此为最终模型。

第五步，利用最终模型预测test数据集，结果提交到kaggle。

因为工作原因，项目历时快2个月，在项目完成过程中，遇到很多挑战，也有很多有意思的事情发生。在刚拿到项目的时候，根本没有头绪，不知道从哪下手，因为跟P5来比较的话，解决这个问题需要的知识空缺太多了，难度上绝对是一个飞跃。没办法，只能硬着头皮上。于是疯狂地利用业余时间，查资料，看keras文档，看论文。

在这个过程中，接触到了很多在图像识别领域很著名的模型，并了解了其特性和在历史中取得的成绩。学习到了迁移学习，这个技术让我豁然开朗，原来这些在大数据上训练的模型还可以这么用，让我跳出了自己用keras一层一层搭建模型的思路。事情变得有趣起来，我开始使用迁移学习技术，很快就使得模型的准确率达到90%以上。我接着尝试不同模型的迁移学习，其中Xception模型甚至可以在验证集上达到98%的准确率。很显然，这些在大数据集上预训练的模型，已经能很好地提取图像的特征，并且能在通用的场景下解决该类图像识别的问题。

但是我遇到的问题是即使是准确率最高的Xception的迁移过来，能达到的最好成绩也不能达到我设定的目标。为了能改进模型，我决定对几个表现较好的三个模型进行微调（fine tune），微调后，这几个模型的准确率都有所提高，其中Xception的最高准确率达到了99.42%，效果显著。接下来，我还尝试了模型融合技术，依次对以上提到的三个模型按不同组合进行融合。融合使得模型的准确率进一步提高，最终的不仅达到设定的目标，而且超过冲刺目标。

通过使用迁移学习+模型融合+模型微调的技术组合，很完美地解决了猫狗分类问题，也可以得出结论，该技术组合同样能有效地利用到其他深度学习问题当中。

5.2 需要作出的改进

在前面我们提到过，有的类型的图片对模型来说其分类的准确率并不是很高。我们可以尝试在对数据做一些数据增强，提高模型的泛化能力。在模型数据分析一节我提到过输入数据的异常值比例小于0.1%，但是查资料发现，该数据集还是存在一些异常值的，也许可以尝试进一步剔除数据集中的异常值。

图15. 知乎作者使用深度学习模型挑出的异常图片

以上是对训练数据的一些处理，另外一方面我们还可以尝试在模型上进一步调整，使用效果更好的模型，比如InceptionResNetV2、DenseNet、NASNet等，或者也可以尝试自己用Keras搭建更先进的模型进行训练。
最后，可以尝试把训练好的最终模型，通过相关技术做成实际的应用，比如使用Flask来制作网页应用，或者直接使用 Apple 提供的 Core ML Tools 把训练出来的 Keras 模型直接转为 iOS 可以使用的模型。

参考资料：

[1] Chollet F. Xception: Deep Learning with Depthwise Separable Convolutions[J]. 2016:1800-1807.

[2] Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the Inception Architecture for Computer Vision[J]. Computer Science, 2015:2818-2826.

[3] He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition[C]// Computer Vision and Pattern Recognition. IEEE, 2016:770-778.

[4] 吴军. 数学之美.第2版[M]. 人民邮电出版社, 2014.

[5] CyberRep. 知乎. https://www.zhihu.com/question/41252833/answer/195901726

[6] https://en.wikipedia.org/wiki/Deep_learning

[7] https://zhuanlan.zhihu.com/p/34068451

[8] https://www.zhihu.com/question/41979241

[9] https://www.cnblogs.com/52machinelearning/p/5821591.html

[10] Keras中文文档. http://keras-cn.readthedocs.io/en/latest/

[11] http://www.sohu.com/a/166062301_465914