机器学习工程师纳米学位毕业项目

猫狗大战

董进贤

2018年4月3日

# 问题定义

## 项目概述

本项目源于kaggle竞赛，解决的问题属于计算机视觉智能的研究领域。本项目要求训练出的模型，在给定的图片中识别出猫狗，并且达到指定的识别率。为方便标准的评估，所涉及的数据集亦是源自kaggle的原始数据集。这个数据集包含25000张已作出标记的用于训练的图片，其中猫狗各占比一半。另外，还有12500张的未标记图片，为测试数据集。

卷积神经网络(Convolutional Neural Network，CNN)，以其共享超参等方式，极大的提高了传统神经网路的训练效率，目前是图像识别领域公认的解决办法。所以我们的模型也以CNN为实现。这种共享超参数，作为卷积核与输入数据作用计算获得结果的方式，等效于矩阵中的卷积运算，故这个神经网络如此得名。

## 问题陈述

数据集中的图像均是来自于现实世界中未经过处理的图像，大小不一，图像中的场景也各不相同，另外具体到猫狗的姿态也有非常大的差异，其品种更是各不相同。相较于传统的机器学习算法，实在难以处理此类问题（比如，我们知道在历史上，第一个成功的卷积神经网络LeNet-5虽然成功完成了数字识别，但在综合表现上反而被后来的SVM所超越）。

虽然我们最终的CNN是经过优化处理，增加了深度的模型。但是归结起来，我们还是采用了四个步骤：一是输入阶段，预处理数据到方便训练的形态；二是卷积层训练，以提取数据在不同通道不同向量方向上的特征数据；三是池化筛选特征数据，用作下一层的卷积训练或者分类处理；最后则是全联接综合所有的维度特征作出最终的分类预测。

## 评价指标

评价指标采用kaggle官方的评价公式，其得分值越低，效果越好，公式如下：

其中：

1. n为测试集中图片的数量；
2. 为猜测图片是dog的概率；
3. 为1时图片为dog，为0时图片为cat；
4. 为自然对数。

目前CNN计算较为完善，各个模型的预测率均较高。由公式计算出的对数损失评价预测效果，而非正确率是合理的。而且在模型在过分肯定预测结果时，在公式的计算下，也不一定得到较好的结果。这点能够发现过度拟合的模型，评价更为客观合理。

# 分析

## 数据的探索

我们采用的数据集来自kaggle，下载后得到三个文件，test.zip、train.zip和sample\_submission.csv。其中test.zip为测试数据集，大小为271MB，train.zip为训练数据集，大小为544MB，sample\_submission.csv应该是提交样例。解压压缩包，我们发现数据集中均是jpg图片，命名格式为“类型.序号.log”的形式，如：cat.1.log，dog.1.log。

进一步探索数据，发现训练集包含25000张图片，猫狗各半12500张，测试集12500张图片（测试集图片只有序号，是没有标记的）。浏览图片，发现图片的大小也是不一致的，清晰度，分辨率也有差别。打开图片，发现图片中的场景各异，而且有错误标注的图片如图1所示。



1. 错误标注

这些错误标注的数据，如果是train中的肯定是需要通过一定的方式进行删选清理的，否则会影响到模型的训练成果。数据集相对较小的情况下，可以直接进行人工清理。而数据量非常大的情况下，我们可以通过预处理模型，先进行简单的预测处理。再根据预测结果，对于筛选后的图片再进行人工判断是否清理。

预处理模型可以大大提高我们清洗数据的工作效率，但是为了更好的训练结果，以及预处理模型本身的识别能力，最终的人工确认，仍然是比不可少。所以有说法，人工智能的数据标注是劳动密集型产业。

另外图片中的猫狗也是形态各异，也有其它主体的干扰，这些都增加了识别难度，如图2所示。示例中也可以看出，图片大小尺寸，也有不小的差异，其中dog.9076.jpg甚至还包含大量的留白。



1. 形态背景各异

## 探索性可视化

在这里我们首选对数据集的大小特性作一定的分析，通过散点图绘制，能够比较直观的发现其中的规律。首先我们对dog和cat的图片，分别以宽为x坐标，高为y坐标绘制了散点图如图3所示。



1. 数据集原始尺寸分布

我们发现，数据集中图片的尺寸，大部分都集中在500以下，但是dog和cat的数据集中，均存在异常的离散点。很显然这个数据属于异常值，并不利于模型的训练，需要去除。

经过排查处理，我们发现，异常的图片分别是dog.2317.jpg和cat.835.jpg。我们将这两张异常的图片删除后，重新统计分析数据集后，绘制散点图如图4所示。



1. 清洗异常图片后尺寸分布

## 算法和技术

猫狗识别问题属于机器视觉的领域，具体到本项目涉及的算法和技术，需要说明的核心概念是，深度学习、卷积神经网络和迁移学习。其中迁移学习的模型，包括InceptionV3、Xception和Inception ResnetV2，后续也会做一定的说明和阐述。

### 深度学习

深度学习的概念源于人工神经网络（Artificial Neural Network，ANN），深度学习通过组合低层特征形成更加抽象的高层表示（属性类别或特征），以发现数据的分布式特征[1]。深度学习是相较于传统机器学习的叫法。普通机器学习通过人工经验抽取样本特征，模型学习后获得单层特性[2]。深度学习通过对原始数据进行逐层特征提取变换，将样本空间的特征表示变换到新的特征空间，自动地学习得到层次的特征表示，从而更有利于分类或特征的可视化[3]。

说道深度学习，必须要理清深度神经网络（Deep Neural Network，DNN）的概念。深度学习所得到的模型结构，包含大量的神经元（感受器、回归函数），每个神经元与大量的其他神经元相互连接，神经元连接之间传递信号的强度（权重），会在学习训练的过程不断修正。这种深层的网络结构符合神经网络的特性[4]，因而命名深度神经网络（Deep Neural Network，DNN）。

### 卷积神经网络

卷积神经网络（Convolutional Neural Network，CNN）是人工神经网络（Artificial Neural Network，ANN）中的一种。CNN的基本结构有输入层、卷积层、取样层、全连接层和输出层构成。卷积层和取样层一般会有若干个，采用卷积层和取样层交替设置。由于卷积层中输出特征面的每个神经元与其输入进行局部连接，并通过对应的连接权值与局部输入进行加权求和再加上偏执值，得到该神经元的输入值，该过程等同于矩阵的卷积过程，故该类神经网络由此命名[5]。

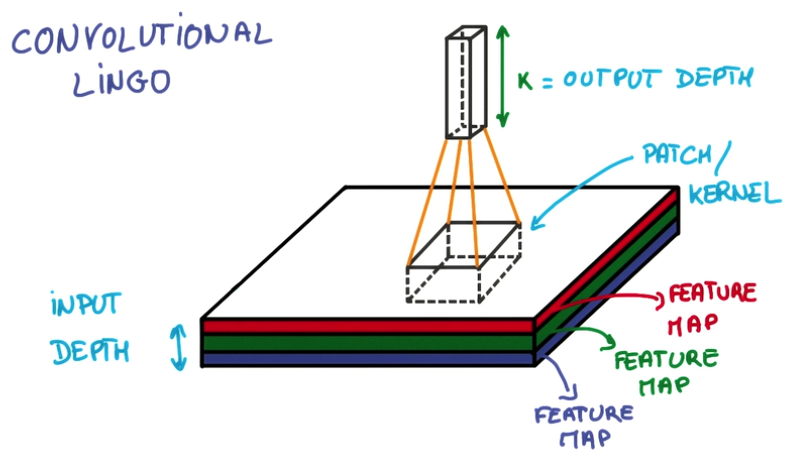
在CNN结构中，深度越深，特征面数目越多，则网络能够表示的特征空间越大、网络学习能力也越强，然而也会使网络的计算更复杂，极易出现过度拟合现象。因而，在实际应用中，需要适当选取网络深度、特征面数目、卷积核的大小及卷积时滑动步长，以在较短的时间和计算量下，训练出更好的模型。

目前CNN在图像分类和识别中应用广泛，几乎是深度学习中，默认的处理方法。本项目也采用CNN进行猫狗的分类识别处理。以下，我们会对卷积神经网络中的关键技术点，作进一步的阐述说明。

#### 卷积

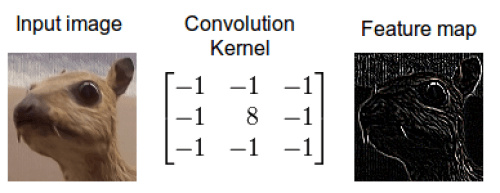
神经网络中的卷积，就是数学上的矩阵卷积计算。含有这种处理方式的神经网络也叫做卷积神经网络。卷积在简化信息处理方面有相当好的效果。

具体到本项目对于图像的处理，我们以第一个卷积层为例。输入图像在RGB三个通道上，以patch为卷积核进行操作，输出深度为k并且缩小了长度和宽度的另一个图像。这个特征提取的过程，正是深度学习中，卷积层所做的工作，如图5所示。



1. 卷积层示例

卷积操作在图片的处理的前几层，往往有一定的物理意义。也可以做一定可视化展示。比如我们常常对图片进行的一系列操作，如锐化（Sharpen）、模糊（Blur）、边缘增强（Edge Enhance）、边缘检测（Edge Detect）、凸印（Emboss）等。总之选用不同的卷积核，进行操作，就产生不同的效果，如所示，展示了边缘检测（Edge Detect）的卷积效果。

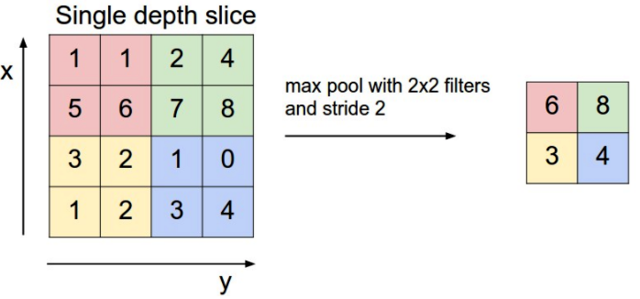


1. 卷积效果展示

#### 池化

在深度神经网络中，池化层通常是出现在卷积层之后，主要作用是进一步降低来自卷积层输出的参数数量。通常有平均池化（Average Pool）和最大池化（Max Pool）的做法，前者对于图片的背景保留较好，而后者对于图片的纹理特征保留较好。

如图7所示，展示最大池化的效果。通过获取特定区域的最大值，将维数据后，我们得到新的矩阵数据。



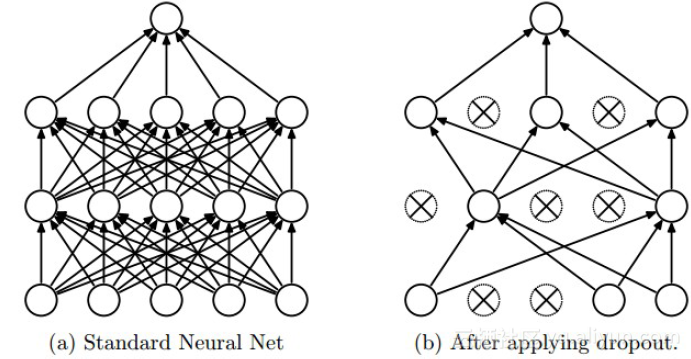
1. 池化效果展示

#### 正则化和dropout

正则化技术并不特定属于卷积神经网络（CNN），但是在神经网络的构建中，其是非常重要的一环。常见的正则化技术有：数据增强、L1和L2正则化、Dropout、Drop Connect和早停法等。

简单的讲，正则化技术保障了算法模型的泛化能力。它以增大训练误差为代价，从而减少了测试误差。

下面说明其中的一种，Dropout。这种正则化处理是，在模型训练时随机的让隐藏层部分节点的权置0，效果等同于舍弃了部分神经元。如图8所示，其简单展示了处理过程。



1. Dropout正则化

### 迁移学习

CNN的核心在于对于图像进行特征提取（浅层的特征如边缘、纹理等），随着神经网络的层数越高，抽线层次也越高。最后的全连接层再基于训练中得到的特征数据，进行分类鉴别，得到最终的分类结果。

从最原始的图像开始训练神经网络，得到超参和特征数据，需要大量的数据，及大量的计算资源。好在猫狗分类问题，已经拥有可以利用的现有模型和预训练权重。通过此，我们可以直接提取到特征数据，在特征数据上附加以分类器，就可以得到非常优良的结果。

在本项目中，我门选用的预训练模型有三个InceptionV3、Xception和Inception ResnetV2。其中我们对InceptionV3作深入讨论。

Xception模型共136层，该文档中不宜展示，可见附件notebook/model\_xception.png。图中对于模型结构，作了清晰的可视化展示。

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。这个过程实际上是加宽了网络。这个模型也输出2048维的特征向量。

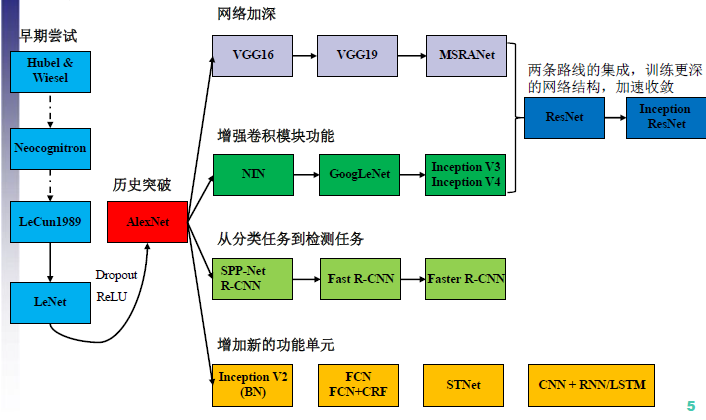
Inception ResnetV2模型共784层，该文档中不宜展示，可见附件notebook/ model\_inception\_resnet\_v2.png。图中对于模型结构，作了清晰的可视化展示。

Inception ResnetV2模型的诞生源于ResNet和GoogleLeNet，是早期的InceptionV3演化而来，其具有相当高的复杂度。该模型中存在shortcuts，使得我们能训练出更深的神经网络，从而提高识别率。

#### InceptionV3

InceptionV3模型共315层，见附件notebook/model\_inceptionv3.png。图中对于模型结构，作了清晰的可视化展示。该模型要求输入图片的分辨率为299\*299，模型的改进主要是增强了卷积模块功能。网络深度也有一定程度的增加。

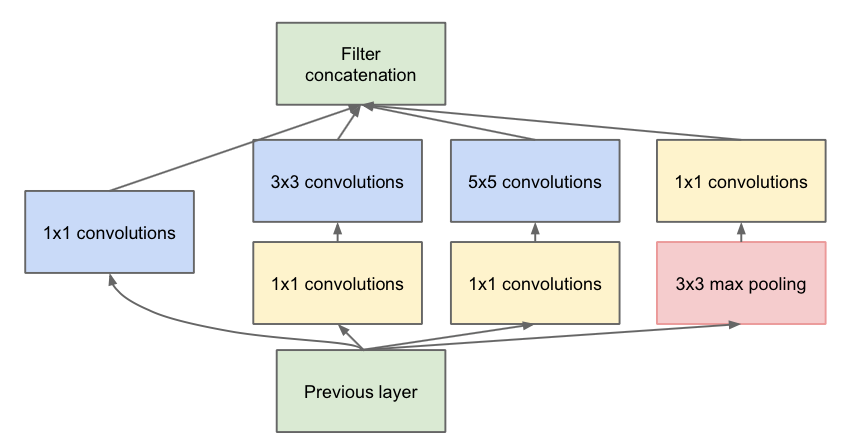
为了更清晰的理解，我首先了解一下CNN的演化，如图9所示。



1. CNN演进简图

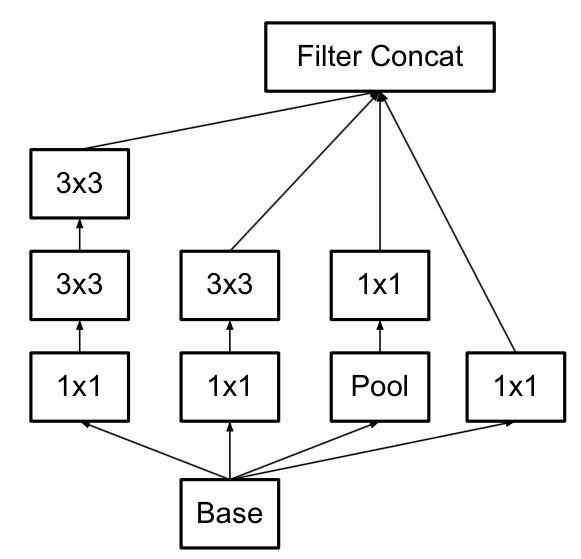
从图中，我们观察到InceptionV3这个演进分支，主要是增强了卷积模块功能。之前，神经网络的改进，主要是增加网络深度即层数，以及增加神经元的数量。但是简单的增大神经网络规模，会导致主要三方面的问题。一是，参数过多，当训练集数据有限时，容易发生过度拟合；二是，巨大的计算复杂度，导致难以应用；三是，过深的网络模型，容易梯度弥散，难以调优。而Inception类的网络模型，解决这些问题，主要是通过在增加网络规模的情况下，控制参数的增加。

比如第一代的Inception模型，增加了1\*1的卷积核，降低了特征图的厚度。通过次降低计算量，如图10所示。



1. InceptionV1主要改进

另一方面，inception也通过替换5\*5的卷积核为3\*3的卷积核，进一步降低参数数量，如图11所示。



1. InceptionV2主要改进

而我们使用的第三代Inception，进一步延续了这种改进的思路。将部分层中7\*7的卷积核分解为1\*7和7\*1。这样加速了计算后，又将1个conv分解为2个，增加了网络的深度。同时将输入由244\*244调整为299\*299。

## 基准模型

我们需要获得kaggle前百分之十的排名（即131名以前），也就是对数损失得分需要小于0.06。由于迁移学习的效果，我们应该会得到远远优于此的得分。

# 方法

## 模型筛选

模型筛选的过程，相似于我们后面要介绍到的异常值清理。对于已经有的模型，我们先单独进行训练，并根据训练的结果，预测我们的数据。对于结果较好的模型我们则优秀挑选使用。具体各个模型的预测结果，在3.3.4章节会有展示。

另外，我这次挑选有所区别的是，我也会根据模型使用简洁性，来挑选最终的模型。比如ResNet50模型完全就有较好的能力。但是最终选择的时候，由于我觉得统一处理299\*299的图片，较为方便，就没有选择。但是在预处理阶段，我却使用它进行初步的数据清洗。

## 数据预处理

数据的预处理解决方案，我们分两个方面进行：一是数据集本身的规划处理，包括图片文件的目录归类，异常值的清洗；二是针对各个模型对于输入数据的不同要求进行预处理。

首先，原始数据为两个压缩文件，test.zip和train.zip，分别是测试集和训练集数据。解压后图片分别存在于文件夹test和train。然后创建train\_bad文件夹，稍后用于存放删除的无效图片文件，可以将这个文件夹理解为垃圾桶，放在其中的图片文件，以后我们并不会再使用。

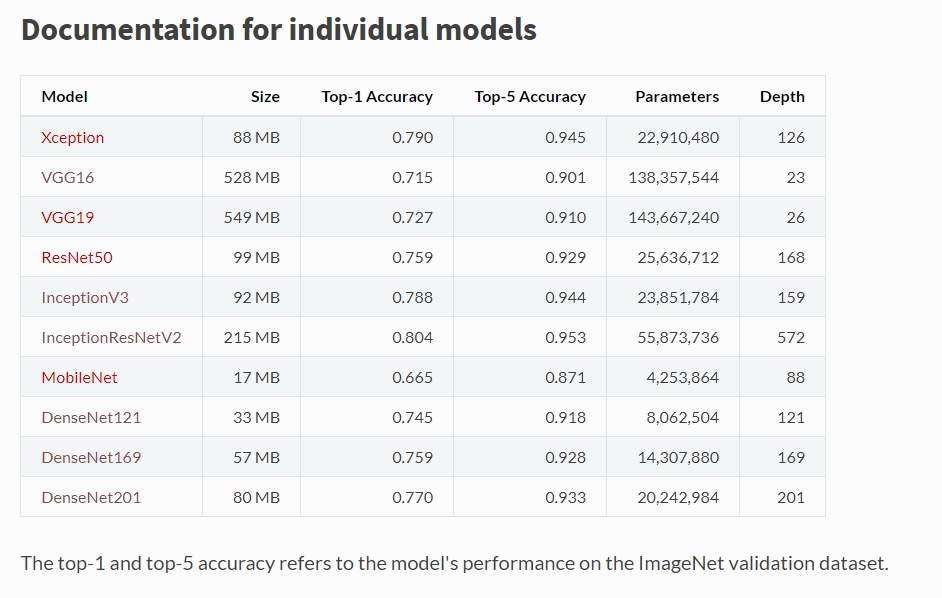
### 图片目录规划

由于我们后面会用到keras中的ImageDataGenerator函数进行数据扩充处理，对于不同种类的图片，我们需要放在不同文件夹中。这里我们只是需要在train目录下，在建立两个文件夹cat和dog，分别存放猫狗的图片即可。关于图片的目录结构和涉及到的shell脚本，见附件中的README.md文件。

### 异常值清洗

这里对数据集数量较小，对于清洗错误标记我们采用预训练模型，先对数据集进行一次预测。然后根据评分结果进行二次筛选。

在imagenet的预处理模型中，狗的分类一共118种，猫的分类一共7种。各个模型对于类型识别的准确率如图5所示。



1. 类型识别准确率

具体的，我们在操作过程中，我会对训练集数据进行预测处理。对于不属于这125类的图片初步筛选。过滤后再进行人工确认。具体的操作过程见附件部分的pick\_bad\_pics.ipyb文件。

### 各个模型下的预处理

由于我们选择的预训练模型，对于输入图片的要求都是299\*299，所以在读取图片到内存的过程中，我门调整图片的尺寸大小为299\*299。这个过程不做持久化处理，保存在磁盘上的图片文件，依然是原始大小。

另外需要说明的是，我们采取的标签是0和1，分部代表猫和狗。这里需要识别的类型并不繁多。我们简单的设置标签即可，没有另行定义归一化函数。

整个处理过程，分别由函数load\_train\_data和load\_test\_data完成，至此完成数据的预处理过程。

## 执行过程

对于完成了预处理的数据，我们通过sklearn的train\_test\_split方法，将数据随机拆分成训练集和测试集（验证集）。为了做最后的陈述分析以及模型的展示分析，我们首先分别使用三个预训练模型训练处理，然后预测结果，分别保存至对应的csv文件中。

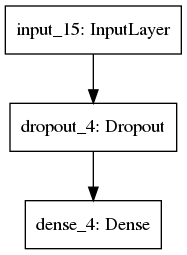
### 导出特征向量

对于InceptionV3、Xception和Inception ResetnetV2三个模型的处理方式，基本一致。估计我们定义函数pick\_features完成此项任务。我们使用model.predict配合原始数据，生成特征向量。GlobalAveragePooling2D 将卷积层输出的每个激活图直接求平均值。提取出的特征向量，实际上就是numpy的三个数组数据，分别保存到feature\_Xception.h5、feature\_InceptionV3.h5和feature\_InceptionResNetV2.h5三个文件中。

每个模型的特征向量导出，都相当耗时，附件的notebook中，我们对每一个导出过程放在单独的checkpoint里面进行。另外，最终生成的三个文件过于庞大，我并没有提交到git工程，如果感兴趣可以运行我的代码，重新生成。

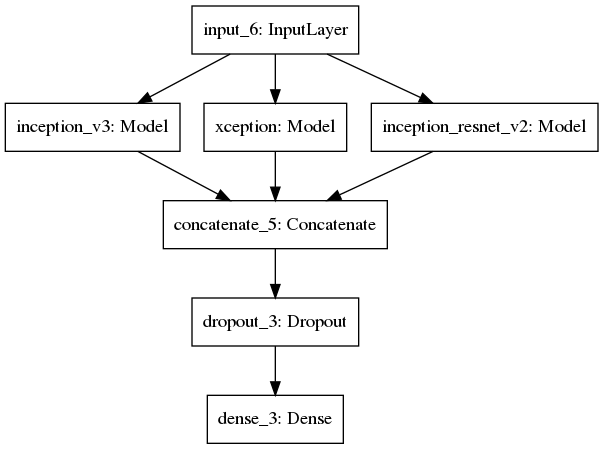
### 构建模型

接下来就是构建我们的模型结构，由于采用了预训练模型提取的特征向量。我们实际的工作，其实就是最后的分类处理。构建最后的全连接层即可完成这个目标。实际使用的模型可视化后如图6所示。



1. 自定义模型

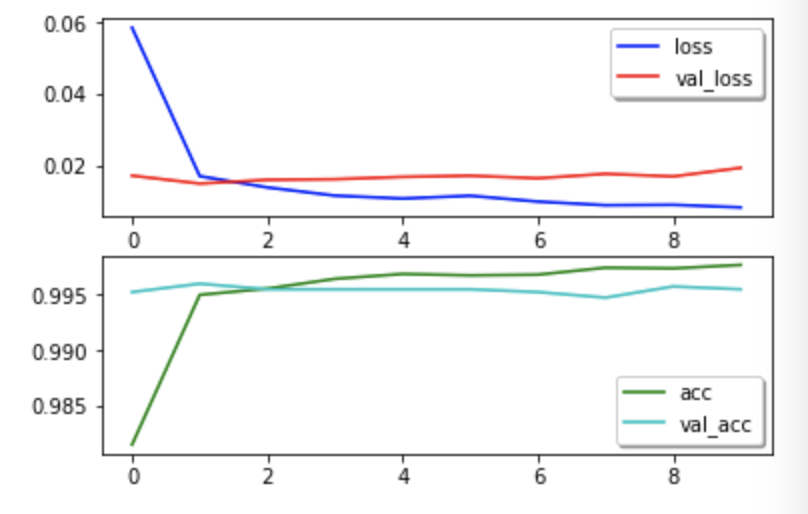
然而这个可视化的模型结构，并不能展示我们整个项目的解决方案要义，我作了额外的工作。将整个解决方案的连接综合展示，得到如图7所示的模型结构图。这张图清晰的展示了，我们这个项目的结构逻辑。即，将原始图片数据预处理后，分别输入三个预训练模型，通过这三个千锤百炼的模型，提取到特征向量后。将便于分类处理的特征数据，输入到最后的全连接层作分类训练。



1. 解决方案模型概览

### 训练

模型构建完成后，我们以特征数据为输入，训练模型。由于模型层次较浅，发现训练速度非常快。并且，由于特征数据质量非常高，我们发现对数损失下降非常快，10个周期后，可以达到0.0081。准确率及对数损失变化如图8所示。



1. 训练曲线

### 预测结果

根据训练好的模型，我们对测试数据进行预测处理。结果保存到final\_submission.csv。提交kaggle后，得到成绩结果如表1所示。这个需要注意的是首创预测非常失败，并不成功。具体的改进，我在模型微调中说明。而调整后到预测结果为0.04030，结果达到我们的要求（小于0.06）。

1. 预测结果

|  |  |  |
| --- | --- | --- |
| 模型 | 首次预测 | 调整后 |
| InceptionV3 | 2.35763 | 0.07372 |
| Xception | 2.26968 | 0.10159 |
| Inception ResnetV2 | 2.40522 | 0.06985 |
| Final\_model | 2.61388 | 0.04030 |

### 改进处理

首次完成这个项目的结果，我发现结果非常的不理想。远远低于我的预期。得到的结果如表1所示。我对于程序进行检测后发现，根本原因是测试集预测结果保存时顺序不一致，导致了重大偏差。调整后得到第二列结果，达到项目要求。

不过在排查问题的过程中，我也作了大量的尝试和改进，如下：

1. 首先数据的预处理阶段，虽然对于不合尺寸的图片进行了删除处理，但是对于错误标注的图片，没有引起足够的重视，予以删除。
2. 对于训练好的模型，还可以进行一定的微调。比如通过fine-tune来解冻靠近输出的层，改善结果。
3. 另外我吧Dropout的参数设置为0.5，防止过度拟合。

# 结果

## 模型的评价与验证

经过前面的测试和实践，我们发现预训练模型确实能够降低我们的设计难度和训练量，但是所得到的结果排名也并非特别优秀。综合几种模型的特征数据，并进行微调处理，才能得到我们可靠结果。

经过微调处理后，我们模型得分达到了预期的水平。证明我们的推测和处理是可靠的。另外，如果想得到更好的结果，我们还可以寻找更优秀的预训练模型，以及进一步微调模型。

## 合理性分析

经过反复的调整参数，我的得分0.04030，达到预期。

# 项目结论

## 对项目的思考

猫狗大战是非常成熟的项目，选用预训练模型也是常规的做法。通过挑选合适的预训练模型，提取到高质量到特征数据。我们可以通过非常小的计算量，达到非常好的预测效果。

这一点，对于缺乏大量数据的公司和个人，有非常好的启发和参考意义！

## 需要作出的改进

本项目采用模型融合到方式，借用了预训练模型的特征向量，再进行分类训练。如果需要进一步改进，可以采用更优秀的模型，导出的特征数据。另外对模型进行对预训练模型进行微调（fine-tune），也是一种方式。

\*\* 在提交之前， 问一下自己... \*\*

* 你所写的项目报告结构对比于这个模板而言足够清晰了没有？
* 每一个部分（尤其**分析**和**方法**）是否清晰，简洁，明了？有没有存在歧义的术语和用语需要进一步说明的？
* 你的目标读者是不是能够明白你的分析，方法和结果？
* 报告里面是否有语法错误或拼写错误？
* 报告里提到的一些外部资料及来源是不是都正确引述或引用了？
* 代码可读性是否良好？必要的注释是否加上了？
* 代码是否可以顺利运行并重现跟报告相似的结果？

**参考文献：**

1. Bengio Y,Delalleau O. On the expressive power of deep architectures[C] // Proc of the 14th International Conference on Discovery Science. Berlin:Springer-Verlag,2011:18-36.
2. Dalal N, Triggs B. Histograms of oriented gradients for human detection[C]// Computer Vision and Pattern Recognition，2005. IEEE Computer Society Conference on. Piscataway, NJ:IEEE Computer Society Conference on. Piscataway, NJ:IEEE,2005:886-893.
3. Hinton GE(1), Osindero S, Teh YW. A fast learning algorithm for deep belief ntes[J]. Neural Computation, 2006 ,18(7):1527-1554.
4. Psaltis D, Sideris A, Yamamura A. A multilayered neural network controller[J]. IEEE Control Systems Magazine, 1988, 8(2): 17-21.
5. Yann LeCun, Leon Bottou, Yoshua Bengio, et al. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998, 86(11):2278-2324.
6. https://github.com/mrgloom/kaggle-dogs-vs-cats-solution
7. https://zhuanlan.zhihu.com/p/34068451
8. https://blog.csdn.net/qq\_20909377/article/details/78976056
9. https://arxiv.org/abs/1512.00567
10. https://arxiv.org/abs/1610.02357
11. https://arxiv.org/abs/1602.07261