

机器学习纳米学位

毕业项目

龚宇海

2018 年 5 月 4 日

I. 问题的定义

项目概述

猫狗大战来自于 Kaggle 上举办的一次有趣的竞赛，目标是训练一个模型，然后通过上传一张猫或狗的照片，来分辨它是猫，还是狗，涉及到机器学习领域中的深度学习和图像分析方向。

项目需要使用到卷积神经网络（Convolutional Neural Network, CNN），是神经网络的一种，它在图像识别和分类领域已被证明非常有效。在卷积神经网络中，卷积的主要目的是从输入图像中提取特征。通过使用输入数据中的小方块来学习图像特征，卷积保留了像素间的空间关系。

一般的神经元是由输入值和激活函数给出感知器最终的输出。如图 1-1 所示：

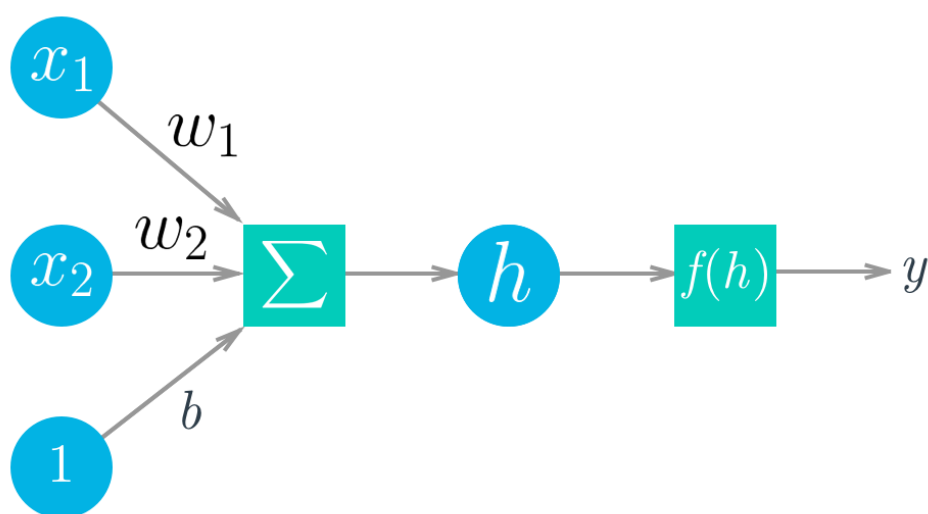


图 1-1 图片引用 [udacity](#)

输入值可以是由权重、输入和偏置项的线性函数组成，如 sigmoid 函数：

$$\text{sigmoid}(x) = 1/(1+e^{-x})$$

另一种 ReLU 函数：

$$\text{ReLU}(x) = \max(0, x)$$

ReLU 函数被更多的用于神经网络的激活函数。

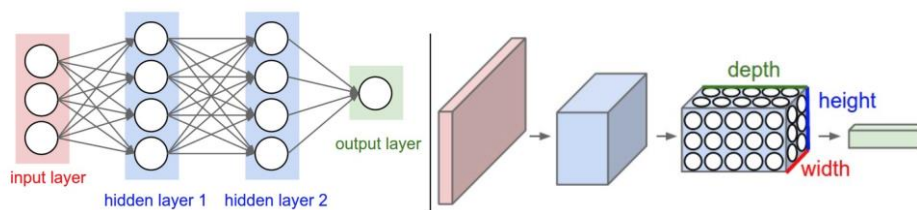


图 1-2 图片引用 [cs231n](#)

由多个神经元组成了神经网络，如图 1-2 左侧所示。在输入和输出之间的都是隐藏层，隐藏层中经过大量的计算，最后输出得到结果。图 1-2 右侧部分是卷积神经网络，它利用输入是图片的特点，把神经元设计成三个维度：width, height, depth，depth 是用来描述神经元的。比如输入的图片大小是 $32 \times 32 \times 3$ (rgb)，那么输入神经元就也具有 $32 \times 32 \times 3$ 的维度。这样的一种网络结构，正好对应图片分类的需求。

构建一个卷积网络需要 3 个最基本的层，分别是卷积层（Convolutional Layer），池化层（Pooling Layer），全连接层（Fully-Connected Layer）。

卷积神经网络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。

池化层的作用在于减少输出值的大小和避免过拟合，通过特征切分取平均值或最大值。

全连接层会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。

项目的数据集由 Kaggle 竞赛项目中提供，训练文件夹包含 12,500 张标记为猫的图像和 12,500 张标记为狗的图像。测试文件夹包含 12,500 个图像，根据数字 ID 命名，未做标记。对于测试集中的每个图像，模型需要预测图像是狗的概率（1=狗，0=猫），很明显，这是一个二分类问题。项目的目的是尽可能的提高准确的预测结果。

问题陈述

从 Kaggle 上获得的数据来自于真实世界中不同地区不同人所拍摄的猫或狗的图像。图像的分辨率质量参差不齐，猫和狗的形态各异，品种多样，背景环境也非常复杂，包括可能含有的异常图片，比如图像中根本就没有猫或狗，而是人或建筑物。这些都增加了分类的难度。

评价指标

训练一个可以有效运行的神经网络，需要使用到损失函数。对于二分类问题，常使用交叉熵作为损失函数对模型的参数求梯度进行更新：

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

其中 n 是样本数量， y 是标签， $y=1$ 表示狗， $y=0$ 表示猫， \hat{y} 表示样本为狗的概率。损失函数的值越小，说明模型对于数据集的分类效果越好。

II. 分析

数据的探索

项目数据集有两个子集，分别是训练数据集和测试数据集。其中训练数据集中的图片按照 `'(dog/cat).(num).jpg'`，如 `'dog.132.jpg'` 或 `'cat.435.jpg'`，相当于对图片做了标签，明确说明图片中的是狗还是猫。

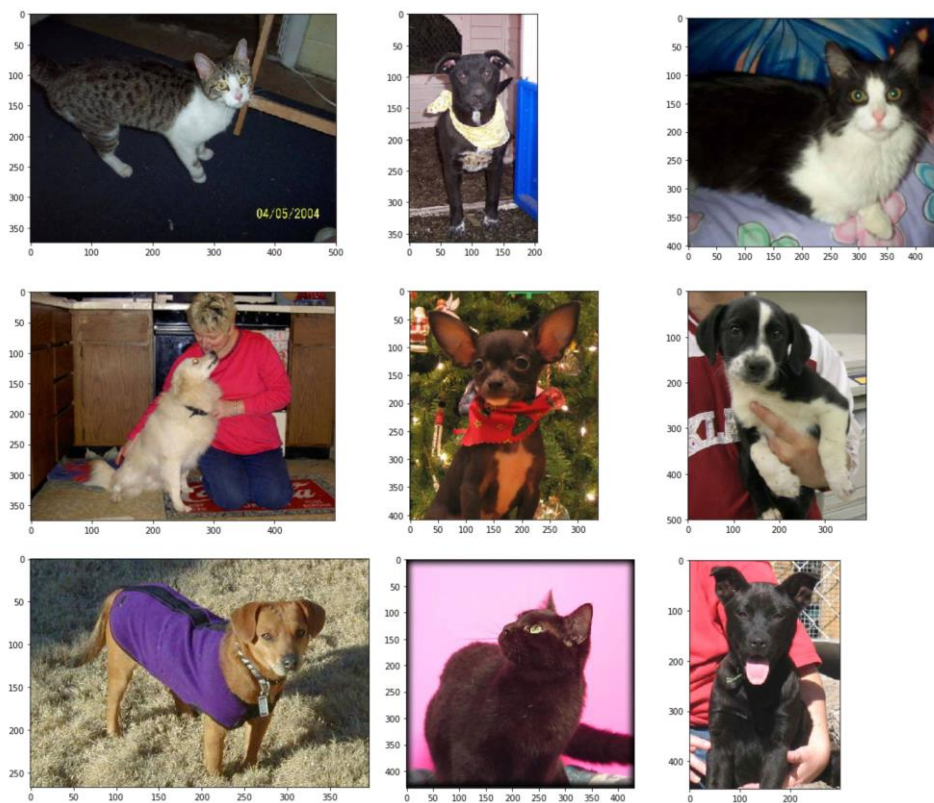


图 2-1 train 中的部分图片

训练集总共有 25000 张图片，猫和狗各 12500 张，分别占一半。从图 2-1 中可以看出，图片拍摄的背景丰富多样，拍摄的光线明暗不均，图片的大小也是各不相同，图 2-2 给出了训练集中图片大小的分布情况，不过大部分图片用肉眼来看，还是可以直观的分辨出猫狗的。

探索性可视化

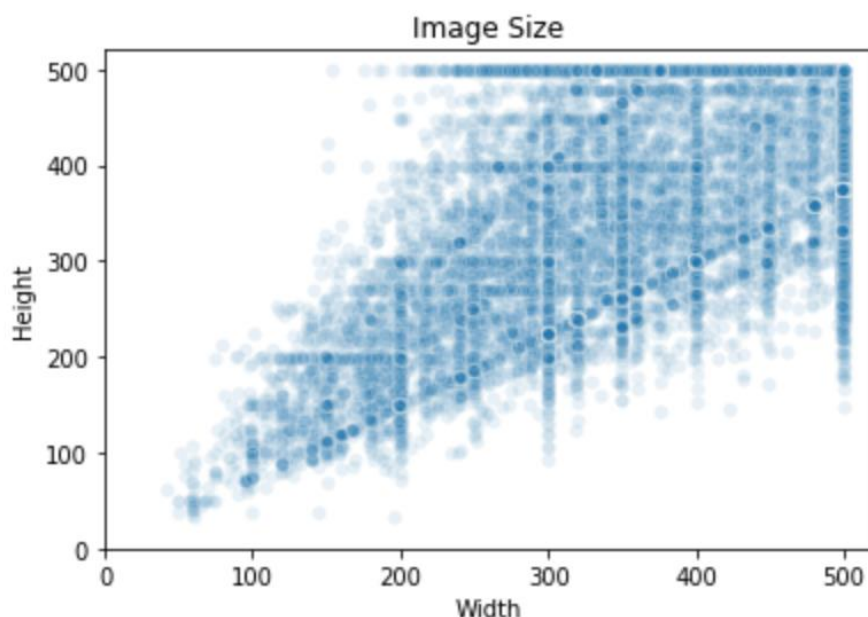


图 2-2 图片大小分布

对于训练模型来讲，图片大小的不一致，会给构建模型带来很大的困难，所以在构建模型之前，需用对所有的图片的尺寸进行统一化伸缩。使用 `openCV` 对图片进行处理，在训练集中，大部分的数据都可以作为神经网络的特征提取，但也有部分的异常值，如 `dog.1773.jpg`、`dog.12376.jpg`。



图 2-3 异常图片，左 `dog.1773.jpg`，右 `dog.12376.jpg`

因为是人为上传的图片，最初标记这些图片的也是人为操作的，难免会有一些错误信息。在实际生活中，遇到的异常值可能还要多，这就需要模型具备一定容错能力。在数据量更大的时候，一个模型的好坏，容错能力也至关重要。这里对这些异常值不做处

理，但我们知道这种情况在很多时候是实际存在的，而且很难避免。

算法和技术

项目使用迁移学习(Transfer learning) 的方法来构建猫狗分类的神经网络。迁移学习顾名思义就是把已学训练好的模型参数迁移到新的模型来帮助新模型训练。具体地，在迁移学习中，我们已有的知识叫做源域(source domain)，要学习的新知识叫目标域(target domain)。迁移学习研究如何把源域的知识迁移到目标域上。特别地，在机器学习领域中，迁移学习研究如何将已有模型应用到新的不同的、但是有一定关联的领域中。传统机器学习在应对数据的分布、维度，以及模型的输出变化等任务时，模型不够灵活、结果不够好，而迁移学习放松了这些假设。在数据分布、特征维度以及模型输出变化条件下，有机地利用源域中的知识来对目标域更好地建模。另外，在有标定数据缺乏的情况下，迁移学习可以很好地利用相关领域有标定的数据完成数据的标定。

迁移学习按照学习方式可以分为基于样本的迁移，基于特征的迁移，基于模型的迁移，以及基于关系的迁移。基于样本的迁移通过对源域中有标定样本的加权利用完成知识迁移；基于特征的迁移通过将源域和目标域映射到相同的空间（或者将其中之一映射到另一个的空间中）并最小化源域和目标域的距离来完成知识迁移；基于模型的迁移将源域和目标域的模型与样本结合起来调整模型的参数；基于关系的迁移则通过在源域中学习概念之间的关系，然后将其类比到目标域中，完成知识的迁移¹。

ImageNet 上已经完成训练的神经网络模型有很多，它们对自然图像特征具备很好的提取和筛选能力。相对于本项目中，会产生干扰问题的背景环境、人物和其他自然实物，这些经过训练的模型都能够很好的将主要物体提取出来，然后可以迁移到新的模型中训练。这样的好处是可以不用自己从零开始构建和训练模型，况且即使是自己从零开始构建的模型，也不一定会比现有的更好，所以选择迁移模型，建立在现有的工具上开始训练新模型会是一个不粗的选择。

¹ [什么是迁移学习 \(Transfer Learning\)? 这个领域历史发展前景如何? - 王晋东不在家的回答 - 知乎](#)

基准模型

在使用 Keras 深度学习库的官网上可以看到，已经有人给我们提供了一些可使用的模型，并且标注了模型的准确率，利用现有的模型以及预训练出的权重，就能很好的表示猫和狗的特征。考虑到准确率，模型大小等因素，基准模型选择的是 ResNet50[1]、InceptionV3[2]、和 Xception[3]这个三个模型组合。结构图如下：

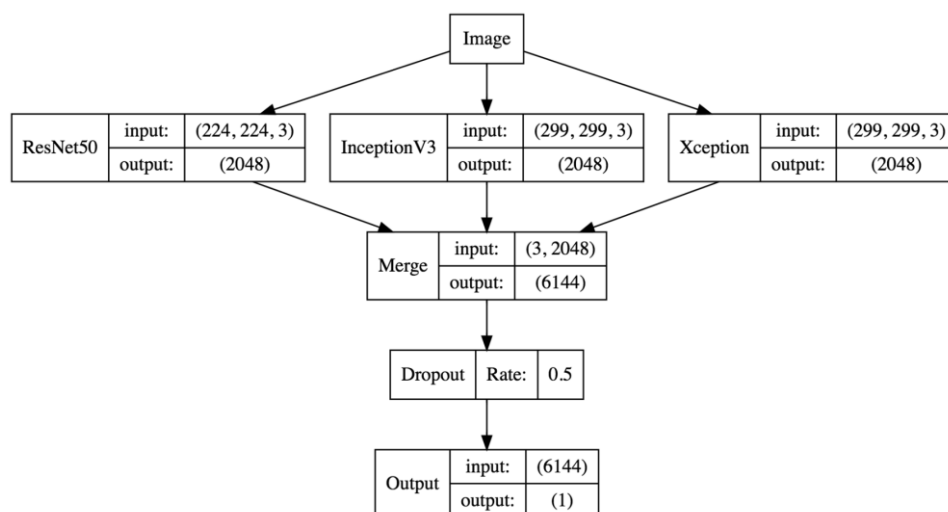


图 2-4 基准模型结构图

各个子模型的初始权重采用 ImageNet 权重，禁用了各个模型分类器的 Top 层，将全局平均池化层作为输出层。在复现模型的过程中，构建好模型后直接训练模型，第一次就在 Kaggle 上得到 0.04008 的分数。相对于项目示例中作者获得 0.04141 稍微好一点，因此希望能够有更好一点的成绩。

III. 方法

数据预处理

基于现有的模型进行迁移学习，输入的数据需要与模型相匹配，ResNet50、InceptionV3 和 Xception 这三种模型对应输入的数据是不同的，需要满足各个模型的默认输入要求。图像数据的表现形式在计算机中表现为：(weight, height, channel)，每个元素的取值范围在 0~255 之间。预训练模型先进行归一化处理，处理完成后作为输入数

据，传递给模型；添加全局平均池化层 `GlobalAveragePooling2D`，作为模型最后的输出数据。`ImageDataGenerator` 作为 `keras` 提供的图像预处理函数，可以将传递过来的图像数据根据指定的大小，传递给模型需要的输入数据。

查阅 `Keras API`，`Application` 模块为部分模型提供预处理好的输入值，如 `InceptionV3` 和 `Xception`。在训练模型时，直接使用 `preprocess_input` 作为输入参数。

执行过程

根据预先构想好的迁移模型方案，分别训练并导出模型 `ResNet50`、`InceptionV3` 和 `Xception` 三个处理好的模型，这个过程需要等待一定时间。等到成功获得对应的模型文件后，读取 3 个模型文件，同时标注训练集和测试集，使用 `Concatenate` 层的函数式接口进行融合。对训练集进行分割，分出 20% 作为验证集，然后对融合后的模型开始训练。

以 `sigmoid` 函数作为激活函数，`adadelta` 作为优化器，以 0.5 的输入比 `Dropout`，直接对模型 `fit`，经过 10 次 `epoch`，在测试集上预测，上传至 `Kaggle`，得分为 0.04008。

完善

使用 `ModelCheckpoint` 回调函数的方法，在获得好的权重的时候保存，这样可以确保获得的权重是每次 `epoch` 中最好的。同样的参数再训练和预测，上传至 `Kaggle`，得分为 0.03843。

| | | |
|---|---------|--------------------------|
| merged_pred.csv 4 days ago by Kevin Gong save bast weight | 0.03843 | <input type="checkbox"/> |
| merged_pred.csv 5 days ago by Kevin Gong merge first | 0.04008 | <input type="checkbox"/> |

图 3-1 二次训练在 `Kaggle` 上的得分

可见采用回调保存最优权重的方法可以提高准确性。对于输入比为 0.5 是不是会有点多，减少一半 0.25，再次尝试，两次对比如下：

```

Epoch 00008: val_loss improved from 0.01616 to 0.01607, saving model to fine_weight.h5
Epoch 9/20
16000/16000 [=====] - 1s 40us/step - loss: 0.0101 - acc: 0.9968 - val_loss: 0.0204 - val_ac
c: 0.9948

Epoch 00009: val_loss did not improve from 0.01607
Epoch 10/20
16000/16000 [=====] - 1s 39us/step - loss: 0.0095 - acc: 0.9968 - val_loss: 0.0163 - val_ac
c: 0.9955

Epoch 00010: val_loss did not improve from 0.01607
Epoch 11/20
16000/16000 [=====] - 1s 40us/step - loss: 0.0095 - acc: 0.9968 - val_loss: 0.0219 - val_ac
c: 0.9945

Epoch 00011: val_loss did not improve from 0.01607
Epoch 12/20
16000/16000 [=====] - 1s 37us/step - loss: 0.0086 - acc: 0.9970 - val_loss: 0.0158 - val_ac
c: 0.9955

Epoch 00012: val_loss improved from 0.01607 to 0.01576, saving model to fine_weight.h5
Epoch 13/20
16000/16000 [=====] - 1s 40us/step - loss: 0.0086 - acc: 0.9974 - val_loss: 0.0157 - val_ac
c: 0.9952

Epoch 00013: val_loss improved from 0.01576 to 0.01575, saving model to fine_weight.h5
Epoch 14/20
16000/16000 [=====] - 1s 39us/step - loss: 0.0076 - acc: 0.9971 - val_loss: 0.0160 - val_ac
c: 0.9945

Epoch 00014: val_loss did not improve from 0.01575

```

图 3-2 Dropout 0.5 时，val_loss 为 0.01575

```

Epoch 00003: val_loss improved from 0.02096 to 0.01681, saving model to fine_weight.h5
Epoch 4/20
16000/16000 [=====] - 1s 36us/step - loss: 0.0126 - acc: 0.9963 - val_loss: 0.0160 - val_ac
c: 0.9948

Epoch 00004: val_loss improved from 0.01681 to 0.01599, saving model to fine_weight.h5
Epoch 5/20
16000/16000 [=====] - 1s 37us/step - loss: 0.0104 - acc: 0.9964 - val_loss: 0.0155 - val_ac
c: 0.9950

Epoch 00005: val_loss improved from 0.01599 to 0.01555, saving model to fine_weight.h5
Epoch 6/20
16000/16000 [=====] - 1s 37us/step - loss: 0.0092 - acc: 0.9971 - val_loss: 0.0162 - val_ac
c: 0.9952

Epoch 00006: val_loss did not improve from 0.01555
Epoch 7/20
16000/16000 [=====] - 1s 43us/step - loss: 0.0084 - acc: 0.9974 - val_loss: 0.0153 - val_ac
c: 0.9955

Epoch 00007: val_loss improved from 0.01555 to 0.01534, saving model to fine_weight.h5
Epoch 8/20
16000/16000 [=====] - 1s 38us/step - loss: 0.0075 - acc: 0.9971 - val_loss: 0.0154 - val_ac
c: 0.9958

Epoch 00008: val_loss did not improve from 0.01534

```

图 3-2 Dropout 0.25 时，val_loss 为 0.01534

同样执行 20 次的 epoch，基本上都会在一定次数之后，停止在一个数值上无法再次提升。这里虽然有不同的结果，但是实际上，即使参数相同，val_loss 也会有一定范围的浮动，在这个浮动中不一定每次都能获得最佳的权重。

IV. 结果

模型的评价与验证

通过迁移模型的方法，将原本模型效果一般的 3 个模型进行融合，得到一个相对于

单个模型较好的组合模型。对于单个模型来说，尝试训练 Resnet50，得到的 Kaggle 分数为 0.07662，结果一般，还有很多可以提高的空间。从最后获得评分上，可以看出，这种迁移融合模型的方式是可行的。

合理性分析

通过 ModelCheckpoint 函数回调的方法，得到了一个比较好的模型权重，但是在第二次、第三次的结果却下降了，说明 Dropout 可能恰巧是不够好的数据，留下了比较好拟合的模型数据，所以才有 0.03843 的成绩，在之后的几次尝试中，并没有能够完全复现这个成绩，不过基本保存在 0.03936。


| | | |
|--|---------|--------------------------|
| merged_pred.t.csv a month ago by Kevin Gong add merged pred train | 0.03961 | <input type="checkbox"/> |
| merged_pred.csv a month ago by Kevin Gong try again merged pred | 0.03936 | <input type="checkbox"/> |
| merged_pred.csv a month ago by Kevin Gong a new fine weight to try  | 0.03936 | <input type="checkbox"/> |
| merged_pred.csv a month ago by Kevin Gong save bast weight | 0.03843 | <input type="checkbox"/> |
| merged_pred.csv a month ago by Kevin Gong merge first | 0.04008 | <input type="checkbox"/> |

图 4-1 Kaggle 成绩显示

v. 项目结论

结果可视化

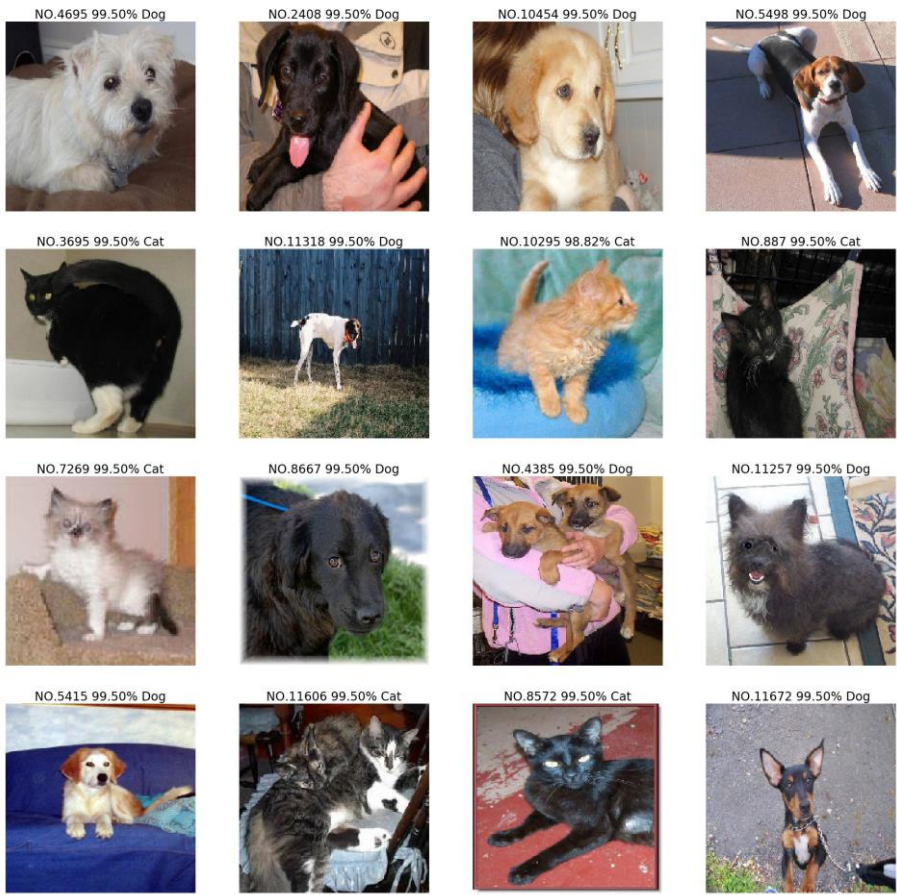


图 5-1 测试图片

从测试图片中，不难看出，总的准确率还可以，但是应该还有更多可提升的范围。

对项目的思考

迁移学习，融合模型的方法可以有效的提升模型的准确率，但也有一定的不稳定性，在获取权重的时候，也有可能获得的权重并不是最佳的，仅保存相对权重较好的值。在 Dropout 的时候，也有可能丢掉了一些较好的模型数据。不过相对于单模型而言，迁移模型的方法会更容易的提升模型效果。还有一种冻结模型的方法，对指定的层范围来训练模型的方法，也会对提升模型效果有一定作用，这次由于时间原因，提炼模型特征消

耗时间较久，并没有对每个模型进行尝试。感谢原项目作者培神²提供的思路方法。

² [手把手教你如何在 Kaggle 猫狗大战冲到 Top2%](#)

VI. 参考文献

- [1]. He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition[J]. arXiv preprint arXiv:1512.03385, 2015
- [2]. Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the Inception Architecture for Computer Vision[J]. arXiv preprint arXiv:1512.00567, 2015.
- [3]. Chollet F. Xception: Deep Learning with Depth wise Separable Convolutions[J]. arXiv preprint arXiv:1610.02357, 2016.
- [4]. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.
- [5]. Szegedy C, Ioffe S, Vanhoucke V, et al. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning[J]. arXiv preprint arXiv:1602.07261, 2016.
- [6]. Hu J, Shen L, Sun G. Squeeze-and-excitation networks[J]. arXiv preprint arXiv:1709.01507, 2017.