优达学城机器学习纳米学位毕业项目：猫狗大战

徐早立

1. 项目概述

人工智能的研究最早兴起于 1960 年代，而其后的发展可谓一波三折 [1]。1980 年代兴起了人工智能的第二波热潮，一批专家系统登上历史舞台，却又很快消失了。从那以后人工智能跌入了近 30 年的寒冬，直到最近几年才又“火”了起来。那么，人工智能为什么又“火”了呢？一个重要的节点是围棋人工智能 AlphaGo 的出现 [2]。2016 年 3 月，AlphaGo 击败了前世界冠军李世乭，惊艳世界。更让人觉得不可思议的是，AlphaGo 甚至可以通过自我博弈就在短时间内进化，并在 2017 年又击败了世界第一的柯洁，全面超越人类最高水平，让人惊呼人工智能这一次是真的来了。通过 AlphaGo，人们开始去了解其背后的原理，于是深度学习和神经网络等词汇开始频繁地出现在大众的视野。事实上，深度学习与神经网络不仅可以用来训练计算机下棋，其作为一种通用的机器学习算法，还被广泛应用到了各种领域，这其中就包括了计算机视觉（Computer Vision）[3]。在 2012 年的 ImageNet 竞赛上，深度学习和神经网络开始展现出其算法的优势，基于卷积神经网络的 AlexNet 算法取得了 15.4%的错误率，远低于第二名的 26.2%。从那之后，卷积神经网络进一步发展，新的网络结构和优化算法使得其逐渐成为处理计算机视觉任务的主流方法。目前具备看图识物能力的人工智能已经可以被训练出来，用来替代人类来完成某些简单枯燥的看图识物任务。

本项目希望能够利用卷积神经网络，训练出可以区分猫和狗的人工智能。这一人工智能需要能够识别出一张彩色图片中的动物是猫还是狗。本项目也同时对应于数据网站 Kaggle 的一个猫狗识别项目（https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition）。在 Kaggle 上这一项目的模型评估标准采用的是平均交叉熵（average cross-entropy），或者叫做 log loss 值，定义为：

$$\mathrm{J} = \frac{1}{m}\sum_{i=1}^{m}[y^i ln\hat{y}^i + \left(1 - y^i\right)ln(1 - \hat{y}^i)],$$

其中 y 表示实际图片中是猫（y=0）还是狗（y=1），而 $\hat{y}$表示神经网络的预测值，m 为训练图片总数。本项目的目标是要建立一个区分猫狗图片的人工智能，其泛化 loss 值小于 0.05629（用训练集的 loss 值来代表泛化 loss 值），这一基准阈值对应于 Kaggle 上公开排行榜的第 100 名。

2. 项目分析

2.1 数据搜集（Data collection）

本项目的数据来自 Kaggle 的猫狗识别项目（https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/data）。数据分为训练集和测试集。数据通过下载的方式全部下载到本地，解压并保存在路径“.../input/train”（训练集）和“.../input/test/”（测试集）下。

2.2 探索性数据分析（Exploratory data analysis）

训练集共有 25000 张图片，猫和狗的照片各有 12500 张，如图 1 所示。每张图片的名字中含有该图片的分类与序号信息。比如，命名为“cat.0.jpg”意味着这张图片中是猫，图片编号是 0。测试集包含 12500 张猫或狗的照片。测试集的图片只有序号，没有图片的分类信息。训练集的 25000 张照片会被用来训练具有猫狗辨识能力的人工智能，而测试集的图片会用来评判这一人工智能的猫狗辨识能力。

训练集中猫和狗的图片示例如图 2 所示（为方便展示，图片进行了缩放调整）。初步来看，训练集的数据特征有：

1. 猫狗照片数量均等，没有类别不平衡的问题；
2. 图片的尺寸不一致。如图 2(a)中猫的照片尺寸为 417×299，而图 2(b)中狗的图片尺寸为 251×359。
3. 有的图片中包含多只猫或狗，如图 2(c)所示。
4. 数据集中含有一些疑似无关的图片，如图 2(d)所示。

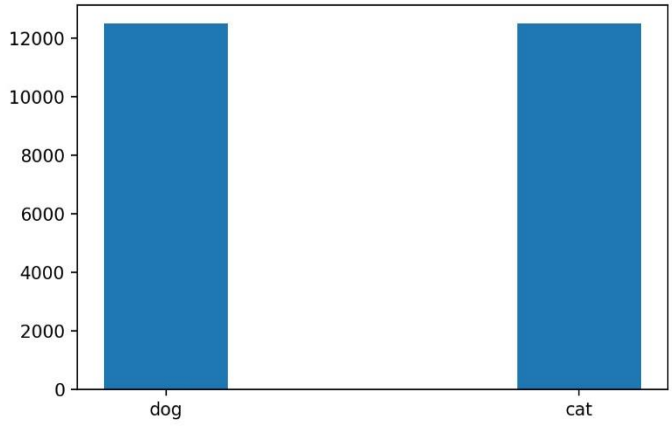

图 1 训练集中猫/狗照片数量柱状图

(a)

(b)

(c)

(d)

图 2 训练集图片示例。 (a) 猫的图片示例； (b) 狗的图片示例；(c) 两只猫的图片；(d) 疑似标记错误的图片示例

2.3 数据预处理 （Data preprocessing）

对于训练集的图片，所做的数据预处理包括：

1. 划分训练验证集。本项目采用“留出法”将训练集划分为互斥的训练/验证集，其训练/验证划分比例为 0.96/0.04。验证集有 1000 张图片，如果两个不同的模型的识别准确率相差 0.1%，那么在验证集上可以体现出来（0.1%×1000=1 张图片）。
2. 调整图片大小。本项目的图片尺寸各一，需要统一图片的尺寸后，才能输入到卷积神经网络进行建模。本项目最终所选用的尺寸大小为 128×128，这是综合建模结果和计算机运算能力的折中选择。若是尺寸选择过小，则图片清晰度不够，难以达到满意的识别精确度；若尺寸选择过大，则难以处理过大的数据量，运算效率降低。统一大小后的图片示例如图 3 所示。可以看到调整大小后的图片有两个特征：第一，猫和狗的形态比例相较于原始图片发生了变形；第二，图片的颜色也发生了变化。
3. 归一化。每张图片都是由像素点阵列存储在计算机中，而每一个阵列点是由 0~255 范围内的数字表示。为加快模型优化速度，采用常规的归一化处理，把所有的像素值用 min-max 归一化成 0~1 范围内的数字。

4. 独热编码。这一项目有猫狗两类， [0,1]表示猫，[1, 0]表示狗。
5. 保存预处理数据。方便直接导入处理过的数据，而无须每次调试模型时再预处理一遍数据。

图 3 统一大小后的图片

对于测试集的图片，所做的数据预处理包括：

1. 调整图片大小，保持和训练集一致。
2. 归一化，保持和训练集一致。
3. 保存预处理数据。

在做数据预处理的过程中，遇到了计算机内存不足的问题。本项目所用的电脑内存为 32GB，若是调整图片大小的时候选择 64×64，则可以一次性处理完全部 12500 张训练图片并保存。若是调整图片大小的时候选择 128×128 或者 224×224，则会报出内存不足的问题。经过分析，造成这一问题的原因在于保存预处理数据时使用的 pickle 库会使用大量的内存[4]。若是图片尺寸过大，则无法一次性保存所有预处理的数据。

解决这一问题可以选择使用 HDF5 [5]或者 Tensorflow 的 TFRecords [6]来代替 pickle，减少内存的消耗。而本项目采取的是另一种方法，对数据集进行分批处理并保存。这一方法与图像分类项目中的做法一致 [7]。简略来讲，为解决数据量过大的问题，将数据集的图片分成几批，每次计算机只处理一批的数据。据试验，若一批中的图片为 4096 时，即使图片大小为 128×128 或者 224×224 都能完成预处理并保存在本地的磁盘中。

2.4 数据分析-建模与优化（Data analysis）

2.4.1 模型架构

本项目通过迭代优化的方式来建立卷积神经网络。最初的网络参照图像分类项目的网络结构，随后参照 AlexNet [8]增加了网络了层数，最后再参考了 VCC-16 [9]完成了卷积神经网络的搭建。参考[9] 的标记方式，本项目最终的模型架构如图 4 所示。

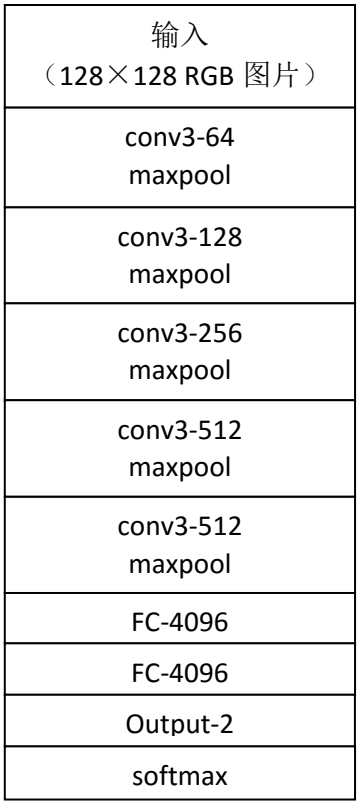

图 4 本项目卷积网络架构

conv3-64 的含义是：3 代表 3×3 的卷积滤波器，64 代表这一卷积层的输出深度为 64。stride 值为 1，采用的 padding 方式为 'same'。其他卷积层的含义以此类推。

maxpool 表示最大池化层，滤波器大小为 2×2，stride 为 2, padding 方式为 'valid'。

FC-4096 为全连接层，输出大小为 4096。

Output-2 为输出层，输出大小为 2。与全连接层相比，输出层没有激活函数，直接输出。

softmax 把输出归一化到和为 1。

另外，卷积层、池化层和全连接层都使用 Relu 作为激活函数。

在每一个最大池化层和全连接层后面都加入 dropout，来防止过拟合。

2.4.2 模型调参

总体来说模型调参是一个试错的过程，大致过程是：从一些经典模型或者常用的设置出发，在结合实际项目的数据情况进行试错调整。

keep_prob: 任何一个给定单元的留存率。keep_prob 可以调整丢弃单元的数量，从而缓解模型过拟合的程度。keep_prob 最终选为 0.6。

stddev: 卷积层、全连接层和输出层的权重值在随机初始化时的标准方差。权重的初始化是一个影响结果的重要因素，合理的权重初始化应该是既能使得模型能够摆脱全对称的状态（梯度消失使得权重值无法得到更新），又不能偏离全对称态太大（各个权重值相差太大导致梯度爆炸）。本项目中 stddev 选择的大小为 0.05。

batch size: 本项目采用 mini-batching 和随机梯度下降来进行优化，batch size 的选择主要是考虑内存大小，在计算允许的前提下尽量增大 batch size。本项目使用的 batch size 值为 128。

epoch: 模型的训练次数，本项目设定的最大 epoch 为 100 次。

2.4.3 模型评估

本项目在建模以及调参过程中，通过监控训练集和验证集的准确率和 loss 值来评估模型的好坏。具体的讲，在每一代优化结束后，计算、打印并储存训练集和验证集的准确率和 loss 值，以此来判断当前模型是欠拟合，还是过拟合。

本项目所建立的卷积神经网络在验证集上的 loss 值为~0.18。将测试集的数据导入模型，计算出结果并导入 kaggle 网站，得到的测试集上的 loss 值为 0.22924。

2.5 数据可视化 (Data visualization)

当前略

结果与讨论

当前的 loss 值还并不能令人满意，没有达到之前设定的小于 0.05629 这一目标。希望能够得到一些反馈完善后再来完成这一部分。

结论

当前略

参考文献

[1] http://mp.weixin.qq.com/s/-wSYLu-XvOrsST8_KEUa-Q, 浅谈人工智能：现状、任务、构架与统一——正本清源，2017，朱松纯，视觉求索

[2] https://en.wikipedia.org/wiki/AlphaGo

[3] https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html, 2016, Adit Deshpande, The 9 Deep Learning Papers You Need To Know About (Understanding CNNs Part 3)

[4] https://shocksolution.com/2010/01/10/storing-large-numpy-arrays-on-disk-python-pickle-vs-hdf5adsf/, 2010, Craig Finch, Storing large Numpy arrays on disk: Python Pickle vs. HDF5

[5] http://machinelearninguru.com/deep_learning/data_preparation/hdf5/hdf5.html#list

[6] http://machinelearninguru.com/deep_learning/data_preparation/tfrecord/tfrecord.html

[7] https://github.com/udacity/cn-deep-learning/blob/master/image-classification/dlnd_image_classification.ipynb

[8] https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf, AlexNet

[9] https://arxiv.org/pdf/1409.1556.pdf, VCC-16