项目报告

一．问题定义

研究背景

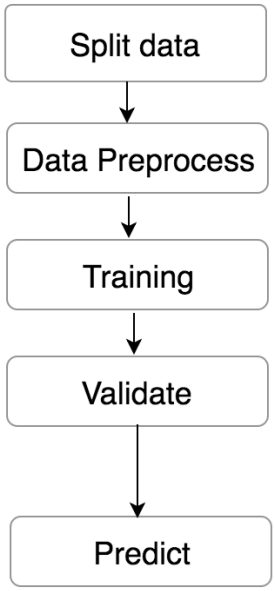
现如今深度学习已经广泛运用在各个领域，近十年来，深度学习取得了快速的发展以及长足的进步，通过深度学习来进行图像识别已经在目前取得了巨大的成功。2015 年微软开发的基于深度卷积神经网络，在 ImageNet1000 挑战中首次超越了人类进行对象识别人类的能力。2016 年 3 月人工智能围棋比赛，DeepMind 公司开发的深度学习围棋程序 AlphaGo 战胜了世界围棋冠军、职业九段选手李世石，这是人类第一次在围棋上战胜世界冠军。2018 年 Waymo 开发的基于深度学习的无人车正式宣布营业，宣告了无人驾驶时代的真正到来。

项目概述

本项目来源于 kaggle 的猫狗大战竞赛，通过训练集的学习让计算机能够分别识别猫和狗的图像。这是一个二元分类项目，即判断一张图片的分类是猫还是狗，1 表示分类结果是狗，0 表示分类结果是猫。

问题陈述

猫狗大战是一个二元分类问题，需要根据 kaggle 提供的训练集图片来训练自己的深度学习模型来对测试集的图片进行预测。具体分为以下步骤：



评价指标

kaggle 一共举行过两次猫狗大战的比赛，第一次使用正确率作为评估标准，第二次使用的是 log 损失函数。因为现在深度学习发展十分的迅速，而深度学习尤其适合处理图像方面的问题，如果依旧是使用正确率作为评估标准，那么大对数选手的模型都是 99%的正确率，不能明显的区分开。如果使用 log 损失函数，不仅仅需要分类正确，还需要对结果有一个较高的可信度，这样就能明显区分各个模型的分类效果，尤其是 Top 模型的分类效果。log 损失函数具体如下所示：

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

- n 表示测试集样本数量
- $\hat{y}_i$ 表示该图片被预测为狗的概率
- y_i 表示如果图片为 dog，则为 1，反之为 0

二．分析

数据探索

kaggle 一共提供了两个数据集: train 和 test（可以从 https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/data 进行数据下载）。其中训练集包含 25000 个样本，cat 和 dog 各 12500 个，测试集有图片 12500 张，但并未标注猫与狗，用于最后的模型测试。通过观察 train 和 test 中的图片，发现图片中不会同时存在猫和狗，但是可额能同时才有两个猫或者两条狗。而且每张图片的清晰度和图片的大小不同，猫与狗占整张图片的比例也有所差别，如下所示：



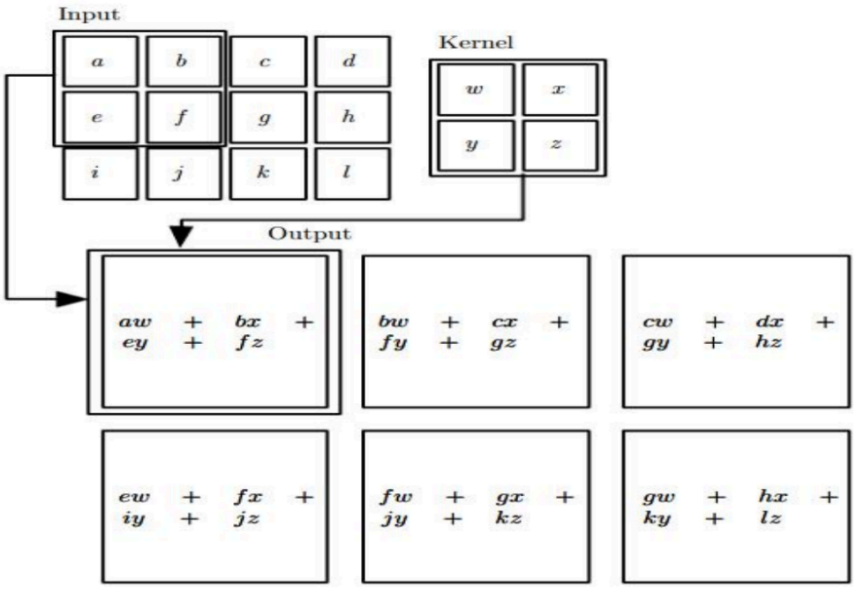



算法与技术

在本项目中我主要使用的是卷积神经网络来构建我的算法模型。卷积神经网络是一种多层神经网络，擅长处理图像特别是大图像的相关机器学习问题。卷积网络通过卷积、池化、全连接等一系列方法，成功将数据量庞大的图像识别问题不断降维，最终使其能够被训练。下面我将分别介绍卷积、池化、RELU 激活函数、dropout 层、迁移学习等知识点。

（1）卷积层

通过卷积操作，讲计算量庞大的图像识别问题不断降维，最终使其恩呢狗狗被神经网络训练。从直观上来看，可以把卷积层看做一系列的可训练/学习的过滤器。卷积过滤器滑过的每个像素，组成新的 3 维输出数据。每个过滤器都只关心过滤数据小平面内的部分特征，当出现它学习到的特征的时候，就会呈现激活的状态。



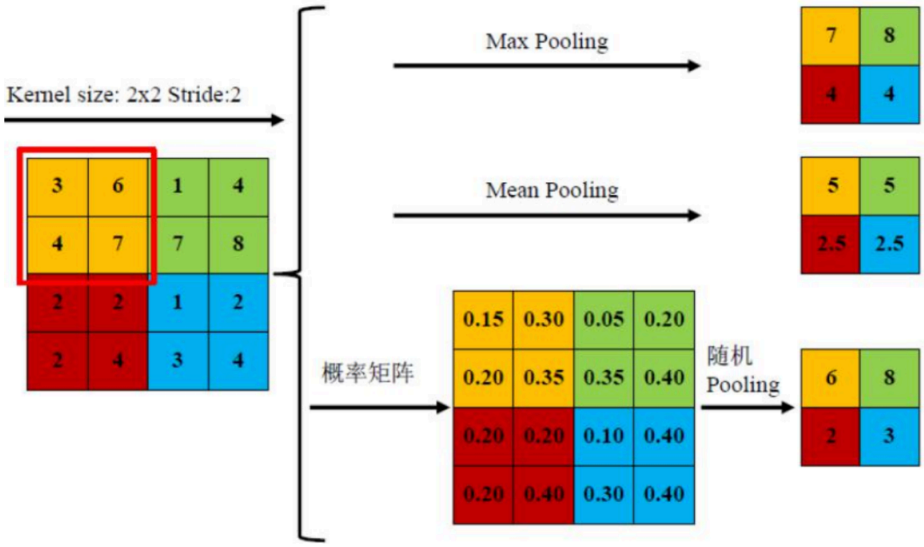
卷积原理

（2）池化层

经过卷积处理后的数据，理论上是可以直接拿来进行训练，但这样的计算量依旧太大。为了降低计算量，以及提高模型的泛化能力，我们会对其进行池化处理。池化处理通常有以下三种方式：

- mean-pooling，即对邻域内特征点只求平均，对背景保留更好
- max-pooling，即对邻域内特征点取最大，对纹理提取更好

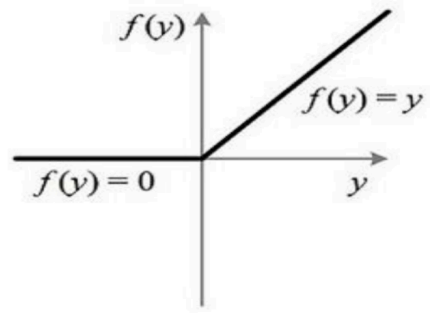
- Stochastic-pooling，介于两者之间，通过对像素点按照数值大小赋予概率，再按照概率进行亚采样



池化原理

（3）RELU 函数

在深度神经网络中，通常使用一种叫修正线性单元(Rectified linear unit, RELU)作为神经元的激活函数。相比于其他激活函数来说，RELU 有以下优势：对于线性函数而言，RELU 的表达能力更强，尤其体现在深度网络中；而对于非线性函数而言，RELU 由于非负区间的梯度为常数，因此不存在梯度消失问题，使得模型的收敛速度维持在一个稳定状态。

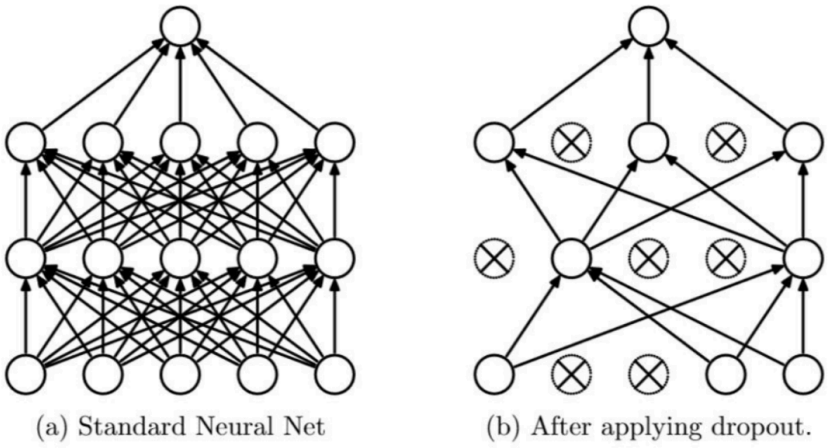


$$ReLU(x) = \begin{cases} x & if\, x > 0 \\ 0 & if\, x \le 0 \end{cases}$$

RELU 函数

（4）Dropout

Dropout 是指在深度学习网络的训练过程中，对于神经网络单元，按照一定的概率将其暂时从网络中丢弃。这种方式可以减少特征检测器间的相互作用，增强模型的泛化能力。

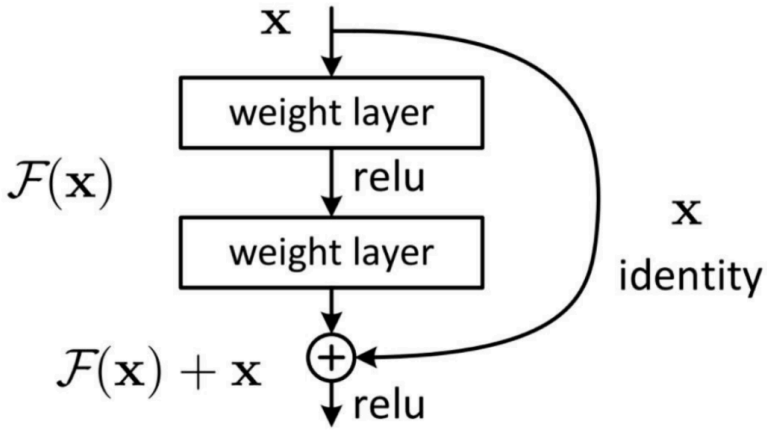


Dropout 原理

（5）迁移学习

迁移学习(Transfer learning)就是把已经训练好的模型参数迁移到新的模型来帮助新模型训练。考虑到大部分数据或任务是存在相关性的，所以通过迁移学习我们可以将已经学到的模型参数通过某种方式来分享给新模型从而加快并优化模型的学习效率。

综合以上技术方法，在本次项目我决定采用经典的 ResNet 模型。ResNet 在 2015 年提出以来，曾在 ImageNet 比赛 classification 任务上获得第一名，因为它“简单与实用”并存，之后很多方法都建立在 ResNet50 或者 ResNet101 的基础上完成的，检测、分割、识别等领域都有着广泛的运用。



ResNet 模型原理

处理 ResNet50 外，我还尝试了比 ResNet50 更深的 InceptionV2，但是在训练集上得到的 accuracy（85%左右）并没有 ResNet50 的好，其中一个可能的原因是数据集的数据量不够大。

最终目标

本项目的最低要求为 kaggle 排行榜前 10%，也就是在 Public Leaderboard 上的 logloss 要低于 0.06127，所以我的最终木匾是 logloss 要小于 0.06127。

三．方法

数据预处理

（1）划分数据集

kaggle 一共提供了两个数据集 train 和 test，所以需要把 train 数据按照一定比例划分出训练集和验证集，划分后文件的结构如下：

```
data/
    train/
        dogs/
            dog001.jpg
            dog002.jpg
            ...
        cats/
            cat001.jpg
            cat002.jpg
            ...
    validation/
        dogs/
            dog001.jpg
            dog002.jpg
            ...
        cats/
            cat001.jpg
            cat002.jpg
            ...
```

文件结构

在本项目中我按照 10%，90%的比例把 train 数据集分为两部分：

- 训练数据集(包含两个文件夹)
    - dog（包含需要参与训练的 11250 张狗的照片）
    - cat（包含需要参与训练的 11250 张猫的照片）
- 验证数据集(包含两个文件夹)
    - dog（包含需要参与训练的 1250 张狗的照片）
    - cat（包含需要参与训练的 1250 张狗的照片）

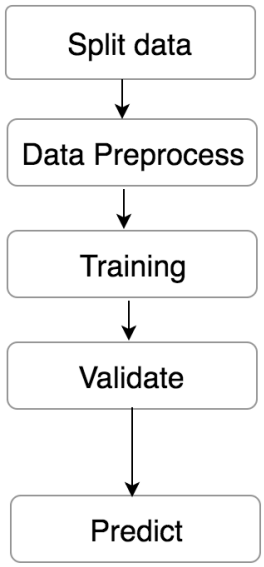
（2）数据预处理

其中包含两部分：ResNet50 所需要的图片大小是(224,224,3)，而训练集里的图片大小不相同，所以需要把图片进行 resize；考虑到当

前的数据集不是很大，所以需要通过 augment 图像增广技术来扩充数据集。这里使用 Keras.preprocessing.image.ImageDataGenerator 方法来完成这两部分数据预处理。

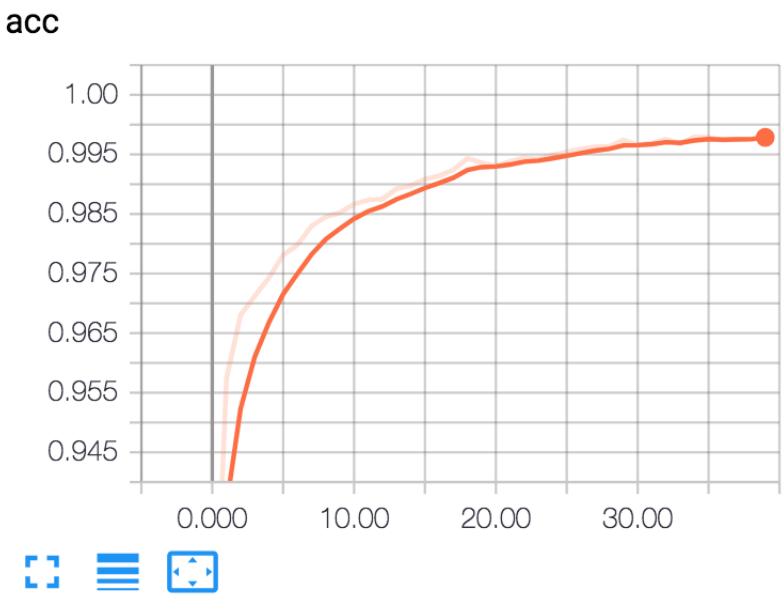
执行

本项目的代码直接运行在Aws p3.2xlarge ec2 instance上，一共运行40个epoch，并且保存每个 epoch 产生的 weights 结果，通过观察tensorboard中的图表结果选择表现最佳的模型。执行过程如下：

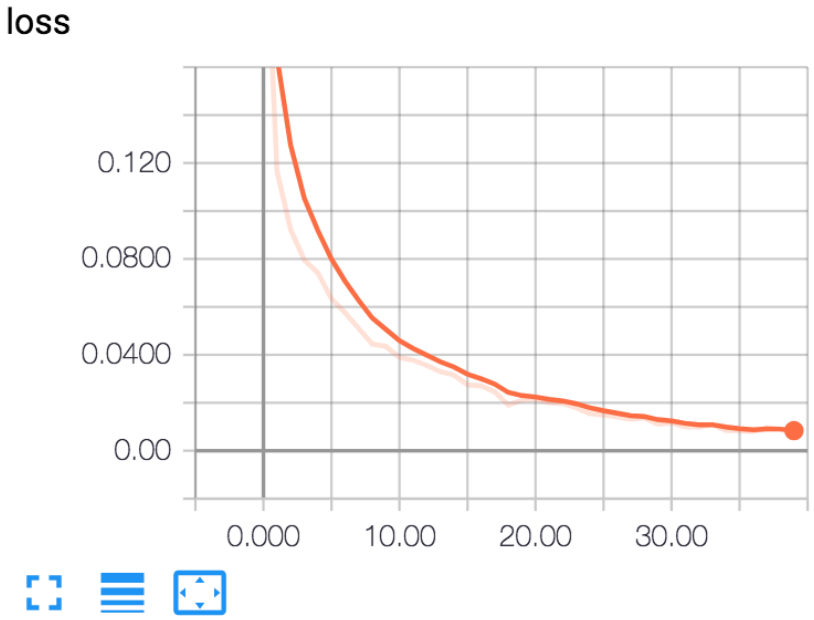


四．结果

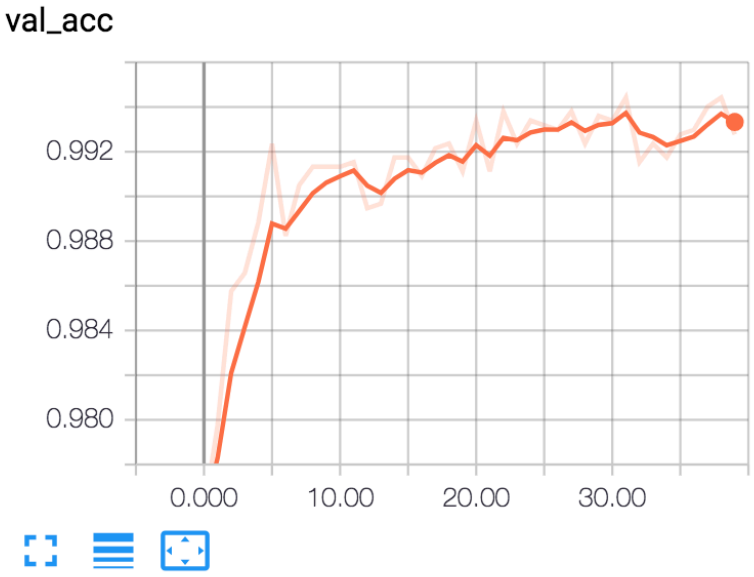
下图是 tensorboard 对运行过程结果展示：



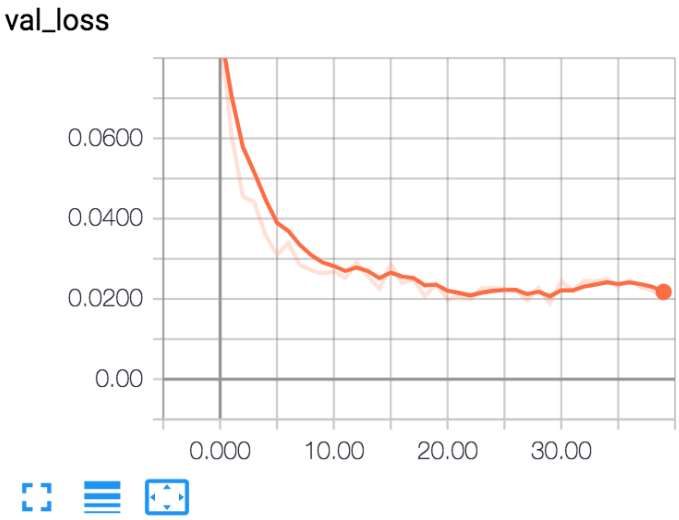
training_acc



training_loss

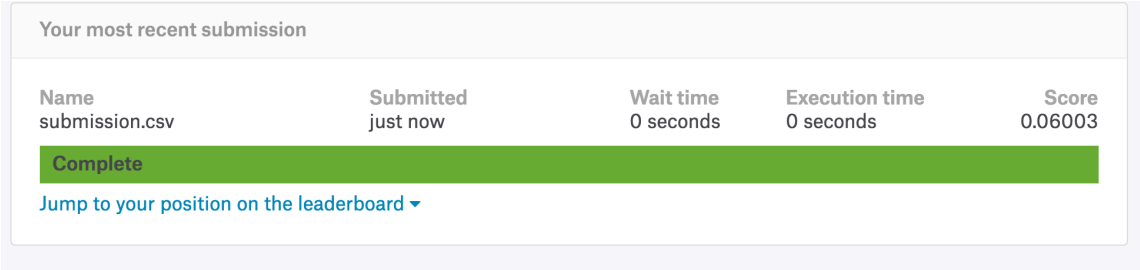


validation_acc



validation_loss

使用 epoch39 产生的 weights 结果测试 test 数据集，得到了 predict 结果 submission.csv 文件，下面是上传 submission.csv 到 kaggle 的结果：



五．项目结论

根据结果显示可以得出本次项目用到的模型是一个满足要求的模型。

对项目的思考

1. 在 AWS 上运行之前，尝试着在本地跑训练时多次导致 out of memory，所以 batch size 的选择是非常重要的。太小可能导致 overfit，太大就对内存要求比较高，如果不是足够大的内存，可能就会发生 out of memory 的问题。
2. training epoch 的选择以及如何保存 weights 结果。一开始训练时 epoch=200，我的想法是通过 tensorboard 看趋势，在 200 个 epoch 中选择最好的结果。但是实际证明这个是非常错误的选择，非常浪费时间。在 Aws 完成一个 epoch 需要大概 10 分钟，跑完 200 个 epoch 就需要 2000 分钟，不管是从时间还是金钱上 cost 太大。而且只保存保存最后一次运行的 weights 结果，也就是说即使我发现了最好的结果，有可能还得重新从头开始执行一次，因为当你发现最好的结果

的时候有可能已经被接下来的 epoch 结果覆盖掉。后来我改为 40 个 epoch，保存每个 epoch 运行后的 weights 结果。

六．参考文献

[1] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In IEEE Int' l Conf. Computer Vision and Pattern Recognition,2009.

[2] A. Krizhevsky, L. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Proc.Neural Information Processing Systems,2012.

[3] kaggle 猫狗大战手记，
http://moverzp.com/2018/01/21/kaggle%E7%8C%AB%E7%8B%97%E5%A4%A7%E6%88%98%E6%89%8B%E8%AE%B0/

[4] Deep Residual Learning for Image Recognition, https://arxiv.org/pdf/1512.03385.pdf

[5] Dogs vs. Cats Redux: Kernels Edition