机器学习纳米学位

毕业项目 wzyanqi

优达学城

2018 年 4 月 16 号

一、问题的定义

研究背景

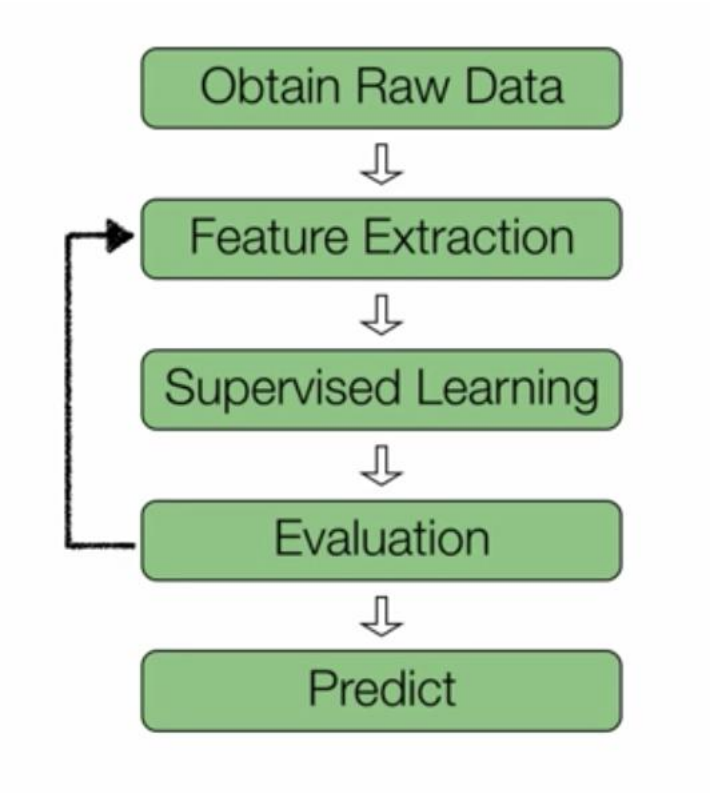
现如今深度学习已经广泛运用在各个领域，近十年来，深度学习取得了快速的发展以及长足的进步，通过深度学习来进行图像识别已经在已经取得了巨大的成功。2015 年微软开发的基于深度卷积神经网络（CNN），在 ImageNet1000 挑战中首次超越了人类进行对象识别分类的能力。2016 年 3 月人工智能围棋比赛，DeepMind 公司开发的深度学习围棋程序 AlphaGo 战胜了世界围棋冠军、职业九段选手李世石，这是人类第一次在围棋上战胜世界冠军。2018 年 Waymo 开发的基于深度学习的无人车正式宣布营业，宣告了无人驾驶时代的真正到来。

项目概述

本文项目来源于 kaggle 竞赛 Dogs vs Cats[1]通过训练集的学习让计算机能够分别识别猫与狗的图像。这是一个二元分类项目。之所以选择猫与狗作为我的毕业项目是因为这其中会涉及到包括图像处理、神经网络等方面的内容。

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：



实施流程

评价指标

本文采用的评价指标与 kaggle 比赛相同，最后所得到的 *Logloss* 越小越好。

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

n：测试集样本数

$\hat{y}_i$：该图片被预测为狗的概率

y_i：如果该图片为狗，则为 1，反之为 0。

二、分析

数据的探索

kaggle 一共提供了两个数据集 train 与 test。其中训练集包含 25000 个样本，cat 和 dog 各 12500 个。

训练集

由于样本为图像数据，大体看了一遍数据集还是发现了一些异常样本。比如 cat.4688，cat.7377，dog.4367，dog.5604 等等。

cat.4688

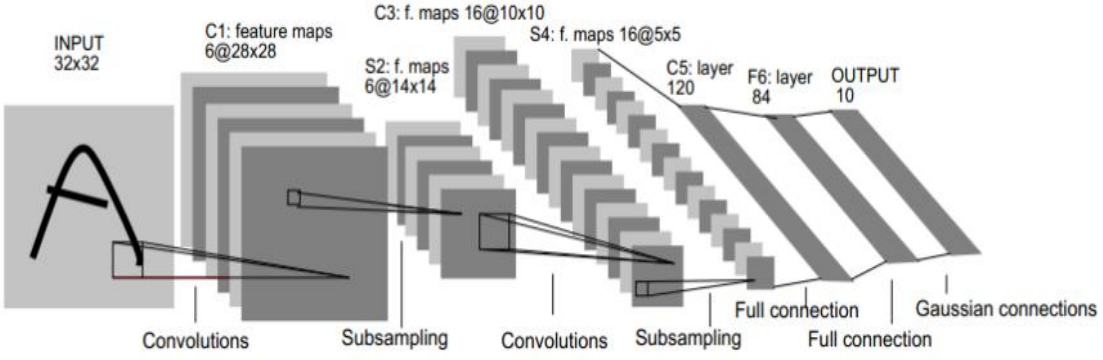
dog.5604

测试集包括 12500 个样本，但并未标注猫与狗，用于最后的模型测试与验证。

测试集

算法与技术

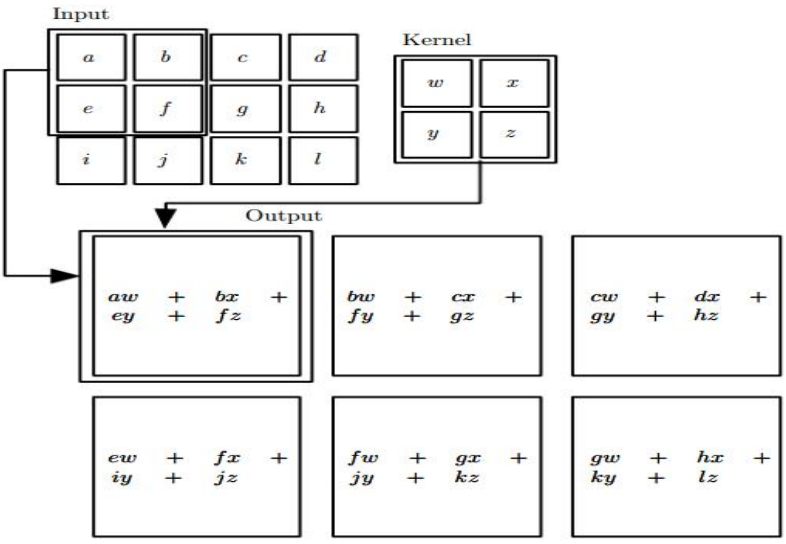
在本项目中我主要使用的是卷积神经网络来构建我的算法模型。卷积神经网络是一种多层神经网络，擅长处理图像特别是大图像的相关机器学习问题。卷积网络通过卷积，池化，全连接等一系列方法，成功将数据量庞大的图像识别问题不断降维，最终使其能够被训练。下面我将分别介绍卷积、池化、RELU 激活函数、dropout 层、迁移学习等知识点。



LeNet-5 卷积神经网络[2]

（1）卷积层

通过卷积操作，将计算量庞大的图像识别问题不断降维，最终使其能够被神经网络训练。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[3]。



卷积原理

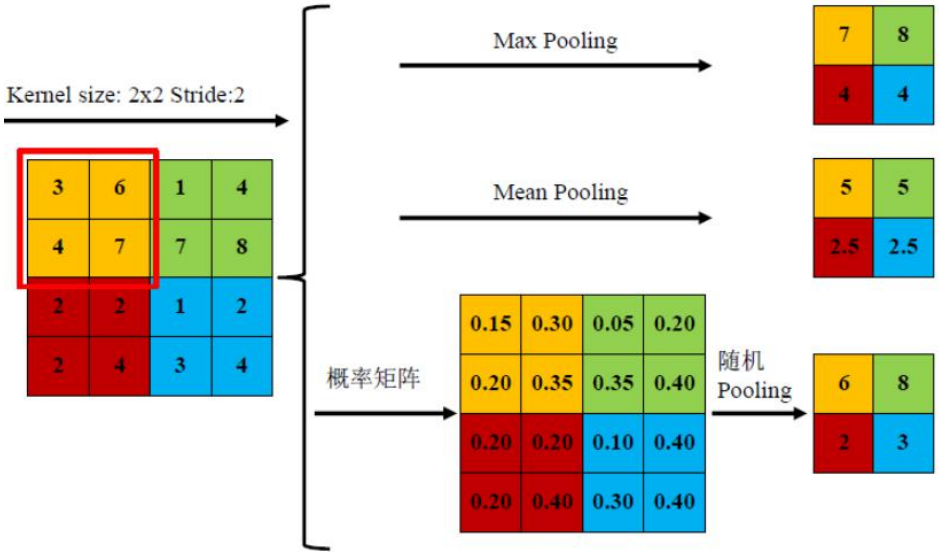
（2）池化层

进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[4]。池化的通常有以下三种方式：

1、 mean-pooling，即对邻域内特征点只求平均，对背景保留更好；

2、 max-pooling，即对邻域内特征点取最大，对纹理提取更好；

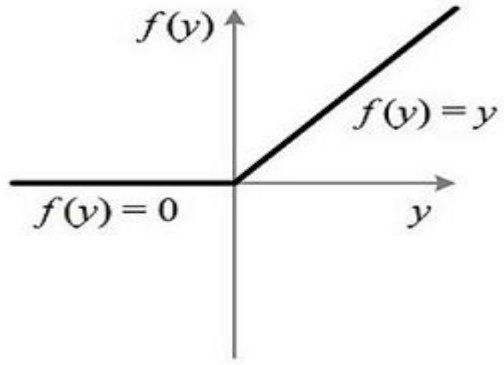
3、 Stochastic-pooling，介于两者之间，通过对像素点按照数值大小赋予概率，再按照概率进行亚采样；



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。相比于其它激活函数来说，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[5]。

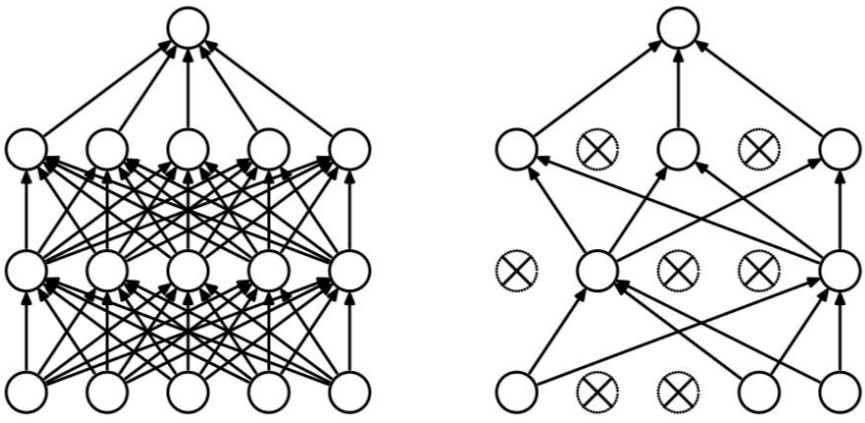


$$ReLU(x) = \begin{cases} x & if x > 0 \\ 0 & if x \leq 0 \end{cases}$$

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。

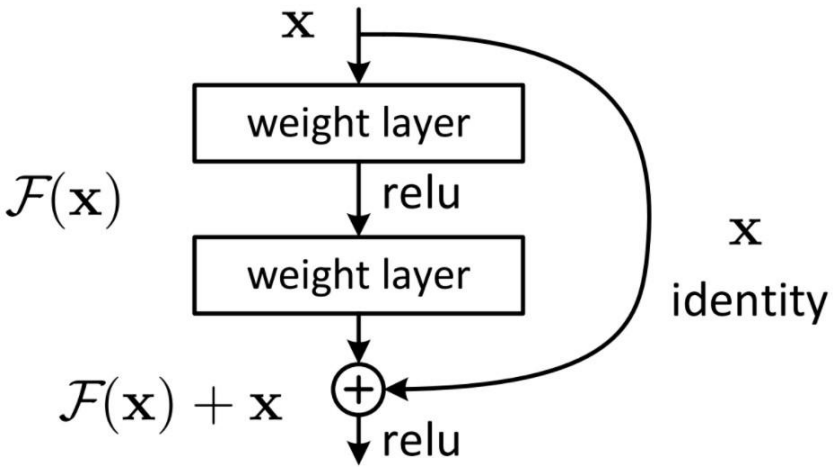


Dropout 原理

（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。

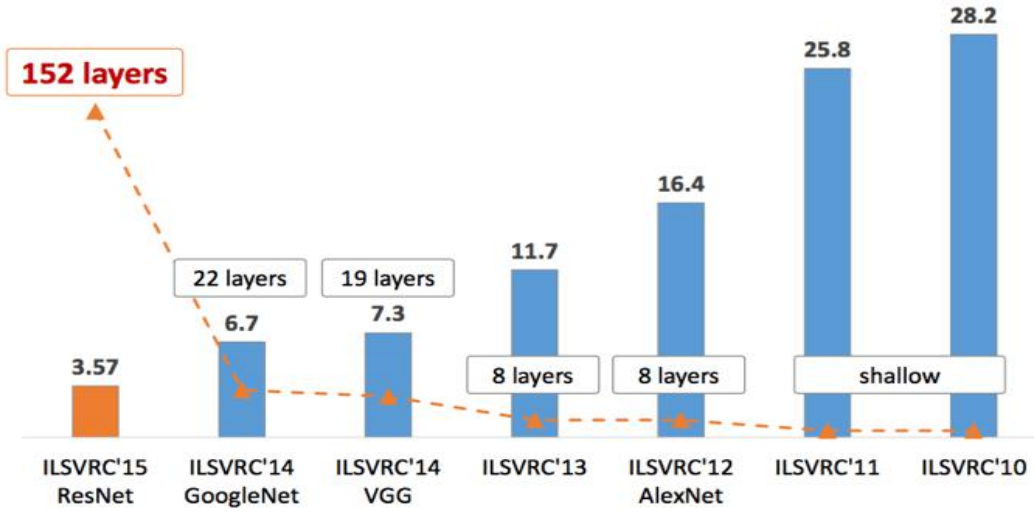
综合以上技术方法，在本次项目我决定采用了经典的 ResNet 模型，ResNet 在 2015 年提出以来，曾在 ImageNet 比赛 classification 任务上获得第一名，因为它“简单与实用”并存，之后很多方法都建立在 ResNet50 或者 ResNet101 的基础上完成的，检测，分割，识别等领域都有着广泛的运用。



ResNet 模型原理

基准模型

关于基准模型，我参考了 ILSVRC 历年的 Top-3 错误率模型（VGG-16、PReLU-net、GoogLeNet）为基准模型。



ILSVRC 历年的 Top 错误率模型

而目标模型 ResNet50 模型的表现超过了以上三种模型[6]，所以最终选择 ResNet50 模型。

model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PReLU-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

ImageNet 比赛错误率

最终目标

目前 kaggle 猫狗大战项目上的 public 榜单上有 1300 人，我的目标能前进入 10%，也就是超过第 130 名（0.6114）。

129	▾ 35	Jeremy Howard		0.06086	11	1y
130	▾ 35	RaviKiranK		0.06114	35	1y
131	▾ 35	Reziproke		0.06127	4	1y
132	▾ 35	mathieuzaradzki		0.06149	32	1y
133	▾ 34	Mouatez		0.06240	39	1y
134	▾ 33	anokas		0.06320	8	2y

10%榜单，截止 2018/4/9

三、方法

数据预处理

（1）因为 kaggle 给我们的 training 图片名是根据 “cat/dog + num.jpg” 命名的，各 12500 张。所以首先需要根据文件名的关键字 cat 和 dog 将训练集根据名字进行分类。

（2）为了增加效果，在使用 glob 读取图片路径后，我就使用了 shuffle 将图片路径进行打乱，之后再进行预处理操作。

（3）由于 ResNet50 所需求的图片大小为(224, 224, 3)，而 training set 里的图片大小各不相同，需要使用了 cv2.resize 命令统一输入给模型。同时为了节约内存分配，我预先还对 training 数据矩阵的大小进行了预分配。

第一次执行

在完成项目中，遇到了许多困难，最让我收获最大的是以下五个问题的解决：

（1）预处理资源不足

Training set 中包含 25000 张图片，每张图片又需要被 resize 成(224, 224, 3)。这就意味着至少需要存储的矩阵大小为(25000, 224, 224, 3)，在每次预处理的时间都会非常长，即使用 np.load 载入之前预处理过的数据，也经常会将 jupyter 卡死。后续我将原有的预处理代码做出了两点改进，一、training 数据进行矩阵大小的预分配取代原来的 append 命令；二、将 dtype 改成 uint8。经过这两点改进后，虽然大大缓解了预处理资源不足的情况，但每次预处理数据时间也要 600s，可见本任务的数据量之大。

（2）训练速度慢

在做之前的机器学习项目中，由于数据量以及计算量都不大。我都是采用自己的笔记本电脑使用 CPU 来训练模型。可这次当我运行 ResNet50 模型后，发现根本跑不动，训练一轮的时间需要 10 个小时以上。为了解决这个问题，我安装了 CUDA9.0 + Cudnn + tensorflow-gpu 改为 GPU 来训练自己的模型。

（3）ResourceExhaustedError

使用 GPU 进行训练的时候，遇到了 ResourceExhaustedError。由于我的电脑显存只有 4GB，模型的参数已经占用了大量的显存资源，无法一次训练过多的数据。我尝试过使用 fit_generator 也无法解决，最后通过调小 batch_size 为 16，才勉强解决了这个问题。虽然调小了 batch_size 会大大影响训练速度，但还是比使用 CPU 来训练的效率更高，训练一轮的时间大概为 70 min。

（4）optimizer 选择

之前对于 optimizer，我一直没有什么研究。一般都是选择 Adam 方法。在本项目中，我发现 Adam 算法对我并不合适。原因有两点，其一，Adam 算法会占用更多的 GPU 资源，而我本身的 GPU 资源已经非常紧缺了。其二，我发现 Adam 算法在本项目中收敛速度并没有我想象的那么快，当准确率达到 90%后，收敛的非常慢。最后我参考了 Keras 文档案例，将 learning rate 设为 0.0001，momentum 设为 0.9。运用该 optimizer ，只需要两轮训练就能使得准确率超过 97%。

$$opt = SGD(lr = 0.0001, momentum = 0.9)$$

（5）设置 Dropout 比例

在 ResNet50 基本模型的基础上，我添加了 Flatten，全连接层，Dropout 来构建我的最终模型。加入 Dropout 是为了防止模型过拟合的发生，但 Dropout 的比例，只能依靠经验与测试，不断调整才行。

在克服上述困难后，我的模型终于可以正常的被训练了。为了能获得更好的准确率，我总共进行了 10 轮训练，将最终生成的模型对 test 集进图片进行了预测，并生成 csv 文件参与 kaggle 评估。

第二次完善

根据 mentor 的建议，我意识到了自己第一次提交的模型的缺点，分别做出了如下修改，并且取得了更好的成绩。

（1）替换预处理方法

之前的模型，我使用的是自己写的预处理方法。后来经过 mentor 提醒，应该要使用 Keras 中 ResNet50 中自带的预处理方法。ResNet50 预处理图片的方法是将 RGB 的图片转成 BGR 然后减 ImageNet 均值。

（2）数据清理

由于整个训练集的有 25000 张图片，通过人工审查图片集不太现实。我使用了最初的模型，对整个训练集进行了预测。删除了所有预测错误的图片，大约有 68 张图片被删除了。我这里将所有预测错误的图片都删除了，并没有对图片的正确性进行分辨。因为即使 68 张全是正确而误删除，但相比于整个训练集 25000 张图片并没有大碍。

（3）Dropout 数值设置

根据 mentor 的建议，我针对现有的模型重新进行了 Dropout 数值的尝试，设置了 0.5 以及 0.75 两个不同的值进行比较。发现在模型训练过程中，两者能达到的准确率基本相同，但在提交最终结果的时候发现当设置为 Dropout = 0.75 的 Loss 会更小一点。

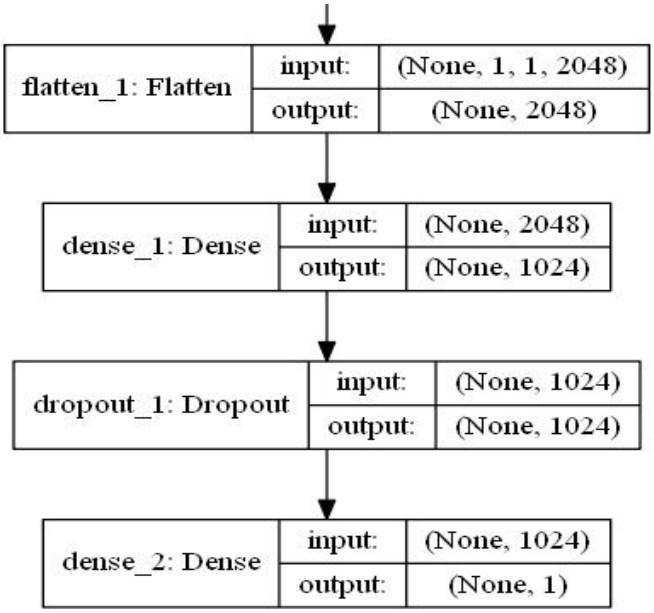
（4）代码风格

之前的我以为良好的 Python 代码风格需要在所有代码的等号两端都要加上空格，经过 mentor 提醒后了解，其实在函数定义里的等号两端是不需要添加空格才能满足 PEP8 的规定。

四、结果

模型的评价与验证

我的最终模型在 ResNet50 基础上添加了 Flatten、Dense、Dropout 层如下图所示（附件 model.png 为完整模型），经过训练后的最终模型的准确率超过了 99%。



模型最后四层

合理性分析

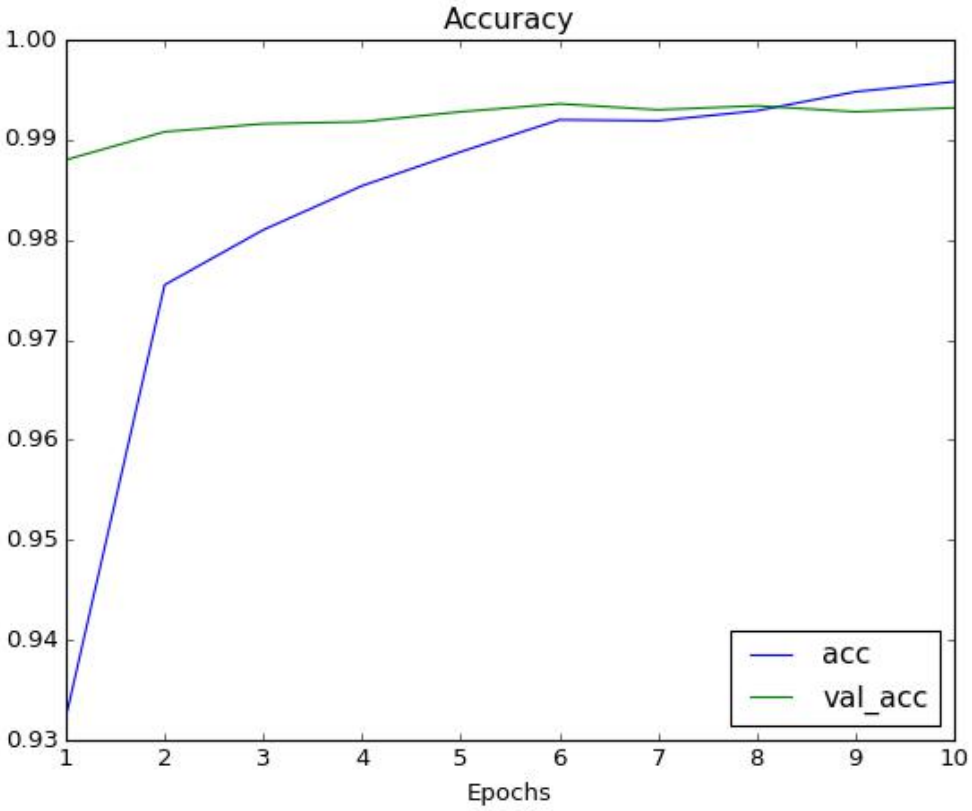
经过若干轮训练后，使用最终的模型对 test 集进行预测，并生成 csv 文件参与 kaggle 评估，最终的评分为 0.06001，比上次提交略微进步了一些，达到了之前定下的目标。

最终结果

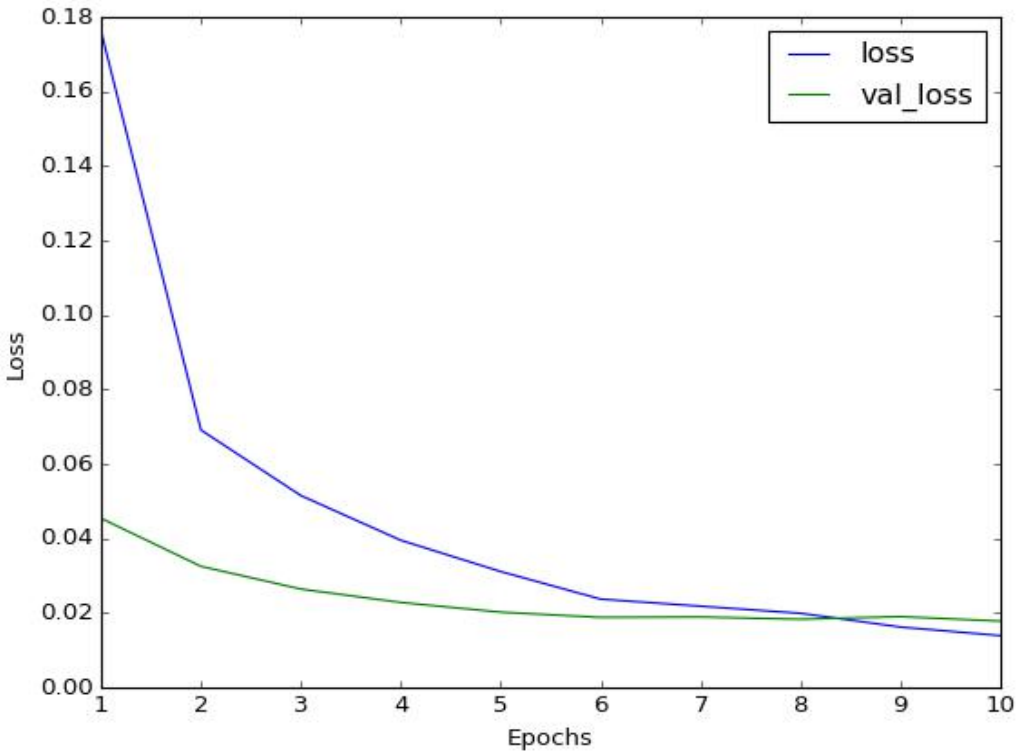
五、项目结论

结果可视化

由下图可以看出，随着训练次数的增加，训练与测试准确率不断提高，一直攀升到 99%+。同时训练与验证 loss 不断下降，达到了 0.02 以下。



Accuracy vs Epochs



Loss vs Epochs

对项目的思考

通过本次项目，我增加了自己对神经网络的理解，主要包含以下两点收获：

（1）预处理的重要性

数据的预处理对任何模型都非常重要。好的预处理可以大大节约计算机内存，减少训练时间，提高模型准确率。如果在预处理方面“偷懒”的话，模型会加倍的“报复”的。

（2）optimizer 参数

不同的 optimizer 选择，对模型的影响不同。，没有一个万能的方法适用于所有项目，只能具体问题具体分析。

需要作出的改进

虽然已经达到了初始目标，在现有的基础上，但仍然有以下几种方法可以改进并提高模型准确率的方法：

（1）模型融合

使用单一模型的方法，通过各种办法提高准确率的方法有限。如果通过模型融合的办法，可以集合多种模型的优点，进一步提高准确率。

（2）增加 train 集数据

可以通过翻转图片，改变图片明暗度等方案，来增加 train 样本数。

六、参考文献

1. Dogs vs. Cats Redux: Kernels Edition
2. Lecun, Y., et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11(1998):2278-2324.
3. 许可. 卷积神经网络在图像识别上的应用的研究. Diss. 浙江大学, 2012.
4. 刘万军, 梁雪剑, and 曲海成. "不同池化模型的卷积神经网络学习性能研究." 中国图象图形学报 21.9(2016):1178-1190.
5. 蒋昂波, and 王维维. "ReLU 激活函数优化研究." 传感器与微系统 2(2018).
6. He, Kaiming, et al. "Deep Residual Learning for Image Recognition." Computer Vision and Pattern Recognition IEEE, 2016:770-778.