优达学城
UDACITY

# 纳米学位论文

## 基于 Android 和微信小程序端的猫狗图像分类

课程： 机器学习（进阶）

作者： 李 果

# 目录


## 1 绪论

1.1 项目概述.....1

1.2 问题陈述.....3

1.3 评价指标.....3

## 2 研究方法

2.1 数据分析.....4

2.2 探索性可视化.....5

2.3 算法和技术.....7

2.4 基准模型.....8

## 3 模型训练

3.1 数据预处理.....9

3.2 从头训练.....12

3.3 迁移学习.....15

## 4 评价与部署

4.1 模型的评价.....17

4.2 在 Android 上的部署.....18

4.3 在微信小程序上的部署.....19

## 5 项目结论

5.1 结果可视化.....20

5.2 对项目的思考.....21

5.3 可以作出的改进.....22

# 第一章 绪论

## 1.1 项目概述

两次工业革命后，机器在诸多领域取代了人力，社会生产力大幅上升。但机器超越了人类四肢的力量，却仍难以企及人类大脑的智慧。但在二战中，对密码破译、导弹弹道计算和原子弹研发的需求促使了电子计算机的诞生。1946 年，第一台电子计算机在宾夕法尼亚大学问世。在此之后，1948 年诺伯特·维纳提出了控制论，克劳德·香农提出了信息论。它们为接下来人工智能的研究奠定了基础。1952 年，手冢治虫开始连载《铁臂阿童木》漫画，人们已经开始畅想一个充满着人造智能机器人的未来世界。

### 1.1.1 人工智能与机器学习

1956 年，在达斯茅斯会议上，约翰·麦卡锡正式提出了“人工智能（Artificial Intelligence）”的概念：让机器的行为看起来像是人所表现出的智能行为一样（这里的“行为”应理解为决策行为而非肢体动作）。后来一般教材将人工智能的领域定义为：智能主体（intelligent agent）的研究与设计。人工智能涉及范围极广，目前没有统一的原理或范式指导其研究，而实现“强人工智能”是它的长远目标。强人工智能，被认为是具备或超越人类同等智慧，能表现正常人类所具有的所有智能行为；相对的，弱人工智能，只能解决特定领域的问题。目前的研究仍停留在弱人工智能级别。

“机器学习（Machine Learning）”是实现人工智能的重要途径，一种定义是：对通过经验自动改进的计算机算法的研究。可见，机器学习就是将控制论的思想（信息反馈）应用于计算机算法上来研究人工智能。1952 年，阿瑟·萨缪尔开发了一个具有自学习能力的西洋跳棋程序[1]，并在 1962 年战胜了美国著名选手尼雷，在当时引起了轰动。这个程序使用的是强化学习算法，除此之外，机器学习的主要算法还包括：神经网络、最近邻、决策树、集成学习、支持向量机等。按基本思想来划分，各算法可分为符号主义（数学）、连接主义（网络）和行为主义（控制）；按输入数据来划分，可分为有监督学习、无监督学习等。

### 1.1.2 神经网络与深度学习

连接主义来源于仿生学，认为智能是由大量神经元所组成的网络产生的。1943 年，沃伦·麦卡洛克和沃尔特·皮茨提出了 MP 神经元模型[2]。40 年代后期，唐纳德·赫布根据神经可塑性的机制提出了赫布型学习[3]，这被认为是一种典型的非监督式学习。1956 年，弗兰克·罗森布拉特创造了感知机[4]，这是一种多层的神经网络。1969 年，马文·明斯基和西摩·帕尔特提出基本感知机无法处理异或回路[5]，使神经网络的研究陷入低潮。1974 年，保罗·韦伯斯创造了反向传播算法[6]，解决了异或问题。1985-86 年，几位研究人员先后提出了感知机与反向传播相结合的理念[7]，为多层神经网络的训练奠定了基础。

1989 年，杨·勒丘恩将反向传播算法用于识别手写数字[8]，提出了“卷积神经网络”的概念。1998 年，勒丘恩提出了 LeNet-5 模型[9]，这是第一个正式的卷积神经网络模型。但由于当时的硬件性能限制，相关研究随后趋于停滞。2006 年，杰弗里·辛顿提出了深度信念网络[10]，以解决深层网络训练中的梯度消失问题，开启了“深度学习”的研究。2012 年，辛顿课程组的 AlexNet 在 ImageNet 图像识别比赛中以巨大优势夺得冠军[11]，引发了深度学习和卷积神经网络的研究热潮。随后又诞生了 VGG、Inception 和 Resnet 等更优秀的卷积神经网络。

### 1.1.3 本研究概述

实现深度学习常用的框架有 MXNET、Caffe、Tensorflow、Torch、Theano 等，目前最流行的是 google 推出的 Tensorflow[12]。16 年 8 月，Google 团队又针对 TensorFlow 开源了 TF-slim 库，它是一个可以定义、训练和评估模型的轻量级软件包，也能对图像分类领域中几个主要网络进行检验和定义模型。同时，伴随着移动智能终端的兴起，业界正致力于将深度学习部署在移动端上。微信是世界上广泛应用的一款移动通信 app，在中国几乎覆盖了所有的智能手机。而倡导“用完即走的”的“微信小程序”在 17 年初推出后，一年时间日活已达到 1.7 亿。

本研究试图使用卷积神经网络 Resnet[13]解决一个二分类图像识别问题，达到较高的识别准确度（低 loss）。在研究过程中先使用小训练集，充分尝试多种数据增强方法和 Resnet 模型参数，以对卷积神经网络的训练有较深入的理解。之

后再使用迁移学习进行训练，以和从头训练的结果进行比较，并期望得到更高的准确度。最后尝试将模型移植到 Android 端和微信小程序上，以取得一定的实际应用价值。本研究的数据集来自 kaggle 猫狗大战（https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition）。

## 1.2 问题陈述

训练一个可以处理二分类问题（是猫还是狗）的神经网络，并将模型和参数导出，部署在 Android 应用程序和微信小程序上。在 Android 应用程序上，对选取的照片以及摄像头的实时图片在移动端进行识别。在微信小程序上，拍摄或使用相册中的图片上传到后台，在服务端进行识别，将结果返回给小程序展示。

训练时，框架选用 Tensorflow，模型选用 Resnet。训练过程中，尝试不同的数据增强方法，并对学习速率等参数进行调节，提高训练准确度。再使用迁移学习，尝试不同的微调结构。最后选择最优的模型和参数，对全训练集进行训练，并把结果上传至 kaggle，排名进入 public leaderboard 前 10%则成功。把训练好的模型和参数导出为 pb 文件，基于 google 提供的 Tensorflow Android Camare demo（https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/android）进行修改，得到可以识别猫狗的 Android app。开发一个简单的微信小程序，将图片传到后台并展示识别结果。在 centos 上使用 node.js 搭建服务端，调用 python 程序对传来的图像进行猫狗识别。

## 1.3 评价指标

kaggle public leaderboard 排名进入前 10%，即 Logloss<0.06127。log 损失函数常用于分类问题中，而排名进入前 10%意味着得到了一个较为精确的模型。

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

其中：n 是测试集中图片的数量；$\hat{y}_i$ 是模型给出的输入图片为狗的概率；$y_i$ 代表输入图片的真实标签，是狗时为 1，是猫时为 0。

# 第二章 研究方法

## 2.1 数据分析

下载的训练集和测试集图片格式都为 jpg。训练集共 25000 张图片，在文件名中进行了类别的标注。标注分为 cat、dog 两类，每类数量各占一半为 12500 张。各图片尺寸不一，在 50*49 到 1023*768 之间，中值约为 400*400。测试集共 12500 张图片，无标注，尺寸在 50*49 到 500*499 之间。为了得到具体的尺寸信息，编写程序对训练集、测试集的各图片尺寸出现次数进行进一步统计绘图。

图 2-1 不同品种的猫（狗）脸部

从上面的猫脸和狗脸对比图可以看到，两者的主要区别是：猫的眼睛较大呈彩色，瞳孔一般较小，鼻子较为娇小。并且不同品种的猫之间脸部差异很小，而不同品种的狗差异更大。事实上，猫的品种数量比狗的也更少——国际爱猫联合会承认 42 个猫咪品种，而世界犬业联盟则收录了 340 种狗狗，这可能是由狗的驯化历史更长造成的。

在 kaggle 讨论版中有人提出了训练集和测试集中都存在异常数据。为了较系统的处理训练集中的异常值，本研究使用训练好的基准模型对训练集的 25000 张图片进行了测试，取出其中较有可能为异常数据的部分图片。为了获取猫狗特征都不明显的数据，将图片按所得 logits 两个值的平方和升序排列，取前 500 张图片；为了获取标注错误的数据，将图片按模型推断的错误程度排列，从猫和狗的标注图片中各取 250 张（可能与前面的 500 张有重复）。

接下来对这些图片进行直接观察，对其中的异常数据按 *无法识别* 和 *标注反了* 两种情况进行统计，其中无法识别又可细分为图中无猫狗、猫狗都有、图像过于抽象和难以区分猫狗（因为人类的观察能力限制，可能存在一定的主观性）四类。具体统计数量见下表：

| 无猫狗 | 猫狗都有 | 过于抽象 | 难以区分 | 标注反了 |
|---|---|---|---|---|
| 15 | 15 | 6 | 28 | 7 |

表 2-1 各类异常图片的统计

具体的异常数据文件名请见项目中的 analysis_outliers.py 文件。与 kaggle 猫狗大战讨论版（https://www.kaggle.com/c/dogs-vs-cats/discussion/5892）中提到的异常数据相比，讨论版上的 21 张异常训练集图片中这里包含了其中的 15 张。对于无法识别的几种情况，将相应的共 64 张异常图片从训练集删除；对于标注反了的数据，将 7 张异常图片的标注进行更正。经此处理后，全训练集的图片数量从 25000 张变为 24936 张。

## 2.2 探索性可视化

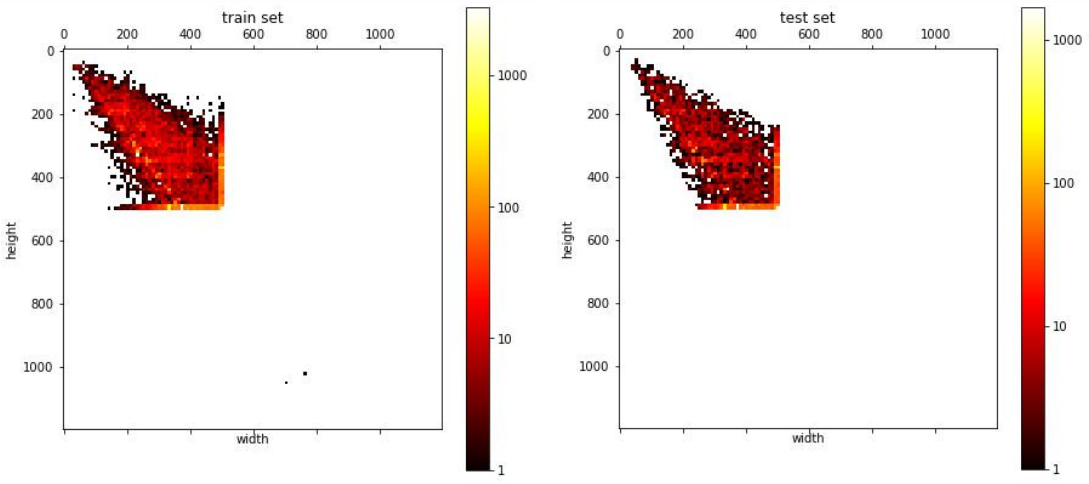



图 2-2 训练集与测试集图片尺寸分布热力图

由上图可知：训练集大部分图片分辨率都在 500*500 以下，长宽比在 1/2 到 2 之间，有少数大分辨率和大比例长宽比图片。而测试集基本上所有图片分类率都在 500*500 以下，长宽比都在 1/2 到 2 之间。

| 宽 | 500-510 | 490-500 | 490-500 | 500-510 | 370-380 |
|---|---|---|---|---|---|
| 高 | 370-380 | 370-380 | 330-340 | 330-340 | 490-500 |
| 数量 | 3251 | 3189 | 482 | 452 | 329 |

表 2-2 训练集图片数量最多的 5 个尺寸区间

| 宽 | 500-510 | 490-500 | 500-510 | 490-500 | 370-380 |
|---|---|---|---|---|---|
| 高 | 370-380 | 370-380 | 330-340 | 330-340 | 490-500 |
| 数量 | 1679 | 1674 | 222 | 190 | 173 |

表 2-3 测试集图片数量最多的 5 个尺寸区间

上面是训练集和测试集中图片数量前 5 的几个尺寸区间，可以看到：训练集和测试集都有 1/4 以上的图片分辨率位于 宽 490-510 高 370-380 的范围内，并且剩余很多图片的分辨率也都在 330*330 以上。

为了进一步获得对数据的直观映像，从训练集和测试集中分别随机抽取 100 张图片样本，对猫狗主要区分特征、猫狗大小占整个图片的比例进行统计。对于各特征，首先不能清晰辨别时不列入计数；其次，不具有明显区分度时不列入计数，如猫的眼球像狗一样占眼睛比例较大时、鼻子像猫一样相比眼睛较大时，狗的耳朵像猫一样立起呈三角形时。对于猫狗占比大小，取猫狗长宽与图片长宽比例较大的一项进行统计，分为 100%-75%，75%-50%，50%-0%三个区间。

| 特征 | 总数 | 眼睛 | 鼻子 | 耳朵 |
|---|---|---|---|---|
| 训练集-猫 | 50 | 42 | 46 | 47 |
| 训练集-狗 | 50 | 39 | 50 | 37 |
| 测试集-猫 | 59 | 53 | 51 | 58 |
| 测试集-狗 | 41 | 32 | 39 | 29 |

表 2-4 训练集与测试集样本 猫狗特征统计

由上表可见，对于训练集，鼻子和耳朵都是识别猫的显著特征，识别率可达 94%；鼻子也是识别狗的显著特征，识别率达 100%。对于测试集，100 张图片中 59 张是猫，占比约 60%，只有耳朵是识别猫的显著特征，识别率约 97%；而鼻子仍是识别狗的显著特征，识别率约 95%。所以，通过鼻子和耳朵的单项特征可以进行较好的识别，但要想达到非常高的准确率，需要对各特征进行综合考虑。

| 占比 | 总数 | >75% | >50% | >0% |
|---|---|---|---|---|
| 训练集-猫 | 50 | 46 | 3 | 1 |
| 训练集-狗 | 50 | 46 | 4 | 0 |
| 测试集-猫 | 59 | 54 | 2 | 3 |
| 测试集-狗 | 41 | 35 | 6 | 0 |

表 2-5 训练集与测试集样本 猫狗占图像大小比例统计

由上表可见，训练集和测试集的猫狗大小占比大都在 75%以上，绝大部分在 50%以上。也就是说，猫狗是绝大部分图片的主体，数据质量较好。无需过多考虑图像中的其它干扰因素，并且可以在进行数据增强时对图像边缘进行裁剪。

## 2.3 算法和技术

ResNet 模型于 2015 年被提出，在当年的 ImageNet 比赛 classification 任务中取得了第一名[14]。它是图像领域中目前广泛使用的神经网络，包括 17 年发布的“围棋上帝”Alpha zero 也使用了 ResNet[15]。Resnet 通过构建残差模块，解决了网络层数加深后的梯度消失问题，得以让极深层的神经网络的训练成为可能。

本研究先使用 Resnet 从头训练数据，采用小数据集分别对模型深度、优化器、学习速率等进行调参。并且在数据增强的预处理中，除了图像标准化、随机剪裁、随机翻转这几个经常使用的方式外，还进行了额外的随机化尝试。

| 数据增强 | 图像标准化，随机剪裁，随机翻转，随机遮挡等 |
|---|---|
| 模型深度 | Resnet18，Resnet34，Resnet50 |
| 优化器 | GradientDescent，Momentum，Adam |
| 学习速率 | 不同固定速率，不同衰减方式的可变速率 |

表 2-6 从头训练的参数选择

再使用迁移学习，基于 slim 的 Resnet 对 ImageNet 的预训练模型开始训练，并尝试不同的模型微调方式。最后，将两种训练方式的结果进行比较，对全训练集进行训练，提交到 kaggle 上查看分数。

对于 Tensorflow 在 Android 上的部署，google 提供的 Tensorflow Android Camare demo（https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/android）是一个很好的样例。Demo 包含四个程序，其中的 TF Classify 可以对图像进行实时分类。本研究在 TF Classify 的 Classifer 文件基础上，对界面进行了重新编写，并提供猫狗识别的 pb 文件。

开发微信小程序时，由于微信小程序只允许访问 https 的合法域名，且不能带有端口号。所以需在服务器上配置 ssl 证书，并使用 https 默认的 443 端口号。在小程序中，使用 chooseImage 和 uploadFile API 来选择和上传图片。在服务端中，由于任务较为简单，故使用轻量高效的 node.js 来快速实现。在 node.js 中将收到的图片保存为文件，再调起 python 程序对图片文件进行识别，最后将结果返回给小程序展示。

## 2.4 基准模型

选用经 ImageNet 数据训练过的 Vgg16 网络作为基准模型[16]，Vgg16 使用 TF-Slim 中的实现，预训练模型从（http://download.tensorflow.org/models/vgg_16_2016_08_28.tar.gz）处下载。在训练集中选取猫狗图片各 750 张进行训练，输入尺寸为 224*244，进行 32 次迭代，学习速率为 0.01，优化器为 rmsprop。最终在 kaggle 得到的 loss 为 0.11217。


图 2-3 基准模型 kaggle 评分结果

# 第三章 模型训练

## 3.1 数据预处理

为了把图片数据转换成 Resnet 可以接受的输入，按以下两个步骤进行处理：

- 读取数据文件夹，得到文件名和分类标签的列表。
- 按照文件名读取图片，根据当前是否在 Training 进行相应预处理，得到 batch_size 长度的图片和分类标签的 Tensor。

由于总训练集只有两万多张图片，数量相对较少，所以有必要进行数据增强。常见的数据增强方法有随机截取和随机水平翻转，此外，随机遮挡，随机对比度、饱和度、亮度、色调也都可以考虑。为了详细分析各个处理的影响，本研究先从不加数据增强开始，再逐步应用各方法考察其效果。具体操作是：

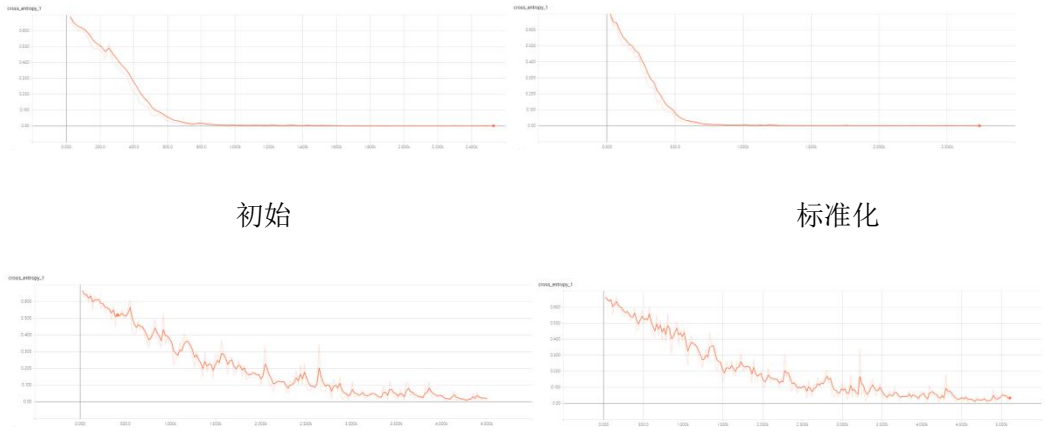
- 不加数据增强时，预处理只是把图片大小 resize 成 224*224。
- 尝试加上图像标准化。图像标准化进行的变换是：x = (x - mean) / adjusted_stddev，其中 x 为图片的 RGB 三通道像素值，mean 分别为三通道像素的均值，adjusted_stddev = max(stddev, 1.0/sqrt(image.NumElements()))。stddev 为三通道像素的标准差，image.NumElements()计算的是三通道各自的像素个数。
- 尝试使用其它数据增强方法。

原始图片　　标准化　　长宽分别截取 4/5　　长宽分别截取 3/4

随机水平翻转　　随机覆盖　　随机亮度与对比度　　随机饱和度与色差

图 3-1 各图像增强方法的处理效果示例

训练时，统一采用 Resnet_v2_18 模型，使用惯性为 0.9 的 Momentum 优化器，学习速率固定为 0.001，权重初始化函数为 variance_scaling，损失函数为 cross_entropy + 2e-4 * l2_loss。为了加快训练速度，使用只有 1500 张图片的小训练集。batch size 设为 64，最大迭代次数为 300 次。采用了 early stop，在原始图片和标准化的训练中，10 次迭代内训练 loss 未超过此前最小值时即停止。在其他启用了数据增强的训练中，因为训练集的数量相当于增加了，故 10 次改为 15 次。cross entropy 曲线如下，横轴是训练步数，纵轴是 cross entropy 值：



初始　　标准化

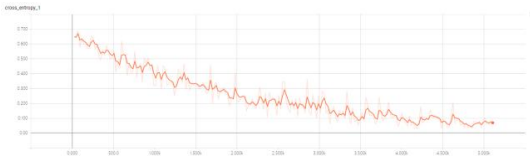
长宽分别截取 4/5　　长宽分别截取 3/4

图 3-2 不同数据预处理方式的 cross entropy 曲线

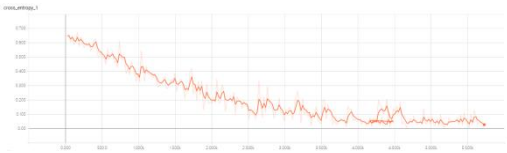
| 预处理方法 | 迭代次数 | 每轮迭代耗时 | 训练集准确度 | 验证集 loss | 验证集准确度 |
|---|---|---|---|---|---|
| 初始 | 110 | 7.4s | 100% | 1.0048 | 69.56% |
| 标准化 | 110 | 7.5s | 100% | 1.0463 | 68.21% |
| 长宽分别截取 4/5 | 195 | 7.5s | 99% | 1.1209 | 76.13% |
| 长宽分别截取 3/4 | 210 | 7.5s | 99% | 0.9796 | 76.81% |

表 3-1 不同数据预处理方式的训练与验证结果

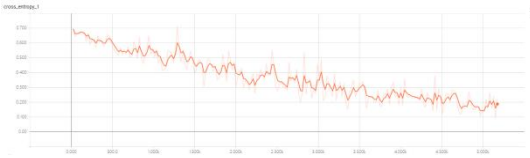
可以看到，对于初始处理，cross entropy 平滑的下降，很快趋近于 0。而在加入数据标准化后，cross entropy 曲线和验证结果都变化不大，但每轮迭代耗时却略微上升，得不偿失。在加入随机截取的数据后，cross entropy 曲线下降速度变慢并开始波动。所需的迭代次数大大增加，但验证集准确度也有了一定提升。而长宽分别截取 3/4 的验证集 loss 较截取 4/5 更为优秀，故保留长宽分别截取 3/4 的数据增强，并在此基础上进行下一步探索。



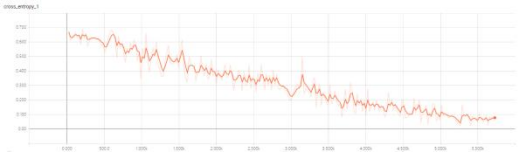
随机水平翻转



随机覆盖



随机亮度与对比度



随机饱和度与色差

图 3-3 更多随机化数据增强的 cross entropy 曲线

| 预处理方法 | 迭代次数 | 每轮迭代耗时 | 训练集准确度 | 验证集 loss | 验证集准确度 |
|---|---|---|---|---|---|
| 随机水平翻转 | 210 | 7.5s | 98% | 0.9467 | 76.09% |
| 随机覆盖 | 240 | 7.5s | 98% | 1.0733 | 75.89% |
| 随机亮度与对比度 | 210 | 7.9s | 93% | 0.9375 | 74.96% |
| 随机饱和度与色差 | 225 | 8s | 97% | 1.5522 | 67.86% |

表 3-2 更多随机化数据增强的训练和测试结果

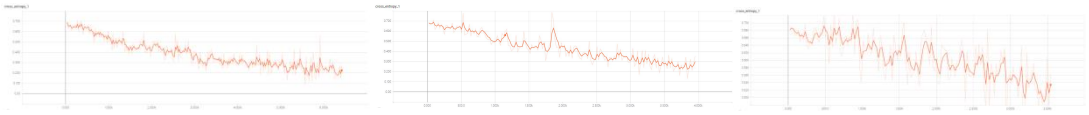
可以看到，随机亮度与对比度的 cross entropy 曲线下降速度显著变慢，说明此法大大增加了训练的难度，其它三条曲线则与此前区别不大。从验证结果来看，随机饱和度与色差的验证集准确度反而有明显下降，说明此法处理后的图片对训练产生了误导作用，有可能在其后加入图像标准化处理能解决这一点。其它三种方法的验证集准确度则区别不大，但随机水平翻转、随机亮度与对比度的验证集 loss 都有所下降。故最终在随机截取的基础上，再采用这两种方法进行数据增强。

## 3.2 从头训练

在研究了数据增强之后，再来对 Resnet 模型的各参数进行调整。需要考虑的有模型深度、优化器、学习速率等。

### 3.2.1 模型深度

Resnet 常见的模型深度有 18、34、50、101、152 五种，模型越深理论上能达到的准确度也更高，但所需数据量也越大、越容易过拟合。考虑到数据集比较小，本研究使用 Resnet18、Resnet34 和 Resnet50 这三种模型进行测试。



Resnet18　　Resnet34　　Resnet50

图 3-4 不同模型深度的 cross entropy 曲线

| 模型类别 | 迭代次数 | 每轮迭代耗时 | 训练集准确度 | 验证集 loss | 验证集准确度 |
|---|---|---|---|---|---|
| Resnet18 | 270 | 7.9s | 90% | 0.8191 | 78.20% |
| Resnet34 | 165 | 12.3s | 88% | 0.6173 | 78.15% |
| Resnet50 | 150 | 24s | 73% | 0.7121 | 62.44% |

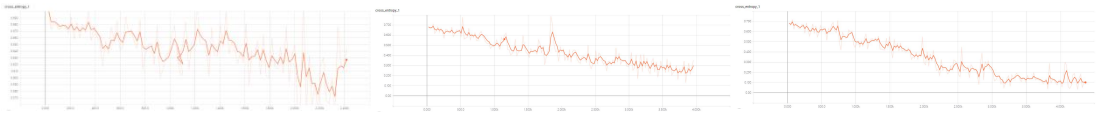
表 3-3 不同模型深度的训练与验证结果

从 cross entropy 曲线可见，深度越大，训练时的波动也越大。Resnet34 相比 Resnet18，训练集和验证集的准确度相近，但验证集 loss 有显著下降。虽然每轮迭代的用时增加了约 50%，但所需迭代次数大幅降低，总用时反而更少。而 Resnet50 的训练集和验证集准确率都非常低，即时通过延长训练时间、调节参数有望提升准确度，但训练耗时大大高于 Resnet18 和 Resnet34。故选择 Resnet34 作为进一步调参的基础。

### 3.2.2 优化器

优化器决定了 Resnet 网络参数更新的方式，也存在多种选择[17]，本研究考察了 GradientDescent[18]，Momentum[19]，Adam[20]这三种。GradientDescent 算法是

在批量梯度下降的基础上，每一次迭代后都计算 mini-batch 的梯度，然后对参数进行更新，是最常见的优化方法。Momentum 算法通过模拟物理中动量的概念，来优化相关方向的训练和弱化无关方向的振荡，本研究使用惯性系数为 0.9 的 Momentum 算法。Adam 算法是一种自适应优化器，它通过计算梯度的一阶和二阶矩估计，为不同的参数设计独立的自适应性学习率，收敛速度很快。



GradientDescent　　Momentum　　Adam

图 3-5 不同优化器的 cross entropy 曲线

| 优化器类别 | 迭代次数 | 每轮迭代耗时 | 训练集准确度 | 验证集 loss | 验证集准确度 |
|---|---|---|---|---|---|
| GradientDescent | 90 | 12.0s | 67% | 0.6323 | 63.86% |
| Momentum | 165 | 12.3s | 88% | 0.6173 | 78.15% |
| Adam | 180 | 12.2s | 95% | 0.3795 | 82.73% |

表 3-4 不同优化器的训练与验证结果

可见，三种优化器得到的结果差异明显。GradientDescent 因早期停止早早结束了训练，效果很不好，说明其训练耗时长，非常依赖对学习速率的调整。Adam 在训练集和验证集的表现都更优于 Momentum，迭代次数也没有太大增加。不同的学习速率对优化器的选择也存在影响，但本研究受篇幅所限不再过多讨论。由上可知 Adam 一定是个不坏的选择，故以 Adam 优化算法为基础继续调参。

### 3.2.3 学习速率

学习速率对训练速度、训练效果都至关重要，速率的设置方式可分为固定速率、随迭代次数下降、根据训练 loss 自动衰减以及周期性热重启[21]。本研究使用的随迭代次数下降方式还包含“预热”，即先使用一个较小的速率开始训练。根据训练 loss 自动衰减，是指先设一个较大的学习速率，随后每当训练 loss 不再下降时，学习速率则自动下降。周期性热重启，是在随迭代次数下降的基础上，学习速率还会周期性的恢复到初始值，以不会被卡在“鞍点”。本研究使用的周期

性热重启，下降时采用余弦衰减，并且由于每次热重启后 loss 都会上升一段时间，故把触发早期停止的迭代次数由 15 提升到 25。

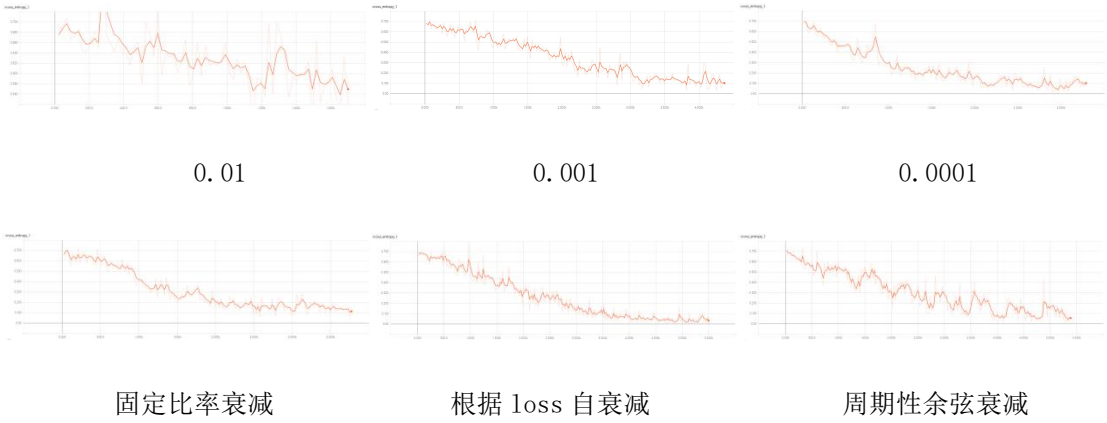


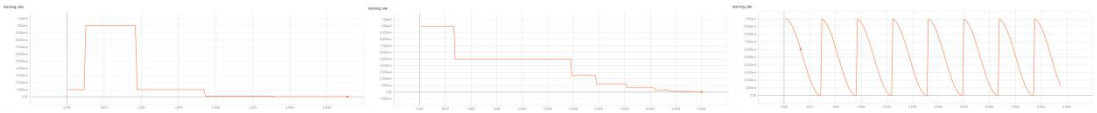

图 3-6 不同学习速率的 cross entropy 曲线

| 学习速率 | 迭代次数 | 每轮迭代耗时 | 训练集准确度 | 验证集 loss | 验证集准确度 |
|---|---|---|---|---|---|
| 0.01 | 60 | 12.2s | 67% | 0.7226 | 50.01% |
| 0.001 | 180 | 12.2s | 95% | 0.3795 | 82.73% |
| 0.0001 | 135 | 12.3s | 94% | 0.5082 | 83.16% |
| 固定比率衰减 | 135 | 12.3s | 94% | 0.4337 | 84.88% |
| 根据 loss 自衰减 | 210 | 12.3s | 98% | 0.4738 | 85.75% |
| 周期性余弦衰减 | 210 | 12.4s | 98% | 0.2923 | 87.62% |

表 3-5 不同学习速率的训练与验证结果

学习速率固定为 0.01 时，cross entropy 起伏很大，很早的就触发早期停止了，没有训练效果。固定为 0.0001 时，cross entropy 下降的比 0.001 更快，但其验证集 loss 较高。

固定比率衰减表现的比固定学习速率更为优秀，验证集准确度有明显提升，虽然验证 loss 比 0.001 略高，但迭代次数少很多，有较大提升空间。根据训练 loss 自衰减显著提升了训练集准确度，但验证集 loss 反而上升了，说明最后过小的学习速率加重了过拟合程度。而对于周期性余弦衰减，从学习速率变化曲线和 cross entropy 曲线都可以看到明显的周期性，其结果也是最优的，验证集 loss 最低、准确度最高。故最终选择周期性余弦衰减的学习速率。



固定比率衰减　　根据 loss 自衰减　　周期性余弦衰减

图 3-7 可变学习速率随迭代次数变化曲线

受篇幅所限，损失函数与权重初始化函数的具体调参过程不再赘述，最后选择的也是常见的两种做法：选用 cross entropy+l2 loss 作为损失函数，选用正态分布的 variance_scaling 作为权重初始化函数。

## 3.3 迁移学习

为了达到更高的准确度，本研究还使用了迁移学习方法对 Resnet 进行训练。预训练模型来源于 slim 框架对 ImageNet 的训练（http://download.tensorflow.org/models/resnet_v2_50_2017_04_14.tar.gz）。其在 ILSVRC-2012-CLS 测试集的 Top 1 精度为 75.6%，Top5 精度为 92.8%。ILSVRC-2012-CLS，即 2012 年 ImageNet 大规模视觉识别挑战赛的图像分类任务，拥有 1000 种类别。其中也包括猫和狗，但进一步细分成了多个子品种。

slim 中已提供了一些迁移学习的样例，本研究在其 flower 样例的基础上进行修改。但 slim 未提供模型的测试代码需自己实现，具体的环境搭建步骤可分为：

- 修改 download_and_convert_flowers.py 和 flowers.py 文件中的相应参数。
- 将训练集按类别分为两个文件夹，执行 download_and_convert_data.py 文件，将训练集转换成 TFRecord 文件；对验证集也做同样的处理。
- 在验证代码的基础上，编写测试代码。测试程序直接从文件夹中读入图片，不需要对测试集进行格式转换。

训练时，本研究尝试了五种模型微调方式：

- 去掉最后输出 1000 类的全连接层，加入新的输出 2 类的全连接层。

- 在加入的输出 2 类的全连接层之前再增加一个全连接隐层，以增加微调模型的复杂性。

- 保留原始的全连接层，对其参数进行训练，在其后加入输出 2 类的新全连接层。以在增加模型复杂性的同时，利用上预训练模型的类别权值。

- 保留原始的全连接层，不对其参数进行训练，在其后加入输出 2 类的新全连接层。从而直接依赖原本 1000 类中猫狗的各已分类品种进行训练。

- 去掉最后输出 1000 类的全连接层，加入新的输出 2 类的全连接层。对 Resnet50 的最后一个 block 也进行训练，以对高层特征的提取也进行微调。

训练时仍采用 1500 张图片的小数据集，使用 Resnet_v1_50 模型，采用 slim 中默认的数据预处理。batch size 设为 16，使用 Adam 优化器，学习速率为 0.001，weight_decay 为 0.00004。得到的结果为：

| 微调模型 | 验证集准确度 |
|---|---|
| 用新输出层替换旧输出层 | 96.57% |
| 用带隐层的新输出层替换旧输出层 | 97.39% |
| 加入新输出层，保留旧输出层并对其进行训练 | 98.11% |
| 加入新输出层，保留旧输出层而不对其进行训练 | 97.2% |
| 用新输出层替换旧输出层，并训练最后一个 block | 98.3% |

表 3-6 不同微调模型的验证结果

带隐层的输出层更优，说明了只对输出层进行训练的情况下，原特征需要被更复杂的组合起来。保留旧输出层的权值并进行训练，结果更优，说明旧输出层对于猫狗二分类问题仍有参考价值。而保留权值但不训练的结果一般，说明原预训练模型虽然包含了诸多猫狗的品种，但各种分类（1000 类）过多，输出精度有待提升。最后，用新输出层替换旧输出层，并对网络中最后一个 block 进行训练的方式得到了最优结果，说明了对高层特征提取的微调对迁移学习十分重要。本研究最终也将采用此方式对全训练集进行训练并测试结果。

# 第四章 评价与部署

## 4.1 模型的评价

### 4.1.1 Kaggle 得分

在从头训练中，根据对各类数据增强方法与参数选择的尝试，最终选择了{ **随机对图片长宽裁剪 3/4，随机水平翻转，随机亮度与对比度** }的系列数据预处理方法，选择了{ **Resnet_v2_34，Adam 优化器，周期性余弦衰减学习率** }的模型与参数。在 1500 张图片的小训练集中，最终得到了{ **验证集 loss 0.2923，验证集准确度 87.62%** }的结果。在此模型与参数的基础上，本研究对全训练集的进行了训练，然后在测试集进行测试并将结果上传至 kaggle。输出结果时使用限制范围的 trick，即将大于 0.99 的输出设为 0.99，小于 0.01 的输出设为 0.01，最终得到了 0.11477 的 loss 评分。


图 4-1 从头训练的最终 kaggle loss 得分

在迁移学习中，根据对各类模型微调方式的探索，最终选择了{ **用新输出层替换旧输出层，并训练最后一个 block** }的模型。在 1500 张图片的小训练集中，最终得到了{ **验证集精确度 98.3%** }的结果。在经过全训练集训练后，使用测试集进行测试，在 kaggle 上得到了 0.05504 的 loss 评分。


图 4-2 迁移学习的最终 kaggle loss 得分

### 4.1.2 与基准模型的比较

与基准模型的 kaggle 得分 0.11217 相比，从头训练的模型的得分 0.11477 要略低，未能优于基准模型，也未能达到进入 kaggle 排行榜 10%的目标。使用了更简单模型 Vgg16、更小数据集、初始参数的迁移学习基准模型，反而要比经过细

致调参的从头训练更好，充分说明了迁移学习的强大。所以，在数据集较小时，应毫不犹豫的先尝试使用迁移学习来解决问题。

而使用 Resnet50 的迁移学习最终的 kaggle 得分是 0.05504，要优于基准模型，并成功在 kaggle 上得到了小于 0.06127 的 loss。因而排名可以进入前 10%，达成了预期目标。

### 4.1.3 模型的鲁棒性

对于使用小数据集的从头训练，由于数据增强中采取了多种随机处理且数据集很小，因而训练结果也存在一定的随机性。另外，在对模型调参时，各参数之间原本也不是互相独立的，因此本研究的调参过程相当于做了简化处理，可能存在更优的参数搭配。

在第二章的数据分析小节中有提到，训练集与测试集抽样的猫狗图片大部分都能看到正脸，模型得以从眼睛、耳朵、鼻子等面部特征进行区别。可以猜想，对于看不到正脸的猫狗图像，识别的准确度将有所下降。此外，狗的品种繁多，训练集对所包含的猫狗品种并无说明，模型很可能对于部分猫狗品种无法识别。

## 4.2 在 Android 上的部署

App 界面采用最常见的底部导航栏结构，使用 BottomNavigationView 方式实现，拥有相册中选图片识别和摄像头实时识别两个页面。成功获取图片后，调用 TensorFlowImageClassifier 进行识别，根据识别结果在页面上展示猫或狗的头像。对于导入的 pb 模型文件，需要由变量名获知输入层和输出层的位置，因而在网络中要在合适的位置设置变量名“input”和“ouput”。

在摄像头实时识别时，对识别速度的要求很高，故最终模型选用最简单的 Resnet18。由于摄像头会不停移动，app 中还实现了摄像头的自动对焦功能。最终，实际的识别效果一般。识别的默认结果是狗，对于猫的照片，需要以较好的角度和图像比例才能正确识别出。这让我感受到了深度学习从研究到实际应用的差距，这个过程中仍有许多工程问题需要解决。

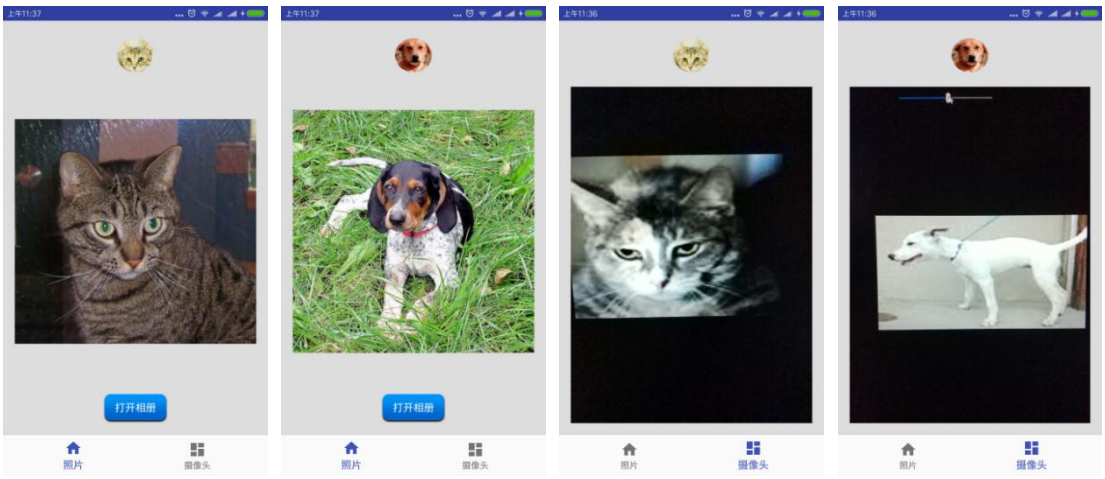



图 4-3 猫狗识别 Android App

## 4.3 在微信小程序上的部署

对于前端的微信小程序，因为后台服务器性能一般识别速度较慢，故设置了一次只能选取一张图片进行上传。对于后台服务器，使用 node.js 的 express 框架搭建服务器，通过 formidable 中间件来获取图片。将图片保存后，在 node.js 中通过 child_process 调用 python 程序进行识别。在阿里云上申请免费的 symantec ssl 证书搭建 https，服务器使用 https 默认的 443 端口号。python 识别程序使用 loss 最低的 Resnet50 迁移学习模型，因而识别效果要优于 Android App。

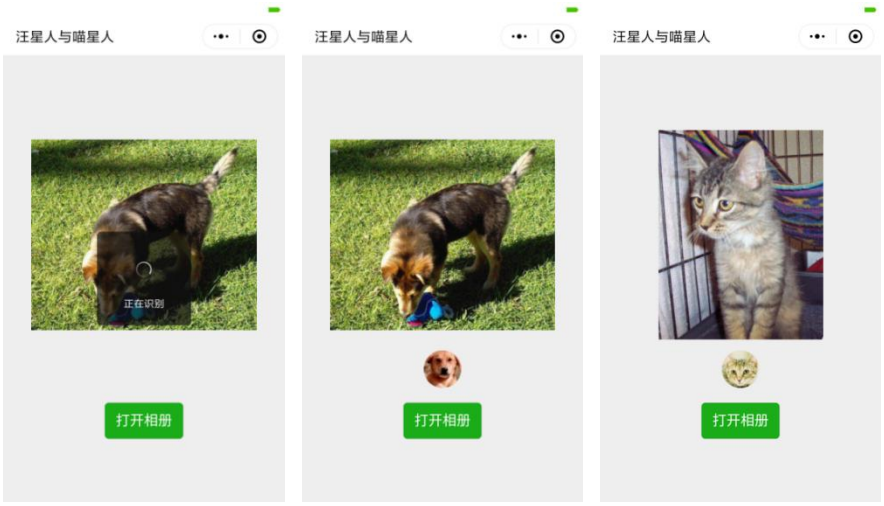



图 4-4 猫狗识别 微信小程序

# 第五章 项目结论

## 5.1 结果可视化

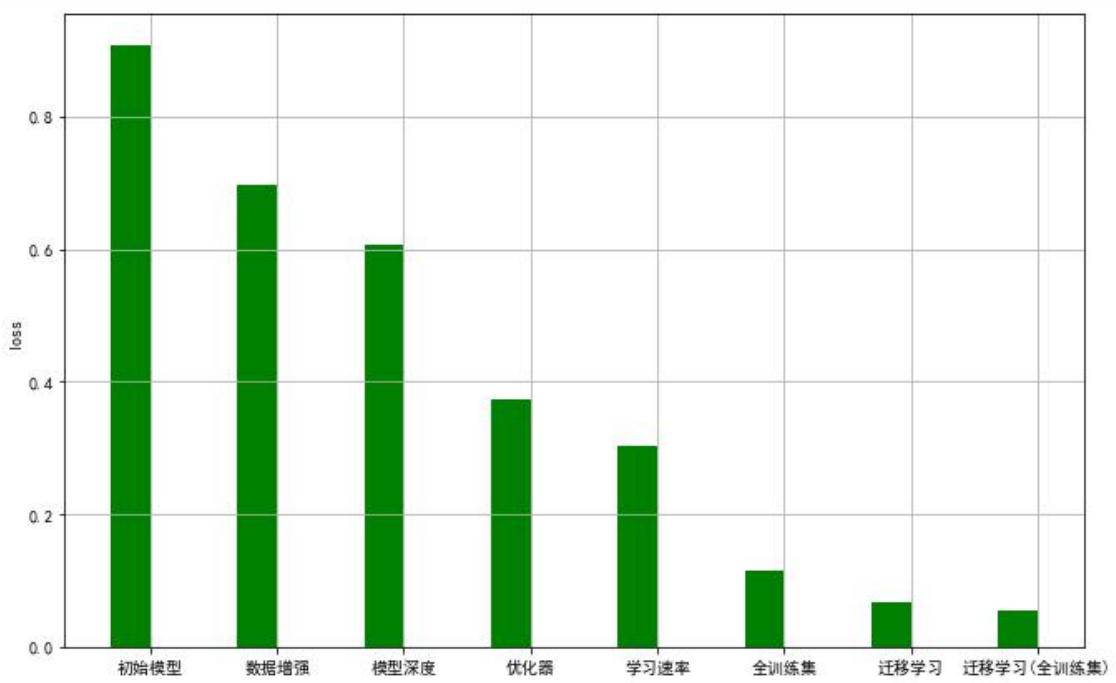



图 5-1 不同模型在 kaggle 上的 loss 得分

图中的前 6 个模型使用的是从头训练，从初始参数开始，递次进行了数据增强、模型深度、优化器和学习速率的调参改进，并最终由小训练集改为全训练集。最后 2 个模型使用的是迁移学习，分别为小训练集上的迁移学习和全训练集上的迁移学习。

可见，随着模型的改进并最终应用全训练集和迁移学习，loss 也不断下降，大约从 0.9 一路下降到了 0.05。其中降幅最大的改进出现在从头训练应用全训练集时，这说明了对于深度学习，数据集仍是训练效果的决定性因素。而使用小数据集的迁移学习比使用全训练集的从头训练更优，也是因为迁移学习使用的预训练模型已经经历过更大规模的数据集训练。最后，使用全训练集的迁移学习相比使用小训练集的迁移学习 loss 下降不大，进一步说明了迁移学习相比从头训练，不那么依赖数据集的大小。要改进迁移学习的效果，可以多从微调方式去着手。

## 5.2 对项目的思考

在完成毕业项目的过程中，几个月的时间里我经历了如下过程：首先是项目选择，在猫狗大战、走神司机、文档归类等项目中，由于我刚使用 Resnet 尝试参加了 AI Challenger 的场景分类比赛，对 Resnet 和图像分类略有了解。并且我此前是 Android 开发工程师，因而决定继续使用 Resnet 来完成猫狗大战，并在此基础上开发出移动应用。随后是技术调研，我成功验证了使用 Tensorflow 在 Android 端和微信小程序上进行图像分类的可行性。在第一次提交开题报告后，评审师提醒了我可以使用迁移学习，于是我又使用迁移学习作为基准模型和从头训练的进一步改进。在从头训练的调参过程中，我搭建了云 GPU，并依据各类资料进行调参尝试。在对模型有了进一步了解后，我又开始进行系统性的调参，然而发现结果仍不能达到要求。随后使用迁移学习进行微调，终于成功的进入了 kaggle 排行榜前 10%。最后基于训练得到的模型构建了简单的 Android app 和微信小程序，并总结实验数据编写论文。

整个项目中，我遇到的主要难点集中在云 GPU 主机环境搭建与改进、代码编写和调参方案的制定：

- 云 GPU 主机我曾先后选择 google 和美团。google 云有免费额度，但我这网络连接不太稳定经常掉线。美团很便宜，性能也足够，但关机后仍会计费，需要不停的删除和重新创建主机。

- 编写从头训练的代码时，Resnet 网络不用做什么改动，但数据增强、训练、验证和测试的代码需要进行编写。在这个过程中，对 Tensorflow 中 tensor 的使用和各种机制需要不断加深理解。此外，迁移学习也需要了解 slim 样例的代码实现，以及编写测试代码。

- 虽然从头训练的结果最后还是没能满足要求，但我在尝试各种模型与参数的过程中收获很大。制定调参方案的过程，需要阅读很多相关资料，大大加深我对卷积神经网络的结构与训练的理解。而执行调参方案的过程，让我积累了训练神经网络的一手经验，开始更主动的去思考如何去更快的训练、如何编写更优秀的代码。

猫狗大战在通用场景下实际应用意义不大，因为它少了一个关键性的分类：非猫非狗。而且迁移学习使用的预训练模型本就包含了诸多猫狗品种，迁移学习的过程大致上只是把这些分类结果合并了一下。但作为卷积神经网络的学习和测试项目，猫狗大战还是相当不错的。

## 5.3 可以作出的改进

本研究在从头训练和迁移学习中使用的都是 Resnet 网络，而在 Resnet 网络之后已经有一些更先进的网络被提出，比如：Inception V4、Inception-ResNet-v2、到最新的 PNASNet 等。

在从头训练的调参过程中，对各参数的调节相对独立。实际上优化器与学习速率的选择有非常紧密的联系，根据相关资料，通过选择合适的学习速率，SGD 有可能会得到比 Adam 等自优化器更好的效果[22]。本研究最后选择的周期性余弦衰减速率，其中也有一些参数可以再仔细调节。而本研究没有过多讨论的 loss 函数，也还一些文章可做。当然，把这些都做到完美恐怕也很难进入 kaggle 排行榜前 10%。要想使用从头训练在两万多张图片的小规模数据集上取得上佳效果，还需做出更底层的改进。

在迁移学习中，本研究只对模型的修改进行了实验，对数据增强和各训练参数的调整还可再加考虑。并且相信很多其它同学都使用了 Resnet50、Inception V3、Xception 三模型融合的迁移学习，这相当于把迁移学习和“集成学习”相结合，可以达到更高的准确度。

# 参考文献

[1] Samuel AL. Some studies in machine learning using the game of checkers. IBM Journal of research and development. 1959 Jul;3(3):210-29.

[2] McCulloch WS, Pitts W. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics. 1943 Dec 1;5(4):115-33.

[3] Hebb DO. The organization of behavior; a neuropsycholocigal theory. A Wiley Book in Clinical Psychology.. 1949:62-78.

[4] Rosenblatt F. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review. 1958 Nov;65(6):386.

[5] Minski ML, Papert SA. Perceptrons: an introduction to computational geometry. MA: MIT Press, Cambridge. 1969.

[6] Werbos P. Beyond regression: new fools for prediction and analysis in the behavioral sciences. PhD thesis, Harvard University. 1974.

[7] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. nature. 1986 Oct;323(6088):533.

[8] LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, Jackel LD. Backpropagation applied to handwritten zip code recognition. Neural computation. 1989 Dec;1(4):541-51.

[9] LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE. 1998 Nov;86(11):2278-324.

[10] Hinton GE, Osindero S, Teh YW. A fast learning algorithm for deep belief nets. Neural computation. 2006 Jul;18(7):1527-54.

[11] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. InAdvances in neural information processing systems 2012 (pp. 1097-1105).

[12] Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M, Kudlur M. TensorFlow: A System for Large-Scale Machine Learning. InOSDI 2016 Nov 2 (Vol. 16, pp. 265-283).

[13] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. InProceedings of the IEEE conference on computer vision and pattern recognition 2016 (pp. 770-778).

[14] Deng J, Dong W, Socher R, et al. Imagenet: A large-scale hierarchical image database[C]//Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009: 248-255.

[15] Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, Hubert T, Baker L, Lai M, Bolton A, Chen Y. Mastering the game of go without human knowledge. Nature. 2017 Oct;550(7676):354.

[16] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.

[17] Ruder S. An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747. 2016 Sep 15.

[18] Robbins H, Monro S. A stochastic approximation method. The annals of mathematical statistics. 1951 Sep 1:400-7.

[19] Qian N. On the momentum term in gradient descent learning algorithms. Neural networks. 1999 Jan 1;12(1):145-51.

[20] Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980. 2014 Dec 22.

[21] Loshchilov I, Hutter F. SGDR: stochastic gradient descent with restarts. arXiv preprint arXiv:1608.03983. 2016 Aug 13.

[22] Wilson AC, Roelofs R, Stern M, Srebro N, Recht B. The marginal value of adaptive gradient methods in machine learning. InAdvances in Neural Information Processing Systems 2017 (pp. 4151-4161).