猫狗大战

I. 问题的定义

项目概述

此项目来源于 kaggle 比赛项目:Distinguish images of dogs from cats,项目属于计算机视觉领域一个传统的图像分类项目，猫狗大战，即通过深度学习区分给出的图片是猫还是狗;

数据集可以从 kaggle 上下载:
https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/data

图像分类过程非常明确，给定已经标注的数据集，提取特征，训练得到分类器。在图像分类领域，比较著名的数据集有 MNIST,ImageNet,MS COCO 等等

问题陈述

此项目是一个典型的监督学习分类问题，有两个类别(猫或者狗)，需要达到的目的是使用深度学习方法识别一张图像是猫还是狗:

输入：一张彩色图片

输出：是猫还是狗

需要通过使用 CNN(卷积神经网络)构建深度学习模型,CNN 非常适合检测图像中的特征，例如颜色、纹理和边，然后使用这些特征识别图像中的物体。首先在训练集上训练模型，然后通过验证集验证模型准确率，最后使用测试集进行预测

评价指标

此项目 kaggle 给定的评判计算方法，对数损失函数:

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

n 是测试集图像数量

yˆi 是图像预测为狗的概率

Yi 如果图像是狗，则值为 1，如果是猫，则为 0

log()是以 e 为底的对数

对数损失值越小，预测结果越优。

项目的预定目标是 kaggle Public Leaderboard 前 10%，即将对数损失控制到 0.06 以内

II. 分析

数据的探索

数据集通过在 kaggle 网站下载，下载完成之后，解压包含 train 和 test 两个文件夹;其中 train 文件夹中包含 25000 张带标签的图片，猫狗图片各 12500 张，每张图片的命名都为 图片类别+id+图片类型，例如:cat.0.jpg (图片 id 从 0 到 12499);

test 文件夹中包含 12500 张图片，图片命名为图片 id+图片类型,例如 1.jpg (图片 id 从 1 到 12500)

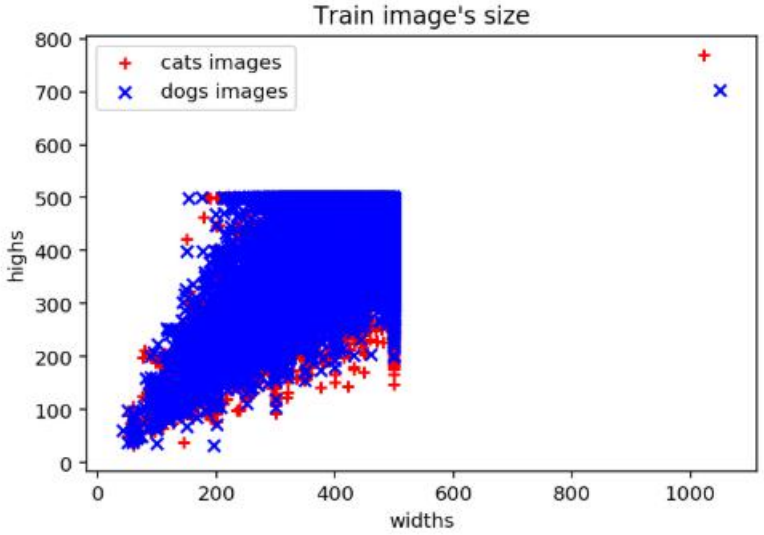
探索性可视化

从数据集中随机抽样 10 张图片数据进行查看:

可以看出图片都是 RGB 图像

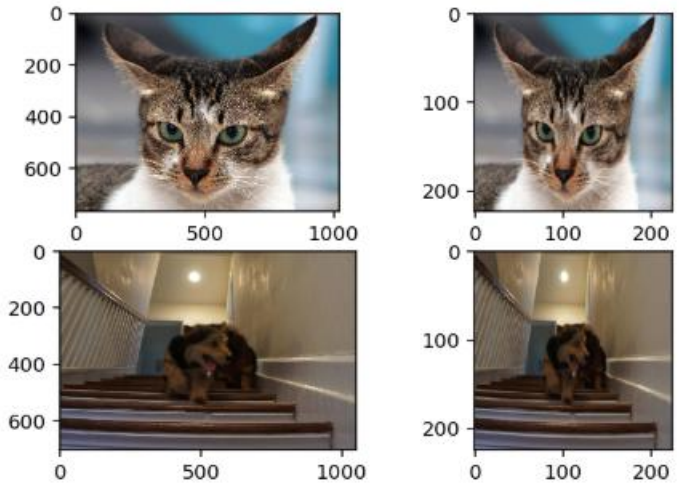
探索图像尺寸分布：

将 train 文件夹下猫狗图像的宽高数据提取出来画散点图，查看图像尺寸是否存在异常值



由上面的散点图可以看出，大部分图片大小像素值都在(500, 500)内，猫狗各有一张图片尺寸明显异常，找出这两张图片:

尺寸异常的猫图像:cat.835.jpg,尺寸为:[1023, 768]
尺寸异常的狗图像:dog.2317.jpg,尺寸为:[1050, 702]



两张尺寸异常的图像经过 resize 缩放到 224*224 大小之后,仍能很容易的辨认出猫或者狗,这说明图像缩放并不会影响到模型的分类,所以尺寸异常这种异常值无需处理

类别标签标注异常探索

由于本项目是猫狗分类,可以看成是 ImageNet 的数据集变种(ImageNet 数据集分类中包含猫和狗),故可以使用在 ImageNet 上预训练过的模型(比如 Vgg16)(ImageNet 上预训练的模型分类器 outsize=1000)来对数据集进行预处理,来检测是否有图像被错误标记。大概检测思路为:使用预训练模型训练数据集,获得预测分类结果最高的 N 个类别,查看猫和狗是否在这些类别当中,若不在则可以认定图片标记错误

①首先找出 ImageNet 数据集的 label.txt 文件记忆里面对应的猫狗分类的 label,然后使用预训练模型对数据集进行预测,先用数据集前 1000 张图片作为样本,探索合适的 N 值,然后在将对应的 N 值应用于整个数据集,需要注意的是,由于 ImageNet 数据集分类猫的类别有 7 中,而狗的类别达到 118 种,所以在查询合适的 N 值时需要区别猫狗分别查找:

②VGG16 预训练模型进行标签异常检测;经过多次试验,确定 N 值猫为 45,而狗为 10,应用于整个数据集筛选出来异常标注的 1 图片猫有 137 张,而狗有 69 张

Densenet121 预训练模型进行标签异常检测;经过多次试验,确定 N 值猫为 180,而狗为 85,应用于整个数据集筛选出来异常标注的 1 图片猫有 92 张,而狗有 73 张

Resnet50 预训练模型进行标签异常检测;经过多次试验,确定 N 值猫为 130,而狗为 45,应用于整个数据集筛选出来异常标注的 1 图片猫有 54 张,而狗有 65 张

③然后将各模型筛选出来的异常标注图片合并起来,重新使用各模型再筛选一次

采用各模型上一步选出的 N 值,VGG16 第二次筛选出的异常图像有猫 90 张,狗 55 张;Densenet121 第二次筛选出的异常图像有猫 92 张,狗 73 张;Resnet50 第二次筛选出的异常图像有猫 54 张,狗 65 张;

将各模型第二次筛选出的异常图片合起来共猫 221 张,狗 186 张

④显示出筛选出来的异常标注图像,然后人工选择

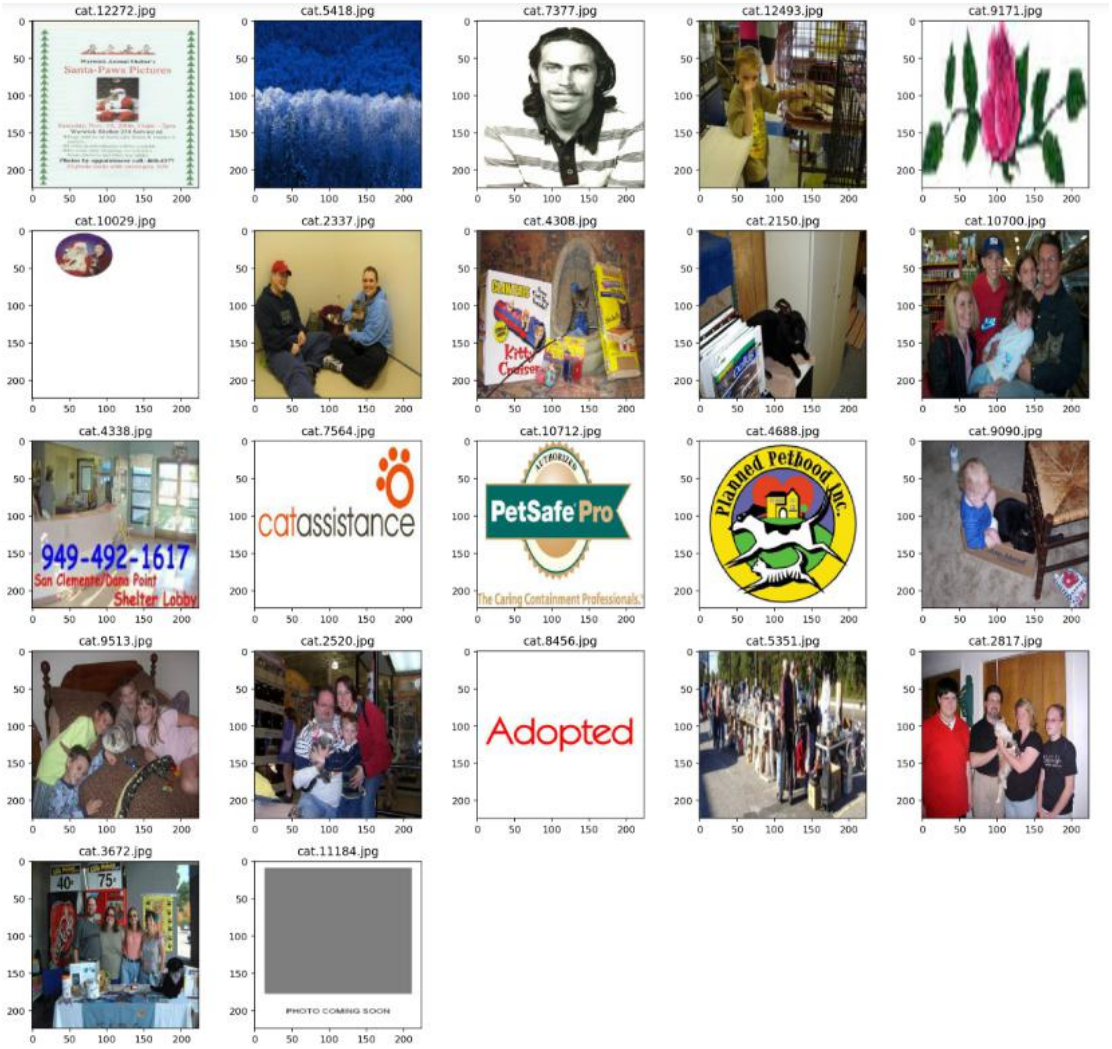
a.针对看起来正常且标记正确,模型却筛选出来的图片,不予处理(毕竟本数据集并不完全是 ImageNet 的子集,故有误差可以理解)

b.针对标记反的图片，例如猫标记成狗或者狗标记成猫，修改标注后追加在对应的图片序列后，例如"cat.12500.jpg，cat.12501.jpg"等

c.针对明显与猫狗无关或者背景过于复杂，猫狗在图像中显示不明显或占比重过小的情况，删除此图像

最终筛选出的需要删除的猫图像为

[12272,5418,7377,12493,9171,10029,2337,4308,2150,10700,4338,7564,10712,4688,9090,9513,2520,8456,5351,2817,3672,11184]，具体情况如下所示



需要删除的狗图像为

[1773,9517,6405,7413,12376,2614,10237,8736,10801,6725,10747,6475,11299,3889,9931,1043,4367,5604]：

图片标注反的图像为(猫)[11222, 3822, 7920, 4085, 5583, 11724, 9444, 335]，狗图像数据集没有发现标记反的

经过数据预处理，标记为猫的图片减少了 30 张，标记为狗的图片减少了 10 张(删除 18 张，错误标记转换过来 8 张)，对于各自 12500 的数据集来说可以忽略不计

从展示结果图片也可以看出模型很好的筛选出了异常标注的图片，最终人工处理的图片只有几百张，在可接受范围内；这些异常标注的图片对模型的猫狗分类没有积极的作用，只会使模型训练时间更长，降低模型的性能，故很有必要从数据集里处理掉；以上分析过程表明使用预训练模型进行异常图片筛选是一种有效的方法，否则人工筛选 25000 张图片将是一件浩大工程

算法和技术

1.卷积神经网络 CNN

卷积神经网络在图像分类，识别领域具有很大的优势；CNN 学习识别基本的直线，曲线，然后是形状，点块，最后是图片中更复杂的物体，最终 CNN 分类器把这些大的，复杂的物体综合起来识别图片。CNN 通过正向和反向传播，自己学习识别物体，而不需要设定特定的特征，CNN 可能有多层网络，每层捕获对象抽象层次中的不同级别。

为什么 CNN 要强于传统的计算机视觉分类方法呢？我们知道图像分类的传统流程包括两个部分，即特征提取和分类，特征提取指的是从原始图像中提取出更具体和高级的特征，这些特征携带者具体信息并能够用于区分各个类。这种特征提取的方式是无监督模式，因为从图像的像素点中提取信息时并没有使用类别标签，例如方向梯度直方图(HOG)等特征提取算法。

完成特征提取后，再使用这些特征与类别标签训练一个分类模型，常用的模型有随机森林，SVM 和 LR 等。传统的图像分类流程存在一个很明显的缺点，那就是特征提取的类型必须在模型训练前确定，而不能根据图像和分类标签进行调整。如果这些提前选择的特征不足以区分各个类别，即缺乏代表性时，训练获得的模型的准确性是不理想的。传统图像分类流程的一个较好方法是使用多种特征提取器，然后组合这些特征以得到一种更好的特征，但这本身是比较困难和复杂的。

而 CNN 就完全不同了，它并没有建立使用硬编码的特征提取器，而是将特征提取和分类两个模块融合在一起。CNN 通过自动识别图像的特征来进行特征提取，并基于分类标签进行分类，

2.迁移学习

迁移学习通俗的讲就是应用已有的知识来学习新知识，在深度学习领域体现在把预训练的模型搬迁至新模型中，以加快新模型的训练。因此应用迁移学习时模型便不用从零开始学习。迁移学习要求预训练模型和新模型之间存在相似性。本项目将应用迁移学习技术，使用预训练的 CNN 特征，具体就是使用在 ImageNet 上大型卷积神经网络预训练的权重。因为 ImageNet 中的标签包含本项目中的分类标签——猫和狗，因此具有相关性，可以进行迁移学习。

随着计算机视觉的发展，基于 ImageNet 数据集已经诞生很多优秀的模型(比如 VggNet,Resnet,DenseNet,Inception v3 等)，可以直接运用迁移学习技术调整其分类器输出类型为 2 以满足本项目需要，然后再在预训练模型的基础上进行参数微调，达到猫狗图片分类的目的

3.早期停止技术

在模型训练过程中，实时监控验证集损失 val loss，当模型的 val loss 降低时保存模型的 checkpoint_pth，当模型的 val loss 不再降低时，记录下 epoch 次数，若 val loss 连续超过 n_epochs_stop(事先定义的早停次数，一般为 10 或者 20)次数都没有降低，则模型停止;避免了设置 epoch 过大导致模型训练过头，错过了 loss 最低点

4.动态学习率

设置模型学习率随着 epoch 的增加而变小，使模型训练过程中逐步减小梯度变化的幅度，从而降低模型过拟合的程度

5. 深度学习框架 Pytorch 和 GPU 计算

PyTorch 于 2017 年初推出，对深度学习领域带来了深远的影响。它是由 Facebook 的 AI 团队开发的，完全开源，已经被各行各业和各个学术领域采用。它比 TensorFlow 和 Keras 更新，它与 Python 本身更相关，就像 Python 编程一样。PyTorch 本质上是 Numpy 的替代者，而且支持 GPU、带有高级功能，可以用来搭建和训练深度神经网络。

GPU 加速运算是指同时利用图形处理器(GPU)和 CPU，以加快计算机运算速度。GPU 加速计算的原理是将应用程序计算密集部分的工作负载转移到 GPU 中运行，同时仍由 CPU 运行其余程序代码

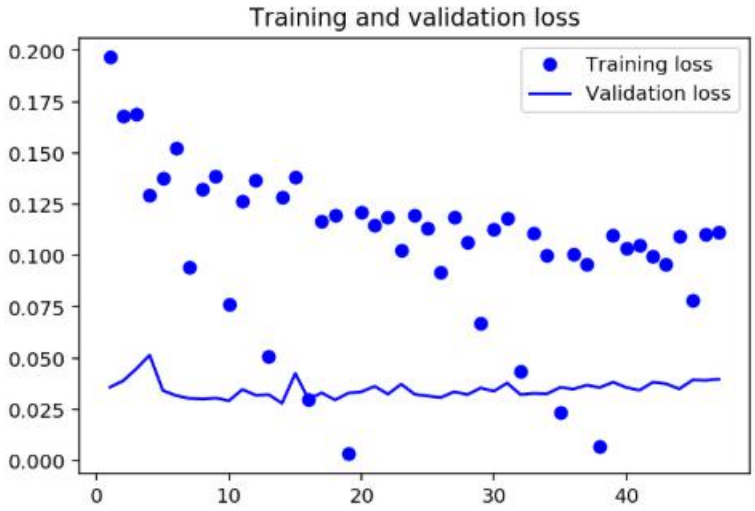
6. 多模型特征向量融合技术

提取多个模型的特征向量，然后在长度上进行堆叠，拼接成一个特征向量，构建一个分类器训练特征向量，进行模型训练，预测，多模型特征融合可以提高模型准确性，降低 loss，以及提升训练速度等

基准模型

考虑到迁移模型的优越性，所以基准模型直接选择 ImageNet 数据集上的预训练模型 Vgg16，epoch 设为 15，droupout 设为 0.5，使用动态学习率，损失函数采用 NLLLoss()

保留 VGG16 的深度卷积层，修改 classifier OutputSize 设为 2，并增加 LogSoftmax 层构建新的模型，进行训练，训练过程中的训练集和验证集的 LogLoss 分布情况如图:



15 个 epoch 训练时间为 82 分 14 秒,模型参数保存时 train loss 为 0.129,val loss 为 0.028;在测试集上的表现为 LogLoss 0.026,准确率为 0.990

将训练好的 Vgg16 模型应用于 kaggle 提供的测试集，将生成的结果提交到 kaggle，得分为 0.05783，满足项目预期

III. 方法

数据预处理

1.数据集分类

首先，从 kaggle 下载下来的数据，解压之后包含 train 和 test 两个文件夹;

其中 train 文件夹中包含 25000 张带标签的图片，猫狗图片各 12500 张;test 文件夹中包含 12500 张图片

首先处理 train 文件夹，在根目录 dogs_vs_cats 下新建 Datasets 文件夹，将 train 中的所有图片移动到 Datasets 下，然后在根目录下新建 val 和 datatest 文件夹，分别在 train、val 和 datatest 文件夹下新建 cat 和 dog 文件夹

然后将 Datasets 中的猫狗图片各随机抽取 10%样本放入 datatest 下的对应文件夹作为测试集，将剩余的 90%张样本中的 10%放入 val 下的对应文件夹作为验证集，将剩余的图片放入 train 下的对应文件夹作为训练集

2.图片处理

在数据探索阶段，已看到图片分辨率不同，故需要进行处理，这里采取一定的数据增强手段，对于训练集，你需要变换数据，例如随机缩放，剪裁，翻转，这样有助于网络泛化，并带来更好的效果。还需要确保输入的数据大小调整为 224*224 像素

验证集和测试集用于衡量模型对尚未见过的数据的预测效果，故不需要进行缩放或旋转变换，但需要将图像剪裁到合适的大小(224*224 像素)

这里需要说明一下的是，不同的卷积神经网络模型对数据预处理的要求是有差别的，例如，模型对图片的尺寸要求，VGGNet 和 ResNet 的默认输入图像分辨率是 224*224，而 Xception 和 Inception v3 则默认为 299*299，故需要根据所选模型需要在数据预处理时做对应的图像剪裁

对于所有三个数据集，都需要将均值和标准差标准化到网络期望的结果。均值为 `[0.485, 0.456, 0.406]`，标准差为 `[0.229, 0.224, 0.225]`。这样使得每个颜色通道的值位于 -1 到 1 之间，而不是 0 到 1 之间。另外也需要将输入数据都转化为张量 Tensor 模式，用于使用 PyTorch 框架

需要注意的是对 kaggle 测试集进行的预处理，自定义 TestDataset(继承 Dataset)类，传入 sample_submission.csv 和测试集文件夹 test/，由于测试集中的图片名称为从 1.jpg 到 12500.jpg 顺序排列，对应了 sample_submission.csv 文件的 id，故 TestDataset 的 __getitem__返回图片 image 和对应的 imageid(即图片名称);由于 Dataset 获取数据集图片并不是顺序提取，故需要根据 imageid 获取图片，方便后续存储结果

用 Dataloader 加载 test_datasets，需要注意的是 batch_size 需要是默认的 1，因为需要一张张预测并记录结果，

对测试集进行预测过程中，根据每张图片的测试结果 classes 以及 probs，获取 kaggle 比赛需要的每张图片为 dog 的概率，即 classes=1 时对应的 probs，存储在 dataframe 中，并根据 id 排序，最后将数据保存为 submission.csv 文件存储在本地用于 kaggle 提交

执行过程

1.构建了应用于 ImageNet 数据集的预训练模型 Densenet121，由于 ImageNet 数据集类别有 1000 种，所以应用于此数据集的模型输出都是 1000 类，故需要针对 Densenet121 模型的分

类器 classifier 进行修改，将 outsize 改为 2，并添加 LogSoftmax 作为激活函数用于求取分类概率的对数

需要注意的是，需要声明 param.requires_grad = False 来保留 Densenet121 模型的特征提取层(即卷积网络层)用作迁移学习，

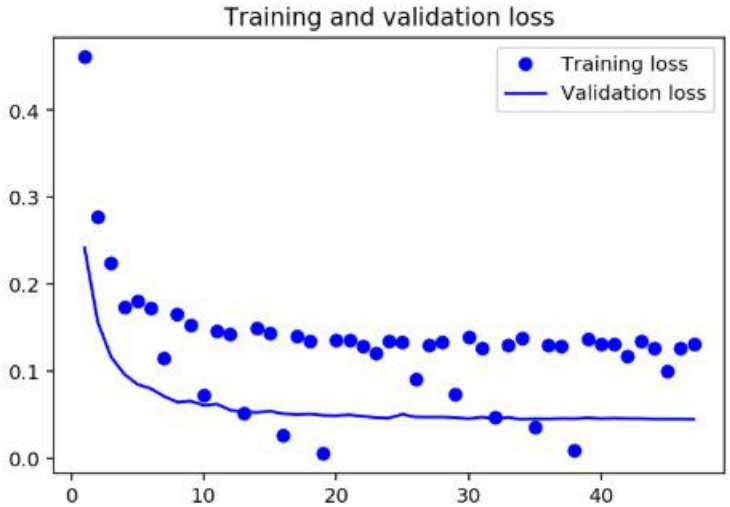
采用 nn.NLLLoss()作为损失函数，(LogSoftmax + NLLLoss 即为 Log Loss);采用 optim.Adam 作为优化器(选 Adam 作为优化器是因为它能够自适应学习率)，优化 model_densenet121.classifier.parameters()，并且设定动态学习率

模型构建完成之后，开始进行深度学习训练，首先确认 GPU 是否可用 (torch.cuda.is_available())，将 model 以及输入的 image 和 label 转移到 GPU 上进行计算，加快计算速度

```
#模型训练时执行，是 dropout 生效
model.train()
#使优化器记录模型梯度变化
optimizer.zero_grad()
#前向传递,输出模型预测值
outputs = model.forward(images)
#使用损失函数计算预测值与真实 label 之间的误差
loss = criterion(outputs,labels)
#反向传播
loss.backward()
#更新权重和偏差，模型优化
optimizer.step()
```

每执行 100(print_every)个 batch，对模型在验证集上的表现进行测试，与在训练集上不同的是，不需要进行梯度计算，只是预测图片，故需要 torch.no_grad()，且设置 model.eval()使 dropout 失效，用以预测，避免信息丢失;

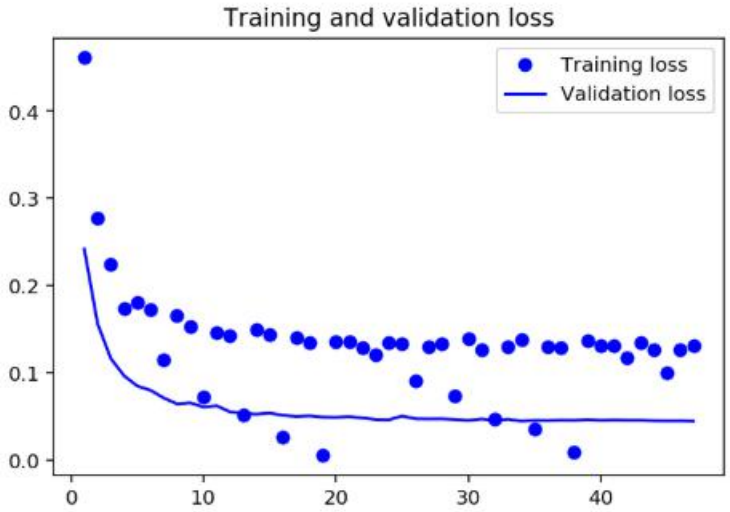
计算模型在训练集上的损失,在验证集上的损失和准确率,实时打印用以评估模型表现，并根据验证集上的损失程度，采用早期停止技术保存训练过程中模型表现最好时对应的 state_dict



Densenet121 模型在 15 个 epoch 训练时间为 58 分 46 秒,模型参数保存时 train loss 为 0.145,val loss 为 0.046;在测试集上的表现为 LogLoss 0.046 准确率为 0.987

将训练好的 Densenet121 模型应用于 kaggle 提供的测试集，将生成的结果提交到 kaggle，得分为 0.06845

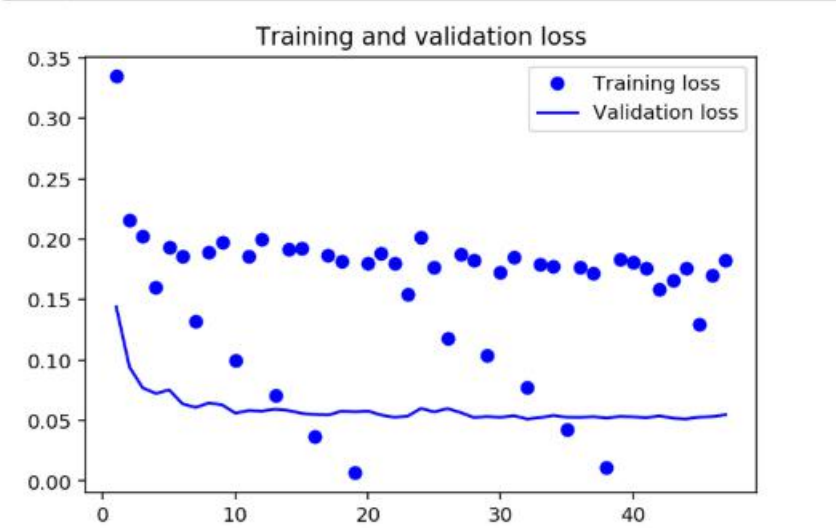
2. 构建以预训练模型 Resnet50 为基础的模型，修改 resnet50.fc 的 OutputSize 为 2，并添加 LogSoftmax 层，进行训练



Resnet50 模型在 15 个 epoch 训练时间为 57 分 21 秒,模型参数保存时 train loss 为 0.138,val loss 为 0.045;在测试集上的表现为 LogLoss 0.042 准确率为 0.987

将训练好的 Resnet50 模型应用于 kaggle 提供的测试集，将生成的结果提交到 kaggle，得分为 0.06008

3. 构建以预训练模型 Inception v3 为基础的模型，需要注意的是，inception_v3 模型输入图片大小默认为 299*299，所以需要单独处理数据集，另外需要将 inception_v3.aux_logits 设为 False，负责无法添加 LogSoftmax 层，修改 inception_v3.fc 的 OutputSize 为 2，并添加 LogSoftmax 层，进行训练



inception_v3 模型在 15 个 epoch 训练时间为 80 分 37 秒，模型参数保存时 train loss 为 0.077,val loss 为 0.051;在测试集上的表现为 LogLoss 0.050 准确率为 0.985

将训练好的 Densenet121 模型应用于 kaggle 提供的测试集，将生成的结果提交到 kaggle，得分为 0.07339

完善

运用多模型特征向量融合技术，步骤如下:

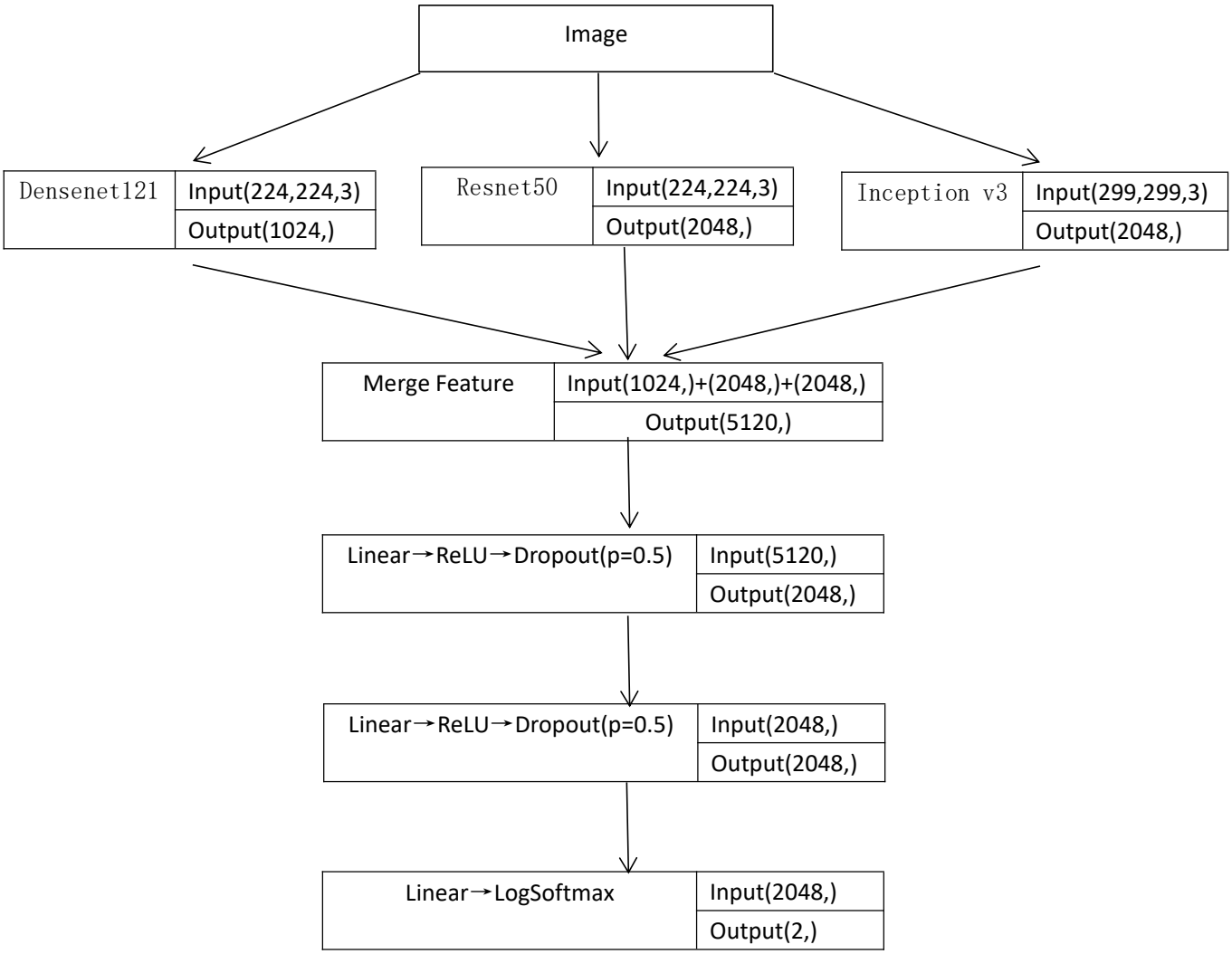
①分别加载训练过的模型(Densenet121,Resnet50 以及 Inception v3),然后修改模型的 classifiler 或者 fc 为 ReLU()以获取模型最终输出的特征向量

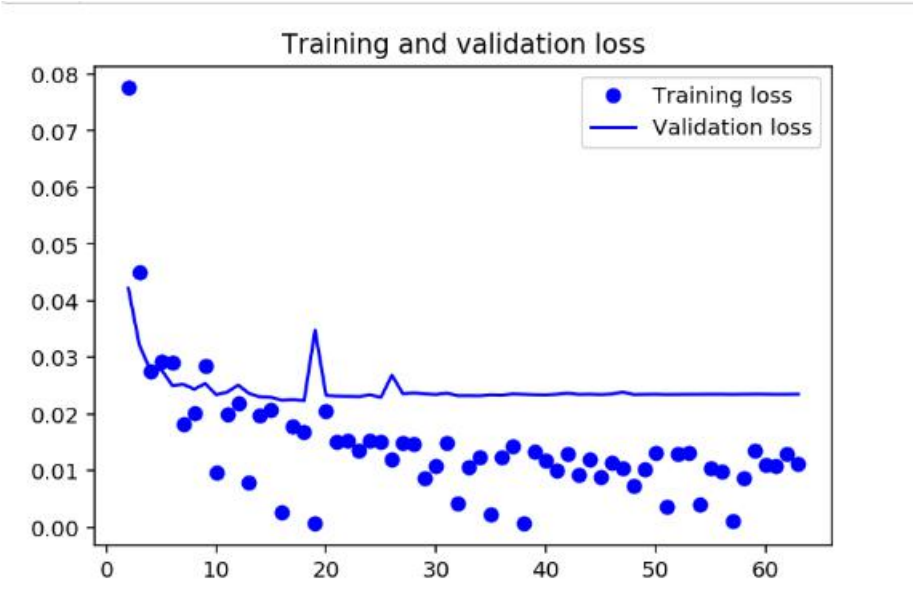
②使用重新构建的模型进行图片数据集预测，把对应的特征向量和 label 保存，对于 kaggle 提供的测试集，label 为对应的图片 id

③进行特征融合，Densenet121 模型输出的特征向量为(1024,)，Resnet50 以及 Inception v3 模型输出的都是(2048,)故最终拼接成的特征向量为(5120,)

④构建简单的分类器模型，将合成的特征向量输入分类器，进行模型训练

应用特征融合技术构建的分类器模型如下图所示





IV. 结果

模型的评价与验证

模型	TrainTime	Train Loss	Val Loss	Test Loss	Kaggle Score
Vgg16	70’2”	0.129	0.028	0.026	0.05783
Densenet121	55’4”	0.145	0.046	0.046	0.06845
Resnet50	54’43”	0.138	0.045	0.042	0.06008
Inception v3	80’37”	0.077	0.051	0.050	0.07339
MergeModel		0.017	0.022	0.023	0.04775

MergeModel 主要耗时在于提取特征向量上，模型训练并未占用多长时间，故在此不做统计

由上模型性能对比表可知，基础模型 Vgg16 kaggle 得分为 0.05783

多个单模型测试结果组合得分:

将训练好的 Densenet121、Inceptionv3、Resnet50 以及 VGG16 模型生成的结果组合求平均，提交到 kaggle 得分为 0.05826

将训练好的 Densenet121、Inceptionv3、Resnet50 以及 VGG16 模型生成的结果按比例组合(Resnet50*0.55 + Densenet121*0.2 + Inception_v3*0.15 + Vgg16*0.1)，提交到 kaggle 得分为 0.05915

特征向量融合模型 kaggle 得分为 0.04775, 比所有单模型表现都好

以上多种方案得分都达到了项目预期

合理性分析

单模型的参数微调，应用的是迁移学习技术，因为预训练模型已经在大数据集(ImageNet)上进行了训练，权重基本上已经是最优的;这对比从零设计的模型具有巨大的优势，节省了大量的训练资源，提高了模型性能。

由 kaggle 得分可以看出，多模型特征向量融合得分最优。多模型融合技术有多种，特征向量拼接融合是其中一种非常简单便捷的方法，其性能也明显优于单模型。

V. 项目结论

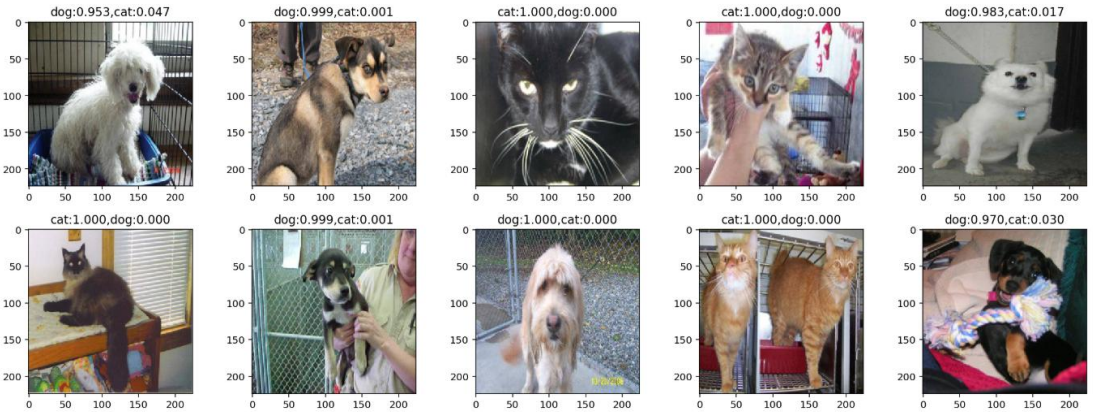
结果可视化

从 kaggle 得分来看，多模型特征向量融合技术比单模型拥有较大的优势，模型更准确，所以确认融合模型作为最终模型

从测试集中随机抽取 10 张图片进行预测及显示:

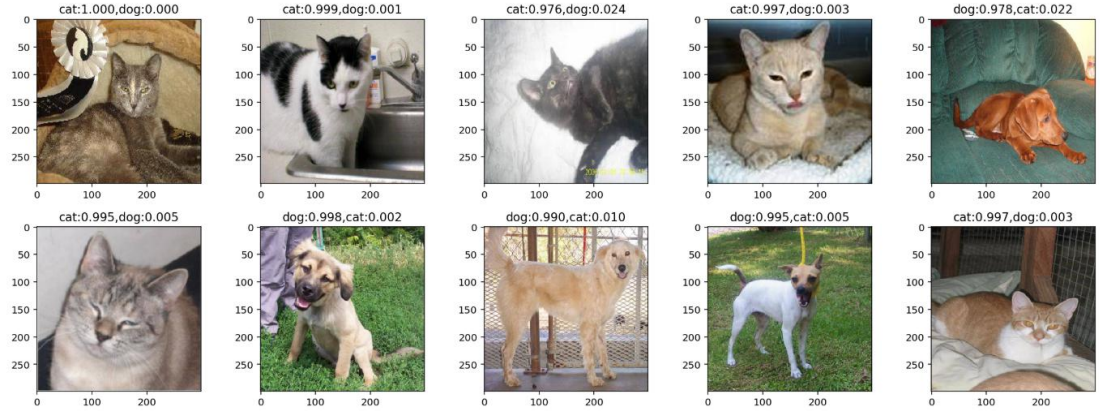
BaseModel(Vgg16)：

Densenet121：

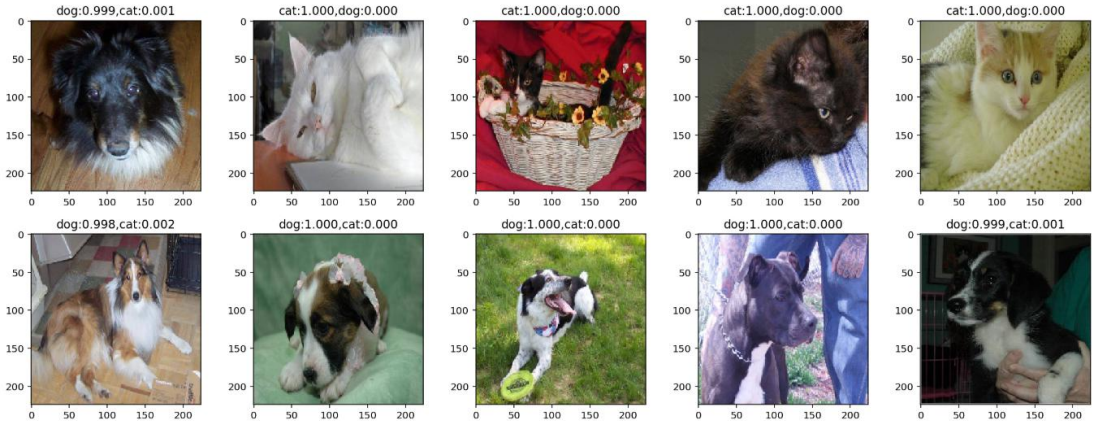


Resnet50

Inception v3



Merge Model



可以看到经过训练的各种模型都能很好的进行图片预测

对项目的思考

猫狗大战是一个非常有意思的竞赛项目，需要多方面考虑，尤其是对基准模型的构建，对深度神经网络有了更深的认识，从训练结果上也证明了迁移学习在机器学习当中的重要型，正所谓”站在巨人的肩膀上才能看得更远”.

CNN 神经网络在图像识别领域有着卓越的表现，同时 CNN 也是一个非常灵活的网络，可以通过各种自定义去实现最适合自己实际项目的模型。在构建模型的过程中，需要有足够的

耐心去逐步尝试，如模型的深度，内核的大小，步长，全连接网络的层数等，还有各种超参数的调节，多模型的对比验证，结果的可视化等

需要作出的改进

由于时间以及算力的限制，从模型的 Test loss 与 kaggle loss 对比可知，模型或多或少都有一些过拟合,resnet50,densenet121,inception_v3 模型可以进一步微调训练，原则上应该可以训练出来比 vgg16 更好效果的

在算力足够的情况下，应该尝试更多超参数组合的训练测试，进一步对 dropout,epoch 次数，优化器的选择，batch_size 大小等调参方式进行试验对比，以观测模型的效果;

选择更多优秀的模型进行对比验证,例如 Xception,NASnet 等,选用更多模型融合方案，例如训练时多模型 logloss 加权平均对 val loss 进行控制

另外结合 K 折交叉验证应该会提高模型性能，降低模型过拟合程度，对于后续项目应该考虑使用 K 折交叉验证

参考文献

1. https://github.com/udacity/cn-machine-learning/tree/master/dogs_vs_cats
2. https://pytorch.org/tutorials/beginner/saving_loading_models.html
3. https://blog.csdn.net/sinat_42239797/article/details/90646935
4. https://www.cnblogs.com/gzshan/p/10628289.html
5. https://pytorch.org/docs/master/torchvision/models.html
6. https://pytorch-cn.readthedocs.io/zh/latest/
7. http://blog.sina.com.cn/s/blog_4b6a6ae50102yda3.html