猫狗大战开题报告

Phil

更改增添的内容以黑体显示

领域背景

在本节中，提供有关提出项目的域的背景信息的简要详细信息。应包括与项目有关的历史信息。应该清楚域中的问题可以或应该解决的方式或原因。本节应适当引用相关的学术研究，包括该研究相关的原因。此外，我们鼓励讨论您调查域中特定问题的个人动机，但不是必需的。

本项目为计算机视觉领域的项目。

计算机视觉的历史是从60年代初开始的，Block World 是由Larry Roberts出版的一部作品，被广泛地称为计算机视觉的第一篇博士论文，其中视觉世界被简化为简单的几何形状，目的是能够识别它们，重建这些形状是什么。
原文：https://blog.csdn.net/qiumei1101381170/article/details/79904176
(https://blog.csdn.net/qiumei1101381170/article/details/79904176)

本项目为kaggle上一个非常有名非常经典的计算机视觉比赛——猫狗大战。这项比赛起始时间为2016年9月，结束为2017年3月，为期6个月。诸多高手过招，为计算机视觉领域提供了许许多多新的调参，模型的实践方法。 这一项目目的的在于将12500张图片分为猫和狗，训练集为25000张，数据集中除了训练集中的图片文件名带有cat，dog字样外，没有其他任何的事先处理和分类，所以在对数据集怎么处理才能使我们将其作为数据集是需要考虑的问题。而最终的预测结果，我们将利用已经在IMAGENET上训练过的一些已有的模型进行预测并输出一个值，该值将以0到1之间[闭区间]的数字呈现，0为猫，1为狗，也就是说越接近0，图片中越可能是猫，反之为狗。 本项目将使用pytorch，在windows10企业版中使用rtx2070，cuda进行训练。

问题陈述

在本节中，清楚地描述了要解决的问题。所描述的问题应该明确定义，并且应该至少有一个相关的潜在解决方案。此外，彻底描述问题，以便明确问题是可量化的（问题可以用数学或逻辑术语表示），可测量（问题可以通过一些度量来衡量并且可以清楚地观察到），并且可以复制（问题可以被复制并发生不止一次）。

这是一个监督学习二分类的问题，我们需要设计一个模型，使其能够根据图片的标记进行拟合，达到将任意一张新的图片分为猫或狗的目的。

那我们再往下需要解决的问题主要有三点，1.将训练图片分放到不同的文件夹，并正确加载数据，使其能够符合我们模型的输入要求。 2.在众多模型当中，找到一个最合适的对该模型进行训练。 3.导出测试集的结果，上传到kaggle以验证得分。

对于第一点，我们可以利用照片文件名里的cat，dog把文件分类到不同的文件夹里面，然后把这些图片文件进行一下转换，将其转换成向量的形式，因为本来计算机中图片通常就是RGB三个数值的向量构成的像素点构成的。由于RGB的数值最大为255，那么我们还需要对其进行归一化处理以避免产生过拟合或欠拟合问题。

第二点，我们可能就需要花些功夫了，这是这一项目的关键，也就是训练模型。我们使用的模型都将是带有卷积层的模型，卷积层会将图片的内容一层一层剥离开来，比如第一层识别线条，第二层识别线条组合，第三层识别矩形，圆形，第四层识别眼睛，耳朵等等。这些层叠加在一起就能使计算机得出一个结论。模型的训练就像教育小孩，你需要使其从错误中学习，设定一个loss函数，让它知道自己‘有多错’，‘错在哪’，还要设定一个改正器optimizer，帮助它从错误中以一定的学习速率进行改正；但训练模型又不像教育小孩，因为教育小孩无可重来，但模型的训练却需要模型结构，loss函数，优化器optimizer，等等各方面的组合才能从中得到最优解。所以解决方案就是多加尝试，并从中找出规律。

第三点比较简单，需要我们用训练好的模型，对测试集进行测试，然后将其提交并验证我们的想法即可。那么kaggle有一个排行榜，榜上总有1314名好汉，我们需要至少进入前10%。也就是说我们的预测准确性应该比榜上90%的人都要好。

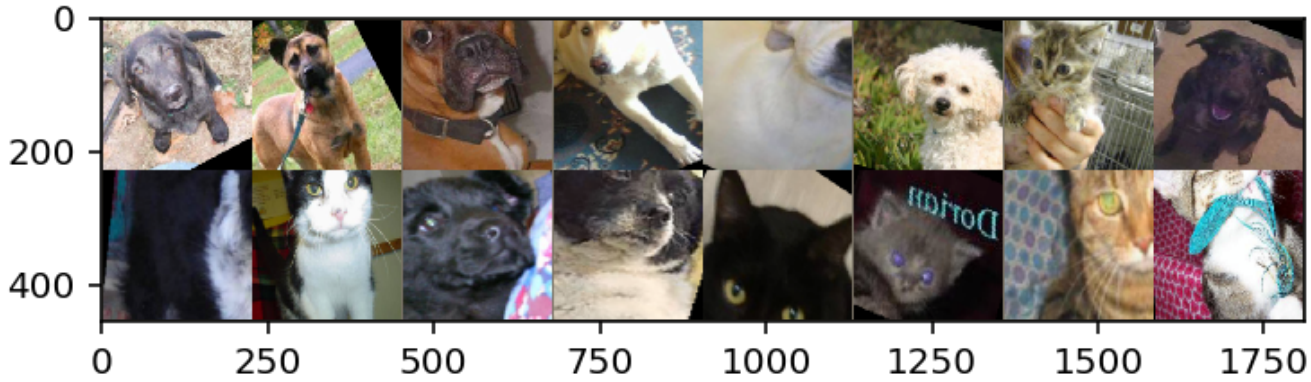
问题是可量化可测量的可复制的：模型参数固定，并且可以设置固定的random seed，虽然pytorch似乎没有这个功能，不过相同的数据集，相同的处理方式，训练出来的模型不会相差太多。而随机种子可以用在最后对模型进行验证的时候，随机抽取图像。

数据集和输入

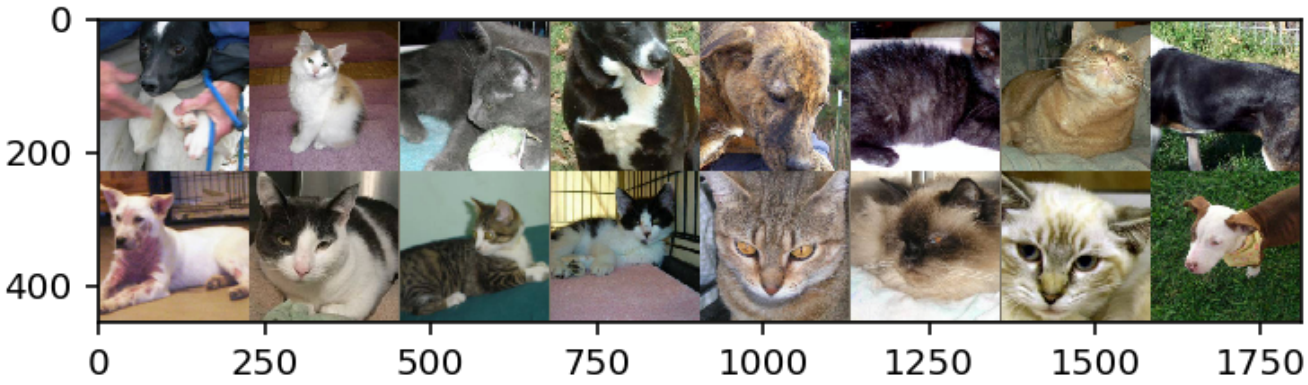
在本节中，应详细描述为项目考虑的数据集和/或输入，例如它们与问题的关系以及为何应该使用它们。获取数据集或输入的方式以及数据集或输入的特征等信息应包含在必要的相关参考文献和引文中。应清楚数据集或输入是如何进行的在项目中使用，以及根据问题的背景，它们的使用是否合适。

数据集为：猫与狗的图片，分为训练集和测试集。
与问题的关系及使用：问题就是从图片中分辨是猫还是狗，所以数据也是猫和狗的图片，训练集用于训练，测试集用于测试。 数据来源：https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/data (https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/data) ，图片具有标记1或者0，1就是狗，0就是猫。 使用是否合理：猫狗的图片一部分用于训练，一部分用于测试，能够使模型得到学习，也可以测试模型好坏，使用它们是合适的。

训练集图片数量为25000，测试集图片数量为12500。
训练集当中所有的文件名都以'猫狗.数字.jpg'的形式构成，猫狗即可以看作是kaggle预先对训练集做出的标记，猫为cat，狗为dog，我们可以利用这个特性将其分开，由于这是kaggle给我们的训练集，所以我们不考虑其中的离群值或者坏掉的没法用的图片，默认全部图片都是高宽大于等于224像素的标记正确的图片，因为大多数迁移学习的模型都需要图片高宽至少为224。
测试集中所有文件名都以‘数字.jpg’的形式构成，意味着没有标记，猫和狗混在一起，图片的性质同上，可以供大多数迁移学习的模型使用。还有一个需要注意的点是，我所使用的pytorch中的imagefolder的功能，并不会将训练集中的文件按照文件名放入datasets中，而是以['1', '10', '100', '1000', '10000', '10001', '10002', '10003', '10004', '10005', '10006', '10007', '10008', '10009', '1001', '10010', '10011', '10012', '10013', '10014']这样的排序，所以将预测结果和具体文件名对应起来，是一个需要处理的问题。
然后我们将数据集中的狗和猫的图片分好类后，可以随机展示一些图片，以便检查有没有分类错误的，遮挡等等的问题。
标记为：['dog', 'dog', 'dog', 'dog', 'dog', 'dog', 'cat', 'dog', 'dog', 'cat', 'dog', 'dog', 'cat', 'cat', 'cat', 'cat']
可以看到没有太大问题，只是有些图片仅展示了身体的一部分。对于这个问题，可以先尝试训练，之后再看这些只显示身体一部分，或者有遮挡的图片对结果是否有较大影响。



再看看测试集中的图片
标记为[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
但真实的文件名为['1', '10', '100', '1000', '10000', '10001', '10002', '10003', '10004', '10005', '10006', '10007', '10008', '10009', '1001', '10010', '10011', '10012', '10013', '10014']



接着随机打印一些图片的高宽，看图片是否适用于迁移学习中的预训练模型。
(336, 447, 3) (346, 499, 3) (189, 249, 3) (408, 499, 3) (342, 500, 3) 可以看到大多数图片宽高都大于224，不存在图片过小的问题。

解决方案描述

在本节中，清楚地描述了问题的解决方案。该解决方案应适用于项目域，并适用于给定的数据集或输入。此外，彻底描述解决方案，以便明确解决方案是可量化的（解决方案可以用数学或逻辑术语表示），可测量（解决方案可以通过一些指标来衡量并清楚地观察到），并且可以复制（解决方案可以被复制并发生不止一次）。

解决方案：针对训练集创建cat，dog两个文件夹。将文件名过滤，并针对含有不同字符cat，dog的，在两个文件夹内分别创建符号链接（可以节省空间，而不用复制粘贴进去）。并针对测试集，对每一个图片，生成一个新的文件夹，并在文件夹内创建符号链接，以备后续在dataloader中自动记录下文件名。

定义transform方式，针对训练集：随机旋转，翻转，改变大小resize，中央剪切，转换成tensor，归一化。针对测试集：改变大小resize，中央剪切，转换成tensor，归一化。然后使用对应的transform将数据加载进datasets,再使用dataloader将datasets转化成生成器，每个batch为16个（过拟合时将调小），这时训练集的shuffle为True。到此最初的数据预处理完毕。

使用resnet152这个较新的模型。将全连接层改为linear层，输出为2，冻结fc（全连接层）之前的所有参数，激活函数为logsoftmax，optimizer为NLLLoss。这样做是因为之前的花卉分类得到的经验，使我默认先进行这样一个尝试。

然后定义训练函数，我们将可以用accuracy和NLLLoss来看到模型学习的变化。然后使用训练好的模型，对测试集进行预测，预测结果提交到kaggle，将会有一个logloss函数计算得分，该得分越低越好，该得分变低了，意味着模型有所进步。再在这一解决方案的基础上，进行优化和改进。

基准模型

在本节中，提供与领域，问题陈述和预期解决方案相关的基准模型或结果的详细信息。理想情况下，基准模型或结果将现有方法或已知信息置于语境中并给出问题，然后可以客观地与解决方案进行比较。通过详细描述，描述基准模型或结果如何可衡量（可通过某种度量衡量并明确观察）。

由于本项目所涉及的指标有一个很统一的标准，就是在kaggle上的leaderboard，而该leaderboard是一个由logloss函数得分排名得出的，那么其中的杨培文的logloss得分是值得参考的。而我所使用的rtx2070应该足以媲美2016年的旗舰单张gpu，再加上模型优化，各项技术的进步，应该无需过多考虑机器性能上的问题。杨培文的模型在验证集上正确率最高达到99.6%，在leaderboard上排名为前2%，logloss得分为0.04008。所以本项目的目标定为，最高准确率能达到99.2%，达到kaggle的leaderboard前5%，logloss得分小于0.05。他的github 上利用tensorflow的猫狗大战代码：https://github.com/ypwhs/dogs_vs_cats (https://github.com/ypwhs/dogs_vs_cats)

评估指标

在本节中，提出至少一个评估指标，可用于量化基准模型和解决方案模型的性能。鉴于数据的上下文，问题陈述和预期的解决方案，您建议的评估指标应该是适当的。描述评估度量的推导方式，并提供其数学表示的示例（如果适用）。复杂的评估指标应明确定义和量化（可以用数学或逻辑术语表示）

评估指标：LogLoss=−1/n∑[yi*log(y_i)+(1−yi)*log(1−y_i)]
n：测试集中的图像数
y_i：图像是狗的预测概率
yi：图像标记，狗为1，猫为0
log()：自然对数
以上为kaggle的logloss函数，最终得分非常重要，它决定了我们能否符合毕业要求，达到相应的分数，才能进入前10%。 kaggle的leaderboard得分排名就是最重要的指标，其他次要指标为损失函数的大小，准确率。

项目设计

在最后一节中，总结了解决问题的解决方案的理论工作流程。提供有关您可能考虑采用的策略，在使用之前可能需要对数据进行哪些分析，或者将为您的实施考虑哪些算法的详细讨论。您提供的工作流程和讨论应与前几节的质量保持一致。此外，建议您包括小可视化，伪代码或图表以帮助描述项目设计，但这不是必需的。讨论应清楚地概述您的顶点项目的预期工作流程。

第一步：预处理数据，将图像文件分类到两个子文件夹。并将其载入datasets，这里需要用到transform，因为resnet，densenet等模型接受的输入数据是一定的。针对训练集：随机旋转，翻转，改变大小resize，中央剪切，转换成tensor，归一化。针对测试集：改变大小resize，中央剪切，转换成tensor，归一化。然后使用对应的transform将数据加载进datasets,再使用dataloader将datasets转化成生成器，每个batch为16个（过拟合时将调小），这时训练集的shuffle为True。到此最初的数据预处理完毕。
第二步：使用resnet152进行训练，不过开始之前，需要将其全连接层换成输出为1，或者2的全连接层，接着将全连接层以前的所有层的参数全部冻结。激活函数使用sigmoid或者logsoftmax，loss函数使用BCELoss或者CrossEntropyLoss。optimizer使用Adam或者Adadelta，SGD。紧接着对模型进行训练。
第三步：对测试集进行预测，pytorch有个特性叫autograd，那么在我们进行预测时需要用model.eval, with torch.no_grad, 或者requires_grad = False 使其不再自动记录梯度。接着将结果提交kaggle查看结果。
第四步：针对不同的模型，重复第二第三步，对激活函数，loss函数进行优化。

参考文献

For Resnet:
Kaiming He, Xiangyu Zhang, Shaoqing Ren & Jian Sun. (2015 Dec 10). Deep Residual Learning for Image Recognition.

For Densenet:
Gao Huang, Zhuang Liu, Laurens van der Maaten & Kilian Q. Weinberger.(2016 Aug 25). Densely Connected Convolutional Networks.

Pytorch Transfer Learning Tutorial:
https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html (https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html)

For Sigmoid, Logsoftmax, BCELoss & CrossEntropyLoss:
https://pytorch.org/docs/stable/nn.html?highlight=sigmoid#torch.nn.Sigmoid (https://pytorch.org/docs/stable/nn.html?highlight=sigmoid#torch.nn.Sigmoid)
https://pytorch.org/docs/stable/nn.html?highlight=logsoftmax#torch.nn.LogSoftmax (https://pytorch.org/docs/stable/nn.html?highlight=logsoftmax#torch.nn.LogSoftmax)
https://pytorch.org/docs/stable/nn.html#torch.nn.BCELoss (https://pytorch.org/docs/stable/nn.html#torch.nn.BCELoss)
https://pytorch.org/docs/stable/nn.html?highlight=crossentropy#torch.nn.CrossEntropyLoss (https://pytorch.org/docs/stable/nn.html?highlight=crossentropy#torch.nn.CrossEntropyLoss)