Homework3 ： 手写字符识别

问题描述

计算机视觉(Computer Vision)是数据分析与人工智能的一个重要研究方向与应用领域，同时受到了研究者和工业界的广泛关注。特别是近年来随着深度学习技术的发展，计算机视觉在多项任务取得了超过人类水平的效果，而诸如“刷脸支付”等技术也逐渐普及在我们的日常生活中。在第三次作业中，我们将探索用深度学习方法处理一个常见但十分重要的计算机视觉问题：手写字符识别，即给定一些手写字符的图片和它们的标签，我们希望训练一个深度学习模型，对于新的手写字符图片，该模型可以预测出其对应的字符。

数据描述

本次作业中，我们将采用 Omniglot 数据 (https://github.com/brendenlake/omniglot) [1][2]。该数据集包涵 50 种不同字母系统中的共计 1623 种字符。对于这 1623 种字符中的每一种字符，数据集收集了由 20 个不同人书写该字符的手写图片。用机器学习的术语，该数据集可以看作一共由 1623*20=32460 个样本组成，样本被均匀分到 1623 类中。

与传统常用的手写字符识别数据集 MNIST 不同（该数据集为识别 0-9 的手写数字），Omniglot 样本类别更多，且覆盖了不同字母系统，因此无论从研究还是应用领域，该数据集都更具有实际意义与挑战性。此外，虽然该数据集类别众多，但每张图片的尺寸较小（我们采用的数据集中，每张手写图片表示为 28x28 的黑白像素点），因此相比于 CIFAR-10、ImageNet 等真实图片分类数据集，该数据集所需计算资源较少（例如，更适合于没有 GPU 计算时使用）。

所有数据均存储于 omniglot_resized 文件夹中，该文件夹中有 50 个子文件夹，对应 50 种不同的符号系统，每个符号系统文件夹内又有若干文件夹，对应该符号系统的字符集，每个字符集中包含 20 张图片。我们在 utils.py 中提供了一个读取图片、划分训练/测试数据集的函数供大家参考（当然，你也可以自己写函数进行读取）。

Omniglot 数据集示意图

[1] Human-level concept learning through probabilistic program induction. Science, 2015.
[2] The Omniglot Challenge: A 3-Year Progress Report. arXiv:1902.03477.

实验要求

本次实验中，我们希望大家围绕深度学习算法在手写字符识别数据上的应用进行探索。以下是一些具体要求：

(1) 预设实验环境：随机从所有类别中取出 50 类进行分类，每个类别中使用 15 张图片作为训练数据，5 张图片作为测试数据。要求对比(a) 至少有一个隐层的全连接神经网络; (b) 一种基于卷积神经网络（CNN）的模型，该模型可以是现有架构，也可以为自己设计。其他超参数（如优化器、梯度下降步长、迭代次数等）均可自行设置。汇报实验结果，并比较不同的方法有何优缺点。

(2) 自行尝试与对比，包括但不限于以下方面：

a) 数据集的划分，例如类别数量、训练/测试样本数量等，如何影响实验结果？

b) 模型的影响，例如不同全连接网络/卷积神经网络架构，如何影响实验结果？

c) 超参数的影响：不同超参数（如初始化方式、优化器、迭代次数、梯度步长等）对结果有何影响？

d) 在实验中是否观察到欠拟合、过拟合等现象？请明确说明判断理由。如果是，如何解决？

e) 若在实验(1)中未使用的话，可以尝试一些深度学习技巧如 dropout, batch normalization 等。

注：此部分较为开放，鼓励大家多做尝试，无固定标准答案，言之合理即可。

报告要求

实验报告中应该包括以下内容：

(1) **(60 分)** 在实验要求(1)中采用的两种模型结构及其他实验设定，汇报实验结果，比较不同方法的优缺点。

作为参考线，我们尝试了一个简单的 CNN（结构如下），可以达到 85%左右的准确率（该结果仅作为参考，并非要求一定达到该结果）

(2) **(30 分)** 在实验要求(2) 中至少进行三个方面的尝试与对比，汇报结果并进行必要讨论。

(3) **(10 分)** 其他的思考、感受等。

- Conv2D: filters = 32; kernel_size = (3, 3); padding='valid'
- BatchNormalization
- Relu
- Conv2D: filters = 32; kernel_size = (3, 3); padding='valid'
- BatchNormalization
- Relu
- MaxPooling2D: pool_size = (2,2)

- Conv2D: filters = 64; kernel_size = (3, 3); padding='valid'
- BatchNormalization
- Relu
- Conv2D: filters = 64; kernel_size = (3, 3); padding='valid'
- BatchNormalization
- Relu
- MaxPooling2D: pool_size = (2,2)

- Dense: 512
- BatchNormalization
- Relu
- Dropout
- Dense: 50
- Softmax

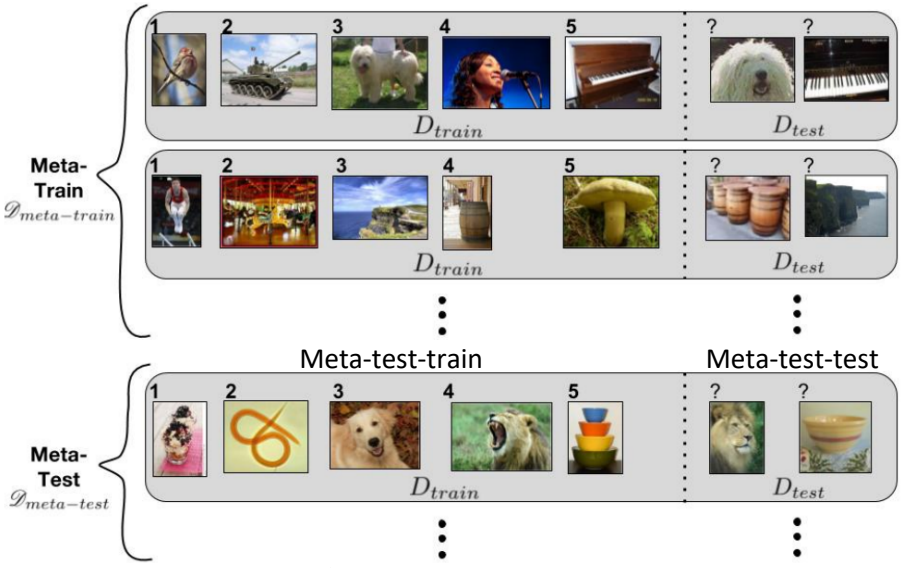
参考线的 CNN 结构图

选做实验：元学习

在前面的实验中，你已经尝试完成了经典的监督学习实验场景。然而，正如其名字中所隐含的，监督学习的一个缺点在于它往往需要较多的监督信息。在我们的实验中，这就代表需要首先人工标记出很多的手写字符，作为机器学习模型的训练样本。然而，在很多场景中，数据集的标注无疑是耗时耗力的。小样本学习（few-shot learning）是专门针对训练样本极少时的一类研究，而其中，基于元学习（meta-learning）的方法是一类具有代表性的方法并受到了广泛关注。

元学习的基本思想是，除了目标问题的少量样本/数据，我们经常还有其他易于获得的数据（这些数据往往与我们的目标问题不完全一致，但有一些相关性）。那么，这些额外数据能否帮助我们更好解决小样本学习的问题呢？考虑在我们人类的学习过程中，如果有一些先验知识，那么学习新的知识会变的更容易。例如，你已经学习过一门编程语言并从中总结出一些规律，那么学习一门新的编程语言时，往往可以“触类旁通”，从而学习的更快。元学习的目标即学习这种快速学习的能力（learning-to-learn）。

基于该思想，元学习一般可以分为两个阶段：元训练（meta-train）与元测试（meta-test）。在元训练阶段，我们利用元训练数据（meta-train data，即易于获得但与我们目标问题不完全一致的数据）得到一些先验知识，这些知识可以是显式、可解释的，也可以是隐式的，如神经网络的权重。然后，在元测试阶段，我们将这些“知识”用到我们的目标问题中。元测试阶段往往可以继续细分为一个训练阶段与一个测试阶段，我们将其记为元测试-训练（meta-test-train）与元测试-测试（mete-test-test）。在元测试-训练中（有时也称为适应阶段，adaptation），我们利用搜集到目标问题的少量样本，将“知识”应用于/迁移于我们想解决的问题中；在元测试-测试阶段，我们测试最终算法的效果。例如，在手写字符识别问题中，我们可以利用一些常见、有大量数据的字符集（例如英文）进行元训练，得到一些元知识，然后在新的字符集中（例如某种使用人数较少的语言）收集少量数据（例如每个字符一个样本），利用这些数据进行迁移（即元测试-训练），然后观察模型的精度（即元测试-测试）。需要注意，元测试-训练阶段往往是必须的，因为我们在元训练中得到的“元知识”可能是宽泛、广义的（例如，人类手写字符的一些共性），必须经过元测试-训练才能将其真正迁移、适应到我们的目标任务中。



元学习设定示意图（Ravi & Larochelle, ICLR 2017）

实验要求：在给定实验设定下，即元测试阶段有50类，每类提供1个样本（即元学习中的50-way-1-shot），元训练数据可以采用数据集中任意非元测试类的数据，至少对比一种非元学习方法（例如，直接使用前面实验中的一种网络模型）与一种元学习方法的效果。元学习方法可以是任意一种经典方法（参考下列资料）或自行设计的方法。汇报实验结果，并分析该元学习方法如何能帮助解决训练样本较少的问题。

该部分视完成情况，最多给与不超过**25分**的加分。

参考资料：

朱文武老师与王鑫老师的tutorial: https://github.com/Yue-Liu/AutoML-Meta-tutorial

相关课程与综述：

[1] Meta-Learning: A Survey, arXiv 1810.03548.

[2] Meta-Learning in Neural Networks: A Survey, arXiv 2004.05439.

[3] CS330: Deep Multi-Task and Meta-Learning by Chelsea Finn, Stanford Univ.

提交要求:

(1) 报告可以用中文或英文，不超过10页A4纸。选做实验部分可额外增加不超过8页A4纸。

(2) 请将所有代码放在一个文件中。注意，我们并不会执行大家的代码，而仅作为查重使用。不要提交任何中间结果或原始数据。

(3) 所有的提交内容应打包成一个压缩文件上传。压缩文件名，以及所有的报告、代码，均需以学号命名，如2020001002.pdf, 2020001002.py, 2020001002.zip. 请仔细检查（因为我们将用一个代码查重软件进行查重，不合规的命名只能手动修改）

(4) 如果不符合上述提交要求，会扣除5%的分数。

(5) 迟交标准：每迟交一周，将会减去20%的分数。计算周数将向上取整，例如，若迟交一天，将按照$\lceil 1/7 \rceil = 1$周，即20%标准扣分。请尽早开始！

(6) 该作业每人独立完成。严禁抄袭他人的代码、结果、报告或其他任何资料，包括在线或开源内容，例如github上的代码或论文作者公开的代码。但我们允许使用软件中的自带功能，如matlab、python中的各种库，或者机器学习框架如pytorch、tensorflow、keras等中的函数、层、自动求导等功能。请注意，抄袭他人成果以及主动提供资料供他人抄袭，均被视为学术不端（所以，请勿将代码、报告等发给别人）。学术不端将会导致该次作业0分，和/或本课程挂科，和/或其他《清华大学学生纪律处分管理规定实施细则》规定的处罚。

学术不端是红线，我们将严格执行，请务必注意！