**《神经网络课程设计》课程报告**

**课题名称：****神经网络框架搭建与测试应用**

**课题小组成员名单**

吉昱阳 2019141460203

琚 理 2019141220183

刘易知 2019141410285

唐 润 2019141460237

王 耀 2019141460240

指导教师： 郭 泉

**评阅成绩：**

评阅意见：

提交报告时间：2021 年 12 月 01 日

# 目录

[目录 1](#_Toc88674955)

[1. 绪论/背景 2](#_Toc88674956)

[1.1. 目的/意义 2](#_Toc88674957)

[1.2. 相关工作 2](#_Toc88674958)

[2. 项目安排 2](#_Toc88674959)

[2.1. 任务分工 2](#_Toc88674960)

[2.2. 进度安排 2](#_Toc88674961)

[3. 方法与实现 2](#_Toc88674962)

[3.1. 总体设计 2](#_Toc88674963)

[3.2. 模块实现 2](#_Toc88674964)

[4. 实验与分析 3](#_Toc88674965)

[4.1. 实验设计 3](#_Toc88674966)

[4.2. 实验结果 3](#_Toc88674967)

[4.3. 结果分析 3](#_Toc88674968)

[5. 结论 3](#_Toc88674969)

绪论/背景

近年来，随着深度学习方法的不断发展和日渐火热，人工智能领域也走向了计算机科学学科舞台的中央。现如今，深度学习算法和深度神经网络在图像、文字、语音等各种数据类型的处理和分析上都占有重要的一席之地。而为了提高神经网络模型的开发效率，研究人员设计了很多方便人工智能科学家和神经网络模型研究者开发新模型的代码框架，并迅速获得了广泛的使用和好评。作为人工智能专业的同学，同时也是人工智能特别是神经网络领域的初学者，我们小组五名同学也基于自己的知识背景和对现存神经网络框架的了解，尝试进行了自主的神经网络框架开发。此外，该神经网络训练框架的开发工作也充分考虑到课程的具体要求，完成了其中的必选模块并扩展完成了大量可选内容。目前，本小组的开发工作已基本完成，而此次的开发经历也对于我们继续深耕人工智能领域有着极大的意义。

目的/意义

总的来说，神经网络训练框架的开发对于深度学习领域发展以及其应用有着重要的意义；另外，自主进行这样的深度学习训练框架开发对于还是学生的我们也是大有裨益的。以下，我将从产业视角和我们的个人发展视角阐述本次神经网络训练框架开发的实践意义。

首先，从宏观的行业需求来讲，神经网络训练框架的产生大大方便了算法工作者对于深度神经网络具体模型的实现，提升了模型构建的效率，有力促进了新的、具有更优异性能模型的诞生。到目前为止，深度学习方法的应用领域基本已经覆盖了传统计算机研究范畴的方方面面，例如处理计算机视觉问题的CNN模型、解决自然语言处理问题的NLP领域模型等等，如此庞大的行业需求使得研究者必须在面对不同的具体问题时高效设计并实现相应的模型。此时，神经网络在训练和学习过程中过于繁杂的底层数学和算法原理又会很大程度上限制模型的开发效率。而众所周知，神经网络和深度学习领域，是非常典型的“站在巨人的肩膀上”不断发展的学科，很多的网络结构、学习算法、神经元实现都是相互借鉴，可以复用的。神经网络训练框架正是因此而诞生，用来将很多可以复用、共享的代码部分封装起来，形成框架，提供接口。从而使得搭建神经网络模型的算法工程师可以不再关注具体的算法实现细节，把模型的构建变得如同搭积木一样便利，进而提高模型搭建的搭建效率。反过来说，也正是因为当前市面上各类神经网络训练框架的存在，才使得神经网络和深度学习方法在各个领域全面开花，爆炸式发展，其现实意义可见一斑。

其次，从我们个人的角度来讲，尝试进行神经网络训练框架的开发不但可以帮助整合、理解目前学过的种种关于神经网络的概念和具体的代码实现，更能让我们学习现如今最为主流神经网络框架的设计理念，并对框架性的编程有更多了解和实践经验。可以说，本次实践任务对于让我们深刻辨析、理解关于神经网络的种种概念，并将其按照一定的体系付诸实践有着极大的意义。

相关工作

关于神经网络训练框架的历史最早也可以追溯到神经网络兴起的初期。在人们开始尝试使用深度学习算法解决实际问题时，深度学习框架就开始产生，并投入广泛使用。从最早由Yoshua Bengio领导的MILA创建和维护的Theano框架，到如今众多深度学习框架的百花齐放（见图1-1），深度学习框架的发展已经十分成熟。

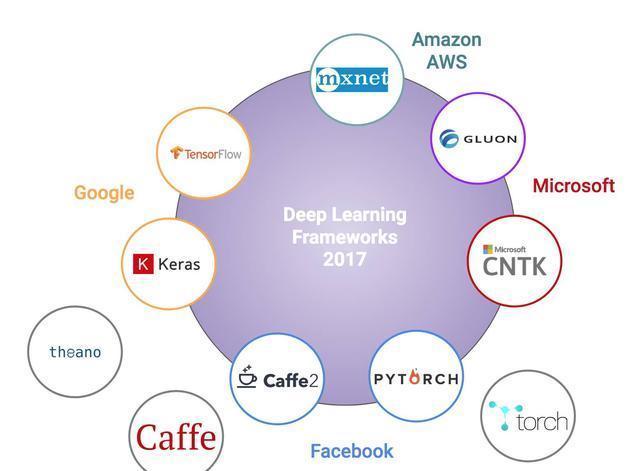


图1-1 2017年深度学习框架的市场局面

而截至现在，使用最广泛、发展最为完善的深度学习框架也正是图中所示的这些基础框架。可以看出，这些框架大多是由当今世界在计算机领域最具有话语权的几家科技公司投资并支持开发的，并且各自都有着自己的特色和庞大的用户群体。，以下，以其中的TensorFlow和Pytorch为，简单的介绍下各自的框架构成与特点。

**TensorFlow**：该框架是一个基于数据流编程(dataflow programming)的符号数学系统，被广泛应用于各类机器学习(machine learning)算法的编程实现，其前身是谷歌的神经网络算法库DistBelief 。Tensorflow拥有多层级结构，可部署于各类服务器、PC终端和网页并支持GPU和TPU高性能数值计算，被广泛应用于谷歌内部的产品开发和各领域的科学研究。（引用：https://baike.so.com/doc/24441014-25277662.html）该框架的最大特点就是对于神经网络的搭建灵活度高，并且数值计算方法设计透明，并且具有很强的移植性。可以说，TensorFlow库非常适合科研人员开发神经网络模型，并能够有效的模型的科研价值与产品价值联系在一起。

Pytorch：Pytorch是torch的python版本，是由Facebook开源的神经网络框架，专门针对 GPU 加速的深度神经网络（DNN）编程。Torch 是一个经典的对多维矩阵数据进行操作的张量（tensor ）库，在机器学习和其他数学密集型领域有广泛应用。与Tensorflow的静态计算图不同，pytorch的计算图是动态的，可以根据计算需要实时改变计算图。作为经典机器学习库Torch的端口，PyTorch为Python语言使用者提供了舒适的写代码选择。（引用：<https://blog.csdn.net/bestrivern/article/details/89433023>）。相较于TensorFlow框架，Pytorch神经网络训练框架更为简洁和清晰易懂，更加贯彻了面向对象的程序设计理念，使得人们在搭建神经网络像搭积木一样简单易行。

而在目前的研究和市场背景下，神经网络框架也是随着整个产业的发展不断更新换代，而很多框架更是因为新的网络模型的提出而不断迭代补充，完善框架的功能。此外，对于不同的框架，也有越来越多的开发者开发了相应的扩展库来补充和更新框架的功能。现如今，还有很多框架更是基于Pytorch、TensorFlow等基础框架搭建出的“高层框架”，它们通常是只针对某一具体领域产生的统一的框架，用于搭建对应领域常用的网络模型。这些框架更是五花八门，如面向自然语言处理的AllenNLP库、面向图数据学习的PyG库等等，在此处也不再过多赘述。

项目安排

任务分工

进度安排

方法与实现

总体设计

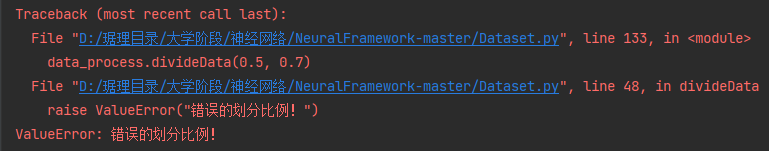
模块实现

### 数据集加载与划分模块

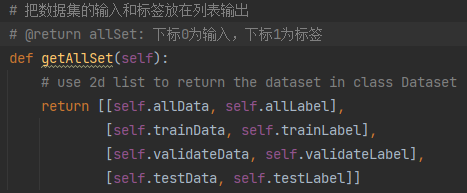
考虑到数据集的一些基本的分割和存取操作在目前的很多框架中都并没有涉及，因此，本框架为方便使用者，特意设计了Dataset类用来管理开发者需要在模型训练中使用的数据。在该模块中，我们主要实现了以下功能：数据集的分割、数据集的部分获取、数据集的本地存储以及本地载入。而同时，若要使用我们所提供的神经网络训练框架，开发者必须首先将数据集使用Dataset类进行管理，并以类对象的方式输入模型，以供模型使用。在Dataset类中，我们实现了以下函数接口：

1. divideData(trainRatio=0.6, validateRatio=0.2)：当初始化输入数据时，用户只提供了整体的数据集，此时用户可以借助该接口将数据按照自己需要的比例（trainRatio参数、validateRatio参数确定）随机划分成不同的集合并存储在类成员中。注意，在此处实现时需要注意进行异常判断，当分割比例异常时及时报错。

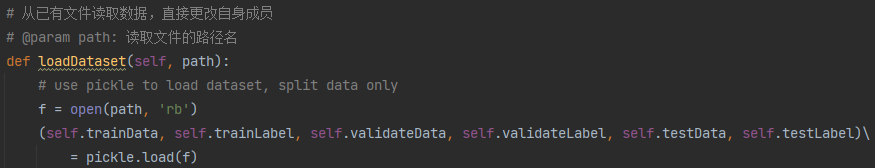




1. getAllSet()：该接口用于将数据样本和标签按照二维列表的形式返回给用户，具体实现如下：



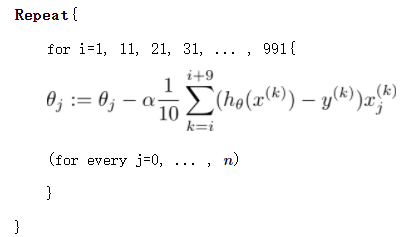
1. setAllSet(allSet)：当用户初始化Dataset成员时没有输入相应的数据集，可以调用该接口重新设置数据集，注意输入时应以二维列表的形式，将每一对数据样本和标签打包在一起传入。
2. saveDataset(path)：该接口能够将Dataset成员中的数据集和标签使用pickle格式存储到指定的路径下，方便下一次或是在其他使用本框架的程序中读取。
3. loadDataset(path)：与上一接口功能相反，用于沿着指定路径读取数据集。注意，此处也仅能读取按照上一接口格式保存的数据，并不能读取任意数据文件。



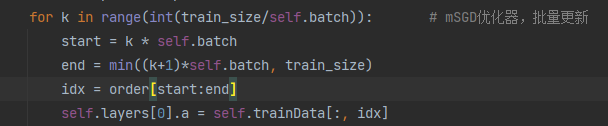
总结起来，该模块能够适应用户各式各样的数据集初始化方式，并提供有效的简单接口用于数据集的分割和保存，极大方便了用户在训练模型中的数据集管理。

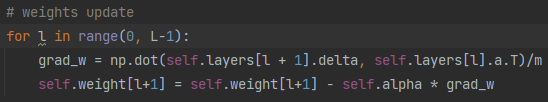
### 前馈网络mSGD模块

该模块作为Model类的子类，重写了Model中的train()函数，主要用于标准的全连接前馈神经网络的反向传播训练。而其中，mSGD是一种十分常用的训练优化方法，译为小批量梯度下降法。该算法是批量梯度下降法和随机梯度下降法的折衷方式，每次采取一个固定batch的数据用来训练模型参数，这样的方法能够兼顾模型的训练迭代次数以及收敛的最终参数训练的准确率。其伪代码形式如下式：

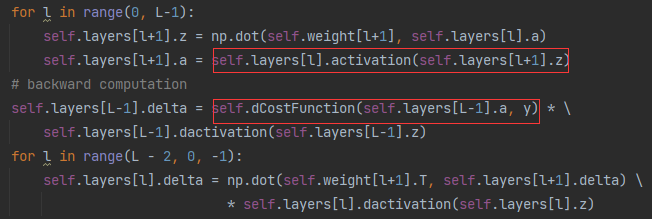


而在代码中，具体的模块实现需要保持代码的内聚性，调用模型中存储好的layers、depth等数据成员实现，更新时也都直接更新这些模型存储的数据成员。而对于mSGD的优化器，其特殊性就体现在每次使用一个固定batch的数据训练参数，而在更新w时也要注意除以对应batch的大小，如下图所示：

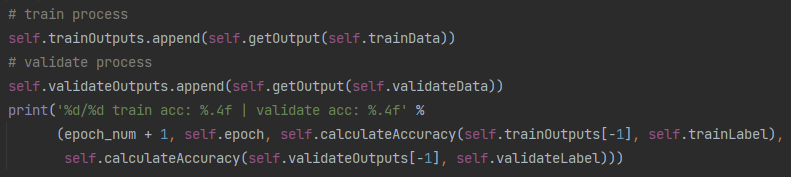


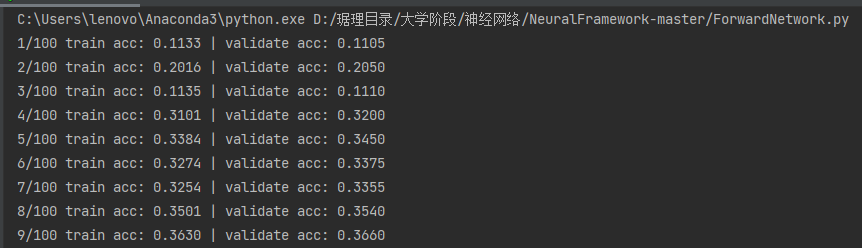


实现的训练过程就是正常点前馈计算和反向传播以及对应的参数更新，此处只需注意权重和盛神经元层的序号即可，以保证传播中数据维度的正确。另外，我们还给用户在每个神经层的激活函数以及模型整体的性能函数选取上都提供了极大的自由度。在对应位置，我们也将调用用户定义的性能函数和激活函数完成前向传播和反向计算的求导过程。



而在训练的迭代过程中，我们也提供了每一轮中间过程的输出，并将中间结果保存至模型对应的数据成员中。以便用户能够清晰的感知目前的训练状态以及模型的训练效果。

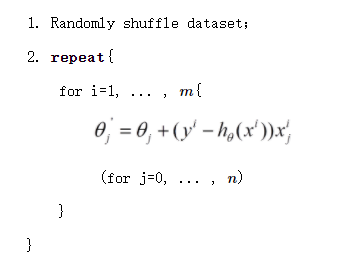




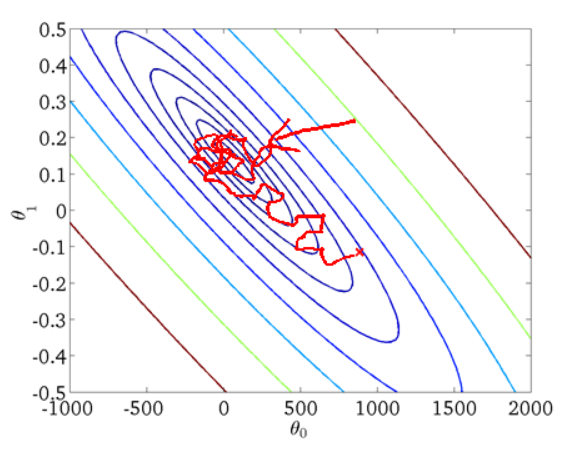
综上，即是mSGD训练优化方法的基本实现，我们小组也在最后针对不同的优化器训练方式设计了大的实验进行横向的性能比较。

### 前馈网络SGD模块

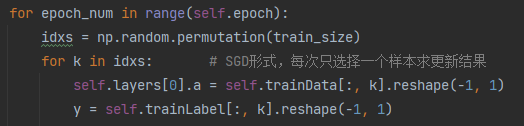
与mSGD模块类似，该模块同样重写了Model的train()方法，也是一种较为基础的训练优化方法，即随机梯度下降优化器。mSGD算法也正是结合了SGD和BGD两种算法的相对优点的折衷方法。在SGD算法中，每一次的参数更新仅仅只需要一条数据样本进行训练，而每轮训练则都要遍历训练集中的所有样本，该优化迭代方法的伪代码如下所示：



可以注意到，使用该优化迭代算法时，对于每一轮的训练都要打乱整个训练集的顺序，确保每次迭代更新选用样本的随机性。而在实践中，SGD算法相较于mSGD算法来说，每次更新参数都只针对一个数据样本而非一堆样本的均值，因此更新的效率高，需要的总迭代次数少。但是，SGD伴随的一个问题是噪音较BGD要多，使得SGD并不是每次迭代都向着整体最优化方向。其收敛曲线如下图所示：



而在具体的代码实现上，所有的前向计算、反向传播、参数更新、中间过程展示等细节都与mSGD模块的代码实现无二，因此不再过多的赘述与展示。唯独只有在参数更新的迭代方式上有显著的变化，此处每次迭代只随机选取训练集的其中一个样本，即：



在这样的情况下，我们只需要设置更少的迭代次数（epoch）就可以达到最终的收敛。



但实际上，由于每次只使用一个样本来更新参数，导致每轮迭代的时间成本更大，因而使得最终10轮迭代所消耗的时间反而比mSGD优化器迭代100轮时还要更长。

实验与分析

实验设计

实验结果

结果分析

结论