《认知科学与类脑计算》实验指导书

第一部分：《认知科学与类脑计算》实验大纲

一、认知科学与类脑计算实验的地位与作用

认知科学与类脑计算是20世纪世界科学标志性的新兴研究门类，它作为探究人脑或心智工作机制的前沿性尖端学科，已经引起了全世界科学家们的广泛关注。一般认为认知科学的基本观点最初散见40年代到50年代中的一些各自分离的特殊学科之中，60年代以后得到了较大的发展。认知科学与类脑计算是一门相当年轻的学科，然而却为揭示人脑的工作机制这一最大的宇宙之谜作出了不可磨灭的贡献。

本课程较系统地介绍了认知科学与类脑计算中的常用算法，内容非常丰富。本课程的学习将为后续课程的学习以及软件设计水平的提高打下良好的基础。《认知科学与类脑计算》课程内容丰富，学习量大，给学习带来一定的困难；所用到的技术多，而在此之前的各门课程中所介绍的专业性知识又不多，因而加大了学习难度；隐含在各部分的技术和方法丰富，也是学习的重点和难点。根据《认知科学与类脑计算》课程本身的技术特性，设置《认知科学与类脑计算》实践环节十分重要。通过实验实践内容的训练，突出学生程序思维训练和动手上机调试程序的能力，目的是提高学生编写类脑计算算法及大型程序的能力。

二、认知科学与类脑计算实验的目的

该课程使学生不仅能够深化理解教学内容，进一步提高灵活运用类脑计算算法和程序设计技术的能力，而且可以在总是分析、总体结构设计、算法设计、程序设计、上机操作及程序调试等基本技能方面受到综合训练。实验着眼于原理与应用的结合点，使学生学会如何把书本上和课堂上学到的知识用于解决实际问题，从而培养类脑计算工作所需要的动手能力。

不少学生在解答习题尤其是算法设计题时，觉得无从下手，做起来特别费劲。实验中的内容和教科书的内容是密切相关的，解决题目要求所需的各种技术大多可从教科书中找到，只不过其出现的形式呈多样化，因此需要仔细体会，在反复实践的过程中才能掌握。

为了帮助学生更好地学习本课程，理解和掌握算法设计所需的技术，为整个专业学习打好基础，要求运用所学知识，上机解决一些典型问题，通过分析、设计、编码、调试等各环节的训练，使学生深刻理解、牢固掌握所用到的一些技术。认知科学与类脑计算中稍微复杂一些的算法设计可能同时要用到多种技术和方法，这就要求学生在掌握基本算法的基础上，掌握分析、解决实际问题的能力。

三、认知科学与类脑计算实验的实验要求

1、阅读实验指导书

每一次实验从阅读实验指导书开始。对于本次实验的实验目的、实验题目、实现提示以及思考题目、选做题目等应认真了解。

2、算法设计

分析实验题目，参考实现提示，进行算法设计。

3、程序设计

根据已完成的算法，用 python 语言进行程序设计。

4、调试和测试

将所编程序在计算机上调试通过，并选取若干组测试数据对程序进行尽可能全面的测试。

5、整理完成实验报告

实验报告一般包括下列内容：

（1） 实验者姓名、学号、专业和班级，课程名称，实验日期等；

（2） 本次实验的实验编号及实验名称

（3） 本次实验的实验目的；

（4） 本次实验的硬件及软件环境；

（5） 程序结构的描述及各模块的规格说明；

（6） 主要算法及其基本思想；

（7） 调试过程简述（调试过程是否顺利，遇到些什么问题，如何解决的，以及上机操作所花费的时间等）；

（8） 测试数据和相应输出的客观纪录，对运行结果的分析讨论。

四、《认知科学与类脑计算实验》实验环境及背景介绍

Python 是一种计算机程序设计语言。是一种动态的、面向对象的脚本语言，最初被设计用于编写自动化脚本(shell)，随着版本的不断更新和语言新功能的添加，越来越多被用于独立的、大型项目的开发。Python 的创始人为荷兰人吉多·范罗苏姆，由于 Python 语言的简洁性、易读性以及可扩展性，在国外用 Python 做科学计算的研究机构日益增多，一些知名大学已经采用 Python 来教授程序设计课程。Python 在设计上坚持了清晰划一的风格，这使得 Python 成为一门易读、易维护，并且被大量用户所欢迎的、用途广泛的语言。由于这些得天独厚的优点，Python 语言在类脑计算领域得到广泛的使用。

PyTorch 是一种简洁优雅且高效快速的深度学习框架。PyTorch 在灵活性、易用性、速度这三个方面具有明显的优势。2017 年 1 月，Facebook 人工智能研究院(FAIR)团队在 GitHub 上开源了 PyTorch，并迅速占领 GitHub 热度榜榜首。作为一个 2017 年才发布，具有先进设计理念的框架，PyTorch 的历史可追溯到 2002 年就诞生于纽约大学的 Torch。Torch 使用了一种不是很大众的语言 Lua 作为接口。虑到 Python 在计算科学领域的领先地位，以及其生态完整性和接口易用性，几乎任何框架都不可避免地要提供 Python 接口。终于，在 2017 年，Torch 的幕后团队推出了 PyTorch。PyTorch 不是简单地封装 Lua Torch 提供 Python 接口，而是对 Tensor 之上的所有模块进行了重构，并新增了最先进的自动求导

系统，成为当下最流行的动态图框架。PyTorch 一经推出就立刻引起了广泛关注，并迅速在研究领域流行起来。

认知科学与类脑计算实验常用的 Python 包有 numpy，sklearn 等。NumPy 系统是 Python 的一种开源的数值计算扩展。这种工具可用来存储和处理大型矩阵，比 Python 自身的嵌套列表（nested list structure)结构要高效的多（该结构也可以用来表示矩阵（matrix））。scikit-learn 是数据挖掘与分析的简单而有效的工具，依赖于 NumPy，SciPy 和 matplotlib。

认知科学与类脑计算实验常用的数据库有：MINIST 和 CIFAR。MNIST 数据集来自美国国家标准与技术研究所, National Institute of Standards and Technology (NIST)。训练集 (training set) 由来自 250 个不同人手写的数字构成, 其中 50%是高中学生, 50%来自人口普查局(the Census Bureau)的工作人员。测试集(test set)也是同样比例的手写数字数据。训练库有 60,000 张手写数字图像,测试库有 10,000 张手写数字图像。训练库 train-images.idx3-ubyte，文件大小 47040016,每个图像拥有 784=28×28 个像素,则训练库共有 784×60000=47040000 个像素；测试库 t10k-images.idx3-ubyte，文件大小 7840016，该测试库拥有 784×10000=7840000 个像素。CIFAR-10 和 CIFAR-100 是带有标签的数据集，都出自于规模更大的一个数据集，这个规模更大的数据集有八千万张小图片。CIFAR-10 数据集由 10 个类的 60000 个 32x32 彩色图像组成，每个类有 6000 个图像。CIFAR-10 数据集共有 50000 个训练图像和 10000 个测试图像。CIFAR-10 数据集分为五个训练批次和一个测试批次，每个批次有 10000 个图像。测试批次包含来自每个类别的恰好 1000 个随机选择的图像。训练批次以随机顺序包含剩余图像,但一些训练批次可能包含来自一个类别的图像比另一个更多。总体来说，五个训练集之和包含来自每个类的正好 5000 张图像。CIFAR-10 数据集的 10 个类分别是：飞机、汽车、鸟、猫、鹿、狗等。CIFAR-100 数据集就像 CIFAR-10 一样，除了它有 100 个类，每个类包含 600 个图像。每类各有 500 个训练图像和 100 个测试图像。CIFAR-100 中的 100 个类被分成 20 个超类。每个图像都带有一个“精细”标签（它所属的类）和一个“粗糙”标签（它所属的超类），如：超类-水生哺乳动物；类别-海狸、海豚、水獭、海豹、鲸鱼。

五、《认知科学与类脑计算实验》考核方式

采用上机情况、程序质量、实验报告相结合的形式。

第二部分：《认知科学与类脑计算》实验步骤和实验报告规范

一、《认知科学与类脑计算实验》实验步骤

随着计算机性能的提高，它所面临的软件开发的复杂度也日趋增加，因此软件开发需要系统的方法。一种常用的软件开发方法,是将软件开发过程分为分析、设计、实现和维护四个阶段。虽然认知科学与类脑计算课程中的实验题的复杂度远不如实际中真正的软件系统,但为了培养一个软件工作者所应具备的科学工作的方法和作风，我们制订了如下所述完成实验的 5 个步骤：

1、问题分析和任务定义

通常，实验题目的陈述比较简洁，或者说有模棱两可的含义。因此，在进行设计之前，首先应该充分地分析和理解问题，明确问题要求做什么，限制条件是什么。注意：本步骤强调的是做什么，而不是怎么做，对所需完成的任务要作出明确的回答。这一步还应该为调试程序准备好测试数据。

2、系统设计

在系统设计这一步骤中需根据问题描述划分模块，定义主程序模块和各函数体模块。在编写程序之前，应该先写出各过程和函数的伪码算法。在这个过程中，要综合考虑系统功能，使得系统结构清晰、合理、简单和易于调试。作为逻辑设计的结果，应写出各个主要模块的算法，并画出模块之间的调用关系图。详细设计的结果是对主程序模块和各函数体模块规格说明作出进一步的求精，写出各函数体模块的定义，按照算法书写规范用 Python 语言写出过程或函数形式的算法框架。在求精的过程中，应尽量避免陷入语言细节，不必过早表述函数结构和局部变量。

3、编码实现和静态检查

编码是把详细设计的结果进一步求精为程序设计语言程序。如何编写程序才能较快地完成调试是特别要注意的问题。程序的每行不要超过 60 个字符。每个过程（函数）体一般不要超过 40 行，最长不得超过 60 行，否则应该分割成较小的过程（函数）。要控制 if 语句连续嵌套的深度，分支过多时应考虑使用 switch 语句。对函数功能和重要变量进行注释。一定要按格式书写程序，分清每条语句的层次，对齐括号，这样便于发现语法错误。

在上机之前，应该用笔在纸上写出详细的程序编码，并做认真地静态检查。多数初学者在编好程序后处于以下两种状态之一：一种是对自己的“精心作品”的正确性确信不疑；另一种是认为上机前的任务已经完成，纠查错误是上机的工作。这两种态度是极为有害的。对一般的程序设计者而言，当编写的程序长度超过 50 行时，通常会含有语法错误或逻辑错误。上机动态调试决不能代替静态检查，否则调试效率将是极低的。静态检查主要有两种方法，一是用一组测试数据手工执行程序（通常应先检查单个模块）；二是通过阅读或给别人讲解自己的程序而深入全面地理解程序逻辑，在这个过程中再加入一些注解。

4、上机准备和上机调试

上机准备包括以下几个方面：

（1） 熟悉 Python 语言用户手册或程序设计指导书；

（2） 熟悉机器的操作系统和语言集成环境的用户手册，尤其是最常用的命令操作，以便顺利进行上机的基本活动；

（3） 掌握调试工具，考虑调试方案，设计测试数据并手工得出正确结果。

上机调试程序时要带一本 Python 语言教材或手册。调试最好分模块进行，自底向上，即先调试低层过程或函数。必要时可以另写一个调用驱动程序。这种表面上麻烦的工作实际上可以大大降低调试所面临的复杂性，提高调试工作效率。调试正确后，认真整理源程序及其注释，印出带有完整注释的且格式良好的源程序清单和结果。

5、总结和整理实验报告

二、《认知科学与类脑计算实验》实验报告规范

实习报告的开头应给出题目、班级、姓名、学号和完成日期，并包括以下 7 个内容：需求分析、概要设计、详细设计、调试分析、测试结果、附录等。

第三部分：《认知科学与类脑计算实验》实验内容

实验一 Hopfield 模型的实现

1982 年，J. Hopfield 提出了可用作联想存储器的互连网络，这个网络称为 Hopfield 网络模型，也称 Hopfield 模型。Hopfield 神经网络模型是一种循环神经网络，从输出到输入有反馈连接。Hopfield 网络有离散型和连续型两种。反馈神经网络由于其输出端有反馈到其输入端；所以，Hopfield 网络在输入的激励下，会产生不断的状态变化。当有输入之后，可以求取出 Hopfield 的输出，这个输出反馈到输入从而产生新的输出，这个反馈过程一直进行下去。如果 Hopfield 网络是一个能收敛的稳定网络，则这个反馈与迭代的计算过程所产生的变化越来越小，一旦到达了稳定平衡状态；那么 Hopfield 网络就会输出一个稳定的恒值。对于一个 Hopfield 网络来说，关键是在于确定它在稳定条件下的权系数。

实验目的：加深对 Hopfield 模型的理解，能够使用 Hopfield 模型解决实际问题

实验原理：Hopfield 网络从输出到输入有反馈连接，在输入的激励下，会产生不断的状态变化，是一种单层反馈神经网络，也可以被视为一种循环神经网络。Hopfield 神经网络是反馈网络中最简单且应用广泛的模型，它具有联想记忆的功能，是神经网络发展历史上的一个重要的里程碑。离散 Hopfield 网络可以视为一个类脑模型，主要是因为其可用于联想记忆，即联想存储器，这是类人智能的特点之一。人类的所谓“触景生情”就是见到一些类同于过去接触的景物，容易产生对过去情景的回味和思忆。对于 Hopfield 网络，用它作联想记忆时，首先通过一个学习训练的过程确定网络中的权系数，使所记忆的信息在网络的 n 维超立方体的某一个顶角达到能量最小，当网络的权重矩阵确定之后，只要向网络给出输入向量，这个向量可能是局部数据．即不完全或部分不正确的数据，但是网络仍然能够产生所记忆信息的完整输出。

离散 Hopfield 网络是一个单层网络，有 n 个神经元节点，每个神经元的输出均接到其它神经元的输入。各节点没有自反馈。每个节点都可处于一种可能的状态（1 或－1），即当该神经元所受的刺激超过其阀值时，神经元就处于一种状态（比如 1），否则神经元就始终处于另一状态（比如 －1）。

实验内容：根据 Hopfield 神经网络的相关知识，设计一个具有联想记忆功能的离散型 Hopfiled 神经网络。要求该网络可以正确识别 0-9 这 10 个数字，当数字被一定的噪声干扰后，仍具有较好的识别效果。

实验要求：

（1） 设计 6*5 数字点阵。有数字部分用 1 表示，空白部分用-1 表示，将数字 0-9 的矩阵设计好存储到 1*30 矩阵中。

（2） 创建网络。

（3） 产生带噪声的数字点阵。带噪声的数字点阵，即点阵的某些位置的值发生了变化。模拟产生带噪声的数字矩阵方法有很多种，如固定噪声产生法和随机噪声产生法。

（4） 数字识别测试。将带噪声的数字点阵输入到创建好 Hopfiled 网络，网络的输出是与该数字点阵最为接近的目标向量，从而实现联想记忆功能。

实验二 感知器模型实现并进行分类

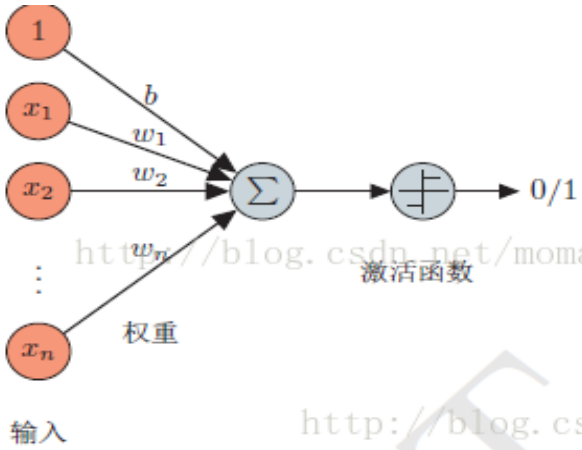
感知器，也可翻译为感知机，是 Frank Rosenblatt 在 1957 年就职于 Cornell 航空实验室(Cornell Aeronautical Laboratory)时所发明的一种人工神经网络。它可以被视为一种最简单形式的前馈式人工神经网络，是一种二元线性分类器。感知器是生物神经细胞的简单抽象，神经细胞结构大致可分为：树突、突触、细胞体及轴突。单个神经细胞可被视为一种只有两种状态的机器——激动时为‘是’，而未激动时为‘否’。神经细胞的状态取决于从其它的神经细胞收到的输入信号量，及突触的强度（抑制或加强）。当信号量总和超过了某个阈值时，细胞体就会激动，产生电脉冲。电脉冲沿着轴突并通过突触传递到其它神经元。为了模拟神经细胞行为，与之对应的感知机基础概念被提出，如权量（突触）、偏置（阈值）及激活函数（细胞体）。

实验目的：加深对感知器模型的理解，能够使用感知器模型解决简单的分类问题

实验原理：参考课本对感知器模型的讲解

实验内容：根据感知器的相关知识，使用 Python 语言实现一个简单的感知器模型，该模型能够实现简单的二分类任务（与或非运算）。

感知器模型如下所示：



权重更新规则如下所示：

$$w_i \leftarrow w_i + \Delta w_i$$
$$b \leftarrow b + \Delta b$$

$$\Delta w_i = \eta(t-y)x_i$$
$$\Delta b = \eta(t-y)$$

其中，t 为 label，y 是预测，t-y=偏差，η 为学习率，w 为权重，b 为偏置，本次实验需要使用到的包为 numpy，可以使用 pip 安装。

实验要求：

（1） 基于与或非真值表构建训练数据，将构建好的训练数据存储到列表中。

（2） 创建网络。

（3） 二分类任务测试。将测试数据输入到创建并训练好的网络中，网络的输出是与该测试数据与或非运算最为接近的数据。

实验三 Hebb 学习

由赫布提出的 Hebb 学习规则是一个无监督学习规则，这种学习的结果是使网络能够提取训练集的统计特性，从而把输入信息按照它们的相似性程度划分为若干类。这一点与人类观察和认识世界的过程非常吻合，人类观察和认识世界在相当程度上就是在根据事物的统计特征进行分类。Hebb 学习规则只根据神经元连接间的激活水平改变权值，因此这种方法又称为相关学习或并联学习。

巴普洛夫实验是一个典型的 Hebb 学习模型，其具体内容是每次给狗送食物以前响起铃声，这样经过一段时间以后，铃声一响，狗就开始分泌唾液。

实验目的：加深对 Hebb 学习模型的理解，能够使用 Hebb 学习模型解决简单问题

实验原理：Hebb 算法最简单可以描述为:如果一个处理单元从另一处理单元接收输入激励信号，而且如果两者都处于高激励电平，那么处理单元之间的加权就应当增强。用数学来表示，就是两节点的连接权将根据两节点的激励电平的乘积来改变，即

$$\Delta w_{ij}=w_{ij}(n+1)-w_{ij}(n)=\eta y_i x_j$$

其中 $w_{ij}(n)$ 表示第(n+1)次调解前，从节点 j 到节点 i 的连接权值；$w_{ij}(n+1)$ 是第(n+1)次调解后，从节点 j 到节点 i 的连接权值；η 为学习速率参考；x_j 为节点 j 的输出，并输入到节点 i；y_i 为节点 i 的输出。

对于 Hebb 学习规则，学习信号简单地等于神经元的输出

$$r = f(W_i^T X)$$

权向量的增量变成

$$\Delta W_i = \eta f(W_i^T X) X$$

这个学习规则在学习之前要求在 $W_i = 0$ 附近的小随机值上对权重进行初始化。这个规则说明了如果输出和输入的叉积是正的，则权增加，否则减小。

实验内容：根据 Hebb 学习模型的相关知识，使用 Python 语言实现一个简单 Hebb 学习模型。

实验要求：

（1） 设计 6*5 数字点阵。有数字部分用 1 表示，空白部分用-1 表示，将数字 0-2 的矩阵设计好存储到列表中。

（2） 创建网络。

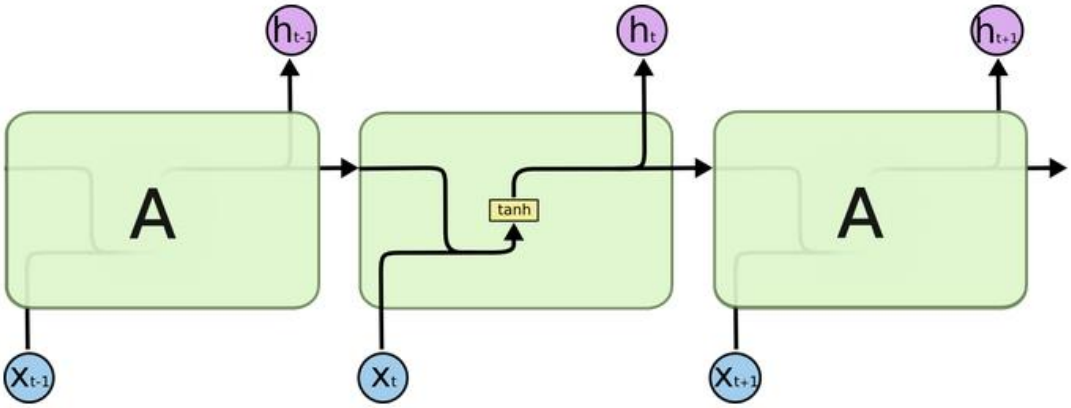
（3） 数字识别测试。将训练数据加入噪声作为测试数据，输入到创建并训练好的网络中，网络的输出是与该数字点阵最为接近的目标向量。

实验四 LSTM 的实现

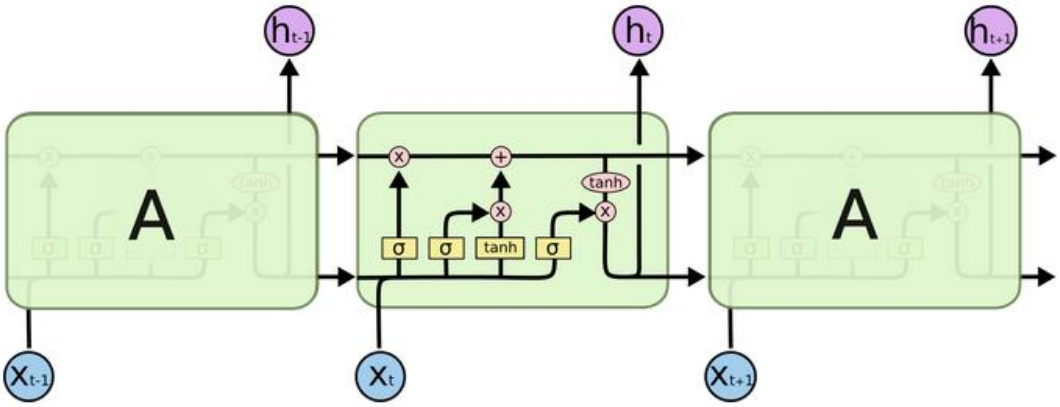
LSTM（Long Short-Term Memory）是长短期记忆网络，是一种时间递归神经网络，适合于处理和预测时间序列中间隔和延迟相对较长的重要事件。LSTM 区别于 RNN 的地方，主要就在于它在算法中加入了一个判断信息有用与否的“处理器”，这个处理器作用的结构被称为 cell。一个 cell 当中被放置了三扇门，分别叫做输入门、遗忘门和输出门。一个信息进入 LSTM 的网络当中，可以根据规则来判断是否有用。只有符合算法认证的信息才会留下，不符的信息则通过遗忘门被遗忘。说起来无非就是一进二出的工作原理，却可以在反复运算下解决神经网络中长期存在的大问题。目前已经证明，LSTM 是解决长序依赖问题的有效技术，并且这种技术的普适性非常高，导致带来的可能性变化非常多。各研究者根据LSTM纷纷提出了自己的变量版本,这就让LSTM可以处理千变万化的垂直问题。

实验目的：加深对 LSTM 模型的理解，能够使用 LSTM 模型解决简单问题

实验原理：LSTM 是一类可以处理长期依赖问题的特殊的 RNN，LSTM 主要用来处理长期依赖问题，与传统 RNN 相比，长时间的信息记忆能力是与生俱来的。所有的 RNN 链式结构中都有不断重复的模块，用来随时间传递信息。传统的 RNN 使用十分简单的结构，如下图所示。



LSTM 链式结构中重复模块的结构更加复杂，有四个互相交互的层（如下图所示）。

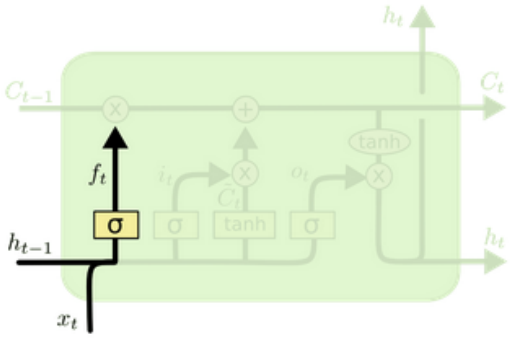


与传统 RNN 相比，除了拥有隐藏状态外，LSTM 还增加了一个细胞状态，记录

随时间传递的信息。在传递过程中，通过当前输入、上一时刻隐藏层状态、上一时刻细胞状态以及门结构来增加或删除细胞状态中的信息。门结构用来控制增加或删除信息的程度，一般由 sigmoid 函数和向量点乘来实现。

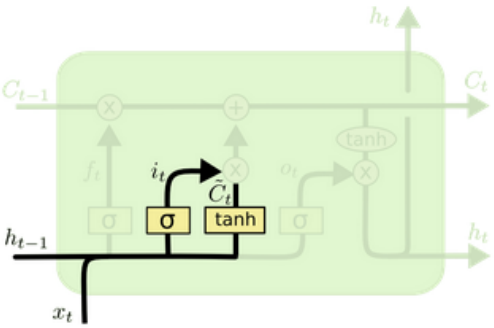
LSTM 共包含 3 个门结构，来控制细胞状态和隐藏状态，下边分别进行介绍。

遗忘门。遗忘门决定上一时刻细胞状态中的多少信息可以传递到当前时刻中。



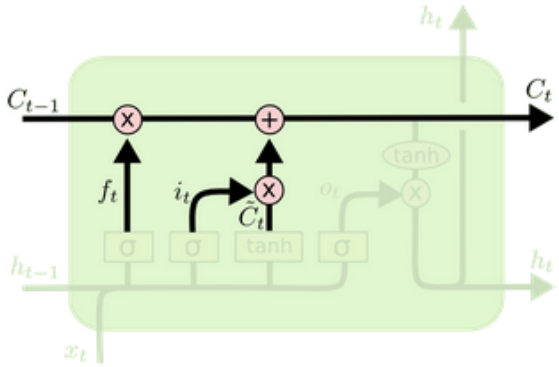
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

输入门。输入门用来控制当前输入新生成的信息中有多少信息可以加入到细胞状态中。C_t层用来产生当前时刻新的信息，i_t层用来控制有多少新信息可以传递给细胞状态。



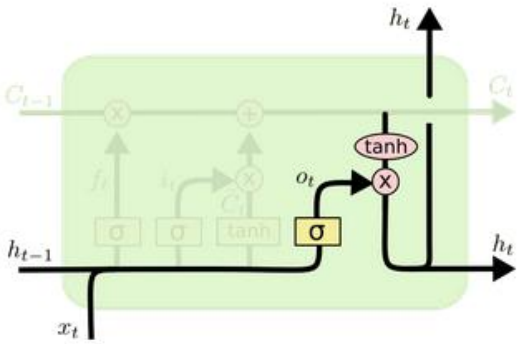
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

更新细胞状态。基于遗忘门和输入门的输出，来更新细胞状态。更新后的细胞状态有两部分构成:一，来自上一时刻旧的细胞状态信息；二，当前输入新生成的信息。



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

输出门，基于更新的细胞状态，输出隐藏状态。



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

实验内容：根据 LSTM 模型的相关知识，使用 Python 语言实现一个简单 LSTM 模型。

实验要求：

（1） 随机产生 0-127 之间的两个八位的二进制整数，作为一组输入数据，将这两个数的和作为一个标签，这三个数据组成一组训练数据，训练数据的组数应尽可能多。

（2） 创建 LSTM 网络。

（3） 实现两个八位的二进制整数的加法运算，网络能够输出正确的加法运算结果。

实验五 HMAX 模型实现

Poggio 以及 Serre 等人于 2007 年提出了一个仿脑、基于特征组合的对象特征提取模型-HMAX 模型，该模型的理论基础是生物学中对视皮层神经细胞进行对象识别机制的研究。HMAX 模型通过 Gabor 滤波、求最大值操作以及交替进行模板匹配模拟了人眼视皮层中神经细胞进行对象识别处理过程。HMAX 模型在模式识别领域主要作用于识别对象的特征提取，所提取的特征称为 HMAX 特征。HMAX 模型是一个层次式的结构，分为 5 层：S1 层、C1 层、S2 层、C2 层、VTU 层。S1 层、C1 层、S2 层、C2 层分别对应视皮层中的简单细胞和复杂细胞，而 VTU 层对应于识别细胞。

实验目的：加深对 HMAX 模型的理解，能够使用 HMAX 模型解决简单问题

实验原理：HMAX 模型是一个层次式的结构，分为 5 层：S1 层、C1 层、S2 层、C2 层、VTU 层。

S1 层。该层事实上是用 4 个方向及 16 个尺度的 Gabor 滤波器组对输入图像进行滤波，得到 64 个响应图。在尺度上，HMAX 算法分的相当细致，它将 16 个尺度分成 8 个子带，每个子带包含两个相邻尺度，以便在之后的 C1 层进行整合。Gabor 函数滤波如下所示：

$$F(x,y) = exp(-\frac{(x_0^2 + \gamma^2 y_0^2)}{2\sigma^2}) \times cos(\frac{2\pi}{\lambda} x_0)$$
$$x_0 = xcos\theta + ysin\theta$$
$$y_0 = -xsin\theta + ycos\theta$$

其中，σ是高斯函数的有效宽度，θ是方向，λ是波长。根据 V1 简单细胞来对参数

进行调整和确定。S1 响应形成 8 个波段，每个波段有 4 个方向和 2 个级别。每个波段有两个滤波器，一共有 64 个不同的 S1 响应，有四个方向（$\theta = 0^{\circ}, 45^{\circ}, 90^{\circ}, 135^{\circ}$）。

C1 层。如前所述，64 个响应图依子带编号分成 8 组，每组包括 S1 层得到的 2 个相邻尺度及所属每个尺度上所有的 4 个方向响应。C1 层的操作是依组依方向进行的。具体做法是先将任一个滤波器响应图划分成 8*8 的格子，在每个格子中求响应的最大值。这样对每个响应图都能得到一张采样过的最大值图；然后，对组内的两个相邻尺度的最大值图的对应像素再求一次最大值，以最终得到具有不变性质的响应。值得注意的是，以上是 8x8 区域没有重叠的情况，在这种情况下，减采样倍数为 8；实际上经常采用 8x8 区域相互重叠一半的情况以进一步增加不变性，在这种情况下减采样倍数为 4，数据经过 C1 层之后，我们在每组中得到的是 4 个方向的不变响应。注意最大值不能跨方向选取。对于物体识别，可以在 C1 层生成的不变响应中抽取每类特征，并加以保存作为训练结果。其具体做法为：随机找到一个子带，在 4 方向不变响应图中抽取一块 m*m*4 的立方体，可以称之为 patch。

S2 层。在该层中，再次使用滤波器对 C1 层的输出进行滤波，产生新一轮的响应图。只是这次滤波器变成了之前随机抽取的那些，而非第一层硬性规定的 Gabor 滤波器。滤波的方式也有所不同，不再是卷积运算，而是直接以 L2 距离作比较，再用高斯核映射成相似度。注意到随机抽取的 patch 是 n*n*4 的，因此做完滤波后，对每一个 patch 我们能得到 8 个组各一张图的响应。假设 patch 数目是 K，则共有 8*K 张响应作为 S2 的输出。

C2 层。对每一个 patch 遍历 8 个组的响应，找到最大的那个值。如此，对每张输入图片，最后得到一个 K 维向量。

VTU 层。该层可以使用 SVM 及其他方法，将 C2 层输出的 K 维向量放入该层进行训练。

实验内容：根据 HMAX 模型的相关知识，使用 Python 语言实现一个简单的 HMAX 模型。

实验要求：

（1） 下载 MNIST 数据集。
（2） 构建 HMAX 模型。
（3） 使用 MNIST 数据集中的训练集训练网络，使用测试集测试训练好的网络。

实验六 TopoICA 模型实现

拓扑 ICA（拓扑结构的独立成分分析）是一个生成模型，他将拓扑映射和 ICA 结合在一起。在所有的拓扑映射中，表征空间中的距离（在拓扑‘网格’）是相似于表征的成分的距离。在拓扑 ICA 中，表征成分的距离通过高阶相关性的互信息定于得到的，这给出了 ICA 上下文中的自然距离测量。在自然图像数据上的应用，拓扑 ICA 给出了类 Gabor 线性特征中的线性分解。与普通的 ICA 相比，这个高阶依赖性显示线性 ICA 不能移除定义一个拓扑顺序使得近邻的细胞倾向于在同一时间激活。这同样暗示这邻居有着与复杂细胞一样的特性。这个方法因此可

以展现复杂细胞特性和拓扑组织的同时发生。这两个特性通过同时邻居的激活值定义的拓扑的原则而得到的。

拓扑 ICA 对经典 ICA 模型进行了一个修正，使得成分之间的依赖性被显式的表达。还要特别提出，独立成分的残余依赖性结构，也就是 ICA 无法消除的依赖性，能够用于定义成分之间的一种拓扑序。这种拓扑序很容易用可视化方式表示，因其与脑建模之间的联系，拓扑序随图像特征提取具有重要意义。

实验目的：加深对 TopoICA 模型的理解，能够使用 TopoICA 模型解决简单问题

实验原理：假定一个静态灰度化图像 I(x,y)，他是由许多特征或基函数 a_i(x,y)的线性组合得到的：

$$I(x,y)=\sum_{i=1}^{m}a_i(x,y)s_i$$

s_i是随机系数，在每个图像 I（x,y）中都是不同的。这里关键的假设是 si 是非高斯的，而且是互相独立的。这样的分解就叫做独立成分分析（ICA），或者换个观点来说，就是稀疏编码。

在给定足够数量的图像或者说图像碎片 I（x,y）的观测值的情况下，估算上述等式中的模型需要对于所有的 i 和（x,y）的情况下决定 s_i，a_i（x，y）的值。我们假定基础的情况下，a_i(x,y)来自于一个可逆的线性系统。然后我们就能对这个系统求逆：

$$s_i=<w_i,I>$$

$$<w_i,I>=\sum_{x,y}w_i(x,y)I(x,y)$$

w_i表示逆过滤器，可以认为是模型简单的细胞的感受野，而且 s_i是当输入是 I(x，y）的激活值。

在经典的 ICA 中，独立成分 s_i并没有特别的顺序，或者其他的关系。独立成分内部固有顺序或关系的缺失，与我们假定他们是完全的统计独立是有关系的。当在自然图像统计上使用 ICA 进行建模时，会清晰的发现在很多地方是违反独立假设的前提的，可以很容易发现所估计的独立成分之间是有着清晰的依赖（非独立）关系的。在这部分中，我们通过使用一个生成模型来定义拓扑 ICA，这个模型是普通 ICA 模型的分层的版本。这个想法是放宽 I（x,y）等式中成分的独立性假设，这样就能使得拓扑中相互靠近的成分不再假定具有独立性。例如，拓扑是由点阵或网格定义的，成分的依赖性就是网格上成分的距离函数。相反的，在拓扑中不相互靠近的成分还是独立的，至少近似独立；因此大多数成分对还是独立的。

近邻成分之间应该选择高阶的依赖性，即能量相关：

$$\mathrm{cov}\,(s_i^2,s_j^2)=E\{s_i^2s_j^2\}-E\{s_i^2\}E\{s_j^2\}\neq 0$$

这里假定 s_i和 s_j是拓扑中的近邻，而且协方差是正的。直观上，这样一个相关性意味着成分倾向于在同一时刻是活动的，即非零，但是 s_i和 s_j的真实值却不是那么容易预测的。即有可能 s_i和 s_j是不相关的，但是他们的能量却不是。

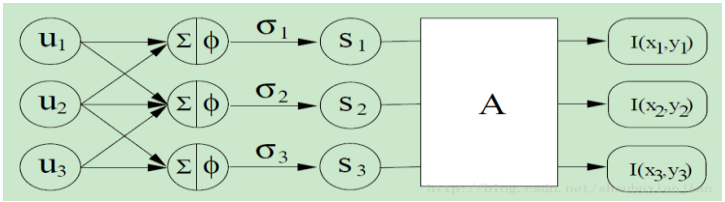
定义一个生成模型，这个模型能够表示拓扑网格中邻近成分的能量相关性。在这个模型中，观测到的图像碎片作为成分 s_i 的线性转换而生成，就像 I（x,y）等式中的基础 ICA 模型一样。关键的点是如何定义一个 s 的联合密度去表示这个拓扑。 我们通过以下的方式来定义 s 的联合密度。s_i 的方差 sigma（i）^2 不是常量，他们被假定成是随机变量，有具体的模型生成。在生成方差后，变量 s_i 是由具体的条件分布来相互之间独立生成的，换句话来说，s_i 是在给定方差下独立的。s_i 之间的依赖性隐含在他们的方差的依赖性中。根据拓扑的原则，近邻成分的相对应的方差应该是(正)相关的，而且相互不靠近的成分的方差是独立的，至少是近似的。

为了具体说明模型的方差 sigma（i）^2，我们首先需要定义拓扑。这可以通过函数 h（i,j）来完成，这个函数能够表示第 i 个和第 j 个成分的相关程度。这个函数能够以自组织映射的同样的方法来定义。所以 h（i，j）是一个超参数矩阵。在本文中，我们认为他是未知的，但是是固定的。

为了使用拓扑相关性 h(i,j)，需要许多不同模型的方差 sigma(i)^2，我们通过一个非线性 ICA 模型来定义他们：

$$\sigma_i = \phi(\sum_{k=1}^{n} h(i,k)u_k)$$

这里 u_k 是高阶独立成分，可以用作去生成方差，而且Φ是标量非线性的。u_k 的分布和Φ的实际形式是模型的附加超参数。限制 u_k 为非负的。这个函数Φ可以被认为是在非负实数集中的一个单调转换。这确保了 sigma（i）是非负的。生成的拓扑 ICA 模型如下所示：



为了估计这个模型，我们采用最大释然估计。为了简化估计，我们通过一个易处理的似然的逼近。为了获得这个逼近，我们首先做出下面的假设。首先，我们固定成分 s_i 的条件密度是高斯的。第二，我们定义非线性Φ是

$$\phi(y) = y^{-1/2}$$

第三，我们简单的假定 s_i 的条件密度对于所有的 i 来说都是相同的。模型的似然度函数定义如下：

$$\log \tilde{L}(w_i, i = 1, ..., n) = \sum_{t=1}^{T}\sum_{j=1}^{n} G(\sum_{i=1}^{n} h(i,j) < w_i, I_t >^2) + T \log|\det \mathbf{W}|.$$

$$G(y) = \log \int \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}uy) p_u(u) \sqrt{h(i,i)u}\, du,$$

简单的得到一个梯度算法，第 i 个向量 w_i 的更新规则如下：

$$\Delta w_i(x,y) \propto E\{I(x,y) < w_i, I > r_i)\}$$

$$r_i = \sum_{k=1}^{n} h(i,k) g(\sum_{j=1}^{n} h(k,j) < w_j, I >^2).$$

实验内容：根据 TopoICA 模型的相关知识，使用 Python 语言实现一个简单的 TopoICA 模型。

实验要求：

（1） 下载 MNIST 数据集。
（2） 构建 TopoICA 模型。
（3） 使用 MNIST 数据集中的训练集训练网络，使用测试集测试训练好的网络。

实验七 CNN 做手写体识别

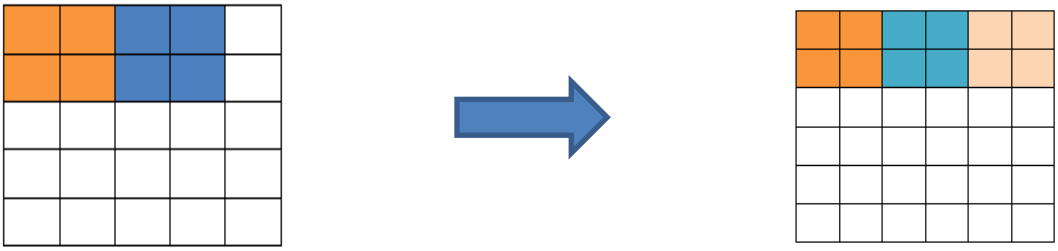
卷积神经网络（Convolutional Neural Networks, CNN）是一类包含卷积计算且具有深度结构的前馈神经网络，是深度学习的代表算法之一。由于卷积神经网络能够进行平移不变分类，因此也被称为“平移不变人工神经网络”。对卷积神经网络的研究始于二十世纪 80 至 90 年代，时间延迟网络和 LeNet-5 是最早出现的卷积神经网络；在二十一世纪后，随着深度学习理论的提出和数值计算设备的改进，卷积神经网络得到了快速发展，并被大量应用于计算机视觉、自然语言处理等领域。卷积神经网络仿造生物的视知觉机制构建，可以进行监督学习和非监督学习，其隐含层内的卷积核参数共享和层间连接的稀疏性使得卷积神经网络能够以较小的计算量对格点化特征，例如像素和音频进行学习、有稳定的效果且对数据没有额外的特征工程要求。

实验目的：加深对卷积神经网络模型的理解，能够使用卷积神经网络模型解决简单问题

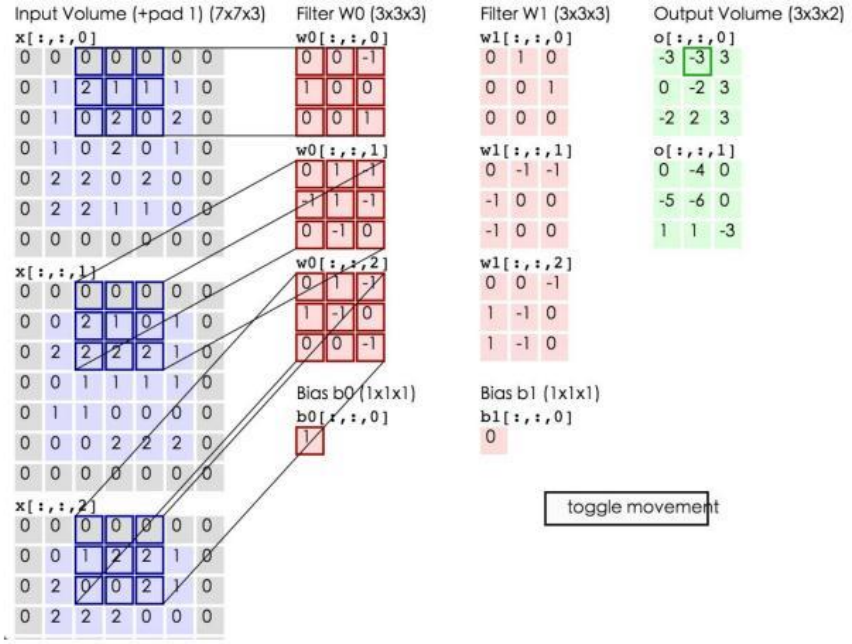
实验原理：卷积神经网络的层级结构一般为：数据输入层、卷积计算层、激励层（最常用的为 ReLU 激活函数）、池化层（平均池化和最大池化）、全连接层（不必须）、输出层。

数据输入层。该层要做的处理是对原始图像数据进行预处理，其中主要包括归一化操作，即将图像数据处理成均值为 0，方差为 1 的数据等。

卷积计算层。在卷积层，有两个关键操作：局部关联，每个神经元看做一个滤波器；窗口滑动，滤波器对局部数据进行计算。先介绍卷积层遇到的几个名词：步长(stride)，即窗口一次滑动的长度；填充值(padding)，一般为 0 值填充。以下图为例子，比如有一个 5*5 的图片（一个格子一个像素），我们滑动窗口取 2*2，步长取 2，那么我们发现还剩下 1 个像素没法滑完，就可以在原先的矩阵边缘加一层填充值，使其变成 6*6 的矩阵，那么窗口就可以刚好把所有像素遍历完。这就是填充值的作用。



卷积的计算。下图的蓝色矩阵就是输入的图像，粉色矩阵就是卷积层的神经元，这里表示了有两个神经元（w_0,w_1）。绿色矩阵就是经过卷积运算后的输出矩阵，这里的步长设置为 2。



激励层。把卷积层输出结果做非线性映射，最常用的为 ReLU 激活函数。

池化层夹在连续的卷积层中间，用于压缩数据和参数的量，减小过拟合。简而言之，如果输入是图像的话，那么池化层的最主要作用就是压缩图像。

实验内容：根据卷积神经网络模型的相关知识，使用 Python 语言实现一个简单卷积神经网络模型，该模型能够实现手写数字识别。

实验要求：

（1） 下载 MNIST 数据集。

（2） 构建卷积神经网络。

（3） 使用 MNIST 数据集中的训练集训练网络，使用测试集测试训练好的网络，测试准确率应达到 90%以上。

实验八 BP 算法分析与实现

BP 算法是由学习过程中信号的正向传播与误差的反向传播两个过程组成。由于多层前馈网络的训练经常采用误差反向传播算法，人们也常把将多层前馈网络直接称为 BP 网络。BP 算法由信号的正向传播和误差的反向传播两个过程组成。

BP 算法的基本思想是，学习过程由信号的正向传播与误差的反向传播两个过程组成。正向传播时，输入样本从输入层传入，经各隐层逐层处理后，传向输出层。若输出层的实际输出与期望的输出(教师信号)不符，则转入误差的反向传播阶段。误差反传是将输出误差以某种形式通过隐层向输入层逐层反传，并将误差分摊给各层的所有单元，从而获得各层单元的误差信号，此误差信号即作为修正各单元权值的依据。这种信号正向传播与误差反向传播的各层权值调整过程，是周而复始地进行的。权值不断调整的过程，也就是网络的学习训练过程。此过程一直进行到网络输出的误差减少到可接受的程度，或进行到预先设定的学习次数为止。

实验目的：加深对 BP 算法的理解，能够使用 BP 算法解决简单问题

实验原理：BP 算法的学习过程由正向传播过程和反向传播过程组成。在正向传播过程中，输入信息通过输入层经隐含层，逐层处理并传向输出层。如果在输出层得不到期望的输出值，则取输出值与期望值的误差的平方和作为目标函数，转入反向传播，逐层求出目标函数对各神经元权值的偏导数，构成目标函数对权值向量的梯度，作为修改权值的依据，网络的学习在权值修改过程中完成。误差达到所期望值时，网络学习结束。

反向传播算法主要由激励传播和权重更新这两个环节组成，通过反复循环迭代，直到网络对输入的响应达到预定的目标范围为止。

激励传播。每次迭代中的传播环节包含两步：前向传播阶段，将训练输入送入网络以获得激励响应；反向传播阶段，将激励响应同训练输入对应的目标输出求差，从而获得输出层的响应误差。

权重更新。对于每个突触上的权重，按照以下步骤进行更新：将输入激励和响应误差相乘，从而获得权重的梯度；将这个梯度乘上一个比例并取反后加到权重上。这个比例将会影响到训练过程的速度和效果，因此称为学习率。梯度的方向指明了误差扩大的方向，因此在更新权重的时候需要对其取反，从而减小权重引起的误差。

实验内容：根据 BP 算法的相关知识，使用 Python 语言实现一个简单 BP 算法模型。

实验要求：

（1） 输入数据[[0.35], [0.9]]，真实值[[0.5]]，输出数据与真实值的误差使用平方和误差表示。

（2） 创建人工神经网络。

（3） 参数更新值 Δw 用链式求导法则求出，最后使得输出数据与真实值的误差小于 0.01。

实验九 生成对抗网络实现

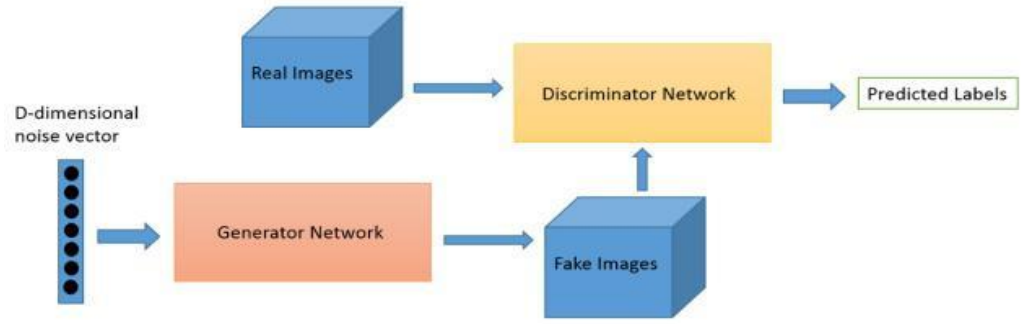
生成式对抗网络（GAN, Generative Adversarial Networks）是一种深度学习模型，是近年来复杂分布上无监督学习最具前景的方法之一。模型通过框架中（至少）两个模块：生成模型（Generative Model）和判别模型（Discriminative Model）的互相博弈学习产生相当好的输出。原始 GAN 理论中，并不要求 G 和 D

都是神经网络，只需要是能拟合相应生成和判别的函数即可。但实用中一般均使用深度神经网络作为 G 和 D 。一个优秀的 GAN 应用需要有良好的训练方法，否则可能由于神经网络模型的自由性而导致输出不理想。

实验目的：加深对生成对抗网络的理解，能够使用生成对抗网络解决简单问题

实验原理：在生成对抗网络中，一共建立了两个神经网络。第一个网络是典型的分类神经网络，称为判别器，我们训练这个网络对图像进行识别，以区别真假的图像。另一个网络称之为生成器，它将随机的噪声作为输入，输出经过生成器生成的图像，它的目的是混淆判别器，使其认为它生成的图像是真的。真的图片在训练集当中，而假的则不在。在这个设置中，两个网络参与了一场竞争游戏，并试图超越对方，同时，帮助对方完成自己的任务。经过数千次迭代后，如果一切顺利，生成器网络在生成逼真的假图像方面变得完美，而判别器网络在判断显示给它的图像是假的还是真的方面变得完美。

最基本的生成对抗网络为 vanilla GAN 网络。首先，从潜在空间采样 D 维噪声向量并馈送到生成器网络。生成器网络将该噪声向量转换为图像。然后将该生成的图像馈送到判别器网络以进行分类。判别器网络不断地从真实数据集和由生成器网络生成的图像获得图像。它的工作是区分真实和虚假的图像。所有 GAN 架构都遵循相同的设计。这是 GAN 的诞生。



原始论文中的目标公式如下：

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[log(D(x))] + E_{z\sim p_z(z)}[log(1-D(G(z)))]$$

上述这个公式就是一个最大最小优化问题，对应的也就是上述的两个优化过程。这个公式既然是最大最小的优化，那就不是一步完成的，其实对比我们的分析过程也是这样的，这里我们先优化 D，然后再去优化 G，本质上是两个优化问题，把目标公式拆解，就如同下面两个公式：

优化 D：

$$\max_D V(D,G) = E_{x\sim p_{data}(x)}[log(D(x))] + E_{z\sim p_z(z)}[log(1-D(G(z)))]$$

优化 G：

$$\min_G V(D,G) = E_{z\sim p_z(z)}[log(1-D(G(z)))]$$

可以看到，优化 D，也就是判别网络时，生成网络并没有参与，后面的 G(z) 相当于已经得到的假样本。优化 D 的公式的第一项，使的真样本 x 输入的时候，得到的结果越大越好，可以理解为需要真样本的预测结果越接近于 1 越好。对于

假样本，需要优化是的其结果越小越好，也就是D(G(z))越小越好，因为它的标签为0。但是优化D的公式是取最大值，所以把第二项改成1-D(G(z))，这样就变成了取最大值，两者合起来也就是取最大值。同样在优化G的时候，不需要真样本，所以把第一项直接去掉了。这个时候只有假样本，但是这时候是希望假样本的标签是1的,所以是D(G(z))越大越好,但是为了统一成1-D(G(z))的形式，那么只能是最小化1-D(G(z))，本质上没有区别，只是为了形式的统一。之后这两个优化模型可以合并起来写，就变成了最开始的那个最大最小目标函数了。

实验内容：根据生成对抗网络的相关知识，使用Python语言实现一个简单的生成对抗网络。

实验要求：

（1） 下载MNIST数据集。

（2） 创建生成对抗网络。

（3） 使用MNIST数据集训练生成对抗网络，使网络生成MNIST假数据。

大作业实验

COCO数据集的MASK RCNN模型实现

微软发布的 COCO 数据库是一个大型图像数据集, 专为对象检测、分割、人体关键点检测、语义分割和字幕生成而设计。COCO API 提供了 Matlab, Python 和 Lua 的 API 接口. 该 API 接口可以提供完整的图像标签数据的加载, parsing 和可视化。此外,网站还提供了数据相关的文章, 教程等。在使用 COCO 数据库提供的 API 和 demo 之前, 需要首先下载 COCO 的图像和标签数据（类别标志、类别数量区分、像素级的分割等 ）：图像数据下载到 coco/images/ 文件夹中，标签数据下载到 coco/annotations/ 文件夹中。

COCO 数据库的网址是:

MS COCO 数据集主页：http://mscoco.org/

Github 网址：https://github.com/Xinering/cocoapi

关于 API 更多的细节在网站: http://mscoco.org/dataset/#download

MASK RCNN是何凯明基于以往的Faster-RCNN架构提出的新的卷积网络，一举完成了对象实例分割。该方法在有效地目标的同时完成了高质量的语义分割。MASK RCNN主要思路就是把原有的Faster-RCNN进行扩展，添加一个分支使用现有的检测对目标进行并行预测。同时,MASK RCNN网络结构比较容易实现和训练，速度5fps也比较快，可以很方便的应用到其他的领域，像目标检测，分割，和人物关键点检测等。并且比着现有的算法效果都要好。

实验目的：加深对MASK RCNN模型的理解，能够使用MASK RCNN模型解决简单问题

实验原理：参考课本对MASK RCNN模型的讲解

实验内容：根据 MASK RCNN 模型的相关知识，使用 Python 语言实现一个简单的 MASK RCNN 模型。

实验要求：

（1） 下载 COCO 数据集。
（2） 创建 MASK RCNN 模型网络。
（3） 使用 COCO 数据集训练并测试 MASK RCNN 模型网络。

BraTS 数据库的图像分割与生存周期预测

Brats2017 数据集主要包含三部分：HGG 和 LGG 以及生存期表单。HGG 表示高级别胶质瘤，共有 210 个病人案例；LGG 表示低级别胶质瘤，共有 75 个病人案例；生存期表单记录了 163 个病人的基本信息以及生存期。HGG 和 LGG 中，每个病人的文件夹下面包含有四个模态的 MR 数据以及一个 Seg Lable 数据，四个模态分别是 FLAIR, T1, T2, T1C，FLAIR 模态能够反映整个肿瘤结构，T2 能够反映肿瘤核结构，T1C 能够表示增强型肿瘤结构。Seg Lable 数据中只有 0,1,2,4 四个数值，分别表示背景、whole tumor、tumor core、enhance tumor，Brats 数据集的图像分割任务就是分辨出上述四个部分。

实验目的：加深对课程学习中模型的理解，能够使用学过的模型解决 Brats 数据集图像分割任务以及生存期预测问题

实验原理：参考课本中对各个模型的讲解

实验内容：根据学过的模型，使用 Python 语言解决 Brats 数据集图像分割任务以及生存期预测问题

实验要求：

（1） 下载 Brats2017 数据集。
（2） 创建网络。
（3） 使用 Brats2017 数据集训练并测试网络，解决 Brats 数据集图像分割任务以及生存期预测问题。