03.12-03.14
幸运的刚发布题目就甲流了，昏昏沉沉晕了三天

03.15
从 0 开始，没有 anaconda 等等，和这些应用搏斗
从头开始理解概念
深度学习，也就是给出输入数据和输出数据，通过函数的变化拟合，找出最合适的函数
DNN（基础版），RNN，CNN
DNN 信息在网络中单向传递，没有循环连接。RNN 具有循环连接，允许信息在网络中进行持续传递。每个时间步都有输入和输出。
影响因素：隐藏层，迭代次数，批量大小，学习率
去 CSDN 找相关结构
损失函数 数据差距，梯度下降法，反方向
张量类似矩阵，图像数据三个维度，需要激活函数
复习了一下 numpy，跟 pytorch 好像

03.16
开始编写前两题，
完成神经网络的搭建，运行和测试函数的编写，但是具体数据没有确定
第三题没有思路，再去学习注意力机制，
正确对我定义的张量分层后用 attention 输出，计算注意力权重
卡在线性变换了，self 有点晕

03.17
开始写第三题，一直在报错，试着去生成随机的数据，但是不知道什么原因一直在报错，
显示我数据形式不匹配，好怪，没时间了，先用一组数据，
开始修正调试前面两题的代码，
中间评估模式结束忘了 model 训练模式了，卡了好久
因为缩进问题卡了好久，因为循环太多内层数据没有包进去
把数据拉到 GPU 上运行，但是没装 cuda
隐藏层数感觉也是跟神经元个数有相似的地方，试了一下 5 层的，数据全完美，过过过拟合。

03.18
第一题学习率不好控制，加上学习率衰减的代码
accuracy 一直是 1.000，设置一部分数据去进行检验避免过拟合
但是整个题代码大修了，
了解 Batch 和 Dropout
混淆矩阵多个分类矩阵相乘，归一化缩减，重塑，线性变换，再变回来
但是最后代码没保存 www，几个小时努力寄了

03.19
起来先把昨天代码补了
增添了最后的画图（一直以为可视化有表就行了）
用 github 上传
优化代码，改掉冗余部分，精简一下格式，一样的放到一个模块（一下午）
整理这个最后的文档，OK

我们有一个损失函数（loss function）来衡量模型的预测与实际数据之间的差异时，我们希望通过调整模型的参数来最小化这个损失函数。梯度下降法就是一种通过计算损失函数关于模型参数的梯度，并按照梯度的反方向更新参数的方法，从而使损失函数逐渐减小。

在深度学习中，张量是一个核心的概念，它可以被理解为数字的容器，是矩阵向任意维度的推广。张量的维度（dimension）叫做轴（axis）也叫做阶（rank）。图像数据通常具有三个维度：高度、宽度和颜色深度。即使灰度图像（比如 MNIST 数字图像）只有一个颜色通道，通常也可以保存在 2D 张量中（即高度和宽度）。然而，按照惯例，图像张量始终都是 3D 张量，其中灰度图像的彩色通道只有一维。

张量的形状有两种约定：通道在后（channels-last）的约定和通道在前（channels-first）的约定。在 TensorFlow 中，通常使用通道在后（channels-last）的约定，即张量的形状为（samples, height, width, color_depth）。这意味着颜色深度轴（或称为通道轴）被放在最后。与此相反，在 Theano 中，通道轴被放在批量轴之后，即（samples, color_depth, height, width）。

在深度学习中处理图像数据时，理解这些张量约定和形状是非常重要的，因为它们直接影响如何组织和处理图像数据。无论是进行前向传播、反向传播还是其他计算，都需要按照正确的张量形状和约定进行操作。

此外，对于视频数据，它可以被看作是一系列帧，每一帧都是一张彩色图像。因此，视频数据的张量形状可以表示为（sample，frames，height，width，color_depth），即（样本，帧，高度，宽度，颜色深度）。

总之，深度学习中的张量通道是处理图像和视频等数据的关键部分，理解其形状和约定对于有效进行模型训练和推理至关重要。全连接层间的激活函数（Activation Function）是在人工神经网络的神经元上运行的函数，负责将神经元的输入映射到输出端。在全连接层中，输入的 inputs 通过加权和求和后，作用在一个函数上，这个函数就是激活函数。

激活函数对于人工神经网络模型去学习、理解非常复杂和非线性的函数来说具有十分重要的作用。它们将非线性特性引入到网络中，使得模型能够表示更加复杂的关系。

隐藏层层数对深度学习模型具有显著的影响，主要体现在以下几个方面：

模型的表达能力：增加隐藏层的数量可以增强模型的表达能力，使其能够捕获数据中更复杂和分层的特征。更深的网络架构可以学习复杂的模式和关系，从而提高在复杂任务上的性能。

特征抽象：每个隐藏层都可以从输入数据中学习越来越抽象和高级的特征。随着隐藏层数量的增加，网络能够逐渐学习到更加抽象和复杂的特征表示，这有助于减少手动特征工程的需要。

处理复杂任务：隐藏层有助于神经网络处理更复杂的任务，如图像识别、语音识别和自然语言处理等。这些任务需要对输入数据进行多层次的抽象和处理，而隐藏层的存在使得网络能够进行更深层次的特征提取和表示学习。

梯度消失和爆炸：然而，非常深的网络也可能导致梯度消失或梯度爆炸的问题。在反向传播过程中，梯度可能变得过小（导致收敛缓慢或训练停滞）或过大（导致数值不稳定）。为了缓解这些问题，可以采用一些技术，如批量标准化和跳过连接（如残差连接）。

训练时间和过拟合：增加隐藏层数量通常会增加网络的复杂度，从而可能导致训练时间增长。此外，过多的隐藏层也可能增加过拟合的风险，即模型在训练数据上表现良好，但在未见过的数据上性能下降。

参数数量：尽管增加隐藏层数可能导致参数增多，但深层网络有时可以使用更少的参数来表示复杂的函数，因为隐藏层可以共享参数和通过层与层之间的连接共享信息.

综上所述，隐藏层层数对深度学习模型的影响是复杂的，需要在模型的表达能力、训练效率、过拟合风险等因素之间进行权衡。在设计深度学习模型时，应根据具体任务和数据特点来选择合适的隐藏层层数。

Batch Normalization（批标准化）和 Dropout 是深度学习中常用的两种技术，它们各自在模型训练中发挥着重要的作用。

Batch Normalization 主要用于解决内部协变量偏移问题，即在训练过程中，每一层的输入分布会发生变化，这可能导致训练困难。通过标准化每一批数据的均值和方差，Batch Normalization 可以使每一层的输出都具有适当的尺度，从而稳定模型的训练过程。这不仅能提高模型的训练速度，还有助于提升模型的性能。

另一方面，Dropout 是一种正则化手段，用于防止模型过拟合。在训练过程中，Dropout 会随机选择神经层中的一些单元并将其临时隐藏，这样可以使模型不太依赖于某些局部特征，从而提高其泛化能力。在测试阶段，所有神经元都被使用，但每个神经单元的权重参数需要乘以一个概率 p，以保持与训练阶段的一致性。

总的来说，Batch Normalization 和 Dropout 都是深度学习模型训练中非常有效的技术。它们各自具有不同的作用机制，但共同的目标都是提高模型的性能和泛化能力。在实际应用中，根据具体任务和数据特点，可以灵活选择是否使用以及如何使用这两种技术。
在 RNN（循环神经网络）中，混淆矩阵（Confusion Matrix）主要用于评估模型的分类性能。RNN 常用于处理序列数据，如文本、时间序列等，特别是在分类任务中，如情感分析、文本分类等。在这些任务中，混淆矩阵是一个重要的工具，用于比较模型的预测结果与实际标签之间的差异。

混淆矩阵是一个 n 行 n 列的矩阵，其中 n 是类别的数量。每一行代表实际类别，每一列代表预测类别。每个单元格的值表示实际为某一类别但预测为另一类别的样本数量。例如，对于二分类问题，混淆矩阵通常包括真正例（TP）、假正例（FP）、真负例（TN）和假负例（FN）四个部分。

通过混淆矩阵，我们可以计算出一系列评估指标，如准确率、召回率、精确率以及 F1 分数等。这些指标帮助我们全面了解模型的性能，包括模型在识别不同类别时的准确性、是否容易将某个类别误分类为其他类别等。

因此，在 RNN 中，混淆矩阵是评估模型分类性能的重要工具，它提供了一种直观且量化的方式来比较模型的预测结果与实际标签之间的差异。
混淆矩阵归一化是将混淆矩阵中的每个元素除以相应行或列的总和，以便于比较各类别之间的差异。这种归一化的目的主要是使混淆矩阵中的值更容易解释，因为它们现在表示的是每个类别的样本百分比，而不是绝对数量。

在可视化混淆矩阵时，归一化特别有用，因为它可以清晰地展示模型在每个类别上的性能，而不受类别样本数量的影响。通过比较每个类别的归一化值，可以更容易地识别模型在各个类别上的分类准确度，并了解模型在不同类别上的性能表现。

请注意，归一化混淆矩阵时，可以选择按行归一化（即每行的和变为 1，这样每个元素表示该类别的预测正确率）或按列归一化（即每列的和变为 1，这样每个元素表示被预测为该类别的实际占比）。具体选择哪种归一化方式取决于你的分析目的。

总之，混淆矩阵归一化是一种重要的数据处理步骤，它有助于更深入地理解模型的分类性能，并为模型优化提供有价值的洞察。