|  |
| --- |
| 电子科技大学  **计算机专业类课程** |
| **实验报告** |
| **课程名称：人工智能综合实验II**  **学院专业：计算机科学与工程学院**  **学生姓名：胡永祺**  **学　　号：2020080602010**  **指导教师：高联丽** |
| **日　　期：2022年 月 日** |
|  |
| 电子科技大学计算机学院实验中心 |

**电 子 科 技 大 学**

**实 验 报 告**

**实验一**

# 一、实验室名称：

电子科技大学清水河校区主楼A2-413

# 二、实验项目名称：

人工智能实验II-1：环境熟悉及深度学习初探

# 三、实验原理：

1.线性回归的基本理论，线性回归模型是利用称为线性回归方程的最小平方函数对一个或多个自变量和因变量之间关系进行建模的一种回归分析。

2．softmax回归的基本理论，softmax回归模型是线性回归模型在多分类问题上的推广，在多分类问题中，类标签y可以取两个以上的值。

3.多层感知器的基本理论，多层感知器模型是在线性回归或者softmax回归的基础上添加多层的操作，并有层与层之间有激活函数。

4. 对比分析在深度学习领域是使用不同参数、模型结构等选择出最优模型，达到分析比较的目的。可视化分析是人工操作将数据进行关联分析，并做出完整的分析图表。

# 四、实验目的：

1） 学习基本的Pytorch的基本数据操作

2） 深度学习模型搭建的基本流程

3） 利用Pytorch实现线性回归、Softmax及多层感知器回归模型

4) 掌握实验结果的分析能力，包括两个方面：对比分析与可视化分析

# 五、实验内容：

1. 问题描述

通过softmax回归模型和多层感知机模型实现对FashionMNIST数据集的分类

2. 算法的概要设计与分析

多层感知机（MLP）算法：将FashionMNIST数据集的图片以Tensor形式输入神经元。

线性回归算法：利用线性回归方程的最小平方函数来计算预测值与真实值的差（loss）,其公式为：

Softmax算法设计：计算MLP输出层loss之后向量的每一个分量的占比，公式为，结果最大的分量对应的标签为网络预测结果。

交叉熵：在实际使用中大部分事物都不是均匀分布的，这时计算信息量就可以使用交叉熵，它是在非均匀分布下信息量的一种表述，其公式为

3. 核心算法的详细设计与实现

多层感知机（MLP）算法：MLP的工作原理如图 1所示，通过公式和激活函数，计算出各个样本的估计值。其中为权重，表示神经元之间的连接强度，为偏置。常见的激活函数有Sigmoid,tanh,ReLU等，可将神经元的输出幅度限制在一定范围内。

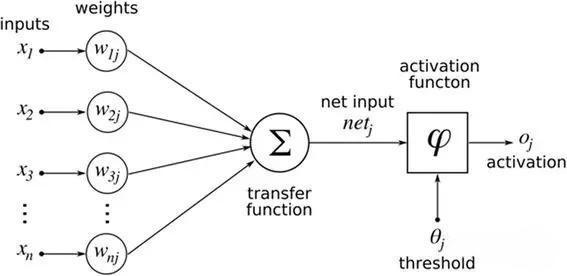


图 1 MLP原理图

最典型的MLP包括包括三层（如图 2）：输入层、隐藏层和输出层，MLP神经网络不同层之间是全连接的，即上一层的任何一个神经元与下一层的所有神经元都有连接。

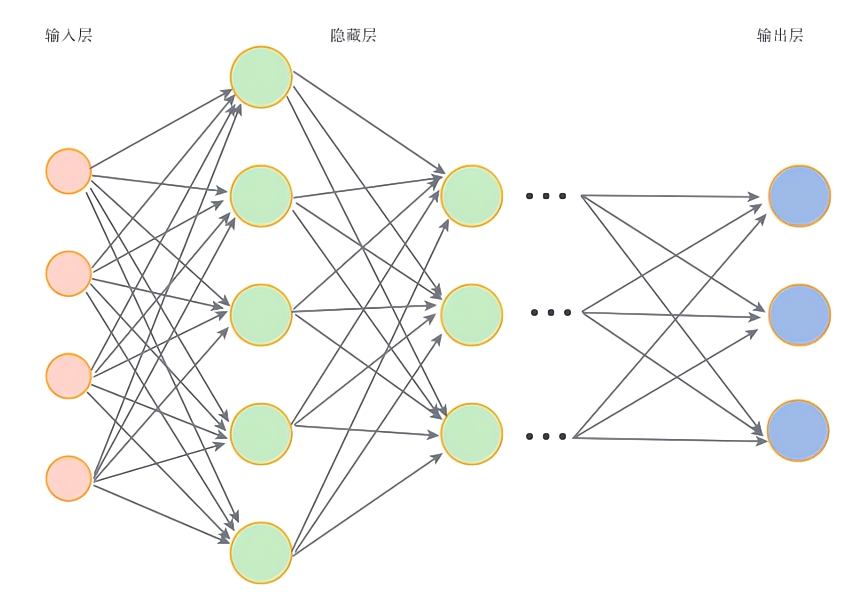


图 2 MLP结构图

MLP的训练分为正向传播和反向传播两种。正向传播FP主要是为了求损失。在这个过程中,我们根据输入的样本,给定的初始化权重值W和偏置项的值b, 计算最终输出值以及输出值与实际值之间的损失值。如果损失值不在给定的范围内则进行反向传播的过程，否则停止,的更新。反向传播BP主要是通过回传误差来更新网络的权重值W和偏置b。将正向传播计算的误差通过隐层向输入层逐层反传,计算每一层W和b对loss的梯度，通过梯度下降法更新W和b使损失减少。然后再次进行正向传播，如此循环，直到满足要求。如图 3所示。

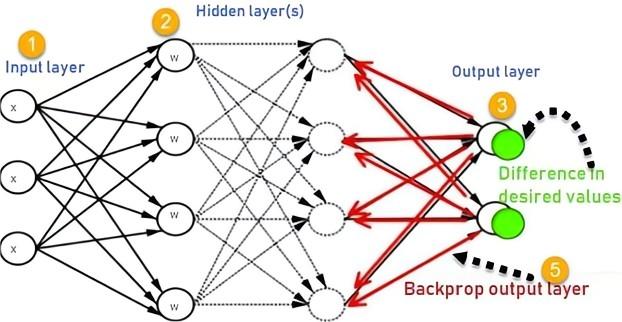


图 3 MLP的工作流程

# 六、实验器材（设备、元器件）：

|  |  |
| --- | --- |
| 硬件平台 | PC端 |
| 开发环境 | 操作系统Windows10 |
| 测试环境 | Python3.9.6+Pytorch1.12.0+cuda11.6 |

# 七、实验步骤：

1. 网络框架结构搭建介绍：

网络框架如图 4所示，使用单层感知机模型，输入层维度为784（），输出层维度为10。

# 

图 4网络框架

1. 核心算法流程：

|  |
| --- |
| **输入：**训练集FashionMNIST  感知机网络net  Adam优化器optim  CrossEntropy损失函数loss\_fun  归一化函数Softmax函数  正确率计算函数evaluate\_accuracy  **过程：**   1. 将数据集传入DataLoader并设置batchsize 2. **for** epoch 1 to 20 3. 从DataLoader中取出img,target 4. 将img传入网络net得到output 5. lossloss\_fun(output,target) 6. accuracyevaluate\_accuracy(output,target) 7. BP算法更新net的权重w和偏置b 8. **end for**   **输出：**loss,accuracy,折线图 |

1. 训练执行流程：

依据单一变量原则和控制变量法。每次训练在最初模型上做一个参数的改动。每次训练都计算训练集上的损失，训练集上的正确率，测试集上的正确率并做出相应的折线图。

1. 核心代码展示：

|  |
| --- |
| 核心代码 |
| from re import L  num\_inputs = 784  num\_hidden\_layer = 256  num\_outputs = 10  *#定义网络框架*  class **LinearNet**(nn.Module):  def \_\_init\_\_(self, num\_inputs, num\_outputs, num\_hidden\_layer=None):  super(LinearNet, self).\_\_init\_\_()  self.net = nn.Sequential(  nn.Linear(num\_inputs, num\_hidden\_layer),  nn.Dropout(0.5),  nn.ReLU(inplace=True),  nn.Linear(num\_hidden\_layer,num\_outputs)  )  self.init\_weights()  def init\_weights(self):  nn.init.kaiming\_normal\_(  self.linear.weight, mode='fan\_in', nonlinearity='leaky\_relu') *# 对权重采用凯明初始化方法*  self.linear.bias.data.fill\_(0)  def forward(self, x): *# x shape: (batch, 1, 28, 28)*  x = x.view(-1, 784)  output = self.net(x)  return output  *#定义device*  device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")  *#输出初始化的参数*  linear\_net = LinearNet(num\_inputs, num\_outputs, num\_hidden\_layer)  linear\_net=linear\_net.to(device)  *#定义损失函数SoftMax和交叉熵函数*  loss = nn.MSELoss() *#nn.MSELoss是交叉熵函数和softmax函数的结合*  loss=loss.to(device)  *# 定义优化器*  import torch.optim  optimizer = torch.optim.Adam(linear\_net.parameters())  *# 定义评价函数*  def evaluate\_accuracy(data\_iter, net, device=None):  if device **is** None **and** isinstance(net, torch.nn.Module):  *# 如果没指定device就使用net的device*  device = list(net.parameters())[0].device  acc\_sum, n = 0.0, 0  with torch.no\_grad():  for X, y **in** data\_iter:  if isinstance(net, torch.nn.Module):  net.eval() *# 评估模式, 这会关闭dropout*  acc\_sum += (net(X.to(device)).argmax(dim=1) == y.to(device)).float().sum().cpu().item()  net.train() *# 改回训练模式*  else: *# 自定义的模型, 3.13节之后不会用到, 不考虑GPU*  if('is\_training' **in** net.\_\_code\_\_.co\_varnames): *# 如果有is\_training这个参数*  *# 将is\_training设置成False*  acc\_sum += (net(X, is\_training=False).argmax(dim=1) == y).float().sum().item()  else:  acc\_sum += (net(X).argmax(dim=1) == y).float().sum().item()  n += y.shape[0]  return acc\_sum / n  *# 模型训练过程*  num\_epochs = 20 *#训练轮数*  losses=[] *#收集损失*  test\_accuracy = [] *#收集测试集正确率*  train\_accuracy = [] *#收集训练集正确率*  for epoch **in** range(num\_epochs):  train\_l\_sum, train\_acc, test\_acc,n = 0.0, 0.0, 0.0,0  for X, y **in** train\_iter:  X=X.to(device)  y=y.to(device) *#将训练数据和标签都放在device上面*  output = linear\_net(X)  temp = torch.zeros(len(y),10)  temp =temp.to(device)  for i,label **in** enumerate(y):  temp[i,label]=1  y = temp *#对标签进行重构，满足[0,...,1,...0]的形式*  l = loss(output, y)  optimizer.zero\_grad() *#清除梯度*  l.backward()  optimizer.step() *#更新网络参数*  train\_l\_sum += l  losses.append(l.item())  n += 1  train\_acc = evaluate\_accuracy(train\_iter, linear\_net,"cuda")  test\_acc = evaluate\_accuracy(test\_iter, linear\_net,"cuda")  train\_accuracy.append(train\_acc)  test\_accuracy.append(test\_acc)  print('epoch **%d**, loss **%.4f**, train acc **%.3f**, test acc **%.3f**'  % (epoch + 1, train\_l\_sum / n, train\_acc , test\_acc))  *# 训练过程可视化*  plt.plot(losses)  plt.title("training losses")  plt.savefig("./loss.png") *#保存loss图片*  plt.show()  plt.subplot(121)*#创建子图1*  plt.plot(train\_accuracy)  plt.title("training accuracy")  plt.xlabel("epoch")  plt.subplot(122) *# 创建子图2*  plt.plot(test\_accuracy)  plt.title("test accuracy")  plt.xlabel("epoch")  ax=plt.gca()  ax.yaxis.tick\_right()  plt.savefig("./accuracy.png") |

# 八、实验数据及结果分析：

1. **实验数据：**

Fashion-MNIST是一个替代[MNIST手写数字集](http://yann.lecun.com/exdb/mnist/)的图像[数据集](https://so.csdn.net/so/search?q=%E6%95%B0%E6%8D%AE%E9%9B%86&spm=1001.2101.3001.7020)。它是由Zalando（一家德国的时尚科技公司）旗下的[研究部门](https://research.zalando.com/)提供。其涵盖了来自10种类别的共7万个不同商品的正面图片。Fashion-MNIST的大小、格式和训练集/测试集划分与原始的MNIST完全一致。60000/10000的训练测试数据划分（即训练集有60000张图片，测试集有10000张图片），图片尺寸为2828。图 5展示了数据集对应10个类别可视化的结果



图 5数据集

1. **结果分析：**
2. **线性回归模型和Softmax模型的对比**

由图 6可知，Softmax的表现要优于线性回归模型（MSEloss），这大概是因为Softmax与交叉熵配合使用，使用指数形式的[Softmax](https://so.csdn.net/so/search?q=Softmax&spm=1001.2101.3001.7020" \t "_blank)函数能够将差距大的数值距离拉的更大，而交叉熵能更好的体现分类，对误分类能有很好的区分。而MSE，对梯度的更新较慢。尤其输出值预测的接近0或1时，梯度几乎为0。

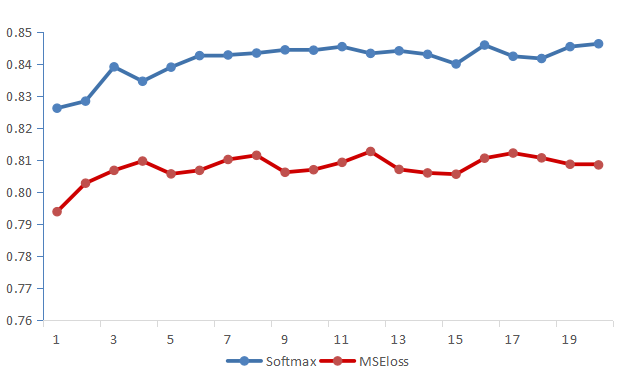


图 6 线性回归与Softmax回归的正确率

1. **不同的迭代次数**  
    探究神经网络在不同迭代次数下的训练结果。图 7是训练50个epoch的traing loss。每一个batchsize=32记录一次loss，可以看到loss从训练开始迅速下降，之后一直在0~0.5之间振荡。网络模型为单层感知机，使用Softmax回归，优化器使用Adam优化器。

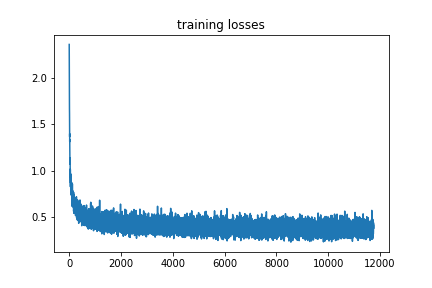


图 7 Epoch从1到50的training loss

图 8为该网路在训练集和测试集上的正确率。可以发现，训练集上的正确率整体高于测试集。训练集的最高正确率为87.4%，测试集的最高正确率为85.1%。

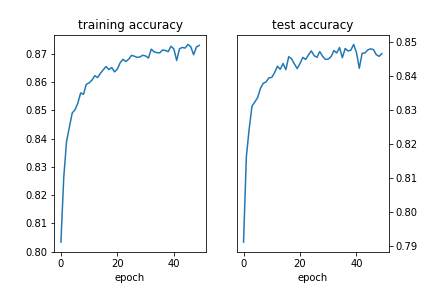


图 8 Epoch从1到50的训练集正确率和测试集正确率

分析正确率的折线图，未出现明显的过拟合，这大概是因为训练epoch并不算大，而且网路由单层感知机构成，网略深度较浅，结构较简单的缘故。

1. **采用不同批训练的大小（batch\_size）**

图 9为不同batchsize下在训练集和测试集上的正确率。选择了32,64,256的batchsize,网络为单层感知机，使用Softmax回归，优化器使用Adam。可以发现32和64的batchsize效果相当，略优于256的batchsize。因而256对于该神经网络是一个较大的batchsize，跑完一次epoch(全数据集)所需的迭代次数减少，要想达到相同的精度，其所花费的时间大大增加，从而对参数的修正也就显得更加缓慢，batchsize增大到一定程度，其确定的下降方向已经基本不再变化（会影响随机性的引入）。

|  |  |
| --- | --- |
|  |  |

图 9 不同batchsize下的正确率

1. **采用不同的优化算法**

图 10为使用不同优化器，网络在测试集上的正确率。选择了Adam,SGD,Adagrad三种优化器。网络结构同上，batchsize取32。结果发现，Adam和Adagrad的效果相似，优于SGD。Adam和Adagrad都是在SGD模型上进行了优化。Adam在SGD上增加了动量算法，既能适应稀疏梯度，又能缓解梯度震荡的问题。Adagrad为不同的变量提供不同的学习率。它增加了罕见但信息丰富的特征的影响。

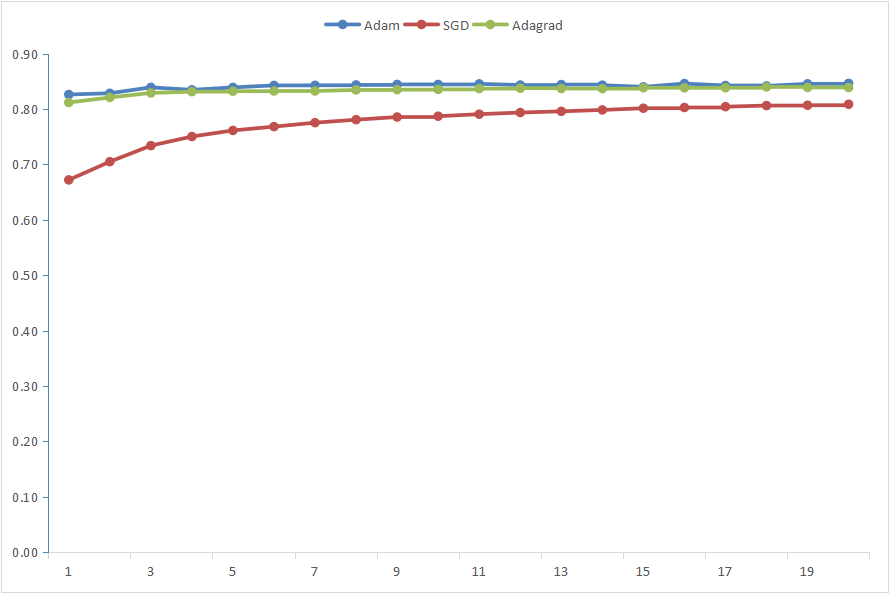


图 10 使用不同优化器的正确率

1. **采用避免过拟合的方式（dropout、正则项等）**

图 11为正则化方法对网络的影响。正则化方法有dropout和L2正则化。L2正则化的效果欠佳，这是因为网络层次较浅，由单层感知机构成，L2正则化在较深的网络有很好的表现。Dropout能以一定概率（此处p=0.5）去除一些神经元，提高网络的泛化性，在本网络中有一定体现。

|  |  |
| --- | --- |
|  |  |

图 11使用不同正则化方法的正确率

1. **不同的学习率（learning\_rate）**

图 12探讨不同学习率对Adam优化器的影响。Adam的默认学习率为1e-3,此处选择了3e-4和1e-2两个学习率来相互比较。1e-3和3e-4的效果相近，而1e-2的效果差于这两个学习率。学习率过大，会导致待优化的参数在最小值附近波动，不收敛；学习率过小，会导致待优化的参数收敛缓慢。学习率为1e-2时可以看到十分明显的波动



图 12 不同学习率下的正确率

1. **网络深度的大小**

图 13为多层感知机模型探究隐藏层维度的影响。结果显示隐藏层的效果欠佳，原因可能是因为多层感知机模型没有对图片特征进行提取，所以加入了影藏层也不能提供更多的信息。还有一个原因是出于对照和单一变量的要求，没有加入激活函数。

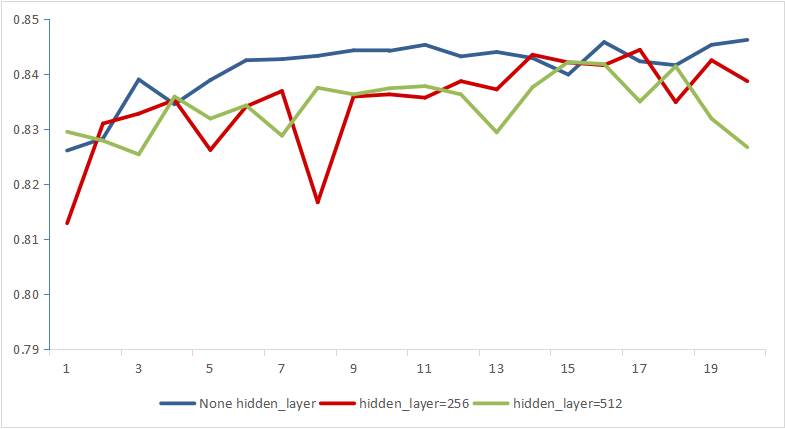


图 13 不同隐藏层维度下的正确率

1. **使用激活函数**

图 14探究激活函数对神经网络训练结果的影响。网络均采用了2层隐藏层，一层维度为512，另一层的维度为256，batchsize为32，训练epoch为20。可以很明显地看到，使用激活函数比不使用效果要好。激活函数为神经网络加入非线性因素的，提高神经网络对模型的表达能力，解决线性模型所不能解决的问题。同时ReLU函数还能解决了梯度消失问题，且计算速度和收敛速度快。

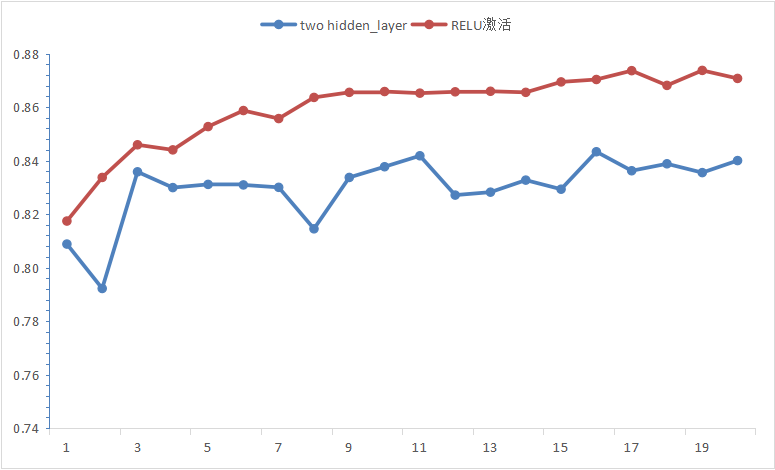


图 14 是否使用激活函数

将不同参数下模型的最高正确率整理为

表 1，考虑单一变量原则和控制变量法，每次实验模型只会改变一个参数，默认参数为:

。

表 1不同参数下模型的最高正确率

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 变化参数 | Batchsize=64 | Batchsize=32 | Batchsize=256 | Epoch=50 | Epoch=20 |
| 最高正确率 | 84.53% | 84.32% | 84.01% | **85.12%** | 84.32% |

续表1：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 变化参数 | Adam | SGD | Adagrad | Hidden layer | dropout |
| 最高正确率 | **84.32%** | 79.35% | 83.75% | 83.87% | 83.33% |

续表2：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 变化参数 | L2正则化 | lr=1e-3 | lr=3e-4 | ReLU | MSEloss |
| 最高正确率 | 82.11% | 84.32% | 84.34% | **87.57%** | 81.26% |

综合各个因素，最优模型如表 2。

表 2 最优模型参数

|  |  |
| --- | --- |
| 模型参数 | 参数取值 |
| batchsize | 32 |
| 优化器 | Adam |
| 学习率 | 1e-3 |
| 输入层维度 | 784 |
| 隐藏层维度 | 256 |
| 输出层维度 | 10 |
| 激活函数 | ReLU |
| 正则化方法 | droput(0.5) |
| 损失函数 | CrossEntropy |
| 最高正确率 | 89.53% |

# 总结及心得体会：

通过本次实验，我掌握了softmax函数，多层感知机，交叉熵，激活函数的基本原理和作用。 1)softmax用于多分类过程中，它将多个神经元的输出，映射到（0,1）区间内，可以看成概率来理解，从而来进行多分类。softmax通过指数进行加权，能够更好的放大不同类别之间的概率差异，从而提升网络的鲁棒性。 2)多层感知机（MLP）模型是一个传统的神经网络，它能够解决单层感知机无法解决的非线性问题（单层感知机只能对线性数据进行分类）多层感知机最初是为了做分类的，但是它也可以处理回归问题，只要在最后将softmax等分类器改成sigmoid回归就行。它的训练方式也是“BP反向传导”。 3) 交叉熵损失函数是按照概率分布q的最优编码对真实分布为p的信息进行编码的长度。在给定q的情况下，如果 p和q越接近，交叉熵越小；如果 p 和 q越远，交叉熵就越大。因而，交叉熵常被用作分类任务的损失函数。 4) 激活函数为神经网络加入非线性因素的，单纯的感知机模型只能学到线性特征，而为感知机模型引入激活函数后能够更好地让网络学习非线性特征。激活函数往往需要连续并可导（允许少数点上不可导）的非线性函数。可导的激活函数可以直接利用数值优化的方法来学习网络参数。激活函数及其导函数要尽可能的简单，有利于提高网络计算效率。激活函数的导函数的值域要在一个合适的区间内，不能太大也不能太小，否则会影响训练的效率和稳定性。常见的激活函数有Sigmoid，tanh,ReLU函数等。其中是ReLU函数是目前最常用的激活函数，它具有单侧抑制、宽兴奋边界的生物学合理性，可缓解梯度消失问题。 5) 根据万能近似定理，对于具有线性输出层和至少一个使用“挤压”性质的激活函数的隐藏层组成的神经网络，只要其隐藏层神经元的数量足够多，它就可以以任意精度来近似任何一个定义在实数空间中的有界闭集函数。随着深度的增加，网络的表示能力呈指数增加，因而更深层的网络具有更好的泛化能力。但是，随着网络层数的增加，网络可能出现梯度消失等问题，使得整个网络很难训练。 6) 因为神经网络的拟合能力强，反而容易在训练集上产生过拟合。因此，在训练深层神经网络时，需要通过一定的正则化方法来改进网络的泛化能力。常见的方法有小批量梯度下降法（mini-batch），它既利于寻找全局最优解，梯度方差小，又训练速度快。L1和L2正则化方法，通过约束参数减小过拟合现象。还有dropout方法，通过随机丢弃一些网络结点，从而简化了网络，防止了过拟合现象。

通过本次实验，我对python基本语法更加熟悉，对pytorch框架更加熟悉。通过较熟练地使用pytorch建立基本的神经网络框架，创建损失函数，优化器等，能够完整地写出训练和测试过程，并将训练和测试放在cuda上。

# 十、对本实验过程及方法、手段的改进建议及展望：

**实验二**

# 一、实验室名称：

电子科技大学清水河校区主楼A2-413

# 二、实验项目名称：

人工智能实验II-2：基于卷积神经网络图像分类算法实现

# 三、实验原理：

卷积网络在本质上是一种输入到输出的映射，它能够学习大量的输入与输出之间的映射关系，它主要被用来识别位移、缩放及其他形式扭曲不变性的二维图像。由于CNN的特征检测层通过训练数据进行学习，所以在使用CNN时，避免了显式的特征抽取，而隐式地从训练数据中进行学习；再者由于同一特征映射面上的神经元权值相同，所以网络可以并行学习，这也是卷积网络相对于神经元彼此相连网络的一大优势。卷积神经网络以其局部权值共享的特殊结构在语音识别和图像处理方面有着独特的优越性，其布局更接近于实际的生物神经网络，权值共享降低了网络的复杂性，特别是多维输入向量的图像可以直接输入网络这一特点避免了特征提取和分类过程中数据重建的复杂度。

# 四、实验目的：

通过Pytorch框架搭建LeNet网络，实现在CIFAR-10数据集上的图像分类，并以此掌握：

1. 卷积神经网络中卷积层、池化层、全连接层的使用场景和作用以及它们参数的具体含义。
2. 了解Pytorch框架下模型的训练流程：训练、测试以及评估方法，并亲自代码实践。
3. 实验结果的分析能力，包括两个方面：量化分析与可视化分析。

# 五、实验内容：

1. 问题描述

基于卷积神经网络图像实现分类算法

1. 算法的概要设计与分析

网络采用了卷积算法，池化层算法，Batch Normalization,MLP模型，BP算法等，通过卷积和池化提取图片的特征，然后用MLP对其进行分类预测，并通过BP算法更新网络权重和偏置，实现网络的优化。

1. 核心算法的详细设计与实现

**卷积算法：**卷积算法是构建卷积神经网络的核心。输入层每个通道卷积范围内的每个待征值乘上一个卷积核对应通道的对应权重再全部求和，等于输出层一个通道上的一个特征值。直观地来说，卷积运算会让滤波器学习到某些类型的视觉特征，具体的视觉特征可能是某些方位上的边界，或者在第一层上某些颜色的斑点，甚至可以是网络更高层上的蜂巢状或者车轮状图案。卷积层可以被看做是神经元的一个输出。神经元只观察输入数据中的一小部分，并且和空间上左右两边的所有神经元共享参数。

**池化算法**：池化算法分为最大池化(Maxpooling)和平均池化(Avaragepooling)。最大池化是取输入层在池化核对应权重的最大值，而平均池化则是去对应范围内所有权重的平均值。通常在连续的卷积层之间会周期性地插入一个池化层。它的作用是对图片进行下采样操作，逐渐降低数据体的空间尺寸，这样的话就能减少网络中参数的数量，使得计算资源耗费变少，也能有效控制过拟合。

**全连接层（MLP）：**对于任一个卷积层，都存在一个能实现和它一样的前向传播函数的全连接层。该全连接层的权重是一个巨大的矩阵，除了某些特定块（感受野），其余部分都是零。而在非 0 部分中，大部分元素都是相等的（权值共享）。在全连接层中，我们将最后一个卷积层的输出展平，并将当前层的每个节点与下一层的另一个节点连接起来。它的功能是为了对提取的特征进行分类。

Batch Normalization（BN）：对每一个输入的mini-batch，计算batch的均值和标准差，对batch中的每一个元素作标准化(是一个很小的数，避免由分母等于0带来的系统错误)。BN算法调整了每层网络输出数据的分布，使其进入激活函数的作用区（原点附近的区域），梯度弥散率低，区分率高。同时，BN会在训练过程中，自己调节数据分布，使其“更合理”地进入激活函数。

# 六、实验器材（设备、元器件）：

|  |  |
| --- | --- |
| 硬件平台 | PC端 |
| 开发环境 | 操作系统Windows10 |
| 测试环境 | Python3.9.6+Pytorch1.12.0+cuda11.6 |

# 七、实验步骤：

1. 网络框架结构搭建介绍

设计的卷积神经网络主体框架如图 15。

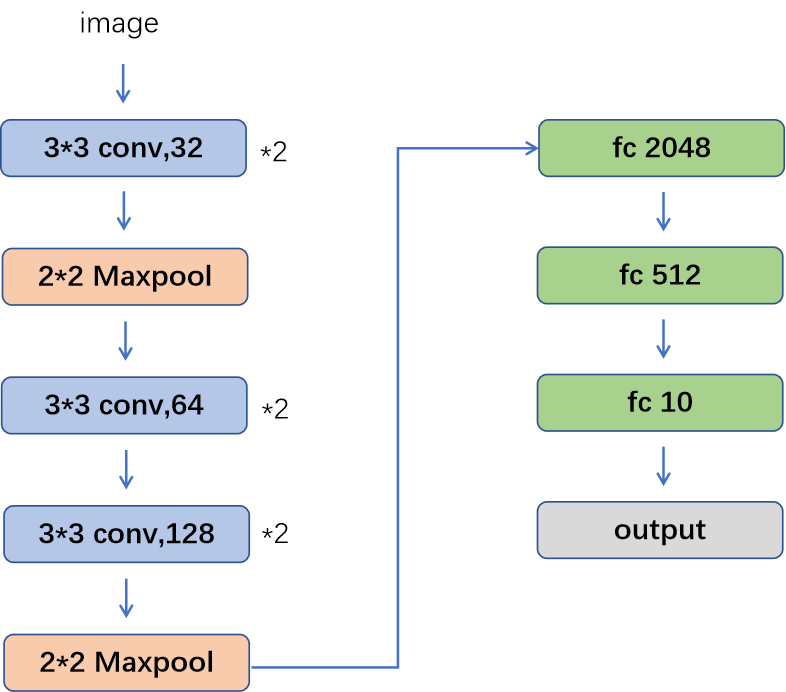


图 15 网络主体框架

网络采用了3个卷积核为33的

1. 核心代码介绍

|  |
| --- |
| **输入：**训练集CIFAR10  卷积神经网络net  Adam优化器optim  CrossEntropy损失函数loss\_fun  归一化函数Softmax函数  **过程：**   1. 将数据集传入DataLoader并设置batchsize 2. **for** epoch 1 to 30 3. 从DataLoader中取出img,target 4. 将img传入网络net得到output 5. lossloss\_fun(output,target) 6. accuracy 7. BP算法更新net的权重w和偏置b 8. **end fo**r   **输出：**loss,accuracy,折线图 |

1. 训练执行流程介绍

总共训练30个epoch。每次训练都计算训练集上的损失，训练集上的正确率，测试集上的正确率并通过tensorboard做出相应的折线图。根据训练结果对网络进行一些调整（如卷积核，池化核的大小等）

# 八、实验数据及结果分析：

1. 实验数据

CIFAR-10 是一个包含60000张图片的数据集。如图 16，其中每张照片为32\*32的彩色照片，每个像素点包括RGB三个数值，数值范围 0 ~ 255。所有照片分属10个不同的类别，分别是 'airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck'。其中训练集有50000张图片，测试集有10000张图片。



图 16 CIFAR10数据集

1. 结果分析：
   1. 量化展示随着训练迭代数目变化准确率以及其变化曲线

图 17为随着训练迭代数目变化，神经网络在训练集和测试集上的正确率，总共迭代epoch为30。从图中可以看到，网络在训练集上的正确率随着迭代次数的增加增长迅速，并最终达到95%左右。而在测试集上的增速比训练集缓慢，且在epoch=20后振荡，无明显增加，最高正确率为85%左右，比训练集的正确率低很多。原因大概是随着训练迭代次数的增加，网络出现了过拟合，这一点在下文展示的网络分别在训练集和测试集上的损失（图 19,图 20）可以更清晰的看到。

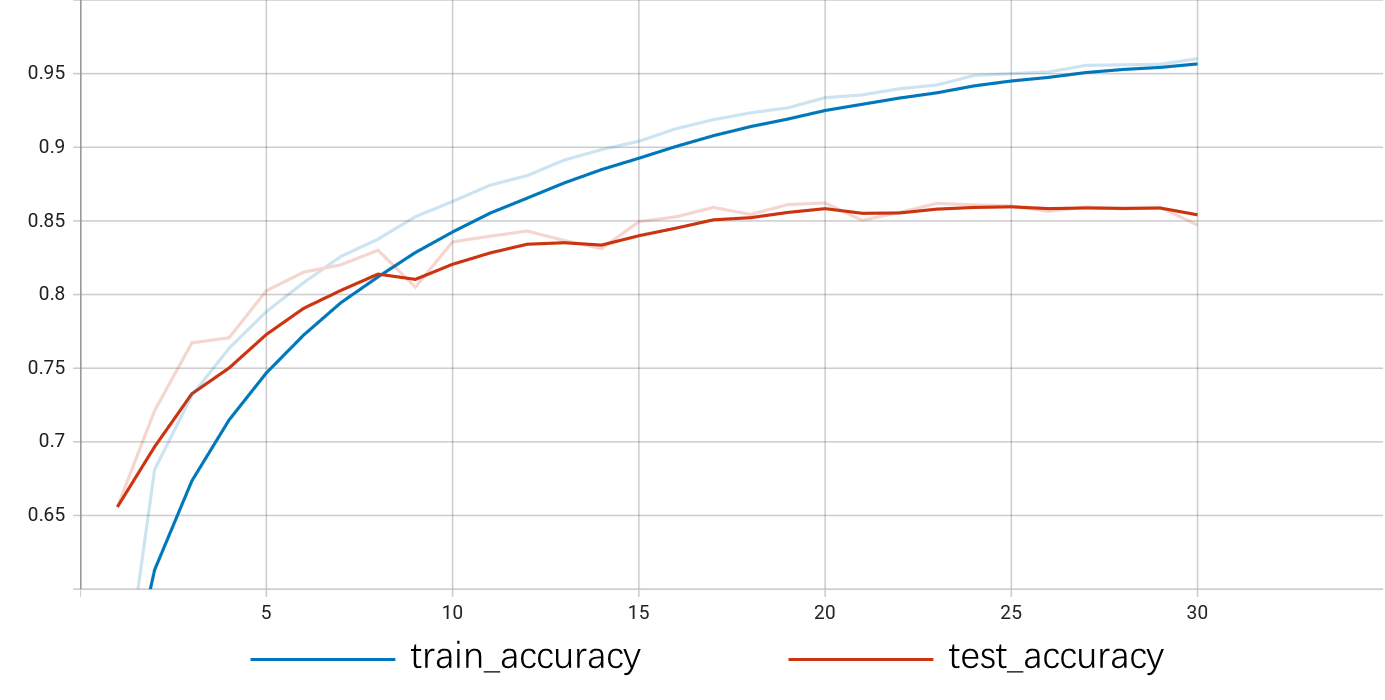


图 17训练30个epoch的正确率

1. 量化展示随着训练数据集大小对于准确率的影响极其变化曲线

图 18展示了模型在不同大小数据集训练和测试的正确率。数据集的大小有训练集和测试集各5k,10k,20k（由于测试集一共只有10k张图片，因此此时取测试集全集10k即可）以及全集50k张图片的训练集和10k张图片的测试集。

|  |  |
| --- | --- |
|  |  |

图 18不同数据集大小对正确率的影响

从图中可以看出，4种大小训练集的正确率整体都呈上升趋势，开始时正确率由低到高依次为5k,10k,20k,fullset。在epoch=30的时候，4者的正确率几乎相当。4种大小测试集的正确率也整体都呈上升趋势，在整个测试过程中，正确率由低到高依次为5k,10k,20k,fullset，且彼此之间差异较大并都低于训练集的正确率。原因是因为训练集数据量越小越容易出现过拟合现象，模型泛化能力较差，在训练集上的表现较差。因而增加数据集大小是一种常见的预防过拟合的方法。

1. 损失下降曲线，是否出现过拟合或者欠拟合现象，考虑如何解决

图 19，图 20为网络分别在训练集和测试集上的损失，可以看到随着训练迭代次数的增加，网络在训练集上的损失整体呈现下降趋势，与网路在训练集上的正确率（图 17）变化相吻合。而网络在测试集上的损失在epoch=0~20时呈下降趋势，在epoch=20后有较明显的上升。这同样与网络在训练集上的正确率（图 17）变化相吻合。损失在测试集的上升与网络在测试集上的正确率波动共同表明网络训练出现了过拟合，进一步验证了上文的猜想。

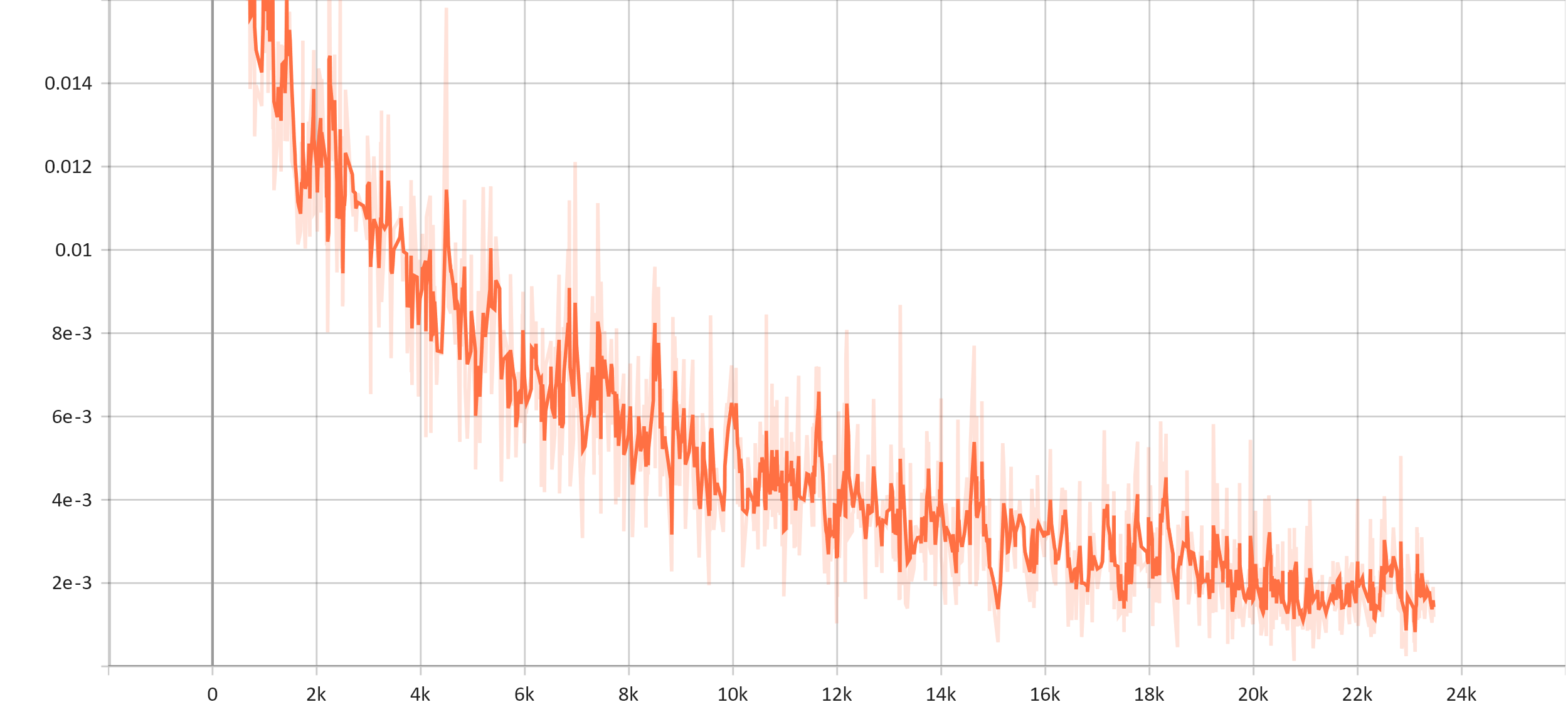


图 19 training loss

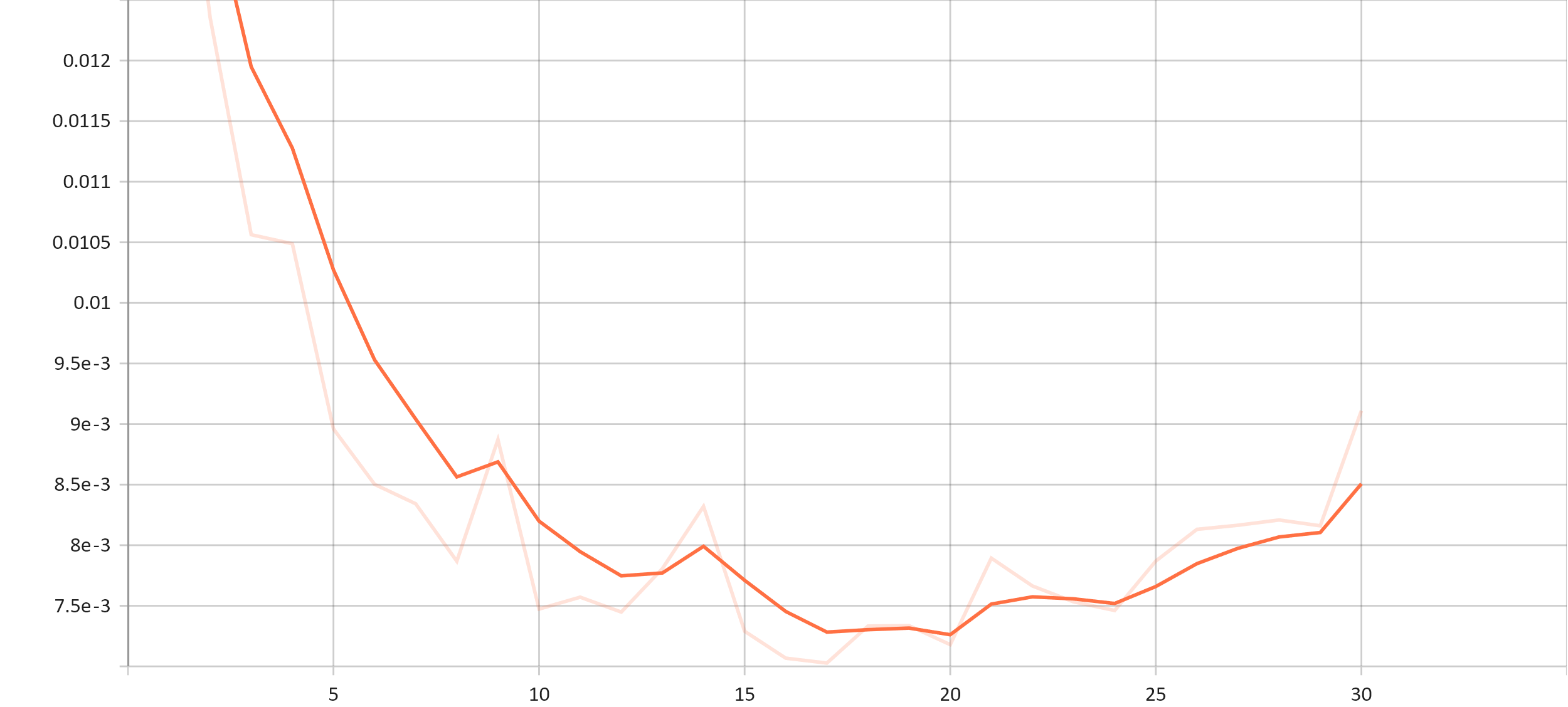


图 20test loss

常见的解决过拟合方法如下：

正则化：正则化是机器学习中最常见的过拟合解决方法，在损失函数中加入正则项来惩罚模型的参数，以此来降低模型的复杂度，常见的添加正则项的正则化技术有L1，L2正则化。L1正则化是基于L1范数的，在损失函数中加入L1范数修正项，修正后的损失函数为。L2正则化是基于L2范数的，损失函数为。

1. Batch Normalization：BM算法是一种非常有用的正则化方法，可以让大型的卷积神经网络快速收敛，同时还能提高分类的准确率。BM算法将每一层的输入值做归一化处理，并且会重构归一化处理之后的数据，确保数据的分布不会发生变化。
2. Dropout是深度学习中最常用的控制过拟合的方法，主要用在全连接层。Dropout方法是在一定的概率上（通常设置为0.5，原因是此时随机生成的网络结构最多）隐式的去除网络中的神经元，提高网络的泛化能力。
3. 可视化分析实验结果，错误分析

选取表现最佳的模型（测试集正确率为87.92%）分别在10个类别上做测试，其结果由高到低如表 3，并进行可视化，如图 21所示。模型分别在10个类别上的准确率

表 3模型分别在10个类别上的准确率

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 类别 | automobile | frog | airplane | truck | ship |
| 正确率 | 94.5% | 93.0% | 92.2% | 91.6% | 91.5% |

续表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 类别 | horse | deer | dog | bird | cat |
| 正确率 | 89.9% | 86.4% | 82.0% | 81.6% | 76.5% |

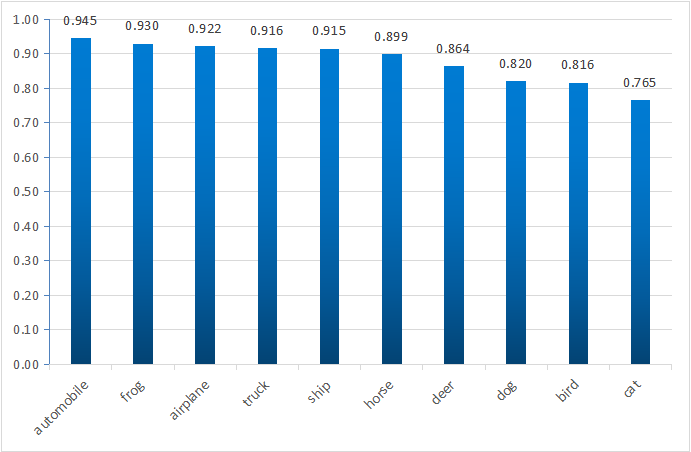


图 21模型分别在10个类别上的准确率

表现最好的类别为automobile，表现最差的类别为cat。大体上在人造物体的表现优于自然生物的表现。这大概是因为自然生物相比人造物体，亚类更多，形态差异性更大，干扰更大，学习难度更大的缘故。

**拓展实验：基于Resnet实现图像分类**

残差网络（Resnet）的主要创新是在卷积网络中增加了直连通道，此前的网络结构是性能输入做一个非线性变换，而Resnet则允许保留之前网络层的一定比例的输出即允许原始输入信息直接传到后面的层中。这样的话这一层的神经网络可以不用学习整个的输出，而是学习上一个网络输出的残差（如图 22）

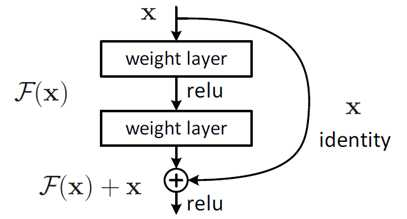


图 22 Resnet残差学习模块

将残差模块堆叠在一起实现不同层数的Resnet，常见的有Resnet18，Resnet50，Resnet101等。本次实验选择了Resnet18，网络框架结果如图 23。由于使用的CIFAR10数据集，因而在原始Resnet18做了部分调整，包括输入图片尺寸为3232，输出为10个特征值。

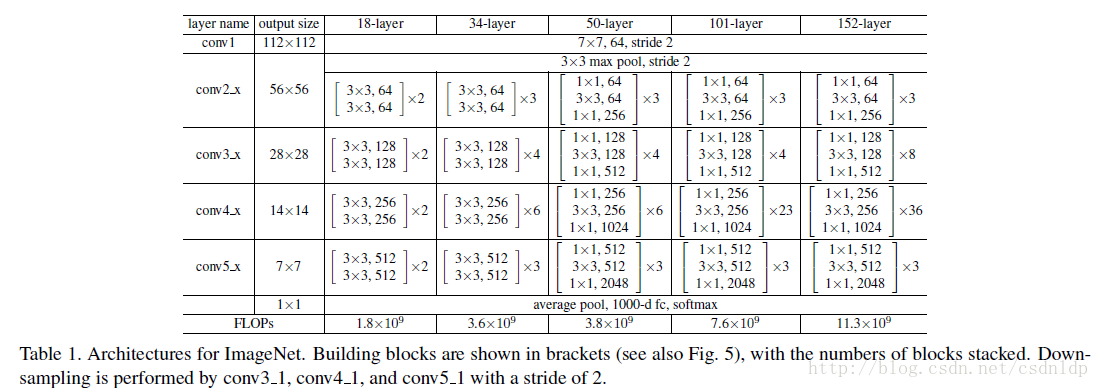


图 23 Resnet不同层数的网络架构

将搭建好的Resnet18在CIFAR10数据集上训练30个epoch，并做出网络在训练集和测试集上的正确率。同时将结果与之前搭建的CNN网络以及pytorch提供的训练好的Resnet18进行比较，做出折线图如图 24。其中Resnet(tr)表示pytorch提供的训练好的Resnet18网络，Resnet(ut)表示自己搭建的Resnet18网络，Mynet表示上文搭建的CNN网络。三者在训练集上的正确率由高到低依次为Resnet(tr)，Mynet，Resnet(ut)。三者最终在测试集上的表现由高到低为Mynet，Resnet(tr)，Resnet(ut)。分析原因大概是因为训练epoch太少，Resnet网络较深，优化较慢，并没有优化到一个特别好的效果。另外，Resnet标准输入的图片尺寸为224224，而本实验使用的图片尺寸为3232，图片尺寸较小，对于较深的网络并不是特别合适。

|  |  |
| --- | --- |
|  |  |

图 24 3种网络分别在训练集和测试集上的正确率

# 总结及心得体会：

通过本次实验，我对卷积神经网络的工作原理与实现有了很好的了解。1）卷积神经网络与普通神经网络的区别在于，卷积神经网络包含了一个由卷积层和子采样层（池化层）构成的特征抽取器。在卷积神经网络的卷积层中，一个神经元只与部分邻层神经元连接。在CNN的一个卷积层中，通常包含若干个特征图(featureMap)，每个特征图由一些矩形排列的的神经元组成，同一特征图的神经元共享权值，这里共享的权值就是卷积核。卷积核一般以随机小数矩阵的形式初始化，在网络的训练过程中卷积核将学习得到合理的权值。共享权值（卷积核）带来的直接好处是减少网络各层之间的连接，同时又降低了过拟合的风险。子采样也叫做池化（pooling），通常有均值子采样（mean pooling）和最大值子采样（max pooling）两种形式。子采样可以看作一种特殊的卷积过程。卷积和子采样大大简化了模型复杂度，减少了模型的参数。 卷积神经网络（CNN）具有共享卷积核，对高维数据处理无压力，通过卷积运算建立了局部连接的稀疏交互网络，相比于全连接前馈神经网络少了很多参数。卷积神经网络无需手动选取特征，训练好权重，即得特征分类效果好的优点，除此之外，卷积网络还具有平移不变性，平移不变性意味着系统产生完全相同的响应（输出），不管它的输入是如何平移的。2）深度学习网络在网络很深的时候，效果却越来越差了。其中的原因之一即是网络越深，梯度消失的现象就越来越明显。对于深度卷积神经网络，由于每做一次卷积（包括对应的激活操作）都会浪费掉一些信息：比如卷积核参数的随机性（盲目性）、激活函数的抑制作用等等。ResNet通过shortcut把以前处理过的信息直接再拿到现在一并处理，起到了减损的效果。因而非常深的残差网络能够很容易的优化，能够容易地从增加的深度中得到精度收益，比先前的网络产生了更好的效果。Resnet使信息更容易在各层之间流动，包括在前向传播时提供特征重用，在反向传播时缓解梯度信号消失，它没有带来额外的参数和计算开销，便于和具有相同结构的“平常”网络进行对比。

通过本次实验，我对python基本语法更加熟悉，对pytorch框架更加熟悉。在实验一的基础上，我能够进一步较熟练地使用pytorch实现卷积操作，池化操作，搭建基本的卷积网络框架。

# 十、对本实验过程及方法、手段的改进建议及展望：

**实验三**

# 一、实验室名称：

电子科技大学清水河校区主楼A2-413

# 二、实验项目名称：

人工智能实验II-3：基于RNN的文本翻译

# 三、实验原理：

RNN作为一种递归式处理序列问题的模型，在机器翻译、文本自动摘要和语音识别中有着成功的应用。巧妙的网络结构设计使得RNN可以捕捉语言中的长距离依赖关系，例如性别一致性和语法结构，而不必事先知道它们，也不需要跨语言进行 1:1 映射。

Seq2seq 是一类特殊的 RNN，它遵循了Encoder-Decoder的设计结构，两个部分均由RNN构成；Encoder将源语句转换为表示语义的向量，然后这个向量通过Decoder可以产生对应的翻译结果。

# 四、实验目的：

通过 PyTorch 框架，搭建神经网络，首先实现基本的RNN模型，以及RNN的变体—GRU。使用GRU搭建seq2seq模型，包括Encoder和Decoder两个部分以进行文本翻译任务。在此过程中需掌握：

1. PyTorch 中文本预处理(词表构建，词嵌入)的实现；
2. RNN和GRU的基本框架、Pytorch代码实现；
3. Seq2seq的基本结构和代码实现。

扩展掌握：

1. 基于RNN的图像描述生成(image caption)基本原理；
2. 具体的实现方式。

# 五、实验内容：

1. 问题描述

用RNN网络实现英法文本互译

2. 算法的概要设计与分析

将数据集中的法语和英语单词进行Embedding编码，使用Seq2Seq模型，采用2个RNN网络，一个RNN作为编码器，一个RNN作解码。采用Adan优化器进行优化。

3. 核心算法的详细设计与实现

**Embedding算法**：Embedding是一个存储向量的查找表，给定一个编号，能返回该编号对应的嵌入向量，嵌入向量反映了各个编号代表的单词之间的语义关系。Embedding能够用低维向量对物体进行编码还能保留其含义。在本实验中，输入是法语或者英语的编码，然后为每一个编码生成一个256维的向量，输出为所有向量构成的Embedding矩阵。

**RNN模型：**RNN网络用于解决训练样本输入是连续的序列,且序列的长短不一的问题。RNN最大的特点在于隐层。如图 25所示，每一层的隐层将上一个隐层的输出乘上权重V，然后与当前时刻输入乘上U进行相拼接，得到一个基础值，通过函数（通常为tanh函数）激活后，更新得到新的。每一层的输出由乘上权重W，然后通过函数（通常为Sigmoid函数）激活后得到。权重U,V,W通过反向传播算法进行更新。

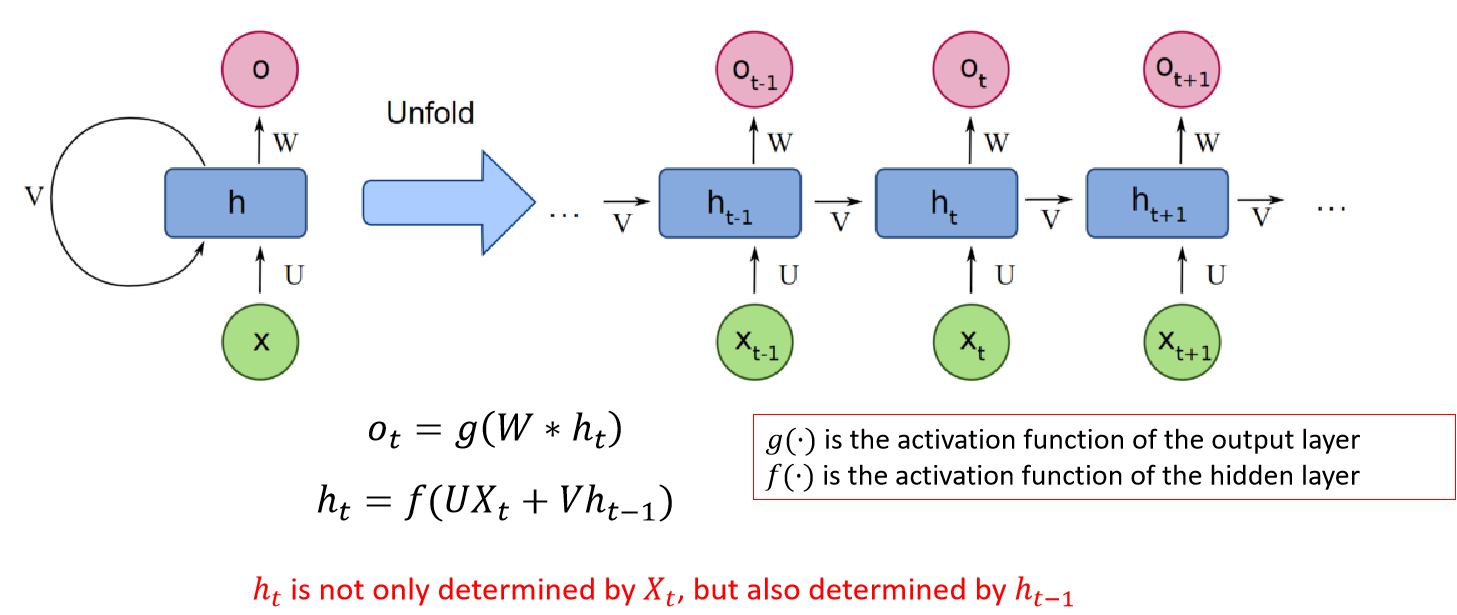
****

图 25 RNN的结构图

**Seq2Seq模型：**Seq2Seq模型是输出的长度不确定时采用的模型，广泛应用与机器翻译任务中，这是因为翻译的原文和译文长度不一定相同，所以输出的长度就不确定了。Seq2Seq模型由两个RNN构成，一个负责Encoder，另一个负责Decoder（如图 26）。Encoder负责将输入序列压缩成指定长度的向量，该向量作为序列的整体语义，这个过程称为编码，Decoder负责根据语义向量生成指定的序列，这个过程称为解码。

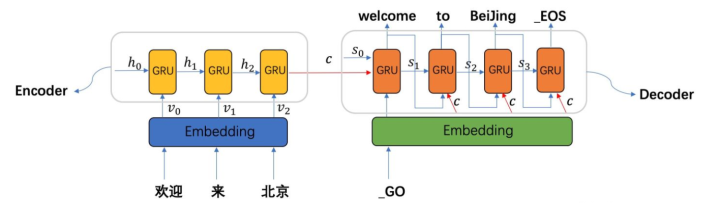


图 26 GRU网络结构

**GRU模型：**GRU在传统的RNN网路上增加了两个门。其中为控制重置的门控（reset gate），为控制更新的门控（update gate）。每一层的和更新方式如图 27所示。GRU最关键的一个步骤是“更新记忆”在这个阶段，GRU同时进行了遗忘了记忆两个步骤。我们使用了先前得到的更新门控（update gate）。更新表达式：。门控信号（这里的）的范围为0~1。门控信号越接近1，代表记忆下来的数据越多，而越接近0则代表遗忘的越多。GRU使用同一个门控同时进行遗忘和选择记忆，因而比LSTM要快，效果往往也更好。

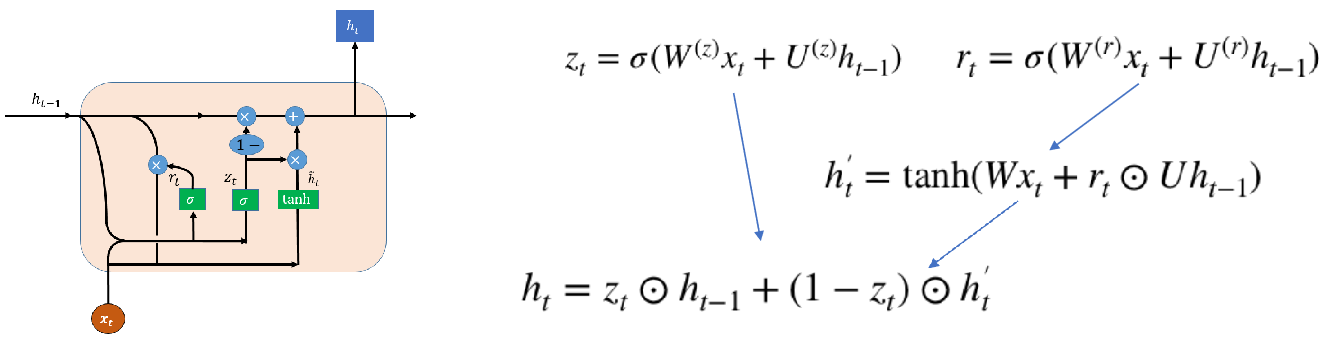


图 27 GRU的模型结构

# 六、实验器材（设备、元器件）：

|  |  |
| --- | --- |
| 硬件平台 | PC端 |
| 开发环境 | 操作系统Windows10 |
| 测试环境 | Python3.9.6+Pytorch1.12.0+cuda11.6 |

# 七、实验步骤：

注：该红色部分的文字在最终版需要删除

1. 网络框架结构搭建介绍
   1. 编码器（Encoder）网络架构

如图 28所示，对于翻译句子对中的每一个法语单词，编码器为其生成一个256维的Embedding向量，将该向量作为输入传入RNN网络。隐藏层的维度也为256，因而RNN网络输出一个256维的向量。编码器将这个向量输出，同时输出一个256维的隐藏层向量。



图 28 Encoder中的RNN结构

* 1. 解码器（Decoder）网络架构

如图 29，对于翻译句子对中的每一个英语单词，解码器为其生成一个256维的Embedding向量，将该向量作为输入传入RNN网络。隐藏层的维度也为256，因而RNN网络输出一个256维的向量。然后将该输出通过一个MLP映射到一个1093（数据集中英语单词的总个数）维的向量，并使用softmax函数，得到解码器的最终输出。解码器还同时输出了RNN的256维隐藏层向量。



图 29 Decoder中的RNN结构

* 1. 整体网络框架

如图 30，对于翻译句子对中的每一个法语单词，输入Encoder，将Encoder的隐藏层输入到Decoder的隐藏层中，同时将翻译句子对中的每一个英语单词输入的Decoder中，Decoder的输出作为预测。

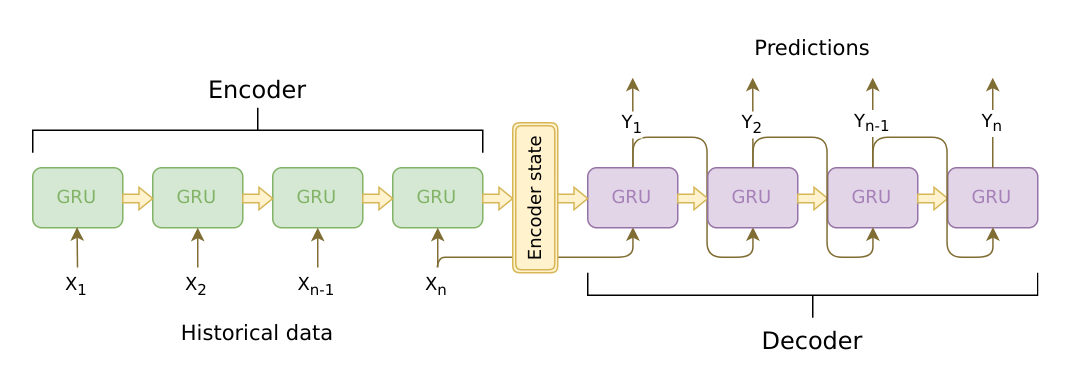


图 30 整体网络框架结构

1. 核心代码介绍

|  |
| --- |
| **输入：**训练集eng-fra  神经网络Encoder  神经网络Decoder  Adam优化器optim  NLLloss损失函数criterion  **过程：**   1. 为数据集中的每一个法语单词和英语单词设置索引 2. for n\_iters 1 to 55000 3. 从数据集中随机选取一个法语-英语句子对 4. 将句子对中每一个法语单词传入Encoder,得到e\_output和e\_hidden 5. 将句子对中每一个英语单词和e\_hidden传入Decoder,得到d\_output和d\_hidden 6. losscriterion(decoder\_output,target) 7. BP算法更新Encoder和Decoder的权重w和偏置b 8. end for 9. 从数据集中随机选取10个句子对进行测试   **输出：**loss,折线图,测试结果 |

1. 训练执行流程介绍

总共训练55000次。每次训练都计算训练集上的损失，每训练5000次打印一次训练结果（包括损失值，完成进度，时长等），每100次记录一次loss,在训练完毕后绘制对应的loss曲线图。

# 八、实验数据及结果分析：

1. 数据介绍：

本次实验使用eng-fra数据集，共有39365对英语-法语句子对，在限定句子最大长度为10后，共整理得到1271个句子对，共有1418个法语单词，1093个英语单词。每次训练或者测试都从1271个句子对中，随机选择一对，并将这对句子按照每个单词的编码进行向量化处理。

1. 结果分析：

测试的时候，从pairs中随机选择10对，通过比较网络翻译和真实翻译来评估网络的效果。

* 1. 使用RNN训练结果：

> il est innocent du crime .

= he is innocent of the crime .

< he is constantly complaining . <EOS>

> je te donne ce que tu veux .

= i m giving you what you want .

< i m glad to see you re happy . <EOS>

> je suis desolee d avoir rompu ma promesse .

= i m sorry i broke my promise .

< i m sorry if i disturbed you . <EOS>

> elle planifie un voyage .

= she is contemplating a trip .

< she is anxious to meet you . <EOS>

> il est toujours en retard a l ecole .

= he is always late for school .

< he is constantly complaining . <EOS>

> elle est beaucoup plus lourde que lui .

= she s much heavier than him .

< she is anxious to meet you . <EOS>

> je m habitue a ce mode de vie .

= i am accustoming to this life style .

< i m not who you think i am . <EOS>

> je crains que nous ne puissions vous aider .

= i m afraid we can t help you .

< i m not who you think i am . <EOS>

> nous appartenons tous a la meme equipe .

= we re all on the same team .

< we re all working together . <EOS>

> je suis gare au coin de la rue .

= i m parked around the corner .

< i m not who you think i am . <EOS>

这10个测试样本中没有一句话是完全翻译正确的，但是对于一些简答的单词（比如I am,he s,we re）等还是能够翻译正确。图 31为RNN训练时的loss曲线图，可以看到最后的loss在0.7左右，因而测试时翻译的准确度不高。

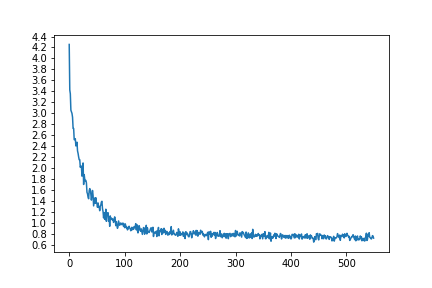


图 31 RNN训练时的loss

* 1. 自己搭建的GRU测试结果：

> je romps ce soir avec ma petite copine .

= i m breaking up with my girlfriend tonight .

< i m breaking up with my girlfriend tonight . <EOS>

> elle se plaint toujours de son travail .

= she is always complaining of her job .

< she is always complaining of her job . <EOS>

> j ai de bonnes relations avec mes voisins .

= i m on good terms with the neighbors .

< i m on good terms with the neighbors . <EOS>

> je suis sure que tom avait ses raisons .

= i m sure tom had his reasons .

< i m sure tom had his reasons . <EOS>

> je n ecarte pas cette possibilite .

= i m not discounting that possibility .

< i m not discounting that possibility . <EOS>

> elle est habituee a se lever tot .

= she s accustomed to getting up early .

< she s accustomed to getting getting ready for tomorrow .

> je vais reflechir a ca .

= i m going to figure this out .

< i m going to figure this out . <EOS>

> je requiers votre aide .

= i m asking you for your help .

< i m asking you for your help . <EOS>

> elle est absorbee par son travail d etude .

= she is absorbed in her study .

< she is absorbed in her study . <EOS>

> il est tres impatient d y aller .

= he is very eager to go there .

< he is deeply attached to this bicycle . <EOS>

可以看到自己搭建的GRU网络在测试的10个样本中有8个是翻译正确的，估计其整体正确率在80%左右。该网络在训练过程中的loss如图 32所示，该网络在训练最终的loss大概为0.3。从训练的loss和测试集上的表现均可以明显看出GRU优于传统的RNN网络，GRU的门控制单元结构能够有效捕捉长序列之间的语义关联, 缓解梯度消失或爆炸现象，因而其表现要优于传统的RNN网络。

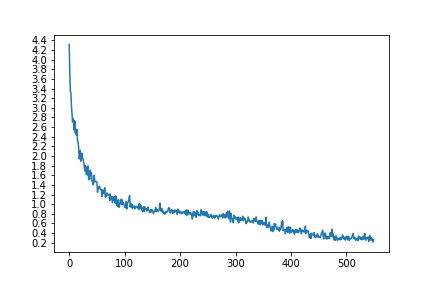


图 32 自己搭建的GRU训练loss

* 1. 使用pytorch提供的GRU模型的训练结果

> demain je ne vais pas au cinema .

= i m not going to the movies tomorrow .

< i m not going to the movies tomorrow . <EOS>

> je pense apprendre le coreen le semestre prochain .

= i m thinking of learning korean next semester .

< i m thinking of learning korean next semester . <EOS>

> il entreprend des experiences dans son laboratoire .

= he is carrying out experiments in his laboratory .

< he is carrying out experiments in his laboratory . <EOS>

> je refuse de te vendre ma voiture .

= i m not selling you my car .

< i m not selling you my car . <EOS>

> je ne suis pas bon pour negocier .

= i m not good at negotiating .

< i m not good at negotiating . <EOS>

> je suis excite a l idee du demenagement .

= i m excited about the move .

< i m excited about the move . <EOS>

> je ne vais pas citer de noms .

= i m not going to name names .

< i m not going to name names . <EOS>

> je suis desolee si je vous ai derangee .

= i m sorry if i disturbed you .

< i m sorry if i disturbed you . <EOS>

> je crains que nous ayons un probleme .

= i m afraid we have a problem .

< i m afraid we have a problem . <EOS>

> je commence a me sentir deja mieux .

= i m starting to feel better already .

< i m starting to feel better already . <EOS>

可以看到pytorch提供的GRU网络在测试的10个样本中全部翻译正确，该网络在训练过程中的loss如图 33所示，该网络在训练最终的loss非常接近0。相比于自己搭建的GRU网络，pytorch内置的GRU模型有更多的优化模块，能够快速的让loss下降，达到一个较好的效果。查询pytorch内置GRU网络结构，发现pytorch中GRU的r和z的权重为经过MLP得到的维度为隐藏层维度的向量，而自己设计的GRU得到的维度为1。这导致权重丢失大量的信息，因而GRU的记忆和遗忘功能不能很好的体现。

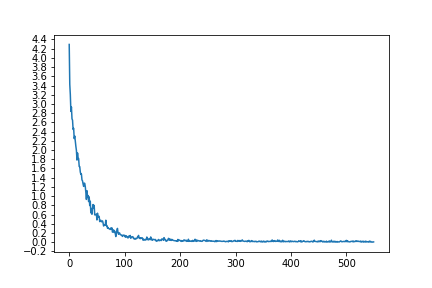


图 33使用pytorch提供的GRU模型的训练loss

# 九、总结及心得体会：

# 十、对本实验过程及方法、手段的改进建议及展望：

**实验四**

# 一、实验室名称：

电子科技大学清水河校区主楼A2-413

# 二、实验项目名称：

人工智能实验II-4：基于自动编码器的手写数字生成

# 三、实验原理：

自动编码器是无监督学习方法中的一种结构。它通过编码器提取输入数据的隐含特征，再通过解码器根据隐含特征重构输入，实现了一种自监督学习的方法。其中，编码器和解码器通常为一个非线性映射函数，隐含特征往往比输入数据更加紧凑。因此，自动编码器能够实现数据降维、压缩和数据隐含分布的投影。随着深度神经网络的广泛应用，人们尝试使用神经网络搭建编码器和解码器，并取得了广泛成功。本次实验基于神经网络搭建自动编码器，以探究其在隐变量映射和图像降维与还原任务中体现出的各种特性。

# 四、实验目的：

通过使用 PyTorch 深度学习框架，搭建神经网络，实现自动编码器以进行手写数字生成等任务，并在此过程中掌握：

1. PyTorch 中全连接网络、卷积网络等的实现方法；
2. 自动编码器的基本框架和设计思路；
3. 探究自动编码器在图像降维和还原中体现的具体功能。

扩展掌握：

1. 自动编码器的隐变量可视化分析；
2. 变分自编码器的实现方式；
3. 对抗生成网络的实现方式。

# 五、实验内容：

* 1. 问题描述

设计自动编码器（AutoEncoder）生成MNIST和CIFAR数据集图像

* 1. 算法的概要设计与分析

AutoEncoder：将数据集中的数据输入Encoder中进行编码，然后将Encoder的结果输入的Decoder中进行解码，生成图片。Loss为生成图片和原图片基于loss函数的差异，然后通过反向传播算法更新网络参数。

VAE：将数据集中的数据输入Encoder编码为k个均值和k个方差，然后将k个对应的正态分布输入的Decoder中进行解码，生成图片。Loss为生成图片和原图片基于loss函数的差异加上0.1权重的先验分布和变分后验分布的klloss，然后通过反向传播算法更新网络参数。

GAN：由生成器和判别器“左右互博”。我们把伪造的图片和真实的图片交给判别器，并训练判别器辨认哪些图片是伪造的，哪些是真实的。生成器则恰恰相反，它根据判别器的判断来调整自己，尽可能去欺骗判别器。具体操作时：根据随机噪声，生成器生成一组伪造图片。把伪造图片和真实图片交给判别器，并告诉它真实标签，训练进行二分类，判断图片是“真”还是“假”。生成器再生成一组图片，同样交给判别器，但这时我们训练去干扰的分类结果。

MS-SSIM算法：用于衡量两幅图像之间的差异。该算法相似度的测量可由三种对比模块组成，分别为：亮度，对比度，结构。在本实验中用MS-SSIM算法来比较自编码器生成图片和原始图片之间的差异从而来评价生成图片的质量。

1. 核心算法的详细设计与实现

AutoEncoder:如图 34所示，AutoEncoder分为encoder和decoder两部分。Encoder对数据进行编码,通常先提取特征，然后对特征进行编码。常用的特征提取方式有线性层网络（MLP）和卷积神经网络（CNN），特征提取后通常用全连接层（FC）对特征进行编码。Decode根据Encoder决定解码方式。如果采用线性层网络，则Decoder也用线性层网络进行解码。如果Encoder采用卷积网络，则Decoder通过反卷积（ConvTranspose）对特征进行解码。

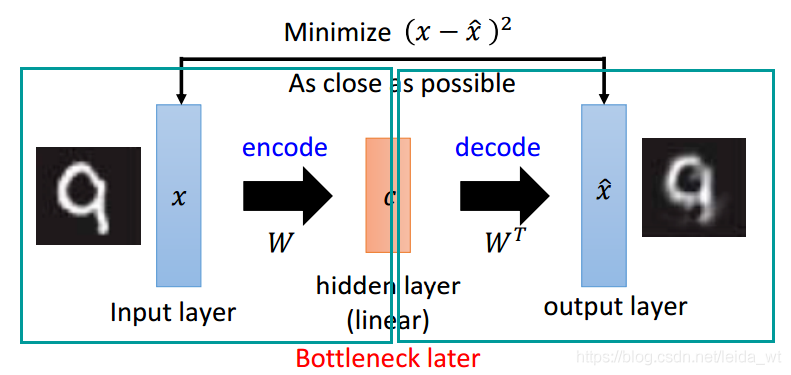


图 34 AutoEncoder结构图

反卷积操作如图 35所示。首先，对输入的feature map进行padding操作，得到新的feature map；然后，随机初始化一定尺寸的卷积核；最后，用随机初始化的一定尺寸的卷积核在新的feature map上进行卷积操作。当s=1时，对于原feature map不进行插值操作，只进行padding操作，这里的padding是卷积操作时设置的padding值，经过padding之后，再经过卷积核进行卷积操作就得到了输出结果。当s>1时，进行插值操作，feature map的元素之间被分隔开了，间隔(s-1)个插入元素0。插值之后的操作同s=1的操作一致。

|  |  |
| --- | --- |
| * 1. S=1 | * 1. S>1 |

图 35 反卷积（ConvTranspose）示意图

Decoder最后要使用Sigmoid函数，将输出固定在0~1之间，这是因为输出的是图像，其像素点大小需要在0~1之间。损失为Decoder生成的图片和真实图片之间的差异，常见的损失函数有MSEloss,binary\_cross\_entropy等。计算出损失后，使用反向传播算法同时对Encoder的参数和Decoder的参数进行更新。

VAE：变分自编码器。它能够学习一个函数/模型，使得输出数据的分布尽可能的逼近原始数据分布。相比AE，VAE不再把分布映射到固定值或变量上，而是映射到一个分布上。如图 36，网络模型中的bottle neck被分解成了两个向量：均值向量方差向量。VAE的Encoder用来计算均值和方差，均值和方差是用来描述正态分布的数字特征，不同的和取值对应不同位置、不同形状的正态分布。这里的均值和方差并不是确定的统计量而是两个函数，是由神经网络训练得到的。其中后验分布是先验分布的近似；是服从标准多元高斯分布的标准化随机变量，是服从高斯分布的随机变量，称为潜在变量或隐变量；是似然分布，即知道变量的分布后根据来估计的分布。当解码时，我们将从每个潜在分布中随机采样，生成一个向量作为解码器模型的输入。由于有了每个属性的概率分布，我们可以简单地从分布中抽取任何值来生成一个新的输出，而且自带过渡效果。

VAE是一种无监督的生成模型，其理论基础是建立在GMM高斯混合模型之上。GMM认为任何一个数据的分布，都可以看作是若干高斯分布的叠加。是由一个标准正态分布所产生的向量，的每一维度都代表了一种属性，且的每一个属性的分布都服从高维高斯分布，通过每一个属性的高维高斯分布来拟合输入样本的真实分布、对于每个采样，会有两个函数和，分别对应到高斯分布的均值和方差，然后在积分域上所有的高斯分布的累加就得到

原始分布的近似，即，整个网络就是把样本经过

降维表达成，再由根据低维特征重构原始输入，对应了一个完整的Encoder-Decoder过程。

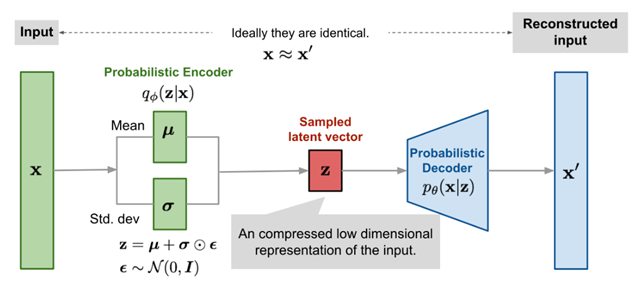


图 36 VAE结构图

我们的目的就是求解的分布。由于是已知的，未知而，于是我们真正要求解的是和两个函数的表达式。通常非常复杂，因而引入两个神经网络进行求解：第一个神经网络是Decoder，用于求解和两个函数，等价于求解。第二个神经网络是Encoder，用于求解，可以表示任何的分布。为了让尽可能接近，我们引入KL散度。KL散度是两个分布之间差异的度量其公式为：



通过最小化KL散度，使两个分布尽可能地相似。因而模型的损失函数除了binary\_cross\_entropy之外，还要加上KL散度，这样最小化损失函数既能是让生成的图片跟真实图片接近又能使生成的分布与原分布尽可能接近，从而提高模型的鲁棒性。

MS-SSIM算法：用于衡量两幅图像之间的差异。该算法相似度的测量可由三种对比模块组成，分别为：亮度，对比度，结构。亮度对比函数为：



其中，为平均灰度，常数是为了避免接近接近0时造成系统的不稳定。通常取

对比度函数为：



其中，为灰度标准差，常数是为了避免接近接近0时造成系统的不稳定。通常取

# 六、实验器材（设备、元器件）：

|  |  |
| --- | --- |
| 硬件平台 | PC端 |
| 开发环境 | 操作系统Windows10 |
| 测试环境 | Python3.9.6+Pytorch1.12.0+cuda11.6 |

# 七、实验步骤：

注：该红色部分的文字在最终版需要删除

网络框架结构搭建介绍

**给出网络设计及核心代码实现，如：**

* 1. **网络类型（全连接或卷积？）；**
  2. **网络每一层的输入输出维度；**
  3. **网络每一层的可学习参数维度等。**

训练执行流程介绍

# 八、实验数据及结果分析：

1. 实验数据

MNIST数据集一共有70000张图片，其中60000张是训练集，10000张是测试集。每张图片是2828的0-9的手写数字图片组成。每个图片是黑底白字的形式，黑底用0表示，白字用0-1之间的浮点数表示，越接近1，颜色越白。如图 37所示。

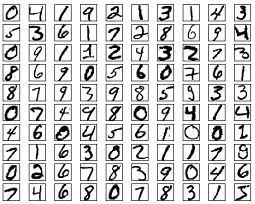


图 37 MNIST数据集

CIFAR-10 是一个包含60000张图片的数据集。如图 38 CIFAR10数据集，其中每张照片为32\*32的彩色照片，每个像素点包括RGB三个数值，数值范围 0 ~ 255。所有照片分属10个不同的类别，分别是 'airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck'。其中训练集有50000张图片，测试集有10000张图片。



图 38 CIFAR10数据集

2. 结果分析

如图 39为使用线性层AutoEncoder生成结果和loss变化曲线。使用了Adam优化器，学习率为1e-3，一共训练了10000次，最终的loss为0.0995。

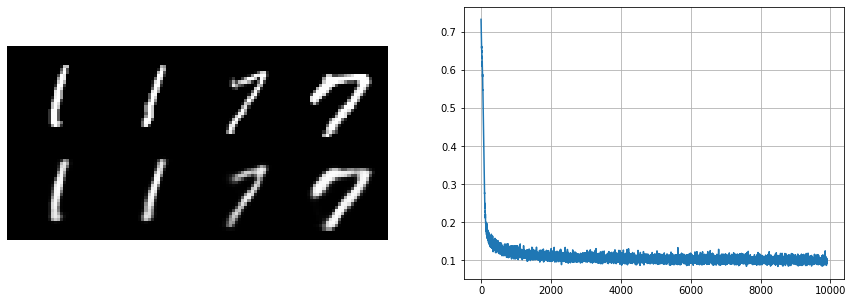


图 39 线性层AutoEncoder生成结果及loss曲线

图 40为使用卷积层AutoEncoder生成结果和loss变化曲线。使用了Adam优化器，学习率为1e-3，一共训练了10000次，最终的loss为0.0678，相比于线性层AutoEncoder,卷积层AutoEncoder生成图片效果更好，最终的loss也更低，并且从loss图中可以看出，卷积层AutoEncoder的loss下降更快，后面的波动也相对更平缓一些。

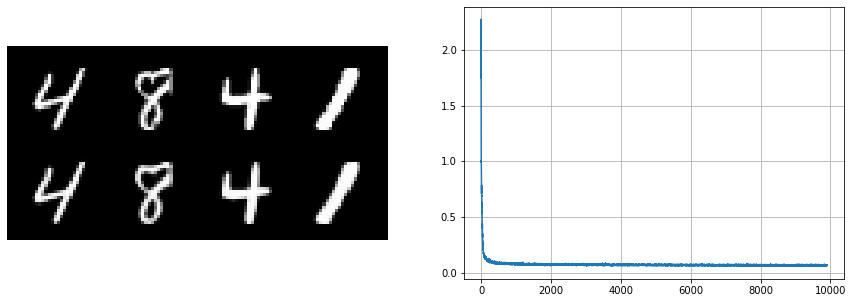


图 40 卷积层AutoEncoder生成结果及loss曲线

上面两个实验均使用了binary\_cross\_entropy作为损失函数，图 41为使用MSELoss后，卷积层AutoEncoder生成结果和loss变化曲线。仍然使用Adam优化器，学习率为1e-3，一共训练了10000次，最终的loss为0.0029。相比于使用binary\_cross\_entropy作为损失函数的AutoEncoder，使用MSELoss生成图片效果更好，最终的loss也更低。这主要AutoEncoder通过生成图片与训练图片之间的loss来更新网络参数。binary\_cross\_entropy更适合于分类任务，也就是它能更好的区分两个样本是否属于同一类。而MSELoss通过L2范数计算生成图片和训练图片之间的差异。显然，直接计算二者的差异比判断二者是否属于同一类的要求和精度会更高，因而使用MSELoss能够使网络参数更快，更好地更新，训练的效果也更好。

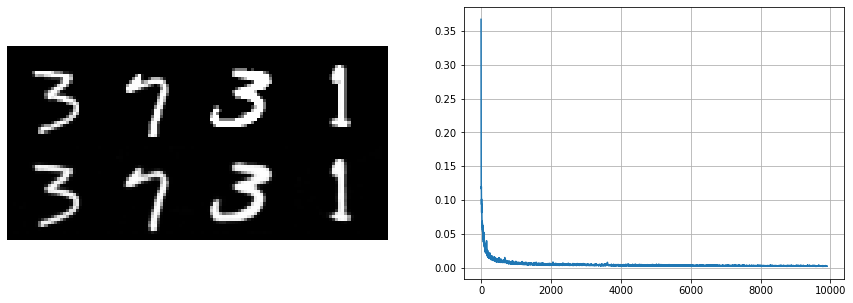


图 41 使用MSELoss卷积层AutoEncoder生成结果及loss曲线

图 42为卷积层AutoEncoder在CIFAR10生成结果及loss曲线，用Adam优化器，学习率为1e-3，一共训练了10000次，最终loss为0.0026。损失函数使用MSELoss。相比于MNIST数据集，CIFAR10的图片是3通道的，因而训练难度更大，生成的图片效果也比MNSIT数据集要差。

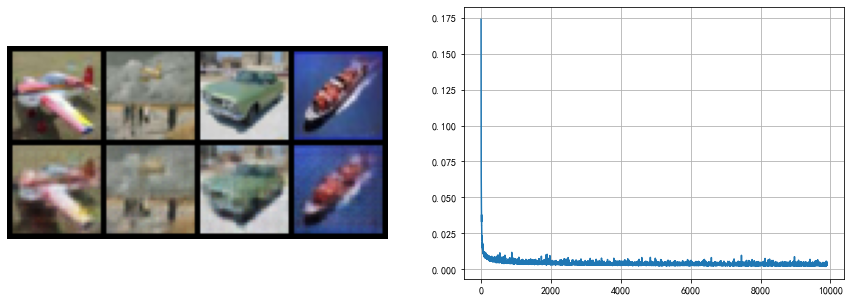


图 42 卷积层AutoEncoder在CIFAR10生成结果及loss曲线

将测试集的所有样本都输入到AutoEncoder中，对其生成的结果作t-SNE,并进行可视化，如图 43所示。a)为线性层AutoEncoder的结果，b)为卷积层AutoEncoder的结果，对比两个子图可以发现使用卷积层AutoEncoder相同类别样本间的距离更近，不同类别样本间的距离更远。这说明卷积层AutoEncoder对数据集特征提取的效果更好，与上文的结论吻合，及卷积层AutoEncoder的效果要好于线性层。

|  |  |
| --- | --- |
| a)线性层AutoEncoder | b)卷积层AutoEncoder |

图 43 AutoEncoder的t-SNE结果

图 44为AutoEncoder生成的10个类别手写数字。a)图为线性层AutoEncoder的结果，b)图为卷积层AutoEncoder的结果，可以直观的看出，卷积层生成的图片清晰图要高于线性层。这一结果与上文结果十分吻合。

|  |  |
| --- | --- |
| a)线性层AutoEncodere | b)卷积层AutoEncoder |

图 44 AutoEncoder生成的手写数字

图 45为VAE生成结果及loss曲线，使用了Adam优化器，学习率为1e-3，一共训练了20万次。最终的loss为0.1173。可以看到VAE生成的图片质量不如AutoEncoder,最终的loss也比AutoEncoder要大。这是因为VAE的Encoder将图片编码为k个均值和k个方差。VAE假设数据服从正太分布，因而将k个正态分布输入到VAE的Decoder中，正态分布的均值和方差对应Encoder的编码结果。相比与直接作反运算的AutoEncoder，VAE的训练难度更大，因而其结果也比AutoEncoder要差。

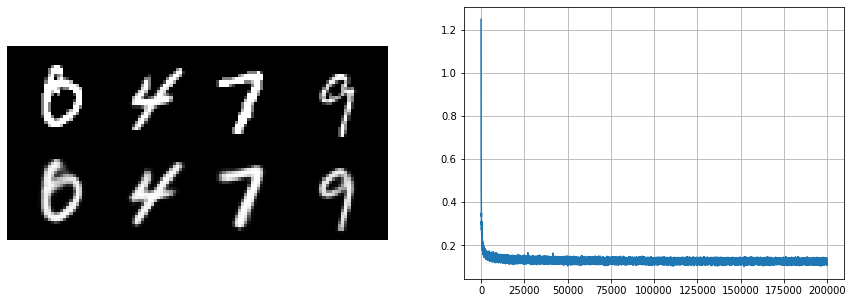


图 45 VAE生成结果及loss曲线

既然Decoder的输入通过采样得来，同时我们已经促使我们的输入所处的分布为‘K’个标准正态分布的混合。因此，我们直接从标准正态分布中随机采样 ‘K’个值，就能当作编码 输入解码器了。图 46为VAE生成结果，此处取K=10。可以看到生成的10个样本基本可以辨识出具体数字，训练出来的VAE效果较优。

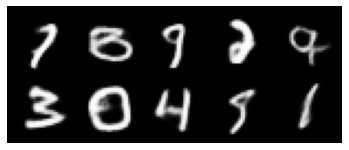


图 46 VAE生成结果

图 47为GAN生成结果及loss曲线。正中间的图为判别器（discriminator）的loss变化曲线，右图为生成器（generator）的loss变化曲线。两个loss都有较大幅度的振荡，这正是生成器和判别器“左右互博”的结果。最理想的结果为生成器的结果让判别器无法辨别，即生成器生成的图片，判别器有0.5的概率预测为真实的图片，有0.5的概率预测为假的图片。此时生成器的loss大概为左右，而判别器的loss大概为生成器的2倍，这是因为由TP和TF两部分构成。生成的图片和抽样的图片完全不一样，这是因为生成器只提取了样本的尺寸（size），我们并没有将抽样图片像素信息传入给生成器，因而生成器生成的图片与抽取的样本并无关系。生成器生成了图片“1978”，较清晰，较真实，效果较好，较难辨别真伪。不过，随着训练次数的增加，生成器生成的种类明显减少，且更频繁生成构成较为简单的数字（如1,7等），这大概是训练的时候loss陷入了局部极小点，生成器发现生成1,7这样简单的数字能更好地逃过判别器的辨识。

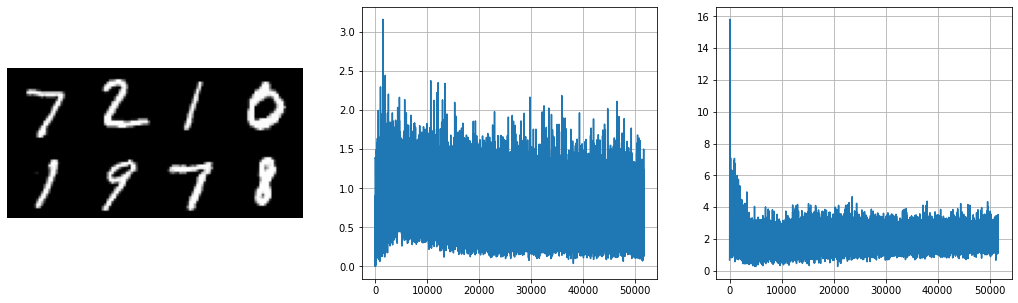


图 47 GAN生成结果及loss曲线

汇总了各个自编码器的MS-SSIM的结果，并由高到低进行了排序，如图 48所示。除了模型括号中的方式外，其余均使用默认方式，默认方式如表 4所示。

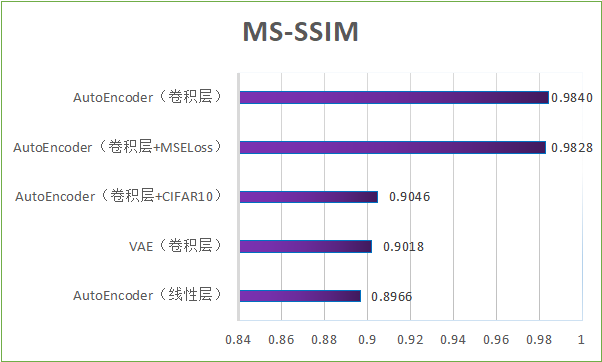


图 48 MS-SSIM条形图

表 4 各个自编码器默认方式

|  |  |
| --- | --- |
| 方式 | 默认形式 |
| 数据集 | MNIST |
| 优化器 | Adam |
| 学习率 | 1e-3 |
| 损失函数 | Binary\_cross\_entropy |
| 训练轮数 | 10000 |
| Batch\_size | 32 |

对于MS-SSIM算法，当计算值越接近1时，表明生成的图片与原始图片越相近，从而该自编码器的效果也越好。从图 48可以看出，AutoEncoder（卷积层）的效果最好，然后依次为AutoEncoder（卷积层+MSEloss），AutoEncoder（卷积层+CIFAR10），VAE（卷积层）。AutoEncoder（线性层）的效果最差。这一结果与前文的分析基本一致，进一步验证了使用卷积运算提取特征后能够显著提升AutoEncoder的效果。

# 九、总结及心得体会：

# 十、对本实验过程及方法、手段的改进建议及展望：

**报告评分：**

**指导教师签字：**