MLVC课程笔记

L.C.

重点

1. 线性拟合(Linear Fitting)、优化(Optimization)和正规方程(NE)

1. 反向传播(Backpropagation), 链式法则(Chain Rule), 位置编码(Position Encoding)

1. 感知机、多层感知机(MLP)及优化、过拟合(Overfitting)

1. 卷积、卷积核(Kernel)、池化(pooling)和滤波(Filter)

1. 卷积神经网络(CNN), 全连接层(Fully-connected layer), 经典架构(AlexNet/ResNet/DenseNet)

1. 反向传播BP, CAM, 和Grad-CAM、权重

1. 风格迁移(Style Transfer), 纹理合成(Texture Synthesis), Gram Matrix，损失函数(L2, MAE, SSIM, MSE, LPIPS), 对比(mean-var)

1. 循环神经网络(RNN)流程原理及问题, LSTM(Chain), Softmax

1. VAE (变分自编码器)和生成对抗网络(GAN)以及对比

1.5. 激活函数(Activation Functions), XOR, 神经网络(NN)

1.5. 梯度下降(GD, batch-GD)与随机梯度下降(SGD)

1.5. 数据划分(Training/Validation/Test)

1.5. 矩阵计算、MSE损失、最小二乘和正则化(LS and Regular)、线性回归(Linear Regression)

1.5. 序列学习(Sequence Learning), 序列模型映射(4种)

1.5. 注意力机制(Attention Mechanism), RNN+Attention

1.5. 自回归模型(Autoregressive Models), 神经隐式表示 (Neural Implicit Representation)

2. 监督学习 vs 无监督学习(Supervised)

2. 特征反演(Feature Inversion), 梯度上升(Gradient Ascent)、正则化，DeepDream

2. Neural Texture Synthesis, 前馈网络(Feedforward Network)

2. 主成分分析(PCA), 自编码器(Autoencoder), 生成模型, 贝叶斯法则

2.5. 扩散过程(Diffusion Process-training and sampling), 高斯, Loss函数, 扩散模型(Diffusion Model)

2.5. U-Net

2.5. Transformer

2.5. 图像检索 (Image Retrieval)-RGB距离\直方图距离\VGG\GRAM

2.5. 降噪(Denoising): 高斯/CNN

3. 特征空间(最近邻和降维)、Visualizing Activation, Saliency

3. Image Pyramid / multiresolution / Features

3. Semantic Tasks

3. 图像、视频、音频和3D表面(光场和视差)表示(Representation), 视觉数据

4. LDM

4. 多项式拟合

复习课

让我从第一章开始讲解 - 优化(Optimization)。

这章介绍了机器学习中的基础优化概念:

1. **Images as Points (图像作为点)**

- 图像可以被看作高维空间中的点
- 一个n×m像素的图像可以表示为nm维向量空间中的一个点
- 这种表示方法使我们能够用数学方法处理图像

2. **Nearest Neighbors (最近邻)**

- 通过计算点之间的距离来找到最相似的样本
- 常用距离度量:
  - 欧氏距离: $d(x, y) = \sqrt{\sum_i (x_i - y_i)^2}$

    - 曼哈顿距离: $d(x,y) = \sum_i |x_i - y_i|$
- K-NN算法是基于这个概念的简单但有效的分类方法

3. **Basic Optimization (基本优化)**

- 目标:找到使损失函数最小的参数
- 通用形式: $\min_\theta L(\theta)$
- 关键步骤:
    - 定义目标函数
    - 计算梯度
    - 迭代更新参数

4. **Normal Equation (正规方程)**

- 用于线性回归的直接解法
- 公式: $\theta = (X^T X)^{-1} X^T y$
- 优点:一步得到最优解
- 缺点:计算复杂度高,不适用于大规模数据

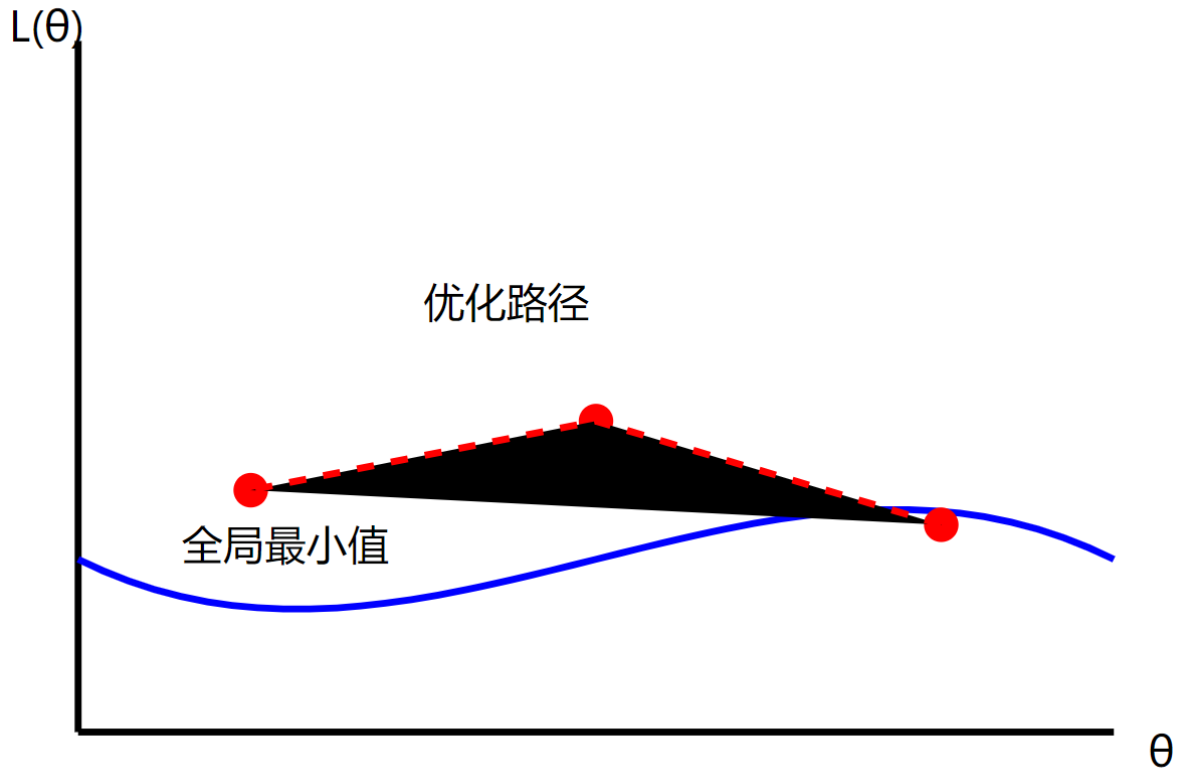
5. **Linear Classifier (线性分类器)**

- 通过超平面将不同类别的数据分开
- 决策边界: $w^T x + b = 0$
- 预测函数: $f(x) = sign(w^T x + b)$

6. **Regularization (正则化)**

- 防止模型过拟合的技术
- 常用方法:
    - L1正则化: $L_1 = \lambda \sum_i |w_i|$
    - L2正则化: $L_2 = \lambda \sum_i w_i^2$

如图展示了典型的优化过程,其中:



- 蓝色曲线表示损失函数曲面
- 红色点和虚线表示优化过程中参数的更新路径
- 最低点表示全局最优解

第二章讲解 - SGD + MLPs（随机梯度下降和多层感知器）

1. **Gradient Descent (梯度下降)**

- 基本公式：$\theta_{t+1} = \theta_t - \alpha \nabla L(\theta_t)$
- 学习率α控制更新步长
- 梯度$\nabla L(\theta_t)$指向函数值增加最快的方向

2. **SGD (随机梯度下降)**

- 每次使用小批量数据计算梯度
- 更新公式：$\theta_{t+1} = \theta_t - \alpha \nabla L(\theta_t; x_i)$
- 优点：
  - 计算效率高
  - 有助于跳出局部最小值
  - 内存需求小

3. **XOR Problem (异或问题)**

- 经典的非线性分类问题
- 输入：(0,0)→0, (0,1)→1, (1,0)→1, (1,1)→0
- 单层感知器无法解决
- 需要至少一个隐藏层的MLP

4. **MLPs (多层感知器)**

- 网络结构：输入层→隐藏层→输出层
- 前向传播公式：
  $h = \sigma(W_1x + b_1)$
  $y = \sigma(W_2h + b_2)$
- 反向传播使用链式法则计算梯度

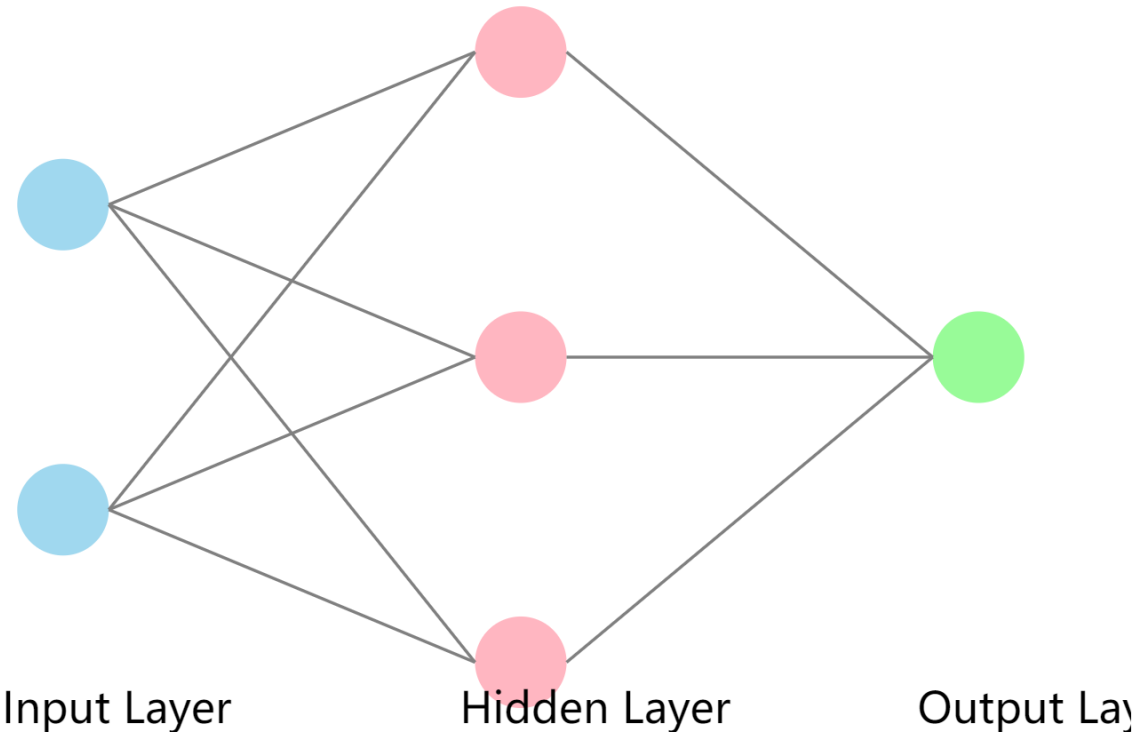
5. **Activations (激活函数)**
   主要激活函数：

- Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$
- ReLU: $f(x) = max(0, x)$
- Tanh: $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

6. **Positional Encoding (位置编码)**

- 用于序列数据的位置信息编码
- 正弦余弦函数编码：
  $PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$
  $PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$

这个图展示了典型的MLP结构：



- 蓝色节点：输入层
- 粉色节点：隐藏层
- 绿色节点：输出层
- 灰色线条表示层间连接和权重

重点需要掌握：

1. **SGD的原理和优势**
   SGD(随机梯度下降)是一种迭代优化算法。每次从训练数据中随机抽取一小批样本来计算梯度，而不是使用全部数据。想象你在爬山，但是闭着眼睛 - 你每次只能感知脚下的一小块区域来决定下一步往哪走。这就是SGD的工作方式。

其主要优势在于：

- 计算效率高：每次只需处理一小部分数据
- 有助于逃离局部最优：随机性可能帮助跳出局部最小值陷阱
- 内存友好：不需要一次加载所有数据到内存中

2. **MLP的基本结构和前向传播过程**
   MLP就像一条信息加工的流水线。假设我们在分类猫狗图片：

- 输入层接收原始数据（比如图片像素）
- 隐藏层进行特征提取和转换（比如识别边缘、纹理等）
- 输出层给出最终预测（比如猫的概率是0.8，狗的概率是0.2）

前向传播过程：

```
# 假设一个简单的两层MLP
h = activation(W1 @ x + b1)  # 隐藏层
y = activation(W2 @ h + b2)  # 输出层
```

3. **激活函数的特点和应用场景**
   激活函数就像神经元的"开关"，决定信息如何传递：

- ReLU：最常用，计算快速，适合深层网络
- Sigmoid：输出范围(0,1)，适合二分类问题
- Tanh：输出范围(-1,1)，适合数据标准化

4. **XOR问题需要多层网络的原因**
   XOR问题是非线性可分的。想象一个平面上的四个点：

```
(0,0) → 0
(0,1) → 1
(1,0) → 1
(1,1) → 0
```

无法用一条直线将输出为1的点和输出为0的点分开。需要至少一个隐藏层来创建非线性决策边界。

5. **位置编码在序列数据处理中的作用**
   想象你在阅读一个句子："我喜欢吃苹果"。每个词的含义不仅取决于词本身，还取决于它在句子中的位置。位置编码就是给每个位置赋予一个独特的表示，帮助模型理解序列中的位置信息。

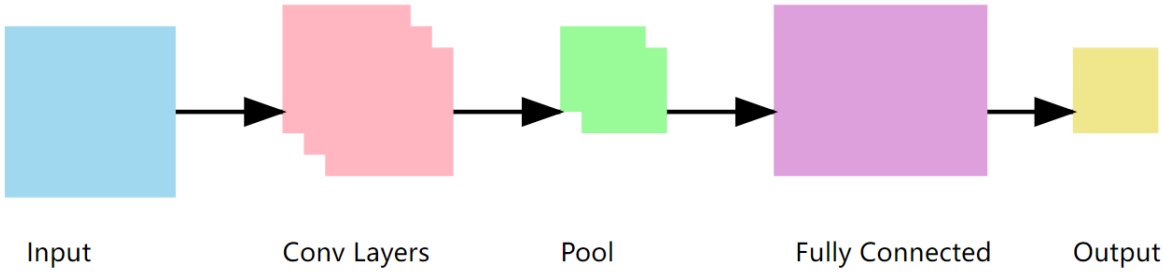
数学公式：
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

这些正弦余弦函数能够编码相对位置信息，使得模型能够理解序列中元素的顺序关系。

这些概念相互关联，共同构成了深度学习的基础。在实际应用中，它们往往是同时使用的。比如，一个使用MLP的图像分类系统会同时用到SGD进行训练，用ReLU作为激活函数，可能还会在某些层使用位置编码来处理序列信息。

让我深入讲解第三章 - Images and CNNs（图像和卷积神经网络）。

1. **CNNs (卷积神经网络)**
   卷积神经网络是专门为处理网格结构数据（如图像）设计的深度学习架构。它的主要特点是使用卷积运算来提取特征，这种方式能够很好地保持数据的空间结构信息。



2. **Convolutions (卷积操作)**
   卷积操作是CNN的核心，它通过一个小的滑动窗口（卷积核）来扫描图像：

```
# 二维卷积的基本操作
output[i,j] = sum(input[i+m,j+n] * kernel[m,n])
```

滤波器（卷积核）的大小通常为3×3或5×5，通过与输入图像的局部区域进行点积运算，可以提取不同类型的特征。

3. **Learned Filters (学习的滤波器)**
   在CNN中，滤波器的权重是通过反向传播学习得到的。不同的滤波器可以学习检测不同的特征：

- 浅层滤波器：边缘、纹理、颜色
- 深层滤波器：更复杂的模式、物体部件

4. **Network Architecture (网络架构)**

CNN的网络架构是一个精心设计的层次结构，就像建筑的blueprint。每一层都有其特定的功能：

1. 卷积层：

- 使用不同的滤波器提取特征
- 早期层检测简单特征（边缘、颜色）
- 深层检测复杂特征（物体部分、纹理模式）
- 典型配置：3×3或5×5卷积核，步长为1或2

1. 激活函数层：

- 通常使用ReLU
- 引入非线性，增强网络表达能力
- 帮助解决梯度消失问题

1. 池化层：

- 最大池化或平均池化
- 降低特征图尺寸
- 提供空间不变性

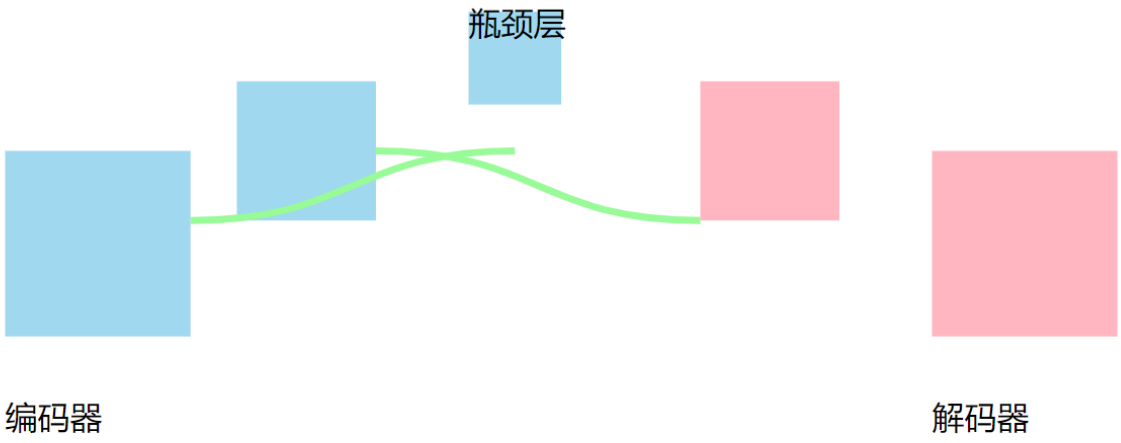
- 典型配置：2×2池化窗口，步长为2

1. 全连接层：

- 通常在网络末端
- 整合所有特征
- 执行最终的分类/回归任务

5. **UNets (U型网络)**

UNet是一种优雅的对称架构，专门为图像分割设计：



1. 编码器路径（下采样）：

- 逐步减小特征图尺寸
- 增加特征通道数
- 捕获上下文信息
- 典型结构：Conv→ReLU→Conv→ReLU→MaxPool

1. 解码器路径（上采样）：

- 逐步恢复空间分辨率
- 减少特征通道数
- 恢复位置细节
- 典型结构：UpConv→Concat→Conv→ReLU

1. 跳跃连接：

- 连接编码器和解码器对应层
- 传递细节信息
- 帮助精确定位
- 缓解梯度消失问题

6. **PCA (主成分分析)**

PCA是一种重要的降维和特征提取技术：

1. 数学原理：

- 寻找数据的主要变化方向
- 通过特征值分解协方差矩阵
- 选择最大特征值对应的特征向量
- 数据投影到新的特征空间

1. 在CNN中的应用：

- 特征可视化：
  - 可视化卷积层学到的特征
  - 理解网络的决策过程
  - 分析网络的表征学习能力
- 网络压缩：
  - 压缩卷积层权重
  - 减少模型参数量
  - 加快推理速度
  - 降低存储需求
- 特征降维：
  - 减少特征维度
  - 去除冗余信息
  - 提取主要特征
  - 加快训练速度

1. 具体实现：

```
pythonCopy# PCA简单实现
def pca(X, k):
    # 中心化
    X_mean = X - np.mean(X, axis=0)
    # 计算协方差矩阵
    cov = np.cov(X_mean.T)
    # 特征值分解
    eigenvals, eigenvecs = np.linalg.eig(cov)
    # 选择前k个主成分
    idx = eigenvals.argsort()[::-1]
    eigenvecs = eigenvecs[:, idx[:k]]
    # 投影
    return np.dot(X_mean, eigenvecs)
```

1. PCA的优缺点： 优点：

- 有效降维
- 去除噪声
- 找到主要特征方向
- 计算简单高效

缺点：

- 只能捕获线性关系
- 可能丢失重要的非线性特征
- 难以确定最佳主成分数量
- 对异常值敏感

重点需要掌握的问题：

1. **卷积运算的本质是什么？**
   答：卷积运算本质是一种局部加权求和操作，通过滑动窗口方式，用卷积核与输入的局部区域做点积运算，从而提取局部特征。这种操作能够保持空间结构信息，且具有平移不变性。
2. **CNN相比传统神经网络有什么优势？**
   答：

- 参数共享：减少参数数量
- 局部连接：捕获局部特征
- 平移不变性：位置无关的特征提取
- 层次化特征学习：从简单到复杂的特征表示

3. **池化层的作用是什么？**
   答：池化层主要有三个作用：

- 降低特征图维度，减少计算量
- 提供一定程度的平移不变性
- 防止过拟合，提高模型鲁棒性

4. **UNet架构的特点和应用场景是什么？**
   答：UNet的主要特点是对称的U形结构和跳跃连接。适用于：

- 医学图像分割
- 语义分割
- 需要精确定位的图像处理任务

5. **PCA在CNN中的应用有哪些？**
   答：PCA在CNN中主要用于：

- 特征降维，减少计算复杂度
- 可视化和分析网络学到的特征
- 网络压缩和加速

让我深入讲解第四章 - Understanding CNNs中的这两个特定主题。

CNN Features (CNN特征)

CNN的特征提取过程是逐层构建的，就像人类视觉系统一样，从简单到复杂：

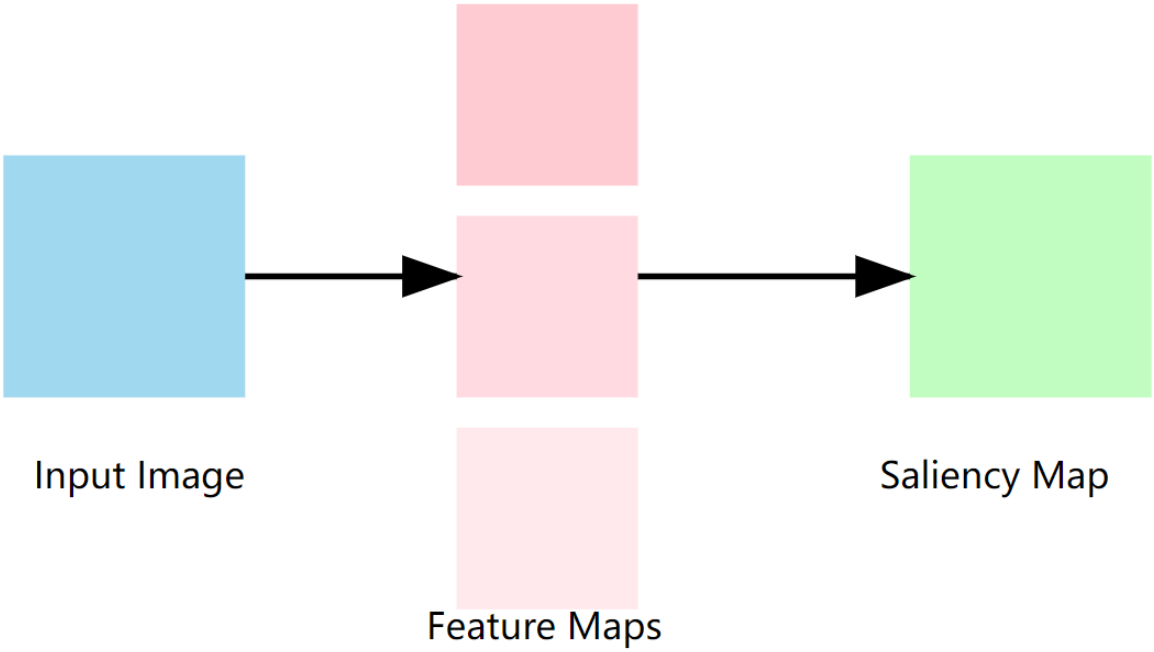
1. 层次化特征：

- 浅层特征：边缘、颜色、简单纹理
- 中层特征：纹理组合、简单形状
- 深层特征：物体部件、复杂模式

2. 特征可视化方法：

- 直接可视化卷积核
- 特征图（激活）可视化
- 最大激活分析
- 反卷积网络

举个例子：如果我们观察一个训练好的CNN在识别猫的过程中：



- 第一层可能检测到胡须的边缘
- 中间层可能组合出眼睛、耳朵的形状
- 深层可能识别出完整的猫脸特征

Saliency和Guided Backprop

这两种方法都是为了理解CNN的决策过程，就像给网络装上了"解释器"：

1. Saliency Maps（显著性图）：

- 计算输入图像对分类决策的梯度
- 显示哪些像素对决策最重要
- 公式：$S_c(x,y) = \left|\frac{\partial f_c}{\partial I(x,y)}\right|$
  其中f_c是类别c的预测分数，$I(x,y)$是输入图像的像素值。

2. Guided Backpropagation：

- 在反向传播时只保留正梯度
- 生成更清晰的可视化结果
- 结合了反卷积和普通反向传播的优点

重点需要掌握的问题：

1. **CNN在不同层级提取的特征有什么区别？为什么会有这种区别？**
   答：CNN的特征提取呈层次化特征：

- 浅层提取低级特征（边缘、颜色），因为卷积核感受野小，只能看到局部信息

- 深层提取高级特征（物体部件、语义信息），因为通过多层堆叠，感受野变大，可以整合更多上下文信息
- 中间层则是过渡状态，提取中等复杂度的特征（纹理组合、简单形状）

2. **Saliency Maps和Guided Backprop的区别是什么？各有什么优势?**
   答：两者都是可视化技术，但原理和效果不同：

- Saliency Maps计算输入对输出的梯度，显示所有对分类有影响的像素
- Guided Backprop只保留正梯度，生成更清晰、更容易解释的可视化结果
- Saliency Maps更完整地展示决策依据，而Guided Backprop更适合看正向激活的特征

让我更详细地解释第5章的内容。

1. Adversarial Attacks（对抗攻击）

对抗攻击是一个引人深思的现象，它揭示了深度学习模型的一个重要弱点。想象一下，一个经过充分训练的CNN可以以99%的置信度正确识别一张猫的图片，但如果我们对这张图片做一些肉眼几乎看不出的微小改动，模型可能会以同样高的置信度认为这是一只狗。

对抗攻击的工作原理：

1. 给定一个输入图像x和目标模型f(x)
2. 计算损失函数L对输入x的梯度：∇xL(f(x), y)
3. 沿着梯度方向添加扰动：x' = x + ε * sign(∇xL)

这里有几个关键的技术细节：

```
def generate_adversarial_example(image, target_label, model, epsilon=0.01):
    # 计算原始预测
    initial_prediction = model.predict(image)

    # 计算损失相对于输入的梯度
    with tf.GradientTape() as tape:
        tape.watch(image)
        prediction = model(image)
        loss = categorical_crossentropy(target_label, prediction)

    # 获取梯度
    gradient = tape.gradient(loss, image)

    # 生成对抗样本
    adversarial_image = image + epsilon * tf.sign(gradient)

    # 确保像素值在有效范围内
    adversarial_image = tf.clip_by_value(adversarial_image, 0, 1)

    return adversarial_image
```

对抗攻击的类型：

1. **FGSM (Fast Gradient Sign Method)**：
   - 最基本的对抗攻击方法
   - 使用梯度符号来创建扰动
   - 计算快速但攻击效果相对较弱

2. **PGD (Projected Gradient Descent)**:
    - 迭代版本的FGSM
    - 多次小步骤更新
    - 攻击效果更强但计算成本更高
3. **DeepFool**:
    - 寻找最小扰动
    - 逐步将样本推向决策边界
    - 生成的对抗样本更难被检测

2. Gradient Ascent（梯度上升）

梯度上升是梯度下降的"反向"操作，它在特征可视化和神经网络理解中扮演着重要角色。

工作原理：

1. 从随机噪声或空白图像开始
2. 计算目标（如某个神经元的激活值）相对于输入的梯度
3. 沿着梯度方向更新输入，以最大化目标

梯度上升的应用：

1. **特征可视化**:
    - 可视化CNN不同层的神经元偏好的模式
    - 理解网络学到了什么样的特征
    - 帮助诊断网络问题
2. **风格迁移**:
    - 优化内容和风格损失
    - 生成具有目标风格的新图像
3. **神经网络解释性**:
    - 理解模型的决策依据
    - 发现模型的潜在偏见

3. Feature Inversion（特征反演）

特征反演是一个非常有趣的概念，它试图回答这样一个问题："如果我们有了CNN某一层的特征激活，能否重建产生这些激活的原始图像？"

想象一下，这就像是在玩一个"逆向猜谜游戏"。假设你的朋友看到一张猫的图片，告诉你图片中有"尖耳朵"、"胡须"和"圆眼睛"这些特征，你需要根据这些描述重新画出那张猫的图片。特征反演就是在做类似的事情。

特征反演的工作过程：

1. 首先，我们通过CNN获取目标图像的特征表示
2. 然后，从一个随机初始化的图像开始
3. 逐步调整这个图像，使其在相同的CNN层产生相似的特征激活
4. 最终得到一个与原始图像在特征空间上相似的新图像

为什么特征反演重要？

- 它帮助我们理解CNN各层捕获的信息量
- 验证不同层级特征的表示能力
- 探索网络的特征空间结构

4. Chain Rule & Backprop（链式法则和反向传播）

这是深度学习中最基础却也最重要的概念之一。想象一个多米诺骨牌效应：推倒第一块骨牌如何影响最后一块的倒下。链式法则就帮助我们理解这种连锁反应。

在神经网络中的应用：

- 假设我们有一个简单的计算链：输入→隐藏层→输出→损失
- 我们需要知道损失对输入层权重的影响
- 链式法则允许我们逐层计算这个影响

举个生活中的例子：
假设你在调整咖啡机的研磨粒度（输入），这会影响咖啡粉的大小（隐藏层），进而影响萃取时间（输出），最终影响咖啡的味道（损失）。要改善咖啡味道，你需要明白每个步骤之间的关联。

反向传播的实际过程：

1. 前向传播：计算每层的输出
2. 计算损失值
3. 反向传播：从输出层开始，逐层计算梯度
4. 更新权重

5. Feature Loss（特征损失）

特征损失是一个革命性的概念，它改变了我们评估图像相似度的方式。传统的像素级损失（如MSE）只关注像素值的直接差异，就像在比较两幅画时只看颜色是否完全一致。而特征损失则更接近人类的感知方式。

让我们通过一个具体的例子来理解：
想象你在看两张猫的照片。即使这两张照片的光线、角度略有不同，你仍然能认出它们是同一只猫。这是因为你的大脑关注的是高级特征（眼睛形状、毛发纹理等），而不是每个像素点的精确颜色。特征损失正是模仿这种思维方式。

特征损失的工作原理：

1. 使用预训练的CNN（通常是VGG网络）作为特征提取器
2. 将源图像和目标图像都输入这个网络
3. 比较它们在不同层级的特征激活
4. 计算这些特征激活之间的差异作为损失

为什么特征损失如此强大？

- 它能捕捉图像的语义信息
- 对细微的空间变形更加鲁棒
- 生成的结果更符合人类感知

6. Gram Matrix（格拉姆矩阵）

格拉姆矩阵是风格迁移中的一个关键创新。它解决了一个看似不可能的问题：如何数学化地描述一幅画的"风格"？

想象你在描述梵高的《星夜》的风格：漩涡状的笔触、强烈的色彩对比、动感的构图。格拉姆矩阵就是将这些抽象的风格特征转化为可计算的数学表达。

格拉姆矩阵的数学本质：

- 它计算特征图之间的相关性
- 捕捉不同特征如何组合和相互关联
- 忽略特征的具体空间位置，只保留风格信息

7. Texture Synthesis（纹理合成）

纹理合成是特征损失和格拉姆矩阵的一个重要应用。它的目标是从一个小的纹理样本生成更大的、视觉上连贯的纹理。

这个过程就像是一个艺术家在观察一小块木纹，然后在更大的画布上重现这种纹理模式。关键在于新生成的纹理要：

- 保持原始纹理的视觉特征
- 避免明显的重复模式
- 在更大尺度上保持自然和连贯

8. LPIPS和SSIM

这两个指标代表了图像质量评估的不同方法：

LPIPS（Learned Perceptual Image Patch Similarity）：

$$L_{LPIPS} = \Sigma_l w_l ||F_l(x) - F_l(y)||_2$$

- 基于深度学习的感知相似度度量
- 更接近人类的主观判断
- 特别适合评估生成模型的输出质量

SSIM（Structural Similarity Index）：

$$SSIM(x,y) = [l(x,y)]^\alpha [c(x,y)]^\beta [s(x,y)]^\gamma$$

- 考虑图像的结构信息
- 评估亮度、对比度和结构三个方面
- 比传统的峰值信噪比（PSNR）更有效

这两个指标的重要性在于：

- 提供了更科学的图像质量评估方法
- 帮助我们开发更好的图像处理算法
- 在图像压缩、超分辨率等任务中提供可靠的评估标准

让我为第五章提供几个核心的重点掌握问题和深入的答案：

1. **详细解释对抗攻击的原理，为什么微小的扰动就能欺骗CNN？**
   答：对抗攻击利用了CNN的线性特性和高维空间的特点：

- 在高维空间中，即使每个维度的扰动很小，累积效应也会很显著
- CNN主要依赖局部特征模式匹配，而不是真正理解图像语义
- 通过计算损失函数对输入的梯度，我们可以找到最能改变模型预测的扰动方向

- 这些扰动虽然对人眼不可见，但能系统性地激活或抑制网络中的特定神经元，从而误导模型的判断

2. **特征损失和像素级损失的区别是什么？为什么特征损失更适合图像生成任务？**
   答：这两种损失函数的区别反映了评估图像相似度的不同思路：

- 像素级损失（如MSE）直接比较像素值的差异，就像在逐点对比两张图片
- 特征损失使用预训练CNN的中间层特征进行比较，模仿人类的视觉感知系统
- 特征损失的优势在于：
    1. 能捕获更高级的语义信息
    2. 对空间变形更加鲁棒
    3. 生成的图像更符合人类审美期望
    4. 允许在保持内容的同时有更大的风格变化空间

3. **格拉姆矩阵如何捕获图像的风格特征？为什么它在风格迁移中如此重要？**
   答：格拉姆矩阵通过计算特征图之间的相关性来表示风格信息：

- 数学上，它计算特征图通道之间的内积，形成相关性矩阵
- 这种计算忽略了特征的具体空间位置，只保留它们如何相互关联的信息
- 在风格迁移中的重要性：
    1. 能够分离内容和风格信息
    2. 捕获纹理、笔触、色彩搭配等风格元素
    3. 使得风格迁移过程可以数学化处理
    4. 允许生成保持原始内容但具有目标风格特征的图像

4. **感知相似度度量（如LPIPS）相比传统度量（如PSNR）有什么优势？**
   答：感知相似度度量提供了更接近人类视觉感知的评估方式：

- 传统度量（PSNR/MSE）：
    1. 仅考虑像素级别的差异
    2. 对细微的空间偏移过分敏感
    3. 可能与人类主观判断不一致
- 感知相似度度量（LPIPS）：
    1. 利用深度网络学习到的特征空间
    2. 考虑图像的语义信息和结构特征
    3. 更好地反映人类对图像质量的主观评价
    4. 特别适合评估生成模型的输出

让我深入讲解第6章 - Sequence Learning（序列学习）

1. RNN (循环神经网络)

循环神经网络是专门为处理序列数据设计的神经网络架构。想象一下你在读一本书，你对每个词的理解都依赖于之前读过的内容。RNN就是模拟这种行为的网络结构。

RNN的核心特点：

```
ht = tanh(whh * ht-1 + wxh * xt + bh)
yt = why * ht + by
```

其中：

- ht 是当前时刻的隐藏状态
- xt 是当前输入
- yt 是当前输出

RNN的主要问题：

- 长期依赖问题：难以捕捉距离较远的信息
- 梯度消失/爆炸：训练过程中梯度可能变得极小或极大

2. LSTM (长短期记忆网络)

LSTM是对标准RNN的改进，通过引入门控机制来解决长期依赖问题。它就像是一个更智能的记事本，能够决定什么信息值得记住，什么信息可以忘记。

LSTM的四个关键组件：

1. 遗忘门（forget gate）：决定丢弃什么信息
2. 输入门（input gate）：决定存储什么新信息
3. 输出门（output gate）：决定输出什么信息
4. 单元状态（cell state）：信息的主要传递线

3. Attention (注意力机制)

注意力机制是深度学习中的一个重大突破。就像人类在阅读时会关注重要的词一样，注意力机制允许模型在处理序列时对不同部分分配不同的重要性。

注意力计算的基本步骤：

1. 计算查询(Query)和键(Key)的相似度得到注意力分数
2. 对分数进行softmax归一化
3. 使用注意力权重对值(Value)进行加权求和

数学表达：

```
Attention(Q,K,V) = softmax(QK^T/√d)V
```

4. Activations (激活函数)

在序列学习中，激活函数的选择尤为重要。不同的激活函数有不同的特性：

1. tanh：常用于RNN的隐藏状态
    - 输出范围：[-1,1]
    - 中心化输出有助于训练
2. sigmoid：常用于门控机制
    - 输出范围：[0,1]
    - 适合表示概率或门控值
3. ReLU：在一些变体架构中使用
    - 避免梯度消失
    - 计算效率高

5.RNN with Attention的结构和工作原理。

这是一个将RNN与注意力机制结合的重要架构，它极大地提升了序列处理的能力。

结构组成和工作流程

1. **输入序列处理**

- x_1, x_2, x_3 是输入序列的元素
- 通过RNN网络，每个输入生成对应的隐藏状态 h_1, h_2, h_3
- 隐藏状态包含了输入信息的编码表示

2. **注意力分数计算**

- $e_{t,i} = g_{att}(s_{t-1}, h_i)$ 表示注意力打分函数
- s_{t-1} 是上一时刻的解码器状态
- h_i 是编码器的隐藏状态
- g_{att} 通常使用一个前馈神经网络来计算相关性分数

3. **注意力权重生成**

- 使用Softmax将分数转换为权重：$a_{t,i}$
- 满足两个约束条件：
    1. $0 \leq a_{t,i} \leq 1$ (每个权重在0到1之间)
    2. $\Sigma a_{t,i} = 1$ (所有权重和为1)
- 这确保了注意力的"软"分配

4. **上下文向量计算**

- $c_t = \Sigma a_{t,i} h_i$$
- 这是一个加权和操作
- 权重越大的隐藏状态对上下文向量贡献越大
- 上下文向量动态地聚合了相关信息

5. **输出生成**

- 将上下文向量c_t和前一时刻的输出y_0结合
- 生成新的状态s_1
- 基于s_1生成当前时刻的输出y_1

为什么这个设计如此重要?

1. **动态关注**

- 模型可以在生成每个输出时动态决定关注输入序列的哪些部分
- 不同输出可能需要关注不同的输入信息

2. **长距离依赖**

- 克服了传统RNN的长序列处理困难
- 可以直接访问任何位置的信息，不受距离限制

3. **并行计算**

- 注意力权重的计算可以并行进行
- 提高了模型的计算效率

举个实际的例子：在机器翻译中，当我们要翻译"我喜欢吃苹果"这句话时：

- 翻译"喜欢"时，模型会更多地关注"我"
- 翻译"苹果"时，模型会更多地关注"吃"
- 注意力权重反映了这种动态的关联强度

这个机制极大地提升了序列到序列模型的性能，也为后来的Transformer架构奠定了基础。

6.Transformer架构:

1. Transformer的本质与创新

想象Transformer是一个超级高效的"阅读理解助手"。传统的RNN/LSTM像人类一样按顺序读取文本，而Transformer能够同时关注所有位置，就像拥有了"上帝视角"。这种并行处理机制使其成为处理序列数据的革命性架构。

2. 核心组件详解

3. 数学原理详解

1. **自注意力机制(Self-Attention)**
   核心公式：

```
Attention(Q, K, V) = softmax(QK^T/√d_k)V
```

其中：

- Q(查询): 当前位置想要获取的信息
- K(键): 用于与查询匹配的键
- V(值): 实际的信息内容
- √d_k: 缩放因子，防止梯度消失

2. **多头注意力(Multi-Head Attention)**
   将输入投影到多个子空间：

```
MultiHead(Q,K,V) = Concat(head_1,...,head_h)W^O
where head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)
```

3. **位置编码(Positional Encoding)**
   使用正弦和余弦函数：

```
PE(pos,2i) = sin(pos/10000^(2i/d_model))
PE(pos,2i+1) = cos(pos/10000^(2i/d_model))
```

4. 关键概念详解

1. **为什么需要Multi-Head?**
   不同的头可以关注不同的模式：

- 有的头可能关注语法关系
- 有的头可能关注语义相似度

- 有的头可能关注位置相关性

2. **残差连接的重要性**

- 防止梯度消失
- 允许信息直接传递
- 帮助训练更深的网络

实践问题与答案：

1. **Q: Transformer为什么要使用多头注意力而不是单个注意力机制?**
   A: 多头注意力允许模型同时从不同的表示子空间学习特征，增加了模型的表达能力。就像人类理解语言时会同时注意语法、语义、上下文等多个方面。
2. **Q: 位置编码的作用是什么?**
   A: Transformer是并行处理的，失去了序列的位置信息。位置编码通过给每个位置添加独特的编码，帮助模型理解序列中元素的相对或绝对位置。
3. **Q: 为什么要进行尺度缩放（Scale）?**
   A: 在点积注意力中，当输入维度增大时，点积的结果也会增大，导致softmax函数梯度消失。除以√d_k可以稳定梯度。

重点掌握问题及答案：

1. **为什么标准RNN会存在长期依赖问题？LSTM如何解决这个问题?**
   答：标准RNN的长期依赖问题源于：

- 梯度在反向传播时会不断相乘，导致梯度消失或爆炸
- 信息在传递过程中会逐渐衰减

LSTM通过以下机制解决：

- 引入细胞状态作为主要信息传递通道
- 使用门控机制控制信息流动
- 允许信息直接传递，不需要经过多次非线性变换

2. **注意力机制的工作原理是什么？为什么它如此重要?**
   答：注意力机制的工作原理：

- 计算输入序列每个位置的重要性分数
- 根据分数对信息进行加权聚合
- 允许模型动态关注相关信息

重要性体现在：

- 突破了固定窗口的限制
- 建立了长距离依赖
- 提供了可解释性
- 大幅提升了模型性能

3. **LSTM中不同门的作用是什么？它们如何协同工作?**
   答：LSTM的门控机制：

- 遗忘门：控制需要遗忘的历史信息

- 输入门：控制当前输入的重要性
- 输出门：控制输出的信息量

协同工作方式：

- 遗忘门先清理无关信息
- 输入门选择性地添加新信息
- 输出门控制信息的输出
- 三个门共同维护长期记忆

让我详细讲解第7章 - 生成模型(第一部分)。

1. **自编码器(Autoencoder, AE)**
   自编码器是一种无监督学习模型，其目标是学习数据的压缩表示。它由两部分组成：

- 编码器(Encoder)：将输入压缩为潜在表示
- 解码器(Decoder)：从潜在表示重建输入

基本公式：

```
# 编码过程
z = encoder(x)  # z的维度通常小于x
# 解码过程
x_reconstructed = decoder(z)
# 损失函数
loss = ||x - x_reconstructed||²
```

2. **变分自编码器(Variational Autoencoder, VAE)**
   VAE是AE的概率版本，它学习的不是确定的潜在表示，而是概率分布。

关键特点：

- 编码器输出均值和方差
- 使用重参数化技巧采样
- 损失函数包含重建误差和KL散度

1. **数学基础**
   VAE的核心是最大化数据的边际似然：
   $\log p(x) = \log \int p(x|z)p(z)dz$

但这个积分难以直接计算，所以引入变分推断，得到ELBO（证据下界）：
$\log p(x) \geq E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - KL(q_\phi(z|x)||p(z))$

其中：

- $q_\phi(z|x)$ 是编码器（近似后验）
- $p_\theta(x|z)$ 是解码器（生成模型）
- $p(z)$ 是先验分布（通常是标准正态分布）

2. **VAE伪代码实现**

```
第一阶段：压缩（编码）
输入数据首先经过编码器网络
```

编码器不是直接输出压缩结果，而是输出两个值：
    一个表示平均值(μ)
    一个表示方差(σ²)

第二阶段：采样
使用这两个值定义了一个正态分布
从这个分布中随机采样得到压缩表示
这个采样过程使用"重参数化技巧"，确保可以进行梯度反向传播

第三阶段：重建（解码）
将采样得到的压缩表示输入解码器
解码器尝试重建原始输入数据

第四阶段：学习
计算两个损失：
    重建损失：重建结果与原始输入的差异
    KL散度损失：确保压缩表示近似标准正态分布
综合这两个损失来更新网络参数

3. **生成对抗网络(GAN)**
4. **数学基础**
   GAN的目标函数是一个极小极大博弈：

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

其中：

- G是生成器，试图最小化这个值
- D是判别器，试图最大化这个值
- p_{data} 是真实数据分布
- p_z 是噪声先验分布

2. **GAN伪代码详细实现**

第一阶段：生成
生成器从随机噪声开始
通过神经网络将噪声转换成看起来真实的数据（如图片）

第二阶段：判别
判别器会看到两种输入：
    真实数据样本
    生成器创造的假样本
判别器需要分辨每个样本是真是假

第三阶段：对抗训练（两个网络轮流训练）
    训练判别器：
使其正确识别真实样本和生成的假样本
判别器参数更新，生成器保持不变

    训练生成器：
生成新的假样本
目标是骗过判别器，让其认为生成的样本是真的
生成器参数更新，判别器保持不变

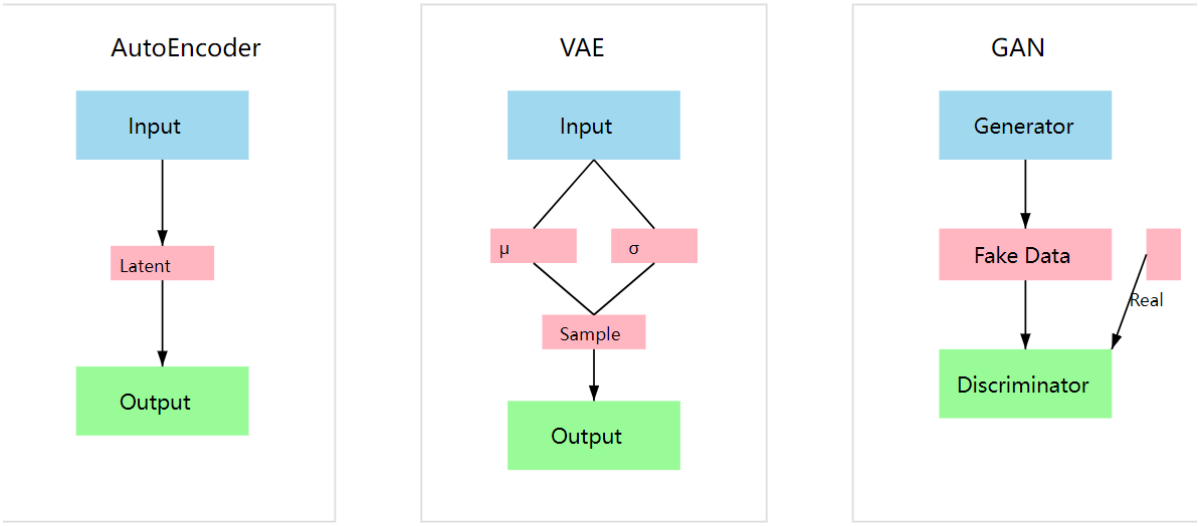
第四阶段：迭代优化
不断重复上述过程
生成器逐渐学会生成更逼真的样本
判别器逐渐提高分辨能力
最终达到纳什均衡，生成器能产生高质量的假样本

3. **高级GAN训练技巧**

- 使用标签平滑：真实标签不用1而用0.9
- 添加噪声到判别器输入
- 使用WGAN损失函数改进稳定性
- 梯度惩罚项(GP)防止梯度爆炸

这个图展示了：

1. GAN的整体训练流程（上半部分）
2. 训练过程中生成器和判别器损失的典型变化（下半部分）



这个图展示了三种生成模型的基本架构。从左到右分别是：

- AE：简单的编码器-解码器结构
- VAE：加入了概率编码的变分自编码器
- GAN：对抗性的生成器-判别器结构

重点需要掌握的问题及答案：

1. **为什么VAE比普通AE更适合生成任务？**
   答：VAE学习的是概率分布而不是确定性映射，这使得它能够生成新样本。通过对潜在空间的正则化（KL散度项），VAE确保了潜在空间的连续性和平滑性，使得在潜在空间中的插值能产生有意义的结果。
2. **GAN训练为什么难？常见问题是什么？**
   答：GAN训练难点主要有：

- 模式崩溃：生成器只生成有限种类的样本
- 训练不稳定：生成器和判别器需要保持平衡
- 梯度消失：判别器太好导致生成器得不到有效梯度
- Nash均衡难以达到：两个网络的博弈过程难以收敛

3. **三种模型各自的优缺点是什么?**
   答:

- AE:简单易实现,但生成能力有限
- VAE:生成结果连续平滑,但可能模糊
- GAN:生成质量最好,但训练困难且不稳定

4. **什么是VAE的重参数化技巧?为什么需要它?**
   答:重参数化技巧是将随机采样操作转换为确定性函数的方法。不直接从分布中采样,而是采样一个标准正态分布的噪声,然后通过均值和方差进行变换。这样可以使得梯度能够通过采样操作向后传播,实现端到端训练。
5. **VAE的KL散度项有什么作用?** 答:KL散度项确保编码器输出的分布接近标准正态分布,这有两个作用:

- regularization:防止过拟合
- 使潜在空间连续且有意义:便于采样和插值

6. **GAN的模式崩溃(Mode Collapse)是什么?如何解决?** 答:模式崩溃是指生成器只产生有限几种样本的现象。解决方法包括:

- 使用MiniBatch判别
- 添加重构损失
- 使用WGAN等改进的GAN变体
- 增加生成器的容量和多样性

最后,让我全面系统地解释第8章:扩散模型(Diffusion Models)。

1. 直观理解

扩散模型的工作原理可以类比为"图片的逐步破坏与重建"过程。想象你有一张清晰的照片:

- 前向过程:慢慢往照片上撒沙子,直到完全看不清原图
- 反向过程:学习如何一点点把沙子清理掉,重现原图

2. 数学原理

扩散模型包含两个关键过程:

A) **前向扩散过程(Forward Diffusion)**
逐步向数据添加高斯噪声:
$q(x_t|x_{t-1}) = N(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$

其中:

- x_t 是第t步的噪声图像
- β_t 是噪声调度表(noise schedule)
- 最终图像变为纯高斯噪声

B) **反向扩散过程(Reverse Diffusion)**
学习去噪步骤:
$p_\theta(x_{t-1}|x_t) = N(x_{t-1}; \mu_\theta(x_t,t), \Sigma_\theta(x_t,t))$

这个过程学习如何逐步预测并移除噪声。

3. 实现过程

主要步骤：

1. **噪声调度设置**
2. **前向扩散**

- 给定一张图片
- 随机选择时间步t
- 添加相应程度的噪声

3. **训练目标**

- 模型学习预测添加的噪声
- 使用简单的L2损失：
  $L = E_{x_0,\epsilon,t}[||\epsilon - \epsilon_\theta(x_t, t)||^2]$

4. **采样过程**

- 从纯噪声开始
- 逐步应用学习到的去噪过程
- 最终得到清晰图像

4. 关键创新

1. **预测噪声而不是图像**
   这让训练更稳定，因为模型不需要直接生成复杂的图像分布。
2. **时间步编码**
   使用sinusoidal position encoding来表示每个时间步：
   $PE(t)_i = \sin(t/10000^{2i/d})$

5. 与其他生成模型的比较

相比GAN和VAE：

- 训练更稳定
- 可以生成更高质量的样本
- 采样过程较慢
- 理论基础更扎实

重要问题解答：

1. **Q: 为什么扩散模型的采样质量通常比GAN更好?**

A: 扩散模型的优势在于其渐进式的生成过程。与GAN的一步生成不同，扩散模型通过多步小改变来生成图像，这让它能够：

- 更细致地捕捉数据分布
- 避免模式崩溃问题
- 在生成过程中有更多的纠错机会

Coursework

1. 线性拟合与多项式拟合

1. **线性回归的本质**
   线性回归是最基础的机器学习模型，它试图找到一个线性函数来拟合数据点。比如在一维情况下：

```
y = wx + b
```

其中w是权重，b是偏置项。

在高维情况下，这个公式扩展为：

```
y = w1x1 + w2x2 + ... + wnxn + b
```

2. **正规方程（Normal Equation）**
   正规方程是求解线性回归的解析方法。其核心公式是：

```
w = (X^T X)^(-1) X^T y
```

这个公式看起来复杂，但实际上就是在最小化预测值与真实值之间的均方误差。

3. **多项式拟合**
   多项式拟合是线性拟合的扩展，它允许我们拟合更复杂的非线性关系。在实验中要求的二次多项式形式是：

```
y = w1x² + w2xy + w3y² + w4x + w5y + w6
```

这里的关键是理解：虽然最终的函数是非线性的，但通过特征转换，我们仍然可以用线性回归的方法求解。只需要将原始特征转换为新的特征空间：

- 原始特征：[x, y]
- 转换后特征：[x^2, xy, y^2, x, y, 1]

4. **实际应用**
   在实验中，你需要用这些方法来预测鸢尾花数据集中的特征。这是一个经典的机器学习数据集，包含了不同鸢尾花的萼片长度、萼片宽度、花瓣长度和花瓣宽度等特征。

2. 图像检索 (Image Retrieval)

图像检索是计算机视觉中的一个基础任务，涉及到课程中的"Image Analysis"和"Feature Representation"的核心概念。让我通过一个简单的比喻来解释：

想象你在寻找一件特定的衣服，你可能会从不同角度来描述它：

1. 颜色分布（对应RGB距离）
2. 整体风格（对应特征直方图）
3. 细节特征（对应VGG特征）
4. 纹理风格（对应Gram矩阵）

让我们详细讲解每种特征表示方法：

2.1 RGB距离

这是最简单的特征表示方法。直接比较两张图片对应位置的像素值差异。

```
# 示意代码
def rgb_distance(img1, img2):
    return np.mean((img1 - img2) ** 2)
```

这种方法的问题是太过简单，对图像的位置变化非常敏感。比如同一张图片稍微移动一下位置，RGB距离就会很大。

2.2 直方图距离

直方图统计了图像中各种颜色出现的频率，不受位置的影响：

2.3 VGG特征

VGG是一个预训练的深度神经网络，它的中间层特征可以捕捉到图像的高级语义信息。这些特征比简单的颜色特征更有意义：

- 浅层特征：边缘、纹理等低级特征
- 深层特征：物体部件、场景结构等高级特征

```
# VGG特征提取示意代码
def vgg_features(image):
    model = models.vgg16(pretrained=True)
    features = model.features(image)
    return features
```

2.4 Gram矩阵

Gram矩阵捕捉了特征之间的相关性，特别适合表示图像的整体风格：

```
def gram_matrix(features):
    b, c, h, w = features.size()
    features = features.view(c, h * w)
    return torch.mm(features, features.t()) / (h * w)
```

3. 图像去噪 (Image Denoising)

图像去噪是一个重要的图像处理任务，这部分内容涉及到课程中的"Tunable image filters"和"Neural networks"两个模块。我们来对比两种方法：

3.1 高斯滤波

高斯滤波是一种传统的去噪方法，使用固定的卷积核：

高斯滤波的原理是用周围像素的加权平均值替代中心像素，权重由高斯函数决定。这种方法简单有效，但可能会模糊图像的细节。

3.2 CNN去噪

卷积神经网络可以学习更复杂的去噪映射：

```
class DenoiseNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 64, 3, padding=1)
        self.conv2 = nn.Conv2d(64, 64, 3, padding=1)
        self.conv3 = nn.Conv2d(64, 3, 3, padding=1)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = F.relu(self.conv2(x))
        return self.conv3(x)
```

CNN的优势在于：

1. 可以学习数据集特定的去噪模式
2. 能够保持边缘和纹理细节
3. 可以处理不同类型的噪声

4. 纹理合成 (Texture Synthesis)

这部分内容与课程大纲中的"Ambiguity and style"以及"Generative modelling"密切相关。纹理合成的目标是生成与给定样例在视觉上相似的纹理图像。

4.1 纹理特征提取

纹理的表示方法有三个层次，逐渐变得更加复杂：

1. **均值(Mean)**
   想象一下油画的调色板，均值就像是整体的色调。它告诉我们纹理的基本颜色倾向，但没有包含任何模式信息。
2. **方差(Variance)**
   方差描述了颜色的变化程度。比如：
   - 大理石纹理：方差较小，颜色变化平缓
   - 豹纹：方差较大，颜色对比强烈
3. **Gram矩阵（Style Matrix)**
   这是最复杂的表示方法，捕捉了特征之间的相互关系。例如：
   - 如果两个特征总是一起出现（比如砖墙的水平线和垂直线），它们在Gram矩阵中的值会很大
   - 如果两个特征很少一起出现，对应的值会很小

原始纹理

均值表示

方差表示

Gram矩阵表示

4.2 纹理合成的关键问题

1. **初始化的重要性**
   - 从随机噪声开始可能导致局部最小值
   - 使用目标纹理的均值或降采样版本作为初始值往往能获得更好的结果
2. **失败案例分析**
   在处理规则纹理（如棋盘格）时常见的问题：
   - 均值只能捕捉颜色，完全丢失了规则排列信息
   - Gram矩阵可以捕捉到局部特征关系，但可能无法保持全局的规则性
3. **改进方法**
   - 使用多尺度合成
   - 添加周期性约束
   - 考虑空间相关性

5. 神经隐式表示 (Neural Implicit Representation)

这是课程中最前沿的内容，涉及到"Neural networks"和"Generative modelling"。这种方法用神经网络来表示连续函数，而不是离散的像素值。

5.1 基本原理

想象一个魔法函数，输入任意坐标(x,y,t)，它能告诉你在那个位置和时间点的像素值。这就是神经隐式表示的本质。

5.2 位置编码（Positional Encoding）

为什么需要位置编码？想象你在给一个很大的房间编号：

- 直接用坐标(x,y)：就像用房间的长和宽
- 加入位置编码：相当于同时考虑房间的位置、大小、朝向等多个维度的信息

位置编码的公式：

```
γ(x) = (sin(2⁰πx), cos(2⁰πx), sin(2¹πx), cos(2¹πx), ...)
```

5.3 关键点

1. **连续性表示**：可以在任意时间点生成帧，而不仅限于训练数据中的时间点
2. **压缩性**：整个视频序列被压缩成网络参数
3. **平滑插值**：可以生成平滑的中间帧

这些概念在现代计算机图形学和视觉计算中越来越重要，尤其是在处理3D场景重建、视频编辑等任务时。

C1 Intro

1. 课程概述与背景

本课程旨在将人工智能应用于创意产业问题,特别关注机器学习如何处理和生成视觉数据。这门课程涵盖了:

- 基础知识:模型、优化、SGD(随机梯度下降)、MLP、位置编码(position encoding)
- 核心技术:CNN(卷积神经网络)、自编码器(autoencoding)、GAN(生成对抗网络)等
- 前沿应用:扩散(diffusion)模型、Transformer等
- 专业应用:渲染(rendering)、3D几何等

2. 视觉计算中的数据表示

课程介绍了多种视觉数据的表示形式:

- 图像/视频:像素网格,随时间变化的像素网格
- 体积数据:体素网格(voxel grid)
- 网格(mesh):顶点/边/面
- 动画:随时间变化的骨骼位置、布料动力学
- 点云(point clouds):点阵列
- 物理模拟:时空流体流动

3. 常见任务
    - 图像分类: Image Classification
    - 图像字幕: Image Captioning
    - 语义分割: Semantic Segmentation
    - 3D内容创作: 3D Content Creation
4. 数据驱动算法

课程介绍了两种主要的机器学习范式:

监督学习: Supervised

- 需要标注数据(supervision data)
- 使用验证数据检查收敛 Validation
- 在测试数据上评估模型
- 通常采用70%(训练):15%(验证):15%(测试)的数据分割比例

无监督学习: Unsupervised

- 不需要标注数据

- 直接从训练数据中学习
- 同样需要验证和测试阶段

5. 机器学习在视觉计算中的特点

视觉数据的特殊性:

- 可以做图像处理和转换任务(IP)
- 有多种输入数据来源(图像、扫描仪、动作捕捉)
- 可以利用合成数据作为监督(synthetic)数据(渲染、动画)
- 涉及光线传输模型和几何不变量
- 生成模型有很多挑战

6. 历史发展

课程也简要介绍了机器学习在视觉领域的重要里程碑:

- 1958: 感知机的提出: Perceptron
- 1974: 反向传播算法: Backpropagation
- 1981: Hubel & Wiesel因视觉系统研究获诺贝尔奖
- 1990s: SVM时代
- 1998: CNN用于手写分析
- 2012: AlexNet在ImageNet竞赛中取得突破

7. 应用实例

课程展示了机器学习在图形学中的多个应用:

- 图像处理:素描简化、上色、去噪
- 3D处理:网格分割、程序化建模、变形学习
- 动画:面部动画、流体模拟
- 渲染:实时渲染、BRDF估计

C2 Optimization

第一部分 - 图像表示和基本概念:

1. 图像表示: image

- 课程一开始介绍了图像如何被表示为数学形式 - 一个点在高维空间中 ($p \in \mathbb{R}^{3WH}$)
- 这表示一个W×H大小、3通道(RGB)的图像可以被展平成一个高维向量

2. 图像分类任务: classification

- 介绍了基本的分类数据结构:训练数据 $X := (I_1, l_1), (I_2, l_2), \ldots (I_n, l_n)$
- 其中 $I_j \in \mathbb{R}^{(r \times c \times 3)}$ 是输入图像
- $l_j \in \mathbb{Z}^k$是类别标签
- 目标：学习一个映射函数 $f : I \to l$

3. 图像比较:

- 介绍了两种计算图像间距离的主要方法:

- L2距离(欧几里得距离): $D(I_1, I_2) := \sum_{i,j}(I_1(i,j) - I_2(i,j))^2$
- L1距离(曼哈顿距离): $D(I_1, I_2) := \sum_{ij}|I_1(i,j) - I_2(i,j)|$

4. 最近邻分类: Nearest Neighbor

- 基本原理:根据训练数据中最近的样本来分类新样本
- 特点:
  - 决策边界呈锯齿状
  - 容易出现孤立区域
  - 容易产生错误
  - 训练时间 O(1)
  - 测试时间 O(n)
- k-最近邻(k-NN):使用k个最近邻进行投票,可提高鲁棒性robust
- 选择k值是一个重要的超参数Hyperparameters

第二部分 - 线性回归与优化基础

1. 参数化学习模型：Learning a Function

$$y = f_w(x) = f(x; w) = w^T x$$

- 输入x∈$\mathbb{R}^n$：特征向量
- 参数w∈$\mathbb{R}^n$：需要学习的权重
- 输出y∈$\mathbb{R}$：预测值
- 线性函数：$f(x) = ax + b$
- 二次函数：$g(x) = ax^2 + bx + c$
- 这里的{a,b,c}都是需要通过训练学习的参数，目标是找到最优的参数组合使得预测误差最小

2. 损失函数(Loss Function)：

- 均方误差(MSE)：
  $MSE_{test} = \frac{1}{N}\sum_i(\hat{y}_i^{test} - y_i^{test})^2 + \lambda w^T w$
- 正则化项$\lambda w^T w$：防止过拟合
- 超参数λ：控制正则化强度

3. 一维优化

- 优化目标：找到函数f(x)的最小值点
- 局部极值条件：在极值点处导数为零 $f'(x^*) = 0$

4. 最小二乘法(Least Squares)拟合

- 对于给定的训练数据xi, yi
- 目标函数：
  $L(w) = \sum_{i=1}^N(\epsilon^i)^2 = \sum_{i=1}^N(y^i - w^T x^i)^2$
- 矩阵形式：
  $L(w) = (y - Xw)^T(y - Xw) = y^T y - 2y^T Xw + w^T X^T Xw$
- 最优解(Normal Equation)：line fitting

$$w^* = (X^T X)^{-1} X^T y$$

- 广义线性回归：$x \to \phi(x), w^* = (\Phi^T \Phi)^{-1} \Phi y$

5. 正则化(Ridge Regression)

- 添加L2正则化项：

$$L(w) = \epsilon^T \epsilon + \lambda w^T w$$

- λ是正则化系数(hyperparameter)
- 正则化解：

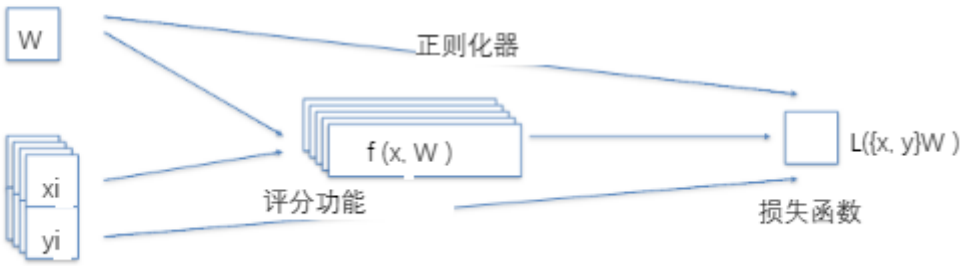
$$w^* = (X^T X + \lambda I)^{-1} X^T y$$

6. 梯度下降优化:

- 初始化x_0
- 梯度下降(Gradient Descent)：

$$w_{t+1} = w_t - \alpha \nabla f(w_t)$$

图像分类器



- α：学习率
- $\nabla f(w_t)$：损失函数在w_t处的梯度
- 关键问题：学习率选择
- 计算梯度的两种方式：
  - 解析梯度：直接计算导数
  - 数值梯度：$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$

7. 梯度下降变体：

a) 批量梯度下降(Batch GD)：

- 使用所有训练数据计算梯度
- 更新稳定但计算开销大

完整loss: $J(w) = \mathbb{E}_{x,y \sim p_{data}} L(x, y, w) = \frac{1}{N} \sum_i L(x^{(i)}, y^{(i)}, w)$

b) 随机梯度下降(SGD)：$\nabla_w J(w) \approx \frac{1}{m} \sum_i \nabla_w L(x^{(i)}, y^{(i)}, w)$

- 每次只用一个或小批量样本mini batch
- 更新快但波动大

c) 动量(Momentum)：

```
v_{t+1} = \rho v_t + \nabla f(w_t)
w_{t+1} = w_t - \alpha v_{t+1}
```

- ρ：动量系数
- 帮助逃离局部最小值

8. 优化挑战：

- 学习率选择：
  - 过大导致发散
  - 过小导致收敛慢
- 局部最小值：优化可能陷入非全局最优解
- 鞍点：梯度为零但非极值点
- 病态条件：不同方向的梯度尺度差异大

第三部分 - 多分类和逻辑回归

1. 多分类线性回归：

- 使用 one-hot 编码表示C个类别
- 示例：4个类别，第i个样本属于第3类表示为 $y^i = (0, 0, 1, 0)$
- 矩阵表示：$Y = \begin{bmatrix} y^1 \\ \vdots \\ y^N \end{bmatrix} = [y_1|\cdots|y_C]$

其中 yc 表示第c类的所有样本标签。$W = [w_1|\cdots|w_C]$

2. 多类别Loss函数：$L(W) = \sum_{c=1}^{C}(y_c - Xw_c)^T(y_c - Xw_c)$

- 每个类别可以独立优化
- 最小二乘解：$w_c^* = (X^TX)^{-1}X^Ty_c$

3. 线性回归的遮罩问题(Masking Problem)：

- 每个类别都有一个线性判别函数：$S_c(x) = w_c^Tx$
- 在多类分类中，某些类别可能永远不会被分配（总是被其他类别的分数覆盖）
- 这是线性回归用于分类的一个主要缺陷

4. 逻辑回归：Logistic regression

- 伯努利条件分布：

```
P(Y=1|X=x;w) = f(x,w)
P(Y=0|X=x;w) = 1-f(x,w)
```

- Sigmoid函数：$g(\alpha) = \frac{1}{1+exp(-\alpha)}$
- 当$\alpha \to -\infty$时，g(α)→0
- 当$\alpha \to +\infty$时，g(α)→1

5. 多类逻辑回归：

- Softmax函数：$P(y=c|x;W)=\frac{exp(w_c^Tx)}{\sum_{c'=1}^{C}exp(w_{c'}^Tx)}\doteq g_c(x,W)$
- 逻辑回归相比线性回归的优势：
  - 不存在遮罩问题
  - 输出可以解释为概率
  - 决策边界更合理
  - 对异常值更鲁棒

C3 MLP和过拟合

1. 线性回归的数学基础:

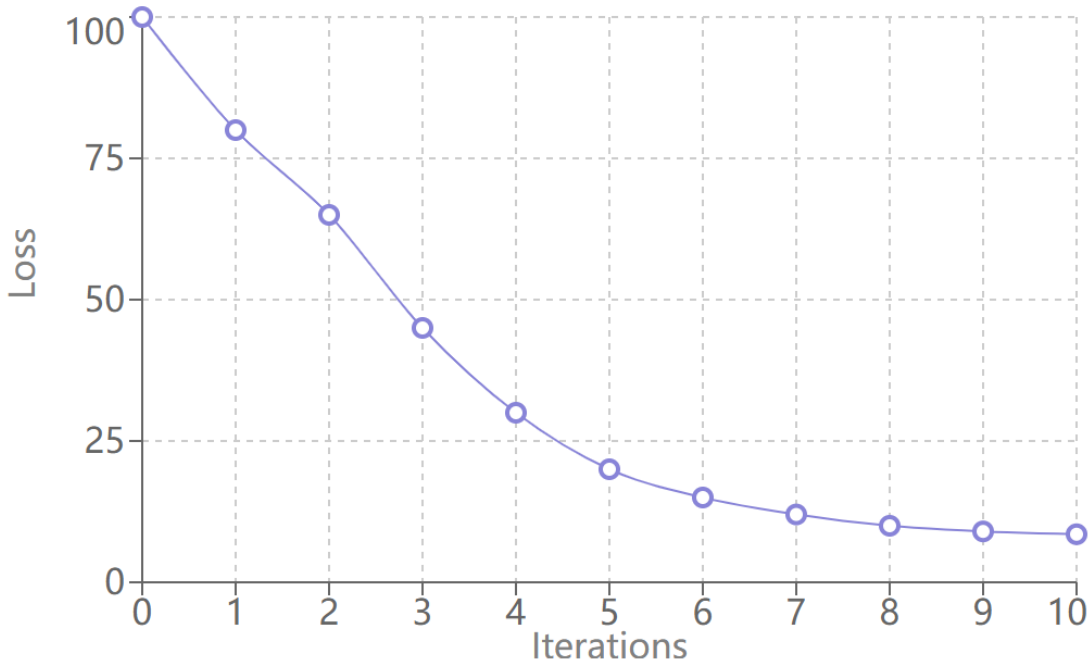
```
x ∈ Rⁿ   # 输入向量
w ∈ Rⁿ   # 权重向量
y ∈ R    # 输出标量

ŷ = <w,x> = wᵀx   # 预测值(点积)

# 均方误差损失
MSEtest = (1/N)Σ(ŷᵢtest - yᵢtest)²

# 梯度下降更新
wᵢ₊₁ = wᵢ - α∇MSE(wᵢ)
```

2. 随机/小批量梯度下降(SGD/Mini-batch GD):

- 目标函数：$J(w)=E[x,y\sim pdata]L(x,y,w)=(1/N)\sum L(x^{(i)},y^{(i)},w)$
- 梯度计算：$\nabla wJ(w)=(1/N)\sum \nabla wL(x^{(i)},y^{(i)},w)$
- 小批量近似：$\nabla wJ(w)\approx(1/m)\sum \nabla wL(x^{(i)},y^{(i)},w)$，其中$m<<N$

3. 重要的超参数:

- initialize_weights: 权重初始化方式
- learning_rate: 学习率
- number of steps: 训练步数/早停条件
- batch size: 小批量大小(常用8,16,32等)
- sampling: 采样策略



随着梯度下降迭代次数增加，损失函数值逐渐下降并收敛

这个可视化展示了梯度下降优化过程中损失函数的变化趋势。随着迭代次数增加，损失函数值不断下降，最终趋于收敛。这反映了梯度下降算法通过不断调整参数来最小化损失函数的过程。

4. 简单分类器与决策边界：

- 在幻灯片中展示了一个二分类的例子（猫与船的分类）
- 分类器试图找到一个决策边界（线性分类器中是一条直线）来分隔不同类别的数据点
- 在特征空间中，决策边界的一侧被分类为一个类别，另一侧被分类为另一个类别

5. Sigmoid激活函数：$\sigma(x) = 1/(1+e^{-x})$

Sigmoid函数的特点：

- 输出范围在(0,1)之间
- 常用于二分类问题
- 导数：$\sigma'(x) = \sigma(x)(1-\sigma(x))$
- 存在梯度消失问题

6. 激活函数汇总：Activation Functions

- ReLU(x) = max(0,x)：最常用的激活函数，解决了梯度消失问题
- Sigmoid：上面已经介绍
- $Tanh(x) = (e^{x} - e^{-x})/(e^{x} + e^{-x})$：输出范围(-1,1)
- Leaky ReLU：$max(\lambda x, x)$，其中λ<<1，解决了ReLU的"死亡"问题
- ELU：指数线性单元，结合了ReLU和Leaky ReLU的优点

7. XOR问题：
   XOR（异或）问题是一个经典的非线性分类问题：

```
x₁ x₂  y
0  0   0
0  1   1
1  0   1
1  1   0
```

- 单层感知机无法解决XOR问题，因为它是非线性可分的
- 需要多层神经网络（至少两层）才能解决
- 解决方案是通过隐藏层进行特征转换：$y = f(z_1, z_2) = f(g_1(x_1, x_2), g_2(x_1, x_2))$

6. 多层感知机(Multilayer Perceptron, MLP)：

    1. 基本结构：
    - 输入层：接收原始数据
    - 隐藏层：可以有多层，每层包含多个神经元
    - 输出层：产生最终预测结果

    每层的数学表达为：
    $h^{(l)} = \sigma(W^{(l)}h^{(l-1)} + b^{(l)})$
    其中：
    - $h^{(l)}$ 是第l层的输出
    - $W^{(l)}$ 是权重矩阵
    - $b^{(l)}$ 是偏置向量
    - σ 是激活函数

    2. 激活函数：
    3. 前向传播：
    4. 反向传播：
    - 计算损失函数关于每层参数的梯度
    - 使用链式法则逐层传播误差
    - 更新权重和偏置
    5. 训练过程：
    - 批量梯度下降
    - Mini-batch梯度下降
    - 随机梯度下降
    6. 正则化技术：
    - Dropout：随机丢弃一些神经元
    - L1/L2正则化：添加权重惩罚项
    - Batch Normalization：标准化层输出
    7. 实际应用注意事项：
    - 数据预处理：标准化、归一化
    - 权重初始化：Xavier/He初始化
    - 学习率选择：建议从小到大尝试
    - 层数和神经元数量：根据任务复杂度选择

8. 优缺点：
   优点：
   - 可以学习非线性关系
   - 结构简单，易于实现
   - 通用性强

   缺点：
   - 参数数量可能很大
   - 容易过拟合
   - 需要大量数据训练

7. 神经网络的基本组成：

```
# 两层神经网络的前向传播
h₁ = w¹₁₁x₁ + w¹₁₂x₂ + w¹₁₃x₃  # 第一个隐藏单元
h₂ = w²₁₁x₁ + w²₁₂x₂ + w²₁₃x₃  # 第二个隐藏单元
s = w¹₁₂h₁ + w¹₂₂h₂  # 输出层
```

第二部分 - 深度网络结构和反向传播：

1. 深度网络结构：

- 基本结构：输入层 → 多个隐藏层 → 输出层
- 数学表达：$f = W_i max(0, W_{i-1} max(0, \dots W_0 x))$
- 典型例子 AlexNet：
  - 包含卷积层、池化层和全连接层
  - 结构：224×224×3输入 → 卷积层 → 池化层 → 多个卷积层 → 全连接层(2048) → 输出层(1000类)
  - 每层都接ReLU激活函数
  - 使用Dropout防止过拟合

2. 损失函数与优化：

```
# 损失函数（以MSE为例）
L(w) = Σᵢ₌₁ⁿ(wᵀxᵢ - yᵢ)²

# 优化目标
min_w L(w)

# 最优解条件
∇wL(w) = 0
```

3. 链式法则（Chain Rule）：
   a) 单变量情况：$df(g(x))/dx = df(g(x))/dg(x) * dg(x)/dx = f'(g(x))g'(x)$

b) 多变量情况：$df(x(t), y(t))/dt = \partial f/\partial x * dx/dt + \partial f/\partial y * dy/dt$

4. 反向传播（Backpropagation）的基本块：

- 每个基本计算单元包含：
  - 前向传播：计算输出值

- 局部梯度：计算当前单元对输入的导数
- 反向传播：将上游梯度与局部梯度相乘传递

基本公式：

```
∂L/∂x₁ = ∂z/∂x₁ * ∂L/∂z  # 下游梯度
∂L/∂x₂ = ∂z/∂x₂ * ∂L/∂z  # 下游梯度
```

5. 反向传播模式的实现：

```
class ComputationalGraph:
    def forward(inputs):
        # 1. 传递输入到输入门
        # 2. 按拓扑排序顺序前向传播
        for gate in self.graph.nodes_topologically_sorted():
            gate.forward()
        return loss

    def backward():
        # 按拓扑排序的反序进行反向传播
        for gate in reversed(self.graph.nodes_topologically_sorted()):
            gate.backward()  # 应用链式法则
        return inputs_gradients
```

6. 反向传播中的基本模式：
   a) 加法（Add）块：

```
z = x + y
∂L/∂x = ∂z/∂x * ∂L/∂z = 1 * ∂L/∂z
∂L/∂y = ∂z/∂y * ∂L/∂z = 1 * ∂L/∂z
```

b) 乘法（Multiply）块：

```
z = x * y
∂L/∂x = y * ∂L/∂z
∂L/∂y = x * ∂L/∂z
```

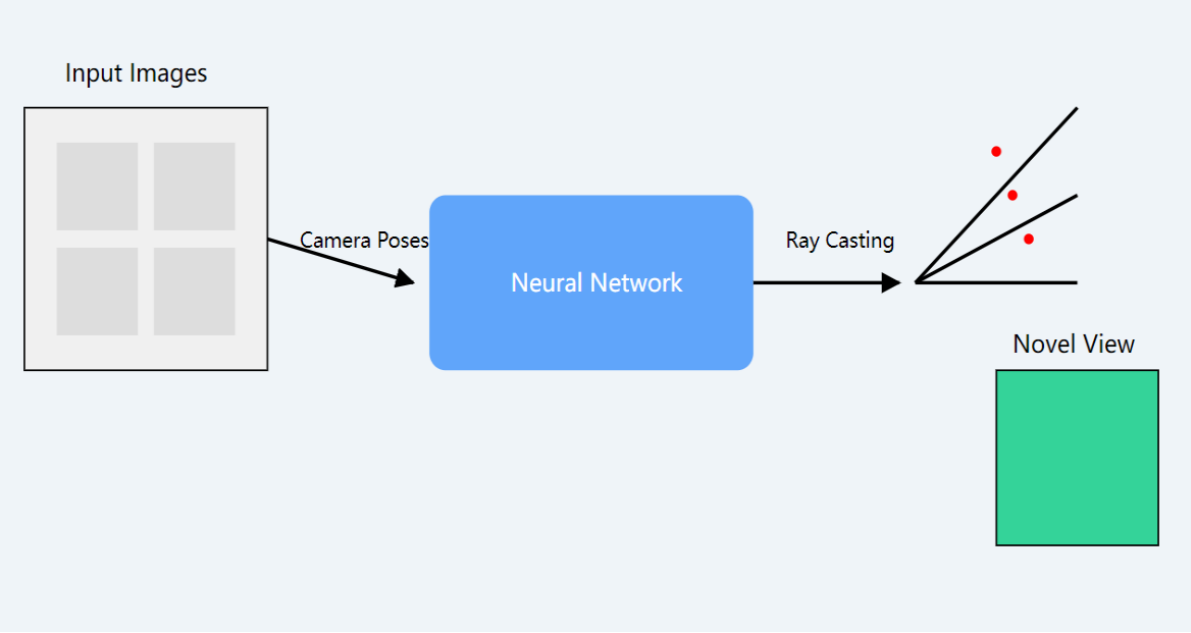
7. Positional Encoding（位置编码）：

- 目的：帮助网络更好地学习空间信息
- 公式：$\gamma(p) = (sin(2^0\pi p), cos(2^0\pi p), \ldots, sin(2^{L-1}\pi p), cos(2^{L-1}\pi p))$
- 在NeRF等任务中特别重要

8. 网络训练过程中的几个关键点：

- 权重初始化的重要性
- 学习率的选择和调整
- 批量大小的影响
- 优化器的选择（如SGD、Adam等）
- 正则化方法（L1/L2正则化、Dropout等）
- 避免过拟合的策略

NeRF(Neural Radiance Fields)与高质量视图合成：



1. NeRF的基本概念与工作流程：

- 输入：多视角的2D图像和对应的相机pose
- 处理：训练一个neural network来重建3D场景
- 输出：能够合成任意新视角的高质量图像

2. NeRF的系统架构：

```
# 5D输入表示
(x, y, z)   # 3D空间位置
(θ, ϕ)      # 2D视角方向

# 网络输出
(RGB, σ)    # RGB颜色和体密度
```

3. 体绘制方程(Volume Rendering Equation)：

```
# 最终颜色计算
Ĉ(r) = Σᵢ₌₁ᴺ Tᵢ(1 - exp(-σᵢδᵢ))cᵢ

# 透明度累积
Tᵢ = exp(-Σⱼ₌₁ⁱ⁻¹ σⱼδⱼ)
```

其中：r: 相机射线, σ: 体密度, c: RGB颜色, δ: 采样点间距, T: 透明度累积项,

4. NeRF的关键技术点：

- 位置编码(Positional Encoding)
  - 将输入坐标映射到高频空间
  - 有助于捕捉高频细节
  - $\gamma(p) = (sin(2^0\pi p), cos(2^0\pi p), \ldots, sin(2^{L-1}\pi p), cos(2^{L-1}\pi p))$
- 分层采样策略(Hierarchical Sampling)
  - 粗采样网络(Coarse Network)

- 细采样网络(Fine Network)
- 提高渲染效率和质量

5. 质量与速度的权衡:

- 渲染质量指标:
  - PSNR (Peak Signal-to-Noise Ratio)
  - 视觉质量和细节保真度
- 速度优化策略:
  - 分辨率调整
  - 采样点数量调整
  - 网络结构优化

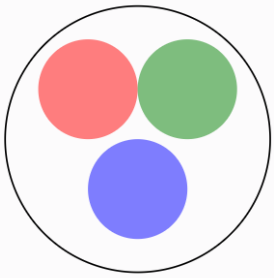
C4 视觉表示和处理 CG

第一部分:视觉表示

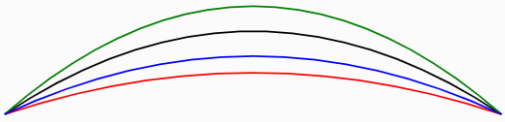
1. 基础概念和挑战

- 本课程需要处理的媒体类型包括:图像、视频、声音、3D表面等
- 主要目标是对这些媒体进行分析(Analysis)、修改(Change)和生成(Generate)
- 选择正确的表示方法对机器学习处理至关重要

2. 图像表示(Image Representation)



RGB Color Space



Human Eye Sensitivity Curves

a) 光和颜色

- 图像本质上是光的记录(radiance)
- 人眼对光的感知主要体现在三个方面:
  - 强度敏感度(Intensity sensitivity): 人眼可以区分大约100个灰度级别,所以我们通常使用8位(256级)来存储强度信息
  - 色彩敏感度(Chromatic sensitivity): 人眼只能感知可见光谱的一部分
  - 空间敏感度(Spatial sensitivity): 人眼对细节的感知能力有限,取决于对比度

b) 颜色空间

- RGB颜色空间: 加色模型,适用于显示器等发光设备
- CMYK颜色空间: 减色模型,适用于打印等
- 其他颜色空间(LMS, Lab, YCrCb): 用于编辑和直观控制

c) 数字图像表示

- 像素数组表示: n行m列的数组
- 采样(Sampling)和锯齿(Aliasing)问题
    - 采样：将连续图像转换为像素数组
    - 锯齿问题：高频信息在采样时可能导致的失真
    - 解决方案：在采样前进行预滤波(Pre-filter)

d) 矢量图形(Vector Graphics)

- 不是按像素定义颜色(显式),而是定义图形的轮廓(隐式)
- 通过控制点定义图形
- 相比位图更容易缩放

3. 视频表示(Video Representation)

a) 时间敏感度

- 人眼对运动的感知有限制
- 时间采样率通常限制在25Hz左右
- 人眼无法看到过快的变化

b) 光流(Optical Flow)

- 描述视频中物体运动的方式
- 不仅仅是简单的2D+时间的3D空间
- 需要考虑帧间的对齐问题

c) 视频压缩和文件格式

- MPEG格式
- 类似JPEG的压缩方式
- 只存储关键帧
- 存储帧间的光流信息

4. 音频表示(Audio Representation)

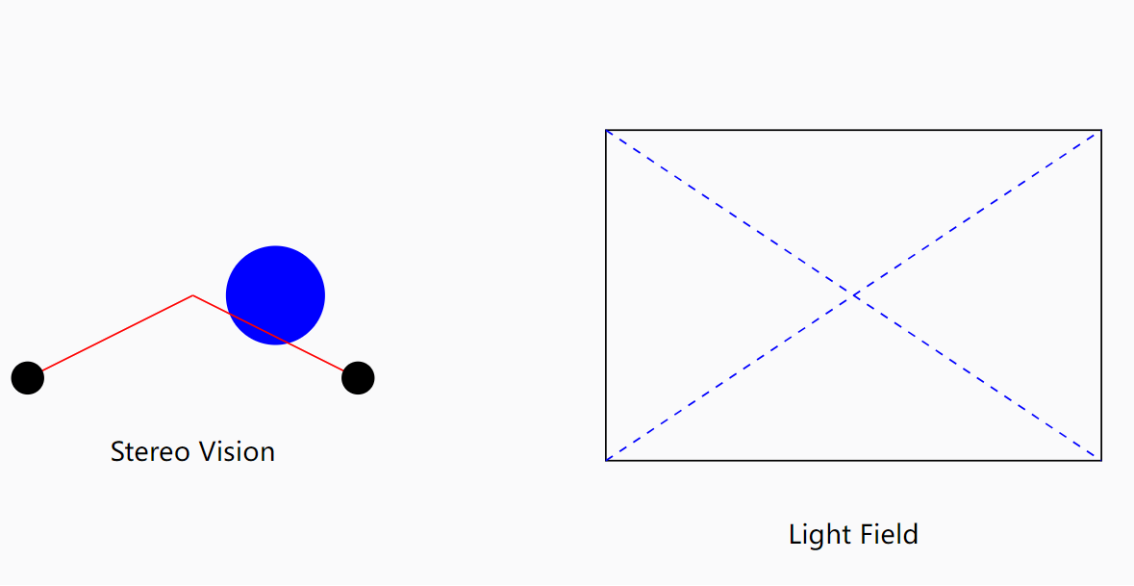
a) 基本概念

- 本质是空气压力的快速变化
- 分贝(Dezibel)：比率的对数
- 方(Phon)：等响度

b) 表示方式

- 波形(Waveform)：时域表示
- 频谱(Spectrum)：频域表示
- 相位(Phase)：信号不仅有幅度差异，还有相位差异
- 时频表示：同时表示时域和频域信息

5. 立体视觉和光场(Stereo and Light Field)



a) 立体视觉

- 单目图像：一个场景的一张图像
- 立体图像：一个场景的两张图像
- 视差(Disparity)：不同部分的移动差异反映深度信息

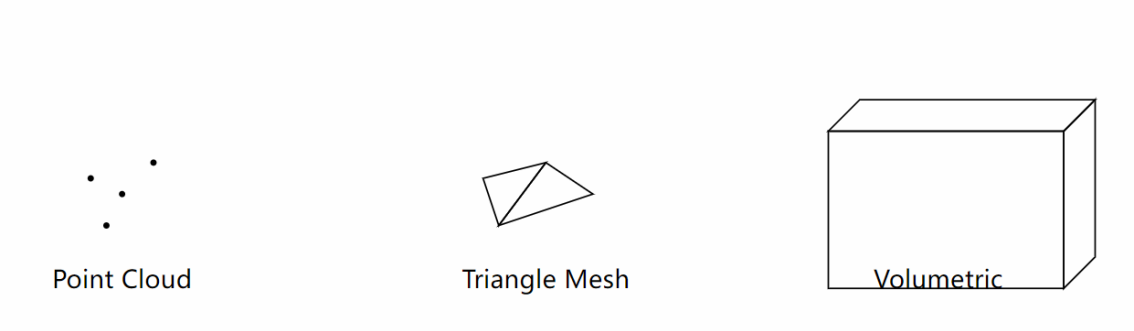
b) 光场

- 不是单一视点的图像
- 记录通过场景的所有光线
- 可以看作是多视角图像的堆叠
- 立体图像是n=2时的特例

c) 光场视频

- 结合光场和时间维度
- 具有更高的数据冗余性
- 为机器学习提供了很好的机会

6. 3D表面表示(3D Surface Representation)



a) 3D表面的主要表示方式：

- 3D体积(Volumes)
- 参数化表面(Parametric Surfaces)
- 点云(Point Clouds)
- 三角网格(Triangle Meshes)

b) 3D体积表示

- 从3D坐标到标量值的映射
- 适合表示内部结构
- 等值面(Iso-surfaces)：标量场取特定值的所有3D点集合

c) 参数化表面

- 从参数向量到坐标的映射
- 适合表示技术对象
- 常用于CAD设计

d) 点云

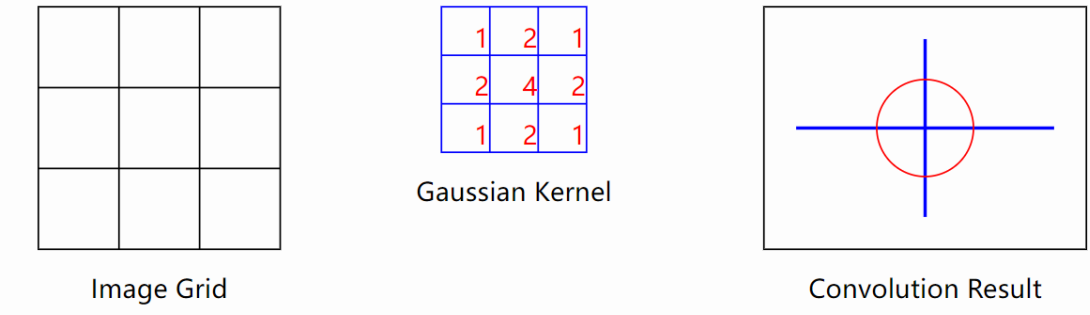
- 简单的3D点列表
- 通常来自3D扫描仪
- 缺乏拓扑信息

e) 三角网格

- 由顶点(vertices)和三角形(triangles)组成
- 可以看作是图结构
- 最常用的3D表示方式

第二部分：视觉处理(Visual Processing)

1. 图像卷积和滤波基础



a) 卷积的基本概念

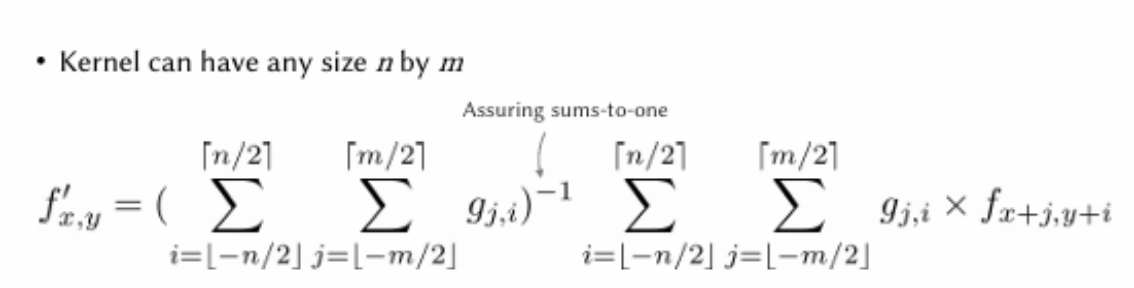
- 卷积是一种基本的图像处理操作，表示为 f * g, f 表示输入图像，g 表示卷积核（滤波器）。卷积的本质是加权求和操作
- 卷积的基本公式：
    1. 标准2D卷积输出大小计算：
    $O = \lfloor \frac{I+2P-K}{S} \rfloor + 1$
    - O：输出特征图大小; I：输入特征图大小; K：卷积核大小; P：填充大小(padding); S：步长(stride)
    2. 二维卷积运算：
    $(f * g)[i,j] = \sum_{m=0}^{k_h-1} \sum_{n=0}^{k_w-1} f[i+m, j+n] \cdot g[m,n]$

    其中：

- f：输入特征图; g：卷积核; k_h, k_w：卷积核的高度和宽度

b) 卷积核(Kernel)的组成

- 大小：通常为奇数×奇数（如3×3, 5×5）
- 权重：核中的每个位置都有一个权重值
- 归一化：权重之和通常需要为1（保持亮度不变）



c) 卷积计算过程

```
对于图像中的每个像素(x,y):
输出像素 = Σ(核权重 × 对应的输入像素)
具体过程：
1. 将核中心对齐到当前处理的像素
2. 将核内每个位置的权重与对应的输入像素值相乘
3. 将所有乘积相加
4. 将结果写入输出图像对应位置
```

2. 常见的图像滤波操作

a) 均值滤波（去噪）

- 最简单的滤波器，所有权重相等
- 核示例（3×3）：

```
1/9  1/9  1/9
1/9  1/9  1/9
1/9  1/9  1/9
```

b) 高斯滤波（平滑）

- 使用高斯分布作为权重
- 核示例（3×3）：

```
1/16  2/16  1/16
2/16  4/16  2/16
1/16  2/16  1/16
```

c) 边缘检测滤波器

- Sobel算子（检测水平边缘）：

```
-1  0  1
-2  0  2
-1  0  1
```

- Sobel算子（检测垂直边缘）：

```
-1  -2  -1
 0   0   0
 1   2   1
```

3. 多分辨率处理(Multi-resolution)

a) 图像金字塔

- 构建不同分辨率层次的图像序列
- 每一层是上一层的降采样版本
- 常见类型：
    - MIP图（游戏中常用）
    - 拉普拉斯金字塔（存储层级间的差异）
    - 可控金字塔（存储不同方向的差异）

b) 降采样操作（Pooling）

- 将2×2像素区域减少为1个像素
- 常见方法：
    - 平均值pooling
    - 最大值pooling
    - 最小值pooling

4. 边界处理（Border Handling）

处理图像边界的几种方法：

- 镜像（Mirror）：使用边界像素的镜像
- 重复（Repeat）：重复边界像素
- 常数（Constant）：使用固定值（如0）

5. 重要概念和注意事项

a) 3D卷积

- 可以同时处理多个通道（如RGB）
- 可以独立处理每个通道
- 也可以进行联合处理

b) 滤波器的选择

- 不同的任务需要不同的滤波器
- 滤波器大小的选择很重要
- 需要考虑计算效率

c) 实际应用中的考虑

- 需要处理边界条件
- 考虑计算效率
- 考虑数值精度

- 可能需要结合多种滤波器

C5 CNN CG

第一部分 - CNN的基本概念和发展:

1. 过滤器(Filters)的来源问题:

- 传统方法中,我们需要手动编写过滤器
- 但一个关键问题是:如何为特定任务选择正确的过滤器?
- 这就引出了让计算机自动学习过滤器的想法

2. 基于优化的过滤器学习:

- 我们可以通过优化方法来学习过滤器的参数
- 基本公式: $argmin_\theta||f \odot g_\theta - h||_2$
- 其中:
  - f 是输入图像
  - g_θ 是待优化的过滤器
  - h 是目标输出
  - ⊙ 表示卷积操作

3. 线性过滤示例:

- 用简单的例子展示了如何区分不同的图像特征
- 以豹子和斑马为例:
  - 豹子的特征是散点状的图案
  - 斑马的特征是平行条纹状的图案
- 通过不同的过滤器可以提取这些特征:
  - (1,1,1)/3 类型的过滤器用于识别点状特征
  - (-1,0,1)/2 类型的过滤器用于识别边缘特征

4. 特征识别和非线性:

- 对于点和边缘的检测:
  - 当特征存在时输出为1
  - 当特征不存在时输出为-1
- 示例解决方案:

```
点检测滤波器:          边缘检测滤波器:
-1 -1 -1              -1  2 -1
-1  8 -1              -1  2 -1
-1 -1 -1              -1  2 -1
```

第二部分 - CNN的架构演进与关键技术:

1. 非线性的重要性:

- CNN中非线性层是必不可少的
- 数学表达: $E[\varphi(f \odot g_\theta)] \neq \varphi(E[f \odot g_\theta])$

- 非线性使网络能够学习更复杂的特征
- 通常在每个卷积层后添加非线性激活函数

2. 深度架构与层级:

- 多层结构的优势:
  - 每层可以学习不同层次的特征
  - 低层学习基础特征(边缘、纹理)
  - 高层学习复杂特征(物体部件、整体结构)
- 数学表示: $\psi(\varphi_1(f \odot g_{\theta 1}) \odot g_{\theta 2})$

3. 多分辨率处理:

- 图像金字塔的概念引入
- 使用池化层(Pooling)降低分辨率:

```
输入分辨率层级:
256x256 -> 32x32 -> 1x1
```

- 池化操作有助于:
  - 减少计算量
  - 提供尺度不变性
  - 扩大感受野

4. 全连接层:

- 将卷积特征映射转换为分类结果
- 结构: $\psi(stack(pool(\varphi_1(f \odot g_{\theta 1})) \odot g_{\theta 2}))$
- 作用:
  - 整合所有空间信息
  - 进行最终的决策

第三部分 - 经典CNN架构:

1. LeNet (1990):

- 第一个成功的CNN架构
- 用于手写数字识别
- 特点:
  - 使用简单的卷积-池化结构; 层数较少(5层); 证明了CNN的可行性

2. AlexNet (2012):

- 深度学习革命的标志性网络
- 创新点:
  - 更深的网络结构，ReLU激活函数，Dropout正则化，GPU加速计算

3. 现代架构创新:

A. ResNet (2015):

- 引入残差连接
- 解决深层网络的梯度消失问题
- 数学表示: $\psi((\varphi_1(f \odot g_{\theta 1}) + f) \odot g_{\theta 2})$

B. 编码器-解码器架构:

- 用于图像到图像的转换任务
- 应用领域:
    1. 语义分割
    2. 深度估计
    3. 风格迁移

C. U-Net结构:

- 添加跳跃连接
- 保留细节信息
- 广泛应用于医学图像分割

第四部分 - 高级应用与挑战:

1. 图像识别与检测:

- 物体检测:
    - 同时预测位置和类别
    - 处理未知数量的物体
- 3D物体检测
- 语义分割

2. 图像合成与处理:

- 风格迁移
- 语义图像合成
- 视频处理

3. 面临的挑战:

A. 对抗样本问题:

- 微小扰动导致错误分类
- 例如: "pig" + 噪声 = "airliner"
- 暴露了CNN的脆弱性

B. 深度学习的局限性:

- 需要大量数据
- 计算资源消耗大
- 解释性差

C6 CNN

1. CNN的结构理解和特征可视化

(1) CNN的基础架构

- 输入接收原始图像(比如224×224×3的RGB图像)
- 通过多层卷积层和池化层提取特征
- 最后通过全连接层生成预测
- 以AlexNet为例,包含96个11×11×3的卷积核,经过多层后可以输出1000类预测

(2) CNN中的特征层次

- 第一层卷积核(浅层特征):
  - 学习简单的视觉特征如边缘、颜色、纹理
  - 不同网络(AlexNet/ResNet/DenseNet)的第一层都显示类似的模式
  - 这些基本特征构成了视觉认知的基础
- 中间层特征:
  - 组合浅层特征形成更复杂的模式
  - 可以识别特定的纹理和部件形状
  - 特征的抽象程度随深度增加而提高
- 深层特征:
  - 最后的全连接层(FC7)生成4096维特征向量
  - 捕获高层语义信息
  - 可用于分类、检索等任务

2. 特征理解和可视化技术

(1) 特征空间分析

- 近邻搜索比较:
  - 像素空间的近邻主要基于视觉外观相似度
  - 特征空间(FC7)的近邻能反映语义相似性
  - 说明CNN学习到了有意义的语义表示
- 降维可视化(PCA/t-SNE):
  - 将4096维特征压缩到2维便于可视化
  - 相似类别的样本在降维空间中聚集
  - 验证了特征的判别性

(2) 激活图分析技术

- Grad-CAM:
  - 通过计算梯度确定图像区域的重要性
  - 生成类激活热力图
  - 可视化网络关注的区域
- Guided Backprop:
  - 反向传播过程中保留正梯度
  - 可视化特定神经元对输入的响应
  - 有助于理解网络的决策依据

(3) 神经元激活分析

- 通过最大激活来研究单个神经元
- 可视化不同层神经元检测的特征
- 从简单几何特征到复杂物体部件

3. 高级特征生成和操作

(1) Feature Inversion(特征反演):$x^* = argmin_I l(\Phi(I), \Phi_0) + \lambda R(I)$

- 从特征重建原始图像
- 通过优化目标函数最小化特征差异
- 帮助理解特征的信息保留程度

(2) Gradient Ascent(梯度上升): $I^* = argmax_I \sum_i f_i(I)^2$

- 生成最大化激活特定神经元的图像
- 揭示网络学到的视觉模式
- 需要合适的正则化来生成自然图像

(3) DeepDream

- 通过放大网络内部激活来生成图像
- 显示网络检测的视觉模式
- 可以产生艺术风格的视觉效果

4. 实践应用

(1) 特征提取

- 使用预训练网络的FC7层作为特征提取器
- 特征可用于图像检索、分类等任务
- 展示了CNN特征的通用性

(2) 可视化应用

- 类激活图用于理解分类决策
- 特征反演用于网络解释
- 生成模型用于创意应用

详细部分：

1. 反向传播(Backpropagation)

- 基本原理:
  - 计算损失函数关于图像像素的梯度
  - 梯度反映了每个像素对最终输出的重要性
  - 通过链式法则逐层计算梯度
- 实现步骤:
  1. 前向传播计算网络输出
  2. 计算损失函数值
  3. 反向传播计算梯度

4. 根据梯度更新参数

2. 类激活映射(CAM, Class Activation Mapping)

- 核心思想:
  - 利用全局平均池化(GAP)层前的特征图
  - 使用类别权重对特征图加权求和
  - 生成类别特定的激活热力图
- 数学表达:

```
Sc = Σ_k wk,c Fk  (类别得分)
Mc,h,w = Σ_k wk,c fh,w,k  (激活映射)
```

其中:

- Fk: 第k个特征通道的全局平均值
- fh,w,k: 特征图在位置(h,w)的激活值
- wk,c: 第k个特征通道对类别c的权重

3. Grad-CAM(Gradient-weighted Class Activation Mapping)

- 改进:
  - 不需要修改网络结构(相比CAM)
  - 适用于任何CNN架构
  - 可以生成更精确的类别定位图
- 计算步骤:

1. 选择卷积层的激活 A ∈ ℝH×W×K
2. 计算类别得分Sc对A的梯度
3. 对梯度进行全局平均池化得到权重αk:$\alpha k = \frac{1}{HW} \sum_{h,w} \frac{\partial Sc}{\partial A_{h,w,k}}$
4. 计算加权激活图:$M^c_{h,w} = ReLU(\sum_k \alpha k A_{h,w,k})$

4. 梯度上升(Gradient Ascent)

- 目标:
  - 找到最大化某个神经元激活的输入图像
  - 通过反向传播优化输入图像
  - 揭示网络学到的视觉模式
- 优化问题:$I^* = argmax_I f(I) + \lambda R(I)$

其中:

- f(I): 目标神经元的激活值
- R(I): 正则化项
- λ: 正则化强度

5. GA正则化

- 主要正则化技术:
    1. L2正则化:控制图像范数
    2. 高斯模糊:保持空间平滑性
    3. 像素值裁剪:限制在合理范围
    4. 梯度裁剪:抑制过大变化
- 效果:
    - 生成更自然的图像
    - 避免高频噪声
    - 增强视觉模式的可解释性

6. Gram矩阵(Style Representation)

- 定义:特征图之间的内积矩阵$G_{ij}^l = \sum_{h,w} F_{h,w,i}^l F_{h,w,j}^l$

其中:

- l: 网络层
- i,j: 特征通道
- F: 特征图
- 应用:
    - 捕获图像的纹理统计信息
    - 用于风格迁移任务
    - 度量图像风格的相似性

实际应用示例:

1. 可视化类别判别依据:
2. 特征可视化:

这些技术的共同点是:

- 利用梯度信息理解网络行为
- 通过可视化揭示网络内部机制
- 提供网络决策的可解释性

它们的区别在于:

- 目标不同:定位(CAM)vs生成(GA)
- 计算方式不同:单次前向/反向传播vs迭代优化
- 应用场景不同:解释vs创造

C7 Gram Matrix和loss

第一部分,我们先来理解什么是神经风格迁移(Neural Style Transfer):

1. 基本概念:
   神经风格迁移是一项将一张图片的艺术风格应用到另一张图片上的技术。它需要三个关键组件:

- 内容图片(Content Image): 我们想要改变风格的原始图片
- 风格图片(Style Image): 提供我们想要模仿的艺术风格的图片

- 生成图片(Generated Image): 最终输出的结果,保持内容图片的内容但具有风格图片的风格特征

2. 经典案例示例:
   在课件中展示了几个典型例子:

- UCL建筑物(内容图片)与抽象艺术画作(风格图片)的结合
- Van Gogh的《星空》风格与各种照片的结合
  这些例子展示了如何将艺术家独特的绘画风格应用到普通照片上。

3. 为什么需要Gram矩阵:
   在进行风格迁移时,我们需要一种方法来捕捉和表示图像的"风格"。这就是为什么要引入Gram矩阵的原因。Gram矩阵能够捕捉图像特征之间的相关性,这些相关性实际上代表了图像的风格特征。
4. 网络架构:
   这个过程通常使用预训练的CNN网络(如VGG)。对于一张224×224×3的输入图像:

- 通过多个卷积层和池化层
- 最终得到4096维的特征向量
  这个网络架构可以提取图像的不同层次的特征,从低层的纹理到高层的语义信息。

第二部分: 纹理合成(Texture Synthesis)与Gram矩阵

1. 纹理合成的基本原理:
   纹理合成是风格迁移的一个重要基础。从课件中我们可以看到一个经典示例:通过一个小的纹理样本(比如那个绿色的网格结构),来生成更大的具有相同纹理特征的图像。这个过程包含几个关键步骤:

(a) 邻域匹配:

- 在原始纹理中定义一个邻域N(称为neighborhood)
- 对于要生成的图像中的每个像素p
- 寻找与其周围已生成区域最相似的纹理片段

2. Gram矩阵的定义和意义:
   Gram矩阵是风格迁移中最核心的概念之一。给定一个卷积层的特征图:

- 输入维度: C (通道数), H (高度), W (宽度)
- Gram矩阵计算公式: $G_{ij} := \sum_k F_{ik}F_{jk}$

这里详细解释一下这个公式:

- F是特征图矩阵
- i,j是通道索引
- k遍历所有空间位置(H×W)
- 最终得到的G是一个C×C的矩阵

3. Gram矩阵的物理意义:

- 它表示不同特征通道之间的相关性
- 捕捉了图像的统计特征,而不是具体的空间布局
- 这正好符合我们对"风格"的理解 - 更关注整体的视觉效果而不是具体的内容布局

4. 神经网络中的纹理合成过程:
   步骤如下:
   1. 计算VGG特征

2. 对输入图像I计算不同层的特征
3. 计算不同层的Gram矩阵
4. 用随机噪声初始化输出图像
5. 计算每一层的特征
6. 计算损失函数并反向传播更新像素值

损失函数定义为:
$E_l = \frac{1}{N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - \hat{G}_{ij}^l)$
$L(I) := \sum_l \alpha_l E_l$

这里:

- E_l 是每一层的损失
- N_l 是特征通道数
- M_l 是特征图大小
- α_l 是每层的权重系数

让我继续深入讲解风格迁移中的关键技术和损失函数。

第三部分：损失函数类型和特征

1. 主要损失函数类型：
   课程介绍了几种重要的损失函数，每种都有其特定的用途和特点：

- L2损失：最基本的均方误差损失
- MAE (Mean Absolute Error)：平均绝对误差
- SSIM (Structural Similarity Index)：结构相似性度量
- MSE/RMSE：均方误差/均方根误差
- LPIPS (Learned Perceptual Image Patch Similarity)：学习的感知图像块相似度

让我们重点关注SSIM和LPIPS这两个较为复杂但非常重要的损失函数：

2. SSIM (结构相似性指数)：
   SSIM是一个衡量图像质量的重要指标，它通过三个关键组件来计算：

亮度对比(l)：
$l(x,y) = \frac{2\mu_x\mu_y + a}{\mu_x^2 + \mu_y^2 + a}$

对比度(c)：
$c(x,y) = \frac{2\sigma_x\sigma_y + b}{\sigma_x^2 + \sigma_y^2 + b}$

结构(s)：
$s(x,y) = \frac{2\sigma_{xy} + c}{\sigma_x\sigma_y + c}$

最终的SSIM计算为：
$SSIM(x,y) = [l(x,y)]^\alpha [c(x,y)]^\beta [s(x,y)]^\gamma$

这里：

- μ表示平均值
- σ表示标准差
- a,b,c是稳定常数

- α,β,γ是权重参数

SSIM的优势在于它能够更好地模拟人类视觉系统对图像质量的感知。从课件展示的Einstein图像示例中我们可以看到，不同类型的图像失真（模糊、噪声、压缩等）会导致不同的SSIM值。

3. LPIPS（学习的感知图像块相似度）：

LPIPS是一个更现代的度量方法，它通过深度学习来模拟人类对图像相似度的判断。其计算公式为：

$$d(x, x_0) = \sum_i \frac{1}{H_l W_l} \sum_{h,w} ||w_l \odot (\hat{y}^l_{hw} - \hat{y}^l_{0hw})||^2$$

这个公式看起来复杂，让我解释一下各个部分：

- x和x_0是待比较的两张图片
- i表示网络的不同层
- h,w是特征图的空间位置
- w_l是学习的权重
- ⊙表示逐元素相乘

LPIPS的工作流程：

1. 将图像通过特征提取网络F
2. 对特征进行标准化和相减
3. 计算空间平均L2范数
4. 最后通过一个判别网络预测感知判断

这种方法的独特之处在于它结合了传统的图像度量和深度学习，能更好地捕捉人类感知中的相似性判断。

网络实现风格迁移的高级方法：

1. 前馈网络风格迁移：
   课件展示了一个更高级的实现方式，使用前馈网络(Feedforward Network)来实现实时的风格迁移。这种方法的架构包括：

- 输入：内容图像和风格图像
- 网络结构：一个前馈网络fw
- 损失网络：用于计算内容和风格的损失
- 输出：生成的风格化图像

这种方法相比传统的迭代优化方法有显著优势：

- 更快的推理速度
- 一次训练可以重复使用
- 风格迁移效果更稳定

2. 实际应用中的关键考虑：

训练策略：

- 从噪声开始，逐步优化生成图像
- 在多个层次上同时计算和优化损失
- 需要平衡内容保留和风格转换的程度

损失函数的选择：

- 内容损失：通常使用高层特征的L2距离
- 风格损失：使用Gram矩阵的差异
- 可以组合多种损失函数来获得更好的效果

3. 课程要点总结：

核心概念：

- Gram矩阵是捕获图像风格的关键工具
- 特征损失提供了度量图像相似度的多种方式
- 不同层次的特征对应不同级别的风格信息

C8 Sequence Learning

1. 序列学习概述

序列学习(Sequence Learning)是机器学习中一个重要的研究方向,主要处理具有时序关系或序列特性的数据。在视觉计算领域,序列学习有很多重要的应用,包括:

- 语音识别(Speech Recognition)
- 音乐生成(Music Generation)
- 机器翻译(Machine Translation)
- 视频生成(Video Generation)
- 情感分析(Sentiment Analysis)
- 音视频合成(Music-Video Synthesis)

2. CNN回顾及序列模型基础架构

在进入序列模型之前,让我们先回顾一下CNN的基本结构:

- CNN包含卷积层(Convolution+ReLU)
- 池化层(Pooling)
- 全连接层(Fully Connected)
- 最终输出预测结果

序列模型的基本设置包含:

1. 输入序列: x_1, x_2, ..., x_t
2. 输出序列: y_1, y_2, ..., y_t
3. 损失函数: $Lt(\hat{y}, y) = -ytlog\hat{y}t - (1 - yt)log(1 - \hat{y}t)$

3. 序列模型的四种基本映射类型

1. **一对一(One-to-One)映射**

- 最基础的映射形式
- 典型应用:图像分类
- 结构: 单个输入x通过模型fθ映射到单个输出y

2. **一对多(One-to-Many)映射**

- 单个输入生成多个输出序列
- 典型应用:图像描述(Image Captioning)
- 结构:一个输入x_1通过递归结构生成多个输出y_1,y_2,y_3

3. **多对一(Many-to-One)映射**

- 多个输入序列生成单个输出
- 典型应用:视频分类、动作检测
- 结构:多个输入x_1,x_2,x_3序列通过递归处理生成一个输出y

4. **多对多(Many-to-Many)映射**

- 输入输出都是序列
- 典型应用:条件图像生成、机器翻译、动画合成等
- 结构:输入序列x_1,x_2,x_3映射到输出序列y_1,y_2

让我继续深入讲解序列学习的核心内容。

4. RNN(循环神经网络)结构与原理

RNN是处理序列数据的基础模型，其核心特点是能够保持和利用历史信息。基本结构如下：

1. **RNN基本公式**:

- 隐状态更新: $h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t)$
- 输出计算: $y_t = W_{hh}h_t$

其中：

- ht 是当前时刻的隐状态
- xt 是当前输入
- Whh, Wxh 是权重矩阵
- tanh 是激活函数

2. **RNN的关键特性**:

- 参数共享：在序列的每个时间步使用相同的权重矩阵
- 记忆能力：通过隐状态ht保持历史信息
- 递归结构：当前输出依赖于之前的计算结果

5. RNN存在的问题

RNN在处理长序列时存在两个主要问题：

1. **梯度消失(Vanishing Gradients)**

- 当序列很长时，早期时间步的梯度会变得非常小
- 导致模型难以学习长期依赖关系
- 使得远距离的信息难以传递

2. **梯度爆炸(Exploding Gradients)**

- 梯度在反向传播时不断累积
- 导致权重更新过大

- 使模型训练不稳定

6. LSTM(长短期记忆网络)

为了解决RNN的问题，LSTM引入了更复杂的门控机制：

1. **LSTM的核心组件**:

```
- 遗忘门(f): 决定丢弃什么信息
- 输入门(i): 决定更新什么信息
- 输出门(o): 决定输出什么信息
- 单元状态(c): 长期记忆
```

2. **LSTM的数学表达式**:

```
[i, f, o, g] = [σ, σ, σ, tanh](W[ht-1, xt])
ct = f ⊙ ct-1 + i ⊙ g
ht = o ⊙ tanh(ct)
```

其中:

- ⊙ 表示元素级乘法
- σ 是sigmoid激活函数
- W是权重矩阵

3. **LSTM的优势**:

- 解决了长期依赖问题
- 控制信息流动更精确
- 训练更稳定

7. 链式LSTM

在实际应用中，我们经常将多个LSTM单元串联起来：

1. **链式结构特点**:

- 每个LSTM单元都接收上一单元的隐状态
- 可以处理变长序列
- 支持双向信息流动

8. 注意力机制(Attention Mechanism)

注意力机制的核心思想来源于人类的视觉认知系统。就像我们在看一幅画时会将注意力集中在特定区域一样，神经网络也可以学会"关注"输入中的重要部分。

8.1 注意力层的基本组成

注意力机制包含三个关键组件：

1. **Query (查询向量 Q)**
    - 代表当前需要关注的信息
    - 可以理解为"我想要找什么"

2. **Key (键向量 K)**
    - 代表信息的索引
    - 可以理解为"信息的标签"
3. **Value (值向量 V)**
    - 代表实际的信息内容
    - 可以理解为"实际的信息"

8.2 注意力计算过程

注意力机制的计算可以用公式表示为：

```
Attention(Q, K, V) = softmax(QK^T/√dk)V
```

这个过程可以分解为以下步骤：

1. 计算Query和Key的相似度：QK^T
2. 进行缩放：除以√dk（dk是Key的维度）
3. 使用softmax得到注意力权重
4. 用权重对Value进行加权求和

9. 注意力机制的类型

9.1 交叉注意力(Cross-Attention)

交叉注意力主要用于处理源序列和目标序列之间的关系：

1. **工作原理**
    - Query来自一个序列
    - Key和Value来自另一个序列
    - 常用于机器翻译等任务
2. **应用示例**
    - 图像描述任务中：
        - Query：当前要生成的文字
        - Key/Value：图像特征
        - 帮助模型关注图像中与当前描述相关的部分
3. **输入处理**:
    - Generated image features: 生成的图像特征
    - Text embedding: 文本嵌入向量
    - 通过不同的映射函数(fQ, fK, fV)生成Q、K、V

    注意力计算:

    ```
    Copy1. 计算Query和Key的相似度矩阵: QK^T
    2. 进行softmax归一化得到注意力权重
    3. 权重与Value相乘得到最终输出: (QK^T)V
    ```

    特点:

- 可以建立图像特征和文本特征之间的对应关系
- 实现精确的局部编辑
- 保持图像的整体一致性

9.2 自注意力(Self-Attention)

自注意力机制让序列中的每个位置都能够关注到序列中的其他位置：

1. **特点**
   - Query、Key、Value都来自同一个序列
   - 能捕捉序列内部的长距离依赖关系
   - 计算高效，可并行化
2. **实际应用**
   - 在图像生成中：帮助模型理解图像不同区域之间的关系
   - 在文本处理中：捕捉句子中词语之间的依赖关系
3. **计算过程**:

```
Copy1. 从同一组图像特征生成Q、K、V
2. 计算注意力图: Attention Maps (H×W)×(H×W)
3. 得到考虑了全局上下文的新特征表示
```

   应用优势:
   - 可以处理任意大小的输入
   - 计算效率高
   - 可以并行处理

10. 案例研究：图像描述生成

1. **输入处理**
   - 使用CNN提取图像特征
   - 将图像特征转换为Key和Value
   - 文本生成过程中的隐状态作为Query
2. **注意力计算**

```
ht = tanh(whhht-1 + wxhxt + wihv)
```

   其中：
   - ht-1 是上一时刻的隐状态
   - xt 是当前输入
   - v 是通过注意力机制得到的图像特征的上下文向量
3. **生成过程**
   - 模型在每一步都会关注图像的不同部分
   - 注意力权重可视化显示模型在生成不同词时关注的图像区域
   - 这种机制使得生成的描述更准确且与图像内容更相关

11. Transformer架构

Transformer的提出源于论文"Attention is All You Need"。这个标题非常形象地表达了其核心思想：我们可以完全依赖注意力机制来处理序列数据，不需要RNN或CNN这样的递归或卷积结构。

Transformer的基本架构

一个完整的Transformer包含以下关键组件：

1. **输入嵌入层（Input Embedding）**
   - 将输入序列转换为固定维度的向量表示
   - 加入位置编码（Positional Encoding）以保留序列中的位置信息
   - 位置编码使用正弦和余弦函数，这样可以表达相对位置关系
2. **Multi-Head Self-Attention（多头自注意力）**
   让我们深入理解这个核心组件：

```
# 多头注意力的计算过程
def multi_head_attention(query, key, value, num_heads):
    # 1. 线性变换，将输入分成多个头
    Q = linear_transform(query)  # [batch, seq_len, d_model]
    K = linear_transform(key)    # [batch, seq_len, d_model]
    V = linear_transform(value)  # [batch, seq_len, d_model]

    # 2. 分割成多个头
    Q_heads = split_into_heads(Q, num_heads)  # [batch, num_heads, seq_len, d_k]
    K_heads = split_into_heads(K, num_heads)
    V_heads = split_into_heads(V, num_heads)

    # 3. 对每个头计算注意力
    attention_outputs = []
    for head in range(num_heads):
        score = dot_product(Q_heads[head], K_heads[head]) / sqrt(d_k)
        attention = softmax(score)
        output = dot_product(attention, V_heads[head])
        attention_outputs.append(output)

    # 4. 合并多头的输出
    return concat_and_project(attention_outputs)
```

3. **Layer Normalization（层归一化）**
   - 在每个子层的输入上应用归一化
   - 帮助训练稳定性
   - 公式：LayerNorm(x) = γ * (x - μ) / (σ + ε) + β
   - 其中μ是均值，σ是标准差，γ和β是可学习参数
4. **Feed-Forward Network（前馈网络）**
   - 两层线性变换，中间有ReLU激活
   - 第一层扩展维度，第二层压缩回原始维度
   - 数学表示：FFN(x) = W2(ReLU(W1x + b1)) + b2
5. **残差连接（Residual Connections）**

- 每个子层都有残差连接
- 帮助解决深层网络的梯度问题
- 公式：x + Sublayer(LayerNorm(x))

Transformer的工作流程

让我用一个具体例子来说明Transformer的工作流程：

假设我们要处理一个图像描述任务：

1. **预处理阶段**：

```
输入图像 → CNN特征提取 → 特征图
文本提示 → 词嵌入 → 位置编码
```

2. **注意力计算阶段**：

```
- 自注意力：图像特征间的关系
- 交叉注意力：图像特征和文本的关系
```

3. **特征整合阶段**：

```
- Layer Normalization
- 残差连接
- 前馈网络处理
```

Transformer的优势

1. **并行计算能力**
   - 所有位置可以同时计算
   - 大大提高了训练和推理速度
2. **全局依赖关系**
   - 可以直接建立序列中任意位置的联系
   - 避免了RNN中的长程依赖问题
3. **灵活性**
   - 可以处理变长序列
   - 易于扩展和修改
4. **模型可解释性**
   - 注意力权重可视化
   - 直观理解模型的决策过程

12. 高级应用: 语义分割

基于注意力机制的语义分割是一个重要应用:

1. Self-Segmentation
   - 利用自注意力特征的语义信息
   - 通过聚类(clustering)得到分割结果

- 不需要额外的标注数据

2. 实现过程

```
Copy1. 提取自注意力特征图 (hw × (h×w))
2. 对特征进行聚类得到语义分割
3. 为每个区域分配标签
```

3. 区域标注(Segments Labeling)
    - 使用cross-attention特征判断区域内容
    - 计算每个区域与文本描述的相关性
    - 为每个区域分配最相关的标签

C9 Generative Modeling

第一部分

1. 生成式建模的基础概念

生成式建模是机器学习中一个革命性的领域，它让机器能够"创造"而不仅仅是"识别"。

2. 学习范式的演进

机器学习在处理视觉任务时有两种主要的学习范式：

监督学习(Supervised Learning):

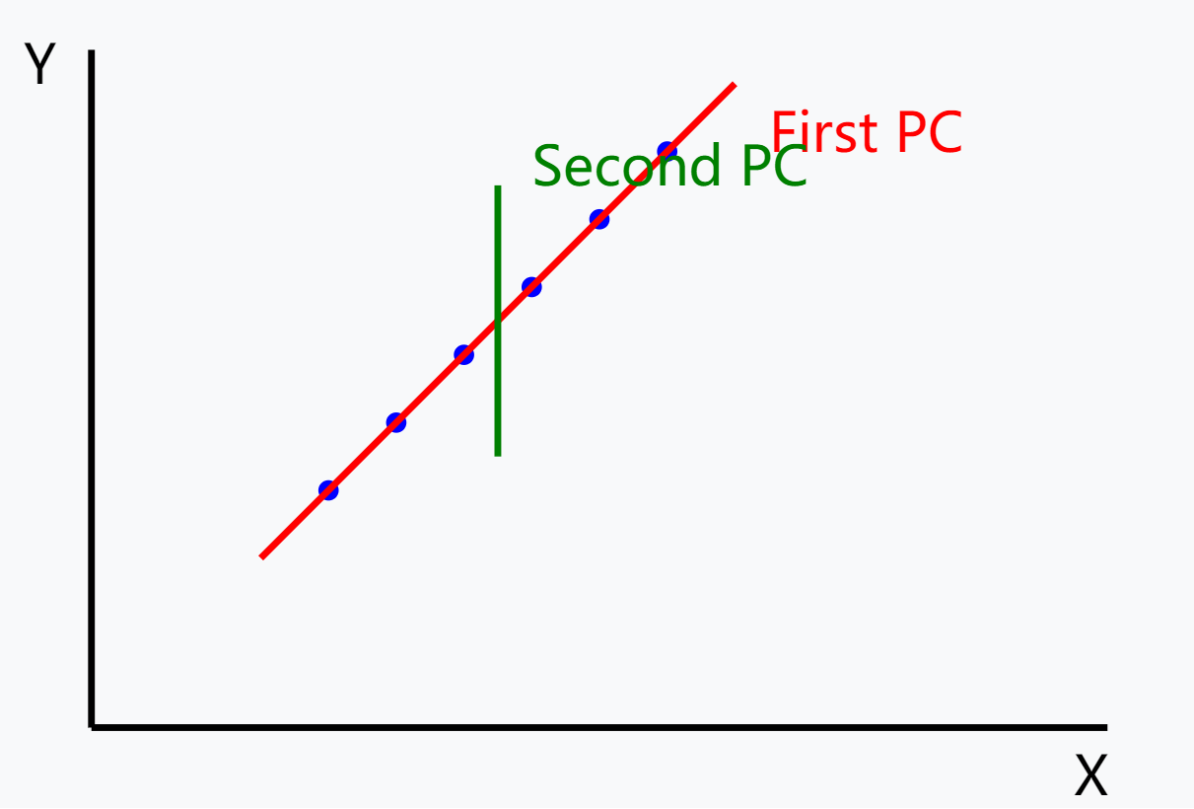
- 需要大量带标签的数据
- 学习输入和标签之间的明确映射
- 适用于分类、检测等任务
- 缺点是获取标签成本高

无监督学习(Unsupervised Learning):

- 直接从原始数据中学习
- 主要包括聚类和降维两大类任务
- 更接近人类的自然学习过程
- 不需要昂贵的标注数据

3. 降维技术：主成分分析(PCA)

PCA是理解生成模型之前的重要基础。



PCA通过四个关键步骤找到数据的主要变化方向：

- 数据标准化：消除量纲影响
- 计算协方差矩阵：描述特征间的相关性
- 特征值分解：找到主要变化方向
- 投影变换：降维的同时保留关键信息

4. 自编码器(AutoEncoder)

自动编码器是生成模型的重要基础。它的核心思想是通过"压缩"再"重建"来学习数据的本质特征：

- 编码器(Encoder)：将高维输入压缩到低维潜在空间
- 解码器(Decoder)：将低维表示重建为原始输入
- 训练目标：最小化重建误差 $x \approx D(E(x))$

这种结构让模型能够学习到数据的紧凑表示，为后续的生成任务奠定基础。

5. 生成模型的分类体系

根据密度估计的方式，生成模型可以分为：

显式密度模型(Explicit Density)：

- 可直接计算的(Tractable)：如自回归模型
- 近似的(Approximate)：如VAE

隐式密度模型(Implicit Density)：

- 直接生成(Direct)：如GAN

这种分类体系反映了不同方法在处理概率分布时的策略差异。

6. 变分自动编码器(VAE)

VAE是自动编码器的概率化版本，它的关键创新在于：

- 将确定性的潜在编码改为概率分布
- 使用重参数化技巧实现随机采样
- 通过变分下界优化目标函数

这让VAE不仅能重建已有数据，还能生成新的样本。从课件的CIFAR-10示例可以看到，VAE能生成多样的、自然的图像。

7. 自回归模型(Autoregressive Models)

自回归模型是一种非常直观的生成式方法，它的核心思想是"逐步生成"。就像Bob Ross画画的例子展示的那样，画作是一笔一笔完成的，每一笔都依赖于之前所有的笔触。

在数学上，自回归模型将联合概率分解为条件概率的乘积：
$p(x) = p(x_1, x_2, \ldots, x_n) = \prod_i p(x_i|x_1, \ldots, x_{i-1})$

这种分解方式具有几个重要特点：

1. 顺序依赖性：

- 每个输出都依赖于之前的所有输出
- 这种依赖关系使模型能够捕捉到数据中的长期依赖关系

2. 可处理性：

- 虽然生成过程是顺序的，但每一步只需要生成一个低维的概率分布
- 比如在图像生成中，每一步只需要预测一个像素的强度值

一个典型的应用是PixelRNN，它的工作原理是：

- 从左上角开始，按照固定顺序生成每个像素
- 使用循环神经网络(RNN)维护历史信息的状态
- 每一步输出一个像素值的概率分布

8. 变分自编码器(VAE)的深入理解

VAE是现代生成式模型的重要里程碑,它的数学原理值得深入理解：

(1) 概率生成过程：

- 首先从先验分布p(z)采样潜在变量
- 然后通过条件分布p(x|z)生成观测数据
- 整个过程可以写作：$p(x) = \int p(x|z)p(z)dz$
- VAE不是直接学习一个确定性的编码，而是学习概率分布
- 编码器输出均值μ和方差Σ，用于参数化高斯分布
- 这种设计使模型具有生成能力，能够产生新样本

(2) 变分推断：

- 后验分布p(z|x)通常难以直接计算
- 使用变分分布q(z|x)来近似
- 最小化两个分布之间的KL散度

(3) 变分下界(ELBO)：
$logp(x) \geq E[logp(x|z)] - KL(q(z|x)||p(z))$

- 左边是对数似然(我们想最大化的目标)
- 右边第一项是重建项(希望生成的数据接近原始数据)
- 右边第二项是正则化项(让潜在空间分布接近标准正态分布)

9. VAE的编码器和解码器

VAE中的编码器和解码器都输出高斯分布的参数：

编码器：

- 输入：原始数据x
- 输出：均值μz|x和方差Σz|x
- 表示了潜在空间中的不确定性

解码器：

- 输入：潜在变量z
- 输出：均值μx|z和方差Σx|z
- 描述了生成过程中的不确定性

10. VAE的应用展示

课件最后展示了VAE的两个重要应用：

(1) 采样(Sampling)：

- 从标准正态分布采样z
- 通过解码器生成新数据
- CIFAR-10数据集上的结果展示了模型可以生成多样的、符合自然图像分布的样本

(2) 编辑(Editing)：

- 在潜在空间中进行插值和操作
- 生成连续变化的图像序列
- 人脸图像示例展示了VAE学习到了有意义的潜在表示

11. 生成对抗网络(GAN)的原理

GANs引入了一个全新的训练范式，它包含两个网络的对抗过程：

1. 生成器(Generator)：

- 目标是生成逼真的样本
- 从随机噪声开始，逐步学习生成目标分布的样本
- 试图骗过判别器，让它无法区分真假样本

2. 判别器(Discriminator)：

- 目标是区分真实样本和生成的样本
- 不断提高鉴别能力，迫使生成器产生更好的样本
- 输出样本的真实概率

这种对抗训练有几个特点：

- 不需要显式定义概率分布

- 可以生成非常逼真的样本
- 训练过程可能不稳定

这三种模型的比较：

- 自回归模型：可处理性好，但生成速度慢
- VAE：训练稳定，但生成质量略差
- GAN：生成质量最好，但训练不稳定

好的,让我继续深入讲解生成式建模的高级应用和数学原理。

12. GAN的进阶应用

条件GAN (Conditional GAN):

- 核心思想是在生成过程中加入条件信息
- 最著名的应用是Pix2Pix,实现了图像到图像的转换
- 判别器不仅要判断图像的真假,还要判断输入-输出对的合理性
- 数学表达式: $min_G max_D E[logD(x,y) + log(1-D(x,G(x)))]$

实际应用包括:

1. 边缘图到实物图转换
2. 草图到真实图片转换
3. 语义分割图到场景图转换
4. 黑白图像到彩色图像转换

StyleGAN的创新:

1. 风格分离

- 将潜在空间z转换为更有结构的中间潜在空间w
- 不同层次的风格可以分别控制
    - 粗粒度:整体姿态、构图
    - 中等粒度:面部特征、发型
    - 细粒度:颜色、纹理

2. 渐进式生成

- 从低分辨率开始,逐步提高分辨率
- 每个阶段都会添加新的卷积层
- 这种策略使训练更加稳定

13. VAE的数学原理与伪代码

VAE的核心是变分推断,其目标函数包含两部分:

1. 重构误差:$E[logp(x|z)]$
2. KL散度正则项:$KL(q(z|x)||p(z))$

完整的ELBO(Evidence Lower BOund)公式:$logp(x) \geq E_z\ q(z|x)[logp(x|z)] - KL(q(z|x)||p(z))$

VAE训练的伪代码:

```
# 初始化参数
encoder_params = θ  # 编码器参数
decoder_params = ϕ  # 解码器参数

for epoch in range(num_epochs):
    for batch in data_loader:
        # 编码器前向传播
        μ, σ = encoder(batch)

        # 重参数化采样
        ϵ ~ N(0,I)  # 从标准正态分布采样
        z = μ + σ * ϵ  # 重参数化技巧

        # 解码器重构
        x_recon = decoder(z)

        # 计算损失
        recon_loss = -E[log p(x|z)]  # 重构误差
        kl_loss = KL(N(μ,σ)||N(0,I))  # KL散度

        total_loss = recon_loss + kl_loss

        # 反向传播和优化
        optimizer.zero_grad()
        total_loss.backward()
        optimizer.step()
```

14. VAE与GAN的深入比较

VAE的优势:

1. 训练稳定性

- 有明确的目标函数
- 不存在模式崩塌问题
- 优化过程更可控

2. 良好的潜在空间结构

- 潜在空间连续且有意义
- 支持插值和属性操作
- 便于下游任务使用

3. 可以同时实现生成和推断

- 编码器提供推断能力
- 解码器提供生成能力

局限性:

1. 生成质量不如GAN

- 重构损失导致模糊
- KL散度可能过度正则化
- 难以捕捉锐利的细节

GAN的优势:

1. 卓越的生成质量

- 不依赖像素级重构损失
- 能生成锐利细节
- 视觉质量最接近真实

2. 灵活的架构设计

- 可以适应各种任务
- 支持条件生成
- 易于扩展新功能

局限性:

1. 训练不稳定

- 需要精心设计损失函数
- 容易出现模式崩塌
- 收敛过程难以控制

2. 缺乏推断能力

- 无法直接进行特征提取
- 需要额外的编码器网络
- 潜在空间结构较差

在实际应用中，我们经常需要根据具体任务选择合适的模型:

- 需要高质量生成 → GAN
- 需要特征提取 → VAE
- 需要可控生成 → StyleGAN
- 需要图像转换 → Pix2Pix

这些模型也在不断发展，比如:

- VAE-GAN: 结合两者优势
- BiGAN: 为GAN添加推断能力
- StyleGAN2/3: 改进生成质量和控制能力

第二部分

1. **语义任务和对象检测**

在生成模型中,语义任务通常包含以下几个关键组成部分:

a) 图像分割:

- 从原始图像到语义分割图的转换(如猫的图像转为区域分割)
- 每个像素都被赋予一个语义类别标签
- 通常使用不同颜色来表示不同的语义区域

b) 对象检测:

- 目标是在图像中定位和识别物体
- 通常输出包含边界框(bounding boxes)
- 可以同时检测多个对象
- 常用方法包括:
    - 单对象检测:专注于检测单一目标
    - 多对象检测:同时处理多个目标的检测任务

2. **检测多个对象的方法**

主要有两种经典方法:

a) 滑动窗口方法(Sliding Window Approach):

- 在图像上使用固定大小的窗口逐步滑动
- 对每个窗口位置进行对象检测
- 优点是简单直接
- 缺点是计算量大且效率较低

b) R-CNN(Region-based CNN)方法:

- 使用选择性搜索(Selective Search)来提出可能包含对象的区域
- 对这些候选区域进行特征提取和分类
- 相比滑动窗口更有效率
- 是现代目标检测的重要基础

好的,我会更详细地讲解后面这些较难的概念。

3. **扩散过程(Diffusion Process)的本质**

扩散过程可以被理解为在两种分布之间建立映射的方法:

a) 基本概念:

- 从一个未知的数据分布(比如真实图像)映射到一个已知的简单分布(通常是标准正态分布)
- 这个过程是双向的,包含正向过程(forward process)和反向过程(reverse process)
- 目标是学习这种转换,使得我们可以从简单分布生成复杂的真实数据

b) 数学表示:

- 数据分布: x_0 (原始数据,如图像)
- 目标分布: x_t (通常是标准正态分布 N(0,I))
- 转换过程: $x_0 \rightarrow x_{t-1} \rightarrow x_t$

4. **高斯(正态)分布的特性与应用**

高斯分布在生成模型中扮演着核心角色:

a) 基本特征:

- 由均值(μ)和方差(Σ)唯一定义: $N(\mu, \Sigma)$
- 具有良好的数学性质,易于操作和计算

b) 重参数化技巧(Reparameterization Trick):

- 公式: $y_i = \mu_i * x_i + \sigma_i$
- 其中 x $N(0,1)$ 是标准正态分布
- 这个技巧让我们可以通过简单的线性变换从标准正态分布得到任意的高斯分布
- 在训练过程中特别有用,因为它使得梯度可以正常传播

c) 组合性质:

- 高斯分布的线性组合仍然是高斯分布
- 这个性质在设计生成模型时非常有用

5. **多步映射过程**

这是扩散模型的核心机制:

a) 正向过程:

- 从原始数据x_0开始
- 通过多个步骤逐渐加入噪声
- 每一步的条件概率由如下公式给出:
  $q(x_t|x_{t-1}) = N(x_t; \sqrt{\alpha_t}x_{t-1}, (1-\alpha_t)I)$

b) 数学推导:

- $x_t = \sqrt{\alpha_t}x_0 + \sqrt{(1-\alpha_t)}\varepsilon_t$
- ε_t是标准正态噪声
- α_t是时间步长t的调度参数
- $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$ 表示累积的缩放因子

这个过程可以被看作是逐步将清晰的图像转换为纯噪声的过程。每一步都是可逆的,这就为生成过程提供了理论基础。

好的,让我从反向过程开始继续详细讲解。

6. **生成建模：反向过程(Reverse Process)**

反向过程是扩散模型中最关键的部分,它实现了从噪声到有意义数据的转换:

a) 基本原理:

- 从噪声分布x_t开始 (通常是N(0,I))
- 逐步去除噪声,最终得到清晰的图像x_0
- 整个过程是原始扩散过程的逆向操作

b) 数学表达:

- $p(x_{t-1}|x_t) = N(x_{t-1}; \sqrt{\alpha_{t-1}}D\theta(x_t, t), (1-\alpha_{t-1})I)$
- Dθ是一个神经网络,用于预测去噪步骤
- θ是需要学习的参数

c) 关键特点:

- 每一步都是条件概率分布
- 使用神经网络来学习最优的去噪过程
- 整个过程是可训练的

7. **损失函数(Loss Functions)**

在扩散模型中有三种等价的损失函数解释:

a) 简单损失函数:$L_{simple}(\theta) = E_{t,x_0,\varepsilon}[C_t||\varepsilon\theta(x_t, t) - \varepsilon||^2]$

- 预测添加的噪声
- 直接优化均方误差
- C_t是时间相关的权重

b) 预测损失函数:$L(\theta) = E_{t,\varepsilon,x_0}[C_t||D\theta(\varepsilon_t(x_0), t) - x_0||^2]$

- 直接预测原始图像
- 基于重建质量优化

c) 基于分数的损失:

- 使用梯度匹配
- 优化score function: $\nabla x_t logp(x_0)$
- 理论上与其他两种方法等价

8. **训练过程**

训练过程涉及以下关键组件:

a) 输入处理:

- ε_t: 随机噪声
- x_0: 原始数据
- t: 时间步长
- $x_t = a_t x_0 + b_t \varepsilon$: 噪声数据

b) 网络结构:

- 使用U-Net作为基础架构
- 条件时间嵌入
- 多尺度特征处理

c) 训练步骤:

1. 采样噪声和时间步长
2. 生成带噪声的数据
3. 通过网络预测
4. 计算损失并更新参数

这个训练过程是迭代的,需要多次重复直到模型收敛。

9. **视觉数据的特殊性**

视觉数据具有一些独特的特点:

a) 维度特征:

- 高维数据空间
- 结构化的空间关系

- 局部和全局特征的层次性

b) 生成特性:

- 推理速度和多样性要求
- 需要维持视觉真实性
- 语义一致性的重要性

c) 特定损失函数:

- 感知损失
- 对抗性损失
- 语义保持损失

10. **潜在扩散模型(LDM)的基本架构**

潜在扩散模型的核心思想是在压缩的潜在空间中进行扩散过程,而不是在原始像素空间中操作。让我们逐步分析其组成部分:

a) 主要组件:

1. 编码器(E):

- 将输入图像 X 压缩到潜在空间
- 降低数据维度,提高计算效率
- 保留图像的关键语义信息

2. 解码器(D):

- 将潜在表示 Z 重建回像素空间
- 生成最终的图像输出 $\tilde{X}$
- 需要确保重建质量

3. 潜在空间扩散过程:

- 在压缩的潜在空间Z中进行
- 使用U-Net作为降噪网络
- 可以更高效地处理高分辨率图像

b) 工作流程:

1. 输入阶段:

```
原始图像(X) → 编码器(E) → 潜在表示(Z)
```

2. 扩散阶段:

- 在潜在空间中添加噪声
- 训练U-Net进行降噪
- 条件控制(如文本提示)在这个阶段起作用

3. 生成阶段:

```
潜在表示(Z) → 解码器(D) → 重建图像(X̃)
```

c) LDM的创新点:

1. 计算效率:

- 在压缩空间中操作大大减少了计算成本
- 使得处理高分辨率图像变得可行
- 训练速度显著提升

2. 灵活的条件控制:

- 可以轻松集成文本、图像等多种条件信息
- 支持跨模态生成
- 便于实现各种创意应用

3. 质量保证:

- 编码器-解码器架构确保了重建质量
- 潜在空间的连续性有助于生成平滑过渡
- 可以更好地保持图像的细节和语义

d) 条件控制机制:

1. 语义映射:

- 将文本描述转换为条件向量
- 指导扩散过程的方向
- 实现文本到图像的精确控制

2. 表示学习:

- 学习不同模态的统一表示
- 支持多样的输入条件
- 提高生成结果的可控性

这个模型架构的重要性体现在:

1. 它是Stable Diffusion等流行图像生成模型的基础
2. 大大降低了计算资源需求
3. 为高质量图像生成提供了可扩展的解决方案
4. 使得实际应用中的实时生成成为可能

第三部分

1. **扩散模型中的无条件采样 (Unconditional Sampling)**

首先让我解释什么是无条件采样。在扩散模型中,无条件采样是指直接从随机噪声生成图像,不需要任何额外的条件输入。这个过程可以用以下公式表示:

$p(x) \approx \nabla x \log p(x)$

这里的p(x)是概率密度,右边的项叫做score function(评分函数)。这个过程就像是从一团混沌的噪声中逐渐"清晰化"出一个图像。

2. **条件采样 (Conditional Sampling)**

条件采样是在生成过程中加入了额外的控制条件y。比如你想生成一只猫的图片,这个"猫"就是条件y。数学表达式变成:

$p(x|y) \approx \nabla x \log p(x|y) = \nabla x \log p(y|x) + \nabla x \log p(x)$

这个公式可以分解为两部分:

- $\nabla x \log p(x)$: 无条件部分
- $\nabla x \log p(y|x)$: 条件项(由分类器提供)

3. **Classifier Guidance(分类器引导)**

这是一个重要的技术创新。通过引入guidance scale(引导尺度)γ,我们可以控制条件的影响强度:

```
γ = 1: 正常的条件生成
γ > 1: 生成更高质量、更符合条件的样本
γ = 10: 非常强的条件控制
```

比如在PPT中的例子,使用γ = 10时生成的狗的图像更加清晰和规范。

4. **Classifier-free Guidance(无分类器引导)**

这是对分类器引导的改进。主要创新点是:

- 不需要额外的分类器
- 在训练时使用dropout策略(10-20%的时间将条件y替换为空值)
- 可以得到更好的条件控制效果

其数学表达式是:
$p(x|y) = (1-\gamma)\nabla x \log p(x) + \gamma \nabla x \log p(x|y)$

5. **条件信号的提供方式**

条件信号通常通过两种注意力机制提供:

- Self-attention: 处理图像内部的关系
- Cross-attention: 处理图像和条件之间的关系

6. **IP-Adaptor技术**

这是一个专门用于图像提示的适配器,它可以:

- 将图像编码为特征
- 通过解耦的交叉注意力机制融合特征
- 实现更精确的图像控制