深度学习

目 录

图 片

表 格

返回目录

1 学术史

姓名	图灵奖	诺贝尔奖
Geoffrey Hinton	√	√
John Hopfield	x	√

表 1: 奖项获得情况

2 神经网络

2.1 概念

深度学习的“深度”：神经网络的深度（层数）。

数据集

- 训练集。
- 测试集。
- 验证集。

参数

- 超参数：网络层数、学习率（策略）、批量大小。不能只在训练集或测试集上获得，要通过训练和验证获得。
- 非超参数：网络权重（不唯一，比如倍数）。

2.2 前馈神经网络（Feed-Forward Neural Network，FNN）

相邻层间特征单向连接。

返回目录

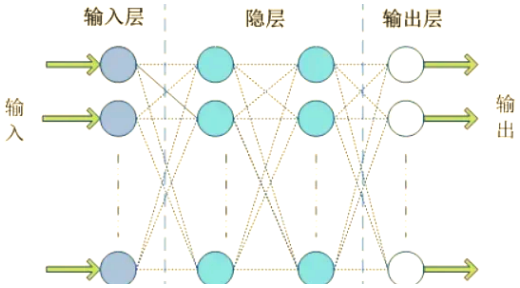


图 1: 前馈神经网络

全连接神经网络（Full Connect Neural Network，FCNN）

- 前向传播：计算结果并保存特征。
- 反向传播：链式规则。

2.3 卷积神经网络（Convolutional Neural Networks，CNN）

2.3.1 结构

- 卷积层：提取局部特征。
- 池化层：降低特征维度。
- 全连接层：分类。

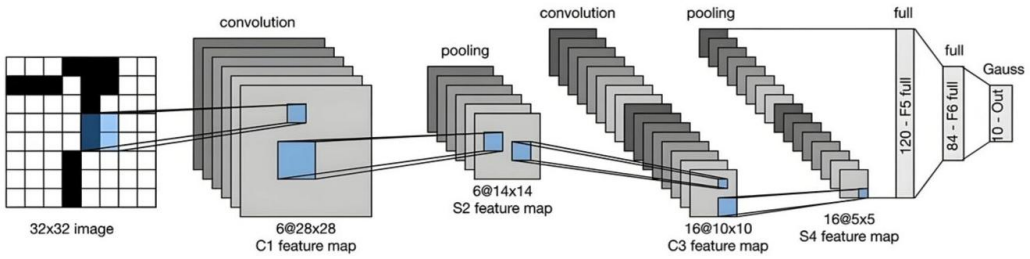


图 2: 卷积神经网络

卷积

计算：

返回目录

- 原图像：$H_0 \times N_0 \times M_0 \times A_0$。
- 卷积核：$B \times F \times F \times A_0$。
- 填充：P。
- 步幅：S。
- 新图像大小：$N_1 = \frac{N_0+2P-F}{S} + 1, A_1 = B$。
- 参数量：$F \times F \times A_0 \times B$。

1×1卷积：

- 维度变换（降/升维）：改变特征图的深度（通道数）。降维有助于降低模型复杂度和计算量，同时保持大部分有用信息。
- 非线性引入：在卷积后添加激活函数，可以在不改变空间尺寸的情况下引入非线性，使模型能够学习更复杂的模式。
- 作为瓶颈层：在一些架构（如ResNet、Inception）中，可以用作瓶颈层（先降维，再进行其他卷积，最后恢复维度）。显著减少参数量和计算成本，同时维持性能。
- 特征融合：融合不同尺度或不同来源的特征图，合并成一个新的特征表示。

池化 参数量为0，计算区域的代表性特征值。

- 最大池化（Max Pooling）：最大值。
- 平均池化（Average Pooling）：平均值。

填充（PADDING）

- 增加感受野，减少信息损失：确保边缘像素能被卷积核充分覆盖，得到有效处理，而不是丢失。
- 控制输出尺寸：通过调整填充量可以精确控制每一层的输出尺寸。

返回目录

步幅（STRIDE）

- 控制输出尺寸、下采样程度：较大步幅可以减小输出的空间尺寸，降低计算复杂度，减少参数量，提高特征图的缩放比例。
- 调整感受野：较大步幅时输出单元会覆盖较大输入区域，增加感受野，减少重叠区域数量。
- 平衡速度与精度：较大步幅可加速计算过程，但可能丢失细节信息；较小步幅能更精细地捕捉特征，但会增加计算成本。

2.3.2 实例

趋势：卷积核变小、层数增加，抛弃池化层、全连接层。

- LeNet：没有使用ReLU。
- AlexNet：最早使用了ReLU、GPU。
- VGGNet：小卷积核（感受野上，$3 \times 3 \times 3 = 1 \times 7 \times 7$）。
- GoogleNet：使用了ReLU，Inception。1×1卷积。
- ResNet：使用了ReLU，恒等映射直连边，残差模块。

3 问题与改进

3.1 问题

3.1.1 拟合

问题	描述	训练收敛的网络	
		训练误差	测试误差
欠拟合	未训练收敛	大	大
过拟合	在训练集上表现良好，但不能推广到测试集	小	大

表 2: 欠拟合和过拟合

返回目录

3.1.2 梯度

	定义	问题	改善方法
梯度消失	网络层数增加，反向传播过程中梯度逐渐变小，甚至趋于零。	权重更新缓慢，甚至停止更新，影响模型的学习能力，使网络难以训练。	使用ReLU及其变体。 采用批量规范化。 选择Xavier初始化或He初始化）。
梯度爆炸	反向传播过程中梯度变得非常大。	权重更新过大，模型学习过程不稳定，甚至发散，无法收敛到最优解。	使用ReLU及其变体。 采用梯度裁断。 使用Adam优化器。 采用批量规范化，稳定梯度，减小影响。

表 3: 梯度消失和梯度爆炸

3.2 改进技术

3.2.1 损失函数

交叉熵损失函数更适用于分类问题，常用于衡量模型预测的概率分布与真实标记的概率分布之间的差异。

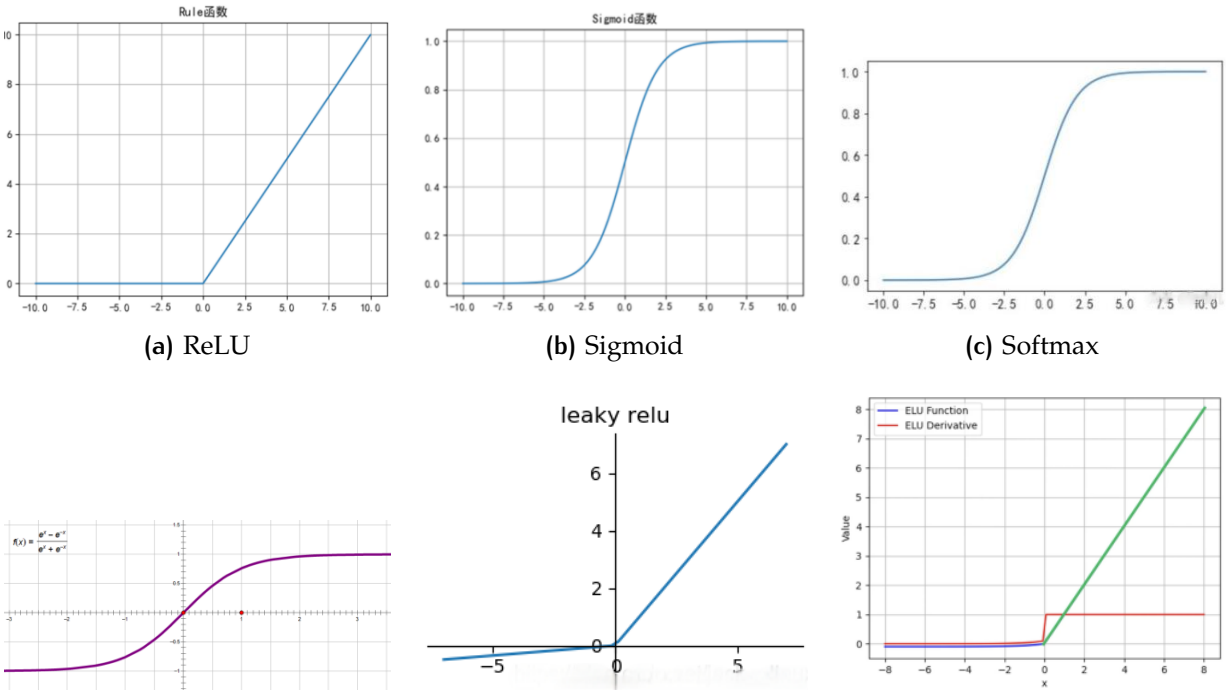
返回目录

3.2.2 激活函数

激活函数	函数	优点	缺点
ReLU	$y = \max(0, x)$ $y' = \begin{cases} 1, & z > 0 \\ 0, & z \leqslant 0 \end{cases}$	计算简单，收敛快。解决了梯度消失问题，允许误差迅速回传。有助于稀疏激活，更简洁且减少过拟合风险。	存在死区问题（神经元输入小于0时将永远不被激活）。 非零中心化，导致下一层权重更新不均衡。
Sigmoid	$y = \frac{1}{1+e^{-z}}$ $y' = y(1-y)'$	输出范围固定在$(0,1)$，适用于二分类问题。函数平滑且处处可微，有利于梯度计算。	非零中心化，正输出，减慢收敛速度。 计算成本高。
Softmax	对单个定义$y = e^x$ $y(x_i) = \frac{e^{x_i}}{\sum_{i=1}^{n} e^{x_i}}$	适用于多分类问题。	存在数值溢出问题，不适合隐藏层。
Tanh	$y = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	零中心化输出，收敛性能好，适用于二分类问题。	易饱合，梯度消失。
Leaky ReLU	$y = \max(0.1x, x)$		
ELU	$y = \begin{cases} x, & x \geqslant 0 \\ a(e^x - 1), & x < 0 \end{cases}$		
Maxout	$y = \max(w_1^T x + b_1, w_2^T x + b_2)$		

表 4: 激活函数对比

返回目录



(a) ReLU (b) Sigmoid (c) Softmax

(d) Tanh (e) Leaky ReLU (f) ELU

图 3: 激活函数图像

RELU与SIGMOID对比

- ReLU优点：
  - 梯度问题：ReLU在正区间内不会导致梯度消失；Sigmoid容易陷入饱和区，在区域内梯度几乎为零，导致梯度消失。
  - 计算效率：ReLU只需判断输入是否大于0，Sigmoid涉及指数运算。
  - 稀疏激活：ReLU有助于产生稀疏激活，使模型更简洁，减少过拟合风险。
  - 收敛速度：ReLU可使随机梯度下降更快收敛。
- Sigmoid优点：
  - 输出范围：Sigmoid将输入映射到$(0,1)$，对特定任务（如二分类问题）更有效。
  - 适用于特定架构或任务：在RNNs中，Sigmoid被用于门控机制，控制信息流。
  - 非线性表达能力：Sigmoid在定义域上平滑且处处可微，非线性表达能力强。

3.2.3 优化器

- 二阶算法，牛顿法。

返回目录

- SGD。
- SGD + Momentum。
- ADam。
- ADamW。

3.2.4 权重初始化

方法	特点	优点
高斯分布随机初始化	基于正态分布，零均值有方差。	确保信号在传播中既不会迅速消失也不会爆炸。
Xavier初始化	确保前向、反向传播过程中激活值和梯度方差保持一致。	适用于使用Tanh或Sigmoid的网络。
Kaiming均匀分布初始化（He初始化）		适用于使用ReLU及其变体的网络。

表 5: 权重初始化方法对比

3.2.5 正则化

- 丢弃法（dropout）：在训练过程中随机失活部分神经元（输出0），减少对训练集的过度依赖，阻止特征间自适应。可以防止过拟合，提高鲁棒性，间接减小计算量，实现模型平均效果。
- L1正则化：使权重稀疏。
- L2正则化：是权重分散。

3.2.6 训练策略/技巧

- 批量规范化（Batch Normalization）：归一化每个小批量样本为标准正态分布，减少梯度消失和梯度爆炸问题，加速收敛。提高模型稳定性、泛化能力，减少对其他正则化技术的依赖。
- 早停。
- 梯度裁断（Gradient Clipping）：限制梯度大小，防止其变得过大

返回目录

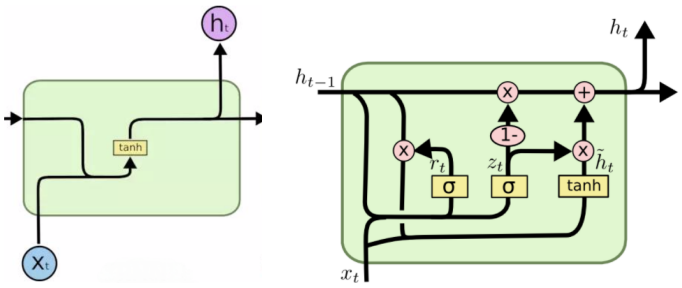
4 循环神经网络

4.1 典型网络

4.1.1 RNN（Recurrent Neural Network，循环神经网络）

循环核计算：

- 隐藏状态更新：$h_t = f_w(h_{t-1}, x_t) = \tanh(w_{xh}x_t + w_{hh}h_{t-1} + b_h)$。
- 输出生成：$y_t = f_{why}(h_t) = Softmax(w_{hy}h_t + b_y)$。



(a) RNN (b) GRU

图 4: RNN与GRU更新单元

4.1.2 GRU（Gated Recurrent Unit networks，门控循环单元网络）

重置门和更新门，没有细胞状态。相比于RNN，主要改善了梯度消失问题。

4.1.3 LSTM（Long Short-Term Memory networks，长短期记忆网络）

遗忘门、输入门、输出门中使用Sigmoid激活函数。

- 细胞状态（Cell State，黄色）：贯穿模型，根据遗忘门和输入门的结果，在整个序列上进行少量的线性交互，使信息流动而不发生太多改变。
- 遗忘门（Forget Gate，红色）：决定从细胞状态中丢弃的信息，查看前一时刻的隐藏状态和当前输入，输出$[0, 1]$的值给细胞状态的每个元素。0意味着完全丢弃，1意味着完全保留。
- 输入门（Input Gate，蓝色）:，包含一个用于确定要更新部分的Sigmoid层，和一个创建新候选值向量的Tanh层。

返回目录

- 输出门（Output Gate，绿色）：决定细胞状态的输出。Sigmoid层决定细胞状态的输出部分，Tanh层缩放细胞状态，与Sigmoid门的输出相乘，得到本时刻输出。

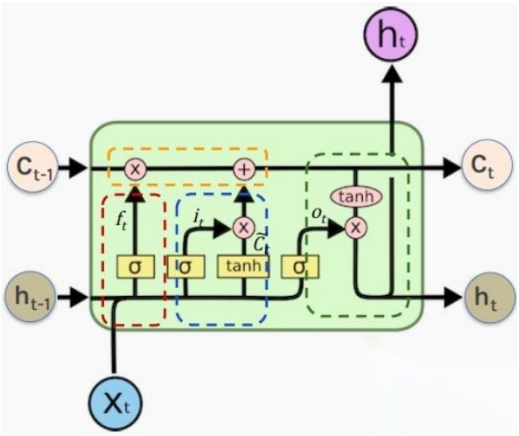


图 5: LSTM更新单元

5 TRANSFORMER

5.1 注意力机制

组件

- 键（Key）。
- 值（Value）。
- 查询（Query）。

多头自注意力（MULTI-HEAD SELF-ATTENTION）

1. 对于输入序列$X = [x_1, x_2, \ldots, x_n]$，其维度是$[n, d_{model}]$。
2. 由权重矩阵W^Q, W^K, W^V生成查询、键、值，维度是$[d_{model}, d_k], d_k < d_{model}$：

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V$$

3. 将其分割成h个头，每个头维度为$\frac{d_k}{h}$：

$$Q_{split} = Split(Q), \quad K_{split} = Split(K), \quad V_{split} = Split(V)$$

返回目录

4. 对每个头计算注意力分数（点乘查询与键，d_k 缩放，Softmax，加权求和）：

$$\text{Attention}(Q, K, V) = \text{Softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

5. 合并头，其中W^o是一个可学习参数矩阵，维度为$[hd_k, d_{model}]$，用来向d_{model}维映射：

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, head_2, \ldots, head_h)W^o$$

在注意力机制中使用位置编码 原始的自注意力机制不包含序列中元素顺序信息，计算仅基于元素间关系。为使模型能有效处理序列数据，并理解不同元素间的相对或绝对位置关系，引入位置编码。位置编码保持了序列信息，使其可以学习更复杂的模式，有助于提高模型的理解能力和表达能力。另外，位置编码还使得模型可以处理任意长度的序列，支持变长输入。

5.2 Transformer

掩蔽注意力（MASKED ATTENTION） 确保模型在生成序列时不依赖未来信息，只关注到它之前的元素。这能模拟真实场景，保证因果关系，维持逻辑性和连贯性，对于生成符合语法和语义规范的句子非常重要。

5.3 生成式模型

步骤

- 预训练（Pre-training）：在大型、通用数据集上训练模型，让模型学习到输入数据中的基本特征和模式。
- 微调（Fine-tuning）：在预训练后，将模型应用到具体任务上，并根据该任务的相关数据集进行进一步训练，调整参数，以优化特定任务的表现。
- 分两步：
  - 迁移学习：预训练模型已经在大量数据上学到了丰富的特征，在新任务上只需进行少量调整就可以获得良好效果。
  - 充分利用数据，适应不同任务需求：有效结合预训练大数据集带来的泛化能力和微调小数据集中特有的领域信息。

返回目录

  - 节省时间和成本：获取大规模标注数据是困难而昂贵的，预训练-微调允许在较小规模的数据集上快速开发和部署模型，大大减少了从零开始训练所需的时间和计算资源，同时还能保持较高的性能水平。

自监督学习 使用输入中的一部分预测另一部分，即将部分输入作为标签，无须人工标注标签。

- 数据效率：使模型能够利用丰富的未标注资源学习有用的表示，而不需要为每个样本都提供昂贵的手工标注。
- 泛化能力：通过在广泛无关数据上训练，可以学到更通用的语言理解能力和其他领域知识，提高模型对不同任务的适应性和泛化能力。
- 减少过拟合风险：相比于在小规模标注数据集上直接训练，在更大规模的数据集上预训练减少了过拟合风险。
- 加速后续学习：预训练得到的特征可以帮助模型更快地收敛到特定任务的最佳解。

6 其它

6.1 物体检测与图像分割

- 语义分割与实例分割。
- 物体检测：R-CNN，FastR-CNN。
- 转置卷积：可学习的上采样方法。

6.2 可视化与理解

- 可视化：卷积核、输出层。
- 理解：重要像素、显著性。
- 对抗性扰动。
- 风格迁移。

返回目录