人工智能引论

L1历史

- 历史会议，七个问题，标志性事件

  时间：1956年夏天（6月至8月）。

  地点：美国新罕布什尔州的达特茅斯学院。

  发起人：由计算机科学家 **约翰·麦卡锡**（John McCarthy）、**马文·明斯基**（Marvin Minsky）、**克劳德·香农**（Claude Shannon）、**内森·罗切斯特**（Nathaniel Rochester）共同发起。

  目的：提出人工智能（Artificial Intelligence，AI）这一新领域，探索如何使机器模拟人类的智能行为。

- 神经元是怎么回事

  神经的可塑性：赫伯法则(Hebbian rule)

  同时被激活的神经元之间的连接被强化，产生新的链路，形成新的记忆 (fire together, wire together)，对概念的记忆存储于由密切连接的神经元组成的细胞群中，激活其中的部分神经元可以唤起对整个概念的记忆

- 智能是和什么特殊相关

  大脑皮层中的神经元数量（?

L2

- 如何搜索

  盲目搜索（深度优先，宽度优先），启发式搜索（A算法，A*算法）

- **皇后问题**

  深度搜索的样例问题

- 两种盲目搜索比较

  深度搜索：对搜索深度加以限制（解决深度问题），记录从初始状态到当前状态的路径（解决死循环问题）

  宽度搜索：

  - 当问题有解时，一定能找到解
  - 当问题为单位耗散值，且问题有解时，一定能找到最优解
  - 方法与问题无关，具有通用性
  - 效率较低且存储量比较大

- **启发式图搜索**（A与A*）

  优先扩展“最佳”节点，是在利用知识来引导搜索，达到减少搜索范围并提高搜索效率的目的

```
OPEN := {s}, f(s) := g(s) + h(s);
LOOP: IF OPEN = ( ) THEN EXIT(FAIL);
n := FIRST(OPEN);
IF GOAL(n) THEN EXIT(SUCCESS);
REMOVE(n, OPEN), ADD(n, CLOSED);
EXPAND(n) → {m_i}, 计算 f(n, m_i) := g(n, m_i) + h(m_i);
OPEN 中的节点按 f 值从小到大排序;
GO LOOP;
```

其中关键在于节点扩展`EXPAND(n) → {m_i}, 计算 f(n, m_i) := g(n, m_i) + h(m_i);`

邻居处理逻辑：扩展节点 n的所有邻居m_i

1. **计算邻居代价**：
   - `g(n, m_i)`：n 到 m_i的实际代价；
   - `h(m_i)`：m_i到目标的启发式估计；
   - `f(n, m_i) = g(n, m_i) + h(m_i)`
2. **邻居状态更新**：
   - 如果 m_i 不在 OPEN 和 CLOSED 中，将其加入 OPEN，并记录 m_i 指向 n 的指针（便于回溯路径）；
   - 如果 m_k已在 OPEN 中，且通过 n 的路径代价更低更新f(m_k)，并修改m_k 的父节点为 n；
   - 如果 m_l在 CLOSED 中，且通过 n 的路径代价更低，将 m_l 从 CLOSED 移回 OPEN，并更新其代价和父节点。

L3

- **a-b剪枝**

  再过一遍例子

  剪枝的条件：

  后辈节点的b值（极小节点上界）≤ 祖先节点的a值（极大节点下界）时， a剪枝

  后辈节点的a值 ≥ 祖先节点的b值时， b剪枝

  伪代码：

```
function AlphaBeta(node, depth, α, β, maximizingPlayer):
    if depth == 0 or node is a leaf:
        return Evaluate(node)  # 评估函数返回当前节点的分数

    if maximizingPlayer:  # Max 层
        maxEval = -∞
        for child in Children(node):
            eval = AlphaBeta(child, depth - 1, α, β, False)
            maxEval = max(maxEval, eval)
            α = max(α, eval)
            if β <= α:  # 剪枝
                break
        return maxEval
    else:  # Min 层
        minEval = ∞
        for child in Children(node):
```

```
            eval = AlphaBeta(child, depth - 1, α, β, True)
            minEval = min(minEval, eval)
            β = min(β, eval)
            if β <= α:  # 剪枝
                break
        return minEval
```

- **蒙特卡洛搜索**（四个步骤）
  - 选择：在现有的搜索树中，从根节点开始，根据某种策略选择一个最值得探索的子节点，直到到达一个叶子节点（尚未完全扩展的节点）；
  - 扩展：如果所选叶子节点未终止游戏，则从该节点随机选择一个可行的操作，生成一个或多个子节点，并将其添加到树中；
  - 模拟：从新扩展的节点开始，进行一次随机模拟（也称为 **playout**），即随机选择动作直到游戏结束，计算该局的结果（胜利、失败或平局）；
  - 回溯跟新：将模拟结果沿搜索树回溯更新，调整每个相关节点的统计信息（如胜率），以反映新信息。

    3. 关键公式

    在选择阶段，通常使用 **UCB1 (Upper Confidence Bound 1)** 公式来平衡探索和利用：

    $$UCB1 = \frac{W_i}{N_i} + c\sqrt{\frac{\ln N_p}{N_i}}$$

    - W_i: 当前节点的累积奖励（胜利次数）。
    - N_i: 当前节点被访问的次数。
    - N_p: 父节点被访问的次数。
    - c: 探索因子，控制探索与利用的平衡（通常取 $\sqrt{2}$ 或其他适当值）。
    - **第一项** ($\frac{W_i}{N_i}$)：表示利用（选择当前最优子节点）。
    - **第二项** ($c\sqrt{\frac{\ln N_p}{N_i}}$)：表示探索（倾向选择访问次数较少的子节点）。

    通过 UCB1，MCTS 能在不确定性较高的情况下优先探索，同时利用已有的优良路径信息。
- 运用和探索的概念

L4

- 头（？）概念
- 评价需要会（F1）

	Correct	Not Correct
Selected	Truth Positive (TP)	False Positive (FP)
Not Selected	False Negative (FN)	Truth Negative (TN)

F1 值的公式为：

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

---

2. 相关概念

精确率 (Precision)

精确率衡量的是模型预测为正的样本中，真正为正的比例：

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- **TP (True Positive)**：预测为正且实际为正的样本数。
- **FP (False Positive)**：预测为正但实际为负的样本数。

召回率 (Recall)

召回率衡量的是所有实际为正的样本中，模型预测为正的比例：

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- **FN (False Negative)**：预测为负但实际为正的样本数。

L5

- K-means
  1. **初始化簇中心：** 随机选择 (K) 个点作为初始簇中心（Centroids）
  2. **分配数据点：** 对于每个数据点 (x_i)，计算它到 (K) 个簇中心的距离（通常使用欧氏距离），将其分配到最近的簇，欧氏距离公式：

$$d(x_i, c_k) = \sqrt{\sum_{j=1}^{d} (x_{ij} - c_{kj})^2}$$

  3. **更新簇中心：** 对每个簇，重新计算簇的中心点，作为该簇所有数据点的均值：

$$c_k = \frac{1}{|C_k|} \sum_{x_i \in C_k} x_i$$

  4. **重复步骤 2 和 3：** 直到簇分配不再变化，或者簇中心的变化小于预设阈值，或者达到最大迭代次数。
- EM算法，大家需要了解

  E步 (Expectation Step)：基于当前的模型参数，计算隐变量的后验分布或期望；

  M步 (Maximization Step)：基于 E 步的结果，重新估计模型参数，使得对数似然函数最大化。
- 隐马尔可夫模型

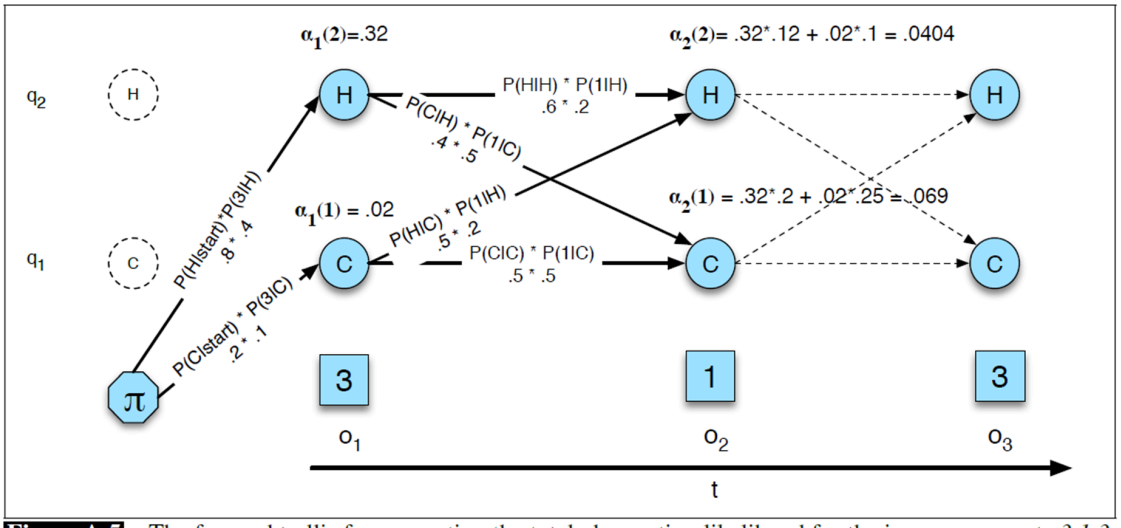


Figure A.5 The forward trellis for computing the total observation likelihood for the ice-cream events *3 1 3*. Hidden states are in circles, observations in squares. The figure shows the computation of $\alpha_t(j)$ for two states at two time steps. The computation in each cell follows Eq. A.12: $\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t)$. The resulting probability expressed in each cell is Eq. A.11: $\alpha_t(j) = P(o_1, o_2 \ldots o_t, q_t = j|\lambda)$.

L6

- 三步骤（基础模型，loss， 优化）
  1. **构建基础模型**
     - **定义网络结构**：根据任务需求选择模型类型，例如线性模型、神经网络、卷积网络等。设计模型的层数、每层的神经元数量和连接方式等。
     - **选择激活函数**：例如 ReLU、Sigmoid 或 Tanh，用于引入非线性能力。
     - **初始化参数**：为模型的权重和偏置赋初始值（随机初始化、Xavier 初始化、He 初始化等），以确保训练过程的收敛性。
     - **输入与输出**：明确模型的输入数据形状和预期输出。例如，对于图像分类，输入可能是像素矩阵，输出是类别标签。
  2. **定义损失函数（Loss Function）**

     损失函数用于量化模型预测值与真实值之间的差异。
     - **目标**：损失值越小，表示模型的预测越接近真实值。
     - **选择损失函数**：如回归任务中，常用均方误差（MSE）或均绝对误差（MAE）；如分类任务中，常用交叉熵损失（Cross-Entropy Loss）或 Hinge Loss；如在其他任务中，例如强化学习中的奖励函数、自定义损失函数等
     - **作用**：损失函数的输出指导优化器更新模型参数。
  3. **优化模型**

     优化的目标是通过更新模型参数，使损失函数的值逐步减小。
     - **优化算法**：常用优化方法有梯度下降（SGD）、动量优化（Momentum）、自适应方法（如 Adam、Adagrad、RMSProp）；
     - **梯度计算**：使用反向传播算法（Backpropagation）通过链式法则计算损失对参数的梯度；
     - **参数更新**：基于梯度下降原理，按一定学习率（Learning Rate）更新参数，使损失函数逐渐收敛。
- 什么是梯度下降

  梯度下降是一种**优化算法**，用于通过迭代调整模型参数，**最小化损失函数**的值，从而提高模型的预测性能。它是许多机器学习和深度学习算法的核心。

假设损失函数为 $\mathcal{L}(\theta)$，参数为 θ，学习率为 η，梯度下降的更新公式为：

$$\theta \leftarrow \theta - \eta \cdot \frac{\partial \mathcal{L}(\theta)}{\partial \theta}$$

- $\frac{\partial \mathcal{L}(\theta)}{\partial \theta}$：损失函数对参数的梯度，表示损失随参数变化的方向和速率。
- η（学习率）：控制每次更新的步长，步长过大可能导致跳过最优点，步长过小可能导致收敛过慢。

梯度下降的类型

1. **批量梯度下降 (Batch Gradient Descent)**
   - 在每次更新时，使用**所有训练数据**计算梯度。
   - 优点：精确计算梯度，方向稳定。
   - 缺点：计算代价高，特别是数据量大时。
2. **随机梯度下降 (Stochastic Gradient Descent, SGD)**
   - 在每次更新时，使用**单个样本**计算梯度。
   - 优点：更新快，适合大数据集。
   - 缺点：更新方向可能不稳定，会在最优点附近波动。
3. **小批量梯度下降 (Mini-Batch Gradient Descent)**
   - 在每次更新时，使用**一个小批量数据**计算梯度（通常为 32、64 或 128 个样本）。
   - 优点：折中计算代价和稳定性，现代深度学习中最常用。

- 概念（batch,epoch弄清楚是什么回事）

1. Batch（批量）

- **定义**：将训练数据划分为较小的子集，每次训练只使用一个子集的数据进行模型参数更新。
- **作用**：
  - 提高训练效率，避免在处理大规模数据时内存不足。
  - 通过小批量数据的随机性帮助模型逃离局部最优。
- **类型**：
  - **Batch Size（批大小）**：每个 batch 中包含的样本数量。常见的值是 32、64、128 等。
    - 小的 batch size：更接近随机梯度下降，更新频繁但可能噪声较大。
    - 大的 batch size：更接近批量梯度下降，方向稳定但计算成本高。
  - 通常通过实验确定最佳的 batch size。

2. Epoch（周期）

- **定义**：一个 epoch 表示模型完成了对整个训练数据集的一次完整遍历。
- **关系**：
  - 假设训练集有 10,000 个样本，batch size 为 100，那么完成一个 epoch 需要 $\frac{10000}{100} = 100$ 次 batch 更新。
- **作用**：通常需要多个 epoch 来让模型逐渐收敛，但过多的 epoch 可能导致过拟合。

3. Iteration（迭代）

- **定义**：一次 iteration 指模型通过一个 batch 完成一次前向传播和反向传播。
- **关系**：
  - 一次 epoch 包含多次 iteration，数量为 $\text{Number of Iterations} = \frac{\text{Training Set Size}}{\text{Batch Size}}$。

10. Regularization（正则化）

- **定义**：通过对损失函数添加惩罚项，抑制模型过拟合。
- **常见方法**：
  - L1 正则化：惩罚参数的绝对值。
  - L2 正则化：惩罚参数的平方。
  - Dropout：随机丢弃部分神经元，减少模型复杂度。

总结：概念间关系

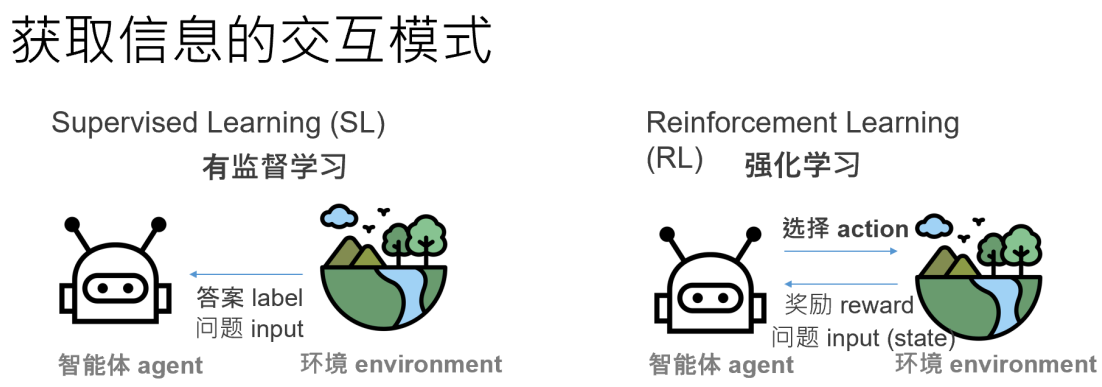
- **一次 iteration** 是模型完成一个 batch 的训练。
- **一个 epoch** 包含多次 iteration，是对整个训练数据集的完整遍历。
- **batch size** 决定了每次 iteration 处理的样本数量。
- **学习率** 和正则化方法影响模型的优化过程和最终性能。

这些概念协同作用，构成了机器学习模型训练的基础流程。

- 主要激活函数

L7

- 无监督学习和强化学习区别



对比总结

特性	有监督学习	无监督学习	强化学习
输入数据	带标签数据 (X, Y)	无标签数据 (X)	状态 (S) 和动作 (A)
目标	学习输入与输出的映射关系	挖掘数据模式和结构	最大化累计奖励
学习过程	基于标签直接优化	基于数据模式提取	试错 + 奖励反馈
典型任务	分类、回归	聚类、降维	游戏、机器人控制
优势	准确预测、清晰目标	数据需求低，无需标注	处理动态、不确定性问题
劣势	需大量标注数据	难以验证模型效果	计算成本高，复杂性大

	有监督学习	无监督学习	强化学习
学习依据	基于监督信息	基于对数据结构的假设	基于评价（evaluative）
数据来源	一次性给定	一次性给定	在交互中产生（interactive）
决策过程	单步（one-shot）	无	序列（sequential）
学习目标	样本到语义标签的映射	同一类数据的分布模式	选择能够获取最大收益的的状态到动作的映射

强化学习的特点

- **基于评估：**强化学习利用环境评估当前策略，以此为依据进行优化
- **交互性：**强化学习的数据在与环境的交互中产生
- **序列决策过程：**智能主体在与环境的交互中需要作出一系列的决策，这些决策往往是前后关联的

注：现实中常见的强化学习问题往往还具有奖励滞后，基于采样的评估等特点

强化学习最大的不同在于有和环境的交互过程；而监督学习和无监督学习都仅仅只有数据本身，不存在交互

- 马尔可夫决策过程（要看懂走迷宫的例子）

  PPT第27页

- 三种策略评估的区别

对比总结

特性	动态规划（DP）	蒙特卡洛方法（MC）	时序差分方法（TD）
依赖模型	需要环境模型	无需环境模型	无需环境模型
更新方式	全状态更新	全轨迹更新	单步更新
计算效率	高效，但要求状态空间较小	样本效率低，需多次采样	高效，无需等待完整轨迹
适用场景	小规模离散状态空间	情节型任务	情节型与非情节型任务均可
优点	理论严谨，收敛快	简单直观，无需模型	结合 DP 和 MC 的优点，更新高效
缺点	依赖模型，扩展性差	高方差，需完整轨迹	有偏差，对学习率敏感

L8

- 了解语言模型发展的过程

  n元模型（序列每个位置的单词只依赖于前n-1个位置上的单词，马尔可夫链，交叉熵）-> 基于矩阵分解的语义分析（term-document矩阵，奇异值分解，LSI案例）-> **神经语言模型**（GPT用于语言生成，BERT用于语言理解，分别对应Transformer的解码器与编码器）

- 神经网络重要人物

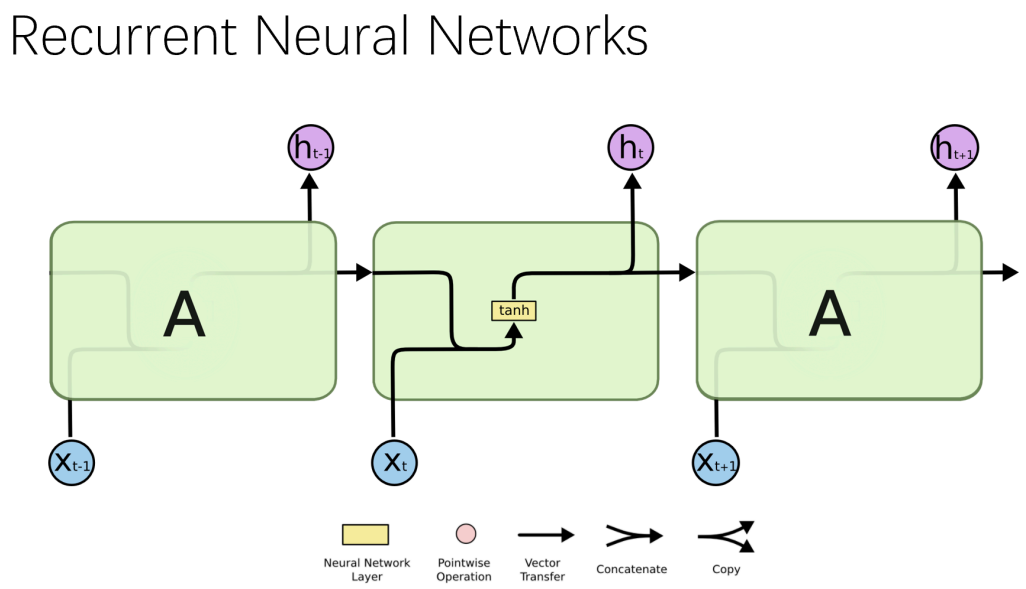
- **杰弗里·埃弗里斯特·辛顿**，FRS（英语：Geoffrey Everest Hinton，1947年12月6日—），又译**杰弗里·欣顿**[11]，英国出生的加拿大计算机学家和心理学家，多伦多大学教授。以其在类神经网络方面的贡献闻名。辛顿是反向传播算法和对比散度算法（Contrastive Divergence）的发明人之一，也是深度学习的积极推动者[12]，被誉为"深度学习教父"[13]。辛顿因在深度学习方面的贡献与约书亚·本希奥、杨立昆共同获得2018年的图灵奖[14]。2024年，辛顿与约翰·霍普菲尔德共同获得诺贝尔物理学奖。
- **约书亚·本希奥** OC FRS FRSC（法语：Yoshua Bengio，1964年3月5日[1]—）是一名加拿大计算机科学家，因其在人工神经网络和深度学习方面的研究而知名[2][3][4]。他是蒙特利尔大学计算机科学和运筹学系的教授以及蒙特利尔学习算法研究所科学主任。
- **杨立昆**（法语：Yann André LeCun，发音：[jan ɑ̃dʁe ləkœ]；1960年7月8日—），本名**扬·安德烈·勒坎**，是一名法国计算机科学家，2018年图灵奖得主，他在机器学习、计算机视觉、移动机器人和计算神经科学等领域都有很多贡献。他最著名的工作是在光学字符识别和计算机视觉上使用卷积神经网络，他也被称为卷积网络之父。[1][2]他同莱昂·博图和帕特里克·哈夫纳（Patrick Haffner）等人创建了DjVu图像压缩技术。他同莱昂·博图开发了Lush语言。

- word2vec（了解即可）
  - CBOW (Continuous Bag of Words, 连续词袋模型)
  - Skip-Gram

L9

- 递归神经网络需要理解

  递归神经网络（RNN）是一种专门用于处理序列数据（如时间序列、文本、语音等）的神经网络架构。它通过保存隐藏状态，使网络能够记住序列中的上下文信息，适用于需要考虑数据时间依赖或序列关系的任务。



L10

- **条件语言模型**（attention是核心的核心）
- **self-attention**
- trasformer架构，bert架构，gpt架构三者区别

四、三者的主要区别

特性	Transformer	BERT	GPT
提出年份	2017	2018	2018
核心架构	编码器 + 解码器	编码器	解码器
注意力方向	双向 + 单向	双向	单向
训练目标	无特定任务	MLM + NSP	自回归语言模型
输入特性	序列对或单序列	单序列或序列对	单序列
下游任务适配	通用	自然语言理解任务（分类等）	自然语言生成任务（生成等）
典型应用场景	机器翻译	情感分析、问答、分类	文本生成、对话、代码生成
训练复杂度	中等	高	较高

共同点：

- 都基于 Transformer 架构。
- 都使用注意力机制处理长序列的依赖问题。
- 都通过预训练学习语言表示，适用于多种下游任务。
- 都可以在大规模语料库上进行训练，具有迁移学习的优势。

改进方向：

- BERT 专注于文本理解（如情感分析）。
- GPT 专注于文本生成（如文章续写）。
- Transformer 作为基础架构，支撑了其他模型的设计。

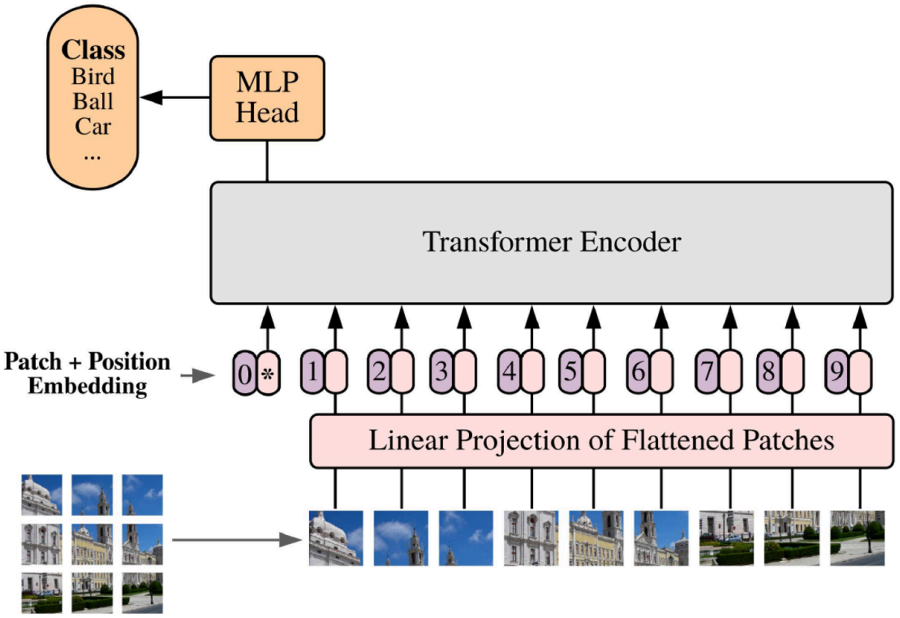
L11

- **cnn必掌握，会考细节**（怎么padding，怎么卷积）

  n*n的图像，用f*f的滤波器进行卷积，padding=p，步长设为s，输出大小 $\frac{n+2p-f}{s}+1 \times \frac{n+2p-f}{s}+1$

  如果商不是一个整数，向下取整，因为滤波必须完全处于图像中或者填充之后的图像区域内才输出相应结果

- vit了解即可



1. **将图像分割成小块（Patches）**：
   - 一张输入图像 x 被划分为固定大小的图像块（例如 16×16 像素）。
   - 每个图像块被展平成一个向量，类似于 NLP 中的单词（Token）。
2. **将图像块作为输入 Token**：
   - 每个图像块被映射到固定维度的向量空间中，形成一系列输入 Token。
   - 为了保留位置信息，ViT 使用**位置编码（Positional Encoding）**来标记每个块的空间位置。
3. **输入 Transformer 架构**：
   - Transformer 的多头自注意力机制可以捕捉输入 Token 之间的全局关系。
   - 与语言模型类似，这些 Token 的相互关系是通过自注意力机制建模的。
4. **分类任务**：
   - 在所有 Token 之前引入一个特殊的 **[CLS] Token**，通过 Transformer 提取该 Token 的表示并送入分类头进行预测。

L12

- 扩散模型是怎么回事

  扩散模型是一类基于概率的生成模型，利用**逐步添加噪声**和**逐步去噪**的过程来生成数据。这种方法最初源于物理中的扩散过程，后来被应用到深度学习中，用于生成高质量的数据（如图像、音频等）。扩散模型因其生成质量高、稳定性好，近年来成为生成式人工智能领域的重要技术之一

- 生成对抗网络大概是怎么回事

  生成式对抗网络（GAN）是一种由 **生成器 (Generator)** 和 **判别器 (Discriminator)** 组成的深度学习模型框架，用于生成高质量的、与真实数据分布相似的新数据。GAN 的基本思想是通过两个网络的对抗训练，使生成器不断改进，生成逼真数据。

  GAN 通过以下两部分组成：

  1. **生成器 (Generator)**：
     - 负责生成新的数据样本。
     - 输入随机噪声 zzz （通常是一个多维的随机向量），输出与真实数据类似的样本 G(z)G(z)G(z)。

    - 目标：欺骗判别器，让其认为生成的数据是真实的。
2. **判别器 (Discriminator)**：
    - 负责区分输入数据是真实数据还是生成数据。
    - 输入真实数据 xxx 或生成数据 G(z)G(z)G(z)，输出一个概率值 D(x)D(x)D(x) 表示是真实数据的可能性。
    - 目标：准确区分真实数据和生成数据。

L13听觉

- 傅里叶变换是在干啥
- 给出公式需要能够套进去，分贝是个啥

Intensity level

- Logarithmic scale
- Measured in decibels (dB)
- Ration between two intensity values
- Use an intensity of reference (TOH)

$$dB(I) = 10 \cdot log_{10}(\frac{I}{I_{TOH}})$$

1. $dB(I)$：
    - 表示声强级（Intensity Level），以分贝（decibel, dB）为单位。
    - 它量化了某个声音的强度与参考强度之间的相对大小。
2. I：
    - 当前声强（Intensity），指的是声音传播过程中单位面积上的功率（单位通常为 $\mathrm{W/m^2}$）。
3. I_{TOH}：
    - 参考声强（Threshold of Hearing，TOH），即人耳可以听到的最低声强。典型值为 $10^{-12}\,\mathrm{W/m^2}$。
4. $\log_{10}$：
    - 对数函数，用来表示两个声强比的数量级差异。以 10 为底的对数表示的是倍数。
5. **常数 10**：
    - 声强级的定义中引入的比例因子，用于将对数值转换为分贝单位。

- 什么是梅尔谱，为什么要用梅尔谱
- 噪声去除是怎么回事

L14

- 每个任务需要举出例子，对图片需要理解
- spell-attention模型需要了解
- feature dis那个看一下就行
- 合成模型需要了解deep voice, tactron, fastspeed，区别在哪?

  1. Deep Voice

  开发者：Baidu Research

  特点：从模块化到端到端的进化，支持多说话人和多语言。

  优点：高音质，版本 3 引入 Transformer 架构提升效率。

  缺点：生成速度较慢（早期版本）。

  适用场景：多说话人、多语言的智能语音助手。

  2. Tacotron

  开发者：Google AI
  特点：端到端语音合成，从文本直接生成频谱图。
  优点：语音自然度高，Tacotron 2 使用 WaveNet 提升音质。
  缺点：自回归架构导致生成速度较慢。
  适用场景：高质量语音生成，个性化 TTS。

  3. FastSpeech

  开发者：Microsoft Research
  特点：非自回归架构，支持并行生成频谱图，生成速度快。
  优点：实时性强，音质接近 Tacotron 2（FastSpeech 2）。
  缺点：音质在早期版本略逊于 Tacotron。
  适用场景：实时语音生成，大规模语音任务。

L15

- 了解case
- 了解最后的总结

选择，填空，简答，计算