# AI学习知识点

# 1. 机器学习与深度学习理论基础

## 监督学习

核心概念： 训练数据不仅包含输入特征（X），还包含与之对应的、人工标注好的正确答案或“标签”（Y）。模型的任务是学习从输入 X 到输出 Y 的映射函数 f(X) -> Y。

类比： 就像老师（数据提供者）给学生（模型）一本习题册，每一道题（输入特征）后面都附有标准答案（标签）。学生通过不断练习和对照答案（训练过程），学习解题的方法（模型参数），目标是能够独立解答新题目（预测）。

目标： 学习一个能够根据新输入数据准确预测其对应标签（未知的正确答案）的模型。

关键特征： **数据有标签**。

主要任务类型：

* + 分类： 预测离散的类别标签（如：垃圾邮件 vs 正常邮件、猫 vs 狗图片识别）。
  + 回归： 预测连续的数值（如：预测房价、预测股票价格、预测销售额）。

常见算法举例：

* + 分类：逻辑回归、支持向量机（SVM）、决策树、随机森林、K近邻（KNN）、朴素贝叶斯、神经网络。
  + 回归：线性回归、岭回归（Lasso/Ridge）、决策树回归、支持向量回归（SVR）、神经网络。

优点： 目标明确，评估模型性能相对容易（因为有真实的标签可以对比）。

缺点： 获取大量高质量的标注数据成本高昂、耗时长。

应用场景（图像分类、文本分类、回归预测）

## 非监督学习

* 核心概念： 训练数据只有输入特征（X），没有任何对应的标签（Y）。模型的任务是探索数据内在的结构、模式或关系，而不需要事先知道“正确答案”是什么。
* 类比： 就像给一个孩子一堆不同形状、颜色、大小的积木（无标签数据），不告诉他怎么玩。孩子自己观察、摸索（训练过程），可能会按颜色分组、按形状堆叠、或者发现一些有趣的排列（发现模式/结构）。
* 目标： 发现数据中隐藏的模式、结构或规律，如分组、降维、异常检测等。

关键特征： **数据无标签**。

主要任务类型：

* 聚类： 将数据点根据相似性自动分组成不同的“簇”。同一簇内数据点相似度高，不同簇之间相似度低（如：客户细分、文档主题分组）。
* 降维： 减少数据的特征数量，同时保留其最重要的信息（如数据压缩、可视化），以减少计算复杂度或去除冗余噪音（如：主成分分析 PCA、t-SNE）。
* 关联规则学习： 发现数据项之间有趣的关联或规律（如：购物篮分析，“买了尿布的人很可能也买啤酒”）。
* 密度估计： 估计数据在特征空间中的概率分布情况。
* 异常检测： 识别与大多数数据显著不同的异常点（如：信用卡欺诈检测、网络入侵检测）。

常见算法举例：

* 聚类：K-Means、层次聚类、DBSCAN、高斯混合模型。
* 降维：主成分分析（PCA）、t-分布随机邻域嵌入（t-SNE）、自编码器。
* 关联规则：Apriori。

优点： 不需要昂贵的标注数据；可以发现人类难以预见的、数据内在的复杂结构和模式。

缺点： 任务目标相对模糊；评估学习结果的好坏通常更为主观或需要特定指标（如轮廓系数评估聚类质量）；结果有时难以解释。

应用场景：用户分群、异常检测、数据降维

## 半监督学习

* 核心概念： 训练数据是混合的——包含一小部分有标签的数据和大量的无标签数据。模型试图同时利用这少量宝贵的标签信息和大量无标签数据的潜在结构信息来进行学习。
* 类比： 老师只给了一小部分习题册标注了答案（少量有标签数据），大部分习题没有答案（大量无标签数据）。学生需要利用那部分有答案的题目学习基本规则，同时观察那些无答案题目的特征和彼此之间的关系（探索无标签数据的结构），来尝试解答剩下的题目（预测无标签数据的标签或提升整体模型性能）。
* 目标： 利用少量标签提供指导，结合大量的无标签数据中蕴含的结构信息，来学习一个比仅使用少量标签数据更好的模型（通常能达到接近使用大量标签数据监督学习的效果），或对无标签数据进行预测。

关键特征： **数据混合（少量标签 + 大量无标签）**。

动机： 在现实世界中，获取大量数据相对容易（无标签），但高质量的人工标注（标签）成本高、耗时长。半监督学习提供了一种折中方案，利用丰富但廉价的未标注数据来弥补标签的不足。

核心思想：

* 一致性假设/平滑性假设： 相似的数据点（在高维空间中靠近的点）很可能具有相同的标签。
* 流形假设： 高维数据实际上分布在一个更低维度的流形结构上。利用无标签数据可以帮助学习这个流形结构，从而更好地推断标签。

常见方法：

* 自训练： 先用有标签数据训练一个初始模型；用这个模型预测无标签数据的标签（伪标签）；将置信度高的预测作为新标签加入训练集；用扩增后的训练集重新训练模型；迭代进行。
* 协同训练： 假设数据可以从两个不同视角（特征集）描述。训练两个不同的模型，每个模型基于自己的视角给无标签数据打伪标签；互相把对方高置信度的伪标签数据加入自己的训练集；迭代训练。
* 生成模型： 如生成对抗网络（GAN）或变分自编码器（VAE）的半监督变体，同时建模数据的分布和有标签信息。
* 基于图的方法： 将数据点视为图中的节点，相似度高的点连接边。标签信息在图上传播。

应用场景： 非常适合标注成本高昂的领域（如医学图像分析、自然语言处理中的语义标注、语音识别）。

优点： 能显著降低对标注数据的依赖，利用廉价的无标签数据提升模型性能。

缺点： 算法通常更复杂；性能提升依赖于无标签数据的质量和相关性与标签任务的一致性；如果假设不成立（如相似点不一定同标签），效果可能变差。

| **特性** | **监督学习** | **无监督学习** | **半监督学习** |
| --- | --- | --- | --- |
| **训练数据** | 输入特征 **+** 对应标签 (X, Y) | **只有** 输入特征 (X) | \*\*混合：\*\*少量(X, Y) + 大量(X) |
| **是否有标签** | **有** | **无** | **部分有** |
| **核心目标** | 学习 X -> Y 的映射，预测新数据的标签 | 发现数据内在结构、模式、关系 | 结合少量标签和大量无标签数据学习更好模型或预测无标签数据 |
| **主要任务** | 分类、回归 | 聚类、降维、关联规则、密度估计、异常检测 | 分类、回归（侧重利用无标签数据提升） |
| **类比** | 老师给答案的习题册 | 探索一堆未知物品 | 老师只给少量习题答案，其余靠自己琢磨和联系 |
| **优点** | 目标明确，评估容易 | 无需标注，发现未知模式 | 降低标注成本，利用大量未标注数据 |
| **缺点** | 标注数据成本高、耗时长 | 目标模糊，评估主观，结果难解释 | 算法复杂，依赖数据假设，效果不稳定 |
| **典型算法** | 线性回归、逻辑回归、SVM、决策树、神经网络 | K-Means、PCA、Apriori | 自训练、协同训练、半监督生成模型、图半监督 |

### 神经网络与深度学习基础

* 神经元与激活函数：输入、权重、偏置、激活函数（ReLU、Sigmoid、Tanh）
* 前向传播与反向传播：计算输出、通过梯度下降优化权重
* 损失函数与优化器：均方误差（MSE）、交叉熵（Cross-Entropy）、SGD、Adam、RMSprop
* 经典网络结构：全连接网络（FCN）、卷积神经网络（CNN）、循环神经网络（RNN）
* 深度学习框架：PyTorch、TensorFlow、JAX

## 2. 大模型发展全景与关键技术

### transformer架构的诞生与应用

* 注意力模型（Attention Model）：捕捉长距离依赖
* 自注意力机制（Self-Attention）：语义理解的钥匙
* 多头注意力（Multi-Head Attention）：并行计算不同子空间的特征
* 位置编码（Positional Encoding）：引入序列位置信息
* 大模型技术突破与前沿趋势（通过deepseek获取）

### 大模型的定义与特点

* 传统模型与大模型的差异
* 稀疏注意力（Sparse Attention）：降低计算复杂度（Longformer、BigBird）
* 混合专家模型（MoE）：动态路由机制（GShard、Switch Transformer）
* 模型压缩与加速：量化（INT8、FP16）、知识蒸馏（小模型学习大模型的知识）

## 4.大模型应用开发平台

### Dify 开发平台简介

平台定位：面向零基础人员的 Agent 开发平台

基础概念：后端即服务（Backend as Service）、LLMOps、工作流、提示词编排等

核心功能与技术架构

核心功能：内置工具接入、变量管理、条件分支、迭代与模板转换

技术架构：前后端分离、API 接口、数据存储（如 MySQL、向量数据库）

Dify 平台使用方法

快速实验：注册开箱即用的 Dify 在线服务

环境准备：Docker 与 Docker Compose 安装、环境变量配置与服务启动

实际操作：新建应用、节点配置、提示词设置、调试预览

常见问题与解决办法说明

使用 Dify 快速搭建 Agent 应用原型

实战：使用 Dify 快速搭建 AI 图像生成应用

实战：使用 Dify 快速搭建智能客服机器人

### LangBot 大模型原生即时通信机器人平台

LangBot 平台简介

平台定位：高稳定性、多模态支持的即时通信机器人开发框架

核心优势：多平台适配（微信/飞书/钉钉等）、插件扩展性、Anthropic MCP 协议兼容

基础概念：多轮对话引擎、工具调用机制、敏感词过滤与访问控制

核心功能与技术特性

多模态对话能力

支持的大模型列表：DeepSeek/Claude

群聊与私聊场景适配、多模态输入输出（文本/图像/文件）

高稳定性设计

限速机制、熔断策略、错误自动恢复

动态配置热更新（通过 JSON/YAML）

快速部署方案

Docker 一键部署 LangBot

Web 管理面板：浏览器管理 LangBot 实例，提供日志、插件管理、访问控制等功能

实战： LangBot 接入个人微信

Docker 网络准备

部署 Gewechat 服务

配置 LangBot 适配器

启动 LangBot 并扫码登录微信

验证私聊和群聊通信功能

### 使用 Dify 和 LangBot 研发多平台智能客服 Agent

多平台智能客服方案概述

介绍 Dify + LangBot 组合的优势：快速接入、统一管理、多模态支持

典型应用场景：智能客服、群助手、工作流自动化

核心组件部署

LangBot 部署：Docker部署（推荐）/手动部署/宝塔面板部署

Dify 服务配置：配置API密钥和超时参数

双向通信原理：LangBot 消息路由机制与 Dify Webhook 对接流程

IM 多平台接入实战

钉钉机器人：开发者后台创建应用→配置Stream模式→获取RobotCode→群聊绑定

微信公众号：

两种模式对比：传统15秒限制模式 vs 异步消息获取模式

回调地址设置：Token/EncodingAESKey配置+HTTPS反向代理

高级功能与优化

多应用类型适配：

Chat应用：基础对话场景配置

Agent应用：工具调用状态跟踪（需开启`track-function-calls`）

Workflow应用：输出键`output-key`动态映射

性能与稳定性：

超时处理：微信公众号15秒限制的应对方案

错误恢复：消息重试机制与日志监控

实战案例：基于行业真实需求的 Agent 开发

电商场景：企业微信-智能客服（RAG）

教育场景：飞书-高考志愿填报智能客服

### 使用 Dify 研发代码自动生成 Agent

需求与应用价值

市场背景：自动化代码生成提升开发效率、降低成本

应用场景：企业工具开发、自动化测试、快速原型、跨平台应用构建

技术方案与系统架构设计

整体方案：用户需求输入 → dify 工作流调度 → 调用 DeepSeek 模型生成代码 → 代码执行与反馈

关键模块：需求解析、代码生成（支持函数调用/ ReAct 推理）、代码校验、缓存与日志监控

核心功能实现

高效提示词设计：指导大模型生成符合需求的代码

模型调用与错误重试：异步执行、自动重试机制与缓存策略

企业级扩展：模块化设计、工具链整合、API 鉴权、安全与本地/云端部署

重要组件：

模型：DeepSeek-Coder 或 CodeLlama

框架：Dify 工作流编排

工具：GitHub API、Linter（Black/ESLint）

实战：企业级代码自动生成 Agent 开发

案例演示：贪吃蛇小游戏代码的构建与调试

案例演示：公司官方网站代码的构建与调试

## 5.模型上下文协议（MCP）简介与开发实战

MCP 概述

MCP 的愿景：提供标准化协议连接 LLM 与数据/工具

MCP 的作用：构建 Agent 和复杂工作流，实现数据与工具整合

MCP 的优势：预构建集成、LLM 供应商灵活切换、安全的数据访问

MCP 架构、组件与技术实现

MCP 技术架构与关键组件

MCP 三驾马车：Prompts, Resources, Tools

使用 Cherry Studio 快速体验 MCP

Cherry Studio 介绍与安装部署

Cherry Studio 模型服务与 MCP 服务配置

基于 12306 MCP Server 实现旅行规划助手

MCP 开源生态：站在巨人的肩膀上

MCP Server 开发与部署

MCP 通信技术架构与原理

MCP Server 开发环境搭建与项目初始化

MCP 天气查询服务器开发（基于 NWS API）

MCP 天气查询应用实战与调试

MCP Host 选择：VS Code + CLINE 插件

MCP Server 本地部署与 Client 配置方法

基于 CLINE 开发与调试行程规划助手

魔塔社区 MCP Server 配置与调试

使用中国天气查询与 12306 API 实现行程规划

深入浅出 MCP 底层技术原理

MCP 协议层与传输层解读

MCP 调用栈与生命周期解析

### A2A 协议与 MCP 的协同之道

AI 协作的黎明：A2A 协议系统性介绍(“Why” & “What”)

AI Agent “能力孤岛”与 A2A 的破局之道

A2A 核心使命：打破壁垒、赋能协作、保护隐私

A2A 设计哲学：仰望星空，脚踏实地

深入 A2A 的运作肌理 (“How”)

A2A 核心架构与 Agent Card 介绍

A2A 交互流程、四要素和两种模式详解

A2A 落地场景：自动化企业招聘和采购，超级智能私人助理

伟大的协同：当 MCP 遇见 A2A (“Synergy”)

Agent 世界观类比：个体（MCP）与社会（A2A）

MCP 与 A2A 全面对比：核心差异，技术栈分层，场景选择，终局展望

MCP 与 A2A 协同使用场景与未来展望

实战 A2A 官方 Python 示例 HelloWorld

## 6.企业知识库与问答系统设计

企业数据治理与知识库搭建现

知识库需求分析与功能规划

企业数据采集、清洗与知识提取

企业知识库的设计原则与架构

向量数据库技术选型：开源 vs 商业服务

如何基于大模型设计问答系统

问答系统的基本架构与技术

如何集成企业知识库与AI大模型

优化问答系统的性能与准确性

企业级应用的挑战与解决方案

在企业环境中实施AI大模型的难点

可扩展性与容错机制设计

高效管理与调度模型资源

### 企业级 RAG 开发框架 FastGPT 快速入门

FastGPT 概述与核心价值

FastGPT 的定位：企业级 RAG 开发框架

核心功能：知识库问答、工作流编排、自动化数据处理

适用场景：智能客服、内部知识管理、复杂业务流程自动化

FastGPT 部署与配置

环境准备：Docker 与 Docker Compose 安装

一键部署：FastGPT 的 Docker Compose 配置文件解析

关键配置：

模型服务连接：DeepSeek

向量数据库设置（Milvus / Chroma）

数据存储配置（MySQL 表结构与优化）

快速使用 FastGPT

知识库构建：

文档上传与自动分段

问答对生成与向量化

简单问答测试：

通过 Web 界面测试知识库问答效果

查看检索结果与生成答案

### 使用 FastGPT 开发企业级智能问答 Agent

FastGPT 高级功能实战

工作流编排：

案例：智能客服流程设计

节点1：用户意图分类（正则匹配或模型判断）

节点2：知识库召回 + 外部 API 调用（如订单查询）

节点3：生成结果过滤与格式化输出

API 集成与鉴权：

鉴权方式：JWT Token 或 API Key

请求示例：Python/Node.js 调用 FastGPT 服务

企业级智能问答 Agent 的设计思路

需求分析：高精度问答、业务流程集成、数据安全

构建高效的问答引擎：信息检索与生成模型结合

技术架构：DeepSeek + 知识库 + 工作流 + 外部 API 集成

数据安全：如何将问答系统部署在本地环境

性能优化与监控

检索性能优化：向量库索引调优（HNSW vs IVF）

生成质量优化：Prompt 模板设计与模型温度调节

监控指标：用户反馈统计、响应时间（P99 指标）

实战：使用 FastGPT 开发企业级智能问答 Agent

## 16. 企业应用中的权限分级控制与数据隐私保护

权限分级设计与实施

角色定义：管理员、开发者、普通用户

权限分配：模型访问、数据读写、API调用

实施工具：RBAC（基于角色的访问控制）

训练中的数据隐私保护策略

差分隐私：训练中加入噪声

联邦学习：分布式训练，数据不出本地

数据脱敏：敏感信息替换或加密

数据加密与隔离策略

传输加密：TLS/SSL协议

存储加密：AES-256算法

数据隔离：多租户架构

数据合规性保障

GDPR合规：数据主体权利、泄露通知

中国法规：网络安全法、数据安全法、个人信息保护法

审计与监控：日志记录、异常检测、定期合规检查

## 17. 跨行业经验复制与应用

如何将行业解决方案移植到其他行业

行业间的差异与相似性分析

将智能客服系统移植到金融与零售行业

复制成功案例并根据行业特点调整

智能客服在不同行业的适配与优化

金融行业的智能客服需求

医疗行业中的智能客服应用

教育行业智能客服的特殊要求

案例：从金融到零售的智能客服实践

金融行业智能客服案例分析

零售行业的智能客服应用实践

行业经验转化与优化

## 18. 跨行业代码生成 Agent 经验复制与应用

如何将代码生成解决方案移植到其他行业

分析不同行业的开发需求与技术共性

探讨如何将企业内部代码自动生成方案扩展到互联网应用、移动端开发等场景

复制成功案例，并根据行业特点进行定制化调整

代码生成 Agent 在不同行业的适配与优化

金融行业：自动生成风控、数据分析与报表代码的需求特点

医疗行业：数据处理、电子病历、统计报表自动生成的实际应用

教育行业：在线课堂、教学工具、作业自动批改系统的代码生成需求

案例：从企业工具到跨平台应用的代码生成实践

企业内部自动化代码生成案例分析

跨平台移动应用自动代码生成实践

行业经验转化、方案优化与效益提升

## 大模型应用的趋势与挑战

未来大模型发展方向预测

多模态模型的崛起：文本、图像、语音的融合。

绿色AI：降低能耗，提升计算效率。

通用人工智能（AGI）：多任务学习、跨模态理解

模型安全与伦理：对抗攻击防御、生成内容审核

面向未来的大模型技术创新

下一代大模型的技术创新方向

量子计算与大模型的结合

多模态AI的进一步发展

行业内外部环境变化对大模型的影响

经济与技术环境对AI发展的影响

全球市场的竞争与技术壁垒

大模型与伦理法规的挑战

未来AI发展的职业机会与挑战

AI人才需求的变化与趋势

企业如何培养AI大模型相关的技术人才

从开发到管理的职业发展路径

## DeepSeek-R1 模型微调与案例分析

微调的基本概念与训练流程

微调的定义与技术要点

数据准备与预处理

微调模型的训练与评估

基于DeepSeek的模型微调案例

实战案例：如何使用DeepSeek进行微调

微调技术在企业中的应用

处理行业特定问题的微调方法

如何定制化训练大模型以满足行业需求

行业应用定制化需求分析

针对行业需求定制化数据集

企业级大模型的个性化训练方案

## 实战 DeepSeek-R1-Qwen 小模型蒸馏

蒸馏技术基础与背景

小模型蒸馏的定义与优势：降低推理成本、提高部署效率

DeepSeek-R1 与 Qwen 模型特点对比：大模型性能与小模型部署的平衡

企业级应用：如何将蒸馏小模型部署到边缘设备与在线服务中

蒸馏流程与关键技术

数据集构建与标注：如何准备高质量蒸馏训练数据

蒸馏策略设计：温度调节、损失函数与知识迁移方法

模型训练与优化：微调蒸馏策略、评估小模型效果

实战：使用 DeepSeek-R1-Qwen 进行小模型蒸馏

环境搭建：配置 GPU 环境与分布式训练工具

训练过程：参数设置、模型收敛监控与实时评估

效果展示：小模型在代码生成、问答等场景下的性能表现

# 模块一大语言模型技术栈与 Prompt 工程

大语言模型调用方式与函数调用（Function Calling）

OpenAI Function Calling 原理与实战：

JSON Schema 定义工具参数结构

Tool Call + Response Chain 流程

示例场景：天气查询、数据库查询、API 调用

HuggingFace Transformers / TGI / vLLM / Ollama 等本地/私有化部署调用方式对比

LangChain 核心组件详解

LangChain 核心组件

LLMChain：Prompt → LLM → Output 的基本流程封装

Agents：

ZeroShotReactDescriptionAgent（ReAct 框架）

Plan-and-Execute Agent（计划+执行模式）

Tools：集成自定义工具（如搜索、数据库访问、计算）

Memory：短期记忆（ConversationBufferMemory）、长期记忆（Redis/MongoDB 存储）

LlamaIndex 全流程实践

文档加载器（Document Loaders）：PDF、网页、SQL、Notion 等格式支持

索引构建（Indexing）：向量化存储、元数据管理

查询引擎（Query Engine）：检索 + 生成一体化

自定义索引结构：扩展 Index 类型以适应特定业务需求

Prompt Engineering 高阶技巧

思维链（Chain of Thought, CoT）：引导模型逐步推理

反思机制（Self-Reflection）：让模型自己评估输出质量

提示模板设计（Prompt Templates）：Jinja2、LangChain Template 支持

外部工具调用提示设计（Tool Calling Prompt）

动态 Prompt 生成：根据用户输入动态构造 Prompt

AutoGen 多 Agent 协作框架

Agent 角色定义（UserProxyAgent、AssistantAgent）

Group Chat 模式：多 Agent 轮流发言、达成共识

Debate 机制：Agent 间进行辩论以提升决策质量

通信机制：Message Passing、工具调用、反馈循环

微调方法比较（LoRA、P-Tuning v2、Adapter）

LoRA（Low-Rank Adaptation）原理与优势

P-Tuning v2：基于可学习 prompt 的轻量微调

Adapter：插入小型神经网络模块进行增量训练

适用场景分析：何时选择哪种微调方式？

Agent 构建与多轮对话逻辑设计

如何设计一个 Agent 的状态机？

多轮对话中如何保持上下文？

如何设计 Agent 之间的消息传递协议？

示例：用户提问 → Agent 分解任务 → 多个子Agent协作 → 汇总结果返回

实践练习：

实践一：基于 LangChain 构建一个多任务问答助手

输入输出定义：

输入：用户自然语言问题（如："帮我查一下今天的天气"、"最近的新闻有哪些？"）

输出：准确答案 + 使用了哪些工具（如天气 API、新闻 API）

扩展项：支持历史对话上下文、缓存结果、错误处理

关键挑战：

如何判断是否需要调用工具？

如何防止无限递归或死循环？

如何记录对话状态？

实践二：构建一个多 Agent 协同客服系统

场景设定：

用户提问："我的订单为什么还没发货？"

系统拆解任务：

Agent A：查询订单状态（调用内部系统）

Agent B：检查物流信息（调用第三方 API）

Agent C：汇总结果并生成回复

技术难点：

如何设计 Agent 之间的通信协议？

如何保证失败重试机制？

如何可视化 Agent 之间的交互过程？

# 模块二深度学习与 NLP 基础

NLP 核心概念回顾

Tokenization：

字符级、词级、子词级（BPE、WordPiece）

分词工具（NLTK、spaCy、HuggingFace Tokenizers）

Embedding：

Word2Vec、GloVe、FastText

Transformer 中的 Positional Encoding

Attention 机制：

Softmax 注意力、Self-Attention、Cross-Attention

多头注意力（Multi-head Attention）详解

Transformer 架构详解

Encoder-Decoder 结构

Self-Attention 的数学推导与代码实现

Positional Encoding 的作用与实现方式

Feed Forward 层结构

Layer Normalization 与残差连接

编码器 vs 解码器的区别

Transformer 变体与扩展

BERT：

MLM（Masked Language Modeling）

NSP（Next Sentence Prediction）

GPT 系列（GPT-2/3/3.5/4）：

自回归生成模型

上下文学习（In-context Learning）

MoE（Mixture of Experts）：

如何在大规模模型中做专家路由？

多模态模型（CLIP、BLIP、Flamingo）：

图像 + 文本联合建模

跨模态检索与生成

使用 PyTorch / TensorFlow 实现文本分类

数据预处理：

加载数据集（IMDB、AG News、SST-2）

Tokenizer 使用与 padding/truncation

模型构建：

LSTM、CNN、Transformer-based 模型对比

训练流程：

Loss 函数选择（CrossEntropyLoss）

Optimizer 配置（AdamW）

学习率调度（LinearWithWarmup）

评估指标：

Accuracy、F1-score、AUC-ROC

HuggingFace Transformers 库实战

安装与基本用法

预训练模型加载（AutoModelForSequenceClassification）

Tokenizer 使用（from\_pretrained）

Trainer API 使用：

自定义 Dataset 类

TrainingArguments 配置

EvaluationCallback 回调函数

LoRA 微调实战（使用 PEFT 库）

LoRA 原理简述：

在权重矩阵中引入低秩矩阵进行参数调整

显存节省、推理效率高

使用 PEFT（Parameter Efficient Fine-Tuning）库进行微调

微调后模型保存与加载

微调效果对比（LoRA vs 全量微调）

模型评估与 A/B 测试

BLEU、ROUGE、METEOR、Perplexity 指标详解

如何设计 A/B 测试实验？

因果推断在 A/B 测试中的应用

统计显著性检验（t-test、bootstrap）

模型迭代策略（持续评估 + 回滚机制）

模型压缩与优化技术

知识蒸馏（Knowledge Distillation）：

小模型模仿大模型输出分布

量化（Quantization）：

INT8、FP16、混合精度训练

ONNX Runtime 部署：

将模型转换为 ONNX 格式

使用 ONNX Runtime 加速推理

模型剪枝（Pruning）：

移除冗余参数以减小模型体积

超参数自动调优（Optuna / Hyperopt）

参数搜索空间定义

使用 Optuna 进行网格搜索 / 贝叶斯优化

并行训练多个模型版本

最佳超参数组合选取

实践练习：

实践一：训练一个意图识别模型并部署为 API

输入输出定义：

输入：用户自然语言句子（如"我要退票"、"查询余额"）

输出：对应的意图标签（如"ticket\_refund", "account\_balance"）

扩展项：

支持热更新模型

提供 Swagger UI 接口文档

添加日志记录与异常处理

实践二：基于 LoRA 微调一个垂直领域客服问答模型并部署为 API

场景设定：

使用医疗或法律领域的 QA 数据集（如 MedQA、LegalQA）

微调一个 BERT 或 LLaMA 模型，提升特定领域问答准确率

技术难点：

如何准备领域语料？

如何评估模型在领域内的表现？

如何将 LoRA 权重合并进原始模型？

# 模块三数据工程与知识增强

数据清洗与预处理

原始数据来源：

PDF、Word、HTML、数据库、API

清洗方法：

正则表达式提取关键字段

使用 Pandas 进行缺失值填充、去重、格式标准化

文本规范化：

分词、去除停用词、大小写统一、拼写纠错

编码转换与乱码处理

多源异构知识库构建

多模态数据整合：

结构化数据（数据库表）

半结构化数据（JSON/XML）

非结构化数据（文本、图像描述）

数据归一化与统一接口设计

数据版本控制（DVC / Git LFS）

文档切片策略优化

固定长度切片 vs 语义切片：

固定长度：容易截断语义

语义切片：基于句子/段落边界、标题识别

重叠窗口机制：

解决信息被割裂问题

使用 LangChain / LlamaIndex 内置分块器（RecursiveCharacterTextSplitter）

自定义分块逻辑（基于标点、换行符、标题等级）

向量数据库原理与实战

向量表示基础：

使用 BERT、Sentence-BERT、SBERT-wk 等模型编码文本

相似性搜索算法：

FAISS 中的 IndexFlatL2、IVF-PQ、HNSW

Milvus 架构介绍：

Standalone vs Cluster 模式

存储引擎（MinIO）、注册中心（Etcd）

向量数据库集群部署

Milvus Standalone 部署：

Docker Compose 安装

Milvus Cluster 模式部署：

Etcd + MinIO + Milvus standalone 组合

FAISS + Redis 缓存加速：

利用 Redis 缓存高频查询结果

设计缓存失效机制（TTL、更新通知）

RAG 系统构建

RAG 流程详解：

Retrieve → Generate

使用 LangChain / LlamaIndex 构建 RAG Pipeline

支持多源检索（混合 BM25 + 向量检索）

评估指标：

Recall@K, MRR, BLEU, ROUGE-L

知识图谱基础与 Neo4j 实践

图数据库基础概念：

节点、关系、属性

Neo4j 安装与 Cypher 查询语言

构建 FAQ 图谱：

问题节点 ←→ 答案节点

问题之间相似性关系

图谱可视化工具（APOC、Gephi）

图神经网络在知识图谱中的应用

PyTorch-Geometric + Neo4j 联合使用

GNN 模型用于关系预测、实体分类

多跳问答示例：

用户提问："A 公司的子公司有哪些？"

图谱中路径：A 公司 → 控股 → B公司 → 控股 → C公司

RAG + KG 联合推理

如何将 RAG 与图谱结合？

先 RAG 检索候选答案，再图谱验证

或者先图谱查找相关实体，再 RAG 生成回答

多跳问答系统构建：

第一次检索 → 得到中间实体 → 第二次检索 → 最终答案

数据增强策略

Back Translation（回译）：

将中文翻译成英文，再翻译回来生成变体

EDA（Easy Data Augmentation）：

同义词替换、随机插入、删除、交换

利用大模型生成伪样本（Self-training）

实践练习：

实践一：构建一个基于 FAISS/Milvus 的 FAQ 检索系统

输入输出定义：

输入：用户自然语言问题（如"如何退货？"）

输出：最相关的 FAQ 条目及其答案

扩展项：

支持多轮对话上下文过滤

支持热更新知识库

提供 RESTful API 接口

实践二：构建一个融合文档检索、图谱推理与 LLM 生成的多跳问答系统

场景设定：

用户问："A 公司的最大股东是谁？"

系统流程：

检索 A 公司相关信息

图谱中查找控股关系

生成最终回答

技术难点：

如何将 RAG 与图谱推理融合？

如何设计联合评分机制？

如何防止错误传播？

# 模块四智能客服系统架构设计

智能客服系统整体架构解析

整体分层架构：

前端接入层（Web / App / 微信 / 第三方接口）

对话引擎层（意图识别、对话状态管理、Agent 执行）

后台服务层（数据库、API 网关、消息队列）

关键子系统：

NLU（自然语言理解）

DM（对话管理）

NLG（自然语言生成）

DST（对话状态追踪）

意图识别与槽位填充

Rule-based 方法：

正则匹配、关键词库、有限状态机（FSM）

ML-based 方法：

使用 BERT、CRF、BiLSTM 等模型进行分类与序列标注

Slot Filling 流程：

提取用户输入中的实体信息（如时间、地点、订单号）

意图识别流水线（Pipeline Design）

多阶段融合策略：

规则引擎兜底

机器学习模型主控

Ensemble 投票机制

模型热更新机制：

加载新模型权重而不重启服务

意图识别服务 API 设计（RESTful / gRPC）

上下文理解与状态追踪（Dialogue State Tracking）

状态表示方式：

FSM（有限状态机）

基于 Rasa 的 Tracker

使用 RL 或 LSTM 模型进行状态建模

Session Context 管理：

存储历史对话状态（Redis / MongoDB）

实现跨轮次的上下文共享

多轮对话设计与管理

Slot Filling 机制：

动态缺失槽位提示

回退机制（Fallback）设计

对话流程控制：

条件分支、循环逻辑、跳转指令

错误处理：

非法输入检测

多轮超时机制

工具调用引擎（Tool Calling Pipeline）

工具注册与发现机制：

插件化加载（Python importlib）

REST/gRPC 工具封装

工具调度流程：

根据意图和槽位选择对应工具

执行并返回结果

异常处理机制：

工具失败重试

超时中断

结果缓存机制

# 模块五多 Agent 协作与通信机制

自定义 Agent 设计

Agent 组件设计：

决策引擎（Rule-based / LLM）

工具调用管理器

记忆存储（短期+长期记忆）

Agent 行为建模：

基于 LangChain 的 Agent Loop

基于状态机的 Agent 流程控制

示例 Agent 类型：

客服 Agent

订单查询 Agent

售后处理 Agent

插件化架构设计（支持动态加载新功能）

插件结构设计：

config.yaml 定义插件元信息

plugin.py 定义插件逻辑

动态加载机制：

使用 Python importlib 或 pkg\_resources

插件热更新机制：

不重启服务更新插件代码

插件生命周期管理：

初始化、运行、销毁钩子函数

多模态输入处理

图像输入处理：

OCR 提取文字内容

CLIP 模型识别图像语义

语音输入处理：

ASR（自动语音识别）转文本

使用 Whisper、DeepSpeech 等开源模型

多模态融合策略：

文本 + 图像联合理解

使用多模态模型（如 BLIP、Flamingo）

实践练习：

实践一：设计一个支持多轮对话的订单查询客服流程

输入输出定义：

输入：用户自然语言（如“我要查订单”、“订单号是 1234567890”）

输出：订单状态、物流信息、退款建议等

扩展项：

支持语音输入（ASR）

支持图像上传（OCR 提取订单号）

支持对话记录持久化（MongoDB）

实践二：设计一个可插拔的意图识别与对话管理模块，支持热更新

场景设定：

新增一个“发票开具”意图

系统无需重启即可加载新意图模型和对话流程

技术难点：

如何实现模型热加载？

如何保证热加载过程中的稳定性？

如何测试新增意图的兼容性？

多 Agent 系统基础概念

Agent 定义：

具备自主性、反应性、目标导向性的软件实体

合作 vs 竞争：

协同完成任务 vs 在资源有限场景下博弈

通信机制：

同步 vs 异步、本地内存 vs 网络通信

Agent 状态管理：

生命周期、状态迁移、上下文共享

主流 Agent 协作框架对比

AutoGen（微软开源）

支持 Group Chat、Debate、自定义角色

优势：支持复杂对话流程、可扩展性强

CrewAI（社区活跃）

提供 Task + Agent + Tools 分层结构

支持串行/并行执行

LangGraph（LangChain 新推出的 Graph-based Agent 框架）

基于状态机和图结构编排 Agent 流程

可视化流程控制

MCP 与 A2A 协议详解

MCP（Multi-Agent Communication Protocol）

标准化 Agent 之间的通信格式（JSON Schema）

包含字段：sender、receiver、content、tool\_call、status

A2A（Agent-to-Agent）协议

规定 Agent 如何在不同平台上进行互操作

支持跨语言调用

构建基于 MCP 的 Client-Server 架构

服务端设计：

接收 Agent 请求（gRPC / REST）

解析 MCP 消息

执行逻辑并返回响应

客户端设计：

封装 MCP 消息发送器

支持异步回调、超时设置

多 Agent 任务调度策略

并行调度：

多个子任务同时执行（如数据采集、分析、生成）

依赖图调度：

任务之间有前后依赖关系（如先检索再总结）

优先级调度：

设置紧急任务高优先级，普通任务低优先级

资源感知调度：

根据 Agent 的负载情况动态分配任务

使用 Ray 实现分布式 Agent 调度

Ray 基础知识：

Actor 模型、远程函数、任务队列

分布式部署：

多节点集群部署

自动任务分发与负载均衡

多 Agent 系统的异常处理与恢复机制

错误类型：

工具调用失败、网络中断、Agent 崩溃

恢复策略：

重试机制（指数退避）

熔断机制（Hystrix）

降级机制（兜底回复）

日志记录与追踪：

记录每个 Agent 的行为日志

支持回放与调试

实践练习：

实践一：开发一个基于 MCP 协议的多 Agent 协作系统

输入输出定义：

输入：用户问题（如“帮我写一篇关于 AI Agent 的文章”）

输出：多个 Agent 协作完成研究、撰写、审核、润色等任务

扩展项：

支持失败重试机制

支持 gRPC 通信

支持任务队列持久化（Redis）

实践二：支持 gRPC 通信 + 状态同步 + 任务队列持久化（Redis）

场景设定：

用户提交任务后，系统将任务拆解为多个子任务

每个 Agent 从 Redis 获取任务并执行

执行结果通过 gRPC 回传并更新状态

技术难点：

如何保证任务不丢失？

如何避免重复执行？

如何实现跨服务的状态同步？

# 模块六DSL 语言设计与执行引擎

DSL 设计原则与应用场景

DSL 的定义：

针对特定领域的轻量级语言，用于描述业务逻辑

优势对比传统硬编码：

更易理解（业务人员可参与）

更易修改（无需重新编译/部署）

更易测试（可视化流程 + 单元测试）

应用场景：

客服流程编排（意图识别 → 分支判断 → 工具调用）

风控审批流程（输入 → 条件判断 → 决策输出）

多 Agent 协作调度（任务分配 → 并行执行 → 汇总结果）

使用 ANTLR / Lark 解析 DSL 语法

ANTLR（推荐用于复杂语法规则）

定义 .g4 语法文件

自动生成词法分析器与语法分析器

支持多种语言（Python、Java、C++）

Lark（适合轻量级 DSL）

简洁的 EBNF 语法

Python 原生支持

更适合快速原型开发

构建 DSL 解释器与执行引擎

抽象语法树（AST）遍历：

使用 Visitor 模式或 Transformer 模式

执行引擎核心组件：

流程控制器（状态跳转）

表达式求值器（条件判断）

工具调用器（绑定函数）

上下文管理器（保存变量状态）

将 DSL 集成到 Agent 框架中（LangChain / CrewAI）

在 LangChain 中集成：

自定义 Agent 类

注册 Tool 到 ToolManager

使用 PromptTemplate 渲染 DSL 模板

在 CrewAI 中集成：

定义 agent + task + tool 的 YAML 描述

支持流程编排与依赖关系

在 AutoGen 中集成：

使用 UserProxyAgent 发起 DSL 流程

AssistantAgent 根据 DSL 执行步骤

支持 DSL 动态生成与运行时参数注入

动态生成 DSL

参数注入机制

热更新机制

版本控制

实践练习：

实践一：设计并实现一套面向 Agent 任务规划的 DSL 语言

输入输出定义：

输入：DSL 脚本文件（YAML 或 JSON）

输出：执行流程日志、最终结果

示例 DSL 功能要求：

包含任务节点（start、step1、step2）

支持条件判断（if...then...else）

支持工具调用（call\_tool）

支持跳转逻辑（goto next\_state）

实践二：开发对应的解析器与执行引擎，支持动态加载与运行

场景设定：

用户上传一个 DSL 文件，系统自动解析并执行流程

支持运行时注入参数（如订单号、用户 ID）

支持热更新 DSL 文件而不重启服务

技术难点：

如何实现高效的 DSL 加载与缓存？

如何防止非法 DSL 引发安全风险？

如何记录执行日志供后续调试？

# 模块七智能 Agent 高级能力构建

记忆管理系统设计

LangMem / Zep / MemGPT 的使用与扩展

短期记忆：

基于 Session Context 存储最近对话历史

长期记忆：

使用向量数据库（FAISS/Milvus）存储关键事实

Agent 可观测性系统构建

集成 LangSmith、Langfuse、Arize

日志追踪结构设计：

支持行为回放与调试

集成 Prometheus + Grafana 实时监控指标（成功率、响应时间、错误率）

多模态 Agent 开发

图像理解 + LLM 推理结合

使用 CLIP 或 BLIP 提取图像语义

将图像描述送入 LLM 生成推理结果

GUI Agent（屏幕识别 + 自动操作）

使用 PyAutoGUI + OCR + LLM 构建自动化助手

移动端大模型部署

NPU 调度原理与优化（Qualcomm Hexagon、Apple Neural Engine）

ONNX Runtime Mobile 部署实战：

模型转换（PyTorch → ONNX）

加载模型并执行推理

CoreML（iOS）与 TFLite（Android）部署实践

模型量化、剪枝、蒸馏优化策略

性能优化技巧：

使用 INT8 量化降低内存占用

利用 NPU 加速矩阵运算

缓存高频词嵌入向量

自主学习 Agent

基于 RL 的 Agent 探索与环境交互

强化学习基础（Q-Learning、PPO）

设计 Reward 函数引导 Agent 学习

自我反思机制（Self-Reflection）

使用 Prompt 引导模型评估自身输出质量

Agent 自动生成（AutoAgent）

基于用户需求自动组合工具与角色

动态生成 DSL 流程并执行

实践练习：

项目一：构建一个具备短期+长期记忆、多模态输入、移动端推理、自我反思能力的智能 Agent

输入输出定义：

输入：文本、图像、语音、GUI 截图

输出：自然语言回答、执行动作、可视化日志

核心功能：

支持图像上传并识别内容

支持语音转文字并理解意图

支持移动端本地推理

支持记忆读写与自我反思

项目二：支持图像识别、语音输入、本地部署、远程监控与调试

场景设定：

用户上传发票图片，Agent 识别金额、日期、公司名称

用户语音提问：“这张发票金额是多少？”

Agent 返回解析结果并记录到长期记忆库

支持远程查看日志、下载模型、更新配置

技术难点：

如何实现跨平台一致性？

如何保证隐私安全（不上传原始数据）？

如何远程管理多个设备上的 Agent？

# 模块八模型部署与服务化

Docker 入门与镜像构建

Kubernetes 编排基础（Pod、Deployment、Service）

Kubernetes 高级配置（HPA、滚动更新、金丝雀发布）

模型服务化部署（FastAPI + Uvicorn + Gunicorn）

模型压缩与量化（OpenVINO、TensorRT、ONNX）

分布式推理服务设计（Ray Serve、Triton Inference Server）

Prometheus + Grafana 监控系统搭建

日志收集与异常检测（ELK Stack + Fluentd）

A/B 测试与灰度发布（Traefik + Istio）

实践练习：

实践一：将意图识别模型打包成容器并在 K8s 中部署

实践二：将一个大模型服务部署到 K8s 集群，支持自动扩缩容与流量控制

场景设定：

部署 LLaMA 或 ChatGLM 模型

使用 Ray Serve/Triton 加速推理

配置 Istio 实现 A/B 测试和灰度发布

技术难点：

如何降低模型延迟？

如何防止资源耗尽？

如何实现无缝升级？

# 模块九Python 高性能编程与并发工程

异步 I/O 底层原理（event loop、async/await）

异步编程（asyncio、aiohttp）

多线程与多进程（concurrent.futures、multiprocessing）

多进程通信与共享内存（multiprocessing.Value, Manager）

使用 Pydantic 进行数据校验

GIL 影响分析与规避策略

线程池与协程池性能对比

使用 Py-Spy / cProfile 定位性能瓶颈

FastAPI 性能调优（连接池、限流、缓存中间件）

RESTful API 开发（FastAPI / Flask）

WebSocket 与 gRPC 实战

单元测试与自动化测试（pytest）

性能分析与优化技巧

底层性能优化 （CUDA 编程基础、TensorRT 加速原理）

实践练习：

项目一：实现一个支持并发的 HTTP+WebSocket 混合通信服务

输入输出定义：

HTTP 接口：接收用户输入并返回结果

WebSocket 接口：实时推送状态更新

技术难点：

如何协调 HTTP 与 WebSocket 的消息传递？

如何防止并发冲突？

如何实现异步事件通知？

项目二：实现一个支持高并发、限流、缓存和日志追踪的聊天代理服务（每秒处理 1000+请求）

场景设定：

用户通过 HTTP 或 WebSocket 提问

服务调用 LLM 并缓存结果

支持限流、错误重试、日志追踪

技术难点：

如何设计缓存结构？

如何保证高并发下稳定性？

如何实现请求追踪 ID？

项目三：“基于 CUDA 加速的向量相似度计算优化”，提升 RAG 检索速度

场景设定：

在 FAISS 中使用 GPU 加速近似最近邻搜索

对比 CPU 与 GPU 的性能差异

技术难点：

如何配置 FAISS 的 GPU 环境？

如何评估加速效果？

如何封装为可复用的组件？

# 模块十项目实战 —— 工程化企业级智能客服平台

支持多租户架构（不同客户/品牌）

高并发接入（Web、微信、App、第三方接口）

插件化设计（支持快速扩展新业务）

可扩展的对话引擎（支持意图识别 + RAG + Agent）

支持多模型切换（GPT、LLaMA、ChatGLM 等）

支持知识库热更新、模型热加载

高可用后台服务（Docker、Kubernetes）

提供可视化后台（知识库管理、对话记录、监控面板）

支持灰度发布、A/B 测试、异常报警机制

支持自定义知识库更新和模型迭代

项目：

提交一份完整的 Agent 平台源码包（含 Web 后台 + 移动 App + 多 Agent 系统 + DSL 引擎）

文档：

架构设计

部署说明

API 文档

DSL 规范

性能报告

演示视频：

展示多 Agent 协作

任务编排

移动端推理等核心功能

# 模块十一行业场景与产品设计

AI 产品设计方法论

需求分析阶段

如何发现用户痛点？

如何从业务流程中识别 AI 可介入环节？

用户画像构建与场景建模

原型设计阶段

使用 Figma / Axure 制作产品原型图

构建最小可行产品（MVP）思路

设计 AI 对话流程图（DSL 或状态机）

用户反馈迭代

A/B 测试设计

用户满意度调查

日志回放 + 人工审核机制

垂直领域经验积累

医疗领域

医疗文本 NER（命名实体识别）：

提取疾病名、药品名、症状、检查项等

医疗问答系统：

基于 RAG 的常见病解答 Agent

医疗术语标准化（ICD 编码映射）

金融领域

风控指标解析：

用户信用评分、逾期率、负债比等

自动报告生成：

基于财报内容自动生成摘要

合规性生成：

自动生成合同条款、风险提示语句

法律领域

法条检索与匹配：

用户输入问题 → 匹配相关法律条文

合同审查辅助：

标记合同中的高风险条款

案例推荐：

相似案例推荐 + 判决结果预测

AI 伦理与法规合规

数据隐私保护

GDPR、CCPA、《个人信息保护法》解读

数据脱敏与匿名化技术（k-匿名、差分隐私）

敏感信息过滤（PII Detection）

AI 偏见与公平性

检测算法偏见的方法（Fairness Indicators）

公平性评估指标（Demographic Parity、Equal Opportunity）

模型去偏策略（Reweighting、Adversarial Debiasing）

实践练习 ：

项目一：在工程化企业级智能客服平台基础上增加“法律咨询 Agent 开发”

场景设定：

用户输入法律问题（如“离婚财产如何分割？”）

系统自动检索相关法条、类案、司法解释

LLM 生成解释性回答并提供参考建议

技术难点：

如何构建法律知识库？

如何实现法条+案例+生成一体化？

如何保证输出的合规性？

项目二：阿里云百炼平台实践案例

场景设定：

在阿里云百炼平台上部署一个客服 Agent

配置 RAG 知识库、多轮对话流程、意图识别引擎

实现日志追踪、A/B 测试、灰度发布

技术难点：

如何接入百炼 API？

如何配置模型参数与 Prompt？

如何进行性能调优与成本控制？

项目三：HuggingFace 平台实践

场景设定：

使用 HuggingFace Spaces 部署一个聊天机器人

使用 Transformers Pipeline 快速部署推理服务

使用 Gradio 构建可视化界面

技术难点：

如何上传模型到 Model Hub？

如何使用 Inference API？

如何优化加载速度与响应时间？

# 第 1 周：AI 概述与基础知识 + 数学基础

知识点

* 课程介绍与环境准备：了解课程目标、形式、带练方式，熟悉本地 Python 环境和阿里云 Notebook 的使用。
* AI 发展与核心概念： 回顾 AI 的发展历程，明确人工智能（AI）、机器学习（ML）、深度学习（DL）和生成式 AI 的基本概念及其相互关系。了解中国 AI 产业化浪潮中互联网大厂的战略布局。
* 主要学派简介与代表方法： 初步了解符号主义（规则与逻辑、知识图谱）、行为主义（强化学习）、连接主义（神经网络与深度学习）的主要思想和代表方法。
* 数学基础（微积分初步）:
  + 函数概念：简单介绍函数的概念（例如，输入到输出的映射）。
  + 导数的概念与直观解释：重点讲解导数的概念，可以从斜率入手，解释导数是函数在某一点的变化率。
  + 偏导数简介：简单介绍偏导数的概念，用于理解多元函数的变化。
  + 导数在优化问题中的应用：强调导数在优化问题中的作用，例如寻找函数的最小值（为后续的梯度下降算法做铺垫）。
* 数学基础（概率论初步）：
  + 概率的基本概念：介绍概率的基本概念（事件、样本空间、概率的定义）。
  + 条件概率简介：简单介绍条件概率的概念。
  + 随机变量：介绍随机变量（离散型和连续型）的概念。
  + 常见概率分布简介：简单介绍常见的概率分布（如均匀分布、正态分布），理解数据分布的概念。

产业案例与实践带练

* 平台体验与环境配置：登录阿里云 Notebook，演示 Python 环境及 GPU 调用。安装 PyTorch 并验证 GPU 是否可用。
* 早期 AI 案例体验：了解早期经典案例，如 GPS（简单介绍）与 Deep Blue（结合搜索与评估函数）。快速演示简化的搜索脚本（如 BFS/A\* 算法应用于物流路径规划）。展示感知机二分类脚本及可视化边界，帮助学员直观理解基本原理。
* 数学概念的应用理解：使用简单的函数进行直观解释导数的概念，并思考导数如何在寻找函数最小值时发挥作用。在数据科学与机器学习中，优化算法是模型训练的核心，了解导数和偏导数的概念可以帮助理解梯度下降等算法的本质。
* 数据准备初步：演示 MNIST、CIFAR 等数据集的下载、准备与初步加载过程。了解数据收集、清洗与预处理的基本概念和常用方法。

# 第 2 周：传统机器学习模型 + 数学基础强化

知识点

* 监督学习基础：理解监督学习的概念，区分分类与回归任务。
* 线性模型：掌握线性回归的原理与应用场景；理解逻辑回归的原理及其在二分类问题中的应用。
* 决策树与集成学习：了解决策树的原理和构建过程；理解集成学习的概念，包括 Bagging 和 Boosting；掌握随机森林和 XGBoost 算法的原理与应用。
* 无监督学习：理解无监督学习的概念，区分聚类与降维任务；掌握 K-Means 聚类算法和 PCA 降维算法的原理与应用。
* 模型评估：了解常用的模型评估指标，如准确率、精确率、召回率和 F1 分数等。

产业案例与实践带练

* 线性模型实践：
  + 在 Notebook 上使用 scikit-learn 实现线性回归模型，并应用于预测任务。
* 决策树与集成学习实践：
  + 在 Notebook 上使用 scikit-learn 实现决策树和随机森林模型进行分类任务。
  + 使用 XGBoost 库实现分类或回归任务。
* 无监督学习与模型评估实践：
  + 在 Notebook 上使用 scikit-learn 实现 K-Means 聚类算法，K-Means 常用于客户细分、市场分析等场景，例如将用户分为不同的群体以便进行精准营销。
  + 使用 scikit-learn 实现 PCA 降维并进行可视化。PCA 降维技术可以用于特征提取和数据压缩，例如在图像处理和基因数据分析中降低数据的维度。
  + 学习如何根据具体的业务问题选择合适的模型评估指标，例如在垃圾邮件分类中，精确率和召回率都很重要。

# 第 3 周：知识表示与检索（聚焦现代知识图谱与向量检索）

知识点

* 符号表示、信息检索与知识图谱基础：理解符号表示与可解释规则；掌握信息检索的基本概念；了解知识图谱的概述和基本构成。
* 向量检索与向量数据库应用：掌握文本向量检索的基础知识；理解向量数据库的概念及其在现代应用中的作用。
* 规则引擎与大模型融合：了解符号推理与规则引擎的基本原理；理解大模型与符号融合的意义和方法。
* 进阶应用与负责任的 AI 考量：了解启发式搜索在实际应用中的作用；对比符号搜索与大模型生成方法的特点；理解负责任的 AI 原则及其重要性。

产业案例与实践带练

* 知识图谱初步体验：
  + 访问一个公开的知识图谱（例如 DBpedia 或 Wikidata），使用其提供的简单查询界面或工具，探索知识图谱的基本结构。
* 向量检索基础体验：
  + 演示文本向量的生成过程，并使用简单的向量相似度计算方法进行文本检索。
* 规则引擎与大模型交互：
  + 通过 Python 实操，演示如何使用简单的规则引擎进行推理。
  + 展示 RAG（Retrieval-Augmented Generation）流程，了解如何结合规则和检索增强大模型的生成能力。
* 启发式搜索与负责任的 AI 讨论：
  + 演示 A\* 算法在路径规划等问题中的应用。
  + 符号搜索与大模型生成结果的对比实验，分析各自的优缺点。

# 第 4 周：控制论与强化学习基础

知识点

* 控制论与强化学习基础：理解控制论的基本概念，如反馈系统与自动控制；掌握强化学习的基本概念，包括智能体、环境、奖励和动作；区分强化学习与传统机器学习。
* Q-Learning：掌握 Q-Learning 算法的原理，理解动作-价值函数 Q(s, a) 的含义；了解 Q-Learning 的更新规则以及探索-利用的平衡策略；理解经验回放的概念。
* 现代应用与课后练习：了解强化学习的现代应用，如深度强化学习（DRL）；初步认识深度 Q 网络（DQN）及其重要性。
* 深度强化学习入门：理解深度强化学习（DRL）的概念，即如何将深度学习与强化学习相结合；掌握 DQN 的基本原理，了解如何使用神经网络近似 Q 函数；理解经验回放与目标网络的作用。

产业案例与实践带练

* 控制论与强化学习初步体验：
  + 演示经典的马尔可夫决策过程（MDP）和简单的强化学习问题，例如一个简单的网格世界。
  + 图示说明控制论中的反馈机制如何应用于机器学习中的决策过程。
* Q-Learning 算法实践：
  + 使用 Q-Learning 在一个简单的迷宫问题中训练智能体。
  + 通过不断更新 Q 表来提高智能体的决策能力，使其找到迷宫的出口。
* 深度 Q 网络（DQN）初步体验：
  + 使用 DQN 实现一个简单的 Atari 游戏（如 Pong）的训练任务。了解如何使用神经网络来学习策略，并观察智能体在游戏中的表现。

# 第 5 周：感知机与浅层网络

知识点

* 感知机提出、线性可分性 & 多层感知机简介：了解感知机的原理，即单层神经网络的基本构造；理解线性可分性以及感知机的局限性；掌握多层感知机（MLP）的概念与基本结构。
* 多层感知机 (MLP) 进阶与正则化：掌握多层感知机的前馈与反向传播（Backpropagation）的基本概念；了解常用激活函数（如 Sigmoid、ReLU）的选择原则；理解过拟合的概念以及常用的正则化技术（如 L2 正则化、Dropout、早停等）。
* 反向传播（BP） & 神经网络复兴：理解反向传播（Backpropagation）算法的基本原理，即如何计算梯度并更新权重；了解梯度消失与梯度爆炸问题；理解网络复兴与深度学习兴起的历史背景。
* 深度学习实践与应用展望：了解深度学习在实际中的应用领域，如图像识别、语音识别、机器翻译等；探讨 MLP 在多领域中的潜力和挑战。

产业案例与实践带练

* 感知机实现与局限性体验：
  + 使用感知机实现简单的二分类任务，例如解决 AND 或 OR 逻辑问题。
  + 比较感知机与多层感知机在解决非线性问题上的差异，直观理解感知机的局限性。
* 多层感知机训练与正则化效果体验：
  + 实现一个简单的多层感知机（MLP）并进行训练与验证，观察其在分类或回归任务上的表现。
  + 调整正则化参数（如 L2 正则化的系数或 Dropout 的概率），观察正则化技术如何改善模型的性能，防止过拟合。
* 反向传播原理理解：
  + 通过一个简单的神经网络结构，手动推导反向传播算法中权重的更新过程（无需实际编写代码，重点理解原理）。
* MNIST 手写数字分类实践：
  + 在 MNIST 数据集上使用 MLP 完成手写数字分类任务，观察模型的训练过程和最终的分类准确率。

# 第 6 周：现代神经网络：CNN 与 RNN

知识点

* CNN 核心 & 经典架构：掌握卷积神经网络（CNN）的基本结构，包括卷积层、池化层和全连接层的作用；了解经典的 CNN 架构，如 LeNet、AlexNet 和 VGG 的特点。
* ResNet & CNN 工业技巧：理解 ResNet（残差网络）的架构以及跳跃连接的概念；了解 CNN 在工业应用中常用的技巧，如 BatchNorm 和 Data Augmentation。
* RNN 核心 & LSTM/GRU：掌握循环神经网络（RNN）的基本原理及其在处理序列数据上的应用；了解 LSTM 和 GRU 的门控机制，以及它们如何解决传统 RNN 的梯度消失问题。
* NLP 基础与 Embedding：了解自然语言处理（NLP）的基础概念，如词级和字符级处理；掌握嵌入技术（Embedding）的基本原理，了解 Word2Vec 和 GloVe 的作用。

产业案例与实践带练

* CNN 图像分类实践：
  + 使用 CNN 进行图像分类任务，例如在 CIFAR-10 数据集上训练一个简单的 CNN 模型。
  + 观察卷积层如何提取图像特征，池化层如何降低维度。
* ResNet 网络体验与工业技巧应用：
  + 使用预定义的 ResNet 网络进行图像分类任务，理解残差连接的作用。
  + 了解 BatchNorm 如何加速网络训练和提高稳定性，以及 Data Augmentation 如何通过人为增加数据多样性来提升模型泛化能力。
* RNN 处理序列数据体验：
  + 使用 RNN、LSTM 或 GRU 处理时间序列数据，例如进行简单的文本生成任务，观察模型如何学习序列中的依赖关系。
* 词嵌入可视化与相似度计算：
  + 使用预训练的 Word2Vec 模型生成词嵌入，并通过可视化工具（如 t-SNE）展示词语在向量空间中的分布。
  + 进行简单的词语相似度计算，理解词嵌入如何捕捉词语的语义信息。

# 第 7 周：Transformer 与大模型

知识点

* Transformer 核心 & 自注意力机制：掌握 Transformer 的基本原理，包括自注意力机制（Self-Attention）和 Query/Key/Value 的概念；理解多头注意力机制和位置编码的作用；了解 Transformer 相较于 RNN 的优势。
* 大模型（BERT/GPT） & 中文预训练：了解 BERT（Bidirectional Encoder Representations from Transformers）和 GPT（Generative Pretrained Transformer）的基本概念；理解预训练与微调（Fine-Tuning）的策略；了解中文大模型（如 ChatGLM 等）应用的特点。
* 生成式 AI 基础与应用：理解生成式 AI 的基本概念；了解生成模型在文本生成、图像生成等领域的应用；掌握生成式对话系统的工作原理。
* 图神经网络（GNN）入门：了解图神经网络（GNN）的概念与基本原理；初步认识 GCN（Graph Convolutional Networks）和 GraphSAGE 的基本算法；理解 GNN 在处理图数据（如社交网络、知识图谱）中的优势。

产业案例与实践带练

* Transformer 模型初体验：
  + 在简单的文本分类任务中，使用现有的深度学习框架实现一个基础的 Transformer 模型。
  + 观察自注意力机制是如何捕捉文本内部不同词语之间的关联的。
* BERT 与 GPT 应用体验：
  + 使用预训练的 BERT 模型进行中文文本分类任务，体验微调策略在实际应用中的效果。
  + 使用预训练的 GPT 模型进行简单的文本生成任务，感受其强大的文本生成能力。
* 生成式 AI 应用演示：
  + 使用预训练的 GPT 模型进行简单的对话生成，体验生成式对话系统的交互过程。
  + 演示文本到图像生成模型（如 DALL·E 或类似的开源模型）的效果，了解生成式 AI 在内容创作方面的潜力。
* 图神经网络初步探索：
  + 了解图数据的表示方式，并尝试使用现有的 GNN 库（如 PyTorch Geometric）实现一个简单的图神经网络模型。

# 第 8 周：深度强化学习进阶与多范式融合趋势

知识点

* DQN 改进与实践：了解 DQN 的改进方法，如 Double DQN 和 Dueling DQN；理解强化学习中稳定性问题及其解决方法；了解 DQN 在实际任务中的应用案例。
* 策略梯度与 PPO：掌握策略梯度方法，初步了解 REINFORCE 算法；理解近端策略优化（Proximal Policy Optimization，PPO）的核心机制；对比策略梯度与 Q-Learning 的异同。
* RL 在工业应用与前沿进展：了解强化学习在无人机控制、机器人导航、工业自动化等领域的应用；初步认识基于人类反馈的强化学习（Reinforcement Learning with Human Feedback，RLHF）的概念和最新进展。
* 多范式融合趋势与案例：理解符号主义、行为主义和连接主义的融合趋势；分析实际案例，了解如何结合符号推理与深度学习等不同范式的方法。

产业案例与实践带练

* DQN 改进算法体验：
  + 尝试实现 Double DQN 或 Dueling DQN 算法，并在 OpenAI Gym 等环境中训练智能体，观察其性能是否有所提升。
* PPO 算法实践：
  + 使用现有的强化学习库（如 Stable Baselines3）实现 PPO 算法，并在 OpenAI Gym 环境中训练智能体完成特定任务。
* 强化学习工业应用案例分析：
  + 分析强化学习在工业界应用的成功案例，例如无人机自主飞行、机器人抓取物体、工业流程优化等，了解 RLHF 在提升大语言模型性能方面的作用。
* 多范式融合案例分析与讨论：
  + 分析一个融合了符号主义、行为主义和连接主义的实际 AI 系统案例，例如一个结合知识图谱和深度学习的智能问答系统，或者一个利用规则和强化学习的机器人控制系统，讨论不同范式结合的优势和挑战。

# 第 9 周：模型部署与推理优化

知识点

* 模型部署基础与容器化：了解模型部署的整体流程；初步掌握容器化技术 Docker 的基本概念和常用操作；学习如何构建包含模型推理代码的 Docker 镜像，为不同环境的部署打下基础。
* 在线推理服务构建：了解如何使用 Flask 或 FastAPI 等框架搭建简单的模型推理 API，为云端和边缘端的在线服务提供基础；理解 RESTful API 设计原则。
* 推理性能优化初步与框架选型：了解影响模型推理性能的主要因素，如延迟、吞吐量和资源占用；初步了解模型量化、剪枝和知识蒸馏等优化技术，以及它们在不同部署环境下的适用性；重点介绍云端（如 TensorFlow Serving, TorchServe）、边缘端（如 TensorFlow Lite, ONNX Runtime）和端侧设备（如 Core ML, MediaPipe）常用的推理框架及其特点和选型考虑。
* 云边端模型部署方案：
  + 云端部署： 了解云平台（如阿里云 ModelArts, AWS SageMaker, Google AI Platform）提供的模型部署服务及其优势。
  + 边缘端部署： 理解边缘计算的概念和优势，了解在边缘设备上部署模型的常见方式和挑战。
  + 端侧部署： 了解在移动设备、嵌入式设备等端侧部署模型的特殊考虑，如模型大小、功耗和硬件加速。

产业案例与实践带练

* 模型容器化与云端推理框架体验：
  + 使用 Dockerfile 构建一个包含模型推理代码的容器。
  + 如何在容器中集成云端常用的推理框架（如 TensorFlow Serving 或 TorchServe），并进行简单的模型部署和测试。
* 在线推理 API 构建与边缘端推理框架初步：
  + 使用 Flask 或 FastAPI 搭建一个简单的 Web 服务，该服务能够接收输入数据，调用预训练的模型进行推理，并将结果返回给用户。
  + 初步体验在本地使用边缘端推理框架（如 TensorFlow Lite 或 ONNX Runtime）加载模型并进行推理，为后续边缘部署打下基础。
* 推理性能优化方法与端侧推理框架了解：
  + 演示如何对一个简单的模型进行量化操作，并观察模型大小和推理速度的变化。
  + 介绍端侧推理框架（如 Core ML 或 MediaPipe）的基本概念和使用场景，了解其在移动设备和嵌入式设备上的优势。
* 云边端模型部署流程了解：
  + 演示如何在阿里云 ModelArts 或类似平台上进行云端模型部署，并了解其提供的不同部署选项（如在线服务、批量预测等）。
  + 介绍边缘计算平台（如阿里云 Link IoT Edge）和端侧部署工具的基本使用流程和注意事项，熟悉不同部署环境的特点和流程，能够帮助选择合适的部署方案。

# 第 10 周：模型监控与自动化运维

知识点

* 模型监控的重要性与关键指标：理解模型监控的目的和意义，为何需要对已部署的模型进行持续监控；掌握关键监控指标，如响应时间、吞吐量、准确率、漂移检测等；了解监控数据的采集与可视化方法，如何通过日志、指标和仪表盘来追踪模型性能。
* 自动化部署与持续集成 / 持续交付 (CI/CD)：了解 CI/CD 的概念和优势，以及在机器学习模型部署中的应用；初步认识常用的自动化构建和部署工具（如 Jenkins 或 GitLab CI）；理解模型版本管理的重要性以及如何确保模型更新的顺利实施。
* 告警与日志管理：掌握告警系统的设置与配置方法，当模型性能下降或系统出现异常时，如何进行告警通知；了解日志记录的最佳实践，以及如何记录与分析模型推理日志；初步了解云平台提供的监控和日志服务。
* AI 系统的安全性、合规性与伦理：了解 AI 系统面临的安全性问题，如模型被攻击或滥用；理解合规性要求，如数据隐私保护等法律法规；认识 AI 伦理问题，如模型决策的透明性、公平性和可解释性。

产业案例与实践带练

* 模型监控指标可视化体验：
  + 演示如何使用简单的监控工具（如 Prometheus 或 Grafana 的开源版本）对一个模拟的在线推理服务进行监控，并创建可视化仪表盘来实时展示模型的响应时间、吞吐量等指标。
* 自动化部署流程初步体验：
  + 演示如何使用简单的脚本或工具（如 GitHub Actions 或 GitLab CI 的免费版本）设置一个基本的自动化构建和部署流程，例如当模型代码发生变化时，自动构建 Docker 镜像并部署到测试环境。
* 告警设置与日志分析初步：
  + 了解如何在云平台或本地设置简单的告警规则，例如当模型的预测准确率低于某个阈值时发送通知。
  + 学习如何查看和分析模型推理日志，从中发现潜在的问题或异常。
* AI 系统安全与伦理案例讨论：
  + 讨论一些常见的 AI 安全风险案例，例如对抗性攻击。
  + 探讨在模型开发和部署过程中需要考虑的合规性问题，例如如何保护用户数据隐私。
  + 分析 AI 伦理方面的案例，例如算法偏见可能带来的社会影响。

# 第 11 周：可解释 AI (XAI) 与模型安全

知识点

* 可解释 AI (XAI) 基础与方法：理解可解释 AI 的重要性，为何需要让 AI 决策过程更加透明；初步了解 LIME（局部可解释模型）与 SHAP（Shapley 值）等常用的可解释性方法；了解可解释性在实际 AI 决策中的应用场景。
* 模型安全与对抗性攻击：了解模型面临的安全风险，如对抗性攻击的概念和常见类型；初步认识对抗性攻击是如何通过微小扰动欺骗模型的；了解防御对抗性攻击的一些基本方法，如对抗训练。
* 模型鲁棒性与泛化能力：理解模型鲁棒性的概念，即模型对不同数据或扰动的适应能力；掌握泛化能力的概念，即模型在未见数据上的表现；区分过拟合和欠拟合及其对模型性能的影响。
* 负责任的 AI：偏见、公平性与可信度：了解负责任的 AI 的关键特征，如公平、透明和可信；认识模型中可能存在的偏见及其来源和影响；理解公平性和可信度在 AI 应用中的重要性。

产业案例与实践带练

* 可解释性方法初步体验：
  + 使用现有的 Python 库（如 lime 或 shap）对一个简单的分类模型（例如使用 scikit-learn 训练的）进行可解释性分析。
  + 观察 LIME 如何解释单个样本的预测结果，以及 SHAP 值如何衡量每个特征对预测结果的贡献。
* 对抗性攻击简单演示：
  + 了解如何使用现有的工具或库生成针对简单图像分类模型的对抗性样本。观察微小的、人眼难以察觉的扰动是如何导致模型做出错误预测的。
* 模型泛化能力评估：
  + 使用交叉验证等方法评估一个模型在不同数据子集上的性能，从而了解模型的泛化能力。
  + 尝试使用正则化等技术来提高模型的泛化能力，避免过拟合。
* 模型偏见检测与公平性分析：
  + 了解在训练数据中可能存在的偏见来源。
  + 使用简单的数据分析方法检测模型在不同群体上的性能差异，初步了解如何评估模型的公平性。

# 第 12 周：多模态学习与前沿应用

知识点

* 多模态学习基础： 理解多模态学习的概念，即如何融合来自不同来源的数据（如图像、文本、音频）；了解多模态数据的特征与挑战，以及常用的融合方法（如特征级融合和决策级融合）。
* 生成式多模态模型： 了解生成式多模态模型的概念，以及如何通过生成模型联合学习不同模态的数据；初步认识扩散模型（Diffusion Model）及其在图像生成等领域的应用。
* AutoML 简介与实践： 了解 AutoML 的概念，即自动化机器学习的定义与发展；初步认识常用的 AutoML 技术与工具（如 AutoKeras、TPOT 等）。
* 联邦学习初步： 了解联邦学习的基本概念和应用场景，特别是在保护数据隐私方面的优势。

产业案例与实践带练

* 多模态数据融合体验：
  + 了解一些常用的多模态数据集（例如包含图像和文本描述的数据集）。
  + 演示如何使用简单的深度学习模型（例如分别处理图像和文本的两个网络，然后将它们的特征进行融合）完成一个基本的多模态任务，如图像描述生成或视觉问答。
* 生成式多模态模型体验：
  + 了解扩散模型的基本工作原理。
  + 使用一些预训练的扩散模型进行图像生成实验，例如输入一段文本描述，观察模型如何生成对应的图像。
* AutoML 工具初步体验：
  + 使用 AutoML 工具（如 AutoKeras 或 TPOT 的简化版本）在一个小型数据集上进行模型搜索和超参数优化。
  + 观察 AutoML 工具如何自动选择合适的模型结构和参数。
* 联邦学习案例了解：
  + 了解联邦学习在医疗、金融等数据隐私敏感领域的应用案例。
  + 理解联邦学习如何在不共享原始数据的情况下，协同训练模型，保护用户隐私。

# 第 13 周：课程回顾与项目展望

知识点

* 课程内容回顾与知识点串联： 回顾整个课程的主要内容，梳理不同学派（符号主义、行为主义、连接主义）和关键技术（传统机器学习、深度学习、强化学习、生成式 AI 等）之间的联系与演进历程。
* 学习资源推荐与未来学习路径：了解 AI 领域优质的学习资源，如经典书籍、在线课程、顶会期刊等；探讨未来在 AI 领域深入学习和发展的路径和方向。
* 期末项目介绍与个人方案设计： 给出以下 两个 期末项目选题方向和具体要求：
* 选题方向一：工业级 ChatGPT 系统设计方案
  + 具体要求： 学员需要独立完成一份详细的工业级 ChatGPT 系统设计方案。
  + 目标： 考察学员对大型语言模型、Transformer 架构、预训练与微调、RLHF（基于人类反馈的强化学习）、模型部署与优化、安全与伦理等相关知识的理解和应用能力。
  + 方案设计应包含但不限于：
    - 系统架构设计： 对 ChatGPT 的核心组成部分进行设计，例如数据收集与处理模块、模型训练模块、推理服务模块、用户交互界面等。
    - 技术选型： 详细说明每个模块拟采用的关键技术和框架（例如：Transformer 模型变种、预训练数据集、微调策略、RLHF 方法、推理优化技术、部署平台等），并阐述选择理由。
    - 数据与模型： 探讨训练工业级 ChatGPT 所需的数据规模、类型和来源，以及模型选择和训练策略。
    - 部署与运维： 考虑模型的部署方式（例如：云端部署、API 服务）和后续的监控、维护、更新策略。
    - 安全与伦理考量：分析工业级 ChatGPT 可能面临的安全风险（例如：对抗性攻击、恶意使用）和伦理问题（例如：偏见、虚假信息），并提出相应的应对方案。
    - 性能指标与评估： 确定衡量系统性能的关键指标（例如：生成质量、流畅度、响应速度）和评估方法。
* 选题方向二：工业级 Manus AI 代理系统设计方案
  + 具体要求： 学员需要独立完成一份详细的 Manus AI 代理系统设计方案。
  + 目标： 考察学员对多智能体系统、大型语言模型、知识检索、任务规划与执行、自主学习、透明性与可解释性等相关知识的理解和应用能力。
  + 方案设计应包含但不限于：
    - 系统架构设计： 对 Manus 的核心组成部分进行设计，例如：多智能体管理模块、知识库模块、任务分解与规划模块、执行引擎、用户交互界面、过程记录与回放模块等。
    - 技术选型： 详细说明每个模块拟采用的关键技术和框架（例如：大型语言模型、知识图谱、规划算法、执行框架、数据库、过程记录方法等），并阐述选择理由。
    - 智能体设计： 描述不同类型智能体的功能和协作方式（例如：规划智能体、知识检索智能体、代码生成智能体等）。
    - 知识与推理： 探讨系统如何获取、存储和利用知识，及如何任务规划和推理。
    - 自主学习与进化： 探讨系统如何通过与环境和用户的交互进行学习和改进。
    - 部署与应用场景： 分析 Manus 的潜在应用场景和部署方式。

产业案例与实践带练

# 第一章：AI 大模型四阶技术总览

* 深度解读 AI 发展四轮浪潮
  + 技术浪潮：弱人工智能、机器学习、深度学习、大语言模型
  + 应用浪潮：高校共识、硅谷创新、中美博弈
  + 把握浪潮：AI 大模型助力超级个体和小团队
* AI 大模型四阶技术总览
  + 提示工程（Prompt Engineering）
  + AI 智能体（Agents）
  + 大模型微调（Fine-tuning）
  + 预训练技术（Pre-training）

# 第二章：大语言模型技术发展与演进

* 统计语言模型
* 神经网络语言模型
* 基于 Transformer 的大语言模型
  + 注意力机制
  + Transformer 网络架构
  + 预训练 Transformer 模型： GPT-1 与 BERT
  + 暴力美学 GPT 系列模型

# 第三章：大模型开发工具库 Hugging Face Transformers

* Hugging Face Transformers 快速入门
  + Transformers 库是什么？
  + Transformers 核心功能模块
    - 使用 Pipelines 快速实践大模型
    - 使用 Tokenizer 编解码文本
    - 使用 Models 加载和保存模型
* 大模型开发环境搭建
  + 搭建你的 GPU 开发环境
  + Google Colab 测试环境
* 实战 Hugging Face Transformers 工具库

# 第四章：实战 Transformers 模型训练

* 数据集处理库 Hugging Face Datasets
  + Hugging Face Datasets 库简介
  + 数据预处理策略：填充与截断
  + 使用 Datasets.map 方法处理数据集
* Transformers 模型训练入门
  + 模型训练基类 Trainer
  + 训练参数与超参数配置 TrainingArguments
  + 模型训练评估库 Hugging Face Evaluate
* 实战使用 Transformers 训练 BERT 模型
  + bert-base-cased 模型（文本分类任务）
  + distilbert-base-uncased 模型（QA 任务）
  + 第五章：大模型高效微调技术揭秘（上）
* Before PEFT：Hard Prompt / Full Fine-tune
* PEFT 主流技术分类介绍
* PEFT – Soft Prompt 技术（Task-specific Tuning）
  + Prefix Tuning （2021 Stanford）
  + Prompt Tuning （2021 Google）
* PEFT – Soft Prompt 技术（Prompt Encoder）
  + P-Tuning v1 （2021 Tsinghua, MIT）
  + P-Tuning v2 （2022 Tsinghua, BAAI, Shanghai Qi Zhi Institute）
  + 第六章：大模型高效微调技术揭秘（下）
* PEFT 基于重参数化（Reparametrization-based）训练方法
  + LoRA 低秩适配微调 （2021 Microsoft）
  + AdaLoRA 自适应权重矩阵微调 （2023 Microsoft, Princeton, Georgia Tech）
  + QLoRA 量化低秩适配微调 （2023 University of Washington）
* UniPELT：大模型 PEFT 统一框架（2022）
* (IA)3：极简主义增量训练方法 （2022）

# 第七章：大模型高效微调工具 Hugging Face PEFT 入门与实战

* Hugging Face PEFT 快速入门
  + PEFT 库是什么？
  + PEFT 与 Transformers 库集成
  + PEFT 核心类定义与功能说明
    - AutoPeftModels、PeftModel
    - PeftConfig
    - PeftType | TaskType
* 实战 PEFT 库 LoRA 模型微调
  + OpenAI Whisper 模型介绍
  + 实战 LoRA 微调 Whisper-Large-v2 中文语音识别

# 第八章：大模型量化技术入门与实战

* 模型显存占用与量化技术简介
* Transformers 原生支持的大模型量化算法
  + GPTQ：专为 GPT 设计的模型量化算法
  + AWQ：激活感知权重量化算法
  + BitsAndBytes（BnB） ：模型量化软件包
* 实战 Facebook OPT 模型量化

# 第九章：GLM 大模型家族与 ChatGLM3-6B 微调入门

* 智谱 GLM 大模型家族
  + 基座模型 GLM-130B
  + 扩展模型
    - 联网检索能力 WebGLM
    - 初探多模态 VisualGLM-6B
    - 多模态预训练模型 CogVLM
    - 代码生成模型 CodeGeex2
* ChatGLM3-6B 微调入门
* 实战 QLoRA 微调 ChatGLM3-6B

# 第十章：实战私有数据微调 ChatGLM3

* 实战构造私有的微调数据集
  + 使用 ChatGPT 自动设计生成训练数据的 Prompt
  + 合成数据： LangChain + GPT-3.5 Turbo
  + 数据增强：提升训练数据多样性
  + 提示工程：保持批量生成数据稳定性
* 实战私有数据微调 ChatGLM3
  + 使用 QLoRA 小样本微调 ChatGLM3
  + ChatGLM3 微调前后效果对比
  + 大模型训练过程分析与数据优化

# 第十一章：ChatGPT 大模型训练技术 RLHF

* ChatGPT 大模型训练核心技术
  + 阶段一：万亿级 Token 预训练语言模型
  + 阶段二：有监督指令微调（SFT）语言模型
  + 阶段三：使用 RLHF 实现人类价值观对齐（Alignment）
* 基于人类反馈的强化学习（RLHF）技术详解
  + 步骤一：使用 SFT 微调预训练语言模型
  + 步骤二：训练奖励模型（Reward Model）
  + 步骤三：使用 PPO 优化微调语言模型
* 基于 AI 反馈的强化学习（RLAIF）技术

# 第十二章：混合专家模型（MoEs）技术揭秘

* 混合专家模型（Mixture-of-Experts, MoEs）技术发展简史
  + 开山鼻祖：自适应局部专家混合（ Michael I. Jordan & Hinton, 1991）
  + 多层次混合：深度MoEs中的表示学习（ Ilya, 2013）
  + 稀疏门控：支持超大网络的MoEs（Hinton & Jeff Dean, 2017）
* MoEs 与 大模型结合后的技术发展
  + GShard：基于 MoE 探索巨型 Transformer 网络（Google, 2020）
  + GLaM：使用 MoE 扩展语言模型性能（Google, 2021）
  + Switch Transformer：使用稀疏技术实现万亿模型（Google, 2022）
* MoEs 实例研究：Mixtral-8x7B-v0.1（Mistral AI, 2023）

# 第十三章：Meta AI 大模型 LLaMA

* Meta LLaMA 1 大模型技术解读
  + 基座模型系列：LLaMA1-7B（13B, 33B, 65B）
  + LLaMA 1 改进网络架构和预训练方法
  + LLaMA 1 衍生模型大家族
* Meta LLaMA 2 大模型技术解读
  + 基座模型系列： LLaMA2-7B（13B, 70B）
  + 指令微调模型：LLaMA2-Chat
  + 申请和获取 LLaMA 2 模型预训练权重

# 第十四章：实战 LLaMA2-7B 指令微调

* 大模型训练技术总结
  + 以模型训练阶段分类：Pre-Training vs Fine-Tuning
  + 以微调权重比例分类：FFT vs PEFT
  + 以模型训练方法分类：Fine-Tuning vs Instruction-Tuning
  + 以模型训练机制分类：SFT vs RLHF
* 实战 LLaMA2-7B 指令微调
  + 指令微调格式：Alpaca Format
  + 数据集：Databricks Dolly-15K
  + 训练工具：HuggingFace TRL
  + 上手训练 LLaMA2-7B 模型

# 第十五章：大模型分布式训练框架 Microsoft DeepSpeed

* 大模型分布式训练框架 DeepSpeed
  + 预训练模型显存计算方法
  + Zero Redundancy Optimizer （ZeRO） 技术
  + DeepSpeed 框架和核心技术
  + 分布式模型训练并行化技术对比
  + DeepSpeed 与 Transformers 集成训练大模型
* DeepSpeed 实战
  + DeepSpeed 框架编译与安装
  + DeepSpeed ZeRO 配置详解
  + 使用 DeepSpeed 单机多卡、分布式训练
  + 实战 DeepSpeed ZeRO-2 和 ZeRO-3 单机单卡训练
* DeepSpeed 创新模块：Inference、Compression & Science

# 第十六章：国产化适配-基于华为昇腾 910 微调 ChatGLM-6B

* 大模型算力设备与生态总结
  + 蓝色星球的算力霸主：NVIDIA
  + 厚积薄发的江湖大佬：Google
  + 努力追赶的国产新秀：华为
* 华为昇腾全栈 AI 软硬件平台介绍
  + AI开发平台：ModelArts
  + 模型开发框架：MindSpore
  + 异构计算架构：CANN
* 实战：华为 Ascend 910B 微调 ChatGLM-6B 模型

### LM 大模型家族介绍

* GLM 系列模型发展历程：从 GLM-10B 到 GLM-4
* GLM-4V：VisualGLM、CogVLM、CogAgent、GLM-4V 的技术变化
* 代码生成模型：CodeGeeX-3 及插件应用
* 图像生成模型 CogView-3
* 超拟人大模型 CharacterGLM
* GLM-4 All Tools

### GLM 模型部署微调实践

* ChatGLM3-6B 开源介绍
* ChatGLM3-6B 快速上手
* 模型微调

### CogVLM 模型部署实践

* CogVLM 开源模型介绍和体验
* CogVLM 开源模型部署