南京大学

智 能 计 算 系 统
实 验 操 作 手 册

实 验 序 号	四
实 验 名 称	基于 MindSpore 实现 GPT 模型的训练和推理
院　　系	计算机学院

2025 年 12 月 4 日

目　录

实验四　基于 MindSpore 实现 GPT 模型的训练和推理

4.1　实验目的

本实验的目的是掌握 MindSpore 深度学习框架的基本使用，了解基于 Transformer 架构的 GPT 模型基本架构以及语言建模任务，使用 MindSpore 构建简易 GPT 模型，并在给定的《金庸武侠小说》数据集上，通过语言建模任务预训练 GPT 模型。然后利用预训练后的 GPT 模型完成一个金庸风格的文本续写任务。具体包括：

1、熟悉和使用 MindSpore 框架、昇腾训练卡和华为云 ModelArts，熟悉 MindSpore 常见 API 的使用方法，熟悉 ModelArts 一站式模型训练和部署平台。

2、使用 MindSpore 构建 GPT 模型，并完成模型的训练和推理。

3、基于预训练后的模型，设计和完成一个金庸风格的文本续写任务。

4.2　背景介绍

4.2.1　MindSpore

MindSpore 是华为推出的深度学习框架，拥有自动微分、并行加持，一次训练，可多场景部署的特性。为全场景 AI 的模型开发、模型运行、模型部署提供端到端能力。MindSpore 支持深度学习算法在 Ascend、GPU、CPU 等硬件平台上开发和部署。

如需查看详情，请参见如下资源：

- MindSpore 教程
- MindSpore Python API

4.2.2 ModelArts

ModelArts 是华为云推出的面向 AI 开发者的一站式开发平台，涵盖了 AI 开发的各个环节，包括数据处理、算法开发、模型训练、模型部署等。从技术上看，ModelArts 底层支持各种异构计算资源，开发者可以根据需要灵活选择使用，而不需要关心底层的技术。同时，ModelArts 支持 Tensorflow、PyTorch、MindSpore 等主流开源的 AI 开发框架。

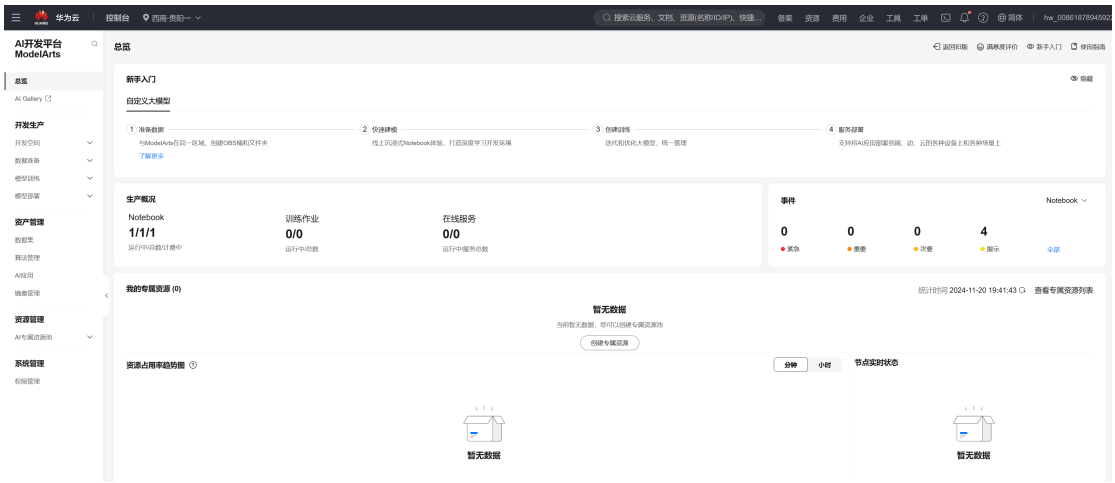

图 4-1 ModelArts 控制台页面

OBS 桶

ModelArts 使用对象存储服务（Object Storage Service，简称 OBS）进行数据存储及模型的备份。OBS 的基本组成是桶和对象。桶是 OBS 中存储对象的容器，对象是代码和数据集文件，每个桶都有自己的存储类别、访问权限等属性。ModelArts 通过访问 OBS 桶中对应的文件来访问数据。因此，在开始训练任务之前，需要将数据和训练脚本上传至 OBS。

上传步骤：

（1）打开 OBS 控制台，创建 OBS 桶（建议区域均选择“西南-贵阳一”，空闲资源较多，且成本较低）。

（2）上传项目目录，包含代码、数据集、以及其他辅助目录（如存放 log 日志文件的目录和存放输出文件的目录）。

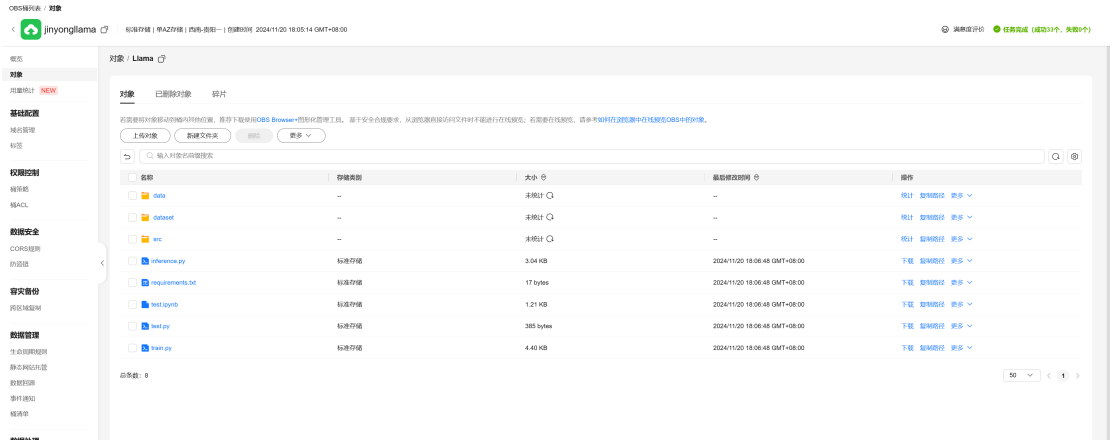

图 4-2 OBS 目录示例

ModelArts 训练和评估页面

在 ModelArts 界面中选择模型训练、训练作业、创建训练作业进入训练作业创建页面，填入相关信息之后，即可启动训练任务。（注意启动方式中选择“Ascend-Powered-Engine”和“mindspore_2.4.10...”）

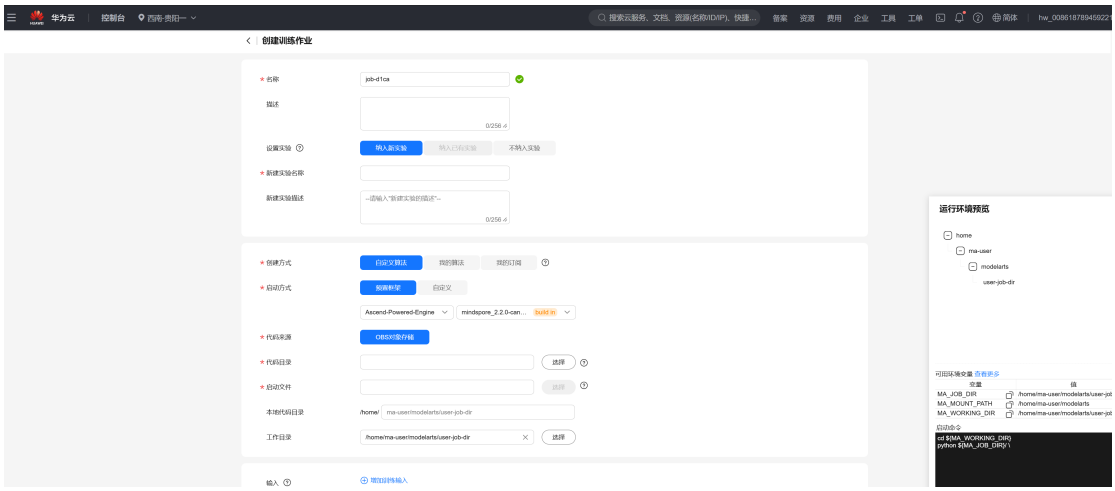

图 4-3 ModelArts 训练作业

通过 Notebook 调试代码

在开始训练和评估之前，通常需要先调试代码，以保证代码的正确性以及执行效率。在 ModelArts 平台上，可以通过平台提供的 Nootbook（Jupyter Lab

环境）进行代码调试和测试。选择“开发空间”、“Notebook”、“创建 Notebook”。创建 Notebook 时，注意选择与训练环境一致的镜像和计算资源规格（mindspore_2.4.10-cann_8.0.0_xxxx）。

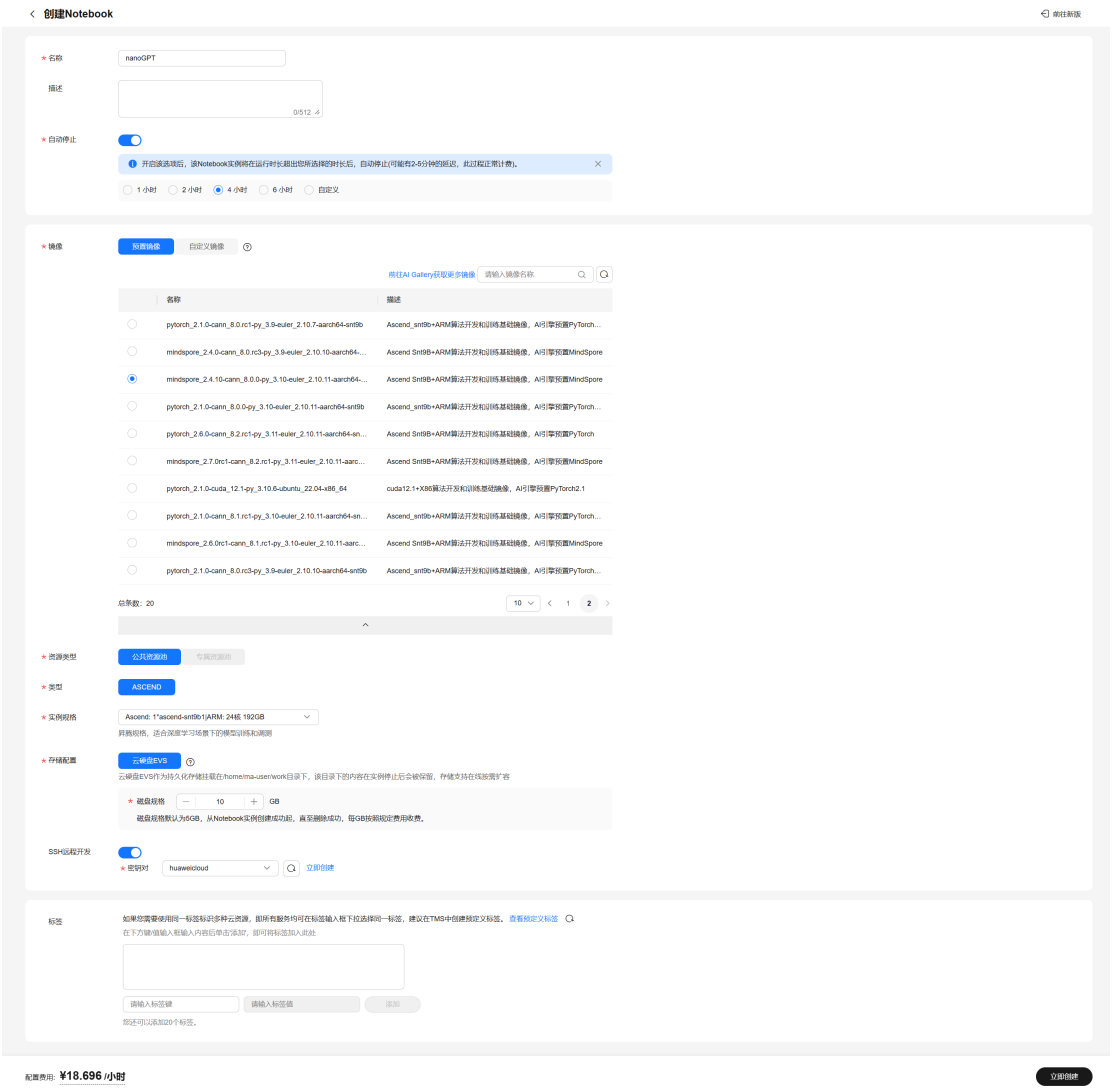

图 4-4 Notebook 创建

进入 Notebook 之后，可以通过 Jupyter Notebook 和终端调试代码。注意，可以通过执行以下代码，将 OBS 中的项目文件，拷贝到当前 Notebook 的工作目录中，以避免重复上传。

```
import moxing
moxing.file.copy_parallel("obs://(OBS桶名称)/(项目目录)", "./(Notebook中的项目目录)")
```

注意：在 Notebook 中更新代码之后，需要将更新同步到 OBS 中的项目目录中，以便在训练和推理中使用。此外，整个训练和推理过程也可以在 Notebook

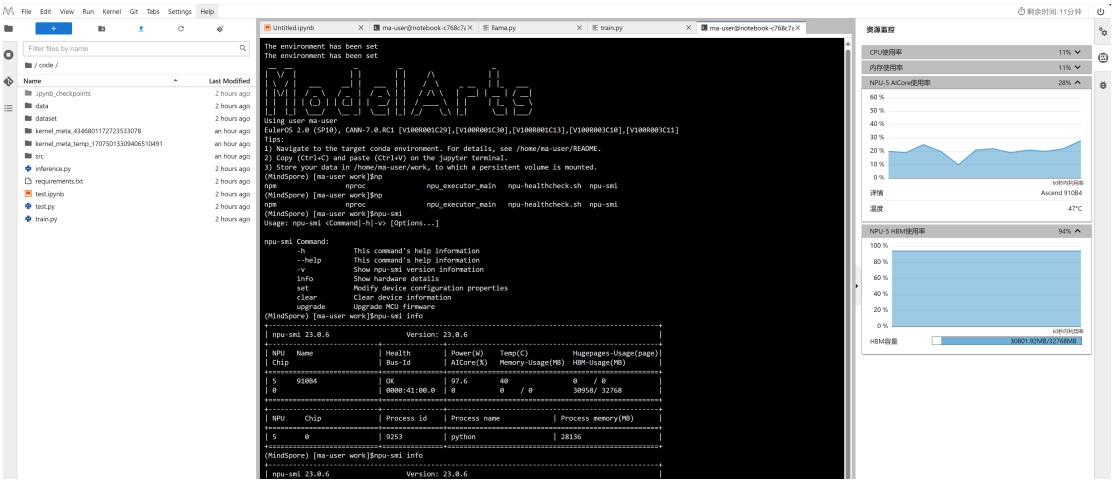

图 4-5 Notebook 页面

中执行。更推荐尝试 ModelArts 提供的训练和推理功能，以体验一站式 AI 开发平台的便捷性。

同时，ModelArts 还提供了 VS Code 扩展和 PyCharm 插件等方式辅助开发和调试代码，感兴趣的同学可以自行了解和使用。

4.2.3 Transformer 架构和 GPT 预训练模型

Transformer 架构通过注意力机制捕捉输入序列中不同位置之间的依赖关系，从而实现对序列数据（如文本）的高效处理，已经在文本和图像任务中展现出。

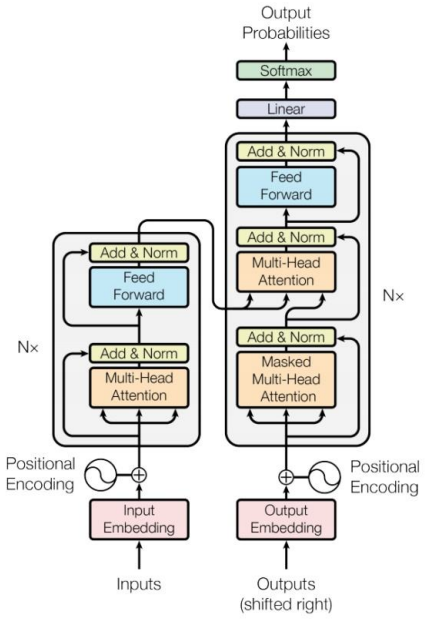

图 4-6 Transformer 架构

GPT（Generative Pre-trained Transformer）是一种基于 Transformer 架构的语言模型，通过大量的文本数据进行预训练，使模型能够理解和生成自然语言。当模型参数量扩展到一定规模后，GPT 模型就拥有了良好的上下文学习（In-context Learning）能力，这也是现代大模型技术的基础。

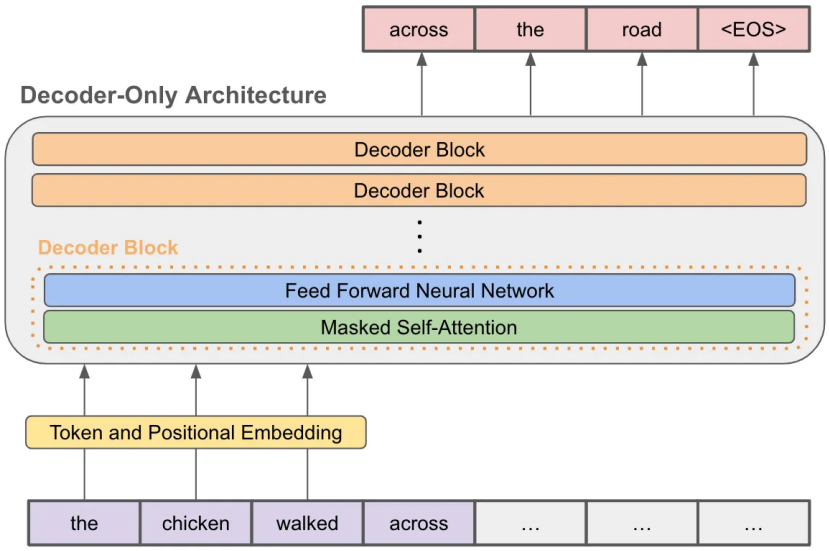

图 4-7 GPT 模型

4.2.4 语言建模任务

GPT 和 Llama 模型的预训练，是通过无监督的语言建模的任务，简而言之，通过一个句子中前面所有的 token 预测下一个 token。即，对于一个序列 $(w_1, w_2, ..., w_T)$，语言模型试图估计：

$$P(w_1, w_2, ..., w_T) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2) \cdots P(w_T|w_1, ..., w_{T-1})$$

因此，除了基本的 Transformer 架构之外，在实现中，需要注意以下两点：

（1）输入数据和标签的构建：为了实现上述语言建模任务，假设从数据集中获取到一个序列 [0, 1, 2, 3, 4]，则给模型的输入应该是 [0, 1, 2, 3]，而模型预测所对应的标签则应该是 [1, 2, 3, 4]。

（2）因果自注意力：在自注意力计算时，为了避免当前 tokens 提前看到后面的 tokens 的信息，需要对注意力计算做掩码，即使得当前 token 对后面 tokens 的

注意力分数都为 0。

4.2.5 大语言模型（Large Language Model, LLM）

大语言模型实际上就是参数和训练数据规模大得多的类 GPT 模型，在本质上是一个概率模型。尽管今天的 ChatGPT 或 Claude 表现出了惊人的推理和对话能力，但它们的核心任务从未改变：给定一段文本序列，预测下一个最可能出现的词（Token）。

GPT 的发展史就是 LLM 从量变到质变的过程：

- GPT-1 (2018)：
  - 理念：先预训练，后微调（Pre-train then Fine-tune）。先在无标注的大规模文本上学习语言知识，再在特定任务上微调。
  - 参数量：1.17 亿。
- GPT-2 (2019)：
  - 理念：Zero-shot（零样本学习）。OpenAI 发现，只要模型够大、数据够多，不需要针对特定任务微调，模型就能通过“续写”完成翻译、问答等任务。
  - 参数量：15 亿。
- GPT-3 (2020)：
  - 理念：暴力美学与 In-context Learning（上下文学习）。参数量扩大了 100 倍。模型展现出了惊人的少样本学习能力，仅需在提示词中给出一两个例子，它就能学会新任务。
  - 参数量：1750 亿。
- InstructGPT / ChatGPT (2022)：
  - 关键一跃：RLHF（基于人类反馈的强化学习）。之前的 GPT-3 虽然强大，但经常胡言乱语。RLHF 引入了人类的偏好，将模型的输出“对齐”（Alignment）到人类的价值观和指令上，使其变得有用且安全。
  - 参数量：同 GPT-3。

4.3 实验环境

环境：华为云 ModelArts 平台

环境依赖：

表 4-1 环境依赖

名称	版本
Python	3.10
MindSpore	2.4.10

数据集：金庸武侠小说全本。使用金庸武侠小说作为预训练语料库。如果时间和余额充裕，可以考虑增加语料库，以提升模型的生成效果。反之，也可以自行选定其中的一本或者几本作为预训练数据集。预训练完成之后，使用模型进行金庸风格文本续写。

4.4 实验内容

本实验基于 Modelarts 平台，使用 MindSpore 深度学习框架，搭建小型 GPT 模型，并使用《金庸武侠小说》作为语料库，在 Ascend910 加速卡上进行预训练和推理。

4.5 实验步骤

代码目录：

```
JinYong_GPT
├─ data                 // 金庸小说数据集
├─ dataset              // 存放 MindRecord 格式数据集文件
   ├─ tokenizer.json // 存放 tokenizer 的词表
   ├─ mindrecord  // MindRecord 格式数据集文件
   ├─ mindrecord.db
├─ generate.py         // 执行推理的代码
├─ train.py            // 模型训练代码
├─ src
│  ├─ dataset.py       // 生成训练数据集
│  ├─ model.py           // 模型主体代码
│  ├─ preprocess.py   // 数据预处理，生成 MindRecord 格式数据集文件
```

```
13 │  ├─ tokenizer.py   // Tokenizer 定义和训练
14 │  └─ utils.py        // 工具方法
15 ├─ requirements.txt            // Python 依赖库
```

实验步骤如下：

1、选择金庸小说作为语料库，通过”src/tokenizer” 训练和获取 Tokenizer。

2、选择金庸小说作为语料库，通过”src/pre_process.py” 预处理数据集，生成”MindRecord” 格式数据集。（前两个步骤可以在 Notebook 中调试和执行，然后将生成的 MindRecord 文件上传到 OBS 中。）

3、搭建模型主体架构。

4、通过 ModelArts 执行训练代码”train.py” 训练模型。（请注意在创建训练任务时，指定正确的 OBS 目录和训练参数。并且按照 ModelArts 要求的方式配置输入和输出路径。）

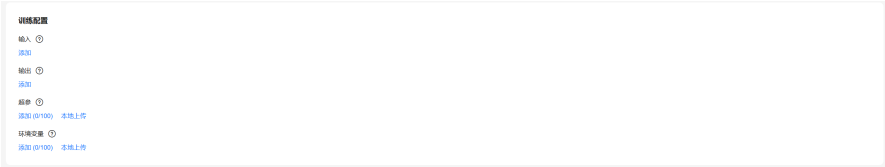

图 4-8 ModelArts 训练任务参数配置

5、执行推理代码”generate.py”，加载模型 ‘checkpoint‘，测试小说续写任务。

注意：请先阅读和理解代码，尤其是各种目录和参数设置，切勿盲目运行。数据集预处理时间开销较大，可以在本地执行之后，再把生成的 MindRecord 文件上传到云端进行训练。

注意：所有代码参数和模型超参数都是可调的，请根据自己的需要以及时间预算和云平台余额，合理调整。

注意：必须使用基本的算子和接口实现 GPT 模型，禁止直接使用 MindSpore 提供的 Attention、Transformer 或 GPT 相关模块。

4.5.1 实验运行

通过 ModelArts 平台运行实验。请注意模型训练的时间开销，避免云平台欠费。预训练未必需要执行多个 epoch ，甚至未必需要执行一个完整的 epoch ，请根据自己的时间预算调整。

4.6 评分指标

评分指标分为两个部分，其中实验代码和结果占 90%，实验报告的撰写（包含实验思考部分）占 10%。实验结果的评分指标如下：

60-80：根据提示，完成 GPT 模型构建和预训练。

80-100：在预训练后的模型上，执行推理任务，构建一个文本续写应用。

4.7 实验思考

1. 你的 GPT 模型生成效果如何？如何进一步提升模型的生成效果？

2. 当前的基于汉字字符的 tokenizer 有何优缺点？是否可以设计更好的 tokenizer 方案？

3. 你的 GPT 模型参数量有多少？如果以 FP32 精度做训练，需要占用多少显存？如果有条件请观察使用不同超参数和模型参数量时，预训练和推理效率的变化情况。

4. 模型训练和推理中的静态图模式和动态图模式分别指什么？有何区别？你使用的是哪种模式？为什么？

5. 是否可以在你的预训练模型基础上，设计一些更有趣的应用？