MODEL-ZOO

蒸馏DeepSeek-R1到自己的模型

在本博客中，我们将介绍如何使用LoRA等技术将 DeepSeek-R1 的推理能力蒸馏到较小的模型（如 Microsoft 的 Phi-3-Mini）中。

admin
Feb 5, 2025 • 9 min read

深度学习模型彻底改变了人工智能领域，但其庞大的规模和计算需求可能会成为实际应用的瓶颈。模型蒸馏是一种强大的技术，它通过将知识从大型复杂模型（教师）转移到较小、更高效的模型（学生）来解决这一挑战。

在本博客中，我们将介绍如何使用 LoRA（低秩自适应）等专门技术将 DeepSeek-R1 的推理能力蒸馏到较小的模型（如 Microsoft 的 Phi-3-Mini）中。

1、什么是蒸馏?

蒸馏是一种机器学习技术，其中较小的模型（“学生”）经过训练以模仿较大的预训练模型（“老师”）的行为。目标是保留老师的大部分表现，同时显着降低计算成本和内存占用。

这个想法最早是在 Geoffrey Hinton 关于知识蒸馏的开创性论文中提出的。它不是直接在原始数据上训练学生模型，而是从老师模型的输出或中间表示中学习。这实际上是受到人类教育的启发。

为什么它很重要：

- 成本效率：较小的模型需要更少的计算资源。
- 速度：非常适合延迟敏感的应用程序（例如 API、边缘设备）。
- 专业化：无需重新训练巨型模型即可针对特定领域定制模型。

2、蒸馏类型

模型蒸馏有几种方法，每种方法都有各自的优点：

数据蒸馏：

- 在数据蒸馏中，教师模型生成合成数据或伪标签，然后用于训练学生模型。
- 这种方法可以应用于广泛的任务，即使是那些 logits 信息量较少的任务（例如开放式推理任务）。

Logits蒸馏：

- Logits 是应用 softmax 函数之前神经网络的原始输出分数。
- 在 logits蒸馏中，学生模型经过训练以匹配教师的 logits，而不仅仅是最终预测。
- 这种方法保留了更多关于教师信心水平和决策过程的信息。

特征蒸馏：

- 特征提炼涉及将知识从教师模型的中间层转移到学生。
- 通过对齐两个模型的隐藏表示，学生可以学习更丰富、更抽象的特征。

3、Deepseek 的蒸馏模型

为了使访问更加民主化，DeepSeek AI 发布了基于 Qwen（Qwen，2024b）和 Llama（AI@Meta，2024）等流行架构的六个蒸馏变体。他们使用 DeepSeek-R1 策划的 800k 个样本直接微调开源模型。

尽管比 DeepSeek-R1 小得多，但蒸馏模型在各种基准测试中都表现出色，通常可以匹敌

尽管比 Deepseek-R1 小得多，但蒸馏模型在各种基准测试中都表现出色，通常可以匹敌甚至超越更大模型的能力。如下图所示

Model	AIME 2024		MATH-500	GPQA Diamond	LiveCode Bench	CodeForces
	pass@1	cons@64	pass@1	pass@1	pass@1	rating
GPT-4o-0513	9.3	13.4	74.6	49.9	32.9	759
Claude-3.5-Sonnet-1022	16.0	26.7	78.3	65.0	38.9	717
OpenAI-o1-mini	63.6	80.0	90.0	60.0	53.8	**1820**
QwQ-32B-Preview	50.0	60.0	90.6	54.5	41.9	1316
DeepSeek-R1-Distill-Qwen-1.5B	28.9	52.7	83.9	33.8	16.9	954
DeepSeek-R1-Distill-Qwen-7B	55.5	83.3	92.8	49.1	37.6	1189
DeepSeek-R1-Distill-Qwen-14B	69.7	80.0	93.9	59.1	53.1	1481
DeepSeek-R1-Distill-Qwen-32B	**72.6**	83.3	94.3	62.1	57.2	1691
DeepSeek-R1-Distill-Llama-8B	50.4	80.0	89.1	49.0	39.6	1205
DeepSeek-R1-Distill-Llama-70B	70.0	**86.7**	**94.5**	**65.2**	**57.5**	1633

Deepseek **提炼模型基准测试**（https://arxiv.org/html/2501.12948v1）

4、为什么要蒸馏自己的模型?

- 特定任务优化

预蒸馏模型在广泛的数据集上进行训练，以在各种任务中表现良好。然而，现实世界的应用程序通常需要专业化。

示例场景：你正在构建一个金融预测聊天机器人。在这种情况下，使用 DeepSeek-R1 为金融数据集生成推理轨迹（例如，股票价格预测、风险分析），并将这些知识蒸馏成一个已经了解金融细微差别的较小模型（例如：finance-LLM）。

- 大规模成本效率

虽然预蒸馏模型效率很高，但它们可能仍然不适合你的特定工作量。蒸馏你自己的模型可以让你针对确切的资源限制进行优化。

- 基准性能 ≠ 真实世界性能

预蒸馏模型在基准测试中表现出色，但基准测试通常不能代表真实世界的任务。因此，你通常需要一个在真实世界场景中表现比任何预蒸馏模型都更好的模型。

- 迭代改进

预蒸馏模型是静态的——它们不会随着时间的推移而改进。通过蒸馏自己的模型，你可以在新数据可用时不断完善它。

5、将 DeepSeek-R1 知识蒸馏成自定义小模型

首先安装库：

```
pip install -q torch transformers datasets accelerate bitsandbytes flash-attn --no-build-
```

5.1 生成和格式化数据集

你可以通过在你的环境中使用 ollama 或任何其他部署框架部署 deepseek-r1 来生成自定义域相关数据集。但是，对于本教程，我们将使用 Magpie-Reasoning-V2 数据集，其中包含 DeepSeek-R1 生成的 250K 思路链 (CoT) 推理样本，这些示例涵盖了数学推理、编码和一般问题解决等各种任务。

数据集结构

每个示例包括：

- 指令：任务描述（例如，“解决这个数学问题”）。
- 响应：DeepSeek-R1 的分步推理 (CoT)。

示例：

```
{
  "instruction": "Solve for x: 2x + 5 = 15",
  "response": "<think>First, subtract 5 from both sides: 2x = 10. Then, divide by 2: x =
}
```

```
```

```
from datasets import load_dataset

# Load the dataset
dataset = load_dataset("Magpie-Align/Magpie-Reasoning-V2-250K-CoT-Deepseek-R1-Llama-70B",
dataset = dataset["train"]

# Format the dataset
def format_instruction(example):
    return {
        "text": (
            "<|user|>\n"
            f"{example['instruction']}\n"
            "<|end|>\n"
            "<|assistant|>\n"
            f"{example['response']}\n"
            "<|end|>"
        )
    }

formatted_dataset = dataset.map(format_instruction, batched=False, remove_columns=subset_
formatted_dataset = formatted_dataset.train_test_split(test_size=0.1)  # 90-10 train-test
```

将数据集构造为 Phi-3 的聊天模板格式：

- `<|user|>`：标记用户查询的开始。
- `<|assistant|>`：标记模型响应的开始。
- `<|end|>`：标记回合的结束。

每个 LLM 都使用特定格式来执行指令跟踪任务。将数据集与此结构对齐可确保模型学习正确的对话模式。因此，请确保根据要提取的模型格式化数据。

5.2 加载模型和标记器

向标记器添加特殊标记 `<think>` 和 `</think>` 。

为了增强模型的推理能力，我们引入了这些标记。

- `<think>`：标记推理的开始。
- `</think>`：标记推理的结束。

这些标记帮助模型学习生成结构化的、分步的解决方案。

```
from transformers import AutoTokenizer, AutoModelForCausalLM

model_id = "microsoft/phi-3-mini-4k-instruct"
tokenizer = AutoTokenizer.from_pretrained(model_id, trust_remote_code=True)

# Add custom tokens
CUSTOM_TOKENS = ["<think>", "</think>"]
tokenizer.add_special_tokens({"additional_special_tokens": CUSTOM_TOKENS})
tokenizer.pad_token = tokenizer.eos_token

# Load model with flash attention
model = AutoModelForCausalLM.from_pretrained(
    model_id,
    trust_remote_code=True,
    device_map="auto",
    torch_dtype=torch.float16,
    attn_implementation="flash_attention_2"
)
model.resize_token_embeddings(len(tokenizer))  # Resize for custom tokens
```

5.3 配置 LoRA 以实现高效微调

LoRA 通过冻结基础模型并仅训练小型适配器层来减少内存使用量。

```
from peft import LoraConfig

peft_config = LoraConfig(
    r=8,  # Rank of the low-rank matrices
    lora_alpha=16,  # Scaling factor
    lora_dropout=0.2,  # Dropout rate
    target_modules=["q_proj", "k_proj", "v_proj", "o_proj"],  # Target attention layers
    bias="none",  # No bias terms
    task_type="CAUSAL_LM"  # Task type
)
```

5.4 设置训练参数

```
from transformers import TrainingArguments

training_args = TrainingArguments(
```

```
    output_dir="./phi-3-deepseek-finetuned",
    num_train_epochs=3,
    per_device_train_batch_size=2,
    per_device_eval_batch_size=2,
    gradient_accumulation_steps=4,
    eval_strategy="epoch",
    save_strategy="epoch",
    logging_strategy="steps",
    logging_steps=50,
    learning_rate=2e-5,
    fp16=True,
    optim="paged_adamw_32bit",
    max_grad_norm=0.3,
    warmup_ratio=0.03,
    lr_scheduler_type="cosine"
)
```

5.5 训练模型

`SFTTrainer` 简化了指令跟随模型的监督微调。 `data_collator` 批量处理示例，`peft_config` 支持基于 LoRA 的训练。

```
from trl import SFTTrainer
from transformers import DataCollatorForLanguageModeling

# Data collator
data_collator = DataCollatorForLanguageModeling(tokenizer=tokenizer, mlm=False)

# Trainer
trainer = SFTTrainer(
    model=model,
    args=training_args,
    train_dataset=formatted_dataset["train"],
    eval_dataset=formatted_dataset["test"],
    data_collator=data_collator,
    peft_config=peft_config
)

# Start training
trainer.train()
trainer.save_model("./phi-3-deepseek-finetuned")
tokenizer.save_pretrained("./phi-3-deepseek-finetuned")
```

5.6 合并保存最终模型

5.6 合并保存最终模型

训练后，必须将 LoRA 适配器与基础模型合并以进行推理。此步骤确保模型可以在没有 PEFT 的情况下独立使用。

```
final_model = trainer.model.merge_and_unload()
final_model.save_pretrained("./phi-3-deepseek-finetuned-final")
tokenizer.save_pretrained("./phi-3-deepseek-finetuned-final")
```

5.7 推理

```
from transformers import pipeline

# Load fine-tuned model
model = AutoModelForCausalLM.from_pretrained(
    "./phi-3-deepseek-finetuned-final",
    device_map="auto",
    torch_dtype=torch.float16
)

tokenizer = AutoTokenizer.from_pretrained("./phi-3-deepseek-finetuned-final")
model.resize_token_embeddings(len(tokenizer))

# Create chat pipeline
chat_pipeline = pipeline(
    "text-generation",
    model=model,
    tokenizer=tokenizer,
    device_map="auto"
)

# Generate response
prompt = """<|user|>
What's the probability of rolling a 7 with two dice?
<|end|>
<|assistant|>
"""

output = chat_pipeline(
    prompt,
    max_new_tokens=5000,
    temperature=0.7,
    do_sample=True,
    eos_token_id=tokenizer.eos_token_id
)
```

```
print(output[0]['generated_text'])
```

下面你可以看到 phi 模型在蒸馏前后的响应。

问题：用两个骰子掷出 7 的概率是多少？

- 蒸馏前的推理

响应简单明了。它直接提供了计算答案的步骤。

```
<|user|>
What's the probability of rolling a 7 with two dice?
<|end|>
<|assistant|>
 The probability of rolling a 7 with two dice can be calculated by finding the number of ways to achieve a s
um of 7 and dividing it by the total possible outcomes when rolling two dice.


Each die has 6 faces, so there are a total of 6 * 6 = 36 possible outcomes when rolling two dice. The combin
ations that give a sum of 7 are (1,6), (2,5), (3,4), (4,3), (5,2), and (6,1). There are 6 combinations that
result in a sum of 7.


The probability (P) of rolling a sum of 7 with two dice is therefore the number of successful outcomes (6) d
ivided by the total number of outcomes (36).


P(rolling a 7) = Number of successful outcomes / Total number of outcomes

P(rolling a 7) = 6 / 36

P(rolling a 7) = 1 / 6


So, the probability of rolling a 7 with two dice is 1/6 or approximately 16.67%.
```

蒸馏前的 Phi 推理

- 蒸馏后的推理

蒸馏后的响应引入了一种更详细和结构化的方法，包括一个明确的“思考”部分，概述了思维过程和推理，这对于为复杂问题生成准确的响应非常有帮助。

```
<|user|>
What's the probability of rolling a 7 with two dice?
<|end|>
<|assistant|>
 <think>
First, I need to determine the total number of possible outcomes when rolling two dice. Each die has six faces, so the total numbe
r of outcomes is 6 multiplied by 6, which equals 36.

Next, I'll identify the combinations of dice rolls that result in a sum of 7. These combinations are (1, 6), (2, 5), (3, 4), (4,
3), (5, 2), and (6, 1), totaling 6 favorable outcomes.

Finally, to find the probability, I'll divide the number of favorable outcomes by the total number of possible outcomes. So, the p
robability is 6 divided by 36, which simplifies to 1/6.
</think>

To determine the probability of rolling a sum of 7 with two dice, follow these steps:

1. **Calculate the Total Number of Possible Outcomes:**

   Each die has 6 faces, so when rolling two dice, the total number of possible outcomes is:
   \[
   6 \times 6 = 36
   \]

2. **Identify the Favorable Outcomes:**

   The combinations of dice rolls that sum to 7 are:
   \[
   (1, 6), (2, 5), (3, 4), (4, 3), (5, 2), (6, 1)
   \]
   There are a total of:
   \[
   6 \text{ favorable outcomes}
   \]

3. **Calculate the Probability:**

   The probability \( P \) of rolling a sum of 7 is the number of favorable outcomes divided by the total number of possible outco
mes:
   \[
   P = \frac{\text{Number of favorable outcomes}}{\text{Total number of possible outcomes}} = \frac{6}{36} = \frac{1}{6}
   \]

**Final Answer:**
\[
\boxed{\dfrac{1}{6}}
\]
```

蒸馏后的 Phi 推理

最后，将蒸馏后的模型权重推送到 huggingface hub（repo_id：`GPD1/DeepSeek-R1-Distill-phi-3-mini-4k-lorar8-alpha16-50000samples`）。

原文链接：How to distill Deepseek-R1: A Comprehensive Guide

汇智网翻译整理，转载请标明出处