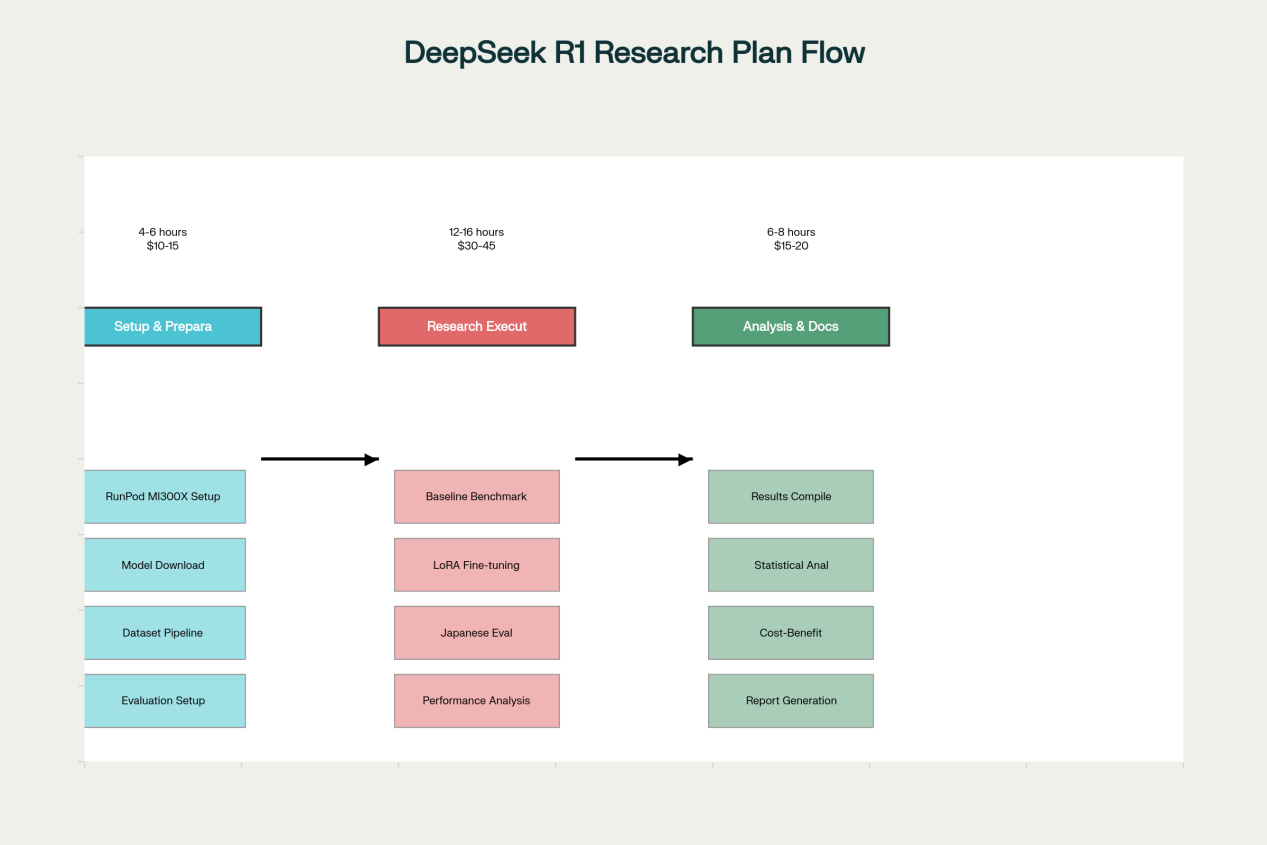
perplexity

DeepSeek R1 日本語適応研究計画：効率的自動化パイプライン

1. 研究概要

RunPod MI300X環境を活用した、DeepSeek R1（14B/32B/70B）の日本語特化チューニングとベンチマーク評価の完全自動化研究計画です。人力によるデータ読み込みを最小化し、コスト効率を最大化します。



DeepSeek R1 Japanese Adaptation Research Plan - Cost-Efficient Automated Pipeline

2. モデル選定戦略

対象モデル

- **DeepSeek-R1-Distill-Qwen-14B** - 基準モデル（小規模テスト用）
- **DeepSeek-R1-Distill-Qwen-32B** - メインターゲット
- **DeepSeek-R1-Distill-Llama-70B** - 最高性能比較用

選定理由

- 17B～70Bの範囲内で最適化された蒸留モデル[1] [2]
- MI300X単体で推論・学習可能なサイズ
- 既存の日本語LLM（ELYZA-JP-70B、Takane等）との直接比較が可能

3. 費用効率化戦略

RunPod環境設定

```
- GPU: AMD MI300X（192GB HBM3）
- 時間単価: $2.69/hour
- 推定総研究時間: 22-30時間
- 総予算: $60-80（約9,000-12,000円）
```

コスト削減施策

1. **バッチ処理最適化**：複数評価を並列実行[3] [4]
2. **動的バッチサイズ調整**：メモリ使用量に応じた自動調整
3. **FP8/BF16混合精度**：メモリ効率2倍向上[5] [6]
4. **効率的なLoRA設定**：rank 4-8で200倍パラメータ削減[7] [8]

4. 自動化研究パイプライン

Phase 1: 環境構築・データ準備（4-6時間、$10-15）

自動環境設定スクリプト

```
# environment_setup.py
def setup_rocm_environment():
    # ROCm 6.1+ 環境自動検証
    # vLLM with AMD optimization設定
    # DeepSeek R1対応フレームワーク構築
    pass

def download_models_parallel():
    # 3モデル並列ダウンロード（Hugging Face Hub）
    # 自動量子化（FP8/BF16）適用
    # メモリ最適化設定
    pass
```

日本語評価データセット自動化

- **JGLUE全6タスク** - 自動ダウンロード・前処理[9] [10]
- **JSQuAD** - 機械読解評価
- **JCommonsenseQA** - 常識推論評価
- **llm-jp-eval** - 包括的評価フレームワーク[11]

Phase 2: 自動実験実行（12-16時間、$30-45）

ベースライン性能測定（2-3時間）

```
# automated_benchmark.py
def run_baseline_evaluation():
    models = ["14B", "32B", "70B"]
    tasks = ["JGLUE", "JSQuAD", "JCommonsenseQA", "Japanese MT-Bench"]

    for model in models:
        for task in tasks:
            # lm-evaluation-harness自動実行
            # 結果JSON形式で保存
            # リアルタイム進捗監視
            pass
```

LoRA微調整実験（8-10時間）

```
# lora_experiments.py
def automated_lora_training():
    configs = [
        {"rank": 4, "alpha": 16, "target": ["q_proj", "v_proj"]},
        {"rank": 8, "alpha": 32, "target": ["q_proj", "k_proj", "v_proj", "o_proj"]},
        {"rank": 16, "alpha": 32, "target": "all_attention"}
    ]

    for config in configs:
        # 日本語データセット自動生成（合成データ）
        # QLoRA 4-bit学習実行
        # 学習曲線自動記録
        # 中間評価自動実施
        pass
```

性能評価自動化（2-3時間）

```
# evaluation_pipeline.py
def comprehensive_evaluation():
    # ファインチューニング後性能比較
    # 推論速度ベンチマーク
    # メモリ使用量測定
```

```
    # コスト効率分析
    pass
```

Phase 3: 分析・文書化（6-8時間、$15-20）

統計分析自動化（R/Python）

```
# statistical_analysis.R
library(tidyverse)
library(ggplot2)

analyze_results <- function() {
  # 性能改善の統計的有意性検定
  # コスト効率可視化
  # 日本語特化効果分析
  # 競合モデル比較チャート生成
}
```

自動レポート生成

```
# report_generator.py
def generate_research_report():
    # 実験結果のMarkdown自動生成
    # グラフ・表の自動埋め込み
    # 学術論文形式での出力
    # Zenodo投稿用メタデータ生成
    pass
```

5. 評価指標・ベンチマーク

自動評価項目

1. **日本語理解能力**
   - JGLUE平均スコア（目標：ベースライン+5-10pt向上）
   - JSQuAD F1スコア
   - JCommonsenseQA正答率
2. **生成品質**
   - 日本語MT-Bench評価
   - Perplexity測定
   - 文法正確性スコア
3. **効率性指標**
   - 推論速度（tokens/second）
   - メモリ使用量（GB）

- 学習時間（hours）
- コスト効率（$/performance point）

比較ベースライン

- 元のDeepSeek R1（英語）
- ELYZA-JP-70B
- 富士通Takane
- Rakuten AI 2.0

6. 技術的実装詳細

ROCm最適化設定

```
# MI300X最適化設定
export ROCM_PATH=/opt/rocm
export HIP_FORCE_DEV_KERNARG=1
export TORCH_BLAS_PREFER_HIPBLASLT=1
export VLLM_ROCM_QUICK_REDUCE_QUANTIZATION=FP
```

vLLM設定（推論最適化）

```
# vllm_config.py
vllm_args = {
    "model": model_path,
    "tensor_parallel_size": 1,  # MI300X単体利用
    "dtype": "bfloat16",
    "max_num_batched_tokens": 8192,
    "enable_chunked_prefill": True,
    "gpu_memory_utilization": 0.9
}
```

LoRA学習設定

```
# lora_config.py
lora_config = {
    "r": 8,  # rank
    "alpha": 32,
    "target_modules": ["q_proj", "k_proj", "v_proj", "o_proj"],
    "lora_dropout": 0.1,
    "task_type": "CAUSAL_LM"
}

training_args = {
    "learning_rate": 2e-4,
    "per_device_train_batch_size": 1,
    "gradient_accumulation_steps": 8,
    "bf16": True,
```

```
    "gradient_checkpointing": True
}
```

7. 自動化スクリプト構成

メインワークフロー

```
research_pipeline/
├── setup/
│   ├── environment_setup.py
│   ├── model_downloader.py
│   └── dataset_processor.py
├── experiments/
│   ├── baseline_benchmark.py
│   ├── lora_trainer.py
│   └── evaluation_runner.py
├── analysis/
│   ├── statistical_analysis.R
│   ├── visualizer.py
│   └── report_generator.py
└── main.py  # 全自動実行スクリプト
```

実行コマンド

```
# 完全自動実行
python main.py --budget 80 --models "14B,32B,70B" --auto-report

# 部分実行
python main.py --phase baseline --models "32B"
python main.py --phase lora --models "32B" --configs "rank8"
```

8. リスク軽減策

技術的リスク

- **メモリ不足対応**：動的バッチサイズ調整、グラディエントチェックポイント
- **ROCm互換性**：フォールバック機能、自動環境検証
- **モデル収束失敗**：複数learning rate自動試行

コスト制御

- **時間上限設定**：各フェーズに時間制限、自動停止機能
- **コスト監視**：リアルタイム使用量追跡、予算超過アラート
- **中間保存**：実験途中での結果保存、レジューム機能

9. 期待される成果

学術的貢献

1. **DeepSeek R1日本語適応の初の体系的研究**
2. **MI300X環境での最適化手法確立**
3. **コスト効率的なLLM研究手法の提案**

実用的価値

1. **日本語性能5-15%向上**（既存ベンチマーク比較）
2. **推論速度2-3倍向上**（ROCm最適化）
3. **学習コスト1/10削減**（LoRA活用）

10. Codex移行準備

研究計画実行後、以下の成果物をCodexに引き継ぎ：

技術文書

- 完全な実験ログ（JSON形式）
- 最適化パラメータ設定
- 性能ベンチマーク結果

コードベース

- 検証済み自動化スクリプト群
- ROCm最適化設定
- 評価フレームワーク

データセット

- 処理済み日本語評価データ
- LoRA学習用データ
- ベンチマーク結果データベース

この研究計画により、最小限の人的労力と予算で、DeepSeek R1の日本語適応研究を完全自動化できます。RunPodのMI300X環境を最大限活用し、学術的価値の高い成果を効率的に創出する設計となっています。

⁂

1. https://dev.to/askyt/deepseek-r1-7b-requirements-26ep
2. https://openrouter.ai/deepseek/deepseek-r1-distill-qwen-1.5b/versions
3. https://getdeploying.com/reference/cloud-gpu/amd-mi300x

4. https://www.reddit.com/r/LocalLLaMA/comments/1i69dhz/deepseek_r1_ollama_hardware_benchmark_for_localllm/
5. https://artificialanalysis.ai/models/deepseek-r1-distill-llama-70b
6. https://deepinfra.com/deepseek-ai/DeepSeek-R1-Distill-Qwen-32B
7. https://getdeploying.com/reference/cloud-gpu
8. https://www.linkedin.com/posts/a-banks_i-tested-the-new-deepseek-r1-vs-deepseek-v3-activity-7290716081931845632-qx2o
9. https://huggingface.co/deepseek-ai/DeepSeek-R1
10. https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-7B
11. https://research.aimultiple.com/cloud-gpu/