Finetuning DeepSeek R1 for a medical use-case

El Arrouf Karim
Data Science, INSEA,Rabat
kelarrouf@insea.ac.ma

ABSTRACT

Large Language Models (LLMs) have demonstrated significant potential across various fields, including medicine, by facilitating clinical reasoning, diagnosis, and medical knowledge retrieval. However, general-purpose LLMs often lack the specialized understanding required to handle complex medical tasks. This study aims to fine-tune DeepSeek R1, an advanced open-weight language model, to enhance its capabilities in medical reasoning and clinical decision-making. To achieve this, we leverage a dataset incorporating Chain-of-Thought (CoT) reasoning, an approach previously used to refine HuatuoGPT-01, a state-of-the-art medical model.We apply advanced optimization techniques, including LoRA (Low-Rank Adaptation) and 4-bit quantization, to reduce memory requirements and improve fine-tuning efficiency without compromising model performance. The training process is conducted using Supervised Fine-Tuning (SFT) with Hugging Face Transformers and TRL (Transformers Reinforcement Learning), employing optimization strategies such as gradient accumulation, dynamic learning rate scheduling, and mixed precision training (FP16/BF16). The fine-tuned model is evaluated on multiple clinical question-answering and medical reasoning tasks, and its performance is benchmarked against state-of-the-art models.The results demonstrate a significant improvement in contextual understanding, diagnostic accuracy, and inferential capabilities, making this model particularly suitable for AI-assisted medical support. This work contributes to advancements in AI-driven healthcare, paving the way for more reliable and interpretable medical language models.

Keywords: Large Language Models (LLMs), DeepSeek R1, Medical AI, Fine-tuning, Chain-of-Thought (CoT) Reasoning, Supervised Fine-Tuning (SFT), LoRA (Low-Rank Adaptation), 4-bit Quantization.

IV. INTRODUCTION

L'intelligence artificielle (IA) révolutionne progressivement le domaine de la médecine en offrant des outils avancés pour l'analyse clinique, le diagnostic et la prise de décision médicale. Parmi ces avancées, les grands modèles de langage (LLMs) ont démontré une capacité exceptionnelle à traiter et à structurer des informations complexes. Cependant, la plupart des LLMs généralistes souffrent d'un manque de compréhension fine des concepts médicaux spécifiques, ce qui limite leur application dans des contextes critiques nécessitant une haute précision et une justification des réponses.

Dans cette étude, nous explorons le fine-tuning de DeepSeek R1, un modèle de langage puissant et open-weight, afin de l'adapter à un chatbot médical conçu pour assister les professionnels de santé dans leurs décisions cliniques et fournir des réponses précises basées sur des connaissances médicales validées. L'objectif est d'améliorer sa capacité à raisonner en médecine en utilisant un dataset intégrant la technique du Chain-of-Thought (CoT) reasoning, déjà employée dans l'entraînement de modèles spécialisés comme HuatuoGPT-01.

Pour garantir une optimisation efficace du fine-tuning, nous adoptons des techniques avancées de réduction de la consommation mémoire et de stabilisation de l'apprentissage, notamment :

• LoRA (Low-Rank Adaptation) : Réduction du nombre de paramètres à entraîner pour minimiser la charge mémoire.

• Quantification en 4-bit : Compression du modèle tout en maintenant ses performances.

• Gradient Accumulation et Precision Mixte (FP16/BF16) : Gestion optimisée des ressources pour un entraînement plus rapide et moins coûteux.

• Optimiseur AdamW 8-bit : Réduction de la charge mémoire tout en conservant la précision du modèle.

L'entraînement est réalisé à l'aide des bibliothèques Hugging Face Transformers et TRL (Transformers Reinforcement Learning), en intégrant une stratégie de Supervised Fine-Tuning (SFT) pour affiner le modèle sur des tâches spécifiques de question-réponse clinique et d'analyse diagnostique.

L'objectif final de ce projet est de déployer un chatbot médical basé sur DeepSeek R1, capable d'assister les professionnels de santé en fournissant des recommandations, des analyses de cas cliniques et des réponses médicales fiables et concises.

IV. Fine-Tuning de DeepSeek R1 sur des Ressources Disponibles

A. Quantification du modèle

Les modèles de langage de grande taille (LLMs) open-source sont généralement publiés en précision flottante 32 bits (FP32), ce qui impose des exigences considérables en termes de mémoire et de calcul. Par exemple, un modèle de 7 milliards de paramètres en FP32 nécessite environ 28 Go de mémoire, rendant son fine-tuning sur des ressources limitées particulièrement coûteux.
La quantification est une technique qui réduit la précision des poids du modèle pour minimiser l'utilisation de mémoire et accélérer l'inférence, tout en maintenant des performances compétitives. Elle est particulièrement utile pour le déploiement sur des environnements contraints en ressources et l'optimisation des coûts énergétiques.

Nous avons exploré plusieurs techniques de quantification :

• FP16 (Floating Point 16) : réduit la précision en utilisant 5 bits pour l'exposant et 10 bits pour la mantisse, ce qui permet d'augmenter la taille du batch et de réduire la consommation mémoire.

• BF16 (Brain Floating Point 16) : maintient la même plage de valeurs que FP32 tout en réduisant la précision à 7 bits, garantissant une meilleure stabilité numérique.

• INT8 (8-bit Quantization) : convertit les poids du modèle en entiers, ce qui réduit la mémoire utilisée et les coûts d'inférence, tout en limitant légèrement la précision.

• 4-bit Quantization (QLoRA) : réduit davantage la taille du modèle en utilisant 4 bits de précision, ce qui permet d'effectuer le fine-tuning sur des GPUs à mémoire limitée, tout en minimisant la perte de performance.

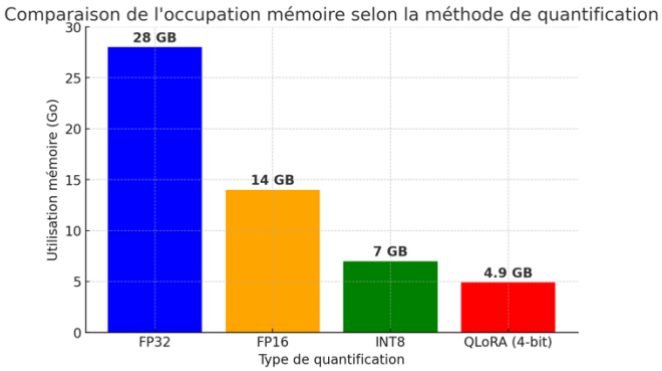

Figure 1: Comparaison de l'occupation de mémoire selon la méthode de quantification.

Dans notre étude, nous avons opté pour QLoRA, qui combine quantification en 4-bit et fine-tuning LoRA, permettant une réduction mémoire de 71% tout en maintenant des performances comparables à celles d'un fine-tuning classique.

B. Accumulation de Gradients

L'entraînement standard des LLMs met à jour les poids après chaque mini-batch. Cependant, cette approche devient impraticable lorsque la mémoire GPU est insuffisante pour stocker un batch de grande taille.
La solution à ce problème est l'accumulation de gradients (Gradient Accumulation), qui permet de simuler un batch plus grand en plusieurs étapes sans augmenter l'utilisation mémoire.

• Au lieu d'effectuer une mise à jour des poids après chaque mini-batch, les gradients sont stockés temporairement.

• Une mise à jour unique des poids est appliquée une fois que plusieurs mini-batches ont été traités.

• Cela permet d'améliorer la stabilité de l'apprentissage et de réduire les besoins en mémoire GPU.

Dans notre configuration, nous avons défini : *gradient_accumulation_steps=4* , ce qui permet de simuler un batch de 8 exemples avec une mémoire équivalente à celle d'un batch de 2 exemples.

C. Fine-tuning Efficace des Paramètres (PEFT – Parameter Efficient Fine-tuning)

Une mise à jour unique des poids est appliquée une fois que plusieurs mini-batches ont été traités.

Le fine-tuning intégral d'un LLM est coûteux en mémoire et en puissance de calcul. Pour rendre l'entraînement plus efficace, nous avons utilisé Parameter-Efficient Fine-Tuning (PEFT), qui consiste à modifier uniquement une petite partie des poids du modèle, réduisant ainsi la charge mémoire et les coûts d'entraînement.
Nous avons exploré plusieurs méthodes de PEFT :

- **LoRA (Low-Rank Adaptation)**

  • Ajoute de petites matrices entraînables à certaines couches clés du modèle au lieu de modifier tous les poids.

  • Diminue le nombre de paramètres mis à jour, ce qui réduit la mémoire GPU consommée.

  • Permet un fine-tuning plus rapide sans perte de performance significative.

- **QLoRA (Quantized LoRA)**

  • Combine LoRA avec une quantification 4-bit, permettant de fine-tuner un modèle 7B sur un GPU 24GB.

  • Réduit l'empreinte mémoire sans dégrader la qualité des prédictions.

```python
# Apply LoRA fine-tuning to the model

model_lora=FastLanguageModel.get_peft_model(
    model,
    r= 16, # Determines the size of the trainable adapters
    target_modules=["q_proj",
                    "k_proj",
                    "v_proj",
                    "o_proj",
                    "gate_proj",
                    "up_proj",
                    "down_proj",
    ], #LoRA will be applied there in order to have a balance between flexibility and performance.
    lora_alpha=16,
    lora_dropout=0, # Full retention of info
    bias="none",
    use_gradient_checkpointing="unsloth", # Saving memory
    random_state=3407,
    use_rslora=False,
    loftq_config= None  # Disabled cause we have 4-bit quantization
)
```

Figure 2: Code Python pour l'application de LoRA fine-tuning

IV. Workflow du Fine-Tuning de DeepSeek R1

A. Présentation du Workflow

Le fine-tuning d'un LLM est un processus structuré en plusieurs étapes essentielles qui garantissent une adaptation efficace du modèle à une tâche spécifique.
Dans le cas de DeepSeek R1, notre objectif est d'affiner ses capacités en raisonnement médical afin de l'intégrer dans un chatbot d'assistance clinique.
Le workflow du fine-tuning suit une séquence logique d'opérations, incluant :

1. Prétraitement des données médicales

2. Tokenization et structuration des données

3. Application de LoRA et quantification 4-bit (QLoRA)

4. Entraînement du modèle avec SFT (Supervised Fine-Tuning)

5. Évaluation et validation des performances

6. Déploiement et intégration du modèle dans un chatbot médical.

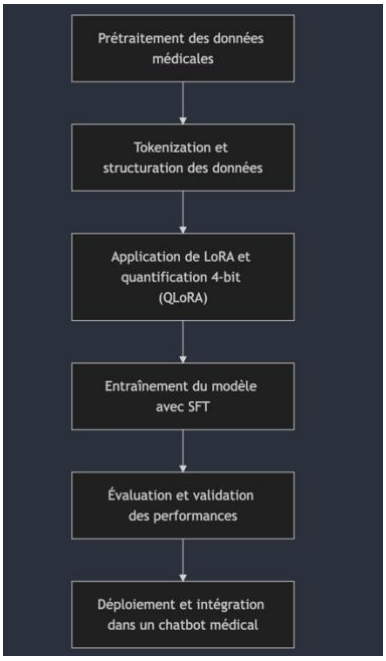

Figure 3: Fine-Tuning Workflow

B. Structure de données

Nous avons utilisé la dataset : "FreedomIntelligence/medical-o1-reasoning-SFT", disponible sur Hugging Face, qui a été conçue pour le fine-tuning des modèles de langage sur des tâches médicales complexes. Cette dataset est spécialisée dans le raisonnement médical avancé, ce qui est essentiel pour affiner DeepSeek R1 en tant que chatbot d'assistance clinique.

La dataset est structurée en trois éléments principaux :

1. Question (Problème Médical) : l'entrée du modèle, posée sous forme de question clinique.

2. Complex_CoT (Raisonnement étape par étape) : Décomposition du raisonnement clinique, permettant au modèle d'apprendre à structurer sa réponse.

3. Réponse Finale (Solution Médicale) → La réponse finale, basée sur le raisonnement médical.

Figure 4: Structure la dataset FreedomIntelligence/medical-o1-reasoning-SFT

C. Pré-traitement et Tokenization de la Dataset.

La dataset a été téléchargé via Hugging Face comme suit :

```
# Download the Dataset using HF
dataset = load_dataset("FreedomIntelligence/medical-o1-reasoning-SFT",
                       "en",
                       split="train[0:500]",
                       trust_remote_code=True)
dataset
```

Figure 5: Chargement de la Dataset

Une fois la dataset chargée, nous avons effectué les étapes de tokenization et structuration. Ces étapes incluent le padding et truncation pour uniformiser les séquences, l'encodage des données en tokens utilisables par le modèle et finalement l'ajout du Cot pour entrainer le modèle à raisonner avant de répondre.

D. Application de LoRA et QLoRA

Nous avons appliqué LoRA et QLoRA afin de rendre l'entraînement plus efficace et compatible avec un GPU à mémoire limitée.En ce qui concerne ,LoRA ,nous avons défini :

• r = 16 : La taille des matrices de projection LoRA, ce qui permet un compromis entre flexibilité et efficacité.

• Couches ciblées : q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj. Cela couvre les principales couches d'attention du modèle, maximisant l'impact du fine-tuning sans nécessiter de modification complète des poids du modèle.

• Lora_alpha = 16 : Un paramètre d'échelle qui ajuste la mise à jour des matrices LoRA.

• lora_dropout = 0 : Aucune perte d'information due au dropout, garantissant une rétention complète des informations durant l'entraînement.

LoRA permet de geler les poids principaux du modèle, ce qui accélère l'entraînement et évite un catastrophic forgetting du modèle pré-entraîné.

Ensuite, nous avons complété LoRA avec une quantification 4-bit (QLoRA) pour réduire encore plus l'utilisation mémoire. Cela nous a permis de fine-tuner DeepSeek R1 sur un GPU 24GB, alors qu'un entraînement standard aurait nécessité plus de 28GB.

Nous avons utilisé :

• Quantification en 4-bit : Réduction des poids du modèle pour minimiser la charge mémoire.

• loftq_config = None : Nous avons désactivé LoftQ, car la quantification 4-bit était suffisante pour nos ressources.

• use_gradient_checkpointing="unsloth" : Activation du gradient checkpointing, qui permet d'économiser la mémoire en ne stockant pas les activations intermédiaires lors de la propagation avant.

Cette approche permet d'obtenir des performances similaires à un fine-tuning complet tout en réduisant drastiquement le coût du calcul.

L'implémentation de LoRA et QLoRA a permis d'optimiser notre fine-tuning de DeepSeek R1, en réduisant considérablement l'empreinte mémoire tout en conservant une qualité de prédiction médicale élevée. Cette approche s'avère particulièrement efficace pour adapter un LLM à des tâches spécialisées sans infrastructure coûteuse.

E. Entrainement du modèle

L'entraînement supervisé (Supervised Fine-Tuning, SFT) permet d'adapter un modèle de langage pré-entraîné (LLM) à une tâche spécifique en l'exposant à un ensemble de paires question-réponse. Dans notre cas, nous avons affiné DeepSeek R1 sur des données médicales, en intégrant une approche de raisonnement Chain-of-Thought (CoT) pour améliorer la cohérence des réponses.
Afin d'assurer un suivi précis de l'entraînement, nous avons utilisé Weights & Biases (W&B), une plateforme permettant de visualiser les métriques d'entraînement, optimiser les hyperparamètres et comparer les performances du modèle au fil des itérations.

Nous avons utilisé les hyperparamètres suivants pour garantir un bon équilibre entre performance et efficacité computationnelle :

TABLE 3: HYPERPARAMÈTRES D'ENTRAÎNEMENT

Hyperparamètre	**Valeur**	**Justification**
Batch size par GPU	**2**	Limiter l'utilisation mémoire
Gradient Accumulation Steps	**4**	Simule un batch de 8 tout en limitant la mémoire
Nombre d'époques	**1**	Évaluation rapide des performances
Warmup Steps	**5**	Stabiliser l'entraînement initial
Max Steps	**60**	Limiter le temps de calcul
Learning Rate	**2e-4**	Compromis entre vitesse et stabilité
Optimiseur	**AdamW 8-bit**	Réduction mémoire supplémentaire
Weight Decay	**0.01**	Prévention du surajustement
Scheduler	**Linear**	Décroissance progressive du taux d'apprentissage
Seed	**3407**	Reproductibilité des résultats
Sortie des résultats	**outputs**	Organisation des modèles entraînés

L'entraînement a été réalisé via SFTTrainer, une classe optimisée pour les LLMs. Weights & Biases (W&B) a été utilisé pour enregistrer et visualiser les métriques de l'entraînement.

```python
# Initialise the trainer
trainer=SFTTrainer(
    model=model_lora,
    train_dataset=dataset_finetune,
    dataset_text_field="text",
    max_seq_length=max_seq_length,
    dataset_num_proc=2,  # accelerate the loading and the preprocessing of data.
    tokenizer=tokenizer,

    #Define the training hyperparameters (arguments)
    args=TrainingArguments(
        per_device_train_batch_size=2,
        gradient_accumulation_steps=4,
        num_train_epochs=1,
        warmup_steps=5,
        max_steps=60,
        learning_rate=2e-4,
        fp16=not is_bfloat16_supported(),
        bf16=is_bfloat16_supported(),
        logging_steps=10,
        optim="adamw_8bit",
        weight_decay=0.01,# small regularization to avoid overfitting
        lr_scheduler_type = "linear",
        seed=3407,
        output_dir= "outputs",
    )
)
```

Figure 6: Initialisation du trainer avec SFTTrainer

Les résultats de l'entraînement ont été suivis en temps réel via W&B (Weights and Biases), nous permettant d'analyser l'évolution des pertes d'entraînement, le taux d'apprentissage et d'autres métriques clés.
Ils indiquent une convergence stable du modèle au fil des itérations. La perte d'entraînement (train/loss) diminue rapidement lors des premières étapes avant de se stabiliser autour de 1.3, suggérant une apprentissage efficace du modèle sans signe de divergence. De plus, la norme des gradients (train/grad_norm) montre une réduction progressive des fluctuations, ce qui confirme que l'optimisation s'effectue de manière contrôlée et que les mises à jour des poids restent cohérentes.
L'utilisation conjointe de QLoRA et de l'optimiseur AdamW en 8-bit a permis de réduire considérablement la mémoire requise, rendant ainsi possible l'entraînement sur une seule carte GPU avec des ressources limitées. Enfin, l'absence de variations soudaines ou de ré-augmentation du loss après 60 étapes indique que le modèle est bien régularisé, suggérant une bonne généralisation aux nouvelles données et un risque limité de surajustement. Ces résultats démontrent l'efficacité du fine-tuning optimisé en mémoire, permettant d'adapter un modèle de langage médical tout en maintenant un coût computationnel réduit.

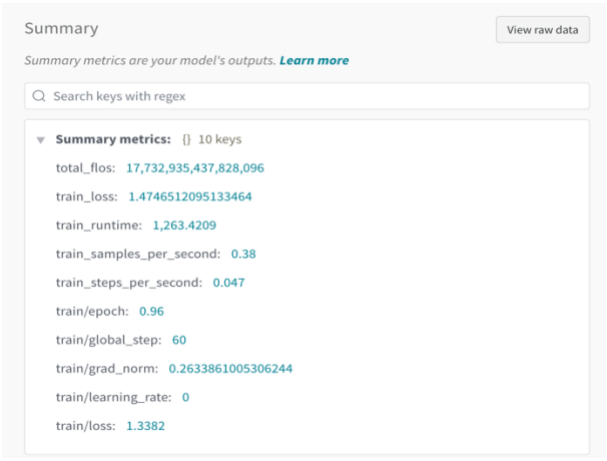

Figure 7: Résumé de la performance du modèle

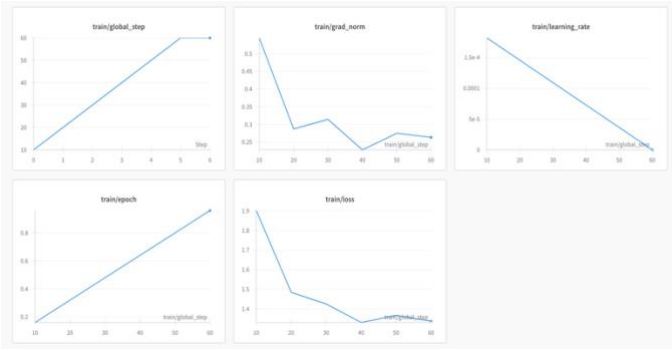

Figure 8: Evolution des metrics d'entrainement

IV. Conclusion et perspectives

Dans cette étude, nous avons exploré le fine-tuning du modèle DeepSeek R1 pour des applications médicales en utilisant une approche optimisée en mémoire basée sur QLoRA et LoRA. Notre méthodologie a permis de réduire significativement la consommation mémoire, rendant le fine-tuning possible sur des ressources GPU limitées tout en maintenant une bonne généralisation du modèle. L'analyse des résultats d'entraînement montre une convergence stable, une réduction progressive de la perte et une stabilisation des gradients, suggérant que l'optimisation choisie permet une adaptation efficace du modèle aux questions médicales complexes.

L'amélioration du modèle passe d'abord par l'extension du volume de données d'entraînement, en intégrant un plus grand nombre de cas cliniques diversifiés et en explorant des données multilingues pour élargir son applicabilité. Ensuite, l'optimisation des techniques de fine-tuning pourrait inclure d'autres approches PEFT, telles que Prefix-Tuning ou IA3, ainsi que l'utilisation de schedulers avancés comme Cosine Annealing pour stabiliser l'apprentissage.
Une validation clinique rigoureuse est également essentielle. Comparer les performances du modèle avec des LLMs spécialisés en médecine et organiser une évaluation par des experts médicaux permettra d'assurer la fiabilité et la pertinence des réponses générées.

Enfin, l'optimisation de l'inférence et du déploiement facilitera l'intégration du modèle dans un chatbot médical interactif. L'amélioration des temps de réponse via DeepSpeed ou ONNX Runtime réduira la latence et favorisera une adoption plus large en milieu clinique. Ces avancées contribueront à faire de DeepSeek R1 un outil fiable et efficace pour l'assistance médicale.

V. REFERENCES

[1] DeepSeek-AI, Daya Guo,DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning

[2] https://dassum.medium.com/fine-tune-large-language-model-llm-on-a-custom-dataset-with-qlora-fb60abdeba07

[3] Mathav Raj J, Kushala VM,Harikrishna Warrier, Yogesh Gupta, FINE TUNING LLMS FOR ENTERPRISE: PRACTICAL GUIDELINES AND RECOMMENDATIONS

[4] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, Luke Zettlemoyer, QLoRA: Efficient Finetuning of Quantized LLMs