REPORT : LLAMA2 IMPLEMENTATION

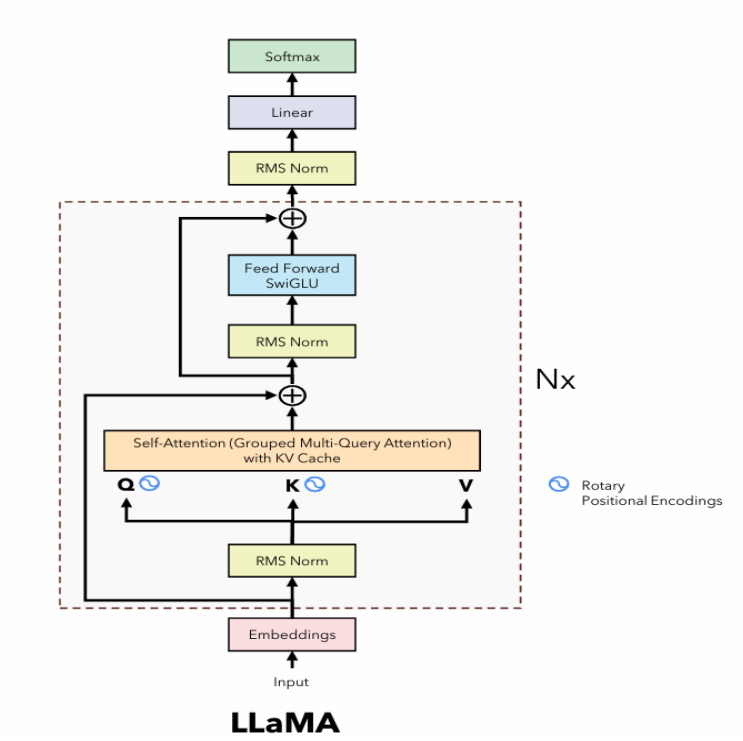
Group 9

Members: Nguyễn Phú Lộc - 22022547

Nguyễn Gia Lộc - 22022554

Nguyễn Quang Huy – 22022582

1. Overall Architecture



LLaMA 2 là một mô hình **Transformer decoder-only**, được tối ưu hóa để tạo văn bản và xử lý ngôn ngữ tự nhiên. Dựa trên sơ đồ, kiến trúc tổng thể của LLaMA 2 bao gồm các thành phần chính sau:

1. **Input & Embeddings**
2. **RMSNorm**
3. **Self-Attention (Grouped Multi-Query Attention) với KV Cache**
4. **Residual Connection**
5. **Feed Forward Network (SwiGLU Activation)**
6. **Residual Connection**
7. **Output (Linear + Softmax)**

Về cơ bản, kiến trúc Llama 2 giống với Llama, tuy nhiên Llama 2 có một số cải tiến quan trọng như sau:

- **Grouped-Query Attention (GQA)** giúp LLaMA 2 tăng tốc độ suy luận mà vẫn duy trì chất lượng đầu ra tốt hơn so với LLaMA 1.
- **Double KV cache** giúp ổn định quá trình huấn luyện và giảm lỗi gradient exploding khi đào tạo mô hình lớn.
- **Mở rộng kích thước FFN** giúp mô hình có khả năng biểu diễn mạnh hơn, đặc biệt với các tác vụ phức tạp.

2. Implementation

2.1 RMSNorm

RMSNorm (Root Mean Square Normalization) là một phương pháp chuẩn hóa chỉ sử dụng giá trị trung bình bình phương (RMS) thay thế cho LayerNorm nhưng không phụ thuộc vào thống kê của batch. Nó được sử dụng chủ yếu trong mô hình Transformer để ổn định quá trình huấn luyện và tăng tốc độ hội tụ.

$$\text{RMS}(x) = \sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2 + \epsilon} \qquad \hat{x} = \frac{x}{\text{RMS}(x)}$$

2.2 Attention

Attention là cơ chế giúp mô hình tập trung vào các phần quan trọng của dữ liệu đầu vào. Nó gán trọng số cao hơn cho thông tin liên quan và giảm trọng số của phần ít quan trọng, giúpmô hình đưa ra dự đoán chính xác hơn.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})$$

2.3. Llama forward function

1. **Chuẩn hóa đầu vào bằng RMSNorm**
2. **Tính Attention**
3. **Residual Connection**
4. **Chuẩn hóa lần hai bằng RMSNorm**
5. **Áp dụng Feed-Forward Network (FFN)**.
6. **Residual Connection lần hai**

```
~/Documents/Workspace/Llama2-Implementation | main !1 ?4
python sanity_check.py
  checkpoint_dict = torch.load(checkpoint, map_location=device)
Your Llama implementation is correct!
```

2.4 Generate function

Hàm `generate` tự động tạo văn bản bằng cách dự đoán và nối từ tiếp theo vào chuỗi đầu vào.

- Nếu quá dài, cắt đầu vào để không vượt `max_seq_len`.
- Dự đoán logits và lấy từ cuối cùng.
- Nếu `temperature = 0.0`: chọn từ có xác suất cao nhất (greedy).
- Nếu `temperature > 0.0`: áp dụng sampling để tăng tính ngẫu nhiên.
- Nối từ mới vào đầu ra và lặp lại cho đến khi đạt `max_new_tokens`.

2.5 RoPE - Rotary Positional Embeddings

Absolute Positional Embeddings

- Chỉ mã hóa vị trí tuyệt đối, không nắm bắt quan hệ giữa các token.
- Khó mở rộng vượt quá độ dài tối đa.

- Embedding hình sin ngoại suy kém, thiếu giải thích hình học rõ ràng.

Rotary Positional Embeddings

- Sử dụng **complex plane** để nhúng thông tin vị trí tương đối trực tiếp vào quá trình truyền tải.
- Không cộng thông tin vị trí tuyệt đối như Transformer truyền thống.
- Xoay vector embedding theo góc dựa trên vị trí từ, giúp mô hình nhận diện quan hệ giữa các từ dễ dàng hơn.

```
~/Documents/Workspace/Llama2-Implementation | main ?4
python rope_test.py
  ref_query_rope_embedding, ref_key_rope_embedding = torch.load("./rotary_embedding_actual.data")
Rotary embedding test passed!
```

2.6 AdamW Optimizer

AdamW là phiên bản cải tiến của **Adam**, tách **Weight Decay** thành một bước cập nhật riêng thay vì kết hợp vào **gradient**. Cách tiếp cận này giúp giảm ảnh hưởng tiêu cực của **weight decay** lên động lực học của **Adam**, cải thiện tính tổng quát của mô hình**.**

Adam Optimizer	AdamW Optimizer
Weight Decay được gộp vào gradient update.	**Weight Decay được tách biệt thành một bước riêng** $\theta_{t+1} = \theta_{t+1} - \eta\lambda\theta_t$

```
~/Documents/Workspace/Llama2-Implementation
python optimizer_test.py
Optimizer test passed!
```

2.7 LlamaEmbeddingClassifier

- **Mã hóa câu (Sentence Encoding):** Mô hình LLaMA2 được sử dụng để trích xuất biểu diễn ẩn (**hidden representation**) của token cuối cùng trong câu đầu vào.
- **Phân loại câu (Sentence Classification):**
  - Áp dụng **dropout** để giảm thiểu hiện tượng **overfitting**.
  - Biểu diễn ẩn được đưa qua một **linear layer** để tạo **logits**.
  - Sử dụng **log-softmax** để chuẩn hóa logits thành phân phối xác suất trên các lớp đầu ra.

3. Finetuning + Inference

Sử dụng pretrained Llama 2 để thực hiện một số tác vụ như: text generation, sentence classfication trên hai tập dữ liệu SST-5 và CFIMBD

3.1. Text Continuation

python run_llama.py --option generate

Temperature is 0.0.

I have wanted to see this thriller for a while, and it didn't disappoint. Keanu Reeves, playing the hero John Wick, is this day. He was playing with his toy car, driving it around the living room. Suddenly, he heard a loud crash. He had broken the car and was very sad. John was angry and he shouted at his little brother. He was only three years old and he was only three. He was only three years old. He was very ups.

Temperature is 1.0

I have wanted to see this thriller for a while, and it didn't disappoint. Keanu Reeves, playing the hero John Wick, is coming a new friend to play with." Iwriting brought a mask, but I wore plain white beats. The sun shone brightly. I be a happy, obedient lung. It'scious to be a true hero. What should I play?"here Ialen Once upon a time, there was a jolly youth named Lucy.

3.2. Zero Shot Prompting

Zero-shot prompting là phương pháp sử dụng trực tiếp mô hình pretrained mà không huấn luyện thêm cũng như cho ví dụ

- Zero-Shot Prompting for SST:

python run_llama.py --option prompt --batch_size 10 --train data/sst-train.txt --dev data/sst-dev.txt --test data/sst-test.txt --label-names data/sst-label-mapping.json --dev_out sst-dev-prompting-output.txt --test_out sst-test-prompting-output.txt

```
load 1101 data from data/sst-dev.txt
load 2210 data from data/sst-test.txt
eval: 100%|
eval: 100%|
dev acc :: 0.213
test acc :: 0.224
```

- Zero-Shot Prompting for CFIMDB:

python run_llama.py --option prompt --batch_size 10 --train data/cfimdb-train.txt --dev data/cfimdb-dev.txt --test data/cfimdb-test.txt --label-names data/cfimdb-label-mapping.json --dev_out cfimdb-dev-prompting-output.txt --test_out cfimdb-test-prompting-output.txt

```
load 245 data from data/cfimdb-dev.txt
load 488 data from data/cfimdb-test.txt
eval: 100%|
eval: 100%|
dev acc :: 0.502
test acc :: 0.213
```

3.3 Classification Finetuning

- Finetuning for SST:

python run_llama.py --option finetune --epochs 5 --lr 2e-5 --batch_size 80 --train data/sst-train.txt --dev data/sst-dev.txt --test data/sst-test.txt --label-names data/sst-label-mapping.json --dev_out sst-dev-finetuning-output.txt --test_out sst-test-finetuning-output.txt

```
load model from finetune-5-2e-05.pt
load 1101 data from data/sst-dev.txt
load 2210 data from data/sst-test.txt
eval: 100%|
eval: 100%|
dev acc :: 0.425
test acc :: 0.445
```

- Finetuning for CFIMDB:

```
python run_llama.py --option finetune --epochs 5 --lr 2e-5 --batch_size 10 --train data/cfimdb-train.txt --dev data/cfimdb-dev.txt --test data/cfimdb-test.txt --label-names data/cfimdb-label-mapping.json --dev_out cfimdb-dev-finetuning-output.txt --test_out cfimdb-test-finetuning-output.txt
```

```
load model from finetune-5-2e-05.pt
load 245 data from data/cfimdb-dev.txt
load 488 data from data/cfimdb-test.txt
eval: 100%|                                                | 25/25 [00:16<00:00,  1.53it/s]
eval: 100%|                                                | 49/49 [00:31<00:00,  1.53it/s]
dev acc :: 0.865
test acc :: 0.457
```

Kết luận:

- **Prompting** phù hợp cho triển khai nhanh nhưng hiệu suất kém, đặc biệt với SST.
- **Finetuning** mang lại kết quả tốt hơn đáng kể, nhưng cần tối ưu thêm (tăng epoch, điều chỉnh lr, hoặc dùng GPU) để đạt hiệu suất cao hơn.

Link github: https://github.com/phulocnguyen/Llama2-Implementation