OPTIMIZATION TECHNIQUES FOR TRANSFORMER

Chi Tran[abc] Huy Thai[abc] Loc Tran[abc] Tuan Vo[abc] Nam Le[abc]

{18127070, 18127109, 10127131, 18127248, 18120061} [at] student [dot] hcmus [dot] edu [dot] vn

[a]Department of Computer Science
[b]Faculty of Information Technology, University of Science, Ho Chi Minh City, Vietnam
[c]Vietnam National University, Ho Chi Minh City, Vietnam

12th August 2022

Mục lục

Kiến trúc mô hình sử dụng CodeBert/ GraphCodeBert pretrain

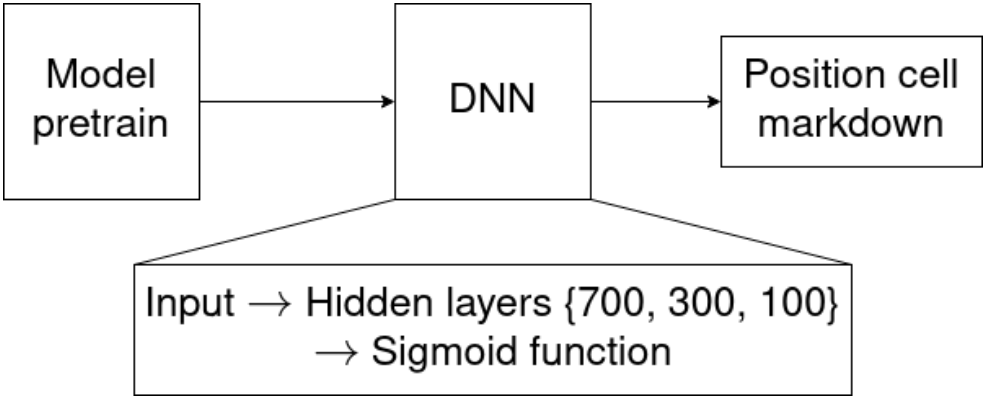


Figure 1: Kiến trúc mô hình sử dụng CodeBert/ GraphCodeBert pretrain.

Kiến trúc mô hình PairWise BertSmall

Phần MarkdownModel, sử dụng phiên bản nhỏ của mô hình BERT gốc với đầu vào là 512 để thời gian huấn luyện mô hình được nhanh hơn.

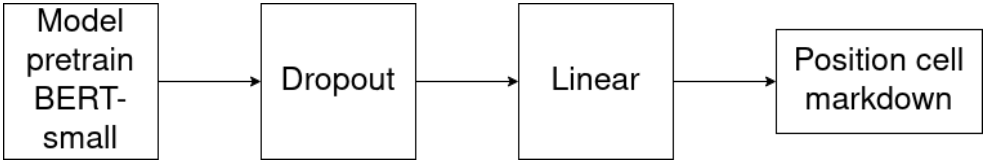


Figure 2: Kiến trúc mô hình sử dụng PairWise BertSmall.

Phần MarkdownDataset, sẽ sử dụng các bộ ba đã tạo thay vì sử dụng `train_df`.

Gradient Accumulation

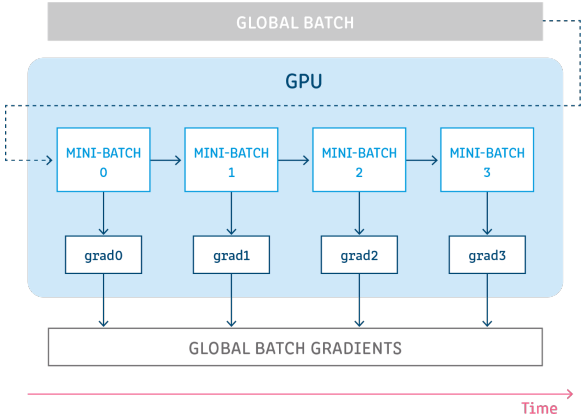


Figure 3: Visualization of how Gradient Accumulation works.

Full-Precision 32-bits vs Half-Precision 16-bits

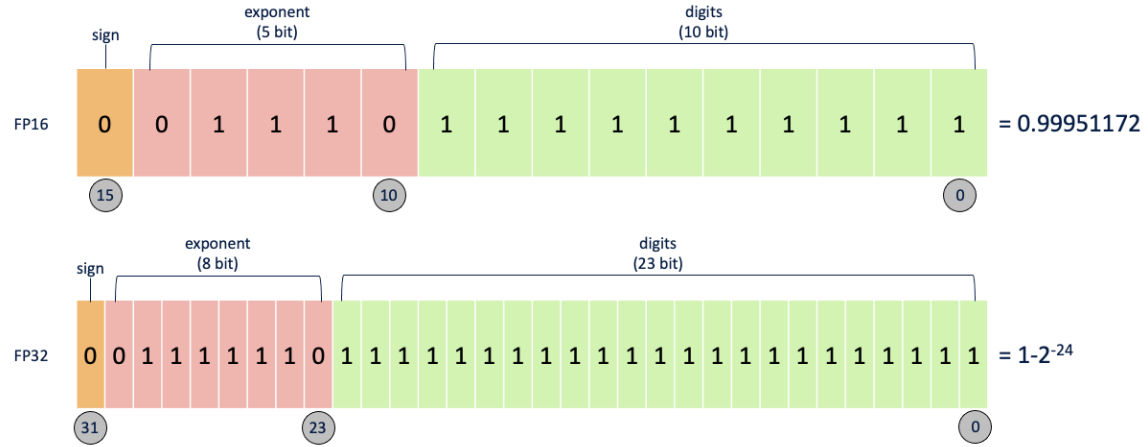


Figure 4: Full-Precision 32-bits vs Half-Precision 16-bits. The number shown, for illustrative purposes, is the largest number less than one that can be represented by each format.

Mixed Precision Training and Loss Scaling Technique

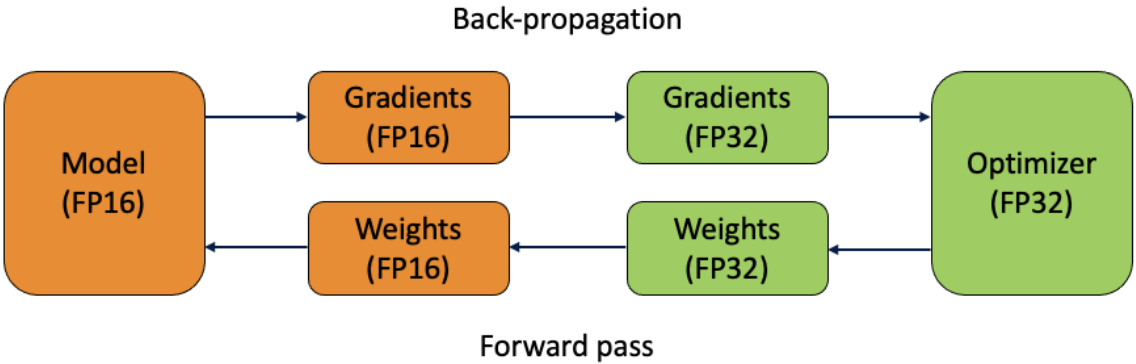


Figure 5: Basic Mixed-Precision Ttraining.

Mixed Precision Training and Loss Scaling Technique

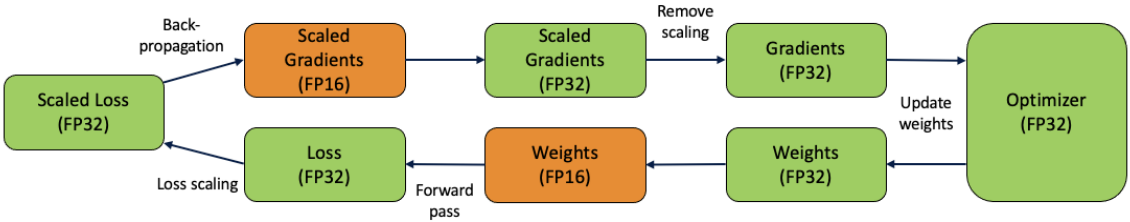
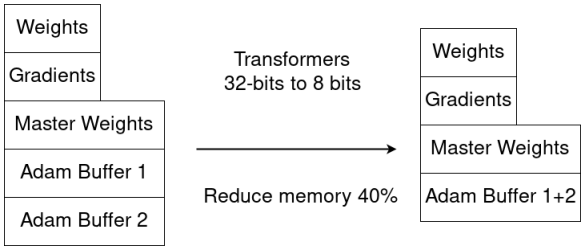


Figure 6: Full architecture Mixed-precision Training.

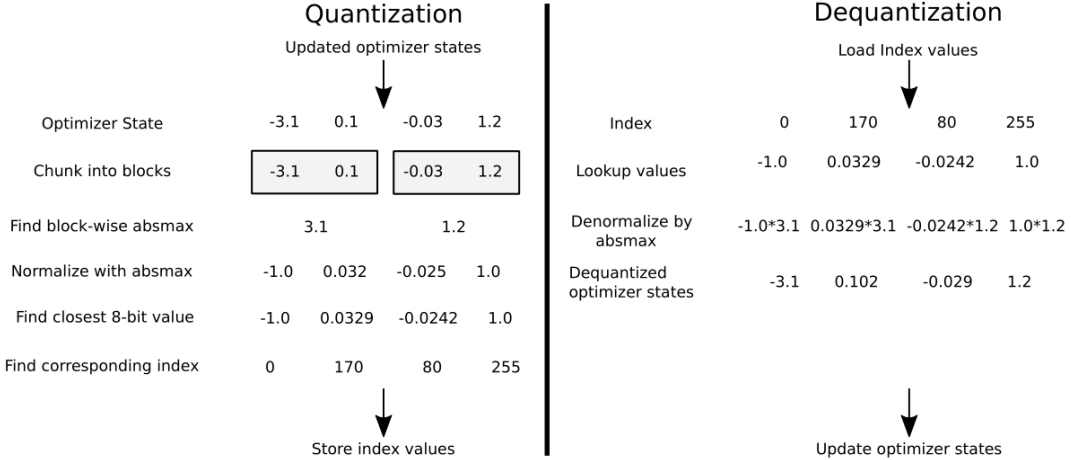
8-bit Optimizers



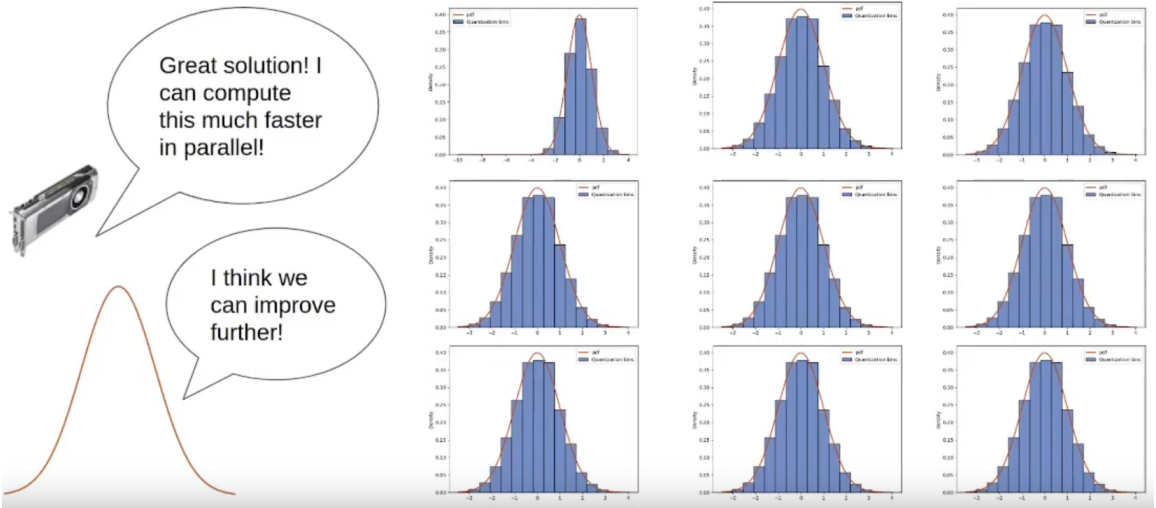
8-bits optimizers hoạt động như thế nào?

- (1) Dequantize optimizers states từ 8 bits thành 32bits;
- (2) Thực hiện optimizer update bình thường ở 32-bits;
- (3) Sử dụng block-wise dynamic quantization trên updated 32-bits states để quantize trở lại 8-bits cho việc lưu trữ;

8-bit Optimizers: Schematic of 8-bit optimizers via block-wise dynamic quantization



8-bit Optimizers: Block-wise quatization advantages



8-bit Optimizers: Dynamic Quantization

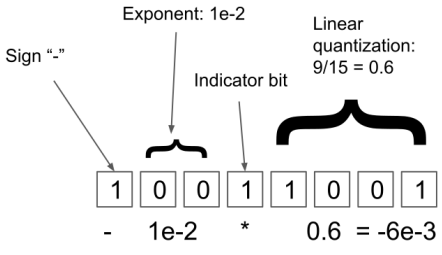


Figure 7: Dynamic tree quantization.

- (1) Bit đầu tiên của kiểu dữ liệu được dành riêng cho việc thể hiện dấu.
- (2) Số bit 0 tiếp theo cho biết độ lớn của số mũ.
- (3) Bit đầu tiên được đặt thành 1 cho biết rằng tất cả các giá trị sau được dành riêng cho
- (4) linear quantization.

Dynamic Quantization chỉ cần mở rộng lượng tử hóa cây động cho tensor đầu vào không có dấu bằng cách định hướng lại bit dấu. Và bởi vì khi sử dụng Adam, trạng thái thứ hai thường luôn dương, do đó bit dấu là không cần thiết.

Fast Tokenizers

HuggingFace cung cấp Fast Tokenizers được cài đặt bằng Rust với tốc độ cải thiện nhanh đáng kể trong các mô hình ngôn ngữ. Áp dụng chiến lược cho KaggleAI4Code như sau:

```
1 class MarkdownDataset(Dataset):
2     def __init__(self, df, model_name_or_path, total_max_len, md_max_len,
          fts):
3         super().__init__()
4         self.df = df.reset_index(drop=True)
5         self.md_max_len = md_max_len
6         self.total_max_len = total_max_len
7         self.tokenizer = AutoTokenizer.from_pretrained(model_name_or_path
              , use_fast=True)
8         if self.tokenizer.pad_token is None:
9             self.tokenizer.add_special_tokens({'pad_token': '[PAD]'})
```

I am Optim"Ú"ization

Đánh giá với mô hình CodeBert và GraphCodeBert

Các bước đánh giá với 2 mô hình CodeBert và GraphCodeBert:

- Khởi tạo đặc trưng học bao gồm đặc trưng ngữ cảnh và xếp hạng.
- Sử dụng mô hình đã huấn luyện để dự đoán.
- Xuất kết quả bao gồm thứ tự các ô.

Đánh giá với mô hình PairWise BertSmall

Các bước đánh giá với mô hình PairWise BertSmall:

- Khởi tạo dữ liệu đầu vào là các bộ ba và đặc trưng xếp hạng.
- Sử dụng mô hình đã huấn luyện để dự đoán.
- Xây dựng lại kết quả dự đoán từ tập các bộ ba đầu vào.
- Xuất kết quả bao gồm thứ tự các ô.

Kết hợp các mô hình

Kết hợp dự đoán từ các mô hình đã huấn luyện để dự đoán

- Từ mô hình CodeBERT
- Từ mô hình GraphCodeBERT
- Từ mô hình PairWise BertSmall

Tỷ lệ kết hợp tốt nhất đã thử nghiệm: a = 0.3, b = 0.5, c = 0.2, với a, b, c lần lượt là trọng số khi kết hợp các kết quả dự đoán từ các mô hình CodeBERT, GraphCodeBERT và PairWise BertSmall

Kết luận

Bằng cách kết hợp 3 mô hình CodeBert, GraphCodeBert và PairWise cùng với bộ tối ưu 8-bits trong phương pháp huấn luyện mixed precision training, đã đem lại nhiều lợi ích:

- Giảm thiểu nhiều thời gian huấn luyện mô hình;
- Tiết kiệm bộ nhớ lưu trữu trọng số, activations và gradients để tăng kích thước các siêu tham số khác;
- 8-bit Adam chủ yếu hữu ích cho việc huấn luyện các mô hình ngôn ngữ lớn từ đầu và ít hơn cho các mô hình tinh chỉnh có tham số $<$1B;

Tài liệu tham khảo I

[1] Khoi Nguyen (suicaokhoailang). Stronger baseline with code cells.

[2] Vadim Irtlach (vad13irt). Optimization approaches for Transformers.

[3] The Devastator. CodeBERT + Pairwise.

[4] Nicholas Broad (nbroad). How to use CuBert.

[5] Deep Learner. Decoupling Code and Markdown Representations w. Poly Encoders.

[6] Khattab, O., & Zaharia, M. (2020, July). Colbert: Efficient and effective passage search via contextualized late interaction over bert. In Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval (pp. 39-48).

[7] Bi-Encoder, Poly-Encoder, and Cross-Encoder for Response Selection Tasks.

[8] Deep Learner. 10 Million Open Source Jupyter Notebooks on GitHub.

[9] Additional Supporting Dataset.

Tài liệu tham khảo II

[10] HechtJP. AI4Code_Inference_XGB_Binary_Classification.

[11] Pretraining on "Kaggle's Language".

[12] Guo, D., Lu, S., Duan, N., Wang, Y., Zhou, M., & Yin, J. (2022). UniXcoder: Unified Cross-Modal Pre-training for Code Representation. arXiv preprint arXiv:2203.03850.

[13] Sharing some ideas: Feature Engineering!.

[14] Tricks From Winning Solutions Of Other Code Competitions.