Report

Implementing Transformers

Florian Kark

Heinrich-Heine-University Düsseldorf
Institute of Computer Science
Department for Dialog Systems and Machine Learning

April 9, 2024

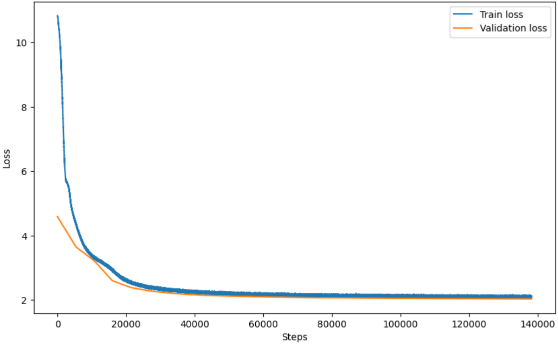
Agenda

1. Hardware and Hyperparameters
2. The Vanishing/Exploding Gradients problem
3. How to optimize the code + Performance tricks
4. Translation Examples

Hardware and Hyperparameters

Starting point is the Vanilla transformer model as in (*)

- Hardware: 1 x Nvidia A100
- Hyperparameter: Base Model variation
- Total Parameters: 69.711.872
- BLEU: 20,3



**Vaswani et. al Attention is all you need*

Base Model Hyperparameter

Hyperparameter	Value
d_{model}	512
Heads	8
Encoder Layers	6
Decoder Layers	6
FFN Dimension	2048
Dropout	0.1
Max token length	64
Weight Decay	0.1
Learning Rate	1.0
β_1, β_2	0.9, 0.98
ϵ	1e-9
Warm up steps	4000
Epochs	30
Batch Size	1024
Random Seeds	1337

Table: Hyperparameters used in Vanilla Base Transformer with AdamW

The Vanishing/Exploding Gradients problem

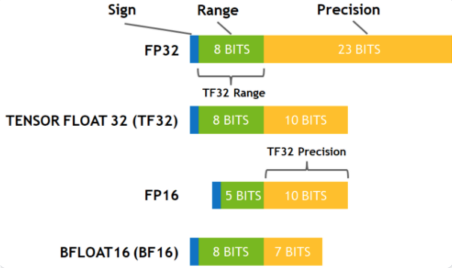
Reasons for vanishing/exploding gradients in vanilla Transformer on GPU

- Causal masks (FP16)
- Activations/Loss (FP16)
- Xavier initialization
- Residual Connections position

The Vanishing/Exploding Gradients problem

1&2/4

- Lower required memory enables training of larger models/minibatches
- But narrows supported numerical range to $[2 \cdot 10^{-24}, 65.504]$



Solution:

- use float(-1e4) in future mask
- use grad clipping (successful value in testing: 1.0) & scaler.scale(loss)
- else use torch.bfloat16 (more range, less precision)

https://docs.nvidia.com/deeplearning/performance/mixed-precision-training/index.html
https://huggingface.co/docs/transformers/v4.15.0/performancefloating-data-types

The Vanishing/Exploding Gradients problem
3/4

- No convergence: input too big & starting weights are too big (Xavier) $\Rightarrow$ big LayerNorm
- Transformer has big signals $\Rightarrow$ need smaller weights

Solution: Use smaller init!

- Original code by authors: uniform unit scaling on $[-\frac{\sqrt{3}}{\sqrt{dim}}, \frac{\sqrt{3}}{\sqrt{dim}}]$ (here -0.07, 0.07)
- BERT and GPT use normal_(mean=0.0, std=self.config.initializer_range)
- I choose torch.nn.init.normal_(module.weight, mean=0.0, std=0.02)

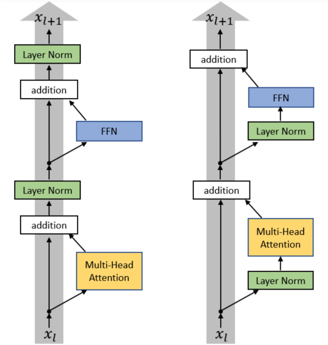
$\Rightarrow$ Now stable again, but only with warm up.
Disadvantage: more hyperparameter tuning with warm up.

https://github.com/tensorflow/tensor2tensor
https://github.com/google-research/bert
https://github.com/huggingface/transformers/tree/v4.39.3/src/transformers/models/gpt2

The Vanishing/Exploding Gradients problem

4/4 Residual Connection

We can remove the warm up if we use the so called PreNorm.



Latest code version by authors contains PreNorm as well!

Xiong et. al On Layer Normalization in the Transformer Architecture
https://github.com/tensorflow/tensor2tensor

Residual Connection

4/4 Why does PreNorm work?

- Still lots of research around the topic!

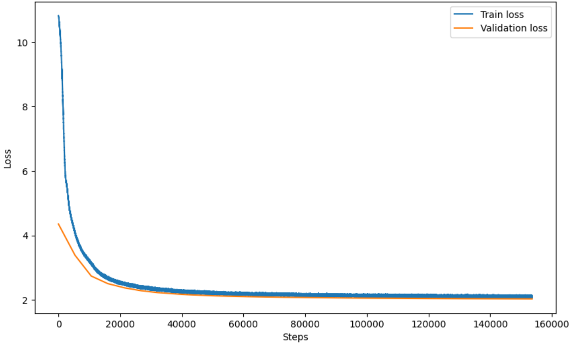
$$\|\frac{\partial \tilde{\mathcal{L}}}{\partial W^{2,L}}\|_F \leq \mathcal{O}(d\sqrt{\ln d})$$

$$\|\frac{\partial \tilde{\mathcal{L}}}{\partial W^{2,L}}\|_F \leq \mathcal{O}\left(d\sqrt{\frac{\ln d}{L}}\right)$$

They find that:
$\Rightarrow$ Gradient scale is a reason for Post-LN Transformer needing a careful learning rate scheduling
$\Rightarrow$ Gradients are large for some layers $\Rightarrow$ large learning rate without warm-up may makes training unstable

Xiong et. al On Layer Normalization in the Transformer Architecture

Transformer with PreNorm and NO warm up



BLEU-Score: 20.8 (+0.5)

How to optimize the code

For stability and higher BLEU score

- More Dropout
- remove bias everywhere
- gradient accumulation $\Rightarrow$ simulates multiple GPUs larger batch size
  $\Rightarrow$ allows higher learning rate & closer to global min

$\Rightarrow$ BLEU: $\sim$+0.3

- +50 max length (from default 64 length)
- Model averaging

$\Rightarrow$ BLEU: $\sim$+0.4-1.0

How to optimize the code

For faster training

- FP16 or BF16
- Pad vocabulary size to a multiple of 64, here 50.048 $\Rightarrow$ unaligned memory accesses significantly reduce efficiency
- model = torch.compile(model) $\Rightarrow$ static graphs instead of dynamic ones

Variation	Diff. %
FP16	134
BF16	125
grad acc steps 2	10
grad acc steps 4	14
grad acc steps 8	17
grad acc steps 16	18
torch.compile	34
pad vocabulary	15

Table: Total speed up of around 200%

Also: optimizer.zero_grad(set_to_none=True), one qkv projection for all heads

Translation examples

The Good

Source	zeiss meditec stellt geräte und ausrüstungen für arztpraxen und kliniken her.
Correct	zeiss meditec produces devices and equipment for doctors practices and clinics.
Generated	zeiss meditec produces devices and equipment for doctors and clinics.
Source	die politische weltlage ist so kompliziert, da gibt es keine einfachen antworten.
Correct	the political situation is so complicated that there are no easy answers to be found there.
Generated	the political situation is so complicated, there is no easy answer.

$\Rightarrow$ Good at short sentences with common words and easy grammar

$\Rightarrow$ Still lots of word by word translation, bad grammar! See next slide ...

Translation examples

The Bad

Source	am imbiss und am angrenzenden gebäude entstand ein schaden von 10000 euro.
Correct	damage amounting to 10,000 euros was caused to the snack bar and the neighbouring building.
Generated	the snack and the adjacent building were damaged by 10000 euros.
Source	der polizeihubschrauber flog etwa eine stunde lang verschiedene gebiete ab - erfolglos.
Correct	the police helicopter flew above various areas for about an hour - without success.
Generated	the police helicopter flew off around one hour of different areas - unsuccessful.

$\Rightarrow$ More training data/larger model needed to understand complicated German grammar + beam search might help a lot

Thank you for your Attention!

In case there are any questions, feel free to reach out to

flkar101@uni-duesseldorf.de