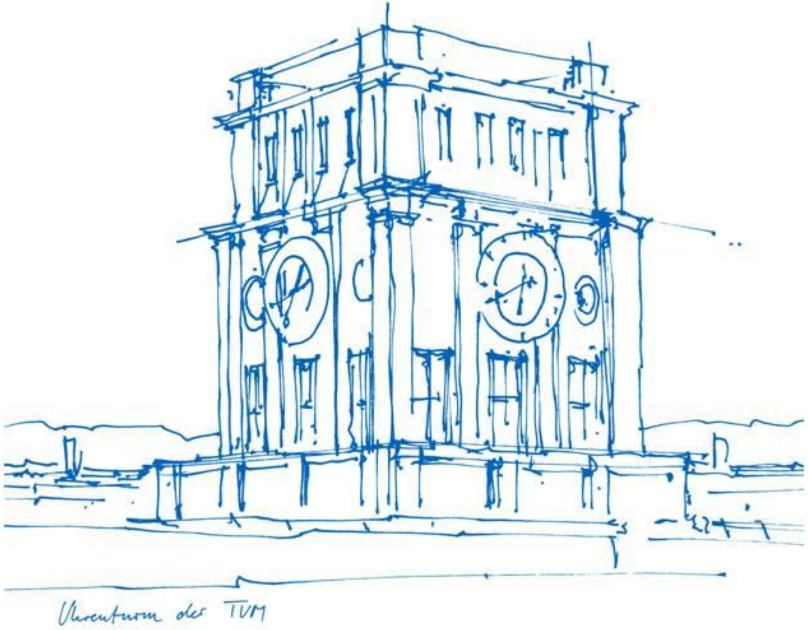
SELECTIVE PARAMETER UPDATING - MEETING 4



Coşku Barış Coşlu

Mato Gudelj

Baykam Say

14 Jun 2023

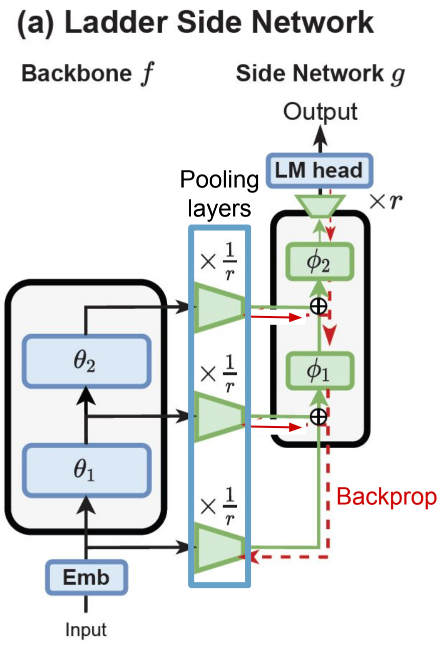
Contents

Plan from last meeting…

- Experiment with ideas on LST and $(IA)^3$
- Record results of those experiments to present in the next meeting
- Add performance-centric metrics to Tensorboard (memory footprint, forward pass latency, etc.)
- Brainstorm for even more approaches
- …keep reading literature

LST Implementation & Distillation Idea [1]

- Fixed LST implementation to work with T5
- New idea: use LST-like architecture for better distillation
    - Do forward pass with both the teacher (backbone) and student (side) networks
    - Do backprop only on the student network
    - During backprop, combine the downsampled intermediate activations form the teacher as a soft target with the gradients from upper layers
    - Separate student from teacher for inference

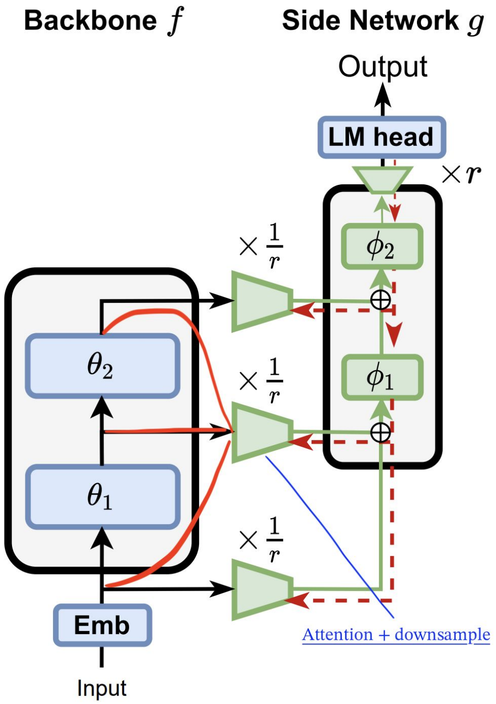


k-ladder LST idea

- Currently: i-th backbone output used as i-th ladder input
- Idea: compute input from fixed 2k+1 length window [i-k, i+k] of backbone outputs
  - Current case is a special case for k=0
  - Maybe mix/combine them with attention?
    - Positional (block-origin) encoding

Why?

- Intermediate inputs aren't solely determined by the previous layer!
  - Ladder network has to 'guess' next backbone output
    - Additive fusion: backbone_future + block_out
    - Useful feature => Requires knowledge of backbone_future
  - Solution: Allow LST to look in the "future"
  - Drawback: Can't do concurrent backbone/side inference

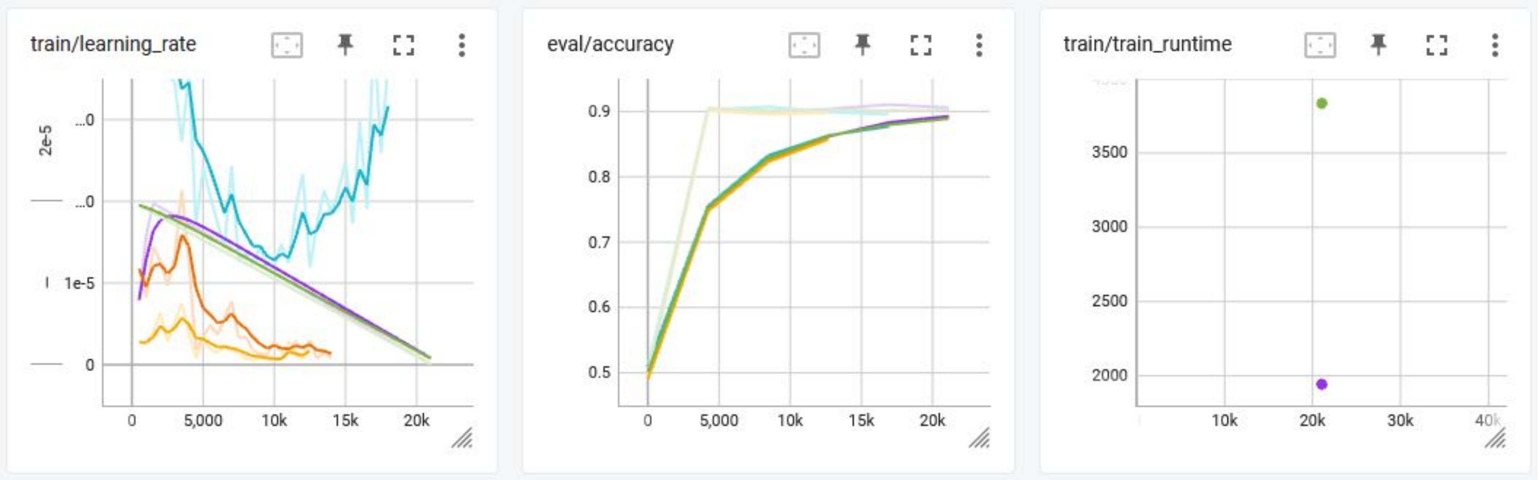


Learning Rate Distillation

- Use distillation loss to compute learning rate for each step
- The idea: if student models outputs are close to the teacher models outputs, lower lr (or potentially skip backprop) assuming learning more wouldn't be beneficial
- Implementation
  - Extend trainer to accommodate teacher model
  - Compute distillation loss after forward pass (between the outputs of student and teacher)
  - Modify adafactor [2] optimizer to take distillation loss into account when updating lr

Learning Rate Distillation - Results

- Underwhelming results, accuracy/f1 is the same as regular fine tuning in the best case with worse performance
- Hard to find a good way to use distillation loss for lr updating



— Full fine tuning with linear decay with warmup scheduler
— Best LR distillation (distillation loss combined with linear decay)

MeZO [3]: Fine-Tuning with Just Forward Passes

- Use two forward passes to estimate the gradients using Simultaneous Perturbation Stochastic Approximation (SPSA)
- Use an in-place implementation of SPSA to reduce memory costs to be same as inference as given in the MeZO paper
- Can be used in combination with other PEFT methods

Definition 1 (Simultaneous Perturbation Stochastic Approximation or SPSA [83]). *Given a model with parameters $\boldsymbol{\theta} \in \mathbb{R}^d$ and a loss function $\mathcal{L}$, SPSA estimates the gradient on a minibatch $\mathcal{B}$ as*

$$\widehat{\nabla}\mathcal{L}(\boldsymbol{\theta};\mathcal{B}) = \frac{\mathcal{L}(\boldsymbol{\theta}+\epsilon\boldsymbol{z};\mathcal{B}) - \mathcal{L}(\boldsymbol{\theta}-\epsilon\boldsymbol{z};\mathcal{B})}{2\epsilon}\boldsymbol{z} \approx \boldsymbol{z}\boldsymbol{z}^\top \nabla\mathcal{L}(\boldsymbol{\theta};\mathcal{B}) \tag{1}$$

where $\boldsymbol{z} \in \mathbb{R}^d$ with $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I}_d)$ and ϵ is the perturbation scale. *The n-SPSA gradient estimate averages $\widehat{\nabla}\mathcal{L}(\boldsymbol{\theta};\mathcal{B})$ over n randomly sampled $\boldsymbol{z}$.*

MeZO: Fine-Tuning with Just Forward Passes

Paper reports comparable performance to full fine-tuning with up to 12x memory reduction

Our progress:

- Core MeZO algorithm implemented
- Prompt-based fine-tuning is required for MeZO to work: Next step

Idea: Combine with LST

- Fine-tune LST's backbone model using MeZO
- Spend more time on training
  - Issue: MeZO requires > 10x more steps
- Maybe get better accuracy in return

	SST-2	SNLI	TREC
Prompt	89.6 (1.2)	65.1 (6.2)	66.7 (6.2)
No Prompt	51.9 (2.9)	34.8 (2.1)	19.5 (9.0)

Experiments with IA^3 [4] and LoRA [5] like models

- IA3-out
  - Use the linear output layer of attention instead of the feed forward layer in IA3
- Feed forward only
  - Put trainable vectors representing the linear layers of the feed forward network, freeze the whole network and train only the new layers
- LoRA for k_v_ffn
  - Only update key, value, and ffn layers using LoRA, freeze the rest of the layers.

Experiment Results

	# of Trainable Parameters	Accuracy	F1 Score	Train Runtime
Full FT	66M	90.59%	-	42 min
LoRA	350K	88.19%	-	34 min
$(IA)^3$	28K	90.02%	-	14.5 min
$(IA)^3$-out	14K	88.42%	88.71%	18 min
FF-only	23K	88.76%	88.99%	16 min
LoRA k-v-ffn	184K	88.53%	88.96%	26.5 min

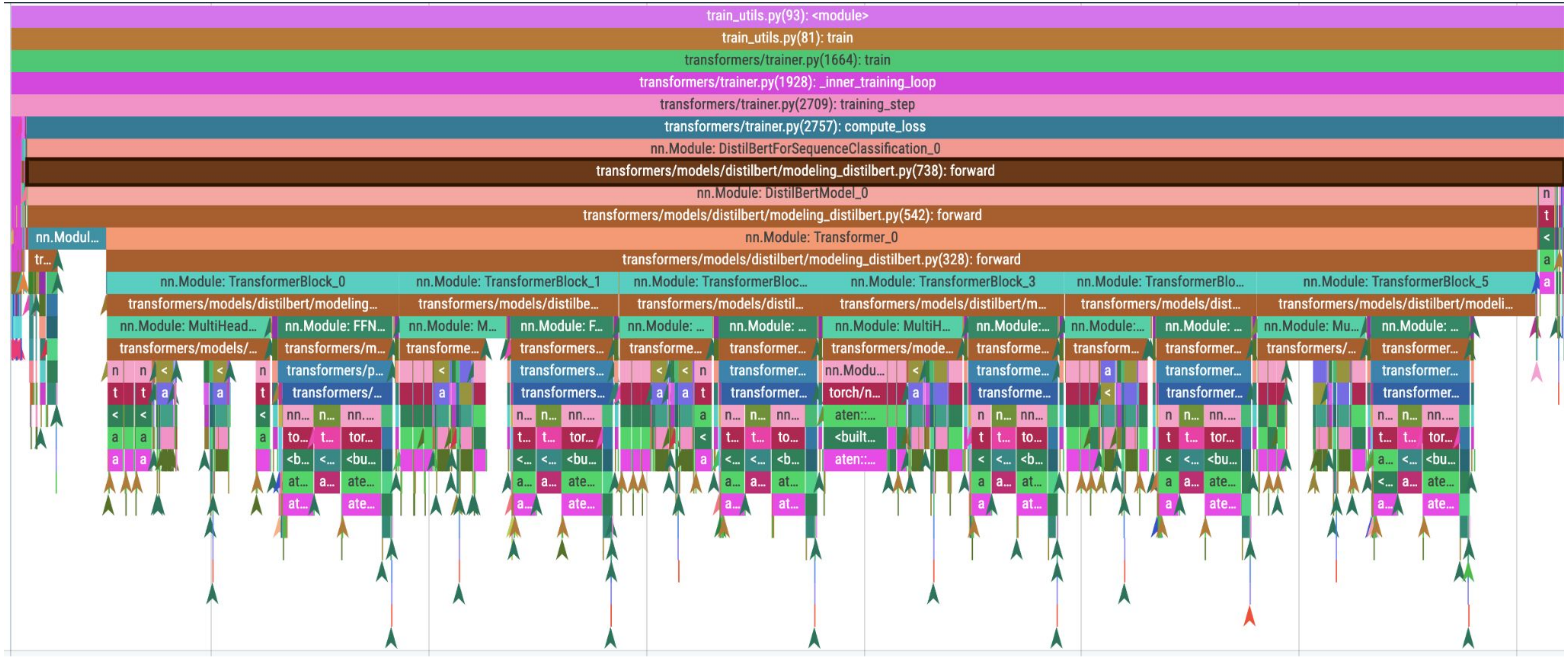
Side LoRA Idea

- LoRA still requires to compute gradients on the original weight matrices which adds a major performance overhead
- Find a way to skip gradient calculations on the original frozen model
- Solution: merge the ideas behind LST and LoRA
  - Different from just adding LoRA to LST
  - Have two models (backbone & side network)
  - Side network is the same size as backbone but it only has low rank decompositions of matrices instead of linear layers
  - After each linear layer of the backbone, add a ladder connection to the output of low rank decomposition matrices
  - Instead of gated connections, just sum the activations
  - Calculate gradients only on the side network

Profiling

- Enables a detailed look into performance characteristics
- Good tool for:
  - precise per-module latency
  - debugging (PEFT method taking longer than expected?)
  - insight into potential candidates for speedup
- Straightforward to add to codebase
- Tensorboard built-in trace viewer doesn’t work with large traces
  - https://ui.perfetto.dev/

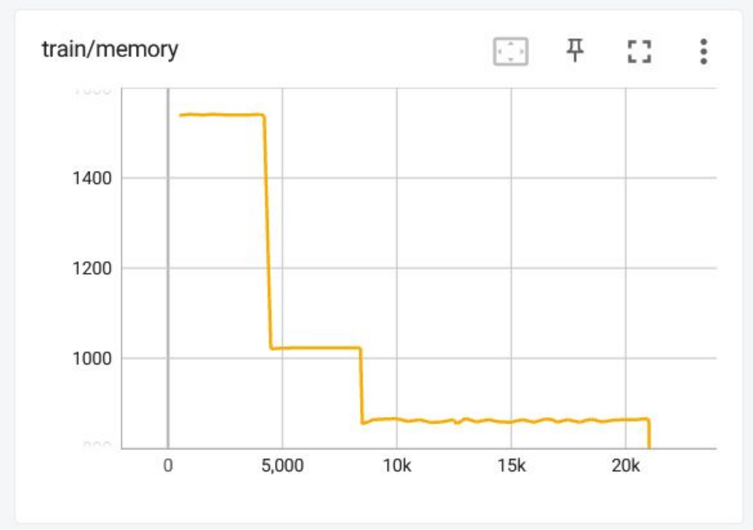
Profiling



train_utils.py(93): <module>
train_utils.py(81): train
transformers/trainer.py(1664): train
transformers/trainer.py(1928): _inner_training_loop
transformers/trainer.py(2709): training_step
transformers/trainer.py(2757): compute_loss
nn.Module: DistilBertForSequenceClassification_0
transformers/models/distilbert/modeling_distilbert.py(738): forward
nn.Module: DistilBertModel_0
transformers/models/distilbert/modeling_distilbert.py(542): forward
nn.Module: Transformer_0
transformers/models/distilbert/modeling_distilbert.py(328): forward
nn.Module: TransformerBlock_0
nn.Module: TransformerBlock_1
nn.Module: TransformerBlock_3
nn.Module: TransformerBlock_5

Tensorboard: Memory Footprint

- Added memory footprint measurement
- Added forward pass latency data
- Image:
  - Gradual model freezing



Open Issues & Solutions

- Issues
  - Current ideas are underwhelming
    - No working novel idea
- Solutions
  - Find and implement new ideas
    - k-ladder LST
    - MeZO + LST
    - Side LoRA
    - Find some variables to decide whether to skip weight updating per step basis (and only update biases)

Plan for the Next Two Weeks

- Implement k-ladder LST and record some results
- Implement MeZO + LST
- Implement Side LoRA
- Start working on the report
- Brainstorm more approaches
- …keep reading literature

References

[1] Yi-Lin Sung, Jaemin Cho, Mohit Bansal: “LST: Ladder Side-Tuning for Parameter and Memory Efficient Transfer Learning”, 2022; arXiv:2206.06522

[2] Shazeer, Noam, and Mitchell Stern. ‘Adafactor: Adaptive Learning Rates with Sublinear Memory Cost’. *arXiv [Cs.LG]*, 2018; arXiv:1804.04235

[3] Malladi, Sadhika, et al. ‘Fine-Tuning Language Models with Just Forward Passes’. *arXiv [Cs.LG]*, 2023; arXiv:2305.17333

[4] Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, Colin Raffel: “Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning”, 2022; arXiv:2205.05638

[5] Hu, Edward J., et al. ‘LoRA: Low-Rank Adaptation of Large Language Models’. ArXiv [Cs.CL], 2021; arXiv:2106.09685