How to train Transformers effectively

Shariq
Peter Wonka Reading Group - 9/16/2021
VCC

What kind of data works best?

- Traditionally
  - Long proven record on **tokenized / quantized** data - mostly language
    - Sequential
    - Token Sets
    - Any data that can be quantized via a **fixed-size finite dictionary of embeddings (vocabulary)**
- More recently
  - Real valued sequences / sets of vectors - Images / Videos
  - DETR, ViT, DEiT etc

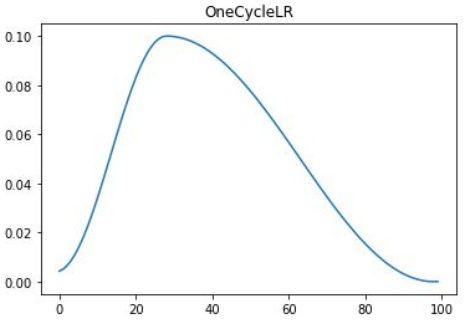
Why is it difficult? Skip connections: A necessary evil

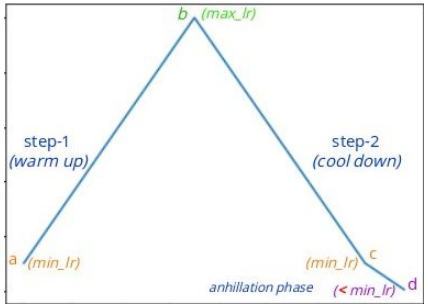
- Pure Attention loses rank **doubly exponentially** with depth! [1]
- What counteracts rank collapse?
  - Skip connections are crucial
  - MLPs help
- Skip connections amplify small parameter perturbations = Unstable training [2]

[1] Dong, Y., Cordonnier, J.B. and Loukas, A., 2021. **Attention is not all you need**: Pure attention loses rank doubly exponentially with depth. *arXiv preprint arXiv:2103.03404*.
[2] Liu, L., Liu, X., Gao, J., Chen, W. and Han, J., 2020. **Understanding the difficulty of training transformers.** *arXiv preprint arXiv:2004.08249*.

Training tips - 1 : Optimization

- Vanilla SGD DOES NOT work
- Use SGD with LR schedules
- More preferably use Adam / AdamW with LR schedules
- **Warmup is essential!**
  - Start small e.g. initial lr = 5e-5
  - warmup + linear or cosine decay
  - OneCycleLR generally works good enough





- Incase of diverged training, Use gradient clipping (e.g. max_grad_norm = 1)

Training tips - 2 - Design

- Model size : Start small but scale up to highest possible
  - Use smaller models first (e.g. layers = 2) for faster debugging.
  - Scale up eventually as much as you can, according to dataset size.

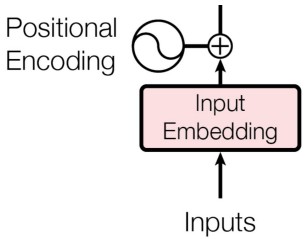
- Batch size : Start with highest possible

- For big datasets: Use as many GPUs as you can for faster results
  - e.g. one 8-GPU experiments one after another > two 4-GPU exps in parallel > eight 1-GPU exps in parallel

Training tips - 3 : ViT

- Lack of inductive bias (vs CNNs) = Extremely data hungry = Massive compute
- Transfer Learning is always the better option
  - Pretrained ViT, DeiT
- Introduce CNNs to form hybrids
  - CNN layers for initial encoding
- Data augmentation > Label smoothing > Regularization
- Increasing Patch Size > Shrinking model size

Other comments



- Position encoding is important (optional if using CNNs)
  - Predefined sine-cosine ≈ Learned encodings
- Try using all the output embeddings for faster convergence even if only one is needed
  - E.g. softmax + pool
- Stay updated with the empirical findings that work

Thank you!
and keep taming transformers!