Fundamentals of Foundation Models
Cheat sheet

Lukas Schüttler

July 17, 2024

1 Transformers

Token Embeddings
$\mathbf{I} \in \mathbb{R}^{\text{voc}\times N}$: Input Sequence - $\mathbf{M}_{\text{emb}} \in \mathbb{R}^{\text{dim\_emb}\times\text{voc}}$: Embedding Matrix - $\mathbf{M}_{\text{pos}} \in \mathbb{R}^{\text{dim\_emb}\times\text{N}}$: Positional Embedding - $\mathbf{E} \in \mathbb{R}^{\text{dim\_emb}\times\text{N}}$: Token Embeddings

$$\mathbf{E} = \mathbf{M}_{\text{emb}} \times \mathbf{I} + \mathbf{M}_{\text{pos}}$$

Self Attention
$\mathbf{W} = [\mathbf{W}_q\ \mathbf{W}_k\ \mathbf{W}_v]^\top \in \mathbb{R}^{3\text{dim\_emb}\times\text{dim\_emb}}$ - $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V} \in \mathbb{R}^{\text{dim\_emb}\times N}$: Query, Key, Value - $\mathbf{A} \in \mathbb{R}^{N\times N}$: Attention Matrix

$$[\mathbf{Q}\ \mathbf{K}\ \mathbf{V}]^\top = \mathbf{W} \times \mathbf{E}$$
$$\mathbf{A} = \text{softmax}\left(\frac{\mathbf{Q}^\top\mathbf{K}}{\sqrt{d_k}}\right) \odot \mathbf{M}_{\text{mask}}$$
$$\mathbf{E}_{\text{att}} = \mathbf{V} \times \mathbf{A}$$

With d_k being the dimension of the key vectors. E.g. pos_emb for single head attention

MLP
$\mathbf{M}_{\text{up}} \in \mathbb{R}^{4\text{dim\_emb}\times\text{dim\_emb}}$: Up projection - $\mathbf{M}_{\text{down}} \in \mathbb{R}^{\text{dim\_emb}\times 4\text{dim\_emb}}$: Down projection

$$E_{\text{mlp}} = \sigma(\mathbf{M}_{\text{down}} \times \sigma(\mathbf{M}_{\text{up}} \times \mathbf{E}_{\text{att}})))$$

With σ being the activation applied elementwise

Computation: FLOPS $\approx 6N \cdot D$ - With D being the number of training tokens.

2 Tokenizers

3 Training

3.1 Datasets

Common Crawl: Database of scraped websites. **WebText**: OpenAI internal dataset. Scraped links from Reddit which received at least 3 karma. Page de-duplication and light-cleaning. *8M* Documents, *40GB* of text.
OpenWebText: Open replication of **WebText**
C4: Colossal Clean Crawled Corpus. Filtered version of **Common Crawl**. Discard pages with fewer than 5 sentences and lines with fewer than 3 words. Filter for unwanted keywords. Remove lines with the word Javascript. Remove pages with "*lorem ipsum*". Remove pages with "*{*". Deduplicate any three-sentence span occurring multiple times. Filter pages that are not in English.
The Stack: Coding dataset.
PeS2o: STEM papers.
DOLMA: Combination of **Common Crawl**, **C4**, **The Stack**, **Reddit**, **PeS2o**, **Project Gutenberg**, **Wikipedia/Wikibooks**
The Pile: *800GB* Dataset of Diverse Text. Academic, Internet, Prose, Dialoque and Misc (GitHub, Math ...)

Dataset Cleaning Pipeline: Language Filtering → Deduplication (by URL) → Quality Filters → Content Filters → Deduplication (by text overlap)

3.2 Evaluation

Perplexity: $\exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P(t_i)\right)$

With t_i being the ith token in the expected output sequence

Benchmarks
Paloma: Perplexity over a diverse set to text.
HellaSwag: QA Benchmark. Most likely answer by perplexity is chosen.
MMLU: QA Benchmark. Answers are part of the Prompt. Model can answer A, B, C or D. Most likely token is chosen.

3.3 Fine-Tuning

Possible with around 1000 high quality prompts and responses or more.
RLHF - Reinforcement Learning from Human Feedback

$$y_1, y_2 \propto \pi_{\text{SFT}}(y \mid x) \qquad\qquad y_w > y_\ell \mid x$$
$$p^*(y_1 > y_2 \mid x) = \frac{1}{1 + \exp(r^*(x \mid y_2) - r^*(x \mid y_1))}$$
$$\mathcal{L}(r) = -\mathbb{E}\left[\log\sigma(r(x \mid y_\ell) - r(x \mid y_w))\right]$$

Where r is the reward model and $\mathcal{L}$ is the loss of the reward model.

$$\max_\pi \mathbb{E}\left[r(x,y) - \beta\,\mathrm{D}_{\text{KL}}(\pi(y \mid x) \mid \pi_{\text{ref}}(y \mid x)\right]$$

D_{KL} to reduce the deviation from the base model (SFT).

DPO - Direct Preference Optimization

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta\log\frac{\pi_\theta(y_\ell \mid x)}{\pi_{\text{ref}}(y_\ell \mid x)}\right)\right]$$

4 Scaling Laws

$$L(D) \approx \frac{A}{D^{\alpha_D}} \qquad L(N) \approx \frac{B}{N^{\alpha_N}}$$

$$R(f_{N,D}) = R(f^*) + (R(f_N) - R(f^*)) + (R(f_{N,D}) - R(f_N))$$
$$R(f_{N,D}) = E + L(N) + L(D)$$

Let $R(f^*)$ be the irreducible error, $R(f_N) - R(f^*)$ the approximation error of a N-parameter model and $R(f_{N,D}) - R(f_N)$ the statistical error Parameter Values depend on data and precise model architecture.

4.1 Compute Optimality

$\arg\min L(N, D)$ for a constant compute budget $C(N, D) = H$ by choosing the best N and D.

Training curve envelope - Plot training curves (x: log(FLOPS), y: $L(N, D)$), Find the envelope (Minimal loss per FLOP), Plot envelope points twice (x: log(FLOPS), y: $\log(D)$, $\log(N)$), Fit a line to the points and extrapolate to the desired FLOPS to find the optimal N and D.

IsoFLOP Curves - Train various model sizes with D such that the final FLOPs are constant, Repeat for different final FLOPs, Plot final loss (x: $\log(N)$, y: $L(N, D)$), Locate optimal model size for a given compute budget (loss valley), Plot optimal models (x: log(FLOPS), y: $\log(D)$, $\log(N)$) and extrapolate.

Parametric fit - Fit parametric Risk function $R(f_{N,D})$ to the training results (Training like IsoFLOP), Plot contours (x: log(FLOPS), y: $\log(N)$), Fit line such that it goes through each iso-loss contour at the point with the fewest FLOPs. Extrapolate to desired FLOPs. Alternative - Plot isoFLOP slice (x: $\log(N)$, y: $L(N, D)$), plot parametric risk for desired compute budget, Locate the minimum.

Key finding: For compute optimal training D should scale proportionally with N - Compute Optimality doesn't consider inference cost

5 Ensembles and MoE

5.1 Ensembles

Linear interpolation - $P(y \mid x) = \sum_m P_m(y \mid x) P(m \mid x)$

Log-linear interpolation - $\mathrm{softmax}\left(\sum_m \log P(y \mid x) \lambda_m(x)\right)$

With $P_m(y \mid x)$ being the output of the model m and $P(m \mid x)$ the "reliability" of the model given the Input. λ_m is an interpolation coefficients for the model m. **Parameter Averaging** - Calculate model weights by accumulating (average, weighted average ...) weights from multiple models

5.2 MoE - Mixture of Experts

Gaussian mixture model - $p(y \mid x) = \sum_k p_{\theta_k}(y) p_k(x)$
Where p_{θ_k} is a gaussian distribution parametrized by θ_k giving the propability of the output y. p_k is the probability of that distribution given the input x.
Allows the approximation of more complex distributions using only simple distributions (gaussian and logistic)

Routing
Shazeer - Mixtral - Switch routing -

6 Scalable Computing

6.1 Multicore Processing

GPUs are optimized for performing the same operation on different data points simultaneously.
SIMD - single-instruction multiple-data
Amdahl's law - Let the non-parallizable part of a program take a fraction s of the time, then m workers can result in a speedup of:

$$\frac{1}{s + \frac{1-s}{m}}$$

Floating Point Numbers

	Sign	Exponential	Mantissa
FP32	1 Bit	8 Bits	23 Bits
FP16	1 Bit	5 Bits	10 Bits
BF16	1 Bit	8 Bits	7 Bits

BF16 - FP32 range with **FP16** precision
Error: $\mid x - \widetilde{x} \mid \leq \epsilon/2 \mid x \mid$ with $\epsilon \neq 0$

6.1.1 CUDA

thread - core; thread block - streaming multiprocessor (SM); kernel grid - CUDA-capable GPU
CUDA Thread - Smallest compute entity
CUDA Block - Group of threads (up to 1024)
Streaming Multiprocessor - Executes one **CUDA Block**

6.2 Distributed Computing

7 Vision Models