What is Language Model?

How do we make the machine generate language quickly?

Konrad Handrick

Workshop 2: stepping stone in AI

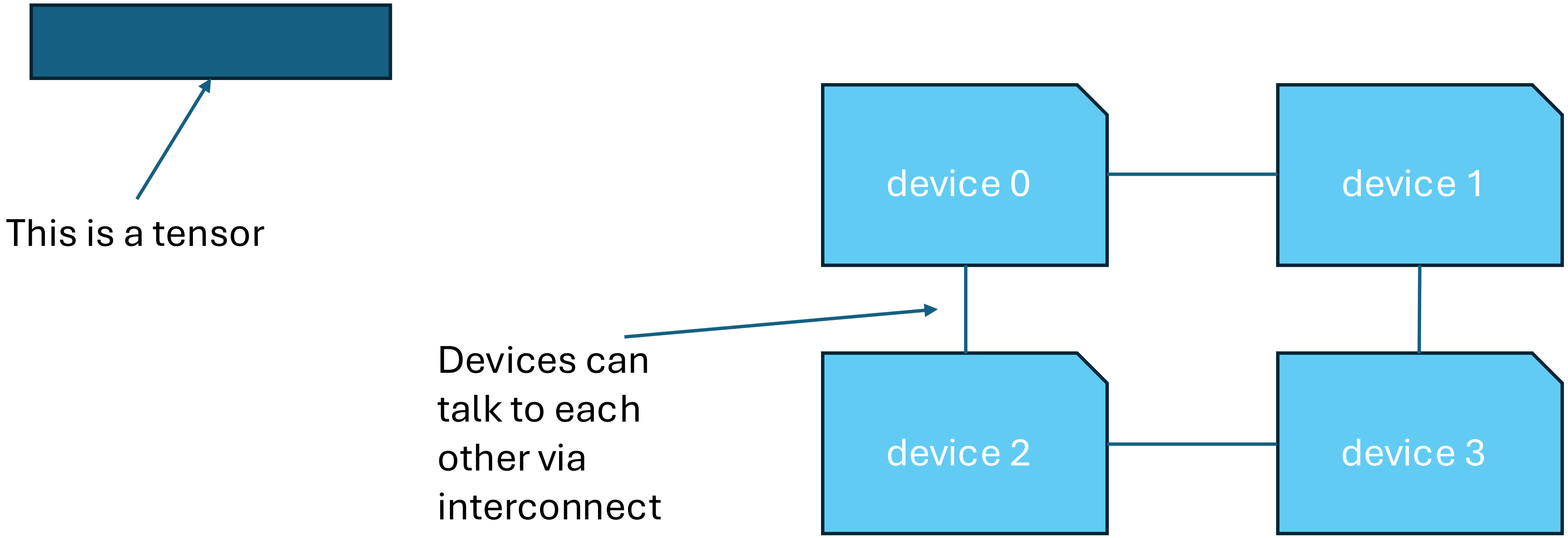
November 2025

Today’s menu

1. Distributed computing primitives
2. What’s a Language Model?
3. How do we split a Language Model on several devices?
4. Next steps

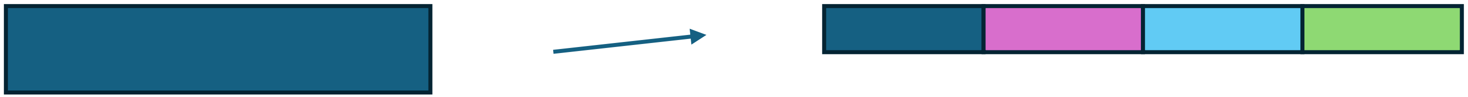
First: Distributed Computing & tensors

Reminder: Everything is Linear Algebra. For all intents and purposes in this section, **a tensor is an array.**



First: Distributed Computing & tensors

Reminder: Everything is Linear Algebra. For all intents and purposes in this section, **a tensor is an array.**

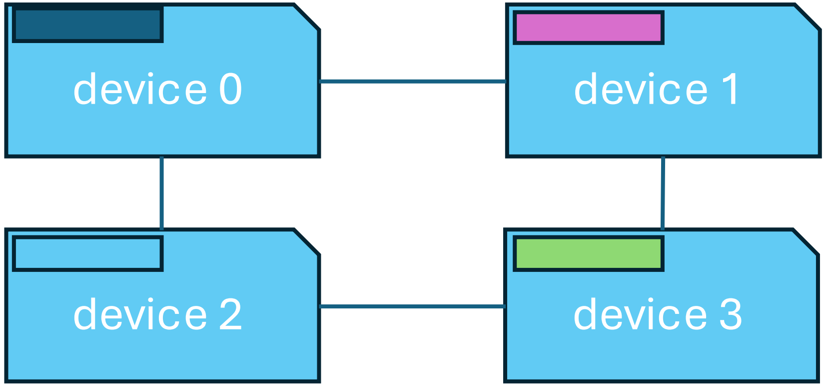


This is a sharded tensor

Often people use the nomenclature
“tensor X sits on rank 0”

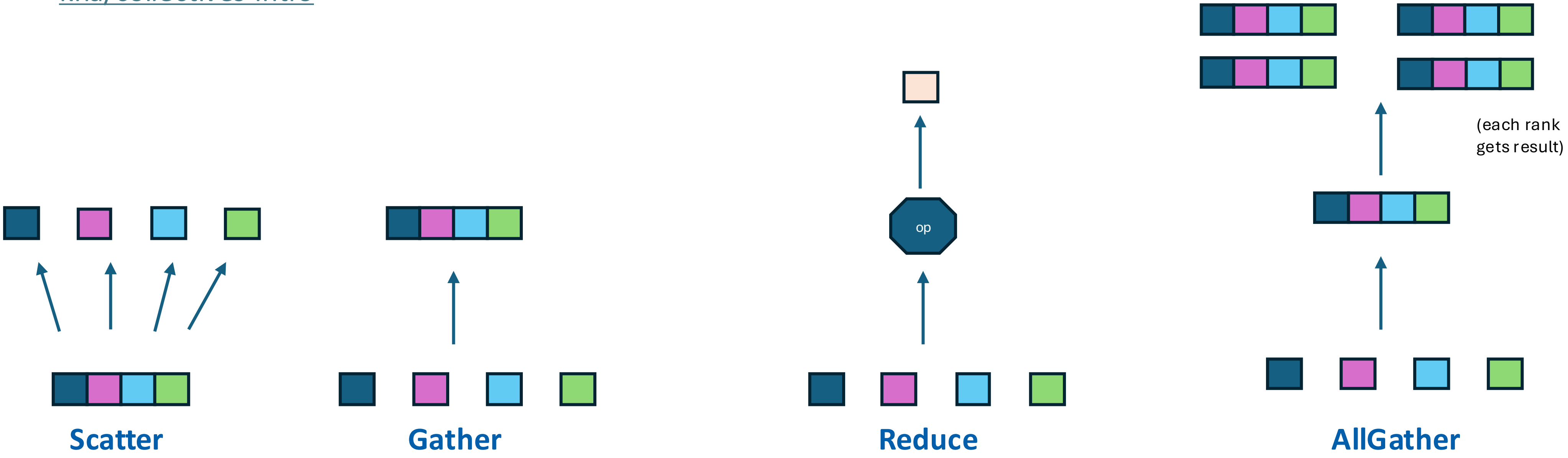
different color now indicates location of a tensor on different device / rank / machine

“device mesh”

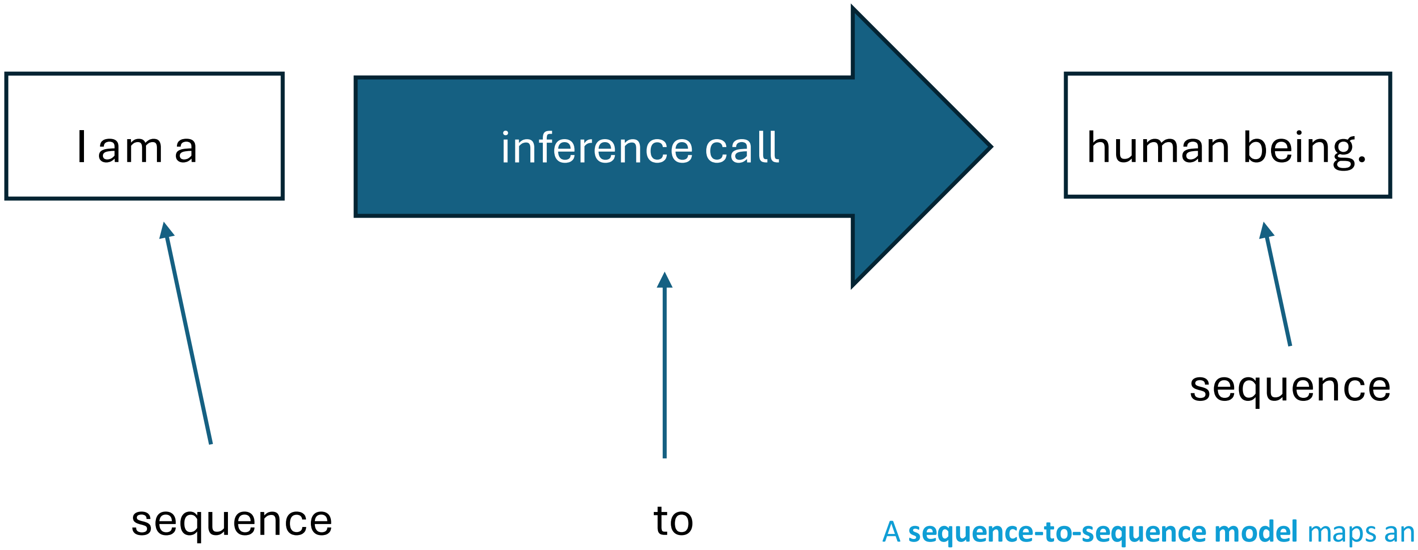


First: Distributed Computing & tensors

Why do we care? Collective operations make up a large part in multi-GPU settings. We'll use the language introduced here to describe operations in later steps. For more, with code examples, see https://git.stepping-stone.ch/eju-kha/collectives-intro



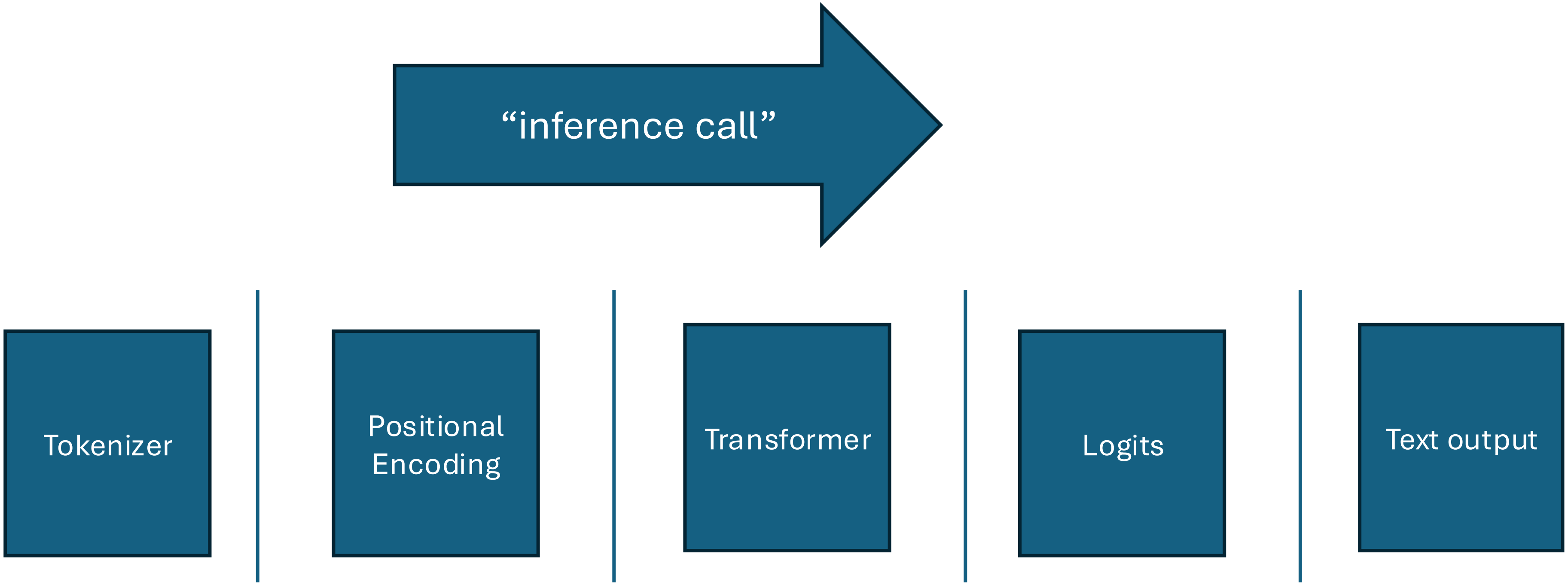
What’s a Language Model?



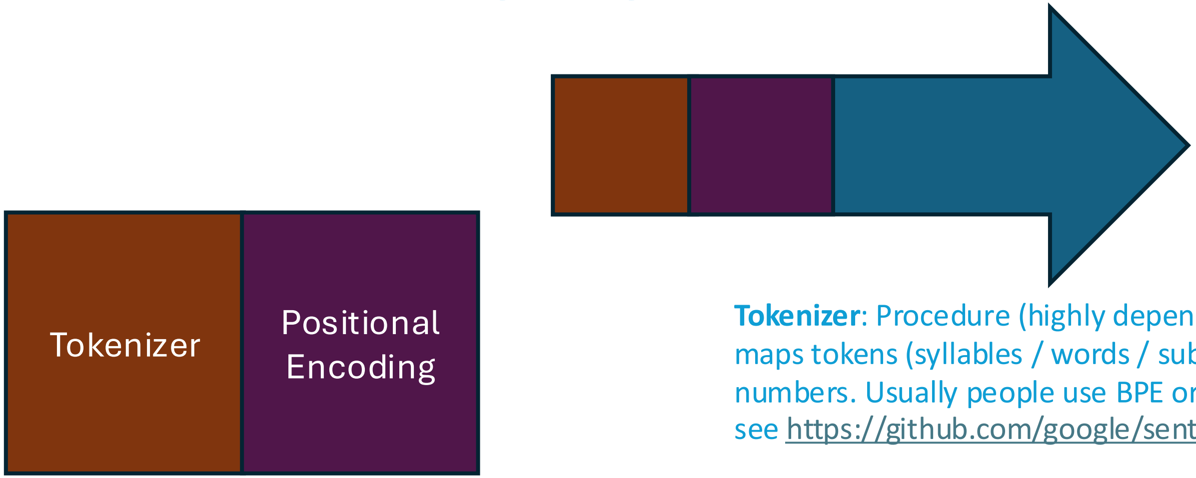
A **sequence-to-sequence model** maps an input sequence to an output sequence of not necessarily equal length. It processes the entire input context to **generate outputs one token at a time**, using the previously generated tokens as additional context for each new prediction.

More text and nice pictures: https://jalammar.github.io/illustrated-transformer/

What’s a Language Model?

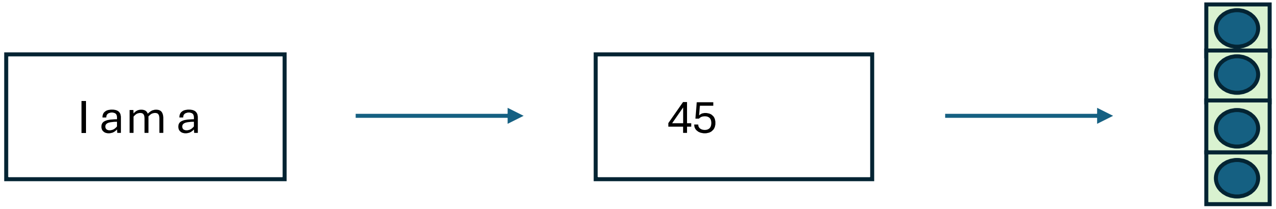


What's a Language Model?



Tokenizer: Procedure (highly dependent on data) that maps tokens (syllables / words / sub-phrases) onto numbers. Usually people use BPE or SentencePiece (e.g. see https://github.com/google/sentencepiece)

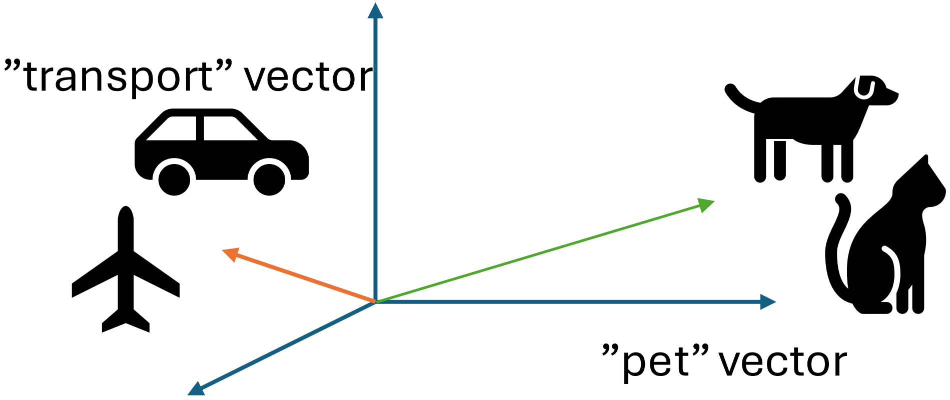
Positional Encoding: Injects information about token position relative to its context. (e.g. see https://shreyashkar-ml.github.io/posts/rope/)

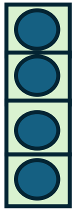


Initial step:
Transform "phrases" into "embeddings" in "latent space".

Embeddings & Latent space

Language lives in a high-dimensional space.





Embedding in latent space

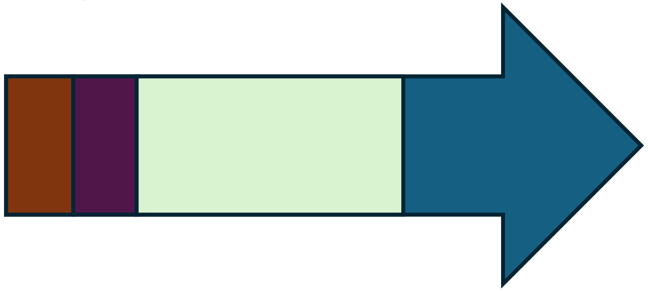
Embedding: A learned mapping from tokens to continuous vectors that preserves semantic relationships.

Latent Space: The vector space where embeddings and hidden representations live, such that semantic similarity corresponds to geometric proximity.

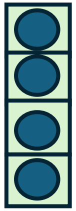
What’s a Language Model?

Attention Is All You Need

Transformer



$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



Embedding in latent space

probabilities (a.k.a. *autoregressive decomposition* [3]):

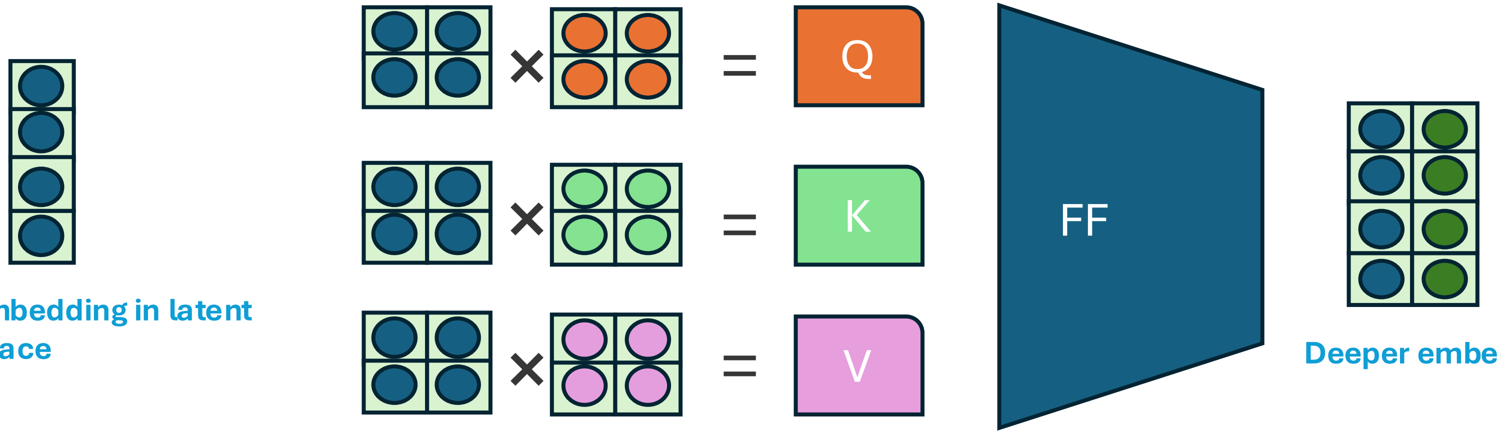
$$P(x) = P(x_1) \cdot P(x_2 \mid x_1) \cdots P(x_n \mid x_1, \ldots, x_{n-1}). \quad (1)$$

Screenshots from: https://arxiv.org/abs/1706.03762

Attention in detail

Transformer

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



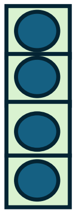
Embedding in latent space

Deeper embedding

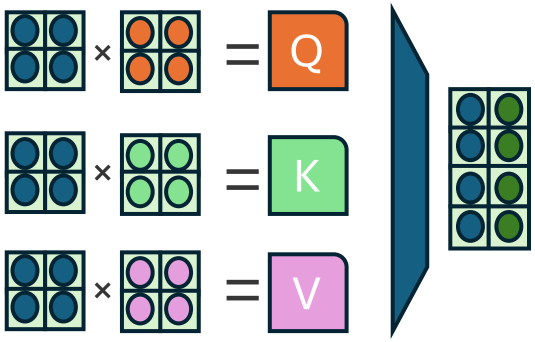
Attention in detail

Transformer

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



Embedding in latent space



K_i, V_i depend on the previous i-1 steps of the calculation – hence we can / should cache them!

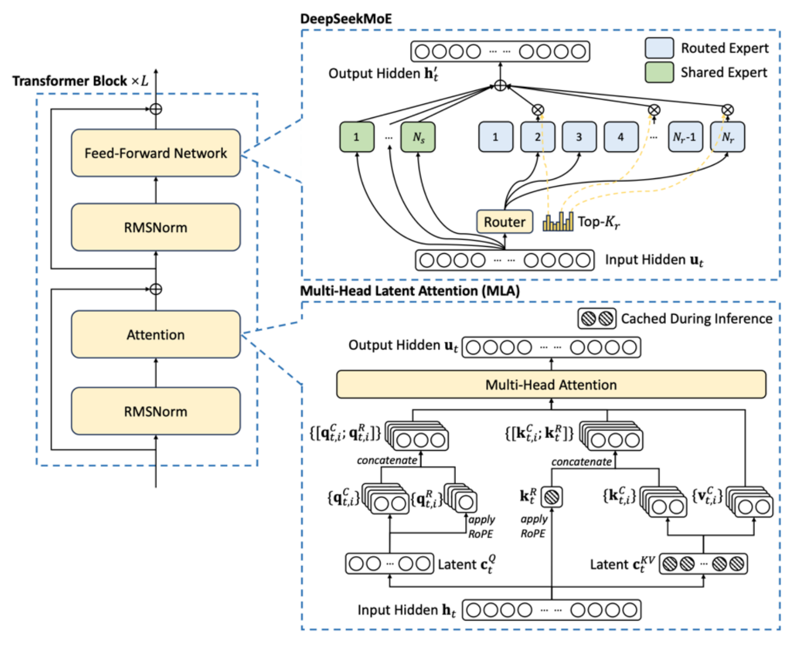
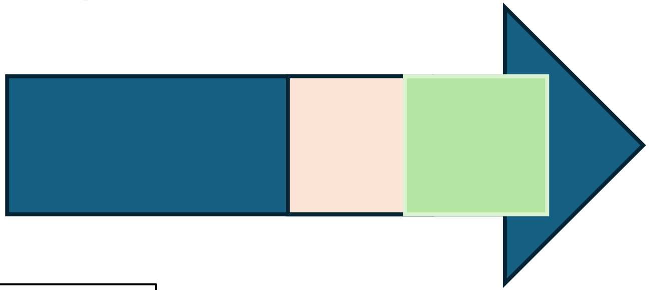


Figure 2 | Illustration of the basic architecture of DeepSeek-V3. Following DeepSeek-V2, we

Right: https://arxiv.org/abs/2412.19437

What’s a Language Model?



Logits

Text output

Aggregating the activations and sampling for the next tokens

Map the output (nvfp4 or so) to a token

… and we repeat until we reached a “kill” token such as EOS (end-of-sequence), repeating increments seq len.

The logits can be guided through several methods; one imporant parameter is “Temperature”: low (< .5) implies less “certainty”, good for logically correct answers; high (~1) good for rambling answers, “high creativity”.

What’s a Language Model?

1. Prompt phase.

Take in a users n tokens and fill (K, V)-cache. x_n+1 depends on all of them, hence this can be massively parallel.

TTFT

2. Autoregressive Generation.
x_n+t depends on all of x_n+t-1, … x_1. This phase completes upon either EOS or reaching max seq_len.

tok/s

Time to first token: How snappy is the model?

Tokens per second: How long do I have to wait for the thing to finish?

What's a Language Model?

Dimensions

How do we get to 7B, or even 450B parameters? (always look inside config.json on HF)

Let's do DeepSeek-v3 (**671B** apparently):

Token embedding (in & out): vocabulary 128000, hidden dim = 7168 -> 2 x 128000 x 7168 ~ 1.835 B

Attn, MLA: weight matrices for the attention mechanisms 11.41 B over 61 layers

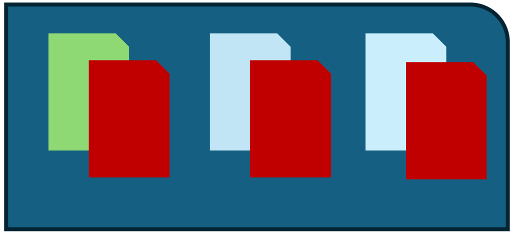
Dense FF (x3): d x (2 x 18432 + 18432) ~ 1.2B

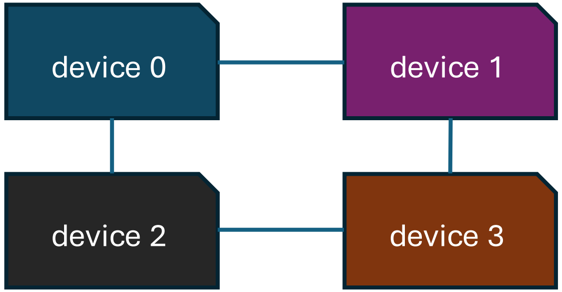
MoE FF: (256 + 1) x 44.04 M x 58 ~ 656 B (and we have to smartly count them all together)

Parallelism Strategies

How do we get our model onto X devices?

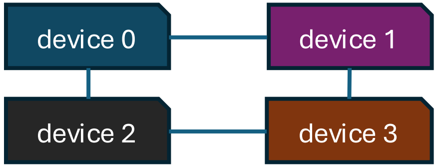
And what operations does that imply?

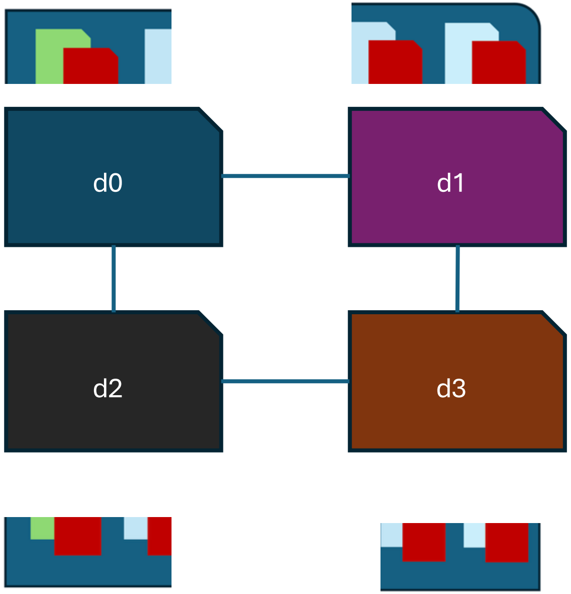




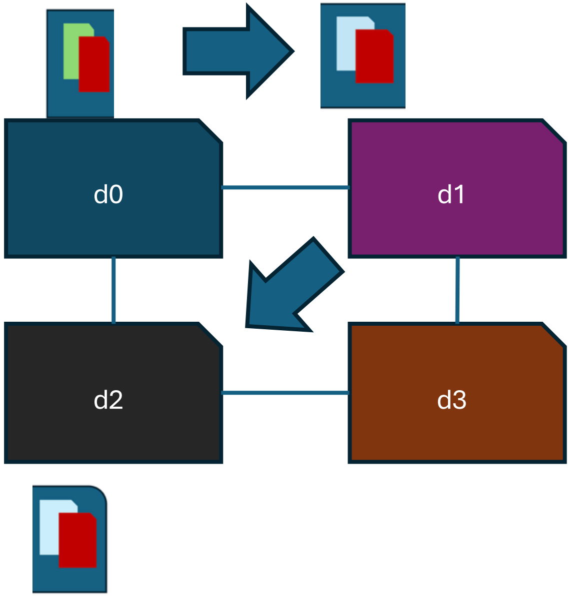
device mesh

Parallelism Strategies

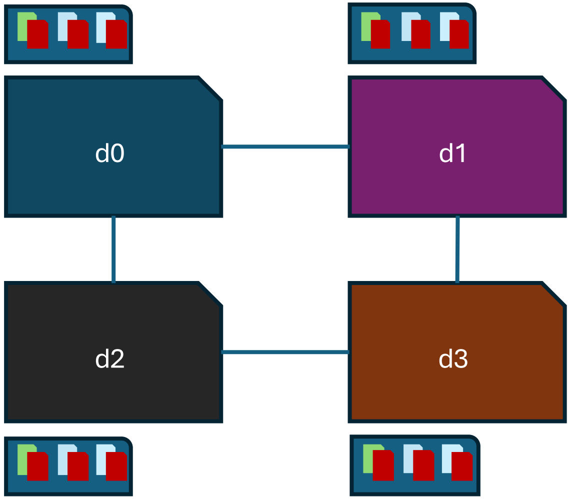




Tensor Parallelism (TP)

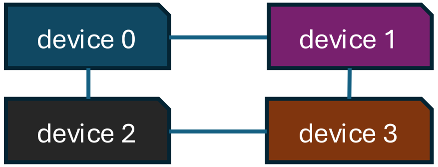


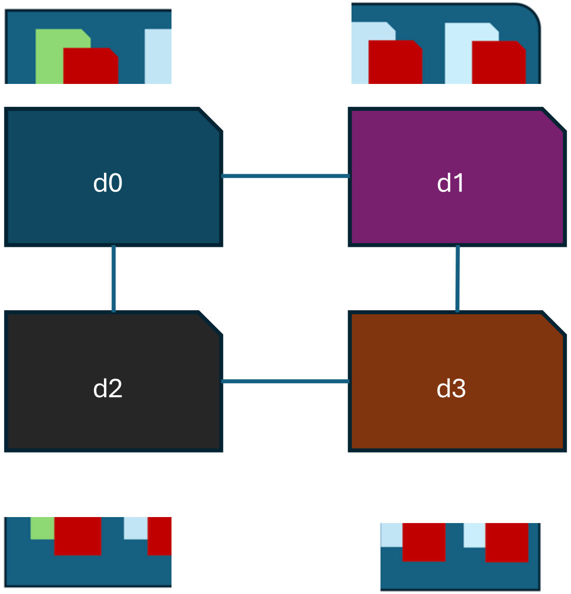
Pipeline Parallelism (PP)



Expert Parallelism (EP)

Parallelism Strategies

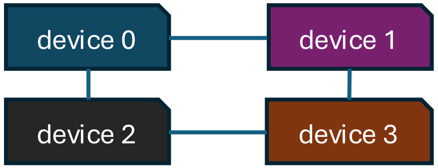




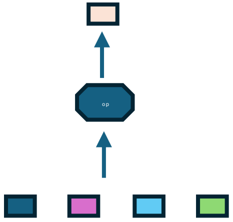
Tensor Parallelism (TP)

Let's split a pooling operation via tensor parallelism.

Parallelism Strategies



For every norm or pooling layer an AllReduce



Reduce

Tensor Parallelism (TP)

Sequentially bottlenecked

Pipeline Parallelism (PP)

Embeddings or activations might be large and need to be reduced / gathered

Expert Parallelism (EP)

What’s a Language Model?

Dimensions
How do we get to 7B, or even 450B parameters?

Let’s do DeepSeek-v3 (**671B** apparently):

Let’s assume 8 GPUs, TP=2, EP=4 and batch_size=16, bf16.

KV Cache calculation
B=16, L=61, c_kv=512, kr=64.

We deduce the size for the KV cache to behave as:

B x L x (c_kv + kr) x 2 ~ 1.07 MB / tok

Prefill is a parameter we set when serving: S = 1024 yields about 1GB.

Number of collective calls:
Attn = 1 allreduce; FFN = 1 allreduce; MoE FF: 8 + 1 experts, dispatch + combine, 58x2 (up & down) all-to-all

d=7168, E=8 hence per GPU we send

108 x 7168 x 2 x (1 + 1) x 2 x 58 + 61 * 16 * 7168 ~ 171 MB / token / GPU

Takeaways & Glossary

Takeaways

Glossary

In the next workshop we'll see

1. ML frameworks
2. LLM inference frameworks
3. Optimizations in LLM inference frameworks
    1. KV caches and PagedAttention
    2. Finding **good** parameter sets for LLMs
    3. LLM inference arithmetic: We will learn more about what I jumped through earlier

    1. And next up: RAG, fine-tuning with RLHF and LoRA

More todos

1. exact annotation of LLMs
2. LLM mechanics deeper
3. diffusion models in general but quick (images + VLM + ...)
4. hybrid models?

stepping stone AG
Wasserwerkgasse 7
CH-3011 Bern

Telefon: +41 31 332 53 63
www.stepping-stone.ch
info@stepping-stone.ch