Training is parallel. So why does an LLM write one token at a time?

GenAI in 50 posts · Post 2/50

Swipe →

01

The transformer in one line

Tokens → vectors + position → repeated attention and feed-forward blocks → next-token scores.

EXAMPLE

"I love" → 2 token vectors → a stack of blocks (32 in some models) refines them → scores for every token → " you" wins.

02

Step 1: tokens become vectors

An embedding table holds one learned vector per vocabulary entry.

EXAMPLE

50,000 tokens × width 768 = 38.4 million numbers. Looking up a token = reading one row.

03

Step 2: add position

Attention alone has no sense of order, so position information is added.

EXAMPLE

"Dog bites man" vs "man bites dog": same words, different events. Position tells them apart (e.g. sinusoidal vectors or RoPE).

04

Step 3: attention mixes tokens

Each position scores the others and takes a weighted mix of their information.

EXAMPLE

"The laptop would not start because its battery was empty." Building "its", attention can draw on "laptop".

05

Step 4: feed-forward per token

A small MLP transforms each position on its own, with the same weights everywhere.

EXAMPLE

Width 768 → 3,072 → 768: a common 4x expansion. Many parameters live here.

06

Who does what

- Attention moves information between tokens
- Feed-forward processes information within each token

07

Step 5: residuals keep it trainable

Each sublayer's input is added back to its output. Normalization keeps values stable.

```
def block(x):
    x = x + attention(norm1(x))
    x = x + feed_forward(norm2(x))
    return x
```

08

Three families

- Encoder-only: sees the whole input
- Decoder-only: sees earlier tokens
- Encoder-decoder: reads a source, writes a target

EXAMPLE

Document classifier → encoder. Chat assistant → decoder. Translation → encoder-decoder.

09

The causal mask

A decoder may only look at earlier tokens. 1 = can see:

```
token 1: 1 0 0 0
token 2: 1 1 0 0
token 3: 1 1 1 0
token 4: 1 1 1 1
```

10

Why training is parallel

During training the whole target sequence already exists. The mask hides the future.

EXAMPLE

A 1,000-token training sequence gives 1,000 next-token predictions in one forward pass.

11

Why generation is sequential

At inference the next token doesn't exist yet. It must be chosen before it can be used.

EXAMPLE

A 300-token answer needs 300 decoding steps, each depending on the last. That's why answers stream in.

12

Why it replaced RNNs

RNNs pass information step by step through a hidden state. Attention connects positions directly.

EXAMPLE

Word 1 matters for word 500: an RNN carries it through 499 updates. Attention links them in one step.

13

Attention has a cost

Full attention compares every pair of positions, so work grows with length squared.

EXAMPLE

2,000 tokens → 4M pair scores per head, per layer.

4,000 tokens → 16M. Double the input, 4x the work.

14

Architecture ≠ checkpoint

The architecture defines the computation.
The checkpoint holds the learned weights.

EXAMPLE

Two models with the same architecture can behave completely differently. Evaluate the checkpoint, not the label.

15

Common misconceptions

- Parallel training ≠ parallel generation
- Attention weights aren't a full explanation
- Same architecture ≠ same model
- Families are typical designs, not hard limits

16

Interview questions

- Why did attention replace RNNs? Direct links + parallel training
- Why residuals? A path for gradients around each sublayer
- Why is generation slower? Tokens must be produced one by one

Full tutorial in the comments. Follow for Post 3/50: Attention

Ali Hassan

linkedin.com/in/ali-hassan-here