Inside GPT-2

How a sequence of numbers becomes a prediction for the next token.

A clear machine worth understanding

GPT-2 is a compact place to learn the machinery that still appears in much larger language models. This guide follows one token through that machinery, then shows what GPT-OSS changes.

1. Represent
Text becomes token IDs, then vectors.

→

2. Transform
Attention and MLP blocks update those vectors.

→

3. Predict
Vocabulary scores become the next token.

The model does not start with English built into these operations. Training adjusts their numerical weights until useful patterns emerge.

READING ROUTE

02-03	Tokens, embeddings and the residual stream
04-05	Causal attention, shapes and a worked example
06-07	Parameters, learning and generation
08-10	GPT-OSS, smaller students and long context
11-12	Glossary, self-check and further reading

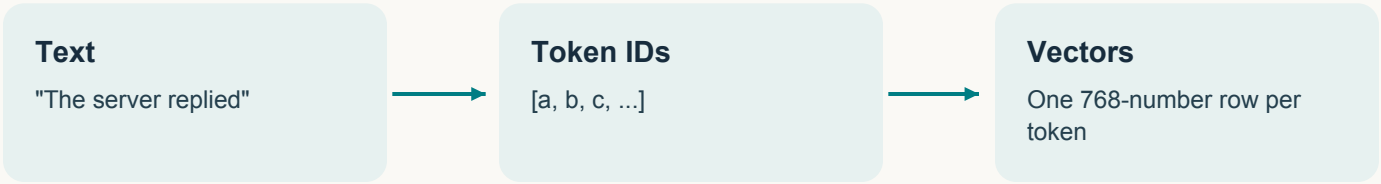
How to use this companion
Read alongside the audiobook or work through it independently. Equations and diagrams are original teaching aids. The small numerical example is illustrative, not a recovered GPT-2 computation.

From text to a vector

A token is a vocabulary item. A vector is the model's working representation.

Tokenization comes before the neural network

GPT-2 uses byte-level byte-pair encoding: text is represented with bytes, and frequent byte sequences are merged into vocabulary entries. A token may correspond to a word, a piece of a word, punctuation, or spacing. Token IDs are labels, not measurements of meaning. [1, 2]



The letters a, b and c above stand for IDs; the drawing does not assert this text has three GPT-2 tokens.

Two lookups, one sum

For token ID t at position i, the starting vector is:
$\mathbf{x_i = E[t] + P[i]}$
E is the learned token embedding table. P is the learned position table. The position vector lets identical tokens in different locations enter the network differently.

Quantity	GPT-2 small	Meaning
Vocabulary	50,257 entries	How many token IDs exist
Context positions	1,024	Learned position-table length
Model width	768	Numbers in each token vector
Sequence tensor	B x T x 768	Batch, tokens, features

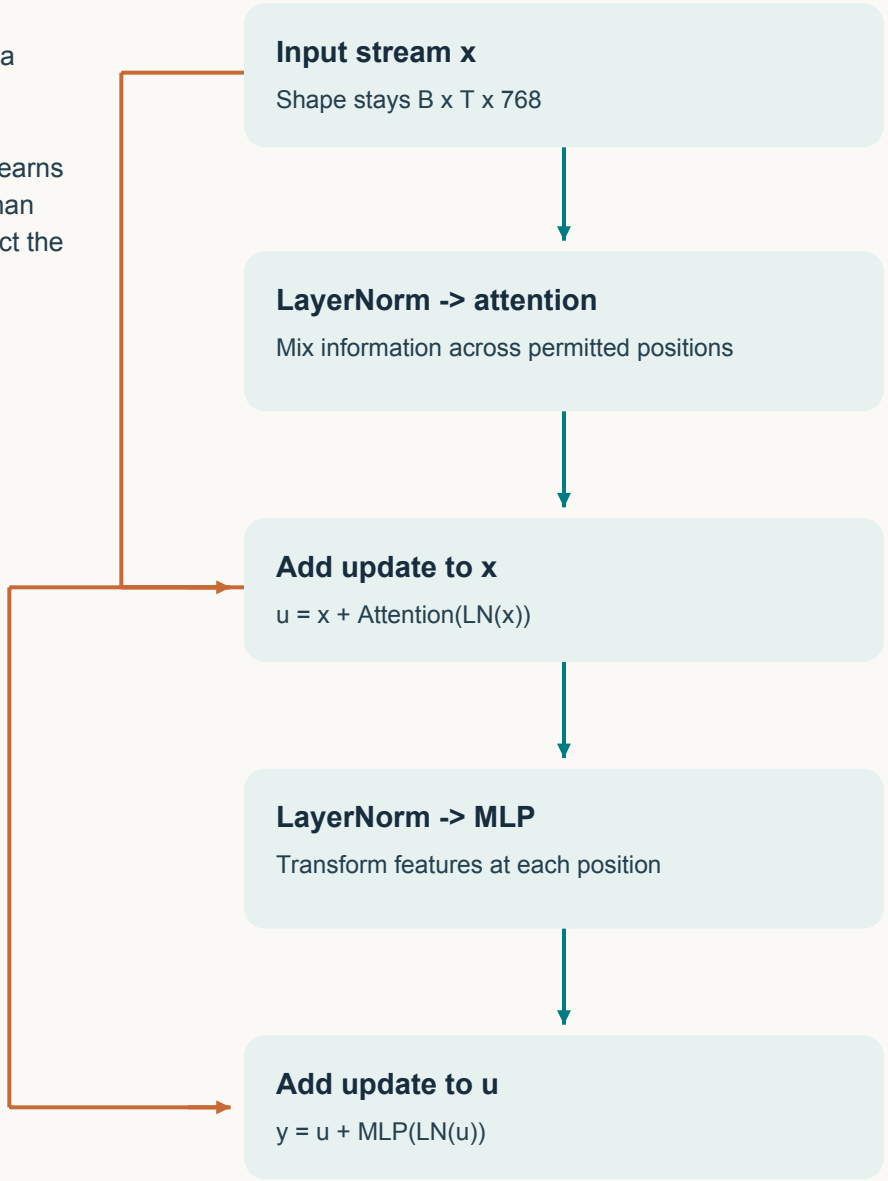
A width of 768 does not mean 768 named concepts. Individual coordinates participate in distributed representations, and their role can change across layers. [1, 3]

Keep a working stream

Twelve blocks repeatedly add updates to a 768-wide residual stream.

RESIDUAL PATH

Additions preserve a direct route for information and gradients. A block learns an update, rather than having to reconstruct the entire stream.



GPT-2 is **pre-norm**: normalization happens before each branch. After the final block, a final LayerNorm precedes the vocabulary projection. [1]

Look back, never ahead

Each position may use itself and earlier positions to predict the next token.

A projection creates queries, keys and values. Queries and keys determine mixing weights; values carry the information being mixed. These are learned vectors, not literal questions, labels and answers. [1, 4]

Tensor in one block	Shape
Input X	B x T x 768
Q, K, V after head split	B x 12 x T x 64 each
Attention score matrix	B x 12 x T x T
Merged heads and output projection	B x T x 768

CAUSAL MASK: ROW = QUERY, COLUMN = KEY



Row 3 can use keys 1, 2 and 3. Key 4 is hidden. A very large negative score is applied before softmax, making its attention weight effectively zero.

$\text{Attention}(Q, K, V) = \text{softmax}(QK^T / \sqrt{64} + \text{mask})V$

Softmax acts across the visible keys for each query. The 12 heads provide 12 learned mixing patterns; concatenation and an output projection bring their results back to width 768. Head roles are not fixed or guaranteed to be human-interpretable.

Attention, without mystery

A second illustrative example, extending the audiobook with query-key arithmetic.

Imagine the current query q = [1, 0]. Three visible keys are k1 = [1, 0], k2 = [0, 1], and k3 = [1, 1]. This toy head has dimension 2, so scale dot products by sqrt(2).

Visible position	q dot k	Scaled score	Softmax weight
1	1	0.7071	0.4011
2	0	0.0000	0.1978
3	1	0.7071	0.4011

For example: exp(0.7071) / (exp(0.7071) + exp(0) + exp(0.7071)) = 0.4011. The three weights sum to 1, allowing rounding.

Now mix the values

Let v1 = [2, 0], v2 = [0, 2], and v3 = [2, 2]. The output is their weighted sum:

0.4011[2, 0] + 0.1978[0, 2] + 0.4011[2, 2]
= [1.6044, 1.1978]

The keys helped decide where to look. The values supplied what was retrieved. A future key would be masked before the weights were normalized, so it would contribute nothing.

What the example leaves out: actual GPT-2 learns its Q, K and V projections, uses 64 dimensions per head and 12 heads, and applies an output projection. An attention weight is a mixing coefficient, not proof of a faithful explanation or a reasoning step.

Where the parameters live

The MLP is large because it expands each token vector, then projects it back.

MLP(x) = GELU(xW1 + b1)W2 + b2
In GPT-2 small, the path is 768 -> 3,072 -> 768. It applies the same learned transformation separately at each position. GELU supplies a smooth nonlinearity; without a nonlinearity, two linear maps would collapse to one. [1, 5]

768 features
Input representation

→

3,072 features
Expand + GELU

→

768 features
Project back

EXACT GPT-2 SMALL COUNT, WITH TIED OUTPUT EMBEDDINGS

Component	Calculation	Parameters
Token embeddings	50,257 x 768	38,597,376
Position embeddings	1,024 x 768	786,432
Attention in one block	4d squared + 4d	2,362,368
MLP in one block	8d squared + 5d	4,722,432
Two norms in one block	4d	3,072
12 complete blocks	12 x 7,087,872	85,054,464
Final LayerNorm	2d	1,536
Total (d = 768)	Embeddings + blocks + final norm	124,439,808

The attention, MLP and norm rows describe one block and are included in the 12-block subtotal. The vocabulary output uses the token embedding matrix again, transposed; it does not add a second 38.6-million-parameter matrix. Biases and normalization scales are included. [1, 3]

Parameters are persistent learned numbers. Activations are temporary values created while processing an input. Both occupy memory, but they are different things.

One objective, two workflows

Training learns the weights. Inference uses the weights to continue a sequence.

TRAINING: MANY POSITIONS AT ONCE

Input positions	The	server	returned
Targets	server	returned	200

The target at each position is the next token. This word-level drawing is conceptual; actual token boundaries may differ. A causal mask prevents a position from seeing its target. Cross-entropy penalizes low probability on the observed next token; backpropagation and an optimizer update the weights.

Loss = average of -log p(observed next token | earlier tokens)

INFERENCE: ONE NEW TOKEN, THEN REPEAT

Process the prompt, take the final position's vocabulary scores, and choose a token. Append it and run the next step. Temperature changes the distribution's sharpness; sampling and greedy selection are different ways to choose. They do not teach the model new weights. [1, 6]

Why keep a key-value cache?

Earlier positions cannot depend on future tokens, so their keys and values can be reused during generation. The cache saves recomputation. It is temporary state for this sequence, not a new permanent memory or an update to the model.

Illustrative GPT-2 small cache at 1,024 tokens, batch 1
2 x 12 layers x 1,024 tokens x 768 features x 2 bytes = 36 MiB.
This assumes a 16-bit cache and excludes weights, activations and runtime overhead.

Parallel training does not mean a single generated continuation can reveal all of its unknown future tokens at once.

What GPT-OSS changes

The decoder idea survives. Attention, normalization and feature processing evolve.

Property	GPT-2 small	GPT-OSS-20B	GPT-OSS-120B
Transformer blocks	12	24	36
Width	768	2,880	2,880
Total parameters	124.4M	About 21B	About 117B
Active per token	Dense model	About 3.6B	About 5.1B
Experts per block	No routed experts	32; select 4	128; select 4
Attention heads / KV heads	12 / 12	64 / 8	64 / 8
Position method	Learned table	RoPE + YaRN	RoPE + YaRN
Context length	1,024	131,072	131,072

Rounded GPT-OSS counts follow OpenAI. Active counts describe the computation path for a token; they are not the amount of model storage required. Configurations and implementation support the other entries. [3, 7, 8, 9]

Four changes to recognize

Grouped-query attention. Several query heads share key and value heads, reducing KV-cache size relative to equally many independent KV heads.

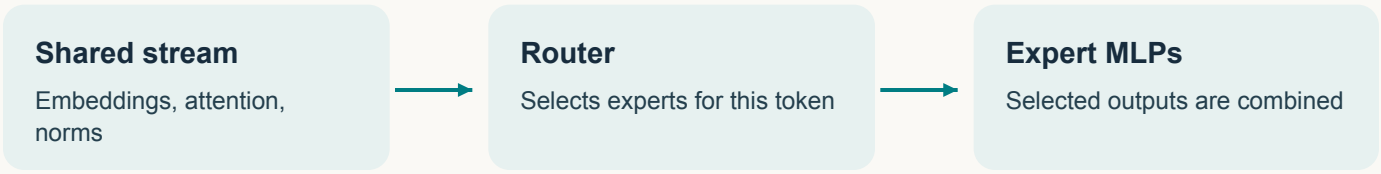
Alternating attention. GPT-OSS alternates 128-token sliding attention and full attention. The first configured block is sliding.

RMSNorm and expert MLPs. RMSNorm replaces LayerNorm; routed expert branches use a modified SwiGLU activation.

MXFP4 expert weights. The released models store expert weights in a microscaled 4-bit format. This changes representation and memory cost; it does not remove parameters.

A smaller model is a new system

An expert is a feed-forward branch inside a larger network, not a complete chatbot.



In GPT-OSS, each block selects 4 experts for each token. Selection can change from token to token and layer to layer. An expert does not include the whole vocabulary system, attention stack or output head. Keeping one expert per layer also changes the computation the original model learned. [7, 8, 9]

Three operations people often conflate

Pretraining from scratch
Initialize a student's weights and optimize next-token prediction over a large corpus. Copying an architecture copies a design, not the language abilities learned by a trained model.

Continued pretraining
Start from an existing checkpoint and keep learning from new text. Much of the existing representation is available at the beginning, although poor data or aggressive updates can damage it.

Distillation
Use a teacher to supply learning targets: generated examples, probability distributions, or selected internal signals. The student still has to be trained. A randomly initialized student needs broad learning, even if some supervision comes from a teacher. [10]

Practical implication
Moving from 20B to 1-2B is not simply choosing a smaller save format. It means choosing a student architecture and training recipe, then measuring what survives. Distillation can help, but cannot guarantee teacher-level breadth or reasoning.

Keep four numbers separate

Parameter count, vocabulary, context length and training tokens answer different questions.

Number	What it tells you	What it does not establish
Parameters	Learned numerical capacity	Quality at a specific task
Vocabulary size	How many token IDs exist	How many words the model knows
Context length	Input positions supported	Reliable use of all those positions
Training tokens	Amount of text processed	Unique, useful or high-quality data

Why 256K is more than a configuration value

GPT-2's original learned position table has 1,024 rows. A much longer window needs positional treatment, training and evaluation appropriate to the new length. It also changes the workload: full attention compares pairs of positions, while cached generation stores history-dependent state.

An idealized scaling comparison
Going from 1,024 to 262,144 tokens is 256 times longer. A fully materialized attention score matrix contains 65,536 times as many entries. Memory-efficient kernels can avoid storing that full matrix; full attention still has quadratic pairwise work.

A successful long-context evaluation should test retrieval, several relevant facts scattered through a document, distractors, and tasks requiring those facts to be combined. Fitting the input in memory is only one requirement.

For a smaller experimental model: judge language quality and task success on held-out data; measure actual memory and throughput; then increase sequence length deliberately. A 1-2B label does not by itself specify a feasible training schedule or useful 256K behavior.

A working glossary

Use these terms to describe what the model is actually doing.

Token	A discrete vocabulary item produced by the tokenizer; not necessarily a word.
Embedding	A learned vector retrieved using a token ID or position index.
Residual stream	The sequence of vectors carried through the model and updated by additions.
Attention head	A learned query-key-value mixing operation over allowed positions.
Causal mask	A restriction that hides later positions from an earlier position.
LayerNorm	Normalization across a token's features, with learned scale and offset.
MLP	The position-wise nonlinear feature transformation inside a block.
Logits	Unnormalized vocabulary scores, before softmax turns them into probabilities.
Cross-entropy	A loss that rewards probability assigned to the observed target token.
KV cache	Stored keys and values from earlier positions, reused during generation.
MoE router	A learned selector that assigns token representations to expert branches.
Quantization	Representation of numerical weights or activations with reduced precision.
Distillation	Training a student with supervisory information supplied by a teacher.

Test your mental model

Explain these answers in your own words, then inspect the implementation.

Why can training process several positions together? Known targets and a causal mask let each position predict its next token without seeing it.

Why is the output matrix not counted twice? GPT-2 ties the vocabulary output projection to the input token embedding table.

Does keeping one expert create a full small GPT-OSS? No. Experts are components inside a shared stack, and changing routing changes the learned computation.

Does a KV cache change the weights? No. It preserves sequence-specific intermediate state during inference.

PRIMARY REFERENCES

[1] GPT-2 implementation: blocks, attention and tied output
[2] GPT-2 byte-level BPE tokenizer
[3] GPT-2 small configuration
[4] Vaswani et al., Attention Is All You Need
[5] Hendrycks and Gimpel, Gaussian Error Linear Units
[6] GPT-2 sampling implementation
[7] OpenAI, Introducing GPT-OSS (counts and release details)
[8] Official GPT-OSS configurations: 20B and 120B
[9] GPT-OSS reference implementation
[10] Hinton et al., Distilling the Knowledge in a Neural Network
120B configuration | Original GPT-2 report | Layer Normalization

THE ARTICLES THAT INSPIRED THIS COMPANION

Sebastian Raschka: From GPT-2 to GPT-OSS
Cameron R. Wolfe: GPT-OSS from the Ground Up

Evidence note: rounded active counts here follow OpenAI (20B: 3.6B; 120B: 5.1B), correcting their reversal in one passage of Wolfe's article. Official configurations establish that GPT-OSS starts with sliding attention. This is an original tutorial, not an abridged reproduction of either article.