GPT: Generative Pre-trained Transformer

OpenAI - Autoregressive Decoder-only Architecture

Decoder-Only Architecture

Decoder Layer N

...

Decoder Layer 2

Decoder Layer 1

Causal Masking

Past → **Present** ✗ Future

Token 1 | Token 2 | Token 3 | Next →

Key Features

- → **Unidirectional** (left-to-right) attention
- Optimized for **text generation**
- **Autoregressive** prediction
- Decoder-only transformer stack

GPT Evolution

2018	**GPT-1**	117M params • Proof of concept
2019	**GPT-2**	1.5B params • Impressive generation
2020	**GPT-3**	175B params — ★ Paradigm Shift

🎓 Training Process (Pre-training)

1

Input Tokenization

Text is converted into tokens using vocabulary

```
"The cat sat" → [1045, 2635, 2068]
```

→

2

Forward Pass

Tokens pass through decoder layers with causal masking

```
Input → Embeddings → Decoder Stack → Logits
```

→

3

Predict Next Token

Model predicts probability distribution over vocabulary

$$P(\text{token}_4 \mid \text{token}_1, \text{token}_2, \text{token}_3)$$

→

4

Compute Loss

Compare prediction with actual next token using cross-entropy

```
Loss = CrossEntropy(predicted, actual)
```

→

5

Update Weights

Backpropagate gradients and update model parameters

$$\theta \leftarrow \theta - \alpha \nabla L(\theta)$$

🚀 Inference Process (Text Generation)

1

Input Prompt

User provides initial text prompt

```
"Once upon a time"
```

→

2

Tokenize & Encode

Convert prompt to token IDs and create embeddings

```
Tokens: [2949,
```

→

3

Autoregressive Generation

Generate one token at a time, appending each to input

```
Loop: predict → sample → append
```

→

4

Sampling Strategy

Use temperature, top-k, or top-p to select next token

```
Temperature = 0.7, Top-p = 0.9
```

→

5

Decode Output

Convert tokens back to text until stopping condition

```
Tokens → "there
```

2402, 257, 640]

lived a princess"

Step-by-Step Example

- Input: `"The weather is"`
- Step 1: `predict → "sunny"`
- Step 2: `"The weather is sunny" → predict → "today"`
- Step 3: `"The weather is sunny today" → predict → "."`
- Final Output: `"The weather is sunny today."`