3Blue1Brown — Transformers, the Tech Behind LLMs

Deep Learning, Chapter 5 — Video Outline

Video by Grant Sanderson Watch on YouTube

I. Introduction — What is a Transformer? (0:00–3:03)

- Breaks down the acronym "GPT": **G**enerative, **P**retrained, **T**ransformer
- Transformer = a specific neural network architecture underlying the current AI boom
- Transformers power text generation, speech synthesis, image generation (DALL·E, Midjourney), and translation
- The variant covered here: a **next-token prediction model** — given text, predict what comes next as a probability distribution
- Text generation works by repeatedly predicting, sampling, and appending — surprisingly effective at scale (GPT-2 vs. GPT-3 comparison)
- Chatbots use a **system prompt** to frame the prediction task as a conversation

---

II. High-Level Data Flow Through a Transformer (3:03–6:39)

- Input is broken into **tokens** (words, subwords, image patches, audio chunks)
- Each token is mapped to a **vector** encoding its meaning; similar meanings → nearby vectors
- Vectors pass through an **attention block**, where they "talk to each other" to update meanings based on context (e.g., "model" in "machine learning model" vs. "fashion model")
- Then through a **multi-layer perceptron (MLP) / feed-forward layer**, where each vector is processed independently in parallel — like asking a list of questions and updating based on answers
- This attention → MLP cycle **repeats** many times
- The final vector in the sequence is used to produce a probability distribution over all possible next tokens

---

III. The Premise of Deep Learning (7:22–12:27)

- Machine learning: using data to determine model behavior instead of hand-coding rules
- Simple example: linear regression (2 parameters); GPT-3 has 175 billion parameters
- Deep learning models scale well thanks to **backpropagation**
- Key structural constraints:
    - Input must be arrays of numbers (tensors)
    - Data is transformed layer by layer
    - Parameters interact with data only through **weighted sums** (i.e., matrix-vector multiplication)
- GPT-3's 175B weights are organized into ~28,000 matrices across 8 categories
- Sharp distinction: **weights** (learned; colored blue/red) vs. **data** (input being processed; colored gray)

IV. Word Embeddings (12:27–18:25)

- Input text is tokenized, then each token is mapped to a vector via an **embedding matrix (W_E)**
- Embedding = treating words as points in high-dimensional space (12,288 dimensions in GPT-3)
- Directions in embedding space carry **semantic meaning**:
    - woman − man ≈ queen − king
    - Italy − Germany + Hitler ≈ Mussolini
    - Germany − Japan + sushi ≈ bratwurst
- **Dot products** measure alignment between vectors — useful for testing semantic hypotheses (e.g., a "plurality direction")
- Embedding matrix: 50,257 tokens × 12,288 dimensions ≈ **617 million parameters**

V. Embeddings Beyond Words — Context and Position (18:25–20:23)

- Vectors aren't just static word meanings; they progressively **soak in context** as they flow through the network
- A vector starting as "king" may end up encoding "a Scottish king who murdered his predecessor, described in Shakespearean language"
- **Context size** limits how much text can be considered (2,048 tokens for GPT-3) — explains why early chatbots "lost the thread" in long conversations

---

VI. Unembedding — Producing the Output (20:23–22:22)

- The last vector is multiplied by an **unembedding matrix (W_U)** to produce a list of 50,000 values (one per token)
- During training, every vector in the final layer simultaneously predicts what comes after it (not just the last one)
- Unembedding matrix adds another **~617 million parameters**, bringing the running total to just over 1 billion

---

VII. Softmax with Temperature (22:22–26:03)

- **Softmax** converts raw output values ("logits") into a valid probability distribution: raise *e* to each value, then normalize
- Largest values dominate but it's "softer" than hard max — nearby values get meaningful weight
- **Temperature (T)**: a parameter that controls randomness in sampling
  - T = 0 → always picks the most probable word (deterministic, predictable)
  - High T → more uniform distribution (creative but risks nonsense)
- Demo: temperature 0 produces a generic Goldilocks story; higher temperature starts originally but degenerates

---

VIII. Preview of What's Next (26:03–end)

- This chapter laid foundations: word embeddings, softmax, dot products as similarity measures, and matrix multiplication as the universal operation
- These are the prerequisites for understanding the **attention mechanism** — the cornerstone of transformers — covered in the next chapter