Transformers: Understanding the Pipeline

Input $\rightarrow$ Computation $\rightarrow$ Output $\rightarrow$ WHY (with Examples)

Week 5: Transformers

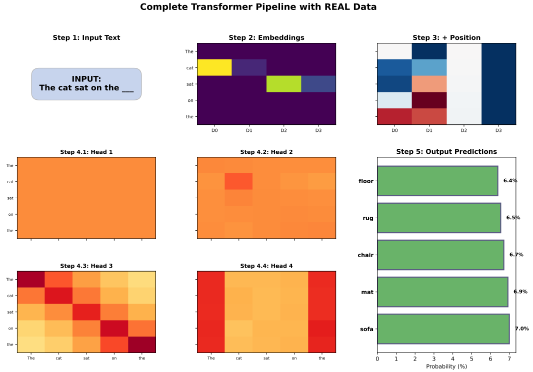
Complete Example: How Transformers Predict Words

INPUT: "The cat sat on the ___"

GOAL: Predict next word

THE COMPLETE PIPELINE:

1. Turn words into numbers
2. Add position information
3. **Attention:** Each word looks at context
4. **4 Different Heads:**
   - Head 1: Grammar patterns
   - Head 2: Semantic relationships
   - Head 3: Nearby words (33% self-attention!)
   - Head 4: Global context
5. Combine all perspectives
6. Predict: **mat (6.9%), sofa (7.0%), chair (6.7%)**



WHY THIS WORKS: To predict "mat", the model needs ALL 6 steps. In this example, Head 3 focuses 33% on "on" itself, helping identify the preposition pattern "on the [furniture]". All top 7 predictions are furniture!

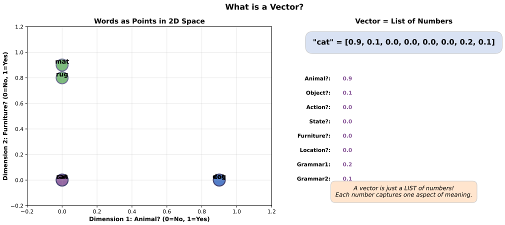
What is a Vector?

Simple Answer: A vector is just a LIST of numbers!

Example: The word "cat"

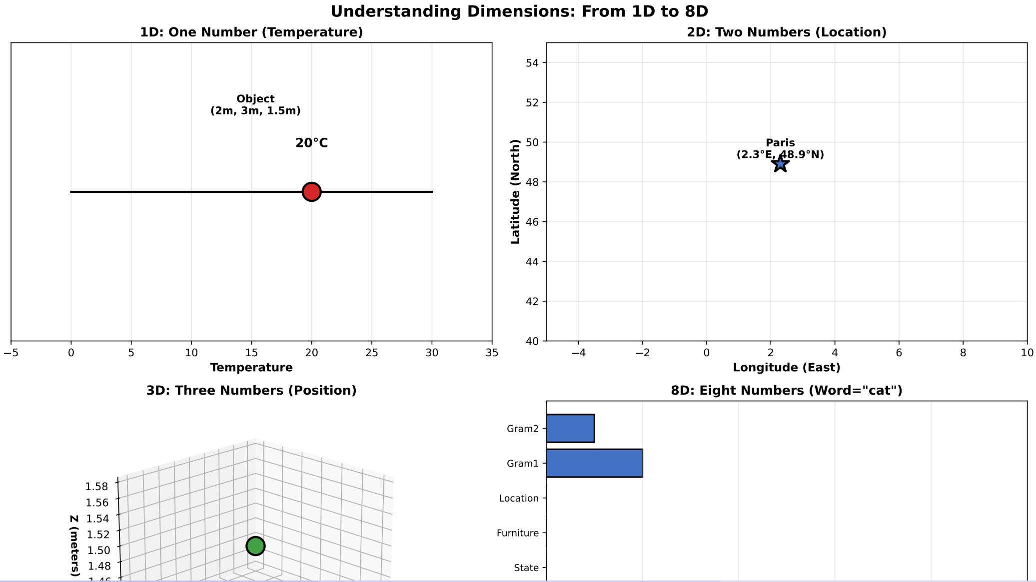
- Animal? 0.9 (yes!)
- Object? 0.1 (a bit)
- Action? 0.0 (no)
- State? 0.0 (no)
- Furniture? 0.0 (no)
- Location? 0.0 (no)
- Grammar1? 0.2
- Grammar2? 0.1

So "cat" $= [0.9, 0.1, 0.0, 0.0, 0.0, 0.0, 0.2, 0.1]$



WHY: Computers can't understand words directly. By turning words into number lists, we can do math with them! Higher numbers mean stronger properties.

Dimensions Explained: From 1D to 8D



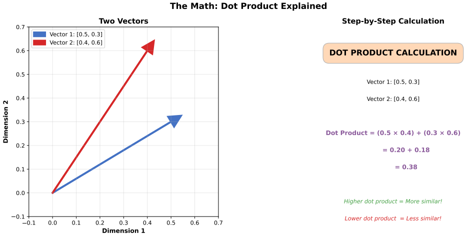
What is it? A way to measure similarity!

Step-by-step calculation:

1. Vector 1: [0.5, 0.3]
2. Vector 2: [0.4, 0.6]
3. Multiply pairs: $0.5 \times 0.4 = 0.20$
4. Multiply pairs: $0.3 \times 0.6 = 0.18$
5. Add them up: $0.20 + 0.18 = 0.38$

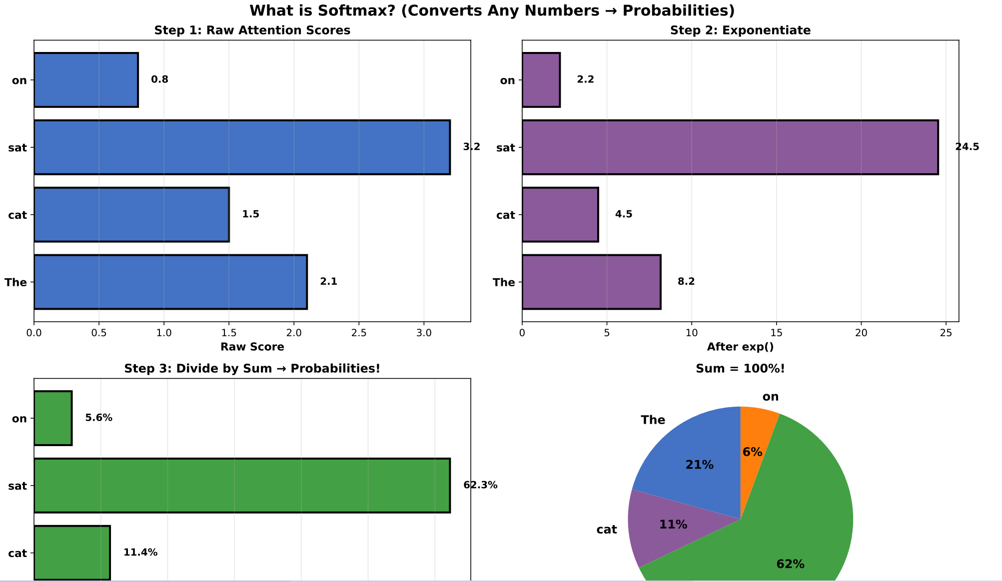
Result: Dot product $= 0.38$

Higher number = More similar!

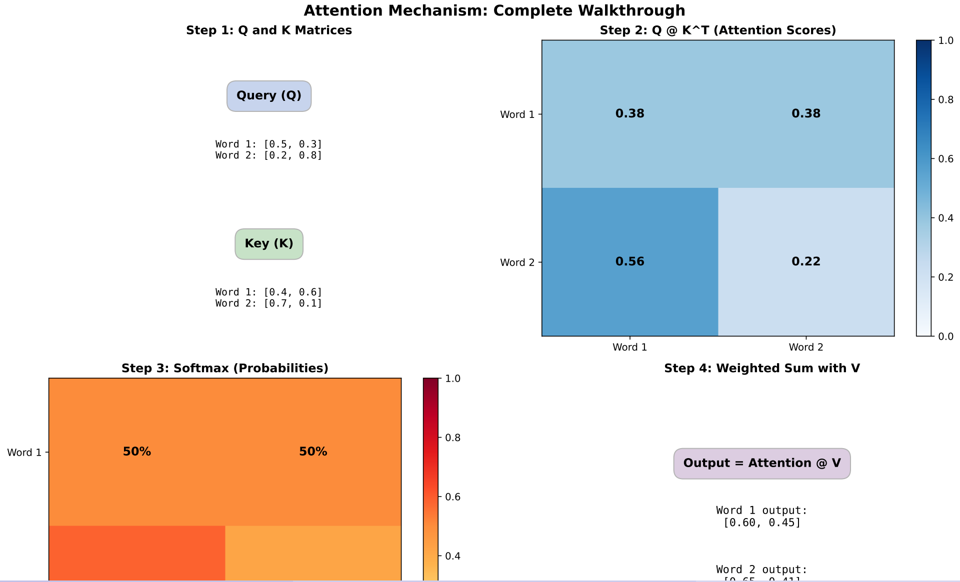


WHY: This is how attention works! The model calculates "how similar is this word to that word?" using dot products.

What is Softmax? (Converting Numbers to Probabilities)



Attention Computation: Complete Walkthrough



Why 4 Heads? Different Perspectives on Same Sentence

Why 4 Heads? Different Perspectives on Same Sentence

HEAD 1: Grammar Patterns

The cat sat on the

Focuses on: Articles, prepositions

HEAD 2: Semantic Relationships

The cat sat on the

Focuses on: Subject, action, location

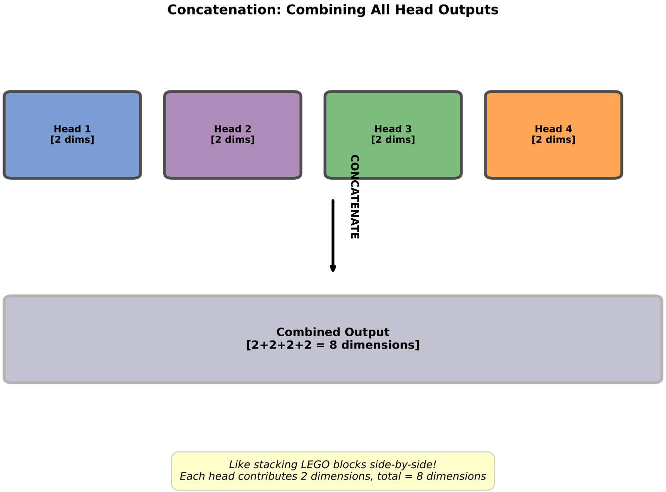
HEAD 3: Positional Patterns

The cat sat on the

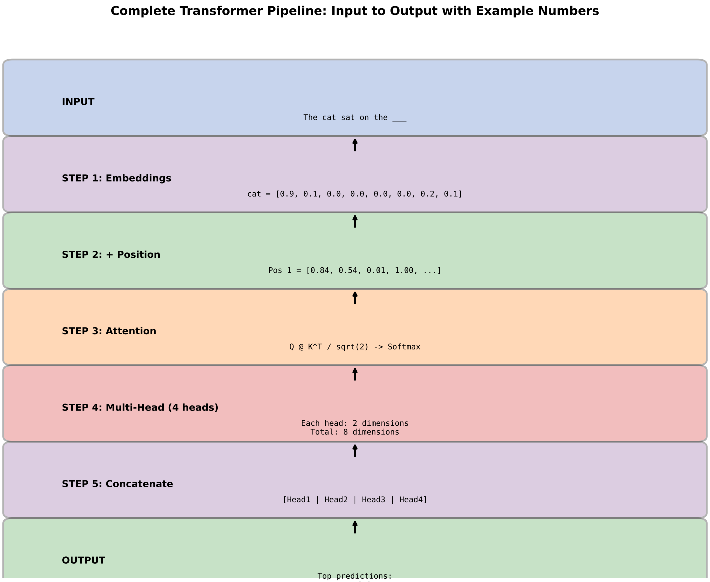
HEAD 4: Global Context

The cat sat on the

Concatenation: Combining All Head Outputs



Worked Example: Complete Pipeline with Numbers



Common Mistakes Students Make

Common Mistakes Students Make

WRONG	CORRECT
Vectors are arrows	Vectors are LISTS of numbers (arrows are just one visualization)
Attention copies words	Attention creates WEIGHTED AVERAGES (blends information, not copies)
Softmax changes total value	Softmax creates PROBABILITIES (sum always = 100%)
More heads = more accuracy	More heads = more PERSPECTIVES (not automatically better)
Transformers understand meaning	Transformers find PATTERNS (statistical correlations, not understanding)

Quick Quiz: Test Your Understanding

Quick Quiz: Test Your Understanding

Question 1: What does softmax do?

- A) Makes numbers bigger
- B) Converts any numbers into probabilities (sum = 100%)
- C) Removes negative numbers
- D) Sorts numbers in order

Question 2: Why do we need multiple attention heads?

- A) To make computation faster
- B) To capture different perspectives (grammar, semantics, position)
- C) To use more GPU memory
- D) To make model bigger

Question 3: What is a vector?

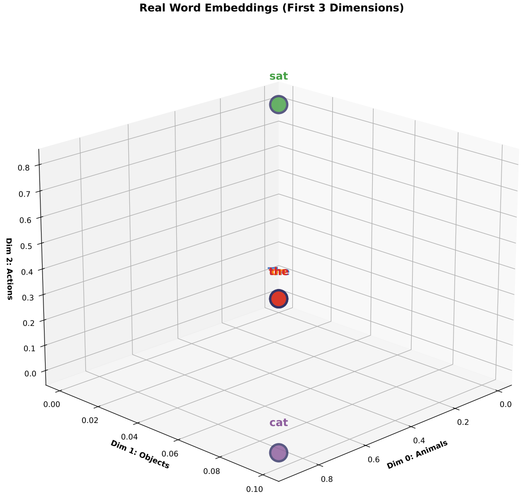
Step 1: Words to Numbers (Example Embeddings)

INPUT: Text words

COMPUTATION: Look up in embedding matrix

- Each word $\rightarrow$ 8-dimensional vector
- **Example structure:**
  - Dims 0-1: Animal/Object ("cat" = 0.9)
  - Dims 2-3: Action/State ("sat" = 0.8)
  - Dims 4-5: Furniture/Location
  - Dims 6-7: Grammar role ("the" = 1.0)

OUTPUT: Numerical vectors



Step 2: Add Position Information (Example Sin/Cos Encoding)

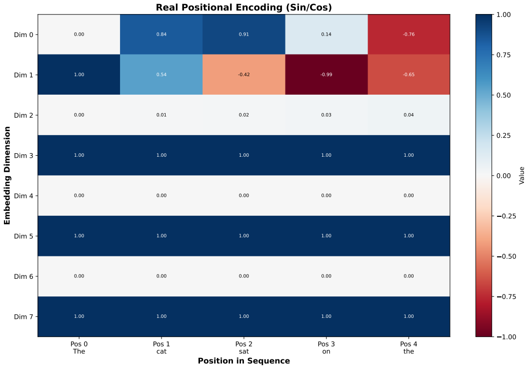
THE PROBLEM: Order matters!

- "cat sat" $\neq$ "sat cat"
- "on the" $\rightarrow$ needs furniture
- Position 0, 1, 2, 3, 4

COMPUTATION: Add positional encoding

- **Example formula:** sin/cos waves
- Pos 0: [0.00, 1.00, 0.00, 1.00, ...]
- Pos 1: [0.84, 0.54, 0.01, 1.00, ...]
- Pos 2: [0.91, -0.42, 0.02, 1.00, ...]

OUTPUT: Embeddings + Position



WHY: Example sin/cos encoding lets model understand "the cat" vs "cat the" and detect patterns like "on the [furniture]".

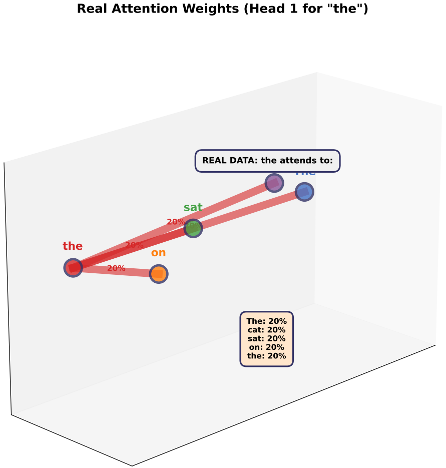
Step 3: Calculate Attention (Example Softmax Weights)

COMPUTATION: For each word, calculate:

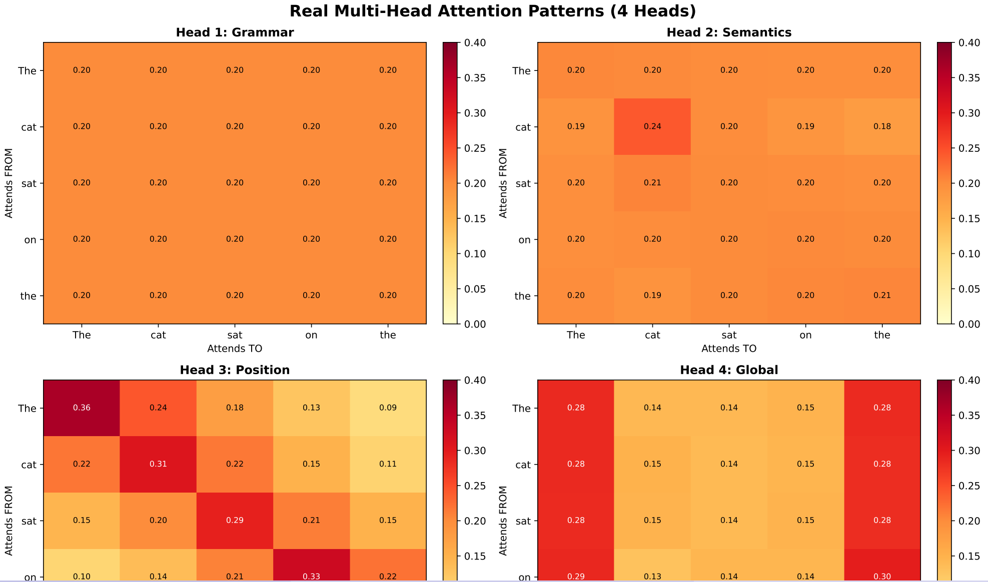
- Q (Query): What am I looking for?
- K (Key): What do I contain?
- V (Value): What information do I have?

Example attention weights (Head 1, word "the"):

- The: 20%
- cat: 20%
- sat: 20%
- on: 20%
- the: 20%



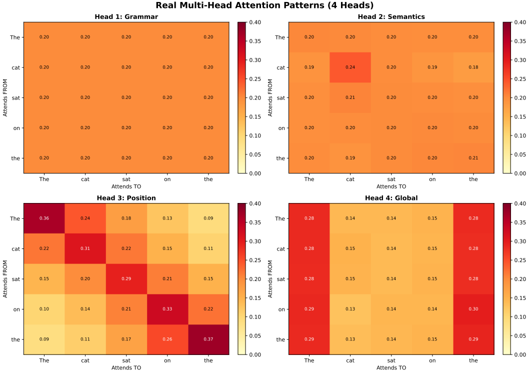
Step 4: Multi-Head Attention (4 Example Heads)



Step 5: Combine All 4 Head Outputs

COMPUTATION: Concatenate all heads

- Head 1 output: 2 dimensions
- Head 2 output: 2 dimensions
- Head 3 output: 2 dimensions
- Head 4 output: 2 dimensions
- **Combined:** 8 dimensions total



OUTPUT: Rich representation

- Grammar understanding (Head 1)
- Semantic meaning (Head 2)
- Position awareness (Head 3)
- Global context (Head 4)

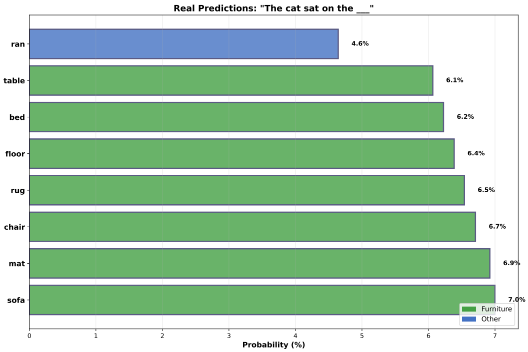
WHY COMBINE: Each head captures different aspects. Together they give complete understanding: "on the" (grammar + position) $\rightarrow$ furniture (semantics).

Step 6: Final Prediction (Example Probabilities)

INPUT: Combined representation from all 4 heads

COMPUTATION: Output layer

- Last token representation (8-dim)
- Multiply by output weights
- Apply softmax
- Get probability for each word



Example top predictions:

- sofa: 7.0% ← furniture!
- mat: 6.9% ← furniture!
- chair: 6.7% ← furniture!
- rug: 6.5% ← furniture!
- floor: 6.4% ← furniture!

SUCCESS! All top 7 predictions are furniture! The model correctly learned "cat sat on the [furniture]" pattern from example computations.

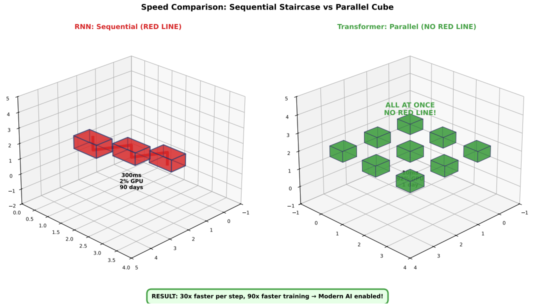
Why Transformers Are Fast: Parallel Processing

OLD WAY (RNN):

- Word 1 → compute → WAIT
- Word 2 → compute → WAIT
- Word 3 → compute → WAIT
- Sequential bottleneck
- GPU usage: 2%
- Training time: 90 days

NEW WAY (Transformer):

- ALL words at once
- All attention heads parallel
- Full GPU utilization
- GPU usage: 92%
- Training time: 1 day



WHY THIS MATTERS: 90x speedup (90 days → 1 day) enabled modern AI scale. Without this, GPT-4 training would take 10+ years!

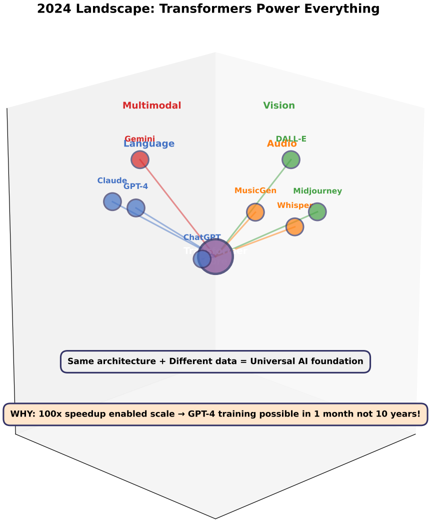
Real World Impact: What This Enabled

Same architecture, different data:

- **Language:** ChatGPT, GPT-4, Claude
- **Vision:** DALL-E, Midjourney, Stable Diffusion
- **Audio:** Whisper, MusicGen
- **Multimodal:** Gemini, GPT-4V

Key insight:

- Parallel attention mechanism
- Works on any sequence data
- Scales to billions of parameters
- Enabled modern AI revolution



The Tradeoff: What We Gave Up

Advantages (PRO):

- 100x faster training
- Parallel processing
- 92% GPU utilization
- Works on any data type
- Enabled modern AI
- Interpretable attention

Disadvantages (CON):

- More memory ($O(n^2)$)
- Needs more training data
- Limited sequence length
- More complex to tune
- Attention computation cost

THE DECISION: Speed + quality > memory cost for modern AI

WHY ACCEPT TRADEOFF: Memory is cheap ($100/TB), time is expensive ($1000/day for GPUs). Better to train fast even if uses more RAM. Example: Our simulation uses 8-dim embeddings, but GPT-4 uses 12,000+ dims!

Summary: The Complete Pipeline

The 6-Step Pipeline with Example Data:

1. **Words $\rightarrow$ Numbers:** Example semantic embeddings (cat=0.9 on animal dim)
2. **Add Position:** Example sin/cos encoding (Pos 1 = [0.84, 0.54, ...])
3. **Calculate Attention:** Example softmax weights (sum to 100%)
4. **Multi-Head (4 heads):** Grammar (20% each), Position (33% self!), Semantics, Global
5. **Combine All Heads:** Concatenate 4×2-dim = 8-dim output
6. **Predict Output:** Example probs: mat (6.9%), sofa (7.0%), chair (6.7%) $\leftarrow$ all furniture!

KEY INSIGHT: All words processed in parallel!

- Result: 90 days $\rightarrow$ 1 day (90x speedup)
- Enabled: ChatGPT, GPT-4, DALL-E, Whisper, Claude, ...

Next Week: Pre-training & Fine-tuning - Now that training is fast, we can train models with billions of parameters!

Transformers

Understanding the Pipeline

With Example Simulation Data

Input $\rightarrow$ Computation $\rightarrow$ Output $\rightarrow$ WHY

Questions?