BasisTech

Studio for startups

Chris Biow
Technologist in Residence
25 March 2024

BasisTech

Under the Hood of a Large Language Model

A visual exploration,
requiring only basic arithmetic

Based on Brendan Bycroft's
LLM Visualization
https://bbycroft.net/llm

BasisTech

SwiftOnSecurity
@SwiftOnSecurity

One time I tried to explain Kerberos to someone. Then we both didn't understand it.

13:00 · 11/21/14

Why Taylor Swift can't authenticate

The Why
of Why Things Work

Why should we understand our underlying tech?

- Sheer curiosity
- Comprehend the layers of abstraction
  - Abstraction enables simplified reasoning
  - Simplification loses detail
    - Insight into capabilities and behavior
    - Anticipate risks
  - Lower layers don't matter until they do
    - Ability to analyze and comprehend what goes wrong
    - Understand reasons for costs and performance

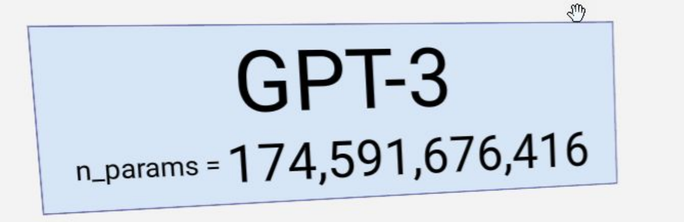
Our layers of abstraction for LLMs

- LLM
- Processing steps
  - Embedding
  - Transformer
  - Normalization
- ~~Training~~
  - ~~Gradient descent, hyperparameters, convergence, grokking~~
  - ~~Math: calculus, statistics, linear algebra~~
  - Data: just text, unsupervised
- Inferencing
  - Math: addition, multiplication (square root, logarithm/exponent)
  - Data structures: tables; columns and rows
  - Toy problem
    - reverse-sort tokens in a vocabulary consisting of letters A, B, C
    - C B A B B C ➔ C C B B B A

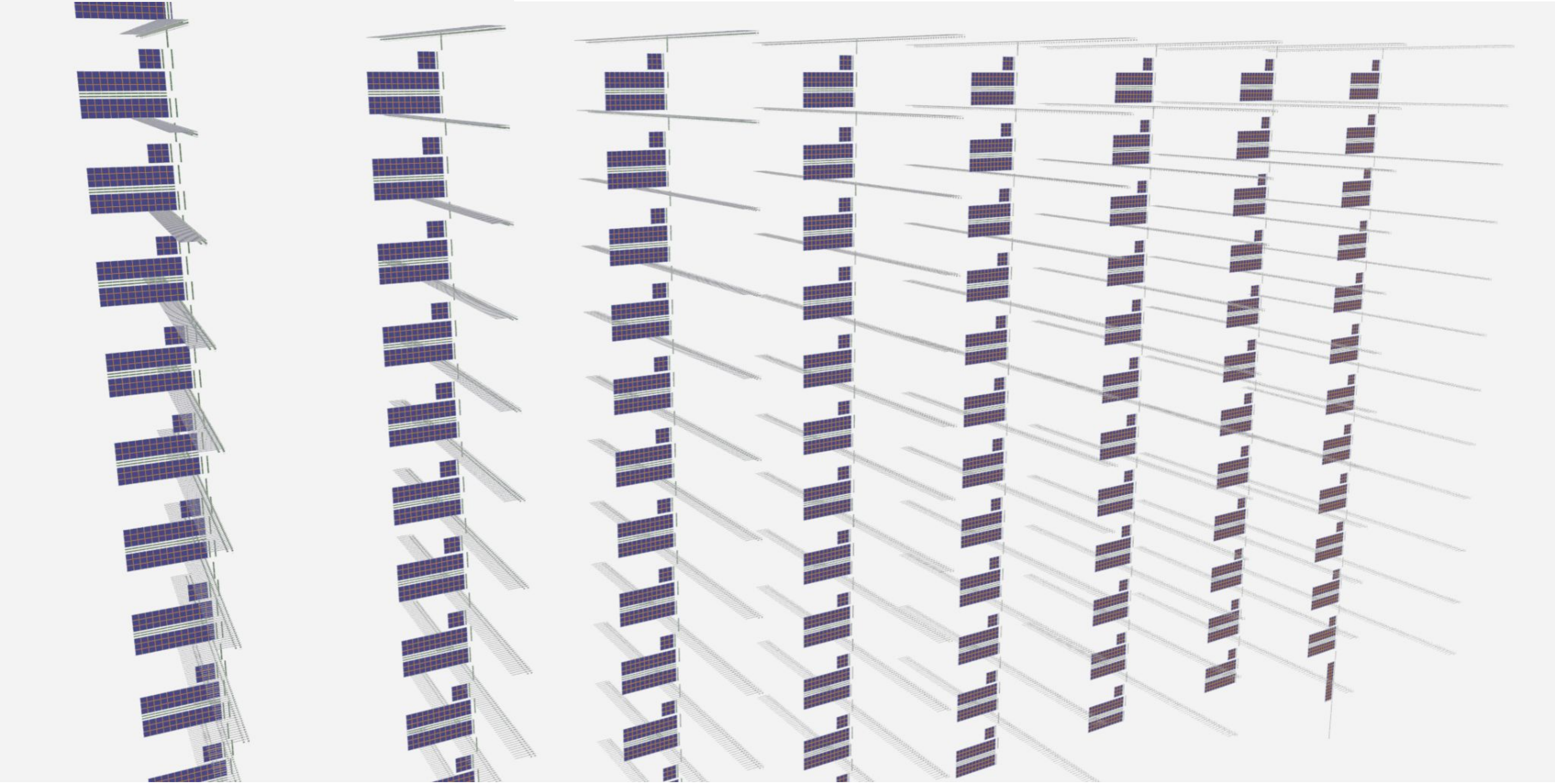
Abstraction Layer 0:
LLMs
by parameter size,
visualizing structural complexity

GPT-4

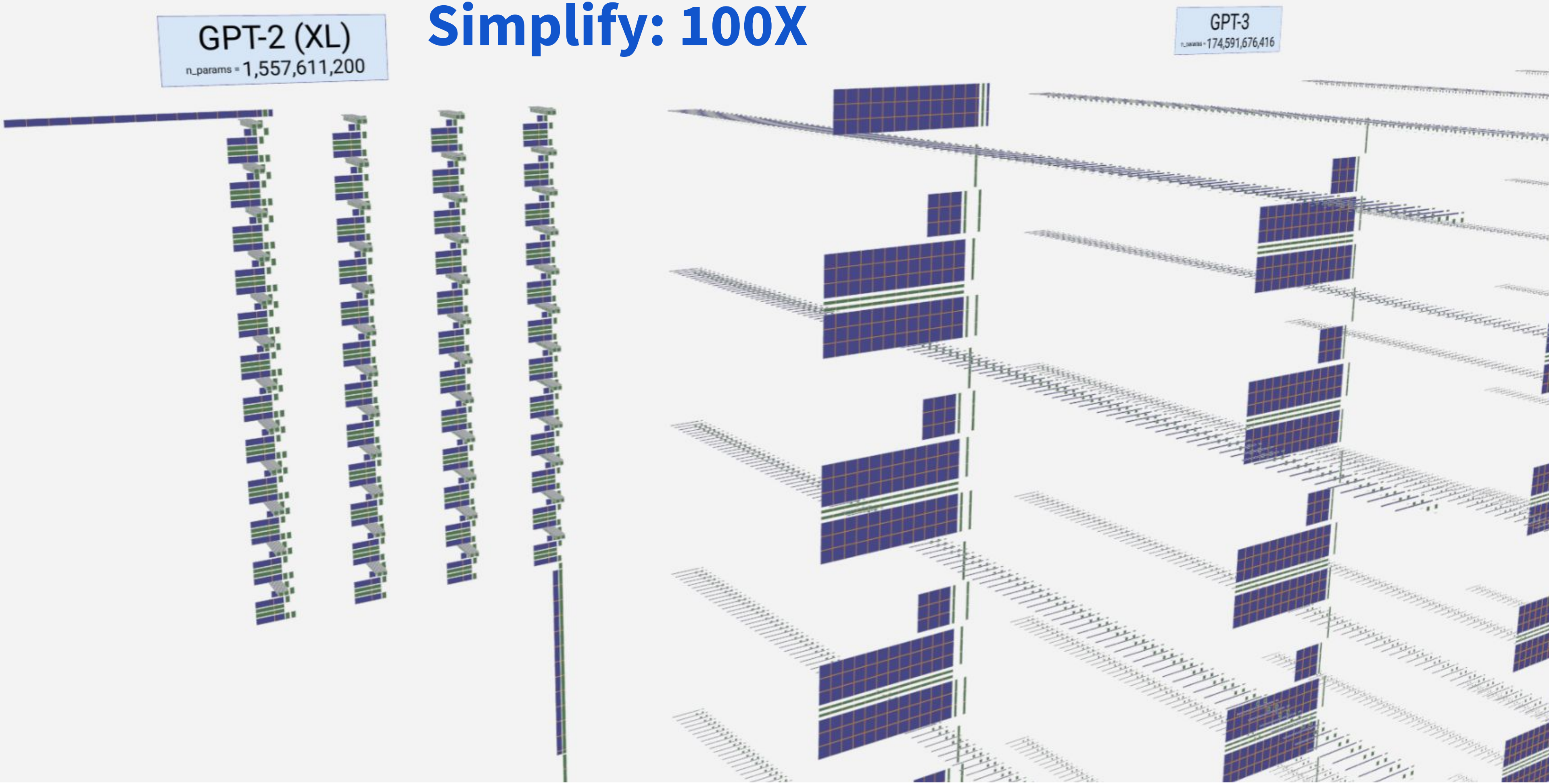
1,760 billion parameters



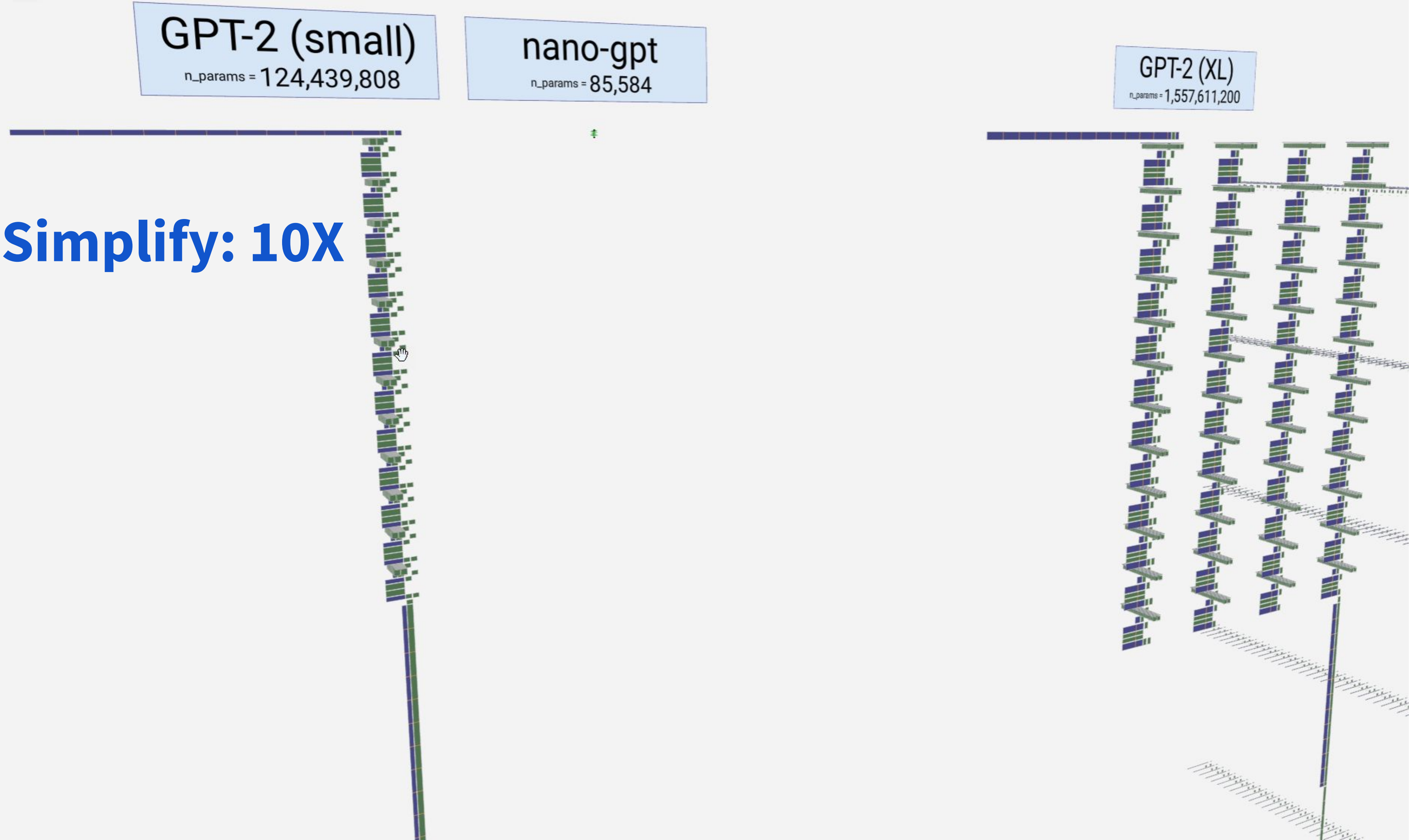
Simplify: 10X
175 billion parameters

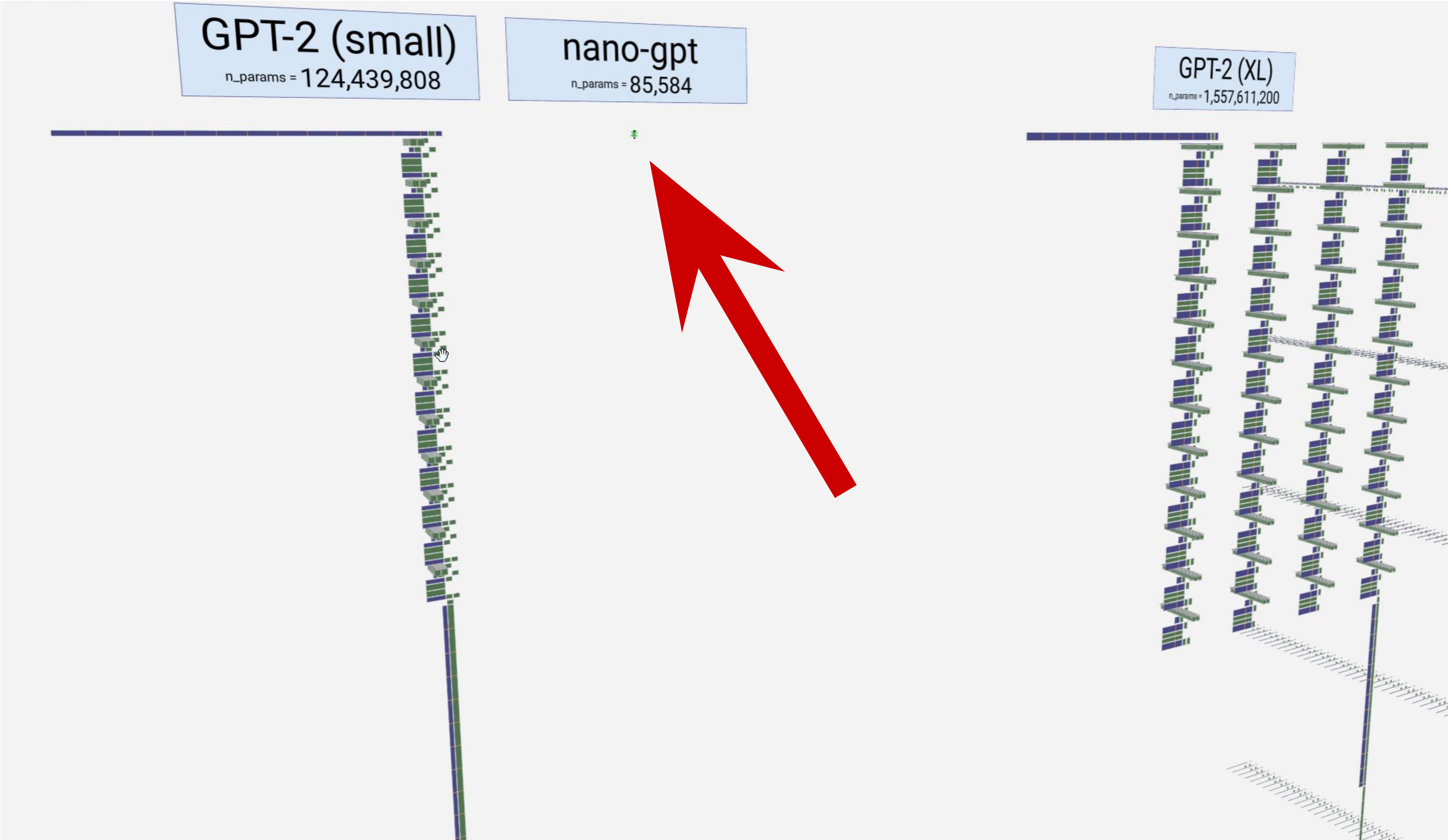


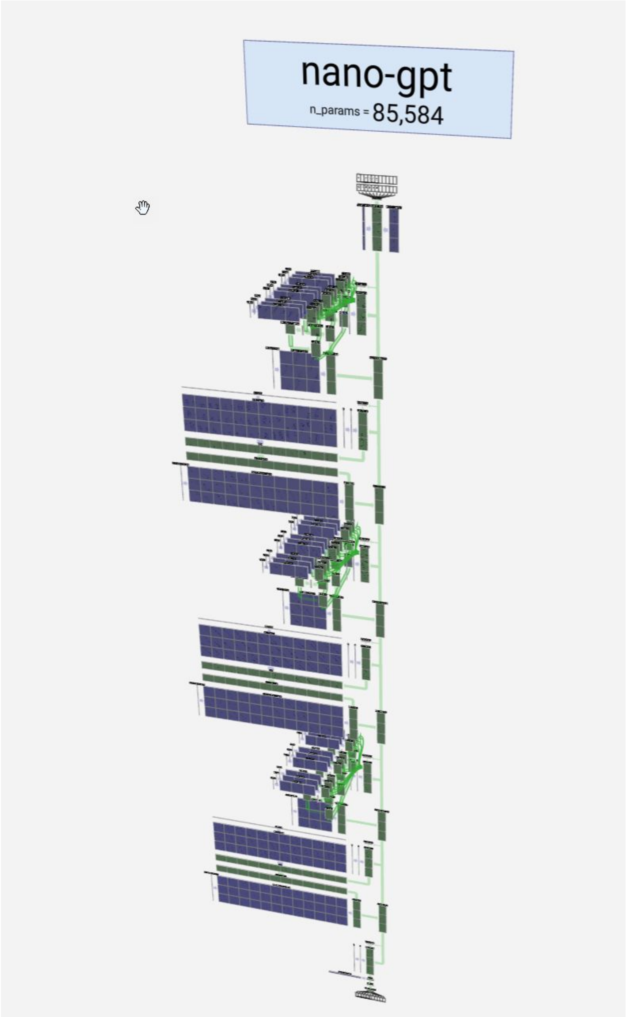
Simplify: 100X



Simplify: 10X




GPT-2 (small)
n_params = 124,439,808
nano-gpt
n_params = 85,584
GPT-2 (XL)
n_params = 1,557,611,200

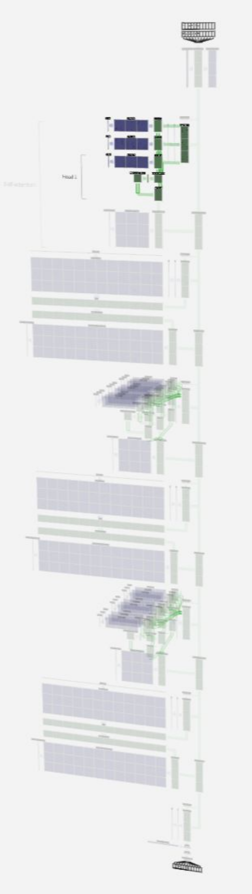


Simplify: 1500X

Abstraction Layer 1: Neural Components

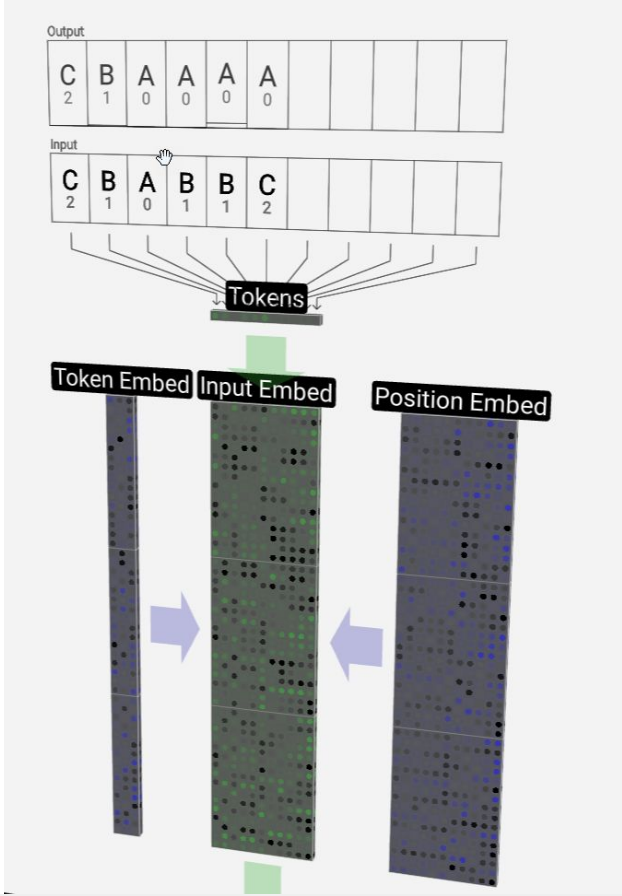
Key components of LLM

1. Embedding
2. Transformer
3. Normalization



Embedding

- Input dimensions = vocabulary size (V)
  - English; V ≅ 1M
  - Toy problem [ABC]; V = 3
- Create a “one-hot” column of Booleans, size V
  - “bottle” = (0, 0, 0, …, 0, 0, **1**, 0, 0, … 0)
    - 1M dimensions; invokes the “curse of dimensionality”
  - “B” = (0, **1**, 0)
    - 3 dimensions
- Reduce to convenient (uncursed) dimensions of Real values
  - English ➔ ~300 dimensions
    - “bottle” = (0.000183, 0.00690, …, 0.0152)
  - [ABC] ➔ 48 dimensions
    - B = (0.000343, 0.00234, …, 0.1436)



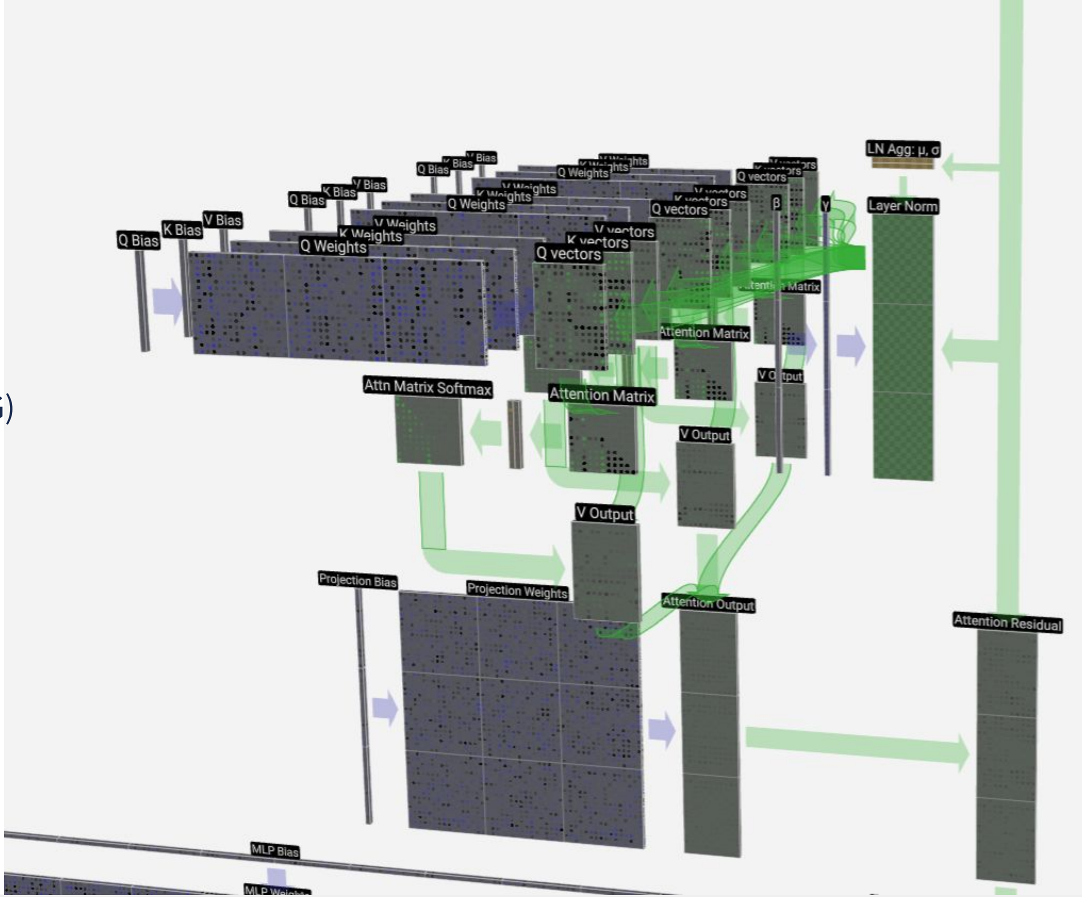
BasisTech

Why embeddings?

- Useful semantics in a tractable number of dimensions
  - king − man + woman ≅ queen
  - king ≅ König ≅ rey ≅ 国王 [guówáng]
- LLM usage
  - Create table with T columns
    - One for each token of input
  - For each token, look up its embedding
    - Column of length 48 / 300
    - Add the column to token embedding table
    - Also create position embedding table
    - Input embedding: sum token embedding with position embedding

Transformer

- *Attention is all you need* [v1: June 2017]
  https://arxiv.org/abs/1706.03762
  - ~~RNN, CNN~~
  - Language translation application
    - 28.4 BLEU English → German
  - Generalization
    - English constituency parsing (CFG)
  - Lots more!
- Four steps
  1. Layer normalization
  2. Self-attention
  3. Projection
  4. Feed-forward,
     multi-layer perceptron



BasisTech

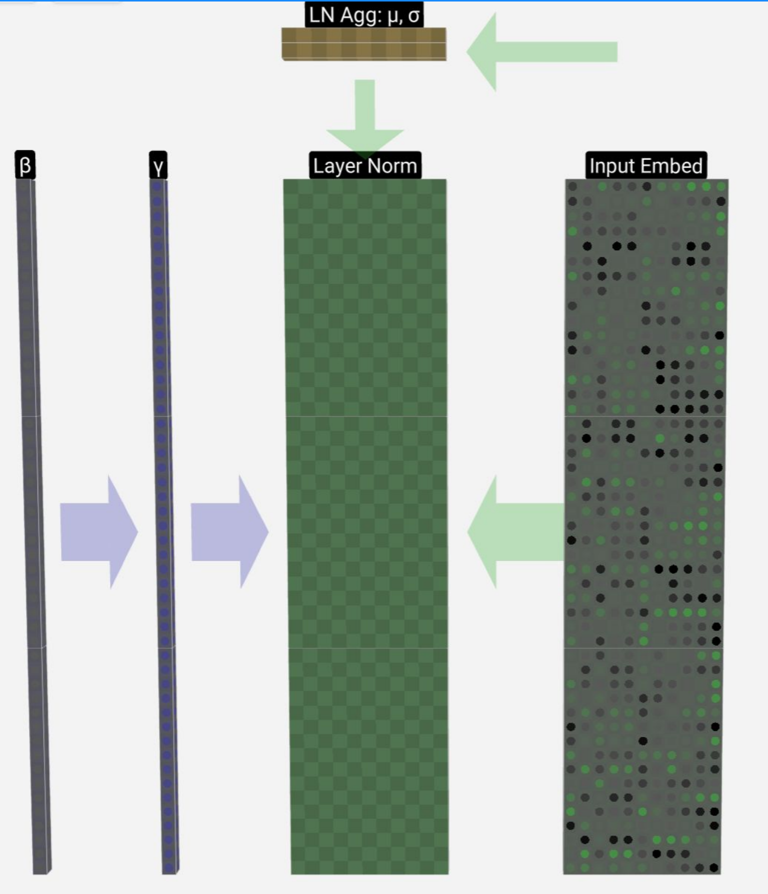
Toy LLM Problem

	Modern LLM	Toy
Words in vocabulary	~1M	3
Embedding dimensions	~300	48
Context window (tokens)	8–128K	11
Transformers	~150	3
Attention heads	9,216	3 x 3
Parameters	1.7T	86K

Abstraction Layer 2: Transformer Operation

Transformer Step 1: Layer normalization

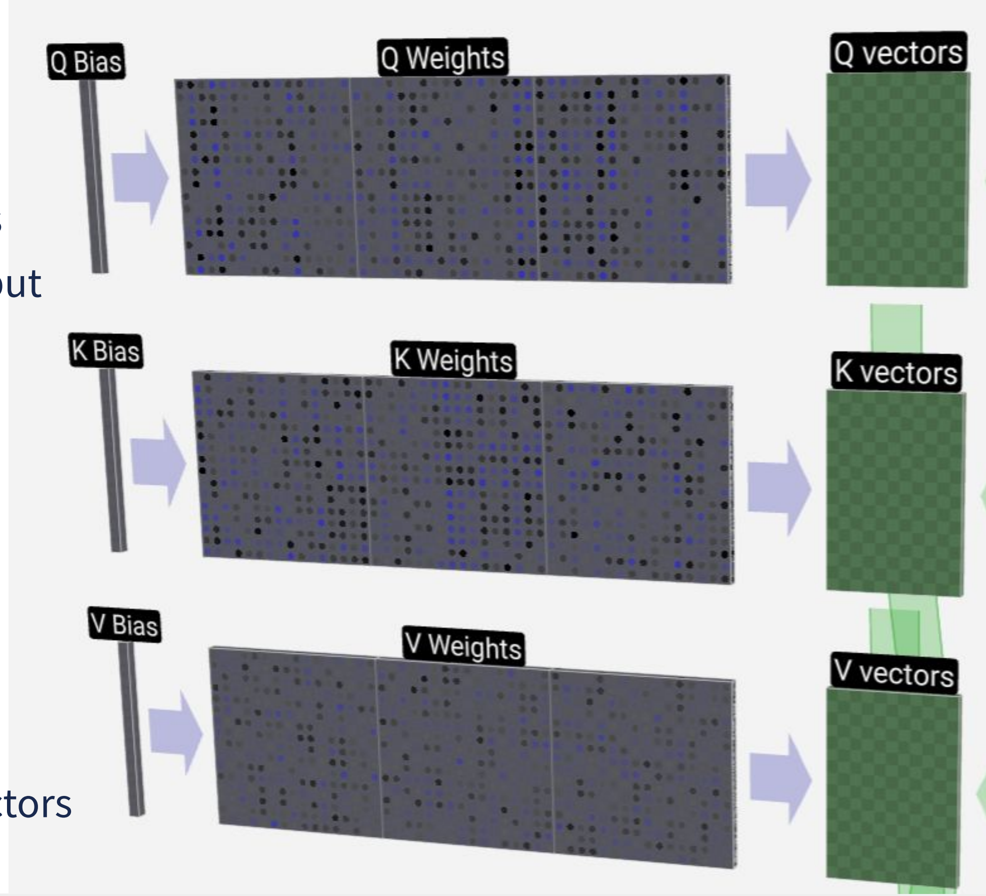
- Scale the columns of input embedding
- For each of the T=11 columns (of n=48 rows)
  - Calculate
    - Average = sum() / n = μ
    - Standard Deviation (SD)= $\sqrt{\text{sum}[(x-\mu)^2]/n} = \sigma$
  - For each cell x: $(x-\mu)/\sigma$
  - Column avg now 0, SD 1
  - Scale with weight (γ), bias (β) values
  - $x \times \gamma + \beta$
  - Column avg now β; SD γ
- Proceed to three parallel self-attention heads
  - Breaks up the space into dimensional chunks



BasisTech

Transformer Step 2a: Self-attention computation

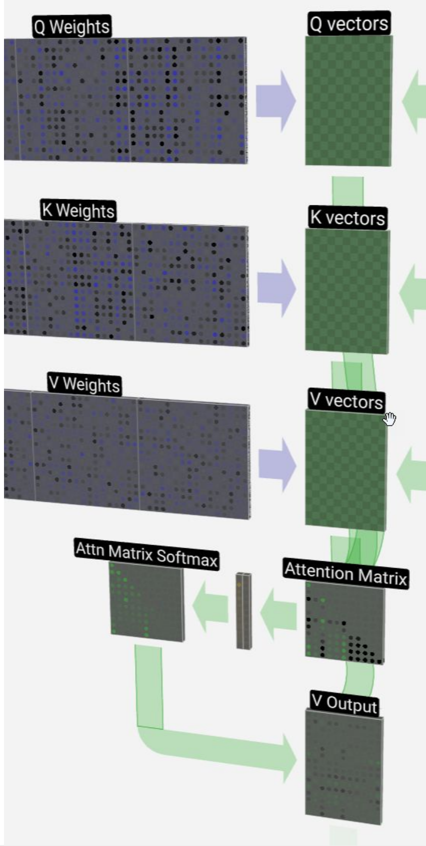
- Three precomputed model Weight tables
  - Q(query), K(key), V(value)
  - Each table has a column of bias values
- For each of the T columns of normalized input
  - For each of [Q, K, V] tables
    - Multiply table by column
    - For each row
      - Dot-product(•)
        - pair up elements
        - multiply each pair
        - add up the products
      - Add the bias for that row
  - Creates tables of Query, Key, Value vectors



BasisTech

Transformer Step 2b: Self-attention query execution

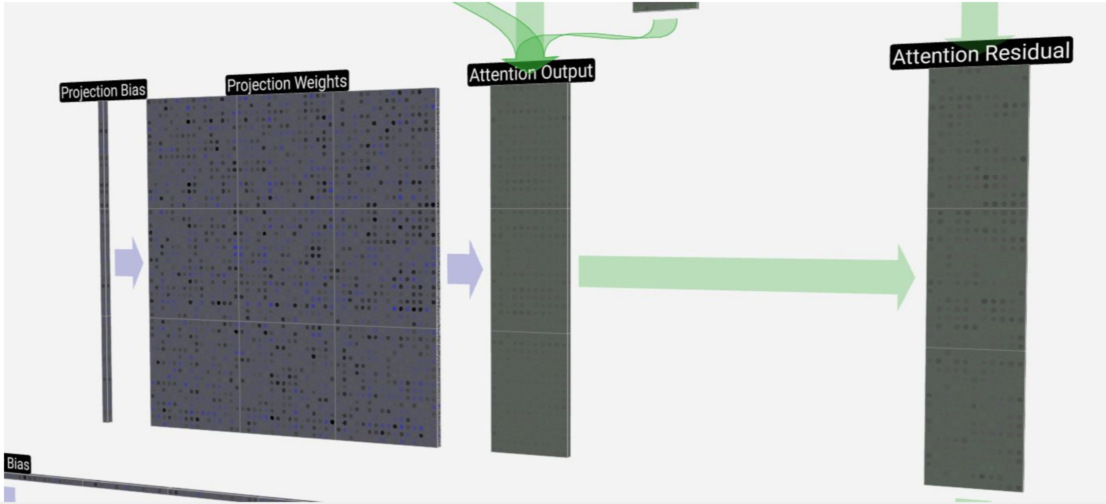
- Build the attention matrix A
  - vectors Q • K
    - Dot product (scales by similarity of vectors)
    - Looks back over all past input columns/tokens
    - Weights the amount of attention paid to them in context of the current token
- Scale A by √(column length)
- Softmax the columns of A: make the values add up to 1. (See final slide)
- Softmax(A) • V vector: produces V Output



BasisTech

Transformer Step 3: Projection

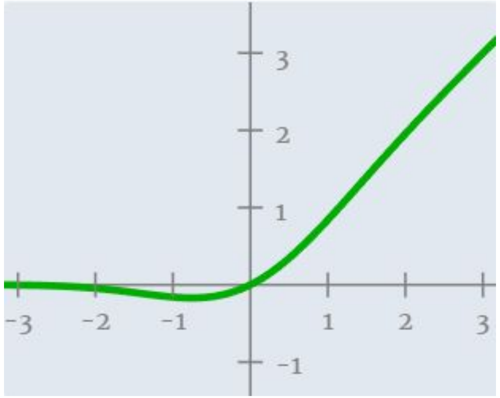
- Stack the V Outputs from each of the multiple attention heads, appending the columns, producing Attention Output
- Apply Projection Weights and Projection Bias to Attention Output
- Add the original input embedding back in to this result, producing Attention Residual
  - Feeding forward the input is another type of normalization
  - Essential for convergence during learning



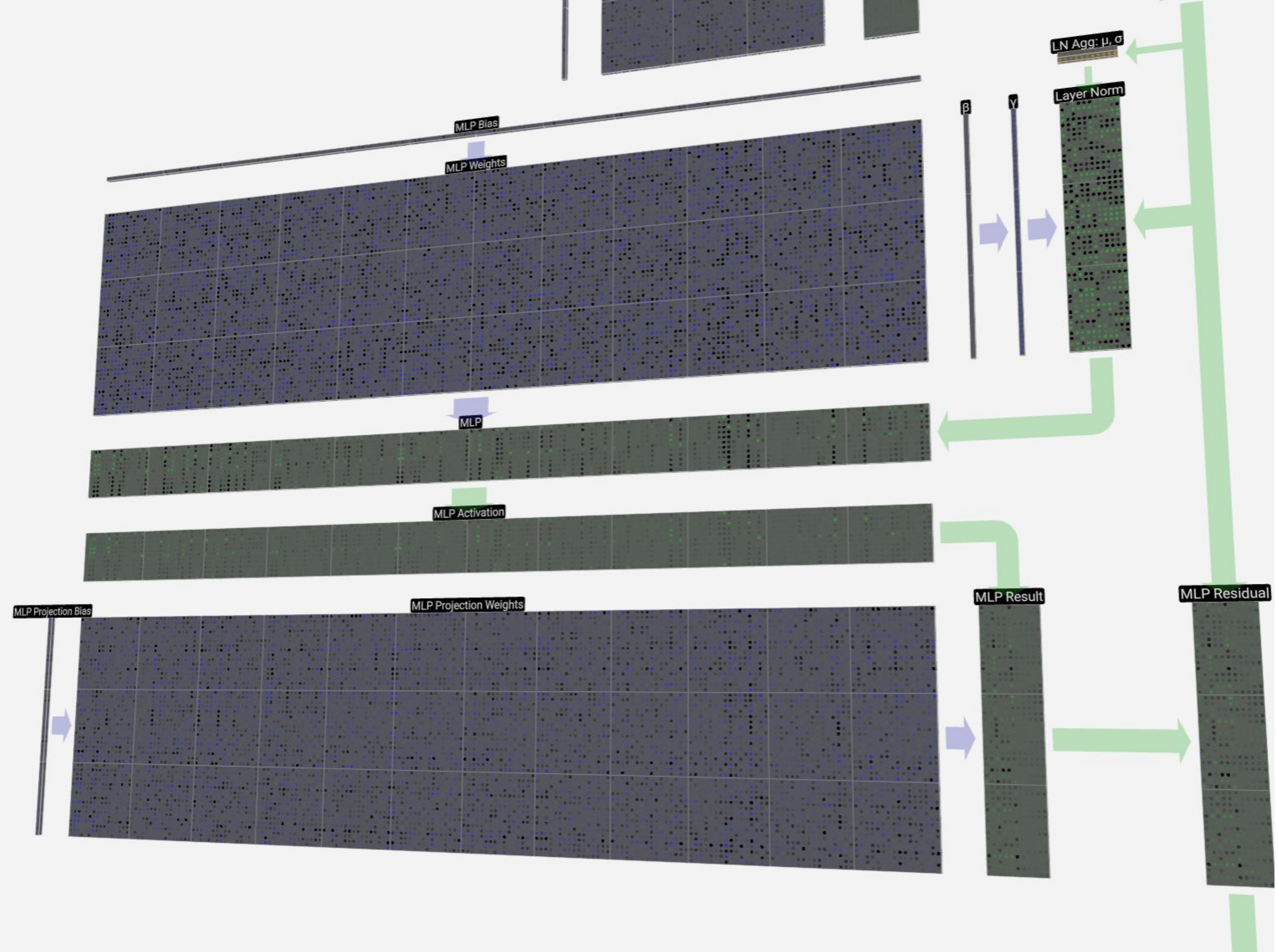
Transformer Step 4: Multi-Layer Perceptron (MLP)

- Normalize (μ, σ) and bias (β, γ) to scale average & standard deviation
- MLP: 2-layer neural network
  - GELU "activation" function
  - Project with a bias vector, collapsing the heads' output
- Add the MLP input back in
  - Feed forward normalization
- This "MLP residual" is the transformer output

Gaussian Error Linear Unit (GELU) function



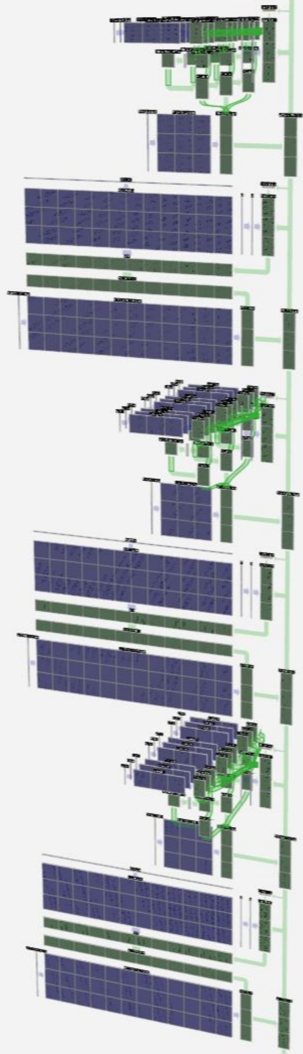
BasisTech


LN Agg: μ, σ
Layer Norm
β
γ
MLP Bias
MLP Weights
MLP
MLP Activation
MLP Projection Bias
MLP Projection Weights
MLP Result
MLP Residual

Transformer iteration

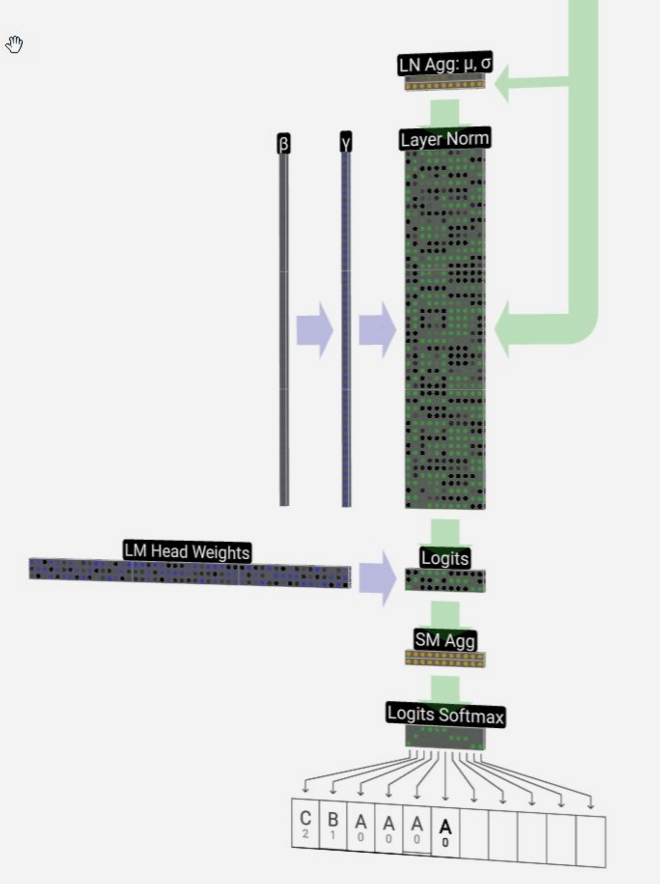
- Feed from each transformer into the next
- Our nano-gpt uses 3 transformers (each with three heads)
- Transformers specialize as they proceed
  - Lower-level feature extraction

    to
  - Higher-level abstractions & relationships



Final Normalization

- Input from final transformer layer
- Normalize (μ, σ) and bias (β, γ) to scale avg & SD
- Final multiplication
  - Scales columns back out to the length of the vocabulary
  - Elements are “logits”
  - Log-probability of the token occurring, summing up to 1
- Final Softmax creates output table
- Choose a path over its columns to produce output
  - Most likely (give me one answer!)
  - Probabilistic / uniform (check veracity later)
  - Temperature parameter
    (sliding scale of likely vs. uniform)



BasisTech

Softmax: why / what

- Vector (row or column) should add up to 1, like probabilities
- For each value
  - Exponentiate
    - All positive values
  - Subtract largest value
    - All negative, except largest, which is now 0
    - Avoids float overflows on division
  - Divide by sum
    - Positive again
    - Adds up to 1

Let's go do it!

Bibliography

Resources

- All credit to Brendan Bycroft
- LLM visualization: https://bbycroft.net/llm
- Visualization project: https://github.com/bbycroft/llm-viz/tree/main/src/llm

BasisTech