Wids

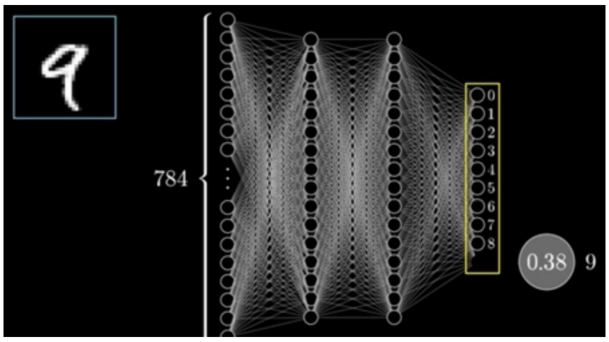
Friday, 12 December 2025 4:02 PM

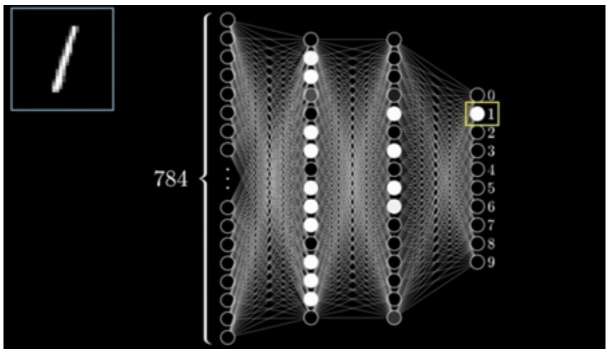
Neural Network:

Neuron:

A thing that holds memory between 0 and 1

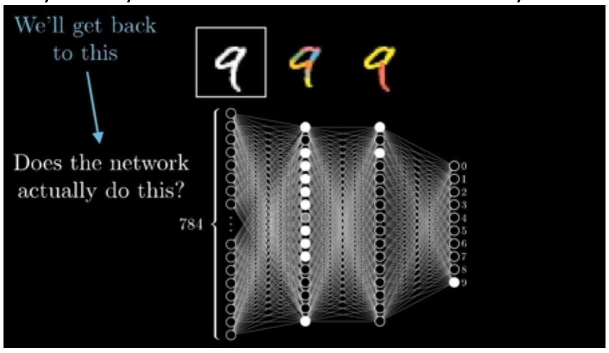
Take a 28*28 pixel image of a number now each cell holds a number between 0 to 1 (0 for dark 1 for white) this is know as activation

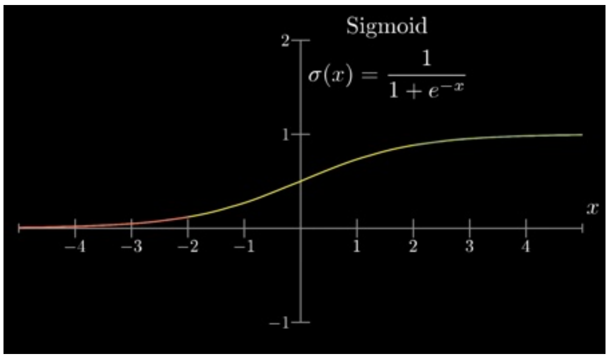




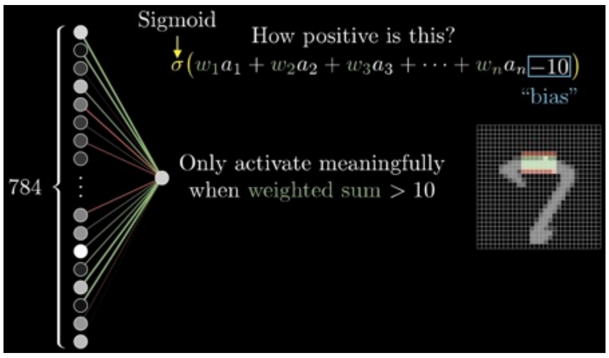
Activation of one layer determines the activation of the next layer and the answer is the brightest neuron in the final layer

Why exactly do we have confidence that this layers work??





Now people are using ReLU because it is working increadabely well for some deep neural networks

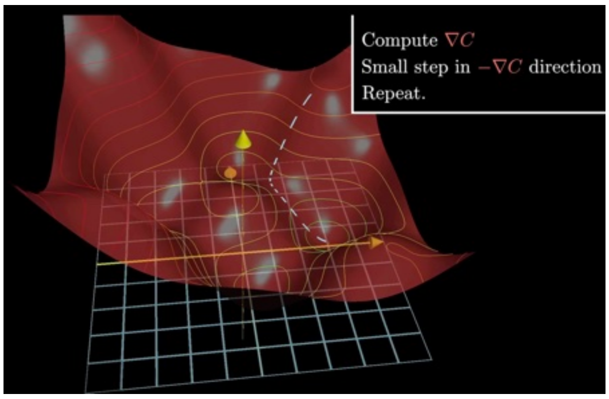


How positive is this?

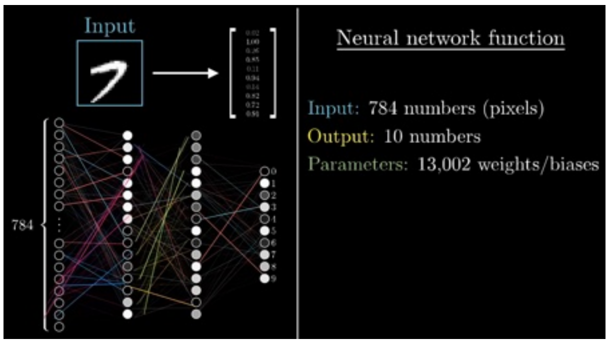
$\sigma(w_1a_1 + w_2a_2 + w_3a_3 + \cdots + w_na_n \boxed{-10})$

Only activate meaningfully when weighted sum > 10

Cost function:
We should find the input(weights and bias) that minimizes this cost function
But it is very complicated for a traumatic function but we can choose something randomly and choose which direction to travel in (find the slope , if slope is >0 shift to the left if the slope<0 shift to the right)

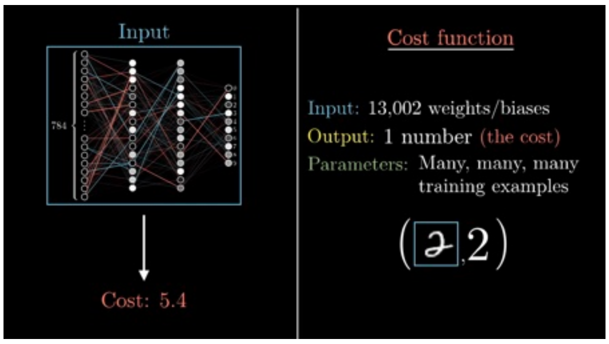


Compute ∇C
Small step in $-\nabla C$ direction
Repeat.



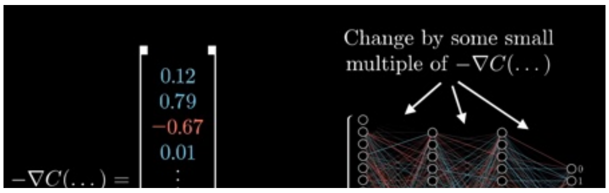
Neural network function

Input: 784 numbers (pixels)
Output: 10 numbers
Parameters: 13,002 weights/biases



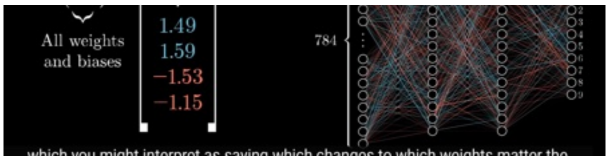
Cost function

Input: 13,002 weights/biases
Output: 1 number (the cost)
Parameters: Many, many, many training examples



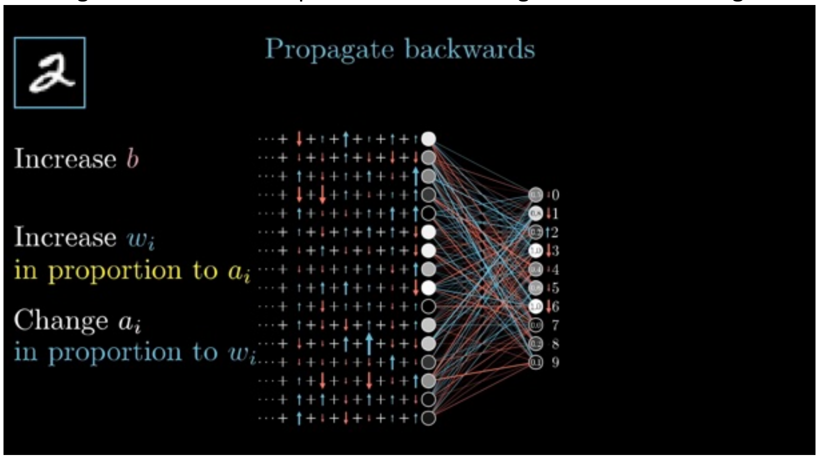
Change by some small multiple of $-\nabla C(\dots)$

$$-\nabla C(\dots) = \begin{bmatrix} 0.12 \\ 0.79 \\ -0.67 \\ 0.01 \\ \vdots \end{bmatrix}$$



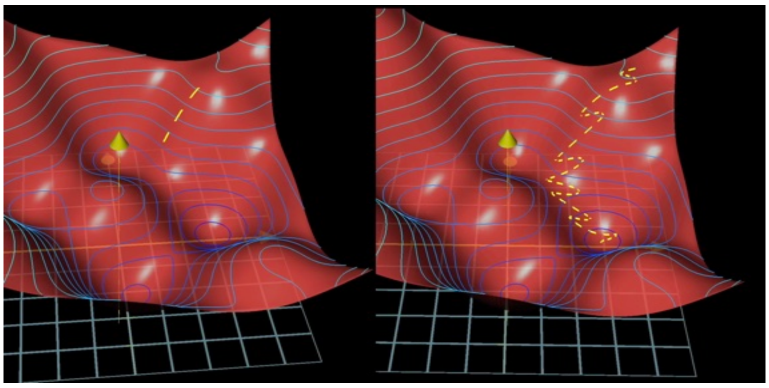
Despite identifying the number I can never draw then
And even if u give some nonsense it confidently says the answer for it

Back Propagation:
Is the algorithm how a single training data example will like to nudge the training weights and bias
A true gradient descent step will take the average for all the training data above and takes the decision


Propagate backwards
Increase b
Increase w_i in proportion to a_i
Change a_i in proportion to w_i

Stochastic gradient descent:

Dividing into small chunks and then rapidly stepping the down the hill with the help of the average of those few rowed data than doing for every data in the training set



MNIST data base

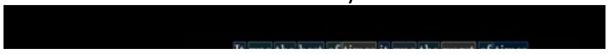
LLM:

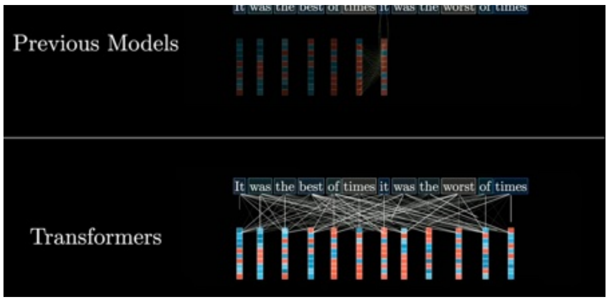
A sophisticated mathematical function which predicts about what word comes next for any piece of text but instead of predicting one certain word It gives all possible words with their probabilities

We have parameters in this which change continuously on training (same as how weights and bias are tuned based on the back propagation)

What all process does the chat bots undergo
1)pretraining
2)RLHF(Reinforcement learning with human feedback)

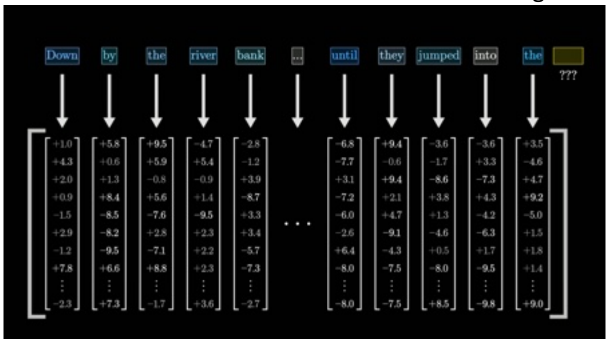
Before 2017 the words are analyzed one to one:



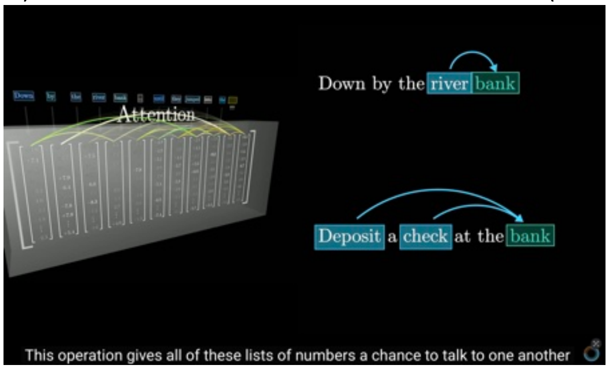

It was the best of times it was the worst of times
Previous Models
It was the best of times it was the worst of times
Transformers

But transformer analyzes all the input text

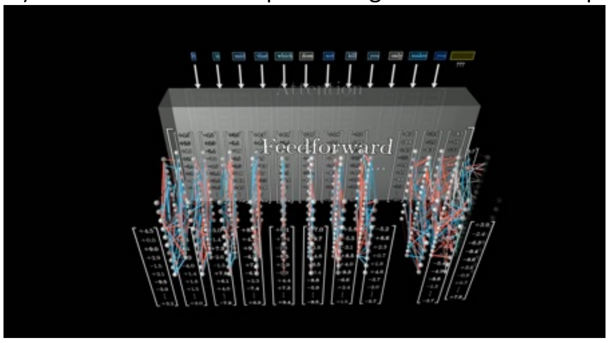
In transformer each word is associated with a large vector of numbers:



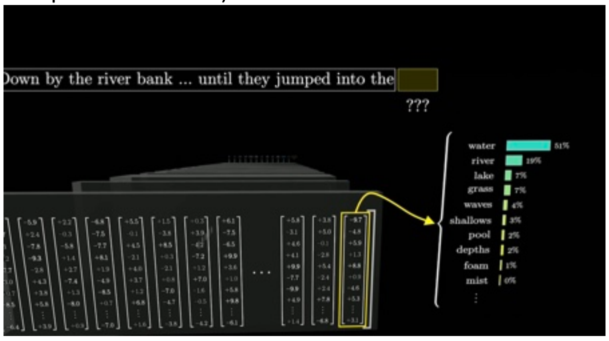
1) Attention: all the words talk with one another(which causes better context based meaning development for each word)


Attention
Down by the river bank
Deposit a check at the bank
This operation gives all of these lists of numbers a chance to talk to one another

2) Feedforward: more processing of the words with pretrained data

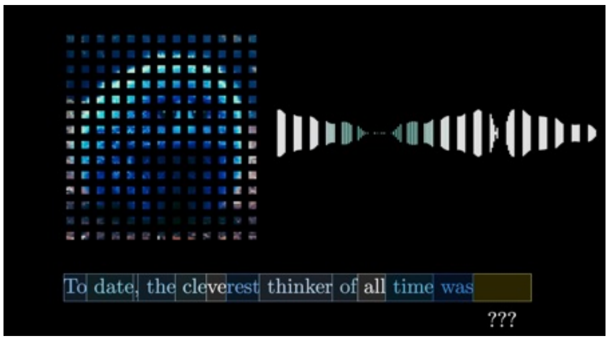


The last step produces a long vector with all the possible words along with their probabilities (this is influenced by our input and pretrained data)



GPT (Generative pretrained training)

Original transformer (in the publication of the Attention is all u need) is introduced for translating the text in one language to the other
Each little chunk is known as tokens:

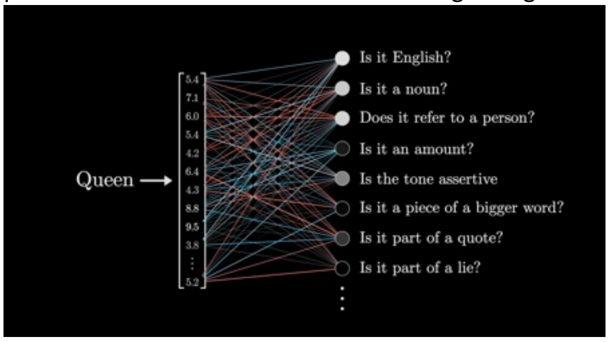


Each of these tokens is associated to a vector(as mentioned above)

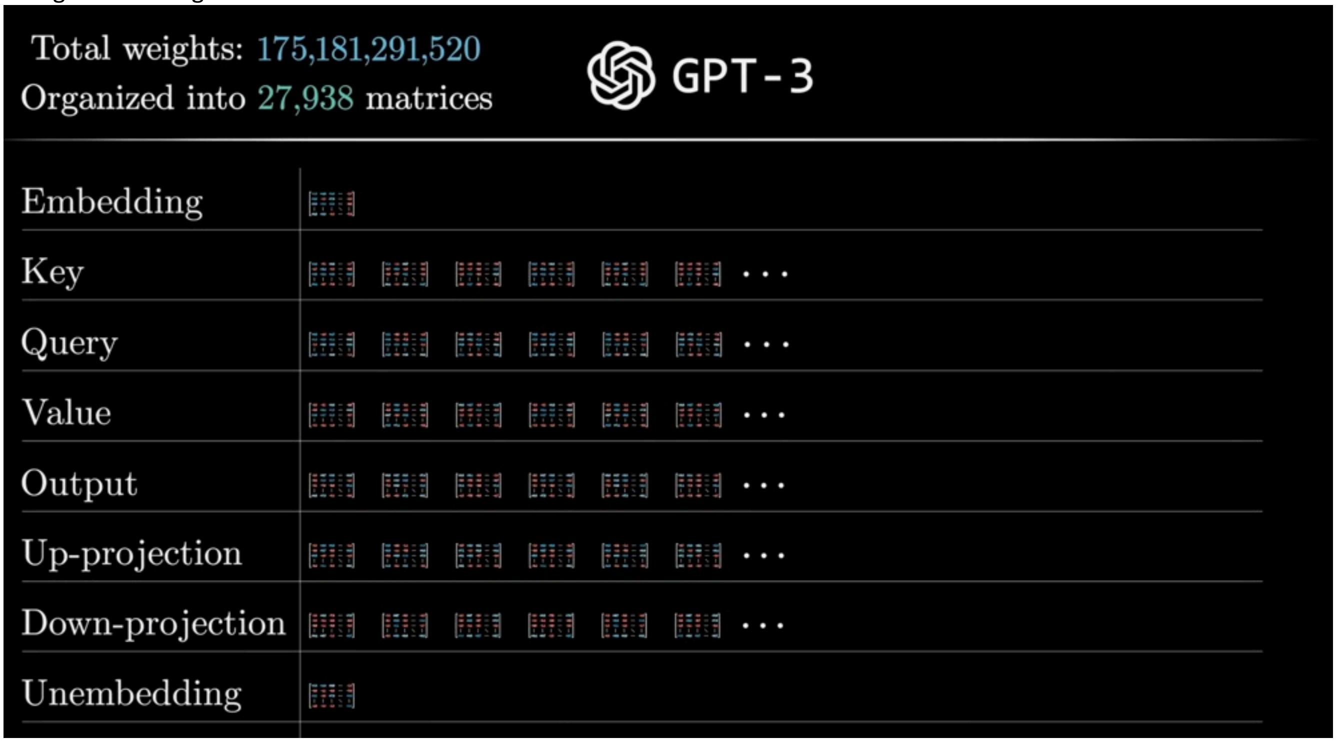
If we think of these vectors in high dimensional coordinate space then words with similar meaning are near to eachother

Then these vectors are passed through the process attention

In multilayer perceptron/feed forward: Vectors don't talk to each other instead they all go through the same process in parallel to each other.It is more like asking a long list of questions to each vector and updating based on their response



Categories of weight


Total weights: 175,181,291,520
Organized into 27,938 matrices

GPT-3

Embedding
Key
Query
Value
Output
Up-projection
Down-projection
Unembedding