

Walk through + Neural Networks

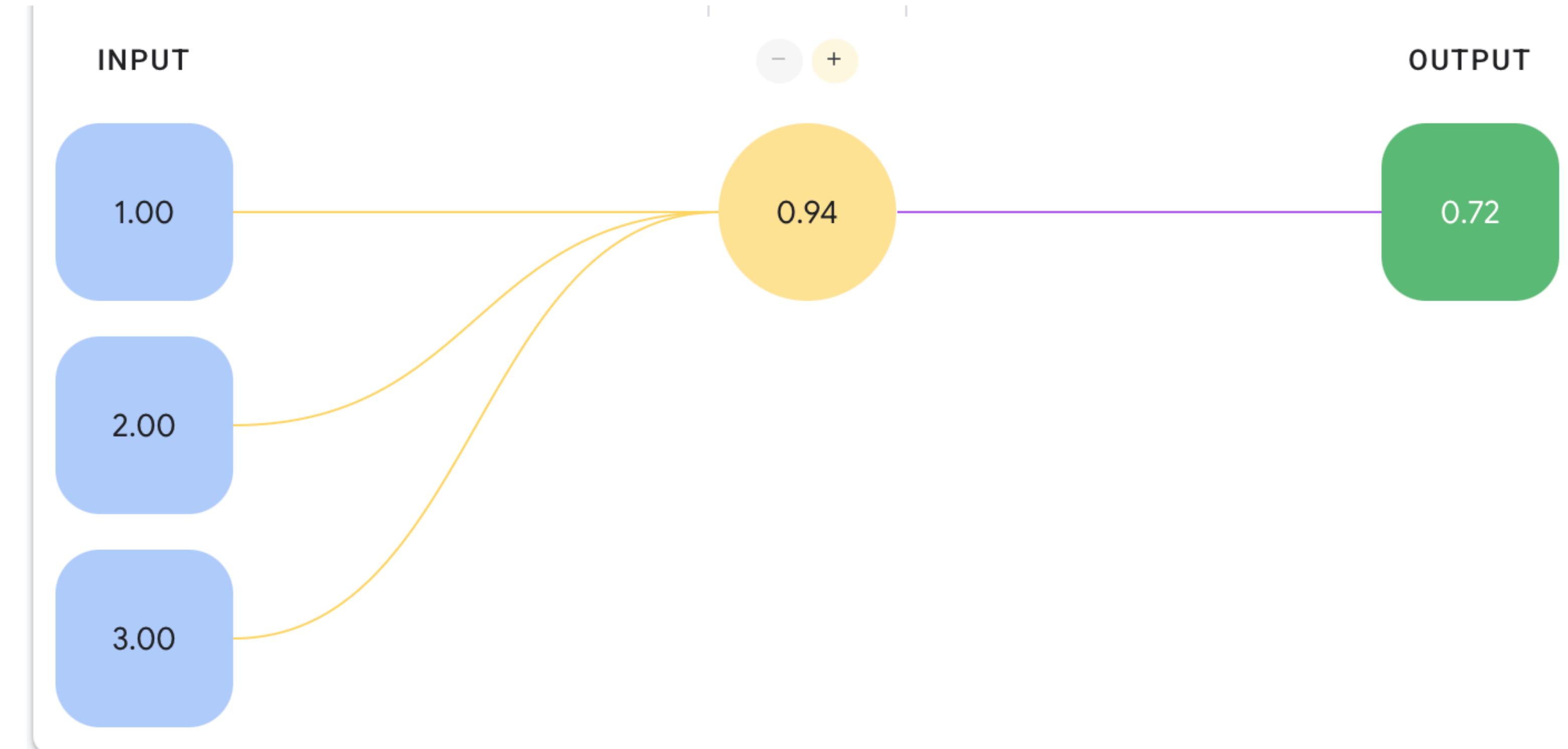
Divij 2025-02-26

The process

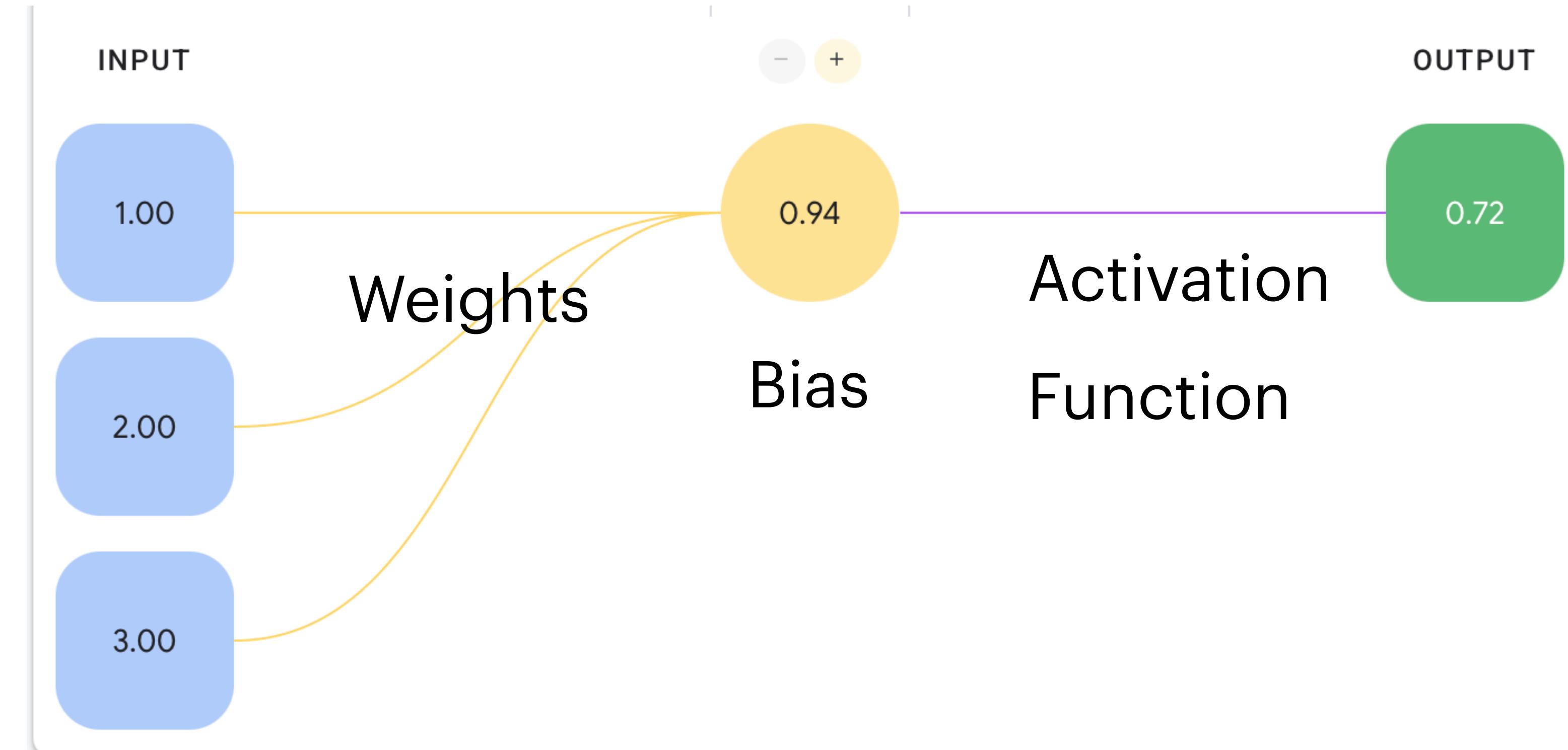
- Let there be some **feature(s)** x
- Let there be some **outcome** y
- Then, there exist some **hypothesis** h of the relationship between x and y
- This hypothesis relies on **parameters** K
- Using some **loss** L that we try to minimize, we estimate the best **parameters** for the given **hypothesis**

Why do this?

- Every model so far has the hypothesis decided
- What if we do not know the hypothesis???
- Neural networks can function as general learning machines, only limited by the complexity of the network

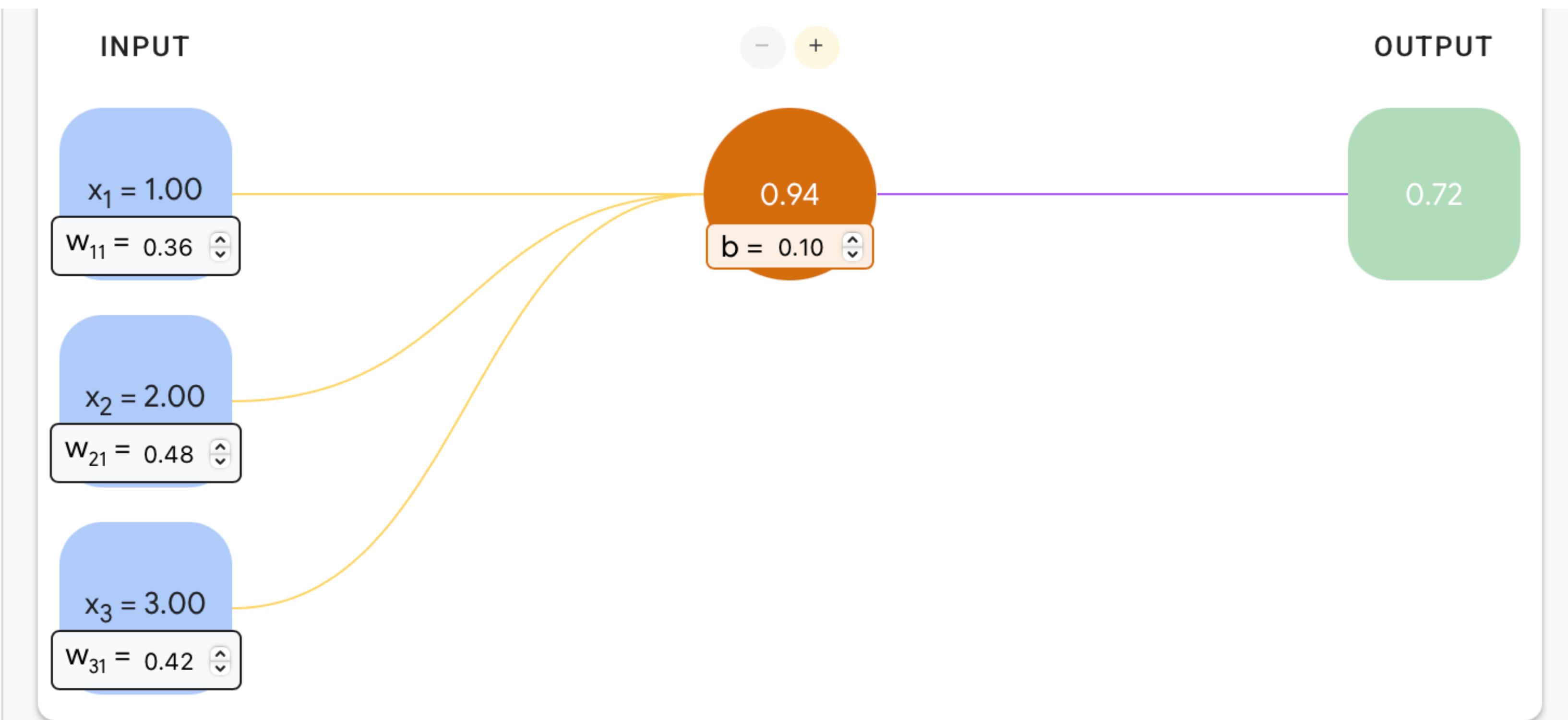


- <https://developers.google.com/machine-learning/crash-course/neural-networks/nodes-hidden-layers>



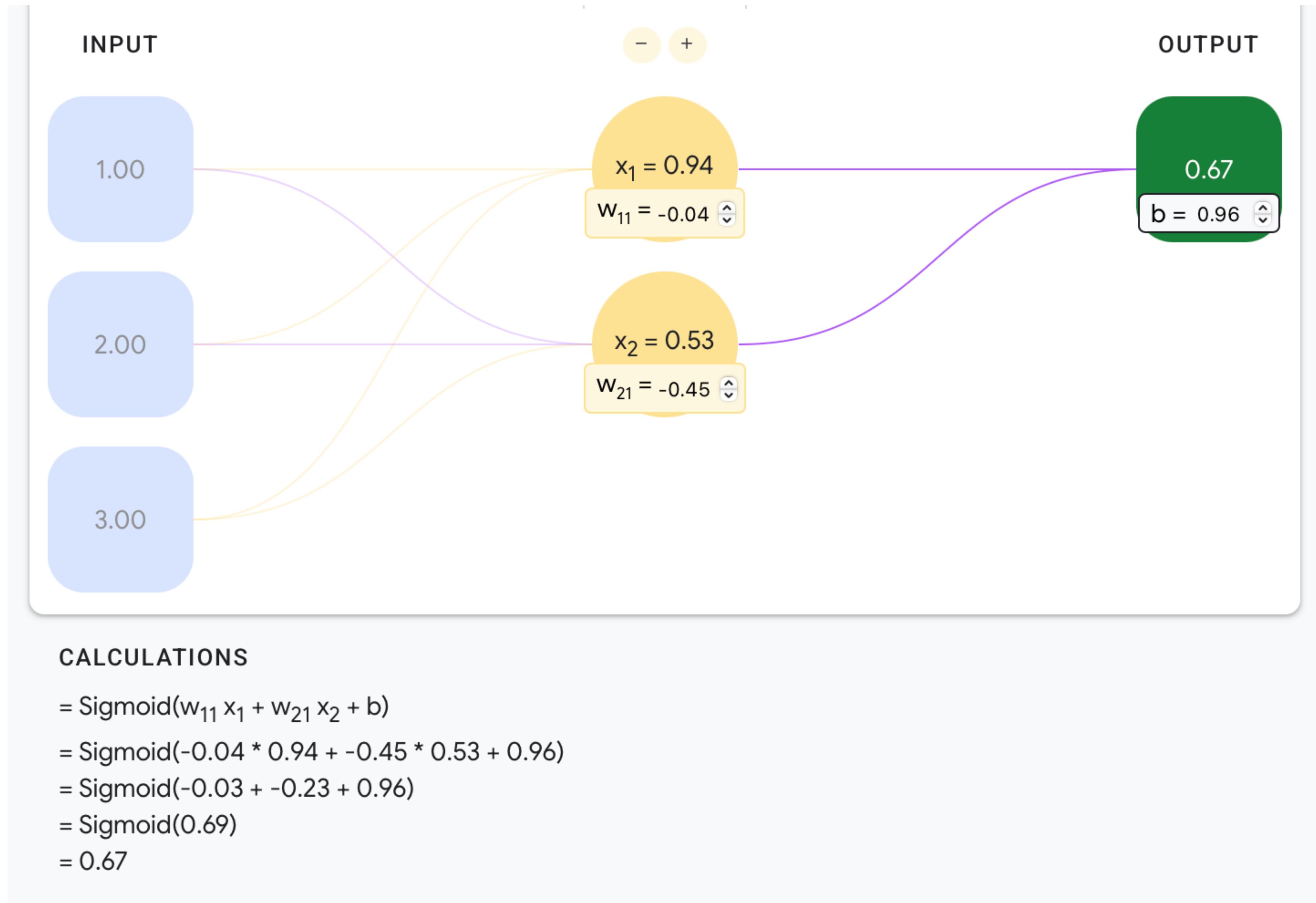
- <https://developers.google.com/machine-learning/crash-course/neural-networks/nodes-hidden-layers>

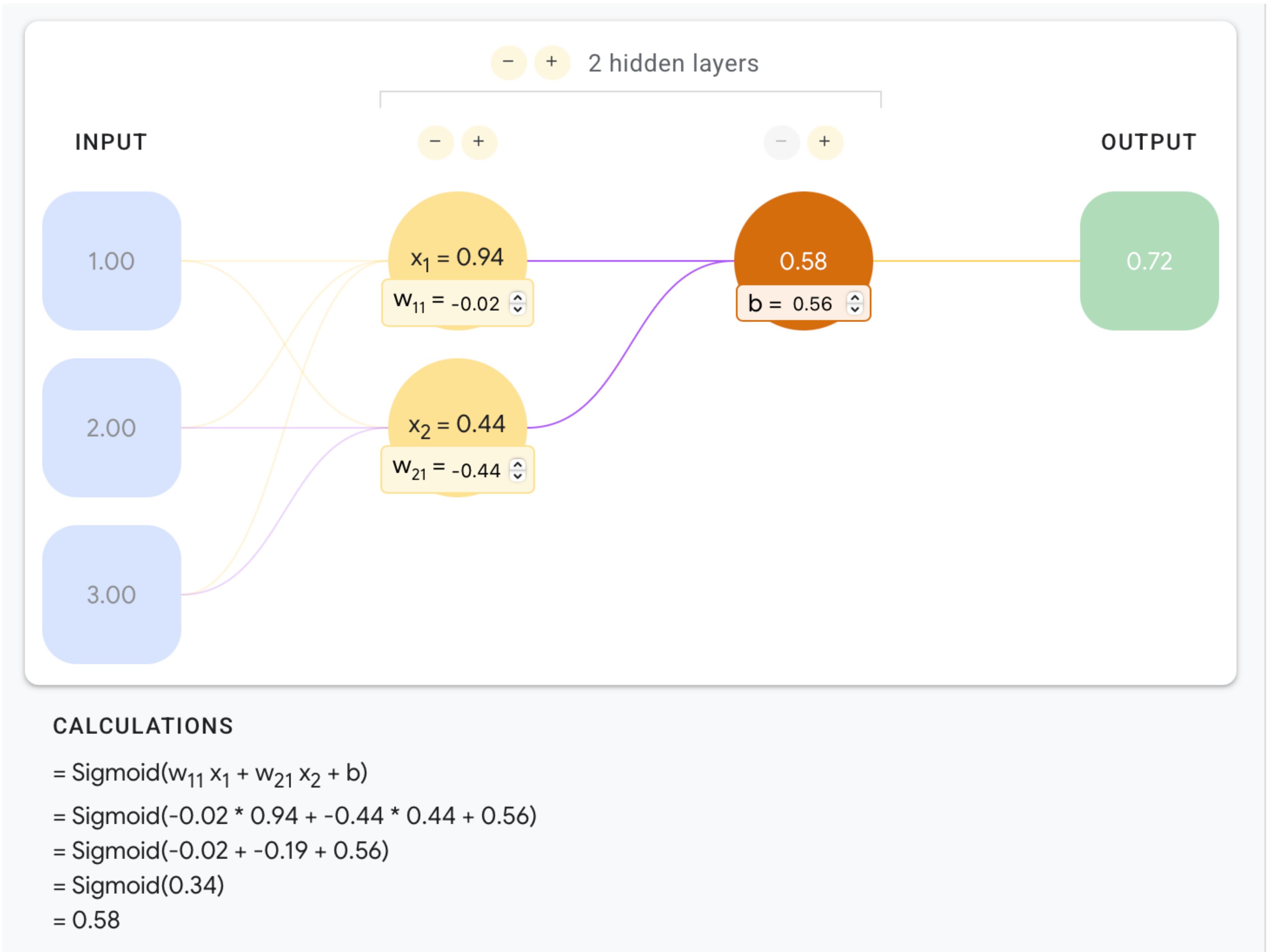
- **Weights**: These are the parameters that determine the strength of connections between neurons
- **Biases**: Biases are additional parameters that allow neurons to “shift” even when their inputs are zeros. They shift the activation function left or right.
- **Activation Functions**: These are mathematical operations applied to the weighted sum of inputs (plus bias) to determine a neuron's output. They introduce *non-linearity*, allowing neural networks to learn complex patterns.

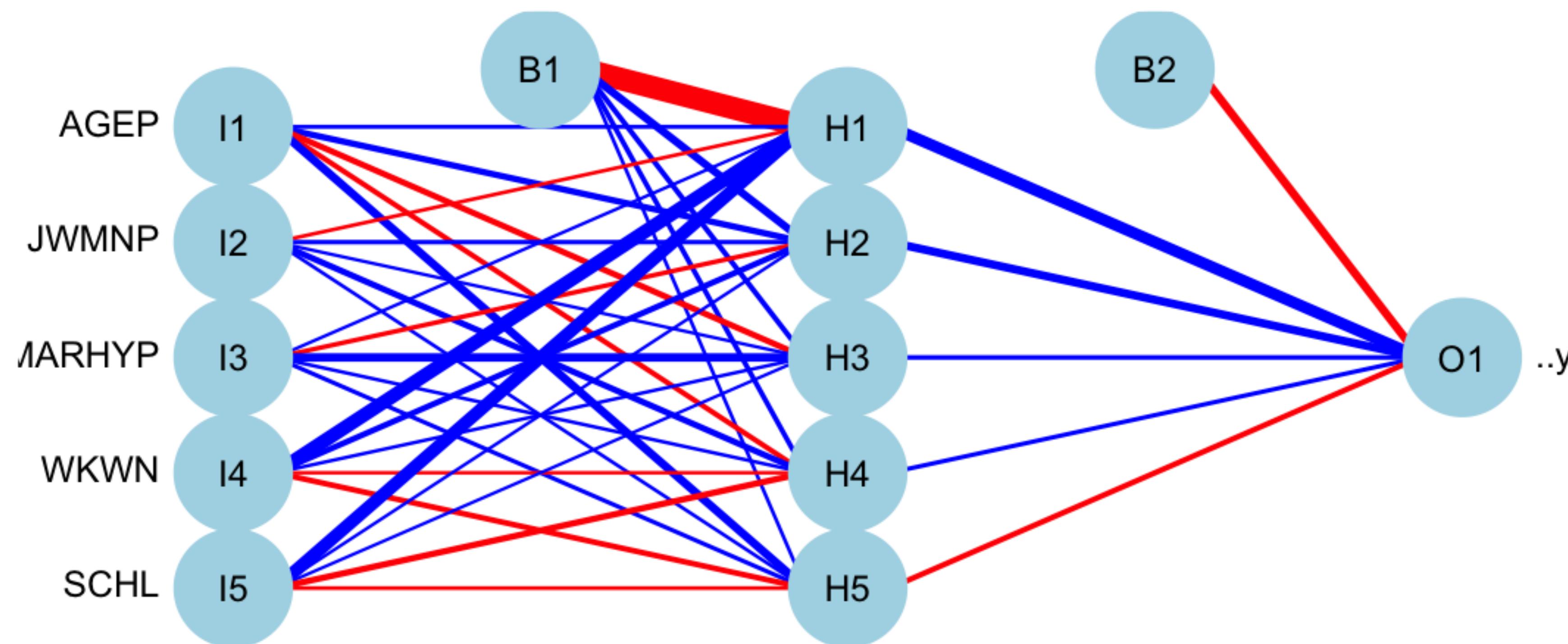


CALCULATIONS

= Sigmoid($w_{11} x_1 + w_{21} x_2 + w_{31} x_3 + b$)
= Sigmoid($0.36 * 1.00 + 0.48 * 2.00 + 0.42 * 3.00 + 0.10$)
= Sigmoid($0.36 + 0.96 + 1.26 + 0.10$)
= Sigmoid(2.67)
= 0.94

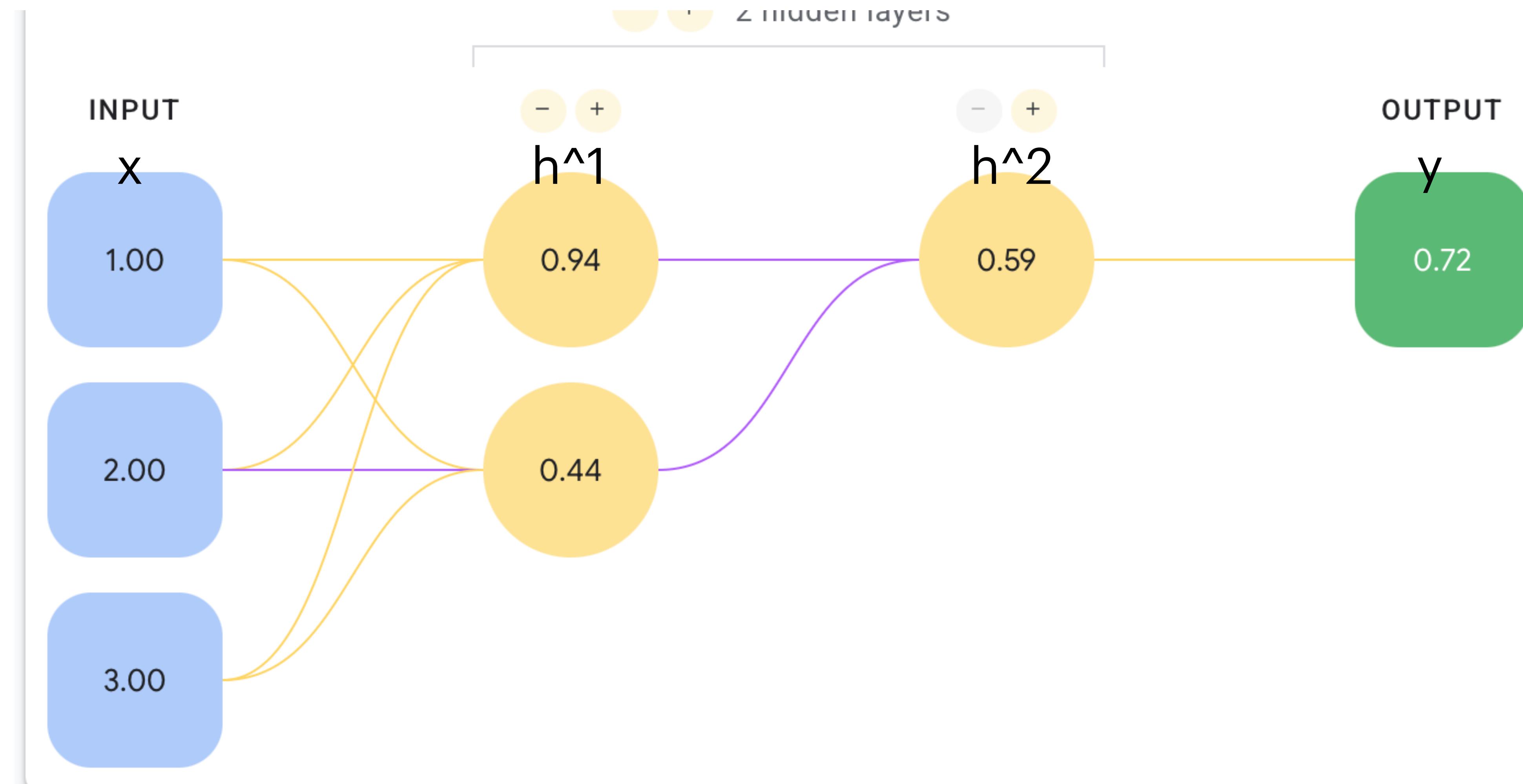






What about classification?

How to train?
Back propagation!



Equations

$$\hat{y} = f(h_1^2 w_{11}^2 + b_1^2)$$

$$h_1^2 = f(h_1^1 w_{11}^1 + h_2^1 w_{12}^2 + b_1^1)$$

$$h_1^1 = f(x_1 w_{11}^0 + x_2 w_{12}^0 + x_3 w_{13}^0 + b_1^0)$$

$$h_2^1 = f(x_1 w_{21}^0 + x_2 w_{22}^0 + x_3 w_{23}^0 + b_2^0)$$

$$L = MSE(y, \hat{y})$$

- What value of say w_{11}^0 will minimize L
- Find the relationship of the change between w_{11}^0 and L
 - How much does L change with respect to changes in w_{11}^0 ?

$$\frac{\delta L}{\delta w_{11}^0}$$

$$L = MSE(y, \hat{y})$$

- $\frac{\delta L}{\delta w_{11}^0}$
 - This is a numerical value of the relationship between them at the current values of all the weights, biases, inputs etc.
 - If positive -> when w_{11}^0 goes up (or down) so does L
 - If negative -> when w_{11}^0 goes up (or down) L does the opposite

- $\frac{\delta L}{\delta w_{11}^0}$
 - We want to reduce L
 - We move AWAY from this direction
 - Gradient descent - We “descend” (reduce the loss) in the loss plane in the opposite of the direction of the “gradient”

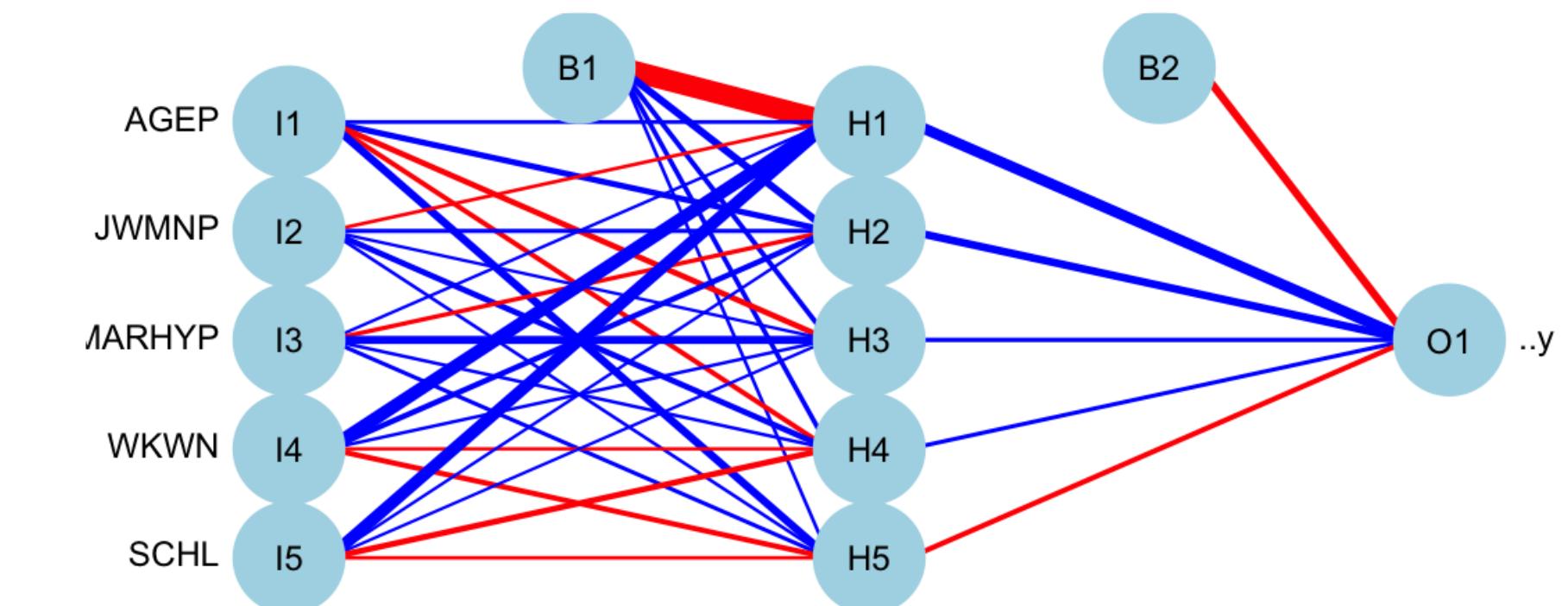
- $NEW\ w_{11}^0 = OLD\ w_{11}^0 - \eta \frac{\delta L}{\delta w_{11}^0}$
- η - Learning rate
 - How much of this to remember vs how much of the past to remember
- Repeat for all variables!

In practice

- Data **MUST** be scaled / normalized - neural networks are very sensitive to scale of data - we try to center around 0, and keep most values between -1 and 1
- Handle missing values or bad data before training the model

That's a lot of work!

- In this case, we have
 - Layer 1 - 5 neurons with 5 inputs each
 - $5 * (5 + 1) = 30$ parameters
 - Layer 2 - 1 neuron with 5 inputs each
 - $1 * (5 + 1) = 6$ parameters
 - Total 36 parameters
 - Each forward pass has 30+ calculations, each backward pass has 5ish? ~200 calculations per forward / backward pass
 - With 120,000 rows, and 10 epochs - 240,000,000 calculations!



Scale of LLMs

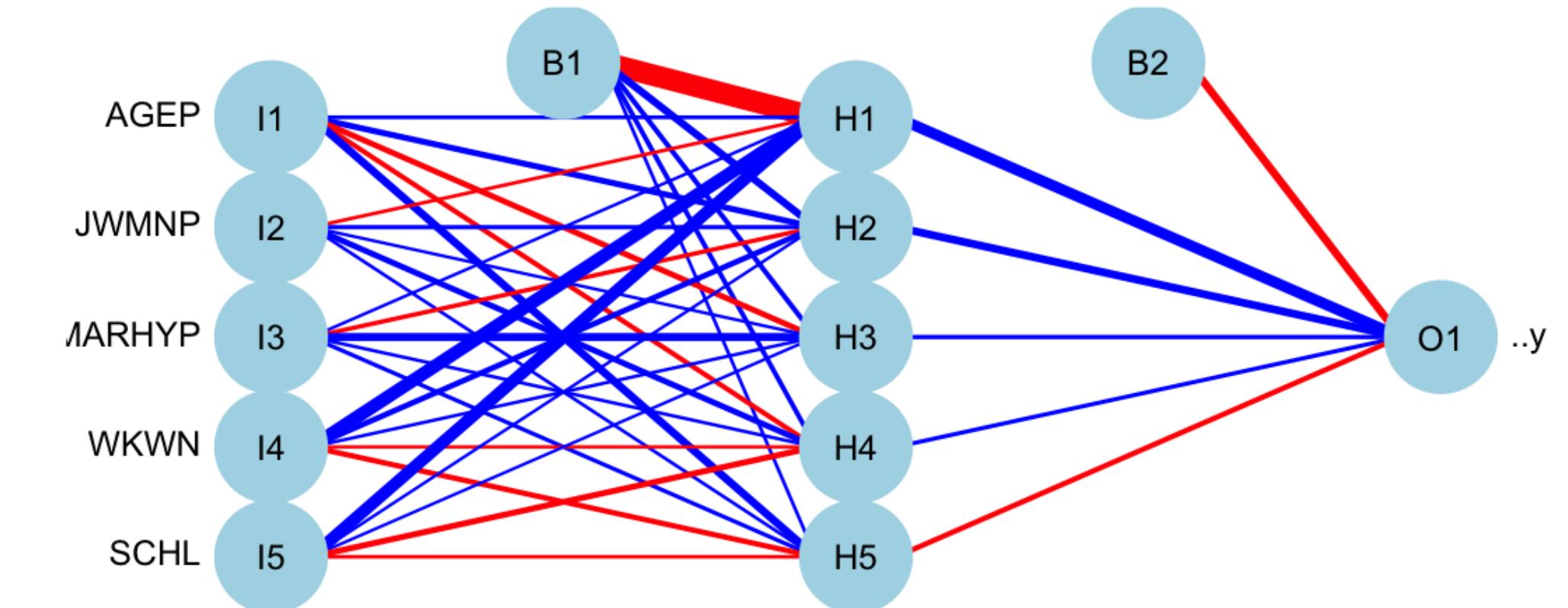
- OpenAI O3 - ~4,000,000,000,000,000,000,000,000 calculations
- My laptop - 3,000,000,000,000 calcs per second
- 1,300,000,000,000 seconds to train – 41,222,730 years
- Functionally just a neural network!! Just a very very very large one

In practice

- Do as many calculations in parallel as possible!
- We move the loss only some some N number of inputs - Batch Size.
- We want to avoid the W/B getting too large or too small -> we ensure that the gradients are in a reasonable range
- We change the learning rate more dynamically, take larger steps earlier then smaller steps, then take some larger steps
- We also do not calculate the gradient every time, do it randomly - stochastic
 - This avoids getting stuck in “local minimas”, generally way faster, and computationally more efficient

Interpretability

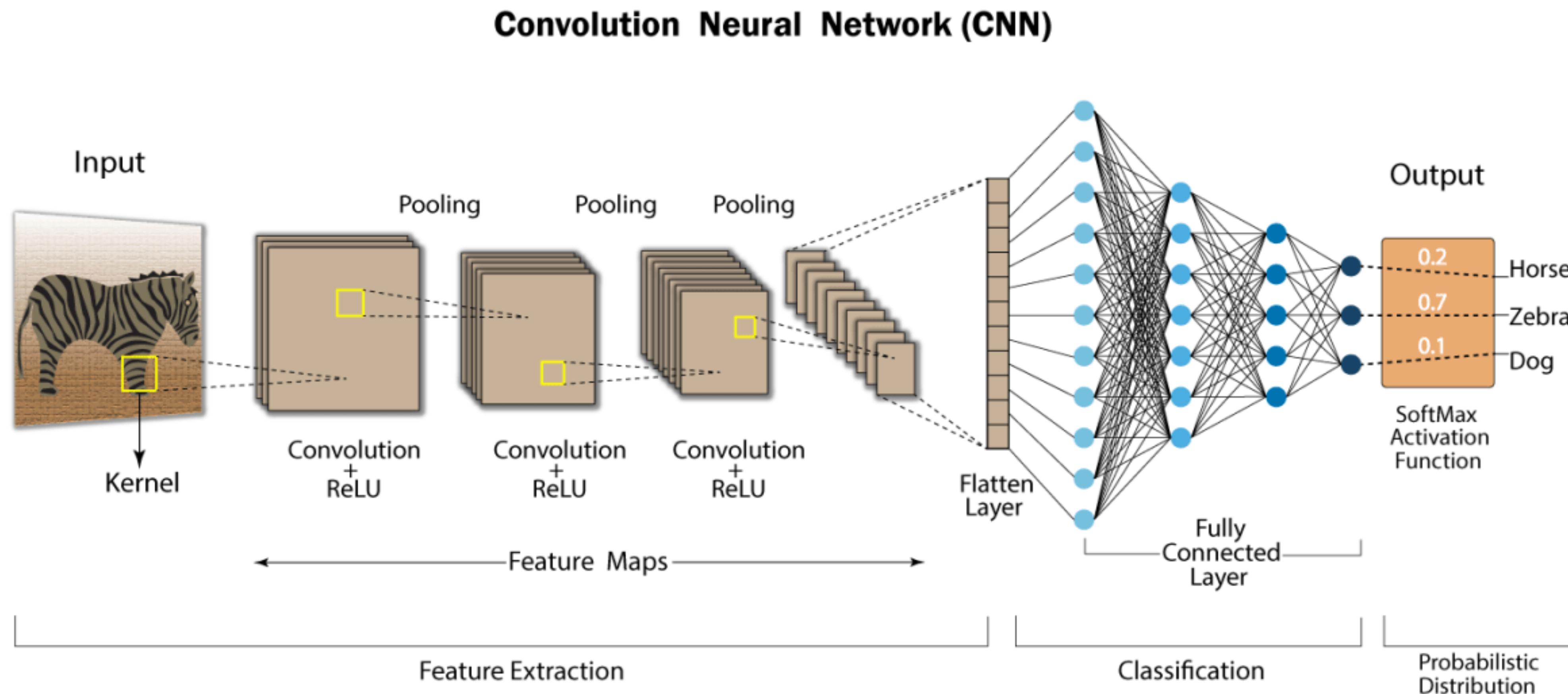
- Hard to interpret even at this size
- We lose almost all interpretability at the scale of LLMs
- We treat it like a black box - adds to the anthropomorphization of LLMs
- They are NOT humans - just large neural networks!



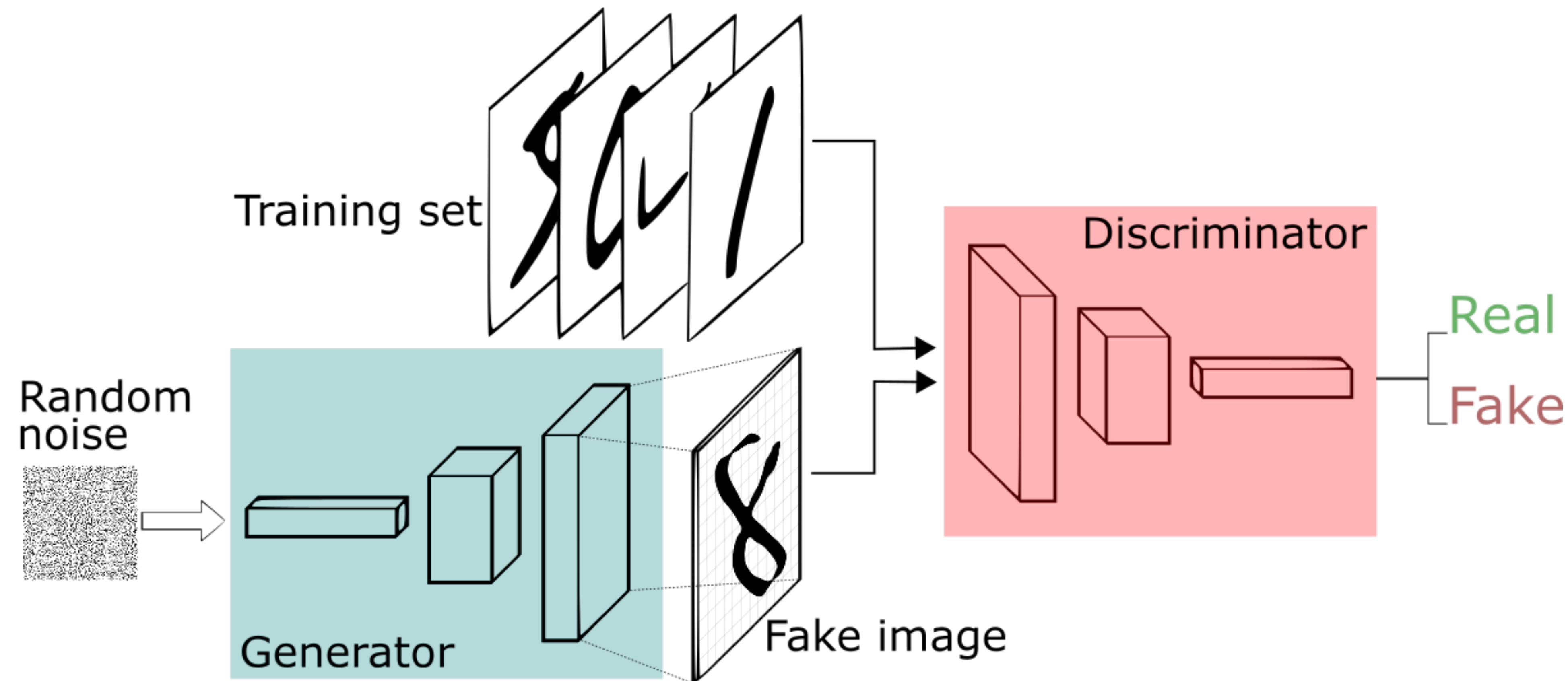
- We are now at the point where our understanding of future networks will not keep pace with what we can reasonably run on our systems
- Therefore, we can classify the ML we now know to be of two kinds
 - Ones we can model, train, and use all by ourselves
 - Ones where we rely on others to model, and train, and we end up being consumers of the models
- We need to learn rules of how to use these LLMs, what works well what does not etc etc.

Other popular architectures

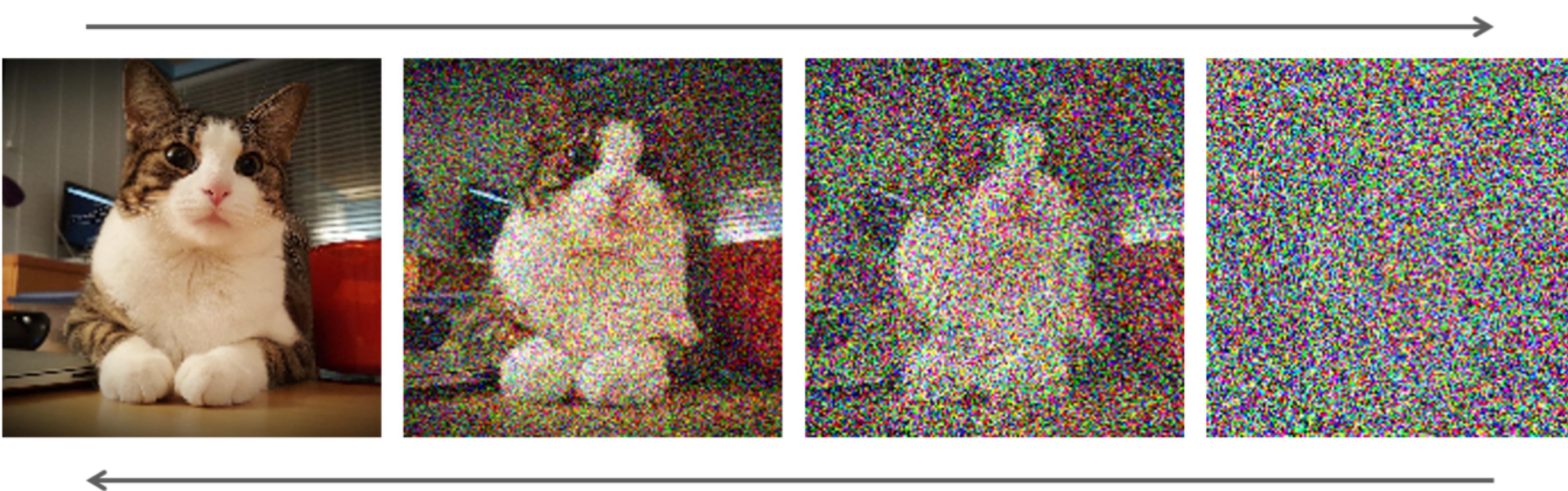
Convolutional Neural Networks



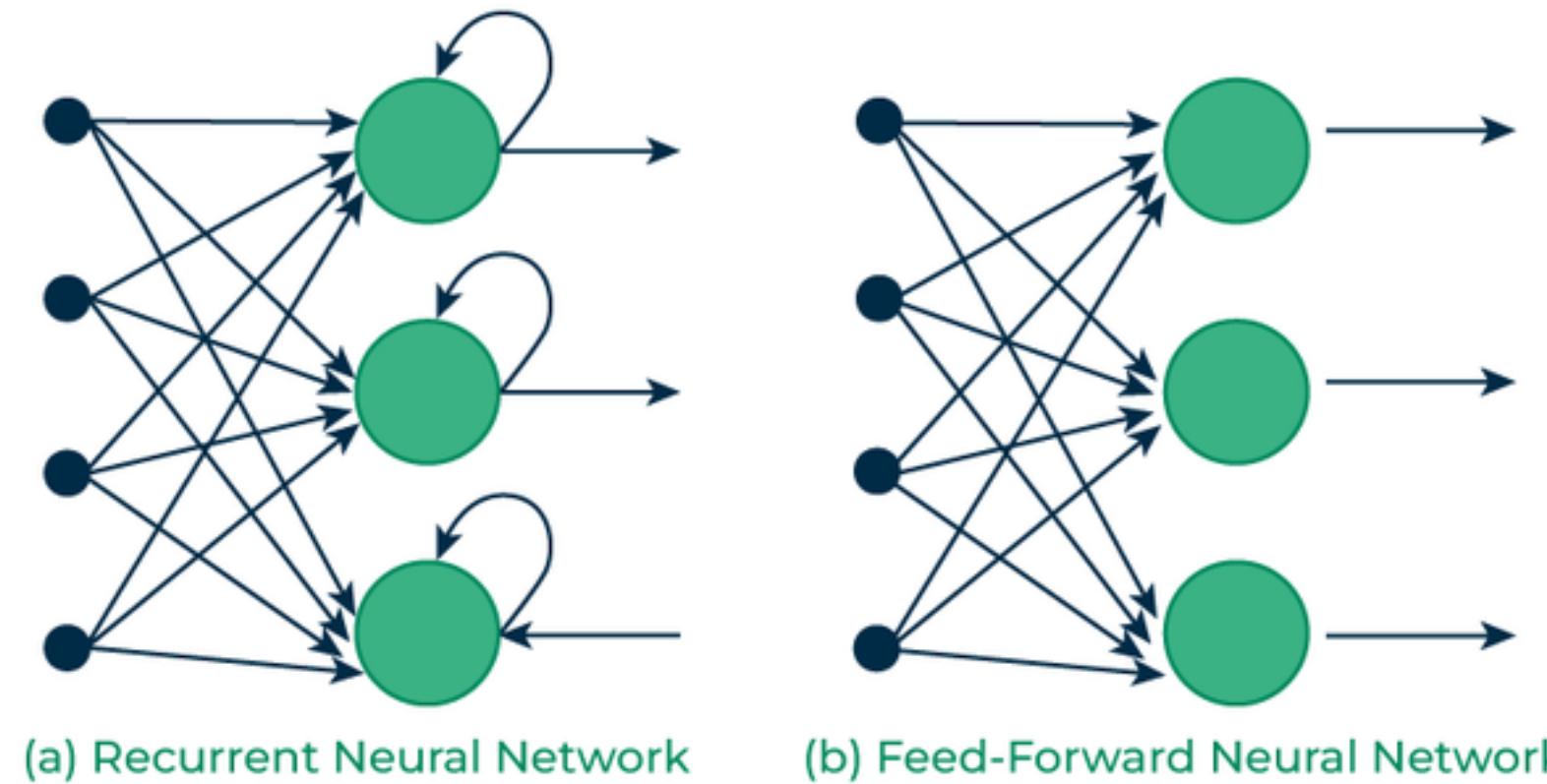
Generative Adversarial Networks



Diffusion Models



Recurrent Neural Networks



A Simple RNN Language Model

output distribution

$$\hat{y}^{(t)} = \text{softmax}(\mathbf{U}\mathbf{h}^{(t)} + \mathbf{b}_2) \in \mathbb{R}^{|V|}$$

Core idea: Apply the same weights \mathbf{W} repeatedly

hidden states

$$\mathbf{h}^{(t)} = \sigma(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_e \mathbf{e}^{(t)} + \mathbf{b}_1)$$

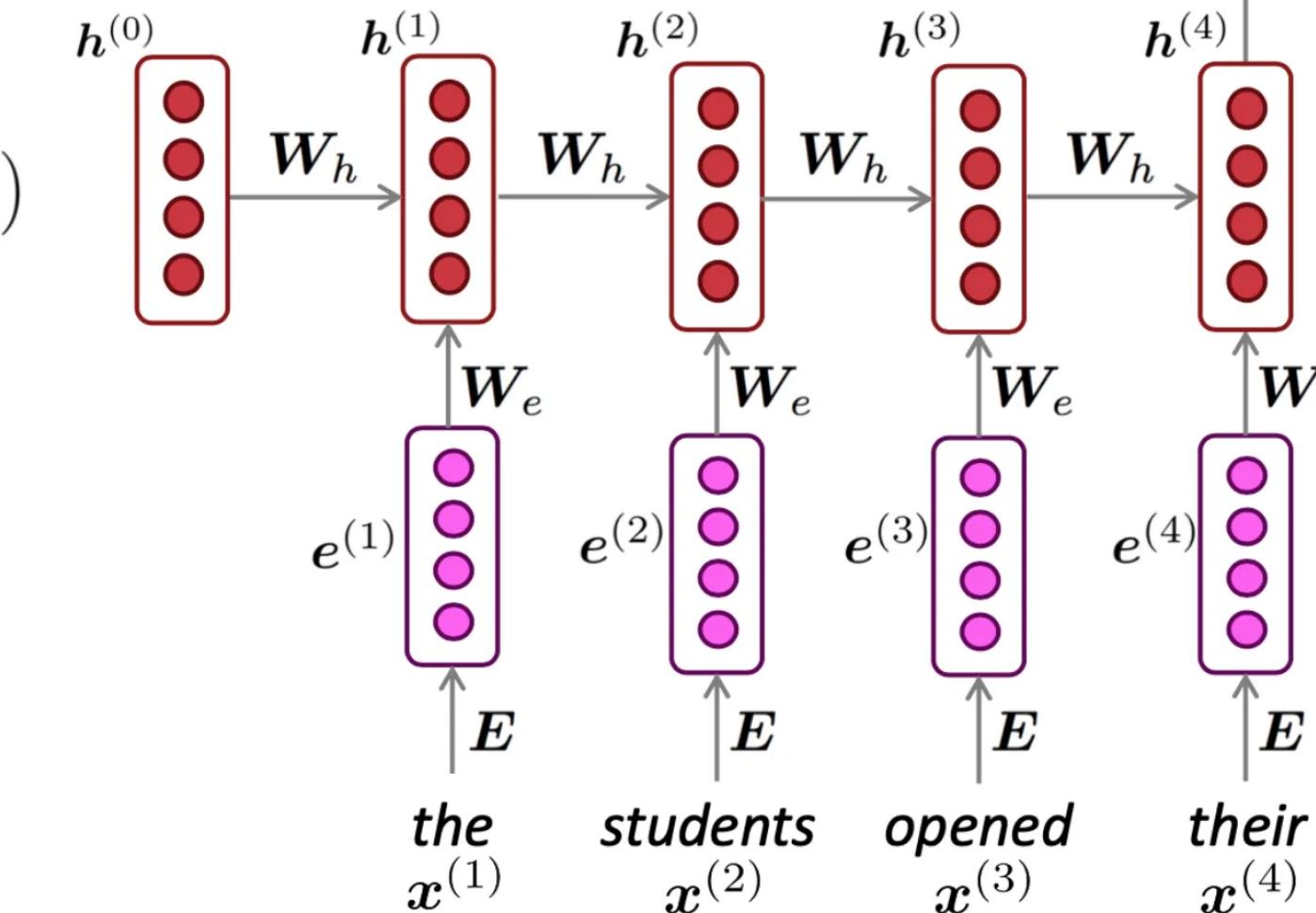
$\mathbf{h}^{(0)}$ is the initial hidden state

word embeddings

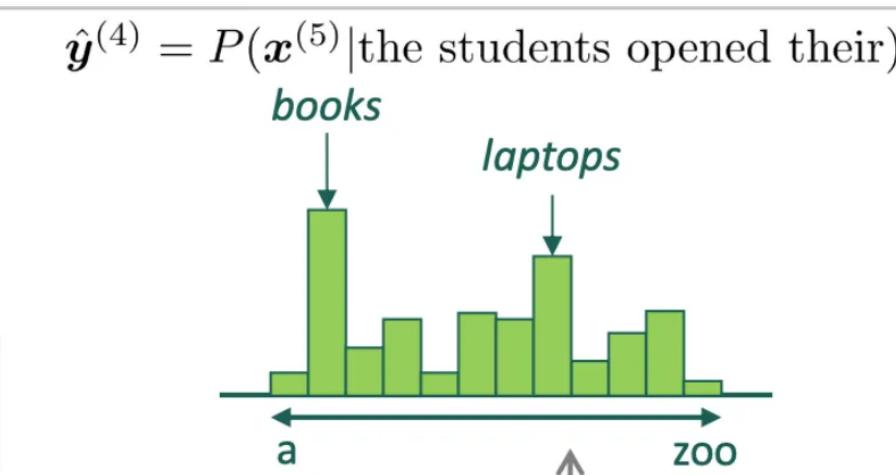
$$\mathbf{e}^{(t)} = \mathbf{E} \mathbf{x}^{(t)}$$

words / one-hot vectors

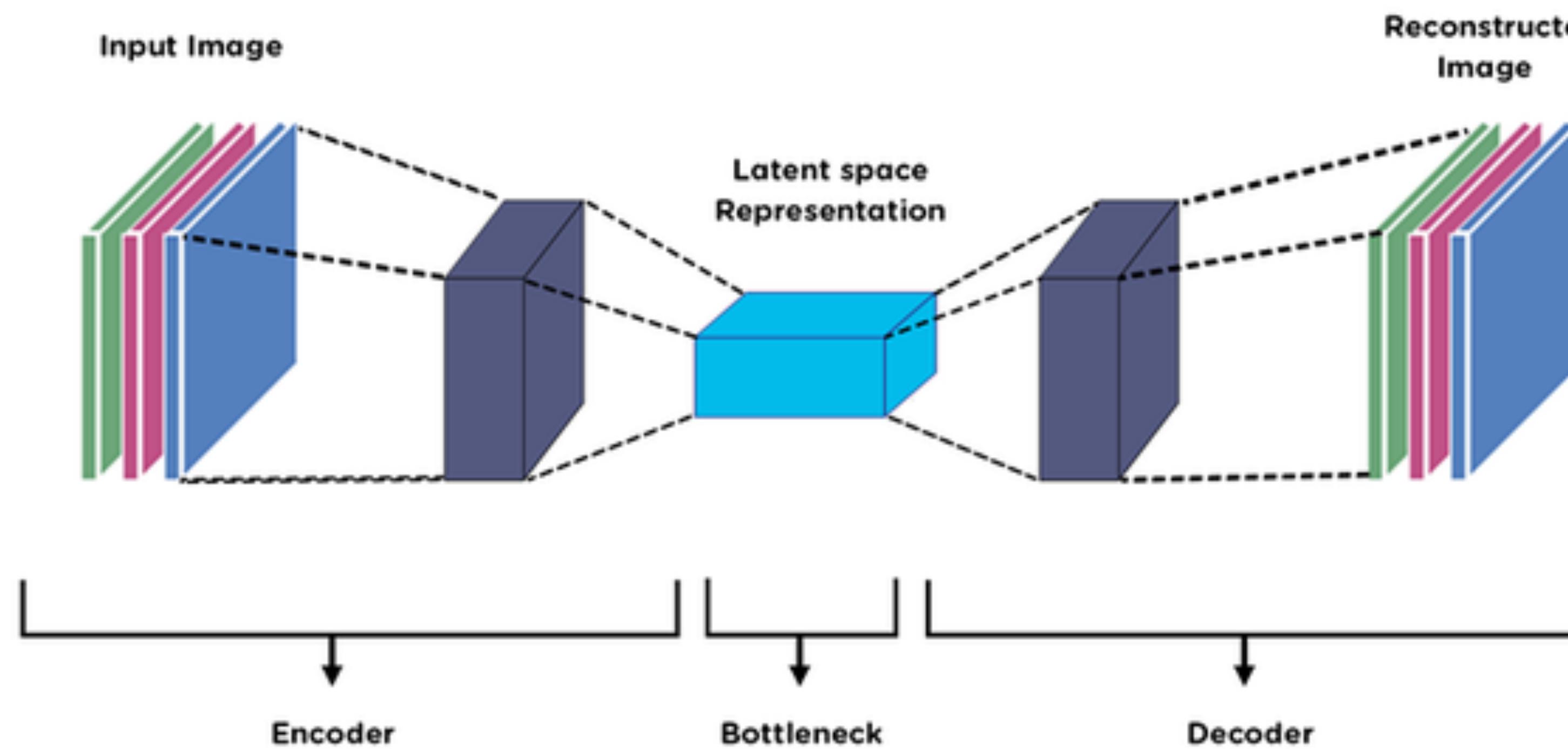
$$\mathbf{x}^{(t)} \in \mathbb{R}^{|V|}$$



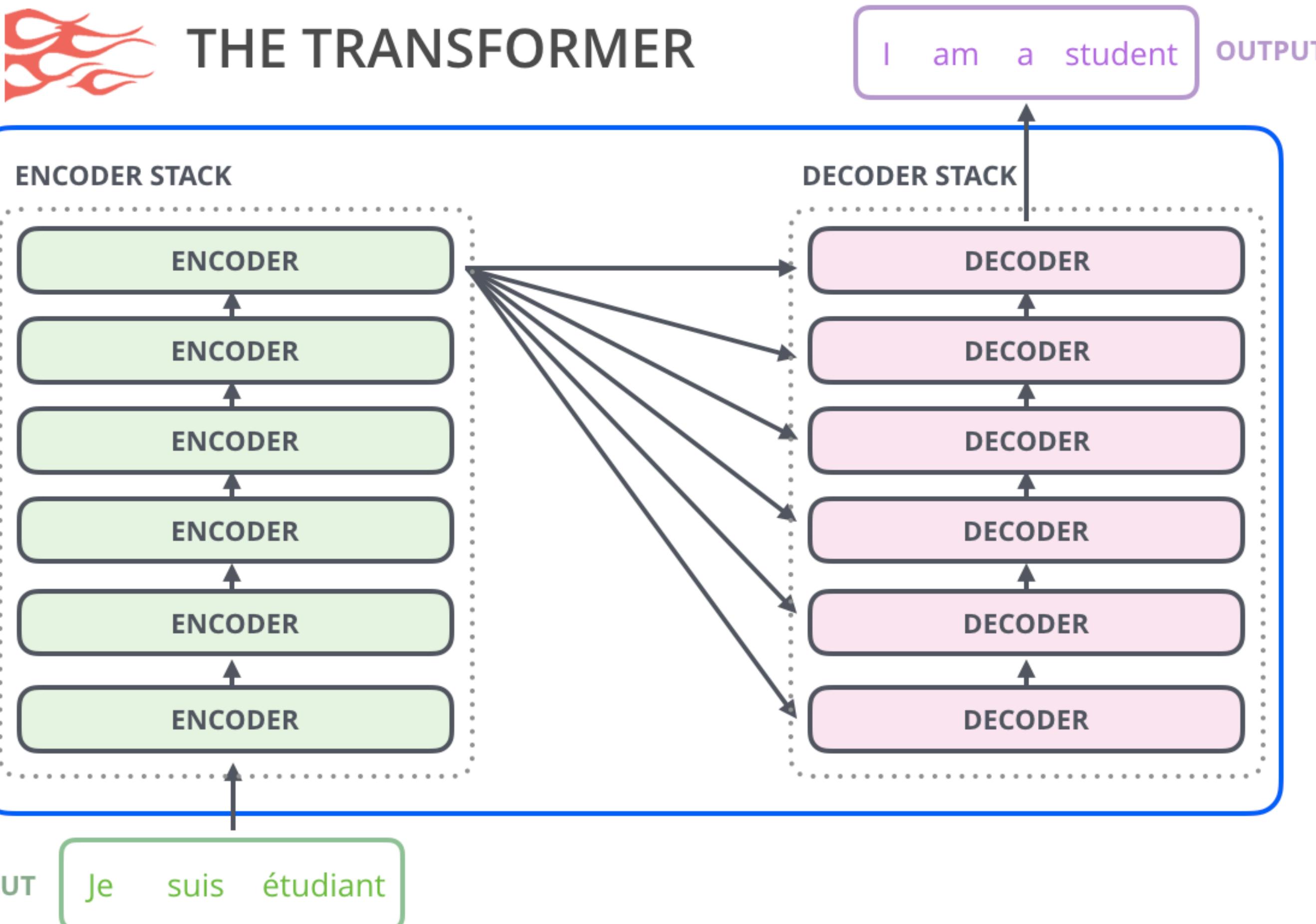
Note: this input sequence could be much longer now!



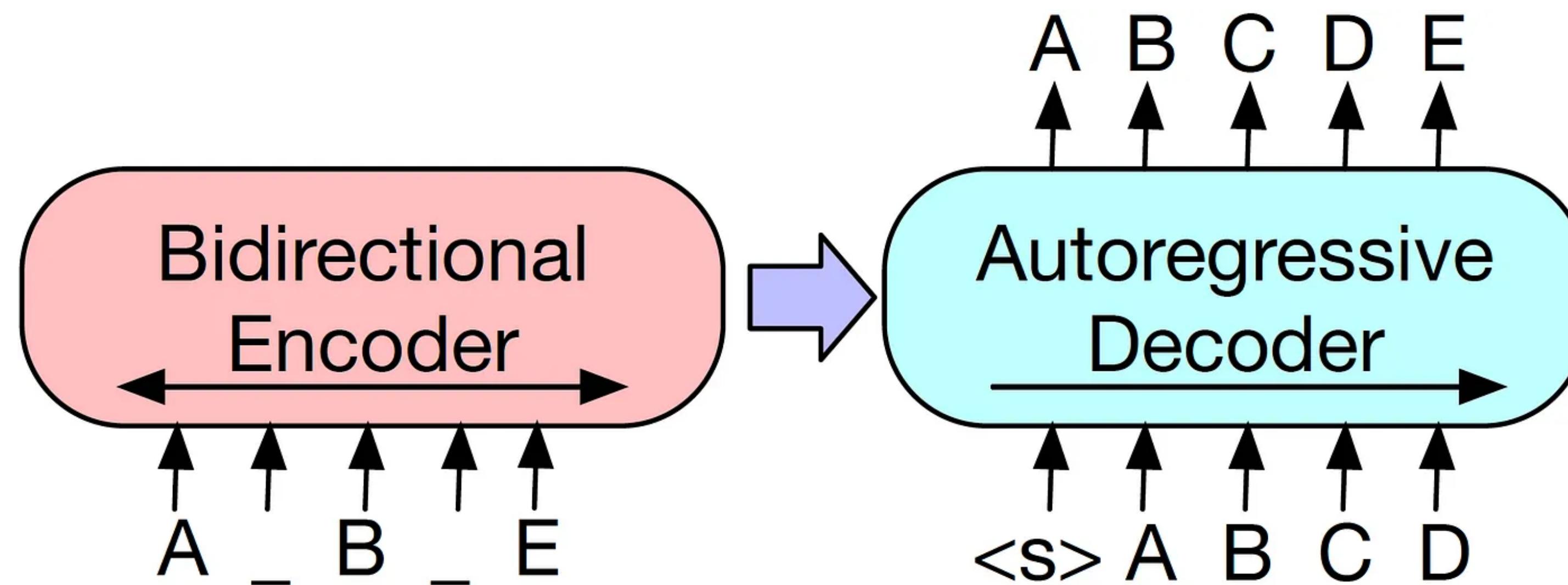
Auto encoders

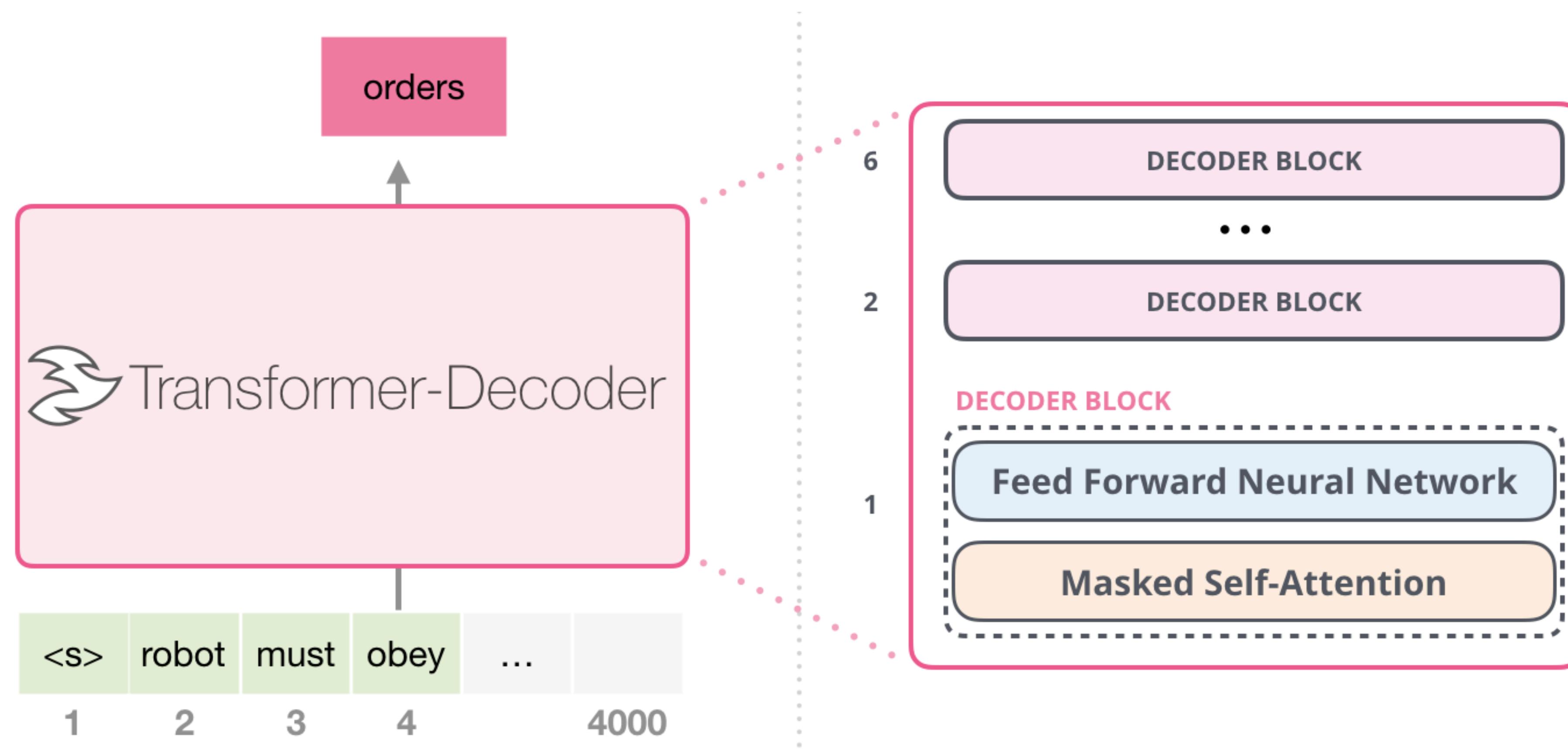


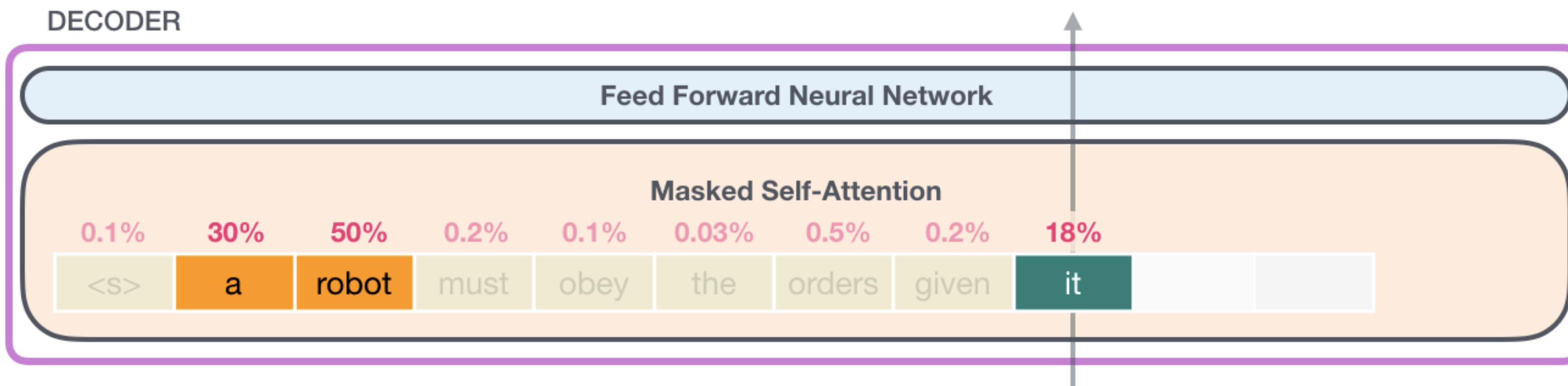
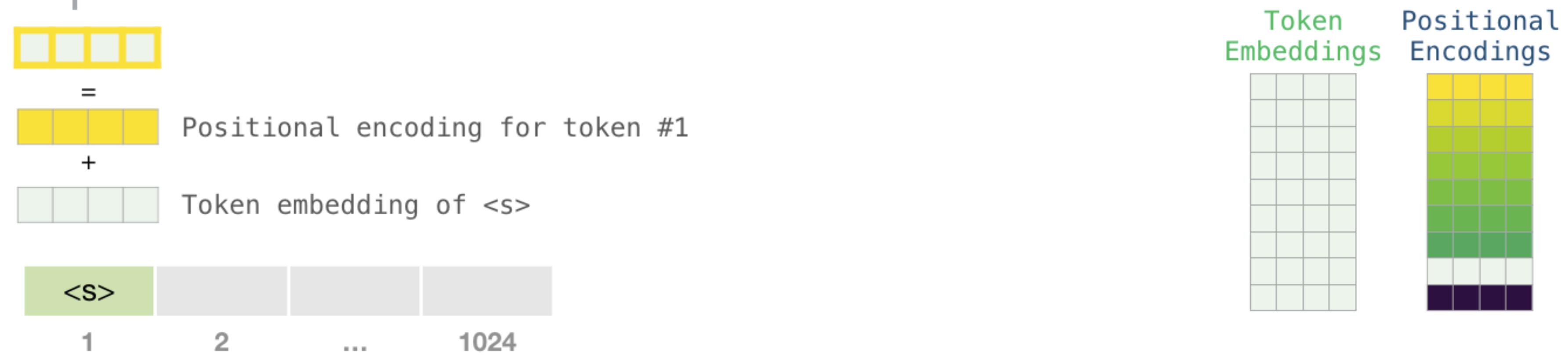
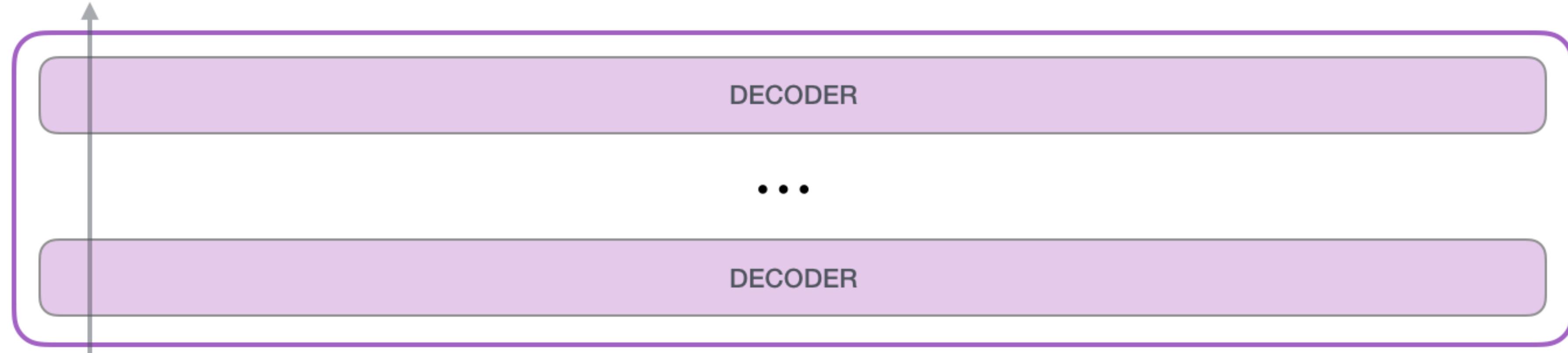
Transformers



Transformers







Cell that turns on inside quotes:

"You mean to imply that I have nothing to eat out of.... On the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

Cell sensitive to position in line:

The sole importance of the crossing of the Berezina lies in the fact that it plainly and indubitably proved the fallacy of all the plans for cutting off the enemy's retreat and the soundness of the only possible line of action--the one Kutuzov and the general mass of the army demanded--namely, simply to follow the enemy up. The French crowd fled at a continually increasing speed and all its energy was directed to reaching its goal. It fled like a wounded animal and it was impossible to block its path. This was shown not so much by the arrangements it made for crossing as by what took place at the bridges. When the bridges broke down, unarmed soldiers, people from Moscow and women with children who were with the French transport, all--carried on by vis inertiae--pressed forward into boats and into the ice-covered water and did not, surrender.

Current state of LLMs

- LLMs seem to be good at quite general purpose tasks.
- Their general structure is outputting the next “token” or “word”
- They use a few tricks and slightly different structures than we are used to
- The question is - how do we leverage what we know about the learning process to have better outputs?
 - Garbage in, garbage out
 - Gold in, gold out

Ways to use LLMs in practice

- Run open models on your own laptop - needs investment, slow, but private
- Run open models on servers - continuing cost, somewhat private
- Use models hosted by others - cheaper, fast, but generally speaking not very private
 - Varying degrees

What about reasoning?

- Being descriptive helps in the input process, it should therefore also help in the output process

“Normal” Training data example

- <user> What is the number of “r” in the word “strawberry” </user>
<agent>There are 3 “r” in strawberry </agent>
- <user> What is the number of “r” in the word “banana” </user>
<agent>There are 3 “r” in banana </agent>

w/ Reasoning Training data example

- <user> What is the number of “r” in the word “strawberry” </user>
<thinking> “strawberry” has the following letters “s” “t” “r” “a” “w” “b” “e” “y”
Their frequencies are: “s” - 1 ; “t” - 1 ; “r” - 3 ; “a” - 1 ; “b” - 1 ; “e” - 1 ; “y” - 1
</thinking>
<agent>There are 3 “r” in strawberry </agent>
- <user> What is the number of “r” in the word “banana” </user>
<thinking> “banana” has the following letters “b” “a” “n”
Their frequencies are: “b” - 1 ; “a” - 3 ; “n” - 2
</thinking>
<agent>There are 0 “r” in banana </agent>

Which do you think will perform better

- Generate some code for me - I want to use R. The context of the code is a single file qmd that runs through a simple ML model, that includes loading in a dataset, cleaning the data, visualising it, and then coming up with a couple of different simple models like random forest, or logistic regression, or linear regression etc, to try and model an output. It also should talk through the metrics for the models, and figure out which is the best one. Ideally including some sort of hyper parameter tuning as well. It should use GGplot, tidymodels, tidyverse, and maybe tidycensus/tigris/sf if needed.

- Create an educational R Quarto (qmd) document for a "ML and AI for Public Policy" course aimed at social science students. This document should serve as a teaching tool that demonstrates how machine learning can be applied to policy-relevant problems.
- Use the "ACS Public Use Microdata Sample" data from the tidycensus package to predict household income levels based on demographic and socioeconomic factors. This example will show students how ML can be used to identify patterns and potential disparities in economic outcomes.
- The document should:
 - 1. Introduction:
 - - Frame the analysis in terms of policy relevance (e.g., understanding income inequality, targeting assistance programs)
 - - Explain ethical considerations when using ML for policy decisions
 - - Discuss how predictive models can inform evidence-based policymaking
 - - List and explain the packages we'll use (tidyverse, tidymodels, ggplot2, tidycensus)
 - 2. Data acquisition and understanding:
 - - Show how to access and load census data using tidycensus
 - - Discuss the importance of understanding government data sources
 - - Explain sampling methodology and limitations of the data
 - - Connect the dataset to real-world policy questions
 - 3. Data cleaning and preparation:
 - - Handle missing values with discussion of how missing data may introduce bias
 - - Transform variables as needed (income brackets, categorical variables)
 - - Create derived features relevant to policy analysis
 - - Discuss how data preparation choices might impact different demographic groups
 - 4. Exploratory analysis through a policy lens:
 - - Create visualizations showing income distributions across geographic regions or demographic groups
 - - Explore relationships between education, employment, demographics, and income
 - - Discuss how visualizations can effectively communicate findings to policymakers
 - - Include maps using sf for geographic analysis of disparities
 - 5. Feature selection and engineering:
 - - Discuss which variables are appropriate to include in a policy context
 - - Address potential sources of bias in feature selection
 - - Create policy-relevant interaction terms
 - - Explain the balance between model complexity and interpretability for policymakers

- 6. Model building:
 - - Split data with discussion of representative sampling
 - - Implement multiple models with increasing complexity:
 - - Linear Regression (baseline for interpretability)
 - - Random Forest
 - - Boosted Trees
 - - Discuss tradeoffs between model complexity and interpretability in policy contexts
- 7. Hyperparameter tuning:
 - - Demonstrate tuning for Random Forest and Boosted Trees
 - - Explain computational resource considerations for government applications
 - - Visualize how parameter choices affect model performance
- 8. Model evaluation with policy implications:
 - - Evaluate models using metrics relevant to policy decisions
 - - Assess fairness across demographic groups
 - - Analyze which features drive predictions (feature importance)
 - - Discuss how to communicate model uncertainty to policymakers
- 9. Final model selection and policy application:
 - - Select best model considering both performance and interpretability
 - - Demonstrate how to use the model for policy scenario analysis
 - - Discuss limitations and potential unintended consequences
 - - Explore how the model could inform specific policy interventions
- 10. Ethical considerations and governance:
 - - Address potential biases in the model
 - - Discuss responsible AI principles for public sector use
 - - Consider transparency requirements for algorithmic decision-making
 - - Outline governance frameworks for ML in public policy
- Throughout the document, include "Policy Insight" sections that connect technical concepts to public policy applications and ethical considerations.