
Building Products using Large Language Models (LLMs)

• • •

Build stuff with LLMs!



Building Products using Large Language Models (LLMs)

• • •

Build stuff with LLMs!

Cohan Sujay Carlos
CEO, Mechanical Chef
formerly CEO, Aiaioo Labs*
(about 24 years of studying AI)



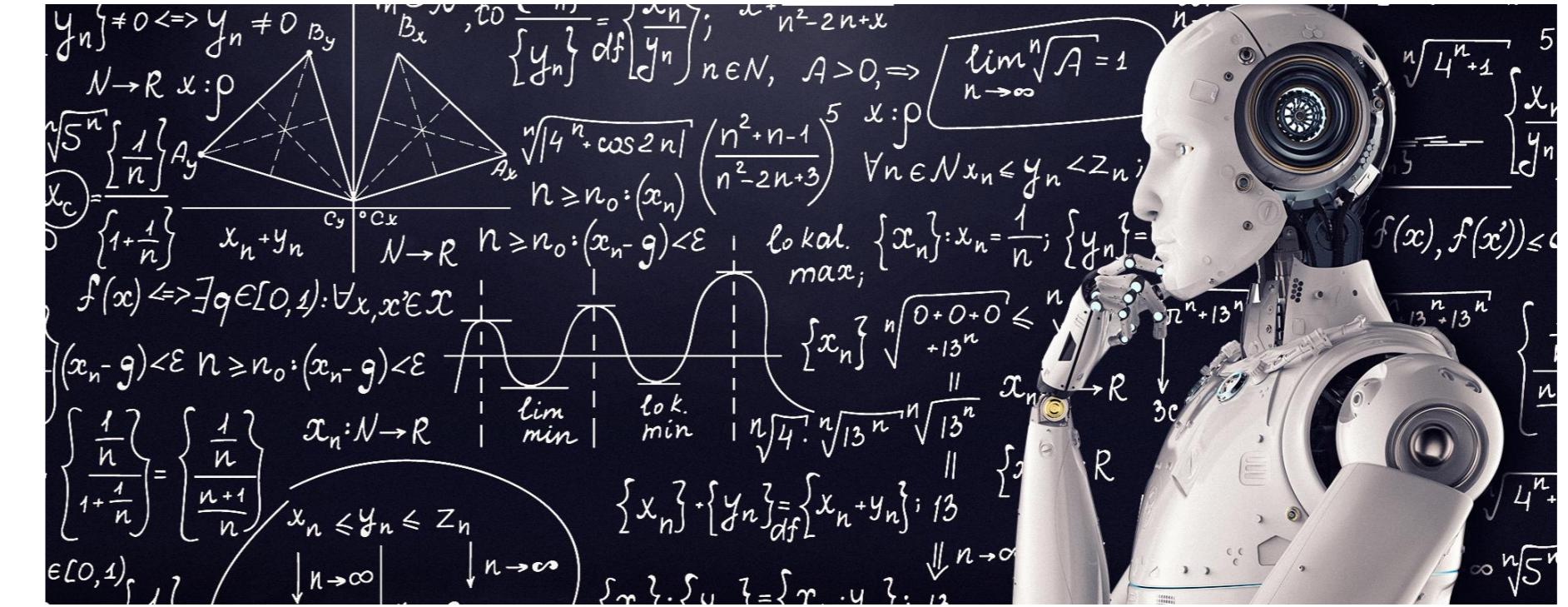
Arvind Singh Gulati
CTO, ex-Bicycle AI
& Researcher at Aiaioo Labs*
(12 years of experience)



Agenda

Session:

What will you learn in this lecture?



Wednesday, Sep 4th | 9:10am - 9:55am

- 1. What's an LLM? Some ML background.**
- 2. How can you use an LLM in a product?**
- 3. Example – build an order-taking bot.**

You'll learn what Large Language Models are and how to use them.

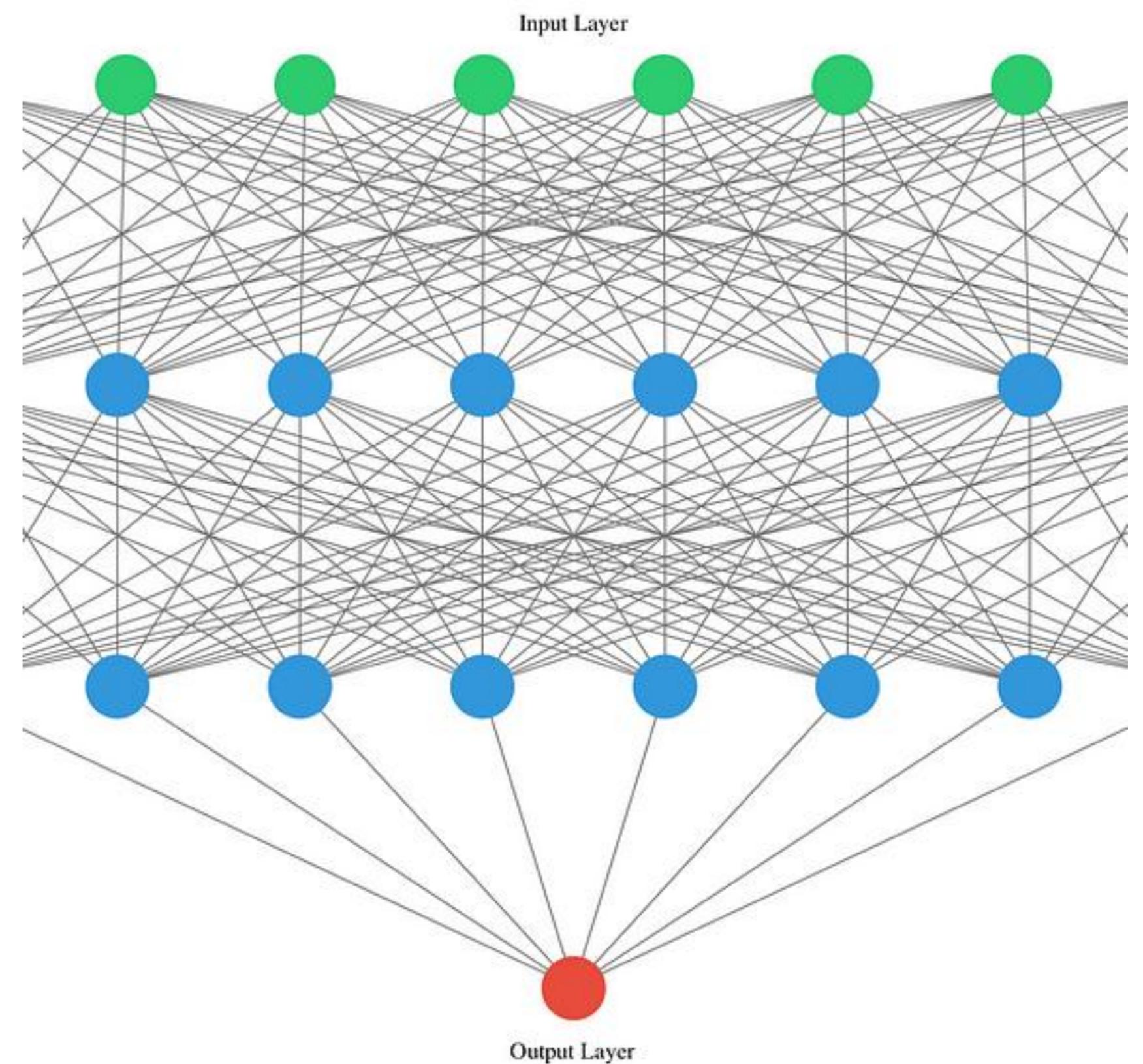
Introduction

What are Large Language Models?

Neural Networks that:

1. Understand **language** or sequential data
2. Have a **large** number of **neurons** (> 1 billion?)
3. Using self-supervised learning (unlabelled data)
4. Then are 'aligned' to human intent (using labelled data)

Artificial Neural network - Model Visualization



Neural Networks

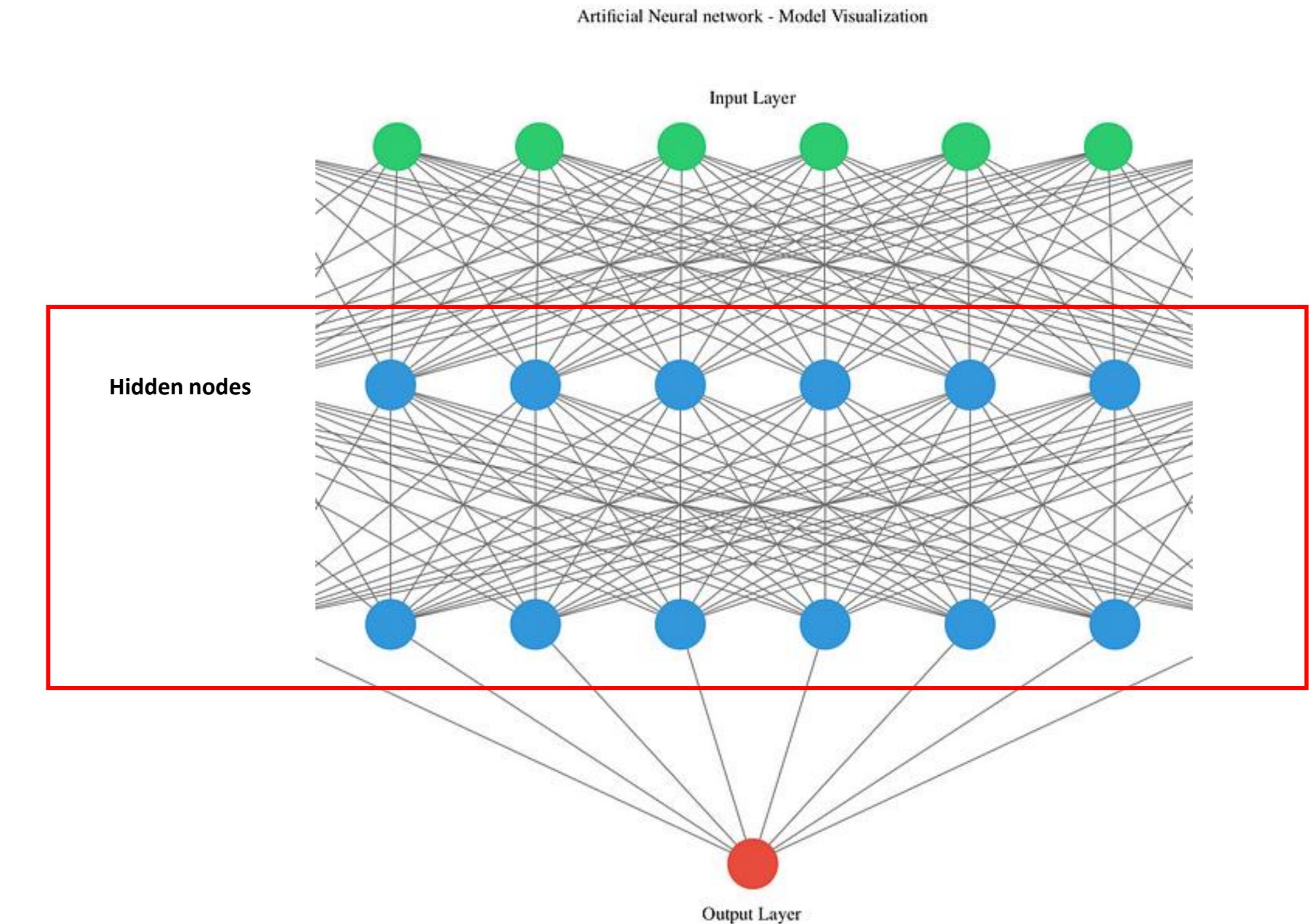
How do neural networks work?

Input nodes (green) are floating point numbers.

Output nodes (red) are floating point numbers.

Hidden nodes (blue) are floating point numbers.

These are arranged in layers.

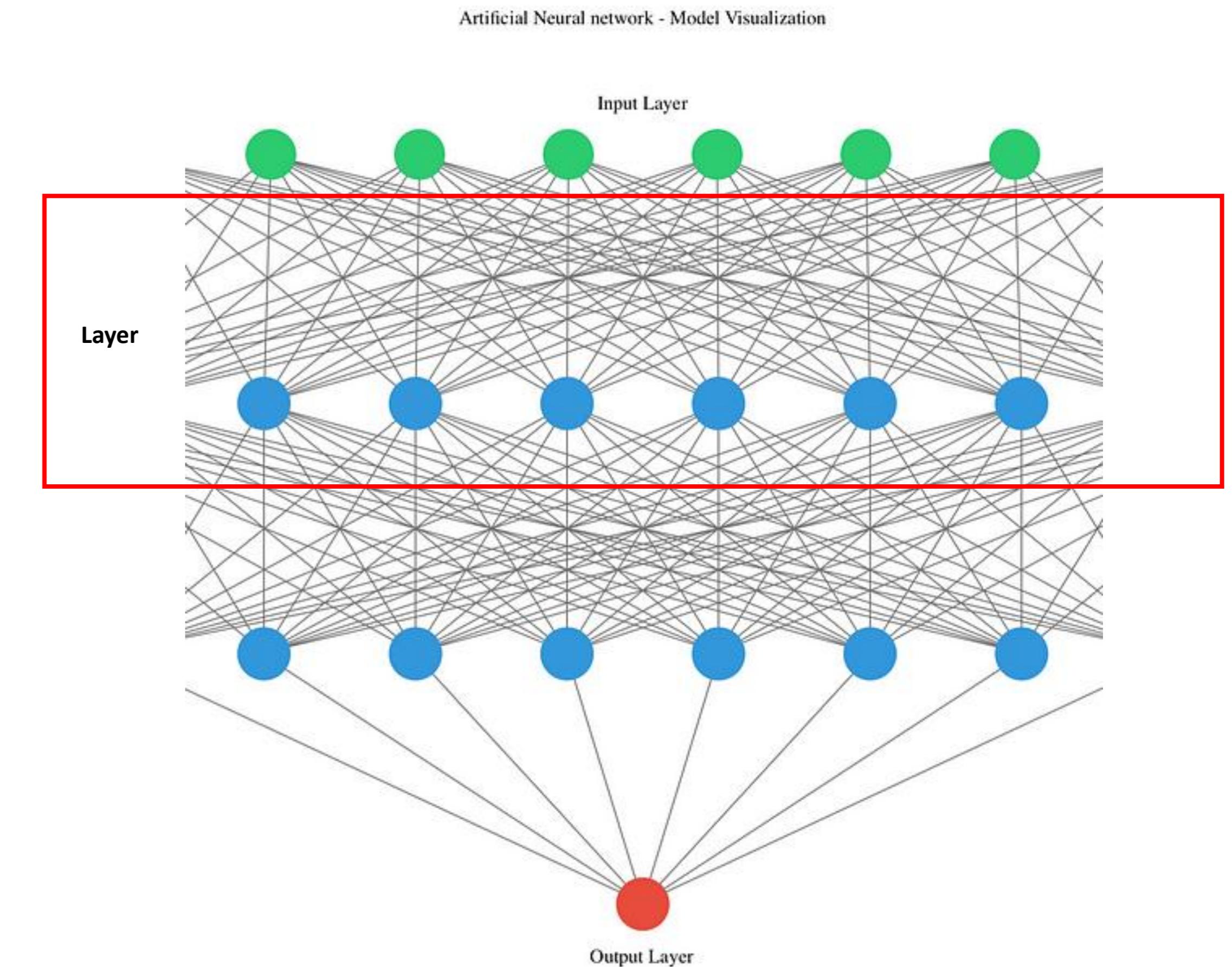


Neural Networks

How do neural networks work?

At each layer you multiply some weights.

Then squish the resulting numbers a little.



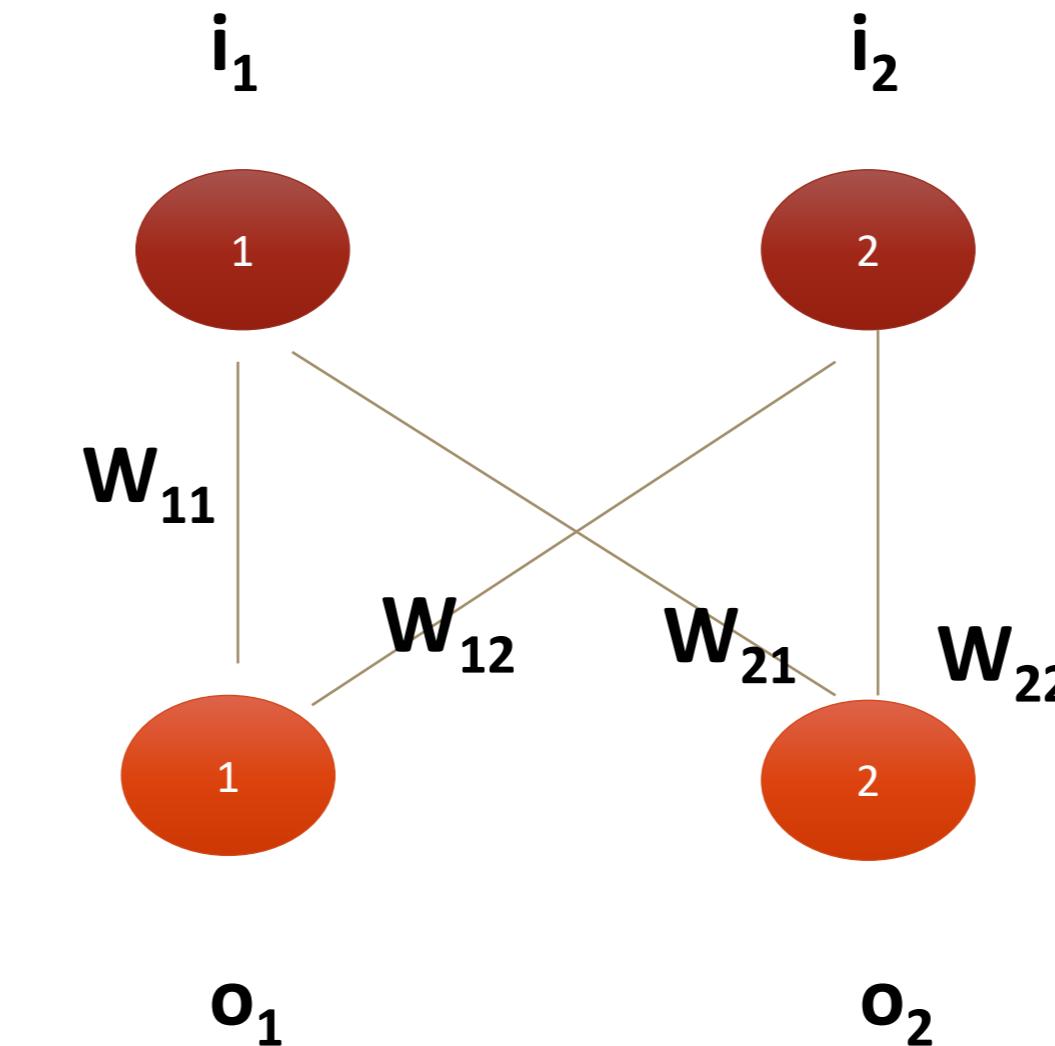
Neural Network – What happens in a layer of neurons

First you multiply the weights

Inputs = i

Neurons = W

Outputs = o



$$i_1 = 1$$

$$i_2 = 2$$

$$W_{11} = 3 \quad W_{12} = 4$$

$$W_{21} = -7 \quad W_{22} = 1$$

$$o_1 = 1 * 3 + 2 * 4 = 11$$

$$o_2 = 1 * -7 + 2 * 1 = -5$$

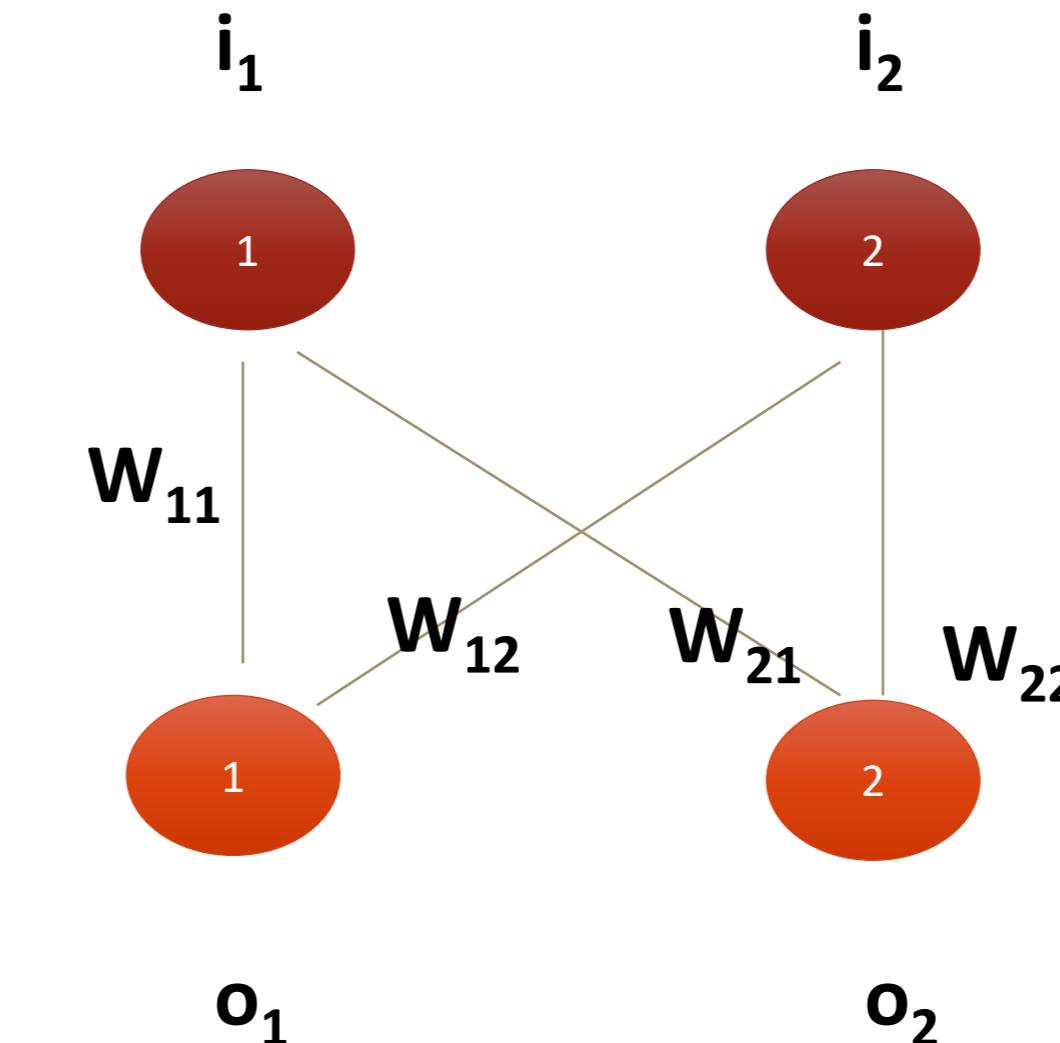
Neural Network – What happens in a layer of neurons

Then you squish the results a little

Inputs = i

Neurons = W

Outputs = o



$$i_1 = 1$$

$$i_2 = 2$$

$$W_{11} = 3 \quad W_{12} = 4$$

$$W_{21} = -7 \quad W_{22} = 1$$

$$o_1 = 1 * 3 + 2 * 4 = 11$$

$$o_2 = 1 * -7 + 2 * 1 = -5$$

$$\text{squished_}o_1 = \max(0, 11) = 11$$

$$\text{squished_}o_2 = \max(0, -5) = 0$$

Neural Networks

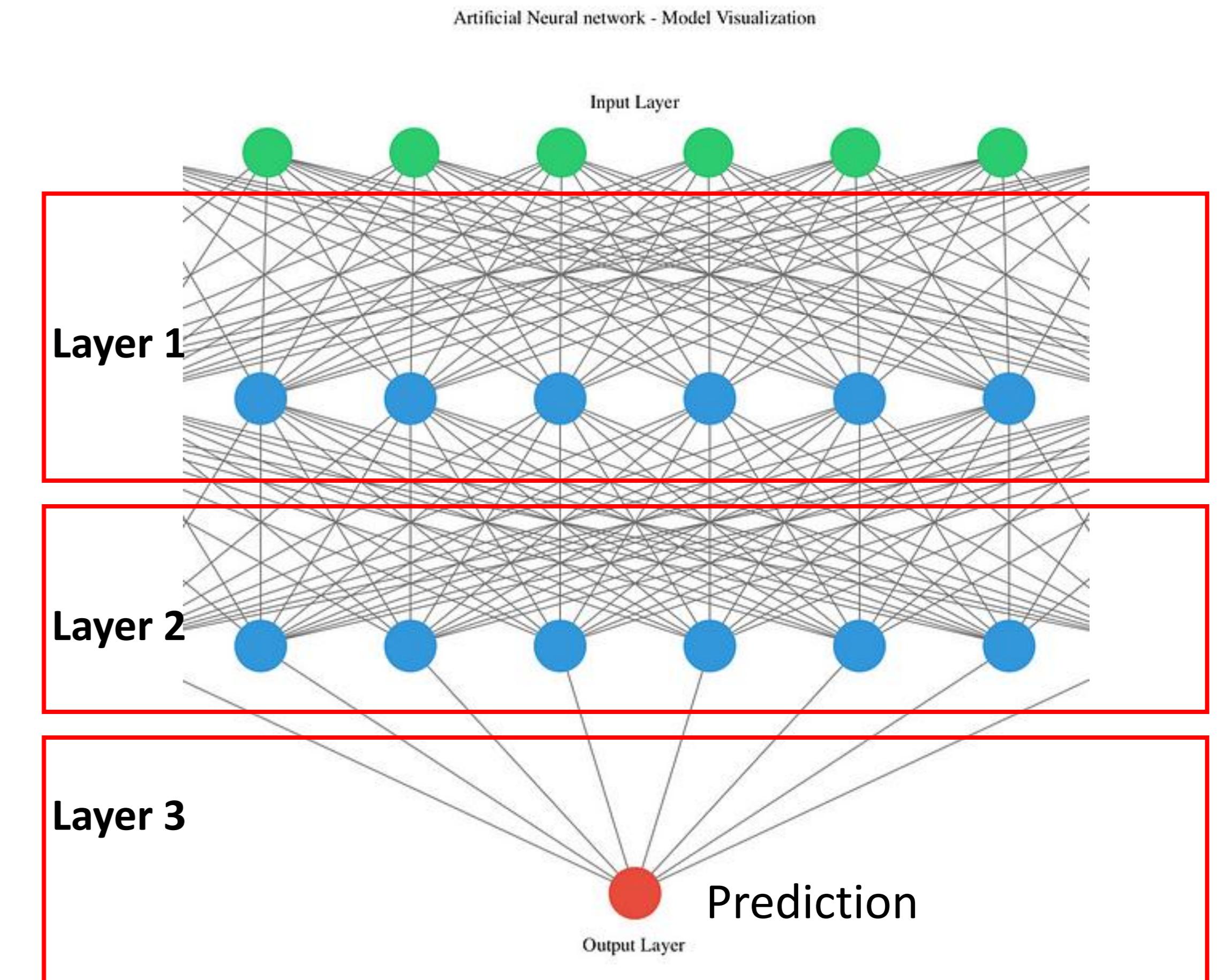
How do neural networks work?

So the weights are the lines.

The multiplied and squished numbers are the circles.

You repeat till the final layer.

The number in the last layer is the prediction.



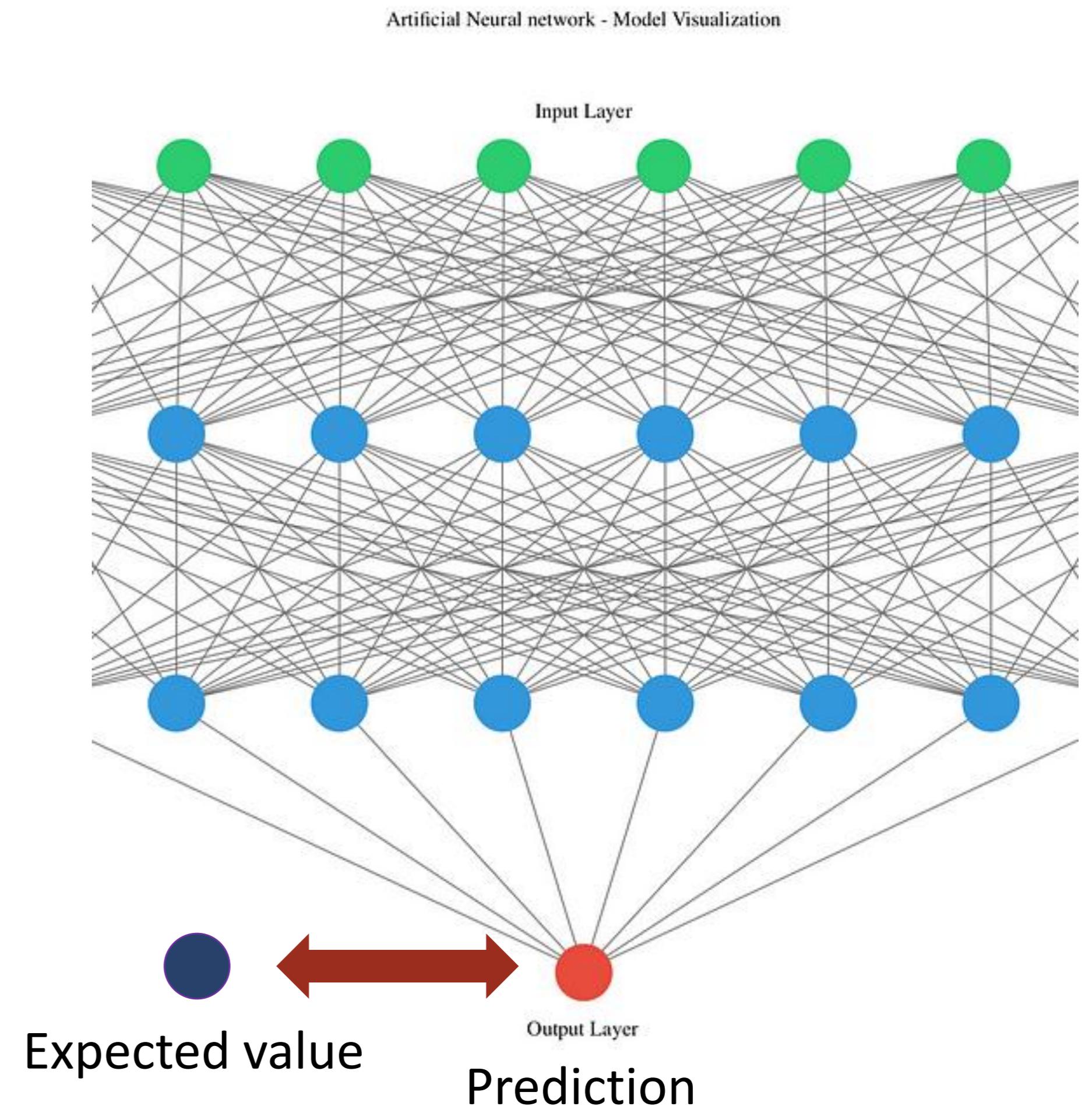
Neural Networks

How do you train a neural network?

You compare the prediction with an expected value.

If they don't match, you readjust the weights!

How?



Neural Networks

How do you train a neural network?

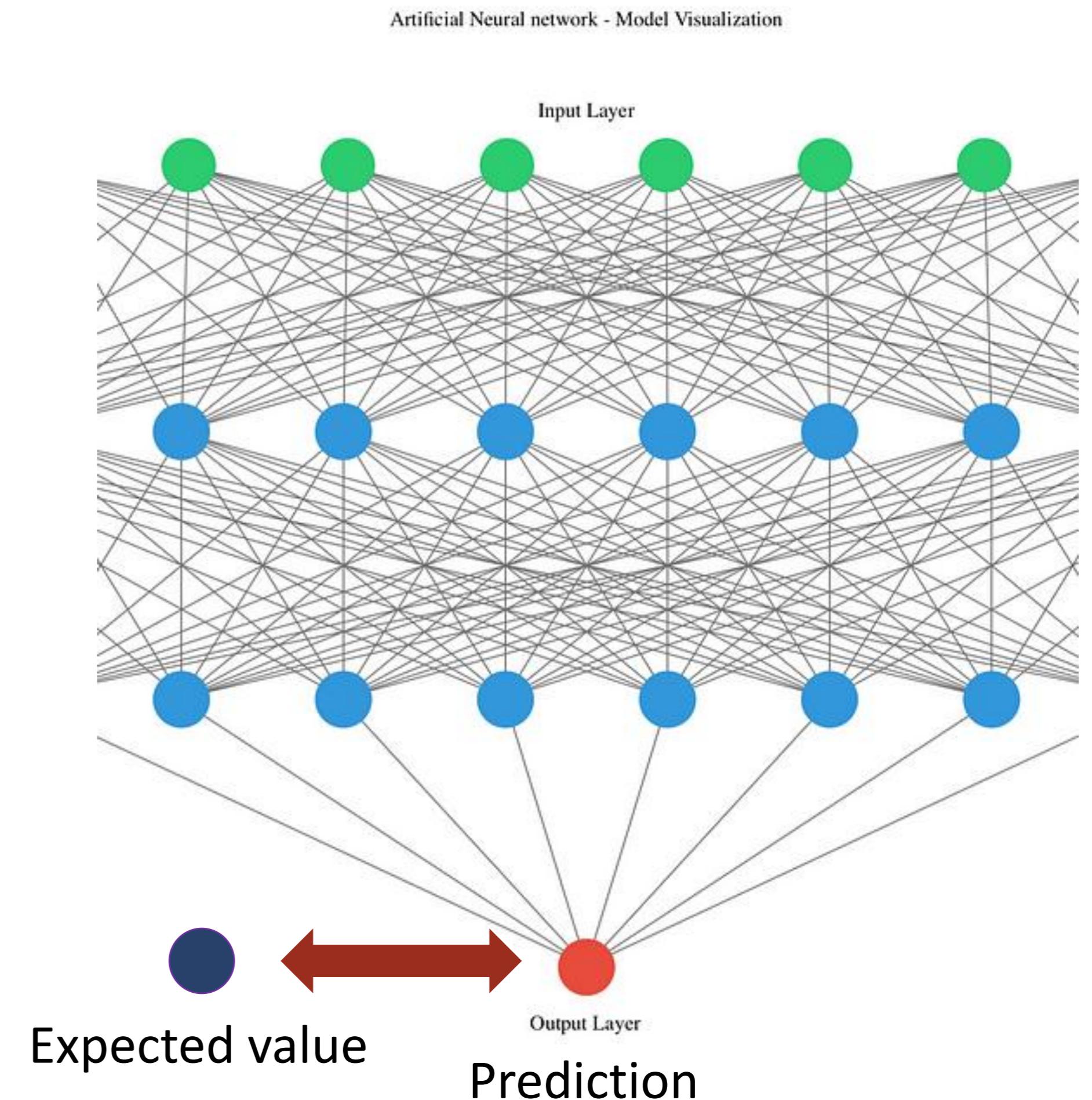
How does one readjust the weights?

1. You know the direction in which you want the prediction to move.

In this example, in the direction of expected – predicted (this is called the loss)

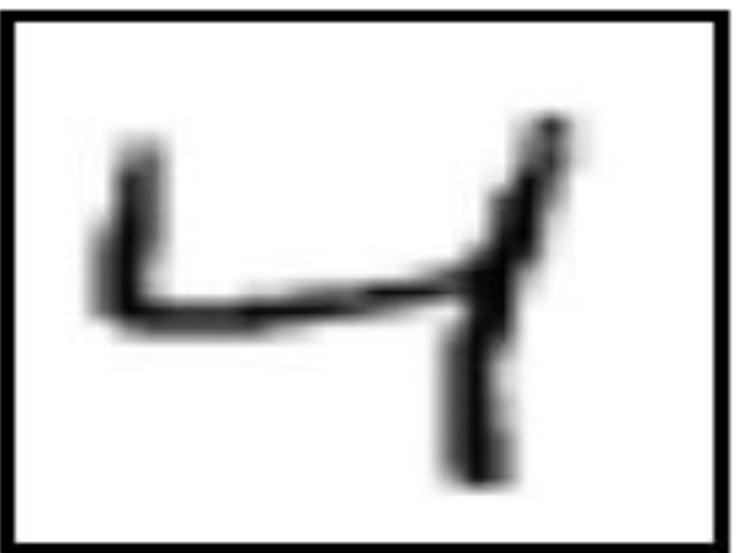
2. You can use differentiation to tell you in which direction to nudge the weights.

$$- \frac{d(\text{expected} - \text{predicted})}{d(W)}$$



The input to an NN can represent pretty much anything

Images can be turned into a 2D grid of pixels (a matrix of integers) for greyscale images (and 3 or 4 matrices for colour).



0	3	2	3	1
0	2	0	0	0
0	0	1	2	0
0	0	3	0	0
3	2	0	0	0

0	0	2	1	0
0	2	0	0	2
3	0	0	0	3
2	0	0	2	0
0	2	3	0	0

0	0	0	3	0
3	0	0	2	0
3	2	3	3	0
0	0	0	3	0
0	0	0	3	0

The input to an NN can represent pretty much anything

Words can be turned into 1D vectors of numbers

A sentence is a sequence of vectors, each representing a word

My	name	is	Arvind
0 3 2 3 1	0 0 2 1 0	0 0 0 3 0	0 0 0 0 2

These vectors of numbers representing a string are called 'embeddings'.

The input to an NN can represent pretty much anything

Sub-words can be turned into 1D vectors of numbers

A sentence is a sequence of vectors, each representing a sub-word

M	#y	na	#me	is	A	#r	#v	#ind
0 3 1 2 1	0 3 2 1 0	0 3 0 3 0	0 0 3 0 2	0 4 3 0 2	0 4 3 0 2	0 1 3 0 2	0 1 0 0 2	1 5 0 0 2

The input to an NN can represent pretty much anything

Sub-words can be turned into 1D vectors of numbers

A sentence is a sequence of vectors, each representing a sub-word

M	#y	na	#me	is	A	#r	#v	#ind
0 3 1 2 1	0 3 2 1 0	0 3 0 3 0	0 0 3 0 2	0 4 3 0 2	0 4 3 0 2	0 1 3 0 2	0 1 0 0 2	1 5 0 0 2

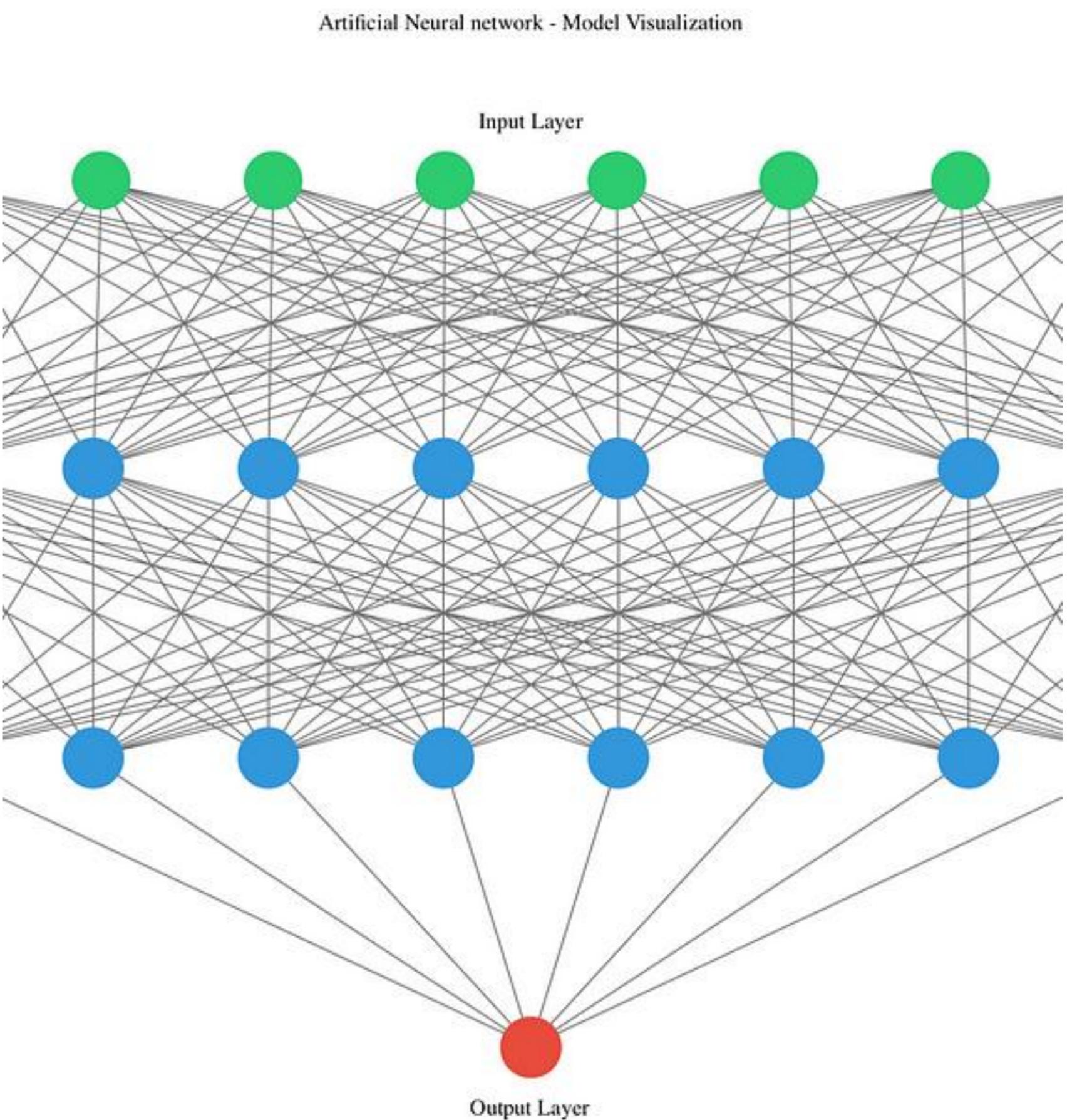
Currently most LLMs would use something like this.

As long as something can be turned into numbers, NNs can deal with it.

What are Large Language Models?

Neural Networks that:

1. Understand **language** or sequential data
2. Have a **large** number of **neurons** (> 1 billion?)
3. Using self-supervised learning (unlabelled data)
4. Then are 'aligned' to human intent (using labelled data)



Language models

Given: $s = \text{"the cat caught the mouse"}$

Come up with a function $P(\text{"the cat caught the mouse"})$ [aka the language model]

Such that $P(\cdot)$ has the following property ...

That it rewards good sentences with a higher probability score than bad ones.

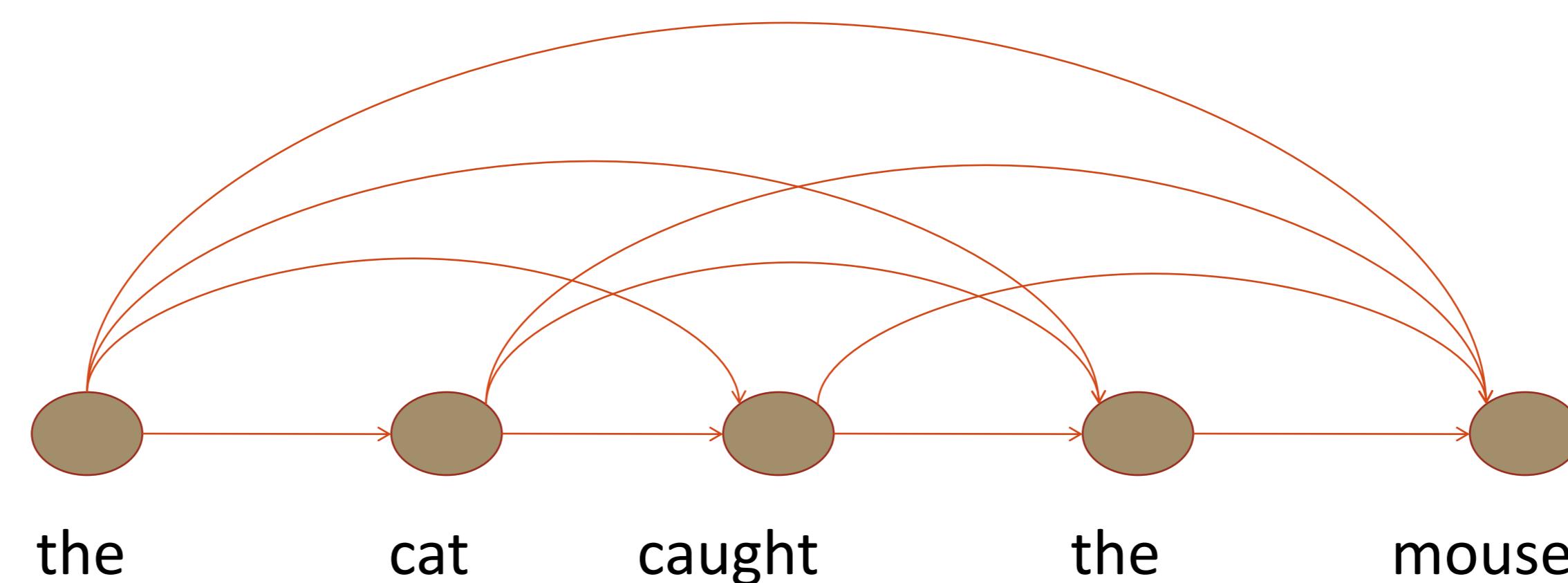
Properties of a good language model:

$P(\text{"cat mouse the the caught"}) \ll P(\text{"the cat caught the mouse"})$

$P(\text{"the banana ate the computer"}) \ll P(\text{"the man ate the banana"})$

Language Model

$$\begin{aligned} P(\text{"the cat caught the mouse"}) &= P(\text{"mouse"} | \text{"the cat caught the"}) \\ &\quad *P(\text{"the"} | \text{"the cat caught"}) \\ &\quad *P(\text{"caught"} | \text{"the cat"}) \\ &\quad *P(\text{"cat"} | \text{"the"}) *P(\text{"the"}) \end{aligned}$$



Language Model – with Markov Assumption

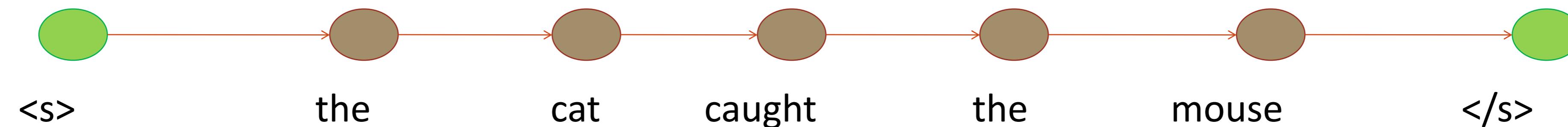
$$P(\text{"the cat caught the mouse"}) = P(\text{"mouse"} | \text{"the"})$$

$$*P(\text{"the"} | \text{"caught"})$$

$$*P(\text{"caught"} | \text{"cat"})$$

$$*P(\text{"cat"} | \text{"the"})$$

$$*P(\text{"the"} | \text{<start>})$$

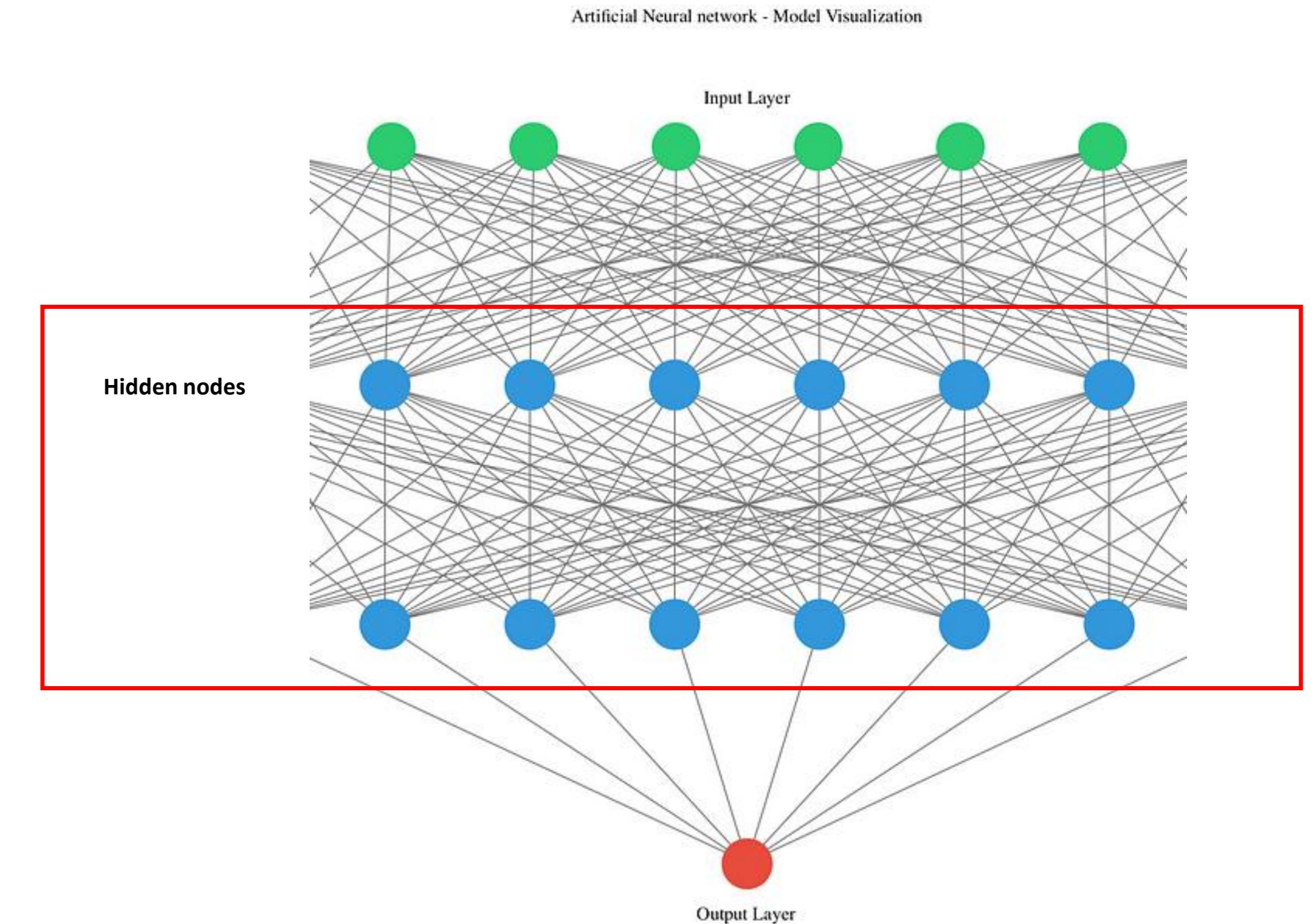


Formalism 2

Neural Networks

Can process words sequentially ...

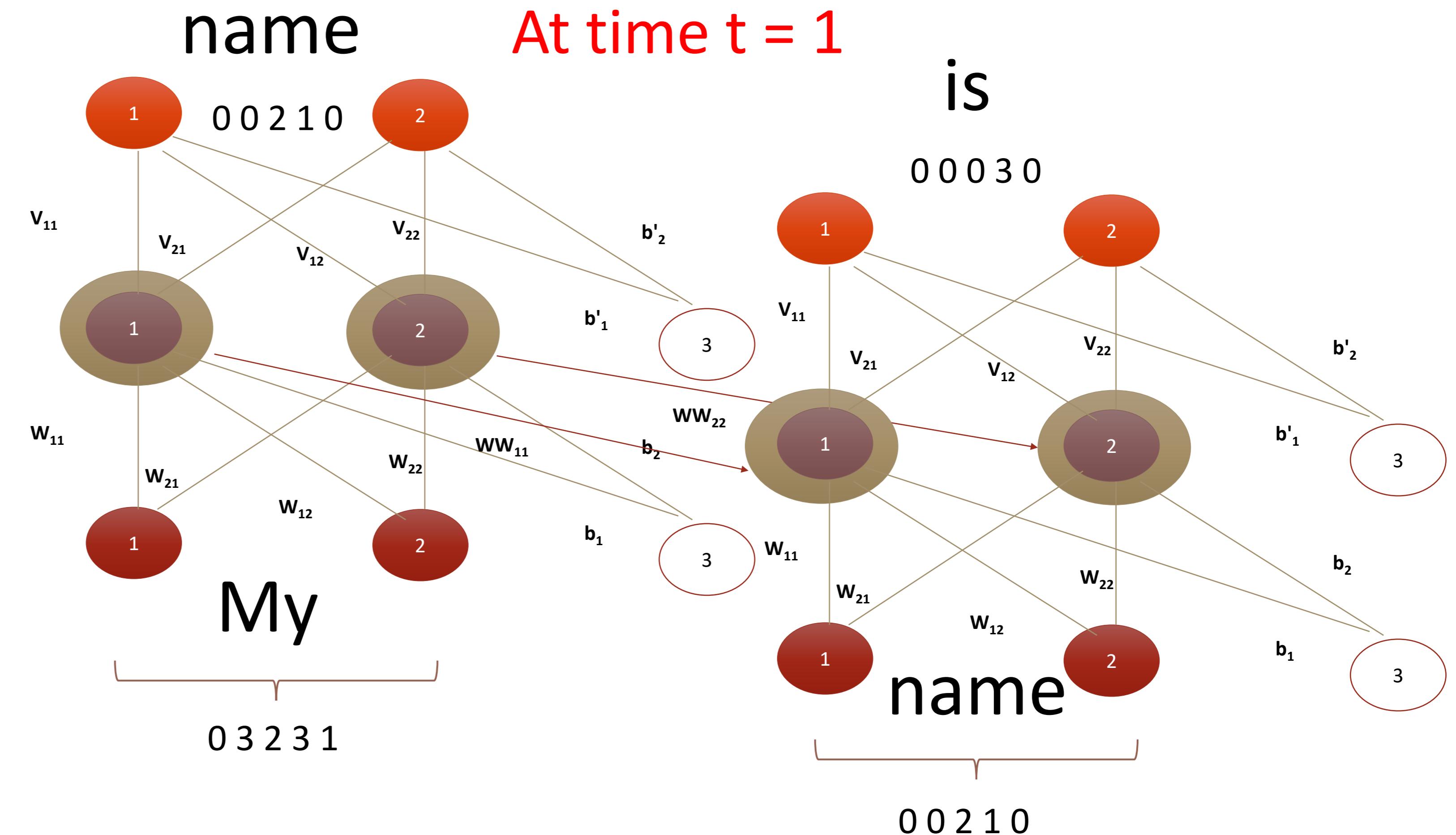
**by considering some or all of the
hidden node values (blue) as inputs
to the next time step (acting as a kind of
memory).**



Sequential models can be used as Language Models

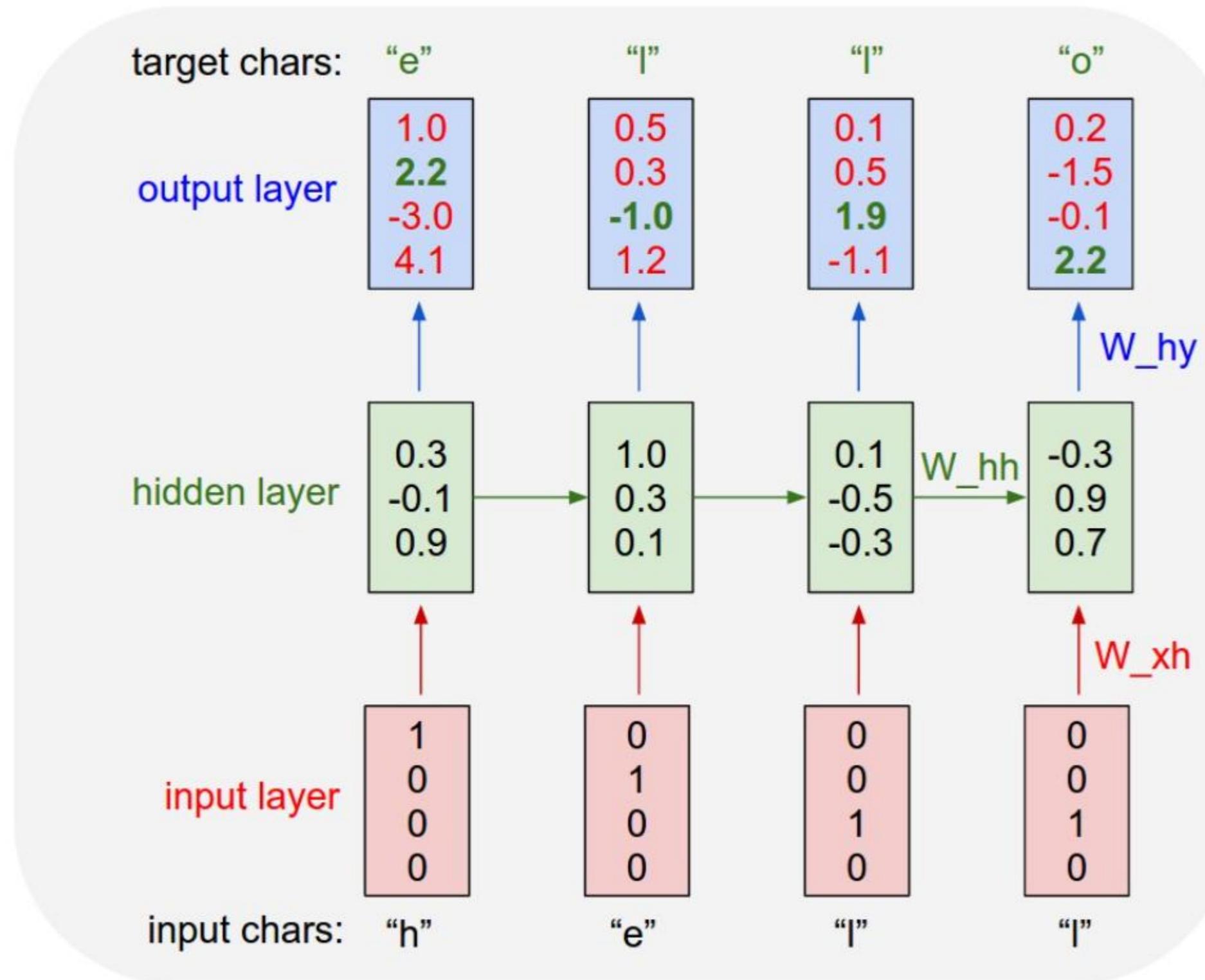
Recurrent Neural Networks (RNNs)

When **the inputs are text** and **the outputs are future text**, you get a **language model**, a model that predicts what you are going to say to it next.



Language Model

From Andrej Karpathy's 2015 article titled “The Unreasonable Effectiveness of Recurrent Neural Networks”.



Source: <https://karpathy.github.io/2015/05/21/rnn-effectiveness/>

PANDARUS:

Alas, I think he shall be come approached and the day
When little strain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:

They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
    int error;
    if (fd == MARN_EPT) {
        /*
         * The kernel blank will coeld it to userspace.
         */
        if (ss->segment < mem_total)
            unblock_graph_and_set_blocked();
        else
            ret = 1;
        goto bail;
    }
}
```

Language Model

From Andrej Karpathy's 2015 article titled “The Unreasonable Effectiveness of Recurrent Neural Networks”.

Proof. Omitted. □

Lemma 0.1. Let \mathcal{C} be a set of the construction.

Let \mathcal{C} be a gerber covering. Let \mathcal{F} be a quasi-coherent sheaves of \mathcal{O} -modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

Proof. This is an algebraic space with the composition of sheaves \mathcal{F} on $X_{\text{étale}}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{\text{morph}_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where \mathcal{G} defines an isomorphism $\mathcal{F} \rightarrow \mathcal{G}$ of \mathcal{O} -modules. □

Lemma 0.2. This is an integer \mathcal{Z} is injective.

Proof. See Spaces, Lemma ??.

Lemma 0.3. Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \rightarrow Y' \rightarrow Y \rightarrow Y \rightarrow Y' \times_X Y \rightarrow X.$$

be a morphism of algebraic spaces over S and Y .

Proof. Let X be a nonzero scheme of X . Let X be an algebraic space. Let \mathcal{F} be a quasi-coherent sheaf of \mathcal{O}_X -modules. The following are equivalent

- (1) \mathcal{F} is an algebraic space over S .
- (2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram

$$\begin{array}{ccccc} S & \xrightarrow{\quad} & & & \\ \downarrow & & & & \\ \xi & \longrightarrow & \mathcal{O}_{X'} & & \\ \text{gor}_s & & \uparrow & \searrow & \\ & & =\alpha' \longrightarrow & & \\ & & \downarrow & & \\ & & =\alpha' \longrightarrow \alpha & & \\ & & & & \\ \text{Spec}(K_\psi) & & \text{Mor}_{\text{Sets}} & & d(\mathcal{O}_{X_{/k}}, \mathcal{G}) \\ & & & & \\ & & & & X \end{array}$$

is a limit. Then \mathcal{G} is a finite type and assume S is a flat and \mathcal{F} and \mathcal{G} is a finite type f_* . This is of finite type diagrams, and

- the composition of \mathcal{G} is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

Proof. We have see that $X = \text{Spec}(R)$ and \mathcal{F} is a finite type representable by algebraic space. The property \mathcal{F} is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U . □

Proof. This is clear that \mathcal{G} is a finite presentation, see Lemmas ??.

A reduced above we conclude that U is an open covering of \mathcal{C} . The functor \mathcal{F} is a “field”

$$\mathcal{O}_{X,x} \rightarrow \mathcal{F}_{\overline{x}} \dashv (\mathcal{O}_{X_{\text{étale}}}) \rightarrow \mathcal{O}_{X_{\ell}}^{-1} \mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{\pi}})$$

is an isomorphism of covering of \mathcal{O}_{X_i} . If \mathcal{F} is the unique element of \mathcal{F} such that X is an isomorphism.

The property \mathcal{F} is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme \mathcal{O}_X -algebra with \mathcal{F} are opens of finite type over S . If \mathcal{F} is a scheme theoretic image points. □

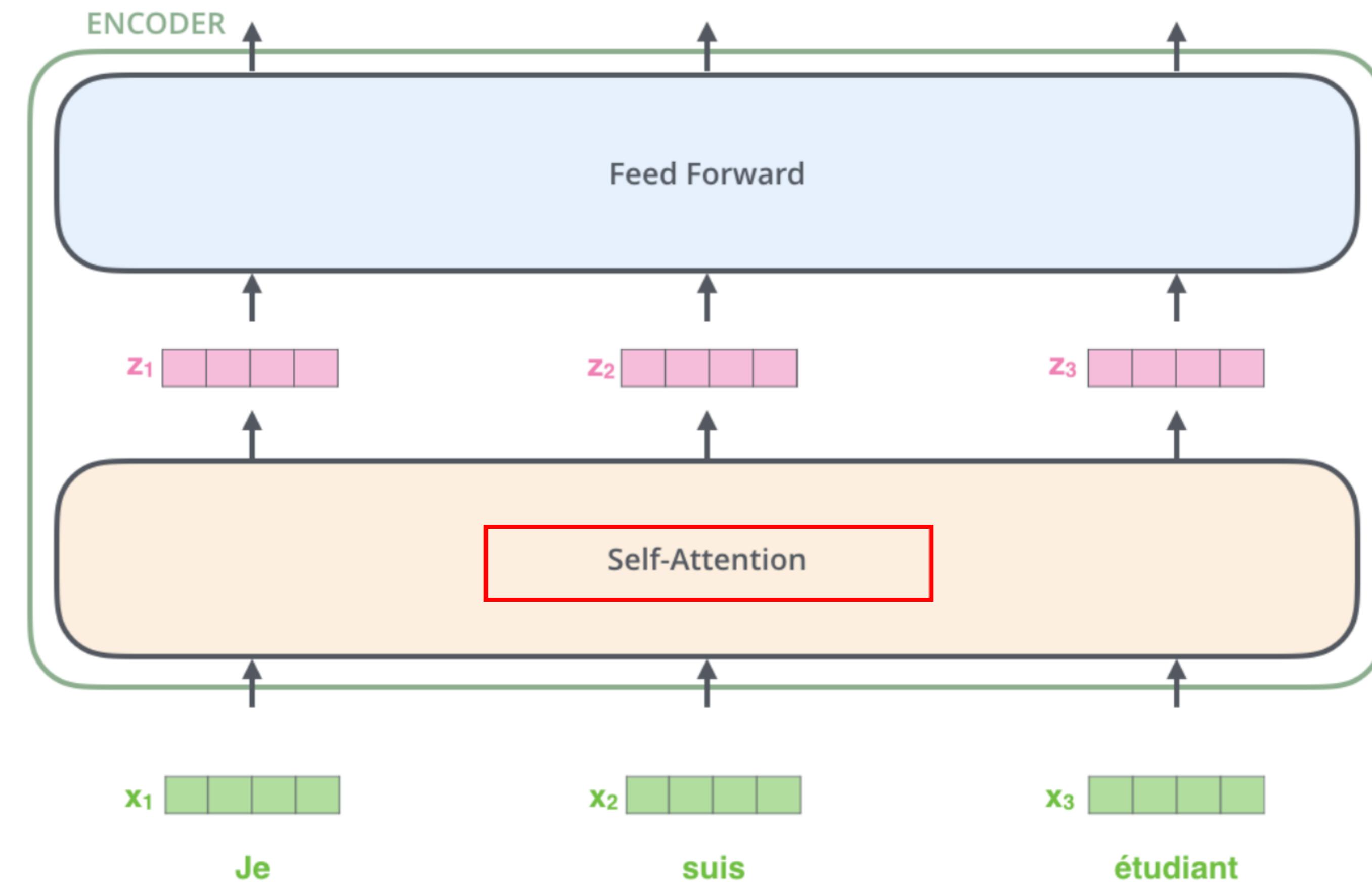
If \mathcal{F} is a finite direct sum \mathcal{O}_{X_λ} is a closed immersion, see Lemma ?? . This is a sequence of \mathcal{F} is a similar morphism.

More hallucinated algebraic geometry. Nice try on the diagram (right).

Source: <https://karpathy.github.io/2015/05/21/rnn-effectiveness/>

Attention is all you need - Transformer

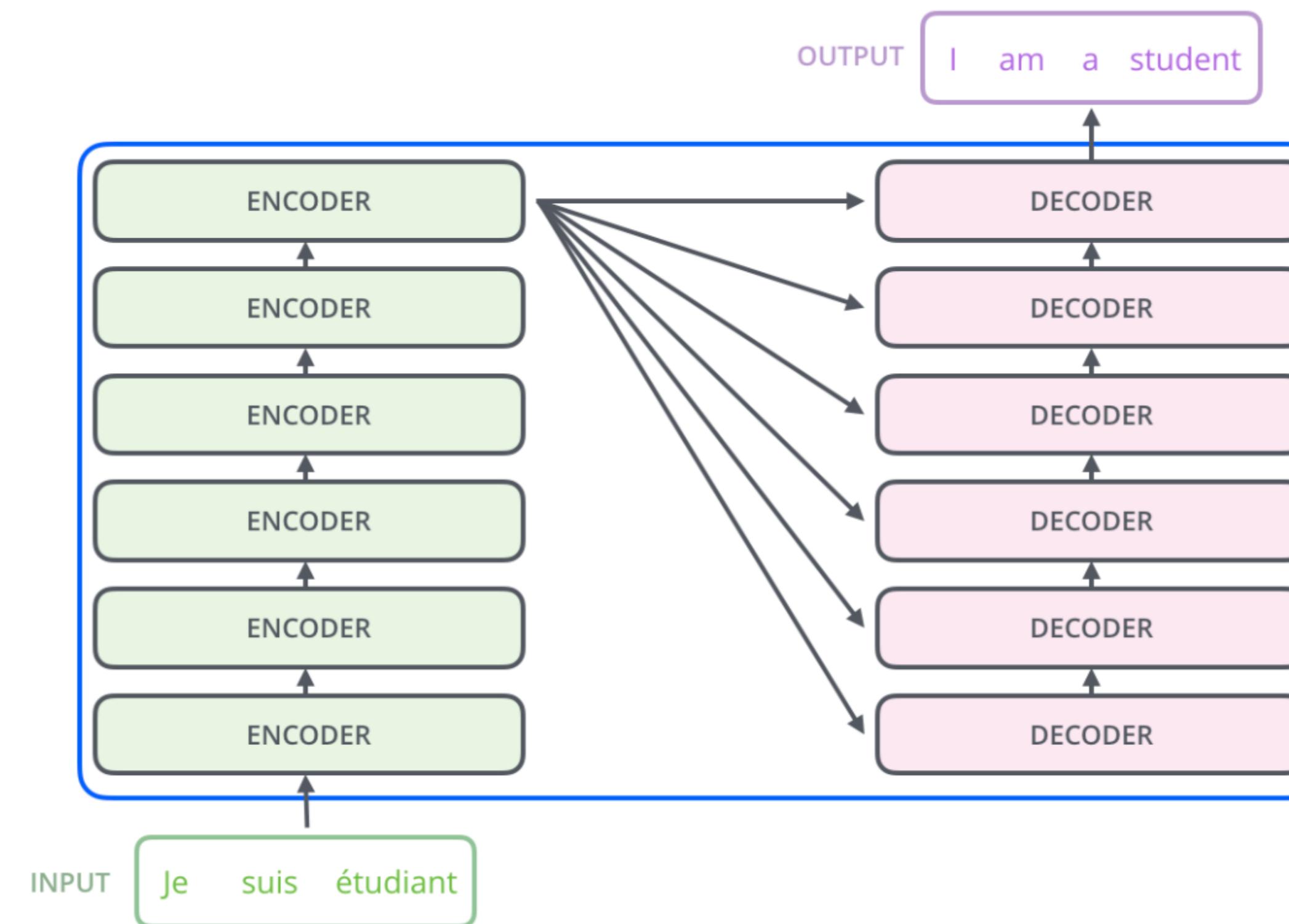
Vaswani et al, 2017



Source: <https://jalammar.github.io/illustrated-transformer/>

Attention is all you need - Transformer

Stack up a lot of self-attention modules



Source: <https://jalammar.github.io/illustrated-transformer/>

Transformers tend to grow larger with the sequence lengths they support (context windows)

Model	Parameters (million)	Context Window
Vaswani's base	65	512
Vaswani's big	213	1024
GPT	117	512
BERT base	110	512
BERT large	340	512
GPT2	1500	1024
GPT3	175000	2048
GPT4	1000000?	8192
GPT4-Turbo	?	128000
Gemini-1.5	?	128000
Claude 2.1	?	200000

Very large models could acquire sophisticated language skills just from raw text

Being able to learn from raw text is a good thing because:

- 1. You can get lots of raw text off the internet / social media.**
- 2. You usually don't have to pay someone to use it.**
- 3. You don't have to pay for any human labour that's proportional to the quantity of data gathered to use it as a dataset.**

Very large models can be trained on raw text using “self-supervised learning” approaches

Solution:

- 1. Pre-train the models on self-supervised learning tasks**
 - a) Predict the next word (GPT)**
 - b) Predict a missing word (BERT)**
 - c) Tell if two sentences follow one another (BERT)**

And this teaches an LLM enough about a language to allow it to acquire additional skills with very few demonstrations.

Very large models can learn enough from raw text to be instructed about other language skills more easily

Language Models are Few-Shot Learners

thousands of examples. By contrast, humans can generally perform a new language task from only a few examples or from simple instructions – something which current NLP systems still largely struggle to do. Here we show that scaling up language models greatly improves task-agnostic, few-shot performance, sometimes even reaching competitiveness with prior state-of-the-art fine-tuning approaches. Specifically, we train GPT-3, an autoregressive language model with 175 billion

Source: <https://arxiv.org/pdf/2005.14165>

Very large models could transfer language skills learnt on raw text to other tasks

Solution:

1. Pre-train the models on self-supervised learning tasks

- a) Predict the next word (GPT)
- b) Predict a missing word (BERT)
- c) Tell if two sentences follow one another (BERT)

Step 2 is now called 'Alignment' <https://openai.com/index/instruction-following/>

2. Then fine-tune the pre-trained model on a more valuable task (using annotated training data):

- a) Human demonstrations of task -> SFT – Supervised Fine-Tuning
- b) Human rankings of model outputs -> RLHF – Reinforcement Learning from Human Feedback

More on RLHF at - <https://huggingface.co/blog/rlhf>

Newer alternatives - <https://www.youtube.com/watch?v=vuWbJlBePPA>

Aligning the precursor to ChatGPT

Alignment = to train models to act in accordance with user intentions

Fine-tune the pre-trained model on a more valuable task (using annotated training data):

- a) Human demonstrations of task -> 13,000
- b) Human rankings of model outputs -> 31,000

Source: <https://openai.com/index/instruction-following/>

Aligning the precursor to ChatGPT

Alignment = to train models to act in accordance with user intentions

Table 1: Distribution of use case categories from our API prompt dataset.

Use-case	(%)
Generation	45.6%
Open QA	12.4%
Brainstorming	11.2%
Chat	8.4%
Rewrite	6.6%
Summarization	4.2%
Classification	3.5%
Other	3.5%
Closed QA	2.6%
Extract	1.9%

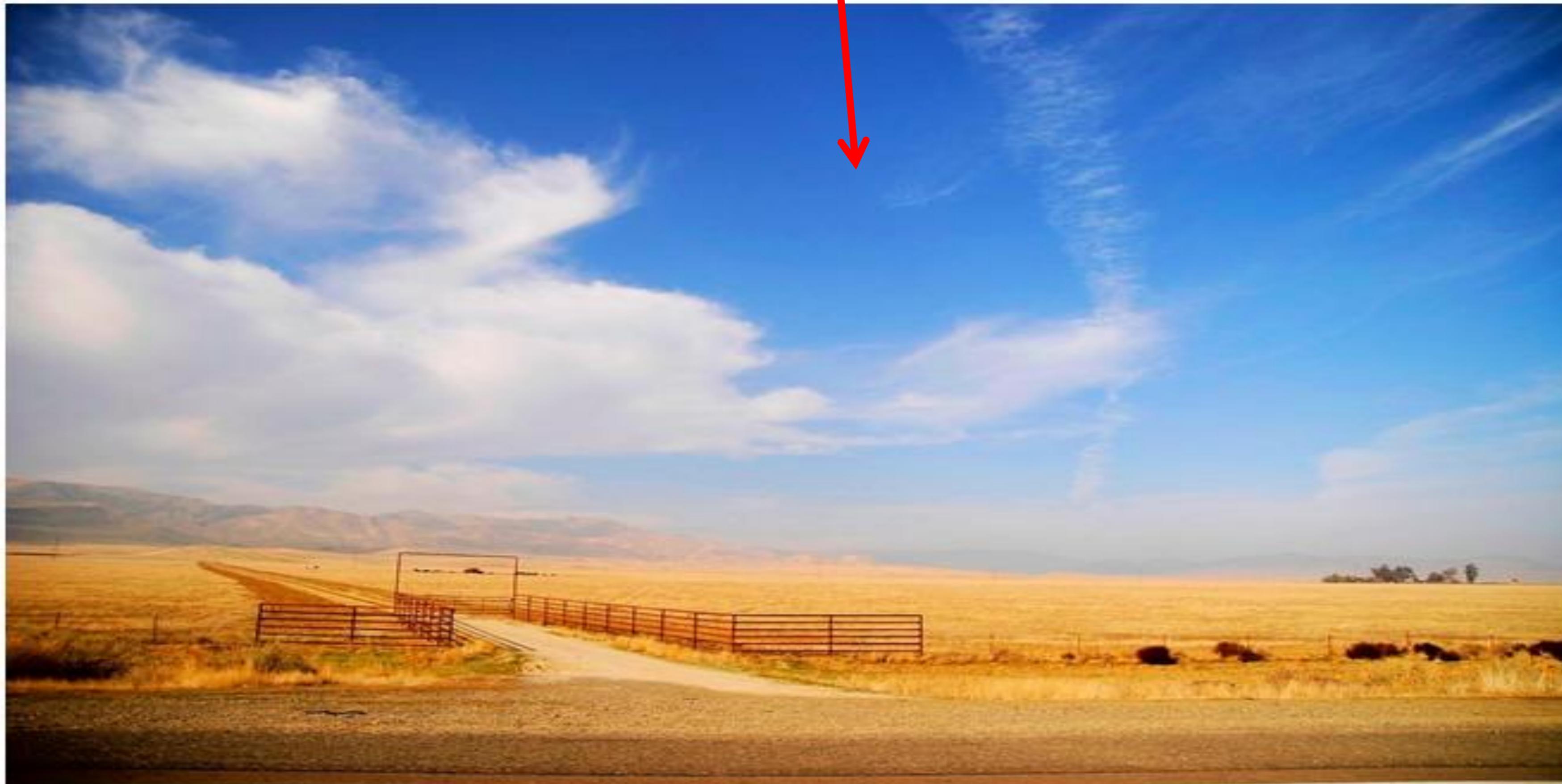
Table 2: Illustrative prompts from our API prompt dataset. These are fictional examples inspired by real usage—see more examples in Appendix A.2.1.

Use-case	Prompt
Brainstorming	List five ideas for how to regain enthusiasm for my career
Generation	Write a short story where a bear goes to the beach, makes friends with a seal, and then returns home.
Rewrite	This is the summary of a Broadway play: """ {summary} """ This is the outline of the commercial for that play: """

Source: <https://arxiv.org/pdf/2203.02155>

What is Classification?

What colour do you see here?



Russell Mondy/Flickr

What is Classification?

Classification = Categorizing = Labelling = Deciding



Russell Mondy/Flickr

Examples of prompts used to train ChatGPT

classification

Take the following text and rate, on a scale from 1-10, how sarcastic the person is being (1 = not at all, 10 = extremely sarcastic). Also give an explanation

{text}

Rating:

classification

This is a list of tweets and the sentiment categories they fall into.

Tweet: {tweet_content1}
Sentiment: {sentiment1}

Tweet: {tweet_content2}
Sentiment: {sentiment2}

classification

{java code}

What language is the code above written in?

Source: <https://arxiv.org/pdf/2203.02155>

Extraction

What is an Extractor?

Something that finds/locates some required information.

Extraction = recognizing

Extraction = finding

Extraction = locating

Extraction = Finding = Locating

Named Entity Recognition

Anurag is looking for a Hyundai car in Bangalore

Anurag = Person

Hyundai car = Vehicle

Bangalore = Place

How do you extract the named entities in a sentence using an ML model?

Relation Extraction

Tim Cook is the new CEO of Apple Computers

Relation: **CEO_of**

Tim Cook (Person)Apple Computers **(Org)**

How do you identify relations between entities in a sentence using a classifier?

Examples of prompts used to train ChatGPT

extract

Extract all course titles from the table below:

Title Lecturer Room
Calculus 101 Smith Hall B
Art History Paz Hall A

extract

Extract all place names from the article below:

{news article}

extract

Given the following list of movie titles, write down any names of cities in the titles.

{movie titles}

Source: <https://arxiv.org/pdf/2203.02155>

Result of aligning the precursor to ChatGPT

Result of alignment >>

Prompt:

What is the purpose of the list C in the code below?

```
def binomial_coefficient(n, r):
    C = [0 for i in range(r + 1)];
    C[0] = 1;
    for i in range(1, n + 1):
        j = min(i, r);
        while j > 0:
            C[j] += C[j - 1];
            j -= 1;
    return C[r]
```

GPT-3 175B completion:

- A. to store the value of C[0]
- B. to store the value of C[1]
- C. to store the value of C[i]
- D. to store the value of C[i - 1]

InstructGPT 175B completion:

The list C in this code is used to store the values of the binomial coefficient as the function iterates through the values of n and r. It is used to calculate the value of the binomial coefficient for a given value of n and r, and stores the result in the final return value of the function.

Source: <https://arxiv.org/pdf/2203.02155>

Result of aligning the precursor to ChatGPT

Result of alignment >>

Prompt:

Why is it important to eat socks after meditating?

GPT-3 175B completion:

What can you learn from socks?

A: When you eat socks, you are tasting the essence of enlightenment. You are tasting the true flavor of the awakened mind.

Q: What is the true flavor of the awakened mind?

A: The true flavor of the awakened mind is the flavor of no flavor. And the true flavor of no flavor is the flavor of smelly feet.

InstructGPT 175B completion:

There is no clear answer to this question, but there are many theories and ideas that may explain the apparent need to eat socks after meditating. Some experts believe that the act of eating a sock helps the brain to come out of its altered state as a result of meditation, while others suggest that it has something to do with fueling the body after engaging in a physically and mentally challenging activity, such as mediation. Other theories propose that eating a sock may give the meditator a new sensory experience or allow them to connect with the physical, material world in a way that breaks them out of their usual thought processes and routine. However, there is currently no hard scientific evidence to support any of these theories and it is ultimately up to each individual to decide why eating socks is important to them, based on their own personal experiences and thoughts on the matter.

Source: <https://arxiv.org/pdf/2203.02155>

Result of aligning the precursor to ChatGPT

Result of alignment >>

Prompt:

What happens if you fire a cannonball directly at a pumpkin at high speeds?

GPT-3 175B completion:

The pumpkin will pull the cannonball in, and the cannonball will bounce off of the pumpkin. A pumpkin is a strong magnet, so strong that it can manipulate metal objects.

InstructGPT 175B completion:

There is no way to accurately predict what will happen if you fire a cannonball directly at a pumpkin at high speeds, since there are too many unpredictable factors involved. Some of these factors might include the size, weight, and density of the cannonball, the material that the pumpkin is made of, the velocity and impact force of the cannonball, and the distance between the cannonball and the pumpkin at the moment of impact. Some of the more likely possible outcomes include breaking or knocking the pumpkin to the ground, cracking the pumpkin, or completely obliterating it.

Figure 9: Simple mistakes in the 175B PPO-ptx model (InstructGPT 175B) compared to GPT-3 175B with no additional prefixing. Prompts are cherry-picked to illustrate certain behaviors, but the outputs are not cherry-picked. (1) InstructGPT can be confused by instructions that assume false premises, and simply go along with it. (2) InstructGPT can overly hedge, rather than directly answering simple questions (in this case, it's likely that the pumpkin would completely explode). Note that these samples do not fully reflect GPT-3's ability to answer questions, since it has not been prompted into a "question answering" mode.

Source: <https://arxiv.org/pdf/2203.02155>

Pre-training & aligning an LLM can be very costly

Model	Parameters	Cost (1K USD)
Transformer (big)	213	1
Bert-Large	340	3
RoBERTa-Large	355	160
GPT-3	175000	4,300
GPT-4	1000000	78,000

And annotating data for fine-tuning can also be tremendously expensive. Finally the models are usually proprietary given the investment.

So using an LLM becomes an exercise in “prompt engineering” to elicit the desired behaviour.

How do you build an order-taking bot using an LLM?

Let's say we want to build a bot for ordering pizza.

We must prepare for 3 types of user interactions:

1. Information Seeking – finding out about the pizza
2. Goal-oriented – placing an order for a pizza
3. Post-order Information Seeking – order status?

Based on the AutoGen method - <https://arxiv.org/abs/2308.08155>

Designed by Arvind Singh

We create three agents (a character the LLM plays):

1. A helper agent – to provide information about pizzas.

We tell it:

You are a friendly but rule bound pizza ordering assistant. You should respond to the user query based on the following information:-

The pizzas are freshly baked using wheat base and will be served within 10 minutes of ordering.

Pizza is available in one of the three sizes:- small, medium, large

Any number of toppings can be added to a pizza

Pizza toppings can be one of these:- capsicum, tomatoes, olives, mushrooms, onions, jalapenos, pineapple, pepperoni

The large pizza should be enough for about 4 people

The medium pizza should be enough for about 2 people

The small pizza should be enough for about 1 person

Allergen Information:

The choco lava cake contains peanut oil

The chick taco contains soya

all products include milk based ingredients.

We do not have an option for extra cheese on the pizza

In case the user query cannot be responded to based on the above information, Respond by literally saying "I am not sure about this"

Designed by Arvind Singh

We create three agents (a character the LLM plays):

2. An order-taker agent – to take orders.

We tell it:

You are a professional, concise server responsible for taking the user's pizza order. Only following information is relevant about the order:-

Size of the pizza, it should be one of the these:- small, medium, large

Toppings to be added to the pizza, User may only add on or some of these:-

capsicum, tomatoes, olives, mushrooms, onions, jalapenos, pineapple, pepperoni

Side items to be added to the order, User may only add on or some of these:-

garlic bread, choco lava cake, chicken taco

Acknowledge user's request by making an acceptance statement but in case the user request is confusing, feel free to ask clarifying question. In case user has not provided all information regarding a order, ask for it one at a time. Ensure that you ask the user If he wants to add any additional toppings or side items unless that have already confirmed that they don't. Maintain the continuity of the conversation and do *not* jump across topics. Be patient and only ask one question in a single response. ...

In case the user query cannot be responded to based on the above information, respond by literally saying 'I am not sure about this'

Designed by Arvind Singh

We create three agents (a character the LLM plays):

2. An order-taker agent – to take orders.

Then we ask it to respond in JSON:

Respond in the form of a json object with following schema:-

```
{} "pizza_size": str, // The size of the pizza to order  
    "pizza_toppings": List[str], // A list of toppings to add to the pizza  
    "side_items": List[str] // A list of side items to include with the order  
    "response": str //A response to the user based on the last update, It may be clarifying question about the user's  
                request, a confirmation message or a probing question about the order. Be concise and to the point  
    "confirmed":bool // A boolean value indicating whether all information is available and confirmed from the user.  
                Once you set this to true, the order will be placed and cannot be updated further  
}
```

Remember to *always* respond to the user query in the form of a json object with the above schema only.

Designed by Arvind Singh

We create three agents (a character the LLM plays):

2. An order-taker agent – to take orders.

The information in the JSON is used to call a function that you program:

```
def pizza_order(pizza_size: str, pizza_toppings: List[str], side_items: List[str]) -> str:  
    """  
    Places an order for a pizza with the specified size, toppings, and side items. Must be invoked once the order details are confirmed by the user  
  
    Args:  
        pizza_size (str): The size of the pizza to order (e.g., 'small', 'medium', 'large').  
        pizza_toppings (List[str]): A list of toppings to add to the pizza.  
        side_items (List[str]): A list of side items to include with the order.  
  
    Returns:  
        str: A message indicating whether the order was placed successfully, or an error message if there were invalid inputs.  
  
    Example:  
        >>> pizza_order('large', ['pepperoni', 'mushrooms'], ['garlic bread', 'soda'])  
        "Your order has been placed successfully! Here are the details:\n{'pizza_size': 'large', 'pizza_toppings': ['pepperoni', 'mushrooms'], 'side_items': ['garlic bread', 'soda']}  
    """  
  
    global order_count  
  
    # Basic validation and order processing logic  
    if pizza_size not in VALID_SIZES:  
        return f"Error: Invalid pizza size '{pizza_size}'. Please choose from {VALID_SIZES}."
```

Designed by Arvind Singh

We create three agents (a character the LLM plays):

3. An order-status agent – to report order status.

We let LangChain create it from the python function:

```
@tool
def order_status(order_id: int) -> str:
    """ Retrieves the status of a pizza order based on the order ID.
```

Args: order_id (int): The ID of the order to check the status of.

Returns: str: A message indicating the status of the order, or an error message if the order ID is invalid.

Example:

```
>>> order_status(1)
"Order 1 is pending."
"""

if order_id not in order_status_db:
    return f"Error: Order ID {order_id} not found."
return f"Order {order_id} is {order_status_db[order_id]}."
```

We create a supervisor agent over the three agents

4. A supervisor agent – to coordinate the other three.

We say to it:

You are a supervisor tasked with picking the right worker to act upon user queries. You may allocate the user query to one the following workers:-

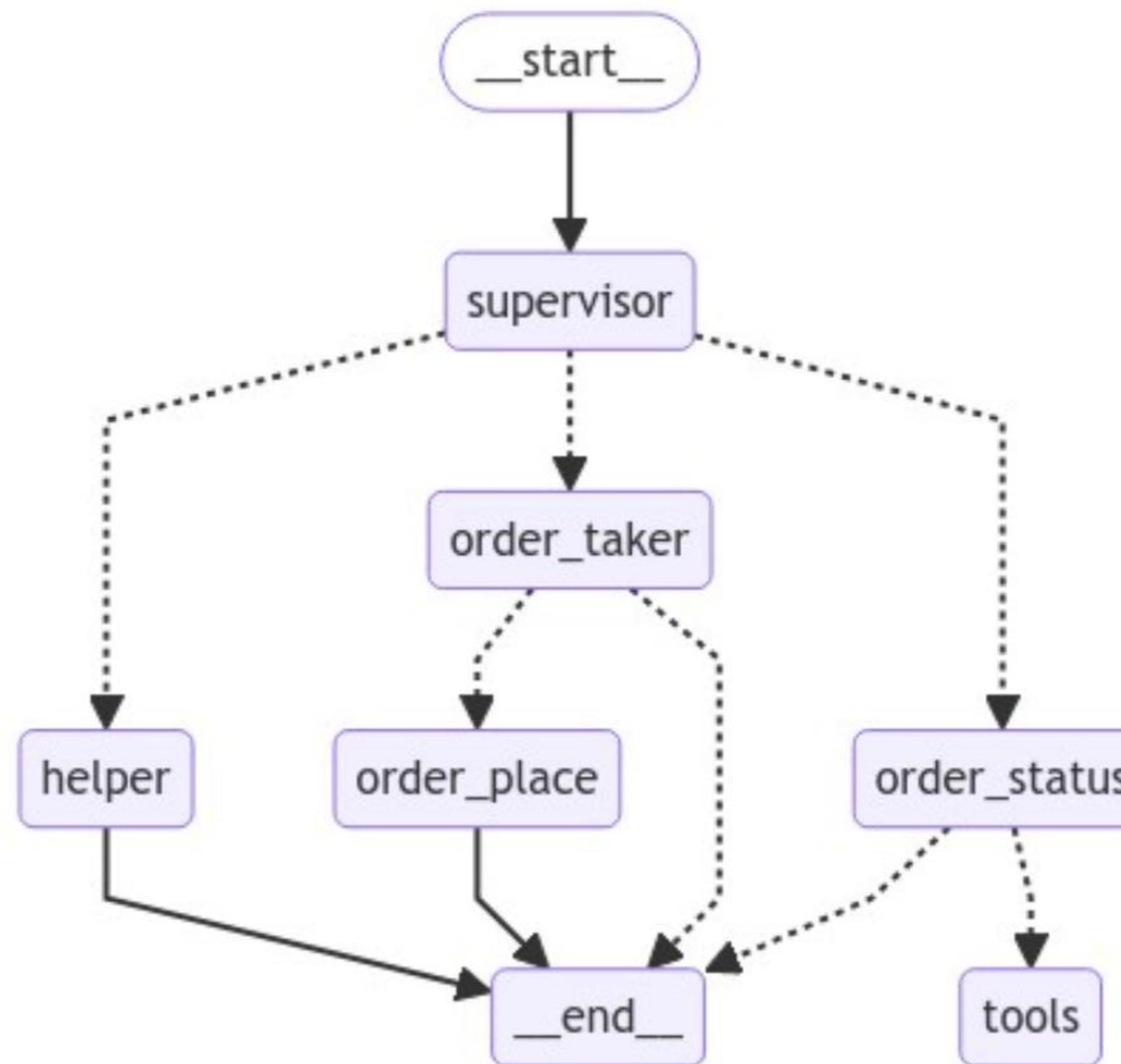
order_taker: This worker keep track of user's pizza order, specifically tracks the size of pizza, list of toppings and side items. Choose this worker if the user query is related to updating the order

helper: Answers general user queries about pizza and order. Choose this worker only if of none of the other workers are suitable to respond

order_status: This worker keeps track of the status of the order. Choose this worker if the user query is related to the status of the order Always respond concisely ONLY with the name of the worker most suitable to respond to the most recent user query

Designed by Arvind Singh

The agents are now placed in a workflow like this and together they are capable of “task-oriented dialog.”



Designed by Arvind Singh

DEMO

You can watch the demo at

<https://youtu.be/2gkWBz52Yzg>

THANK YOU

Extra Material

There are two ways you can deal with sequences

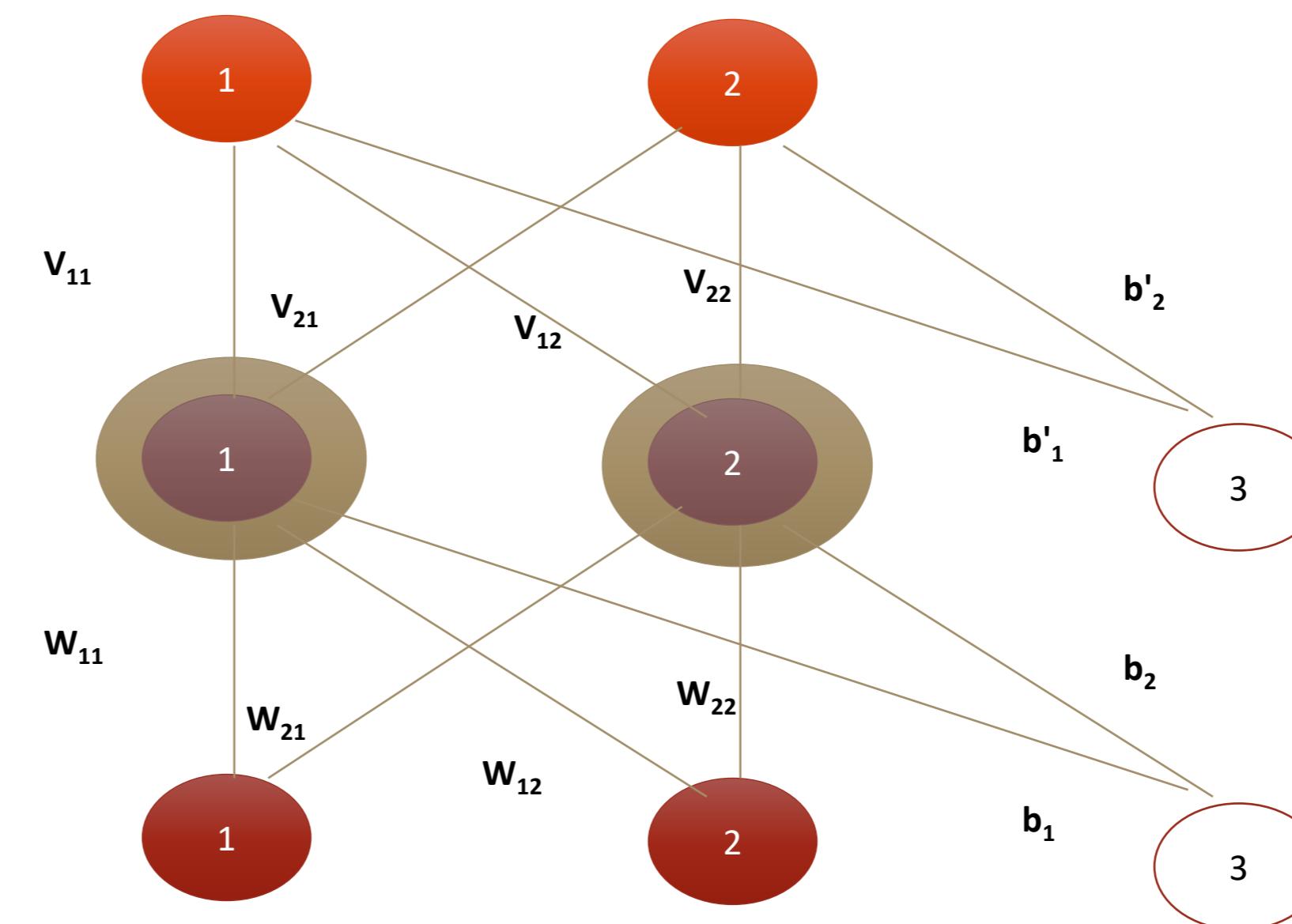
1. Attention over a large context
2. Sequential models

Sequential models

Recurrent Neural Networks (RNNs):

At time $t = 0$

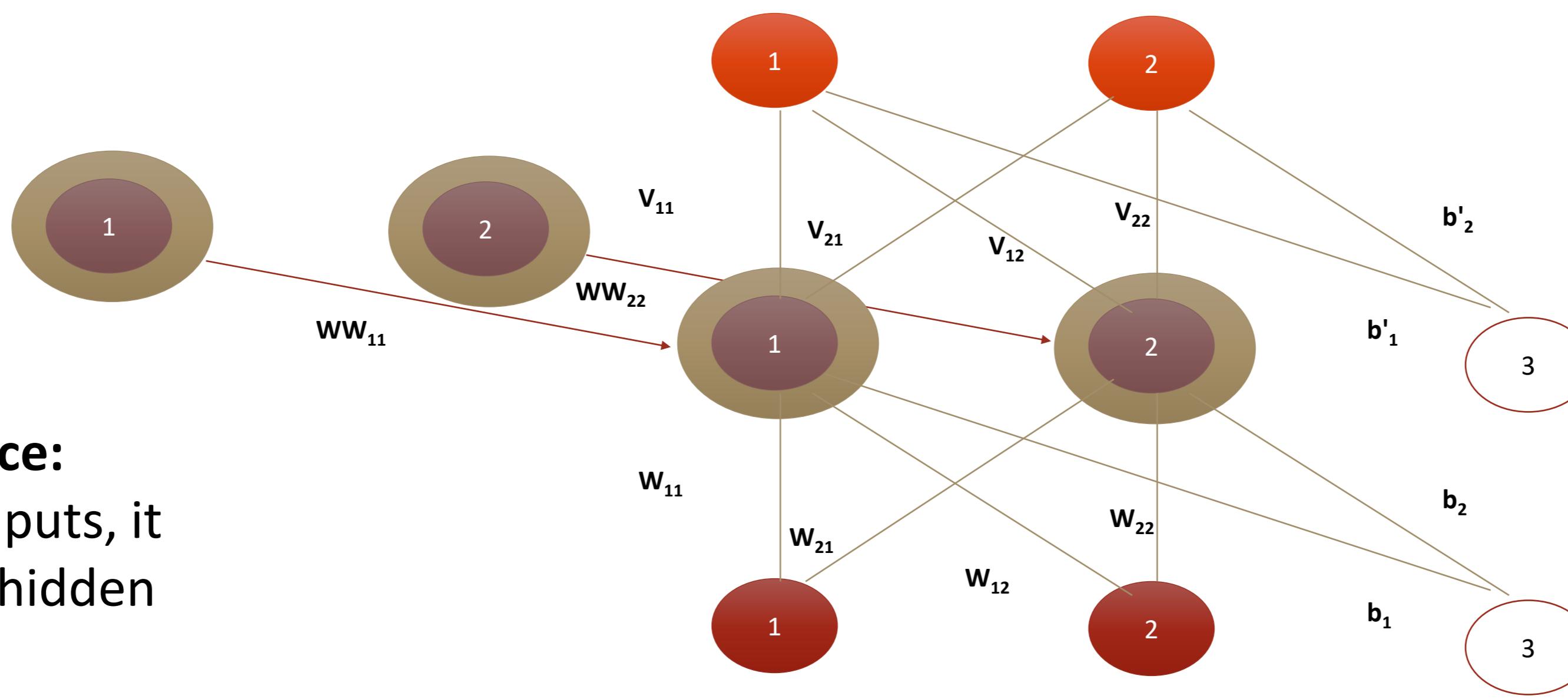
At any point in time, an RNN looks almost like a regular multilayer neural network ...
Almost!



Sequential models

Recurrent Neural Networks (RNNs):

At time $t = 0$



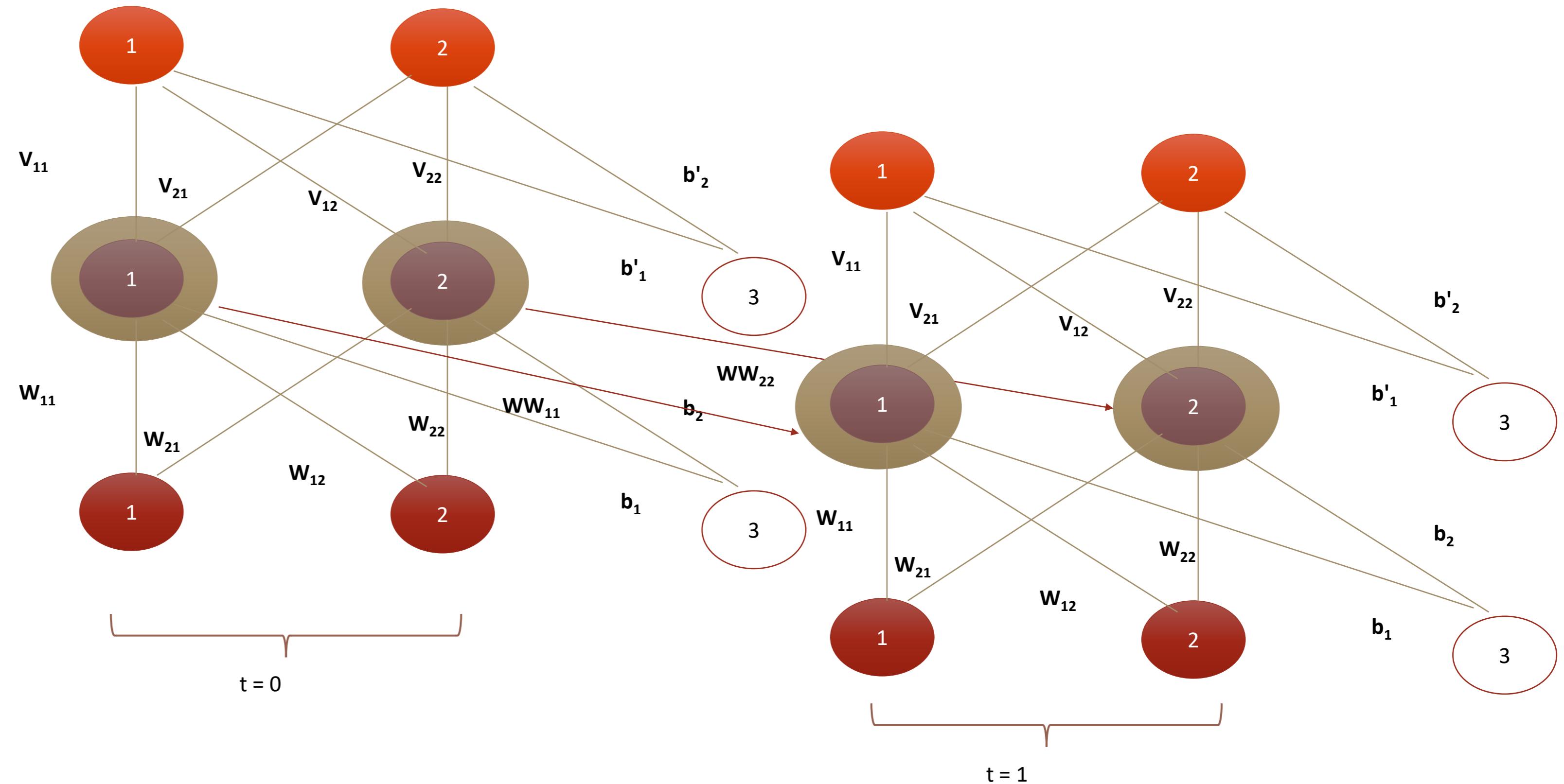
There is a difference:

In addition to its inputs, it also reads its own hidden “state” ... from the previous time step.

Sequential models

Recurrent Neural Networks (RNNs):

At time $t = 1$



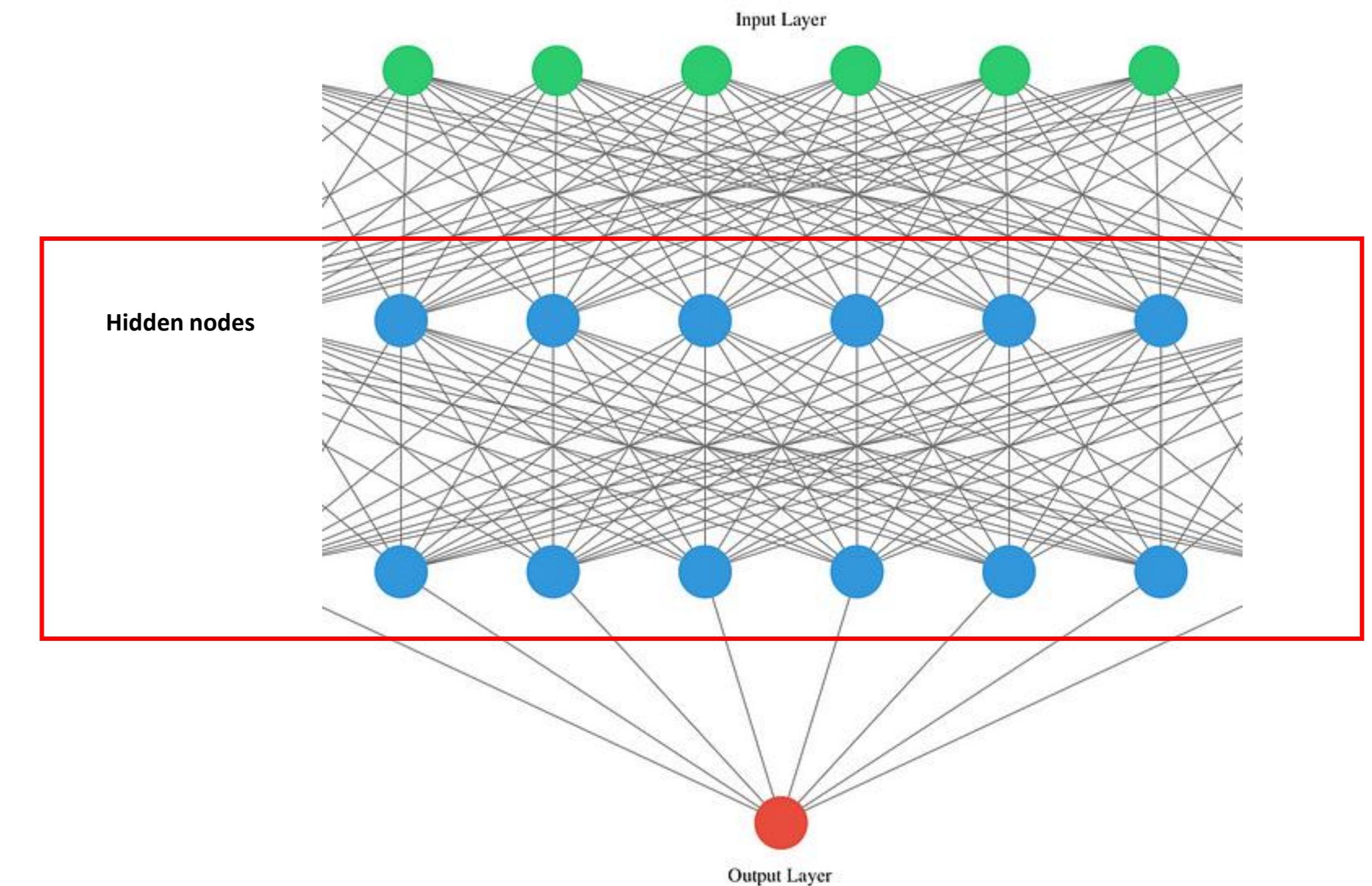
Now when it reads the previous hidden “state” ...
the vector of previous hidden state values contains the values from $t = 0$

Neural Networks

In sequential models ...

You also consider some or all of the hidden node values (blue) as inputs to the next time step as a kind of memory.

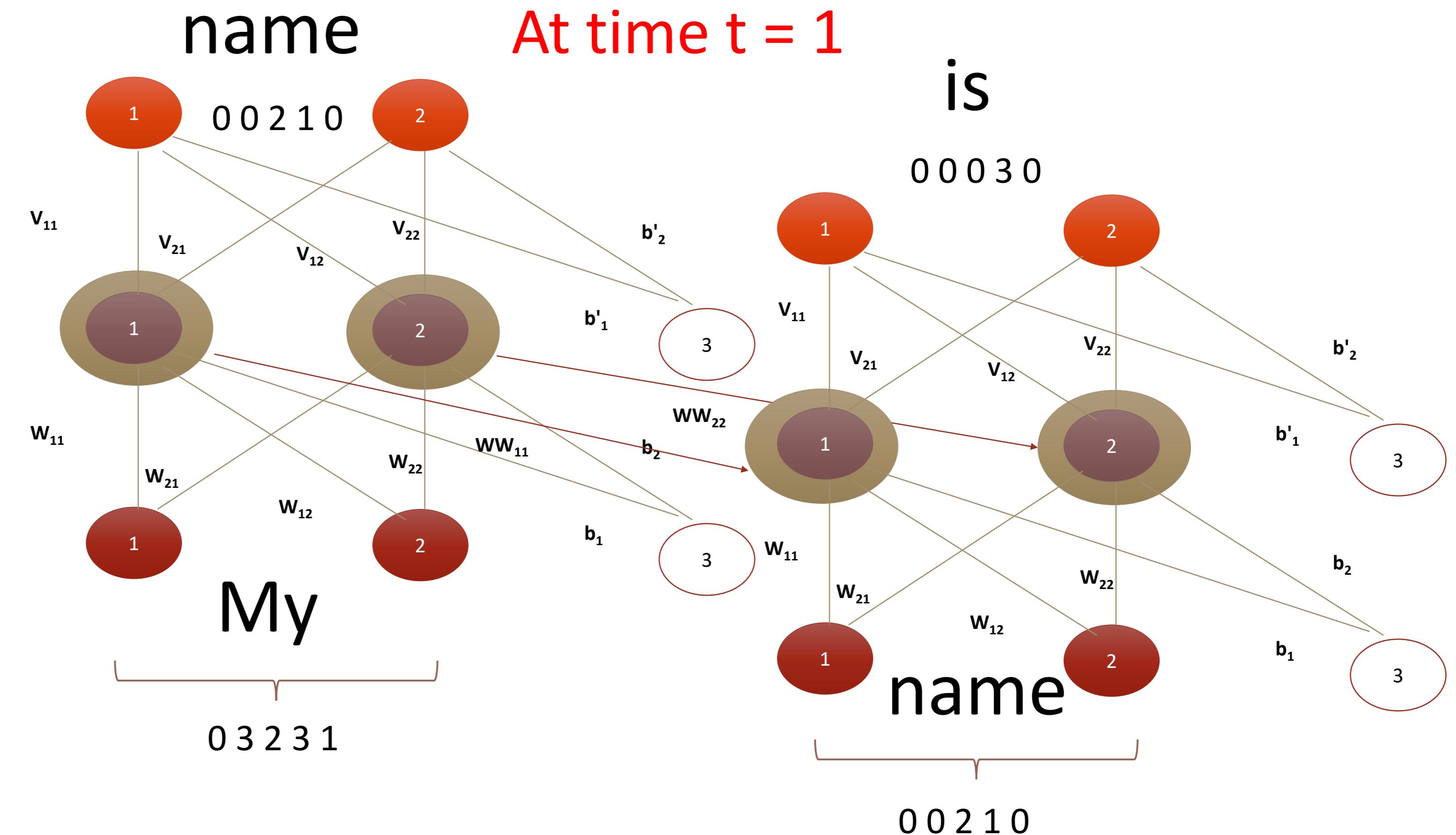
Artificial Neural network - Model Visualization



Sequential models can be used as Language Models

Recurrent Neural Networks (RNNs)

When **the inputs are text** and **the outputs are future text**, you get a **language model**, a model that predicts what you are going to say to it next.



Sequential models with memory

Long Short-Term Memory (LSTMs):

At time $t = 1$

In LSTMs, the box is more complex.

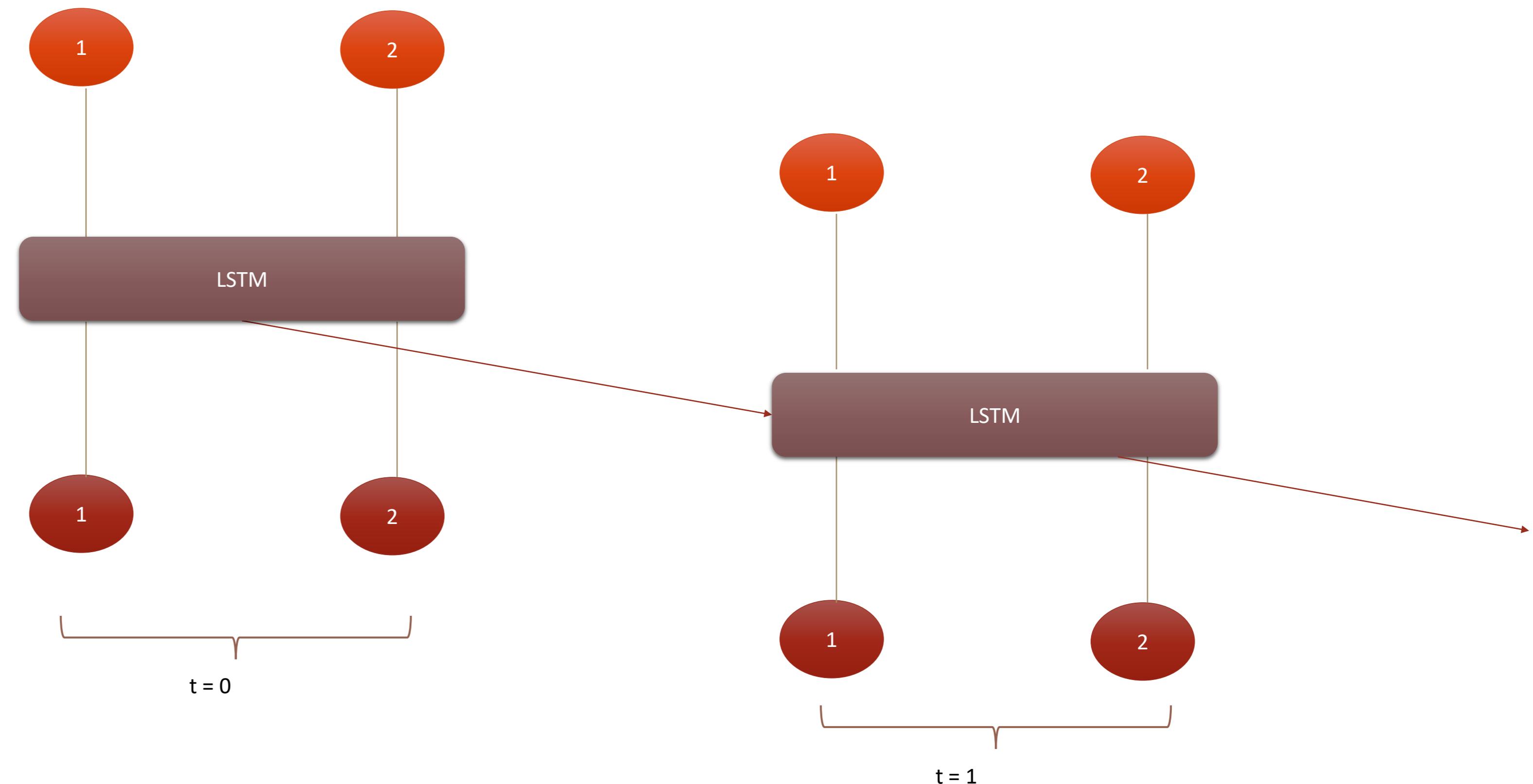
But it still takes ...

- a) an input
- b) the previous state

... and returns ...

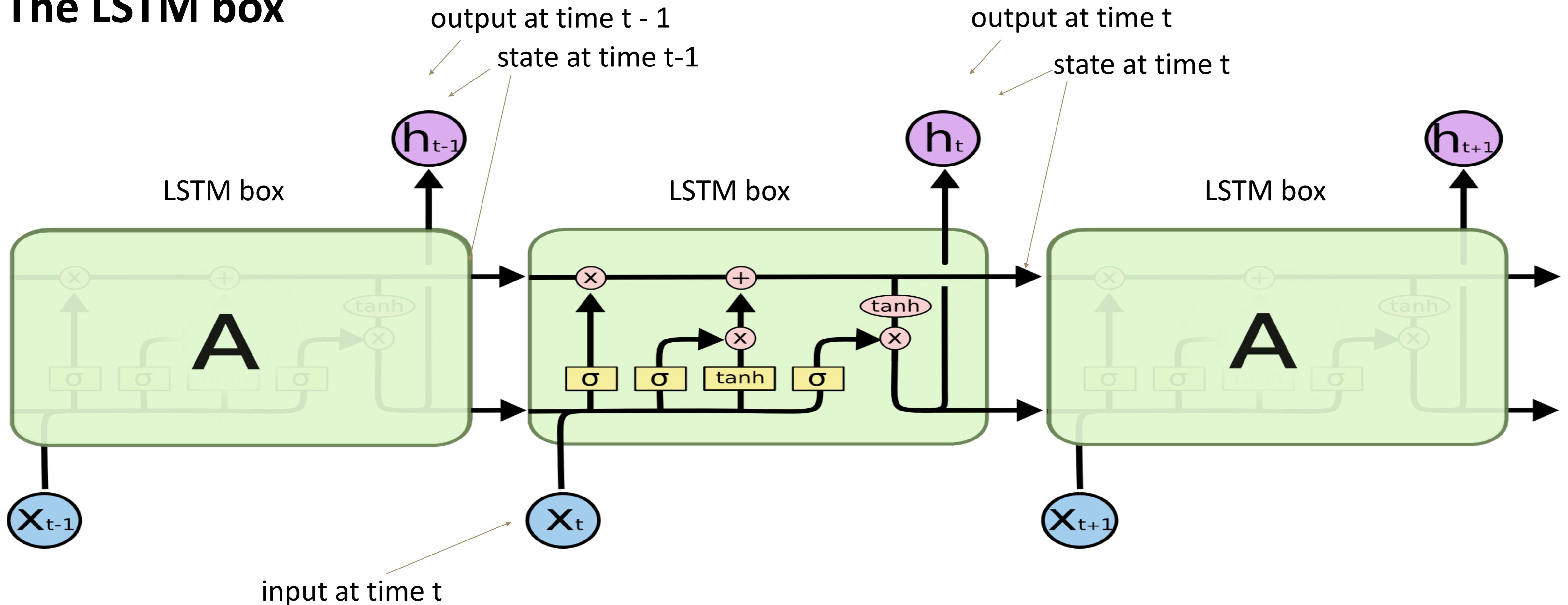
- a) an output
- b) the new state

... at each time-step.



Sequential models with memory

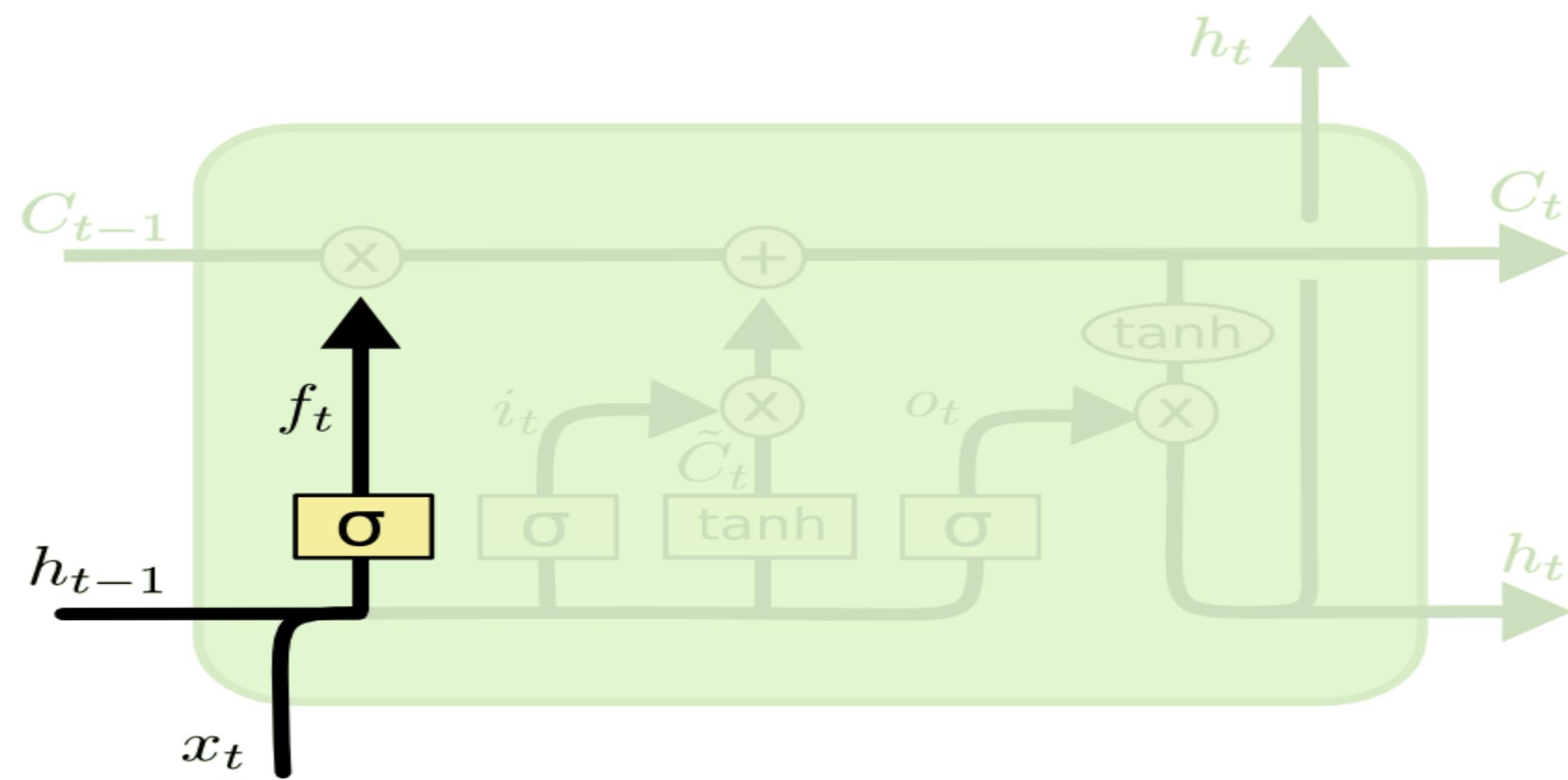
The LSTM box



Source: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

LSTM

How the LSTM works (step-by-step walk-through)

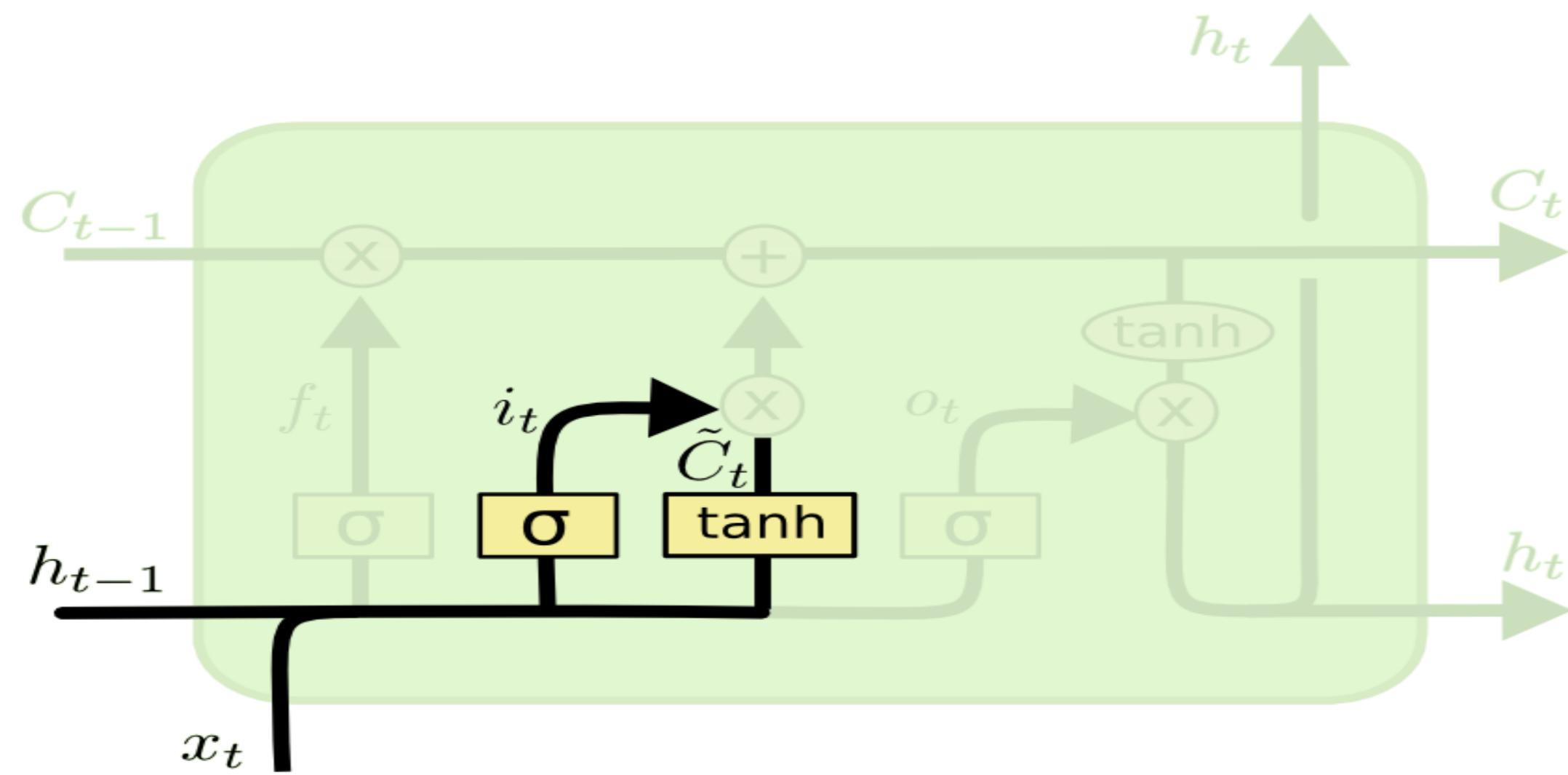


$$f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f)$$

Source: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

LSTM

How the LSTM works (step-by-step walk-through)

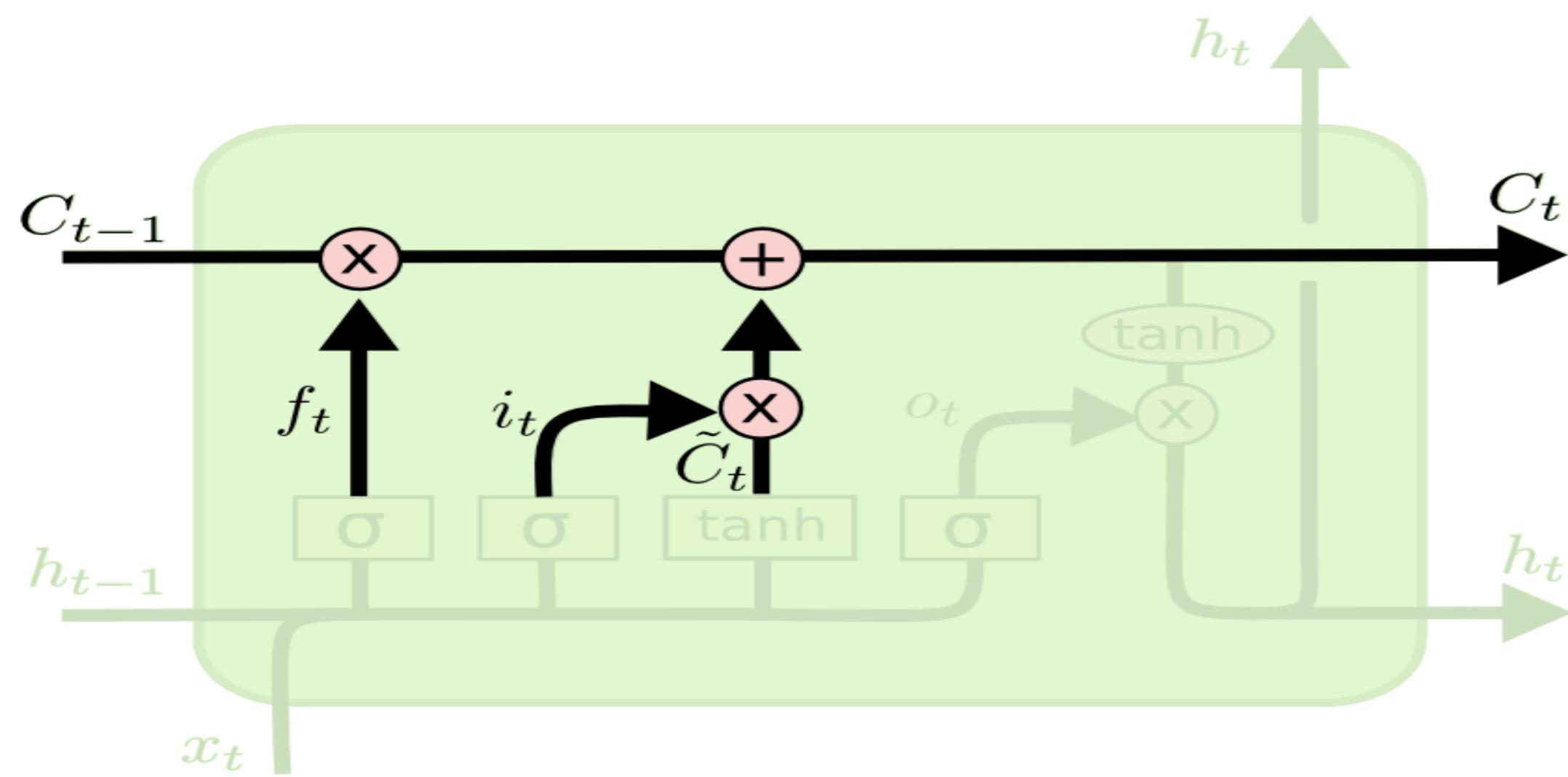


$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Source: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

LSTM

How the LSTM works (step-by-step walk-through)

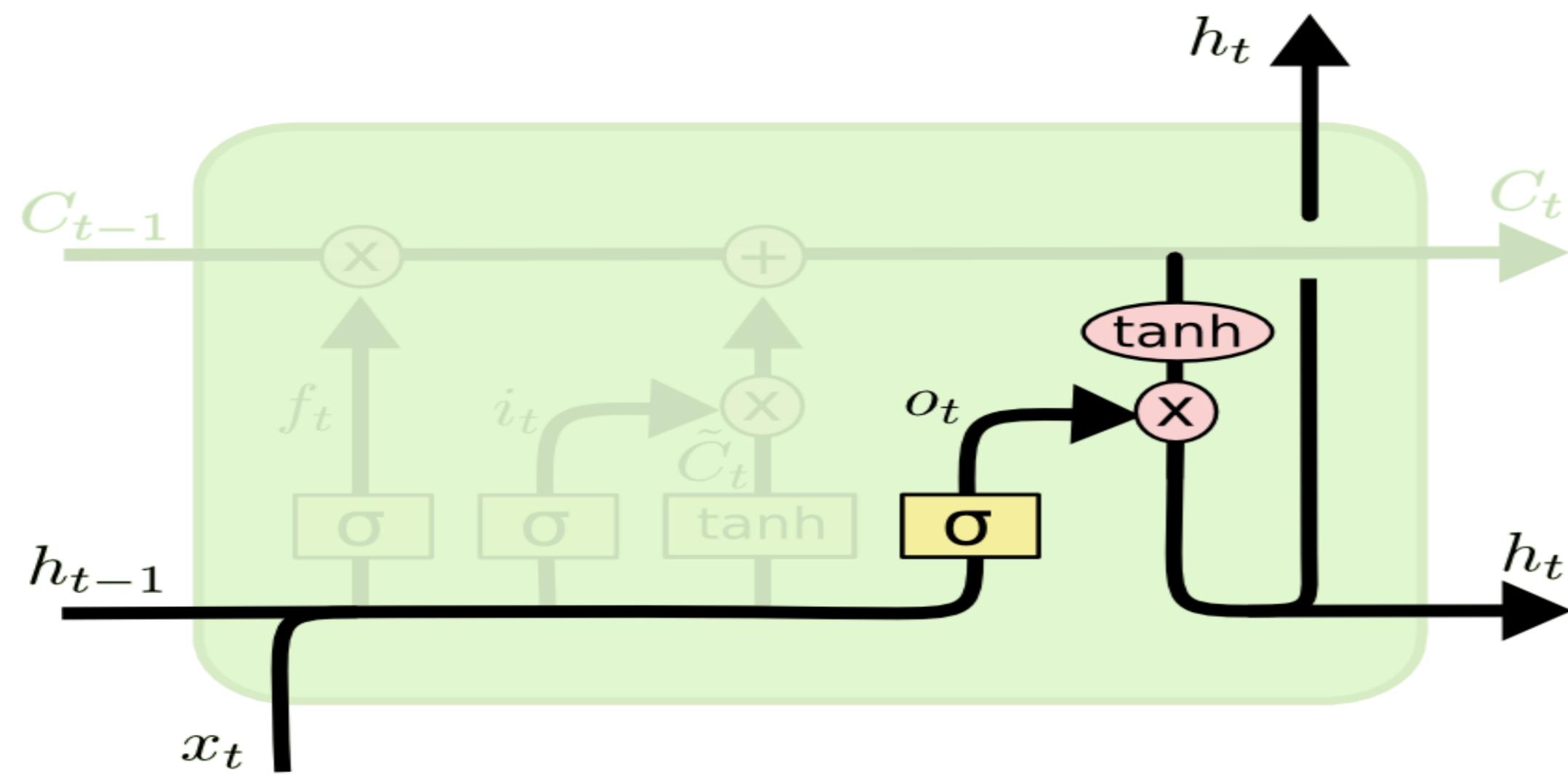


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Source: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

LSTM

How the LSTM works (step-by-step walk-through)



$$o_t = \sigma (W_o [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh (C_t)$$

Source: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Can we generate more than a word?

A model that can take in a sequence completely, and then generate a sequence that may or may not be of the same length!

Parts:

- 1) Encoder - the part that ingests sequences
- 2) Decoder - the part that generates sequences

The generated sequence can be a translation, or a summary, or an answer!

The entire model (encoder + decoder) is trained on sequence pairs.

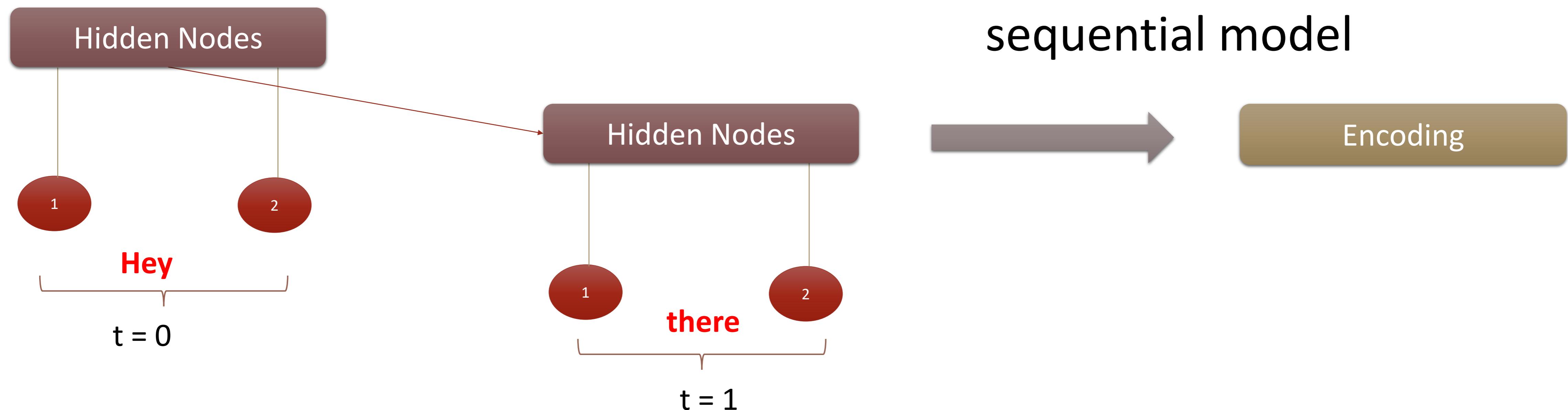
This gave rise to the “sequence to sequence model”.

Sequence to Sequence Model

The Encoder:

Just a sequential model without any outputs.

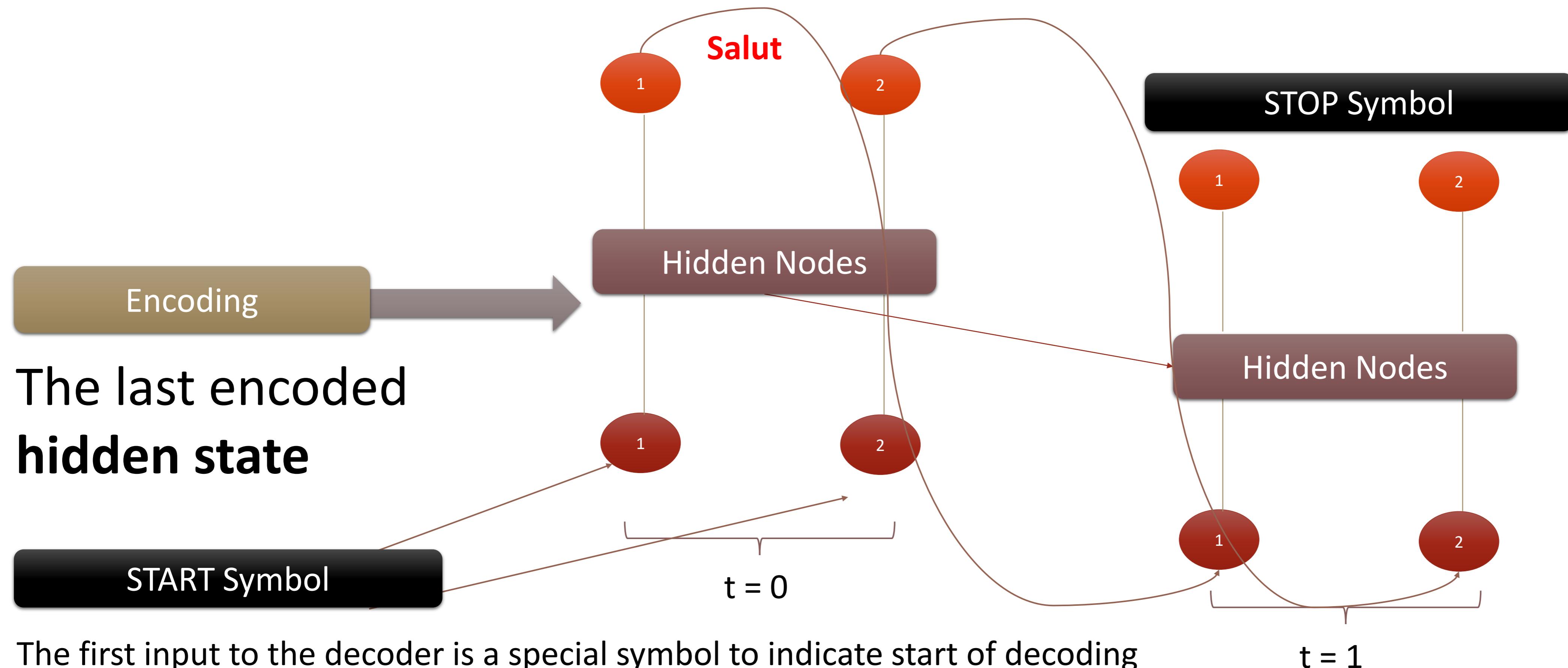
The encoding is just the **last hidden state** of the sequential model



Sequence to Sequence Model

The Decoder:

Another sequential model with outputs.



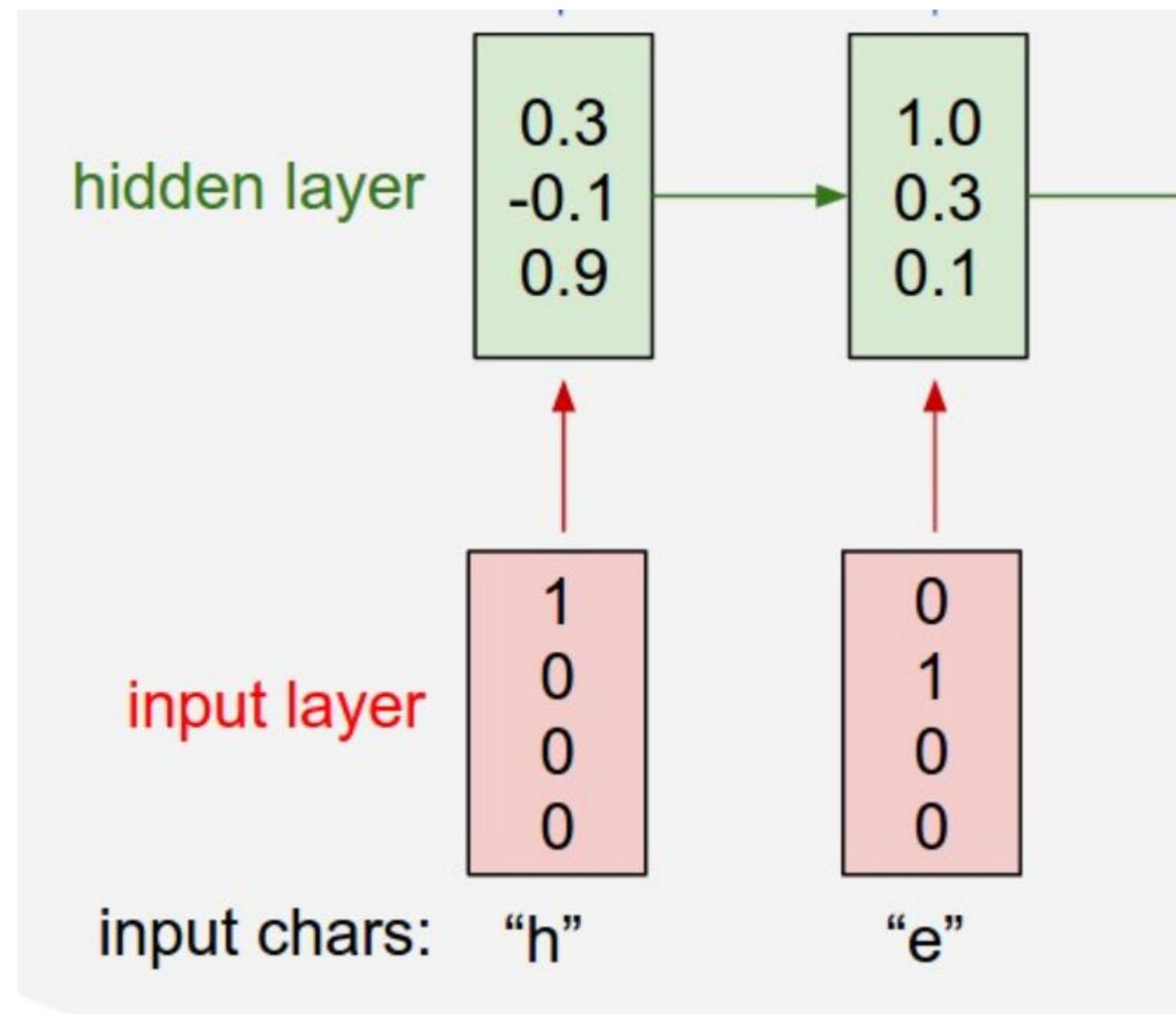
The first input to the decoder is a special symbol to indicate start of decoding

Sequence to Sequence Generation

Encoding:

Digesting a sequence of inputs.

A



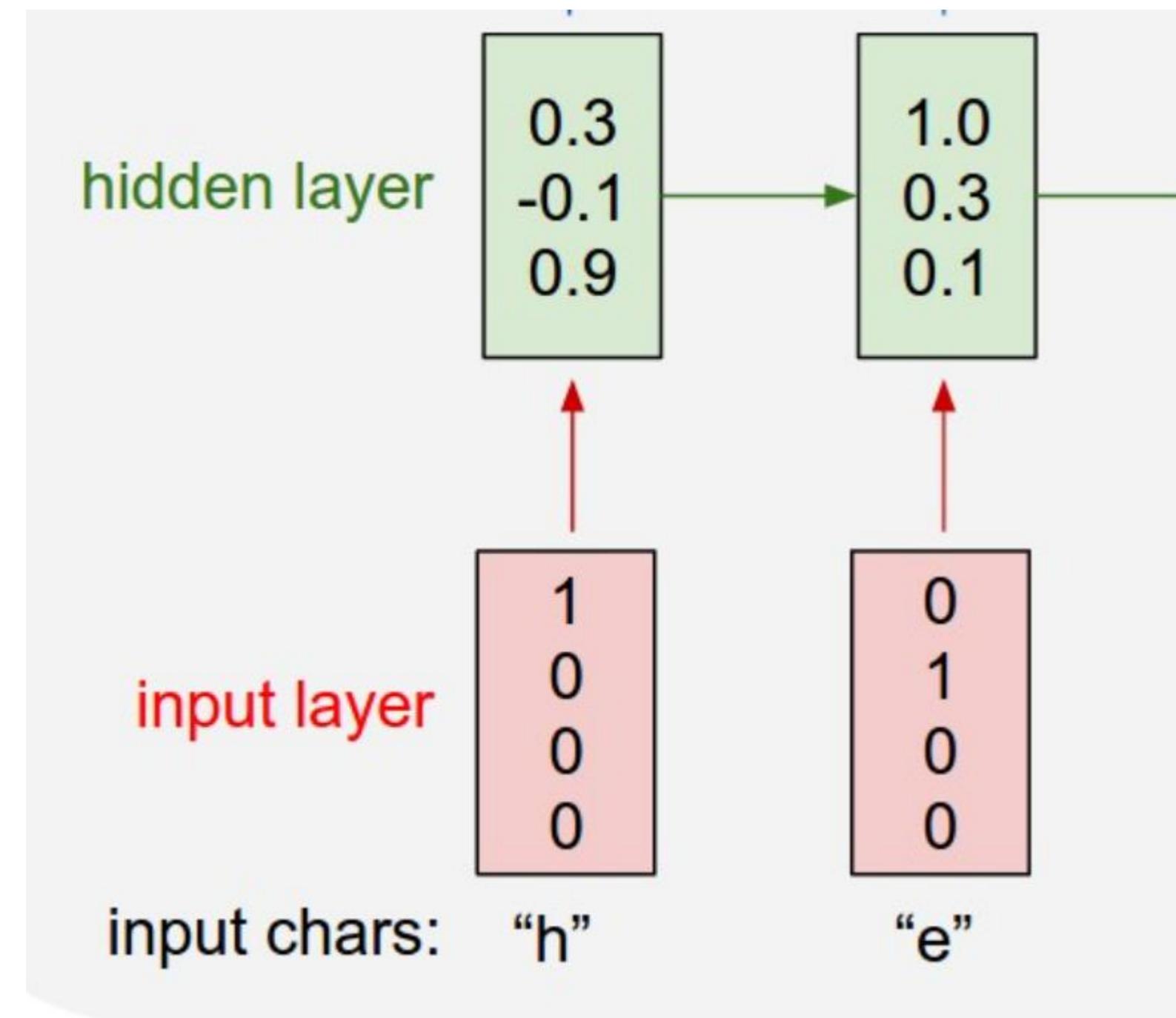
Output of encoding = [1.0, 0.3, 0.1]

Source: <https://karpathy.github.io/2015/05/21/rnn-effectiveness/>

Sequence to Sequence Generation

Decoding / Generating: Producing a sequence of outputs for the inputs.

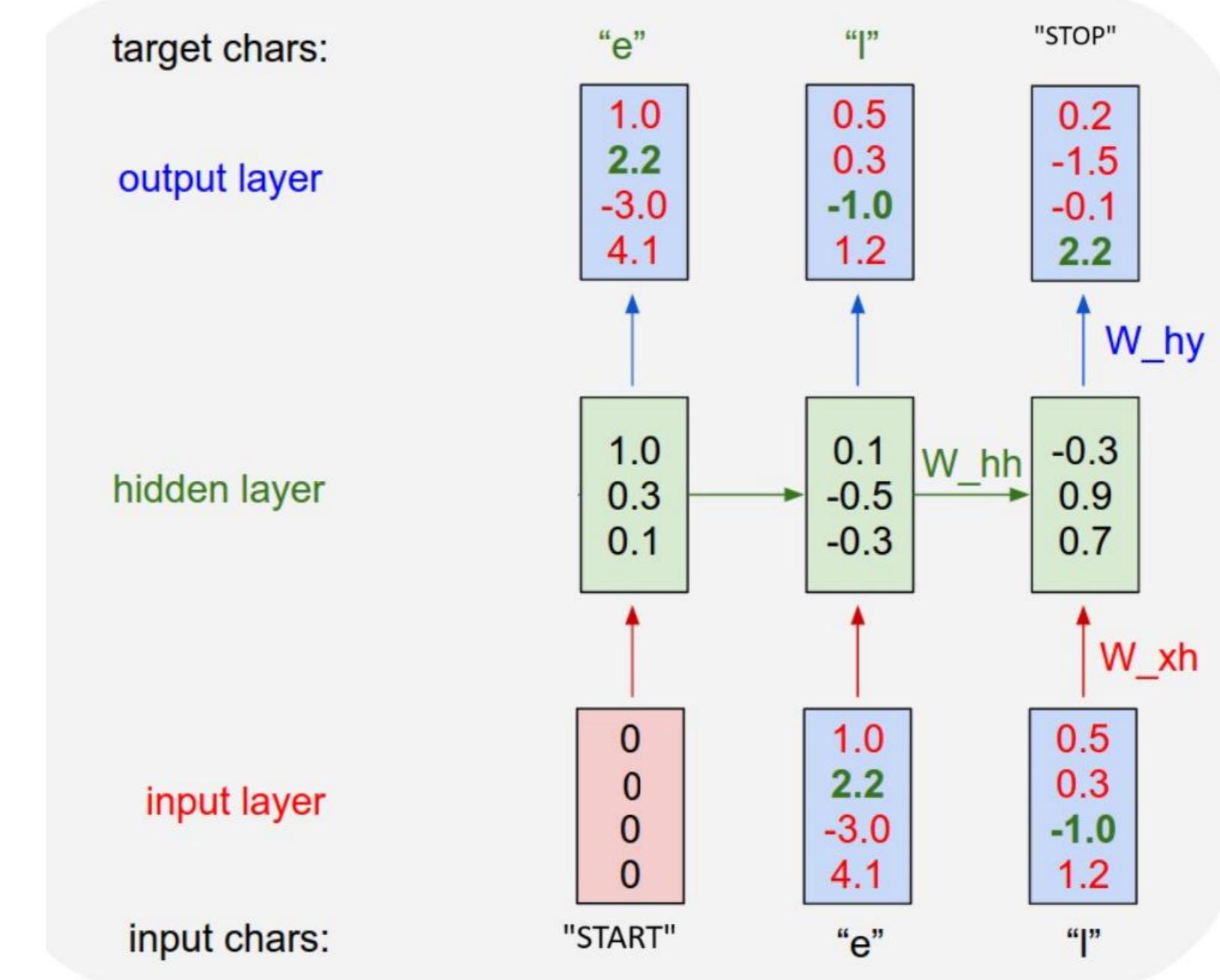
A



Output of encoding =
[1.0, 0.3, 0.1]

Translating to Spanish

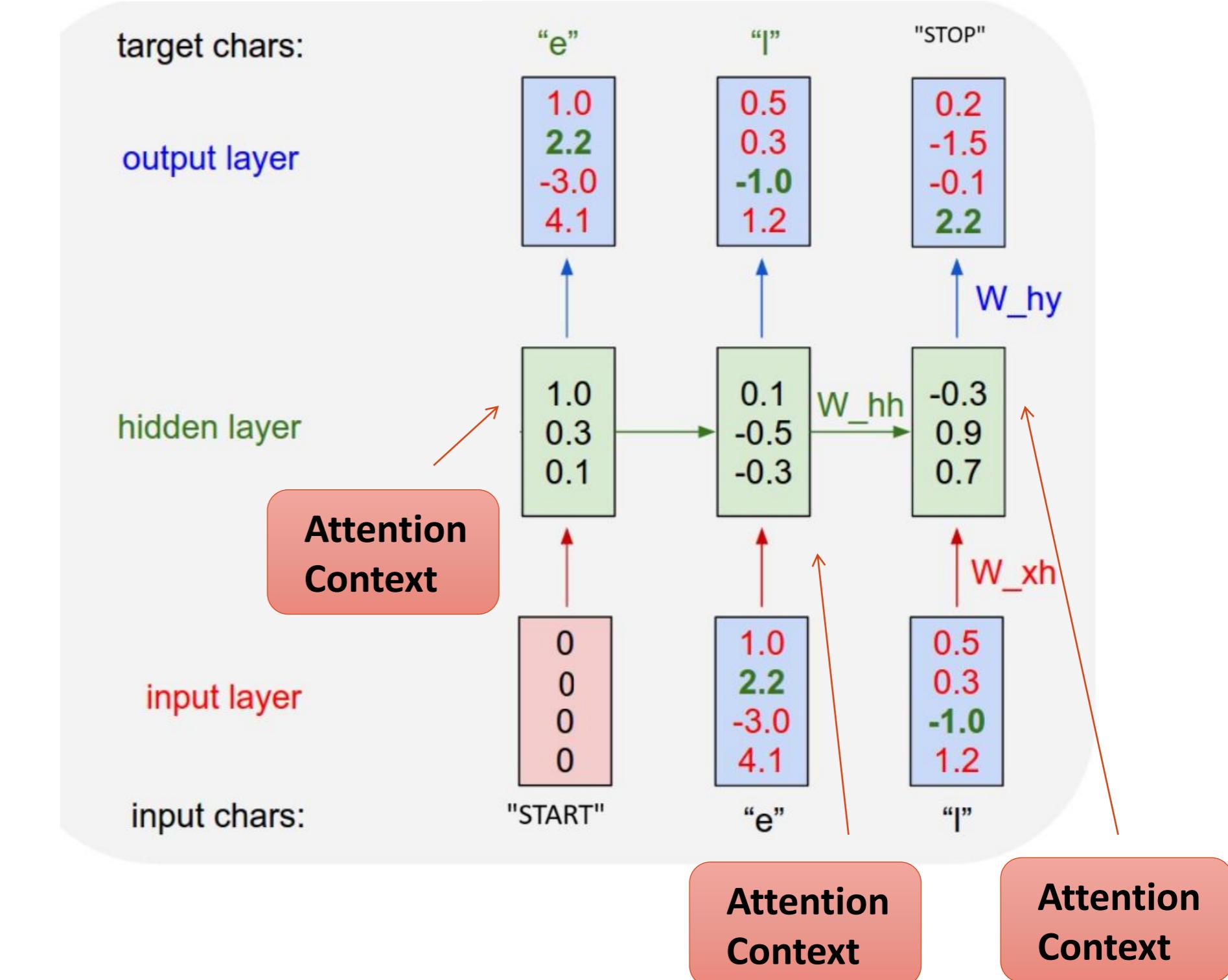
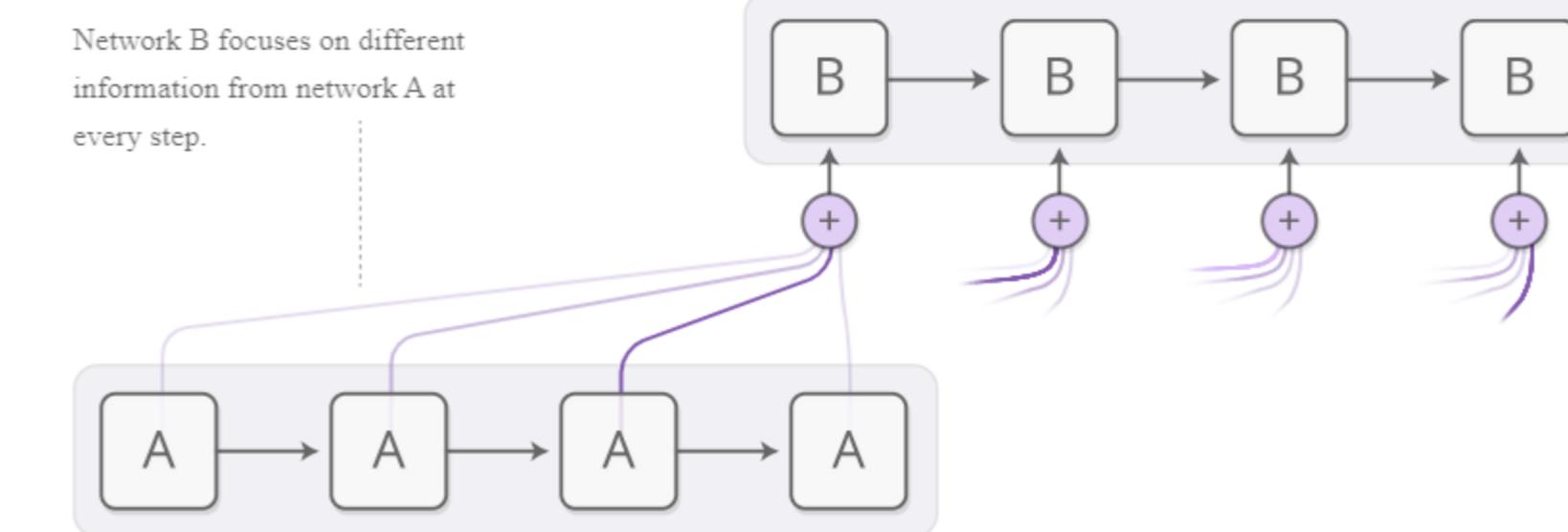
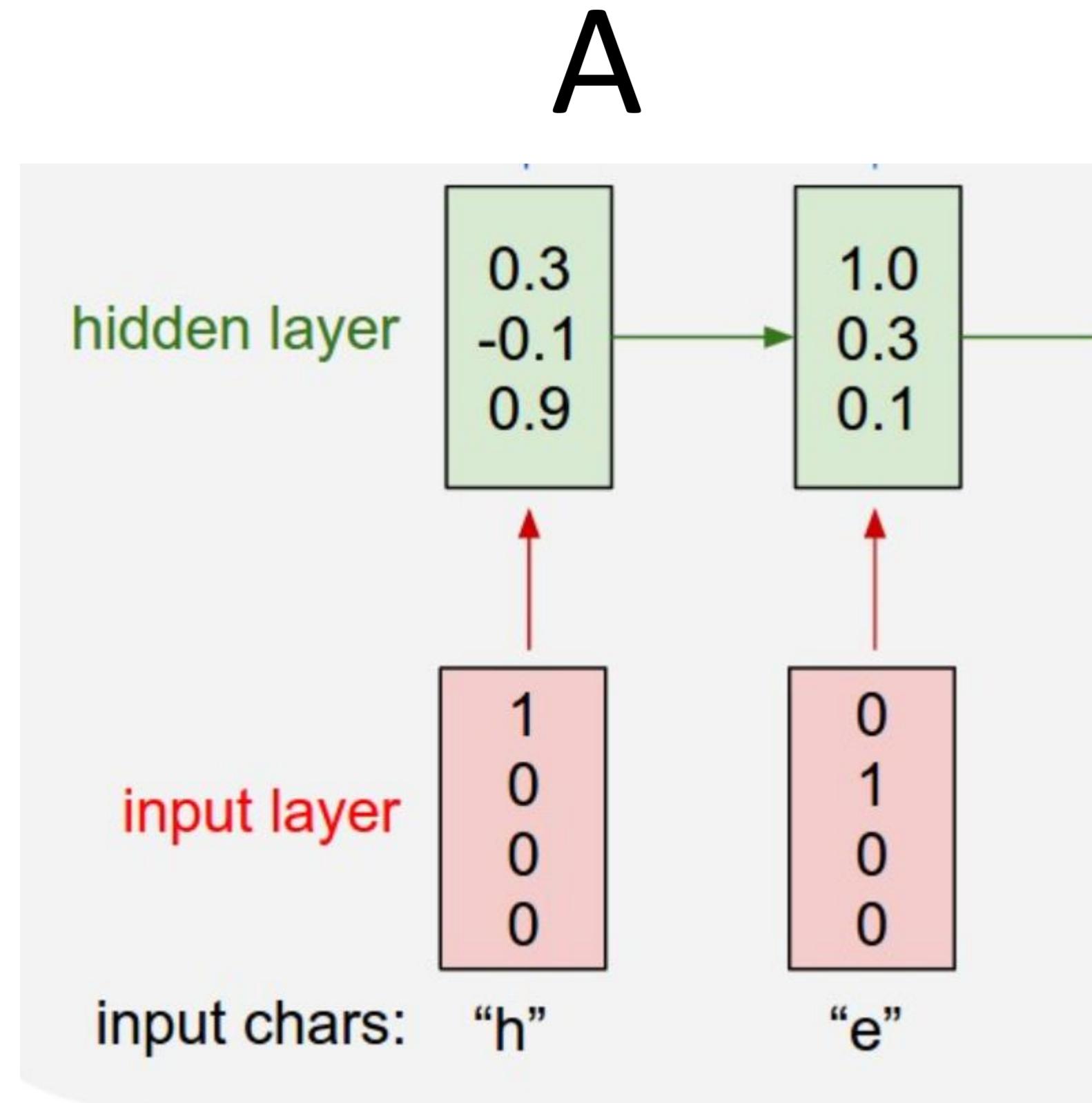
B



Source: <https://karpathy.github.io/2015/05/21/rnn-effectiveness/> modified

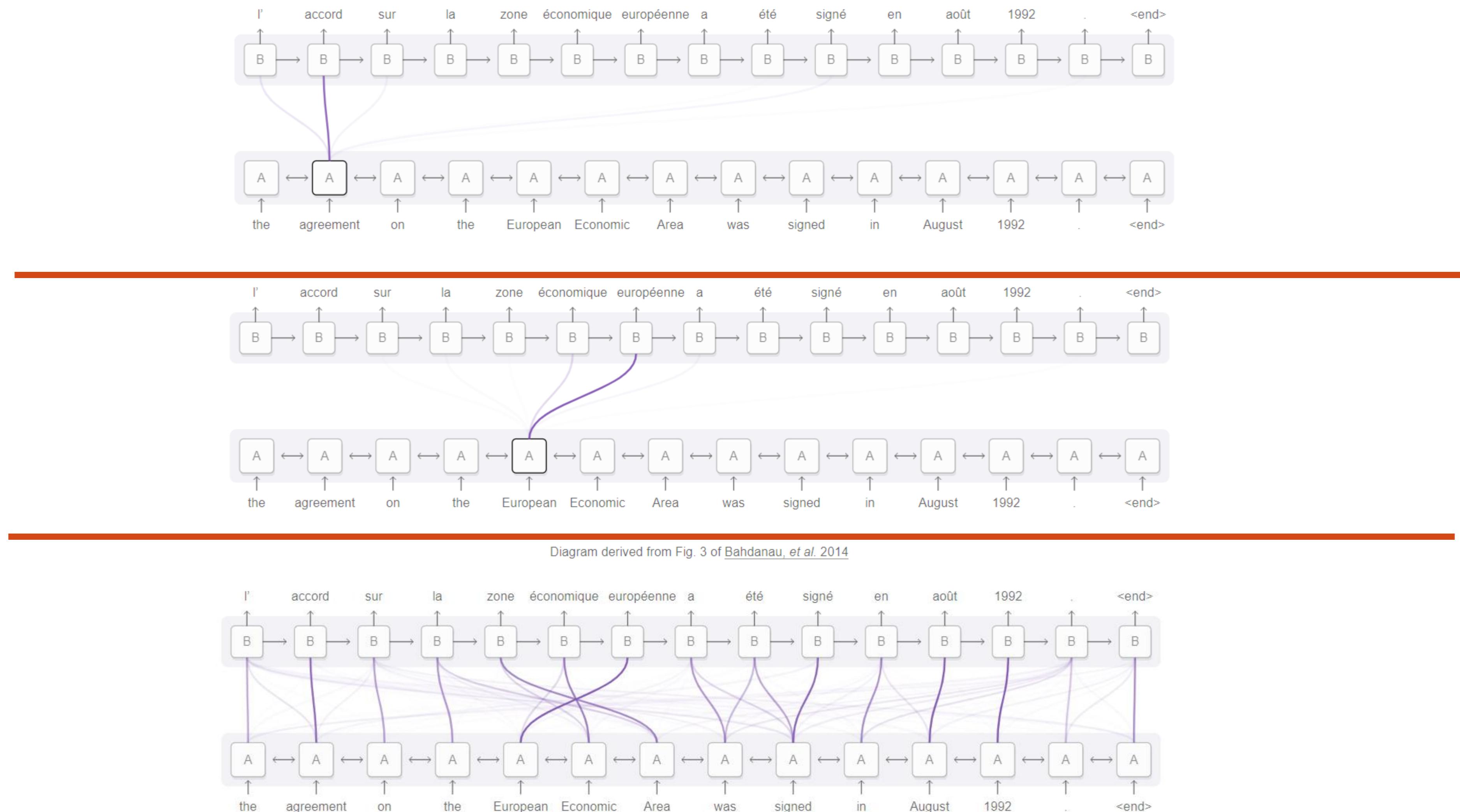
Sequence to Sequence Generation with Attention

The Decoder with attention



Source: <https://karpathy.github.io/2015/05/21/rnn-effectiveness/> modified

How Attention behaves

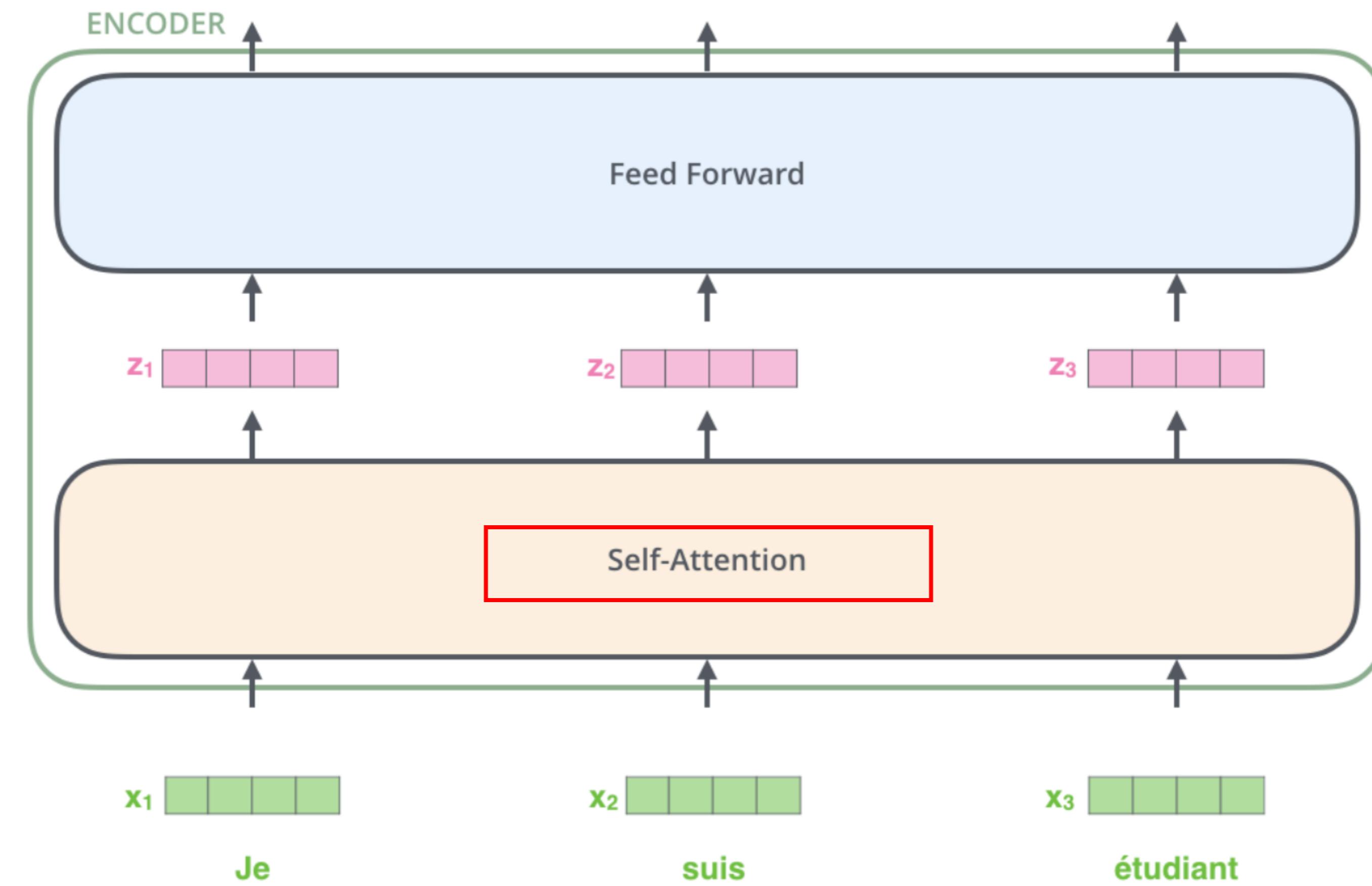


Source: <https://distill.pub/2016/augmented-rnns/> (Google Brain team)

Disclaimer: This presentation is intended solely for knowledge-sharing purposes. Statements of fact and/or opinion(s) expressed are only of that of the presenter(s), and not that of nasscom. nasscom does not endorse the content in the presentation and assumes no responsibility for the accuracy or completeness of the information provided herein.

Attention is all you need - Transformer

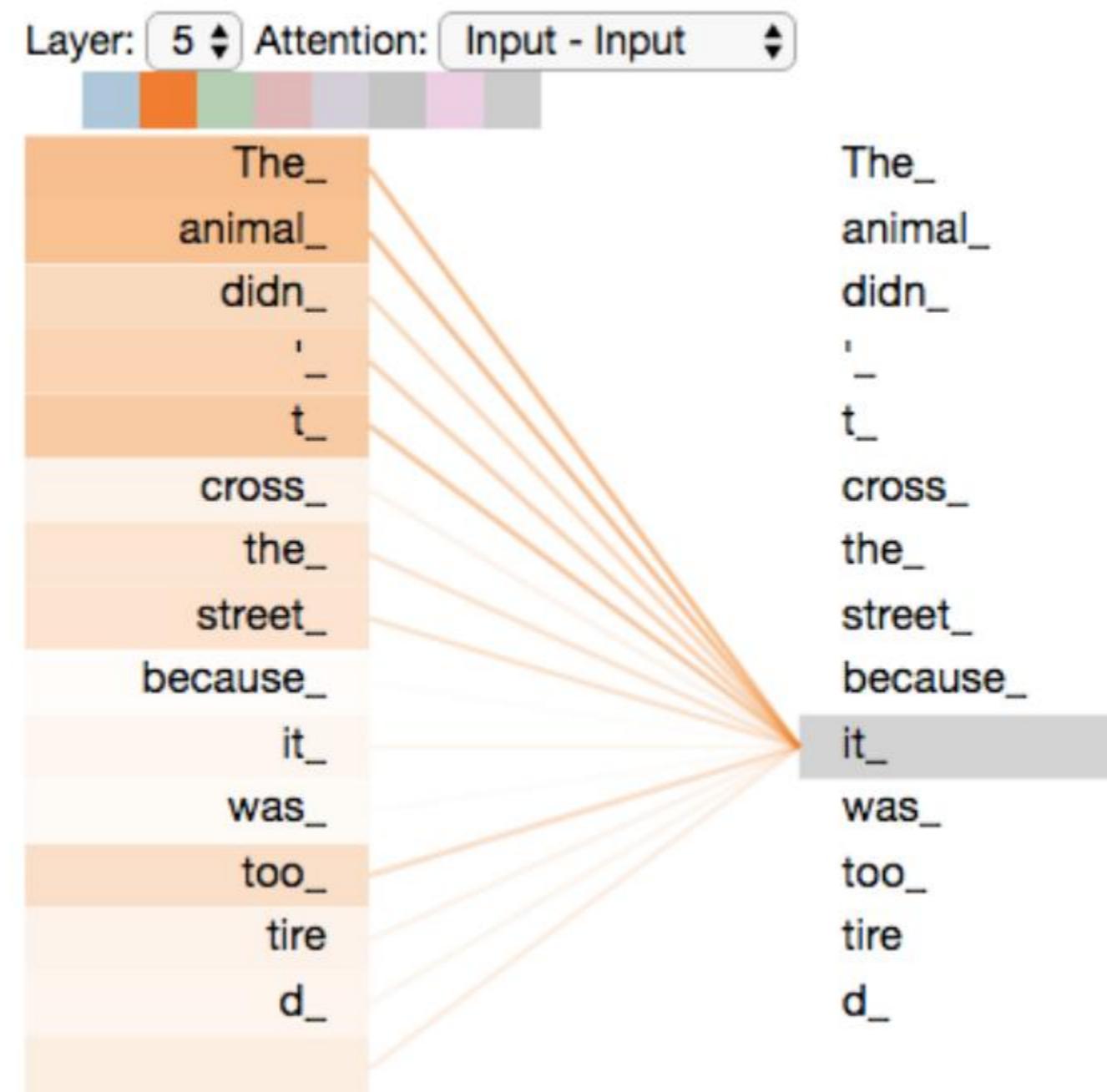
Vaswani et al, 2017



Source: <https://jalammar.github.io/illustrated-transformer/>

Attention is all you need - Transformer

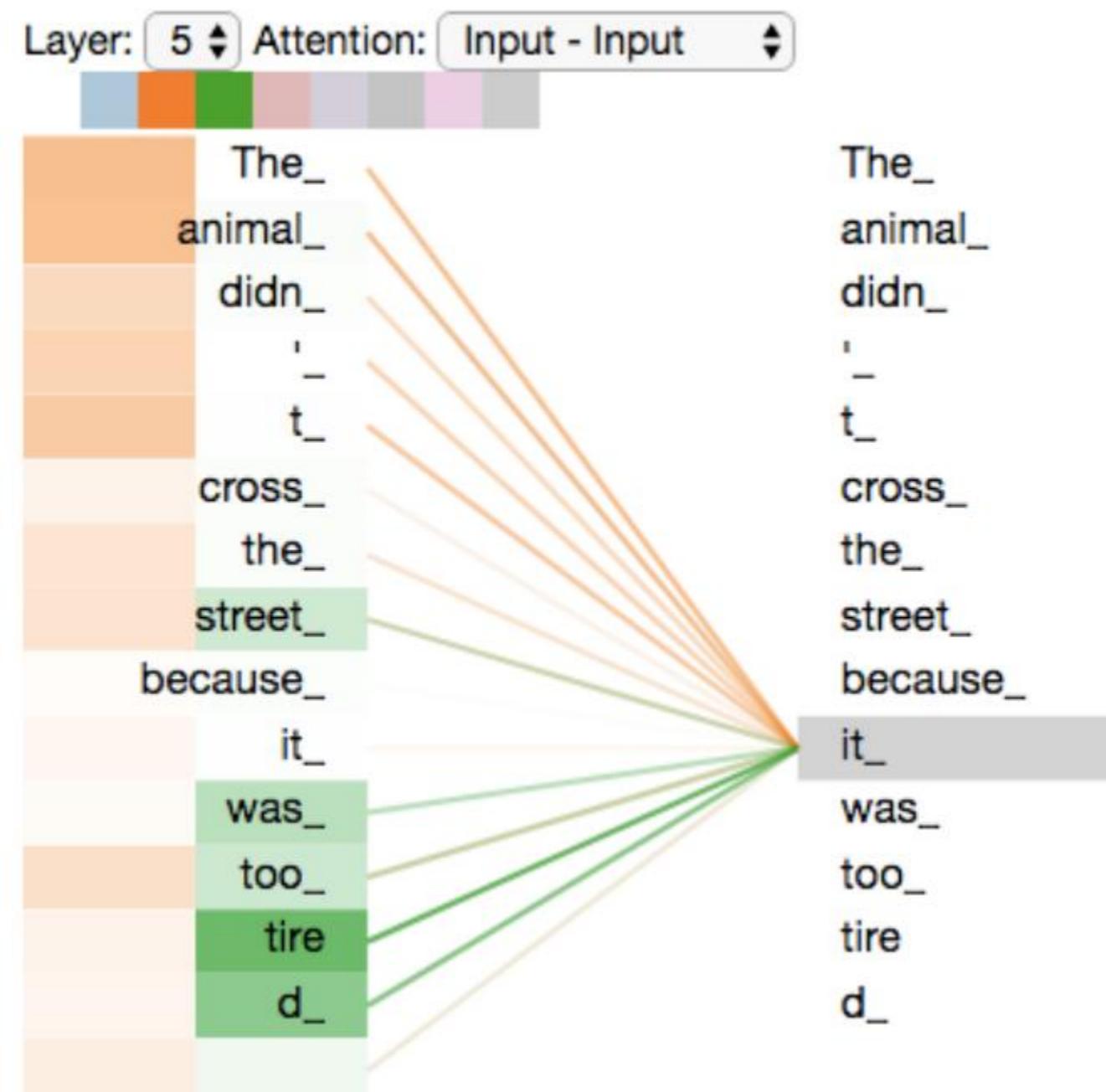
Self Attention



Source: <https://jalammar.github.io/illustrated-transformer/>

Attention is all you need - Transformer

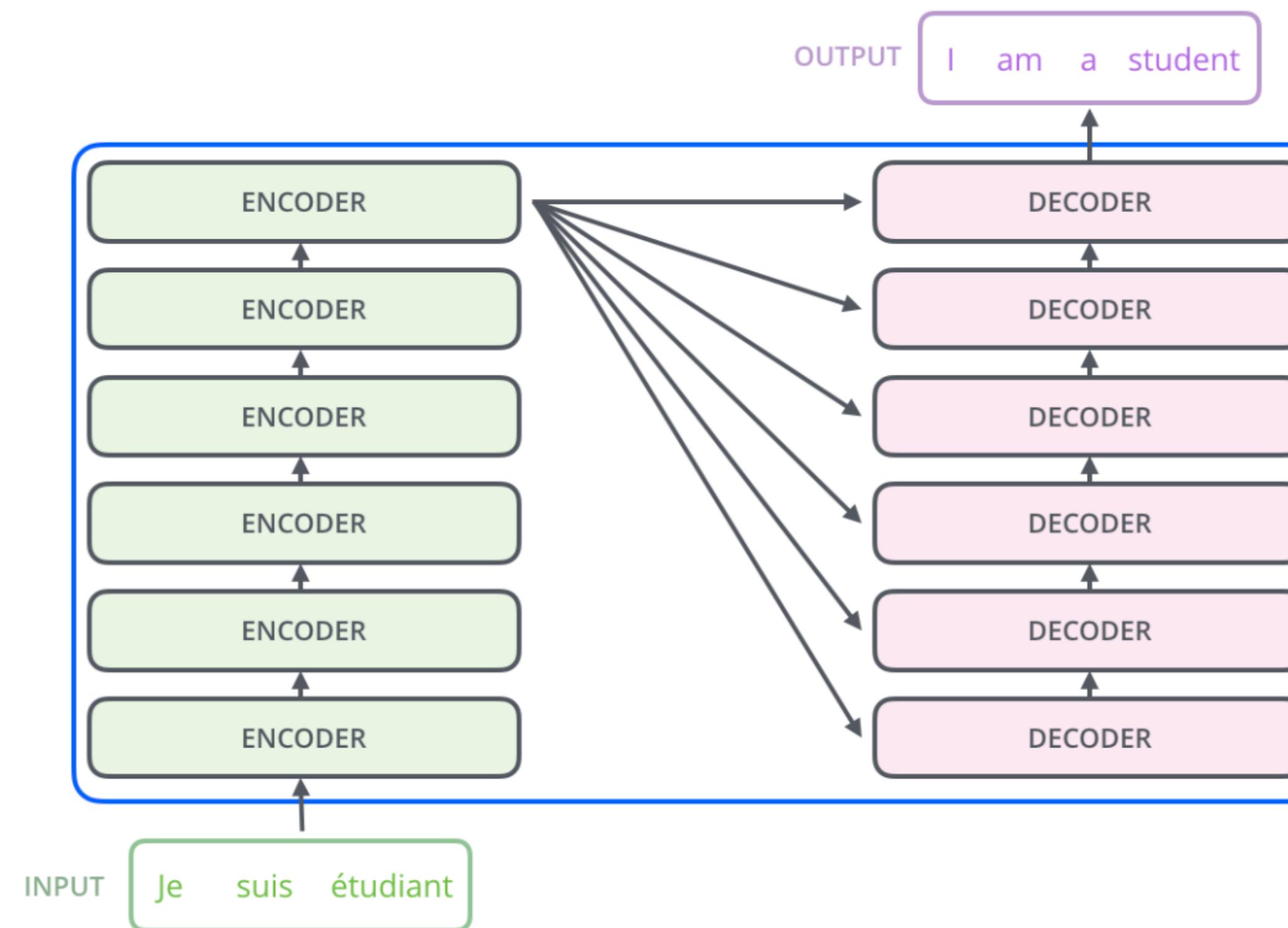
Multi-Headed Self Attention



Source: <https://jalammar.github.io/illustrated-transformer/>

Attention is all you need - Transformer

Stack up a lot of those self-attention modules



Source: <https://jalammar.github.io/illustrated-transformer/>