# BRIDGING THE GAP BETWEEN BASIC NEURAL LANGUAGE MODELS, TRANSFORMERS, AND MEGATRON

MAGNUS EKMAN, PH.D., DIRECTOR ARCHITECTURE

JARED CASPER, PH.D., SENIOR DEEP LEARNING SCIENTIST

# AGENDA

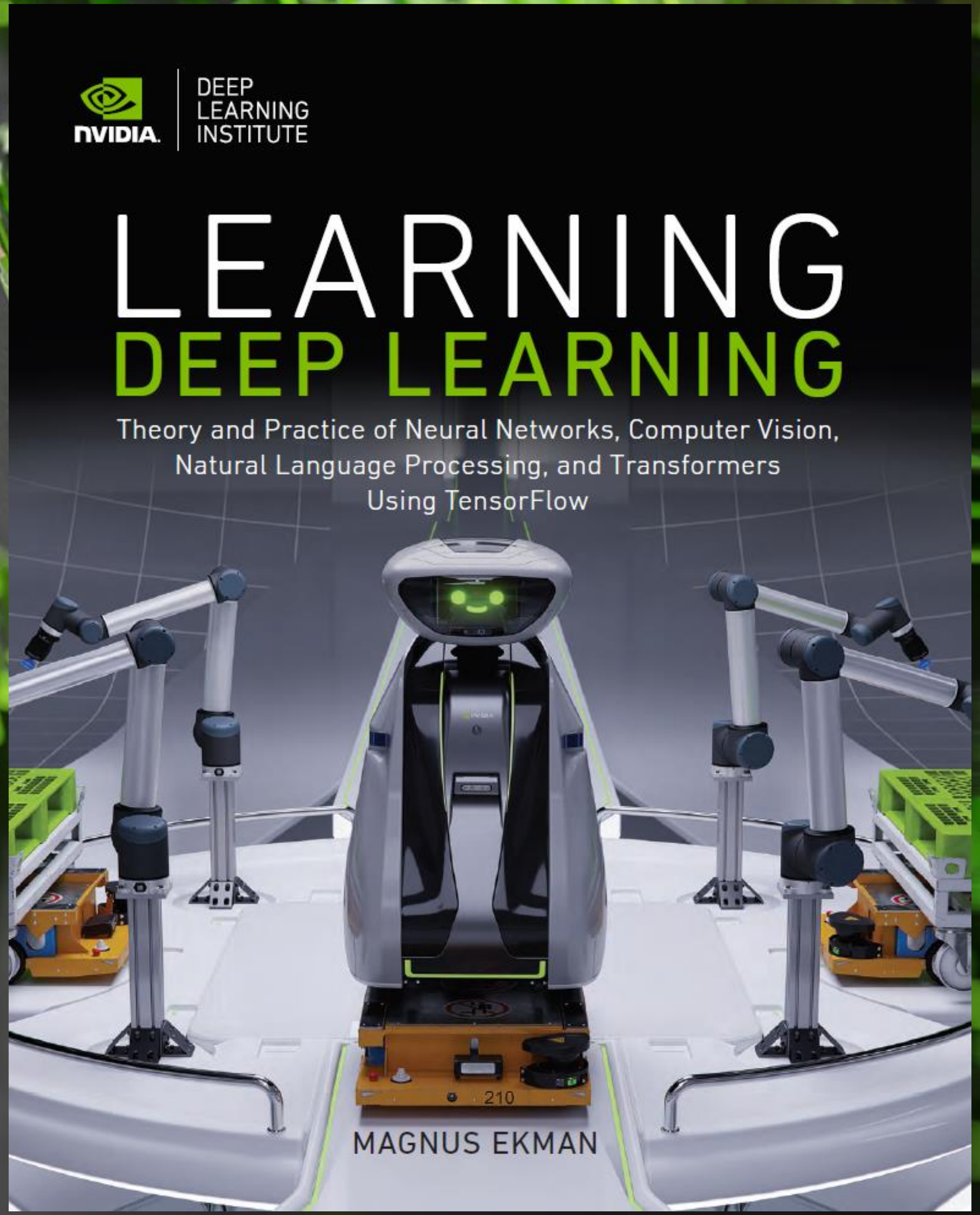
Basic neural language models and autoregression

Encoder-decoder network for language translation

The attention mechanism

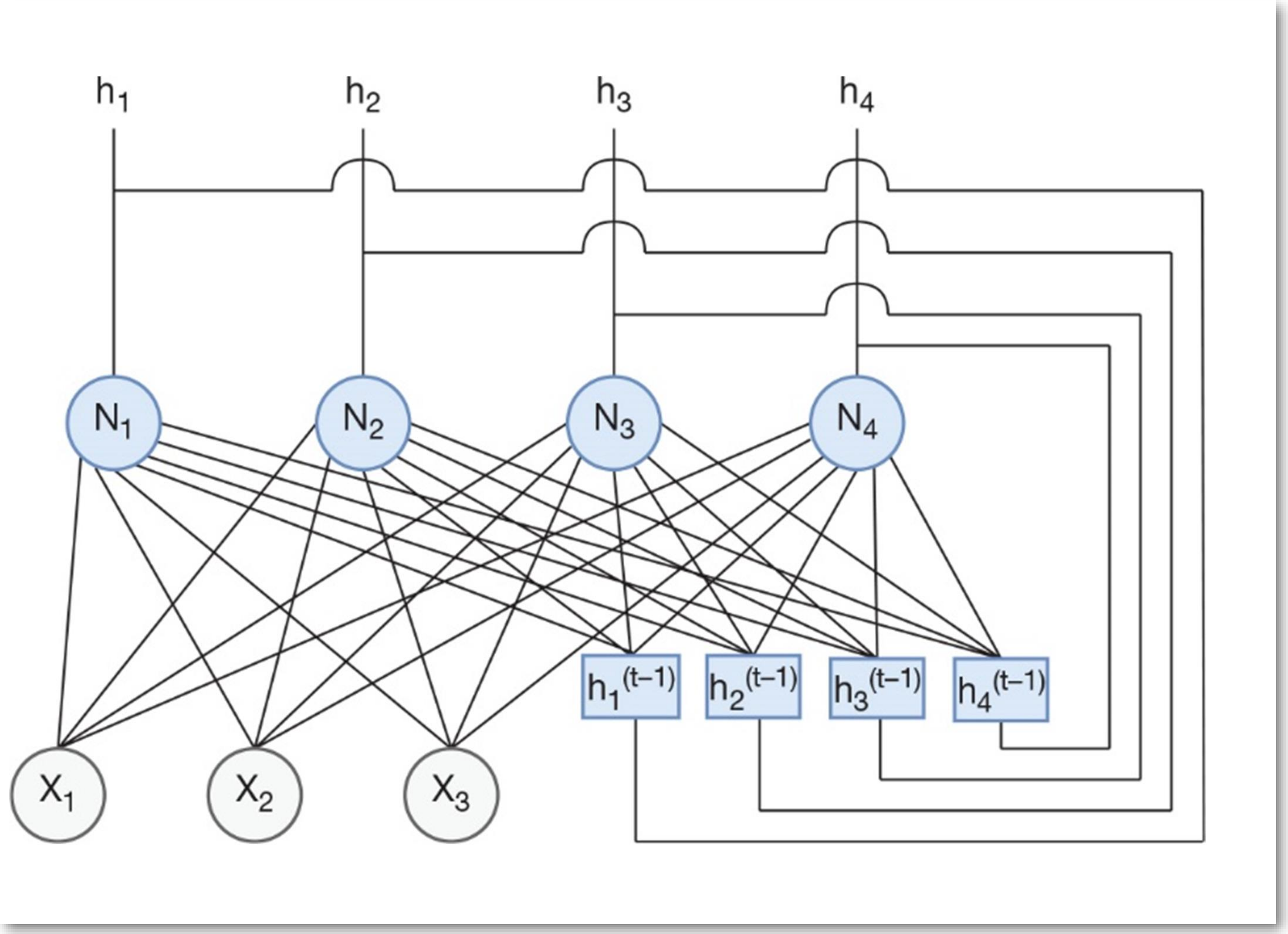
The Transformer, GPT, and BERT

Scaling transformer models with Megatron



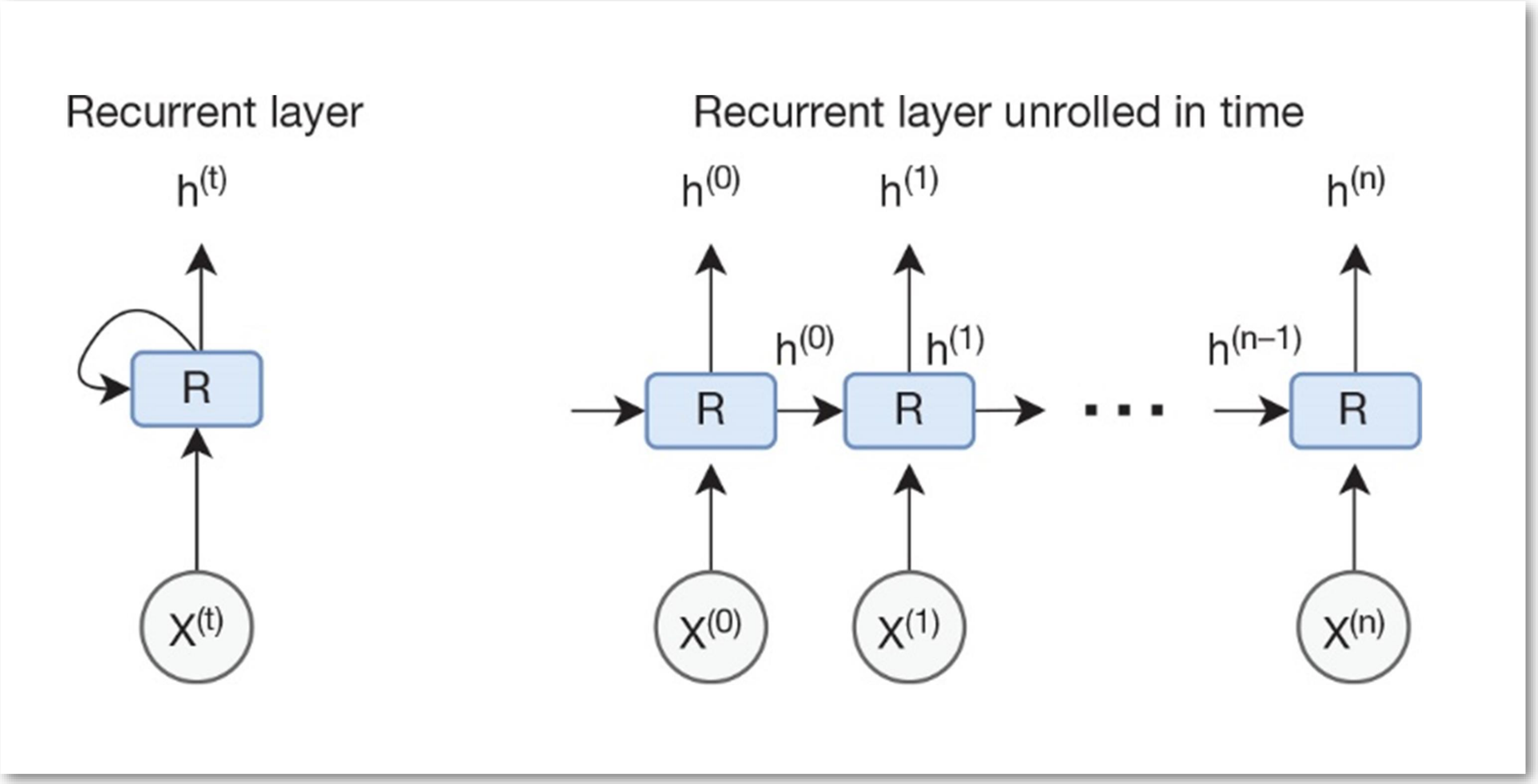
# PREDICTING SEQUENTIAL DATA

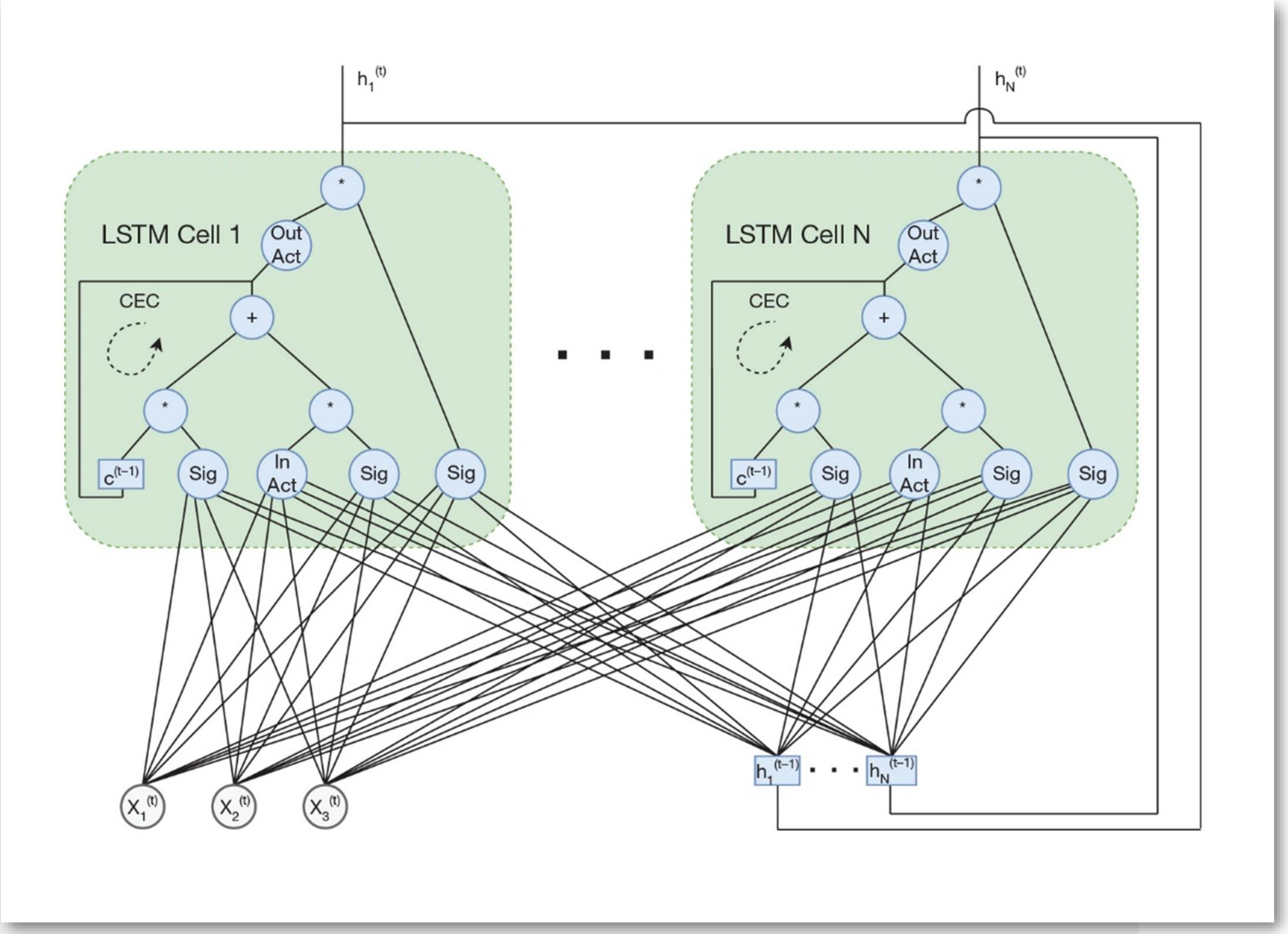
Using a recurrent neural network (RNN)



# UNROLLING A RECURRENT NETWORK IN TIME

Converts it into a feedforward network





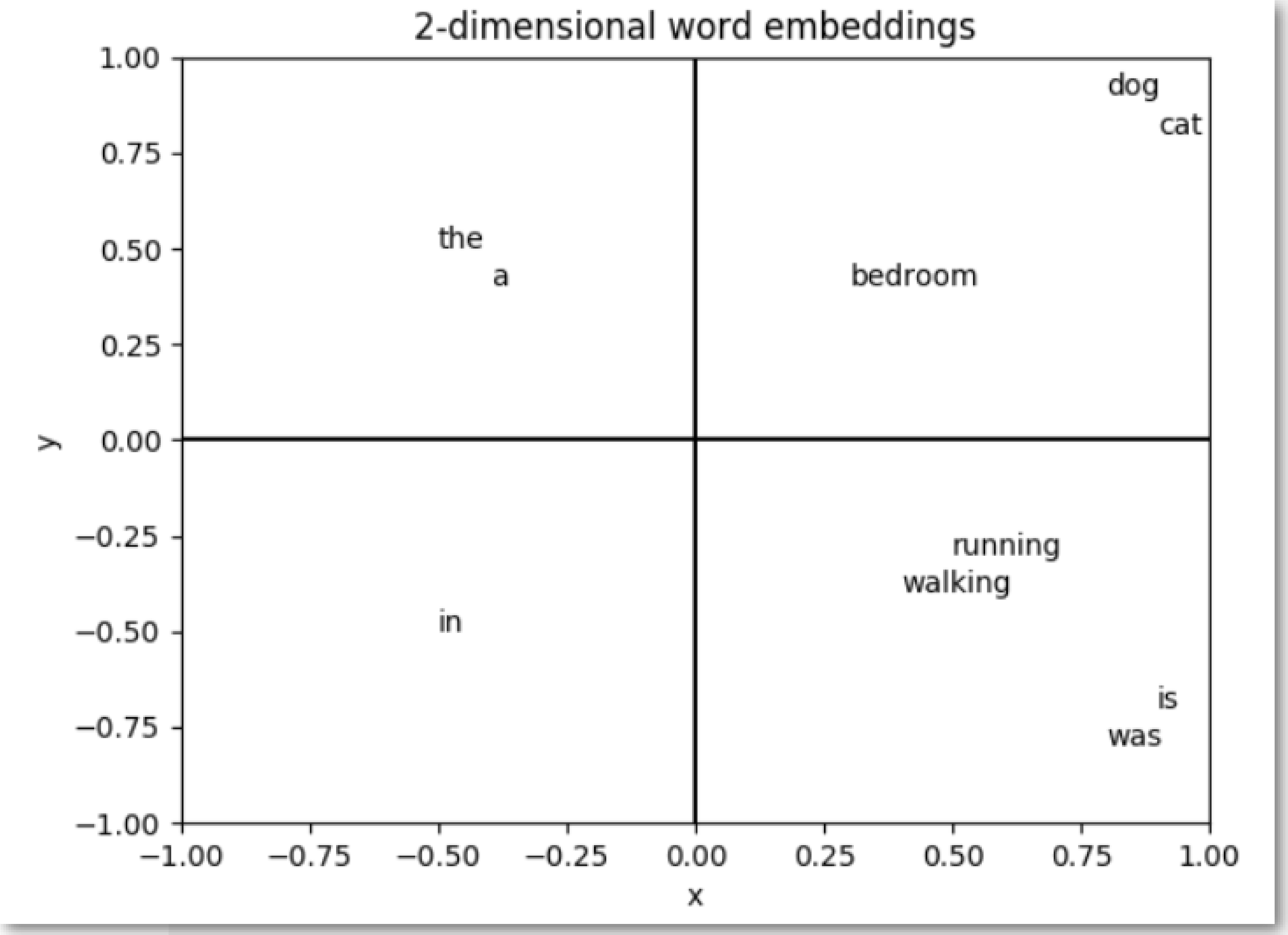

# LONG SHORT-TERM MEMORY

Drop-in replacement for simple unit in RNN

- Gated units
- More weights to train
- Better at capturing long-term dependencies

# NETWORK LAYERS USED FOR WORD INPUT/OUTPUT

- Input: Embedding layer
- Output: Softmax



# BASIC NEURAL LANGUAGE MODELS AND AUTOREGRESSION

# WHAT IS A LANGUAGE MODEL?

A model that describes how likely a sequence of words is

- Example use-case: Text autocompletion
- Likelihood of sequence depends on training data

**Input sequence**

“Deep Learning”

Language model

**Probability for each word in the vocabulary**

| Word | Probability |
|---|---|
| Book | High |
| Model | High |
| … | … |
| Food | Low |
| travel | Low |

# MORE LANGUAGE MODEL USE CASES

**Speech recognition - what did they say?**

*"Recognize speech using common sense"*

*"Wreck a nice beach you sing calm incense"*

**Which translation is most likely for "Je suis étudiant"?**

*"I am student“*

*"I am a student"*

*"Student am I"*

*"A student I am"*

# WHAT IS AN N-GRAM?

n consecutive words from a body of text

Training Text

*"The more I read, the more I learn, and I like it more than anything else."*

*n-grams with n=2*

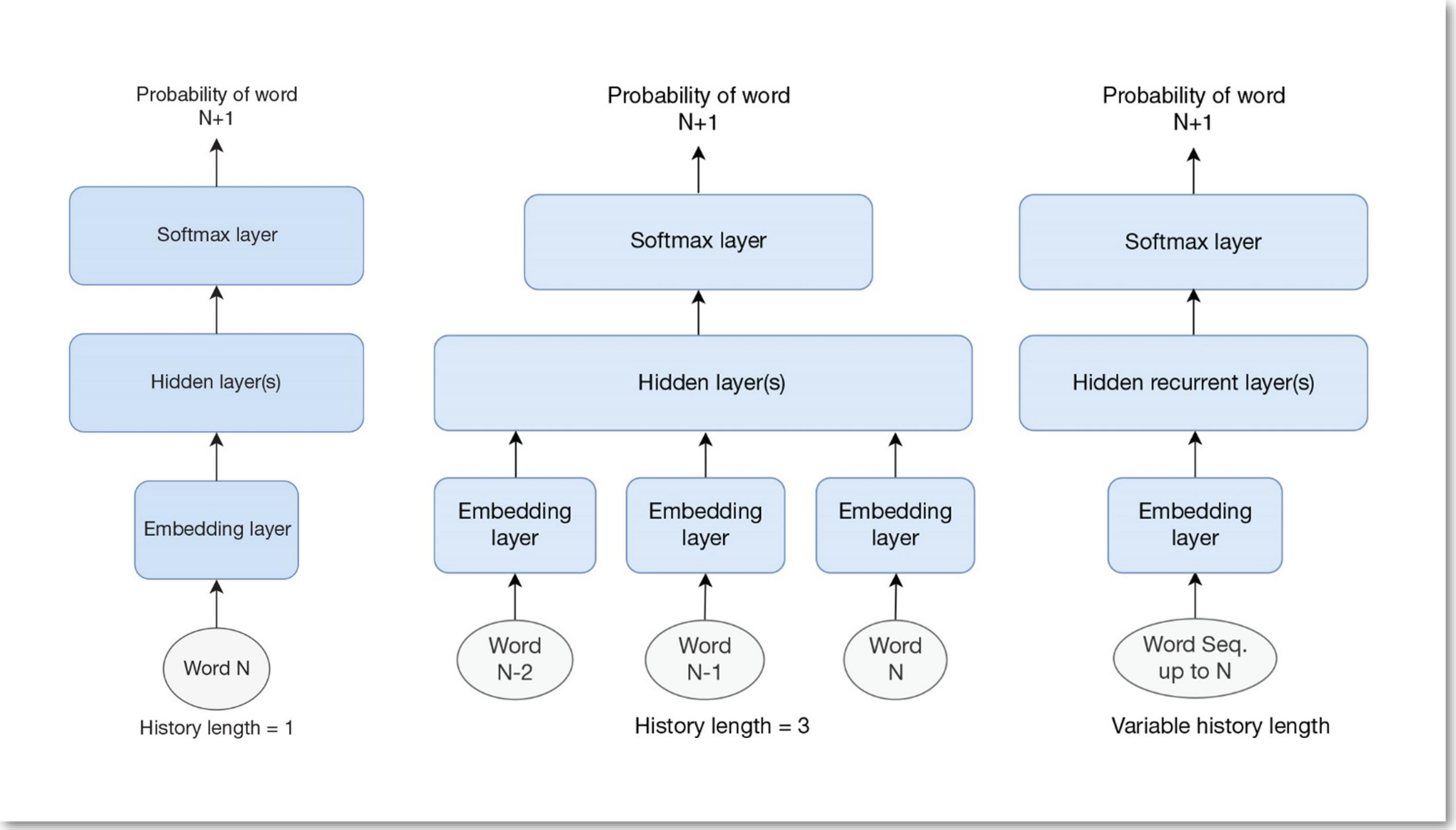
*/the more/ /more i/ /i read/ /read the/ /the more/ /more i/ /i learn/ /learn and/ /and i/ /i like/ /like it/ /it more/ /more than/ /than anything/ /anything else/*

# PREDICT NEXT WORD WITH 2-GRAM MODEL

| First word | Predicted word | # of occurences | Probability given starting word |
|---|---|---|---|
| and | i | 1 | 100% |
| anything | else | 1 | 100% |
| i | learn | 1 | 33% |
| | like | 1 | 33% |
| | read | 1 | 33% |
| it | more | 1 | 100% |
| learn | and | 1 | 100% |
| like | it | 1 | 100% |
| more | i | 1 | 67% |
| | than | 2 | 33% |
| read | the | 1 | 100% |
| than | anything | 1 | 100% |
| the | more | 2 | 100% |

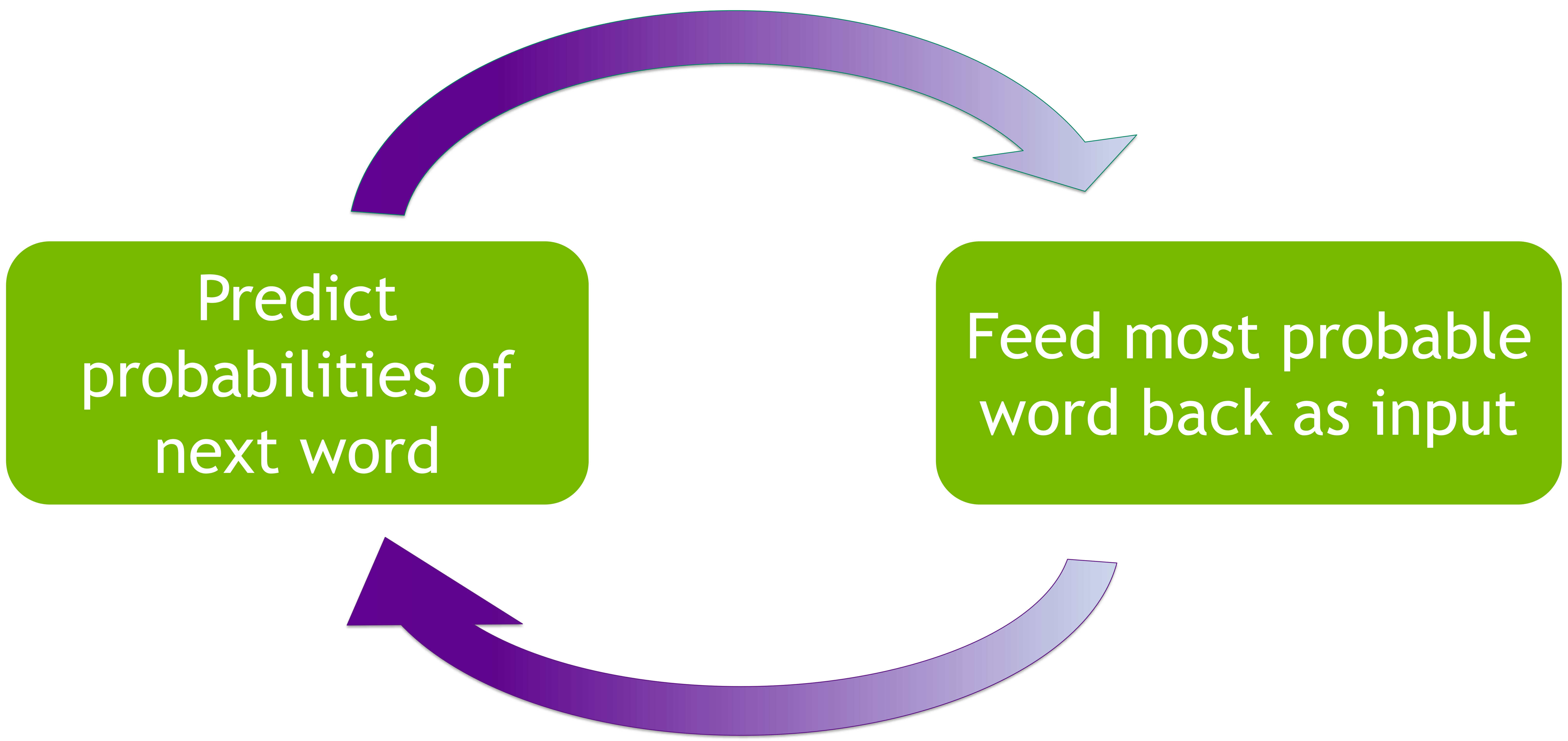
# NEURAL LANGUAGE MODELS

Fixed or variable length



# AUTOREGRESSION

Dynamically generate input sequence



# AUTOREGRESSION EXAMPLE

1. Input: “Deep” → Prediction: “learning”

2. Input: “learning" → Prediction: “is"

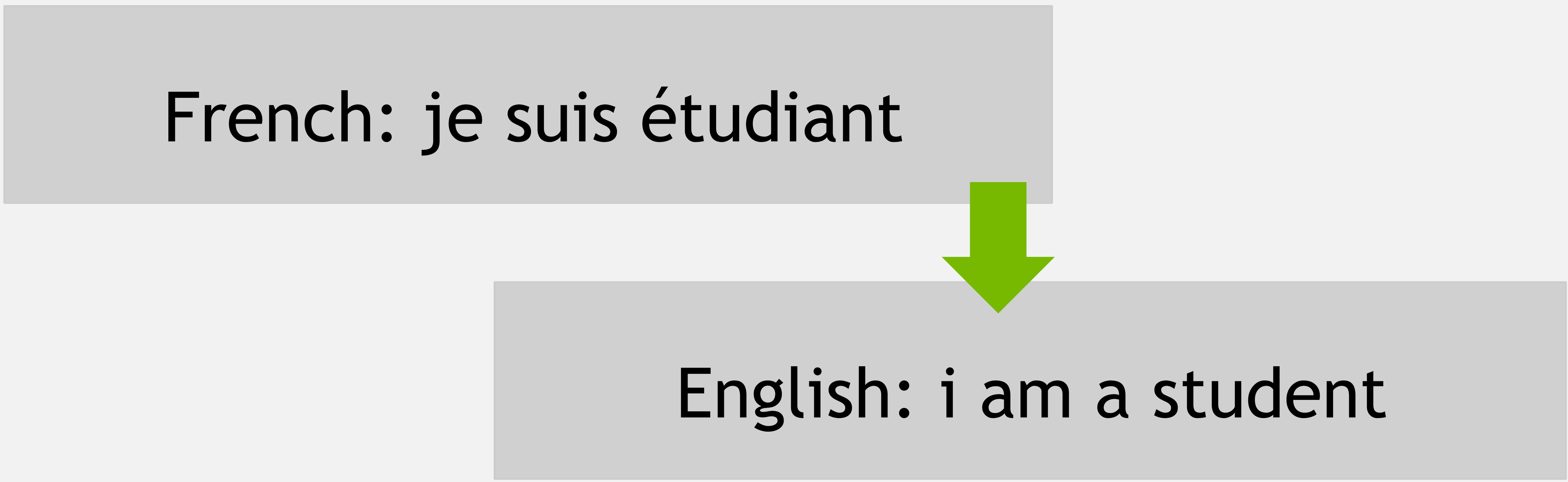
3. Input: “is” → Prediction: “awesome”

Result: “Deep learning is awesome”

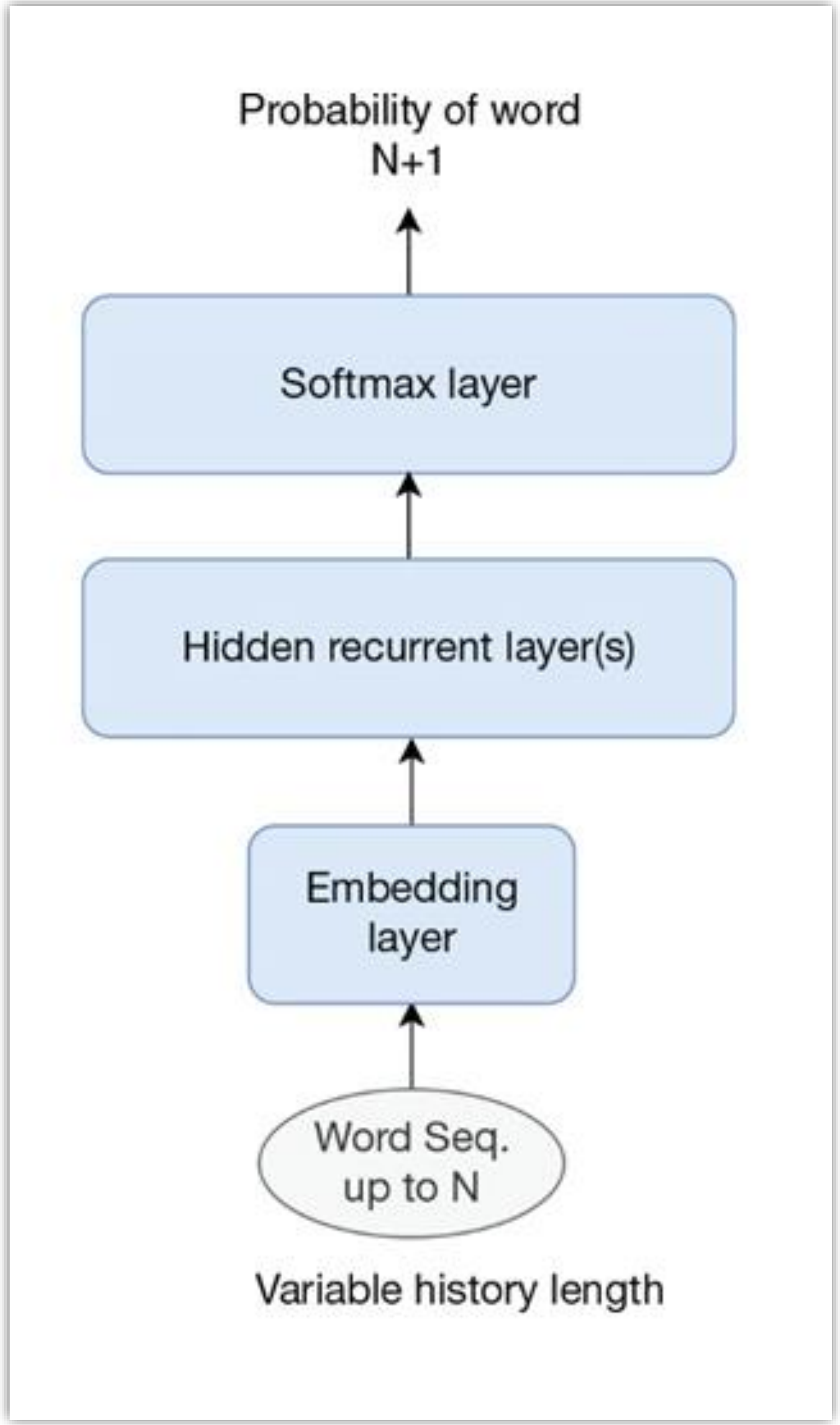
# ENCODER/DECODER NETWORK FOR LANGUAGE TRANSLATION

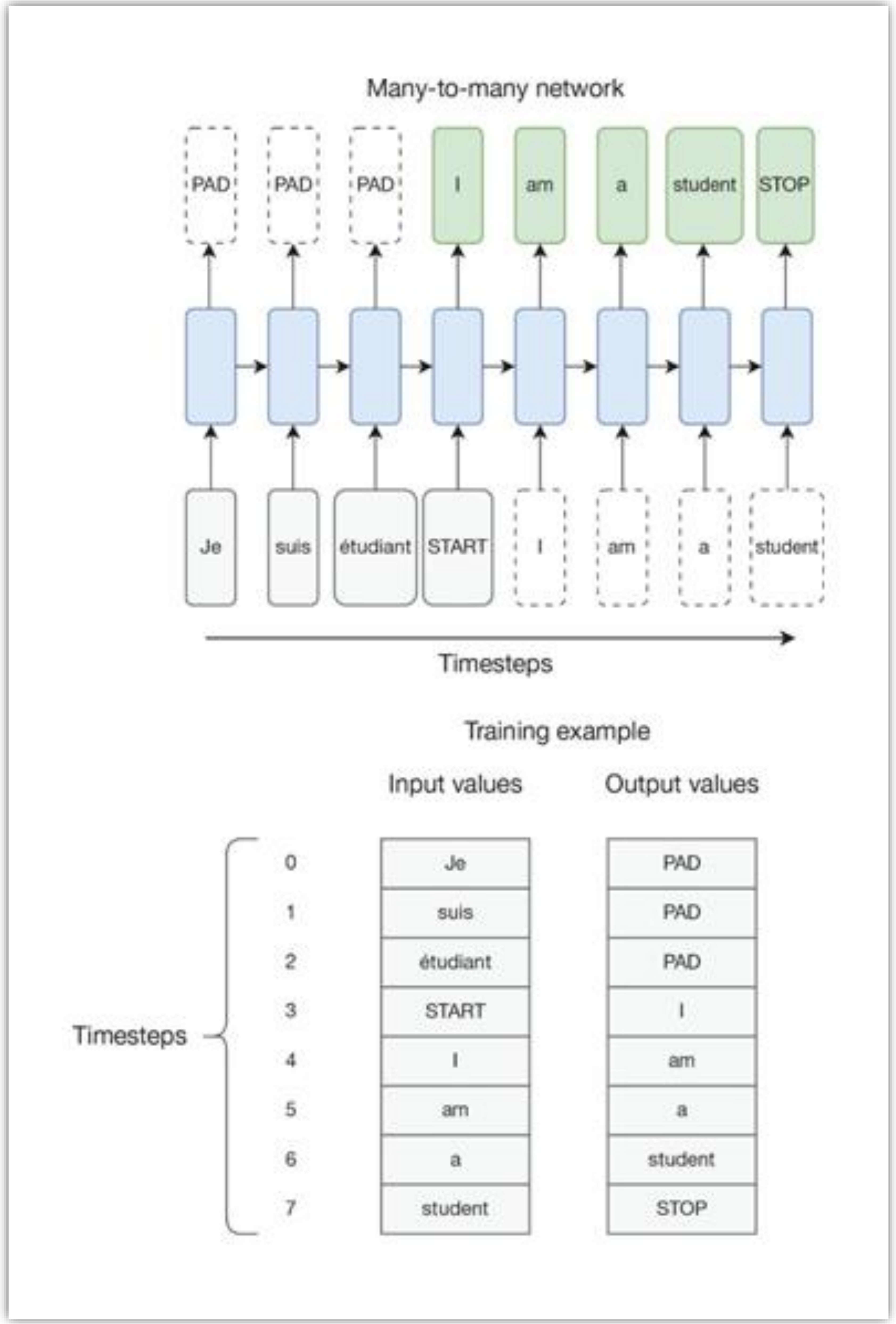
# LANGUAGE TRANSLATION

- View it as a text autocompletion problem
- Training sequence: “je suis étudiant START i am a student STOP”
- Now complete the sequence “je suis étudiant START”
- Just use a neural language model!

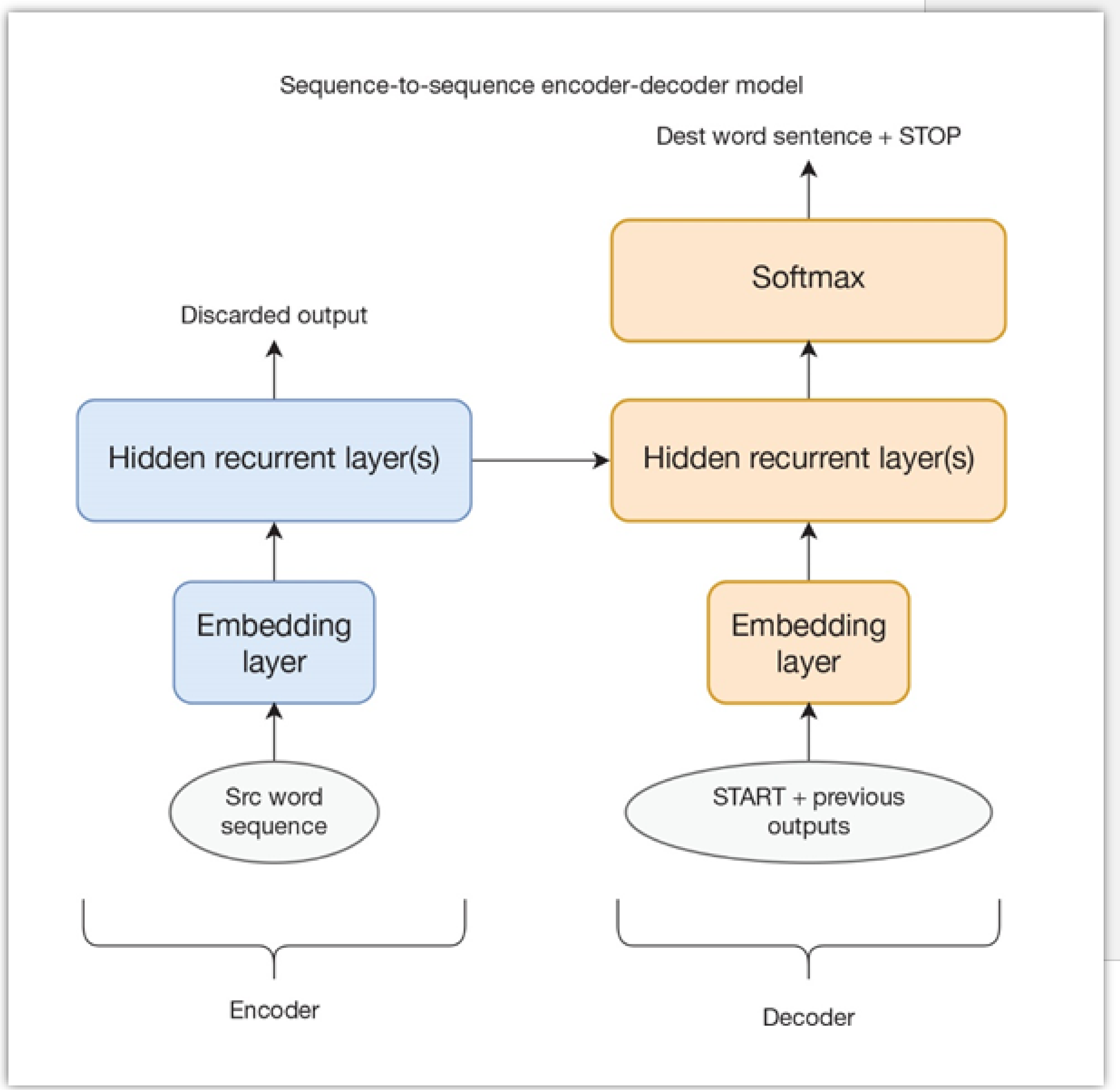


# LANGUAGE MODEL APPROACH



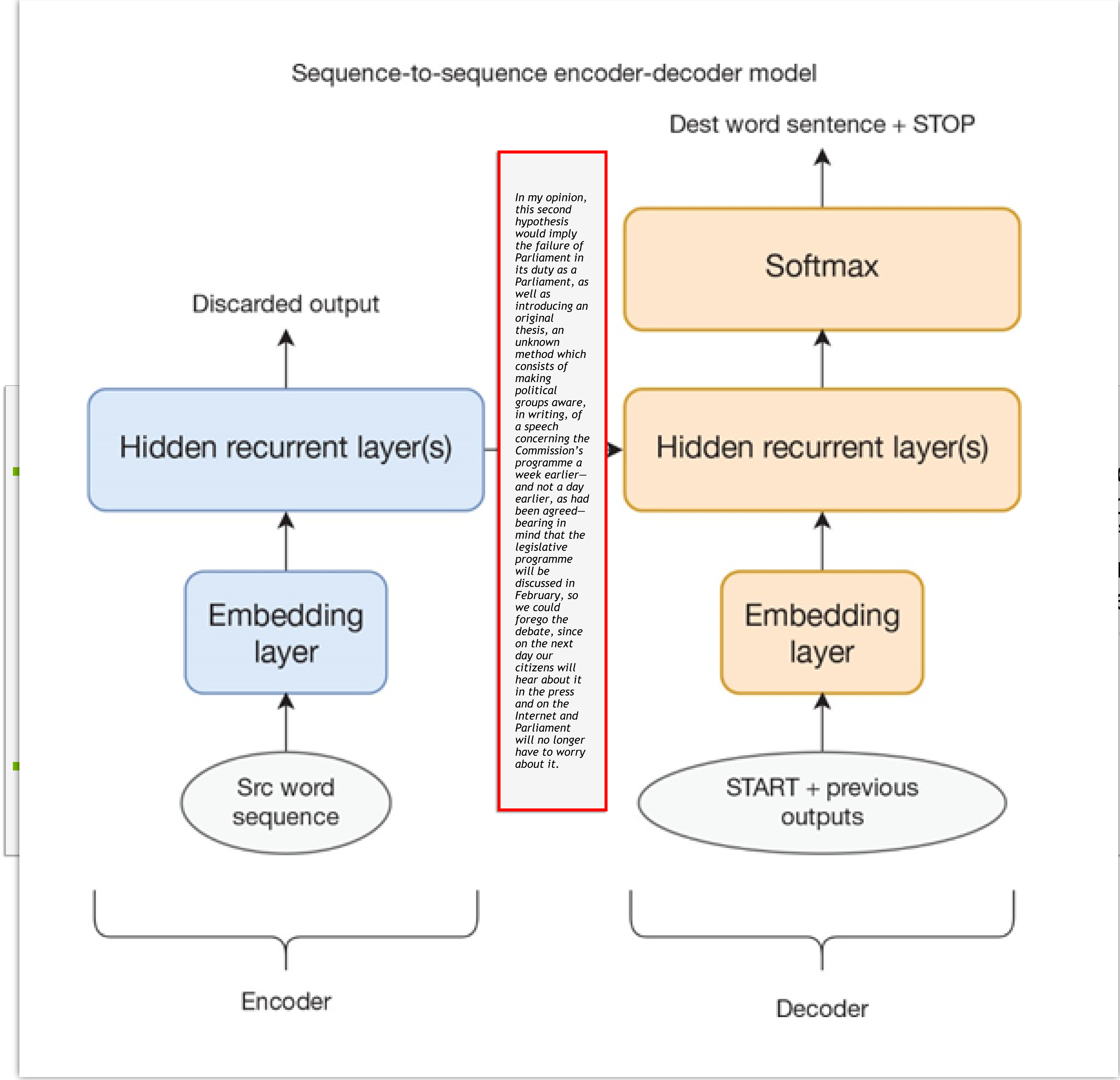



# ENCODER/DECODER NETWORK




- Encoder encodes French sentence into an intermediate representation
- Decoder decodes this intermediate representation into an English sentence
- The decoder is just a neural language model
- Encoder embedding layer works with French words and decoder works with English words

# THE ATTENTION MECHANISM




duty as a
onsists of
's
mind that
ebate,
et and

# RICHER INTERMEDIATE REPRESENTATION

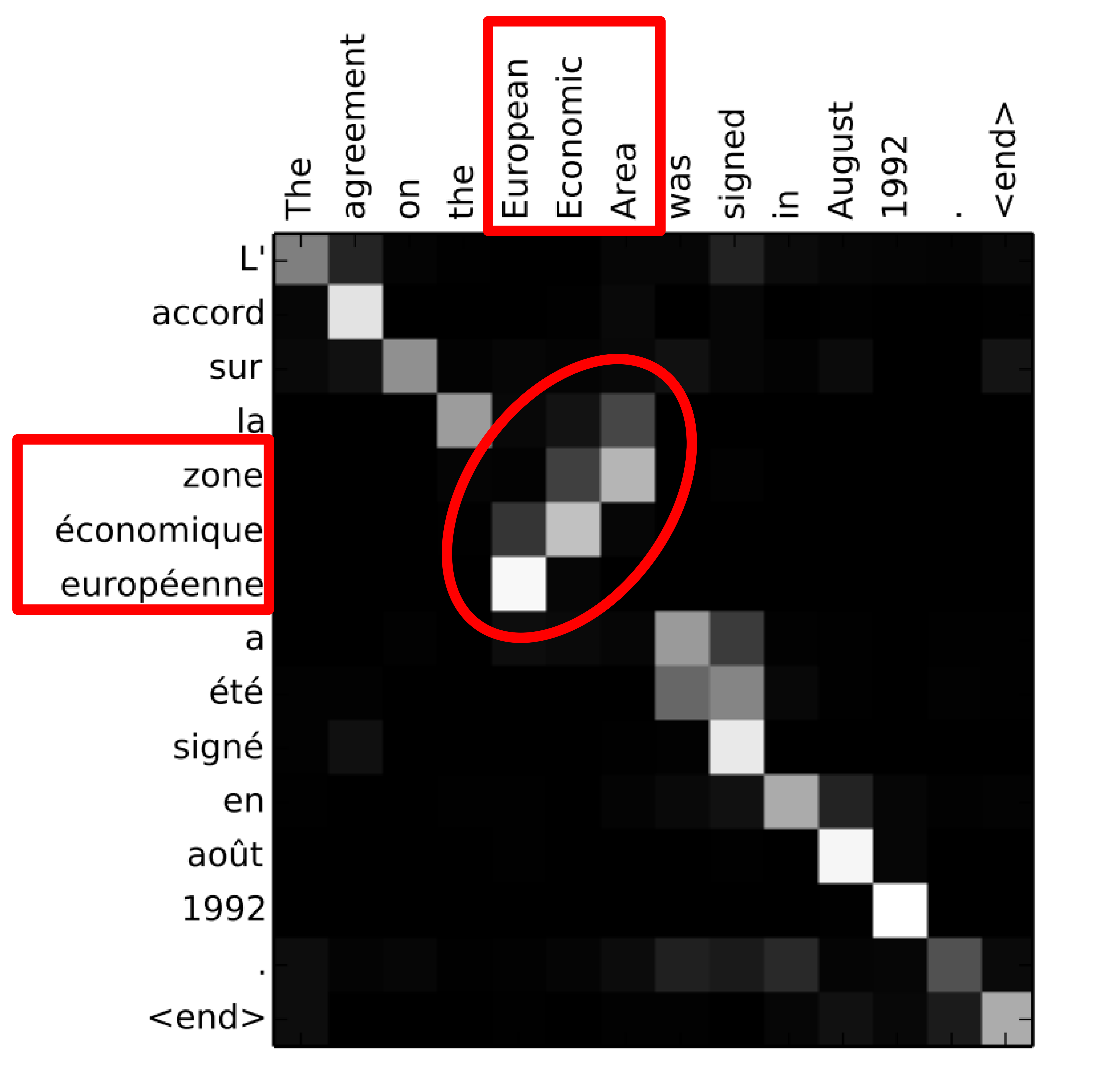
Let the decoder decide what to attend to

# SOFT ATTENTION

For now, ignore how alignment vector is computed

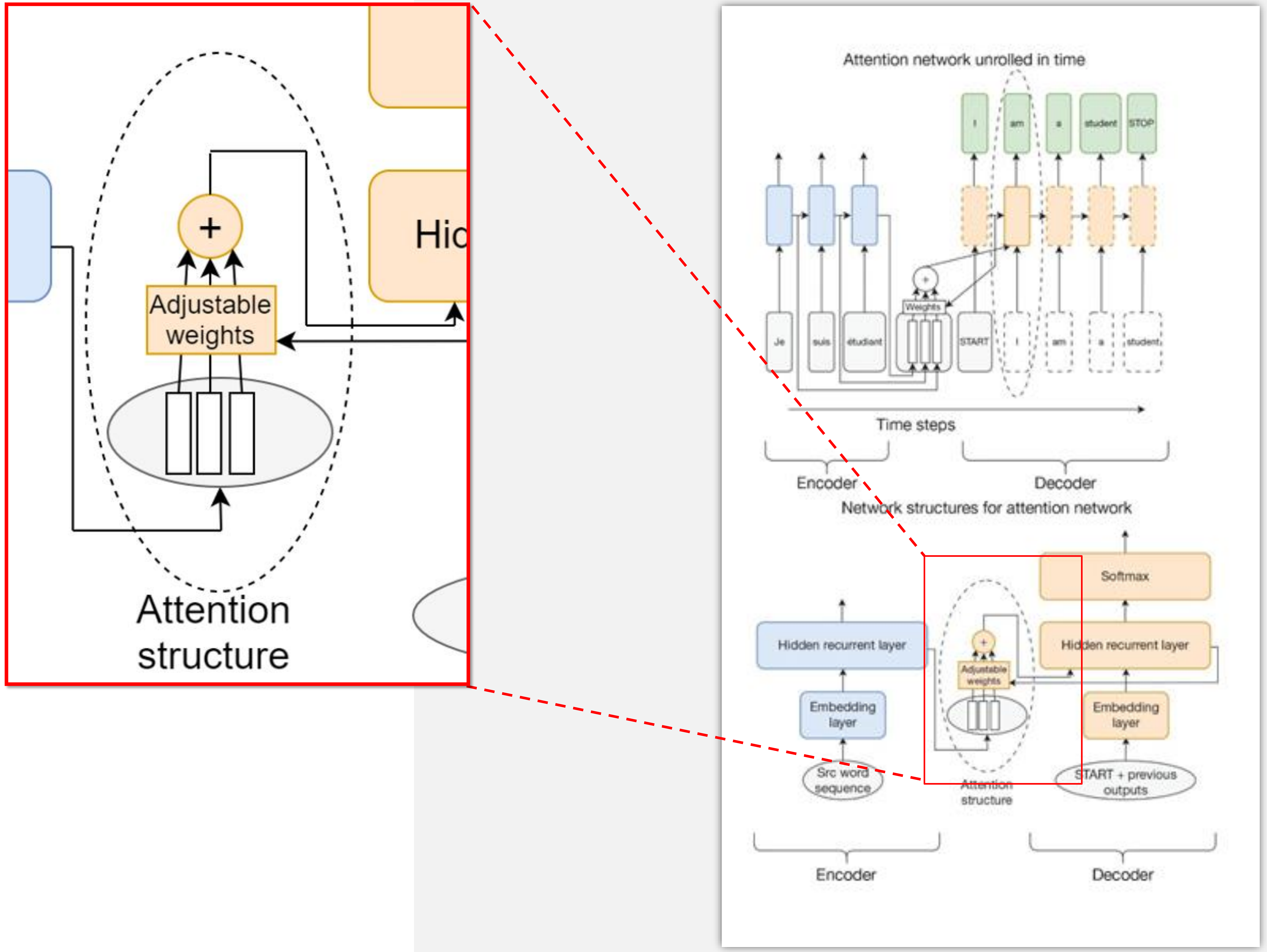
# ATTENTION EXAMPLE

- French: L'accord sur la zone économique européenne a été signé en août 1992.
- English: The agreement on the European Economic Area was signed in August 1992.



# TRANSLATION NETWORK WITH ATTENTION

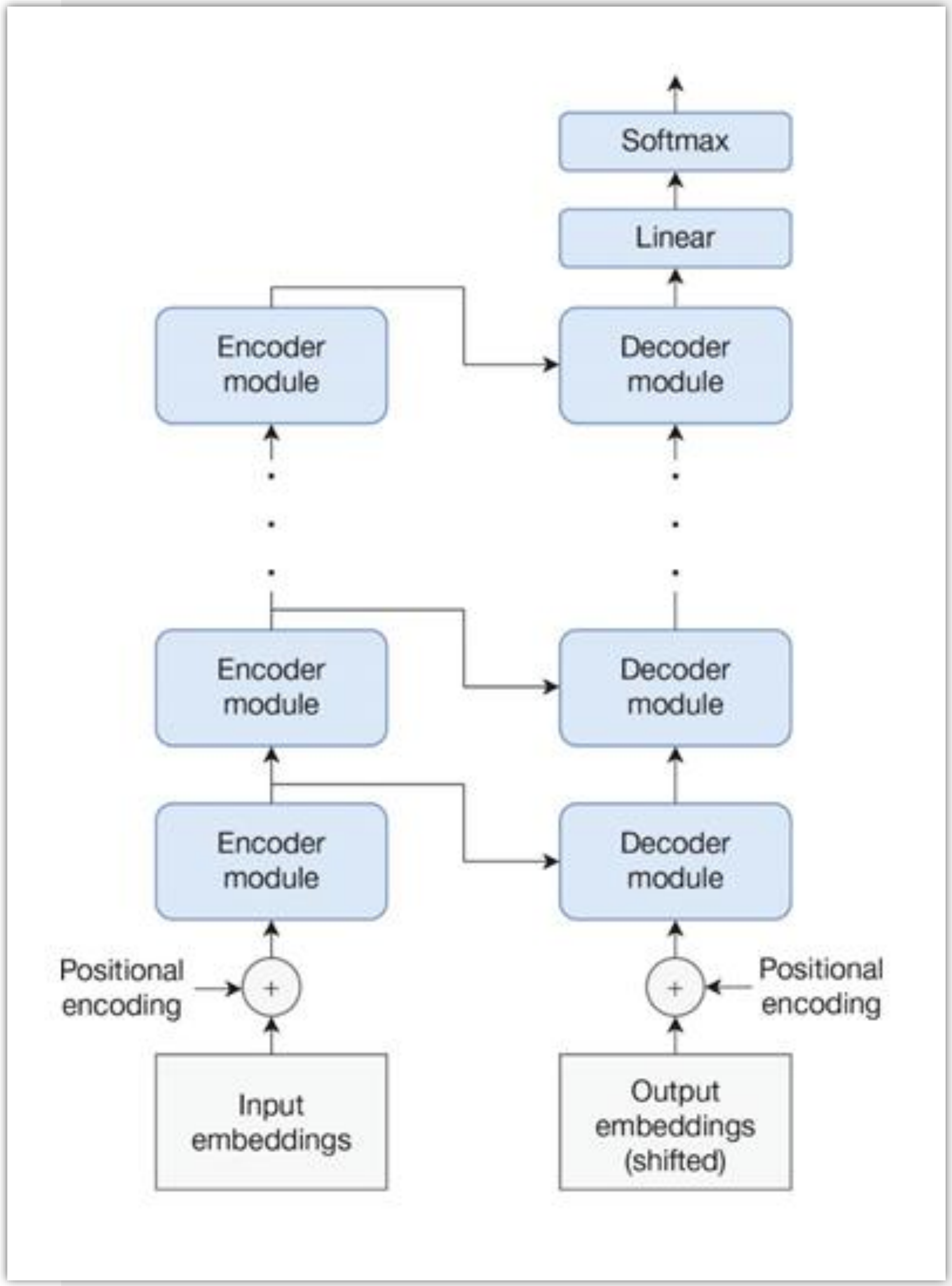
- Encoder produces one state vector for each input word
- Decoder creates an alignment vector for each output word
- Input to decoder is the weighted sum of encoder state vectors, where alignment vector serves as weights



# THE TRANSFORMER, GPT, AND BERT

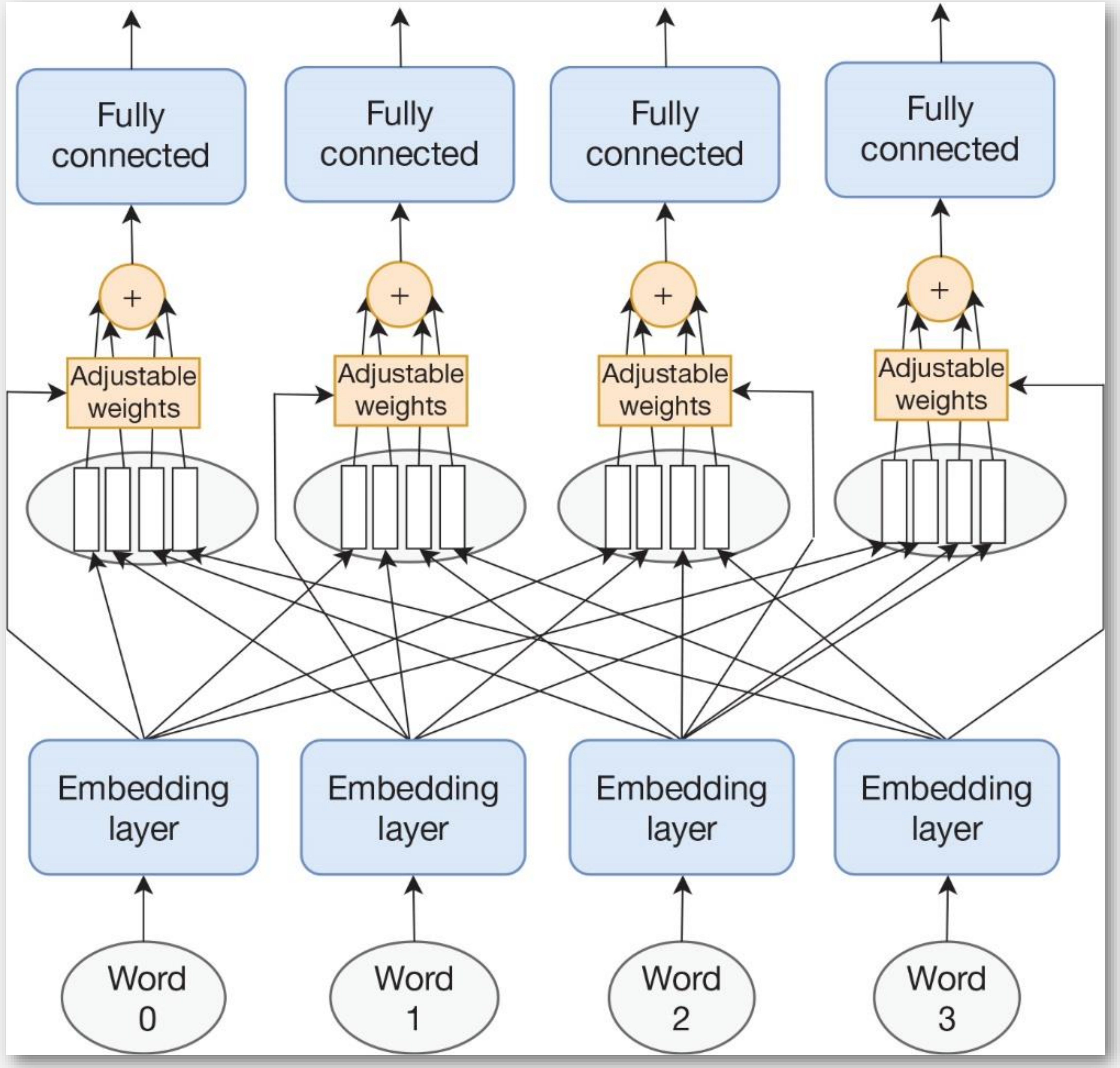
# THE TRANSFORMER

- Encoder/decoder network with attention
- Encoder and decoder additionally use multi-headed self-attention
- No recurrent layers
- Decoder uses autoregression

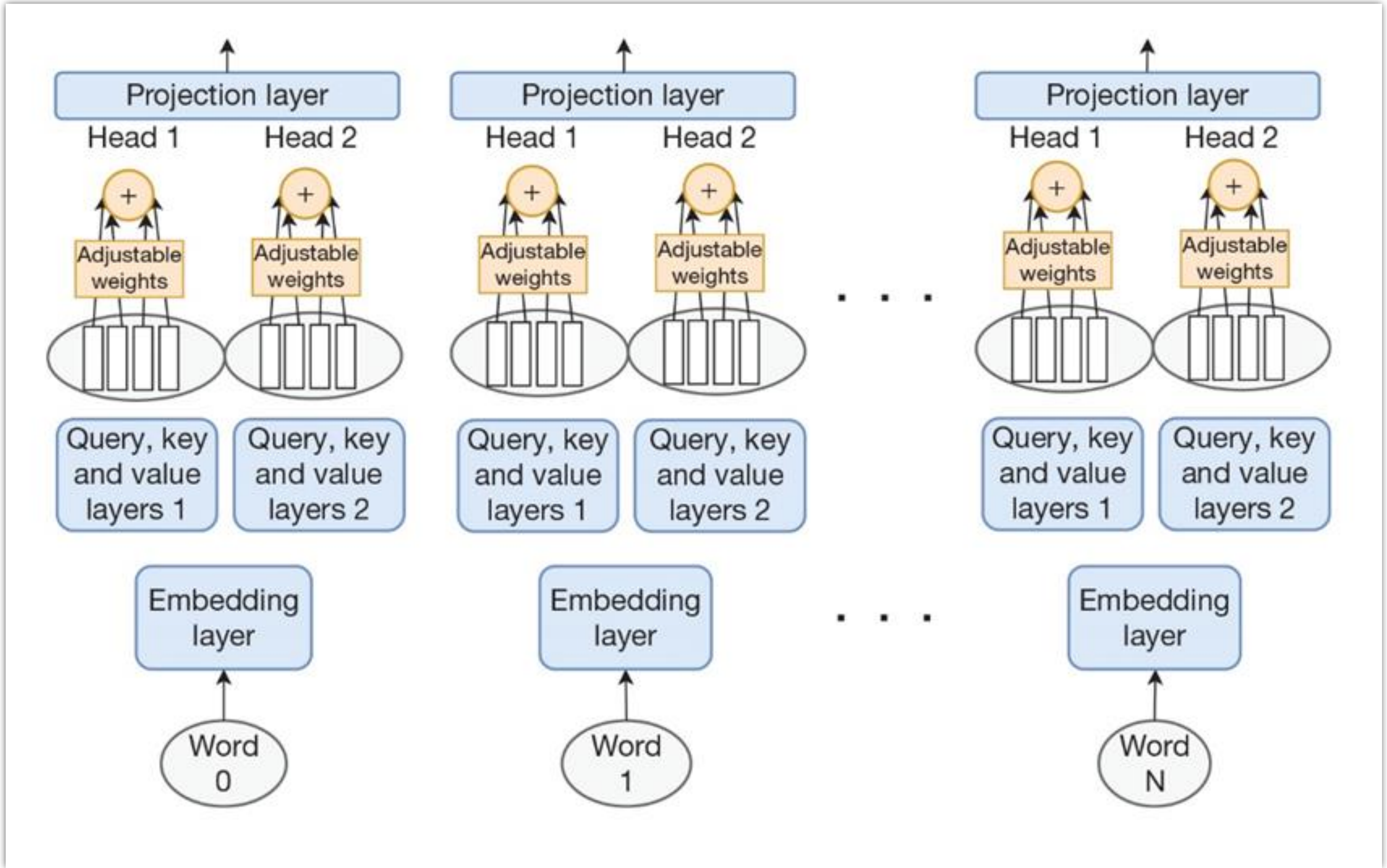


## SELF-ATTENTION

- In traditional attention the decoder attends to output of the encoder
- In self-attention a layer attends to the output of the preceding layer
- An important difference is what data is used to compute the weights

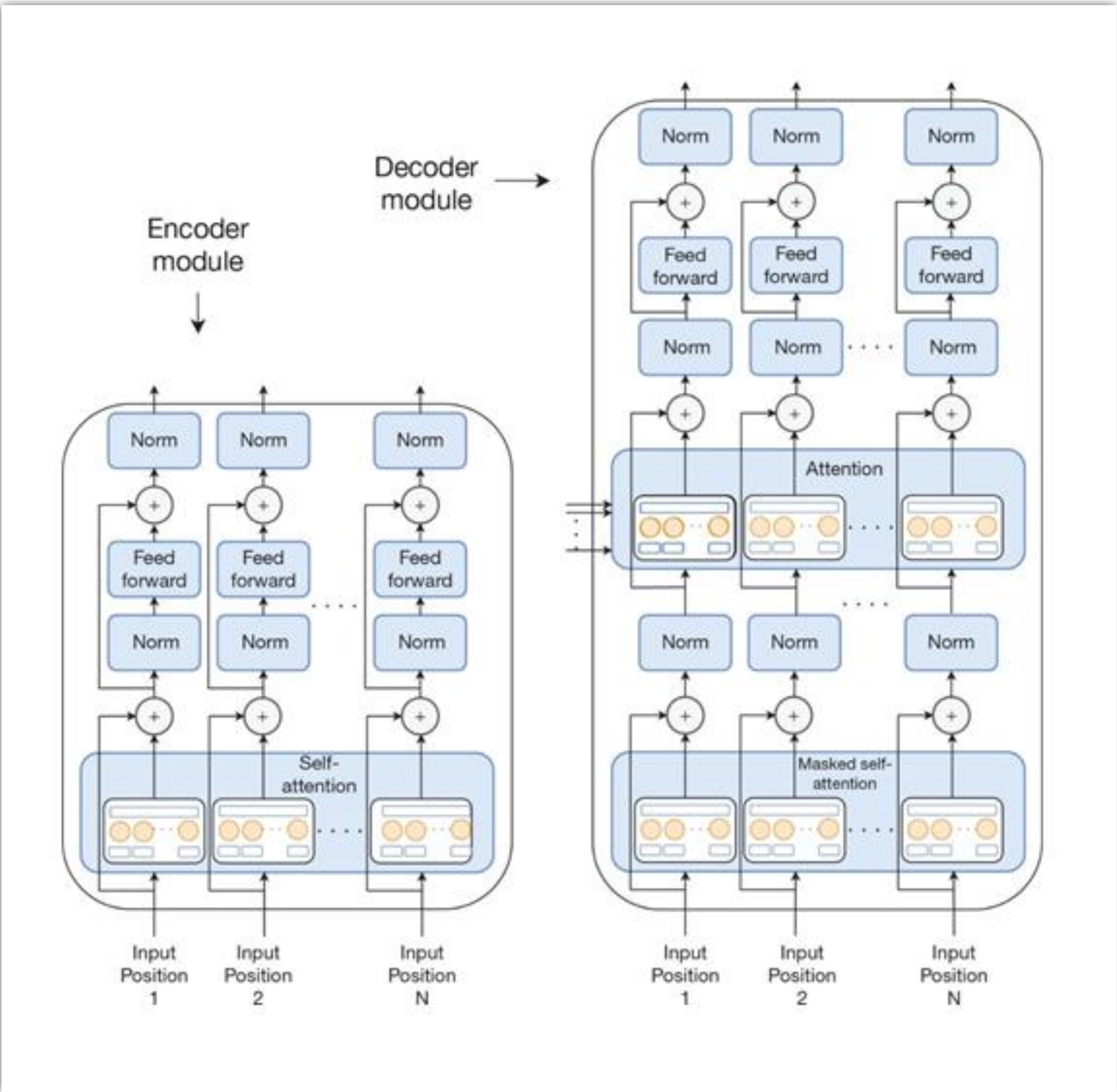


# MULTI-HEADED SELF-ATTENTION

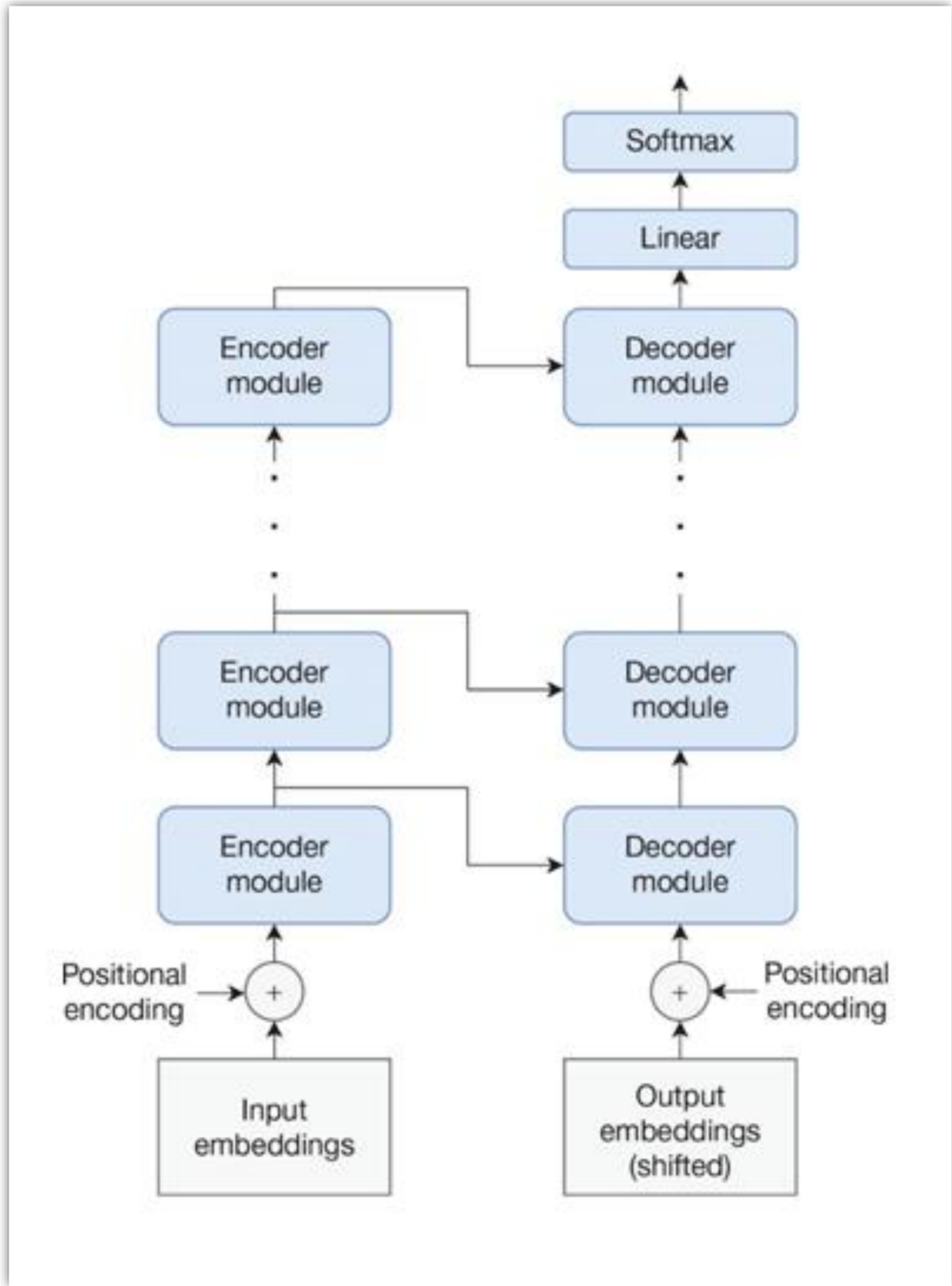



NVIDIA

# PUTTING IT ALL TOGETHER




Building blocks




The Transformer

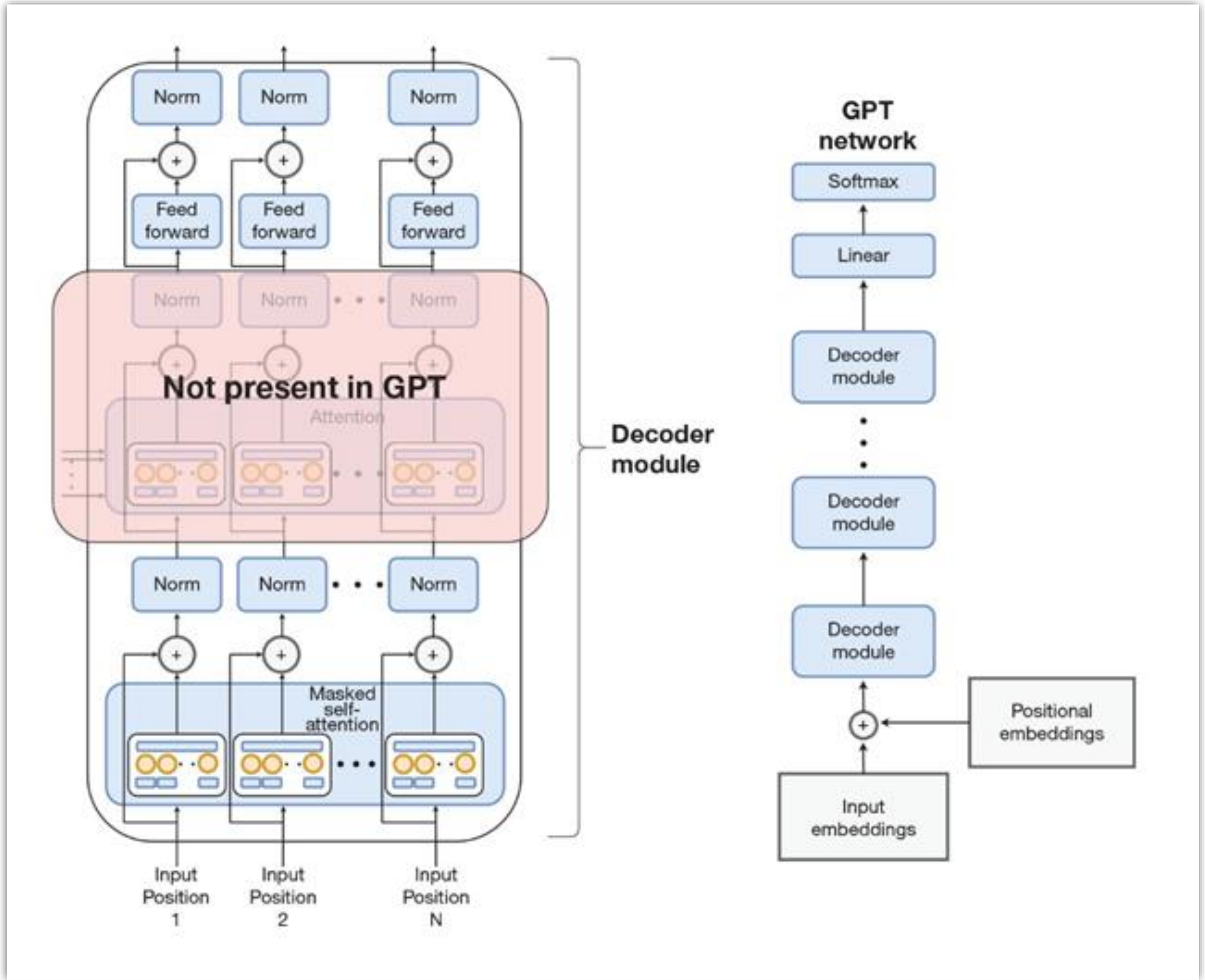
# GPT

Generative Pretraining with Transformers

Model based on Transformer decoder

Pre-trained as language model

Fine-tuned for other NLP tasks in end application

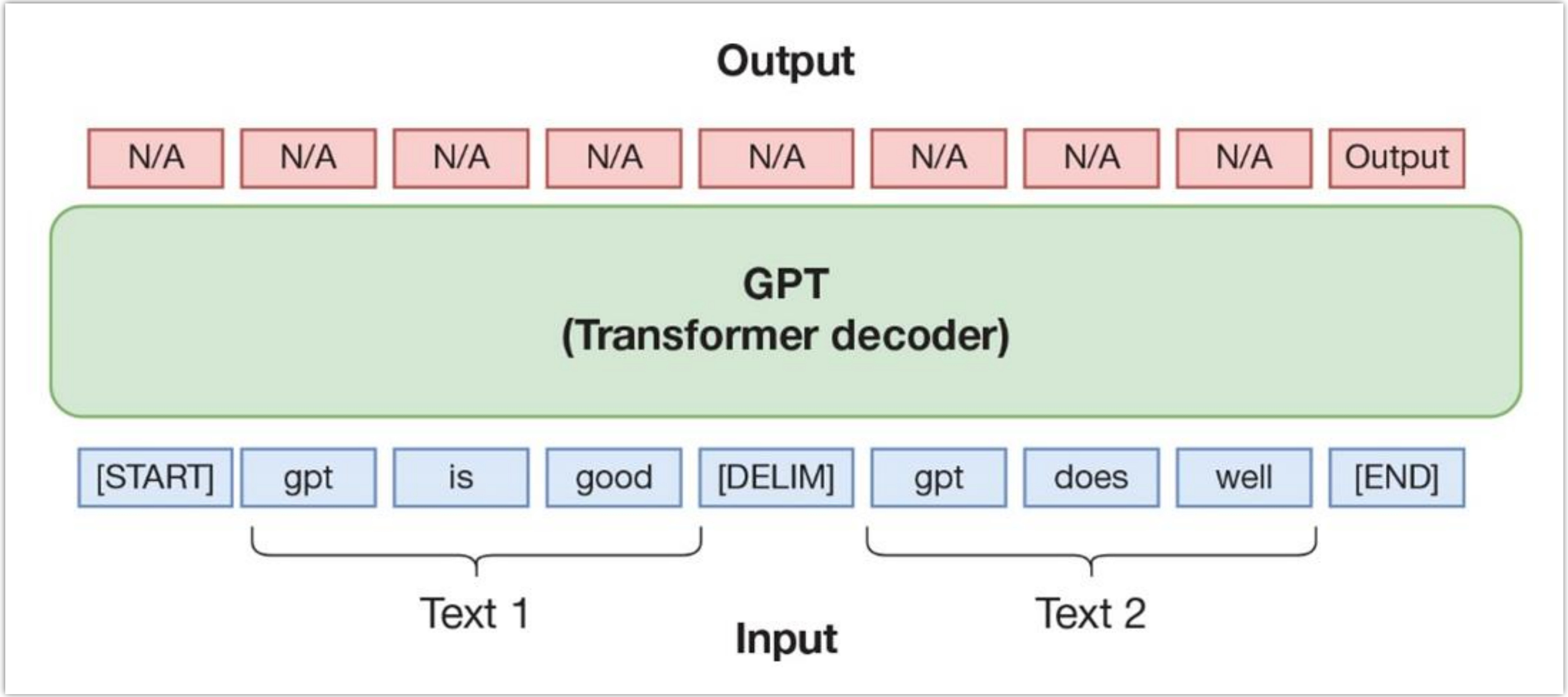


# GPT

Pre-training

# GPT

Fine-tuning for similarity task

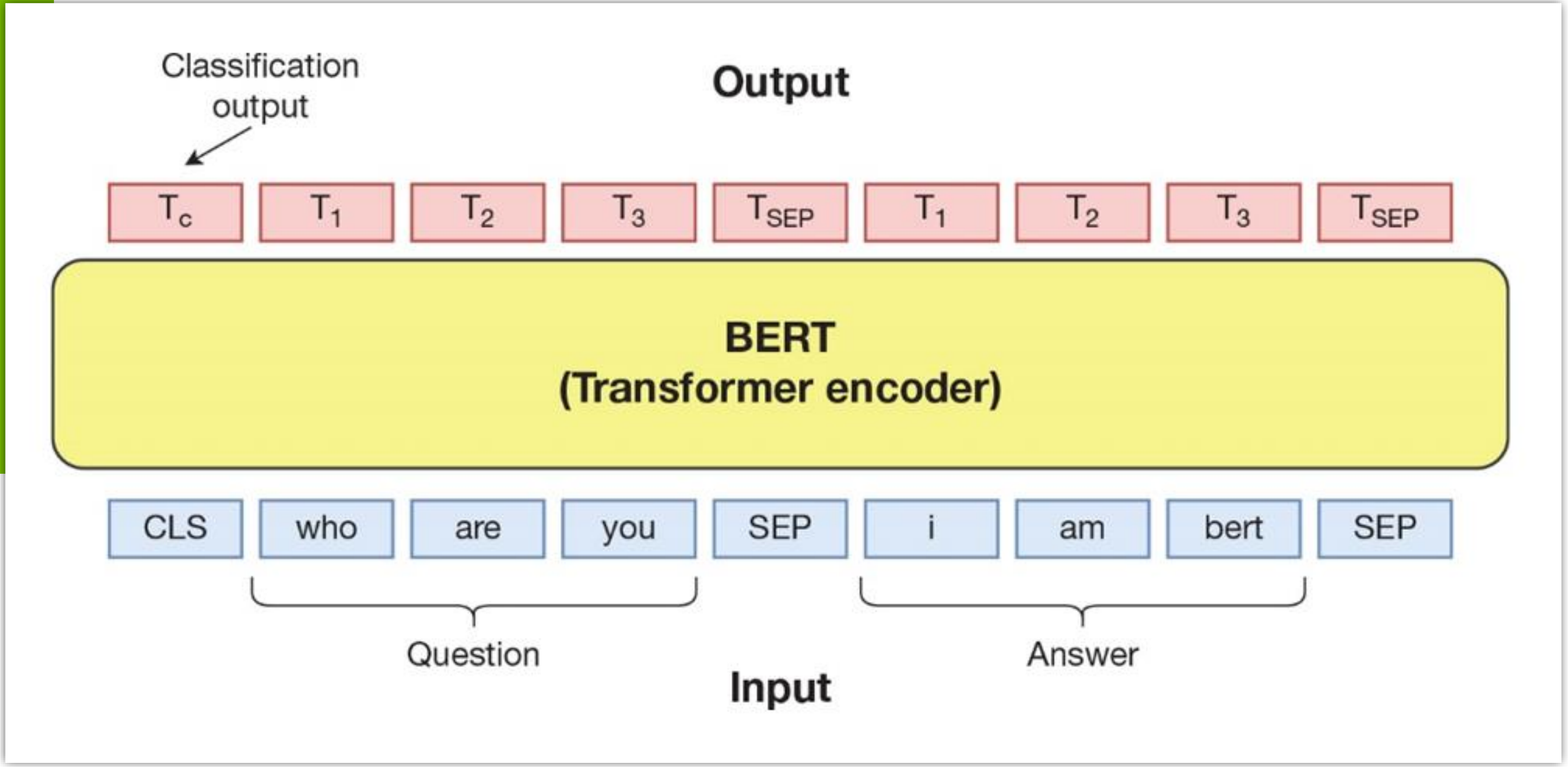


# BERT

Bidirectional Encoder Representations from Transformers

Model based on Transformer encoder

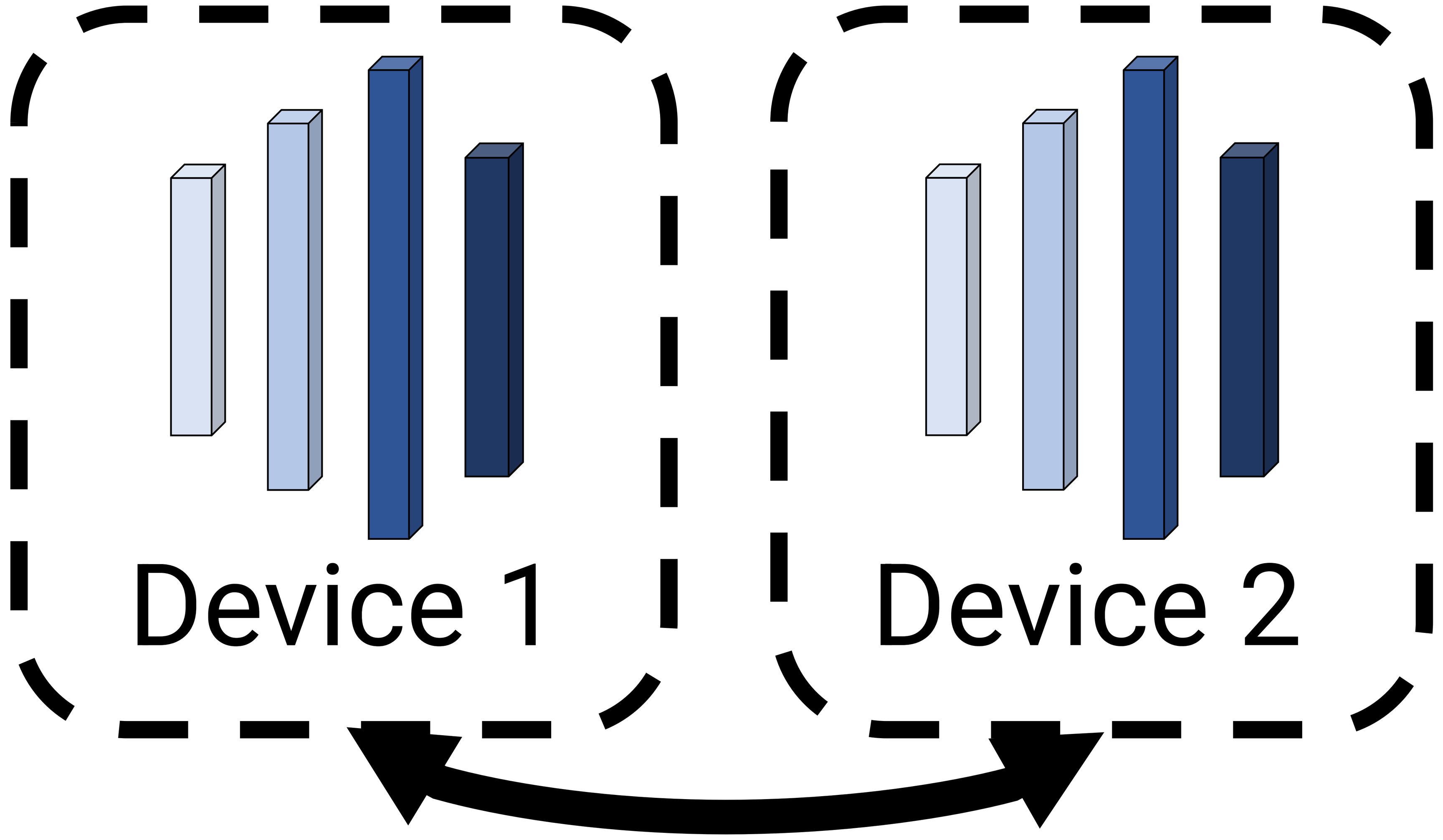
Pre-trained as masked language model and next-sentence prediction



# SCALING TRANSFORMER MODELS WITH MEGATRON

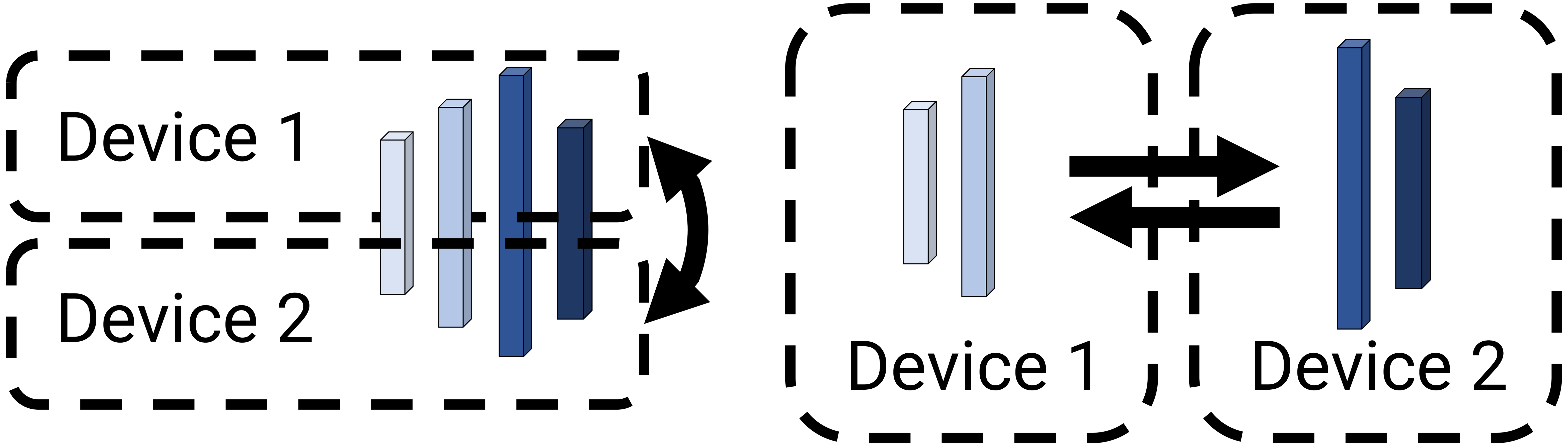
# PARALLEL TRAINING

## Data Parallelism (DP)




$n$ copies of model parameters

## Model Parallelism (MP)




Tensor MP

Pipeline MP

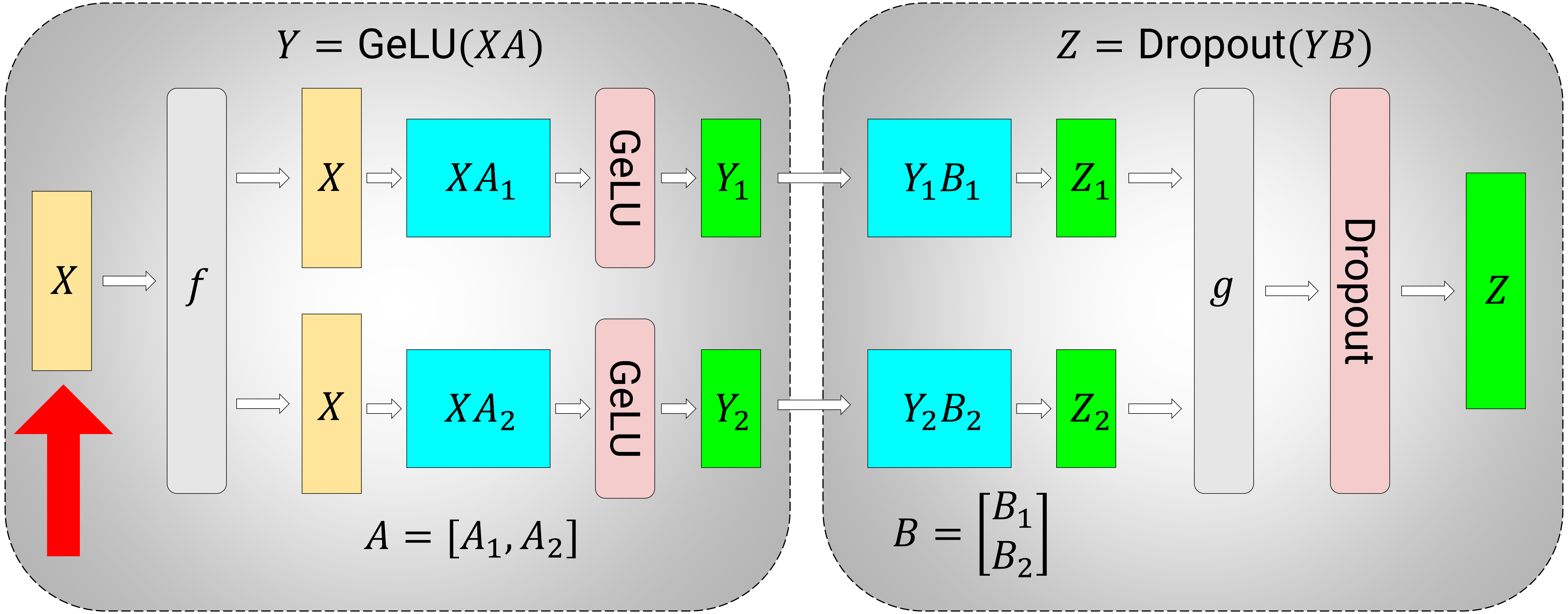
Single copy of model parameters

# DATA PARALLELISM


- All-reductions of weight gradients after every iteration
- Model size limited by device memory – Worked through GPT-2 ~2 years ago
  - Cannot be used in isolation for large models
- Parallelism limited by batch size
  - Too much data at a time leads to inefficient steps

# TENSOR MODEL PARALLELISM

Each layer of model is partitioned over multiple devices




$g \rightarrow$ All-reduction $(Y_1B_1 + Y_2B_2)$ in forward pass

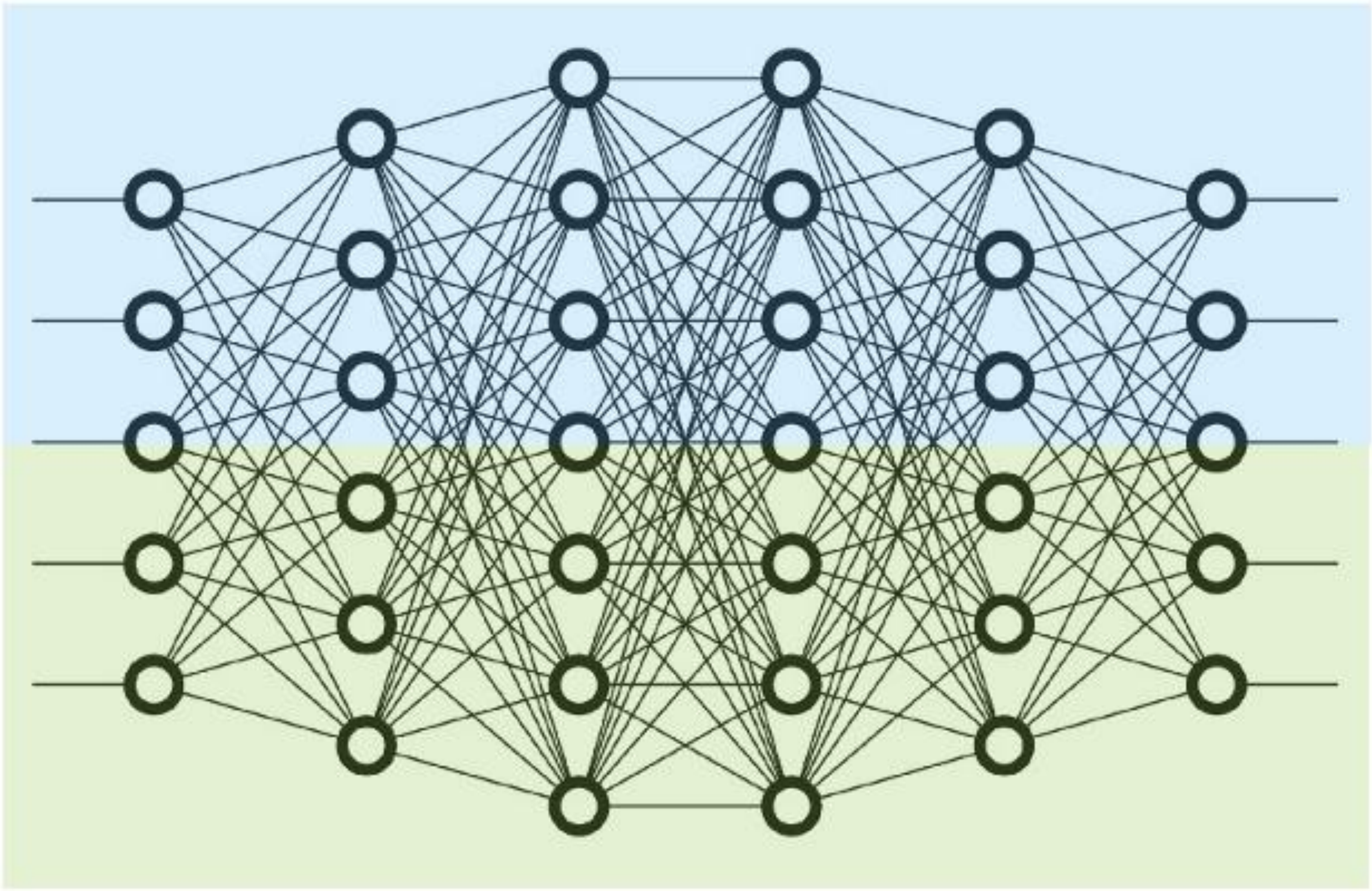
**Slow across inter-server communication links**

[1] Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism, arXiv:1909.08053, Shoeybi et al.

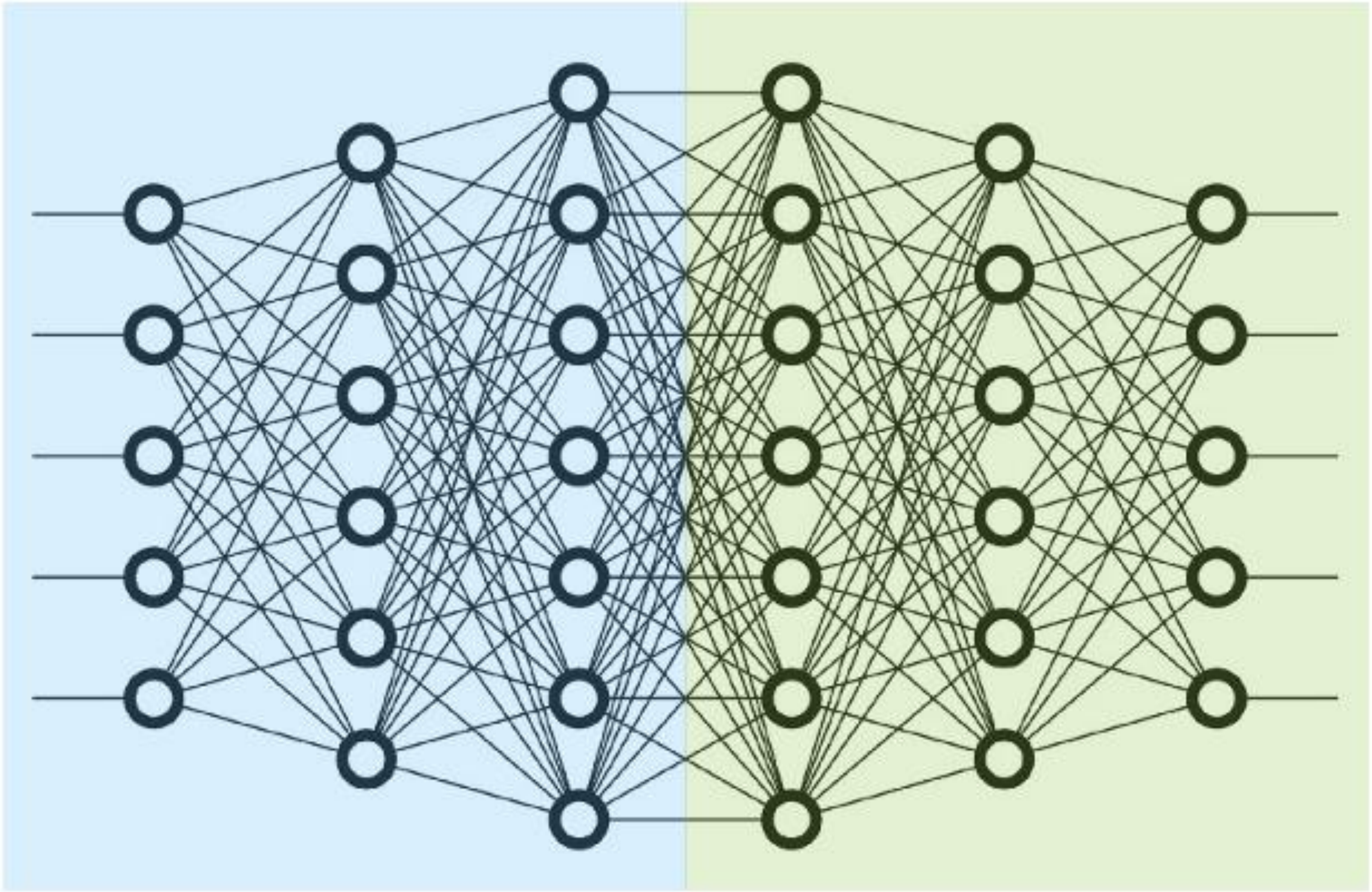
# MODEL PARALLELISM

Pipeline (Inter-Layer) Parallelism

- Tensor (Intra-Layer) Parallelism
  - Split individual layers across multiple devices
  - Both devices compute difference parts of Layer 0,1,2,3,4,5

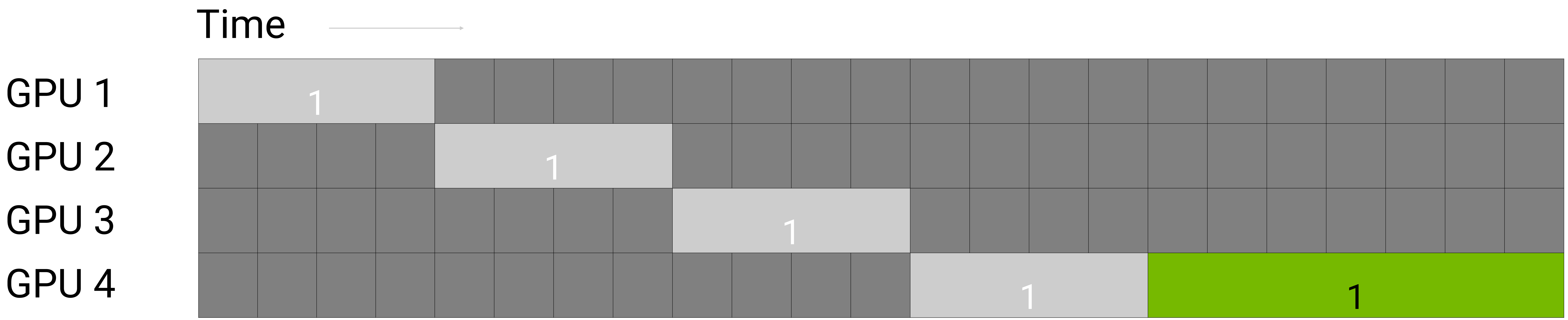


- Pipeline (Inter-Layer) Parallelism
  - Split sets of layers across multiple devices
  - Layer 0,1,2 and layer 3,4,5 are on difference devices



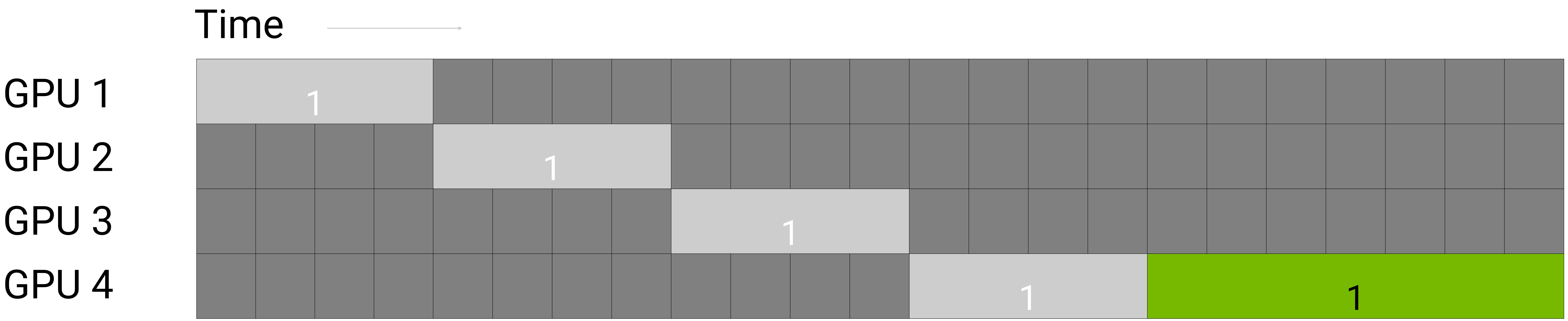
NVIDIA

# PIPELINING

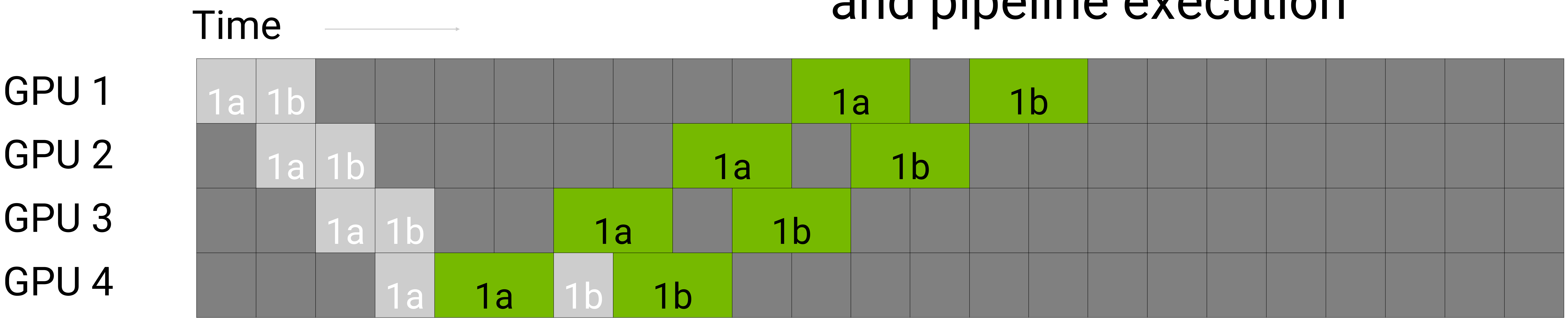



Split batch into microbatches and pipeline execution

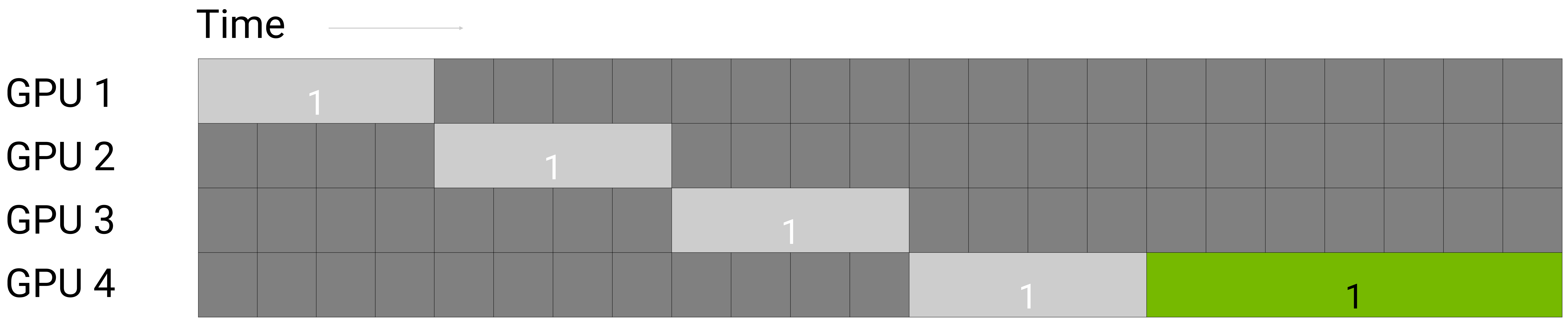
# PIPELINING



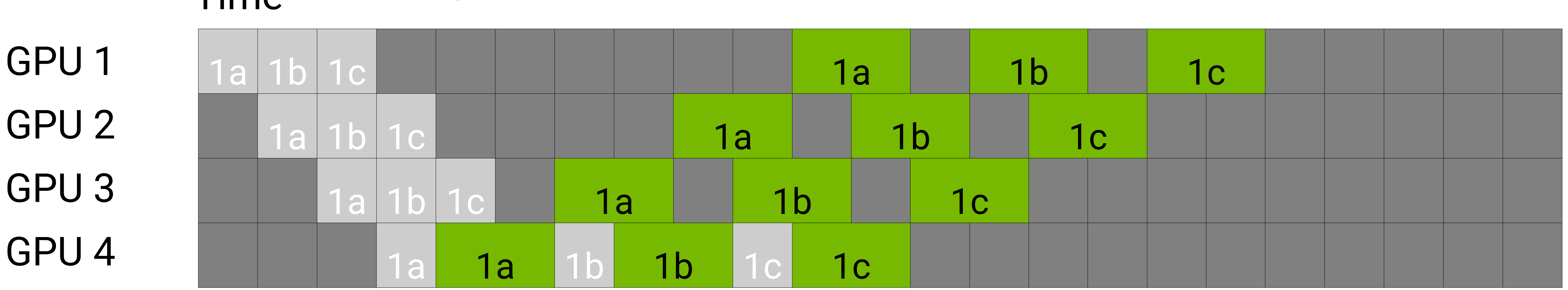

Split batch into microbatches and pipeline execution



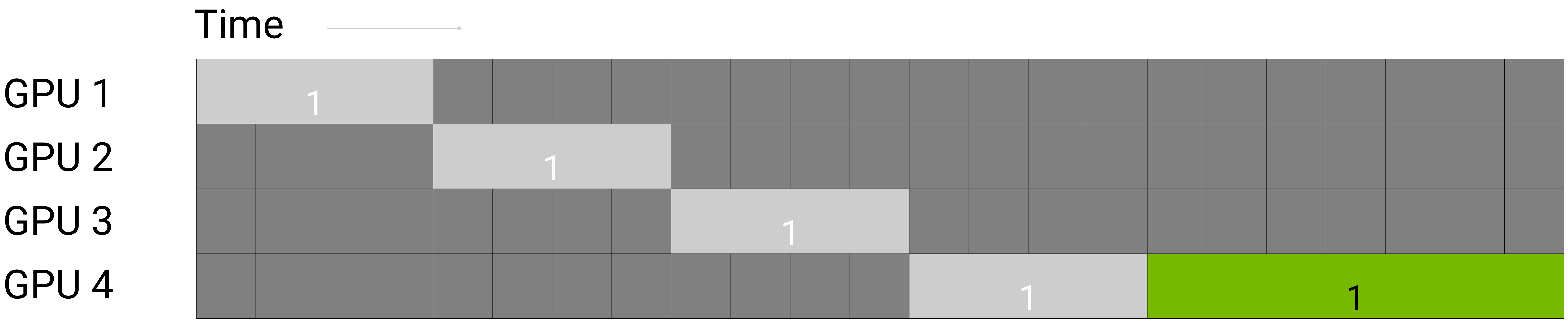
# PIPELINING



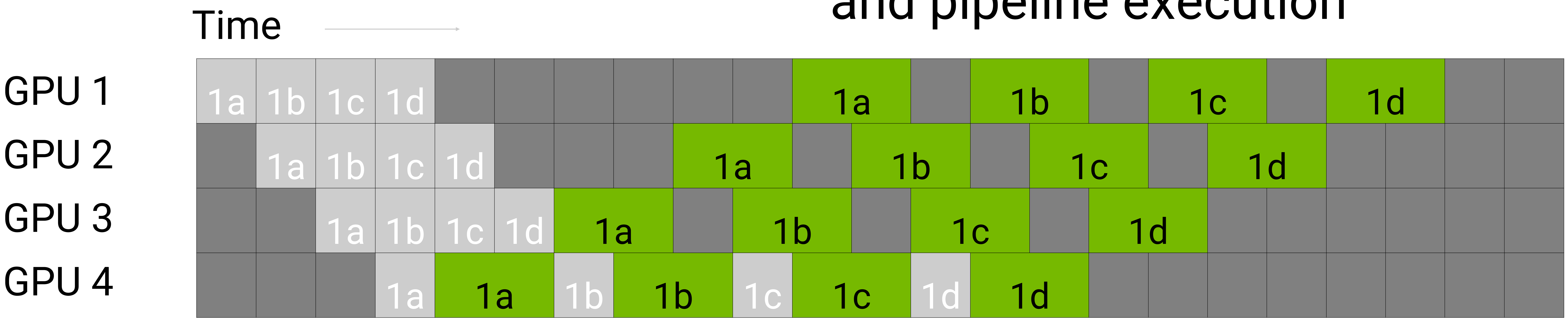

Split batch into microbatches and pipeline execution



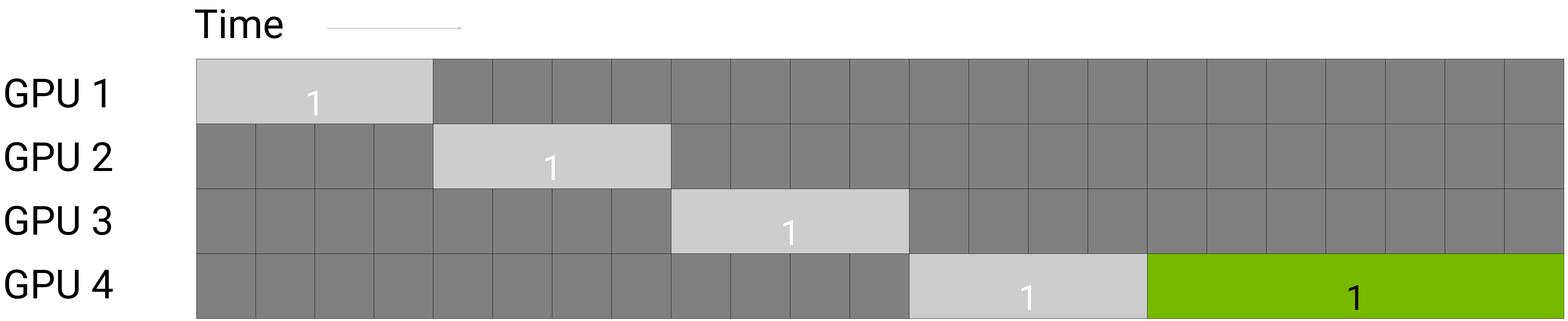
# PIPELINING



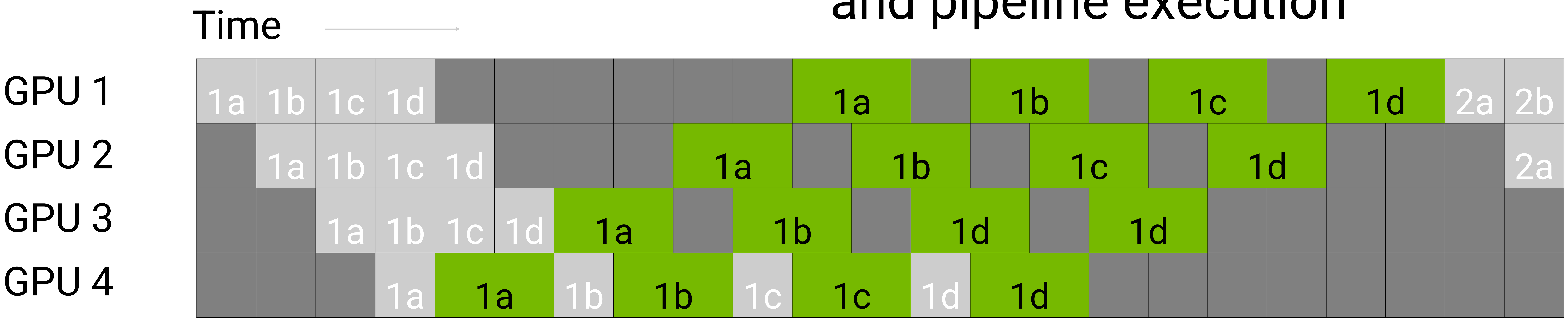

Split batch into microbatches and pipeline execution



# PIPELINING




Split batch into microbatches and pipeline execution



# PIPELINE BUBBLES


$p$ : number of pipeline stages

$m$ : number of micro batches

$t_f$ : forward step time

$t_b$ : backward step time

$$\text{total time} = (m + p - 1) \times (t_f + t_b)$$

$$\text{ideal time} = m \times (t_f + t_b)$$

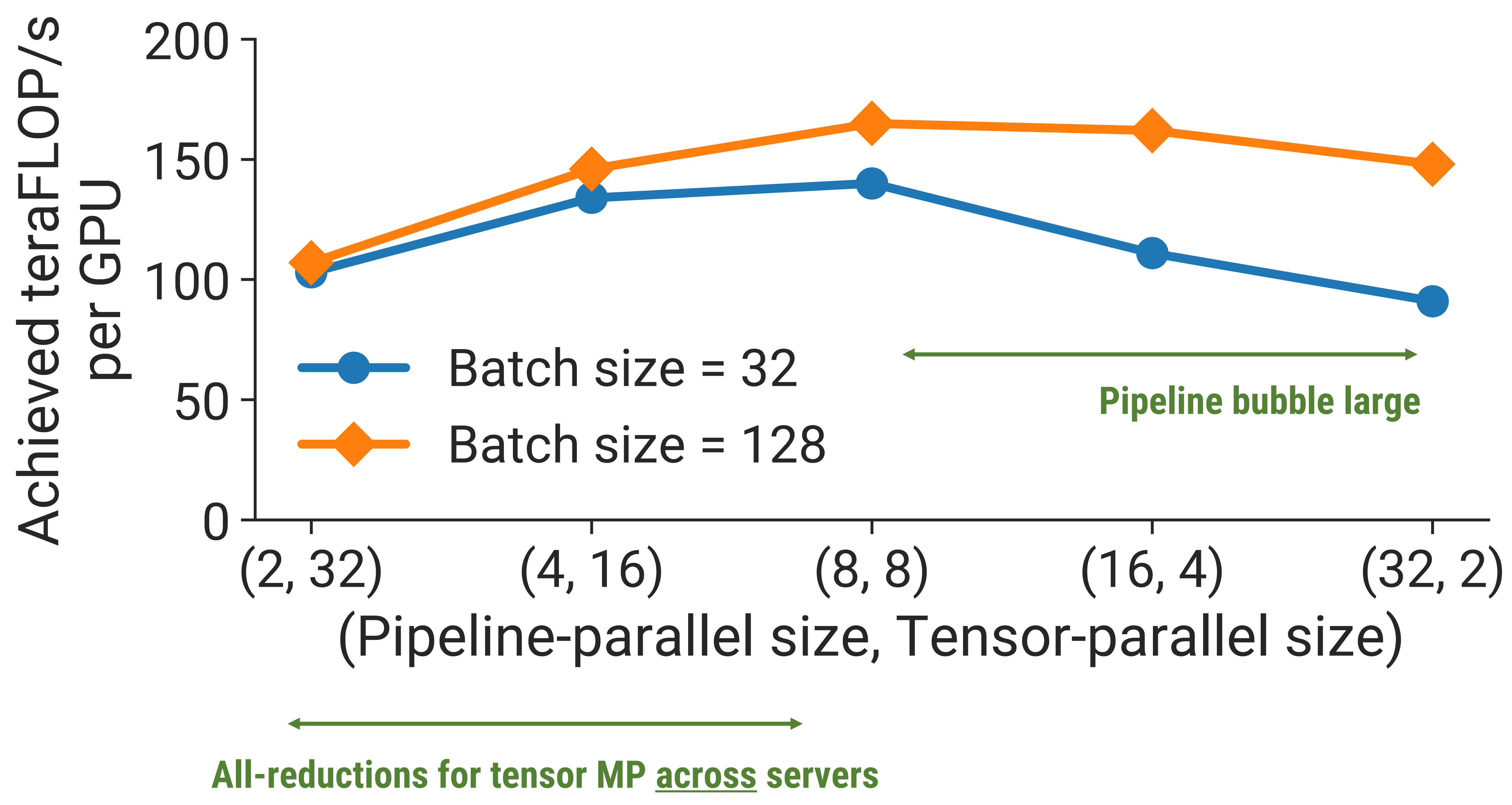
$$\text{bubble time} = (p - 1) \times (t_f + t_b)$$

$$\text{bubble time overhead} = \frac{\text{bubble time}}{\text{ideal time}} = \frac{p-1}{m}$$

# DIFFERENT SCHEMES HAVE DIFFERENT TRADEOFFS

- Naïvely combining parallelism dimensions leads to poor throughput
- Using parallelism efficiently thus requires one to reason through the interactions between different parallelism modes
- Each parallelism mode makes tradeoffs, and determining the optimal degrees of parallelism requires reasoning through these tradeoffs

162B GPT model
64 80-GB A100 GPUs



# MORE EFFICIENT PIPELINE SCHEDULES

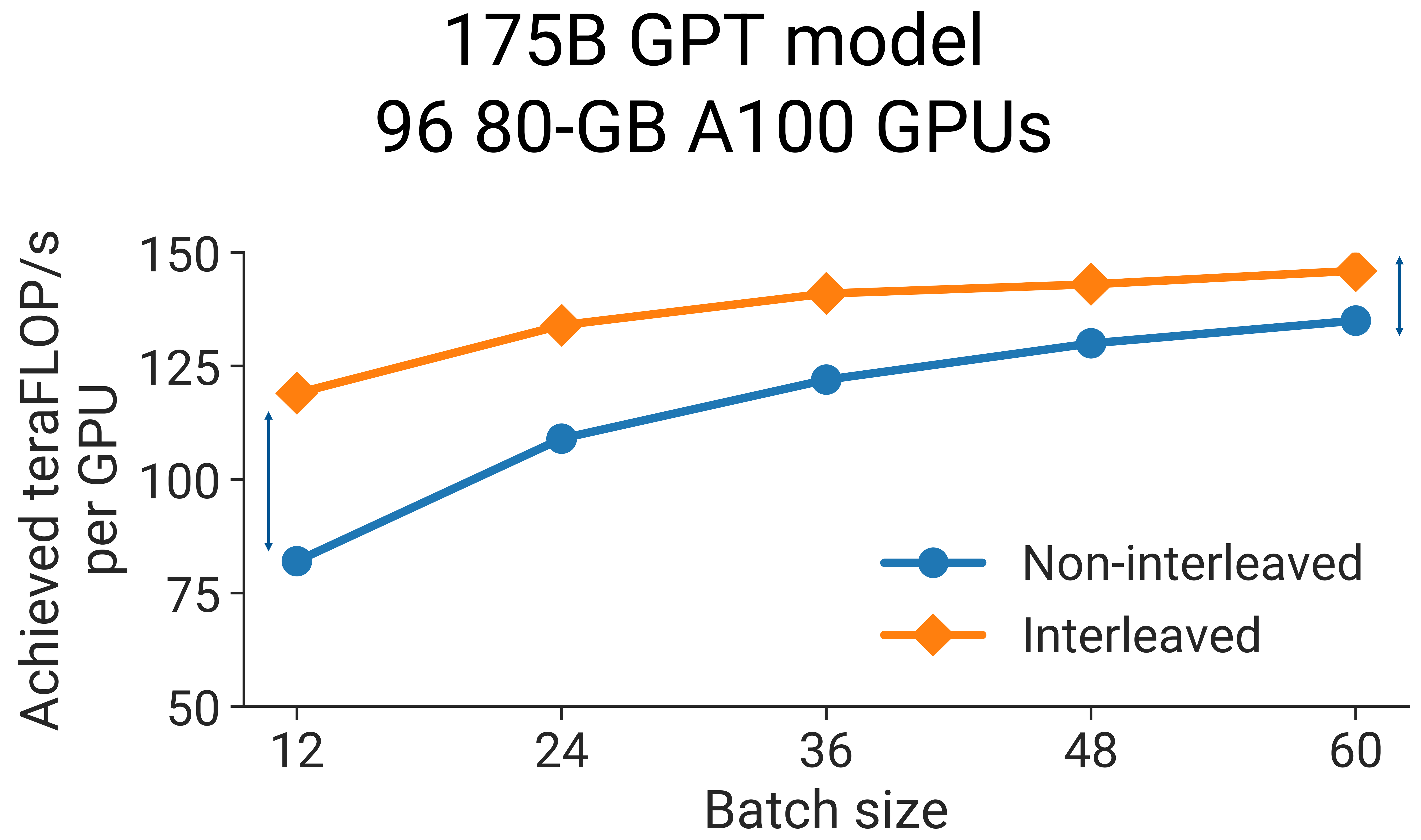

Assign multiple stages to each device (interleaved schedule)


Forward Pass

Backward Pass

NVIDIA

# MORE EFFICIENT PIPELINE SCHEDULES




**Large throughput increases at small batch sizes, smaller at large batch sizes**

# HOW DO WE NAVIGATE THIS CONFIGURATION SPACE?

Degree of pipeline, tensor, and data parallelism

Pipelining schedule

Global batch size

Microbatch size

Each of these influence amount of communication, size of pipeline bubble, memory footprint

See our recent paper Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM

NVIDIA

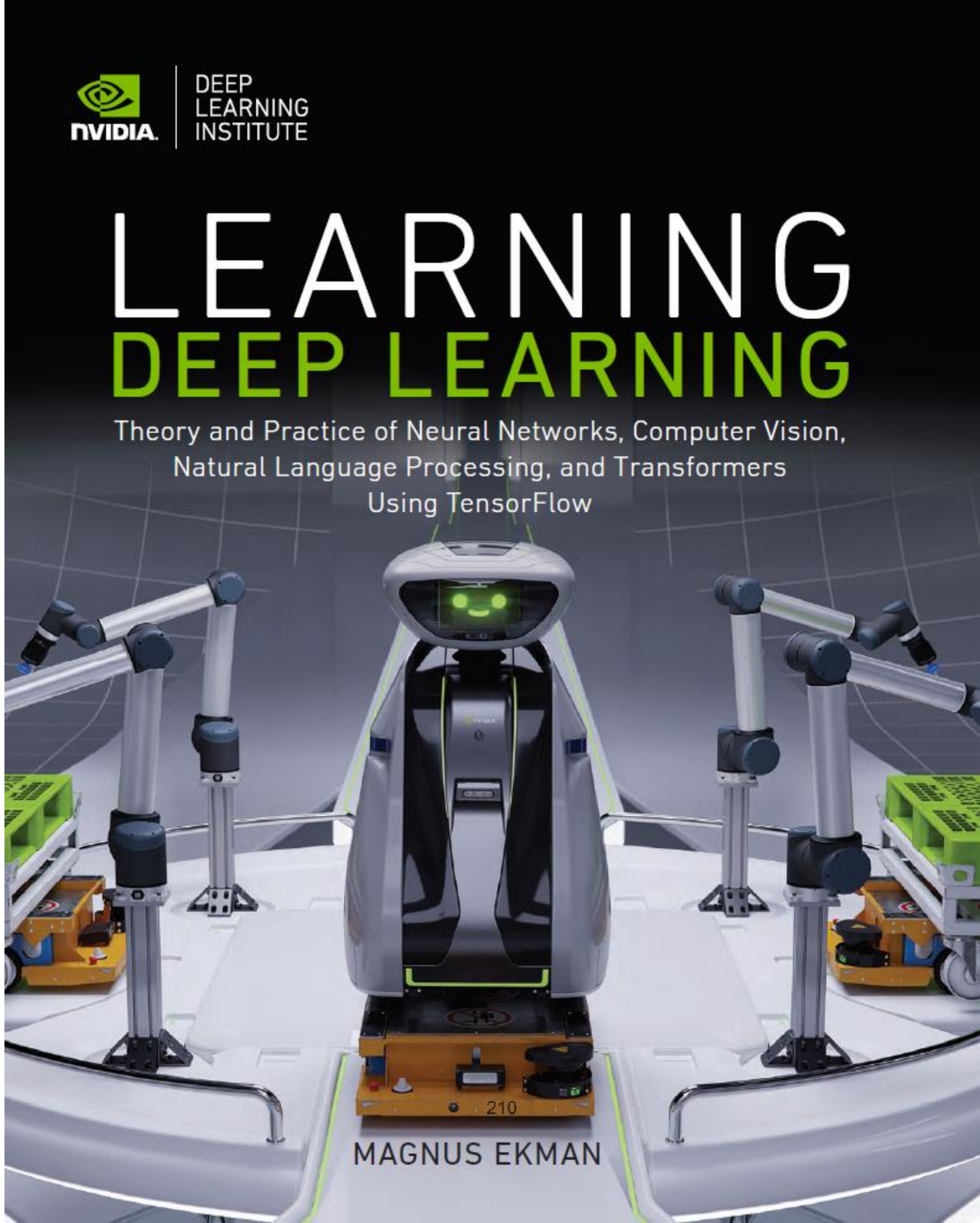
# IMPLEMENTATION: MEGATRON-LM

- Built on top of PyTorch
- Supports various transformer models like GPT and BERT
- Good performance on smaller models (e.g., BERT-Base and BERT-Large) and smaller scales as well

**Implementation available at https://github.com/nvidia/megatron-lm**

## RESOURCES FOR A DEEP DIVE

- https://nvidia.com/en-us/training/books/
- https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/Translation/Transformer
- https://github.com/NVIDIA/Megatron-LM




NVIDIA