May 5th, 2026
Ay 119

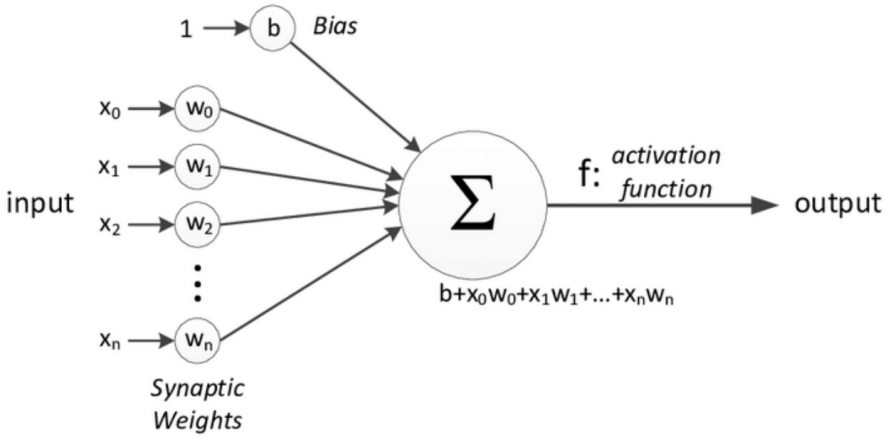
Advanced Deep Learning:

A Brief View to Transformers

Matthew J. Graham
and Pavlos Protopapas

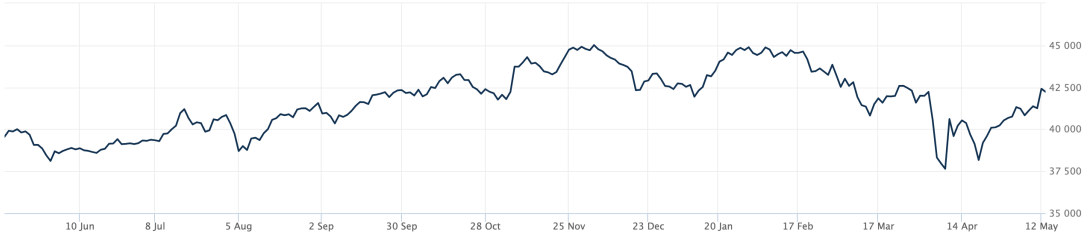
Deep learning: recap from last seminar

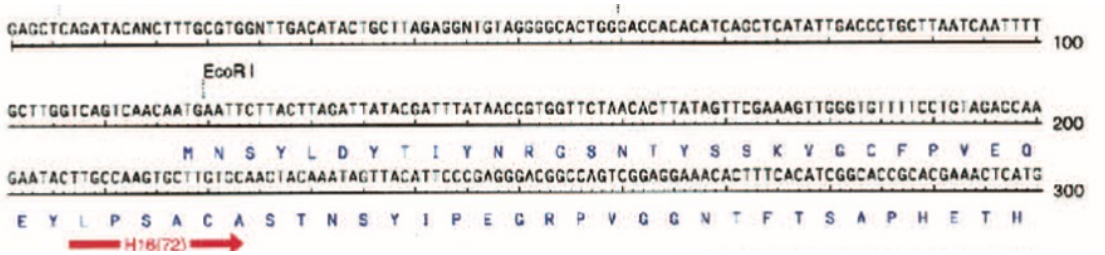
- The basic neuron: $y = f(\boldsymbol{w} \cdot \boldsymbol{x} + b)$
- Choice of activation function
- MLP or feedforward NN trained by backpropagation
- Loss functions for different purposes
- Regularization to prevent overfitting
- Optimizers
- Different types of neural architecture:
  - Convolutional Neural Networks
  - Recurrent Neural Networks
  - Autoencoders
  - GANs

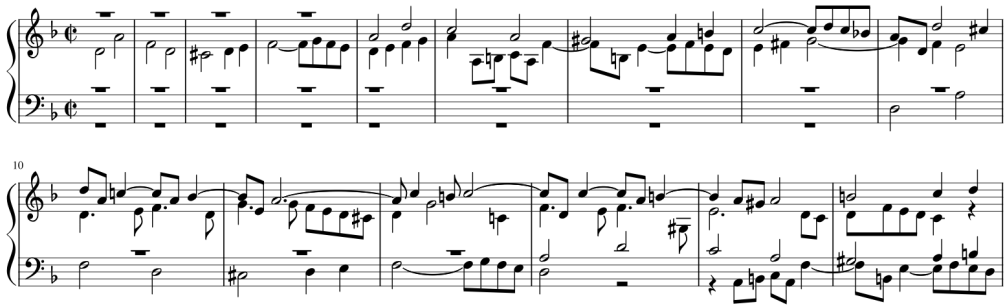


Sequential data is everywhere

[p.004] What clashes here of wills gen wonts, oystrygods gaggin fishygods! Brékkek Kékkek Kékkek Kékkek! Kóax Kóax Kóax! Ualu Ualu Ualu! Quaouauh! Where the Baddelaries partisans are still out to mathmaster Malachus Micgranes and the Verdons catapelting the camibalistics out of the Whoyteboyce of Hoodie Head. Assiegates and boomeringstroms. Sod's brood, be me fear! Sanglorians, save! Arms apeal with larms, appalling. Killykillkilly: a toll, a toll. What chance cuddleys, what cashels aired and ventilated! What bidimetoloves sinduced by what tegotetabsolvers! What true feeling for their's hayair with what strawng voice of false jiccup! O here here how

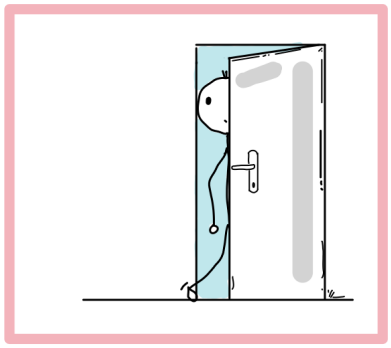






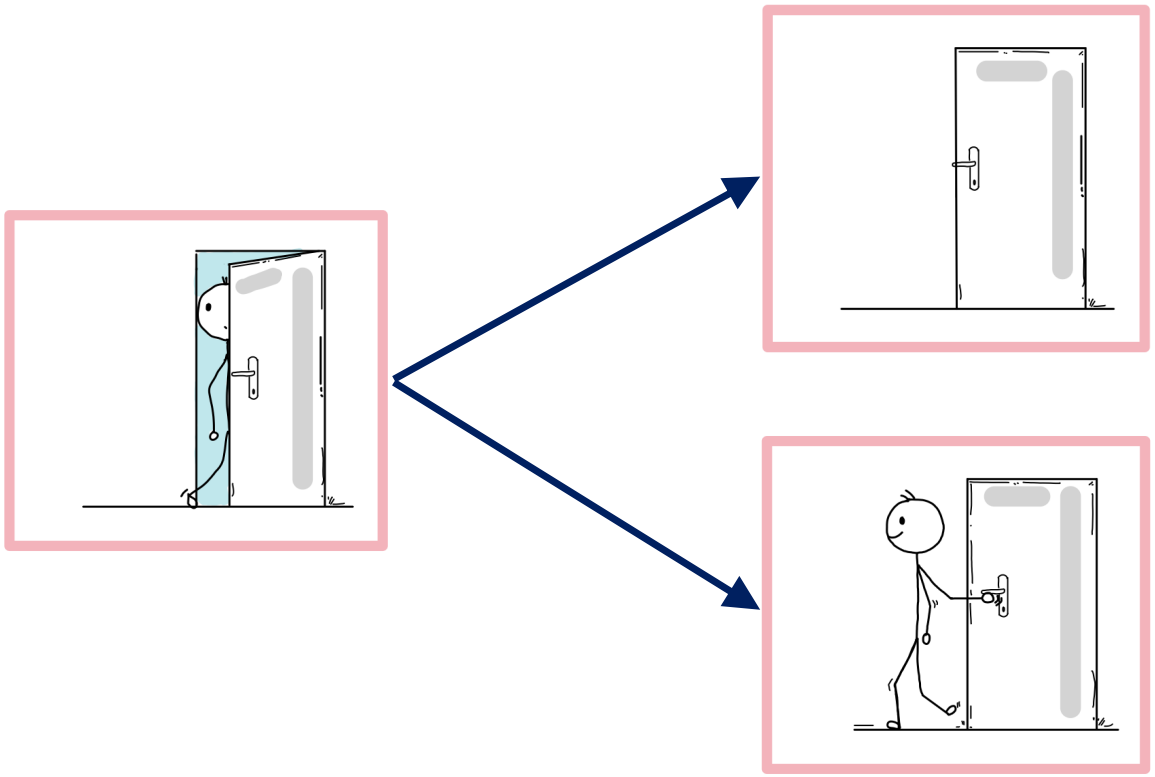
Let’s think about sequences

Given this frame, what do you think is the next likely frame?



Let's think about sequences

Given this frame, what do you think is the next likely frame?



The person might be going in

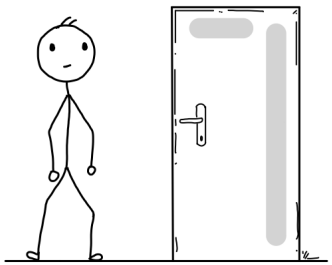
The person might be coming out

Based on only the first frame, it would be difficult to predict the next frame.

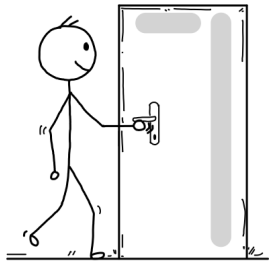
Let's think about sequences

However, if we give the model the previous frames, it is easy to predict the next:

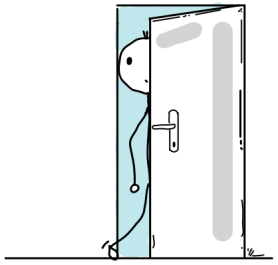
Frame 1



Frame 2



Frame 3



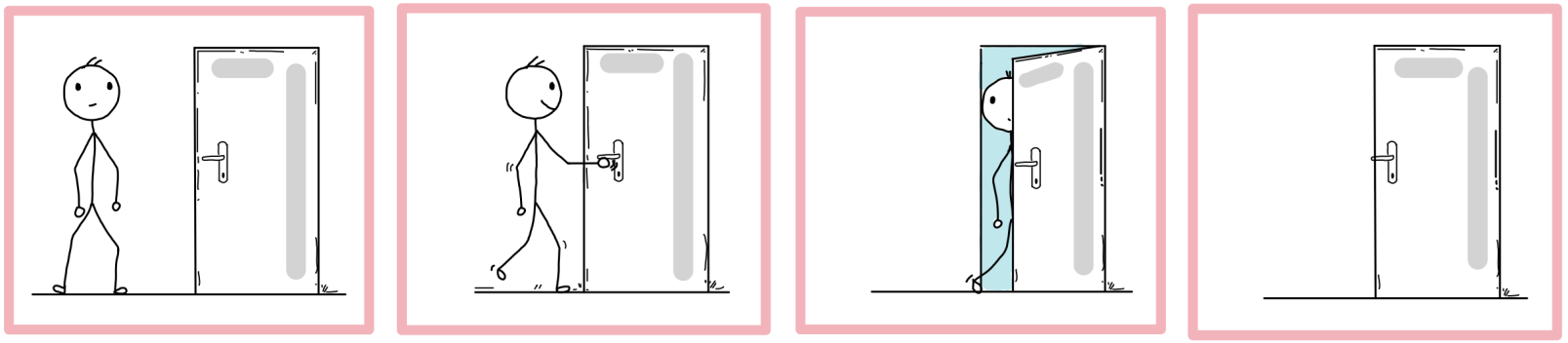
Frame 4

?

Let's think about sequences

However, if we give the model the previous frames, it is easy to predict the next:

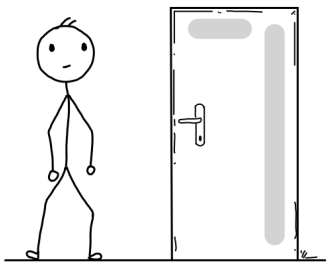
Frame 1 Frame 2 Frame 3 Frame 4



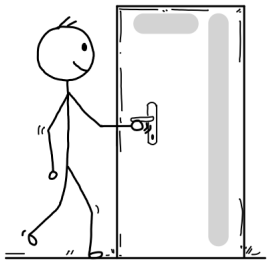
Let’s think about sequences

However, if we give the model the previous frames, it is easy to predict the next:

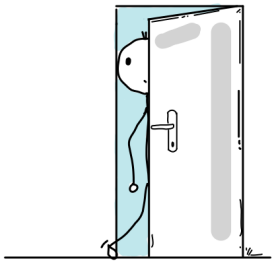
Frame 1



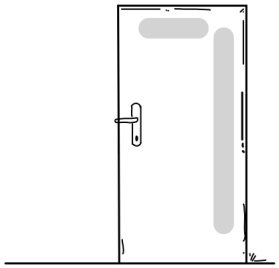
Frame 2



Frame 3



Frame 4



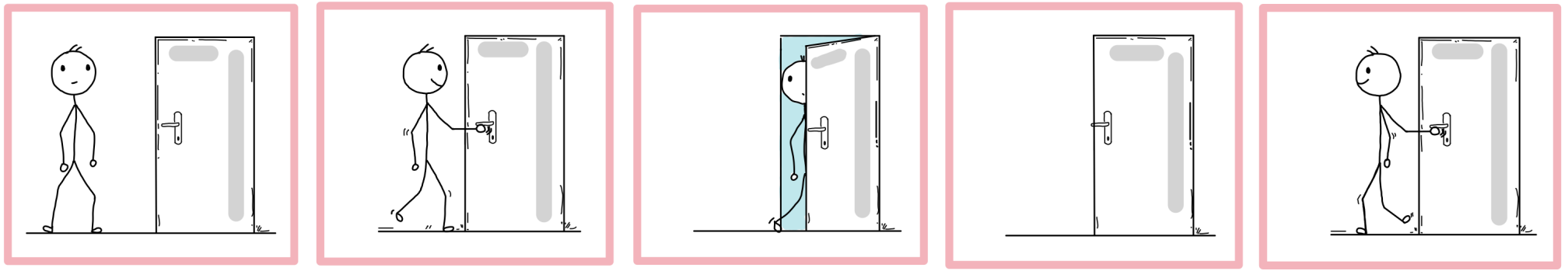
Frame 5

?

Let's think about sequences

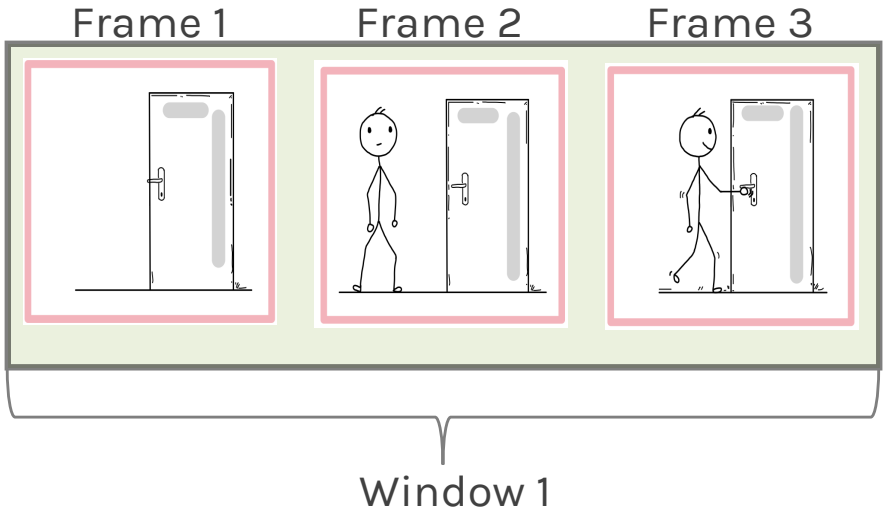
However, if we give the model the previous frames, it is easy to predict the next:

Frame 1 | Frame 2 | Frame 3 | Frame 4 | Frame 5



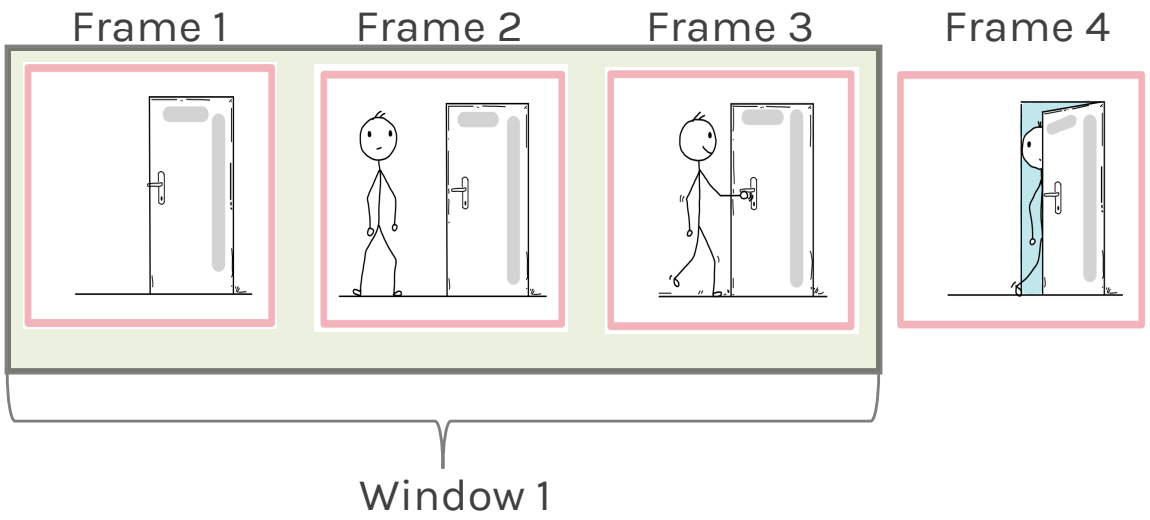
Sequences play an important role for forecasting and predictions.

Let's think about sequences



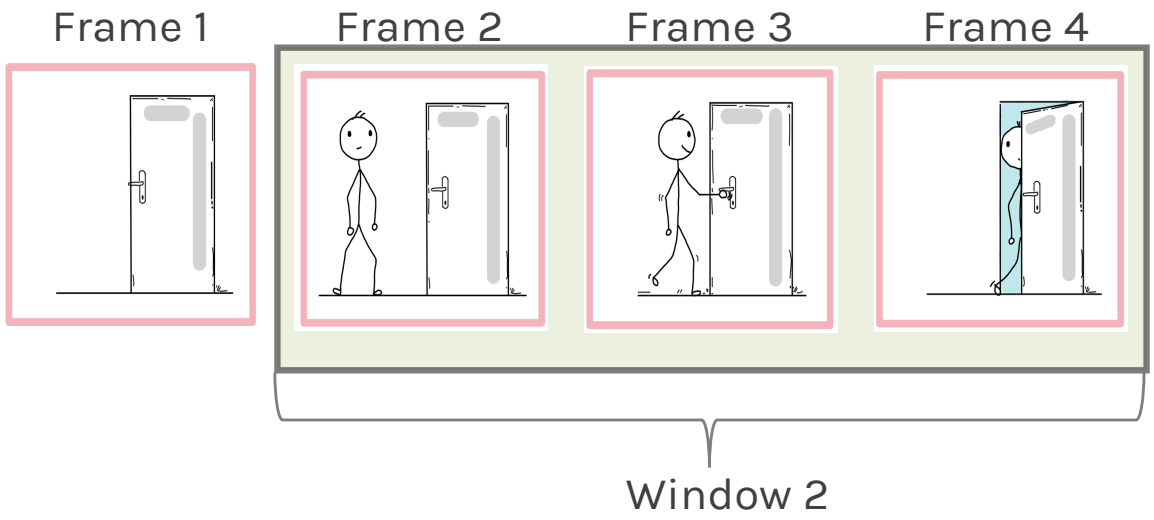
If we window a fixed number of frames as input to a NN like MLP or CNN then prediction will work

Let's think about sequences



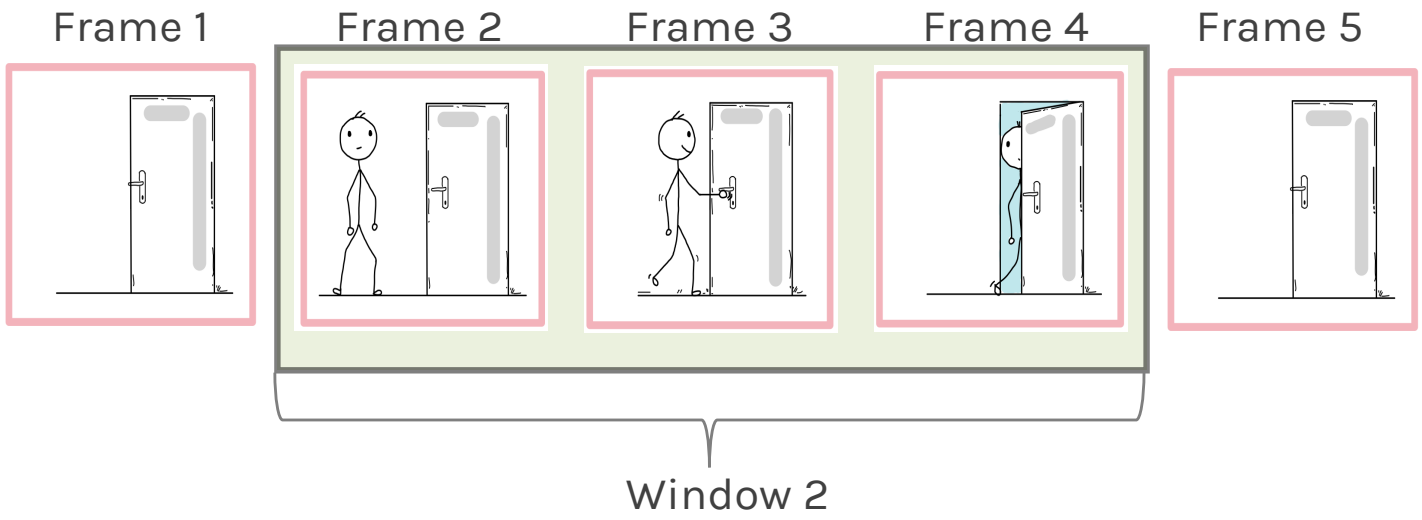
If we window a fixed number of frames as input to a NN like MLP or CNN then prediction will work

Let's think about sequences



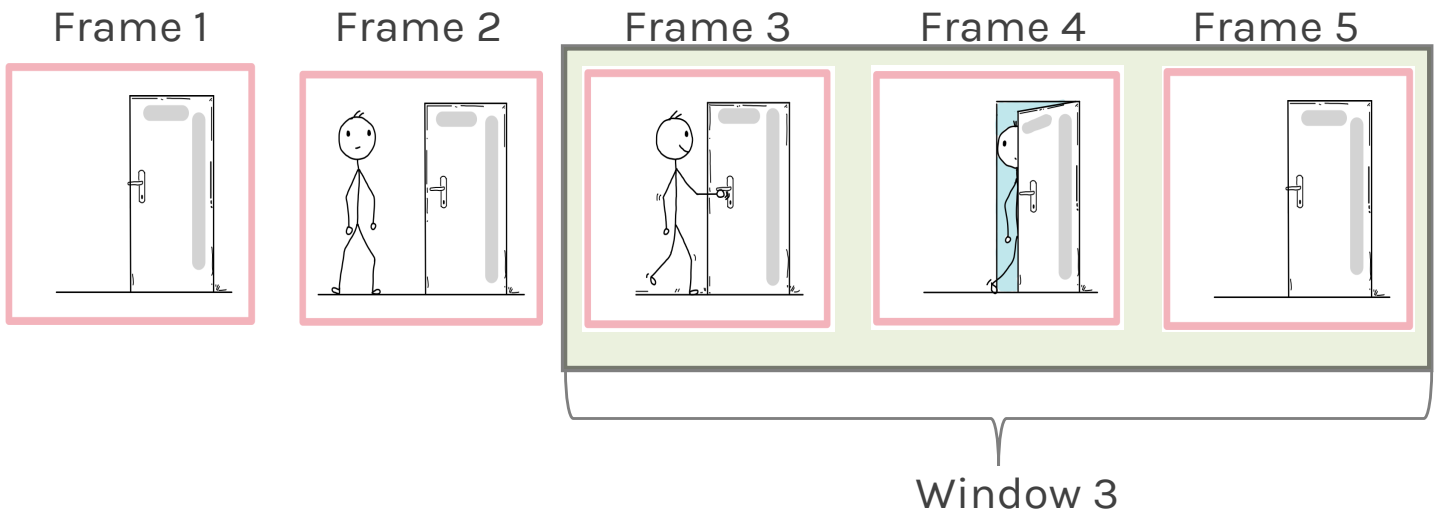
If we window a fixed number of frames as input to a NN like MLP or CNN then prediction will work

Let's think about sequences



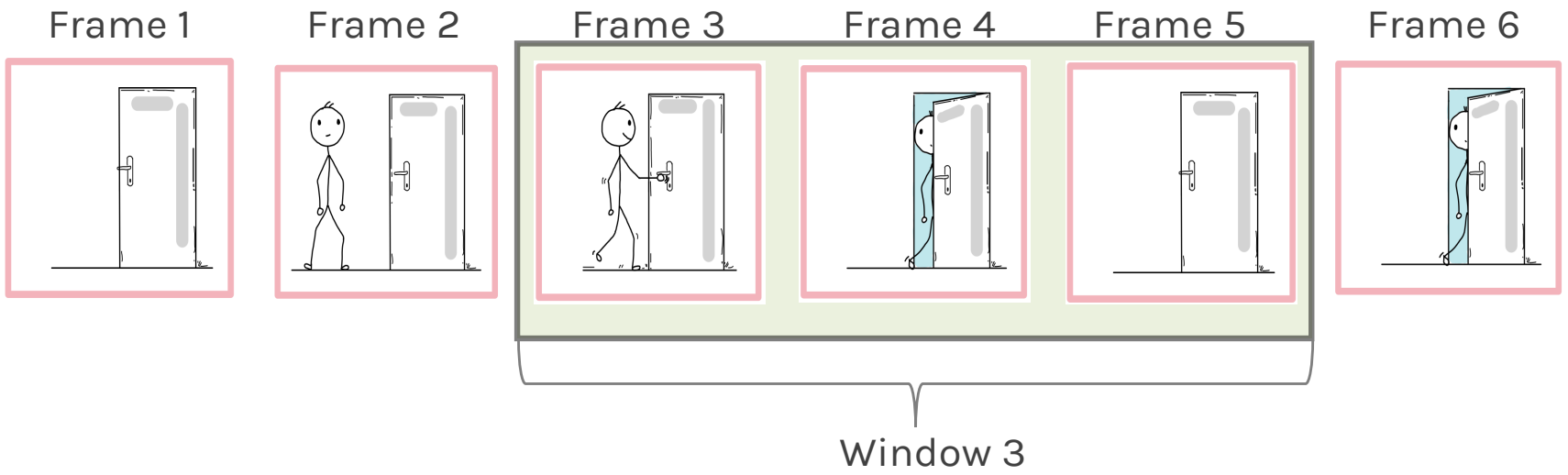
If we window a fixed number of frames as input to a NN like MLP or CNN then prediction will work

Let's think about sequences



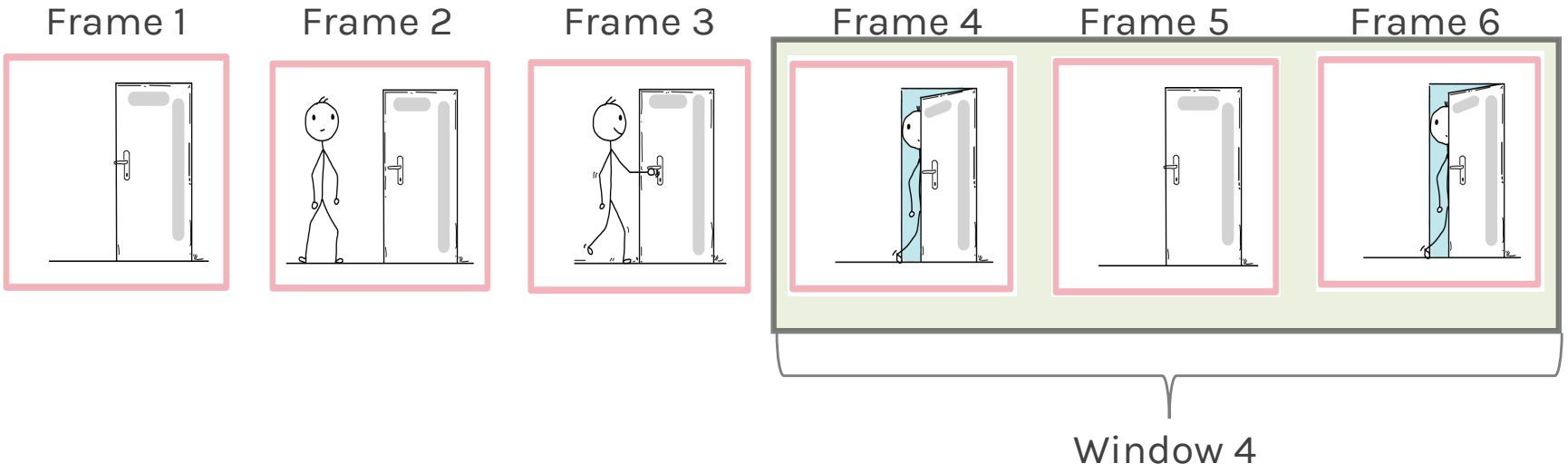
If we window a fixed number of frames as input to a NN like MLP or CNN then prediction will work

Let's think about sequences



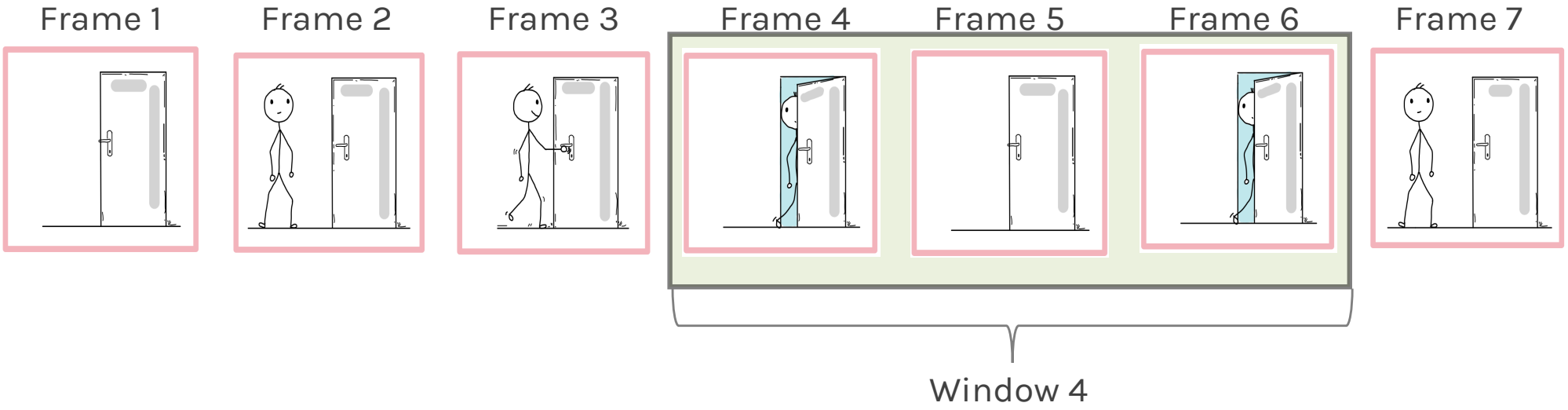
If we window a fixed number of frames as input to a NN like MLP or CNN then prediction will work

Let's think about sequences



If we window a fixed number of frames as input to a NN like MLP or CNN then prediction will work

Let's think about sequences



If we window a fixed number of frames as input to a NN like MLP or CNN then prediction will work

Let’s think about sequences

However, consider the following sequence of frames:

Let’s think about sequences

What is the next frame?

Let's think about sequences

What is the next frame?

There are many options: the door shutting, the person coming out, etc.

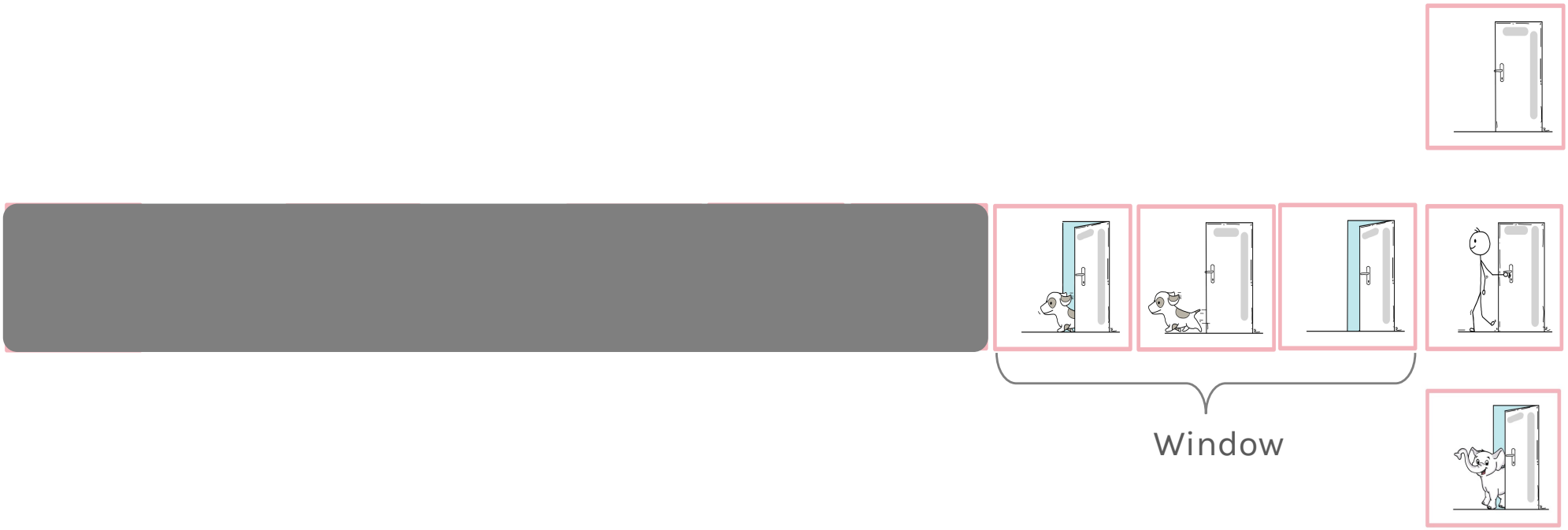
Let's think about sequences

What is the next frame?

There are many options: the door shutting, the person coming out, etc.

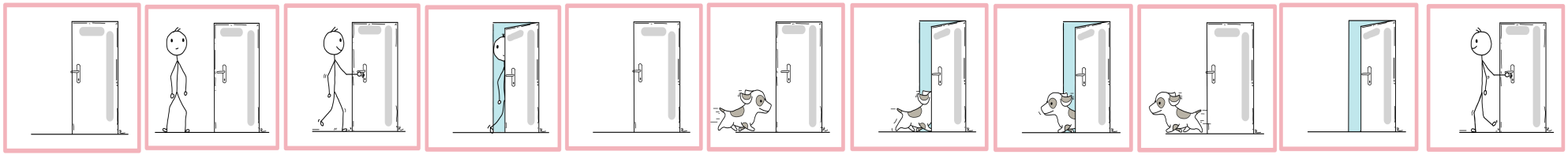
Let's think about sequences

What is the next frame?



There are many options: the door shutting, the person coming out, etc. ⋮
So a longer memory is needed.

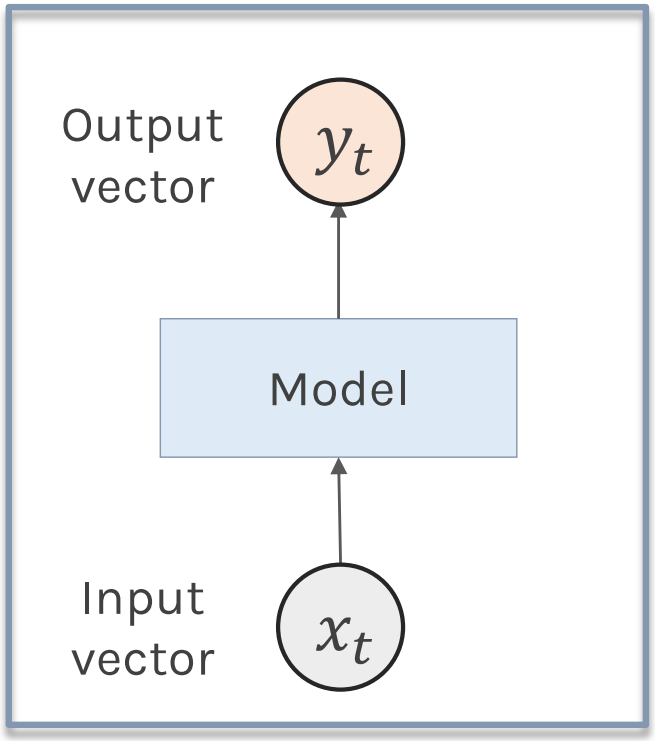
Let’s think about sequences



With longer memory, it is easier to predict

Intro to RNNs

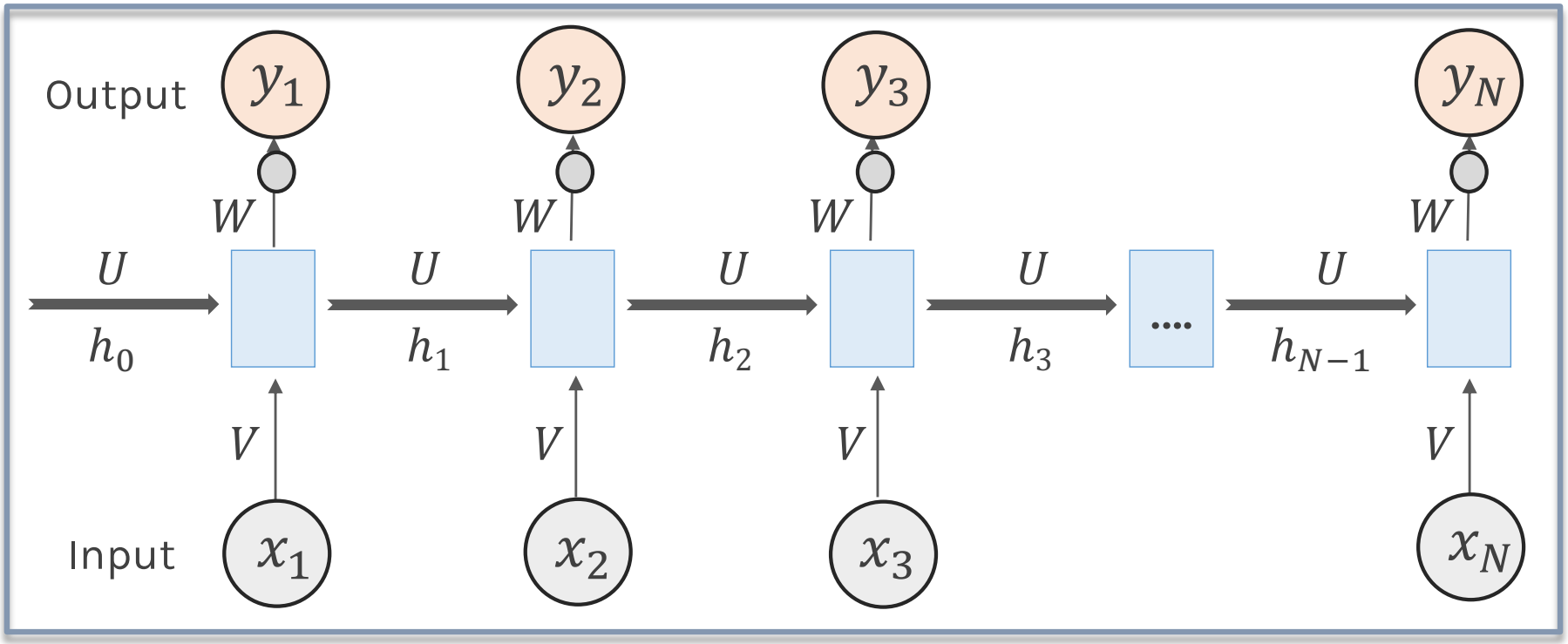
FEED FORWARD
NEURAL NETWORK



Cannot maintain
previous information

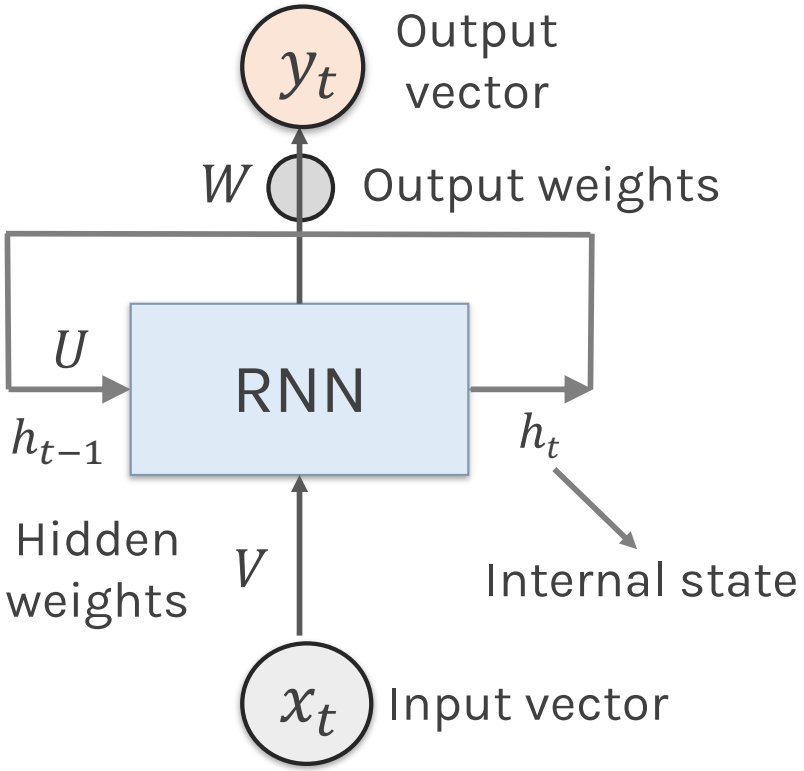
Intro to RNNs

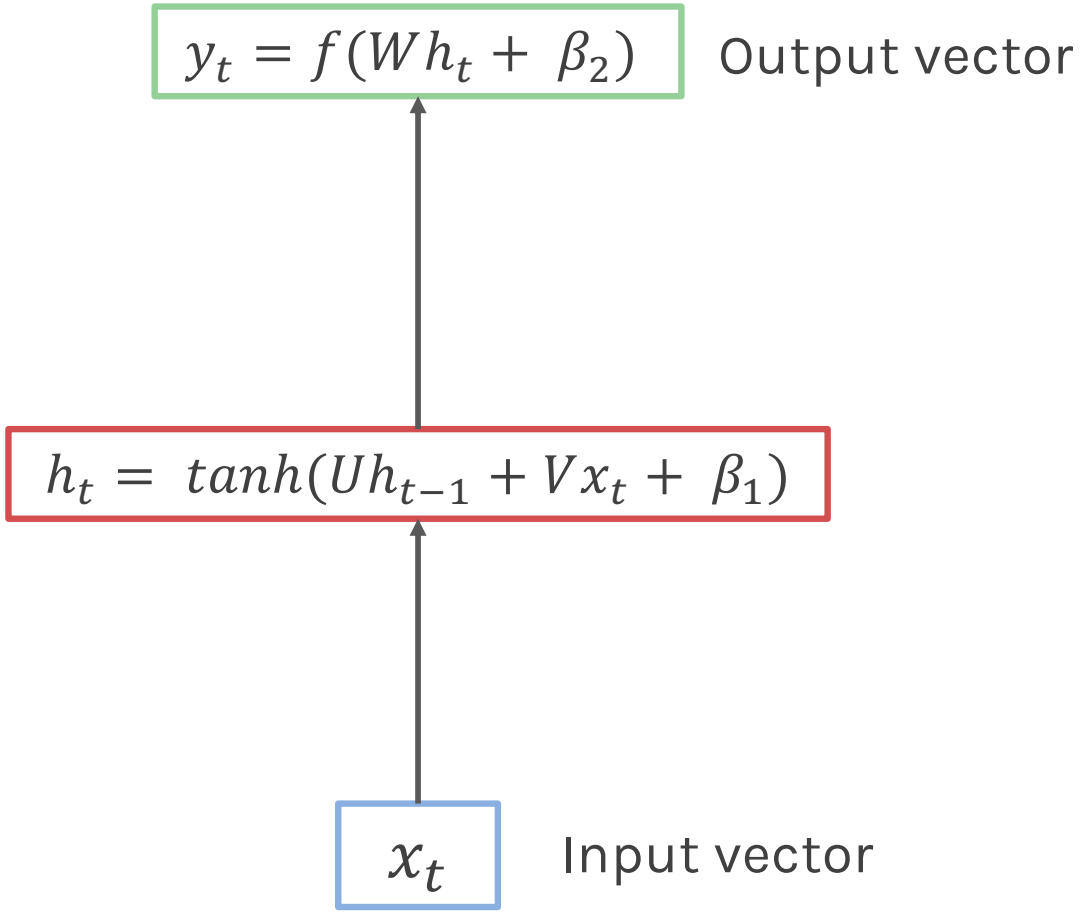
RECURRENT NEURAL NETWORK



- The term recurrent comes from the fact that information is being passed from one time step to the next internally within the network.
- Network has loops for information to persist over time.

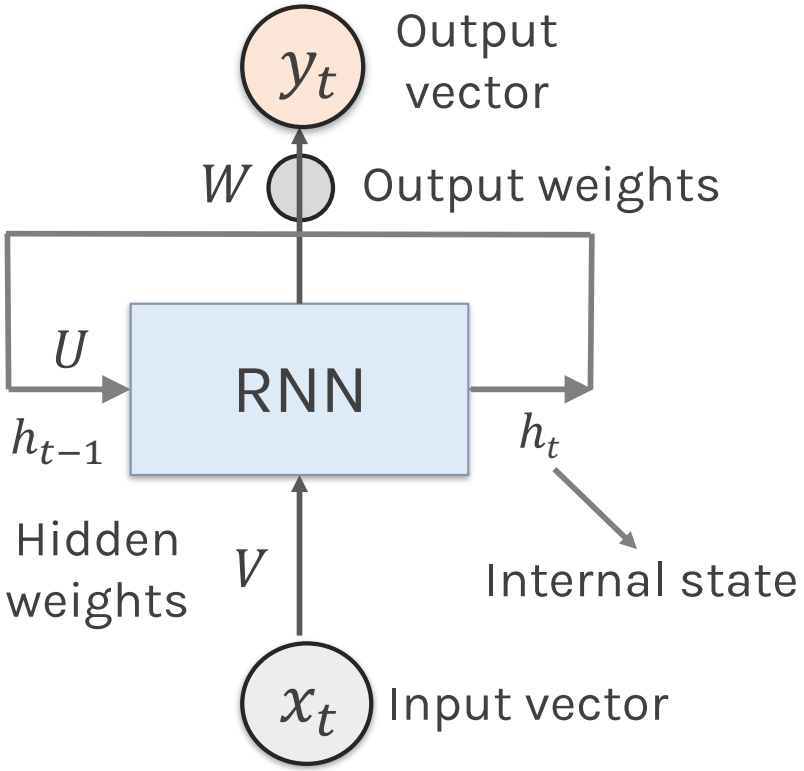
Intro to RNNs





U, *V* and *W* are three different weight matrices learned during training

Intro to RNNs



$y_t = f(Wh_t + \beta_2)$ Output vector

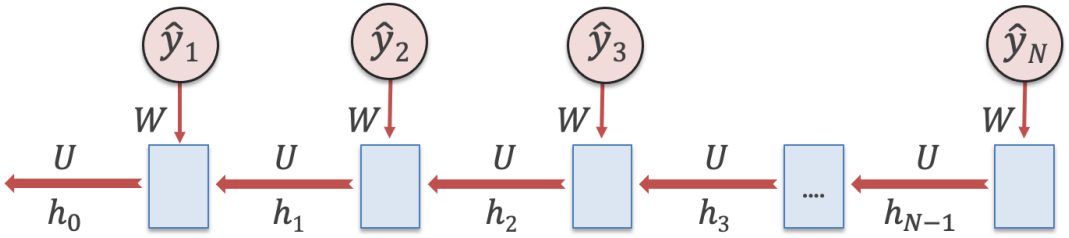
Sigmoid is used for binary classification. For multi class classification, we use softmax and add more nodes in the prediction layer. For regression we use a linear activation.

$h_t = tanh(Uh_{t-1} + Vx_t + \beta_1)$

x_t Input vector

U, *V* and *W* are three different weight matrices learned during training

Intro to RNNs: training issues



For longer sequences, we must backpropagate through more time steps.

This requires the gradient to be multiplied many times which causes the following issues:

If many values < 1, then the product, i.e., the gradient, will be close to zero. This is called the **vanishing gradient problem**.

This causes the parameters to update very slowly.

If many values > 1, then the product, i.e., the gradient, will explode. This is called the **exploding gradient problem**.

This causes an overflow problem.

Let's think about tokens

How do we process a sequence? **Tokenization** converts a sequence into a symbolic or structured representation (token) that can modeled

For **language**, this can be:

- a word: "I am here" -> ["I", "am", "here"]
- rule-based (regex): "can't" -> ["ca", "n't"]
- subword: "unhappiness" -> ["un", "happiness"]
- character: "cat" -> ["c", "a", "t"]

GPT models use Byte-Pair Encoding (BPE): subwords are formed iteratively from mergers of the most frequently adjacent character pairs

For **time series:**

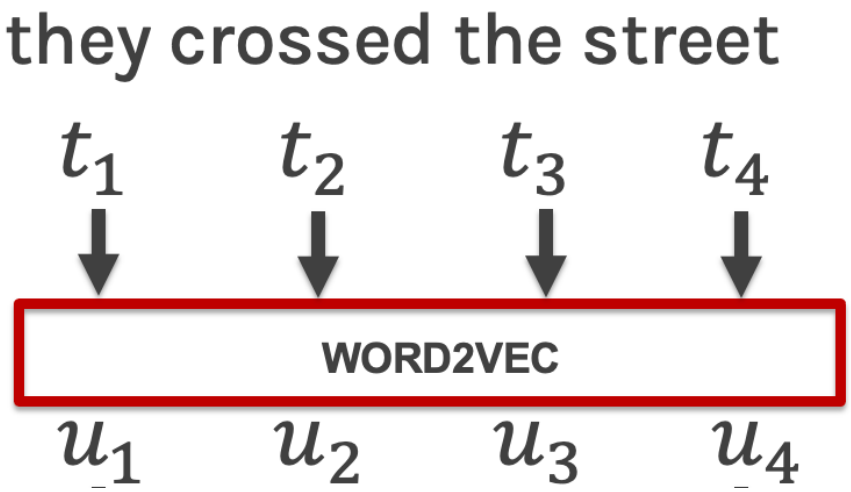
- discretization
- event-based
- chunking (fixed window)
- autoencoder to a discrete latent space (codebook token)

Let's think about embedding

How do we represent high dimensional discrete data (like words, categories, or nodes in graph) into a dense, continuous vector representation in a lower dimensional space?

An **embedding** is a learned mapping from a discrete input space to a continuous vector space. Similar items have closer vectors.

Word2Vec is a neural network model developed by Mikolov et al. (2013) at Google that learns word embeddings.

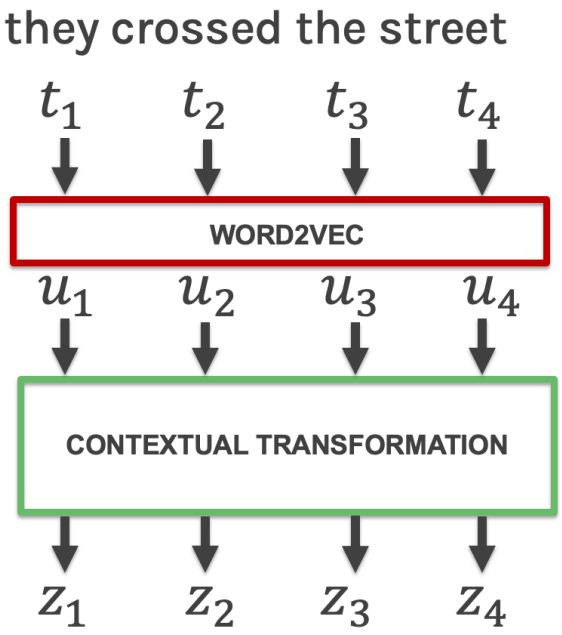


Let's think about context

Donald was hit by a bus because **they** crossed the street

How do we find what "they" is referring to?

Make a contextual embedding that is a linear combination of the other word embeddings:



$$z_1 = \alpha_{11}u_1 + \alpha_{12}u_2 + \alpha_{13}u_3 + \alpha_{14}u_4$$

$$z_2 = \alpha_{21}u_1 + \alpha_{22}u_2 + \alpha_{23}u_3 + \alpha_{24}u_4$$

$$z_3 = \alpha_{31}u_1 + \alpha_{32}u_2 + \alpha_{33}u_3 + \alpha_{34}u_4$$

$$z_4 = \alpha_{41}u_1 + \alpha_{42}u_2 + \alpha_{43}u_3 + \alpha_{44}u_4$$

Let's think about context

The coefficients are weights that encapsulate the degree to which new contextual embeddings should incorporate the influence of other tokens

For words, we use the cosine similarity:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

$$u_1 \circ u_1 = s_{11} \qquad \alpha_{11}$$

$$u_1 \circ u_2 = s_{12} \qquad \alpha_{12}$$

$$u_1 \circ u_3 = s_{13} \qquad \alpha_{13}$$

$$u_1 \circ u_4 = s_{14} \qquad \alpha_{14}$$

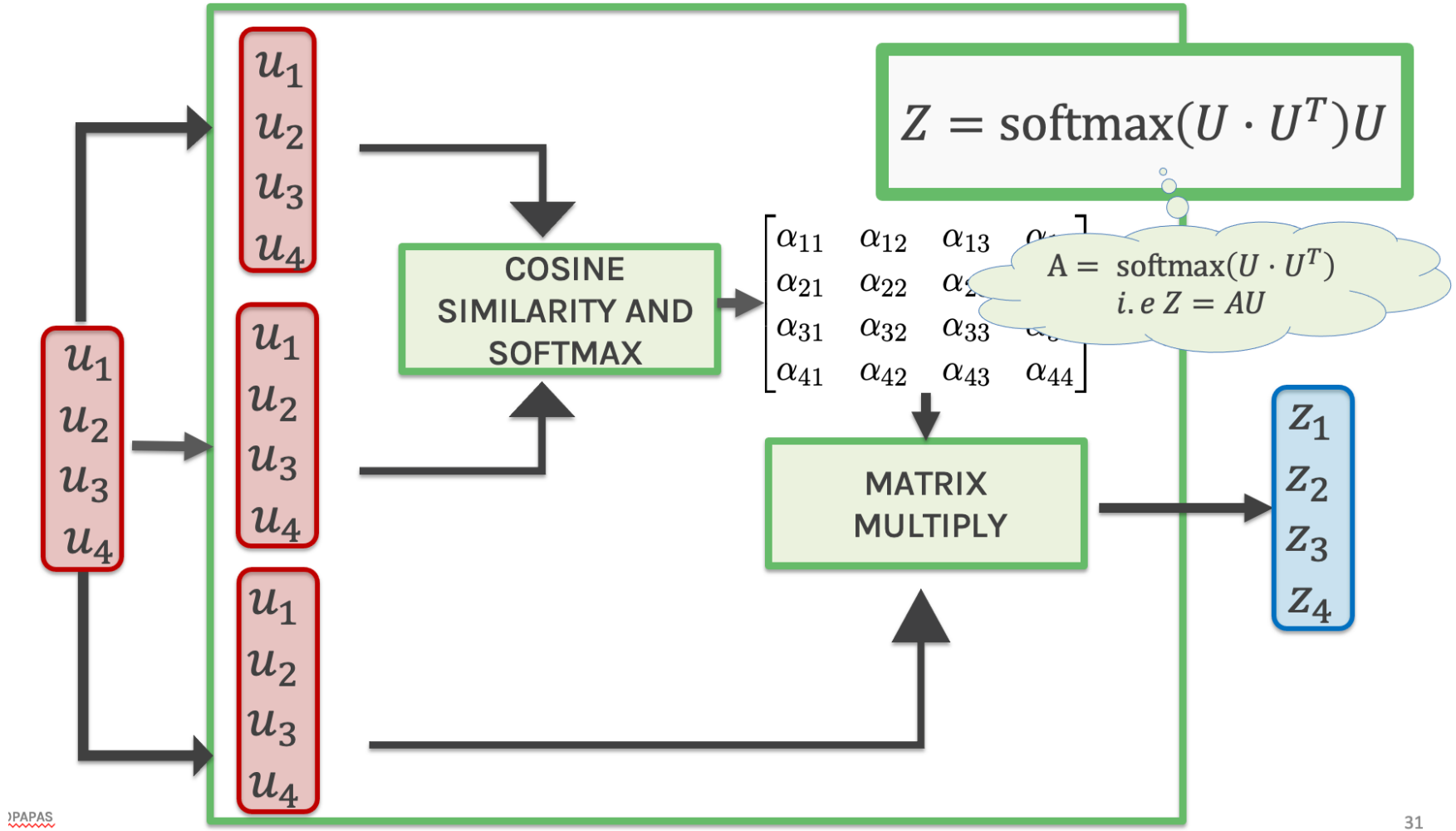
softmax(.)

WEIGHTS NORMALIZATION

$$\Sigma \alpha_{1i} = 1$$

$$z_1 = \alpha_{11} u_1 + \alpha_{12} u_2 + \alpha_{13} u_3 + \alpha_{14} u_4$$

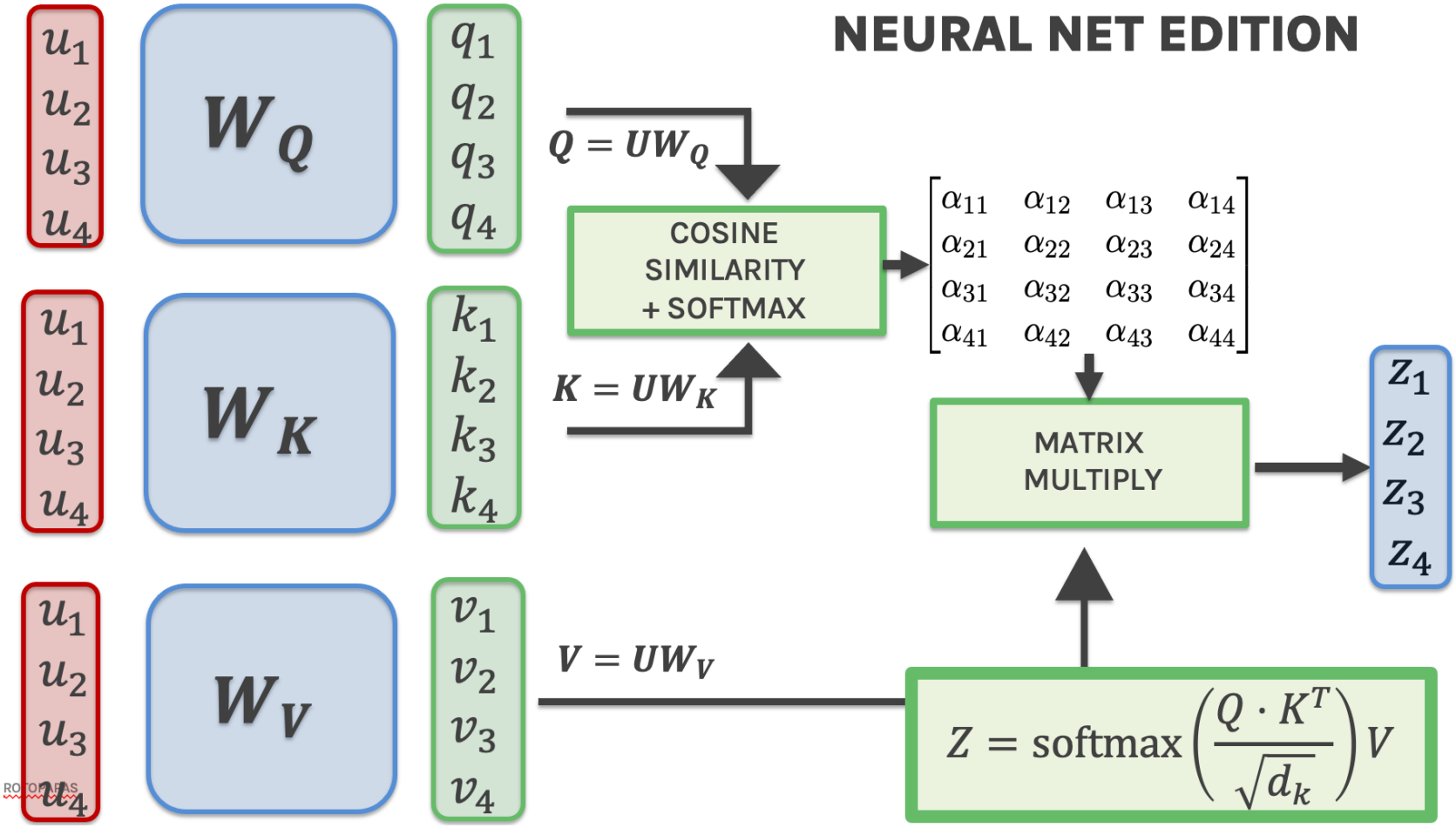
Let's think about attention



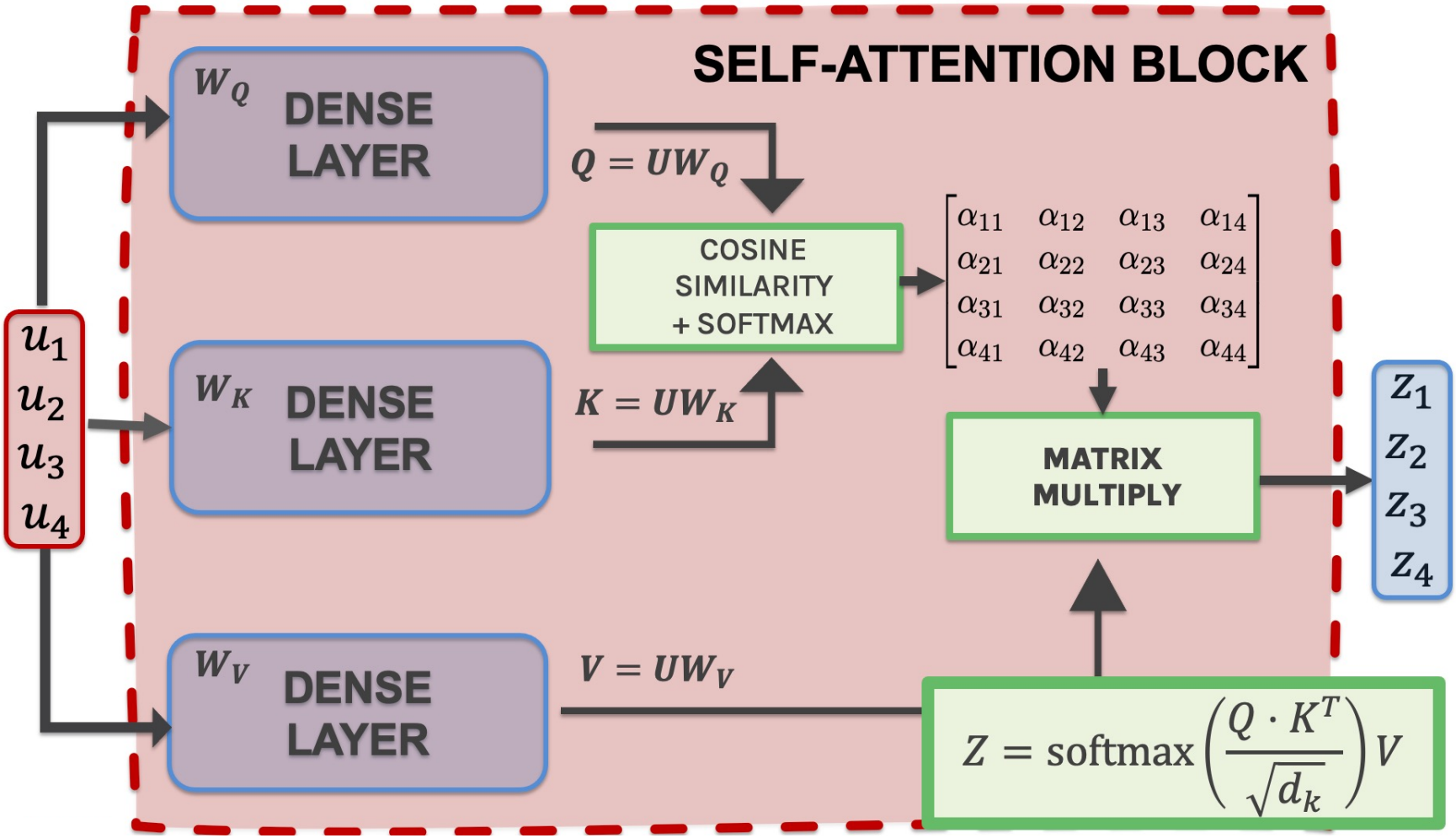
31

Let's think about attention

- As currently defined, two words will also have the similarity: no weights have been trained from a corpus

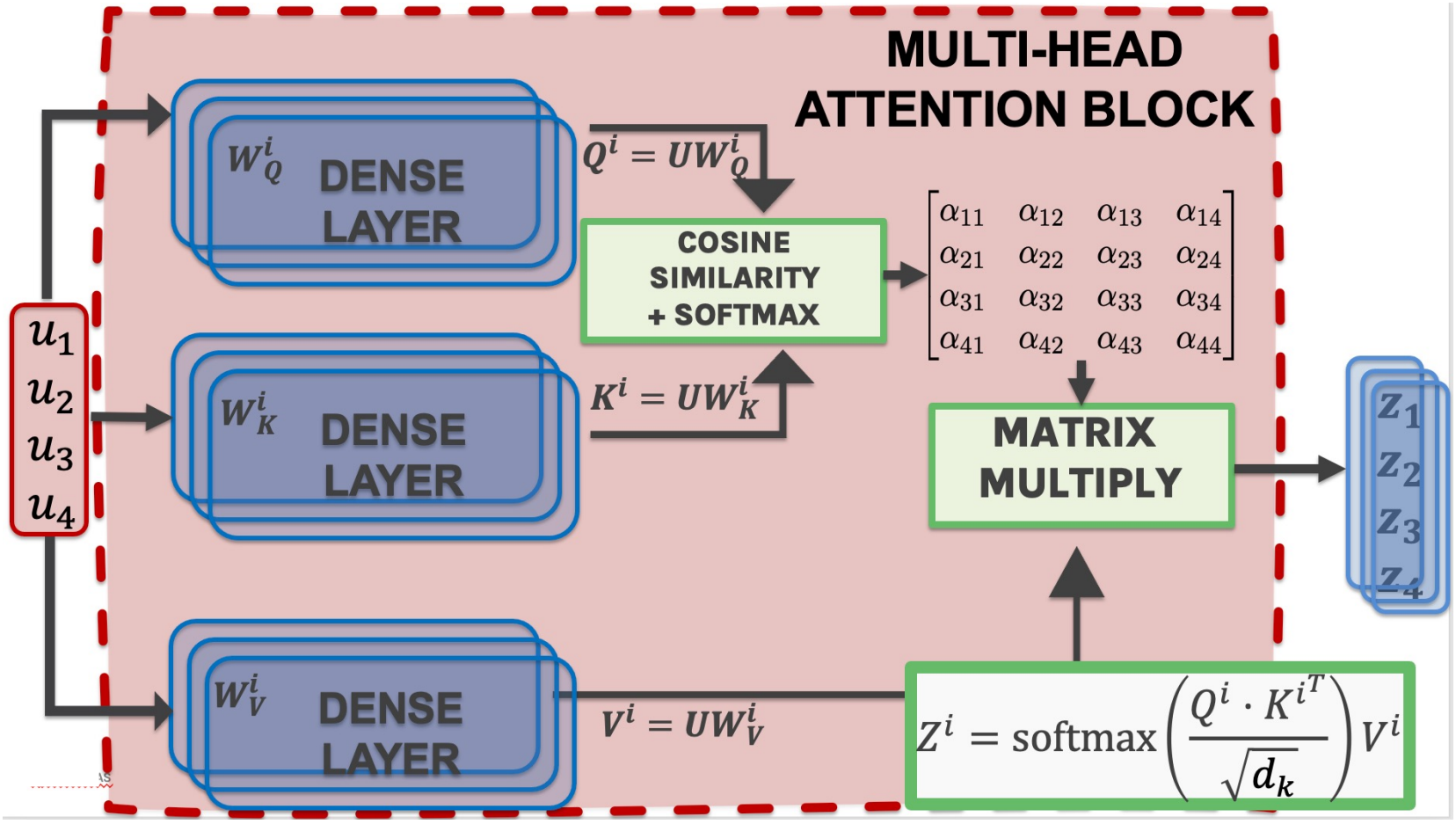


Self-attention block

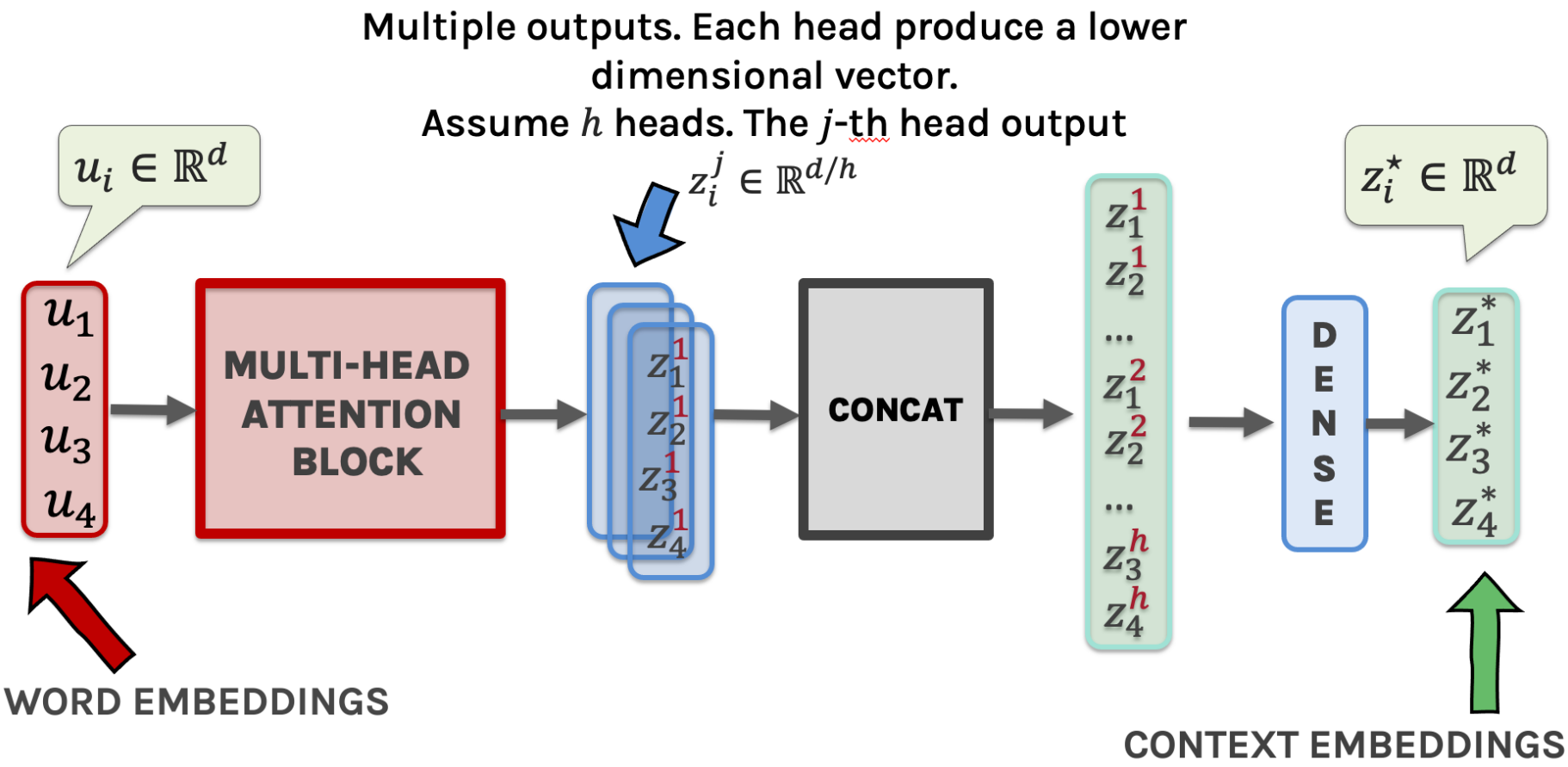


Multi-head attention block

- Attention leads to limited contextual mapping
- We can have multiple attention blocks to look for different relations (cf. CNNs with multiple filters to learn different features)

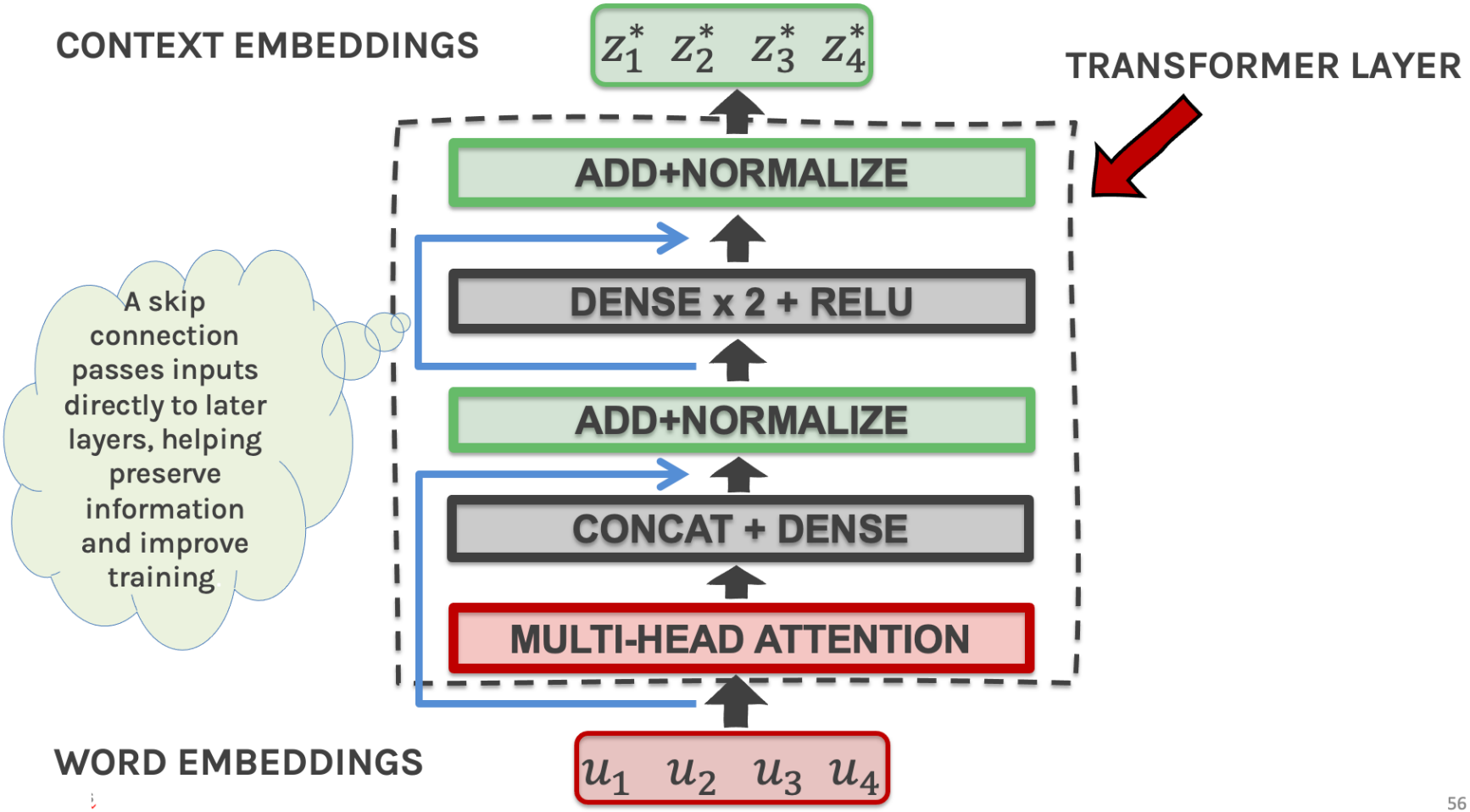


Multi-head attention block



Transformer layer

- Add skip connections to avoid the issue of vanishing gradients
- Add normalization layers to help gradients flow better during backpropagation



Let's talk about positional information

George spoke to Matthew about the Ay 119 class projects

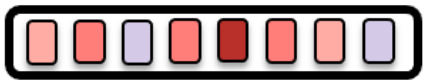
Matthew spoke to George about the Ay 119 class projects

RNNs inherently take the **order** of words into account

Multi-head attention blocks do not take order into account

Consider the input embedding u_i for a word at position i:

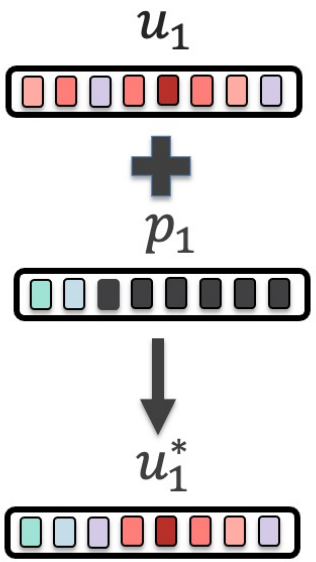
$$u_i$$



This will be the same for any position in the sentence

Let's talk about positional information

Modify the input embedding with a positional vector:



The same input embedding will have a different value depending on its position in the sentence

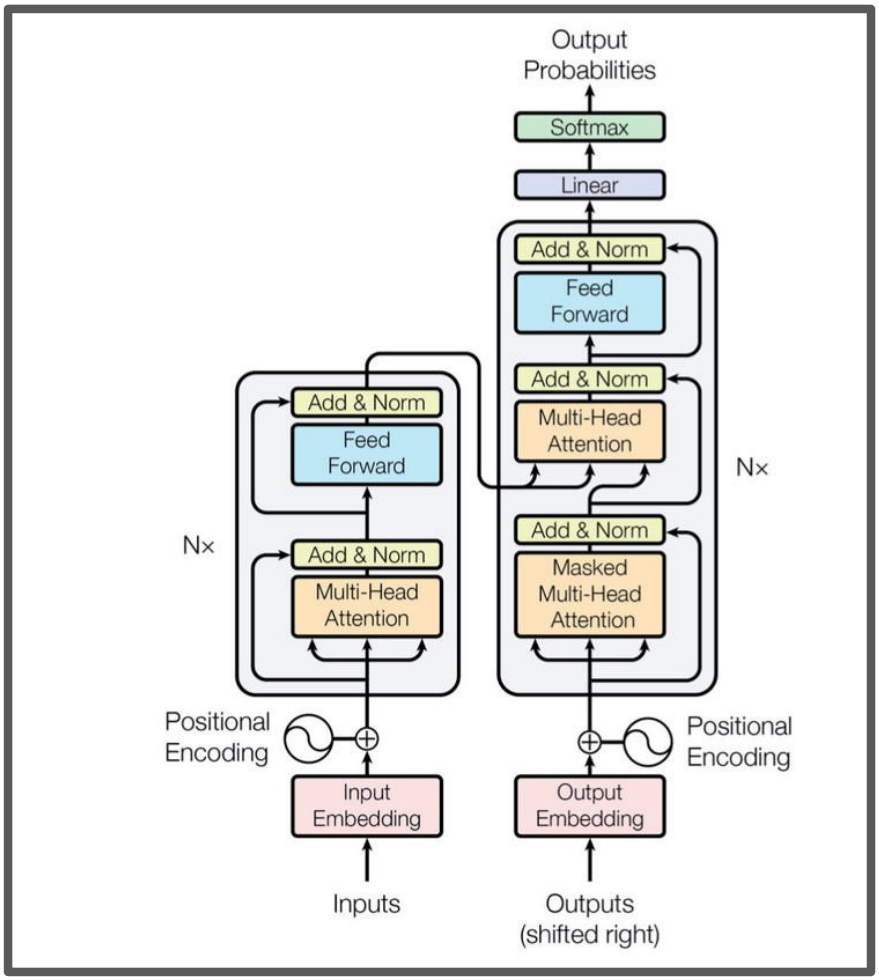
Let's talk about positional information

Each position in a sentence is assigned a vector that encodes its position (from Attention is All You Need):

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

The encoding uses a mix of trigonometric functions to ensure each position has a unique yet repeatable pattern

Putting it all together: transformers



Putting it all together: transformers