

Transformers



Dr. Manuel Castillo-Cara

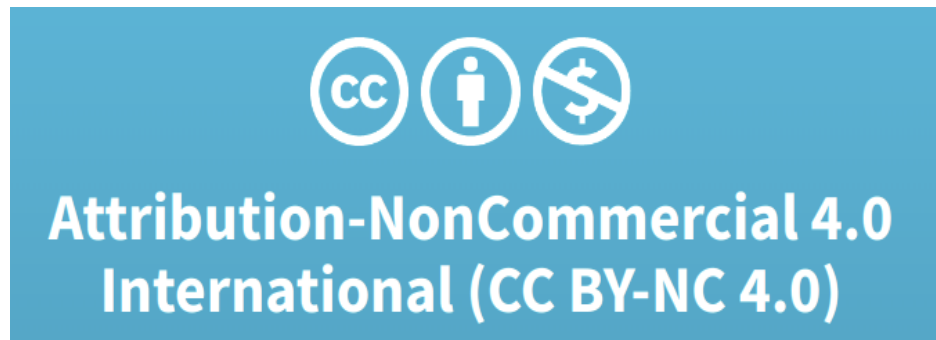
www.manuelcastillo.eu

**Departamento de Inteligencia Artificial
Escuela Técnica Superior de Ingeniería Informática
Universidad Nacional de Educación a Distancia (UNED)**

Preliminar



- Improving Deep Learning by Exploiting Synthetic Images © 2024 by Manuel Castillo-Cara is licensed under Attribution-NonCommercial 4.0 International



Índice



Background

Attention

Transformers

Vision Transformers

Inferencia en ViT

Conclusiones



Background



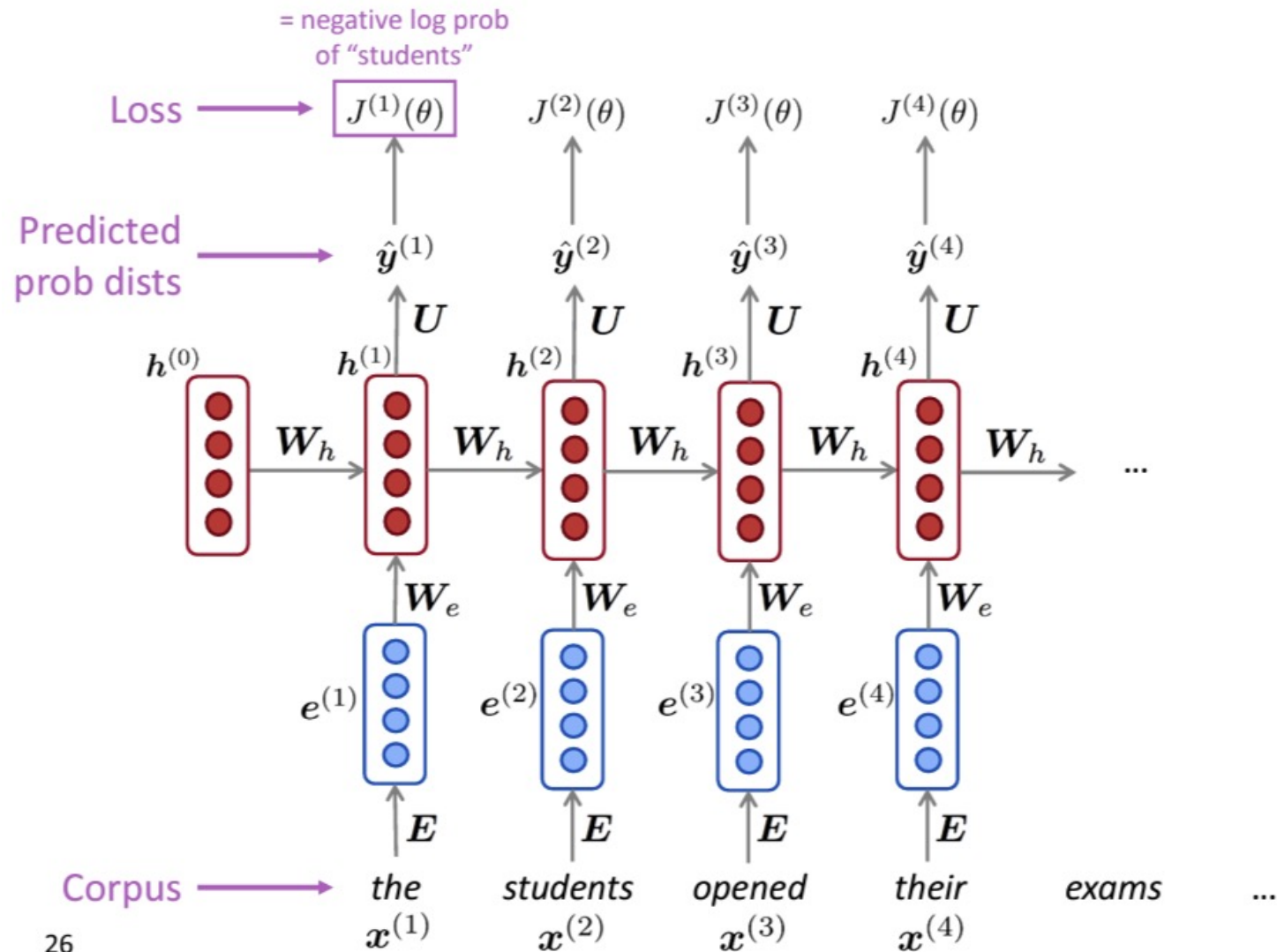
ETS de
Ingeniería
Informática



UNED

Recurrent Neural Networks

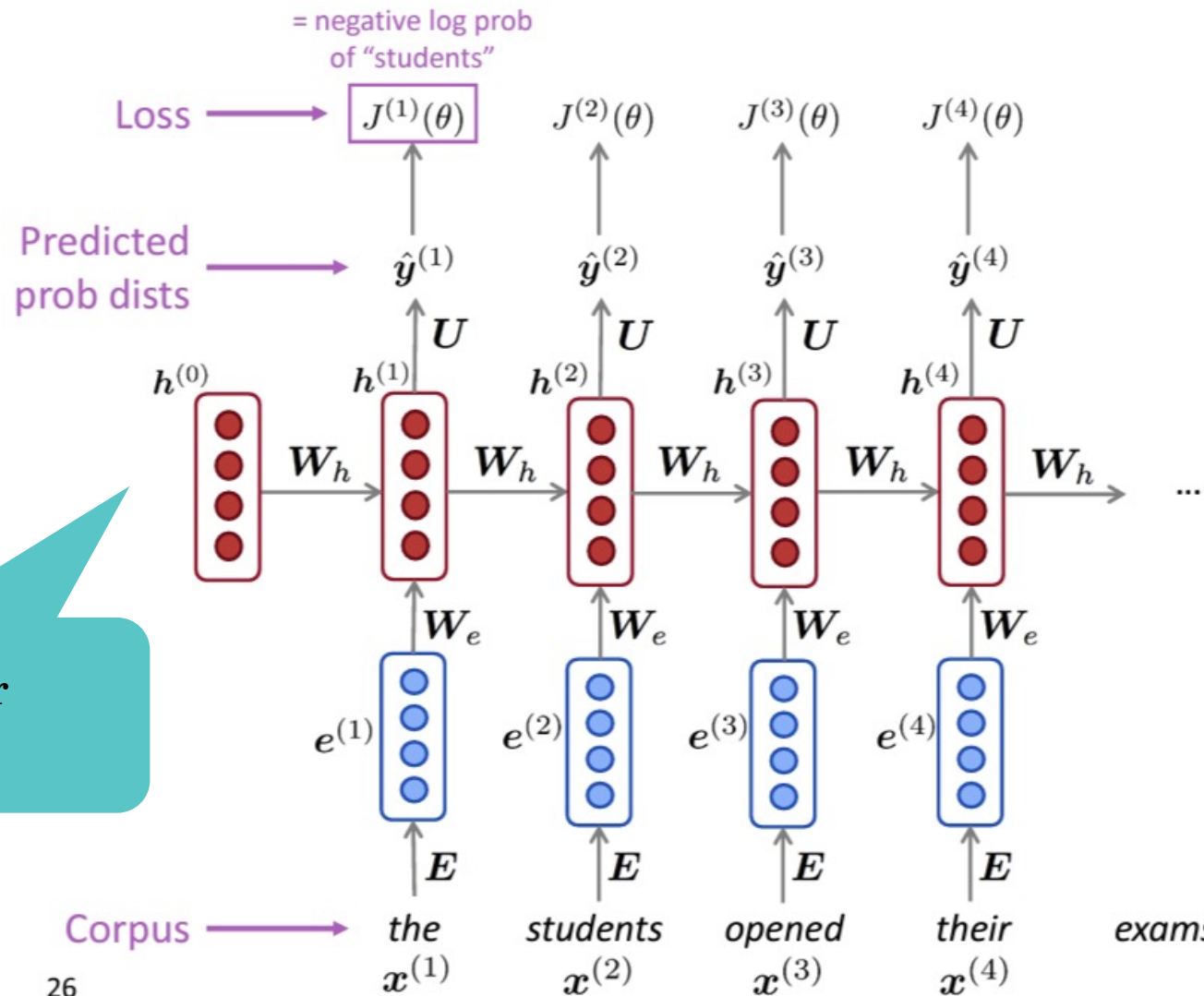
Problem





Recurrent Neural Networks

Problem



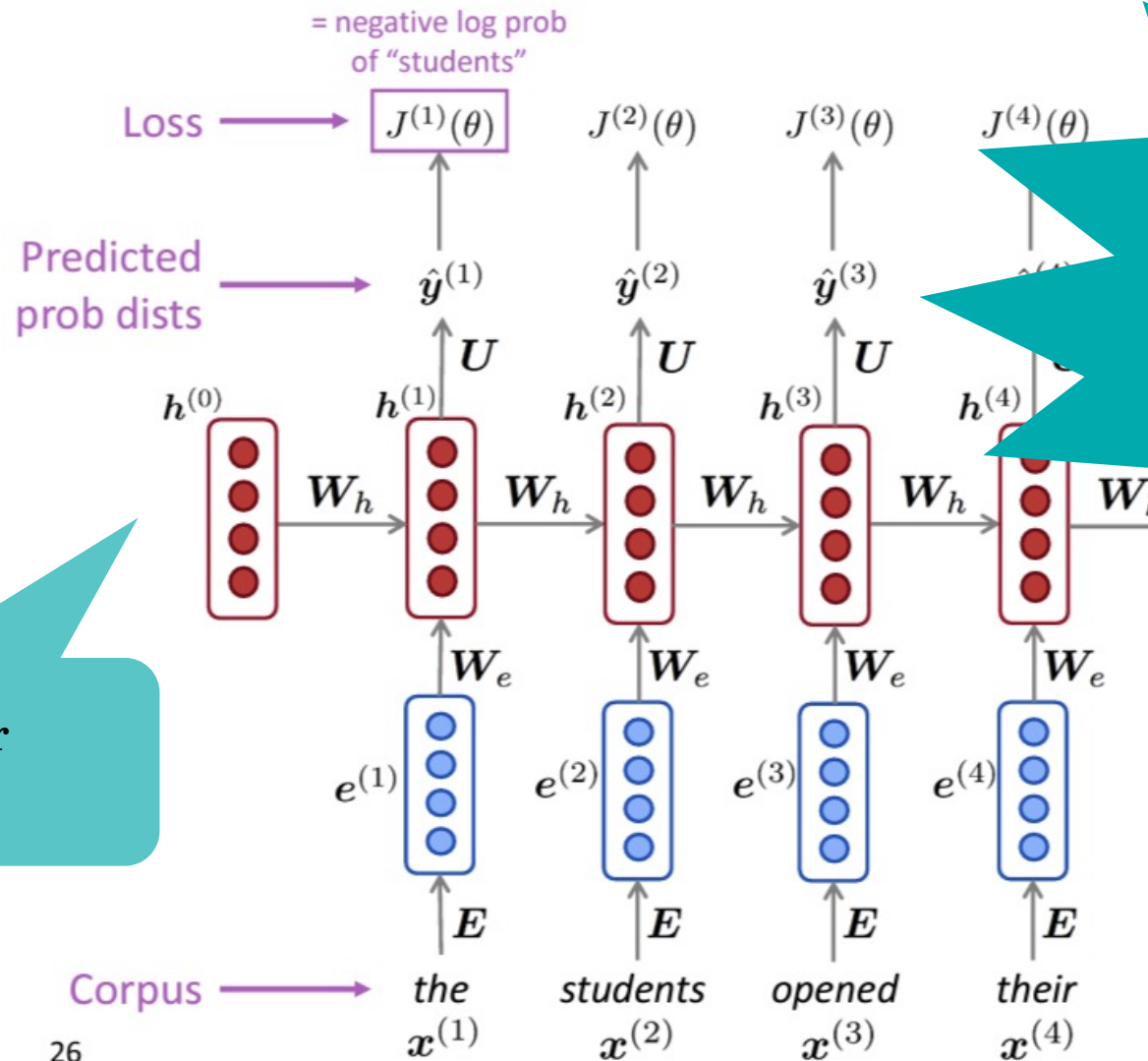
First Layers Minor Weight \rightarrow Forget

Last Layers - **more** Weight \rightarrow Memory



Recurrent Neural Networks

Problem



First Layers Minor Weight \rightarrow Forget

Last Layers - **more** Weight \rightarrow Memory

Recurrent Neural Networks

Problem



El jaguar dormía en la rama del árbol con su cola de soporte.

Recurrent Neural Networks

Problem



El jaguar dormía en la rama del árbol con **su** cola de soporte.



With NRNs, it is forgotten that
"su" refers to the jaguar

Recurrent Neural Networks

Problem



El jaguar dormía en la rama del árbol con **su** cola de soporte.



**Attention solves
this problem**

With NRNs, it is forgotten that
"su" refers to the jaguar



Attention

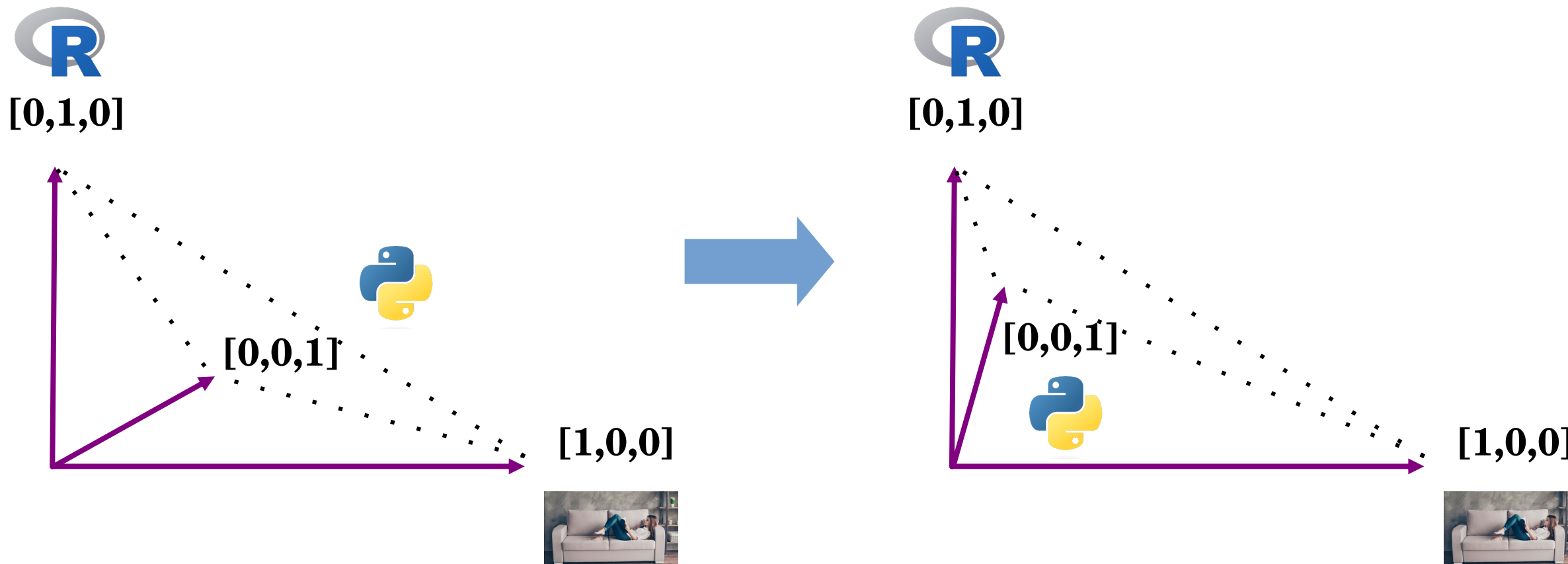
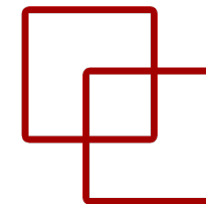


ETS de
Ingeniería
Informática

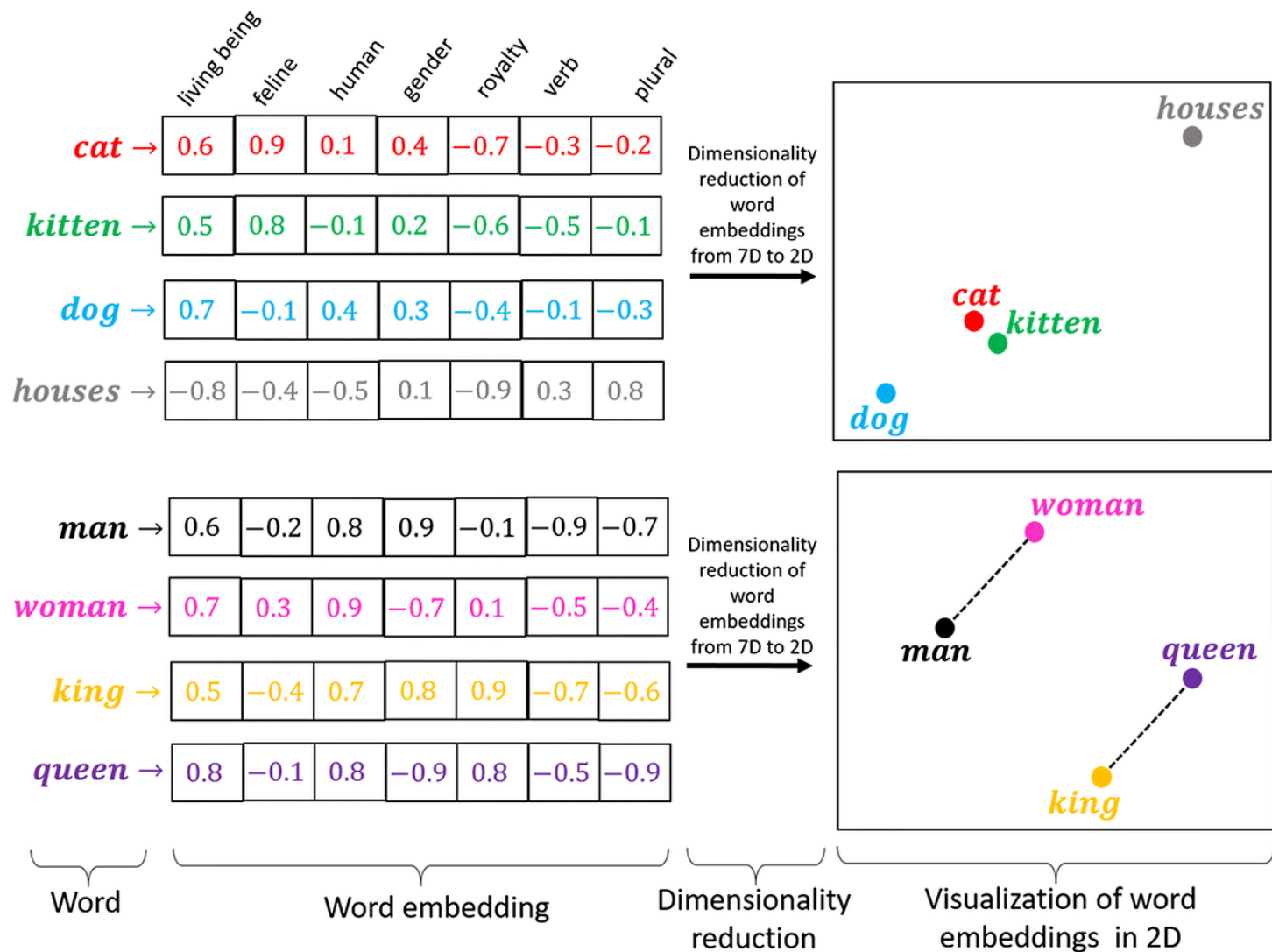
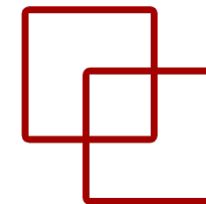
UNED

Encoding

One-Hot Encoding – Problem



Word embedding



Attention

Background



El

jaguar

dormía

en

su

árbol

El

jaguar

dormía

en

su

árbol

Attention

Background



El

jaguar

dormía

en

su

árbol

...

El

jaguar

dormía

en

su

árbol



Attention Background

El

jaguar

dormía

en

su

árbol

...

El

jaguar

dormía

en

su

árbol

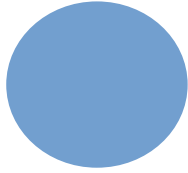
Look up the relationship of all
the words to all the other words!



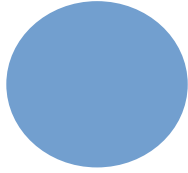
Vectors

Key - Search

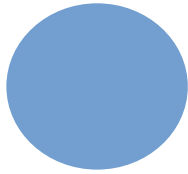
El



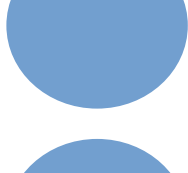
jaguar



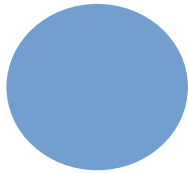
dormía



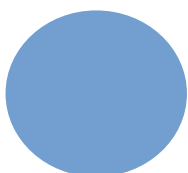
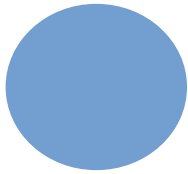
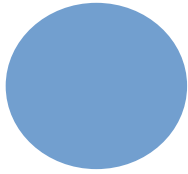
en



su



árbol



El

jaguar

dormía

en

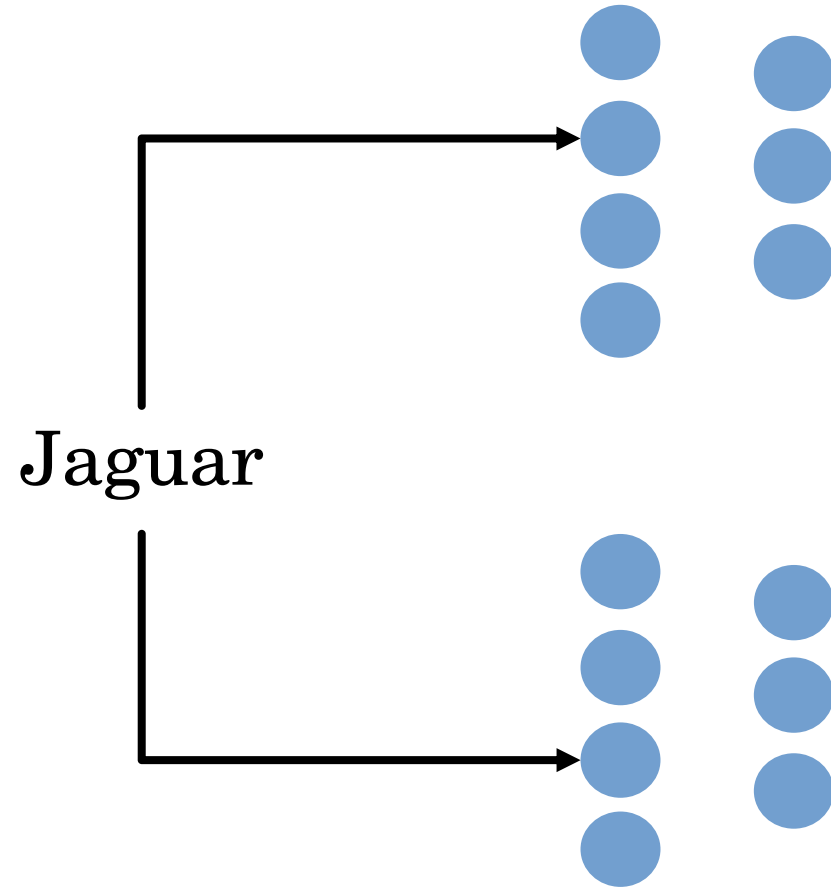
su

árbol

And it is neural networks that learn to find these relationships

Vectors

Key - Search

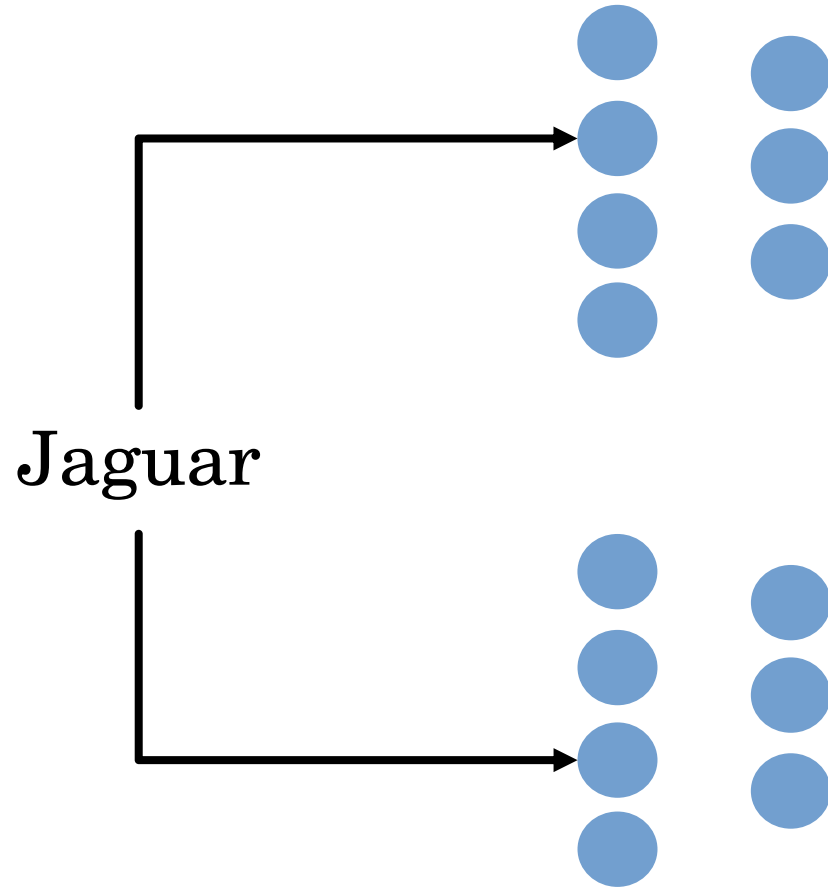


Two different NNs to generate two different vectors



Vectors

Key - Search



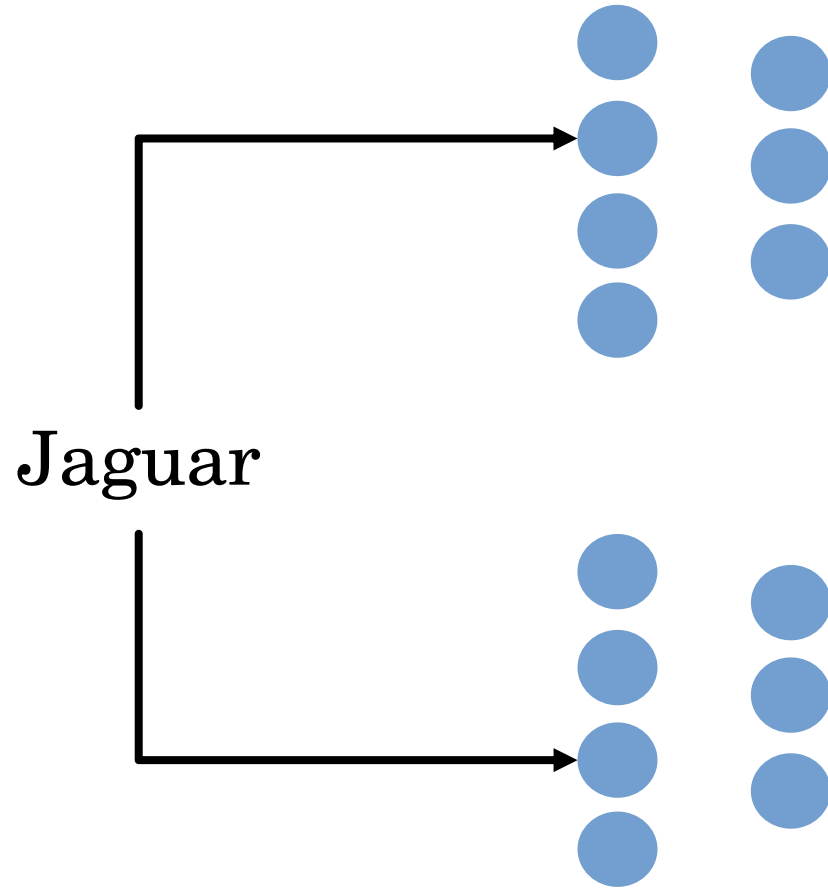
[0,74 -0,44 ...0,54]

Vector Identifier (key):
Obtains the properties that describe the word → **Meaning**



Vectores

Key - Query



[0,74 -0,44 ...0,54]

Key Vector:

Obtains the properties that describe the word → **Meaning**

[0,34 -0,14 ...0,34]

Query Vector :

Gets the properties of what the word is looking for → **Context**



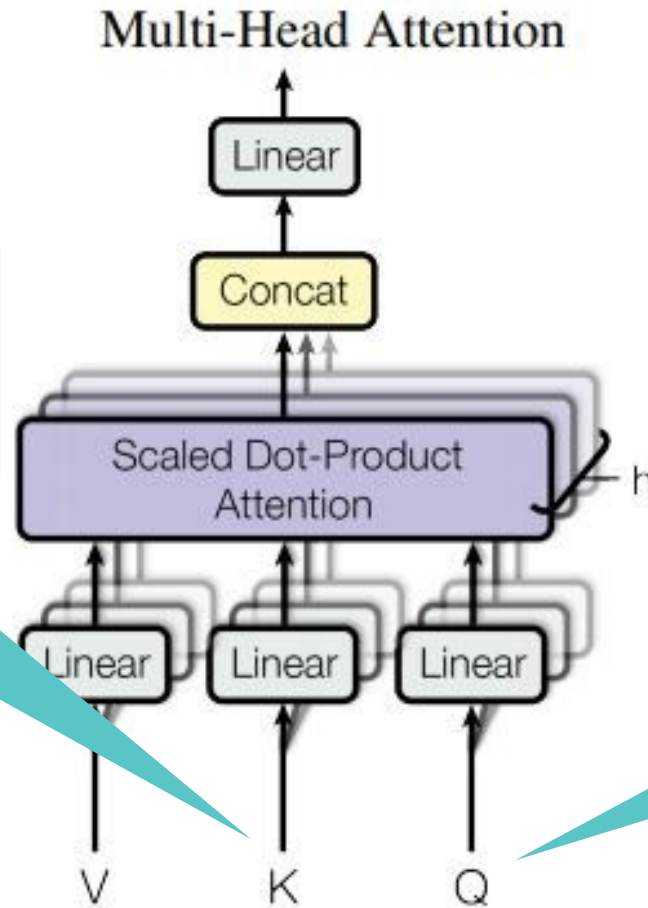
Vectores

Key - Query

Key Vector:

Obtains the properties that describe the word → **Meaning**

Jaguar

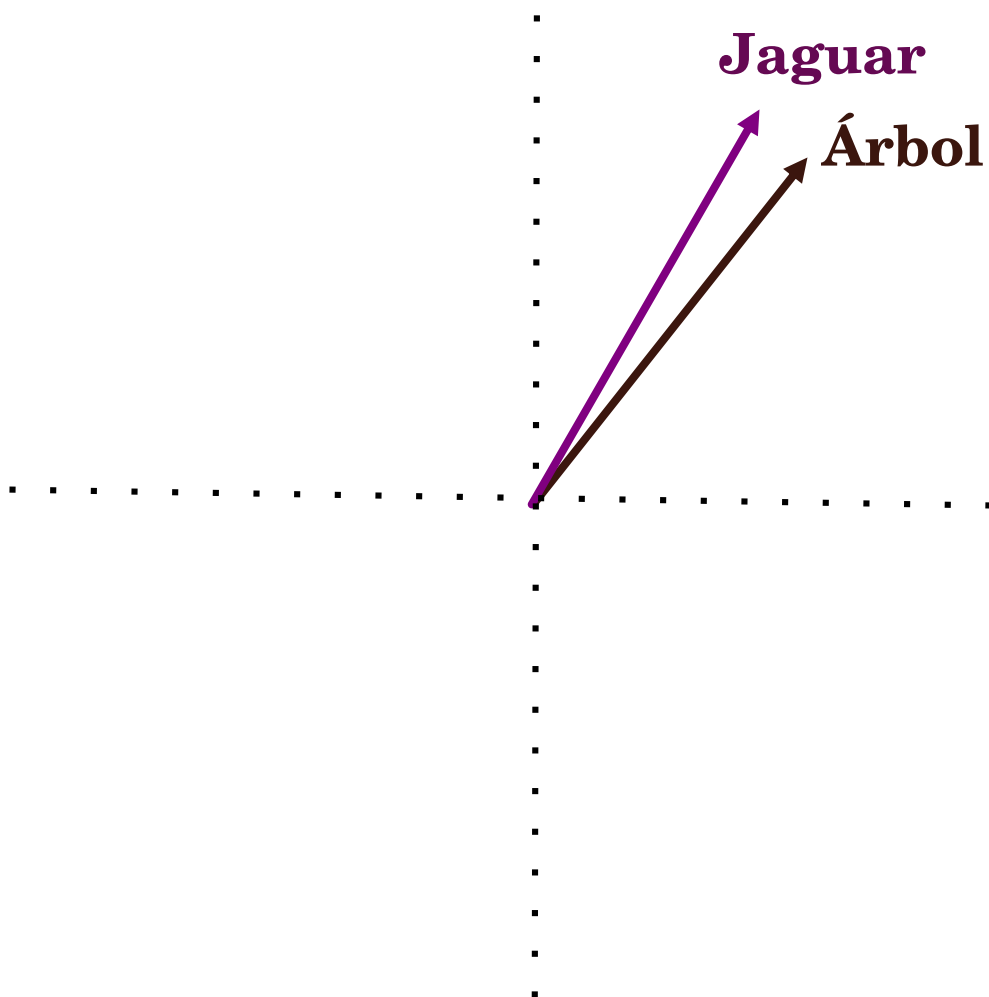
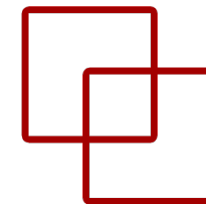


Query Vector :

Gets the properties of what the word is looking for → **Context**

Attention

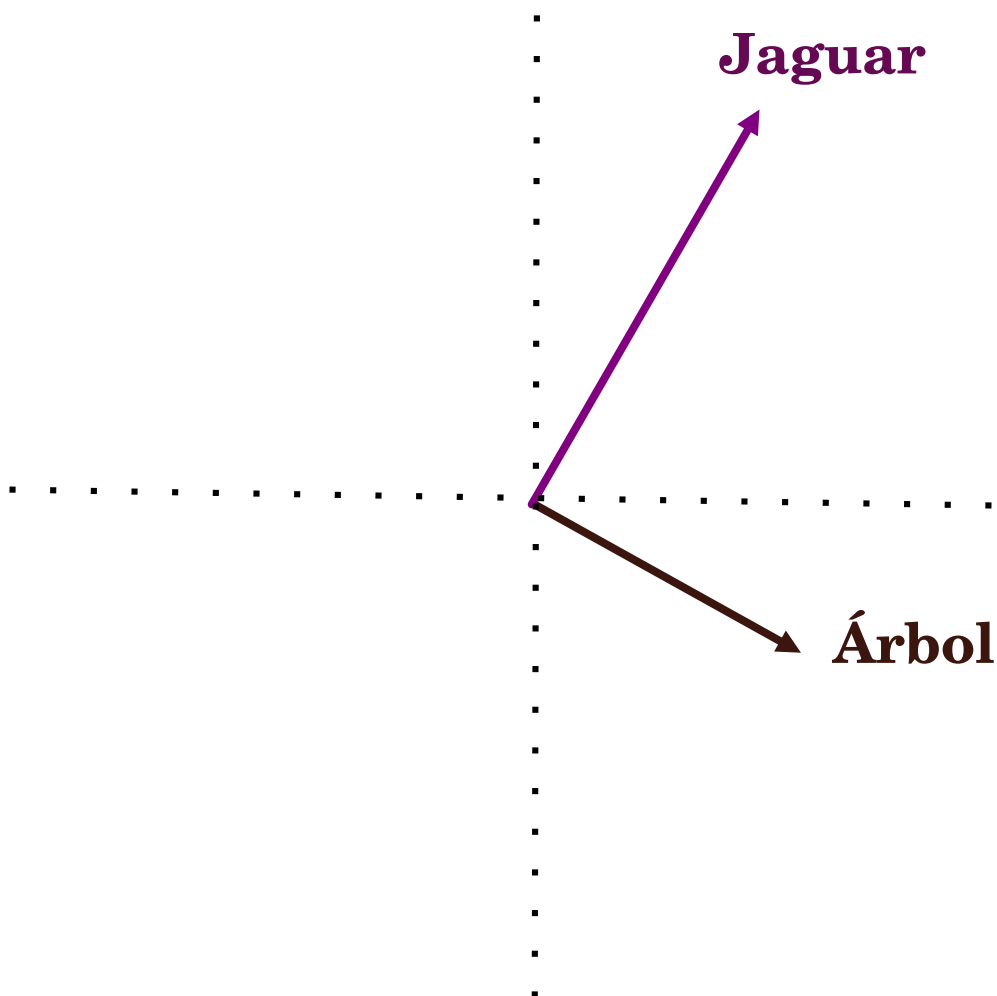
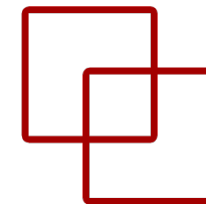
Scalar Product



If the vectors are close then they are **coincident words** → they are in their context

Attention

Scalar Product



If the vectors are **far apart** then they are not matching words → they are **not** in their **context**



Attention Background

El

jaguar

dormía

en

su

árbol

...

Key Vector

El

jaguar

dormía

en

su

árbol

Query Vectors



Attention Background

El

jaguar

dormía

en

su

árbol

...

Key Vector

El

jaguar

dormía

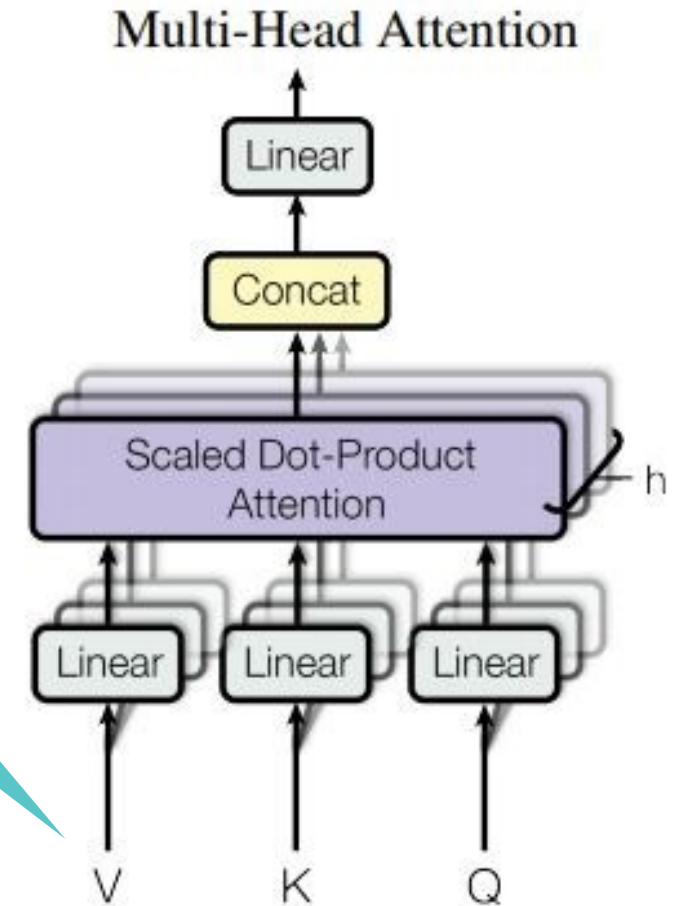
en

su

árbol

Query Vector

And Value Vector?



Attention

Background



El

jaguar

dormía

en

su

árbol

El

jaguar

dormía

en

su

árbol



Attention Background

El

jaguar

dormía

en

su

árbol

El

jaguar

dormía

en

su

árbol

You can calculate the **compatibility** of each *key* word with all other *query* words



Attention Background

El

jaguar

dormía

en

su

árbol

El

jaguar

dormía

en

su

árbol

You can calculate the **compatibility** of each *key* word with all other *query* words

**It's called vector
Value**



Attention

Background

El	El	0,34
jaguar	jaguar	-0,44
dormía	dormía	0,19
en	en	-0,24
su	su	0,64
árbol	árbol	0,84

Calculate the scalar product:
 $v(key) * v(query) \rightarrow$ the result
the greater the result, then the
greater the **compatibility**
between those words



Attention Background

El

jaguar

dormía

en

su

árbol

El

jaguar

dormía

en

su

árbol

0,34

-0,44

0,19

-0,24

0,64

0,84

This is **ATTENTION**



Values vector

Input



El **jaguar** dormía en su árbol

Attention



Values vector

Input



El **jaguar** dormía en su árbol

Attention

0,34

0,14

0,75

0,11

0,52

0,26



Values vector

Input



El **jaguar** dormía en su árbol

Attention

0,34

0,14

0,75

0,11

0,52

0,26

**Values
Vector**

0,34

-0,44

0,19

...

0,84



Values vector

Input



El jaguar dormía en su árbol

Attention

0,34 0,14 0,75 0,11 0,52 0,26

**Values
Vector**

**0,34 0,31
-0,44 -0,54
0,19 0,49
... ...
0,84 0,34**



Values vector

Input



El **jaguar** dormía en su árbol

Attention

0,34 0,14 0,75 0,11 0,52 0,26

**Values
Vector**

0,34	0,31			0,45
-0,44	-0,54			-0,14
0,19	0,49			0,26
...
0,84	0,34			0,43



Values vector

Input



El **jaguar** dormía en su árbol

Attention

0,34 0,14 0,75 0,11 0,52 0,26

**Values
Vector**

0,34		0,31				0,45
-0,44		-0,54				-0,14
0,19	+	0,49	+		+	0,26
...	
0,84		0,34				0,43



Output vector



Values vector

El

jaguar

dormía

en

su

árbol

El

jaguar

dormía

en

su

árbol

Vector output #1

Vector output #2

Vector output #3

Vector output #4

Vector output #4

Vector output #5



Values vector

El

jaguar

dormía

en

su

árbol

El

jaguar

dormía

en

su

árbol

Vector output #1

Vector output #2

Vector output #3

Vector output #4

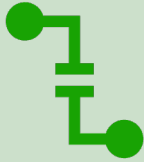
Vector output #4

Vector output #5

These value vectors
now if they have the
context of the entire
text



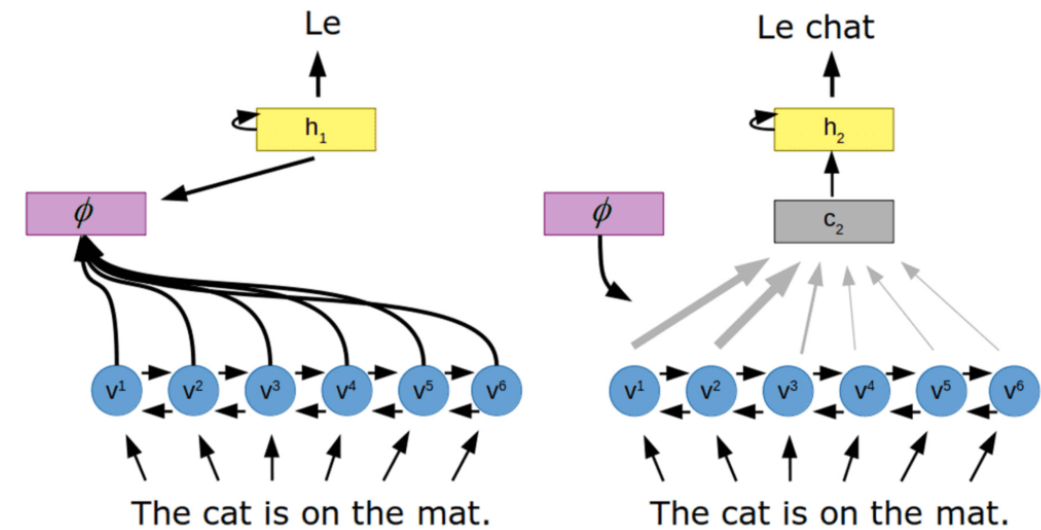
Encoder-Decoder with attention



The **attention mechanism** (ϕ) learns a set of attention weights that capture the relationship between the **encoded vectors** (v) and the **hidden state of the decoder** (h) to generate a **context vector** (c) through a weighted sum of all the hidden states of the encoder.



This way, the decoder would have **access to the entire input stream**, focusing specifically on the **most relevant** input information to generate the **output**.





Transformers



ETS de
Ingeniería
Informática

UNED



Consideraciones previas



The architecture of the Transformer dispenses with any recurrence and relies solely on a self-attention mechanism.



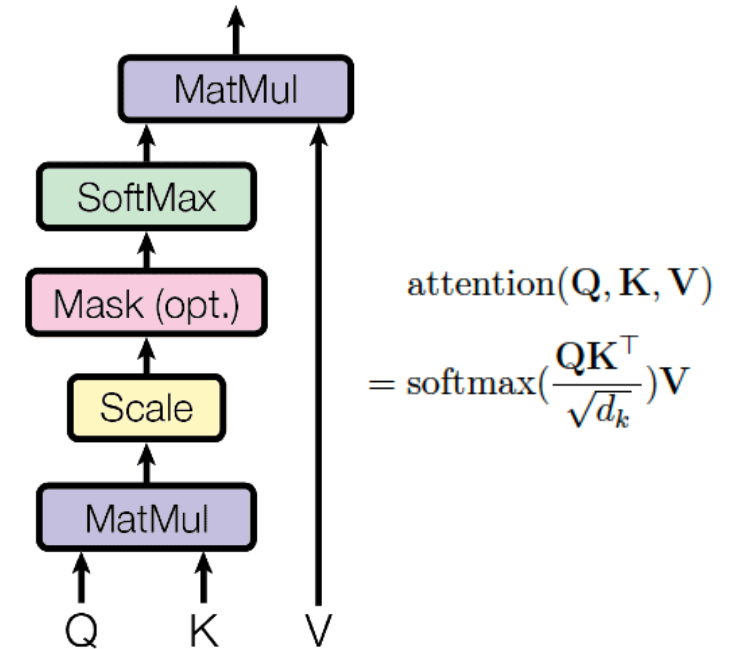
The self-attention mechanism is based on the use of queries, keys, and values, which are generated by multiplying the encoder's representation of the same input sequence with different weight arrays.



The Transformer uses scalar product attention, in which each query is compared to a database of keys using a scalar product operation in the process of generating the attention weights.



These weights are then multiplied by the values to generate a final attention vector.





Consideraciones previas



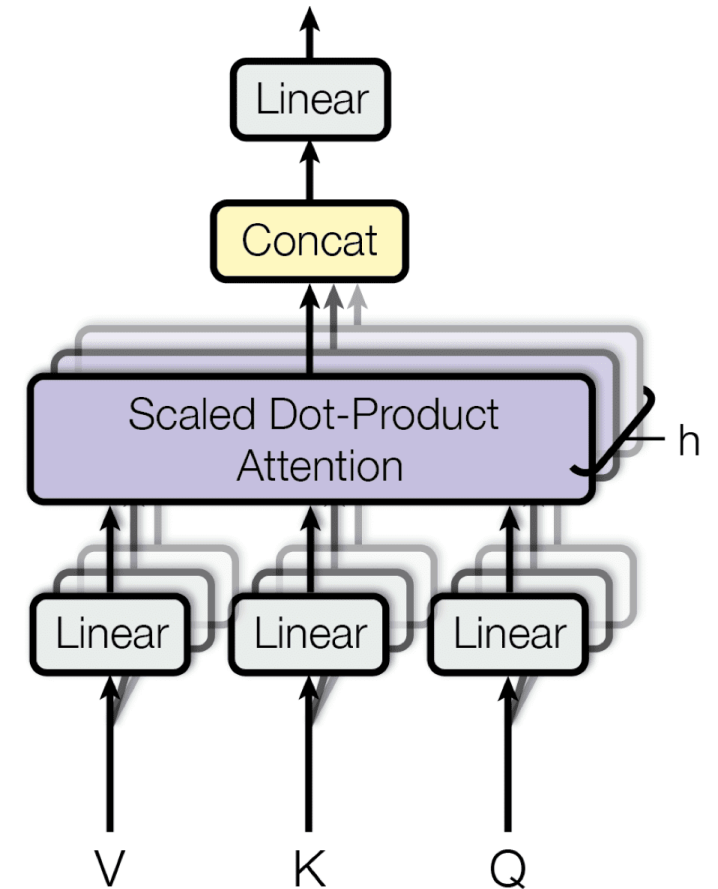
In addition, several layers of attention can be stacked in parallel (**multi-head attention**).



Each **head** works in parallel on different linear transformations of the **same input**, and then the outputs of the heads are **concatenated** to produce the **final attention** result.



Since multiple heads of attention can work **independently** and in parallel, a multi-head model can cause each head to attend to **different elements** of the sequence (**projections**).





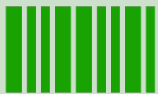
Arquitectura



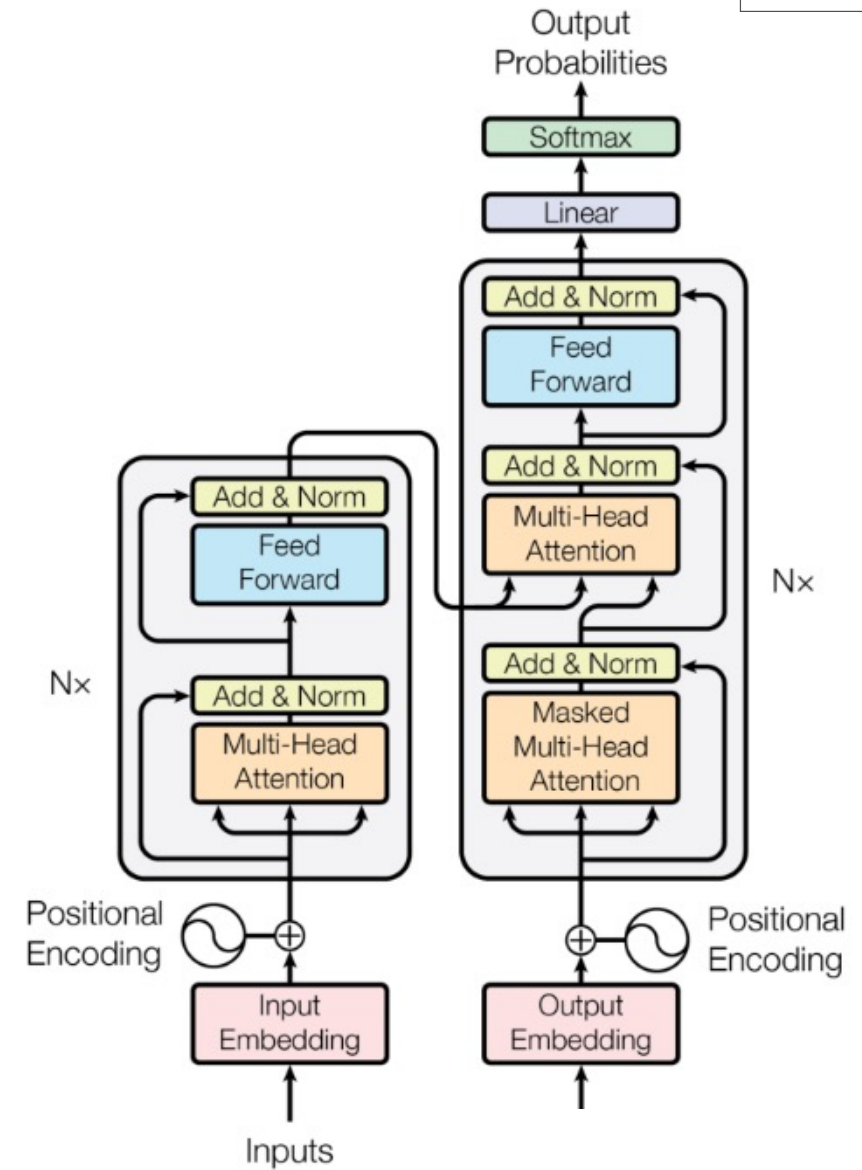
Transformer follows an encoder-decoder architecture but does **not** rely on **recurrence** or **convolutions** to generate an output.



The **encoder's** task (left half) is to **map an input sequence** to a sequence of continuous representations, which is then fed into a decoder.



The **decoder** (right half) **receives** the output of the encoder along with the output of the decoder in the **previous** time step to generate an output sequence.

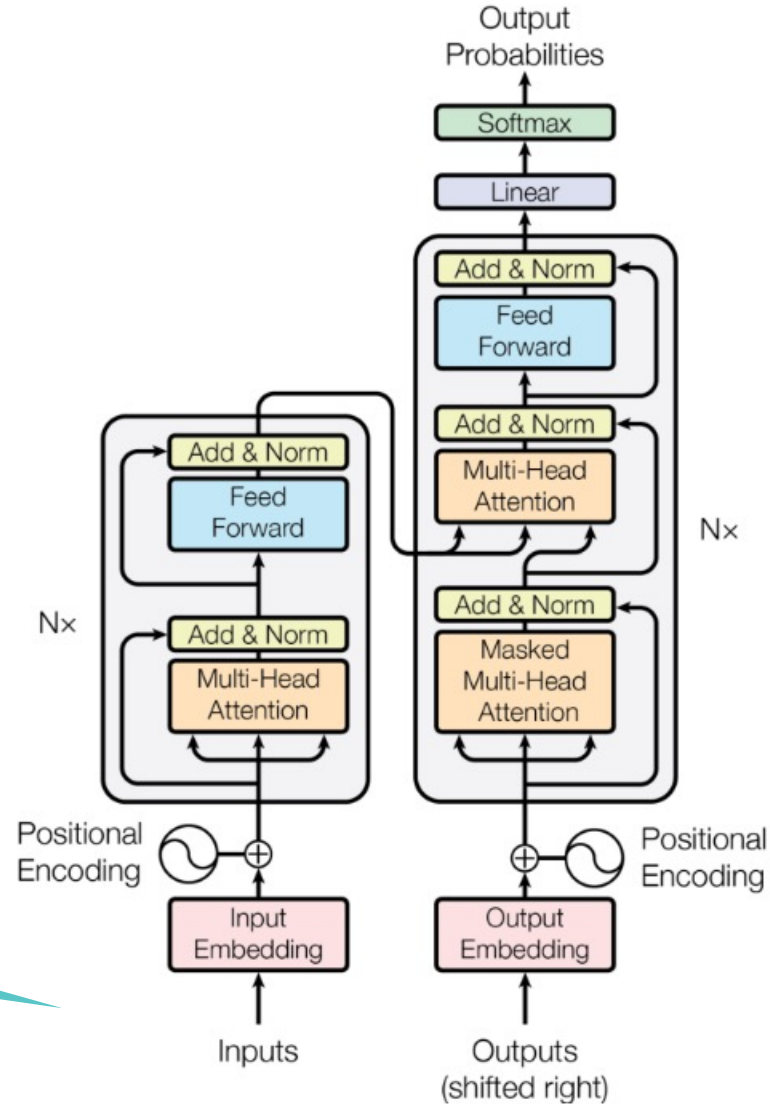




Architecture

Input

We have an input embedding, Transformers assume **all the words at once** (as opposed to the RNN which was word-for-word)



Architecture

Input



TRANSFORMER

↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
El jaguar dormía en la rama del árbol con su cola de soporte.

Architecture

Input



TRANSFORMER

↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
El jaguar dormía en la rama del árbol con su cola de soporte.

All the words come in at once

Architecture

Input



TRANSFORMER

↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
dormía jaguar su la soporte rama del cola árbol con en El de.

All the words enter at the same
time → **Regardless of the Order!!!**

Architecture

Input



TRANSFORMER

↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
dormía jaguar su la soporte rama del cola árbol con en El de.

All the words enter at the same time → **Regardless of the Order!!!**

And how do you take context into account?



Positioning

El **jaguar** dormía en su árbol



Positioning

El jaguar dormía en su árbol			
0,34	0,31	0,45	
-0,44	-0,54	-0,14	
0,19	0,49	0,26	
...
0,84	0,34	0,43	

Vectores de palabras



Positional Encoding

El		jaguar		dormía		en	su	árbol
0,34	1	0,31	2	0,45	3			
-0,44	1	-0,54	2	-0,14	3			
0,19	1	0,49	2	0,26	3			
...	
0,84	1	0,34	2	0,43	3			

Word Vectors

But we add its
position in the text



Positional Encoding

El		jaguar		su	árbol
0,34	1	0,31			
-0,44	1				
0,19	1	0,49			
...
0,84	1	0,34	2	0,43	3

Problem: if we have very large text we would have very high positions

Word Vectors

But we add its **position** in the text



Positional Encoding

El		jaguar		dormía		en	su	árbol
0,34	1/6	0,31	2/6	0,45	3/6			
-0,44	1/6	-0,54	2/6	-0,14	3/6			
0,19	1/6	0,49	2/6	0,26	3/6			
...	
0,84	1/6	0,34	2/6	0,43	3/6			

Solution: **Normalize** the ranking vector by the total number of words



Positional Encoding

Position Vector:
 $4/6 = 0,66$

El jaguar dormía en su árbol



Positional Encoding

El jaguar dormía en su árbol

Position Vector:
 $4/6 = 0,66$

El jaguar dormía

Position Vector:
 $2/3 = 0,66$



Positional Encoding

El jaguar dormía en su árbol

Position Vector:
 $4/6 = 0,66$

El jaguar dormía

Position Vector:
 $2/3 = 0,66$

So, is 0.66 position 2 or 4?



Binary Encoding

El

jaguar

dormía



3 3 3 ... 3 3 3



0 0 0 ... 0 1 1

en

su

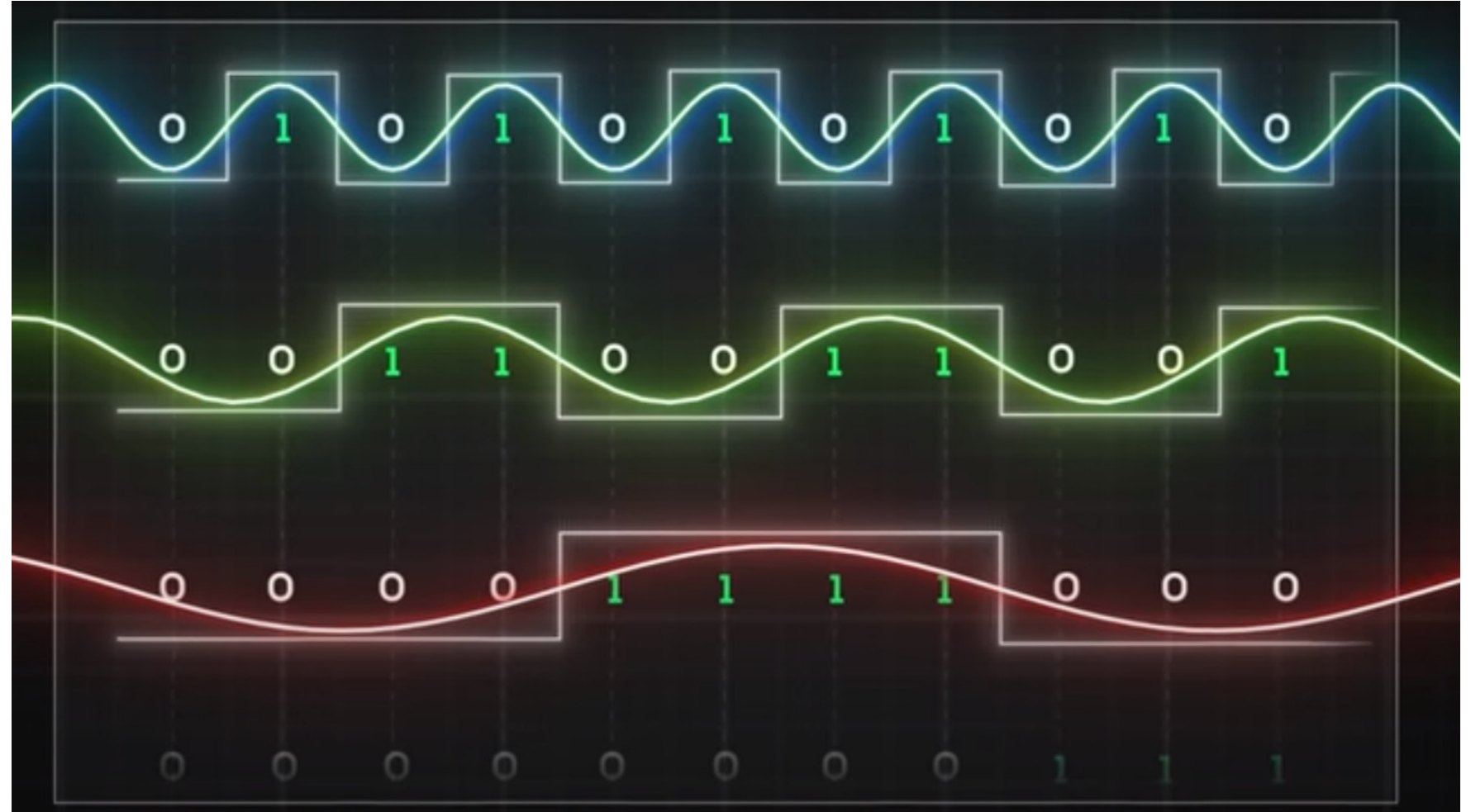
árbol

Binary Encoding



Binary Encoding

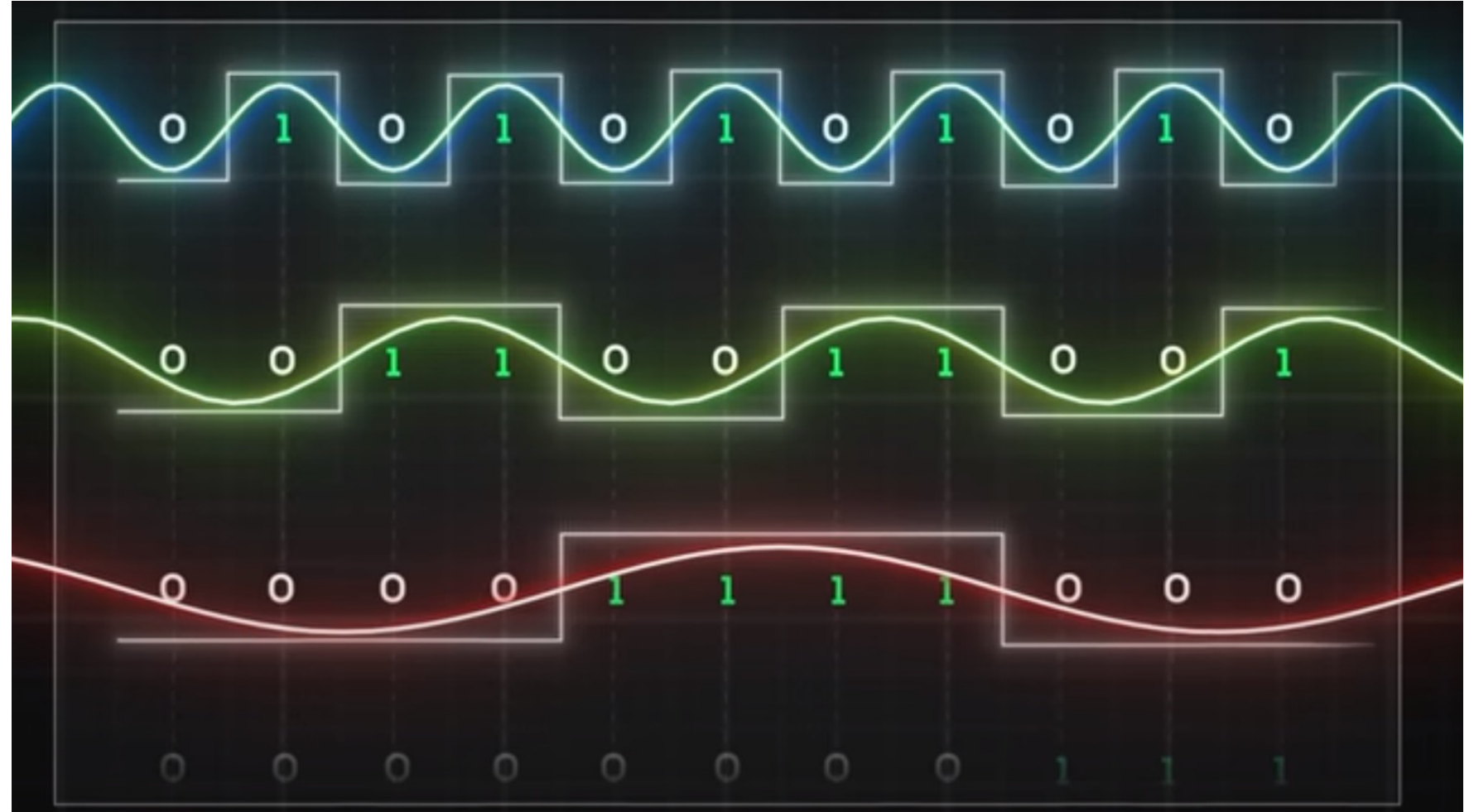
Translates to a
Sinusoidal function





Binary Encoding

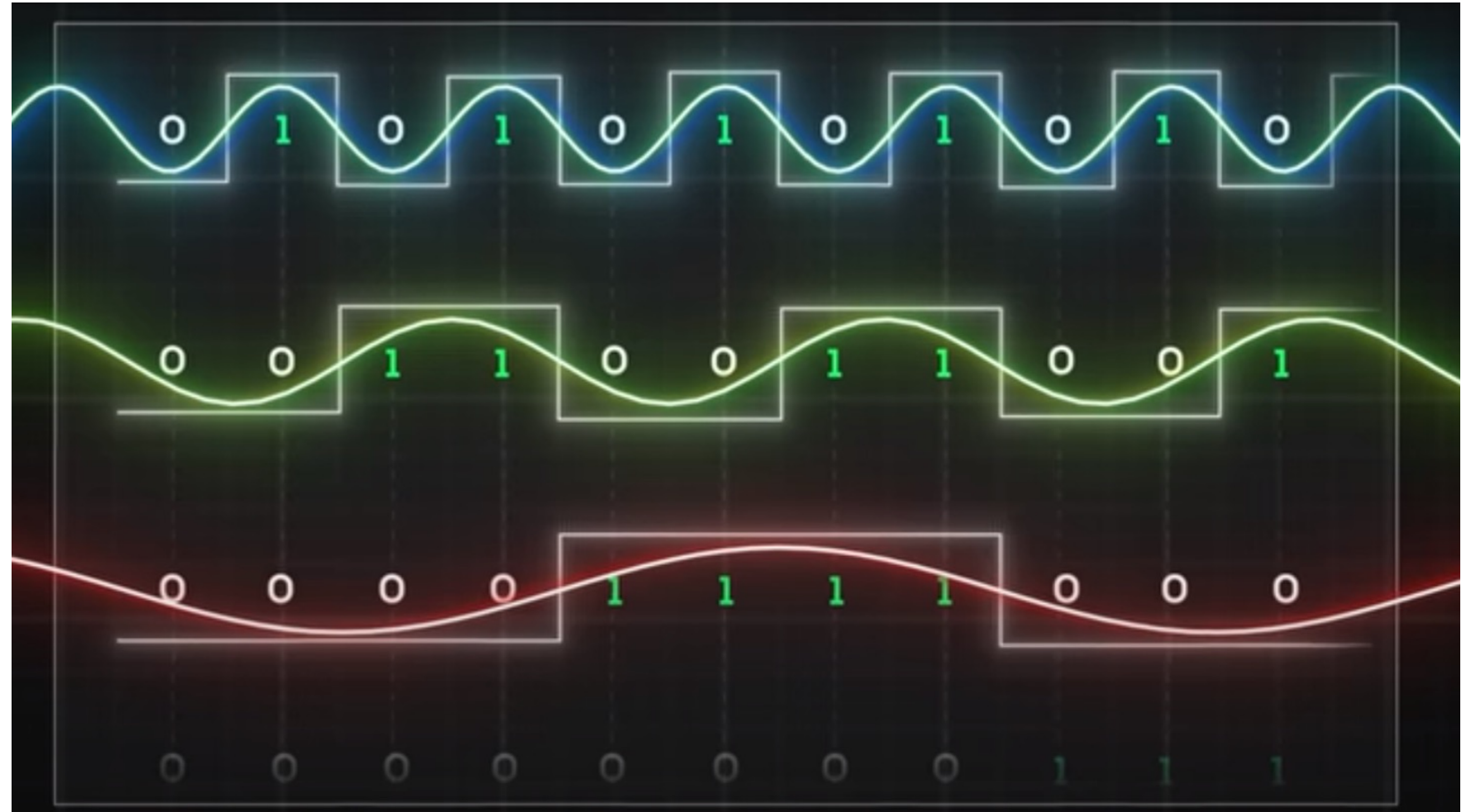
$\text{Sen}(1/1 * \text{pos})$





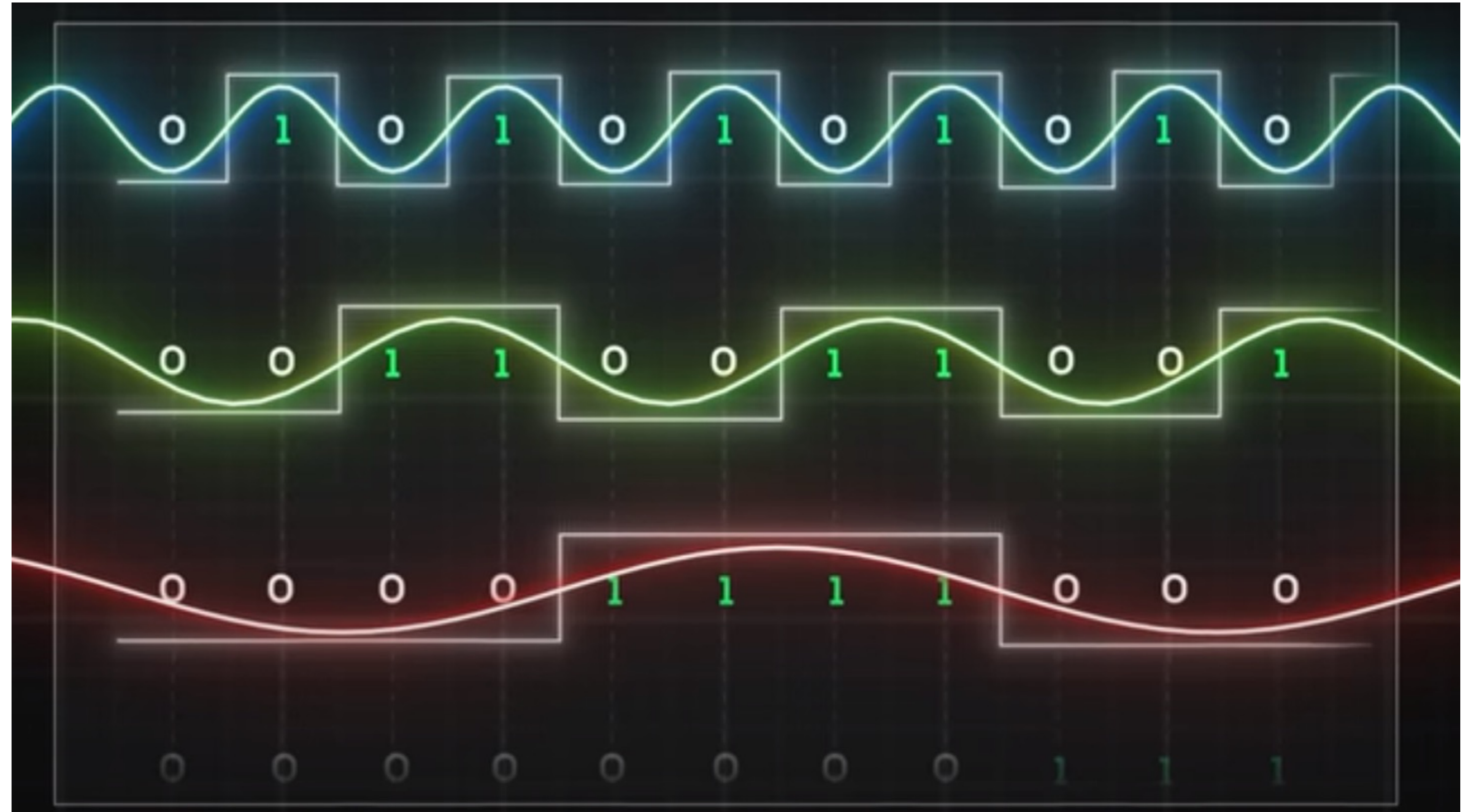
Binary Encoding

Sen($\frac{1}{2} * \text{pos}$)





Binary Encoding



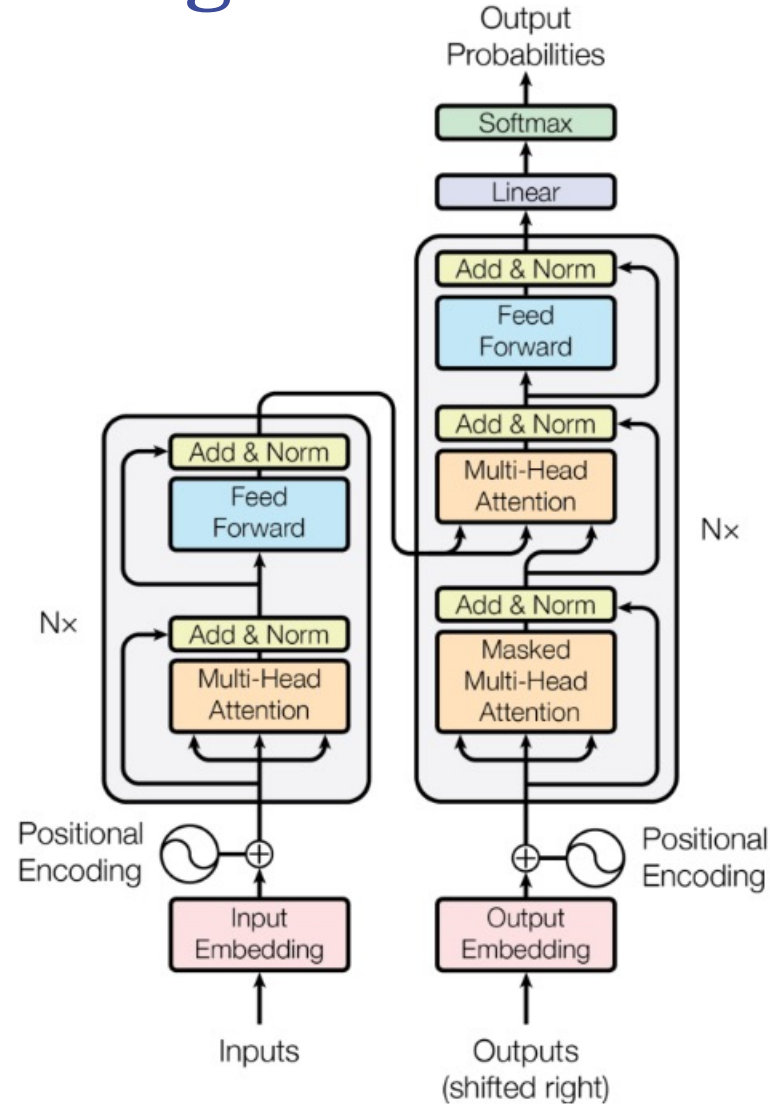
Sen($\frac{1}{3} * \text{pos}$)



Architecture

Positional Encoding

This is!



Architecture

Input

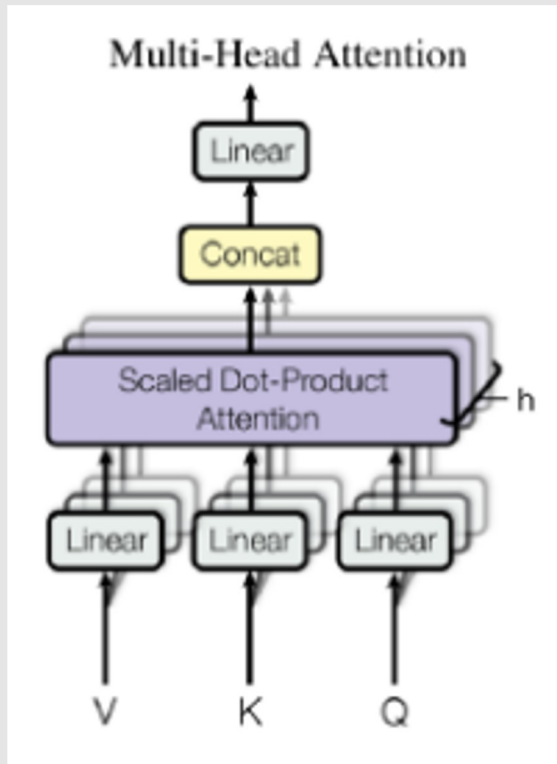


TRANSFORMER

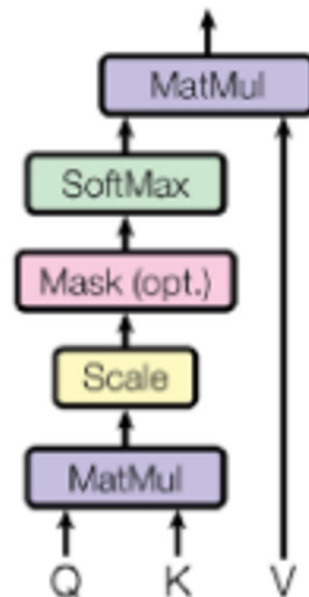
↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
dormía jaguar su la soporte rama del cola árbol con en El de.

And by passing all the words at the same time and regardless of the order → can be **parallelized!!**

Transformer – Encoder

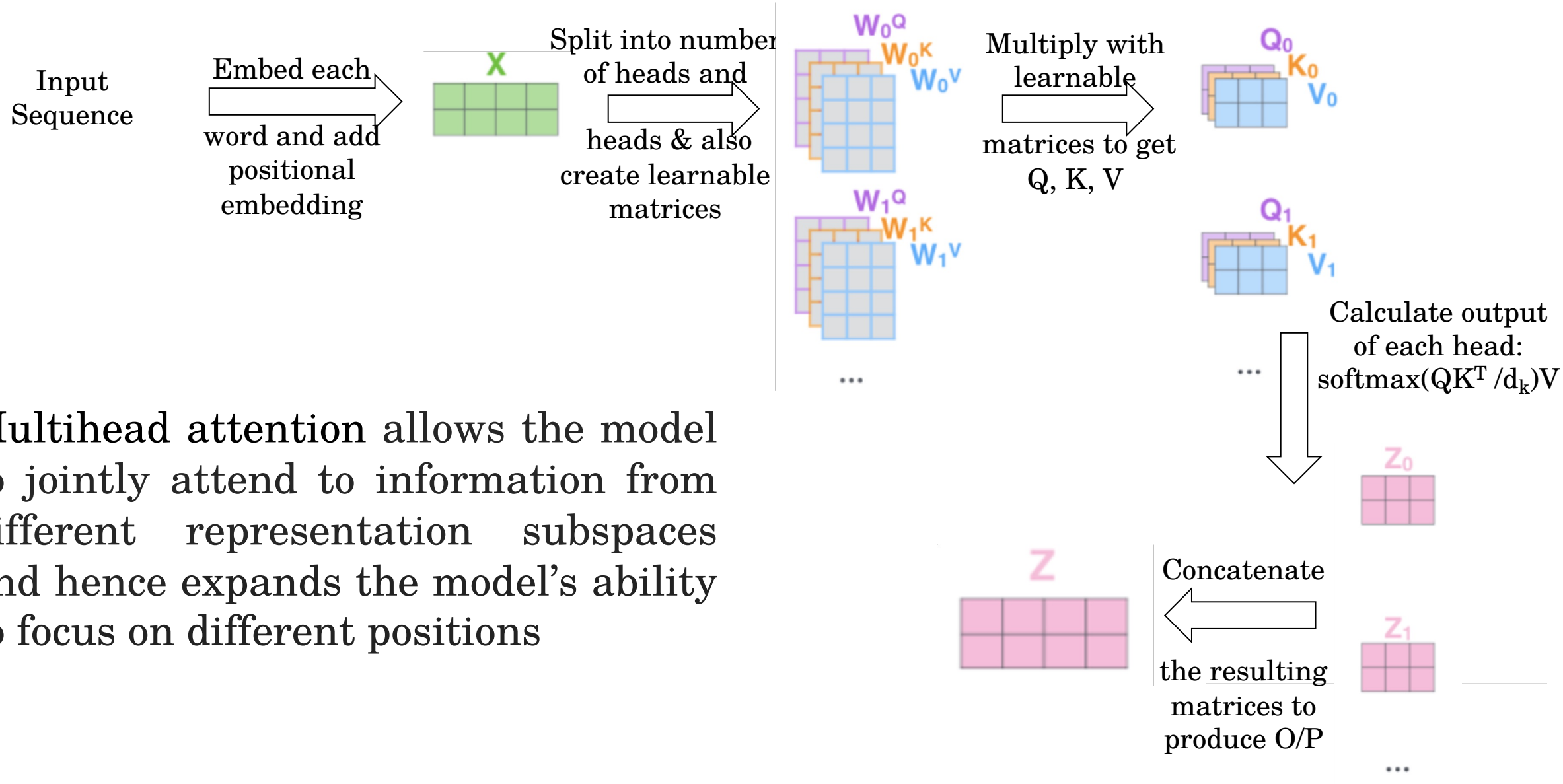


Scaled Dot-Product Attention




- Like LSTM, Transformer is an architecture for transforming one sequence into another one with the help of two parts (Encoder and Decoder), but it differs from the existing seq2seq models because it does not imply any Recurrent Networks (GRU, LSTM, etc.).
- 3 main parts : Position Embeddings + Multi Head Attention + Feed- forward Layers.
- Q (query) matrix : vector representation of one word in the sequence
- K (keys) matrix : vector representations of all the words in the sequence
- V (values) matrix : vector representations of all the words in the sequence. For the encoder, V consists of the same word sequence than Q.
- Attention weights = $\text{softmax}(QK^T/d_k)$
- These weights are defined by how each word of the sequence (represented by Q) is influenced by all the other words in the sequence (represented by K).
- Those weights are then applied to all the words in the sequence that are introduced in V.

Multi-Head Attention



Multihead attention allows the model to jointly attend to information from different representation subspaces and hence expands the model's ability to focus on different positions



Vision Transformer (ViT)



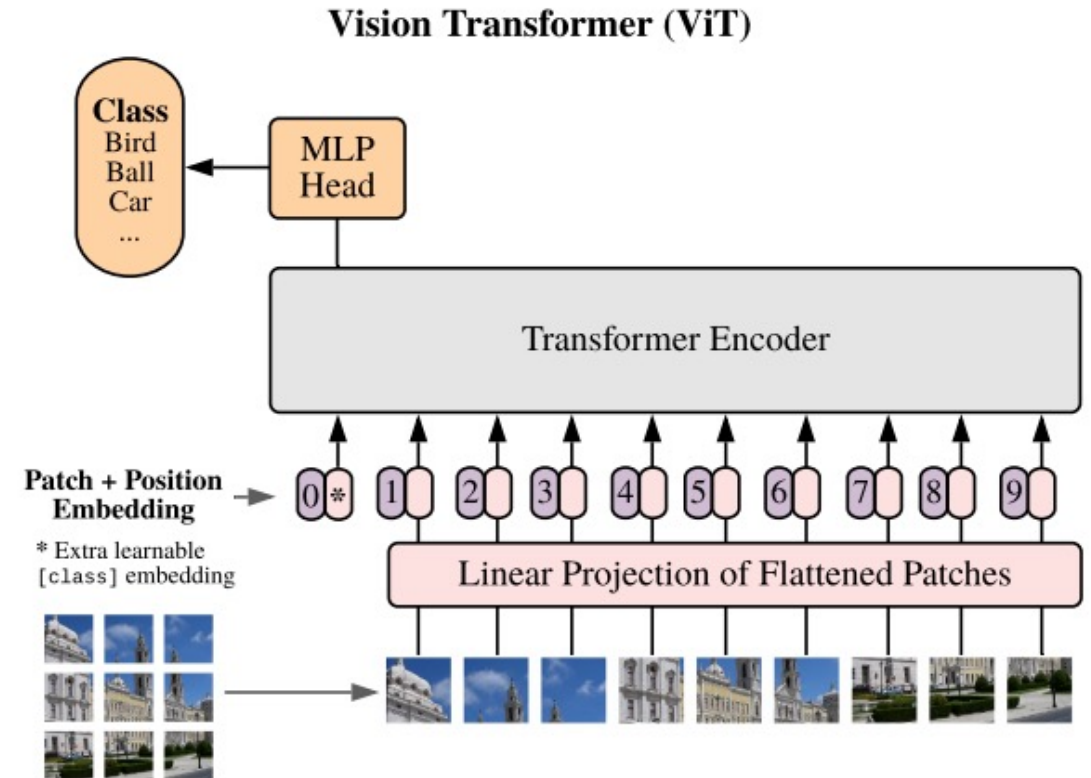
ETS de
Ingeniería
Informática



UNED

Background

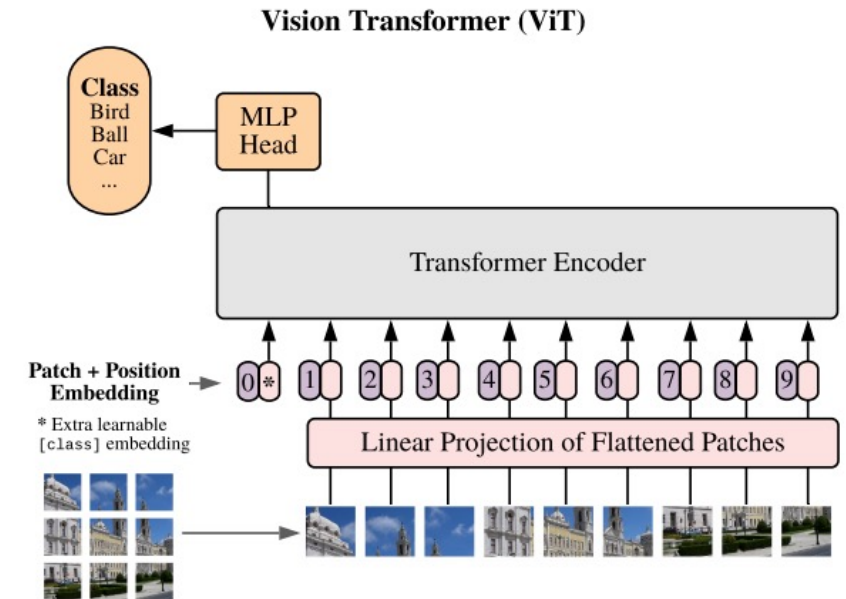
- The input data to Transformer is provided in the form of **two-dimensional images**.
- The input image, with height (H), width (W), and channels (C), is divided into smaller two-dimensional **patches**.
- This results in an $N=HW/P^2$ **number of patches**, where each patch has a resolution of $P \times P$ pixels.



ViT Operations

- Each image patch is **flattened** into a **vector** of \mathbf{x}_p^n length $P \times C$ where, $n=1, \dots, N$.
- A sequence of embedding image patches is generated by assigning the flattened patches to D **dimensions**, with a **trainable linear projection** E .
- An x_{class} class is added to the sequence of embedding images. The value of x_{class} represents the result of the y classification.
- Embedding patches are eventually augmented with one-dimensional E_{pos} positional embedding, thus introducing **positional information** into the input, which is learned during training.
- The resulting sequence of embeddings vectors:

$$\mathbf{z}_0 = [\mathbf{x}_{class}; \mathbf{x}_p^1 \mathbf{E}; \dots; \mathbf{x}_p^N \mathbf{E}] + \mathbf{E}_{pos}$$

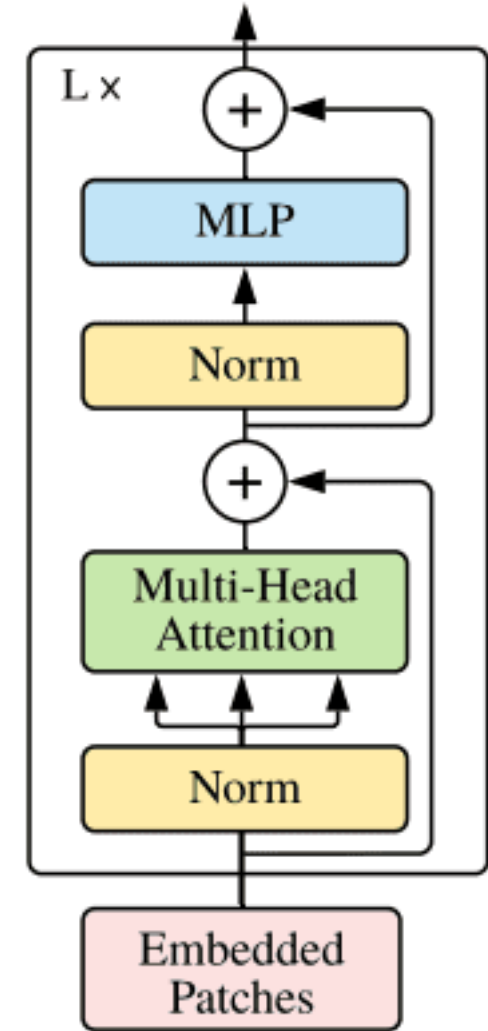




ViT Operations

- To perform the classification, they feed z_0 into the Transformer's encoder input, which consists of a stack of identical layers.
- Then, they proceed to take the value of x_{class} at the L-layer of the encoder output and feed it into a classification MLP head.
- The classification MLP implements Gaussian error linear unit (GELU) nonlinearity.
- ViT uses the Encoder part of the original Transformer architecture.

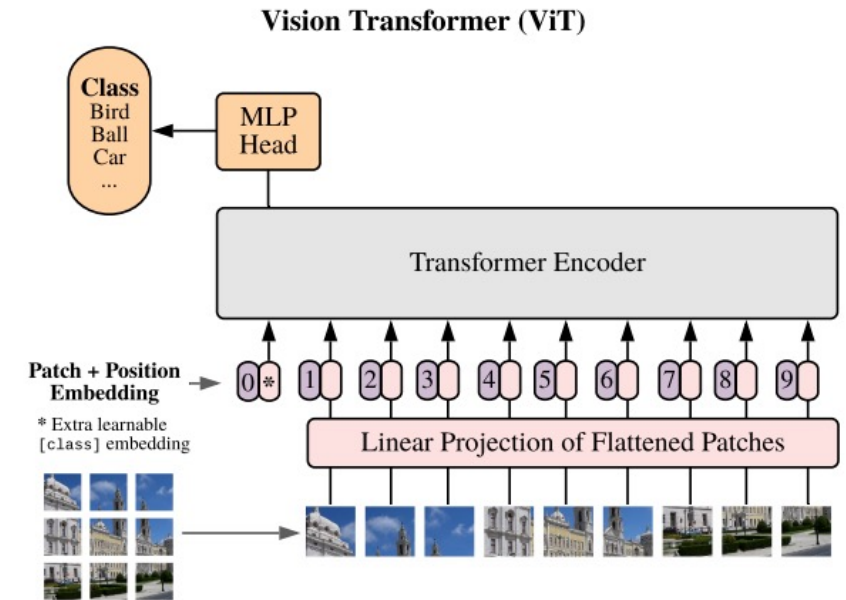
Transformer Encoder



$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}}; \mathbf{x}_p^1 \mathbf{E}; \dots; \mathbf{x}_p^N \mathbf{E}] + \mathbf{E}_{\text{pos}}$$

Operating Mechanism

- The input to the encoder is a sequence of **patches** of embedded images, which is also augmented with **positional information**.
- A classification head follows the output of the Encoder receives the value of the **embedded class** that can be learned to generate a classification output based on its state.
- Compared to a convolutional layer, ViT does **not** generate **separate feature maps** for the entire image. Instead, each patch in the image will become an embedding in which several features are rendered.
- Alternatively, the original image can be **fed into a CNN** before being passed to the Encoder. The sequence of image patches would then be obtained from the CNN feature maps, while the subsequent process of embedding the feature map patches, adding a class token and augmenting with positional information remains the same.





ViT Vs ResNet

Exp. 1: Fine-tuning and testing on ImageNet



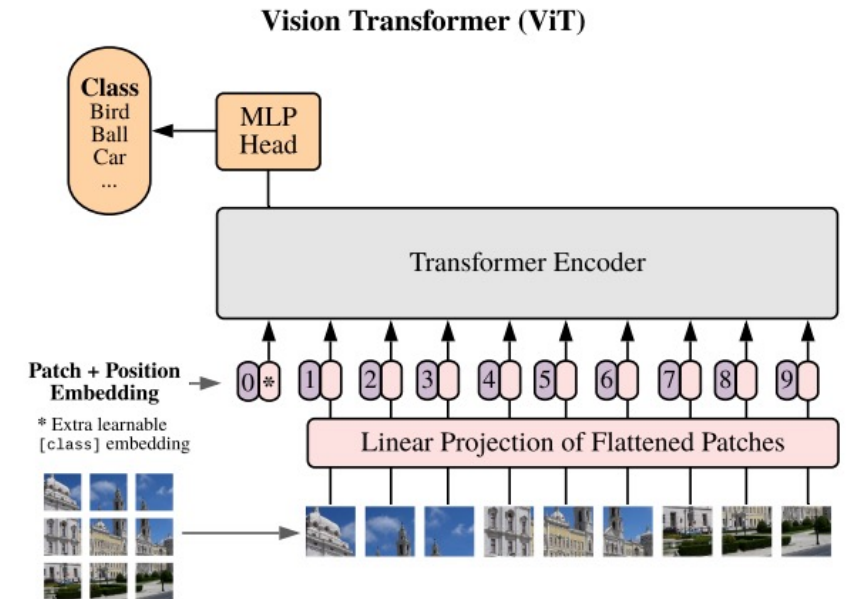
In the smaller dataset (ImageNet), the two largest ViT models underperformed compared to their **smaller counterpart**. The performance of all ViTs remains below that of ResNets.



When trained on a larger dataset (ImageNet-21k), the three ViT models performed **similarly** to each other, as well as to the ResNets.



When trained on the largest dataset (JFT-300M), the performance of the larger ViT models **exceeds** the performance of the smaller ViT and ResNets.



$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}}; \mathbf{x}_p^1 \mathbf{E}; \dots; \mathbf{x}_p^N \mathbf{E}] + \mathbf{E}_{\text{pos}}$$



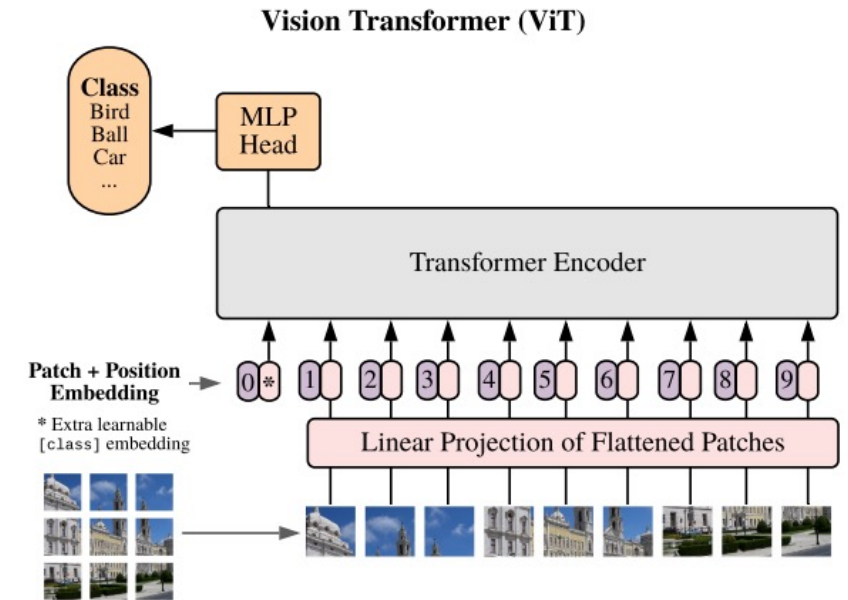
ViT Vs ResNet

Exp. 2: Investigate the effect of the dataset size

Training on random subsets of different sizes of the JFT-300M dataset and testing on ImageNet to further investigate the effect of dataset size:

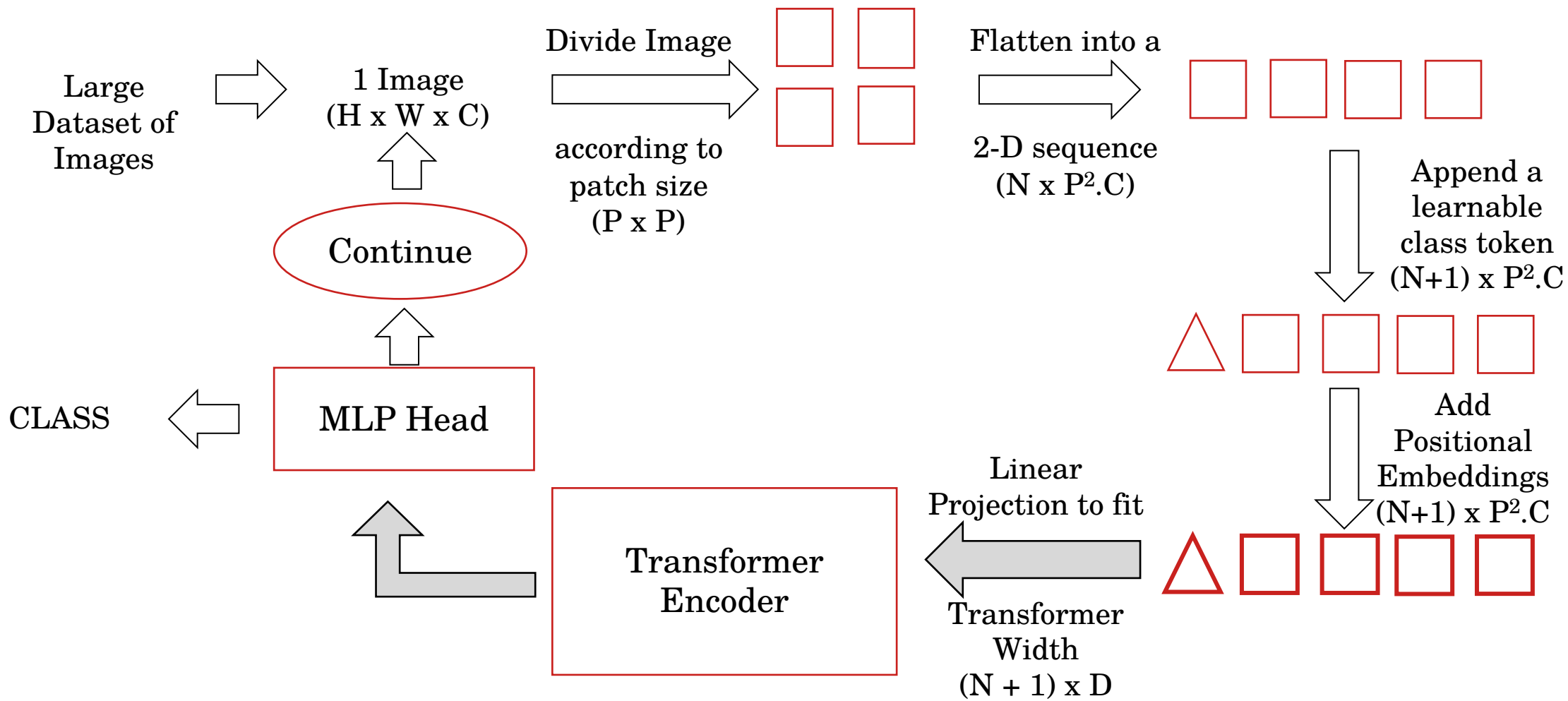
In smaller subsets of the dataset, ViT models **are more fine-tuned** than ResNet and have significantly lower performance.

In the **largest subset** of the dataset, the performance of the largest ViT model **exceeds** the performance of the ResNets.



$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}}; \mathbf{x}_p^1 \mathbf{E}; \dots; \mathbf{x}_p^N \mathbf{E}] + \mathbf{E}_{\text{pos}}$$

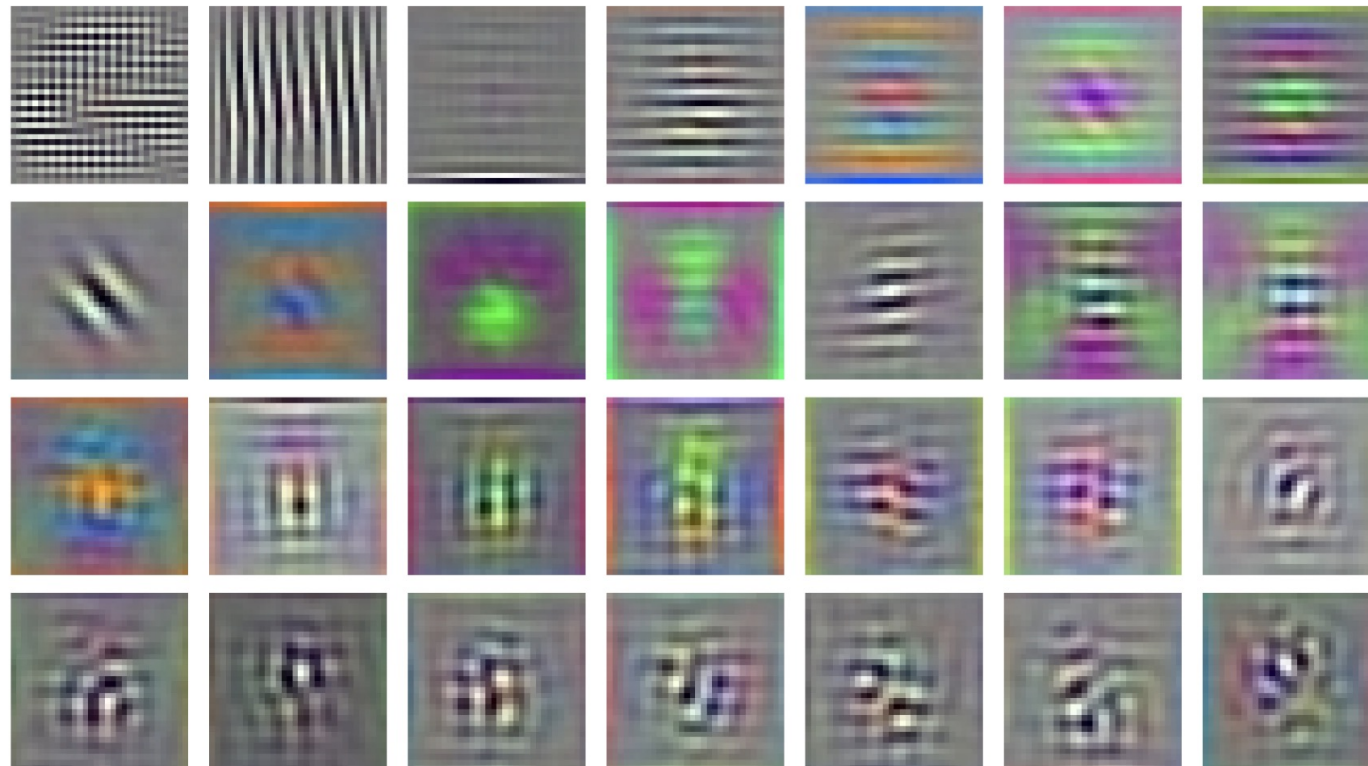
Methodology





Internal Data Representation

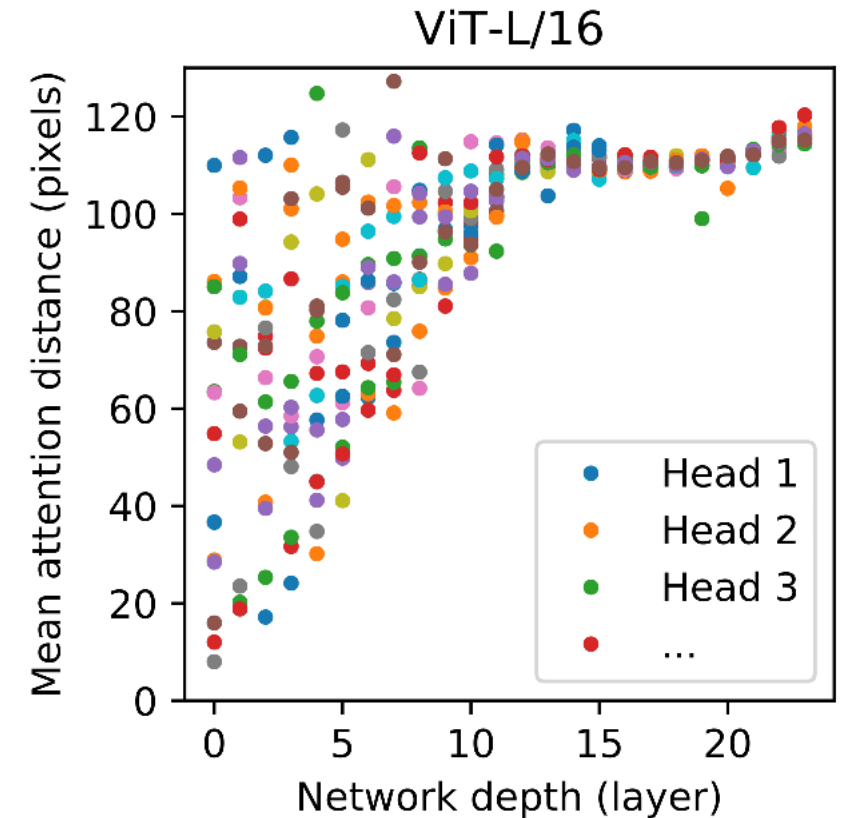
- The learned embedding filters that are initially applied to image patches in the first ViT layer resemble **basic functions** that can extract the low-level features within each patch





Internal Data Representation

- Multiple multi-head attention in the lower layers of the model already cater for most of the information in the image (based on their attention weights), demonstrating the ability of the self-attention mechanism to integrate information **throughout the image**.
- The image shows the size of the area of the image served by different head self-attention
- We notice that each head specializes in different patches.





Inference

ETS de
Ingeniería
Informática



UNED

Inference from our Results

- The size of the patch in the ViT decides the length of the sequence. **A smaller patch size leads to a greater exchange of information during the self-care mechanism.**
- This is proven by the best results obtained by using a smaller patch size, 4 out of 8, in a 32x32 image.
- Increasing the number of ViT layers should ideally lead to better results, but the results of the 8-layer model **are slightly better** than those of the 12-layer model, which can be attributed to the small datasets used to train the models.
- More **complex** models require **more data** to capture the characteristics of the image.
- **Pretrained ViT** performs better than methods because it has been trained on **huge data sets** and therefore has learned better.



Conclusions



ETS de
Ingeniería
Informática

UNED



Vision Transformers

Bibliografy

Original paper:

- [An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale](#)

ViT Variants:

- [Swin Transformer: Hierarchical Vision Transformer using Shifted Windows](#)
- [Training data-efficient image transformers & distillation through attention](#)
- [Tokens-to-Token ViT: Training Vision Transformers from Scratch on ImageNet](#)
- [DeepViT: Towards Deeper Vision Transformer](#)
- [MViTv2: Improved Multiscale Vision Transformers for Classification and Detection](#)

Explicabilidad:

- [LeGrad: An Explainability Method for Vision Transformers via Feature Formation Sensitivity](#)

¡Gracias!



Dr. Manuel Castillo-Cara

www.manuelcastillo.eu

**Departamento de Inteligencia Artificial
Escuela Técnica Superior de Ingeniería Informática
Universidad Nacional de Educación a Distancia (UNED)**