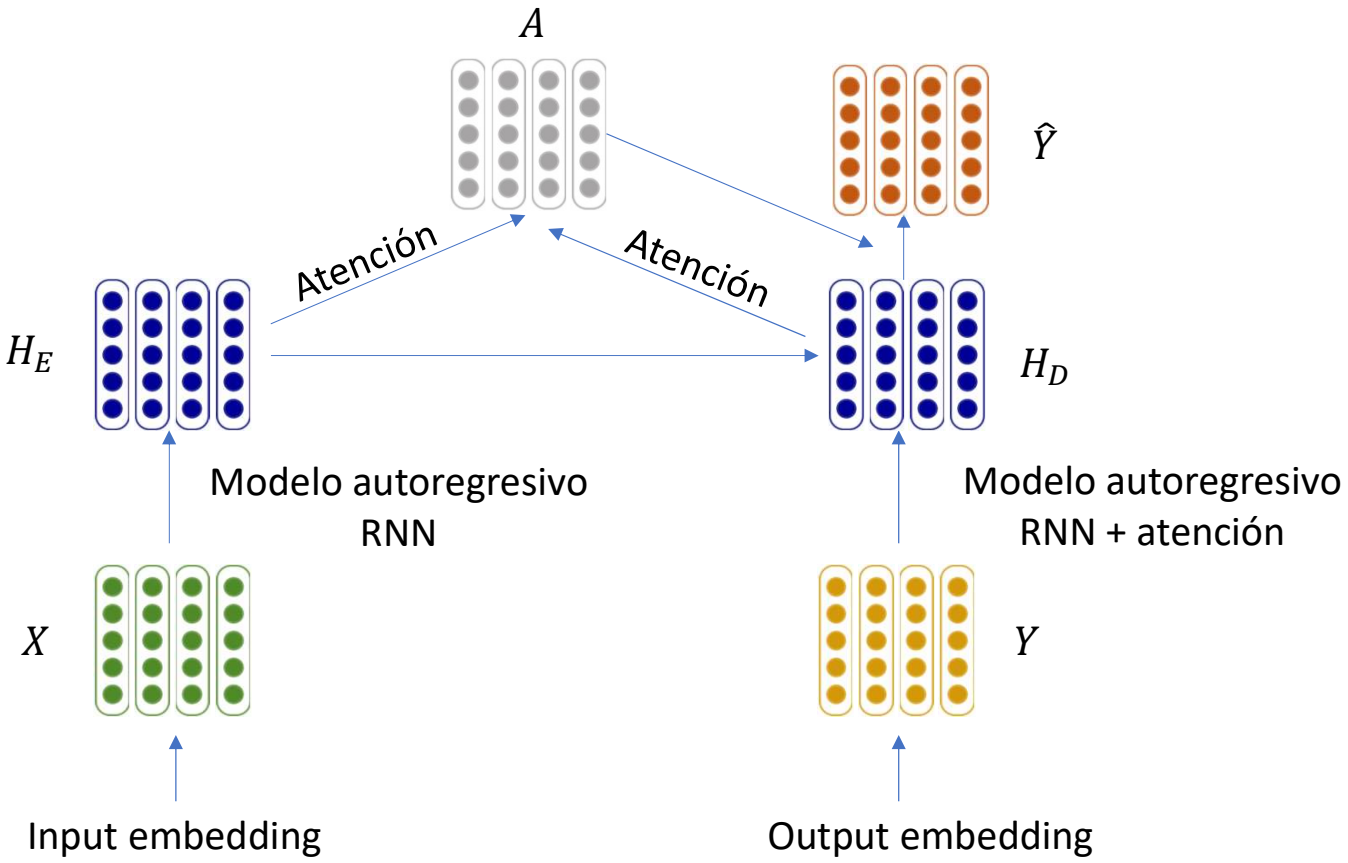
Deep learning

Transformers

Neural Machine Translation + Attention

Input: El bebé está llorando
Output: The baby is crying

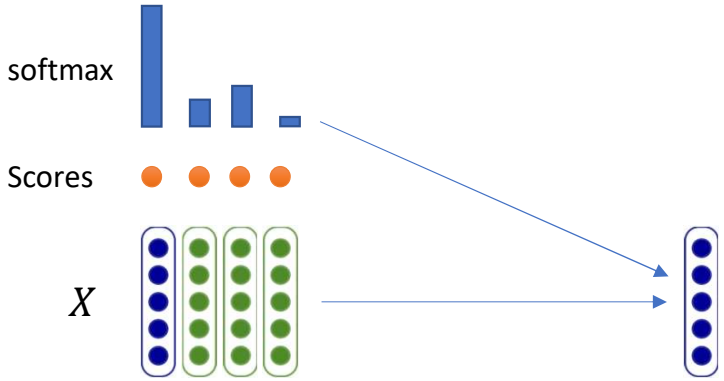


Dados dos conjuntos de vectores (que representan algo, por ejemplo una secuencia), computar nuevos vectores combinados que traten de encontrar asociaciones entre los datos originales.

Self-Attention

Se podrá aplicar atención sobre el mismo conjunto de vectores?

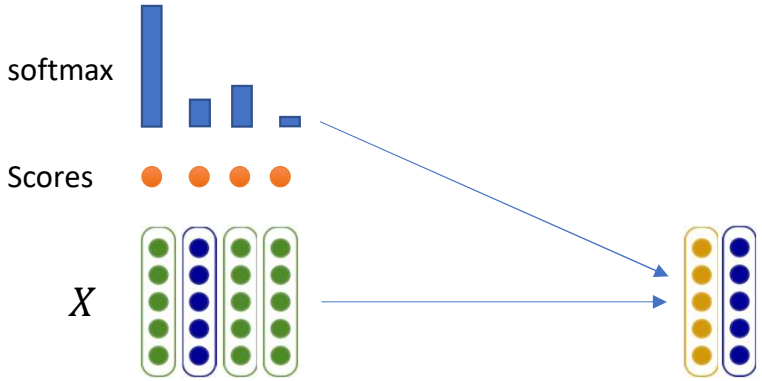
Cada vector de la colección aplica atención sobre los vectores de la colección de entrada y luego combina las atenciones para construir nuevos vectores.



Self-Attention

Se podrá aplicar atención sobre el mismo conjunto de vectores?

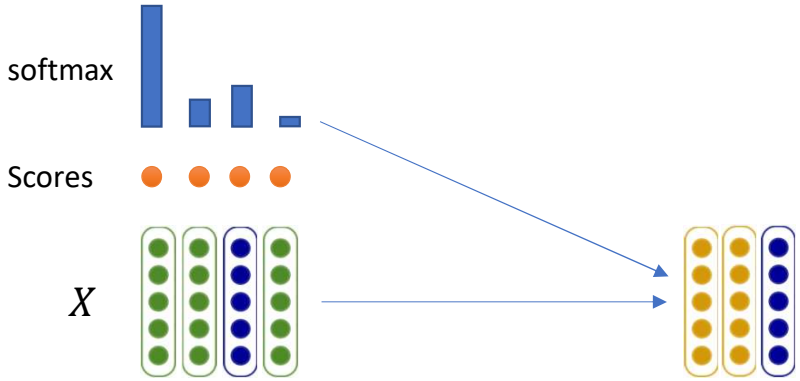
Cada vector de la colección aplica atención sobre los vectores de la colección de entrada y luego combina las atenciones para construir nuevos vectores.



Self-Attention

Se podrá aplicar atención sobre el mismo conjunto de vectores?

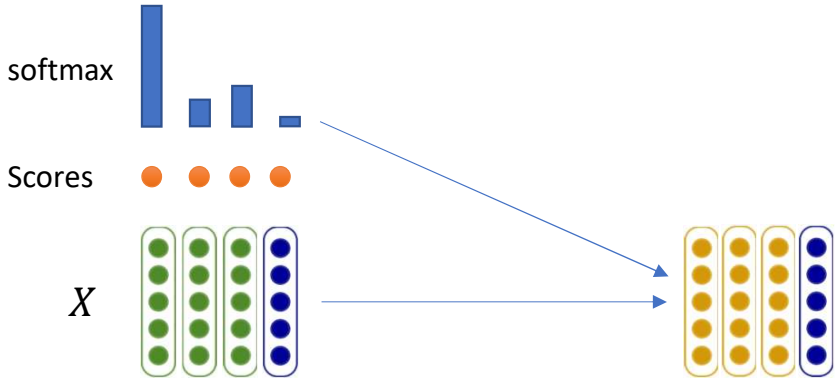
Cada vector de la colección aplica atención sobre los vectores de la colección de entrada y luego combina las atenciones para construir nuevos vectores.



Self-Attention

Se podrá aplicar atención sobre el mismo conjunto de vectores?

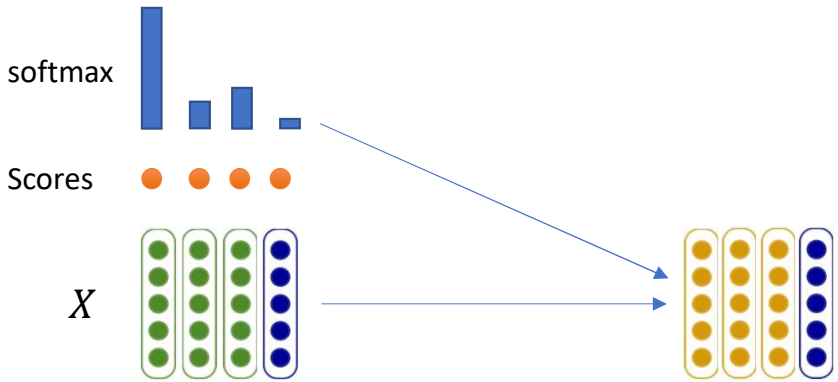
Cada vector de la colección aplica atención sobre los vectores de la colección de entrada y luego combina las atenciones para construir nuevos vectores.



Self-Attention

Se podrá aplicar atención sobre el mismo conjunto de vectores?

Cada vector de la colección aplica atención sobre los vectores de la colección de entrada y luego combina las atenciones para construir nuevos vectores.



Podemos generar nuevos datos a través de la atención
Ya no necesitamos la recurrencia!
Asumimos que los datos vienen en matrices de dimensión $seq \times dim$
Todo se puede calcular en paralelo, así:

$$H = softmax(XX^T)X$$

Pero parece no tener mucho sentido aplicar la auto-atención. Es muy posible llegar a la transformación de identidad entre X y H

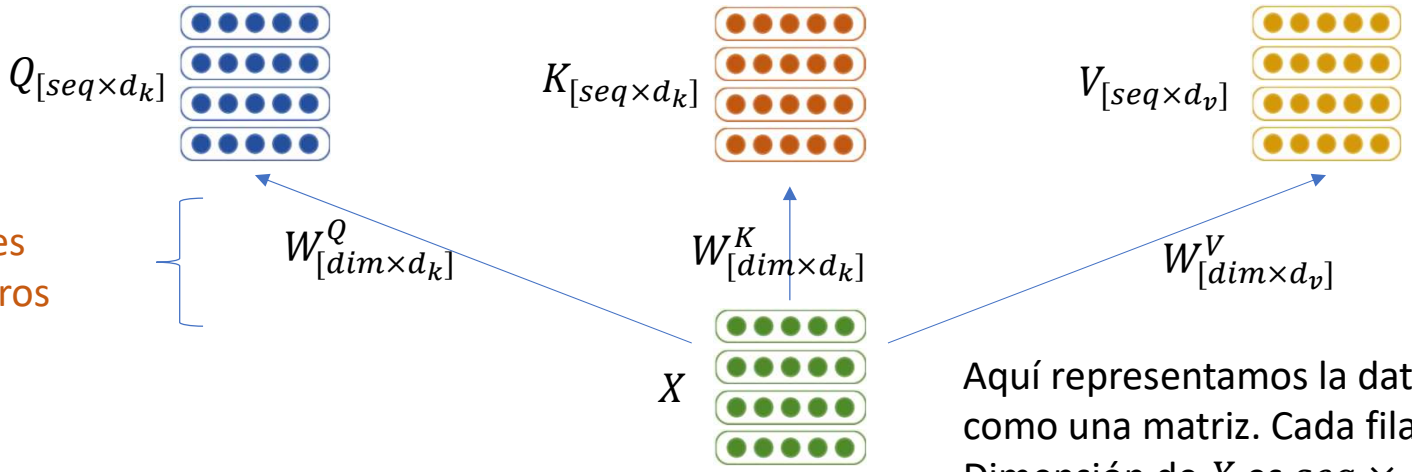
Self-Attention

Solución: Agregar un nivel de abstracción a los vectores que se usan en la auto-atención

$$H = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

QK^T es una matriz cuadrada de scores
"similitud entre cada par de embeddings de la secuencia"

El resultado es $H_{[dim \times d_v]}$



Aquí representamos la data de entrada como una matriz. Cada fila es un word embedding.
Dimensión de X es $seq \times dim$

Multi-head Self-attention

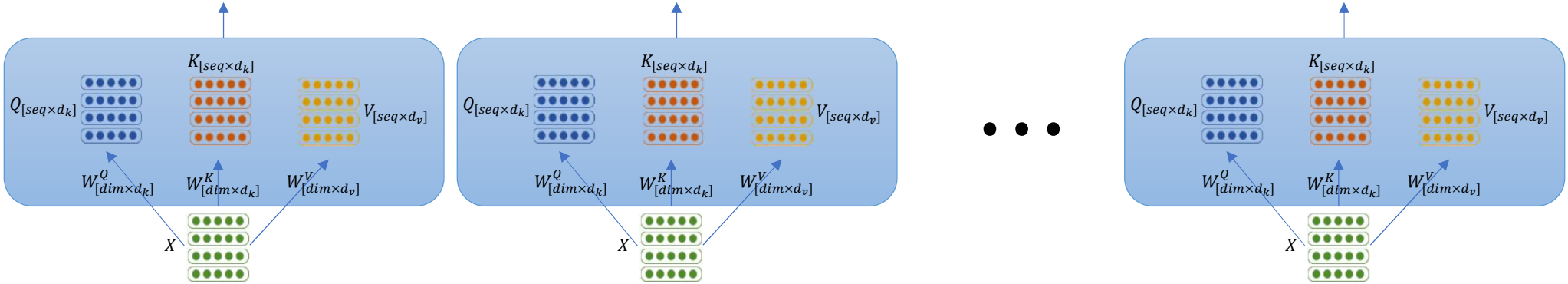
Auto-atención es una transformación de los datos de entrada. Para potenciar su uso, se usa varias veces, esperando que cada aplicación extraiga diferente información.

$$H^{output}_{[seq\times dim]} = H_{[seq\times(h\times d_v)]} W^{O}_{[(h\times d_v)\times dim]}$$

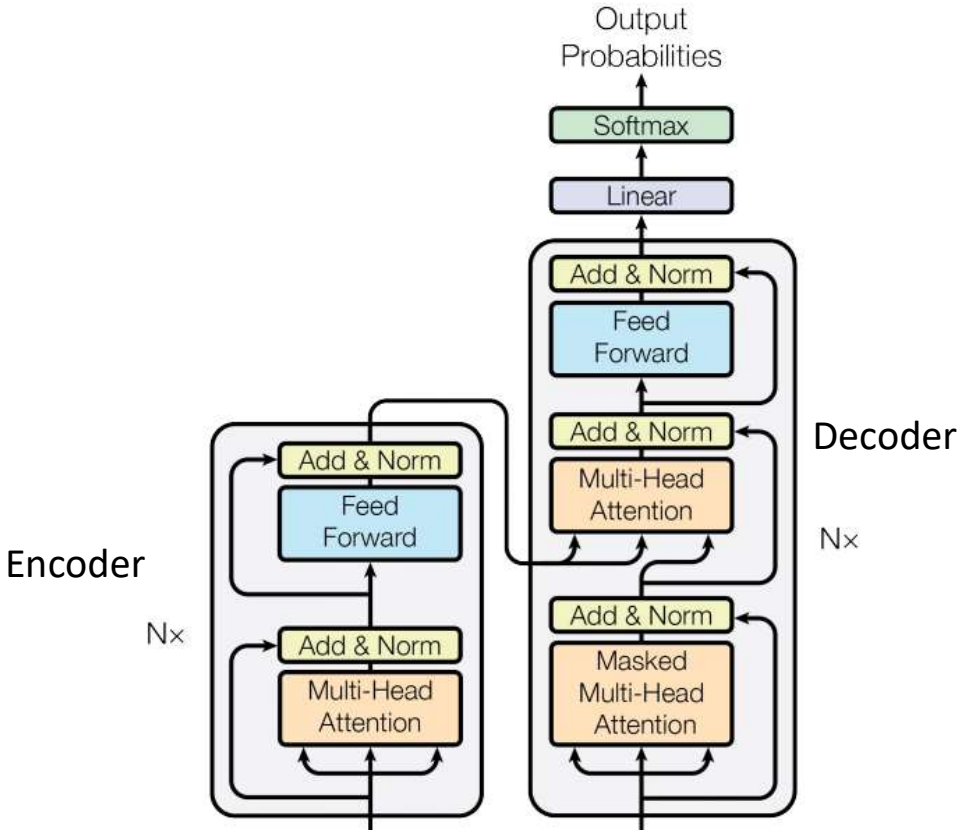
Capa lineal: hace que la salida tenga el mismo tamaño que el input

$$H_{[seq\times(h\times d_v)]} = [H^1_{[seq\times d_v]}, H^2_{[seq\times d_v]}, \dots, H^h_{[seq\times d_v]}]$$

Concatenación



Transformer



Encoder

- Shortcuts y normalización después de cada atención multi-cabeza
- MLP de dos capas (con ReLU)
- Este bloque se repite N veces

Decoder

- Self-attention sobre la secuencia target
- Atención sobre la salida del encoder
- Shortcuts y normalización
- MLP al finalizar el bloque

Un MLP y softmax al final

Transformer

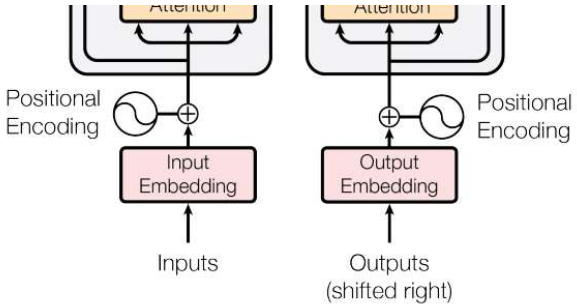
Qué sucede con el orden de la secuencia en esta arquitectura?
Hemos perdido información del orden temporal

Solución: positional encoding

- Agregar información al embedding inicial para saber en qué parte de la secuencia sucede

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

- La propuesta original es usar funciones periódicas en función de posición y dimensión del embedding

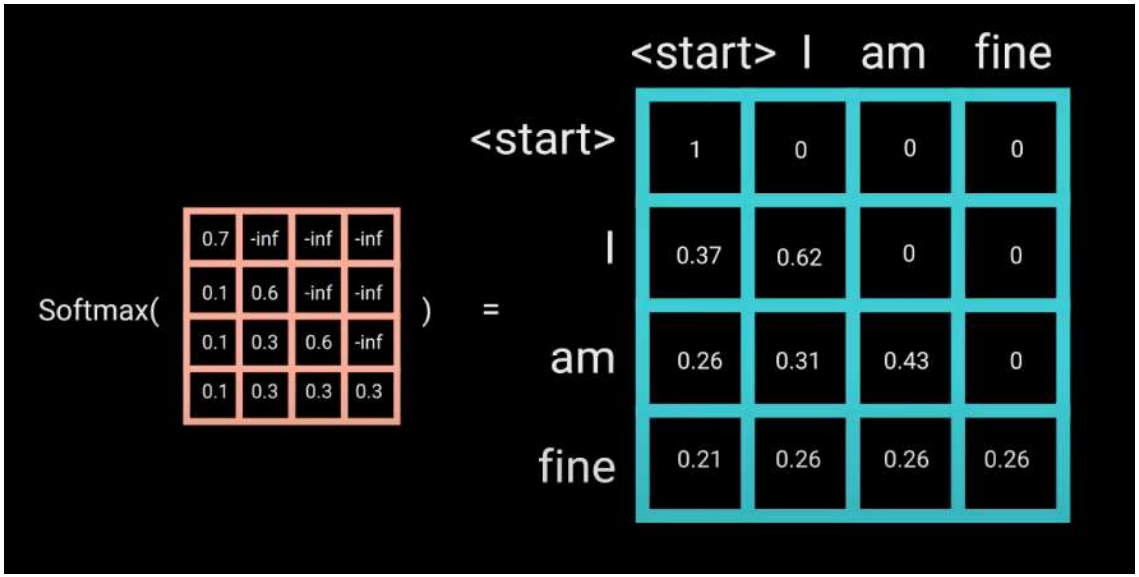


Transformer

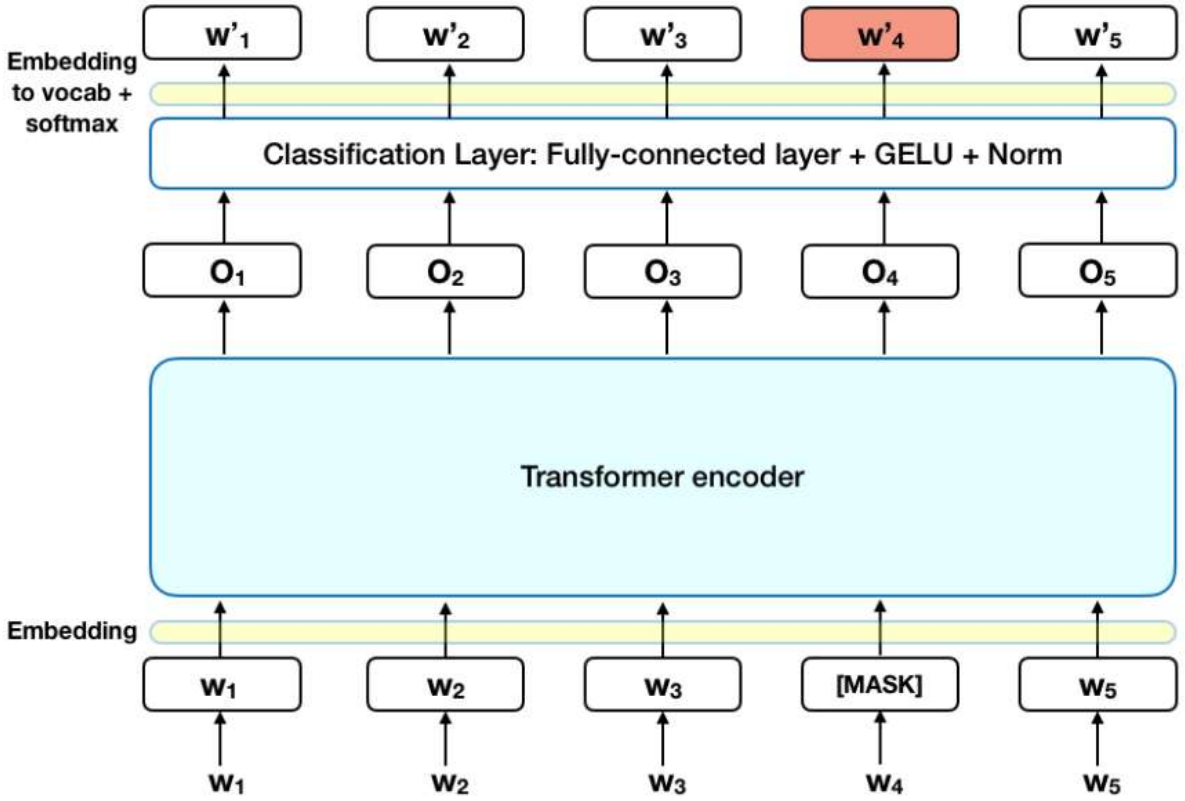
Un detalle más: el decoder debería aprender a poner atención sólo a los tokens que ocurrieron hasta un determinado momento. Al momento de hacer inferencia, no se conoce el futuro.

Esto se puede resolver fácil en el mismo cómputo de la auto-atención.

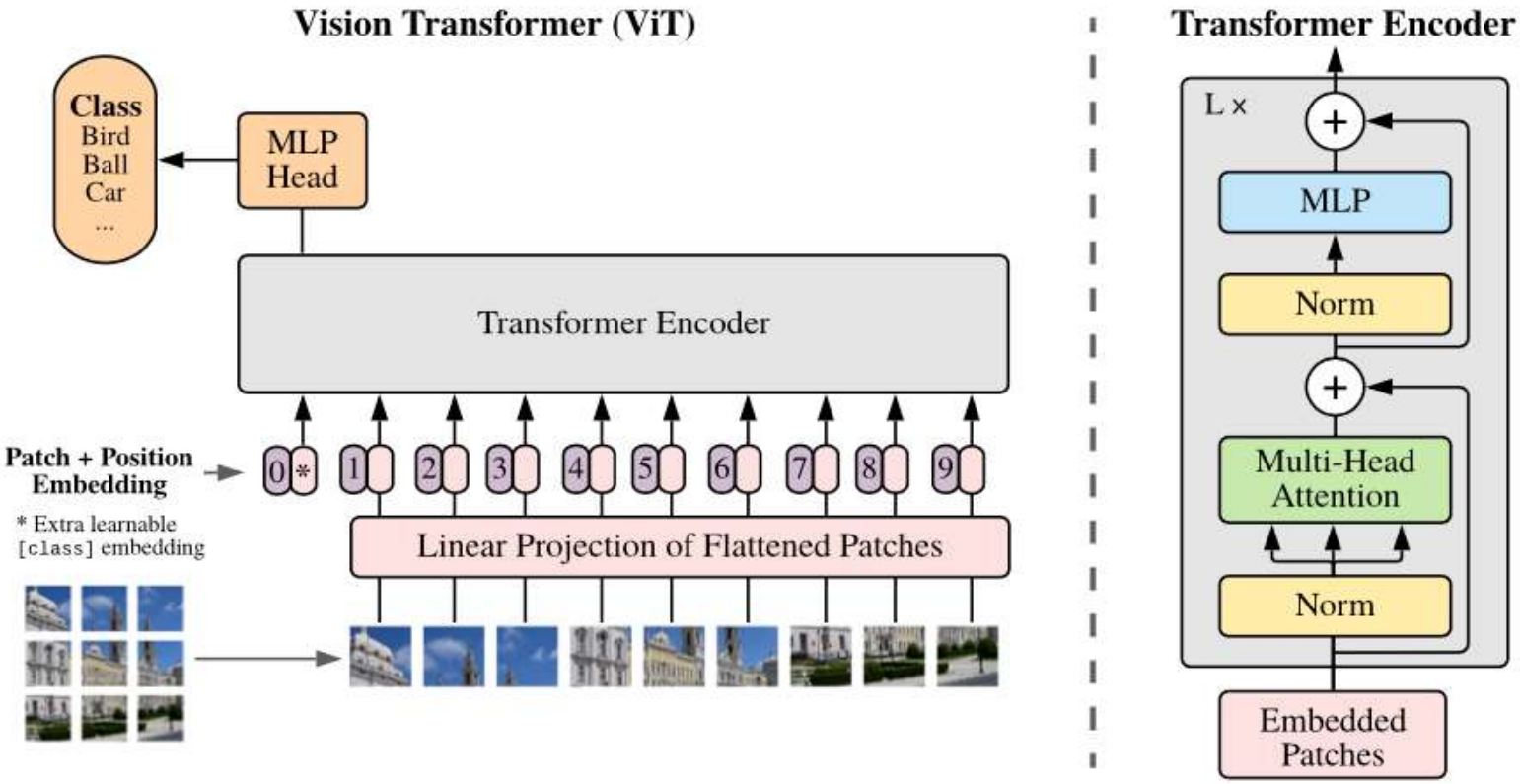
Usar a propósito un score de atención $-inf$ para posiciones futuras. Softmax se hará cargo de poner en cero los scores de atención.



BERT


w'1
w'2
w'3
w'4
w'5
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
w1
w2
w3
[MASK]
w5
w1
w2
w3
w4
w5

ViT


Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
0 * 1 2 3 4 5 6 7 8 9
Linear Projection of Flattened Patches
Transformer Encoder
L x
MLP
Norm
Multi-Head Attention
Norm
Embedded Patches