

Mecanismos de Atenção

Prof. Cícero Ferreira Fernandes Costa Filho

Tópicos:

- 1. Introdução**
- 2. Mecanismo de atenção para tradução de texto**
- 3. Transformer**
- 4. Transformer e mecanismos de atenção em NLP**
- 5. Mecanismos de atenção em visão computacional**
- 6. Transformer e mecanismos de atenção em visão computacional**

1. Introdução

- Os seres humanos não utilizam ativamente toda a informação de que dispõem do ambiente em qualquer momento. Em vez disso, concentram-se em porções específicas de os dados que são relevantes para a tarefa em questão.
- Esta noção biológica é referida como atenção. Um princípio semelhante pode também ser aplicado às aplicações de inteligência artificial. Os modelos com atenção utilizam a aprendizagem de reforço (ou outros métodos) para se concentrarem em porções mais pequenas dos dados que são relevantes para a tarefa em questão.
- Ultimamente, o desempenho de tais métodos foi significativamente melhorado.

1. Introdução

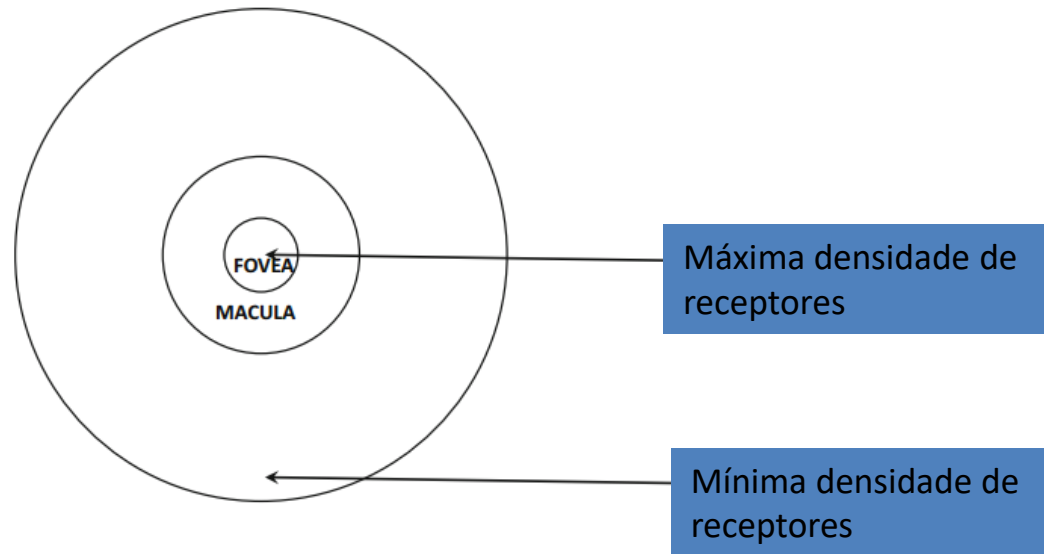
Mácula e Fóvea

- Contém os cones responsáveis pela visão colorida;
- Grande concentração de receptores, com alta resolução.
- Diâmetro da fóvea (1.5mm)

Retina:

- Contém os bastonetes responsáveis pela visão em tons de cinza;
- Baixa concentração de receptores, com baixa resolução.

Ao ler o número de uma casa a região do número concentra-se na fóvea. O olho obtém uma visão com poucos detalhes do resto da imagem.

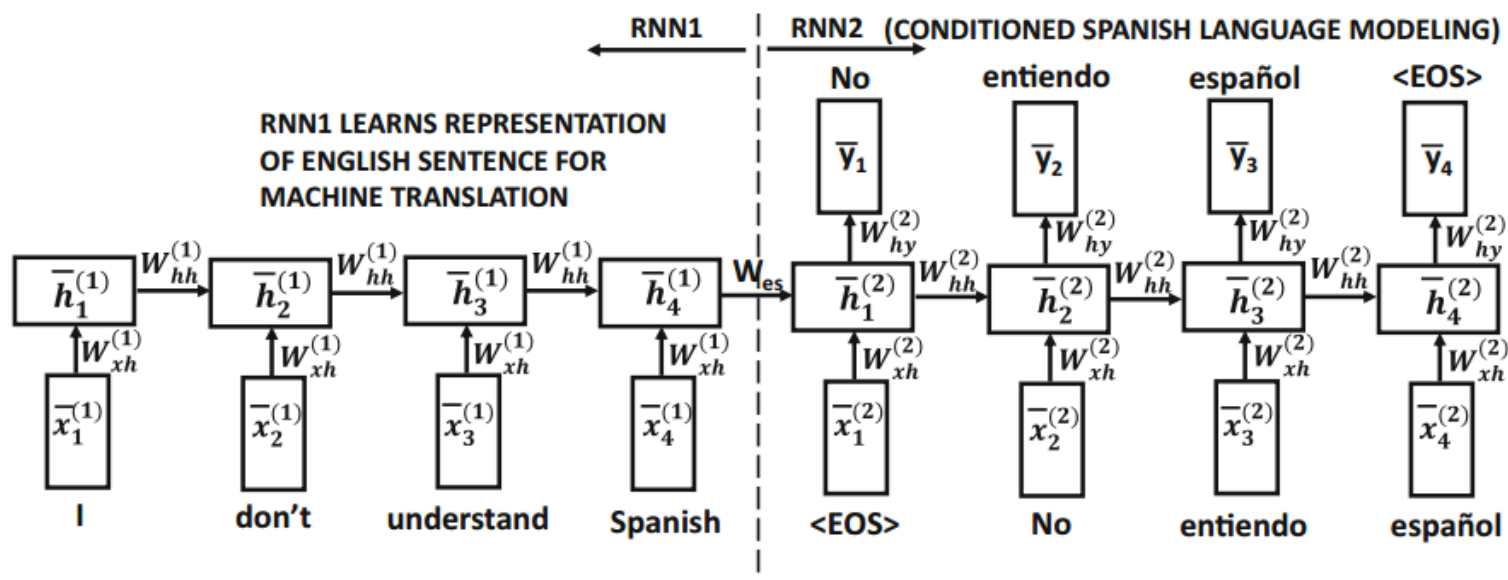


Apenas 0.5% da imagem correspondente a fóvea é transmitida em alta resolução para o cérebro, reduzindo o processamento para a tarefa específica, no caso a identificação do número da casa.

2. Mecanismos de Atenção para tradução de texto

2.1 Mecanismo de Atenção com Redes Recorrentes.

Numa aplicação de tradução de texto com redes recorrentes, a tradução de todas as palavras fica dependente apenas de um único vetor, que tem que codificar toda a informação necessária para a tradução, conforme mostrado na figura abaixo.



[1] Neural Machine Translation by Jointly Learning to Align and Translate, Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio. arXiv:1409.0473, 2014.

2. Mecanismos de Atenção para tradução de texto

2.1 Mecanismo de Atenção com Redes Recorrentes.

A ideia explorada no mecanismo de atenção é permitir a busca de uma correlação mais forte entre uma palavra gerada na tradução com todas as palavras de entrada. Ou seja, para geração de um estado de saída, o mesmo olhará para o “contexto” das palavras de entrada, procurando palavras de entrada mais significativas para a geração de uma palavra de saída.

Em aprendizado de máquina um primeiro mecanismo de atenção que possibilitou a implementação da ideia mencionada anteriormente foi proposta por Bahdanua [1]. Nessa proposta, o mecanismo de atenção é dividido em três etapas.

- Cálculo de escores
- Cálculo dos pesos
- Cálculo do vetor de contexto

[1] Neural Machine Translation by Jointly Learning to Align and Translate, Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio. arXiv:1409.0473, 2014.

2. Mecanismos de Atenção para tradução de texto

2.1 Mecanismo de Atenção com Redes Recorrentes.

1. **Escore:** O modelo de alinhamento considera os estados ocultos codificados, h_i , e a saída anterior do decodificador, s_{t-1} , para calcular uma pontuação, $e_{t,i}$, que indica quão bem os elementos da sequência de entrada se alinham com a saída corrente na posição, t . O modelo de alinhamento é representado por uma função, $a(\cdot)$, que pode ser implementada por uma rede neural de propagação direta:

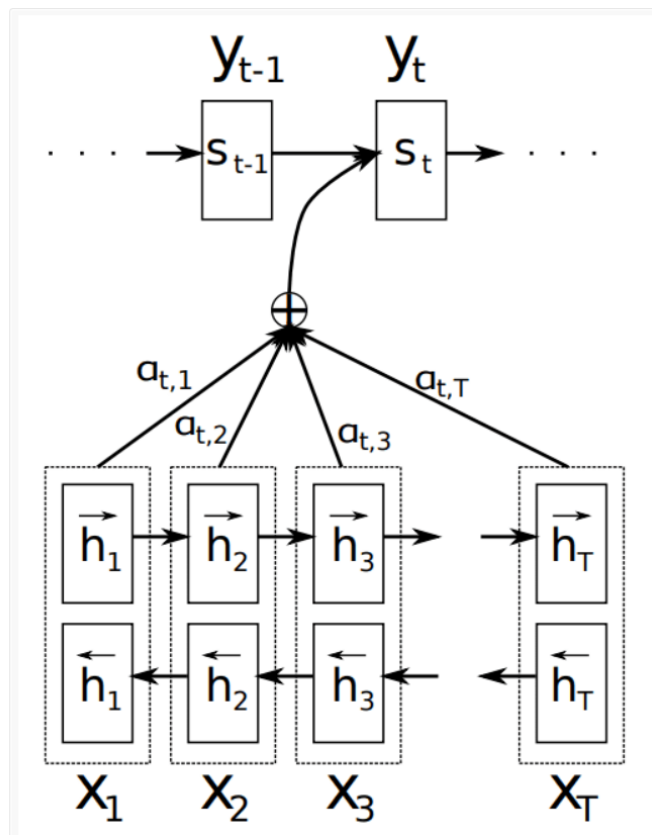
$$e_{t,i} = a(s_{t-1}, h_i)$$

2. **Pesos:** Os pesos, $\alpha_{t,i}$, são calculados aplicando-se a função softmax aos escores previamente calculados:

$$\alpha_{t,i} = \text{softmax}(e_{t,i})$$

3. **Vetor de contexto:** Um único vetor de contexto, c_t , seve como entrada do decodificador. Esse vetor é calculado usando uma soma ponderada no tempo de todos os estados escondidos h_i

$$c_t = \sum_{i=1}^T \alpha_{t,i} h_i$$

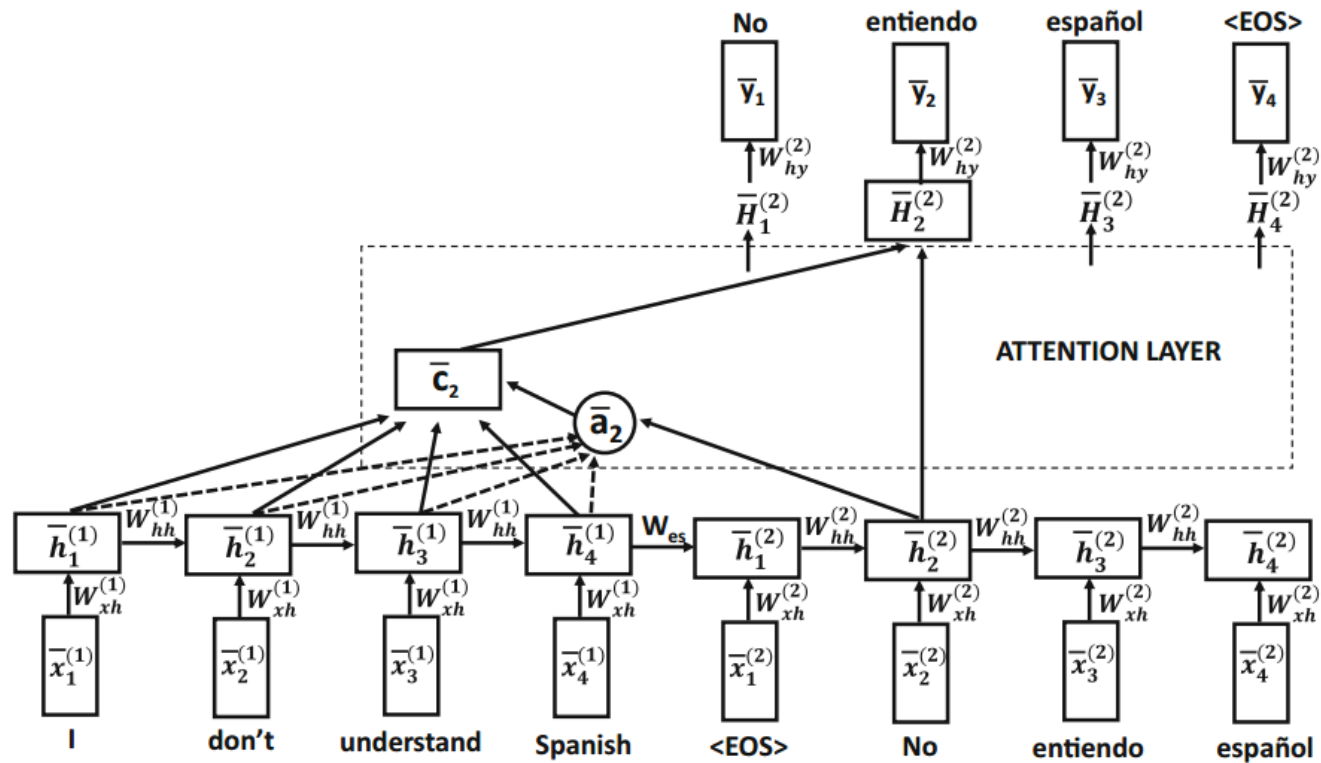


2. Mecanismos de Atenção para tradução de texto

2.2 Redes LSTM e mecanismo proposto por Luong et al.

- A seguir mostraremos uma implementação do mecanismo de atenção descrito anteriormente, com pequenas diferenças, feita por *Luong et al* [2] para uma aplicação de tradução de texto. Para simplificação será utilizada apenas uma rede recorrente de uma camada
- No equacionamento a seguir, os estados escondidos da rede contendo a sequência de entrada serão denotados por $h_t^{(1)}$, enquanto que os da sequência de saída serão denotados por $h_t^{(2)}$.
- Nos métodos de atenção, os estados $h_t^{(2)}$ são transformados em estados enriquecidos $H_t^{(2)}$, através de um processamento adicional por uma camada de atenção.

[2] M. Luong, H. Pham, and C. Manning. Effective approaches to attention-based neural machine translation. arXiv:1508.04025, 2015.



Rede LSTM com mecanismo de atenção proposta por Luong et al.

- No equacionamento a seguir, os estados escondidos da rede contendo a sequência de entrada serão denotados por $h_t^{(1)}$, enquanto que os da sequência de saída serão denotados por $h_t^{(2)}$.
- Nos métodos de atenção, os estados $h_t^{(2)}$ são transformados em estados enriquecidos $H_t^{(2)}$, através de um processamento adicional por uma camada de atenção.
- O objetivo da camada de atenção é incorporar o contexto dos estados ocultos da entrada nos estados ocultos da saída, para criar um novo e melhorado conjunto de estados ocultos da saída.
- Para cada estado de saída t , $h_t^{(2)}$, cria-se um vetor de contexto c_t através dos passos descritos anteriormente, com uma alteração no primeiro passo:

$$1. e_{t,i} = a(h_t^{(2)} \cdot h_i^{(1)}) = h_t^{(2)} \cdot h_i^{(1)} \quad (\text{Principal diferença em relação ao método de Baudinaud, que usa } s_{t-1})$$

$$2. \alpha_{t,i} = \text{softmax}(e_{t,i}) = \frac{\exp(h_i^{(1)} \cdot h_t^{(2)})}{\sum_{i=1}^{T_s} \exp(h_i^{(1)} \cdot h_t^{(2)})}$$

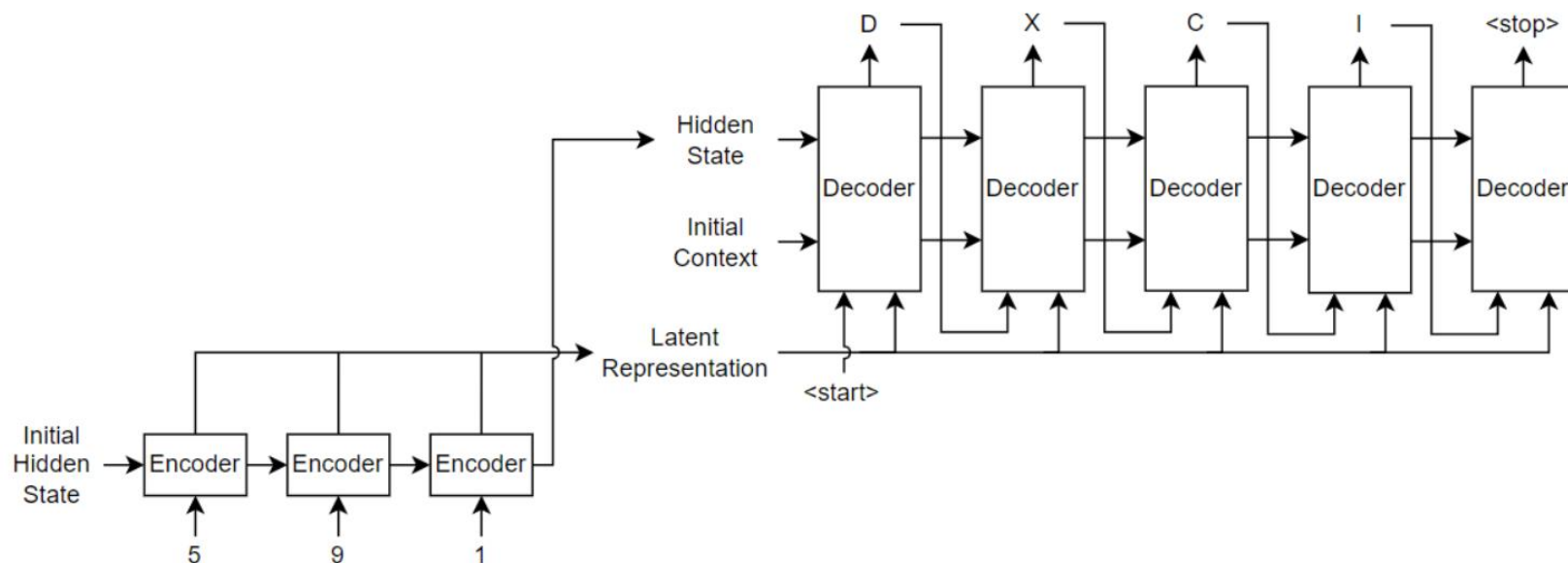
$$3. c_t = \sum_{i=1}^{T_s} \alpha_{t,i} \cdot h_i^{(1)}$$

[1] M. Luong, H. Pham, and C. Manning. Effective approaches to attention-based neural machine translation. arXiv:1508.04025, 2015.

O novo estado escondido $H_t^{(2)}$ pode ser gerado por:

$$H_t^{(2)} = \tanh \left(W_c \begin{bmatrix} c_t \\ h_t^{(2)} \end{bmatrix} \right)$$

- Exemplo no Matlab: **Sequence-to-Sequence Translation Using Attention**



Conversão de um número decimal para algoritmos romanos

3. Transformer

O *transformer* implementa, em visão computacional, um outro modelo de atenção que não aquele descrito anteriormente. Esse novo modelo é denominado de “modelo geral de atenção”. A seguir apresentamos o modelo:

A atenção tem tido grande sucesso na área de processamento da linguagem natural (NLP). Recentemente, mostrou também o potencial para se tornar uma ferramenta dominante na visão computacional. Em visão computacional a atenção é utilizada como um mecanismo de atenção espacial para captar informação global da imagem, com muitas aplicações, como: classificação, detecção de anomalias, reconhecimento de objetos, etc.

Devido à operação localizada das redes convolucionais, as CNN têm campos receptivos com abrangência pequenos, o que limita às redes CNNs de compreenderem as cenas a nível global.

A arquitetura *transformer* revolucionou o uso da atenção ao dispensar a recorrência e as convoluções, nas quais os mecanismos de atenção eram baseados.

O *transformer* é baseado apenas em um mecanismo de auto-atenção (*self-attention*).

O *transformer* utiliza um “modelo geral de atenção”, diferente do modelo de atenção descrito anteriormente e usado em conjunto com as redes recorrentes.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017

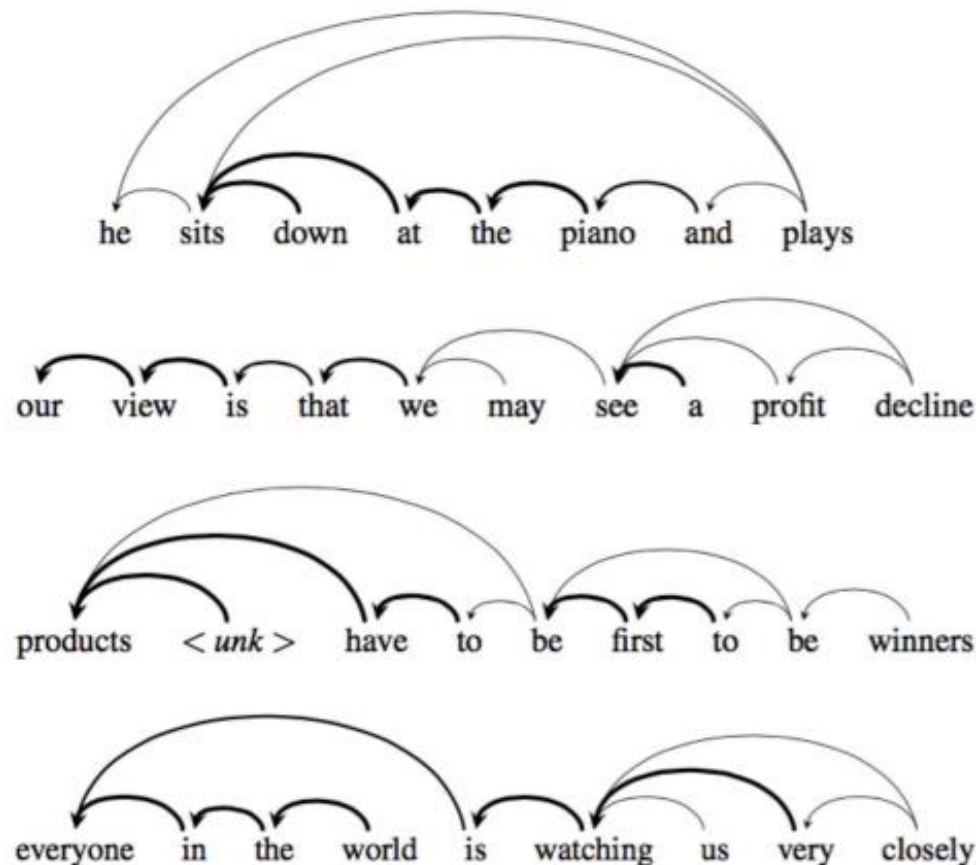
3. Transformer – Mecanismo de Atenção

1. Suponha uma aplicação em que é dada a seguinte frase: ***“As aliens entered ou planet”***
2. Suponha que a tarefa do *transformer* seja completar esse texto. Suponha que ele complete com: ***“and began to colonized Earth, a certain group of extraterrestrials began to manipulate our society through their influences of a certain number of the elite to keep and iron grip over the populace.”***
3. Para tanto cada nova que o transformer prevê exige que ele olhe para todas as palavras já entradas, conforme ilustrado na figura a seguir:



3. Transformer – Mecanismo de Atenção

Exemplos do mecanismo de atenção, mostrando que palavras influenciam a geração de palavras seguintes numa frase.



3. Transformer – Mecanismo de Atenção

- Mas o poder do mecanismo de atenção é que ele não sofre de memória a curto prazo. As RNN's têm uma janela mais curta para referência, por isso quando a história se torna mais longa, as RNN's não podem acessar as palavras geradas mais cedo na sequência.
- Isto também é verdade para as redes GRU's e LSTM's, embora tenham uma maior capacidade para alcançar memória a longo prazo, tendo, portanto, uma janela mais longa para referência.
- O mecanismo de atenção, em teoria, dado recursos computacionais suficientes, tem uma janela infinita para a referência, sendo assim capaz de utilizar todo o contexto da história enquanto gera o texto traduzido.

3. Transformer

- O modelo geral de atenção utiliza três componentes principais: consultas Q, chaves K e valores V.

"A chave de consulta e o conceito de valor vêm de sistemas de recuperação de informação. Por exemplo, quando escreve uma consulta para pesquisar algum vídeo no Youtube, o motor de busca irá mapear a sua consulta contra um conjunto de chaves (título do vídeo, descrição, etc.) associadas aos vídeos candidatos na base de dados, e depois apresentar-lhe os vídeos (valores) mais adequados."

- Os passos do modelo geral de atenção, também chamado atenção do produto interno, que foi proposto por Vaswani et al. (2017) são os seguintes:
1. Cada vetor de consulta q é correlacionado com uma base de chaves, com o objetivo de calcular um escore. Essa correlação é calculada através do produto interno entre a consulta específica com cada chave k_i .

$$e_{q,k_i} = q \cdot k_i$$

3. Transformer

2. Os escores são calculados através de uma função softmax:

$$\alpha_{q,k_i} = \text{softmax}(e_{q,k_i})$$

3. A atenção generalizada é calculada então por uma soma ponderada do vetor v_{k_i} com escore:

$$\text{attention}(q, K, V) = \sum_i \alpha_{q,k_i} v_{k_i}$$

A utilização desse algoritmo no *transformer*, utiliza os seguintes parâmetros:

- q e k , representando vetores de dimensão d_k , contendo as consultas e as chaves, respectivamente.
- v representado vetores de dimensões d_k , contendo os valores.
- **Q, K e V, representando matrizes que agrupam todos as consultas, chaves e valores, respectivamente.**

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin.
Attention is all you need. In *NIPS*, 2017

3. Transformer

A implementação do modelo geral de atenção utilizando as matrizes Q, K e V é mostrado na figura a seguir. A versão mostrada do *transformer* é denominada de atenção do **produto interno escalonado**, devido a operação de escalonamento.

1. Cálculo de $e_{q,k_i} = q \cdot k_i$. Se as matrizes Q e K são $(n \times d_k)$ temos:

$$Q \cdot K^T = \begin{bmatrix} q_{11} & q_{12} & \dots & q_{1d_k} \\ q_{21} & q_{22} & \dots & q_{2d_k} \\ \dots & \dots & \dots & \dots \\ q_{n1} & q_{n2} & \dots & q_{nd_k} \end{bmatrix} \cdot \begin{bmatrix} k_{11} & k_{21} & \dots & k_{n1} \\ k_{12} & k_{22} & \dots & k_{n2} \\ \dots & \dots & \dots & \dots \\ k_{1d_k} & k_{nd_k} & \dots & e_{nd_k} \end{bmatrix} = \begin{bmatrix} e_{11} & e_{12} & \dots & e_{1n} \\ e_{21} & e_{22} & \dots & e_{2n} \\ \dots & \dots & \dots & \dots \\ e_{n1} & e_{n2} & \dots & e_{nn} \end{bmatrix}$$

Em que: e_{ij} representa o produto interno da consulta q_i pela chave k_j

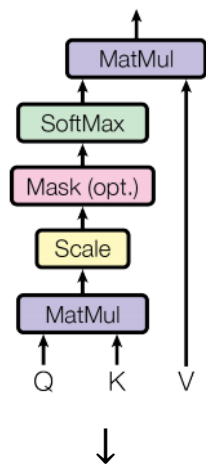
2. Escalonamento (Essa fase não está presente no algoritmo original):

$$\frac{Q \cdot K^T}{\sqrt{d_k}} = \begin{bmatrix} e_{11}/\sqrt{d_k} & e_{12}/\sqrt{d_k} & \dots & e_{1n}/\sqrt{d_k} \\ e_{21}/\sqrt{d_k} & e_{22}/\sqrt{d_k} & \dots & e_{2n}/\sqrt{d_k} \\ \dots & \dots & \dots & \dots \\ e_{n1}/\sqrt{d_k} & e_{n2}/\sqrt{d_k} & \dots & e_{nn}/\sqrt{d_k} \end{bmatrix}$$

3. Cálculo de $\alpha_{q,k_i} = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right)$:


$$Q \cdot K^T = \begin{bmatrix} \text{softmax}(e_{11}/\sqrt{d_k}) & e_{12}/\sqrt{d_k} & \dots & e_{1n}/\sqrt{d_k} \\ \text{softmax}(e_{21}/\sqrt{d_k}) & e_{22}/\sqrt{d_k} & \dots & e_{2n}/\sqrt{d_k} \\ \dots & \dots & \dots & \dots \\ \text{softmax}(e_{n1}/\sqrt{d_k}) & e_{n2}/\sqrt{d_k} & \dots & e_{nn}/\sqrt{d_k} \end{bmatrix}$$

Scaled Dot-Product Attention



**produto
interno
esalonado**

3. Transformer

Softmax() =

	Hi	how	are	you
Hi	0.7	0.1	0.1	0.1
how	0.1	0.6	0.2	0.1
are	0.1	0.3	0.6	0.1
you	0.1	0.3	0.3	0.3

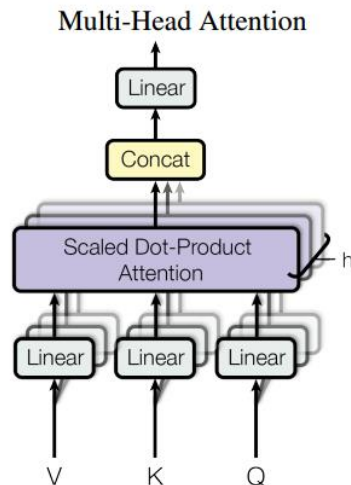
$$\text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

3. Transformer

4. Cálculo da multiplicação: $atenção(q, K, V) = \sum_i \alpha_{q,k_i} v_{k_i}$

$$atenção(q, K, V) = \begin{bmatrix} softmax(e_{11}/\sqrt{d_k}) & e_{12}/\sqrt{d_k} & \dots & e_{1n}/\sqrt{d_k} \\ softmax(e_{21}/\sqrt{d_k}) & e_{22}/\sqrt{d_k} & \dots & e_{2n}/\sqrt{d_k} \\ \dots & \dots & \dots & \dots \\ softmax(e_{n1}/\sqrt{d_k}) & e_{n2}/\sqrt{d_k} & \dots & e_{nn}/\sqrt{d_k} \end{bmatrix} \cdot \begin{bmatrix} v_{11} & v_{12} & \dots & v_{1d_k} \\ v_{21} & v_{22} & \dots & v_{2d_k} \\ \dots & \dots & \dots & \dots \\ v_{n1} & v_{n2} & \dots & v_{nd_k} \end{bmatrix}$$

O Transformer utiliza um bloco mais complexo, com várias blocos de produto interno escalonado, denominado de **Multi-head self attention (MSA)** – Auto atenção de múltiplas cabeças. Esse bloco é mostrado a seguir.



$$multi - head(Q, K, V) = concat(head_1, head_2, \dots, head_h) W^o$$

Em que:

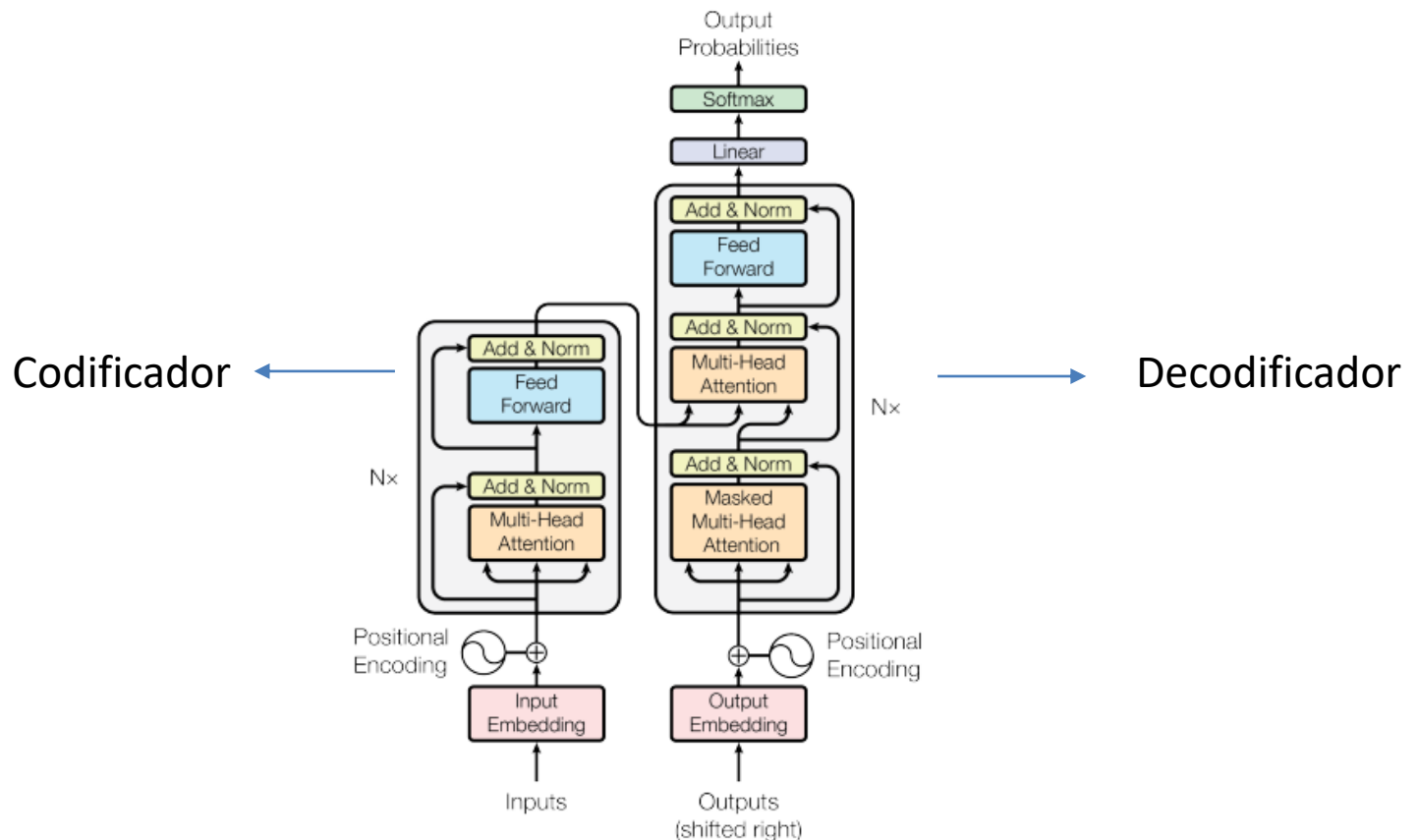
W^o - pesos da camada linear

$$head_i = attention(linear(Q, K, V))$$

$$head_i = attention(QW_i^Q, KW_i^K, VW_i^V)$$

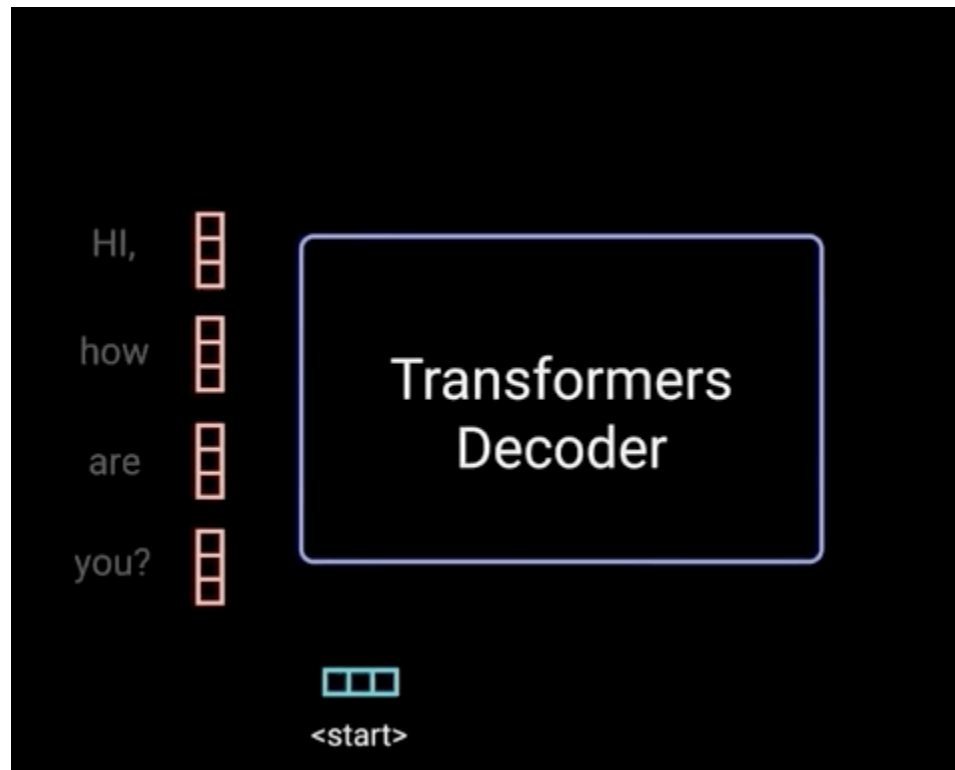
3. Transformer

- A figura abaixo mostra a arquitetura proposta do *transformer* utilizando o bloco MSA>.
- O *Transformer* foi originalmente proposto para fazer a tradução de um idioma para outro. Assim como as redes recorrentes, o mesmo é constituído por duas partes, um codificador e um decodificador. Para se traduzir o texto apresenta-se o mesmo à entrada do codificador (todas as palavras ao mesmo tempo). A saída do decodificador vai gerando as palavras traduzidas uma a uma. A seguir mostra-se a arquitetura do codificador e do decodificador .



3. Transformer

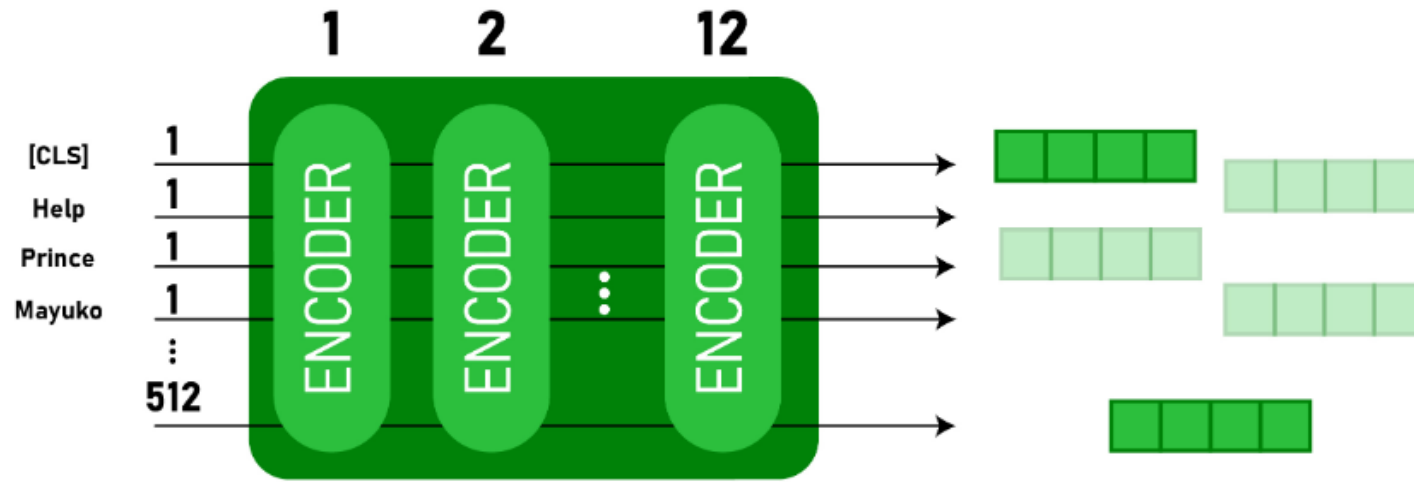
Em um nível elevado, o codificador mapeia uma sequência de entrada numa representação abstrata contínua que guarda toda a informação aprendida dessa entrada. O decodificador toma então essa representação contínua e, passo a passo, gera uma única saída enquanto também é alimentada a saída anterior.



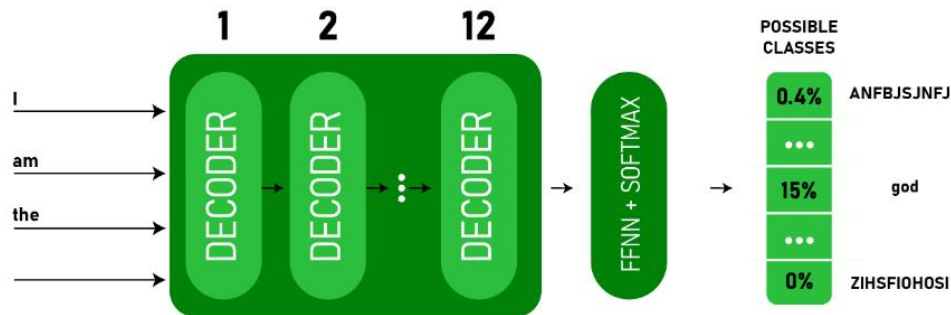
O decodificador é auto-regressivo, ou seja, ele usa a saída anterior para prever a próxima

3. Transformer

- Uso do **ENCODER** do **TRANSFORMER** para a tarefa de tradução:

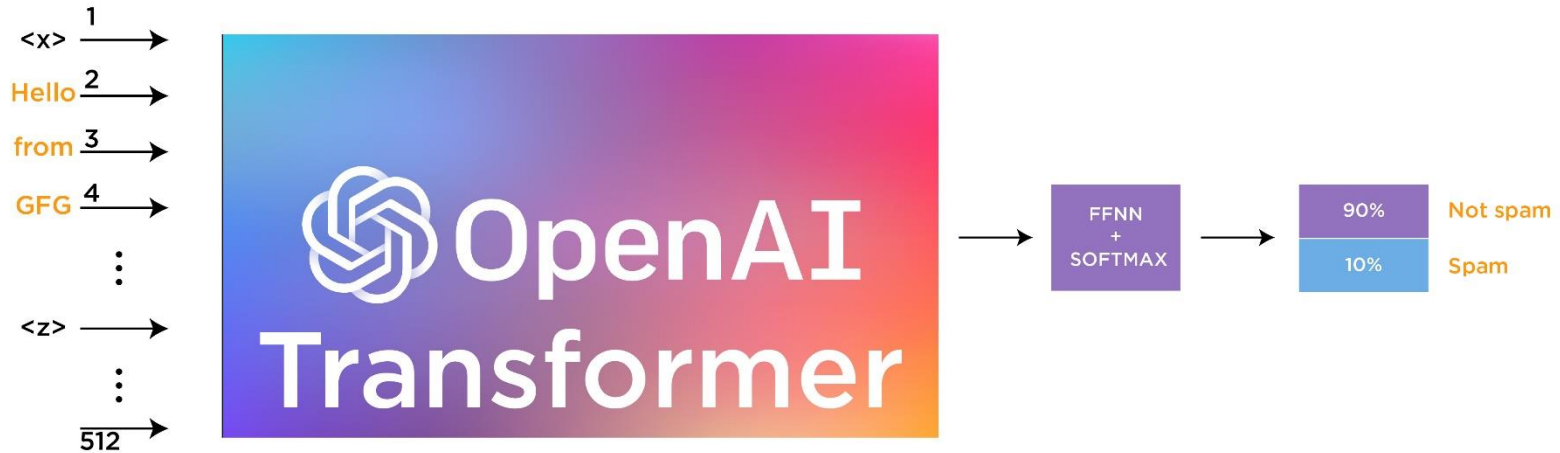


- Uso do **DECODER** do **TRANSFORMER** para classificação de sentenças ou geração da próxima palavra de uma frase (*Language Modeling*):

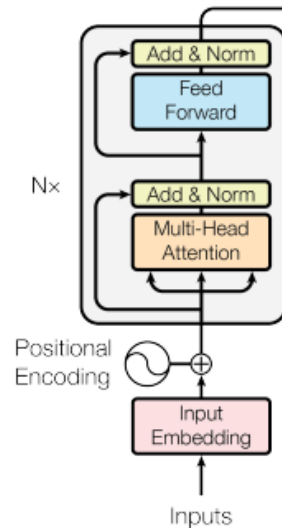
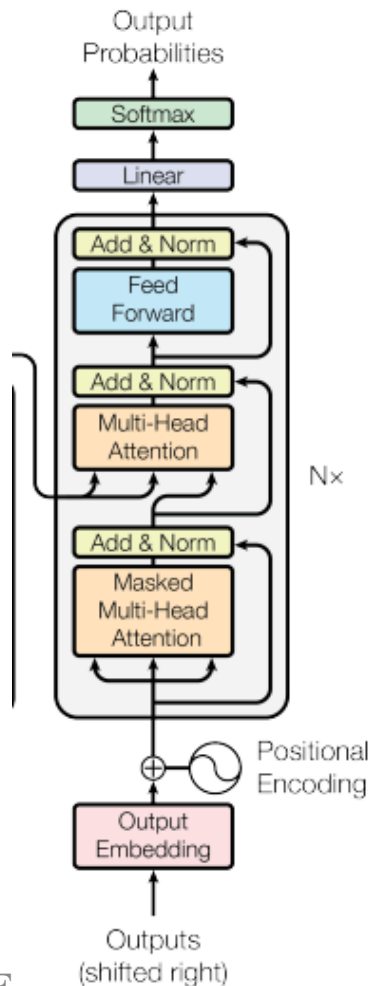


3. Transformer

- Exemplo de Classificação de Sentença: classificação de uma mensagem em SPAM ou Não SPAM



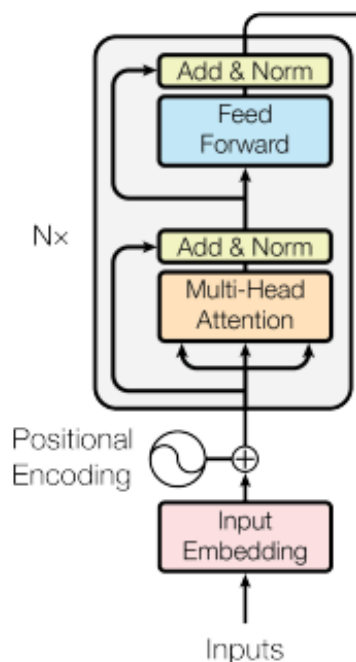
3. Transformer



- No codificador, tem-se apenas um bloco de atenção com múltiplas cabeças.
 - Na tradução entre dois idiomas, os vetores de procura, as chaves e os valores utilizados no decodificador assumem o mesmo valor, que é uma matriz contendo as palavras de entrada.
 - Na atenção do codificador, cada palavra foca a atenção em todas as outras.
 - O codificador é formado por 6 camadas idênticas a essa mostrada à esquerda.
- No decodificador, tem-se dois blocos de atenção com múltiplas cabeças.
 - Na tradução entre dois idiomas, os vetores de procura, as chaves e os valores utilizados no primeiro bloco assumem o mesmo valor, que é a saída do decodificador no instante anterior. No segundo bloco, os vetores das chaves e os valores vêm da saída do codificador, enquanto que os vetores de procura vêm da saída do primeiro bloco.
 - O decodificador é constituído também por 6 camadas.

3.1 Codificador do Transformer

Na figura a seguir um bloco denominado “codificador do transformer”, que utilizaremos nesse curso para processamento em NLP e para o processamento de imagens.



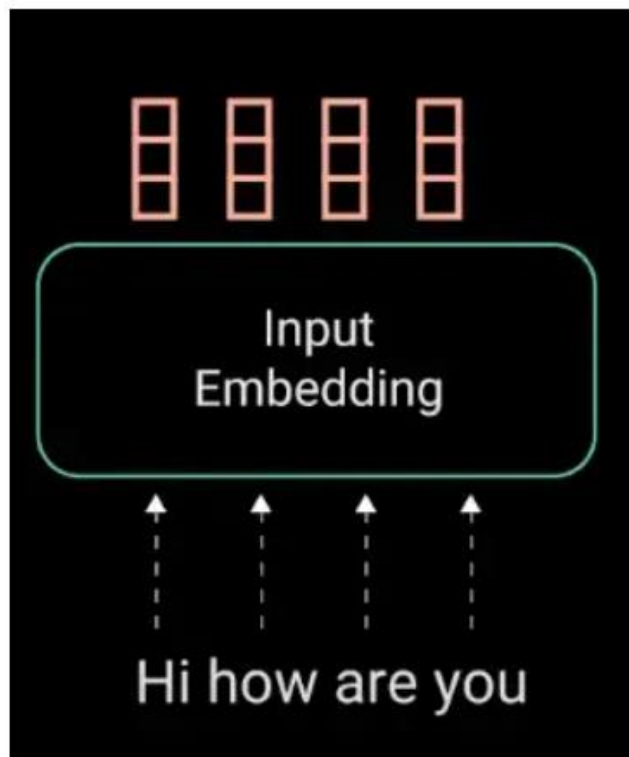
O codificador ao lado é formado por uma camada MAS seguida por uma camada *feedforward*. Antes de cada uma das camadas tem-se uma operação de normalização, que é feita subtraindo-se os valores de uma valor médio e dividindo-se pelo desvio padrão.

A entrada, denominada *embedded inputs*, associa-se a sinais de codificação de posição (*positional encoding*), gerando as marizes Q, K e V vistas anteriormente.

Na sequência veremos a codificação de posição é Implementada no transformer.

3.1.1 Embedding

Codifica cada palavra através de uma camada de *embedding* conforme foi visto em NLP.



3.1.2 Codificador de Posição

A codificação de posição é feita somando-se a matriz de *embedding* à matriz de codificação de posição E^{pos} . Abaixo mostra-se um matriz de posição para um *embedding* com d posições.

Sequence	Index of token	Positional Encoding Matrix			
I	0	P_{00}	P_{01}	...	P_{0d}
am	1	P_{10}	P_{11}	...	P_{1d}
a	2	P_{20}	P_{21}	...	P_{2d}
Robot	3	P_{30}	P_{31}	...	P_{3d}

Positional Encoding Matrix for the sequence 'I am a robot'

A codificação posicional descreve a localização ou posição de uma entidade numa sequência, de modo a que cada posição é atribuída uma representação única. Há muitas razões pelas quais um único número, como o valor da posição, não é utilizado para representar a posição de um item em modelos de *transformers*.

Para sequências longas, os índices podem crescer muito em magnitude. Se se normalizar o valor do índice para situar-se entre 0 e 1, pode-se criar problemas para sequências de comprimento variável, pois seriam normalizados de forma diferente.

3.1.2 Codificador de Posição

Determinando os elementos da matriz E^{pos} :

Suponha que temos uma sequência de entrada de comprimento L e exigimos a posição do k-ésimo objeto dentro desta sequência. A codificação posicional é dada pelas funções seno e cosseno de frequências variáveis:

$$P(k, 2i) = \text{sen}\left(\frac{k}{n^{2i/d}}\right) \quad P(k, 2i + 1) = \text{cos}\left(\frac{k}{n^{2i/d}}\right)$$

Em que:

k: Posição de um objeto na sequência de entrada, $0 \leq k < L-1$;

d: Dimensão do espaço de inserção de saída (dimensão do *embedding*);

i: Usado para mapear índices da coluna $0 \leq i < d/2$.

$P(k,j)$: Valor da coluna (k,j) da matriz posicional. Os valores de P para as colunas pares são calculados com a função senoidal, enquanto que os valores de P para as colunas ímpares são calculados com a função cosseno.

n: escalar definido pelo utilizador. Definido para 10.000 pelos autores;

3.1.2 Codificador de Posição

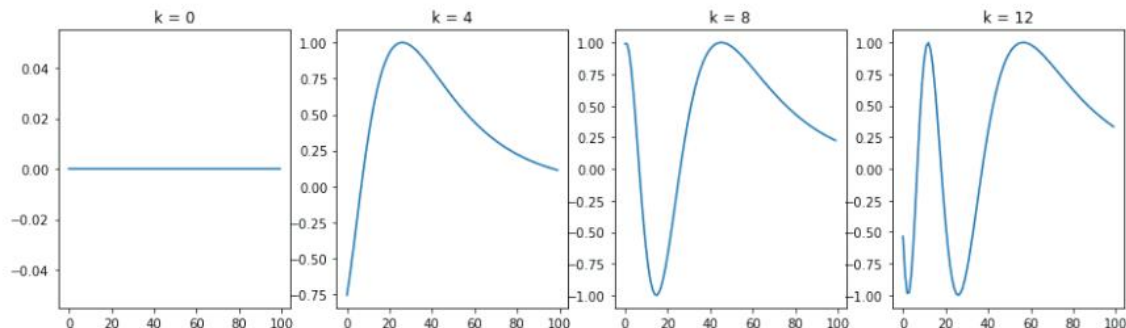
Para compreender as expressões anteriores, tomemos um exemplo da frase “I am a robot”. Seja $n=100$ e $d=4$. A tabela seguinte mostra a matriz de codificação posicional para esta frase. Na realidade, a matriz de codificação posicional seria a mesma para qualquer frase de 4 letras com $n=100$ e $d=4$

Sequence	Index of token, k	Positional Encoding Matrix with $d=4$, $n=100$			
		$i=0$	$i=0$	$i=1$	$i=1$
I	0	$P_{00}=\sin(0)$ = 0	$P_{01}=\cos(0)$ = 1	$P_{02}=\sin(0)$ = 0	$P_{03}=\cos(0)$ = 1
am	1	$P_{10}=\sin(1/1)$ = 0.84	$P_{11}=\cos(1/1)$ = 0.54	$P_{12}=\sin(1/10)$ = 0.10	$P_{13}=\cos(1/10)$ = 1.0
a	2	$P_{20}=\sin(2/1)$ = 0.91	$P_{21}=\cos(2/1)$ = -0.42	$P_{22}=\sin(2/10)$ = 0.20	$P_{23}=\cos(2/10)$ = 0.98
Robot	3	$P_{30}=\sin(3/1)$ = 0.14	$P_{31}=\cos(3/1)$ = -0.99	$P_{32}=\sin(3/10)$ = 0.30	$P_{33}=\cos(3/10)$ = 0.96

Positional Encoding Matrix for the sequence 'I am a robot'

3.1.2 Codificador de Posição

A figura abaixo mostra, para $n=10000$ e $d=512$, a função seno para diferentes valores de k . Como pode ser observado, cada posição tem uma função codificadora diferente.

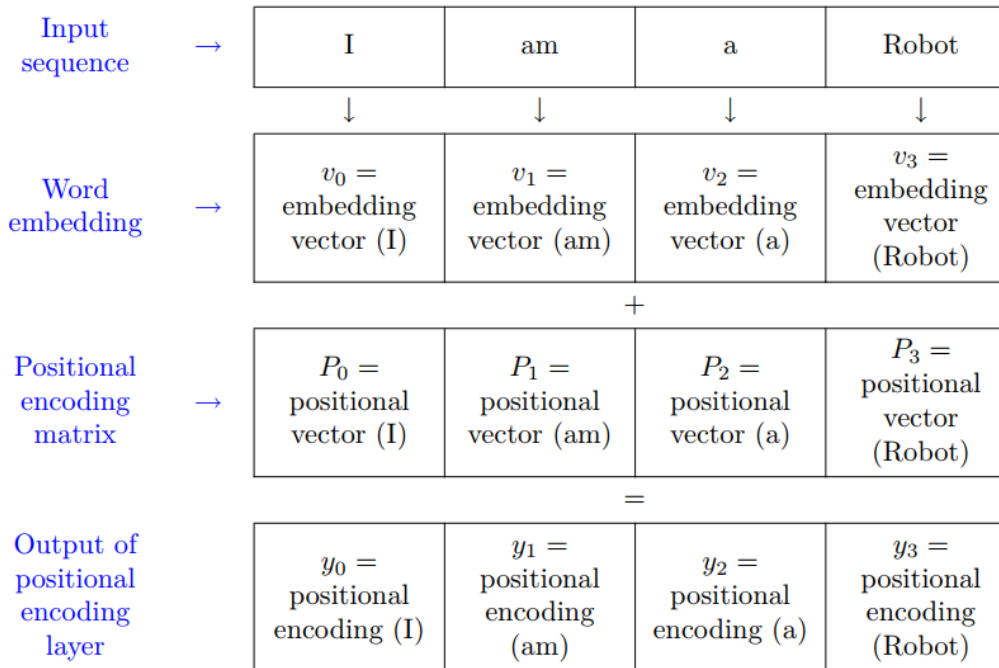


O esquema de codificação posicional tem uma série de vantagens.

1. As funções seno e cosseno têm valores em $[-1, 1]$, o que mantém os valores da matriz de codificação posicional num intervalo normalizado
2. Como a senoide para cada posição é diferente, tem uma forma única de codificar cada posição.
3. Tem uma forma de medir ou quantificar a similaridade entre diferentes posições, permitindo-nos, assim, codificar as posições relativas das palavras.

3.1.2 Codificador de Posição

Resultado da Codificação de Posição:



3.1.3 *Feedforward sub-layer*

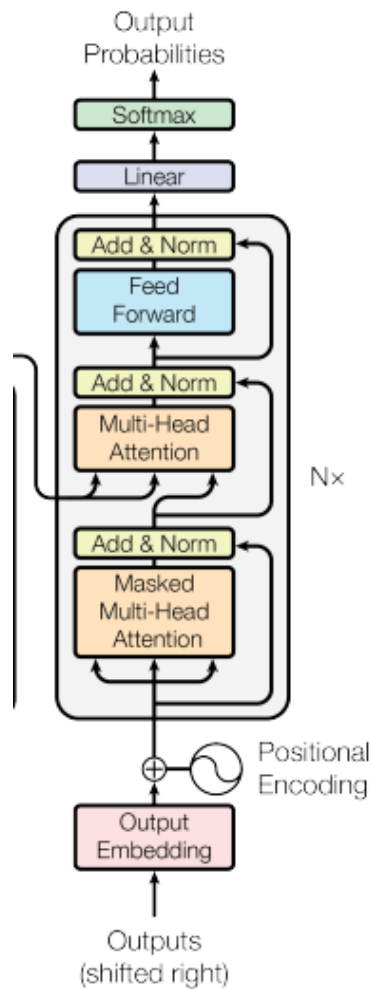
A sub-camada de propagação direta do codificador consiste de uma operação linear aplicada a uma função ReLU, conforme mostrado na expressão abaixo:

$$FFN(x) = (ReLU(x))W_2 + b_2$$

$$FFN(x) = \max(0, xW_1 + b) W_2 + b_2$$

A expressão acima corresponde a duas convoluções com tamanho do kernel igual a 1.

3.2 Decodificador do Transformer



A saída prevê a palavra seguinte através de uma codificação *hot-encoded* fornecida pela camada Softmax.

3.2.1 Decodificador do *Transformer* – *Masked Multhead Attention (MMA)*

Suposição 1: o *transformer* está traduzindo de inglês para alemão a seguinte frase:

- Inglês: I love you
- Alemão: Ich liebe dich

Suposição 2: cada frase é codificada com os seguintes vetores de *embeddings*.

- Inglês: (11, 12, 13)
- Alemão: (21, 22, 23)

Com a utilização do método MMA, as gerações das palavras de saída podem ser feitas em paralelo, conforme mostrado a seguir:

Operações:

Entradas: 11, 12, 13;	Predição: 21; Todo o vetor de saída é mascarado (escondido)
Entradas: 11, 12, 13 e 21;	Predição: 22; Os valores 22 e 23 são mascarados (escondidos)
Entradas: 11, 12, 13, 21 e 22;	Predição: 23; O valor 23 é mascarado (escondido)

3.2.1 Decodificador do *Transformer* – *Masked Multhead Attention (MMA)*

Operação de mascaramento:

$$\text{mask}(Q \cdot K^T) = \text{mask} \left(\begin{pmatrix} e_{11} & e_{12} & \dots & e_{1n} \\ e_{21} & e_{22} & \dots & e_{2n} \\ \dots & \dots & \dots & \dots \\ e_{n1} & e_{n2} & \dots & e_{nn} \end{pmatrix} \right) = \begin{pmatrix} e_{11} & -\infty & \dots & -\infty \\ e_{21} & e_{22} & \dots & -\infty \\ \dots & \dots & \dots & \dots \\ e_{n1} & e_{n2} & \dots & -\infty \end{pmatrix}$$

O mascaramento torna o decodificador unidirecional (ao contrário do codificador bidireccional)

4. Transformer e mecanismos de atenção em NLP

A arquitetura que mostraremos foi publicada no artigo de Devlin et al. (2019) e é conhecida pelo nome de “*Bidirectional Encoder Representations from Transformers*” - BERT.

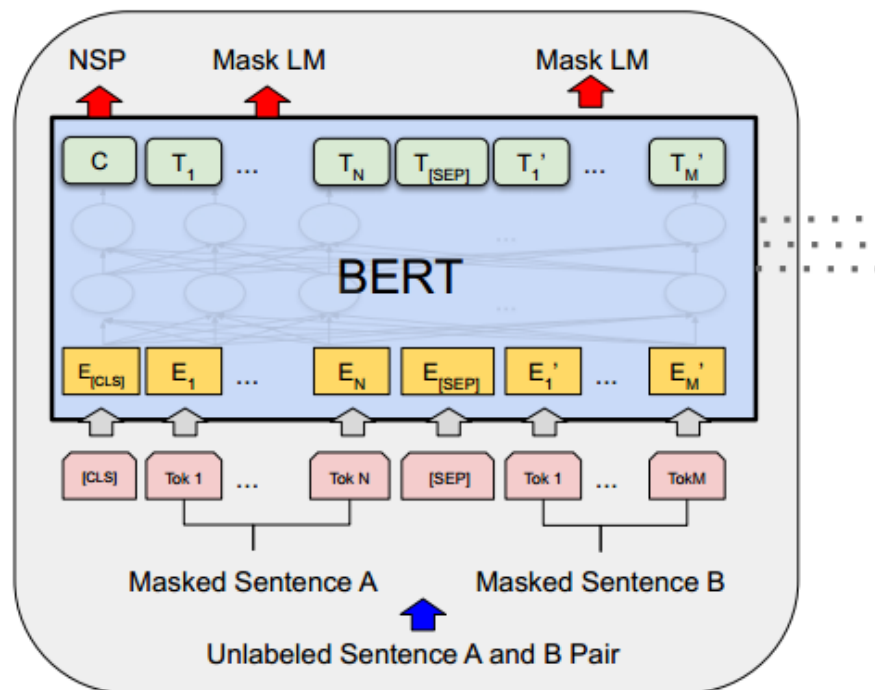
As principais características dessa arquitetura são as seguintes:

- A mesma arquitetura é utilizada para as mais diversas aplicações de NLP: tradução; perguntas e respostas, continuidade de sentença; equivalência entre duas perguntas; dado duas sentenças dizer a relação da segunda com a primeira é de implicação, contradição, ou neutralidade; análise de sentimento, etc.
- O treinamento é realizado em duas fases: **pré-treinamento** e **ajuste fino**.

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova, Google AI Language, arXiv:1810.04805v2 [cs.CL] 24 May 2019

4. Transformer e mecanismos de atenção em NLP

ARQUITETURA DE REDE BERT



Caraterísticas:

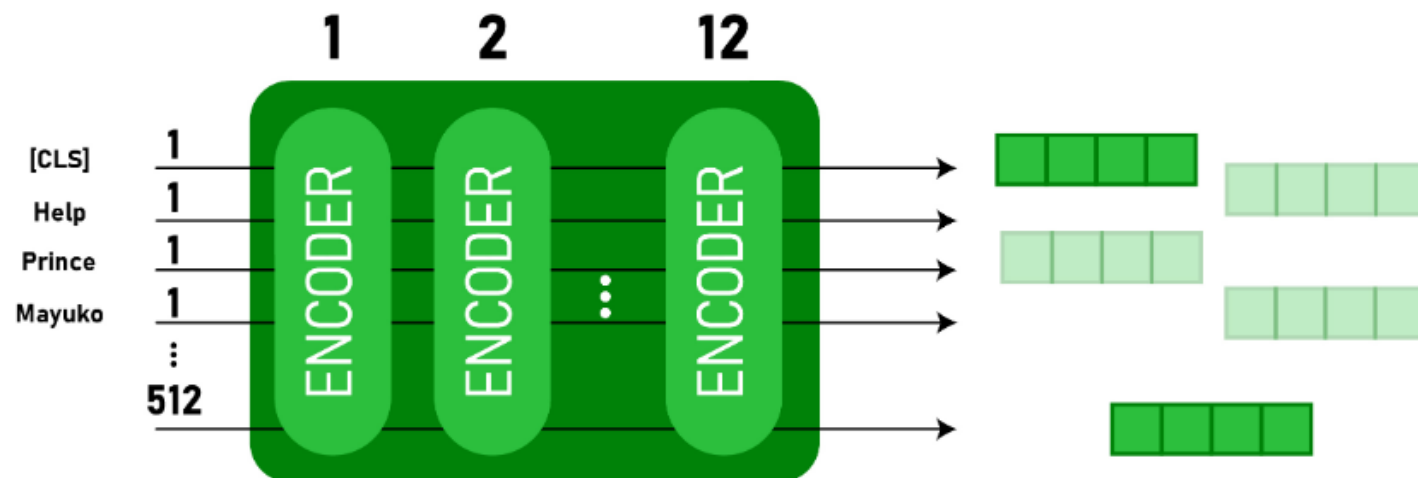
- Duas sentenças
- A primeira sentença começa com um token $[CLS]$.
- Separando as duas sentenças tem um token $[SEP]$.
- Número de blocos *transformers*: L
- Tamanho da representação das camadas escondidas: H
- Número de *heads*: A

• Arquiteturas:

- $BERT_{BASE}$: $L = 12, H = 768, A = 12, Total\ Parameters := 110M$
- $BERT_{LARGE}$: $L=24, H=1024, A=16, Total\ Parameters = 340M$

4. Transformer e mecanismos de atenção em NLP

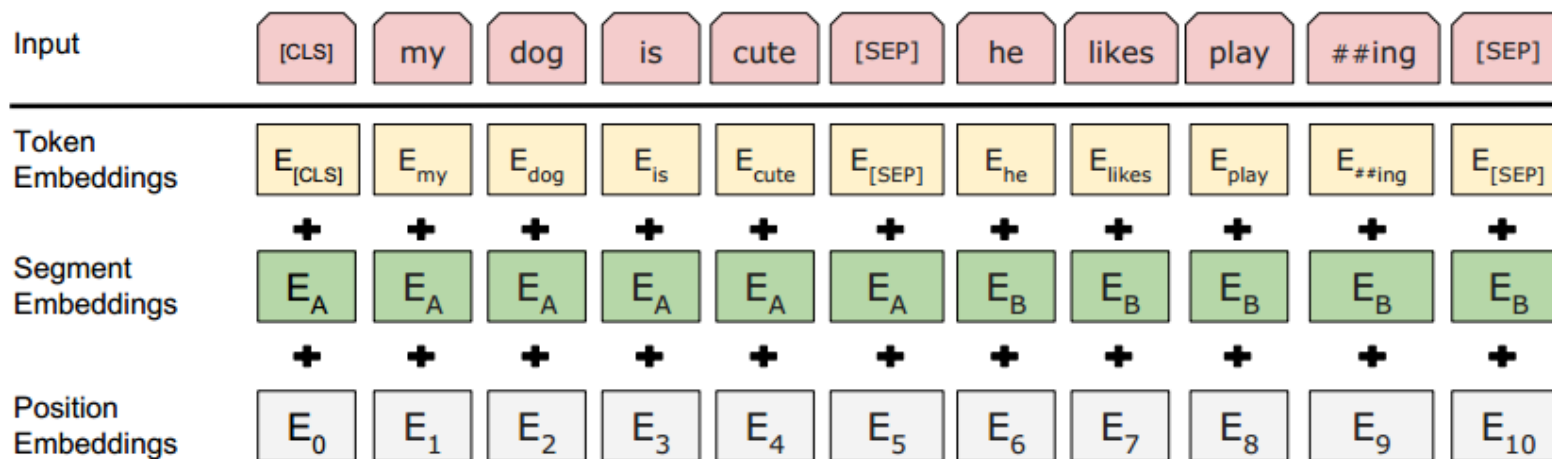
BLOCO DO TRANSFORMER NA ARQUITETURA DE REDE BERT



4. Transformer e mecanismos de atenção em NLP

ARQUITETURA DE REDE BERT

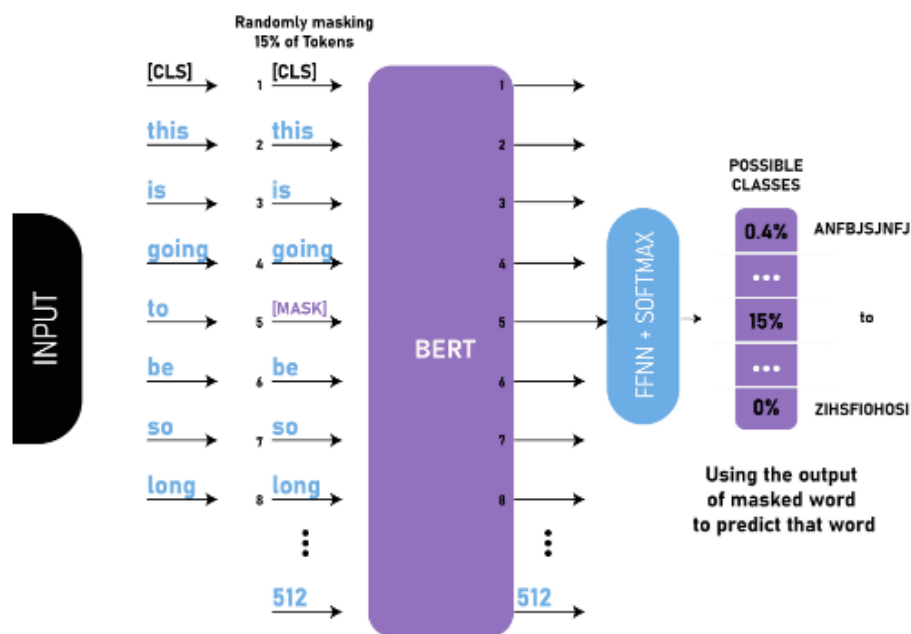
- A entrada da rede BERT é o resultado da associação do token correspondente a palavra, com um codificador de segmento (se o token pertence a palavra A ou B) e um codificador de posição, conforme mostrado na figura a seguir:



4. Transformer e mecanismos de atenção em NLP

PRÉ-TREINAMENTO

- O pré-treinamento utiliza um modelo de linguagem com máscara" (MLM)
- O modelo MLM impõe uma máscara em alguns tokens, sendo o objetivo do treinamento prever o vocabulário original dos tokens mascarados em função do contexto em que o mesmo está inserido.
- O pré-treinamento é comum, independente da tarefa de NLP a ser realizada. O ajuste fino é feito em função da tarefa a ser realizada. A figura a seguir mostra a arquitetura utilizada no pré-treinamento.



4. Transformer e mecanismos de atenção em NLP

- **Metodologia do pré-treinamento para LM (Language Modeling)**
 - As frases A e B são frases aleatórias da linguagem
 - 15% dos tokens de entrada são mascarados com um token *[MASK]*. Quando um *i-ésimo* token é selecionado para ser substituído pelo token *[MASK]* o seguinte esquema de substituição é utilizado:
 - Em 80% das épocas de treinamento, o token da palavra é substituído pelo token *[MASK]*.
 - Em 10% das épocas de treinamento, o token da palavra é substituído por um token aleatório.
 - Em 10% das épocas de treinamento, o token da palavra é usado.
 - Na saída, o token T_i da palavra é utilizado para prever o token original utilizando *cross entropy loss*.

Exemplo: Supor que tenha uma frase em uma das entradas: ***meu cão é peludo***, e que seja escolhida a quarta palavra “***peludo***”

Então:

- 80% das vezes a palavra ***peludo*** é substituída por *[MASK]*: ***meu cão é [MASK]***
- 10% das vezes, a palavra peludo é substituída aleatoriamente: ***meu cão é apple***
- 10% das vezes, a palavra peludo não é substituída: ***meu cão é peludo***

4. Transformer e mecanismos de atenção em NLP

- **Metodologia do pré-treinamento para NSP (*Next Sentence Prediction*):**
Perguntas e respostas, ou NLI (*Natural Language Inference – Próxima sentença*)
 - As frases A e B são frases que obedecem ao seguinte esquema:
 - Em 50% das épocas de treinamento, a frase B é a resposta da frase A, no caso de perguntas e respostas, ou a frase seguinte, no caso de NLI.
 - Em 50% das épocas a frase B é um frase aleatória.
 - A saída C é utilizada para indicar, na saída, se a frase B é ou não a resposta, ou a próxima sentença.

Exemplo:

Entrada: [CLS] o homem entrou [MASK] loja [SEP] ele comprou um galão [MASK] leite

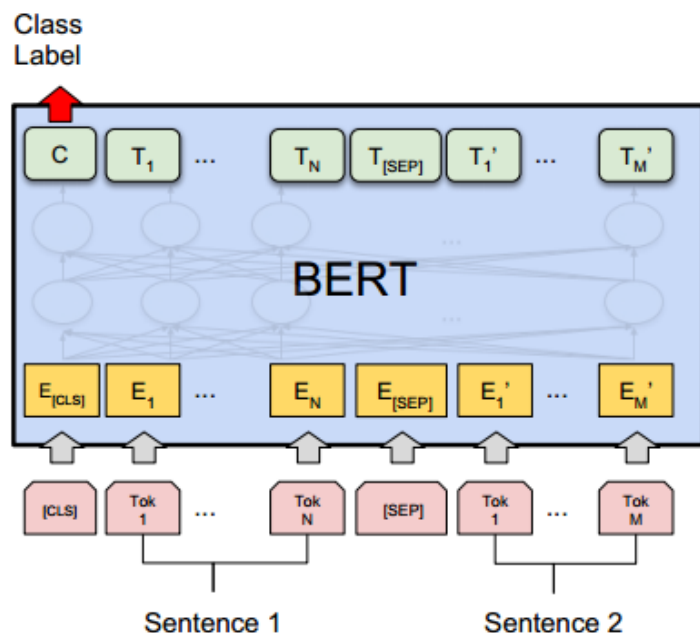
Label: IsNext

Entrada: [CLS] o homem [MASK] na loja [SEP] pinguins vivem no [MASK]

Label: NotNext

4. Transformer e mecanismos de atenção em NLP

AJUSTE FINO 1: Tarefas de Classificação



Arquitetura utilizada para as seguintes aplicações

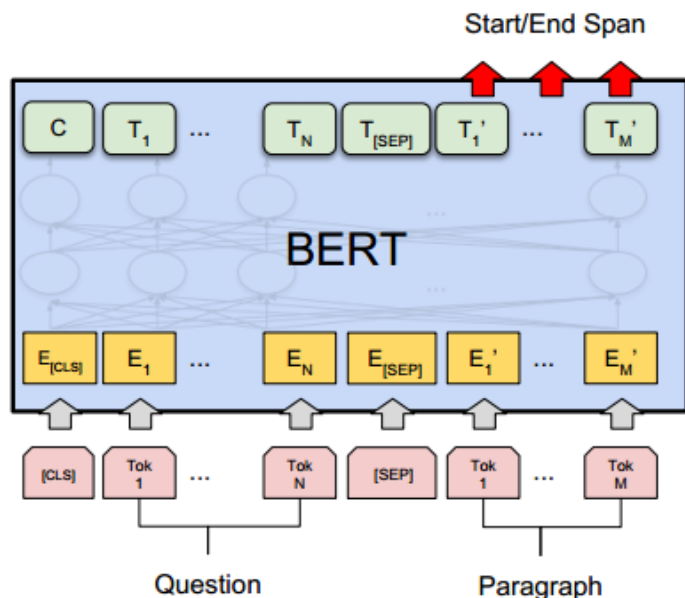
MNLI (Multi-Genre Natural Language Inference): dado um par de sentenças, o objetivo é prever se a segunda frase é uma implicação, contradição, ou É neutra em relação à primeira.

QQP (Quora Questions Pairs): dadas duas sentenças, o objetivo é saber se as mesmas são semanticamente equivalentes.

QNLI (Question natural language inference): dadas duas sentenças, o objetivo é saber se a segunda contém a resposta correta da primeira.

4. Transformer e mecanismos de atenção em NLP

AJUSTE FINO 2: Tarefa de pergunta e resposta



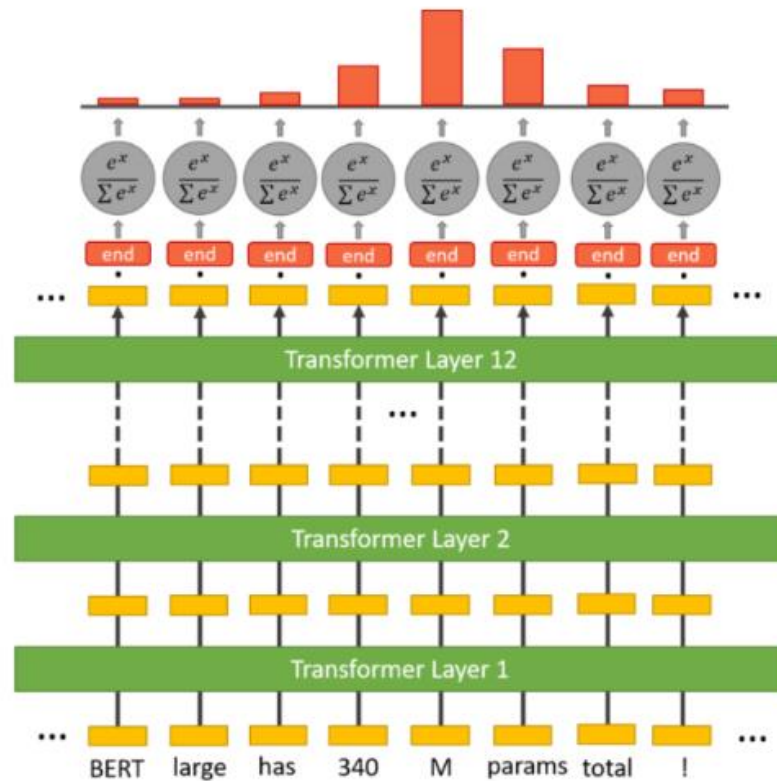
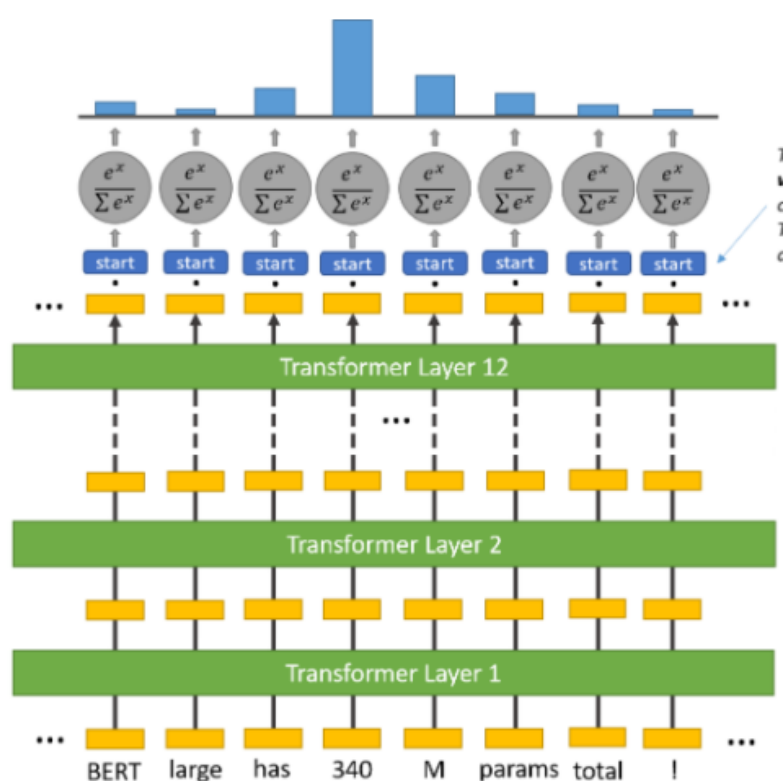
Arquitetura utilizada para as seguintes aplicações

Question and Answering: o objetivo é dada uma pergunta e um texto longo da wikipedia contendo a resposta dentro dele, identificar onde começa e onde termina a resposta. (**Base de dados: SQuAD**)

Na etapa de ajuste fino, é introduzida uma nova camada, onde é feito o produto interno de um vetor **[Start]** e um vetor **[end]** com todos os tokens de saída. Em seguida é calculada uma probabilidade com a função softmax. Aquele token que apresentar uma maior probabilidade para os vetores **[Start]** e **[end]** serão escolhidos como os tokens de início e fim da resposta, simultaneamente.

4. Transformer e mecanismos de atenção em NLP

AJUSTE FINO 2: Tarefa de pergunta e resposta



4. Transformer & MATLAB

NLP & MATLAB: <https://matlabsimulation.com/natural-language-processing-using-matlab/>

→ ↻ 🔒 matlabsimulation.com/natural-language-processing-using-matlab/

Matlab Simulation
Technology with Trust

Matlab Projects ▾ Simulink Projects ▾ Matlab Guidance ▾ Contact Us ☰

▶ Matlab Simulink

▶ power Electronics

▶ Renewable Energy

▶ Mobile Communication

▶ Satellite Communication

▶ Optical Communication

▶ Video processing


▶ Artificial Intelligence

▶ Advanced Robotics

▶ Networking

Related Tools

▶ SIMULINK



NATURAL LANGUAGE PROCESSING USING MATLAB


WWW.MATLABSIMULATION.COM

matlabsimulation.com/wp-content/uploads/2021/11/Implementing-Natural-Language-Processing-Using-matlab-programming.jpg

ex For Matlab


4. Transformer & MATLAB

<https://www.mathworks.com/matlabcentral/fileexchange/107375-transformer-models>

 [Products](#) [Solutions](#) [Academia](#) [Support](#) [Community](#) [Events](#) [Get MATLAB](#)

File Exchange

[MATLAB Central](#) [Files](#) [Authors](#) [My File Exchange](#) [Publish](#) [About](#)



Transformer Models

version 1.2 (92.7 KB) by [David Willingham](#) **STAFF**

Deep Learning Transformer models in MATLAB

<https://github.com/matlab-deep-learning/transformer-models>

[+ Follow](#)

[Overview](#) [Functions](#) [Reviews \(2\)](#) [Discussions \(0\)](#)

Transformer Models for MATLAB

tests **passing**

This repository implements deep learning transformer models in MATLAB.

Translations

- [日本語](#)

Requirements

BERT and FinBERT

- MATLAB R2021a or later

Requires

- [MATLAB](#)
- [Deep Learning Toolbox](#)
- [Text Analytics Toolbox](#)

MATLAB Release Compatibility

Created with R2021b
Compatible with any release

Platform Compatibility

☒ Windows ☒ macOS ☒ Linux

Tags

[bert](#) [deep learning](#) [finbert](#)

5. Mecanismos de Atenção em Visão Computacional

- Em visão computacional mecanismos de atenção são métodos para concentrar a atenção em partes mais importantes da imagem.
- Num sistema de visão, um mecanismo de atenção pode ser tratado como um processo de seleção dinâmico que é realizado por parâmetros adaptativos, de acordo com a importância da entrada.
- Os mecanismos de atenção têm proporcionado benefícios em muitas tarefas visuais, por exemplo, classificação de imagens, detecção de objetos, segmentação semântica reconhecimento de face, reconhecimento de ações, processamento de imagens médicas, geração de imagem, visão 3D, tarefa multimodal.
- Em função do método utilizado, existem 4 categorias básicas de sistemas de atenção em visão computacional: atenção em canal, atenção espacial e atenção temporal

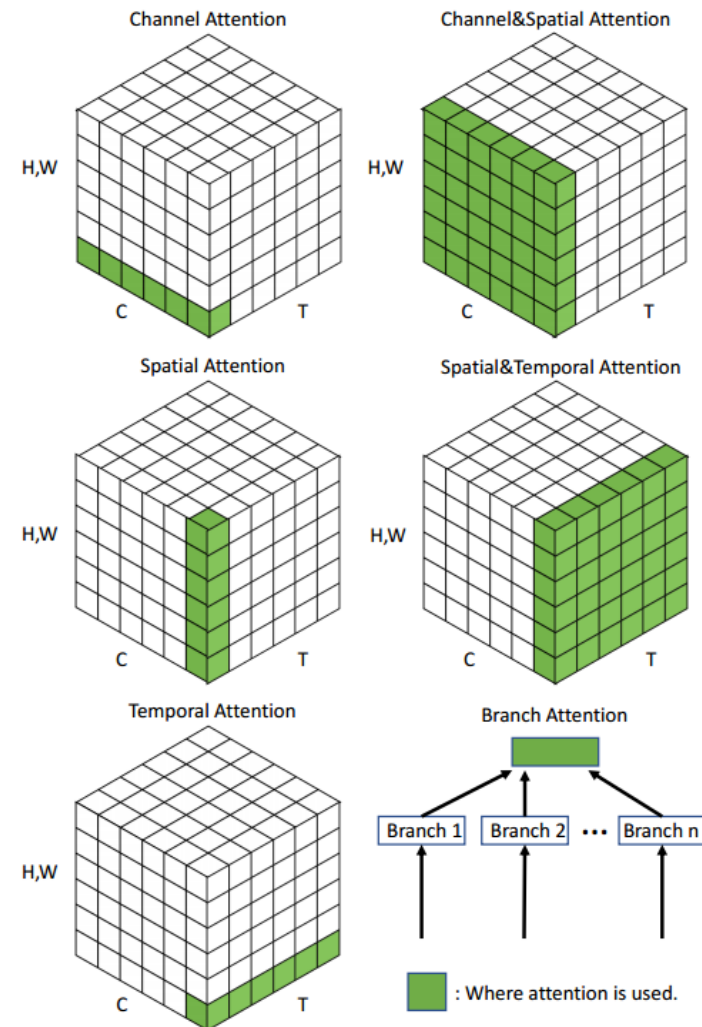
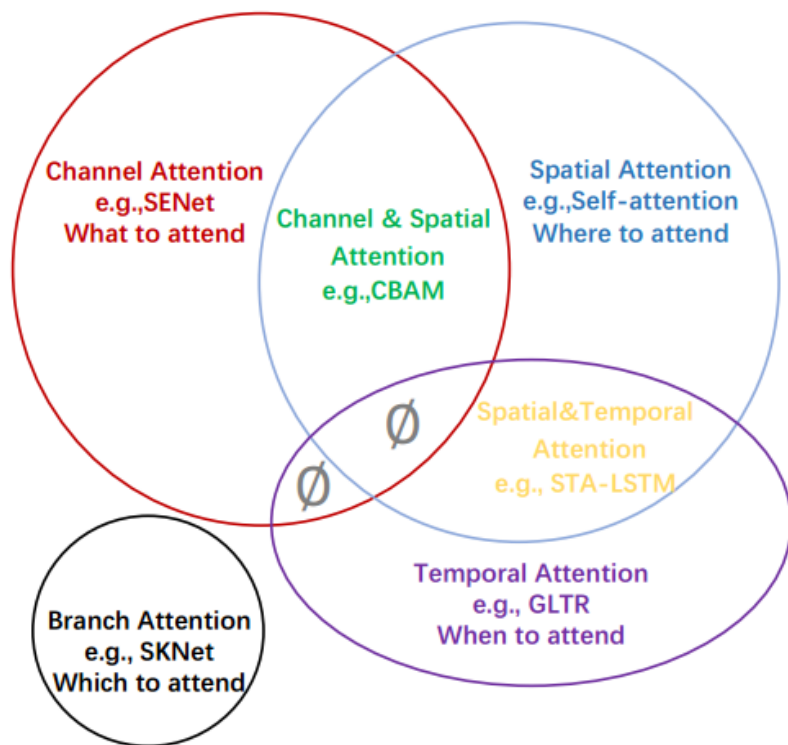
Attention Mechanisms in Computer Vision: A Survey. Meng-Hao Guo, Tian-Xing Xu, Jiang-Jiang Liu, Zheng-Ning Liu, Peng-Tao Jiang, Tai-Jiang Mu, Song-Hai Zhang, Ralph R. Martin, Ming-Ming Cheng, Senior Member, IEEE, Shi-Min Hu, Senior Member, IEEE. arXiv:2111.07624v1 [cs.CV] 15 Nov 2021

5. Mecanismos de Atenção em Visão Computacional

- **Atenção de canal:** gera uma máscara de atenção através do domínio do canal, aplicada a informação apresentada à entrada da rede, e utiliza-a para selecionar canais importantes. Exemplo: canais de uma imagem RGB.
- **Atenção espacial:** gera uma máscara de atenção através de domínios espaciais, aplicada a informação apresentada à entrada da rede, e utiliza-a para selecionar regiões espaciais importantes. Exemplo: Qualquer imagem.
- **Atenção temporal:** gera uma máscara de atenção no tempo, aplicada a informação de entrada apresentada à rede e utiliza-a para selecionar frames importantes. Exemplo: Seleção de frames importantes em um vídeo.
- **Atenção de Ramo:** combina saídas geradas de forma diferente. Por exemplo, a combinação da saída gerada por kernels de tamanhos diferentes em uma camada convolutiva. Inspirado no fato de que investigação na comunidade neurocientífica sugere que aos neurónios corticais ajustam adaptativamente os tamanhos dos seus receptivos campos (RFs) de acordo com o estímulo de entrada.

Attention Mechanisms in Computer Vision: A Survey. Meng-Hao Guo, Tian-Xing Xu, Jiang-Jiang Liu, Zheng-Ning Liu, Peng-Tao Jiang, Tai-Jiang Mu, Song-Hai Zhang, Ralph R. Martin, Ming-Ming Cheng, Senior Member, IEEE, Shi-Min Hu, Senior Member, IEEE. arXiv:2111.07624v1 [cs.CV] 15 Nov 2021

5. Mecanismos de Atenção em Visão Computacional



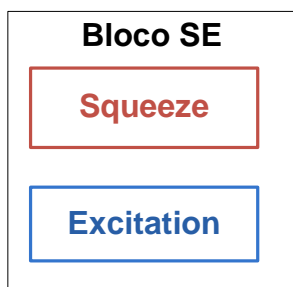
5. Mecanismos de Atenção em Visão Computacional

- No dia a dia, podemos observar uma cena e focar em regiões específicas da mesma, visando uma tomada de decisão. Por exemplo, ao procurarmos por um endereço em uma rua, olhamos para uma casa e centramos a atenção no número.
- Matematicamente, essa atenção por ser representada pela equação abaixo:
- $s = f(g(x), x) \rightarrow \text{vetor de atenção}$
- A função $g(x)$ pode representar o processo de atenção, que corresponde ao foco da atenção em regiões críticas com poder discriminativo.
- $f(g(x), x)$ significa processar a entrada x com base na atenção $g(x)$ a regiões específicas da imagem.

5. 1 Mecanismos de Atenção de Canal

5.1.1 SENet

- SENet – Squeeze Excitation Net (Rede de Compressão e Excitação)
- O núcleo do SENet é formado por um bloco de compressão e excitação (SE) que é usado para coletar informações globais, capturar relações entre canais e melhorar a capacidade de representação.



As informações espaciais globais são coletadas no módulo de compressão por amostragem de média global de cada canal (*global average pooling*)

O módulo de excitação captura relações entre canais e gera um vetor de atenção usando camadas inteiramente conectadas e camadas não lineares (ReLU e sigmoide).

5.1.1 SENet

- Um bloco de Compressão e Excitação F_{se} (com parâmetros θ), com entrada X e saída Y é formulado matematicamente por:

$$s = F_{se}(X, \theta) = \sigma \left(W_2 \delta(W_1 \text{GAP}(X)) \right) \quad (1)$$

$$Y = s \odot X \quad (2)$$

Em que:

s - vetor de atenção

σ - função sigmoide

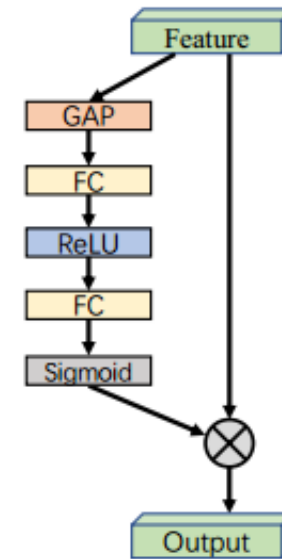
W_2 - matriz da segunda camada inteiramente conectada

δ – função de ativação ReLU

W_1 - matriz da primeira camada inteiramente conectada

GAP – *Global average pooling de cada canal de X (1 canal gera um número real)*

- Equação (2): Cada canal de entrada é escalonado multiplicando-se pela coordenada correspondente do vetor de atenção.



Bloco SE

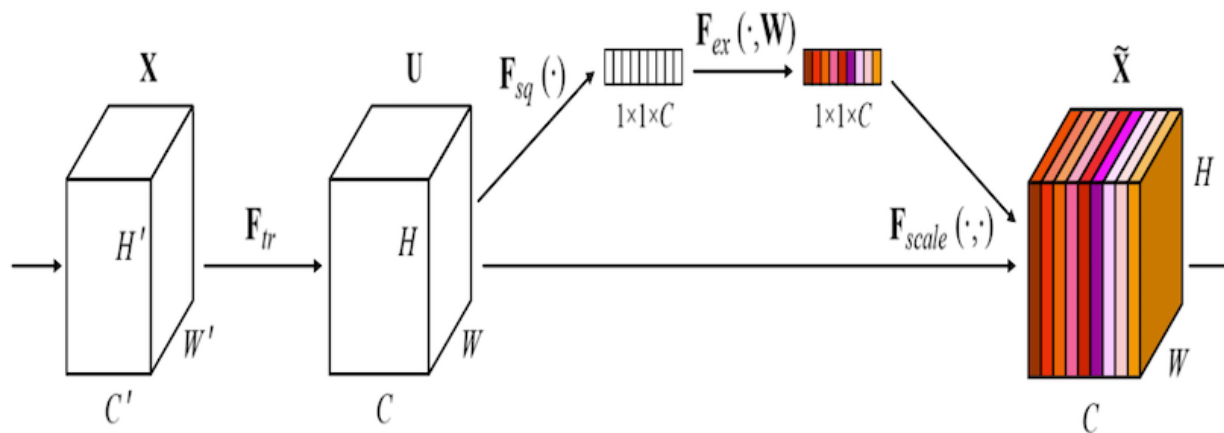
Problemas dos blocos SE:

- No módulo de compressão, a operação *global average pooling* é muito simples para capturar informações globais complexas.
- Na excitação as camadas inteiramente conectadas aumentam a complexidade do modelo.
- O bloco SE pode ser inserido após a saída de cada camada convolutiva.

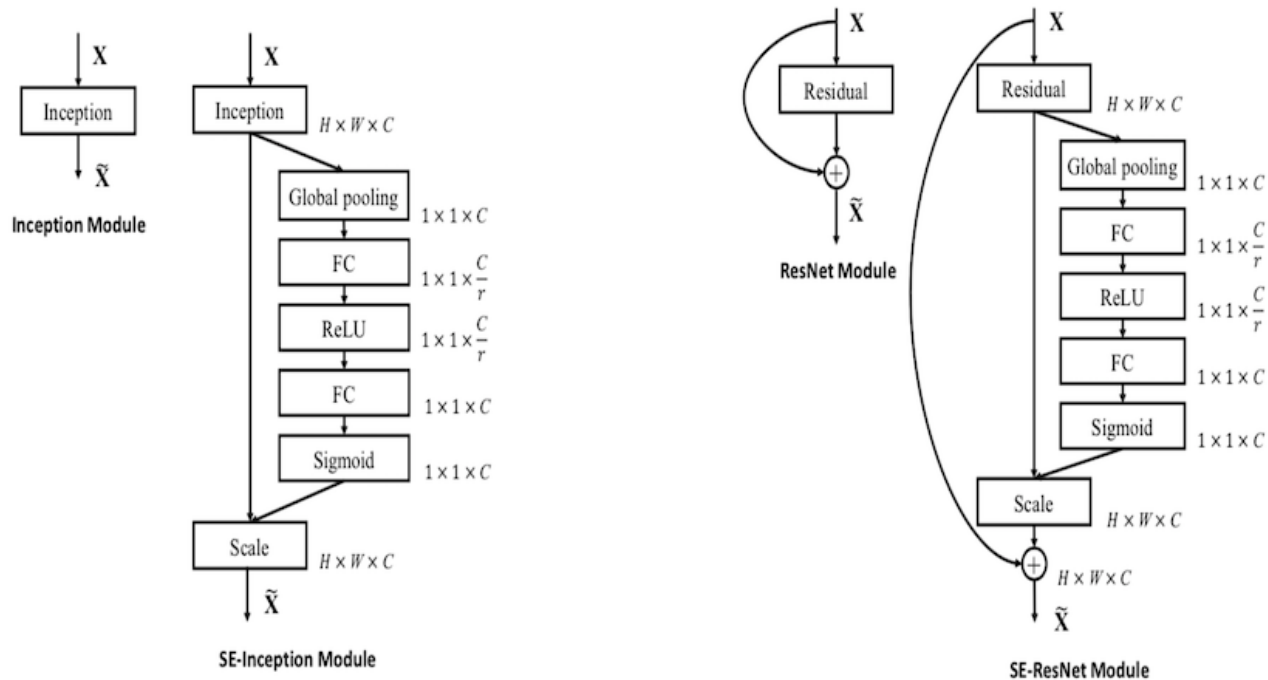
5.1.1 SENet

Squeeze: Nesta parte, primeiro transforma-se o mapa de características 2D em mapas de características 1D. $(\text{Batch_size}, H, W, C) \rightarrow (\text{Batch_size}, 1, 1, C)$. Depois alimenta-se o tensor numa camada densa (totalmente conectada) que pode ter menos filtros/unidades do que os filtros/unidades de entrada. Esta redução de filtros/unidades é para poupar a complexidade computacional. O vetor 1D é chamado de descritor de canais, o qual agrega mapas de características através da operação de amostragem.

Excitation: Nesta parte, o sinal irá passar por uma camada Densa (totalmente ligada) que tem o mesmo número de filtros/unidades que os filtros/unidades de entrada. Em seguida, utiliza-se ativação sigmóide para produzir as saídas para cada canal do tensor original. Utiliza-se estas saídas para aprender a importância das dependências de cada canal. No final, as saídas são multiplicadas pelo canal correspondente no tensor original para aumentar/diminuir a importância do mesmo.



5.1.1 SENet





Problemas dos blocos SE:

- No módulo de compressão, a operação *global average pooling* é muito simples para capturar informações globais complexas.
- Na excitação as camadas inteiramente conectadas aumentam a complexidade do modelo.
- O bloco SE pode ser inserido após a saída de cada camada convolutiva.



5.1.1 SENet

Biblioteca Keras github



 RayXie29 / SENet_Keras Public

 Notifications

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#)

master  1 branch  0 tags

[Go to file](#) [Code](#)

 RayXie29 update readme file bce1bfb on 22 Sep 2019  13 commits

imgs	ver1	3 years ago
.gitignore	ver1	3 years ago
Cifar10_Keras_example.ipynb	ver1	3 years ago
Cifar10_Keras_example.py	ver1	3 years ago
README.md	update readme file	3 years ago
SENet.py	ver1	3 years ago

☰ README.md

Squeeze-and-Excitation Networks


Environments


keras version : 2.2.5
python version : 3.6.4


Info


About

Implementation of Squeeze-and-Excitation Network on Keras

 Readme

 19 stars

 3 watching

 7 forks


Releases



No releases published

Packages

No packages published

Languages



 Jupyter Notebook 96.3%  Python 3.7%

5.1.2 GSoP Net

- Propõe melhorar o módulo de compressão usando uma operação de *pooling* global de segunda ordem (*global second-order pooling* - GSoP), para modelar estatísticas de ordem mais elevada durante a coleta de informações globais.
- Como a SENet é formado também por um bloco de compressão e excitação (SE)



Um bloco de compressão primeiro reduz o número de canais de c para c' ($c' < c$) usando uma convolução 1×1 , depois calcula uma matriz de covariância $c' \times c'$ para os diferentes canais para obter sua correlação. Em seguida, a normalização por linha é executada na matriz de covariância. Cada elemento (i, j) no valor normalizado da matriz de covariância relaciona explicitamente o canal i ao canal j

Um bloco de excitação executa a convolução por linha para manter as informações estruturais e gerar um vetor. Em seguida, uma camada totalmente conectada e uma função sigmoide são aplicadas para obter um vetor de atenção c -dimensional.

5.1.2 GSoP Net

- Um bloco de Compressão e Excitação GSoP (com parâmetros θ), com entrada X e saída Y é formulado matematicamente por:

$$s = F_{gsop}(X, \theta) = \sigma \left(WRC \left(Cov(Conv(X)) \right) \right) \quad (1)$$

$$Y = s \odot X \quad (2)$$

Em que:

s - vetor de atenção

σ - função sigmoide

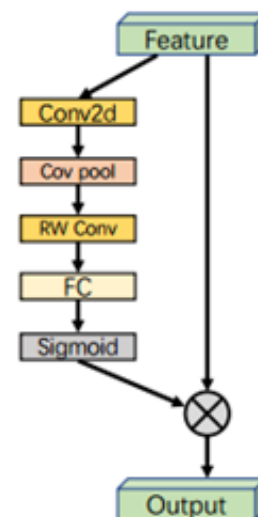
W – matriz da camada inteiramente conectada

RC - convolução baseada em linhas que gera um vetor a partir da matriz de covariância, preservando a informação em cada canal c' .

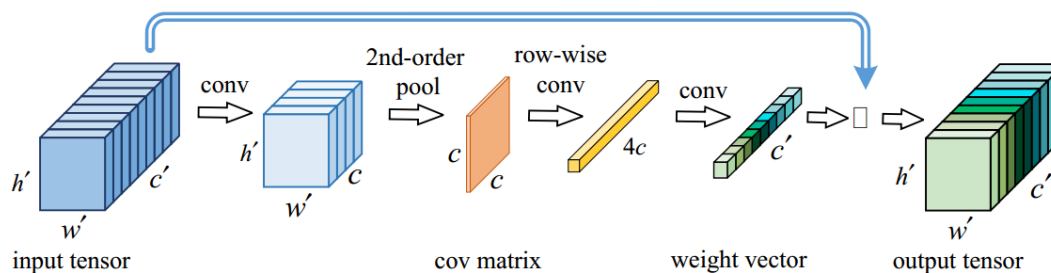
Cov – calcula a matriz de covariância

$Conv2d$ – Convoluções de 1x1 para reduzir o número de canais para c'

- Equação (2): Cada canal de entrada é escalonado multiplicando-se pela coordenada correspondente do vetor de atenção.



Bloco GSoP



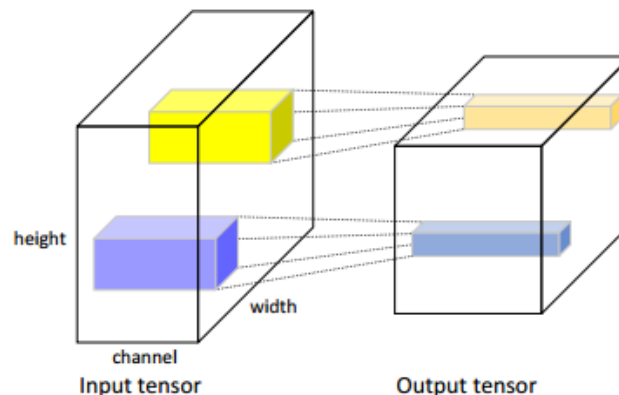
5.1.2 GSoP Net

Comentários:

- Os blocos GSoP melhoraram a capacidade de coletar informações globais sobre o bloco SE. No entanto, isso tem um custo adicional de computação.
- Assim, um único bloco GSoP é normalmente adicionado após vários blocos residuais.

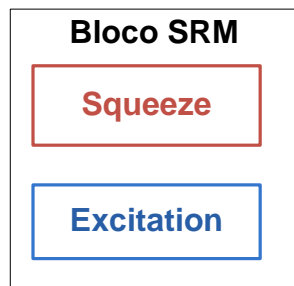
Ideia explorada pelo bloco GSoP:

Operações convolucionais clássicas não conseguem captar dependência holística do tensor 3D devido a uma receptividade limitada ao tamanho do kernel. Por exemplo, os dados em pequeno tensor azul não podem interagir com a do tensor amarelo em posição distante devido a tamanho de arquivo receptivo limitado. A GSoP-Net aborda esta questão modelando as correlações em pares do tensor holístico.



5.1.3 SRM Net

O bloco SRM (*style-based recalibration module*) melhora tanto os módulos de compressão como de excitação, e pode ser adicionado após cada camada convolutiva como o SE.



A principal contribuição na compressão é a utilização de “*style pooling*”, que utiliza tanto o valor médio quanto o desvio padrão das características de entrada para melhorar a capacidade de captar informação global

Para diminuir a complexidade do bloco SE visto anteriormente, utiliza uma camada inteiramente conectada baseada em canal.

5.1.3 SRM Net

- Um bloco de Compressão e Excitação SRM (com parâmetros θ), com entrada X e saída Y é formulado matematicamente por:

$$s = F_{srm}(X, \theta) = \sigma \left(BN \left(\textcolor{red}{CFC}(\textcolor{teal}{SP}(X)) \right) \right) \quad (1)$$

$$Y = s \odot X \quad (2)$$

Em que:

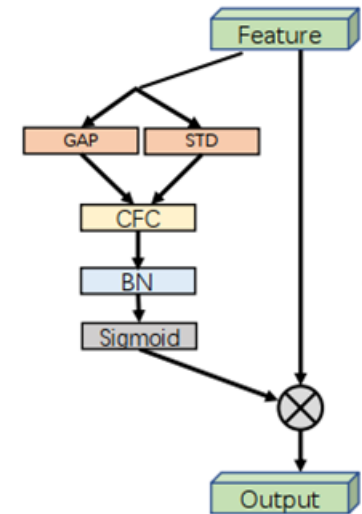
s - vetor de atenção

σ - função sigmoide

SP – *style pooling*: combina amostragem média com amostragem de desvio padrão.

CFC – *channel wise fully connected layer*

BN – *batch normalization*



5.2 Mecanismo de Atenção Espacial

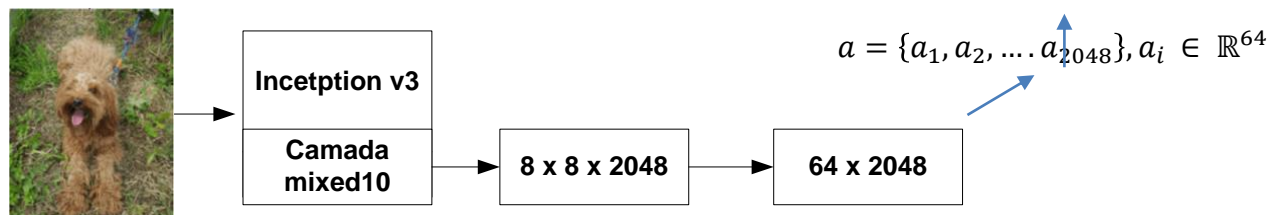
- A atenção espacial pode ser vista como um mecanismo de seleção adaptativo de regiões espaciais, indicando onde prestar atenção. Veremos dois mecanismos.
- O primeiro foi proposto por [1] e tem duas versões, o mecanismo de atenção *hard* e o mecanismo de atenção *soft*. Nesse texto veremos o mecanismo de atenção *soft*.
- O segundo utiliza uma estrutura chamada “*transformer*” e foi proposto em [2].

5.2.1 Mecanismo Soft

Esse mecanismo de atenção é mostrado no diagrama em blocos da figura a seguir. O mesmo é constituído por três blocos distintos. Geração de características ou codificador, geração do vetor de contexto e decodificador com mecanismo de atenção. A figura no próximo slide ilustra a composição desses três blocos.

Geração de Características ou codificador:

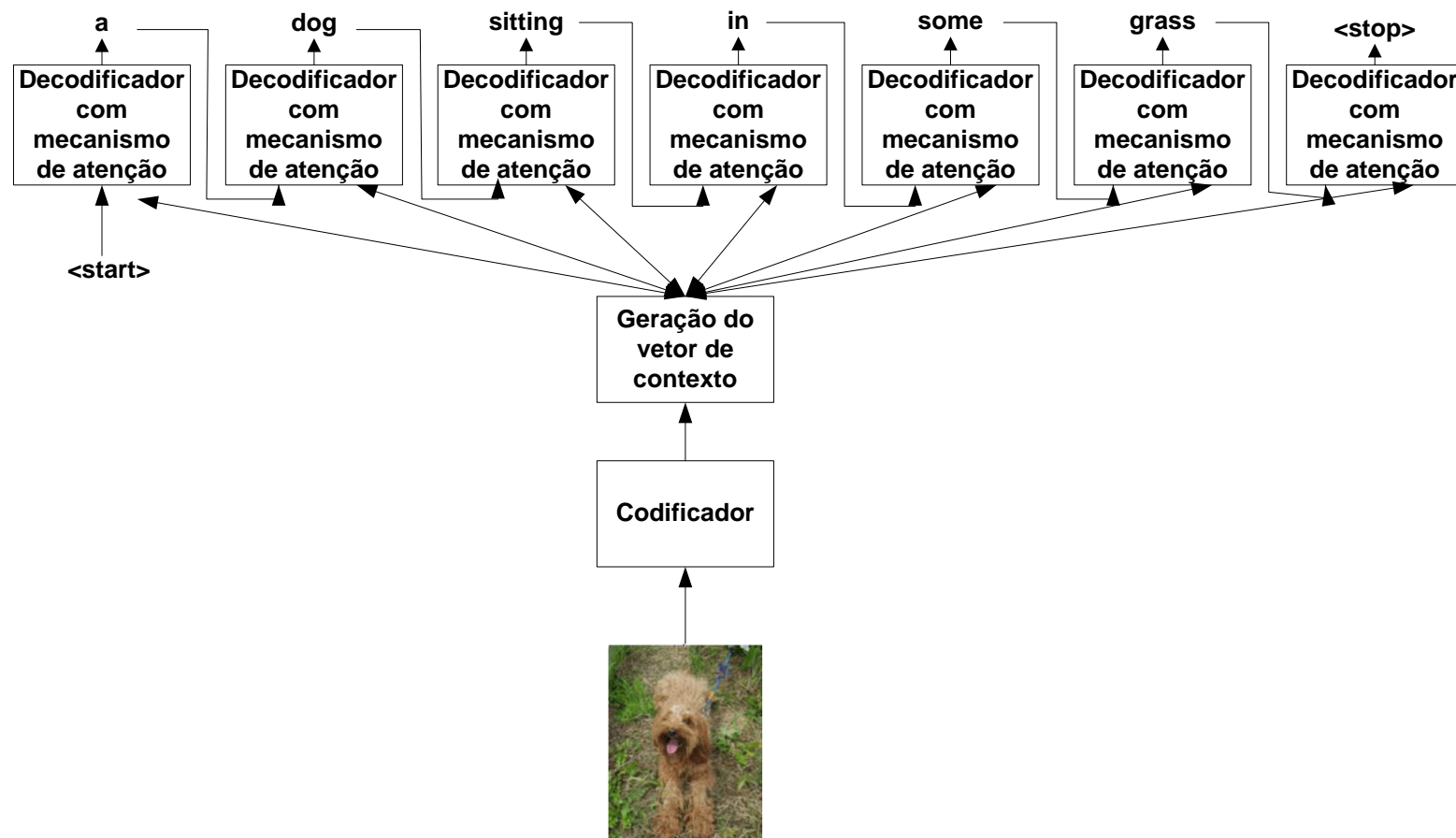
As características são extraídas a partir da camada “*mixed10*” de uma rede pré-treinada inception-v3. A figura a seguir ilustra onde se encontra essa camada na rede inception-v3. As características geradas são vetores, conforme mostrado abaixo.



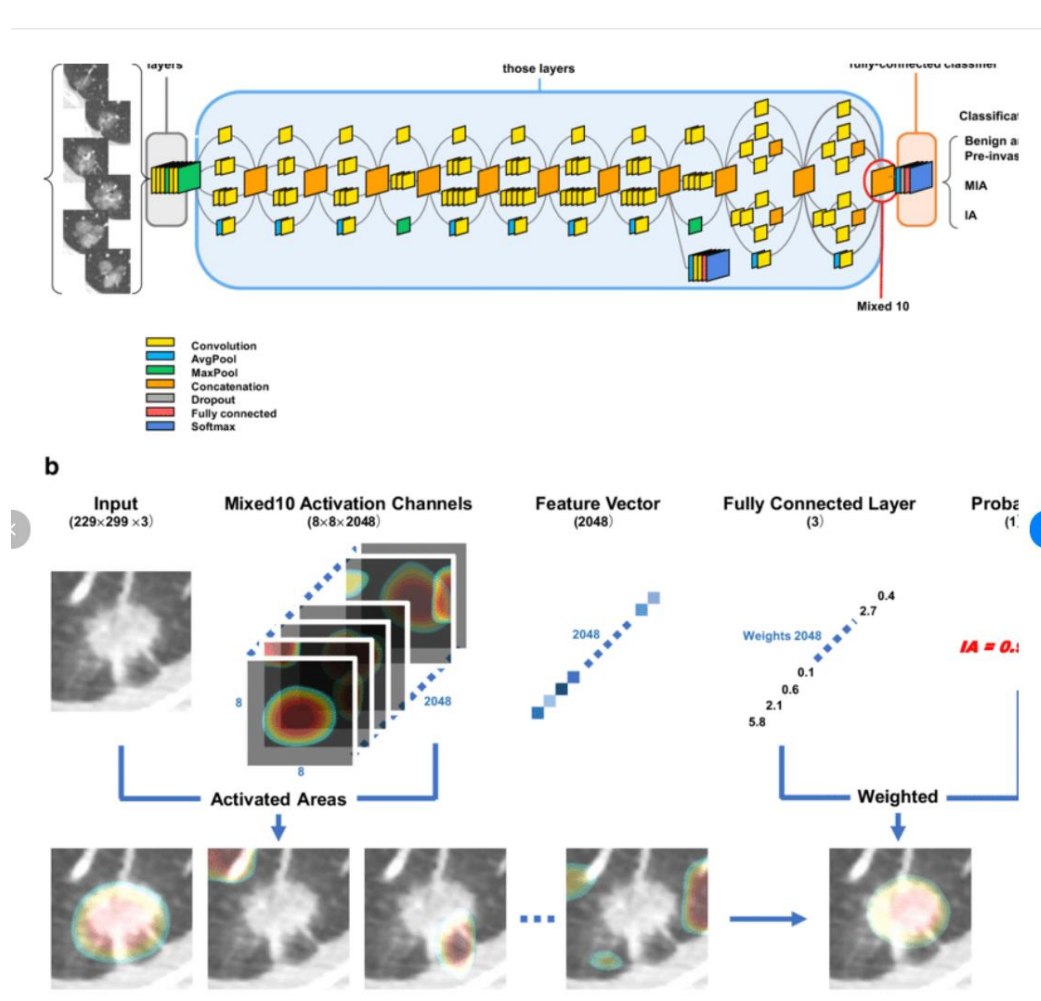
[1] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, “Show, attend and tell: Neural image caption generation with visual attention,” in International conference on machine learning. PMLR, 2015, pp. 2048–2057

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2017.

5.2.1 Mecanismo Soft



5.2.1 Mecanismo Soft



Utilização da rede inception v3 para diagnóstico de nódulos no pulmão em imagens de [CT](#)

Jiang, Beibei & Zhang, Yaping & Zhang, Lu & Bock, Geertruida & Vliegenthart, Rozemarijn & Xie, Xueqian. (2021). Human-recognizable CT image features of subsolid lung nodules associated with diagnosis and classification by convolutional neural networks. *European Radiology*. 31. 10.1007/s00330-021-07901-1.

5.2.1 Mecanismo Soft

Geração do vetor de contexto:

A partir desses vetores de características é obtido o vetor de contexto \mathbf{z}_t de entrada da rede recorrente através da função:

$$\mathbf{z}_t = \sum_i \alpha_{ti} \mathbf{a}_{ti}$$

Em que:

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{k=1}^L \exp(e_{tk})}$$

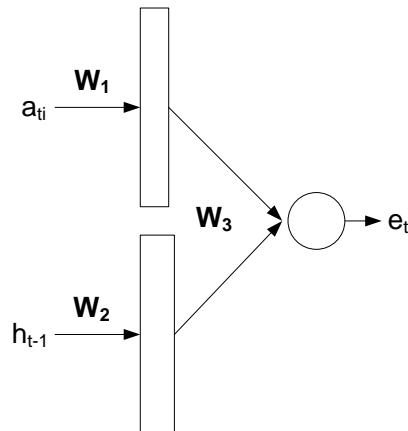
O valor e_{ti} é calculado a partir de um modelo de atenção f_{att}

$$e_{ti} = f_{att}(a_{ti}, h_{t-1})$$

Em que h_{t-1} é o estado escondido da RNN usado no decodificador.

O modelo de atenção f_{att} implementado é uma perceptron multicamadas, conforme mostrado abaixo:

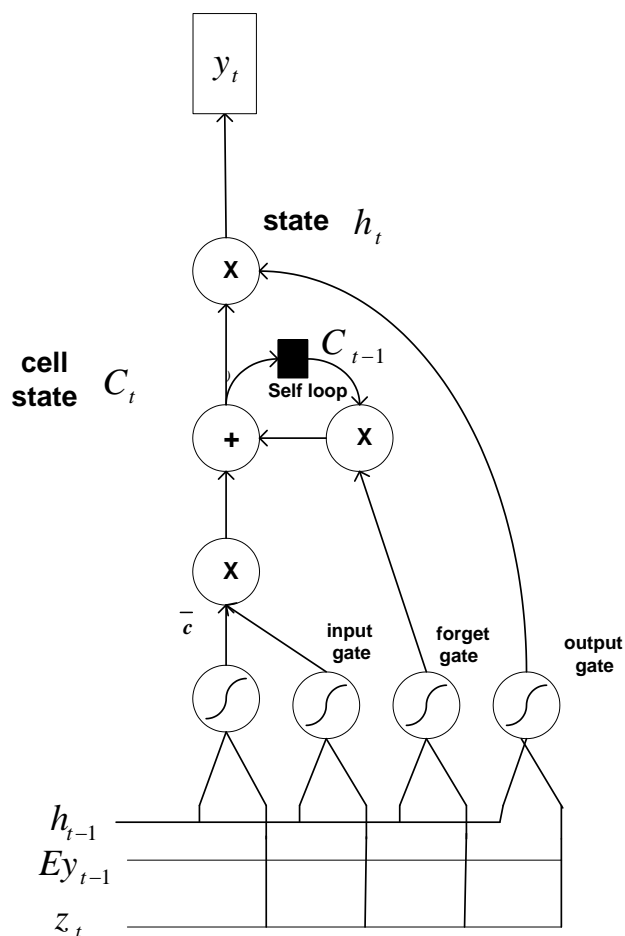
$$e_{ti} = W_3(W_1 a_i + W_2 h_{t-1})$$



5.2.1 Mecanismo Soft

Decodificador com mecanismo de atenção

A figura a seguir mostra a unidade da rede LSTM utilizada na decodificação com atenção.



Saída da rede recorrente

$$y = \{y_1, y_2, \dots, y_C\}, y_i \in \mathbb{R}^K$$

Sinais de Controle da rede recorrente

$$\begin{bmatrix} \bar{i} \\ \bar{f} \\ \bar{o} \\ \bar{c} \end{bmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \text{tanh} \end{pmatrix} W \begin{bmatrix} Ey_{t-1} \\ h_{t-1} \\ z_t \end{bmatrix}$$

Em que:

K – tamanho do vocabulário.

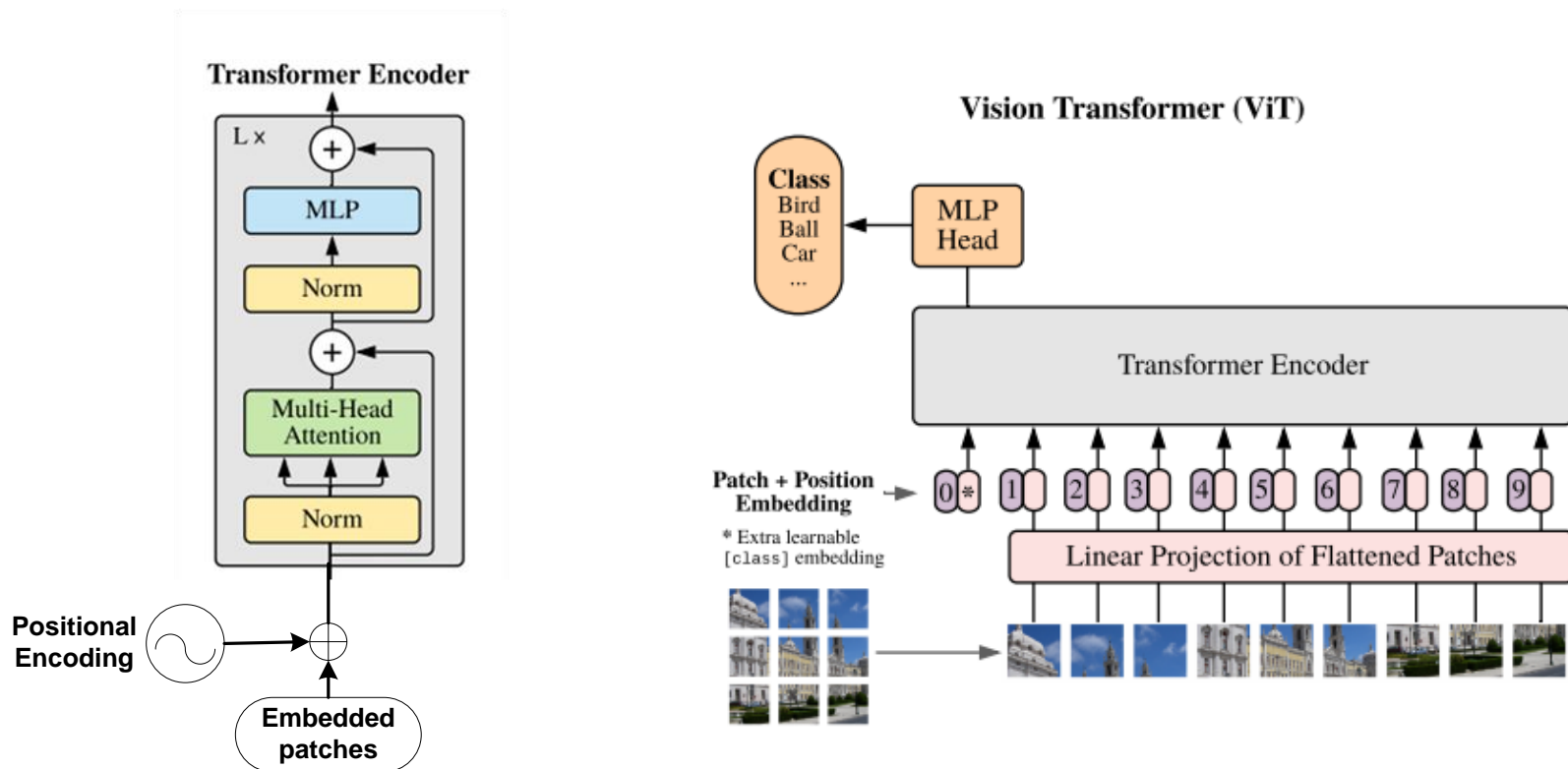
E é uma matriz $\mathbb{R}^{m \times k}$ obtida através do treinamento

z_t é o vetor de contexto

y_{t-1} é a saída da rede no instante $t - 1$

6. Transformer e mecanismos de atenção em visão computacional

Utilizando a arquitetura do *codificador do transformer*, Dosovitskiy et al. propôs a utilização do *transformer* para atenção visual, conforme mostrado no modelo abaixo. O objetivo dessa arquitetura é a classificação da imagem. Observe que não se utiliza redes convolucionais, mas tão somente o *transformer*. A seguir veremos os passos utilizados para determinação das matrizes Q, K e V.



A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," ICLR, 2021.

6. Transformer e mecanismos de atenção em visão computacional

O transformer recebe como entrada um sequência de tokens 1D, usando a seguinte transformação:

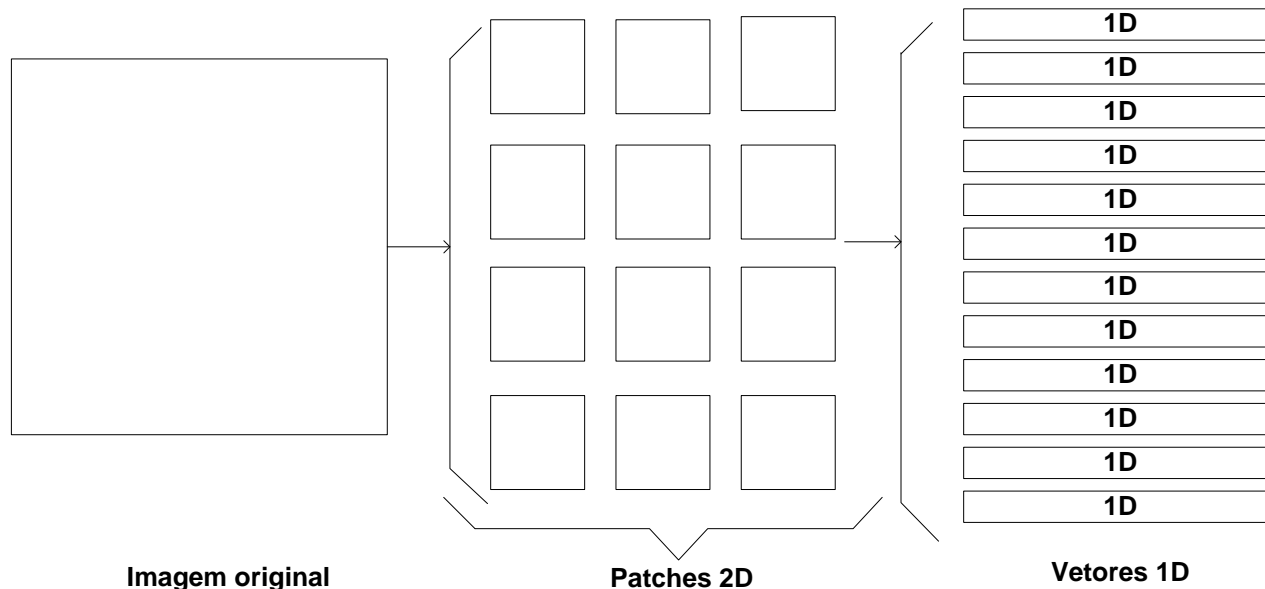


Imagem: $x \in \mathbb{R}^{H \times W \times C}$

N Patches 2D: $p_p \in \mathbb{R}^{P \times P \times C}$

N Vetores 1D: $x_p \in \mathbb{R}^{(P^2 C)}$

Em que:

H, W – dimensões da imagem original

C – Número de canais da imagem original

$P \times P$ – dimensões de cada patch

$N = HW / P^2$ – número de patches

6. Transformer e mecanismos de atenção em visão computacional

A Projeção Linear gera as matrizes Q, K e V, que são iguais e chamadas de z_0

$$z_0 = \begin{bmatrix} x_{class} \\ x_p^1 \cdot E \\ x_p^2 \cdot E \\ \dots \\ x_p^N \cdot E \end{bmatrix} + E_{pos}$$

Em que:

E – matriz de projeção $\in \mathbb{R}^{(P^2 C) \times D}$

E^{pos} – matriz de posicionamento $\mathbb{R}^{(N+1) \times D}$

x_{class} – vetor D dimensional utilizado para classificação da imagem pelo perceptron.

Matriz de Posicionamento E^{pos} :

Cada patch no transformer visual é codificado com um vetor que indica a sua posição. Essa codificação da posição é realizada através da matriz E^{pos} . As linhas dessa matriz contém a codificação da ordem de cada patch (1,2, ...N).

6. Transformer e mecanismos de atenção em visão computacional

Veremos a seguir o equacionamento do *codificador do visual transformer*, que é constituído por L blocos do *codificador transformer* mostrado anteriormente.

Bloco do *codificador do visual transformer* :

$$z'_l = MSA(LN(z_{l-1}) + z_{l-1})$$

$$z_l = MLP(LN(z'_l) + z'_l)$$

Lembrete: saída da projeção linear:

$$z_0 = \begin{bmatrix} x_{class} \\ x_p^1 \cdot E \\ x_p^2 \cdot E \\ \dots \\ x_p^N \cdot E \end{bmatrix} + E_{pos}$$

A normalização LN das saídas de uma camada é efetuada subtraindo-se cada valor da média das saídas da camada, dividindo-se pelo desvio e padrão e multiplicando-se por um fator de ajuste

O vetor de classificação da imagem de saída é feita através de um perceptron de uma camada, aplicada a saída da última camada do bloco MAS:

$$y = LN(z_L^0)$$

Lembrete: $z_0^0 = x_{class}$

6. Transformer e mecanismos de atenção em visão computacional

O site <https://huggingface.co/models> contém uma série de modelos pré-treinados e Publicados na literatura usando o *transformer* para várias aplicações.

The screenshot displays the Hugging Face website's 'Models' section. The browser's address bar shows 'huggingface.co/models'. The page features a search bar and navigation links for Models, Datasets, Spaces, Docs, Solutions, Pricing, Log In, and Sign Up. On the left, there are filters for Tasks (Image Classification, Translation, Image Segmentation, Fill-Mask, Automatic Speech Recognition, Token Classification, Sentence Similarity, Audio Classification, Question Answering, Summarization, Zero-Shot Classification, and 22 more) and Libraries (PyTorch, TensorFlow, JAX, and 30 more). Below these are filters for Datasets (mozilla-foundation/common_voice_7_0, squad, wikipedia, common_voice, glue, emotion, bookcorpus, xtreme, and 305 more) and Languages (English, French, Spanish, German, Chinese, Japanese, Russian, Portuguese). The main content area shows a list of models, including gpt2, xlm-roberta-base, bert-base-uncased, openai/clip-vit-large-patch14, roberta-base, distilbert-base-uncased, facebook/bart-base, bert-base-cased, Jean-Baptiste/camembert-nli, dmis-lab/biobert-base-cased-v1.2, roberta-large, vblagoje/bert-english-uncased-finetuned-pos, cardiffnlp/twitter-roberta-base-sentiment, and bert-base-chinese. Each model entry includes its name, update date, download count, and like count. The bottom of the image shows a Windows taskbar with various application icons and a system tray displaying the date and time as 02/11/2022 10:57.

6. Transformer e mecanismos de atenção em visão computacional

Fontes de código didático:

ViT transformer from scratch:

<https://www.kaggle.com/code/utkarshsaxenadn/vit-vision-transformer-in-keras-tensorflow>

<https://www.kaggle.com/code/raufmomin/vision-transformer-vit-from-scratch/notebook>

transformer fine-tuning

<https://www.kaggle.com/code/raufmomin/vision-transformer-vit-fine-tuning>

How to Implement Scaled Dot-Product Attention from Scratch in TensorFlow and Keras

<https://machinelearningmastery.com/how-to-implement-scaled-dot-product-attention-from-scratch-in-tensorflow-and-keras>

How to Implement Multi-Head Attention from Scratch in TensorFlow and Keras

<https://machinelearningmastery.com/how-to-implement-multi-head-attention-from-scratch-in-tensorflow-and-keras>

Implementing the Transformer Encoder from Scratch in TensorFlow and Keras

<https://machinelearningmastery.com/implementing-the-transformer-encoder-from-scratch-in-tensorflow-and-keras/>