Transformers, Efficient Attention and Vision-Language Retrieval Model

Prof. Gianluca Moro*, Dott. Ing. Stefano Salvatori^
Department of Computer Science and Engineering
University of Bologna, Cesena
*name.surname@unibo.it, ^s.surname@unibo.it

Notebook with code at the of the slides

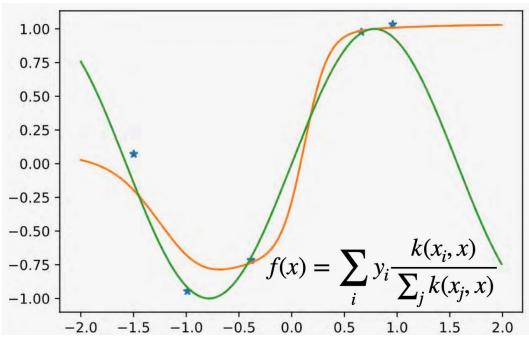
SEMINAR

Attention Mechanism: Rediscovering Old Ideas

- Data $\{x_1, ...x_m\}$ and labels $\{y_1, ...y_m\}$
- Estimate the value label *y* for new data *x:* the world's trivial estimator: average over all labels $y = \frac{1}{m}\sum_1^m y_i$
- Better idea: predicting the label *y* of a new data x by weighting the labels according to locations (Watson, Nadaraya, 1964)
- $f(x) = \frac{1}{m}\sum_1^m \phi(x, x_i) \bullet y_i$

query key value

- where $\phi(x, x_i)$ is a kernel function, e.g. gaussian kernel



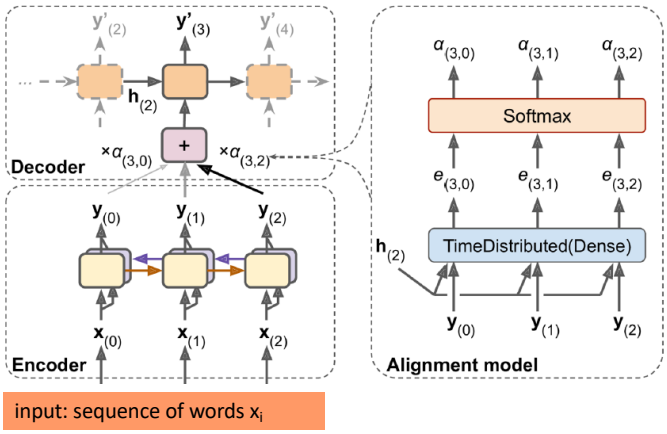
- Simple solution: no free parameters, information is in the data, no weights to learn
- Deep Learning Variant: learn weighting function

Attention Mechanism in Recurrent Neural Nets

It has been firstly included in recurrent neural networks based on Encoder-Decoder models for NLP

the figure depicts a bidirectional recurrent neural network where the input is supplied as a sequence of words to the encoder, **here the focus is in the 3rd step output y'$_{(3)}$**

The $y_{(i)}$ encoder output for each word is the output combination of the two directional RNN



The Attention consists in combining each encoder output $y_{(i)}$ (here i=0,..,2) in a new input *alphas* for the next RNN, i.e. the decoder, applying to them an operation, i.e. the concatenation (+)

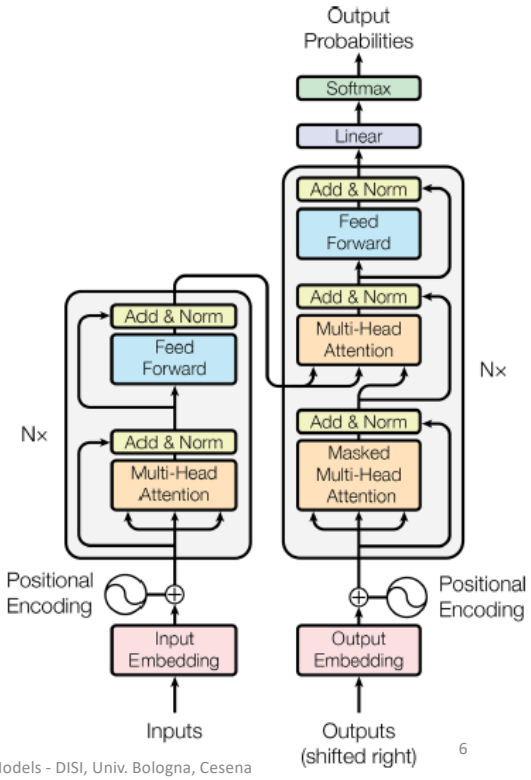
The attention learns the dependencies among words by learning the new weights alphas from the results $e_{(i,j)}$ achieved by combining the hidden states *h* of the decoder with the $y_{(i)}$ output of encoder

Concatenative and Multiplicative Attention

- The alpha weights are the result of a softmax function in the last layer of the network
  - $\alpha_{(t,i)} = \frac{\exp(e_{(t,i)})}{\sum_j \exp(e_{(t,j)})}$
  - Called $\boldsymbol{y}(i)$ the output of the encoder at timestep i, and $\boldsymbol{h}_{(t)}$ the decoder hidden state at timestep t, the way with which $e_{(t,i)}$ is computed defines different types of attentions
- *Concatenative Attention* [*Bahdanau et al., 2014]*
  - $e_{(t,i)} = \boldsymbol{v}^{\top}\tanh(\boldsymbol{W}[\boldsymbol{h}_{(t)}; \boldsymbol{y}(i)])$
  - $\boldsymbol{v}$ is a scaling parameter and [;] is the concatenate operator
- M*ultiplicative Attention [Luong et al., 2015]*
  - $e_{(t,i)} = \boldsymbol{h}_{(t)}^{\top}\boldsymbol{W}\boldsymbol{y}(\boldsymbol{i})$
  - multiplicative attention is faster and more space-efficient in practice as it can be efficiently implemented using matrix multiplication
  - The $\boldsymbol{W}$ matrix could also be non-trainable and be set to the identity matrix (***Dot Product Attention***)

"Attention is All You Need" Transformer

- The *Transformer [Vaswani et al. 2017]* is a model that allows to use the **attention mechanism without the need of recurrent neural networks**
- It replaces the recurrent cells of the RNNs with the *Multi-Head Attention Layer* that learns the relationships of an element with each other in the input, considering different semantic meanings
  - It is also known as ***self-attention*** **mechanism**
  - In the first step of the decoder this layer is 'masked': each word is related only to the previous ones
- The architecture can contain multiple encoding and decoding blocks (6 were used in the original paper)

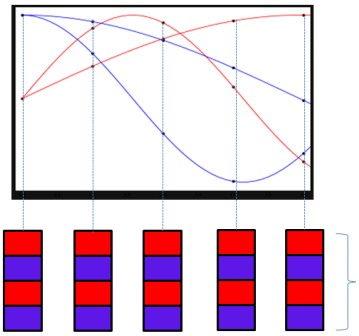


Transformer: Input and Positional Embeddings

- Both the Encoder and the Decoder take as input a matrix $\boldsymbol{Z}$ obtained by transforming each input element into a relative 'input embedding';
  - called L the input length and d the size of the embeddings, $\boldsymbol{Z}$ will have dimension $L \times d$
- The $\boldsymbol{Z}$ matrix is then summed to a matrix $\boldsymbol{P}$, called *positional embedding* matrix
  - The model can have different behaviors based on the position of an element in the sentence
  - **Positional embeddings can be fixed or learned from the network itself;** both solutions produce equivalent results
  - In the original paper, called i the position of the word in the sentence and j the index in the embedding, the positional embeddings are computed using the following formula

$$p_{i,j} = \begin{cases} \sin\left(\dfrac{i}{10000\dfrac{j}{d}}\right) & if\ j\ is\ even \\ \cos\left(\dfrac{i}{10000\dfrac{j-1}{d}}\right) & if\ j\ is\ odd \end{cases}$$

  - 10000 is an arbitrary selected coefficient to ensure a period of sin and cos sufficiently large to map each points

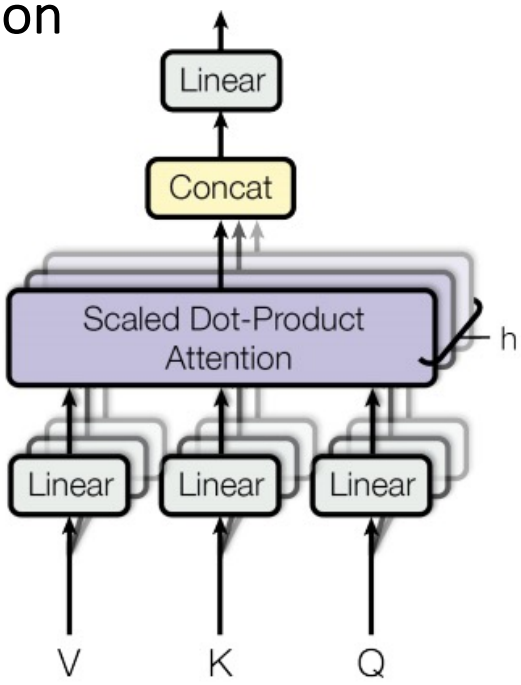


Es. dati 5 embedding di dim. 4(d_{model}) si può notare come gli indici i pari(in rosso) siano mappati su 2 frequenze di sinusoide mentre gli indici i dispari(in blu) siano mappati su 2 Frequenze di cosinusoide.
Ciò permette di mappare un gran numero di valori di embedding mantenendo però una coerenza posizionale.
(N.B. Per semplicità di esempio al il grafico corrisponde alla costante 10 invece di 10000.

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

Transformer: Multi Head Attention

- The matrix $\boldsymbol{X} \in \mathbb{R}^{L \times d} = \boldsymbol{Z} + \boldsymbol{P}$ obtained by summing input embeddings and positional embeddings, gets then processed by the **Multi Head Attention layer**
- This layer computes the *Scaled Dot-Product Attention* h times with different weights
  - Each output is indeed obtained with different projections of the embeddings in the $\boldsymbol{X}$ matrix
  - This layer therefore learns different semantic relationships between the input elements
- The h results are eventually concatenated and brought back to $L \times d$ dimension trough a lineare layer



Transformer: Scaled Dot-Product Attention

- The scaled dot-product attention is computed employing 3 matrices: keys ($\boldsymbol{K}$), queries ($\boldsymbol{Q}$) e values ($\boldsymbol{V}$)
  - $\boldsymbol{V_i} \in \mathbb{R}^{L \times d_v} = \boldsymbol{X}\boldsymbol{W}_i^V,\ W_i^V \in \mathbb{R}^{d \times d_v}$
  - $\boldsymbol{K_i} \in \mathbb{R}^{L \times d_k} = \boldsymbol{X}\boldsymbol{W}_i^K,\ W_i^K \in \mathbb{R}^{d \times d_k}$
  - $\boldsymbol{Q_i} \in \mathbb{R}^{L \times d_k} = \boldsymbol{X}\boldsymbol{W}_i^Q,\ W_i^Q \in \mathbb{R}^{d \times d_k}$
  - The matrices are different for each multi-head attention layer *h*, therefore $i = 0, 1, \ldots h-1$
  - Usually we have $d_k = d_v = \frac{d}{h}$
- Given $\boldsymbol{Q}, \boldsymbol{K}\ and\ \boldsymbol{V}$ matrices, the scaled dot-product attention is obtained as follows

$$Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}}\right)\boldsymbol{V}$$

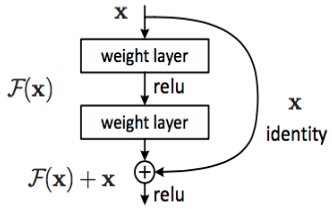
- $\boldsymbol{Q}\boldsymbol{K}^\top \in \mathbb{R}^{L \times L}$ represents the similarity between different projections of the emebddings
  - $\sqrt{d_k}$ is a scaling factor used to avoid saturation of the softmax function
- Time and Space complexity are $O(L^2 d)$ and $O(L^2 + Ld)$ respectively

Residual Learning Building Block

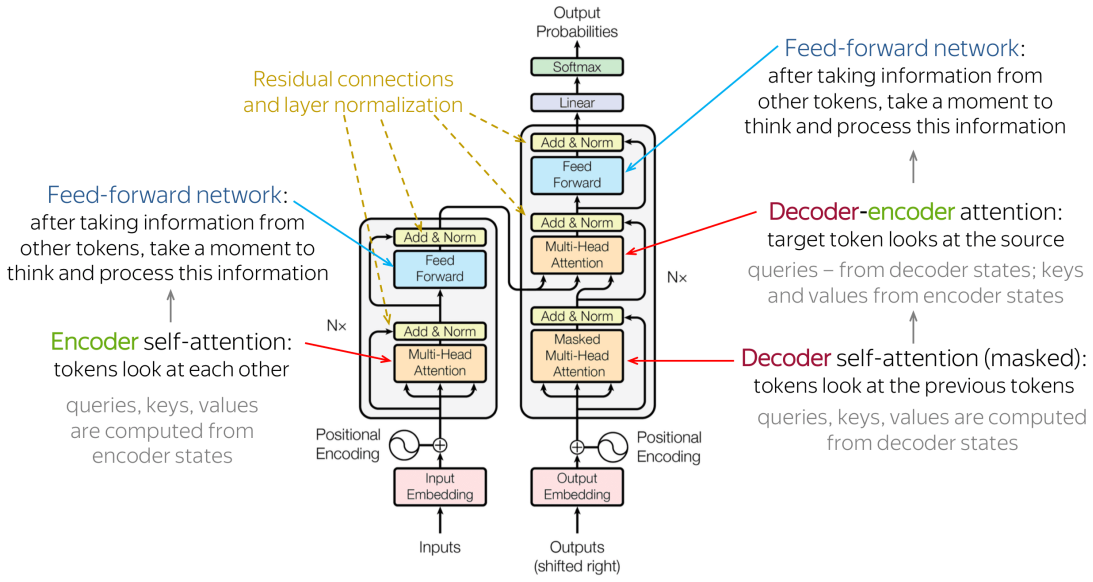
- Deep neural networks tend to *forget* some features of their input data-set samples during training
- This problem is circumvented by summing the input x to the result of a typical feed-forward computation $\mathcal{F}$:

$$\mathcal{F}(x) + x = [W_2 \cdot \sigma(W_1 \cdot x + b_1) + b_2] + x$$

- The residual connection is schematically represented as



Transformer: Residual Connections

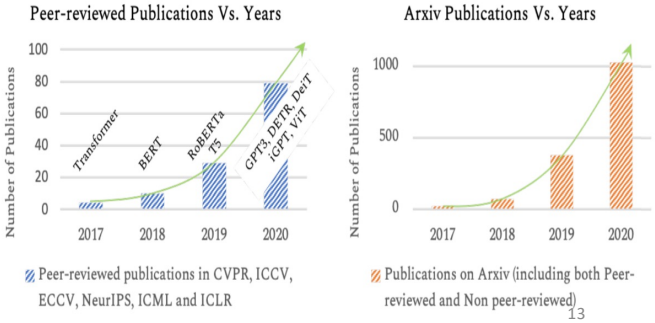


Why Residual Connections in Transformers ?

- Residual connections are motivated in very deep networks
- But attention blocks perform very little computations compared to the networks that were outperformed
  - *Deep Residual Learning for Image Recognition* K. He et al., 2015
- So, what is the motivation of residuals in the attention-blocks of transformer architectures ?
- Two reasons
  1. mitigates the vanishing gradient problem (with ReLU activation function the gradient is zero for theoretically half of the cases)
  2. attention mechanism allows an arbitrary information flow in the network and thus arbitrary, also insensate, permutations of the input tokens. Residual connections *remind* the original state, i.e. the contextual representations of the input tokens are still tokens

Transformer: Results

- Transformers improved the state-of-the-art in several NLP tasks with lower training costs since recurrent connections are not used
- Applications in: Text Classification, Machine Translation, Summarization, Question Answering…
- The most significant advances were made in NLP thanks to the pre-training and fine-tuning strategy introduced in *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*
- The figure on the right shows the number of times keywords such as BERT, Self-Attention and Transformers have appeared in peer-reviewed and arXiv article titles in recent years (in Computer Vision and Machine Learning)

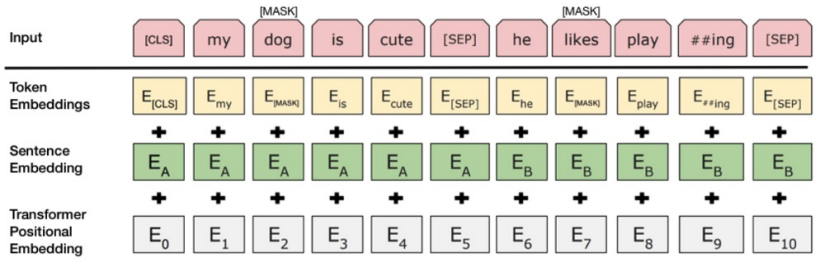


BERT - Bidirectional Encoder Representations from Transformers

- BERT [Devlin et al., 2018] is a network for Natural Language Processing based on the Transformer architecture
- It employs a stack of N encoding blocks, removing the decoder from the model
- It is called *Bidirectional* since, using Transformers, attention is computed in both directions
  - Every word in the sentence is related to every other one, including the following ones
  - This is a substantial difference with respect to previous works (e.g. OpenAI GPT *[Radford et al, 2017])* in which a word could only be related to the preceding words
- BERT introduces two self-supervised tasks for pre-training
  - **Masked Language Modeling***: t*he model is asked to predict randomly masked words within a sentence
  - **Next Sentence Prediction***:* the network is given two sentences and is asked it to predict whether the second sentence semantically follows the first one

BERT: Input

- BERT uses positional embeddings learned during training
- Each token is also assigned a *sentence embedding* (or *token type embedding*) which is summed to the input matrix
  - This is used to differentiate multiple inputs (e.g., the two sentences in the Next Sentence Prediction Task)
- Each sentence in input is separated by a special *[SEP]* token
- A special token *[CLS]* is added at the beginning of the input, which can be eventually used for classification



BERT: Masked Language Modeling

- The purpose of this task is to make the network **predict masked words within a sentence**
- 15% of the words in each sentence are masekd in different ways
  - 80% substituted with a *[MASK]* token (*my dog is hairy => my dog is [MASK])*
  - 10% replaced with a random word (*my dog is hairy => my dog is apple)*
  - 10% of the times they are left untouched (*my dog is hairy => my dog is hairy)*
- The fact that words are not always replaced with the [MASK] token is required to facilitate the fine-tuning phase in which this token will not be present
- The loss used is the *cross-entropy loss* computed over the masked-out tokens
  - Called M the number of masked token, t the one-hot encoded vectors of the correct words and p the softmax output returned by the model

$$L_{MLM} = -\frac{1}{M}\sum_{i=1}^{M} \mathbf{t}_i \log(\boldsymbol{p}_i)$$

BERT: Next Sentence Prediction

- The purpose of this task is to make the model **understand if a given sentence logically follows another one**
- During training 50% of sentence pairs are given in the correct order, while the remaining half are pairs of sentences randomly selected from the corpus
- According to the authors, pre-training the network on this task leads to an improvement in performance on question answering and natural language inference problems
  - This type of tasks requires indeed to understand relationships between sentences, a skill that is more difficult to learn using only the masked language modeling task
- The loss used is a *binary cross entropy loss*
  - Called $y \in (0,1)$ the correct class and p the probability assigned by the model

$$L_{NSP} = -(y \log(p) + (1-y)\log(1-p))$$

BERT: Training

- BERT is pretrained on *BookCorpus* (800M words) and *English Wikipedia (*2,500M words)
- Each sentence is tokenized using *WordPiece [Wu et al., 2016]*
- The pair of phrases in input to the network was chosen so that the length of their concatenation did not exceed **512 tokens**
- Two versions of BERT have been trained; they are distinguished by: number of encoding blocks (L), size of hidden layers (H) and size of multi head attention layer (A)
  - $BERT_{BASE}$: L=12, H=768 e A=12 (110M di parametri)
  - $BERT_{LARGE}$: L=24, H=1024 e A=16 (340M di parametri)
- After the pre-training phase, **finetuning can be performed adding an additional output layer** which should be designed to address the specific downstream task

BERT: Results

- **GLUE:** *General Language Understanding Evaluation [Wang et al., 2018]* is a collection of diverse natural language understanding tasks.
  - $BERT_{BASE}$ managed to outperform its predecessor OpenAI GPT by improving the state of the art by 4.5% on average across all tasks
  - $BERT_{LARGE}$ has obtained even better results improving average results from 79.6% to 82.1%

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$BERT_{BASE}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$BERT_{LARGE}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

BERT: Results

- **SQUAD**: *Standford Question Answering Dataset [Rajpurkar et al., 2016]* is a collection of 100k crowdsourced question/answer pairs
  - Given a question, the model must find the answer within a provided text passage
  - Results are measured in terms of F1-score and Exact Match (percentage of answers that exactly match the correct one)
- Also in this case BERT, and in particular $BERT_{LARGE}$ trained on the additional TriviaQA dataset *[Joshi et al., 2017],* has improved the results achieved by all previous works
  - In the ensemble version, 7 models are used with different pre-training checkpoints and fine-tuning seeds

System	Dev		Test	
	EM	F1	EM	F1
Leaderboard (Oct 8th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
#1 Single - nlnet	-	-	83.5	90.1
#2 Single - QANet	-	-	82.5	89.3
Published				
BiDAF+ELMo (Single)	-	85.8	-	-
R.M. Reader (Single)	78.9	86.3	79.5	86.6
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$BERT_{BASE}$ (Single)	80.8	88.5	-	-
$BERT_{LARGE}$ (Single)	84.1	90.9	-	-
$BERT_{LARGE}$ (Ensemble)	85.8	91.8	-	-
$BERT_{LARGE}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$BERT_{LARGE}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

BERT: Results

- **Named Entity Recognition**: This task consists in recognizing entities in a text. The *CoNLL-2003* dataset was used *[Tjong Kim Sang and De Meulder, 2003]*, consisting of 200k words annotated as Person, Organization, Location, Miscellaneous or Other (unnamed entity)

System	Dev F1	Test F1
ELMo+BiLSTM+CRF	95.7	92.2
CVT+Multi (Clark et al., 2018)	-	92.6
$BERT_{BASE}$	96.4	92.4
$BERT_{LARGE}$	**96.6**	**92.8**

Results on Named Entity Recognition

- **SWAG:** *Situations With Adversarial Generations [Zellers et al., 2018]* is a datest containing 113k sentence pairs. Given a sentence, the task is to choose the most plausible continuation among four choices

System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
$BERT_{BASE}$	81.6	-
$BERT_{LARGE}$	**86.6**	**86.3**
Human (expert)†	-	85.0
Human (5 annotations)†	-	88.0

Results on SWAG

BERT: Variants - RoBERTa

- **RoBERTa** (*Robustly Optimized BERT Pretraining Approach) [Liu, Yinhan, et al., 2019]* is a BERT variant that shows how the pre-training strategy proposed in the orginal BERT paper could be improved
- The main modifications proposed by the authors are the following
  - **Dynamic Masking**: Masking is performed dinamically during training for each sentence, instead of using static pre-processed masking
  - **Remove Next Sentence Prediction Task***:* The authors showed that this task is not needed to improve downstream results on the benchmark datasets
  - **Bigger batch size***:* BERT was pre-trained with a batch size of 256 for 1M steps; they propose to use a batch size of 2K sequences for a total of 125K steps, proving that this leads to better results with reduced training times

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
$\text{BERT}_{\text{LARGE}}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$\text{XLNet}_{\text{LARGE}}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-

BERT: Variants - BART

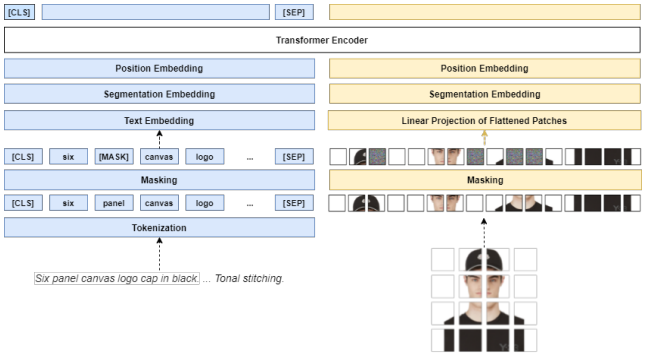
- **BART** *[Lewis, et al. 2020]* is designed to combine BERT bi-directional encoder and GPT autoregressive decoder in a single architecture
- BART main objective is text generation, but it also obtains competitive results in other tasks
- It is pre-trained in different *denoising* tasks to train the model to reconstruct document corrupted in different ways
  - **Token Masking***:* As in BERT, some tokens may be replaced with a special [MASK] token
  - **Token Delition***:* Random tokens are deleted from the input. The model must predict the missing indices
  - **Token Infilling***:* Token Masking applied on sequences of tokens
  - **Sentence Permutation***:* A document is divided into sentences based on full stops, and these sentences are shuffled in a random order
  - **Document Rotation***:* A token is chosen uniformly at random, and the document is rotated so that it begins with that token. This task trains the model to identify the start of the document.
- BART performs comparably to RoBERTa on discriminative tasks, and achieves new state-of-the-art results on several text generation tasks

Cross-Modal Transformers

- Given the success of Transformers in NLP tasks, several works in the literature have tried to extend their architecture to be able to manage inputs of various types. These models are often referred to as **Cross-Modal** or **Multi-Modal Transformers**
- Many solutions have been proposed in particular to combine texts and images (**Vision and Language (V+L) Transformers**)
  - Often, they differ by the training strategies and the way in which they process the images inside the transformer
- Both general purpose (e.g., *ViLBERT, OSCAR, ImageBERT, Unicoder VL, ViLT)* and domain specific (e.g., *FashionBERT, KaleidoBERT* for the fashion domain)
- Possible applications are
  - Visual-Question Answering
  - Text-Image Classification
  - Image Captioning
  - Cross-Modal Retrieval

Sample Architecture for a Vision-Language Transformer

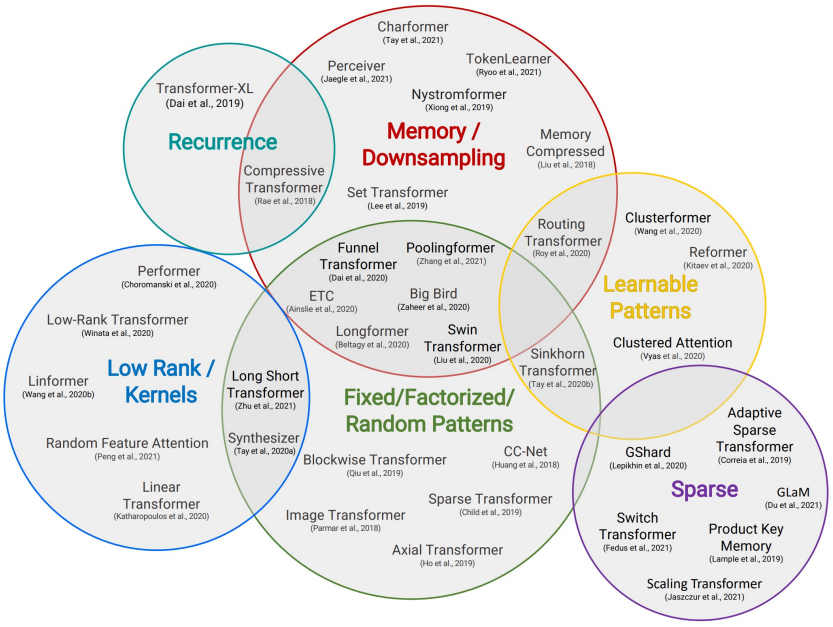
- Two channels: one for text (blue in the figure) and one for images (yellow)
  - The image $x \in \mathbb{R}^{C \times H \times W}$ is split in P squared patches which are then flattened into a sequence $x_p \in \mathbb{R}^{N \times P^2 \cdot C}$ with $N = \frac{HW}{P^2}$; then they are transformed into embeddings through a convolutional layer
  - The representations obtained for the two modalities are concatenated and passed to the Transformer attention layers
  - Masking is optional and is usually used during pre-training
- The CLS token can be used as a holistic representation of the input text and image



Optimizations for the Attention Mechanism

- The quadratic space-time complexity of the attention mechanism led to the development of approximate but efficient algorithms for its computation
  - They solve the problem of the limited input length
  - They reduce training times
  - They allow to use bigger batch sizes
- The most popular strategies proposed in literature are the following
  - ***Fixed/Learnable Patterns****:* compute the attention only between blocks of size B (with $B \ll L$) of the input; complexity from $O(L^2)$ to $O(B^2)$; the way in which the blocks are built can be learned from the network itself
  - ***Memory****:* use additional "memory" in the form of global tokens that allow to aggregate information and reduce the number of attended tokens
  - ***Low-Rank Methods****:* use a low-rank approximation of the attention matrix which projects the vectors into a lower dimensional space
  - ***Kernels****:* they use kernel functions and mathematical re-writing to be able to express attention without explicitly calculating the $L \times L$ matrix
  - ***Recurrence****:* they are based on the 'fixed patterns' mechanism but introduce recurrent connections between blocks

Optimizations for the Attention Mechanism



Tay, Yi, et al. *"Efficient transformers: A survey."* (2020)

Performer

- Kernel-based solution proposed in *[Choromanski et al., 2020]* to bring the complexity of the attention mechanism from quadratic to linear
  - Time complexity is reduced from $O(L^2 d)$ to $O(Lrd)$ while spatical complexity goes from $O(L^2 + Ld)$ to $O(Lr + Ld + rd)$ (with r hyperparameter such that $0 < r \ll L$)
- We can rewrite attention using the following formulation:

$$Att(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{V}, \qquad \boldsymbol{A} = \exp\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}}\right), \qquad \boldsymbol{D} = diag(\boldsymbol{A}\mathbf{1}_L)$$

- The entries in the matrix $\boldsymbol{A} \in \mathbb{R}^{L \times L}$ can be thought as the result of a kernel K: $\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$ applied to the matrices $\boldsymbol{Q}$ an $\boldsymbol{K}$
  - In transformers attention we have $K(\boldsymbol{x}, \boldsymbol{y}) = \exp(\boldsymbol{x}^\top \boldsymbol{y})$
- Performer manages to avoid explicitly computing $\boldsymbol{A}$ using an approach called FAVOR+ (*Fast Attention Via Positive Orthogonal Features*) which allows to approximate, the kernel K

Performer: FAVOR+

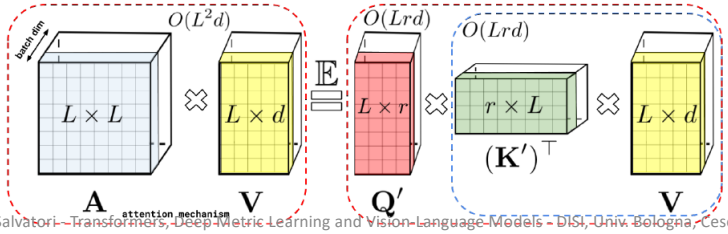
- The idea is to define a function ϕ called *positive random feature map* $\phi: \mathbb{R}^d \to \mathbb{R}_+^r$ such that:

$$K(\boldsymbol{x}, \boldsymbol{y}) = \mathbb{E}[\phi(\boldsymbol{x})^\top \phi(\boldsymbol{y})]$$

  - It can be shown that this holds if, given $r > 0$, we define $\phi(\boldsymbol{x}) = \frac{h(\boldsymbol{x})}{\sqrt{r}} f(\boldsymbol{W}\boldsymbol{x})^\top$ where $\boldsymbol{W} \in \mathbb{R}^{r \times d}$ is a matrix of random orthogonal vectors, $h(\boldsymbol{x}) = \exp(-\frac{\|\boldsymbol{x}\|^2}{2})$ and $f = \exp(\cdot)$
- With this trick, Attention can be approximated in linear time with respect to the size of the input by rewriting the formula as follows:

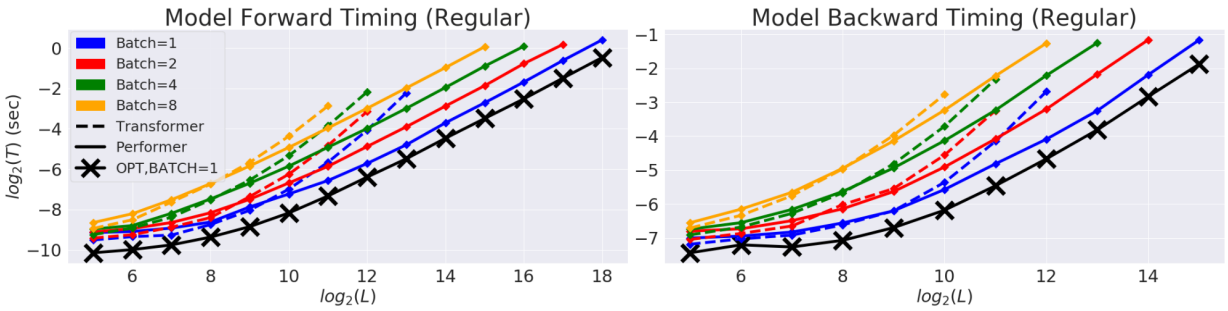
$$\widehat{Att}\,(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \widehat{\boldsymbol{D}}^{-1}\left(\boldsymbol{Q}'\left((\boldsymbol{K}')^\top \boldsymbol{V}\right)\right), \qquad \widehat{\boldsymbol{D}} = diag(\boldsymbol{Q}'\left((\boldsymbol{K}')^\top \mathbf{1}_L\right))$$

  - With $\boldsymbol{Q}', \boldsymbol{K}' \in \mathbb{R}^{L \times r}$ computed applying the function ϕ to the rows of the matrices $\boldsymbol{Q}$ *and* $\boldsymbol{K}$



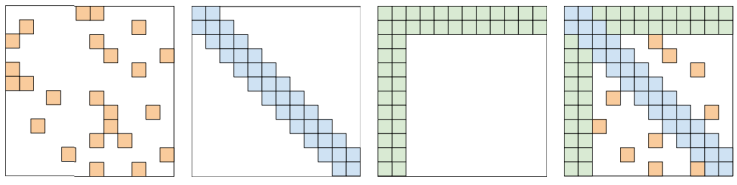
Performer: Results

- The log-log graph below shows, as L varies, the differences between training times for a vanilla Transformer (dashed lines) and Performer (solid lines); the × represent the maximum achievable speed (calculated using a 'dummy' attention block that does not perform any calculations); data is shown until the training has produced a 'memory error'



BigBird: Sparse Graph Attention

- BigBird *[Zaheer et al., 2020]* is a model with linear complexity that approximates attention using fixed pattern and memory mechanisms
  - Use sparse graphs in which the nodes are tokens of the input sequence, and the arcs represent the connections in the attention matrix
- BigBird combines 3 types of sparsification:
  - **random attention***:* each node is randomly linked to other r nodes
  - **window attention**: each node is connected to its w neighboors ($\frac{w}{2}$ on its left and $\frac{w}{2}$ on its right); this choice comes from the heuristic consideration for which the most important information is generally local information
  - **global attention***:* g nodes are completely connected
  - the authors proved that with this type of connections, BigBird becomes a universal approximator of sequence-to-sequence functions and is Turing complete



(a) Random attention (b) Window attention (c) Global Attention (d) BigBird

BigBird: Results

- Two models have been proposed
  - **BigBird-ITC** *(Internal Transformer Construction):* Global tokens are selected from within the input sequence
  - **BigBird-ETC** *(Extended Transformer Construction)*: Global tokens are newly added tokens at the start of the input
  - A LARGE version was also implemented for both of them with 24 hidden layers of size 1024 and 24 'heads' of attention

Parameter	BIGBIRD-ITC	BIGBIRD-ETC
Block length, b	64	84
# of global token, g	$2 \times b$	256
Window length, w	$3 \times b$	$3 \times b$
# of random token, r	$3 \times b$	0
Max. sequence length	4096	4096
# of heads	12	12
# of hidden layers	12	12
Hidden layer size	768	768

- State-of-the-art in question answering and text-classification
- Improved results in summarization tasks since **it can handle longer input sequences (i.e. up to 4096 token**)
  - In particular BigBird-Pegasus, which is specifically pre-trained to generate summaries

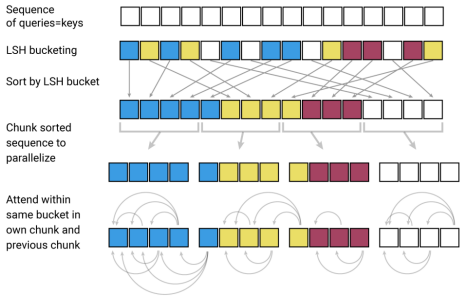
	Model	Arxiv			PubMed			BigPatent		
		R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
Base	Transformer	28.52	6.70	25.58	31.71	8.32	29.42	39.66	20.94	31.20
Base	+ RoBERTa	31.98	8.13	29.53	35.77	13.85	33.32	41.11	22.10	32.58
Base	+ Pegasus	34.81	10.16	30.14	39.98	15.15	35.89	43.55	20.43	31.80
Base	BIGBIRD-ROBERTA	41.22	16.43	36.96	43.70	19.32	39.99	55.69	37.27	45.56
Large	Pegasus (Reported)	44.21	16.95	38.83	45.97	20.15	41.34	52.29	33.08	41.75
Large	Pegasus (Re-eval)	43.85	16.83	39.17	44.53	19.30	40.70	52.25	33.04	41.80
Large	BIGBIRD-PEGASUS	**46.63**	**19.02**	**41.77**	**46.32**	**20.65**	**42.33**	**60.64**	**42.46**	**50.01**

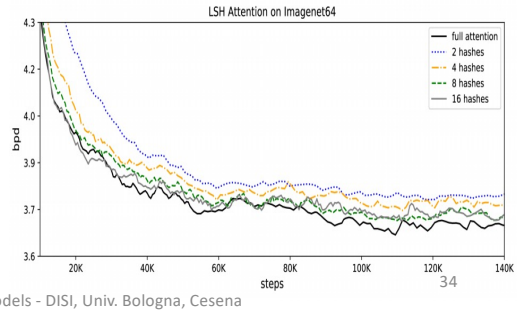
Reformer

- Reformer *[*Kitaev *et al., 2020]* is a model with space-time complexity $O(L\ logL)$ which approximates attention using *learnable patterns*
  - It is base on a technique called **Locality Sensitive Hashing (LSH)** which allows to reduce the number of tokens for which attention is computed
- Reformer also uses shared parameters between $\boldsymbol{W_Q}$ and $\boldsymbol{W_K}$ matrices (i.e $\boldsymbol{Q = K}$)
- It introduces *Reversible Residual Layers* in the Transformer architecture
  - They compute the input of each layer on demand during the backpropagation, rather than saving it in memory during the forward pass
  - This further reduces memory occupation of the model
- It can handle sequences up to 1M tokens with 16GB of memory
  - It was tested with text lengths of 64K tokens on enwik8 and also with images treated as sequences of 12K tokens on ImageNet
- It achieves almost equivalent results to a normal Transformer but with less memory

Reformer: LSH Attention

- *Locality Sensitive Hashing* consists in using a hash function h that assigns the same bucket to similar vectors
  - Reformer uses $h(x) = argmax([xR; -xR])$
  - $x \in \mathbb{R}^d$ is the input vector, $R \in \mathbb{R}^{d \times \frac{b}{2}}$ is a matrix of random gaussian floats and b is the number of buckets; $[u; v]$ is the concatenate operation
- Reformer computes attention weights only between vectors of $\boldsymbol{Q}$ and $\boldsymbol{K}$ belonging to the same bucket
  - They are the most similar to each other and therefore would contribute more to the attention result
  - The same hash function can be applied several times with different matrices R to reduce the chance that similar vectors end up in different buckets





Transformer-XL

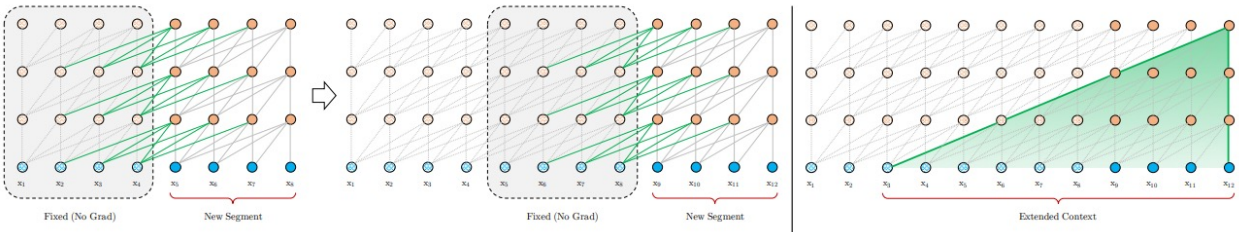
- Unlike previous models, Transformer-XL *[Dai et al., 2019]* does not directly reduce the computational complexity of the attention, which remains $O(L^2)$, but still allows to potentially process very long documents
  - To process a long input with a normal Transformer, it is necessary to divide it into segments and process each one separately, losing the inter-dependencies
  - Transformer-XL introduces a recurrence mechanism that allows to model the dependency between consecutive segments with a sort of 'additional memory'
- The idea is that the outputs computed for a certain input are stored in a cache to be reused as extended context in the arrays $\mathbf{W_K}, \boldsymbol{W_V}$ of the subsequent inputs.
- *Relative Positional Encoding* must be used instead of *Absolute Positional Encoding*
  - This is necessary to ensure that the position embeddings are consistent between segments processed consecutively

Transformer-XL: Segment-Level Recurrence

- During training, the n-th hidden state of segment s_τ (i.e., $h_\tau^n \in \mathbb{R}^{L \times d}$) is cached and used as input to the hidden layer $h_{\tau+1}^{n+1}$ of the next segment $s_{\tau+1}$

$$\tilde{h}_{\tau+1}^{n-1} = [SG(h_\tau^{n-1}); h_{\tau+1}^{n-1}]$$
$$q_{\tau+1}^n, k_{\tau+1}^n, v_{\tau+1}^n = h_{\tau+1}^{n-1} W_q^\top, \tilde{h}_{\tau+1}^{n-1} W_k^\top, \tilde{h}_{\tau+1}^{n-1} W_v^\top$$
$$h_{\tau+1}^n = TransformerEncoder(q_{\tau+1}^n, k_{\tau+1}^n, v_{\tau+1}^n)$$

  - $SG(\cdot)$ is the stop-gradient function, $[u; v]$ is the concatenate operator
  - the maximum dependence length grows linearly with respect to the number of layers and the number of segments (i.e., $O(N \times L)$)

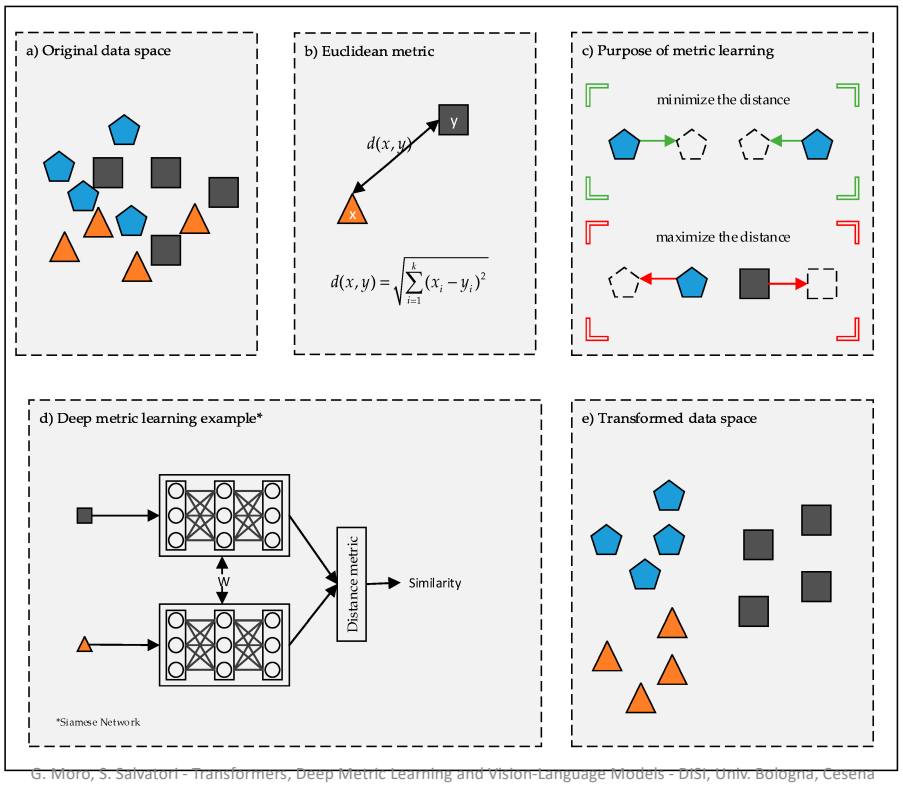


(a) Training phase. (b) Evaluation phase.

Deep Metric Learning and Information Retrieval

Metric Learning

- Aims to find a latent embedding space that allows to separate the data according to a given metric, keeping semantically similar records close to each other while separating the dissimilar ones
- The generated space can be used to perform various tasks
  - K-NN classification
  - Clustering
  - **Information Retrieval**
- 2 types of training:
  - **Supervised Metric Learning:** Each data is labeled and belongs to a specific class; the objective in this case is to learn a metric that brings the records belonging to the same class (also called positive) closer together while moving the records of different (or negative) away
  - **Weakly Supervised Metric Learning**: There are no labels available, but positive and negative pairs can be inferred from the dataset



Kaya, Mahmut, and Hasan Şakir Bilge. "Deep metric learning: A survey." *Symmetry* 11.9 (2019): 1066.

Mahlanobis Distance and Deep Metric Learning

- Mathematically we want to find a Mahlanobis matrix M (positive semidefinite), such that, taking two similar elements a and p and one dissimilar n, we have $d_M(a,b) < d_M(a,n)$
  - $d_M(x,y)$ is the Mahlanobis distance and is equal to $\sqrt{(x-y)^\top M(x-y)}$

- Each Mahlanobis matrix can be written as $M = W^\top W$ so we have
  - $d_M(x,y) = \sqrt{(x-y)^\top W^\top W(x-y)} = \sqrt{(Wx-Wy)^\top(Wx-Wy)} = \|Wx-Wy\|_2$
  - The Mahlanobis distance in the original data space corresponds to the Euclidean distance between the data projected into a latent space via a transformation W

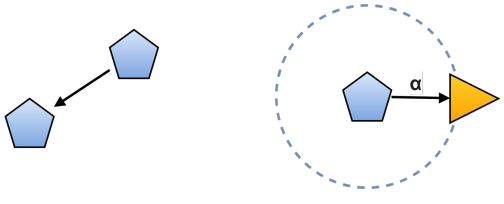
- In **deep metric learning,** deep neural networks (e.g., Transformer, CNN) are used with trainable weights θ to learn the transformation W_θ
  - Deep neural networks allow to use raw input, without manual extraction of the features, which can be complex when working with unstructured data as is for text or images

Loss Functions: Contrastive Loss

- Is necessary to define losses that force embeddings of similar elements (i.e., with the same label) to be closer than the dissimilar ones
- One of the first and most intuitive loss proposed was the ***Pairwise Ranking Loss or Contrastive Loss***
  - Given input x_1 and x_2 with labels $y_1$and y_2 the loss is defined as:

$$L_{contrastive} = \begin{cases} \|W_\theta x_1 - W_\theta x_2\|_2^2 & if\ y_1 = y_2 \\ \max(0, \alpha - \|W_\theta x_1 - W_\theta x_2\|_2^2) & if\ y_1 \neq y_2 \end{cases}$$

  - Encourages inputs with the same label to have a low distance, and inputs with different labels to have a distance greater than a margin α

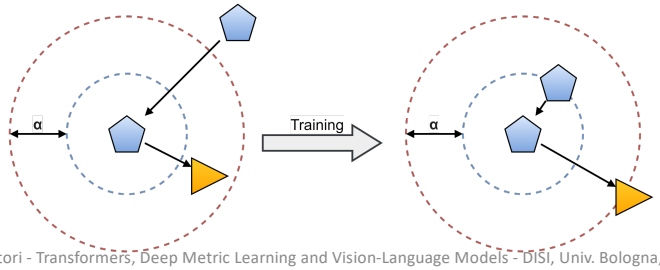


Loss Functions: Triplet Loss

- The most popular loss used in metric learning is the **Triplet Loss**, which works with triplets of elements x_a, x_p, x_n with $y_a = y_p \neq y_n$

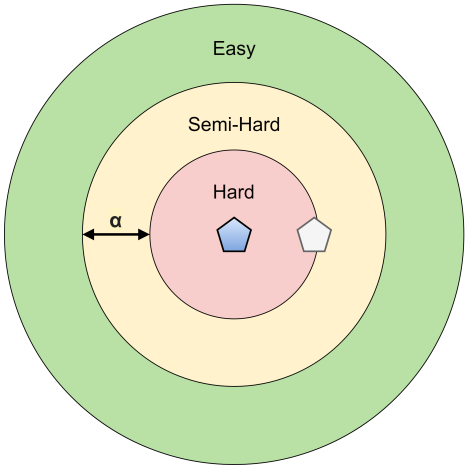
$$L_{triplet} = \max(0, \| W_\theta x_a - W_\theta x_p \|_2^2 - \| W_\theta x_a - W_\theta x_n \|_2^2 + \alpha)$$

  - x_a, x_p, x_n are called *anchor, positive and negative*
  - Same idea of the Contrastive Loss, but uses both a positive and a negative sample at the same time
  - We want the distance between anchors and positives to be smaller than the distance between anchors and negatives by at least a margin α



Triplet Mining

- Triplet Mining refers to the strategy used to select negative samples. We can have 3 types of negatives:
  - **Easy**: They already satisfy the loss constraint and therefore do not contribute during training
  - **Hard:** They are those negatives that have a shorter distance from the anchor than a positive record
  - **Semihard:** They are those negatives that have a greater distance than the positive but do not exceed the margin α
  - Using hard and semi-hard negatives usually speed-up training convergence; however, they can lead to *mode collapse* issues where the model reaches too fast a local minimum



Metric	Sample Selection	Topic	Dataset	Purpose	Year
Contrastive Loss	Hard negative	Image recognition, Object recognition	MNIST, NORB	calculates a contrastive loss function that aims to obtain a higher value for pairs of dissimilar objects and aims to obtain a lower value for pairs of similar objects	2006
Triplet Loss	Easy sampling	Image recognition, Object recognition	MNIST, CIFAR10, SVHN, STL10	calculates the distance difference between anchor-positive samples and anchor-negative samples and aims to bring similar objects closer	2014
Structured Loss	Hard negative	Image retrieval	CUB-200-2011, Online Products, CAR-196	aims a new metric learning algorithm using the lifted dense pairwise distance matrix within the batch throughout the training.	2016
N-Pair Loss	Multiple negative "class"	Image retrieval, Image clustering, Face verification Face identification, Object recognition Object verification	CUB-200-2011, Online Products, Flower-610, CAR-196, LFW, Car-333	aims to develop triplet loss focusing on pushing a positive sample away from multiple negative samples at each training stage	2016
Magnet Loss	Hard negative	Image recognition, Image annotation	Stanford Dogs, Oxford-IIIT Pet Oxford 102 Flowers, Object Attributes	aims to retrieve a whole local neighborhood of nearest clusters and punish their overlaps	2016
Angular Loss	Multiple negative	Image retrieval, Image clustering	CUB-200-2011, Online Products, CAR-196	focuses on limiting the angle in the negative sample of triplet triangles.	2017
Quadruple Loss	Semi-hard negative	Patient similarity	The Ischemic Heart, The Cerebrovascular	aims to capture the degree of similarity between patients effectively	2017
Hierarchical Triplet Loss	Anchor-Neighbor sampling	Image retrieval Face recognition	CUB-200-2011 ,Online Products, CAR-196 LFW, In-Shop Clothes Retrieval	aims to collect informative samples and capture global data context with an online class-level tree update	2018
Part Loss	Easy sampling	Person re-ID	Market-1501, CUHK03, VIPeR	aims to reduce empirical classification risks for training and representation learning risks for test by dividing images to K parts	2019
Multi-Similarity Loss	General pair weighting	Image retrieval	CUB-200-2011, Online Products, CAR-196 In-Shop Clothes Retrieval	aims to collect informative paired samples, and weights these pairs both their own and relative similarities	2019

Kaya, Mahmut, and Hasan Şakir Bilge. "Deep metric learning: A survey." *Symmetry* 11.9 (2019): 1066.

Metric Learning: Applications

- **Computer Vision:** Appropriate techniques are required in computer vision to compare images or videos with each other. Working with their latent representations is often more effective
  - Image Classification
  - Object Recognition
  - Face Recognition
  - Image Annotation

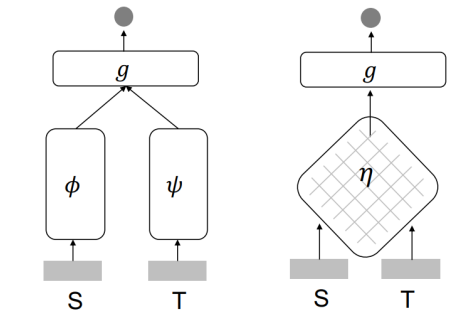
- **Information Retrieval**: The goal of information retrieval systems is to provide the user with the most relevant documents based on their queries. This ranking can be obtained by using the distance between the latent representations of queries and documents
  - Document Retrieval
  - Cross-Modal Retrieval
  - Question Answering

Information Retrieval and Neural Ranking Models

- **Information retrieval (IR)**: Return the most relevant results to a given query $s \in \mathcal{S}$ from a set of documents $\mathcal{T} = \{t_1, t_2, \dots, t_n\}$
  - A function $f(s,t)$ which assigns a similarity score between a query and a document must be used. In general, we have $f(s,t) = g(\psi(s), \phi(t), \eta(s,t))$
  - ψ , ϕ extract features from queries and documents respectively
  - η models the relationship between queries and documents
  - g is the evaluation function that returns the similarity score
- In traditional IR, ψ , ϕ and η are manually defined functions
- Recently, the use of deep neural networks is becoming increasingly pervasive. These models applied to IR are also referred to as **Neural Ranking Models** *[Guo et al., 2020]*
  - All or part of the functions ψ , ϕ, η, g are defined through deep neural networks

Representation vs Interaction Architectures

- **Representation-Focused (a):** Explicitly define ψ and ϕ and compute the similarity using $g\big(\psi(s), \phi(t)\big)$. **They can be trained using *metric learning***
  - *Advantages:* they are efficient as they allow to pre-compute the document embeddings
  - *Limits: t*hey are usually less accurate since the interaction between query and documents is not explicitly modeled
- **Interaction-Focused (b):** They don't directly define ψ and ϕ and focus on the interaction between queries and documents through the function η
  - *Advantages*: they allow to learn deep relationships between queries and documents and often achieve higher accuracies
  - *Limits*: they are inefficient for online retrieval



(a) Representation-focused (b) Interaction-focused

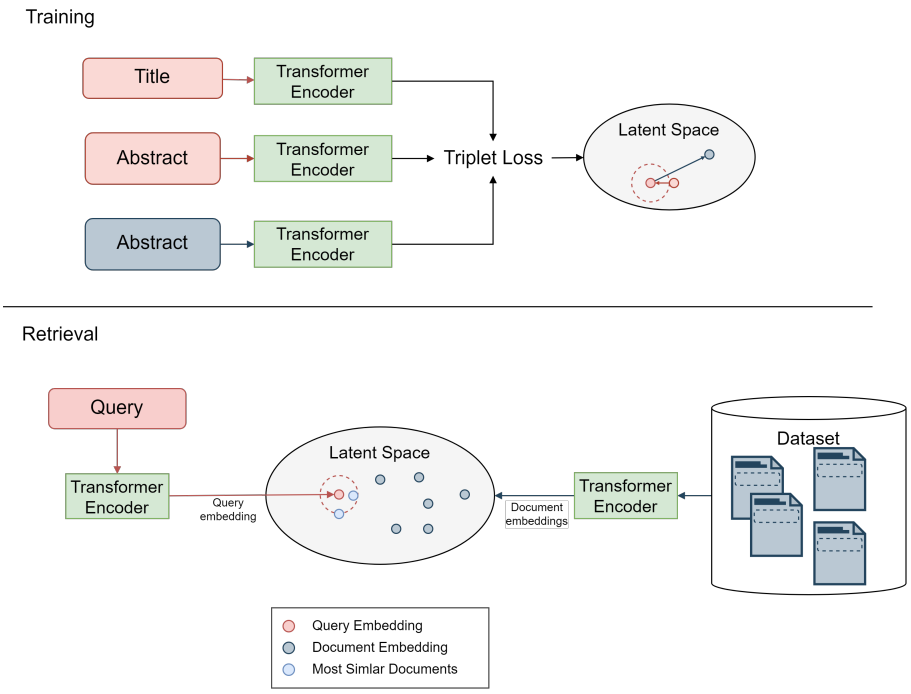
Guo, Jiafeng, et al. "A deep look into neural ranking models for information retrieval." Information Processing & Management 57.6 (2020): 102067.

Text-to-Text Information Retrieval with Metric Learning

- Suppose that we want to implement a text-to-text information retrieval system for scientific articles
  - We want to provide a textual query and retrieve its most semantically relevant articles

- **This problem can be solved with metric learning**
  - We can use a Transformer model as text encoder to generate latent vector representations (*representation-focused solution*)
  - Title and Abstract from a given article are considered a positive pair
  - The same Title and a random Abstract are considered a negative pair
  - Then we can finetune the Transformer model with Triplet Loss to increase the distance between negatives and reduce the distance between positives

- To perform retrieval, we simply **rank each document by its distance to the query**
  - The top-K results are the most semantically relevant according to our model



Cross-Modal Information Retrieval with Metric Learning

- In cross-modal retrieval, queries and documents are of different modalities (e.g., text-to-image, image-to-text, text-to-audio…)
  - Much studied in literature is **Text-to-Image** and **Image-to-Text** retrieval, in which queries and documents can be text or images

- Both Representation-Focused and Interaction-Focused solutions have been developed for cross-modal retrieval
  - **Representation-Focused:** They were the first to be proposed and typically use LSTMs as text encoders and pretrained-CNNs as image encoders, both trained using metric learning (VSE, VSE++, SCAN, PFAN, …)
  - **Interaction-Focused:** They are recently becoming more popular and are based on cross-modal Transformers (ImageBERT, OSCAR, ViLT, FashionBERT, …)

Cross-Modal Retrieval: SOTA Solutions and Limits

- SOTA solutions in the literature are those based on ***Cross-Modal Transformers Interaction-Focused*** which train the model to associate a similarity score $\mathrm{g}(\eta(s,t))$ to each query-document pair
  - There are no explicit latent representation ψ and ϕ
  - They don't use metric learning

- They are effective thanks to the direct interaction between queries and documents and also thanks to the cross-modal pre-training

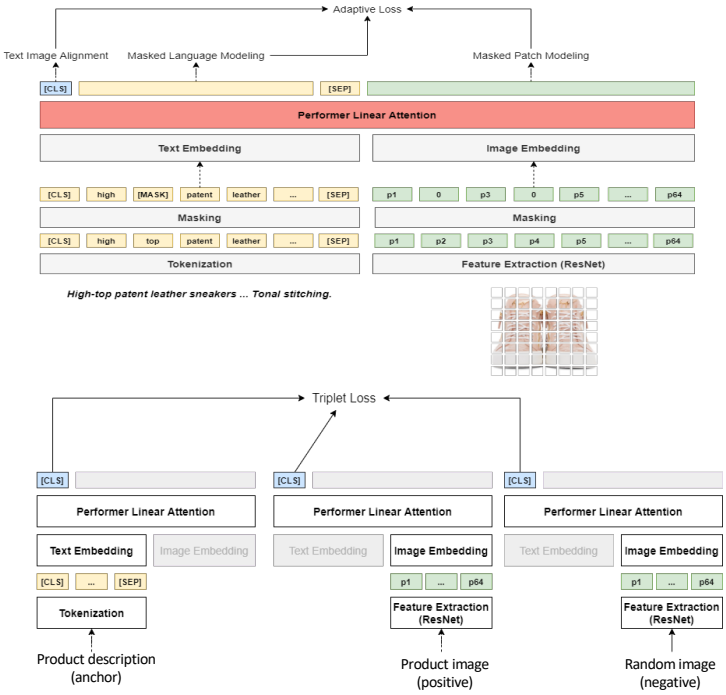
- However, they are usually inefficient for two reasons
  - **Computational Complexity:** The quadratic complexity of the attention mechanism becomes an important limitation when working with long texts and big images
  - **Slow Online Search:** Due to the coupling of the input, it is not possible to use multidimensional indexes to optimize the search.

Our Solution: Performer and Metric Learning

- To reduce the computational complexity, an efficient Transformer can be used to approximate the attention mechanism
  - This allows to work with longer text and bigger images or increase the batch size to improve training
  - We indeed developed a **first Efficient V+L Transformer based on Performer**
- To improve retrieval efficiency, text and images must be decoupled ad their embeddings should be separately computed
  - **We used Metric Learning to accomplish this and to generate a latent space that preserves the semantic similarity between the two modalities**
- Two stage training:
  - *First phase*: cross-modal pre-training with coupled text and images to exploit the ability of Transformers' attention to learn deep semantic relationships between the two modalities
  - *Second phase*: metric learning to learn the embeddings of the two modalities and generate the common latent space ($\mathbb{R}^{768}$ in out case)

Moro, G., & Salvatori, S. (2022, September). Deep Vision-Language Model for Efficient Multi-modal Similarity Search in Fashion Retrieval. In *Similarity Search and Applications: 15th International Conference, SISAP 2022, Bologna, Italy, October 5–7, 2022, Proceedings* (pp. 40-53). Cham: Springer International Publishing.

- **First Phase**: Pre-training with text-image interaction
  - *Masked Language Modeling:* Predict masked tokens in the sentence
  - *Masked Path Modeling:* Predict masked patches in the image
  - *Text Image Alignment:* Predict whether the input text describes the image
- **Second Phase***:* Training with Triplet Loss, decoupling text and images
  - *Anchor* and *positive* are an image and its caption; the *negative* is a random image in the dataset
  - Triplet mining can be performed offline or online. In the first case, one can choose negatives which are more difficult to distinguish



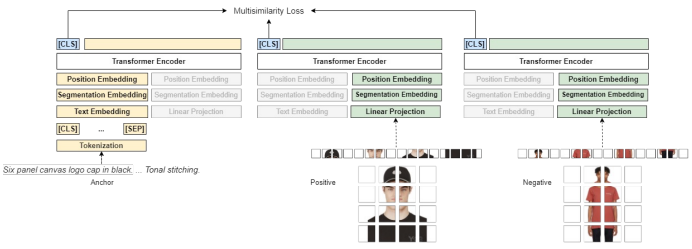
Results on the FashionGen Dataset

- The table below shows the results obtained on the FashionGen dataset in the image-to-text (ITR) and text-to-image (TIR) retrieval tasks
  - The table shows also the results of other models proposed in the literature on the same dataset
- Thanks to cross-modal Transformers and the combination of pre-training and metric learning, **Rank@K** is higher than in previous SOTA models
  - The model is also memory and time efficient since it is based on a linear attention mechanism
- The results prove that the latent space learned with metric learning correctly encodes the relationships between text and images

Tasks		VSE	VSE++	SCAN	PFAN	ViLBERT	VLBERT	Image Bert	OSCAR	Fashion Bert	Kaleido Bert	Our
ITR	R@1	4.01%	4.59%	4.59%	4.29%	20.97%	19.26%	22.76%	23.39%	23.96%	27.99%	**34.70%**
	R@5	11.03%	14.99%	16.50%	14.90%	40.49%	39.90%	41.89%	44.67%	46.31%	60.09%	**70.50%**
	R@10	22.14%	24.10%	26.60%	24.20%	48.21%	46.05%	50.77%	52.55%	52.12%	68.37%	**83.70%**
TIR	R@1	4.35%	4.60%	4.30%	6.20%	21.12%	22.63%	24.78%	25.10%	26.75%	33.88%	**35.80%**
	R@5	12.76%	16.89%	13.00%	20.79%	37.23%	36.48%	45.20%	49.14%	46.48%	60.60%	**70.20%**
	R@10	20.91%	28.99%	22.30%	31.52%	50.11%	48.52%	55.90%	56.68%	55.74%	68.59%	**83.30%**

Improving Efficacy and Removing the Visual Embedder

- Same idea of the two-stage training but
  - We keep the quadratic attention layer
  - We reduce model footprint by **removing ResNet** preprocessing
- We replace the Triplet Loss with a more recent one: **Multsimilarity Loss**
  - Encodes multiple similarities between anchors, positives, and negatives to weight and sample triplets during training



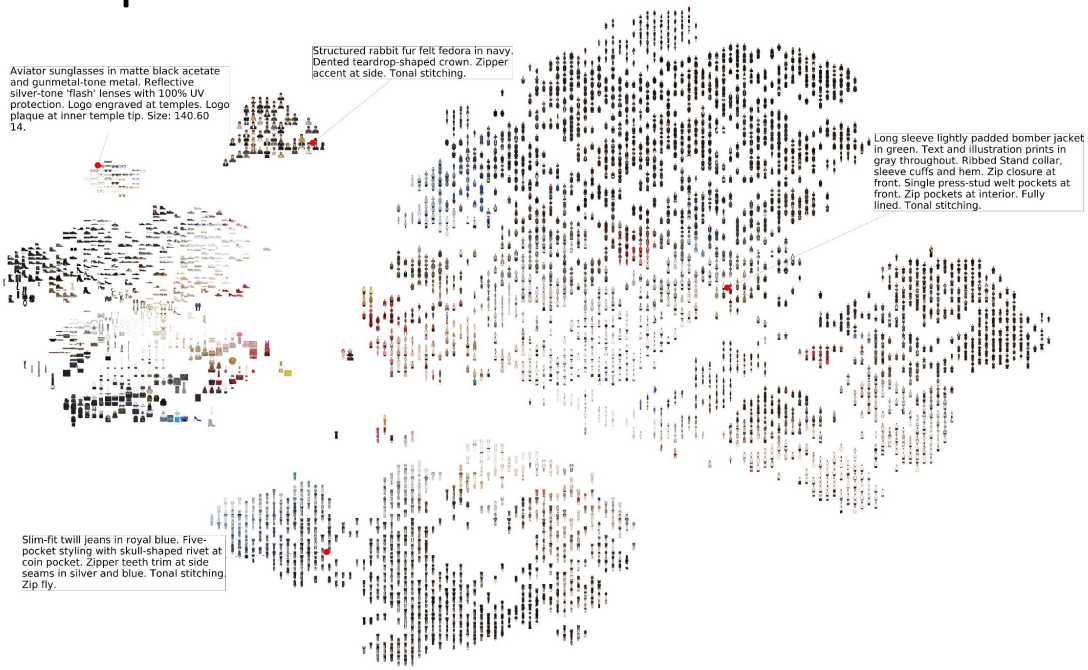
Results on the FashionGen Dataset

- Thanks to the lightweight visual embedder we can split images in smaller patches to focus on fine-grained details
  - From 32x32 patches to 16x16 patches
- The gap with SOTA solutions is even larger than the previous approach
- We keep the same retrieval efficiency as before since we still have separate text and image embeddings

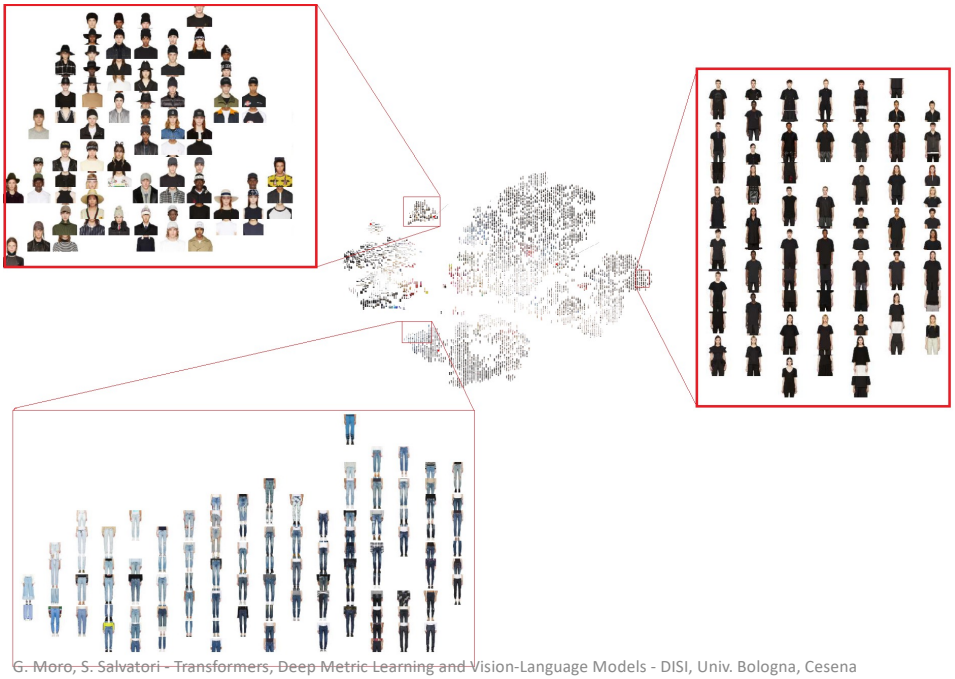
Tasks		FashionBERT	KaleidoBERT	Our (Performer + ResNet + Triplet)	Our (Patch Size=16 + Multisimilarity)	FashionVil
ITR	R@1	23.96%	27.99%	34.70%	**46.70%**	65.54%
	R@5	46.31%	60.09%	70.50%	**80.00%**	91.34%
	R@10	52.12%	68.37%	83.70%	**89.30%**	96.30%
TIR	R@1	26.75%	33.88%	35.80%	**43.10%**	61.88%
	R@5	46.48%	60.60%	70.20%	**76.60%**	87.32%
	R@10	55.74%	68.59%	83.30%	**87.60%**	93.22%
SumR		251.53	319.52	378.20	**423.30**	495.60

Current SOTA but with >6x training data and +3 pretraining tasks. Still uses ResNet

Latent Space Visualization with T-sne

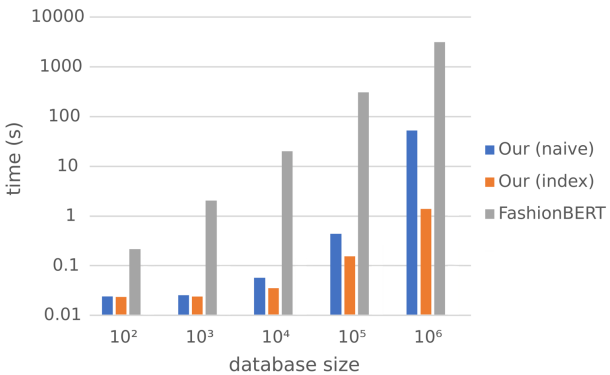


Latent Space Visualization with T-sne



Multidimensional Indices

- One of the advantages of having a latent embedding space in which queries and documents are decoupled, is that it is also possible to define multidimensional indices on document embeddings to **drastically reduce retrieval times**
- The figure shows the time required to perform retrieval on different database sizes
  - **Blue**: latent space with no indices
  - **Orange**: latent space with ball-tree index
  - **Grey**: FashionBERT model which does not generate a latent space and must compute a score for each pair at runtime



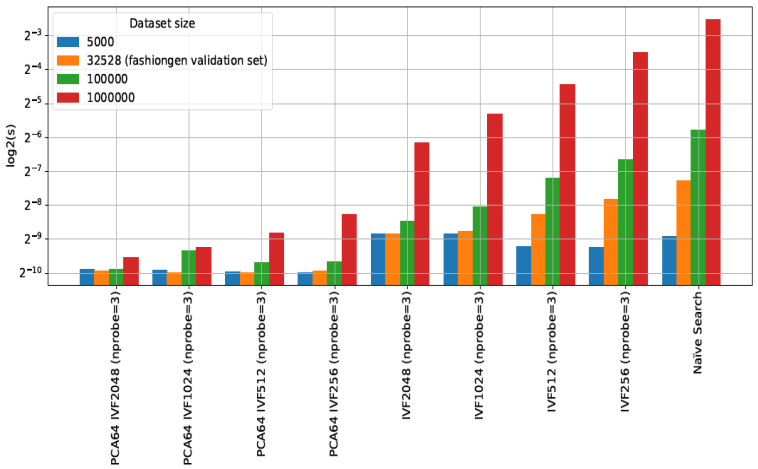
- The solution with embedding and multidimensional indices is **up to 1000 times faster** than FashionBERT

Multidimensional Indices: Approximate Search

- Performance can be further improved by employing approximate search techniques (known as *Approximate Nearest Neighbor Search* or ANN)
  - They are faster but usually lead to less acurate results
  - Available in python through the **Faiss** library *[Johnson et al., 2020]* developed by facebook
- **Inverted File Index (IVF):** Subdivide the space into V Voronoi cells. The query is compared only with the embeddings of the documents in the same cell
  - It is possible to increase accuracy by including the embeddings of the n neighboring cells called $nprobes$
- **Principal Component Analysis (PCA):** Reduce the dimensionality of the embeddings to speed up the computation of distances
- And others
  - Product Quantization
  - Hierchical Clustering
  - Locality Sensitive Hashing
  - …

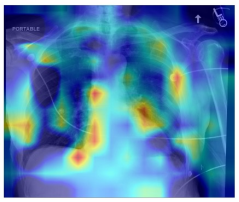
Multidimensional Indices: Approximate Search

- The figure shows the retrieval times required by different ANN techniques as the dataset size varies
  - ***IVF<X>*** indicates that an Inverted File Index with X cells was used
  - ***PCA64*** indicates that embeddings were first reduced to $\mathbb{R}^{64}$
  - ***Naive Search*** indicates the solutions with no ANN techniques used
- As shown in the figure, performance can be further improved combining together multiple strategies
  - However in that case accuracies are even lower



Application to Radiography Retrieval

- We also conducted studies on the Biomedical domain, in particular in the radiography retrieval task
  - Matching **between medical reports and radiological images**
- This domain is still challenging since:
  - Medical reports are hard to interpret
  - Radiography images are similar to each other and differ by details
- We still get promising results (38% Rank@1) and this domain allows us also to extend our solution with explainability algorithms

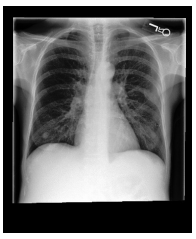


[CLS] final report indication : _ _ _ m with am ##s / / acute process ? technique : portable ap upright view of the chest comparison : none findings : heart is normal size and card ##iom ##ed ##ias ##tina ##l con ##tour ##s are within normal limits . lungs are symmetrical ##ly expanded and clear . there is no pl ##eur ##al e ##ff ##usion or p ##nc ##um ##otho ##ra ##x . no pulmonary ed ##ema . impression : no acute intra ##thor ##ac ##ic process . [SEP]

Thesis Proposals

- **Explainability of Vision-Language Models**
  - Understand what deep v+l models have learned
  - Attention Rollout, Layer Integrated Gradients, ...

- **Text Generation from Images**
  - Automatically generation of medical diagnose reports from radiological images



Report:
The lungs are clear of focal consolidation, pleural effusion or pneumothorax. The heart size is normal…

Notebook link for the lab in python:
https://colab.research.google.com/drive/1g_Bo4EDWv97n3_MR7Xnhg9sXbBXqnOIP?usp=sharing

References

- Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." ICLR (2014).

- Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. "Effective approaches to attention-based neural machine translation." EMNLP (2015).

- Vaswani, Ashish, et al. "Attention is all you need." NIPS (2017).

- Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." NAACL-HLT (2018).

- Wu, Yonghui, et al. "Google's neural machine translation system: Bridging the gap between human and machine translation." arXiv preprint arXiv:1609.08144 (2016).

- Wang, Alex, et al. "GLUE: A multi-task benchmark and analysis platform for natural language understanding." EMNLP (2019).

- Rajpurkar, Pranav, et al. "Squad: 100,000+ questions for machine comprehension of text." EMNLP (2016).

- Sang, Erik F., and Fien De Meulder. "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition." arXiv preprint cs/0306050 (2003).

- Moro, G., Salvatori, S. "Deep Vision-Language Model for Efficient Multi-modal Similarity Search in Fashion Retrieval", SISAP (2022)

References

- Zellers, Rowan, et al. "Swag: A large-scale adversarial dataset for grounded commonsense inference." EMNLP (2018).
- Liu, Yinhan, et al. "Roberta: A robustly optimized bert pretraining approach." arXiv preprint arXiv:1907.11692 (2019).
- Lewis, Mike, et al. "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension." ACL (2020).
- Tay, Yi, et al. "Efficient transformers: A survey." arXiv preprint arXiv:2009.06732 (2020).
- Choromanski, Krzysztof, et al. "Rethinking attention with performers." ICLR (2021).
- Zaheer, Manzil, et al. "Big Bird: Transformers for Longer Sequences." NeurIPS. (2020).
- Guo, Jiafeng, et al. "A deep look into neural ranking models for information retrieval." Information Processing & Management 57.6 (2020): 102067.
- Kaya, Mahmut, and Hasan Şakir Bilge. "Deep metric learning: A survey." Symmetry 11.9 (2019): 1066.
- Johnson, Jeff, Matthijs Douze, and Hervé Jégou. "Billion-scale similarity search with gpus." IEEE Transactions on Big Data (2019).
- Kitaev, Nikita, Łukasz Kaiser, and Anselm Levskaya. "Reformer: The efficient transformer" ICLR (2020).