

Machine Learning Course - CS-433

Self-supervised Learning

Dec 12, 2023

Martin Jaggi
Last updated on: December 12, 2023

credits to Atli Kosson and Oğuz Kaan Yüksel



Self-supervised Learning

Problem: Labelled data for supervised learning is often limited and expensive to acquire.

Transfer learning: Fine-tune an existing model trained on a related task giving it a good internal representation of the input data. This can allow us to achieve high performance on the **downstream task** with significantly fewer labels. However, we still need supervision (labels) for the **base task**.

Self-Supervised Learning: Use the input itself to generate supervisory signals. The base task is an artificial **pretext task** that requires learning useful features and structure in the input data.

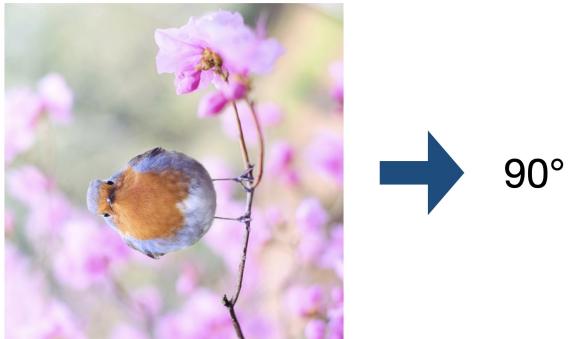
In the pretext task we want to learn a model f :

$$f : \mathbf{x}_{\text{in}} \mapsto \mathbf{x}_{\text{out}}$$

Where we use a transformation $g : \mathbf{x} \mapsto (\mathbf{x}_{\text{in}}, \mathbf{x}_{\text{out}})$ to create an input and a target from an (unlabelled) datapoint $\mathbf{x} \in \mathcal{X}$. Note that g often involves some randomness.

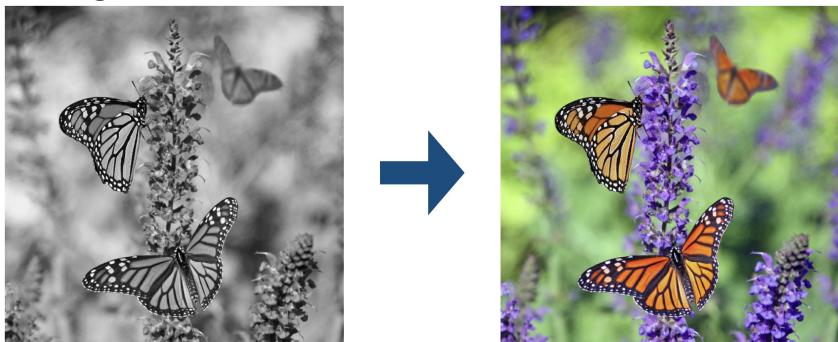
Example Pretext Tasks

Predicting Image Rotation:



90°

Image Colorization:



Predicting Relative Patch Placement:



Second patch
is above and
left of the first

Upcoming:

- Masked Language Modelling
- Next Token Prediction

Masked Language Modelling (MLM)

Learn to predict masked (hidden) words for example:

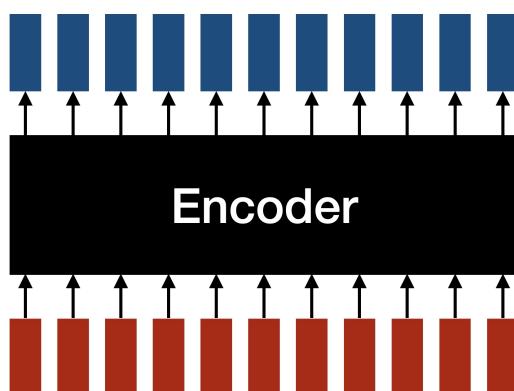
She [MASK] her coffee \mapsto She drank her coffee

Forces the model to learn a comprehensive representation of the language that is very useful for many downstream tasks. Can train on huge corpora of text (books, Wikipedia, etc) without any labelling.

BERT

Bidirectional Encoder Representations from Transformers is a well known family of [language models](#) based on the encoder-only transformer architecture and trained on masked language modelling.

[Encoder](#): Sees the whole sequence at once, every token can generally attend to every other token (both previous and later ones, hence bidirectional). Generates a fixed sized output, typically one token per input.



Training Inputs: BERT is trained on input sequences consisting of two sentences and special tokens, for example:

[CLS] The cat is sleeping.
 [SEP] It's on the sofa. [SEP]

Where [CLS] is a special classification token and [SEP] is used to separate the two sentences. Around 15% of the standard tokens are selected to be replaced by [MASK]. Occasionally the selected tokens are replaced by a random word or not replaced to reduce distribution shift for downstream tasks that do not have [MASK]. Positional and segment encodings are added to the token embeddings.

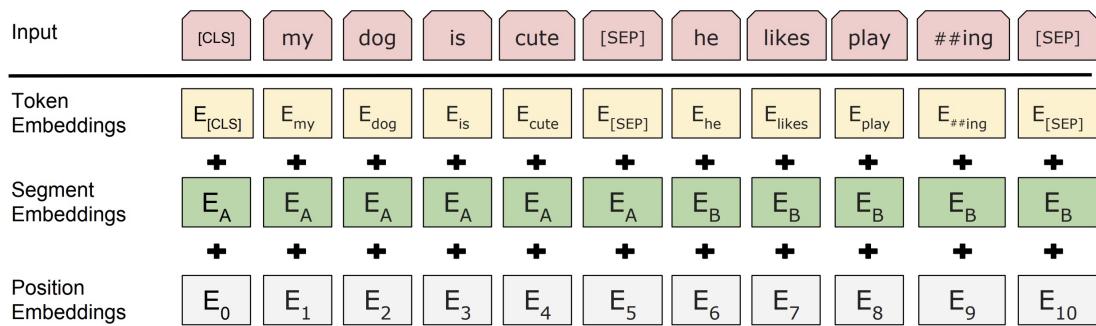
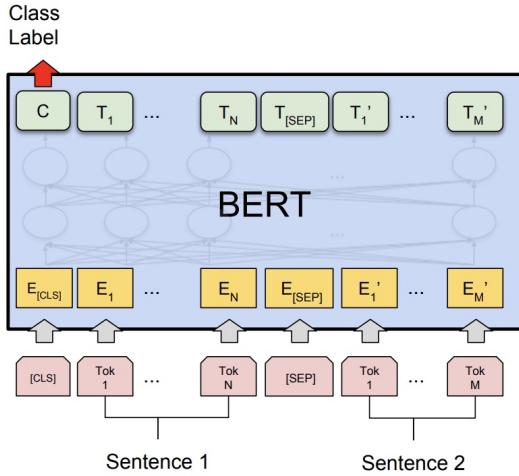


Figure from the original BERT paper: <https://arxiv.org/abs/1810.04805>

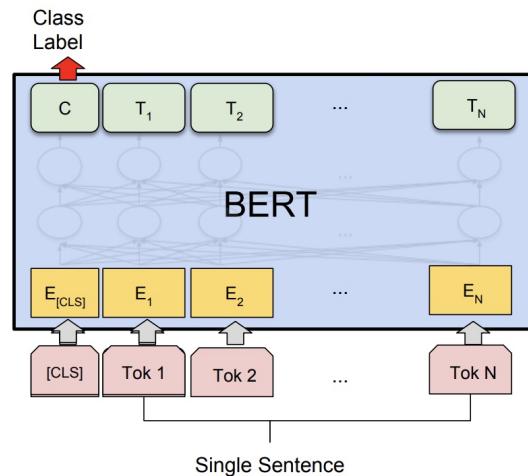
Training Objective:

- For each selected (replaced / masked) token: Predict the original token. Softmax cross-entropy loss over all possible tokens.
- For the [CLS] token: Predict whether the second sentence immediately follows the first sentence or not (binary classification).

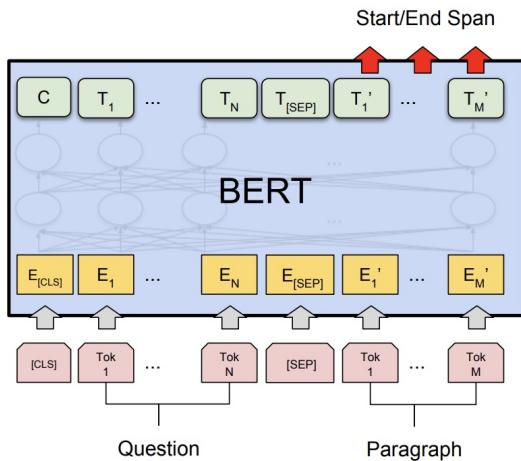
Downstream tasks: The structure of the BERT training task allows fine-tuning for a variety of downstream tasks.



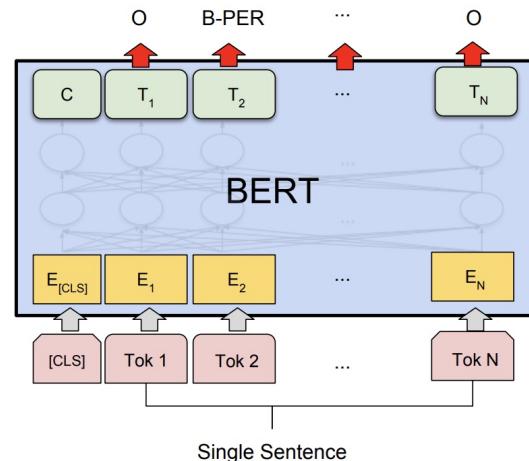
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



(b) Single Sentence Classification Tasks:
SST-2, CoLA



(c) Question Answering Tasks:
SQuAD v1.1



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

Figure from the original BERT paper: <https://arxiv.org/abs/1810.04805>

- Classifying sentences or pairs (e.g. sentiment analysis): Use the **[CLS]** token.
- Word level classification (e.g. named entity recognition): Use the output for each token.
- Extracting a relevant passage (e.g. search): Use the individual outputs to define a span.

Next Token Prediction

Predict the next token given previous tokens, e.g.:

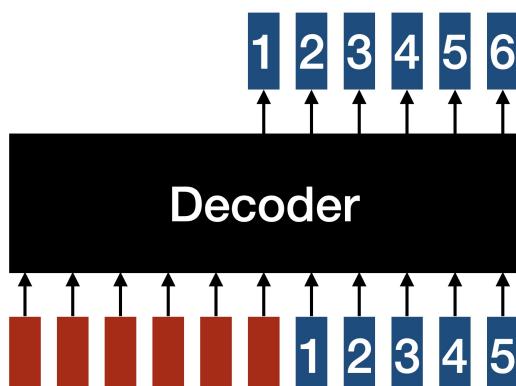
She drank her \mapsto coffee

Similar to Masked Language Modelling but is [causal](#), we can only see previous words, not later ones. This makes it better suited for generating arbitrary length responses.

GPT

Generative **P**re-trained **T**ransformers are a family of language models based on the decoder-only transformer architecture and trained on next token prediction. Most generative [Large Language Models \(LLMs\)](#) like GPT-4 and LLAMA are members of this family. They can be fine tuned for a variety of purposes, famously as chatbots / assistants (ChatGPT, Bard).

[Decoder](#): Each token can only attend to prior tokens (operates causally). Used [autoregressively](#) during inference, each generated output token is fed back into the model as an input to generate the following output token.



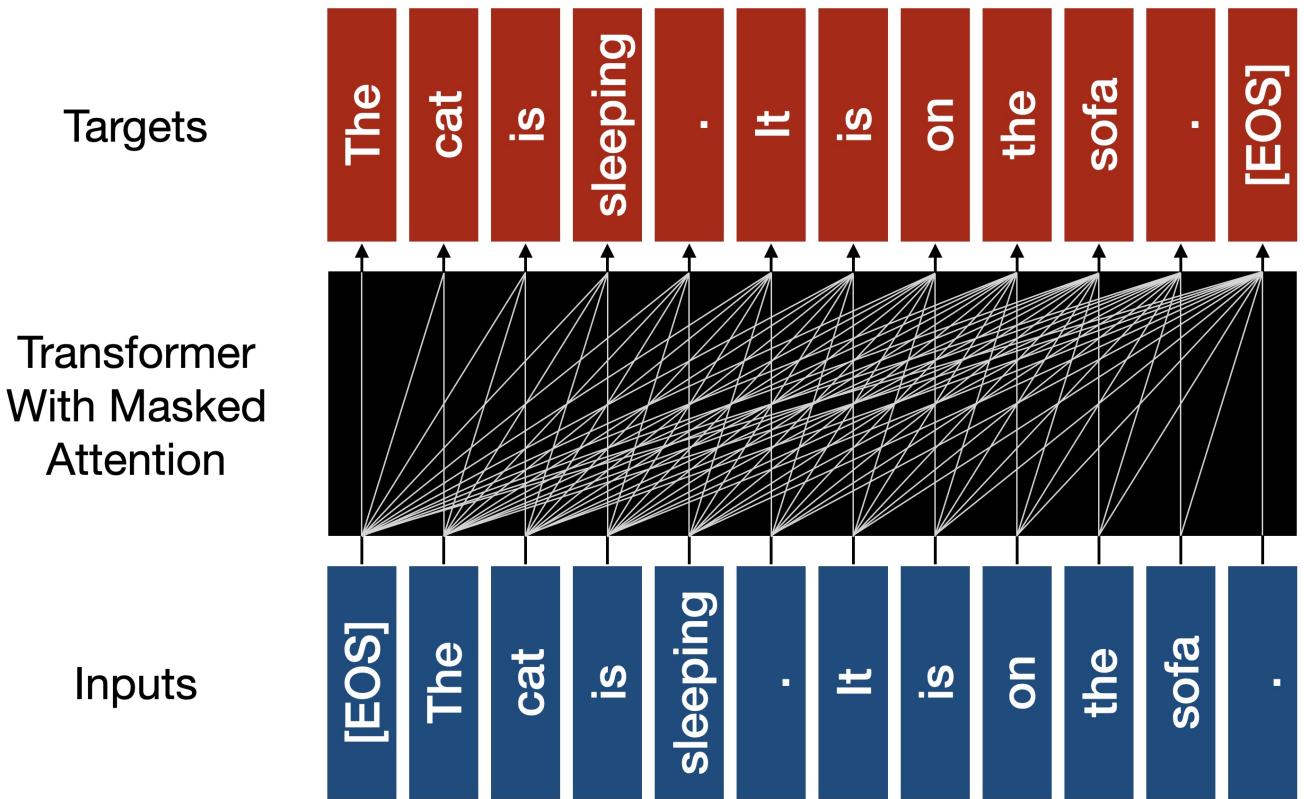
Training Inputs: The training is simply done on token sequences with minimal changes e.g.:

[EOS] The cat is sleeping. It's on the sofa. [EOS]

Here the only special token used is [EOS] which signifies the end of text and is used to separate unrelated chunks of text (but this can vary between implementations).

Training Procedure: Standard training involves [teacher forcing](#), where we don't use the model outputs as subsequent inputs (like in inference) but rather use the actual correct sequence. This allows training on all tokens in a sequence to happen in parallel rather than sequentially. We need to [mask](#) the attention to prevent tokens from attending to future tokens, preserving the causality needed for inference.

We obtain a loss for every token in the sequence corresponding to the prediction of the next token. This is a standard classification loss over the possible vocabulary (softmax cross-entropy). The total loss for a sequence is the mean loss over all tokens in the sequence.



Downstream Tasks: Decoders are very general sequence-to-sequence models and can be adapted to most natural language tasks. GPTs have been fine tuned for various applications. The base models are also often capable of performing various tasks without further tuning.

In-Context Learning: Large GPT models can sometimes complete new tasks with just a few examples of successful completion provided as inputs, without any model updates. The model uses patterns it has learned during its initial training to infer how to handle these new tasks. Example input:

Translate English to French:

sea otter => loutre de mer

cheese =>

which would hopefully result in **fromage** as an output.

The search for an effective input (prompt) to achieve desired output behavior is known as [prompt engineering](#).

Joint Embedding Methods

Learn an encoder invariant to certain transformations on the data, e.g., to rotation:



Rather than learning a model f : $\mathbf{x}_{\text{in}} \mapsto \mathbf{x}_{\text{out}}$, learn $f : \mathcal{X} \rightarrow \mathbb{R}^d$ s.t.

$$f(\mathbf{x}_1) \approx f(\mathbf{x}_2)$$

where $g : \mathbf{x} \mapsto (\mathbf{x}_1, \mathbf{x}_2)$ is a transformation that creates two [views](#) from the same input datapoint $\mathbf{x} \in \mathcal{X}$.

g is usually a random function that applies multiple data augmentations, each with certain probability.

Problem: constant f is a trivial encoder that is invariant!

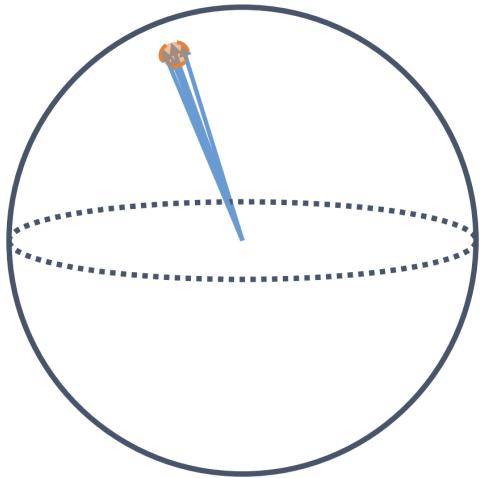


Figure from <https://arxiv.org/abs/2110.09348>

Solved by contrastive learning (next!) or non-contrastive methods with regularization, e.g.,

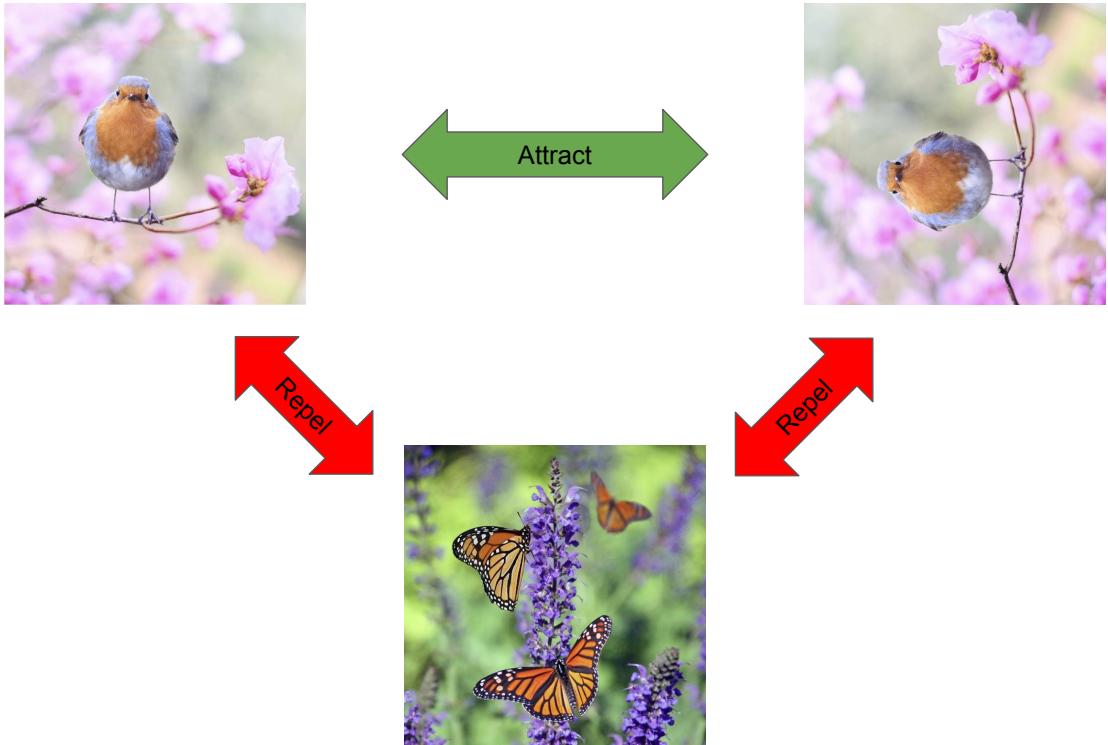
- **BYOL:** use two encoders f_1 , f_2 where f_2 is the exponential moving average of f_1 and force

$$f_1(\mathbf{x}_1) \approx f_2(\mathbf{x}_2).$$

- **VicReg:** force non-zero variances to avoid collapse but minimize the covariance to reduce information overlap.

Contrastive learning

Push away representations of unrelated views from different data points!



Objective: given a **positive** pair \mathbf{x}, \mathbf{x}^+ from the datapoint \mathbf{x} , and a **negative** view \mathbf{x}^- from a different datapoint $\mathbf{x}' \neq \mathbf{x}$,

$$s(f(\mathbf{x}), f(\mathbf{x}^+)) > s(f(\mathbf{x}), f(\mathbf{x}^-)),$$

where the s function quantifies the similarity of two embeddings.

SimCLR

A contrastive learning framework with optimized choices.

1. classification-like objective with N negative samples:

$$L(\mathbf{x}) = -\mathbb{E} \left[\log \frac{\exp(s(f(\mathbf{x}), f(\mathbf{x}^+)))}{\sum_{i=1}^N \exp(s(f(\mathbf{x}), f(\mathbf{x}_i^-)))} \right].$$

- maximize the score between \mathbf{x} and \mathbf{x}^+ ,
- minimize the score between \mathbf{x} and all \mathbf{x}_i^- .

2. use an additional [projector](#) to map the encoder output to the similarity (possibly lower-dimensional) space:

$$f(\mathbf{x}) = \underbrace{f_2}_{\text{projector}} \circ \underbrace{f_1}_{\text{encoder}}(\mathbf{x})$$

- learn f_1 and f_2 jointly in an end-to-end fashion,
- use only f_1 for downstream tasks.

3. use [cosine similarity](#) with temperature scaling τ :

$$s(\mathbf{e}_1, \mathbf{e}_2) = \frac{\langle \mathbf{e}_1, \mathbf{e}_2 \rangle}{\|\mathbf{e}_1\|_2 \|\mathbf{e}_2\|_2} / \tau.$$

4. generate views with data augmentation:

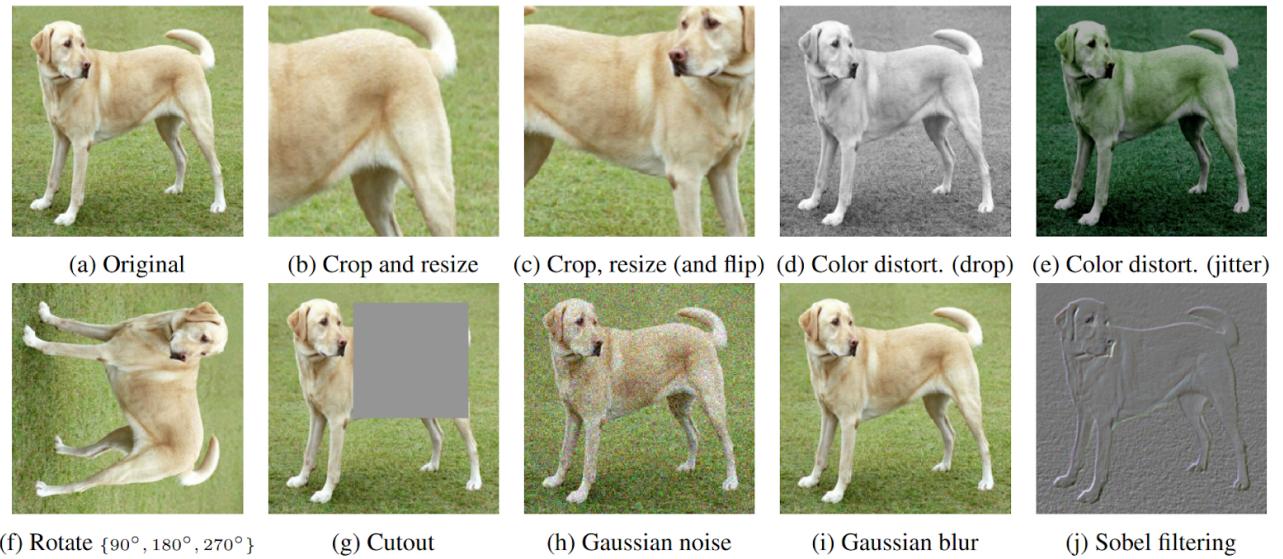


Figure from the SimCLR paper: <https://arxiv.org/abs/2002.05709>

- stronger data augmentations than those used in supervised learning,
- **random crop** creates global to local ($B \rightarrow A$) views or adjacent view ($D \rightarrow C$) prediction tasks.

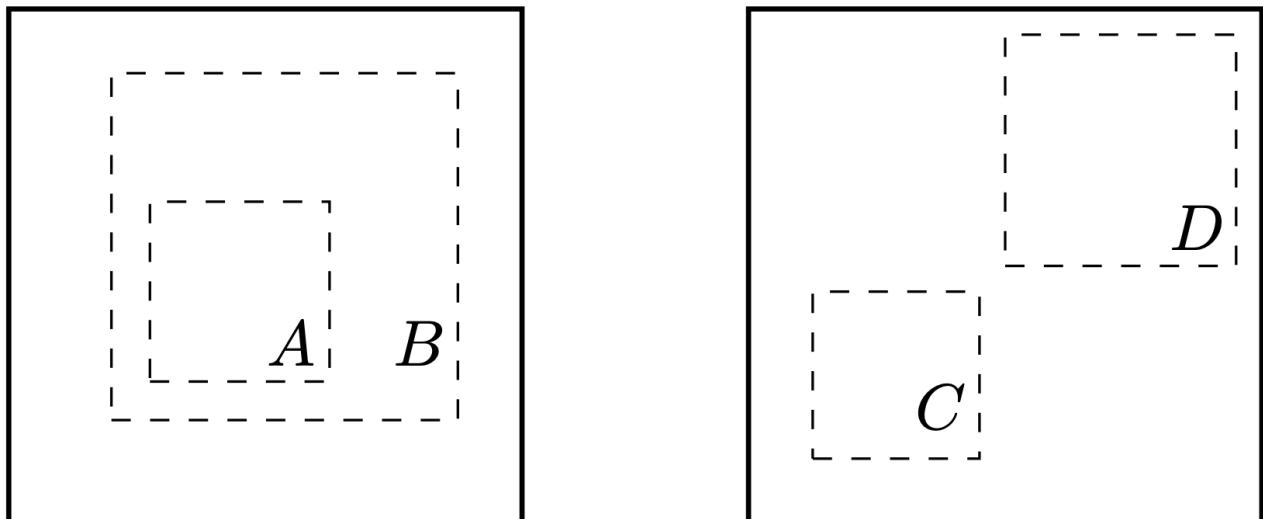


Figure from the SimCLR paper: <https://arxiv.org/abs/2002.05709>

CLIP

Use captioned images to learn a joint **multimodal** embedding space.

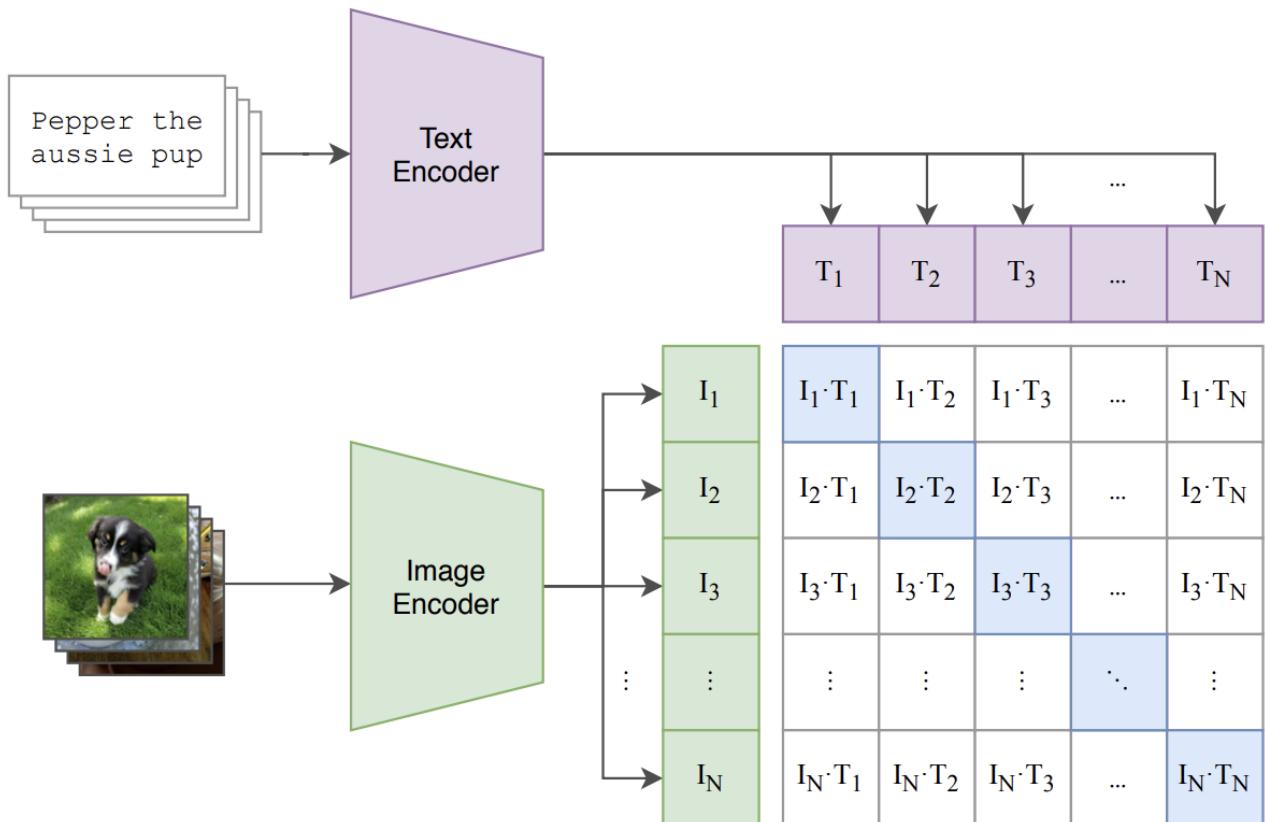


Figure from the CLIP paper: <https://arxiv.org/abs/2103.00020>

Objective: Maximize cosine similarity of caption and image embeddings while minimizing unrelated pairs.

Few-shot learning: learn to classify new classes with only a few labelled examples. CLIP is able to **zero-shot** classify with the following text input: “A photo of [class]”.

Further Pointers

1. The Illustrated BERT:

<https://jalammar.github.io/illustrated-bert/>

2. The Illustrated GPT:

<https://jalammar.github.io/illustrated-gpt2/>