

Machine Learning Course - CS-433

Self-supervised Learning

9 Dec 2025

Robert West

Last updated on: December 8, 2025

credits to Martin Jaggi, 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

Examples of self-supervised learning for text:

- Masked Language Modelling
- Next Token Prediction (as in previous LLM lecture)

... for images:

Predicting Image Rotation:

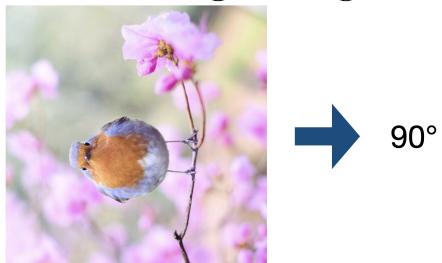


Image Colorization:



Predicting Relative Patch Placement:



Masked Language Modelling (MLM)

Learn to predict masked (hidden) words for example:

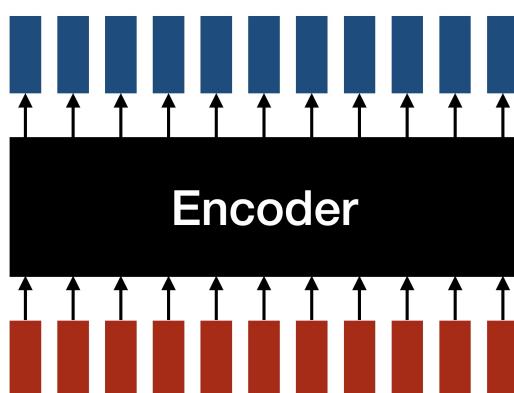
She [MASK] her coffee \mapsto She drank her coffee

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.

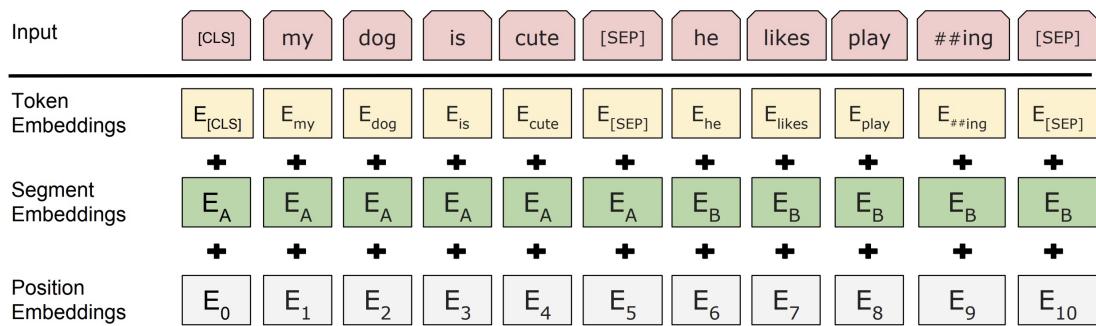
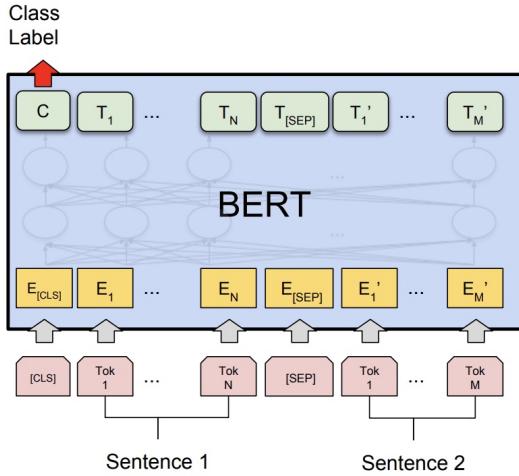


Figure from the original BERT paper: <https://arxiv.org/abs/1810.04805>

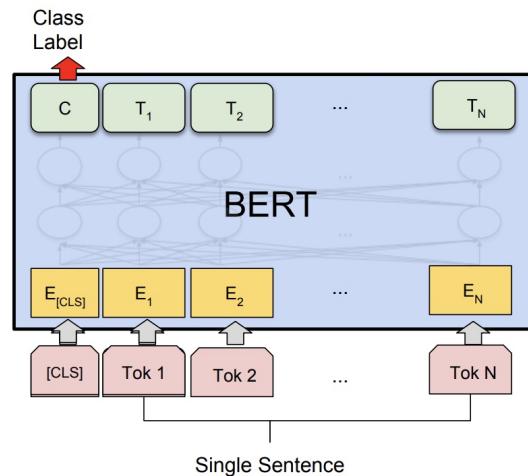
Training Objective:

- 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).

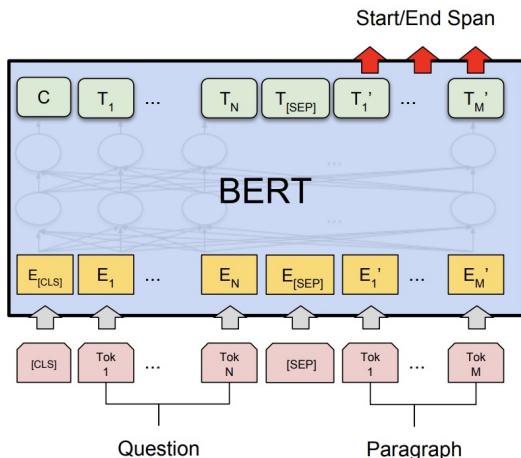
Downstream tasks: The structure of the BERT training task allows fine-tuning for a variety of downstream tasks.



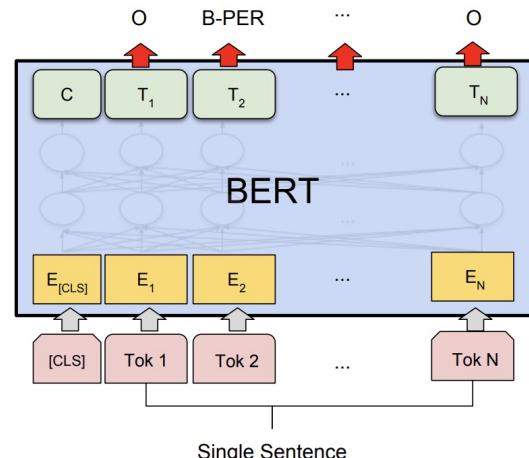
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



(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, LLMs

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

She drank her \mapsto coffee

Similar to Masked Language Modelling but is causal (auto-regressive), we can only see previous words, not later ones.

This makes it better suited for generating arbitrary length responses. In line with common use-cases of LLMs, such as prompting, in-context learning.

(See previous lecture)

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!

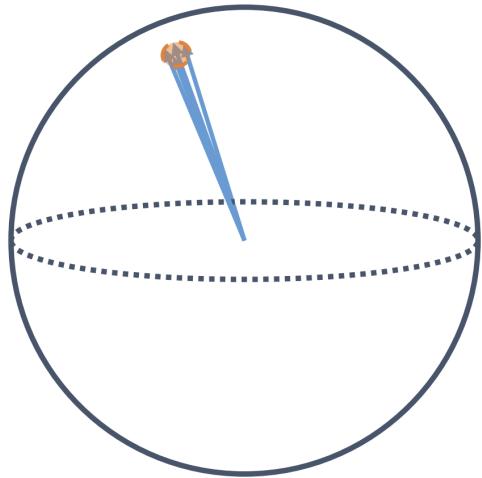


Figure from <https://arxiv.org/abs/2110.09348>

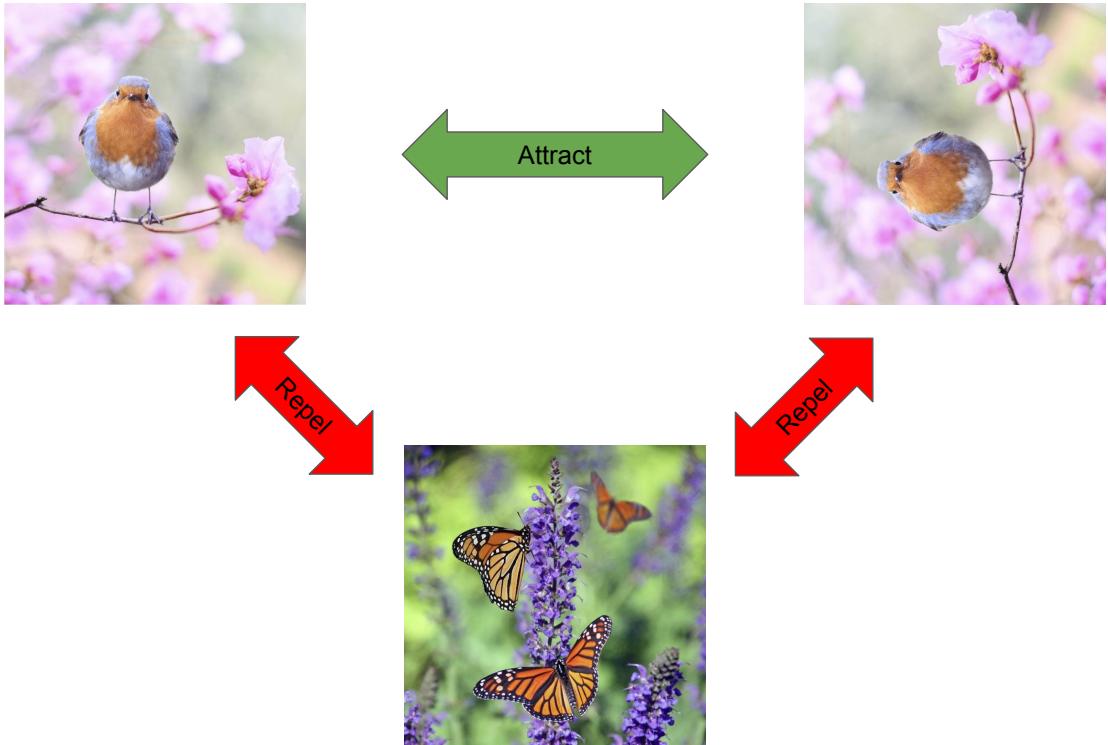
Solved by contrastive learning
(next!) or non-contrastive methods
with regularization, e.g.,

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

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

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 objective with N negative samples:

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

- 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:

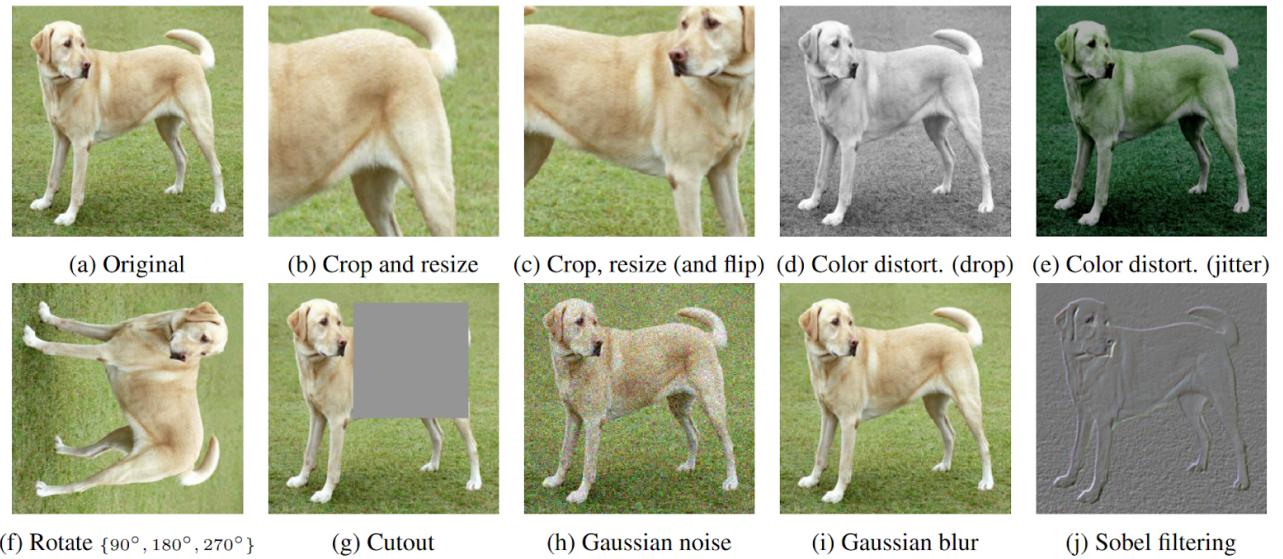


Figure from the SimCLR paper: <https://arxiv.org/abs/2002.05709>

- 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.

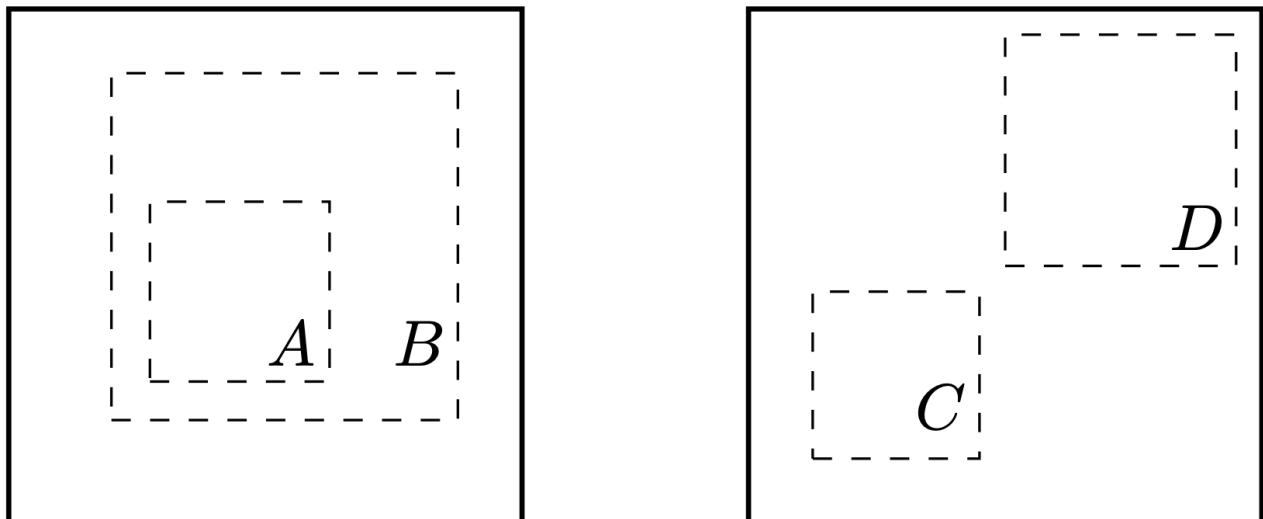


Figure from the SimCLR paper: <https://arxiv.org/abs/2002.05709>

CLIP

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

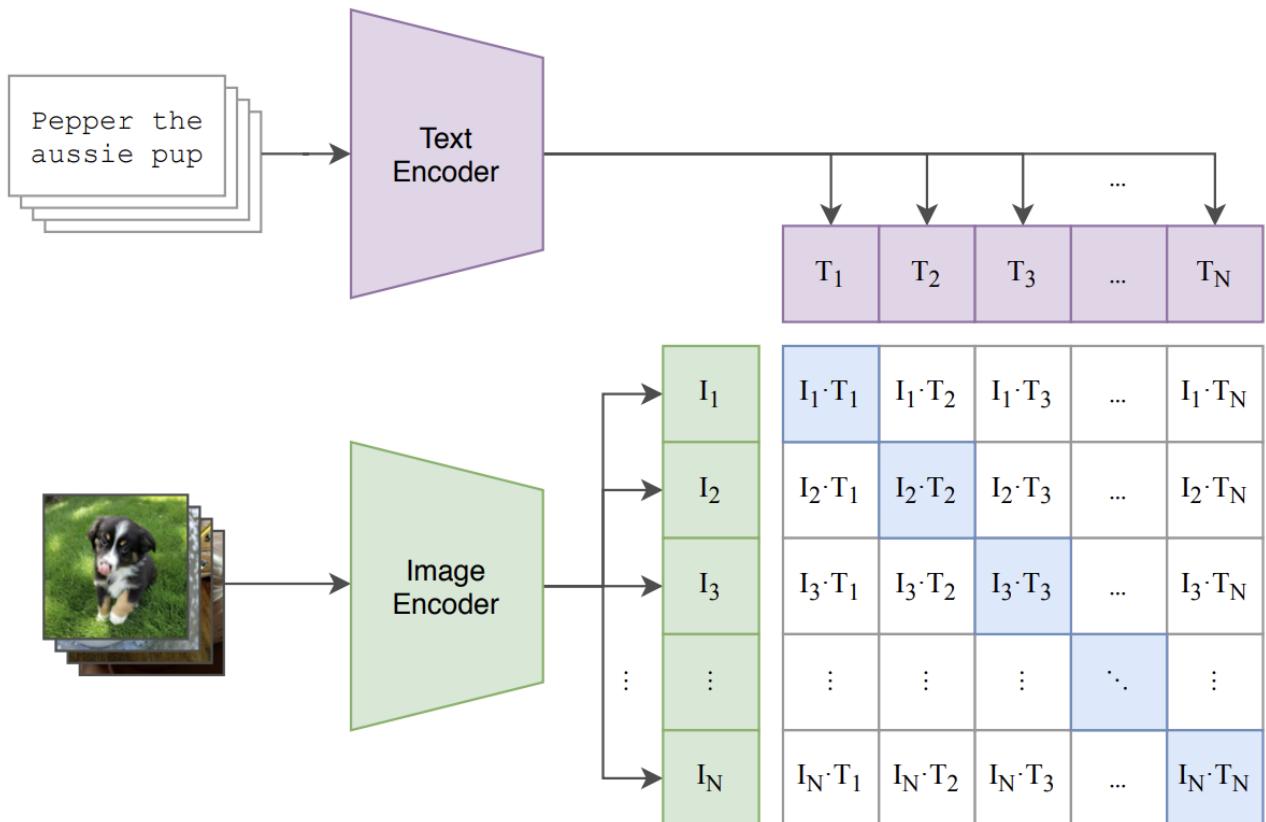


Figure from the CLIP paper: <https://arxiv.org/abs/2103.00020>

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]”.