**Ludwig-Maximilians-Universität München**
**Institut für Informatik**
Prof. Dr. Matthias Schubert
Maximilian Bernhard, Niklas Strauß

Munich, 09.12.2024

# Deep Learning and Artificial Intelligence

WS 2024/25

## Exercise 8: Self-supervised Learning

### Exercise 8-1 Contrastive Learning

(a) What are positive and negative samples in contrastive learning?

**Detailed Solution**

Given a sample, a positive sample with respect to this sample is a sample that has the same semantic content, e.g., an augmented version of the original sample. Conversely, negative samples have (probably) different semantics from the first sample. Thus, we want to maximize the similarity between embeddings for positive sample pairs and minimize the similarity between embeddings for negative sample pairs.

(b) How are negative samples chosen in SimCLR? What would happen if we did not use negative samples in SimCLR?

**Detailed Solution**

In SimCLR, all samples in a batch of $n$ samples act as negative samples for the $n-1$ other samples. This means that the set of negative samples can actually contain some samples that should be considered positive. However, there is no easy way to identify these ”wrong” negative sampels as we do not have access to data annotations. Instead, SimCLR mitigates the effect of this problem by choosing a large batch size $n$ such that the ”correct” negative samples will consistently dominate the training objective.

If we did not use any negative samples in SimCLR, the network could simply map all samples to the very same embedding (i.e., becoming a constant function) and achieve a perfect loss. However, in this case, the ”learned” embeddings are entirely useless.

(c) Discuss and compare the contrastive learning frameworks SimCLR, SwAV, BYOL, and SimSiam from the lecture. Thereby, pay specific attention to the training objectives, the usage of negative samples, the usage of (different) networks, and gradient flow.

**Detailed Solution**

- **SimCLR** uses a single network to encode samples. Positive pairs are formed with augmentation, negative pairs are formed by using all other samples in a batch. The training objective is a variant of the cross-entropy and maximizes/minimizes the similarity of embeddings for pairs of positive/negative samples, respectively.
- **SwAV** uses a similar overall framework as SimCLR but does not measure the similarity between sample embeddings directly. Instead, it clusters sample representations to obtain cluster prototypes, which are then used as attractors/repellors for the training samples.

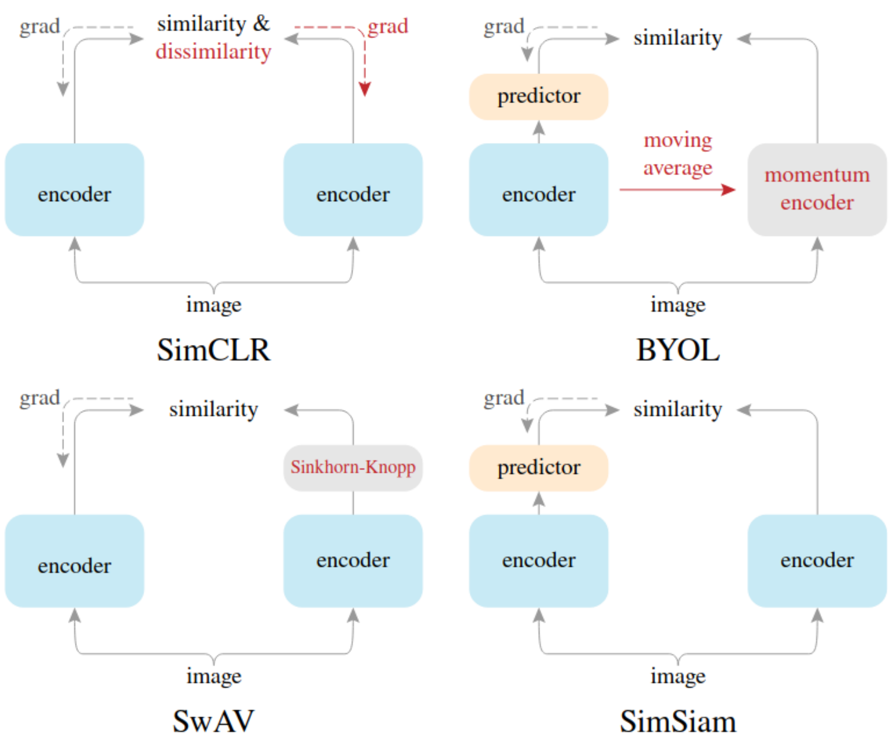

Figure 1: Contrastive learning frameworks. Source: `https://arxiv.org/pdf/2011.10566`

- **BYOL** uses an asymmetric framework, where one branch encodes samples with an encoder network, and the other branch encodes augmented samples with an exponential moving average of the encoder of the other branch. Then, the similarity between the embeddings from both branches is maximized as a training objective, but the gradients for the exponential moving average encoder are suppressed.
- **SimSiam** represents a simplistic framework for self-supervised representation learning. It is most similar to BYOL but instead of using an exponential moving average encoder, it uses the same encoder for both branches and solely stops the gradients for one branch in the backpropagation of the similarity maximization objective. That is, this framework is able to learn rich embeddings of samples without the need of multiple networks, negative samples, or other sophisticated tricks.

(d) How does CLIP differ from the contrastive methods above? What are the advantages of that?

**Detailed Solution**

While the aforementioned contrastive methods consider uni-modal data, CLIP is a multi-modal contrastive foundation model. It was trained to contrast images and their textual descriptions by aligning image and text embeddings in a joint latent space. This is useful as it creates a bridge between the image and text domain, allowing recognize image features based on textual descriptions, even in a zero-shot manner (i.e., without explicitly training a classifier for queried semantic concepts/classes).

## Exercise 8-2 Self-supervised Training in NLP

(a) What are CBOW and Skip-gram in Word2Vec?

**Detailed Solution**

They are both training objectives for the word embedding model Word2Vec.

- **CBOW:** The model has to predict a masked-out word in the middle of a context of a fixed size, e.g., ”the mother loves her daughter” $\rightarrow$ ”the mother `[?]` her daughter”
- **Skip-gram:** Only a single word is provided to the model, which has to reconstruct the rest of the context window, e.g., ”the mother loves her daughter” $\rightarrow$ ”`[?] [?]` loves `[?] [?]`”

(b) What are Next Sentence Prediction (NSP) and Masked Language Modeling (MLM)? What famous model were they used for?

**Detailed Solution**

- **NSP:** Given a sentence, the model has to predict, which of (i) the actual next sentence from the text and (ii) a randomly selected sentence from the text corpus is the subsequent sentence.
- **MLM:** In MLM, a certain amount of input tokens is replaced by a so-called `[MASK]` token. The model's task is it then to reconstruct the original, replaced input tokens.

These two objectives are self-supervised as no labels or data annotations are used but solely the sequentiality of language and text are leveraged to define pre-training tasks. These tasks are used in the pre-training of BERT.

## Exercise 8-3 Autoencoder Techniques

(a) What is an autoencoder? Why ”auto”?

**Detailed Solution**

An autoencoder is neural network, typically consisting of an encoder and a decoder, that maps input data to a latent representation and then back to its original form (therefore ”auto”=”self”; the input is encoded such that itself can be reconstructed). Autoencoders are often designed in a way such that the latent representation have desirable properties, e.g., lower-dimensional, rich and condensed information.

(b) What three examples of autoencoders have you seen in the lecture? What are the key ideas behind these methods? What is the central high-level similarity of these methods?

**Detailed Solution**

- Autoencoders with information bottleneck: By enforcing a substantially lower dimensionality than the input dimensionality on the latent representations, the autoencoder is forced to represent the relevant information in a condensed way and to leave out irrelevant and noisy features.
- Denoising autoencoders: The input is distorted with noise and the model has to reconstruct the original, denoised version of the input. In doing so, the model learns meaningful features in the data and to denoise the data.
- Masked autoencoders: Similar to denoising autoencoders, the input is corrupted by masking out certain parts of its representation (e.g., some tokens in text, some patches in images). By learning to reconstruct the missing parts, the model learns meaningful high-level features as it has to understand the relations within different aspects of the data. Masked autoencoders work well with transformer architectures as single tokens can be easily replaced or omitted.

In general, the central commonality of these methods is that the decoder has to **reconstruct the original samples given some degraded representation**, where degradation is achieved through either an information bottleneck, the addition of noise, or masking.