LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR MATHEMATIK, INFORMATIK UND STATISTIK
INSTITUT FÜR INFORMATIK
LEHRSTUHL FÜR DATENBANKSYSTEME UND DATA MINING

Lecture Notes on

Deep Learning and Artificial Intelligence

Winter Semester 2024/25

Self-Supervised Learning

Lectures: Prof. Dr. Matthias Schubert

Tutorials: Maximilian Bernhard

Outline

- learning without ground truth
- local context
- information bottlenecks in encoder-decoder architectures
- object augmentation and contrastive losses
- learning from context and sequence completion

Data for Large Neural Networks

- fitting large models requires large amounts of samples
- in many domains, there are petabytes of observation data
- ground truth data is often missing:
  - manual labelling (by an expert)
  - label quality is often questionable
  - data collection is digital, but labelling is manual

⇒ find new supervision signals to utilise large amounts of training data

⇒ learn a ”good” representation in an unsupervised or semi-supervised way and then require limited labels to train a supervised head under limited ground truth

⇒ open-vocabulary learning: use a language model to describe object meanings. (does not know all target classes in advance)

Unsupervised Learning

What else can we do if no information about what to learn is given?

- Information bottleneck: learn a low-dimensional representation preserving information as well as possible
  - find a lower dimensional manifold containing the data
  - separate noise from essential content
- Use weak labels: if the intended ground truth is unavailable, use available information correlated to Y.
  - images: point supervision for object detection, box supervision for segmentation
  - classification: predict super-classes
  - description texts instead of class labels (e.g. CLIP)
- use context: predictions are based on the correlation between observations (X) and target values (Y)
  - use other information which is correlated to X as a supervision signal
  - use parts of the input to predict other parts
    - sentence completion
    - word2vec
    - patching
  - generate this type of data with data augmentation

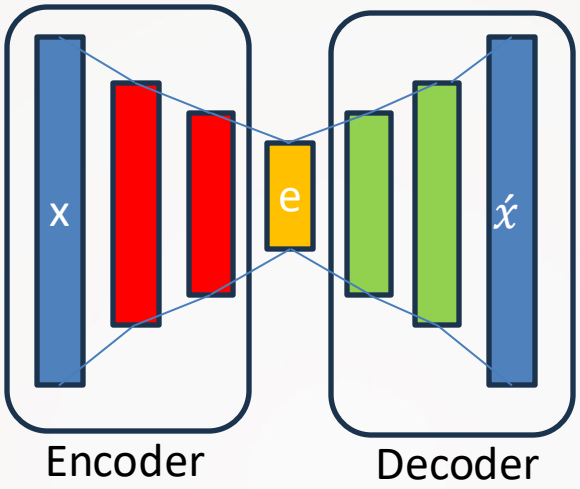
Unsupervised Representation Learning

- learn a more meaningful representation of the data:
  - less noise
  - less (and maybe independent) features
- for example, PCA maps data into a lower dimensional representation by deleting low-variance dimensions: $\Sigma = V \Lambda V^T$
  - PCA work well for univariate multi-variate Gaussians
  - in this setting, we can pick independent features
- in general:
  - no restriction to a linear or invertible mapping to/from the embedding space
  - features do not need to be pairwise-linear independent
  - instead of deleting low-variance dimensions, minimise reconstruction loss:

$$min|X - UV^T|$$

⇒ Auto Encoders use neural networks for encoding and decoding instead of linear mappings

Auto-Encoders



- information bottleneck: Map input x ∈ $\mathbb{R}^d$ to an embedding vector $e \in \mathbb{R}^l$ with $l < d$. With $enc(x, \theta_e) \rightarrow \mathbb{R}^l$ and $dec(e, \theta_d) \rightarrow \mathbb{R}^d$, $dec(enc(x, \theta_e), \theta_d) = \acute{x}$.
- train with square loss over *N* samples : $\downarrow \frac{1}{N} \sum_{i=0}^{N} \|x - \acute{x}\|$
- the decoder and the encoder are trained independently though decoder could be the direct inverse function.
- the depth of both architecture can be stepwise increased and outer layers can be pretrained. (avoid vanishing/exploding gradients)

How do information bottlenecks work?

- reducing the dimensionality of a vector potentially drops information
- the reconstruction loss minimises the amount of information that has to be dropped
- If $\|x - \acute{x}\| = 0$, the e has the same information as x.
  ⇒ x contains redundant information
  ⇒ the manifold of X has a lower internal dimensionality than the feature space it is embedded in (c.f. PCA)
- reducing the number of dimensions is intended to remove redundant information and directly model the manifold as a lower dimensional vector space.
- in practice, autoencoders lose information as input data may contain noise
- However, if the embedding has a too small dimension, the Autoencoder cannot learn a suitable representation as it must drop important information

Denoising Autoencoders

- can we learn an AE with a full-dimensional embedding space? (dimensionality cannot be too small to keep data)
- a full-dimensional embedding yields the danger of learning the identity function
- Can we remove information without mapping to a lower-dimensional space?

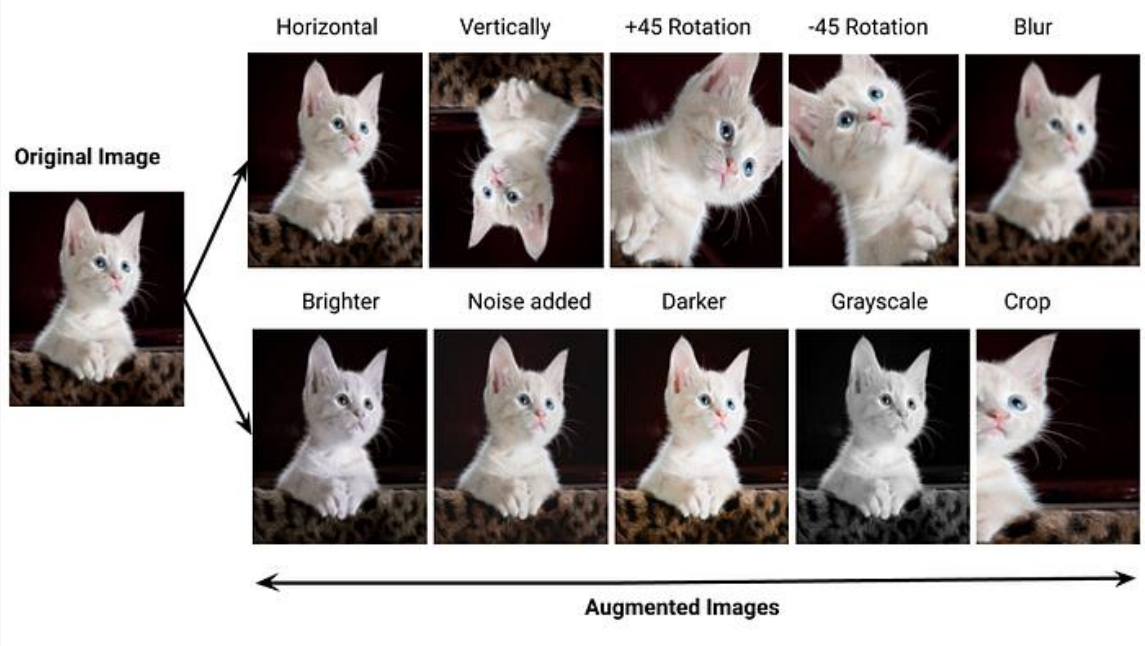
idea: Add noise for every stage of the encoder or drop-out activations?

- provide a noisy input and denoise the output
- here, the embedding vector contains less information due to noise
- the decoder learns to remove noise from the input

http://www.opendeep.org/v0.0.5/docs/tutorial-your-first-model

Data Augmentation

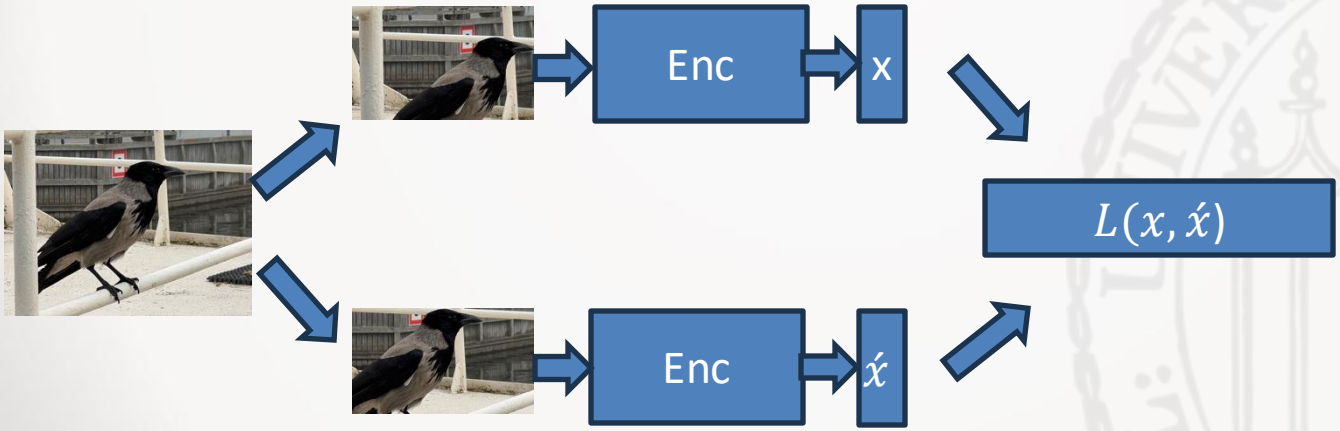
- data augmentation: Apply a transformation to each instance which modifies its content but should not change the meaning.
  $g(x) = \acute{x} \neq x$, but f$(x) = f(\acute{x})$ where f(x)=y is an optimal prediction function.
- Examples: flipping, cropping, adding noise, remove color etc. does not change the object shown on an image.



https://miro.medium.com/v2/resize:fit:720/format:webp/0*rvldTiAUPDiqf_-B.png

Data Augmentation and Self-Supervision

- initially, data augmentation is used to diversify the training set and make predictors robust against augmentation effects
- contrastive learning: learns a data representation that gets invariant concerning augmentation. When using enough augmentation functions, representations yield content but not instance-specific description
- There are various Siamese architectures which learn a joint representation



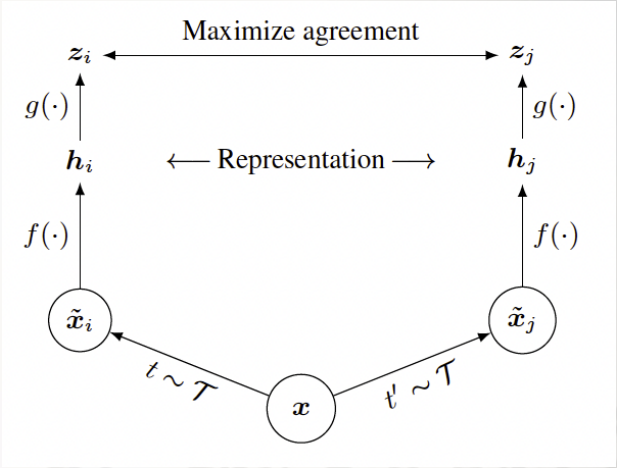
SimCLR

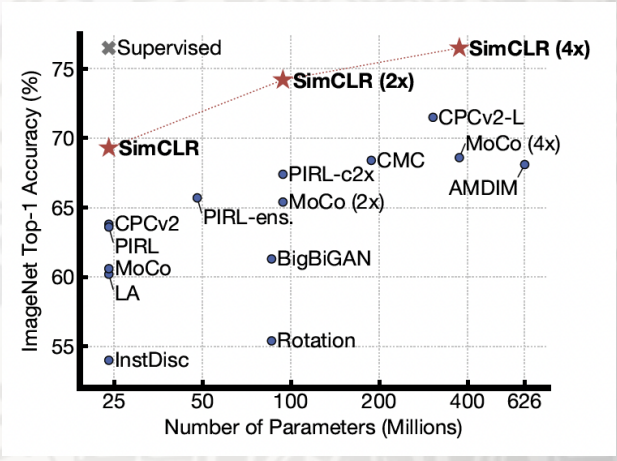
- contrastive loss based on cosine similarity

$$L_{i,j} = -log \frac{\exp(\frac{sim(z_i, z_j)}{\tau})}{\sum_{k=1}^{2N} \mathbb{I}_{[k \neq i]} \exp(\frac{sim(z_i, z_k)}{\tau})}$$

- the loss requires negative samples which are taken from other instances from the batch (requires huge batch sizes ≈2048 instances)

- ImageNet Top-1 accuracy of linear classifiers trained on representations learned with different self-supervised methods (pretrained on ImageNet). Gray cross indicates supervised ResNet-50.





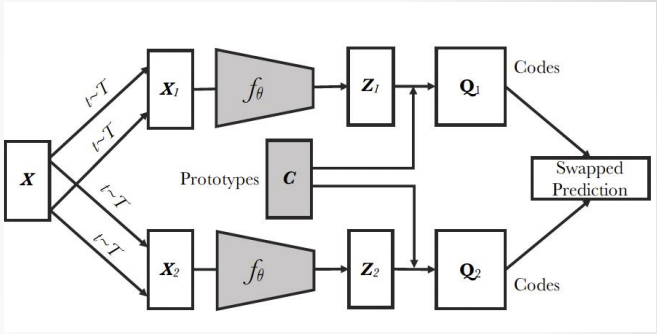
Chen, Ting, et al. "A simple framework for contrastive learning of visual representations." *International conference on machine learning*. PMLR, 2020.

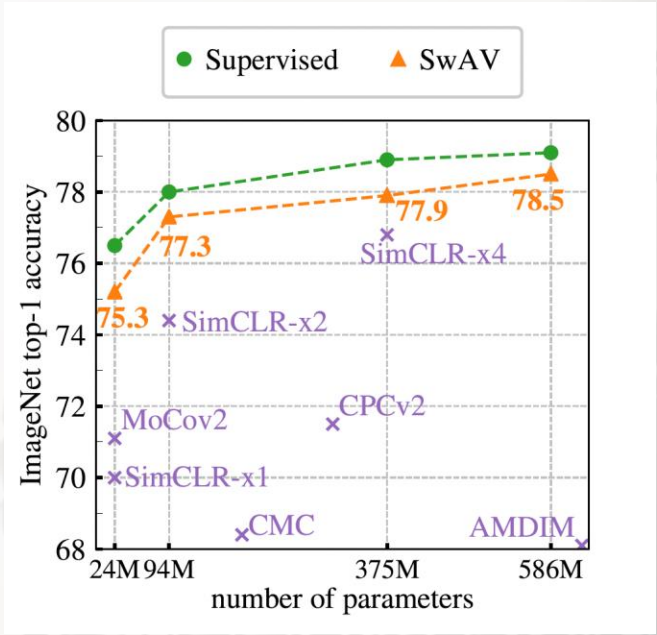
SwAV

- compares representations relative to a set of cluster centres(prototypes) C
- codes q represents closeness to prototypes
- loss between representation z and code q is computed as cross-entropy between q and the assignment of z to C.

$$L(z_t, q_s) = -\sum_k q_s^{(k)} \, log\left(p_t^{(k)}\right)$$

where $p_t^{(k)} = \frac{\exp(\frac{1}{\tau} z_t^T c_k)}{\sum_{\acute{k}} \exp(\frac{1}{\tau} z_t^T c_{\acute{k}})}$

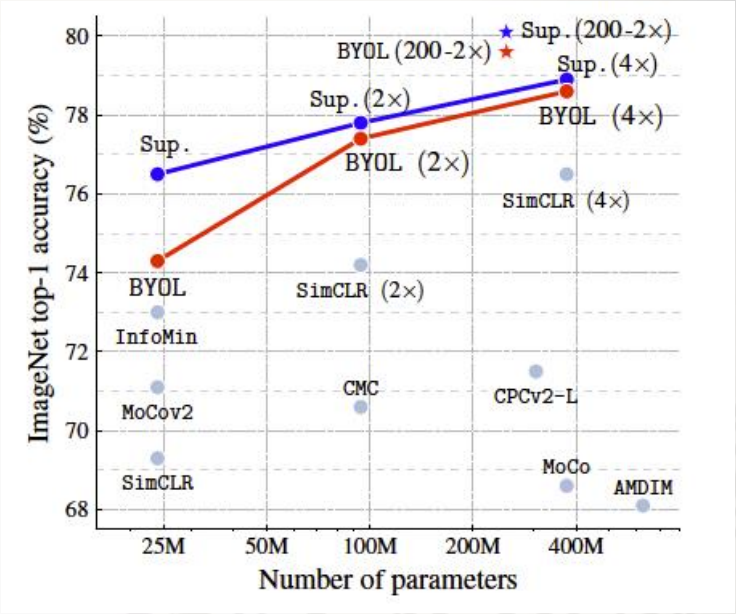


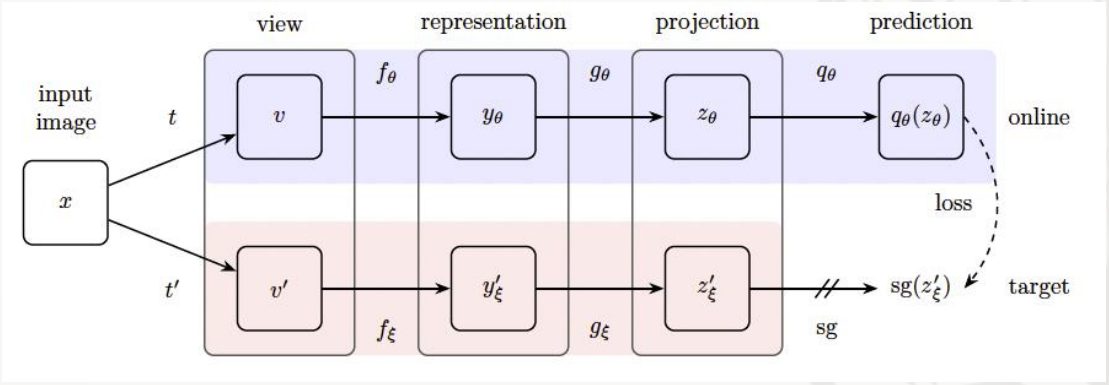


Caron, Mathilde, et al. "Unsupervised learning of visual features by contrasting cluster assignments." *Advances in neural information processing systems* 33 (2020): 9912-9924.

BYOL

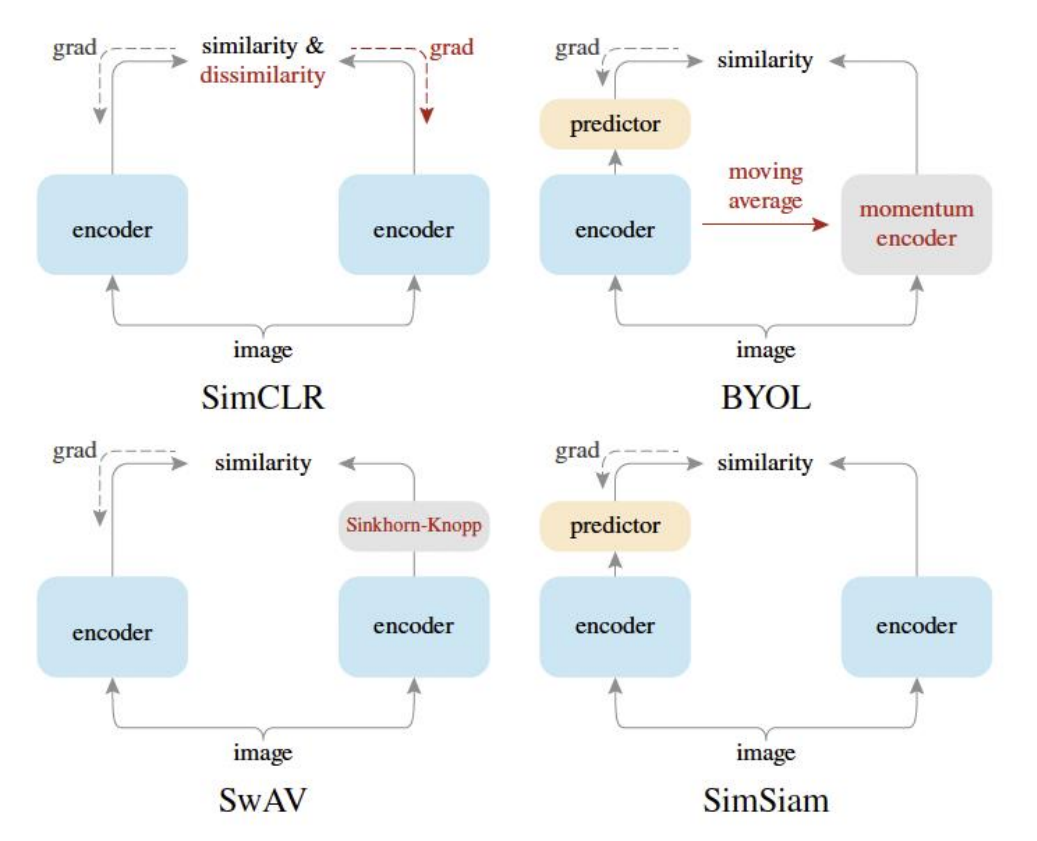
- avoids using negative pairs from the batch
- stop-gradient: only update one branch of the architecture with SGD.
- moving average over θ to describe ξ for the non-SGD branch
- loss based on cosine similarity





Caron, Mathilde, et al. "Unsupervised learning of visual features by contrasting cluster assignments." *Advances in neural information processing systems* 33 (2020): 9912-9924.

SimSiam



Chen, Xinlei, and Kaiming He. "Exploring simple siamese representation learning." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021.

Contrastive Learning Conclusion

- SimSiam performs comparably well but is very simple (only stop gradient)
- more approaches e.g. by varying the loss
- data augmentation works well for images but is often more problematic for other data, e.g., graphs
- Siamese methods also work well if there are multiple views on the same image (e.g. text-image, speech-text, …)

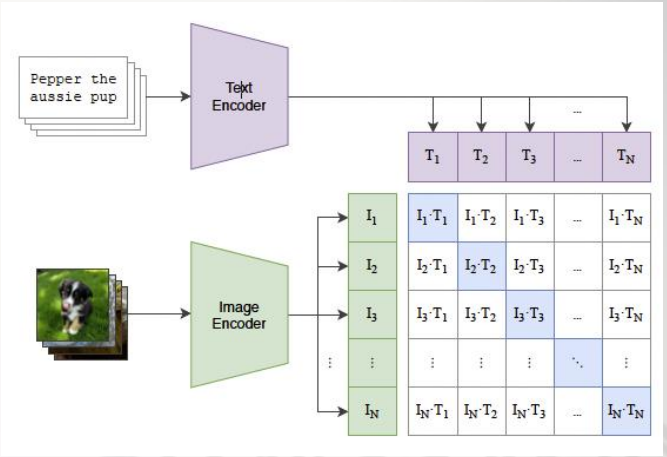
Algorithm 1 SimSiam Pseudocode, PyTorch-like

```
# f: backbone + projection mlp
# h: prediction mlp

for x in loader: # load a minibatch x with n samples
   x1, x2 = aug(x), aug(x) # random augmentation
   z1, z2 = f(x1), f(x2) # projections, n-by-d
   p1, p2 = h(z1), h(z2) # predictions, n-by-d

   L = D(p1, z2)/2 + D(p2, z1)/2 # loss

   L.backward() # back-propagate
   update(f, h) # SGD update

def D(p, z): # negative cosine similarity
   z = z.detach() # stop gradient

   p = normalize(p, dim=1) # l2-normalize
   z = normalize(z, dim=1) # l2-normalize
   return -(p*z).sum(dim=1).mean()
```

method	batch size	negative pairs	momentum encoder	100 ep	200 ep	400 ep	800 ep
SimCLR (repro.+)	4096	✓		66.5	68.3	69.8	70.4
MoCo v2 (repro.+)	**256**	✓	✓	67.4	69.9	71.0	72.2
BYOL (repro.)	4096		✓	66.5	**70.6**	**73.2**	**74.3**
SwAV (repro.+)	4096			66.5	69.1	70.7	71.8
SimSiam	**256**			**68.1**	70.0	70.8	71.3

Multi-Modal Contrastive Learning

- learn joint multi-modal representations
- can be used to describe one mode via the other (text from images or images from text)
- most well-known image-text vocabulary foundation model is CLIP
- CLIP jointly trains an image and a text encoder to derive similar descriptors
- loss is based on scaled pairwise cosine similarity
- which is processed in a symmetric cross-entropy of both modes
- CLIP models were pre-trained on large text-image data sets and can be used as foundation models without further training.

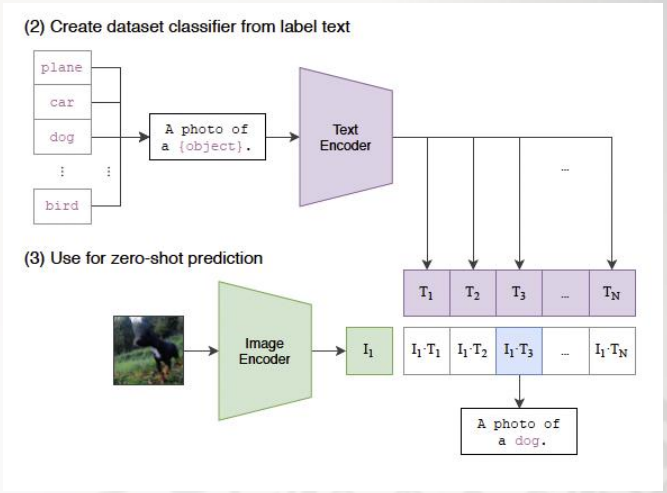



```
# image_encoder - ResNet or Vision Transformer
# text_encoder  - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

Radford, Alec, et al. "Learning transferable visual models from natural language supervision." *International conference on machine learning*. PMLR, 2021.

CLIP usage

- CLIP can be used for zero-shot inference:
  - encode class description (A photo of …)
  - compare image encoding to the class description in the embedding space
- Open Vocabulary: there is no need to specify classes before the training. Any class which can be described can be predicted
- CLIP is used for replacing fixed labels and concepts with text descriptors



Clipping and Completion

another target for contrastive learning is to remove parts of the input and try to complete the object based on the remaining part:

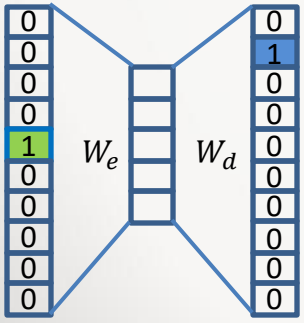
- predict missing tokens of a sentence
- completing sentences
- predict the surrounding words (Word2VEC)
- learn whether a sentence follows the previous one
- remove crops from images and predict the rest
- clip channels from an image and predict their values from the remaining channels.

This allows for supervised losses as a particular value needs to be predicted.

Word2Vec with Skip-Gram

- Given an alphabet $A = \{a_1, .., a_m\}$ and dataset $S = \{S_1, .., S_n\}$ sequences $S_i = [\alpha_1, .., \alpha_l]$ where $\alpha_j \in A$.
- context $C_w(\alpha_i) = [\alpha_{i-w}, ., \alpha_i, ., \alpha_{i+w}]$ of length w of element α_i
- the one-hot encoding V(α_i) is a m-dimensional vector with $v_i = 1$ for $v_i = 0$ for $i \neq j$.
- the encoder E(α_i) maps V(α_i) to a k-dimensional embedding vector e_i
- the decoder $D(e_i)$ maps e_i to $V(\alpha_j)$ where $\alpha_j \in C_w(\alpha_i)$
- Both E(α_i) and $D(e_i)$ are modelled as linear functions:

$$\mathrm{E}(\alpha_i) = V(\alpha_i)^T W_e \text{ and } \quad \mathrm{D}(e_i) = e_i^T W_d$$



NOTE: Each $\alpha_j \in C_w(\alpha_i)$ is predicted separately. Thus, there are 2w many pairs for each word.

Training Skip-Grams

- The output of $\mathrm{D}(e_i) = e_i^T W_d$ is generally not sparse, but the $\mathrm{V}(\alpha_i)$ is a one hot encoding:

  => apply softmax function: $\mathrm{S}(x) = \frac{e^{x_i}}{\sum_{j=0}^{d} e^{x_j}}$

- for each pair α_i and $\alpha_j \in C_w(\alpha_i)$, we now maximize $P(\alpha_j|\alpha_i)$ or minimize $-\log P(\alpha_j|\alpha_i)$
- Let $f_j(\alpha_i) = \mathrm{D}\left(\mathrm{E}(\mathrm{V}(\alpha_i))\right)_j$ denote the decoder output in dimension j

  minimize: $-\log\left(\frac{e^{f_j(\alpha_i)}}{\sum_{l=0}^{d} e^{f_l(\alpha_i)}}\right) = -f_j(\alpha_i) + log \sum_{l=0}^{d} e^{f_l(\alpha_i)}$

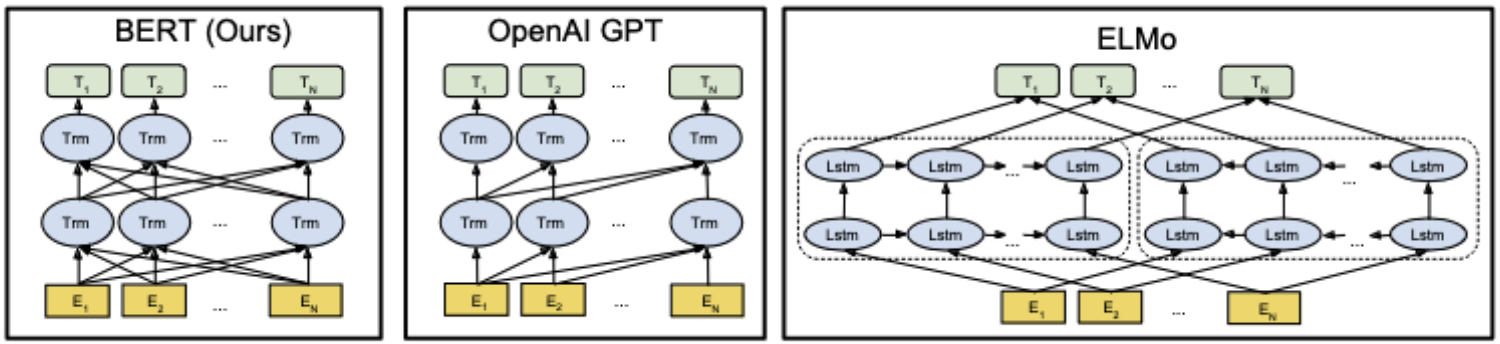
- training is done by gradient descent
- Continuous Bag of Words (CBOW): Switches input and output of the model, i.e. $\alpha_j \in C_w(\alpha_i)$ is the input to predict α_i

Improving training

- **Subsampling Frequent Words**
  - To avoid over training on frequent words (e.g. the, is,..), we can randomly delete words based on their appearance frequency
  - Rare words are deleted less likely than frequent words
  - All pairs containing a deleted word are ignored
- **Negative Sampling**
  - Estimating $log \sum_{l=0}^{d} e^{f(\alpha_l)}$ in the loss function has to iterate over the complete output vector
  - The term aims to make $f(\alpha_l)$ small for words which are not the context ⇒ all words except one are trained to become small
  - Since training is done very frequently anyway, sample the negative words, i.e. only train five of them
  - Negative samples are drawn w.r.t. appearance frequency of the word in the training dataset

Bidirectional Encoder Representations from Transformers (BERT)

- first popular Large Language Model, which trains a general text representation which allows for fast fine-tuning to particular tasks
- idea: train a transformer architecture to complete sentences
- as we learn a language model, BERT is an encoder without a specific decoder
- bidirectional means that the complete sentence is fed into the encoder
- masked training (MLM): replace ca. 15 % of tokens with a MASKED-token and try to predict the original word.
- next sentence prediction (NSP): built pairs of consecutive sentences (pos) and non-consecutive sentences (neg) and train a binary classifier

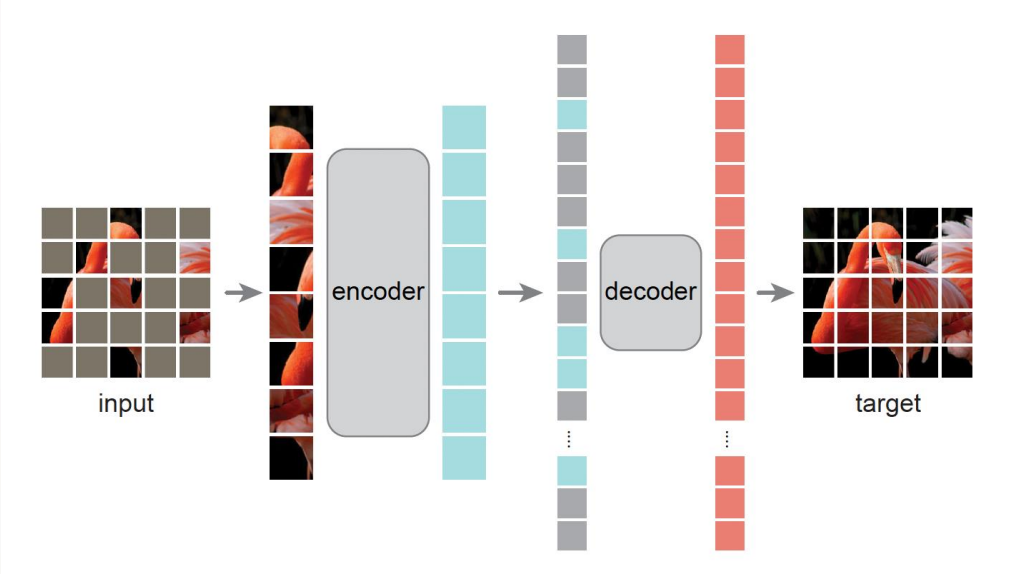


Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

Masked Autoencoders

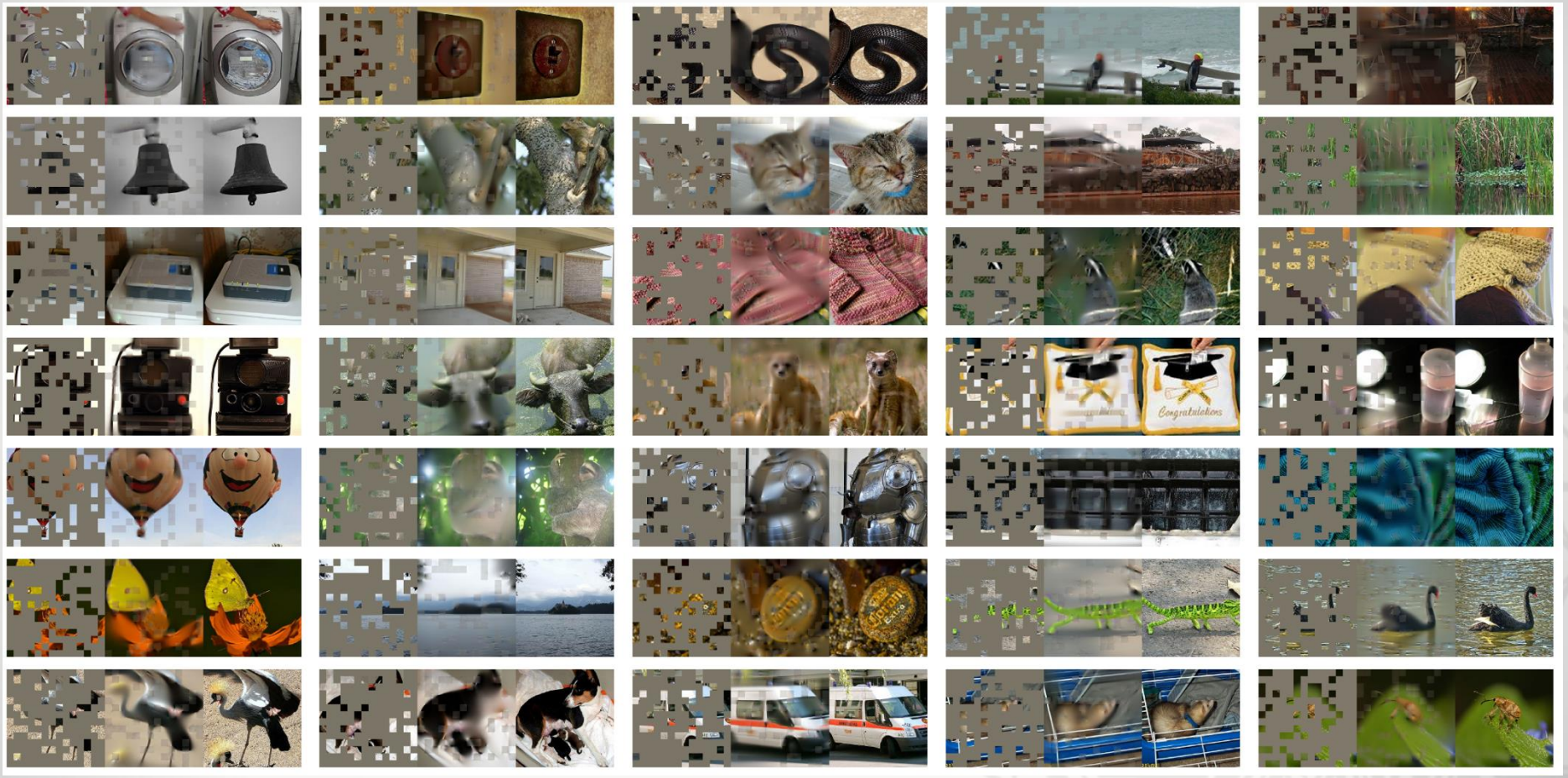
- train ViT encoder on subset of input tokems (only 25 % sufficient)
- train ViT decoder on all token (masked tokens with positional embedding)
- compute loss on pairs of original and learned tokens
- goal unsupervised training of a ViT backbone on unlabeled image sets
- encoder receives only unmasked tokens
- decoder can be lightweight for backbone training

method	pre-train data	ViT-B	ViT-L	ViT-H	ViT-H_{448}
scratch, our impl.	-	82.3	82.6	83.1	-
DINO [5]	IN1K	82.8	-	-	-
MoCo v3 [9]	IN1K	83.2	84.1	-	-
BEiT [2]	IN1K+DALLE	83.2	85.2	-	-
MAE	IN1K	83.6	85.9	86.9	**87.8**



He, Kaiming, et al. "Masked autoencoders are scalable vision learners." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022.

Examples: Masked Autoencoders



Conclusions

- self-supervised training is one of the major success stories in deep learning
- it is not a neural network method, but it usually requires some redundancy and correlations between the observed features
- discussed mechanisms:
  - information bottlenecks
  - noising
  - data augmentation
  - multi-modal learning
  - Masking
- self-supervised learning might not yield optimal task-specific training
- well-suited for foundation models where there are huge training sets and no particular goal (feature maps like resnet101, ViT, LLM like BERT, GPTx)
- Open Vocabulary is related as it relaxes the label space of supervised learning to a textual embeddings