

Representation Learning

Representation Learning

- Techniques that transform a form of raw data into a representation that can be effectively exploited for machine learning tasks
- Representation learning typically refers to learning such a transformation that can generalize across tasks

What are good representations?

- Representation encodes priors about data distribution(s)
 - Smoothness: close inputs map to close outputs
 - Compactness: input dimension \gg output dimension
 - Robustness: features are insensitive to input noise
 - Abstraction and invariances -> problem driven

What are representations?

- The feature vector can be used by other models to produce outputs, e.g.,
 - Classification



feature

[81, 20, 84, 64, ...]



"CAT"

What are representations?

- The feature vector can be used by other models to produce outputs, e.g.,
 - Reconstruction



feature

[81, 20, 84, 64, ...]



What are representations?

- The feature vector can be used by other models to produce outputs, e.g.,
 - Generation



What are representations?

- Representation examples
 - Handcrafted attribute
 - Gender: {"female": 0, "male": 1}
 - Eye color: {"blue": 0, "brown": 1}
 - Hair color: {"black": 0, "blond": 1}



feature

[1, 0, 1]

What are representations?

- Representation examples
 - Binary (one-hot vector)
 - {"Paris": 0, "London": 1, "Munich": 2, ...}



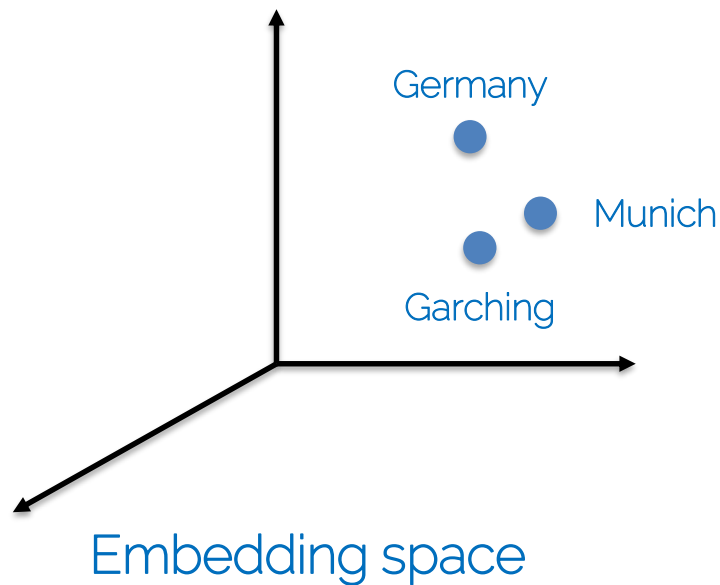
What are representations?

- Representation examples
 - Embedding vector

"Germany"

"Munich"

"Garching"



Representation in Computer Vision

Supervised

Constrained on task(s), e.g.,
classification



"CAT"

Unsupervised

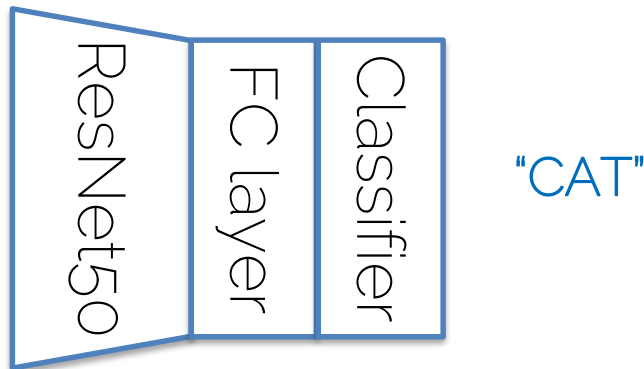
Constrained on data itself,
e.g., reconstruction



Supervised Approaches

- Classification
 - Train ResNet50 on ImageNet
 - Use the features in the last layer as image representations

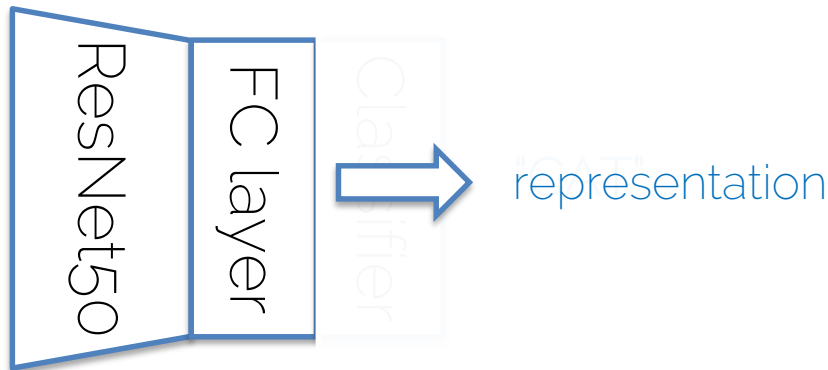
During training:



Supervised Approaches

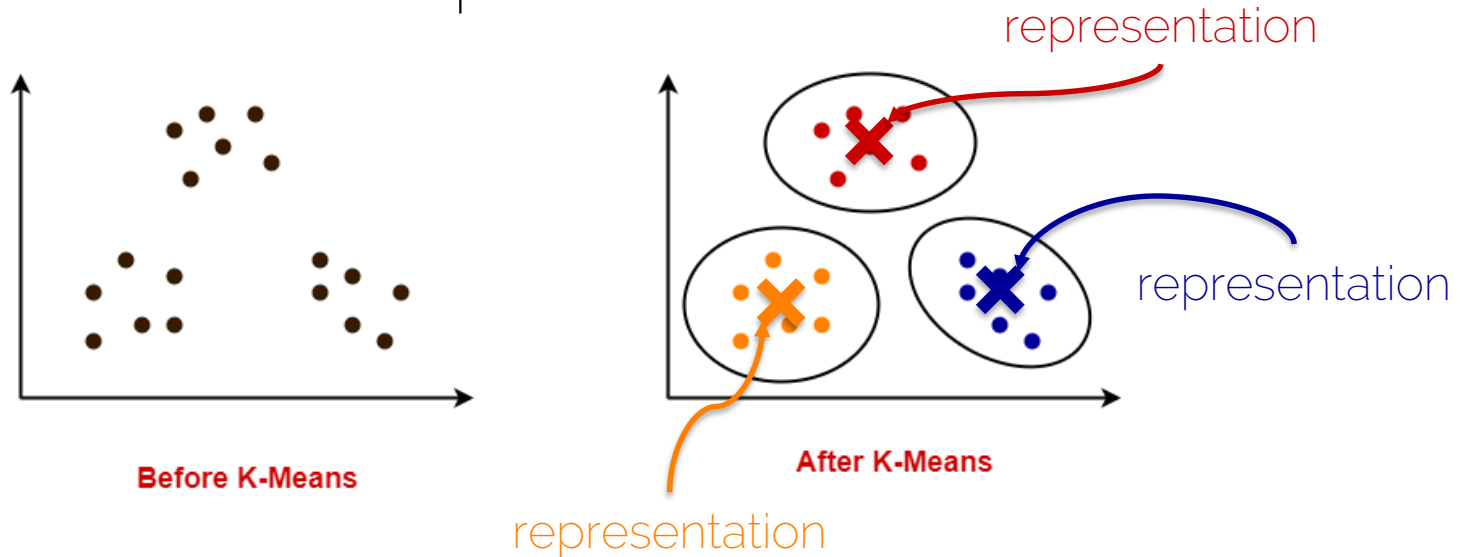
- Classification
 - Train ResNet50 on ImageNet
 - Use the features in the last layer as image representations

After training:



Unsupervised Approaches

- Clustering (K-Means)
 - Mean vectors as representations

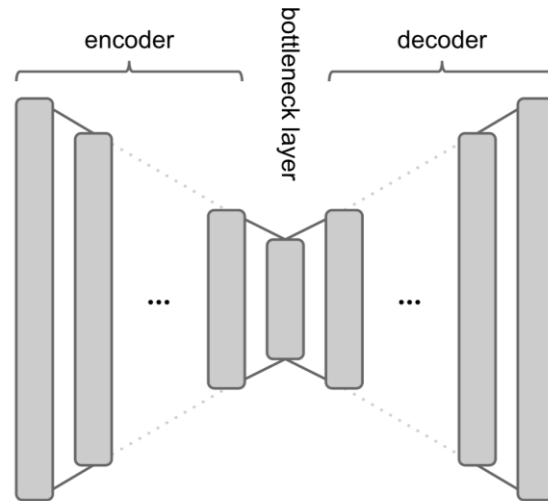


Self-supervised Approaches

- A form of unsupervised learning approaches where the data provides the supervision for itself
- With a proxy loss, e.g., reconstruction loss, the network is forced to learn the features we care about, e.g., semantic representations
- Why self-supervised?
 - Hard and expensive to obtain annotations
 - Alternative to the strong supervisions (labels)

Self-supervised Approaches

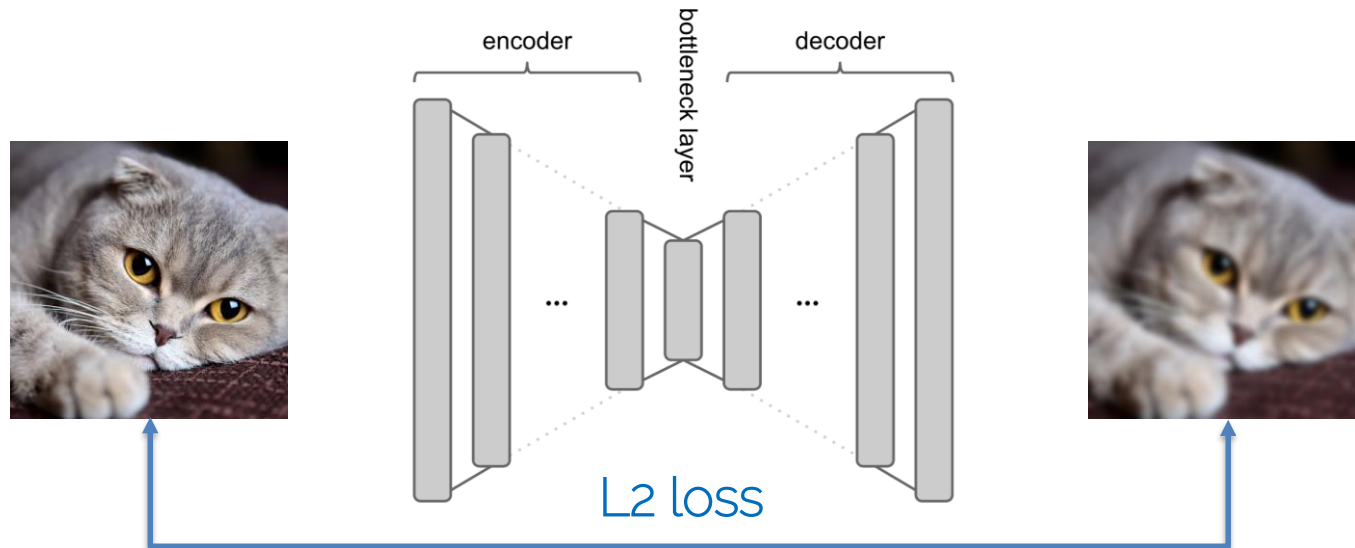
- A form of unsupervised learning approaches where the data provides the supervision for itself



Reconstruction
No label!

Self-supervised Approaches

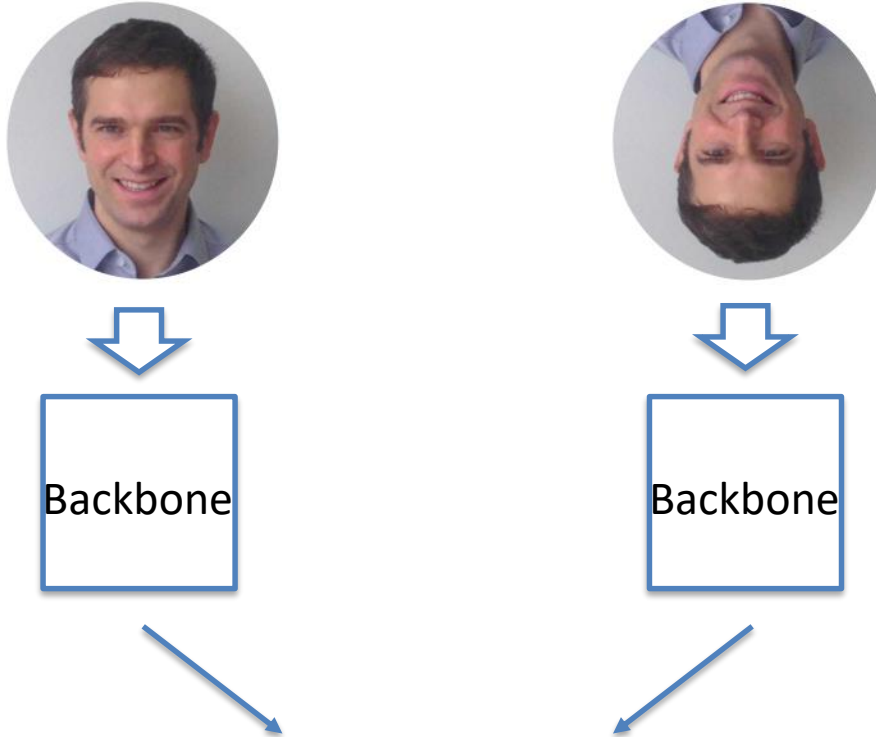
- With a proxy loss, e.g., reconstruction loss, the network is forced to learn the features we care about, e.g., [semantic representations](#)



Self-supervised Approaches

- Why self-supervised?
 - Hard and expensive to obtain annotations
 - Make the most out of the existing unlabelled data
 - Instagram: >1 billion images uploaded / day
 - YouTube: >300 hrs of vides uploaded / minute
 - Alternative to the strong supervisions (labels)

Self-supervision by Augmentation



Loss

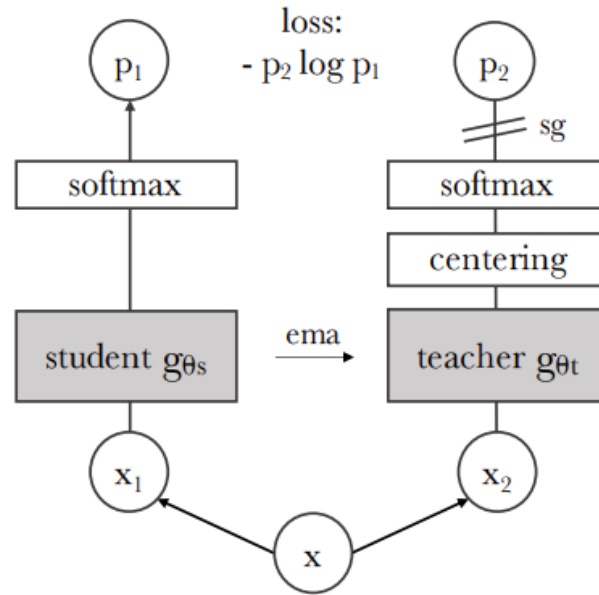
Augmentation is an Art:

- Image vs patch basis
- Color variations
- Geometric transforms
- ...

Losses we have already seen some
-> contrastive learning is popular

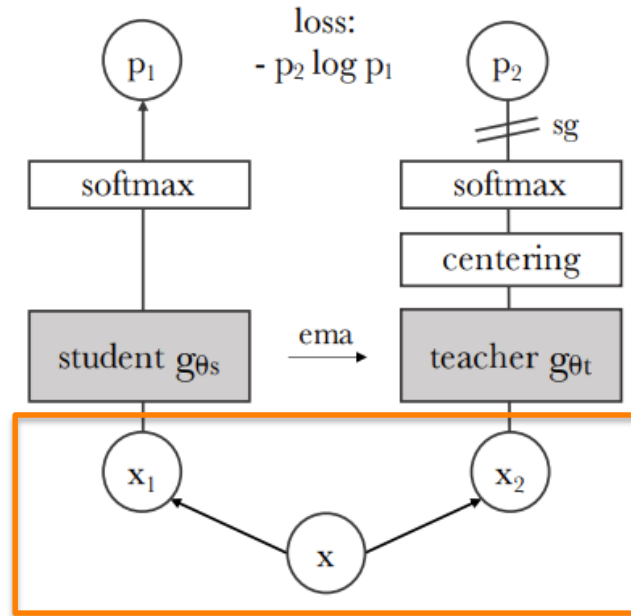
DINO

- Self-distillation with **no** labels



DINO

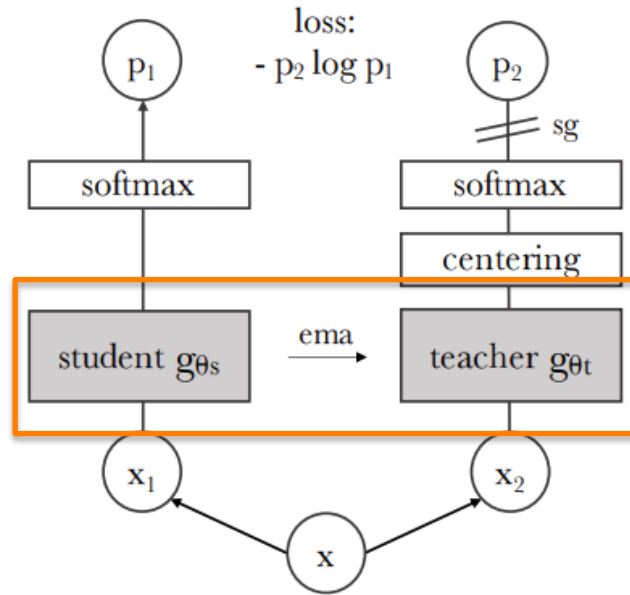
- Self-distillation with **no** labels



A pair of two random transformations of the image input

DINO

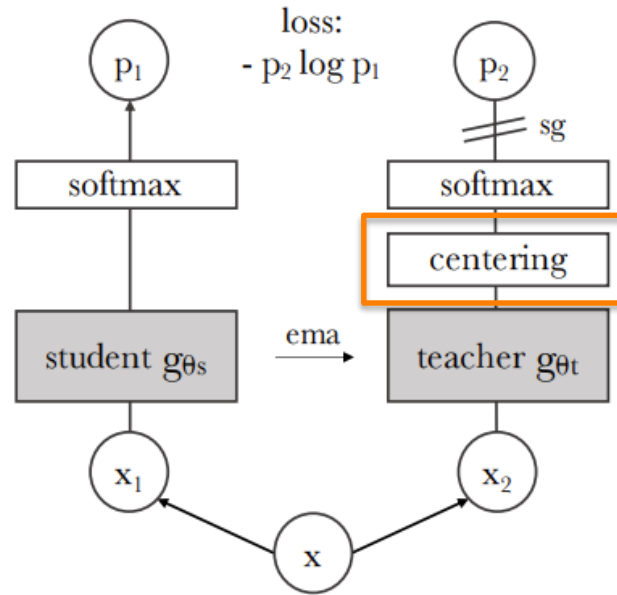
- Self-distillation with **no** labels



Same image encoder,
e.g., ResNet50, but
different parameters

DINO

- Self-distillation with **no** labels



Teacher's output is centered over the batch

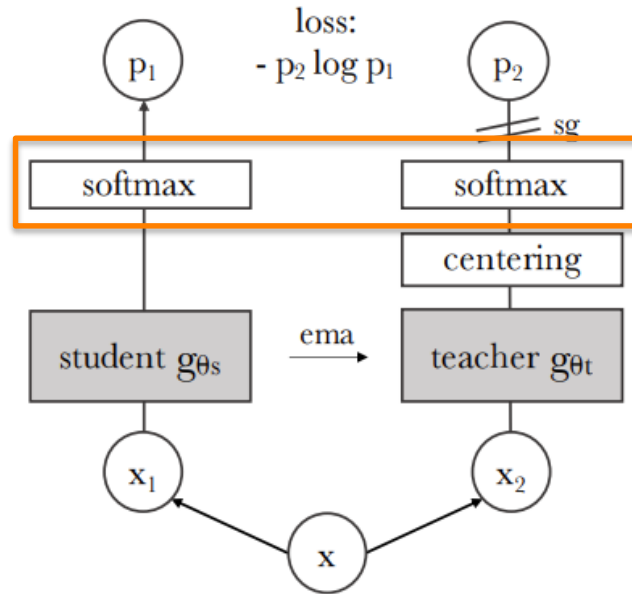
DINO

- Self-distillation with **no** labels

For a batch with K features

$$P_s(x)^{(i)} = \frac{\exp(g_{\theta_s}(x)^{(i)}/\tau_s)}{\sum_{k=1}^K \exp(g_{\theta_s}(x)^{(k)}/\tau_s)}$$

Network outputs



Network outputs are normalized by a temperature softmax over the feature dimension

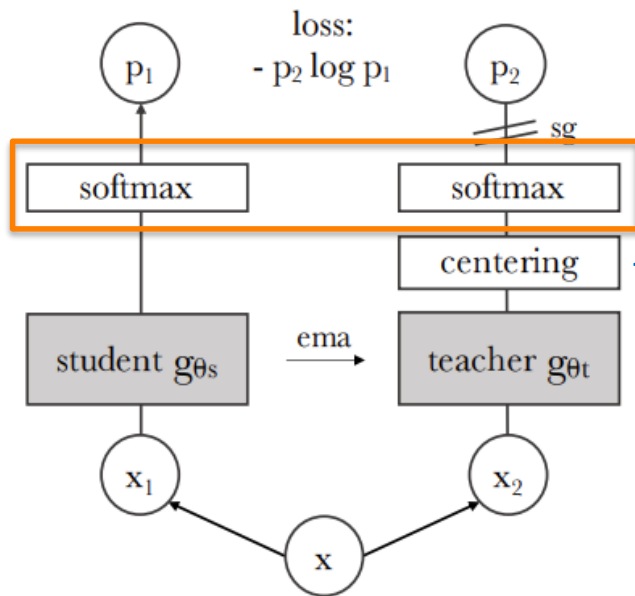
DINO

- Self-distillation with **no** labels

For a batch with K features

$$P_s(x)^{(i)} = \frac{\exp(g_{\theta_s}(x)^{(i)}/\tau_s)}{\sum_{k=1}^K \exp(g_{\theta_s}(x)^{(k)}/\tau_s)}$$

Temperature



Network outputs are normalized by a temperature softmax over the feature dimension

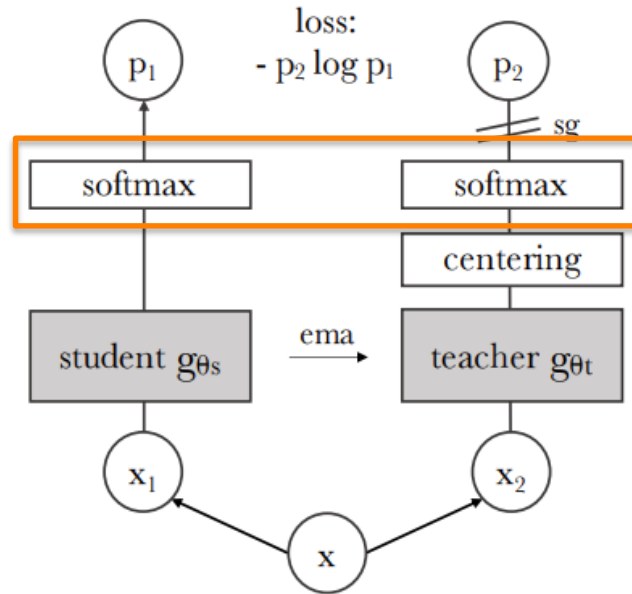
DINO

- Self-distillation with **no** labels

For a batch with K features

$$P_s(x)^{(i)} = \frac{\exp(g_{\theta_s}(x)^{(i)} / \tau_s)}{\sum_{k=1}^K \exp(g_{\theta_s}(x)^{(k)} / \tau_s)}$$

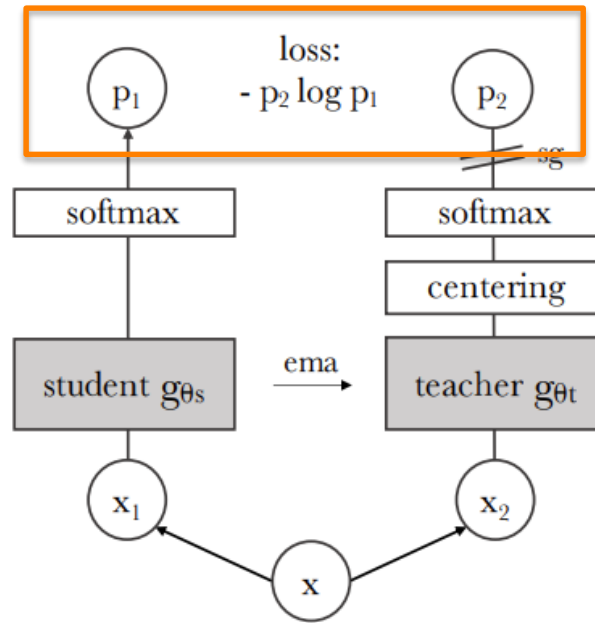
Normalized features



Network outputs are normalized by a temperature softmax over the feature dimension

DINO

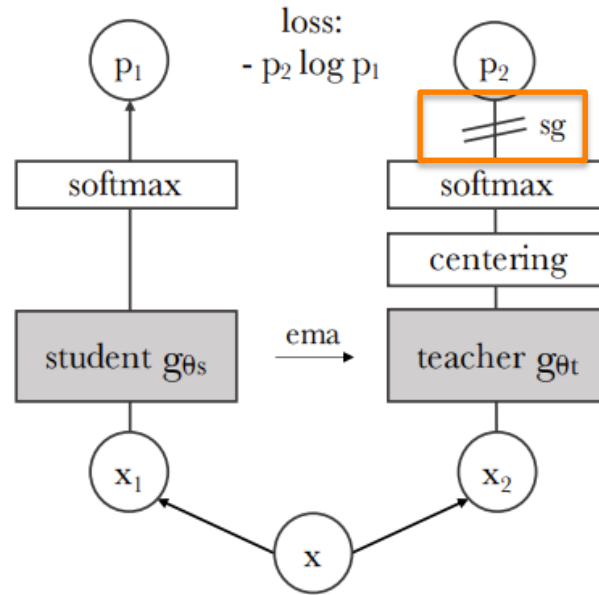
- Self-distillation with **no** labels



The similarity of two outputs is measured by a cross-entropy loss

DINO

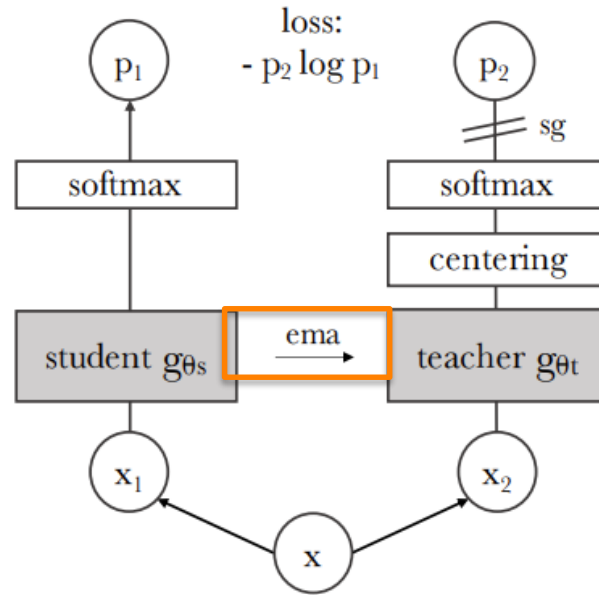
- Self-distillation with **no** labels



A stop-gradient operator is applied on the teacher to back-prop gradients only through the student

DINO

- Self-distillation with **no** labels



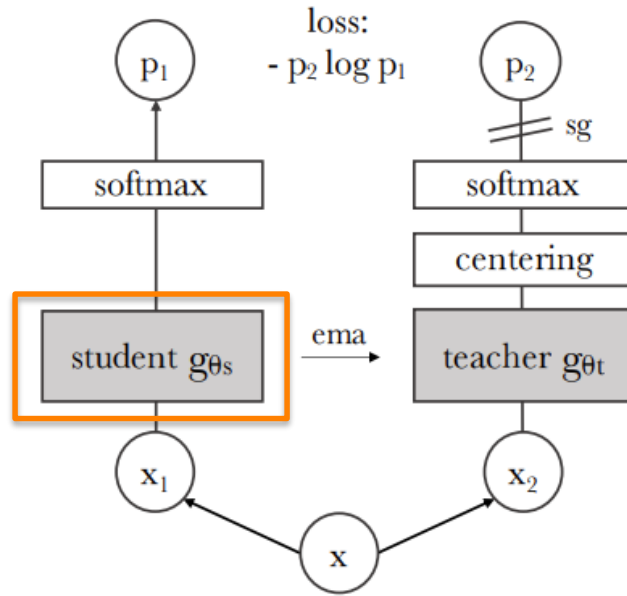
The teacher is essentially built from past iterations of the student

The teacher parameters are updated with an exponential moving average of the student parameters

DINO

- Self-distillation with **no** labels

The trained student network is used for feature extraction



Algorithm 1 DINO PyTorch pseudocode w/o multi-crop.

```
# gs, gt: student and teacher networks
# C: center (K)
# tps, tpt: student and teacher temperatures
# l, m: network and center momentum rates
gt.params = gs.params
for x in loader: # load a minibatch x with n samples
    x1, x2 = augment(x), augment(x) # random views

    s1, s2 = gs(x1), gs(x2) # student output n-by-K
    t1, t2 = gt(x1), gt(x2) # teacher output n-by-K

    loss = H(t1, s2)/2 + H(t2, s1)/2
    loss.backward() # back-propagate

    # student, teacher and center updates
    update(gs) # SGD
    gt.params = l*gt.params + (1-l)*gs.params
    C = m*C + (1-m)*cat([t1, t2]).mean(dim=0)

def H(t, s):
    t = t.detach() # stop gradient
    s = softmax(s / tps, dim=1)
    t = softmax((t - C) / tpt, dim=1) # center + sharpen
    return - (t * log(s)).sum(dim=1).mean()
```

DINO

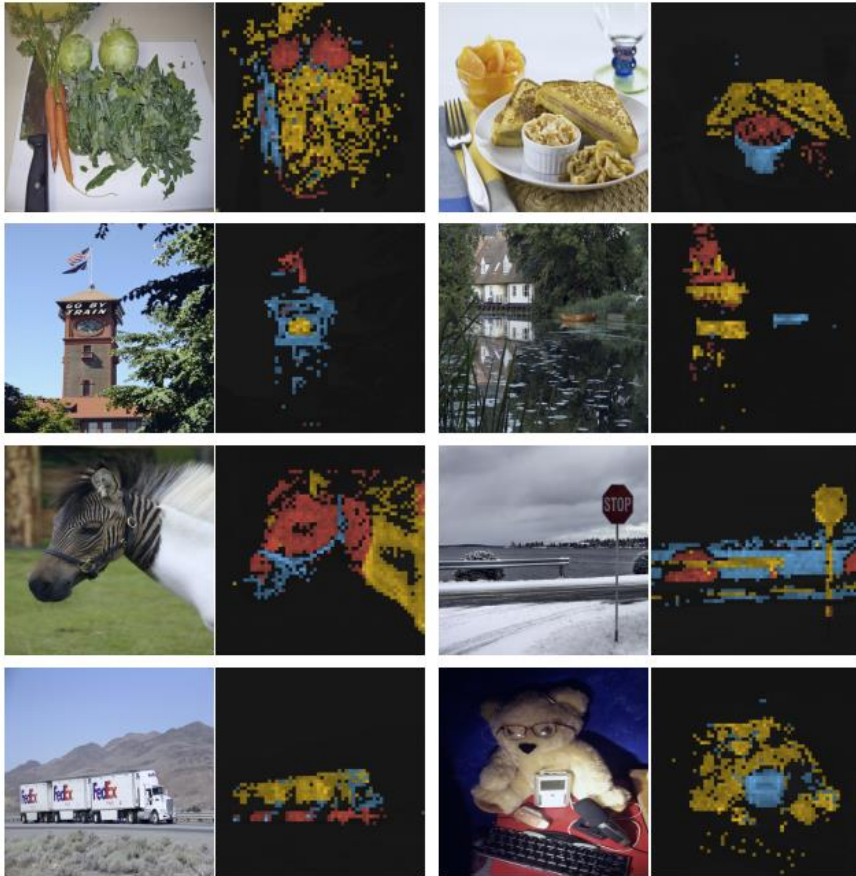
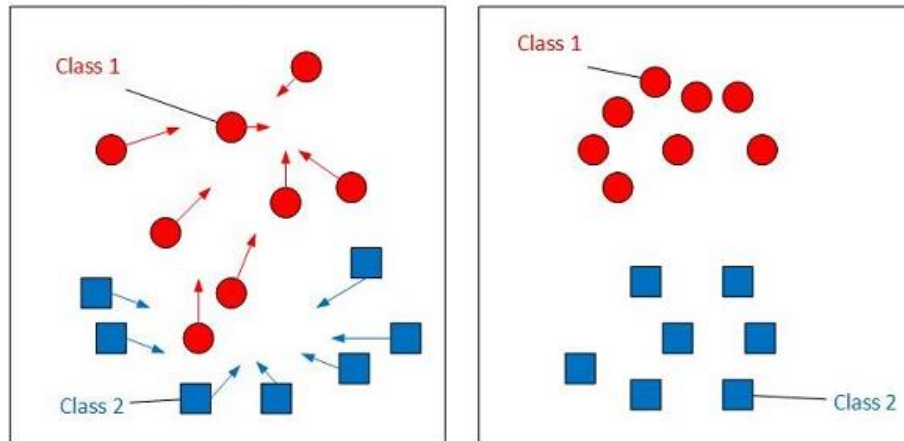


Figure 3: **Attention maps from multiple heads.** We consider the heads from the last layer of a ViT-S/8 trained with DINO and display the self-attention for [CLS] token query. Different heads, materialized by different colors, focus on different locations that represents different objects or parts (more examples in Appendix).

Contrastive Learning Approaches

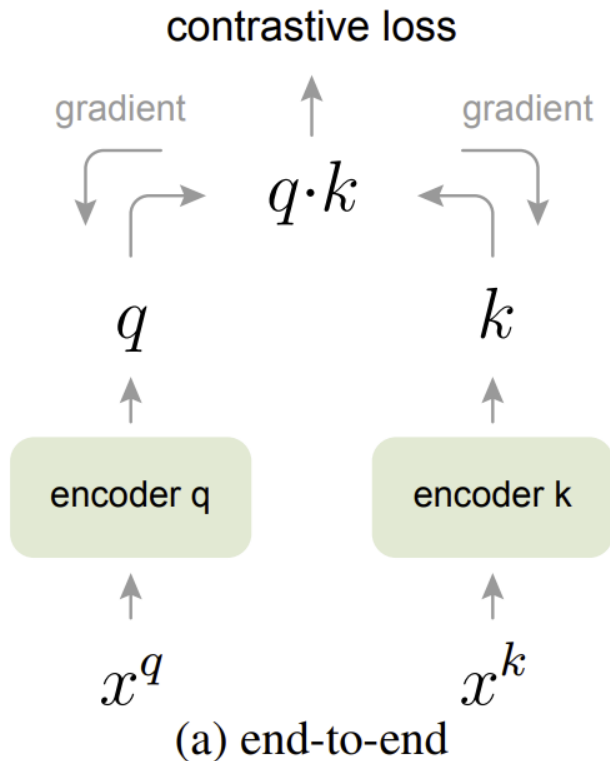
- What is contrastive learning?
 - To learn an embedding space in which similar samples pairs stay close while dissimilar ones repel



Contrastive Learning Approaches

- Can be both supervised and unsupervised
 - With labels? Without labels?
- Can be even used in semi-supervised setting -> some samples are annotated, others not
- When working with unsupervised data, it is one of the most powerful approaches in self-supervised setting

Contrastive Learning Approaches

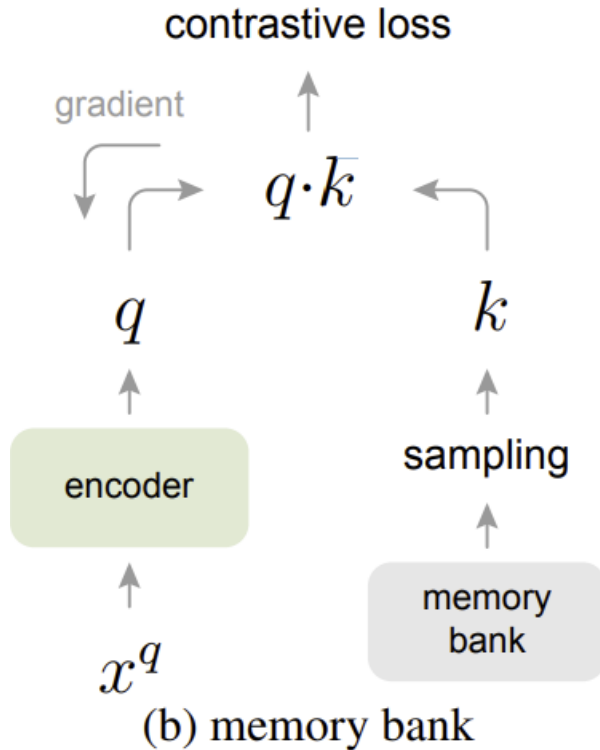


$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k_+ / \tau)}{\sum_{i=0}^K \exp(q \cdot k_i / \tau)}$$

InfoNCE Loss: $(k+1)$ -way softmax classifier

Issue: k is coupled to the mini-batch size which limits k by GPU memory

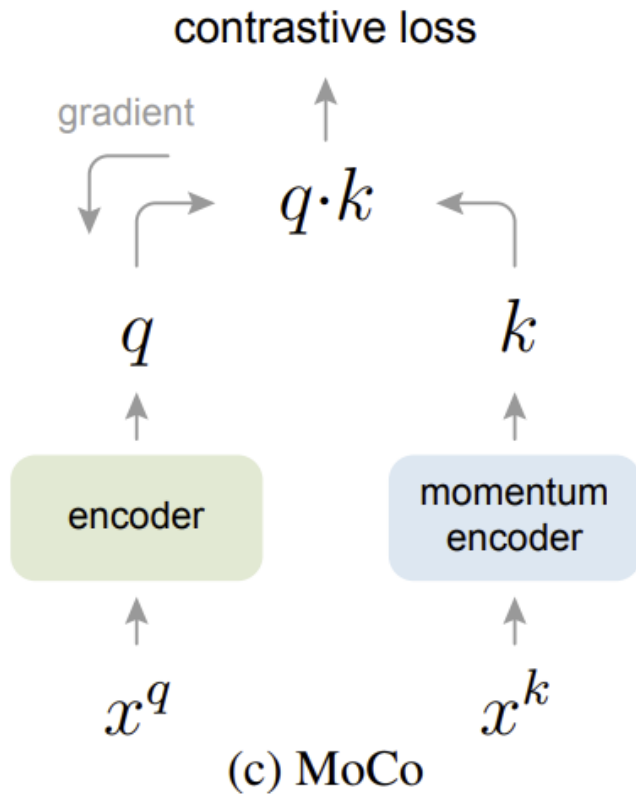
Contrastive Learning Approaches



Idea: don't update keys at the same time but compare to encodings from memory bank

-> allows for large k but encodings are not up to date (typically once per epoch)

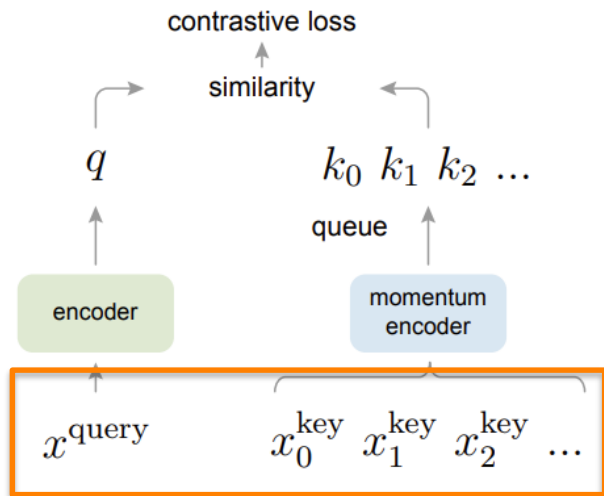
Contrastive Learning Approaches



- **M**omentum **C**ontrast for unsupervised visual representation learning
- A self-supervised learning algorithm with a contrastive loss
- Enables learning a large and consistent visual representation

MoCo

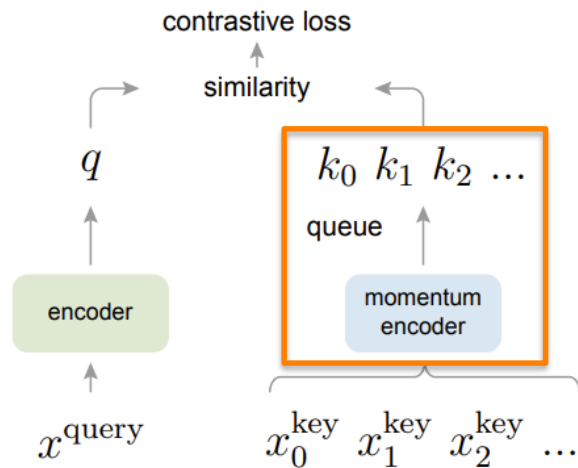
Can be thought of as building a dynamic dictionary



Samples from the dataset

MoCo

Can be thought of as building a dynamic dictionary

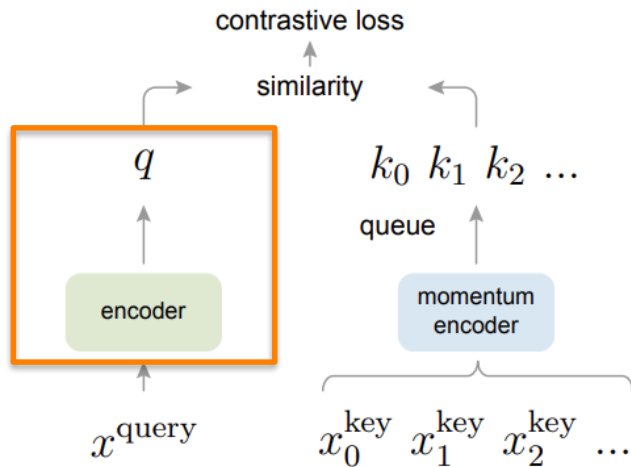


“key”: samples encoded on-the-fly by a slowly updating encoder

MoCo

Can be thought of as building a dynamic dictionary

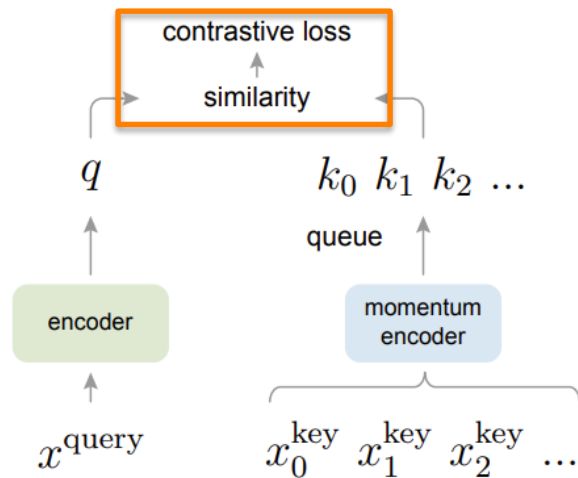
“query”: samples encoded by another encoder to match the keys in dictionary



MoCo

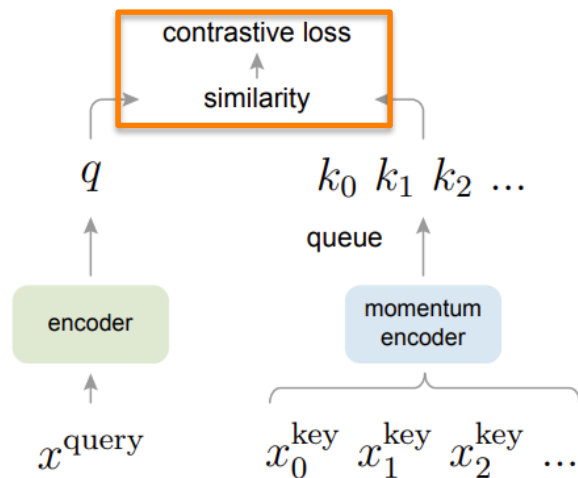
Can be thought of as building a dynamic dictionary

The similarities between the query and keys are supervised by a contrastive loss



MoCo

Can be thought of as building a dynamic dictionary



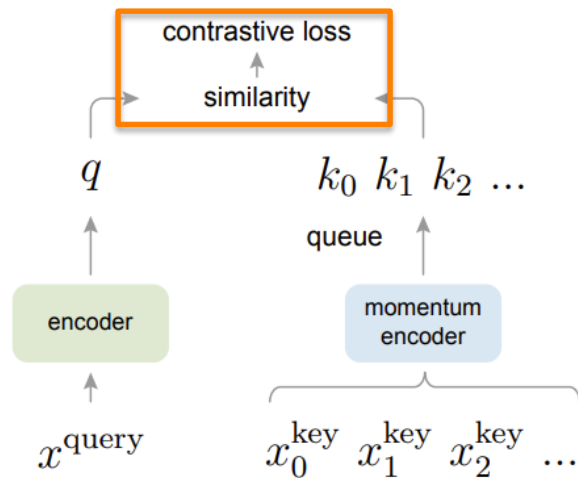
InfoNCE loss function:

$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k_+ / \tau)}{\sum_{i=0}^K \exp(q \cdot k_i / \tau)}$$

Query

MoCo

Can be thought of as building a dynamic dictionary



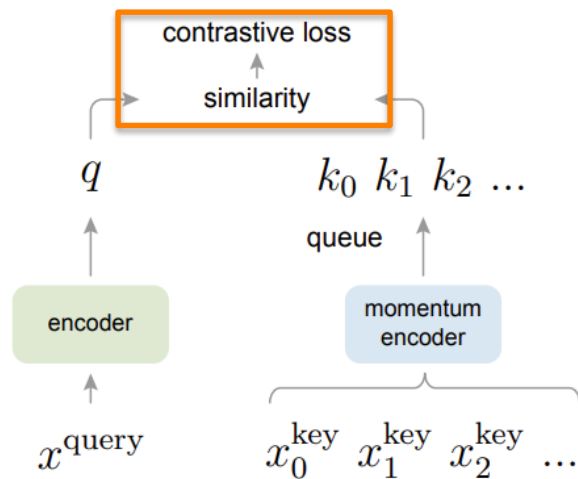
InfoNCE loss function:

$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k_+ / \tau)}{\sum_{i=0}^K \exp(q \cdot k_i / \tau)}$$

Matched key

MoCo

Can be thought of as building a dynamic dictionary



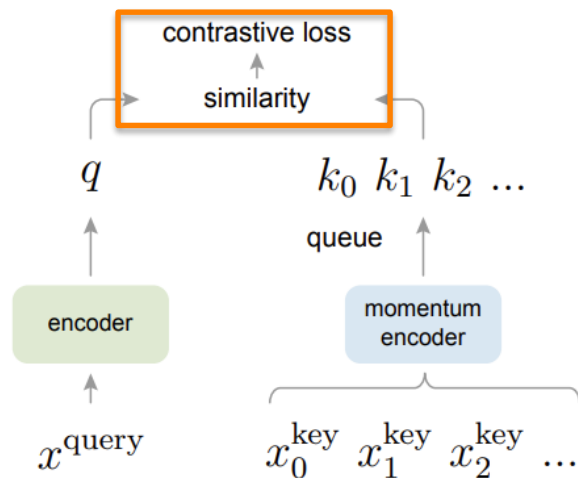
InfoNCE loss function:

$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k_+ / \tau)}{\sum_{i=0}^K \exp(q \cdot k_i / \tau)}$$

All keys

MoCo

Can be thought of as building a dynamic dictionary



InfoNCE loss function:

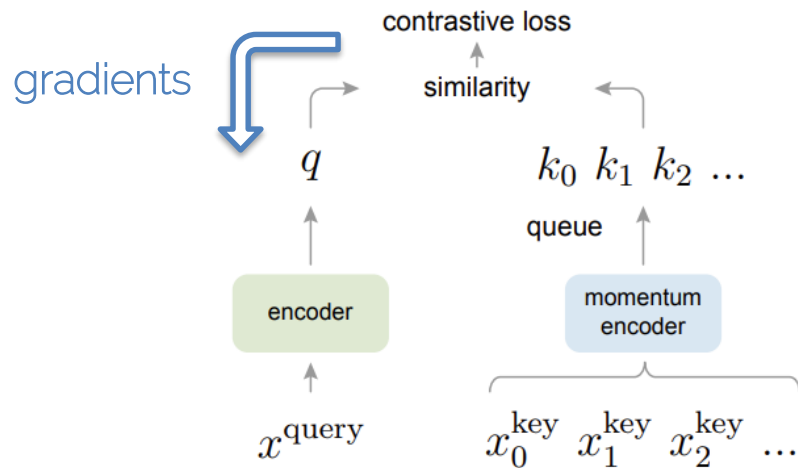
$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k_+ / \tau)}{\sum_{i=0}^K \exp(q \cdot k_i / \tau)}$$

temperature

Dot products
as similarities

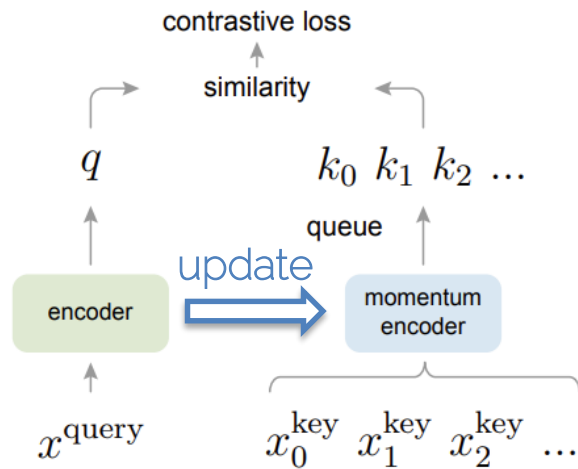
MoCo

Can be thought of as building a dynamic dictionary



MoCo

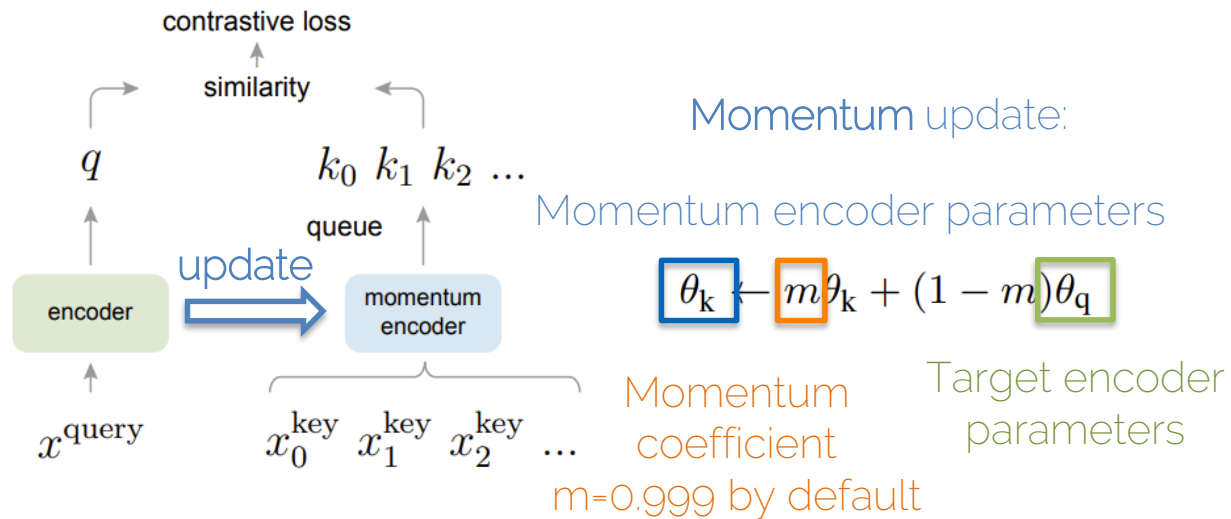
Can be thought of as building a dynamic dictionary



The momentum encoder is driven by a momentum update with the query encoder

MoCo

Can be thought of as building a dynamic dictionary

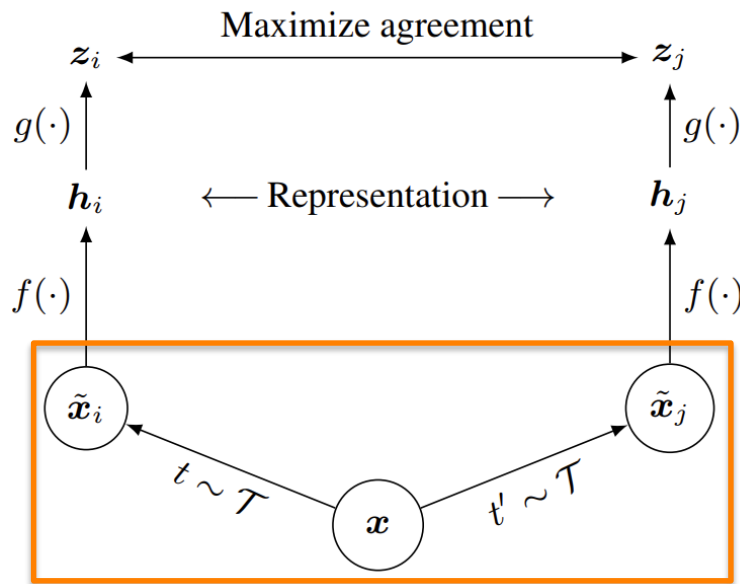


SimCLR

- A **S**imple Framework for **C**ontrastive **L**earning of Visual **R**epresentations
- Learns visual representations by maximizing agreement between differently augmented views of the same data samples
- Supervised via a contrastive loss in the latent space

SimCLR

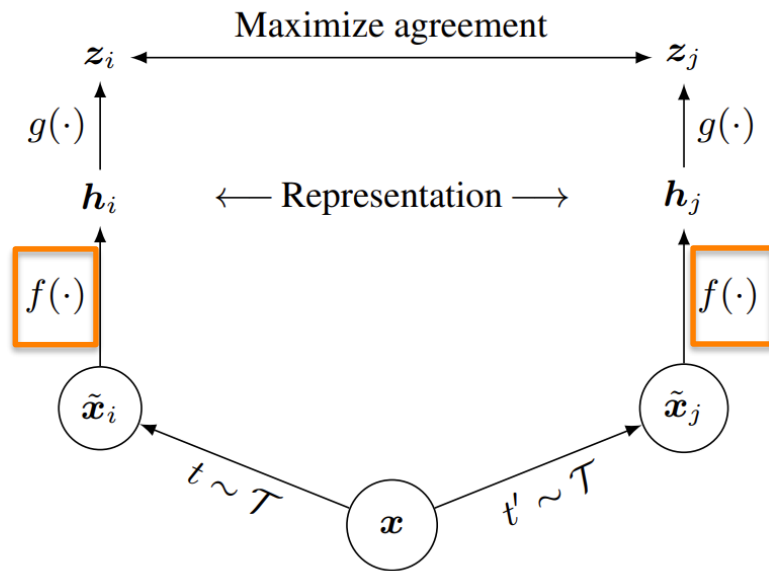
- A **Simple** Framework for **Contrastive** Learning of Visual **Representations**



One input sample is augmented to two views by two different operators from the same family, e.g., different rotations

SimCLR

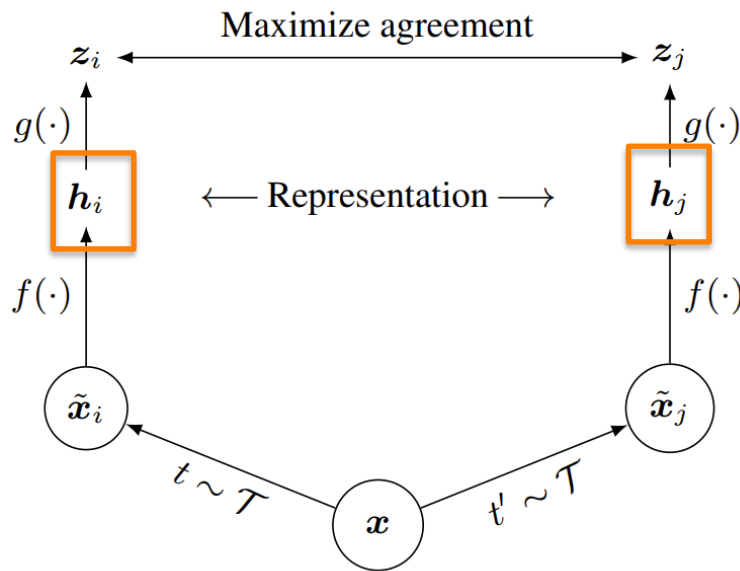
- A **Simple** Framework for **Contrastive** Learning of Visual **Representations**



Encoder network,
e.g., ResNet50

SimCLR

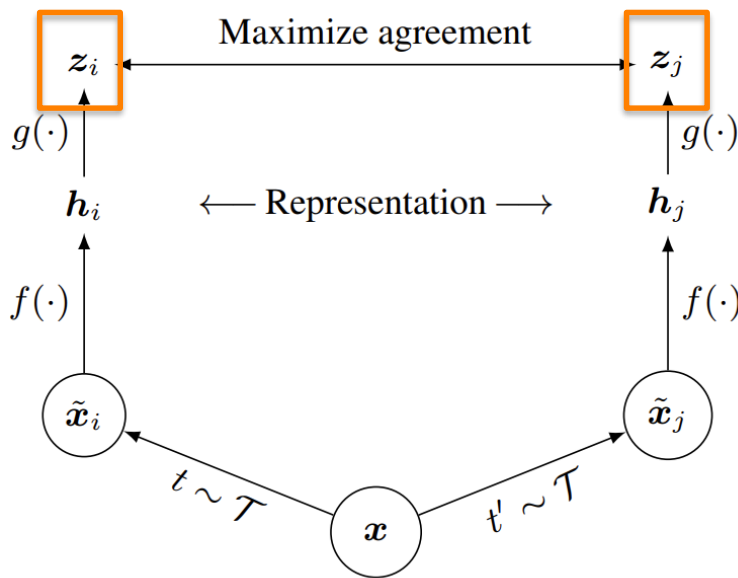
- A **Simple** Framework for **Contrastive** Learning of Visual **Representations**



Visual
representations

SimCLR

- A **S**imple Framework for **C**ontrastive Learning of **V**isual **R**epresentations

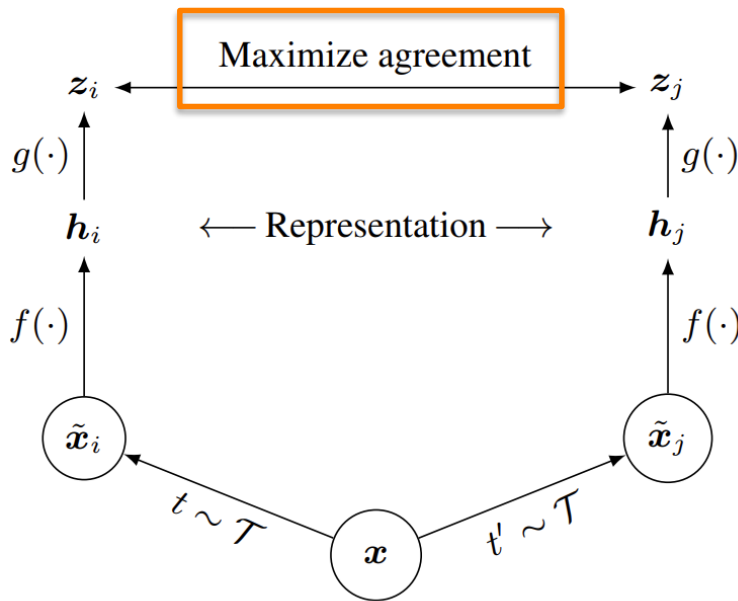


Projected
representations

SimCLR

- A **Simple** Framework for **Contrastive Learning** of Visual **Representations**

For a batch with N samples, $2N$ augmented samples are produced

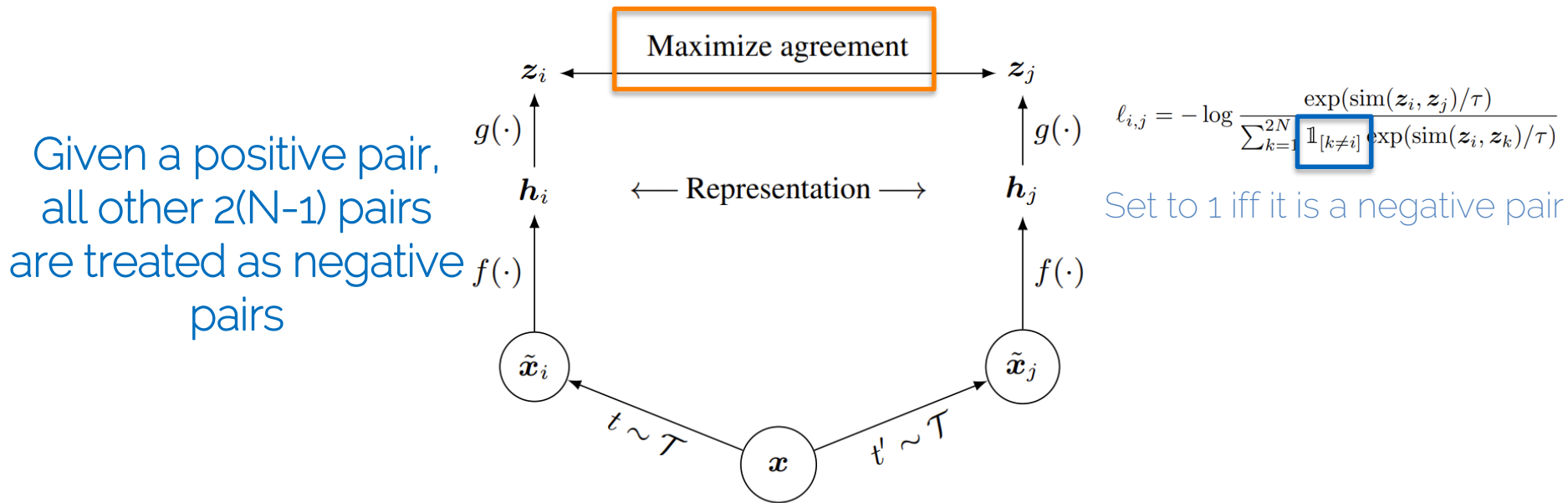


$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(z_i, z_k)/\tau)}$$

Normalized temperature-scaled cross entropy loss

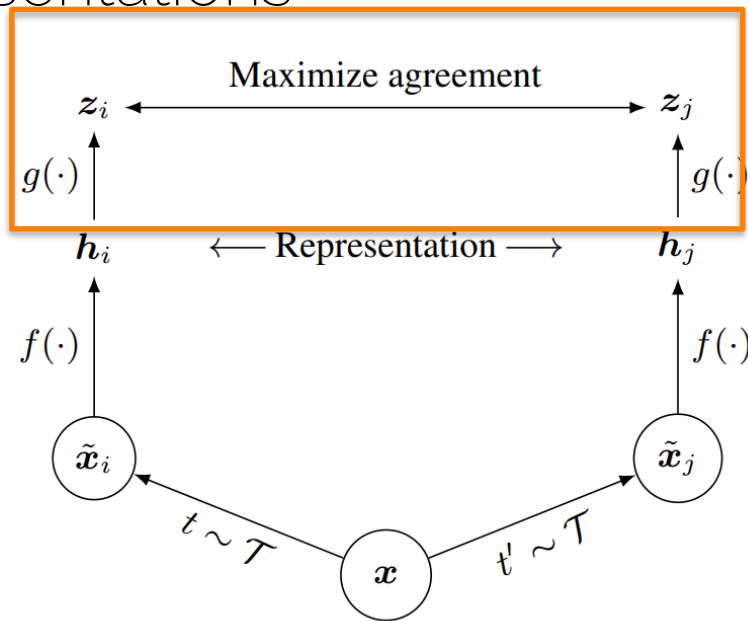
SimCLR

- A **Simple** Framework for **Contrastive** Learning of Visual **Representations**



SimCLR

- A **Simple** Framework for **Contrastive** Learning of Visual **Representations**



The projection head is removed for downstream tasks

SimCLR

- Key takeaways from SimCLR
 - Larger batch (4k or 8k) to provide more negative samples
 - Apply a MLP on the ResNet outputs to encode the final features during training, use the ResNet outputs directly during inference
 - Stronger data augmentations help

Augmentation is an Art



(a) Original



(b) Crop and resize



(c) Crop, resize (and flip)



(d) Color distort. (drop)



(e) Color distort. (jitter)



(f) Rotate $\{90^\circ, 180^\circ, 270^\circ\}$



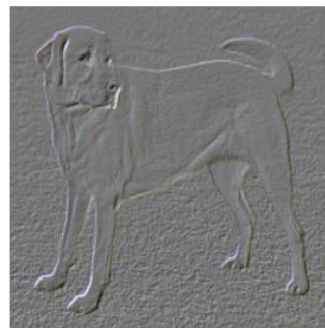
(g) Cutout



(h) Gaussian noise



(i) Gaussian blur



(j) Sobel filtering

Augmentations in SimCLR

Multi-Model Representation Learning

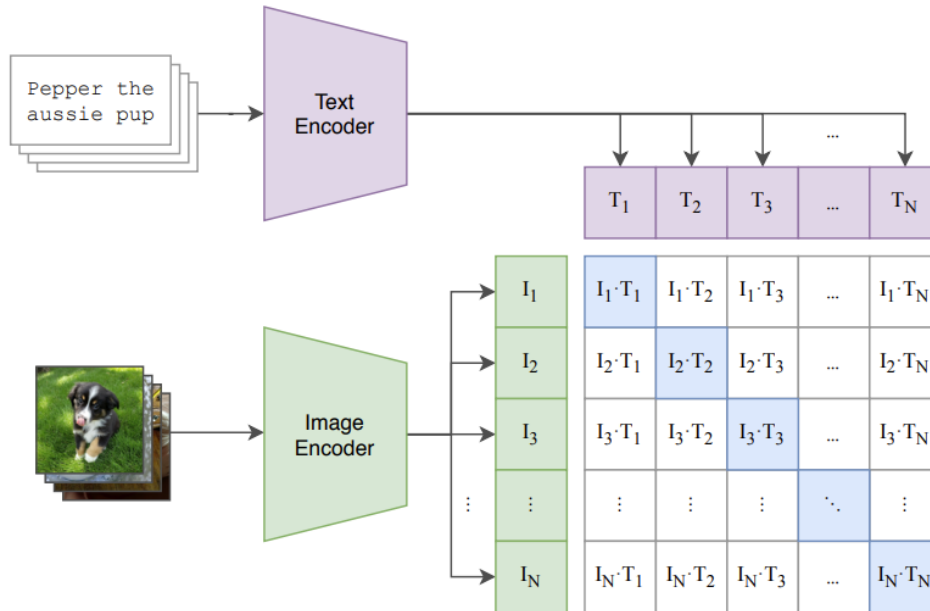
- Augmentations are key for contrastive learning: why not use matching samples from different modes?
- Image \leftrightarrow text is a prime example since there are millions of training pairs on the web

CLIP

- Contrastive Language-Image Pre-training
- Trained on a new dataset of 400 million image-text pairs
- Use a very large batch size of 32,678

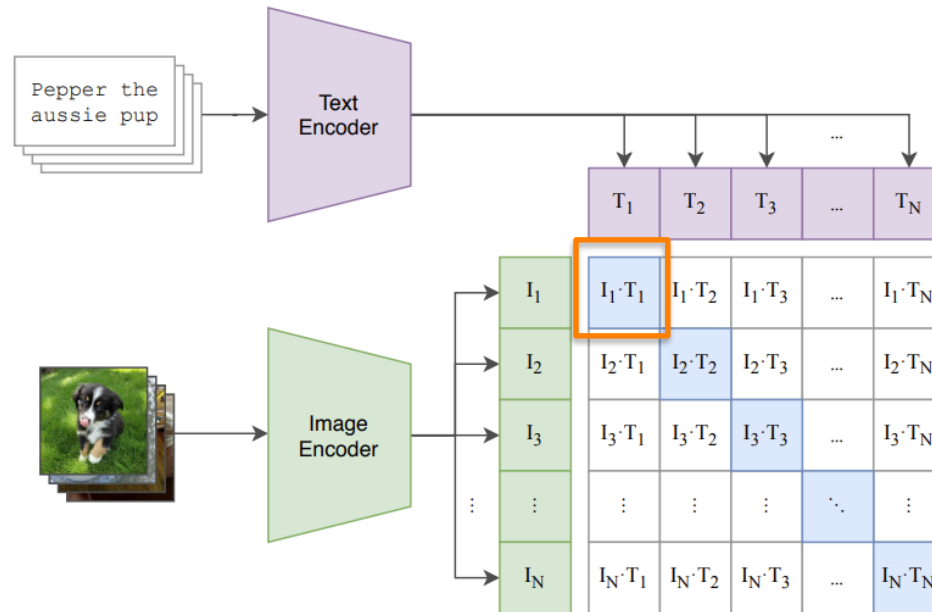
CLIP

- Contrastive Language-Image Pre-training
Contrastive pre-training



CLIP

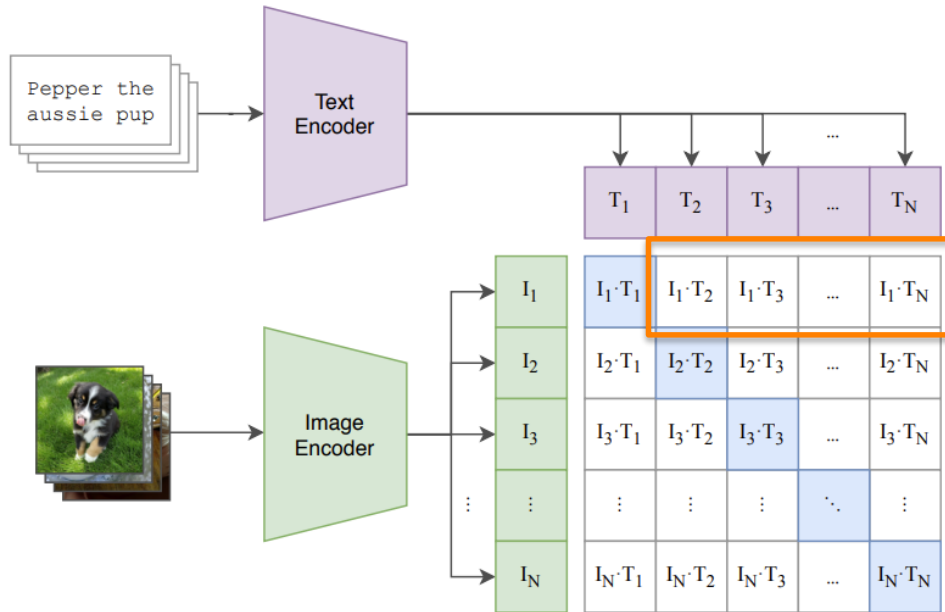
- Contrastive Language-Image Pre-training
Contrastive pre-training



Positive pairs

CLIP

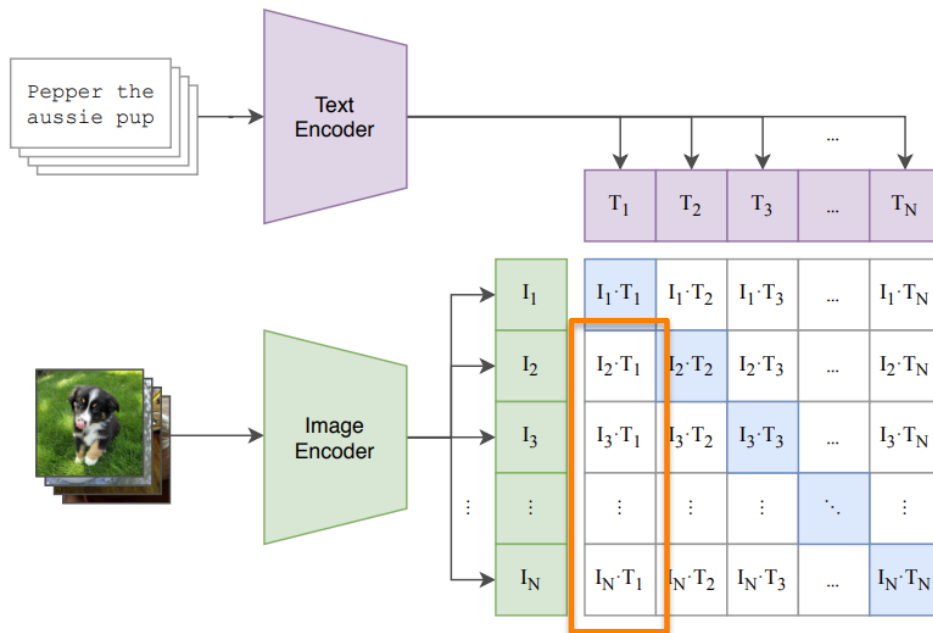
- Contrastive Language-Image Pre-training
Contrastive pre-training



Negative
image-text pairs
(image as anchor)

CLIP

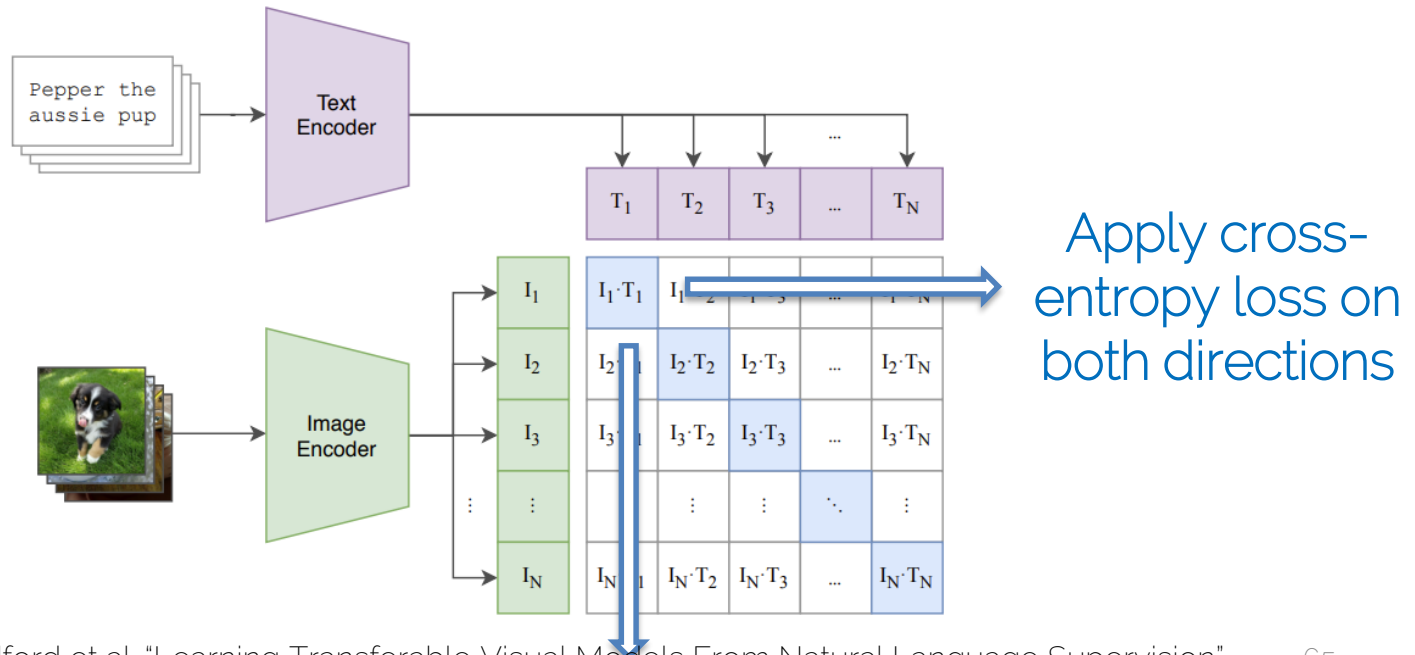
- Contrastive Language-Image Pre-training
Contrastive pre-training



Negative
text-image pairs
(text as anchor)

CLIP

- Contrastive Language-Image Pre-training
Contrastive pre-training



CLIP

- Contrastive Language-Image Pre-training

Contrastive pre-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)

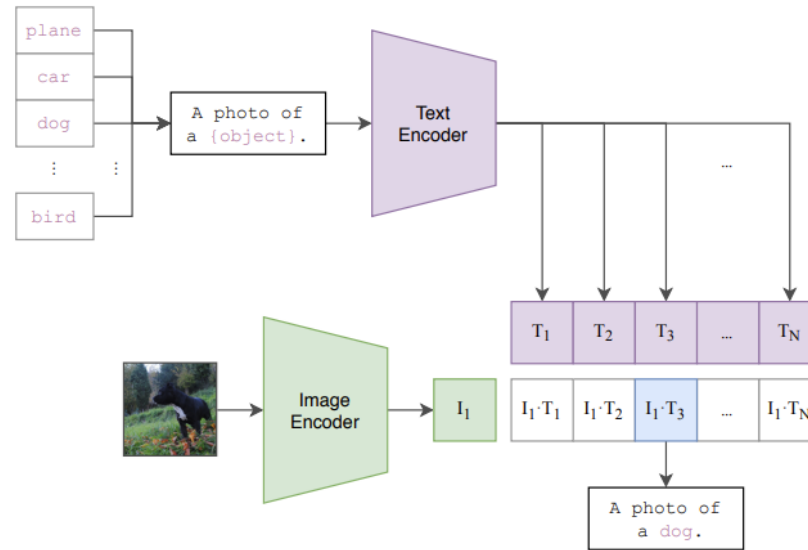
# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# 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
```

Simple
implementation

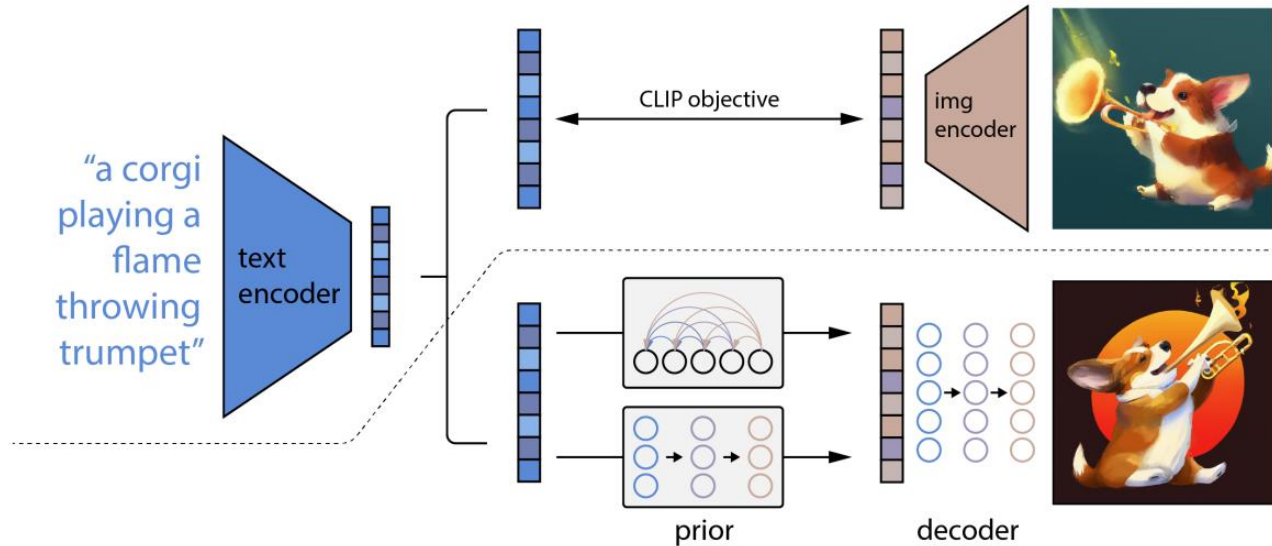
CLIP

- Contrastive Language-Image Pre-training
Inference: zero-shot classification



CLIP

- Contrastive Language-Image Pre-training
Use case: text-to-image generation (DALL-E 2)



CLIP

- Contrastive Language-Image Pre-training
Use case: text-to-image generation (DALL-E 2)

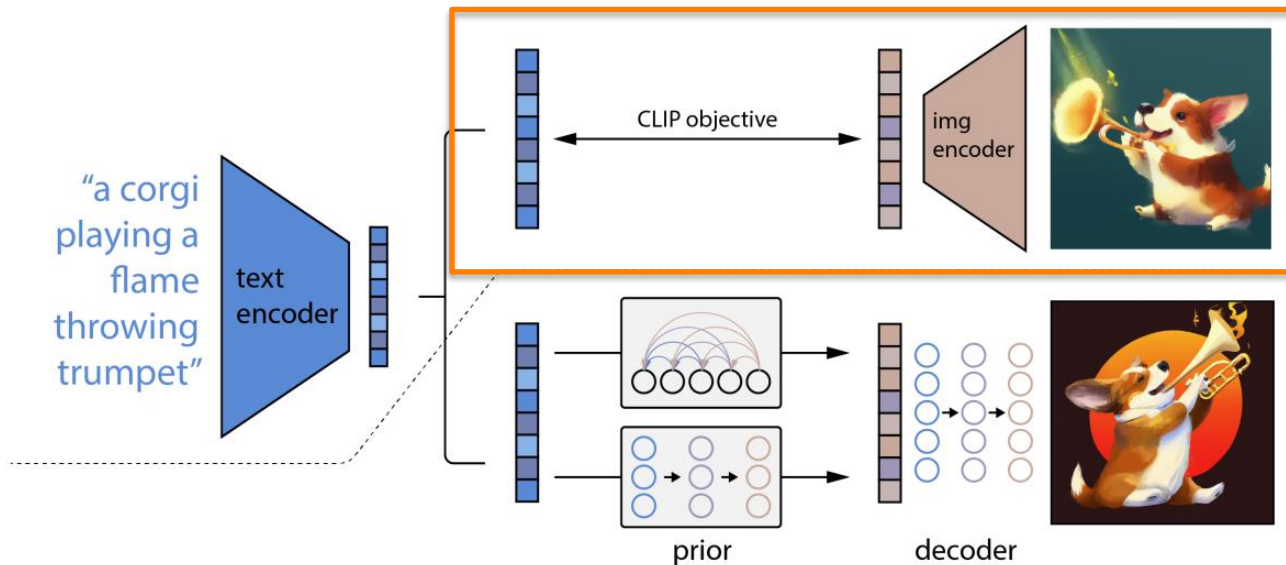
"a shiba inu wearing a beret
and black turtleneck"



CLIP

- Contrastive Language-Image Pre-training

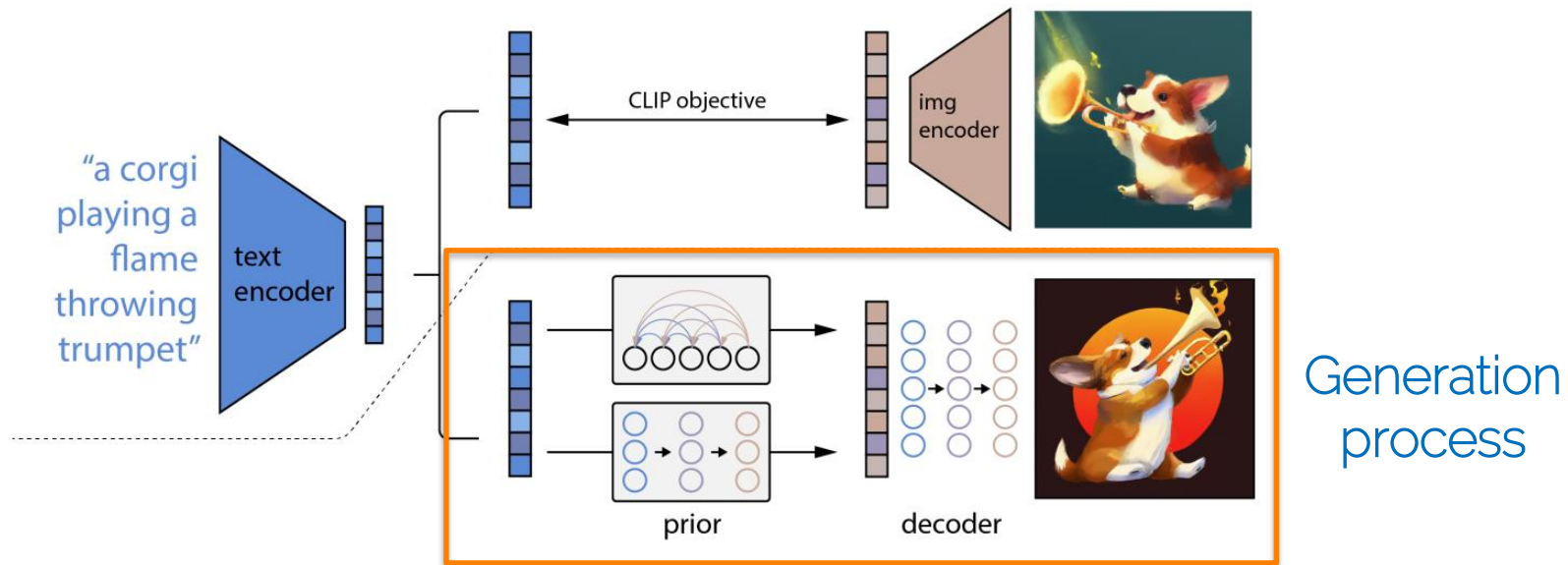
Use case: text-to-image generation (DALL-E 2)



CLIP
training
process

CLIP

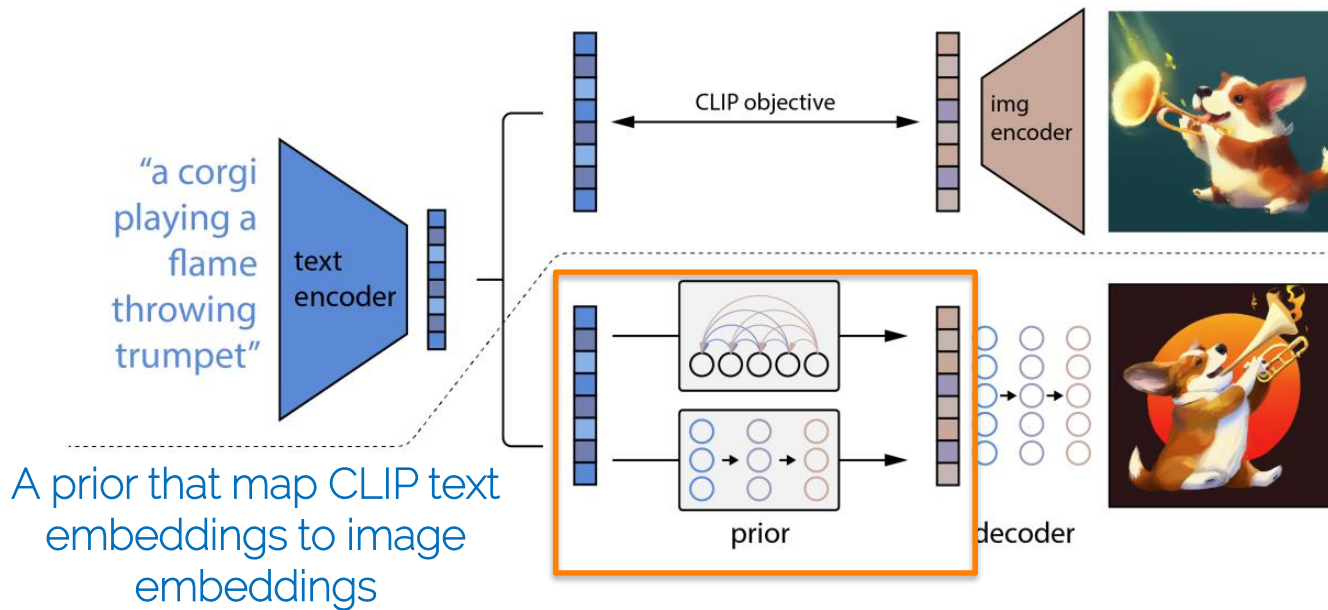
- Contrastive Language-Image Pre-training
Use case: text-to-image generation (DALL-E 2)



CLIP

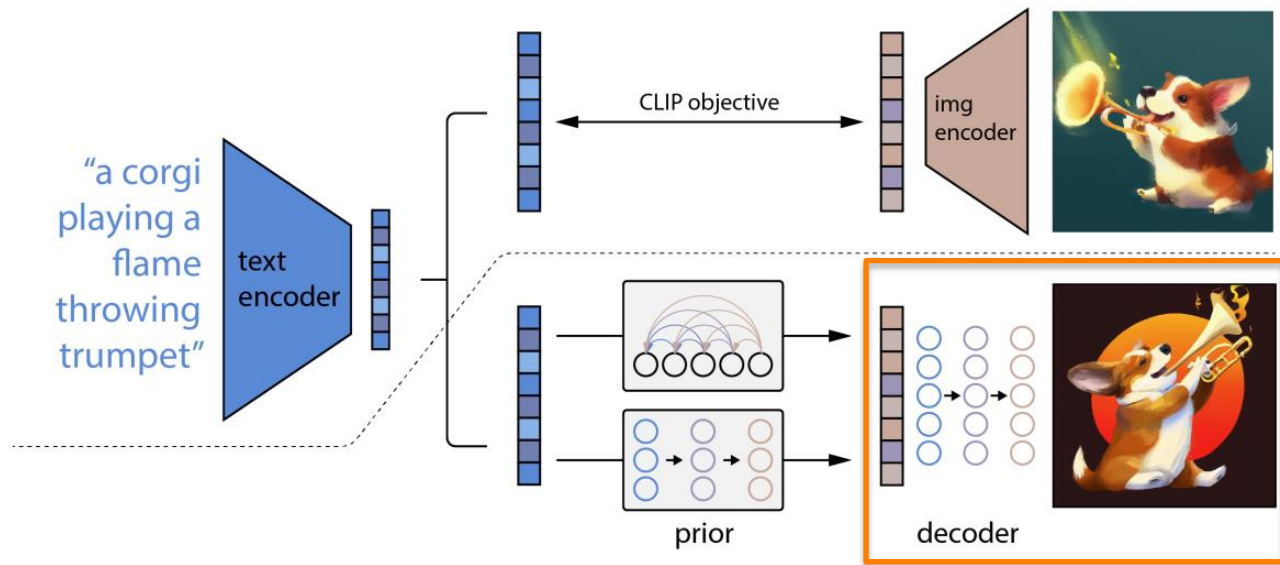
- Contrastive Language-Image Pre-training

Use case: text-to-image generation (DALL-E 2)



CLIP

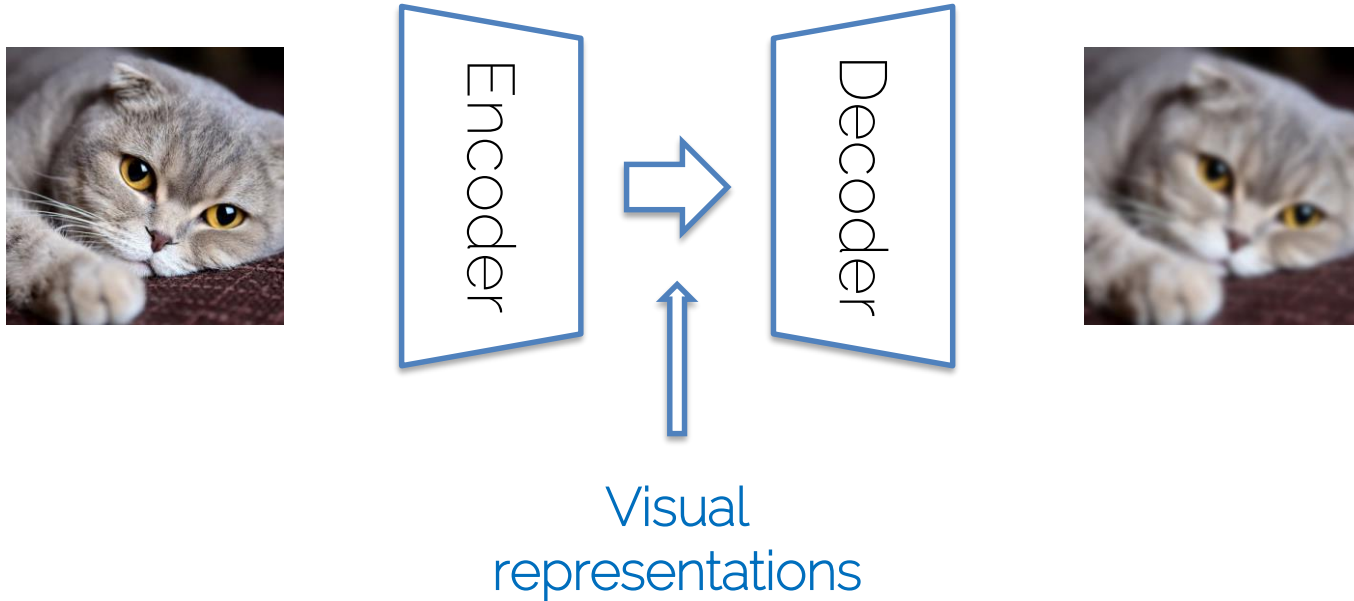
- Contrastive Language-Image Pre-training
Use case: text-to-image generation (DALL-E 2)



A decoder produces images conditioned on CLIP image embeddings

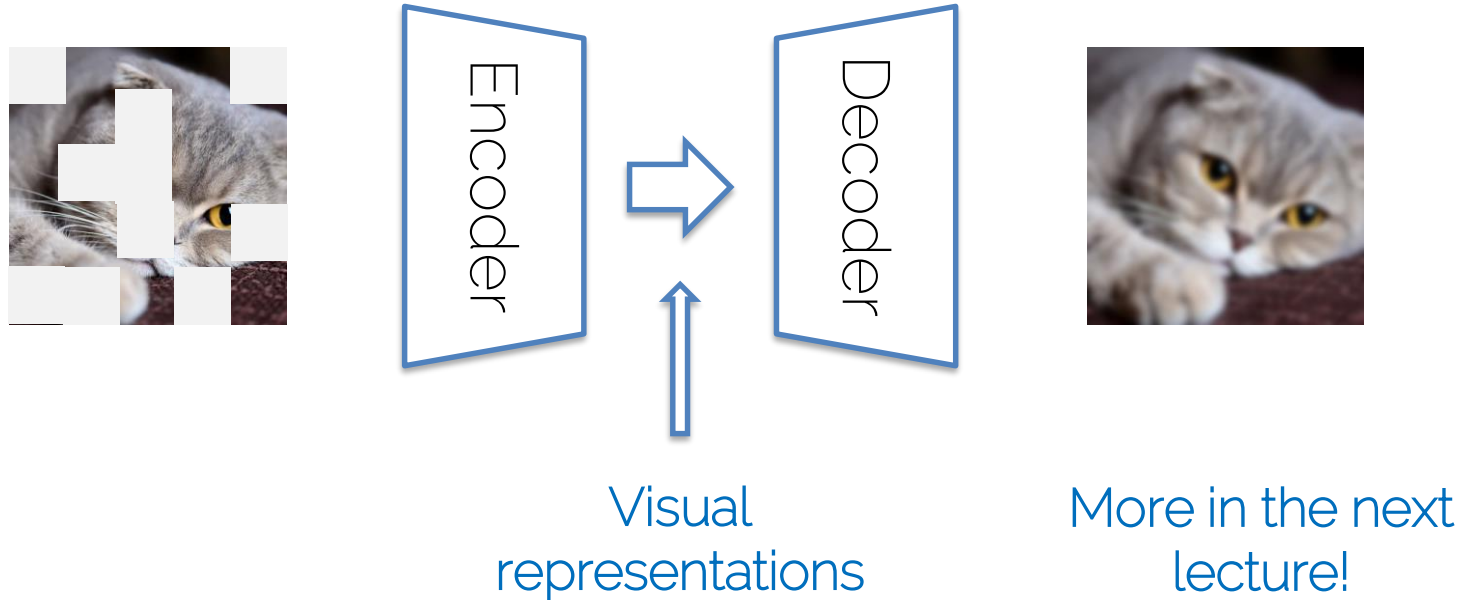
Other Self-supervised Approaches

- Autoencoder to Masked autoencoder



Other Self-supervised Approaches

- Autoencoder to Masked autoencoder



Representation Learning Caveats

- Lots of hyperparameters make difficult to asses what made improvements possible:
 - Better engineering vs method idea
- Long training cycles make things difficult to reproduce, in particular, for class projects
- Improvements can be small but require lots of effort to produce (training + hyperparam finding)

Reading Homework

- MoCo v2: [Chen et al. 2020] Improved Baselines with Momentum Contrastive Learning
 - <https://arxiv.org/pdf/2003.04297v1.pdf>
- MoCo v3: [Chen et al. 2020] An Empirical Study of Training Self-Supervised Vision Transformers
 - <https://arxiv.org/pdf/2104.02057v4.pdf>
- Masked autoencoder: [He et al. 2021] Masked Autoencoders Are Scalable Vision Learners
 - https://openaccess.thecvf.com/content/CVPR2022/papers/He_Masked_Autoencoders_Are_Scalable_Vision_Learners_CVPR_2022_paper.pdf

Literature

- DINO: [Caron. 2021] Emerging Properties in Self-Supervised Vision Transformers
 - <https://arxiv.org/pdf/2104.14294.pdf>
- MoCo: [He et al. 2019] Momentum Contrast for Unsupervised Visual Representation Learning
 - <https://arxiv.org/pdf/1911.05722.pdf>
- SimCLR: [Chen et al. 2020] A Simple Framework for Contrastive Learning of Visual Representations
 - <https://arxiv.org/pdf/2002.05709.pdf>
- CLIP: [Radford et al. 2021] Learning Transferable Visual Models From Natural Language Supervision
 - <https://arxiv.org/pdf/2103.00020.pdf>
- DALL-E 2: [Ramesh et al. 2022] Hierarchical Text-Conditional Image Generation with CLIP Latents
 - <https://arxiv.org/pdf/2204.06125.pdf>

Thanks for watching!