Deep Learning

Lecture 4

Representation Learning

Giovanni Chierchia

Table of contents

Introduction

- What is representation learning?
- What makes a good representation?
- How to evaluate a representation?

Domains vs Tasks

Domain

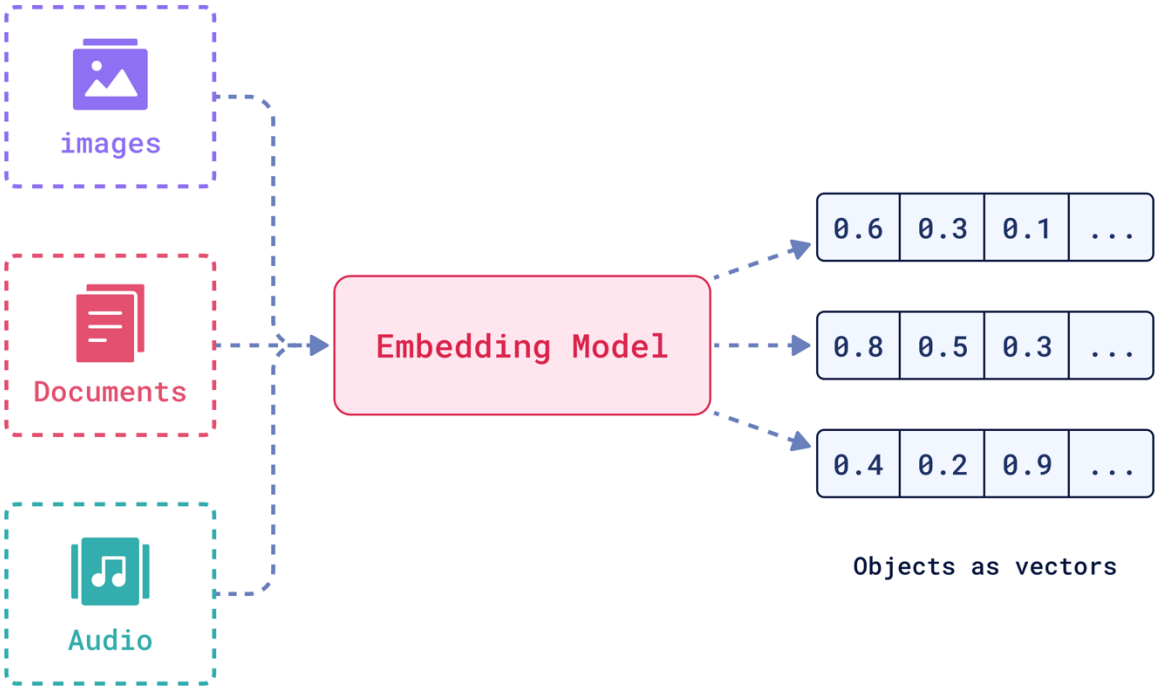
- The type of **data or field** that learning is applied to
- *Examples*
  - Images (Computer Vision)
  - Text (NLP)
  - Audio (Speech Recognition)
  - Structured Data
  - …

Task

- The specific **goal or problem** to solve within a domain
- *Examples*
  - Image Classification
  - Object Detection
  - Sentiment Analysis
  - Speaker Recognition
  - …

What is representation learning?

- **Representation Learning**
  - *Train a neural network to discover how to **transform** input data into features that are more adapted to some tasks of a domain*



Importance of representation learning

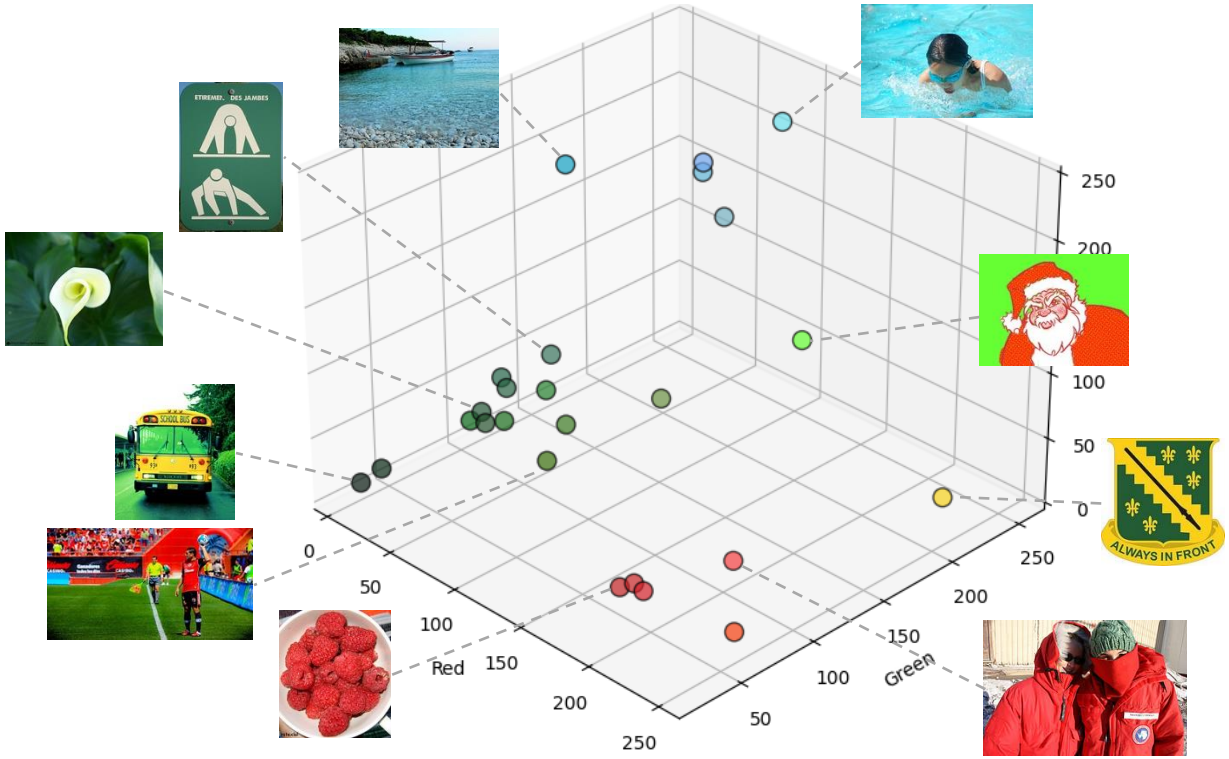
- **Goal ➜** Extract good numerical features from raw data

- **Motivation**
  - *Raw data is high-dimensional, noisy, redundant*
  - *Poor generalization due to overfitting*

- **Benefits**
  - *Learned features are "easier to process" in downstream tasks*
  - *Better generalization and task performance*
  - *Scalable solutions for large datasets*

Example of representation

- **Example ➜** Images represented by their dominant color
  - *Is this a good representation?*



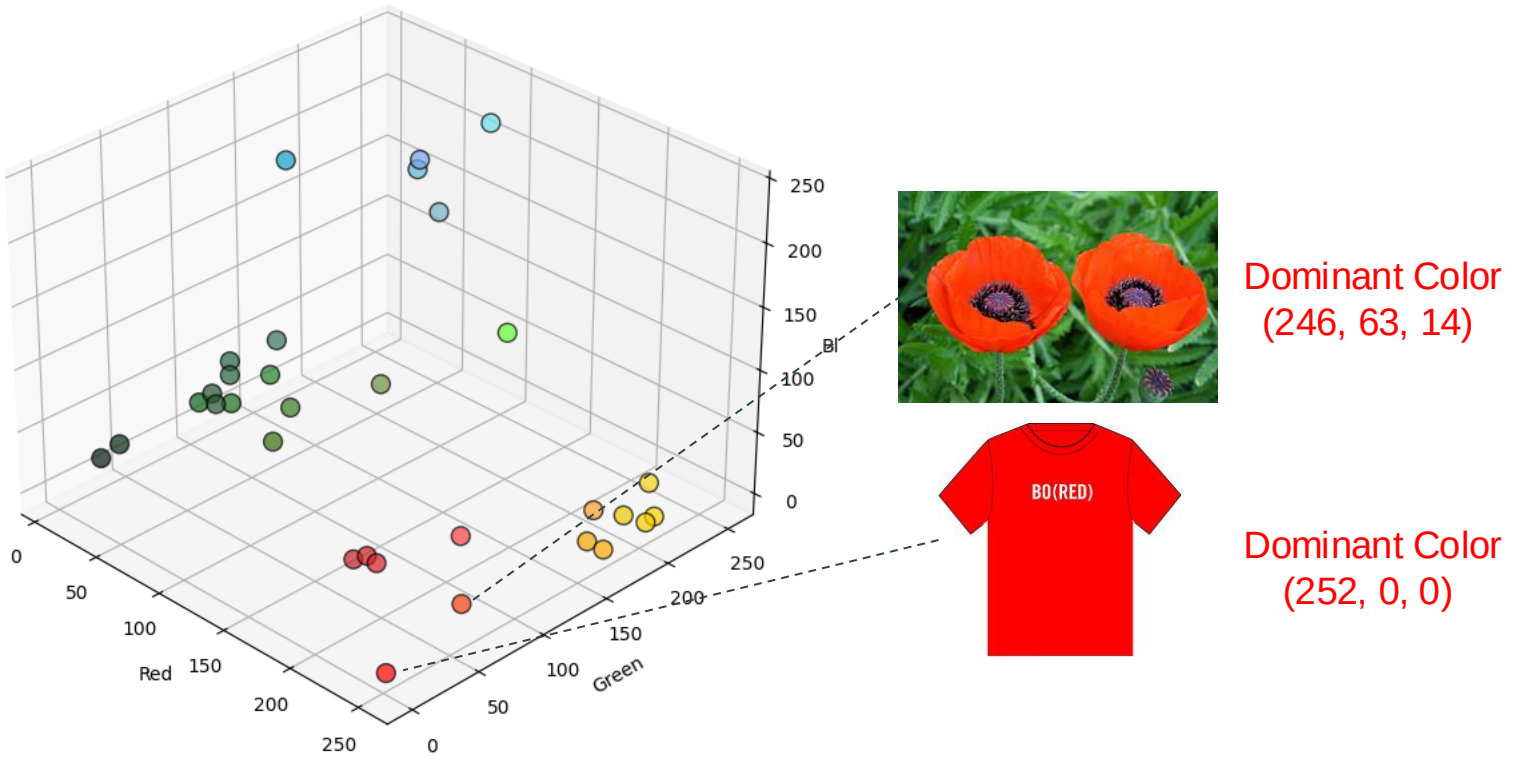
What makes a good representation?

- **Representation** ➜ Set of numerical values
  - *It should capture meaningful patterns in the original data point, leaving out irrelevant or redundant information*
  - *What is significant or irrelevant depends on the selected tasks*

- Effective representations…
  - *… are robust to small variations*
  - *… ignore irrelevant transformations*
  - *… group similar points together*
  - *… use as few dimensions as possible*
  - *… separate independent factors*

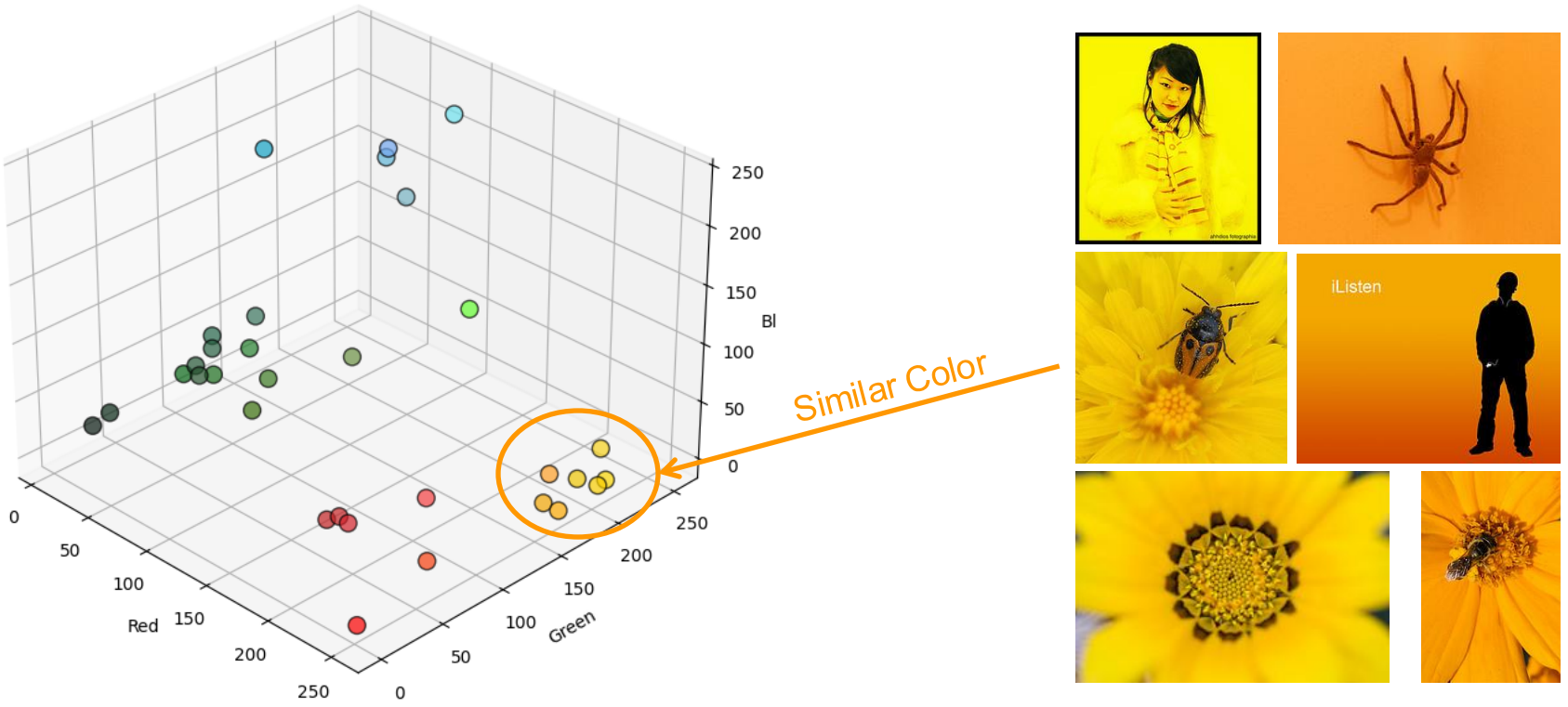
Properties (1/5)

- **Smoothness** ➜ A small change in the original data point should result in a small change in the representation



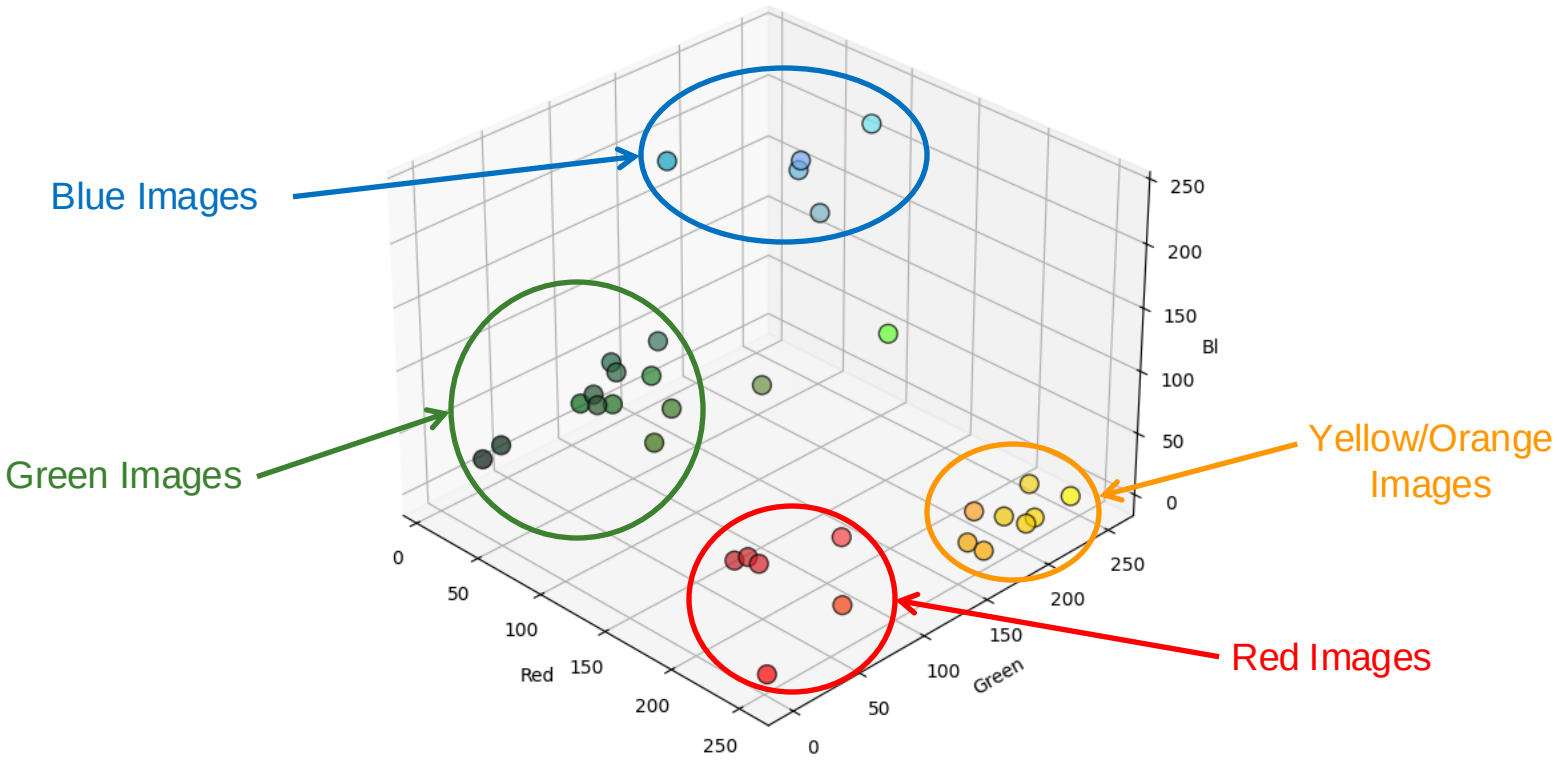
Properties (2/5)

- **Invariance ➜** Representations should remain consistent despite transformations irrelevant to the task



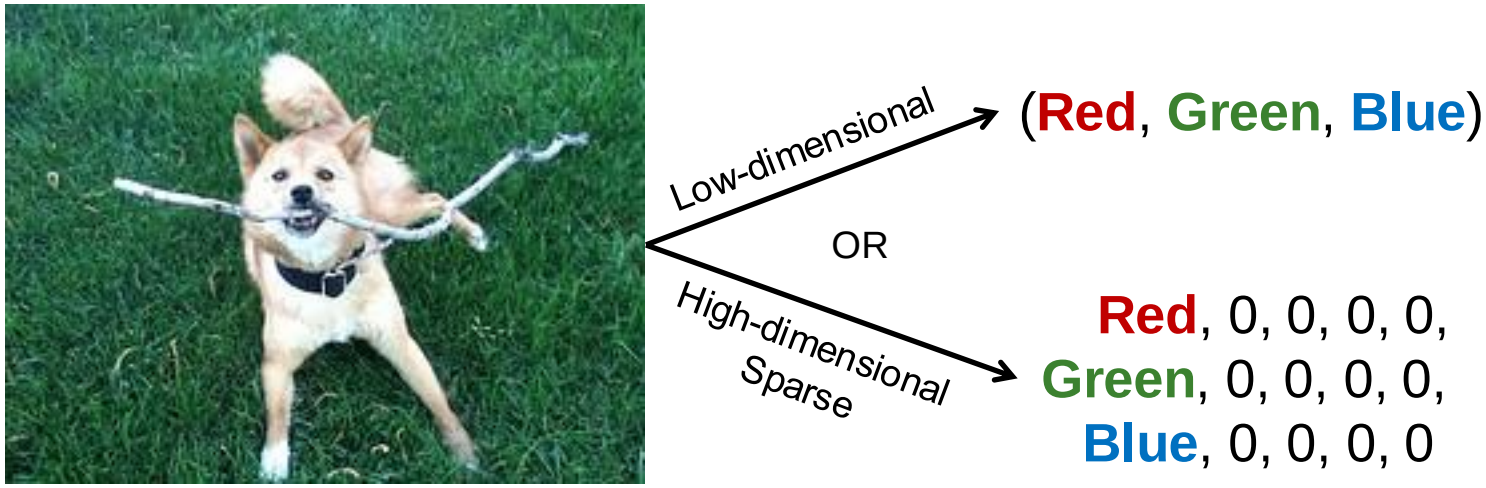
Properties (3/5)

- **Clustering** ➜ Similar data should be close in the representation space (and dissimilar data should be far apart)



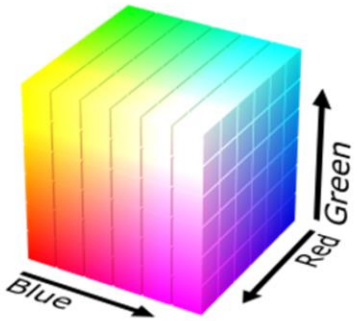
Properties (4/5)

- **Sparsity** ➜ Representations should use as few dimensions as possible

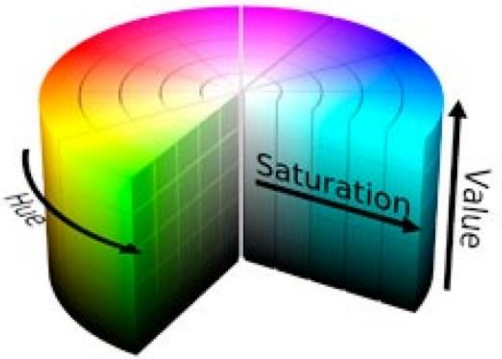


Properties (5/5)

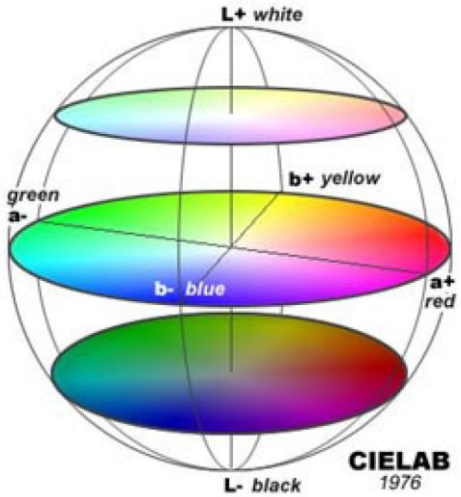
- **Disentanglement** ➜ Independent factors of variation should be clearly separated in the representation



RGB Space
Colors are decomposed into Red, Blue, Green



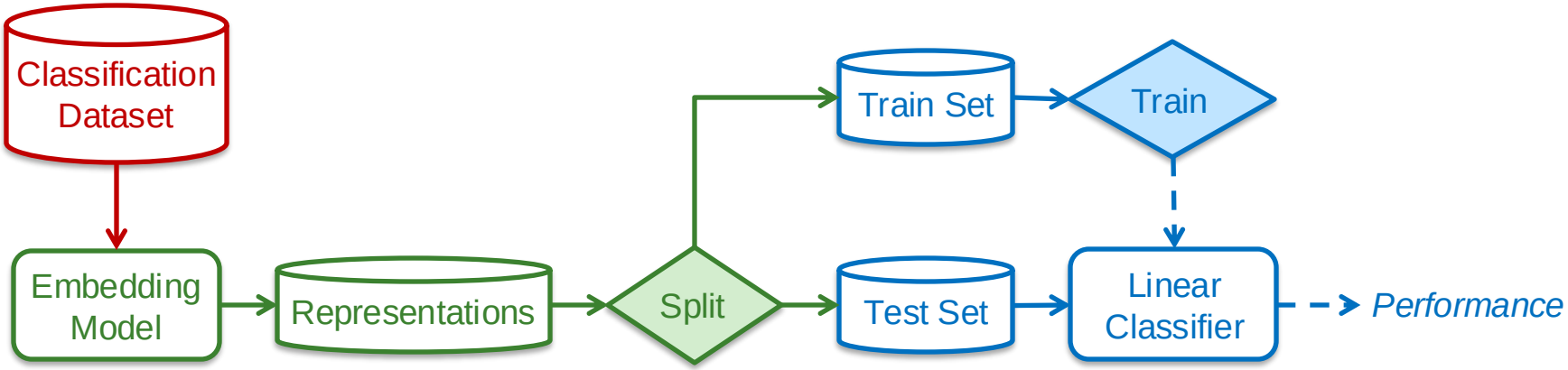
HSV Space
Colors are split into Hue, Saturation, Brightness



LAB Space
Colors are split into Black-White, Blue-Yellow, Red-Green

Linear probing

- How to **measure** the quality of a representation?
  - *Choose a classification dataset with **labels***
  - *Transform the input data with the embedding model*
  - *Fit a **simple classifier** on the training set*
  - *Evaluate the performance of the test set*



Summary

- **Representation learning** simplifies complex tasks by transforming raw data into meaningful sets of features

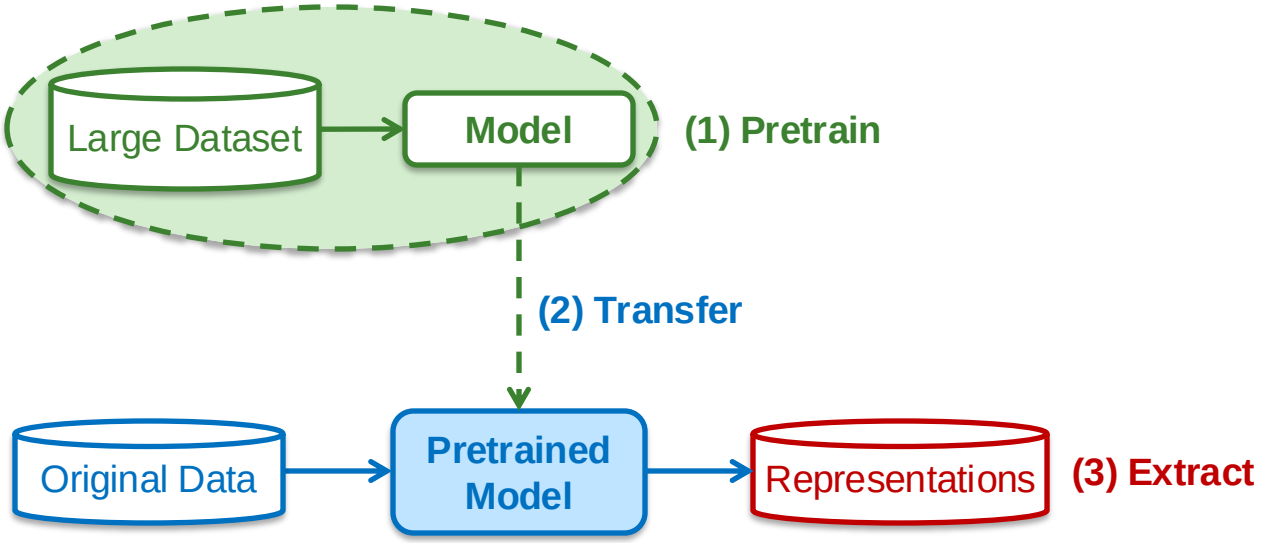
Smoothness	Small changes in the input data lead to small changes in the representations
Invariance	Representations maintain consistency under irrelevant transformations
Clustering	Similar data points are placed close to each other in the representation space
Sparsity	Representations use as few dimensions as possible
Disentanglement	Independent factors of variation are decomposed into different “axes” of the representation

Supervised Representation Learning

- Transfer learning
- Metric learning

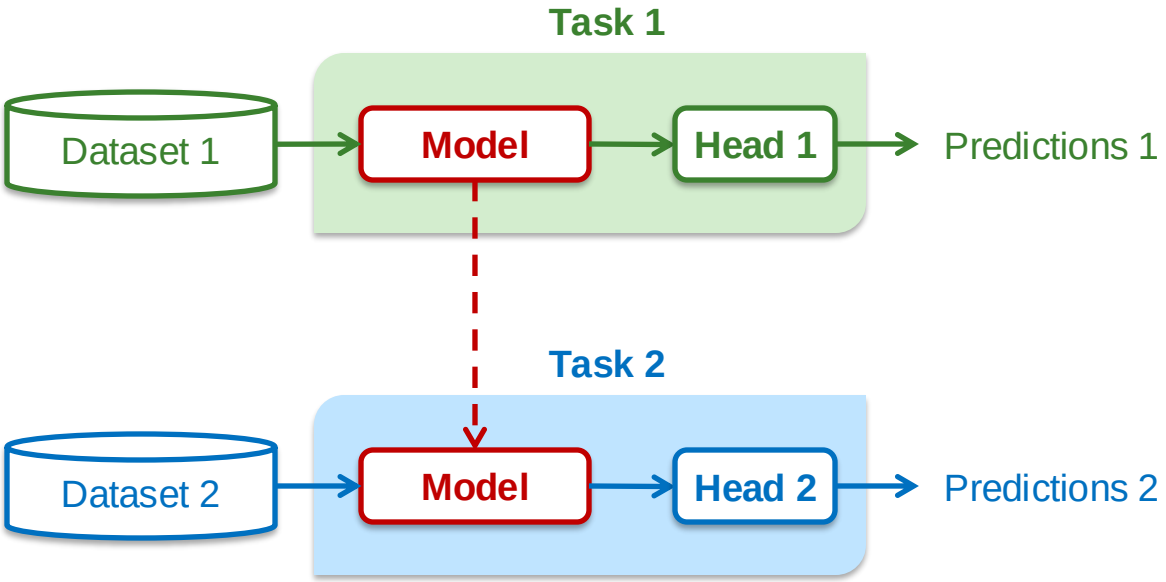
Introduction

- **Supervised representation learning**
  - ***Goal*** ➜ *Learn task-specific features in a supervised manner*
  - *Model is trained on a **supervised task** using real labelled data*
  - *Trained model is then reused to extract representations*



Transfer learning (1/2)

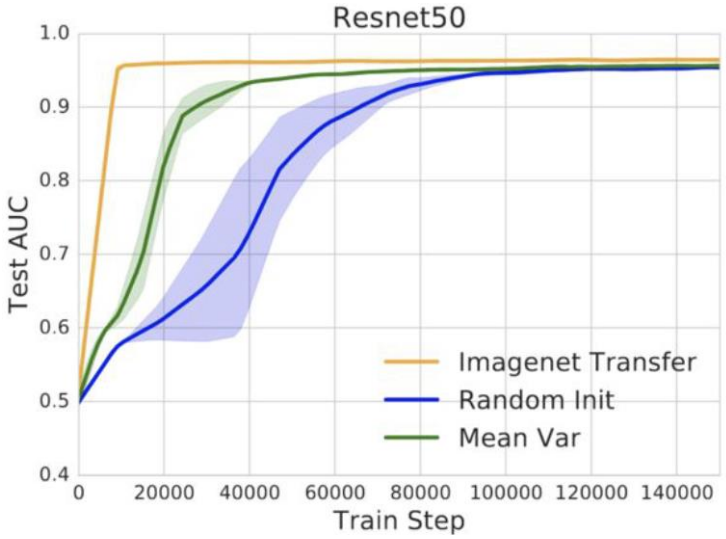
- **Transfer learning**
  - *Reuse trained model on different task to improve performance*
  - ***Option 1*** ➜ *Extract representations with pretrained model*
  - ***Option 2*** ➜ *Train new model starting from pretrained model*



Transfer learning (2/2)

- **Benefits**
  - *Faster training time*
  - *Reduce overfitting on small datasets*

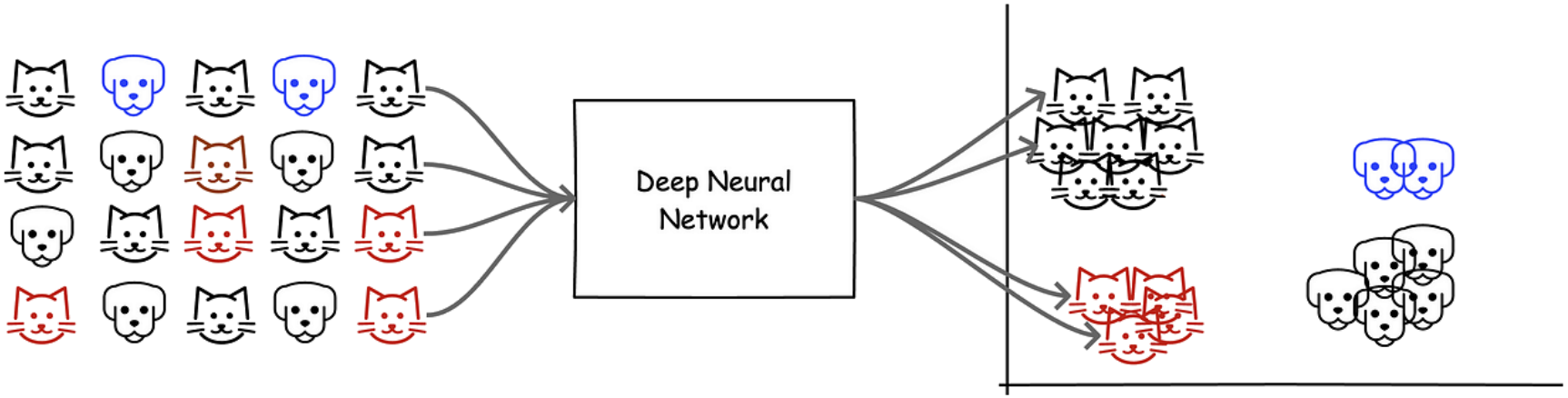
- **Limitations**
  - *Large, high-quality labeled datasets are costly to create*
  - *Representations may not transfer well to tasks very different from the training task*



- **Advise ➜** Use pretrained models whenever possible !!!

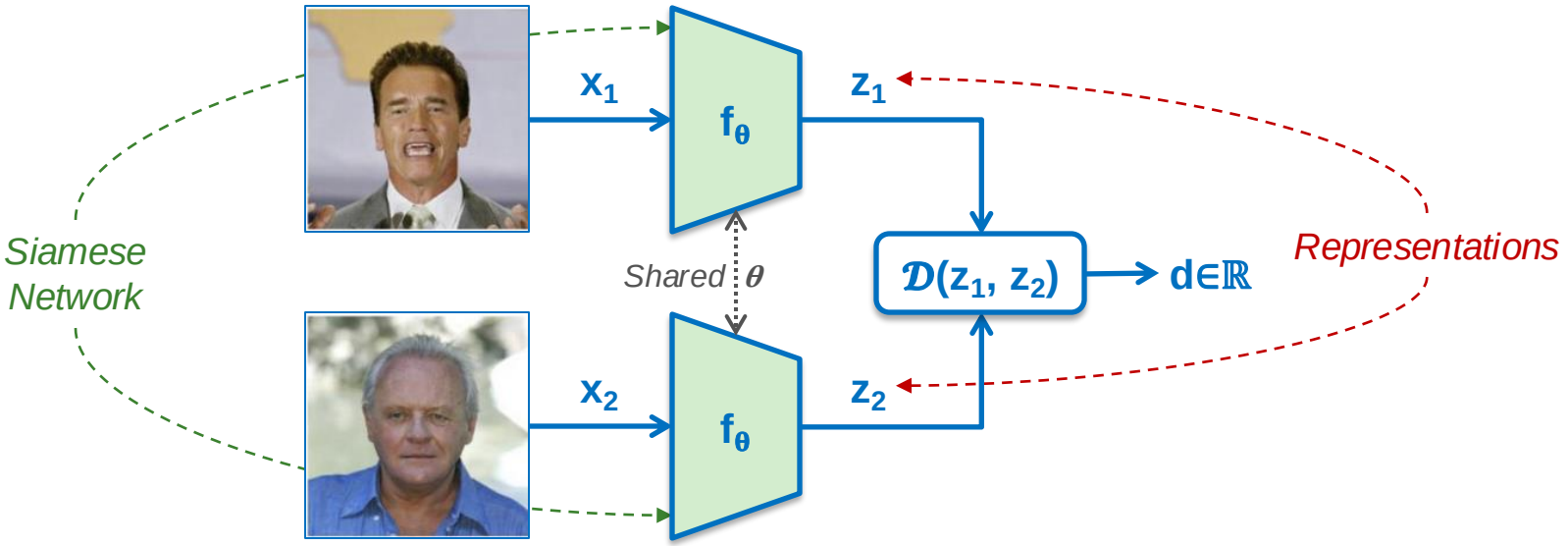
Metric learning

- **Metric learning**
  - *Learn representations that are comparable by distance*
  - ***Small distance*** ➜ *Original data points are similar*
  - ***Large distance*** ➜ *Original data points are dissimilar*
  - *The notion of similarity is specific to a task or domain*



Siamese network

- **Siamese network** *(Chopra et al., 2005)*
  - *Model that computes the representation of a data point*
  - *Trained to optimize a "distance function" between representations*
  - ***Distance function*** → *Euclidean distance, angular separation, …*



Training a Siamese network

- What it takes to train a Siamese network
  - ***Supervision*** ➜ *Set of data points and their corresponding labels*

$$\mathcal{S} = \{(\mathbf{x}_n, y_n) \in \mathcal{X} \times \mathcal{Y} \mid n = 1, \ldots, N\}$$

  - ***Network*** ➜ *Model to convert data points into representations*

$$f_\theta : \mathcal{X} \to \mathbb{R}^K$$

  - ***Metric*** ➜ *Function to measure distance between representations*

$$\mathcal{D} : \mathbb{R}^K \times \mathbb{R}^K \to \mathbb{R}$$

  - ***Loss function*** ➜ *See next slide…*

Loss function

- The **loss function** guides the Siamese network to learn a representation space where distances are consistent
  - ***Attraction*** → *Representations of similar data are pulled together*
  - ***Repulsion*** → *Representations of dissimilar data are pushed away*

- There exist two families of loss functions
  - ***Contrastive approaches***
    - *Minimize distance for similar pairs and maximize it for dissimilar ones*
    - *Pair loss, triplet loss, quadruplet loss, N-pair loss, structured loss*
  - ***Non-constrastive approaches***
    - *Optimize distance to "class centers" determined while training*
    - *Center loss, sphere loss, cosine loss, arc loss*

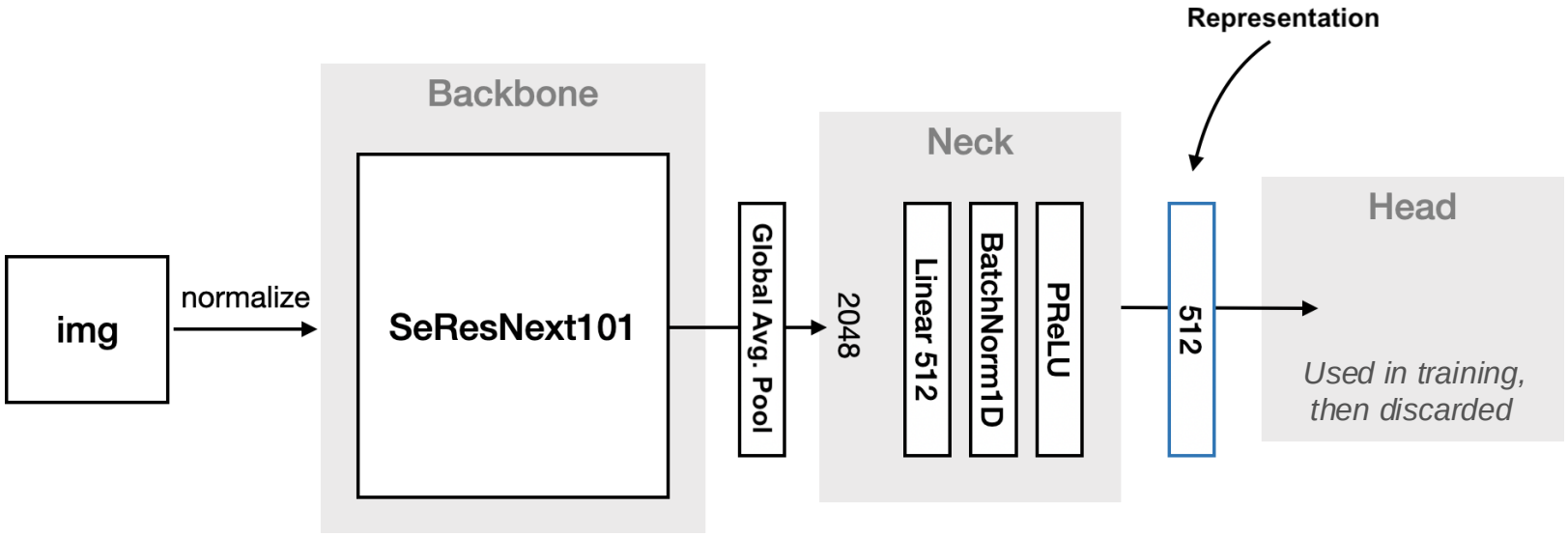
Applications in computer vision

- Any task that involves the **comparison** of two images is a possible application of metric learning in computer vision
  - ***Face Recognition*** ➜ *Decide if two images are of the same person*
  - ***Image Retrieval*** ➜ *Find similar images in a database*

Sketch-based image retrieval *(Bui et al., 2017)*

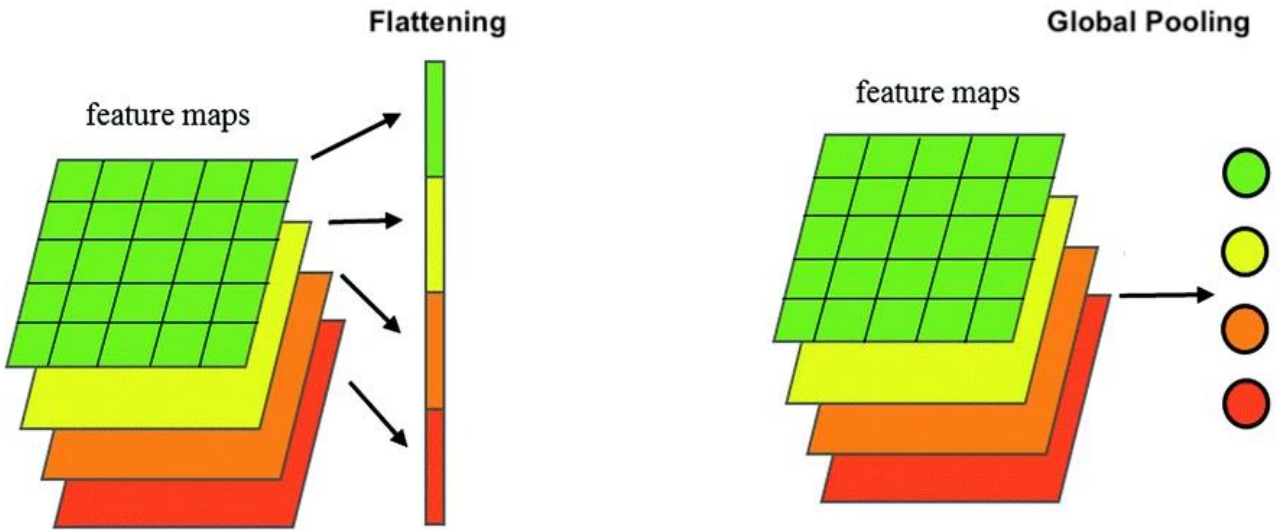
Architecture for computer vision

- Siamese network for **computer vision**
  - ***Backbone*** → *Mix of convolutional and max-pooling layers*
  - ***Flattening*** → *Global average pooling*
  - ***Neck*** → *Dense layer + Batch normalization + Parametric ReLU*
  - ***Head*** → *Only used in training by some loss functions*



Global average pooling

- **Global pooling** reduces dimensionality from 3D to 1D
  - *Pooling is computed with the largest window size*
  - ***Output*** ➜ *A scalar value for each input channel*
  - *Subsequent layers do not depend on the image size*



Summary

- **Why Metric Learning?**
  - *To capture semantic relationships between data*
  - *To produce discriminative representations for tasks like*
    - *Face recognition, Image retrieval, Anomaly detection*

- **Contrastive approach**
  - *Training requires triplets of similar and dissimilar data*
  - ***Usage*** ➜ *Weakly-supervised tasks*

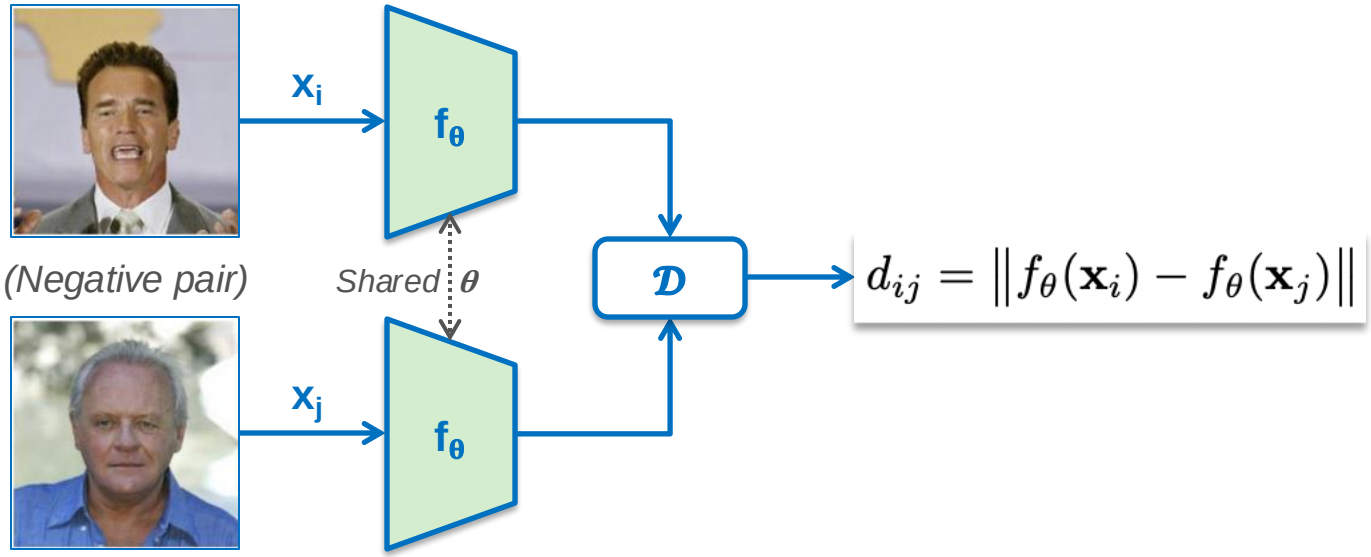
- **Non-contrastive approach**
  - *Training is like classification with some adjustments*
  - ***Usage*** ➜ *Supervised tasks with class imbalance*

Loss Functions for Metric Learning

- Contrastive
  - *Pair loss, Triplet loss*
- Non-contrastive
  - *Center loss, Cosine loss*

Pair loss (1/2)

- Training is performed on pairs of data
  - ***Positive pair*** ➜ *Inputs that are similar*
  - ***Negative pair*** ➜ *Inputs that are dissimilar*



Pair loss (2/2)

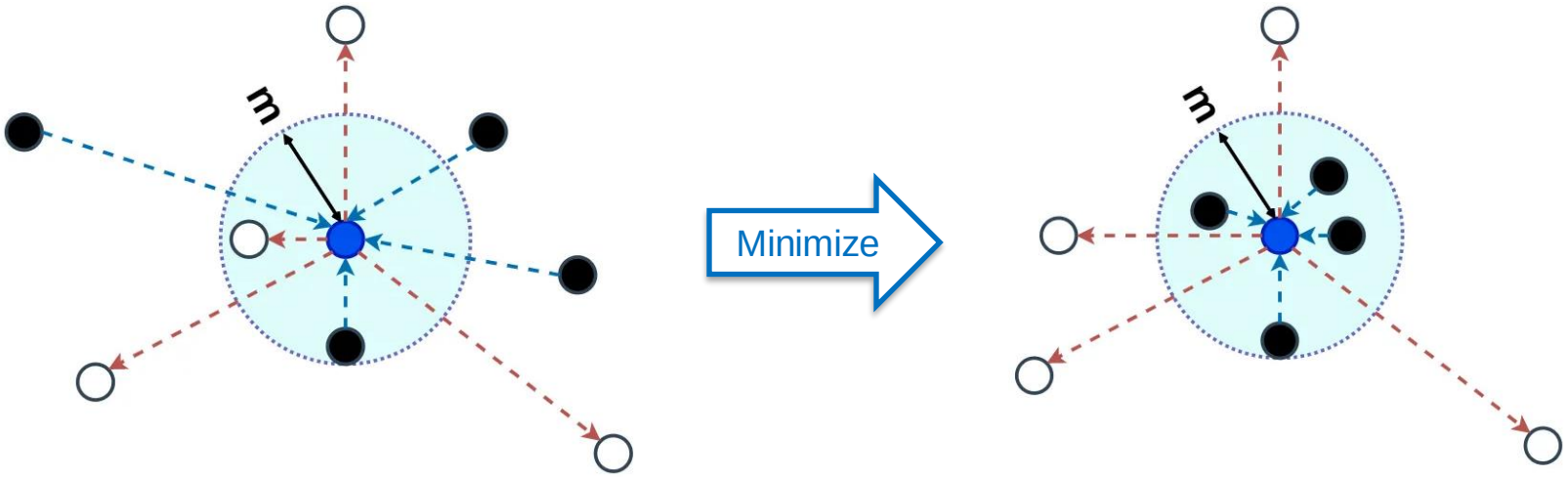
- The **pair loss** encourages the distance to be **small** for positive pairs and **large** for negative pairs

$$\mathcal{L}_{\text{pair}}(\theta) = \sum_{i,j \in \mathcal{P}} \left(y_{ij}\, d_{ij}^2 + (1 - y_{ij}) \max\{0, m - d_{ij}\}^2 \right)$$

1 for positive pairs

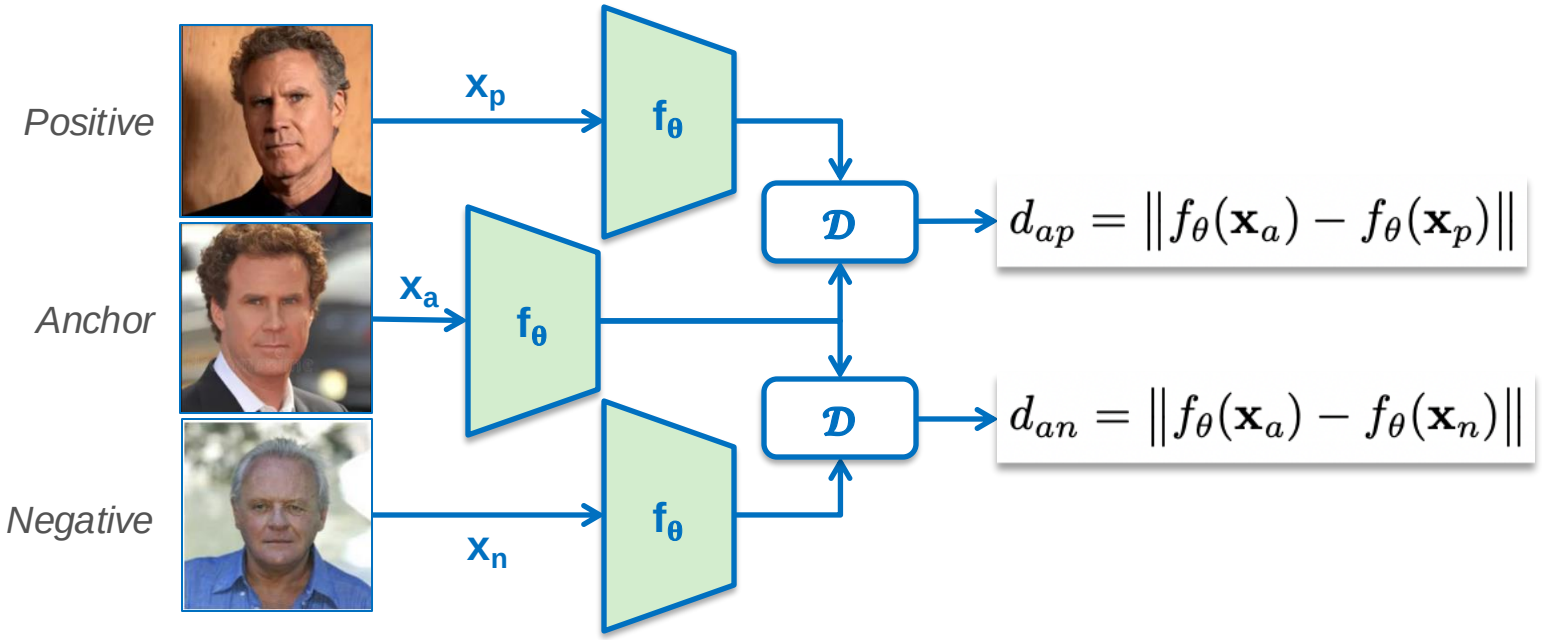
0 for negative pairs

Minimum distance



Triplet loss (1/2)

- Training is performed on triplets of data
  - ***Anchor*** ➜ *Some input*
  - ***Positive*** ➜ *A second input similar to anchor*
  - ***Negative*** ➜ *A third input dissimilar to anchor*

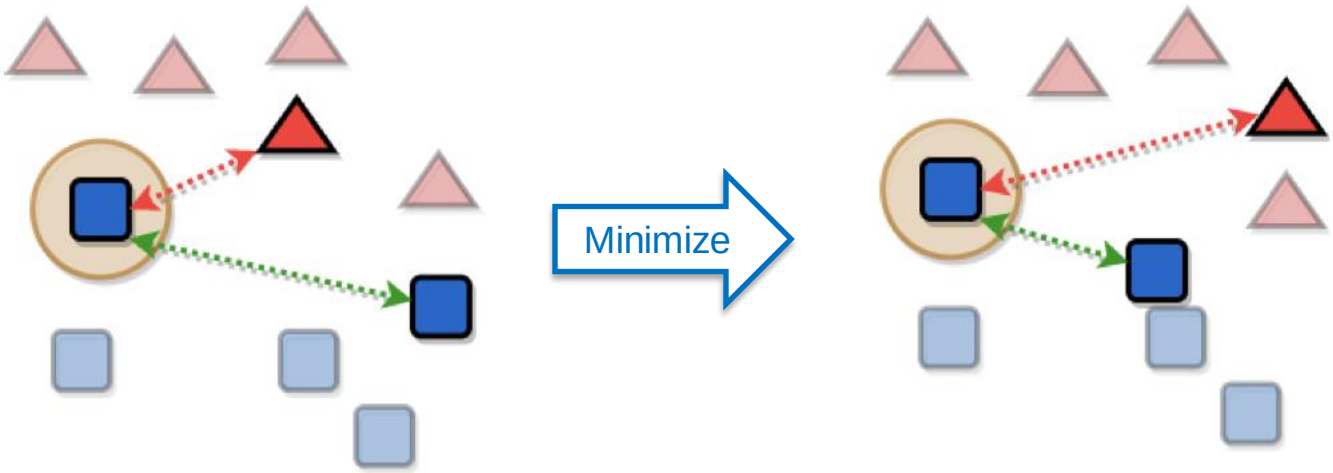


Triplet loss (2/2)

- The **triplet loss** makes the anchor-positive distance **smaller than** the anchor-negative distance **by a margin**

$$\mathcal{L}_{\text{triplet}}(\theta) = \sum_{a,p,n \in \mathcal{T}} \max\{0, d_{ap}^2 - d_{an}^2 + m\}$$

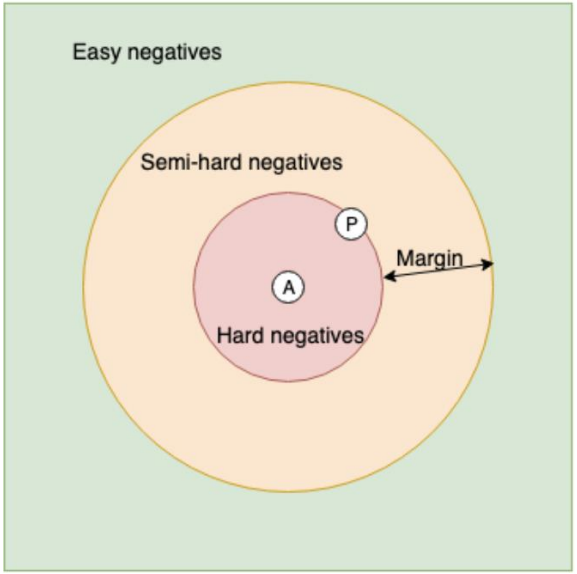
Minimum separation



Triplet mining (1/2)

- The choice of negative pairs is crucial for training
  - *This is easier to do with triplets!!!*

- **Types of triplets**
  - ***Easy*** ➜ $\mathfrak{D}_{an} > \mathfrak{D}_{ap} + m$
  - ***Hard*** ➜ $\mathfrak{D}_{an} < \mathfrak{D}_{ap}$
  - ***Semi*** ➜ $\mathfrak{D}_{ap} < \mathfrak{D}_{an} < \mathfrak{D}_{ap} + m$



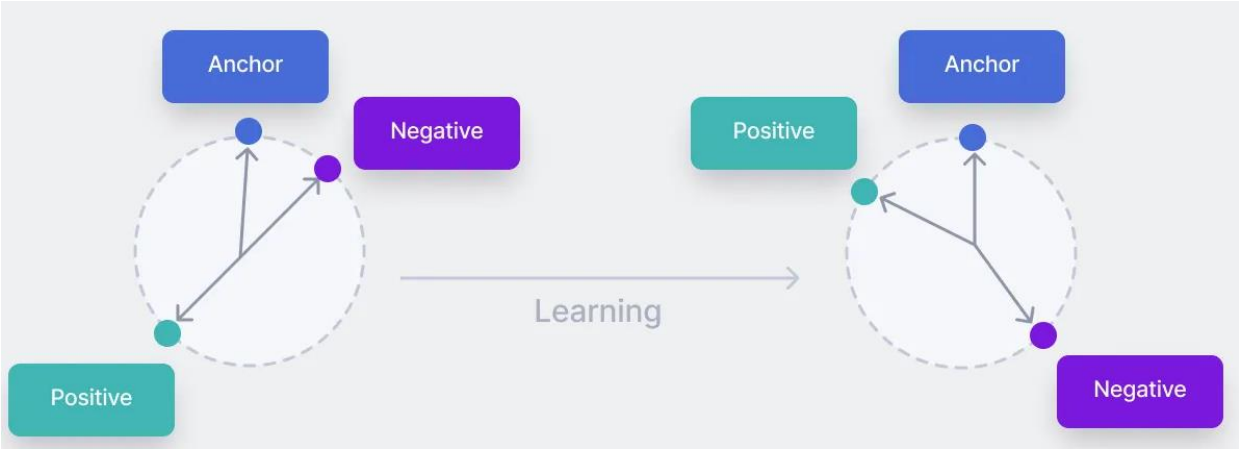
- Remark
  - *Easy triplets do not contribute to the loss*
  - *Training should focus on hard and semi-hard triplets*

Triplet mining (2/2)

- **Online triplet mining** *(Hermans et al., 2017)*
  - *Sample a batch of inputs and compute their representations*
    - *Select K classes randomly, then sample M data points per class*
    - *Batch size = K × M (larger batches are preferred)*
  - *For each sample of the batch, select the following triplet*
    - ***Anchor*** → *The sample itself*
    - ***Positive*** → *Similar point in batch at max distance from anchor*
    - ***Negative*** → *Dissimilar point in batch at min distance from anchor*
    - *(Alternatively, select all valid triplets except the easy ones)*
  - *Compute the triplet loss and update the Siamese network*
  - *Repeat until training is over*

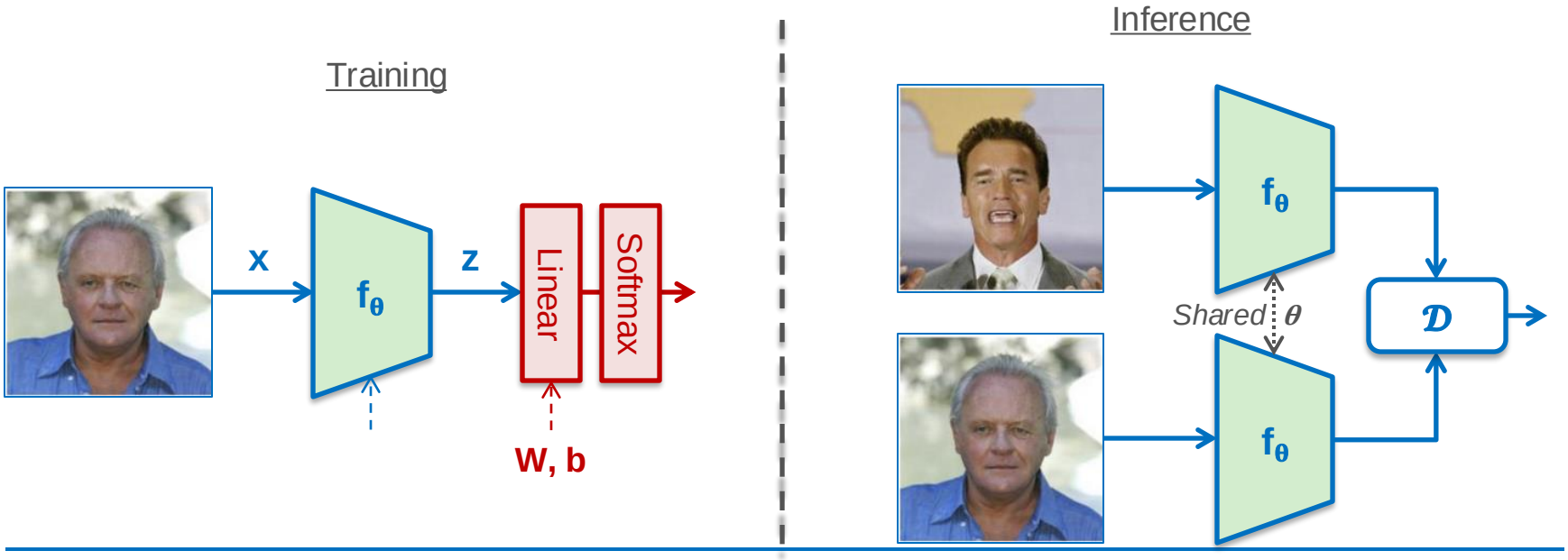
Moving away from triplet loss

- **Limitations of triplet loss**
  - Performance depends on the quality of mined triplets
    - *Increasingly likely to sample easy triplets as training progresses*
  - Triplets focus only on the “local” structure of representations
    - *Anchor is only guaranteed to be far from the selected negatives*
    - *Difficult to group all the positives into a common region of space*



Center loss (1/3)

- Training is performed like **classification**
  - *Inputs are batched as usual and sent to the network*
  - *A classification head is required* *(only in training)*
  - ***Head*** ➜ *Dense layer + Softmax*



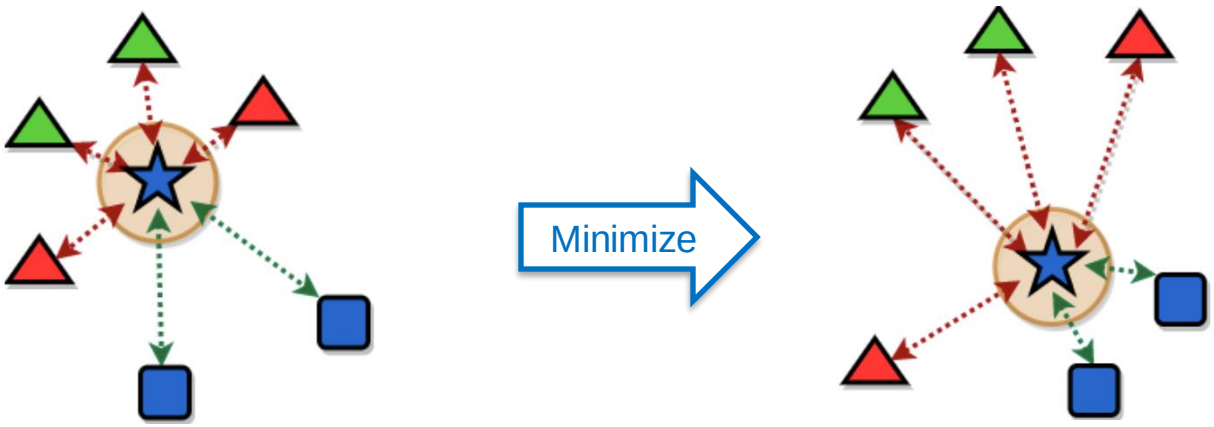
Center loss (2/3)

- The **center loss** separates representations by their class while forcing them to cluster around their **class centers**

$$\mathcal{L}_{\text{center}} = \mathcal{L}_{\text{CE}}(\theta, W, b) + \frac{\lambda}{2} \sum_{n=1}^{N} \| f_\theta(\mathbf{x}_n) - \mathbf{c}_{y_n} \|^2$$

Cross-Entropy

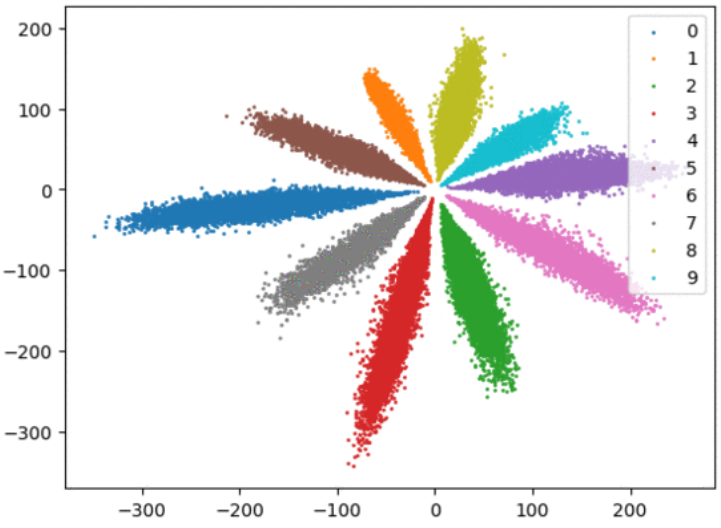
Center of class y_n



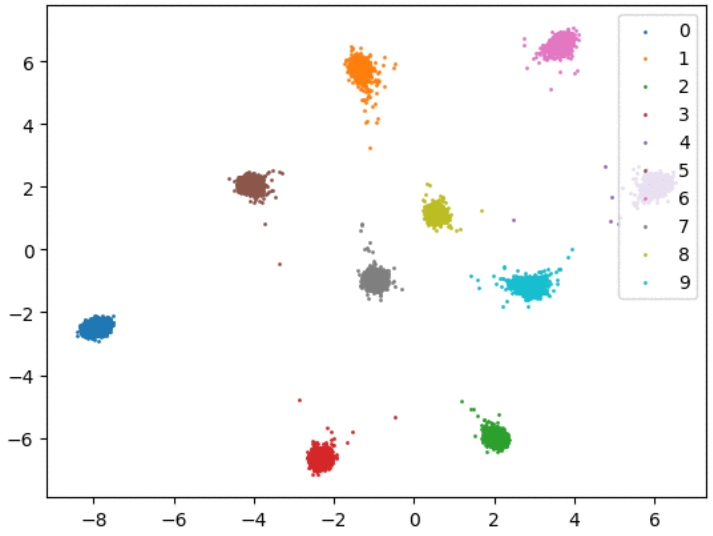
Center loss (3/3)

- This loss focuses on the **global** structure of representations
  - *"Cross-entropy" encourages separability between classes*
  - *"Distance to centers" enforces compactness within classes*
  - ***Drawback*** ➜ *No control over the separation of class centers*

Cross-entropy loss

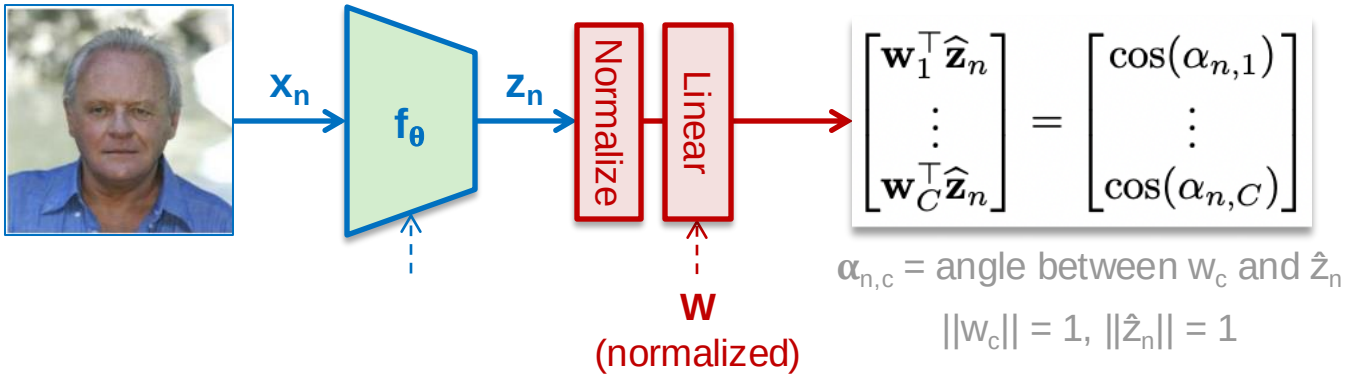


Center loss



Cosine loss (1/2)

- Training is performed like **classification**
  - *Inputs are batched as usual and sent to the network*
  - *A normalization head is required* *(only in training)*
  - ***Head*** ➜ *Normalize + Dense layer* *(no bias, weights normalized)*

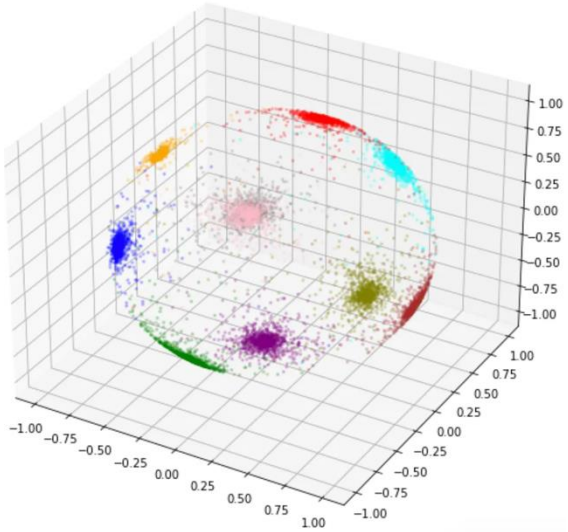


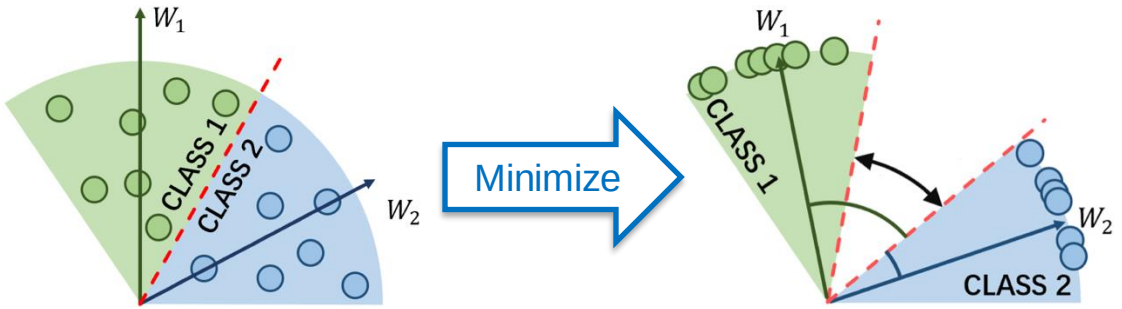
Cosine loss (2/2)

- The **cosine loss** projects representations on the sphere and increases the **angular distance** between classes

$$\mathcal{L}_{\cos}(\theta, W) = \sum_{n=1}^{N} -\log\left(\mathbf{y}_n^{\top}\,\text{softmax}(\mathbf{t}_n)\right) \dashrightarrow t_{n,c} = \begin{cases} s(\cos(\alpha_{n,c}) - m) & \text{if } c = y_n \\ s\cos(\alpha_{n,c}) & \text{if } c \neq y_n \end{cases}$$

One-hot encoding of class y_n — Scaling — Margin





Further improvements (1/2)

- Performance improves significantly with these tricks
  - ***Arc loss*** *(Deng, 2019)* ➜ *Put the margin "inside" the cosine*

$$\cos(\alpha_{n,c}) - m \quad \xrightarrow{\text{replace with}} \quad \cos\left(\alpha_{n,c} + m\right)$$

  - ***Scaling*** *(Zhang, 2019)* ➜ $s \approx \sqrt{2} \cdot \log\left(C - 1\right)$ *(**C** = number of classes)*

  - ***Sub-centers*** *(Deng, 2020)* ➜ *Allow multiple centers per class*

$$\tilde{\alpha}_{n,c} = \arccos\left(\max_k \mathbf{w}_c^{(k)\top} \widehat{\mathbf{z}}_n\right)$$

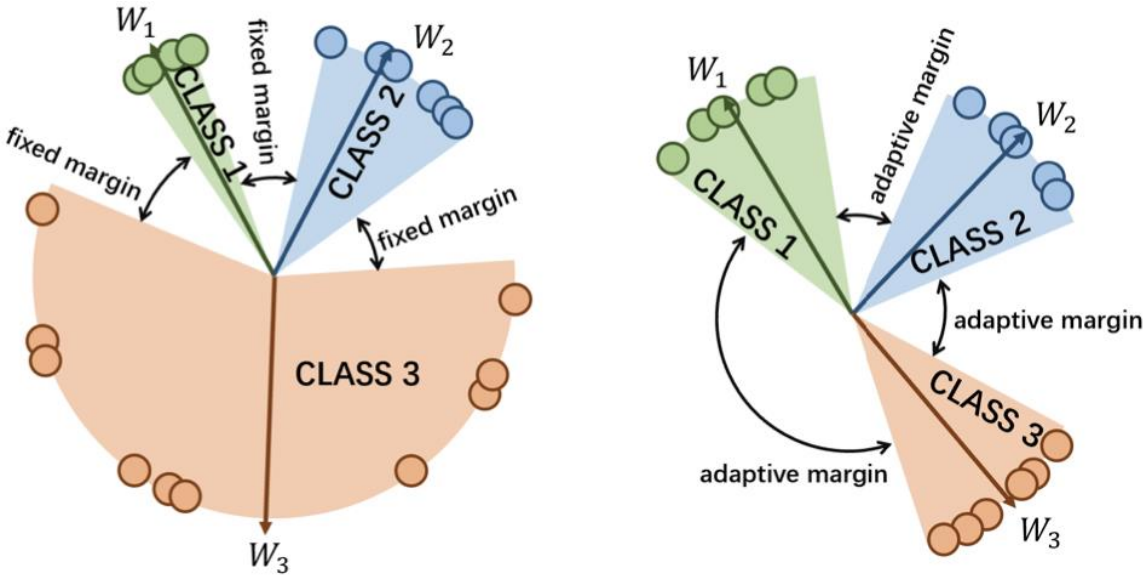
- *Source: https://hav4ik.github.io/articles/deep-metric-learning-survey/*

Further improvements (2/2)

- … *(continued from previous slide)*
  - ***Adaptive margins*** *(Ha, 2020)*
    - *Use a different margin per class when the dataset is imbalanced*

$$m_c = aN_c^{-\lambda} + b$$

(N_c = samples of class 'c')



Summary

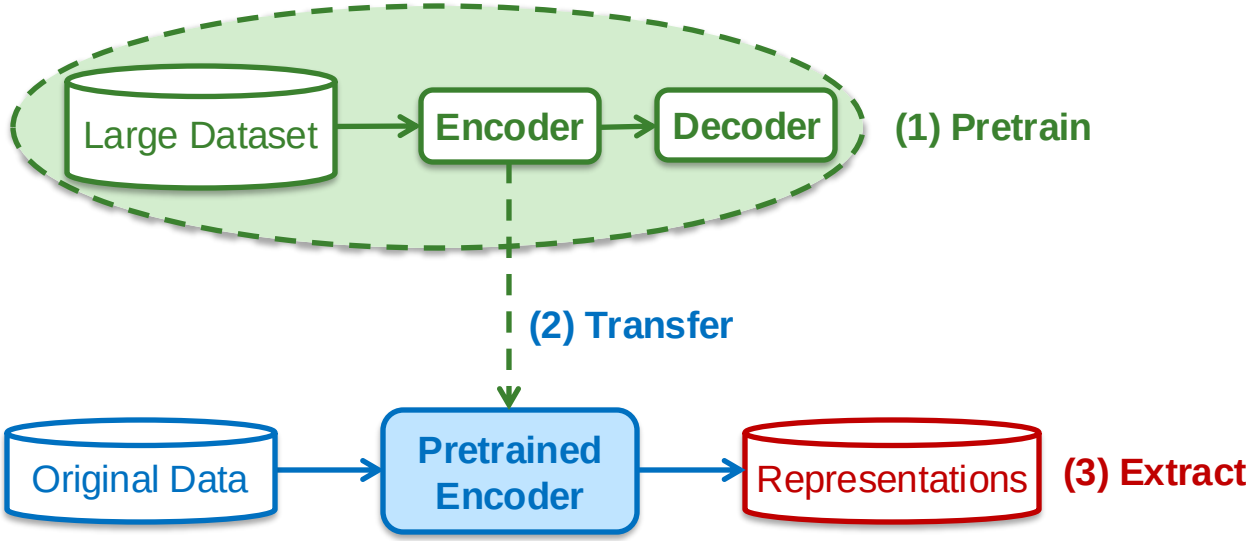
- Loss functions for metric learning
  - ***Triplet loss***
    - *Training requires triplets of similar and dissimilar data*
    - *Performance depends the quality of mined triplets*
    - *Still very useful for weakly-supervised tasks*
  - ***Cosine/Arc loss***
    - *Training requires the same data as classification tasks*
    - *Loss function is a simple variant of "softmax + cross-entropy"*
    - *More effective for supervised tasks, even with class imbalance*
  - ***Takeaways***
    - *Both approaches are designed to learn representations where distances reflect semantic relationships in the original data*
    - *They are best suited for tasks requiring similarity measurement*

Self-Supervised Representation Learning

- Pretext tasks
- Invariant representations
- Representation collapse

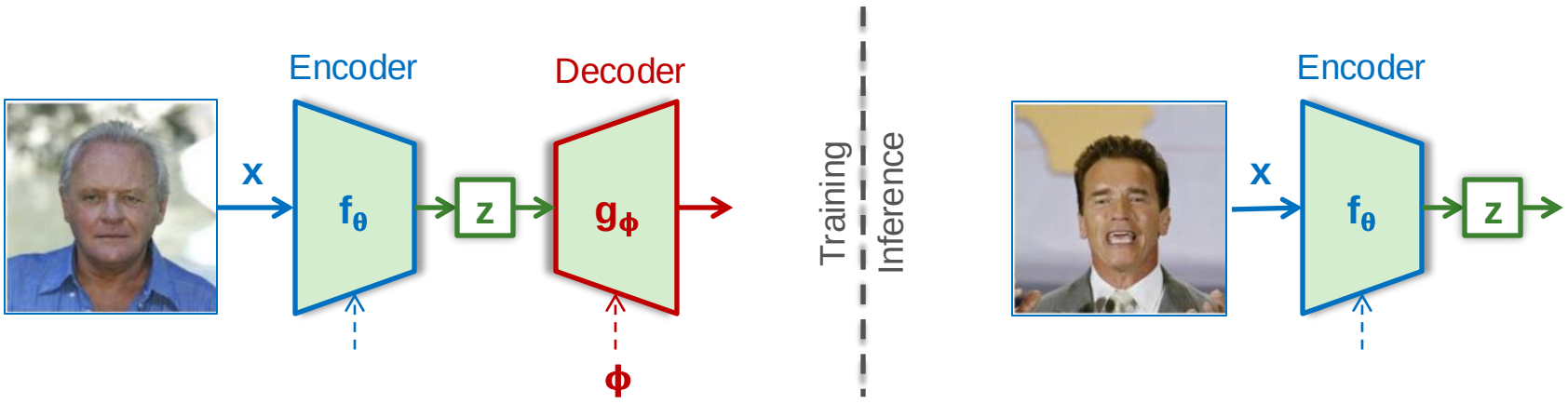
Introduction

- **Self-supervised representation learning**
  - ***Goal*** ➜ *Learn generic features from unlabeled data*
  - *Model is trained on a **fake task** using labels created from the data*
  - *A part of the trained model is then reused to get representations*



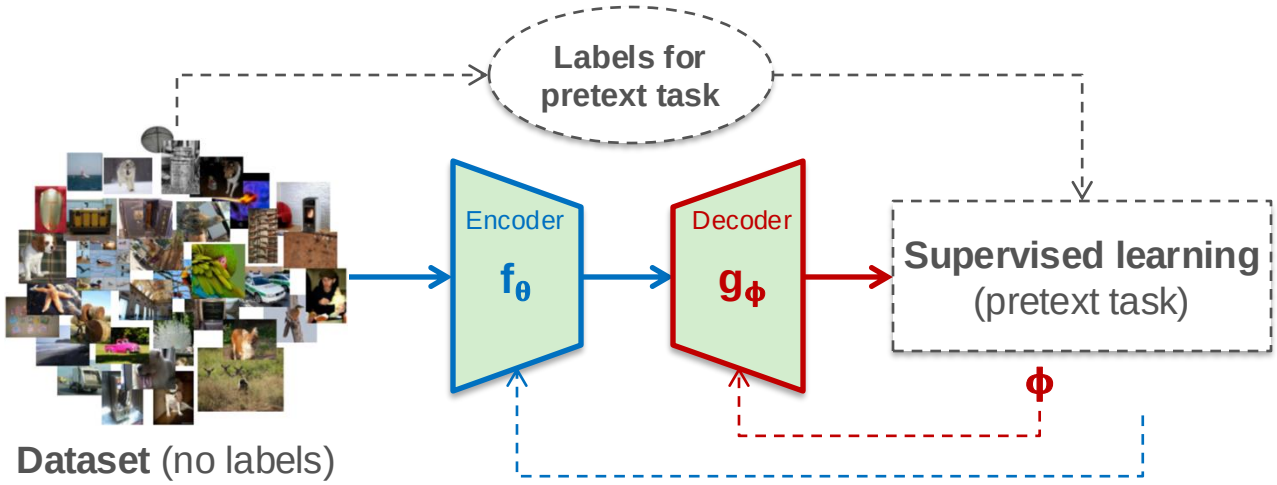
Encoder-decoder architecture

- Architecture used in **self-supervised** learning
  - ***Encoder*** → *Model that extracts representations from input data*
    - *Does not depend on the type of labels used for self-supervision*
  - ***Decoder*** → *Model that predicts the desired outputs from repres.*
    - *Tightly coupled with the type of labels used for self-supervision*
    - *Discarded after training*



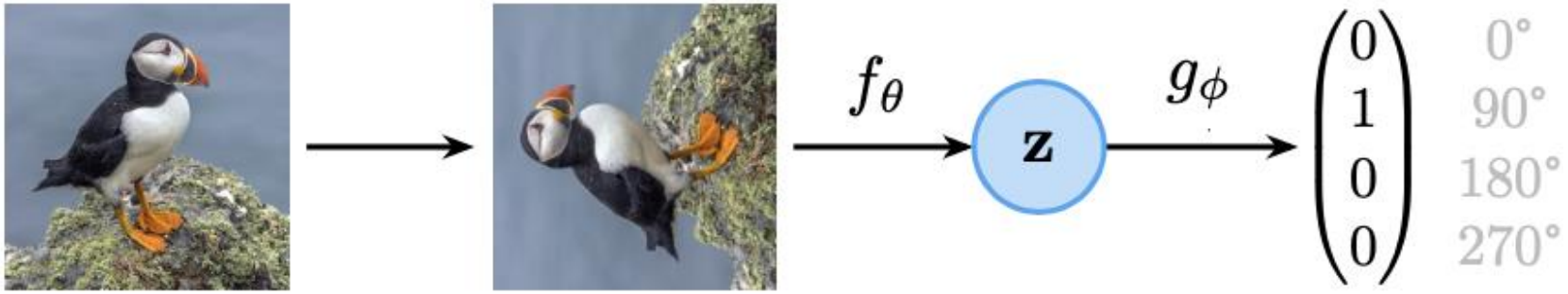
What is a pretext task?

- **Pretext task**
  - *Task designed to predict information derived from the data itself*
  - *Unsupervised learning conducted in a "supervised" manner*
  - ***Goal*** → *Encourage the encoder to learn useful representations*



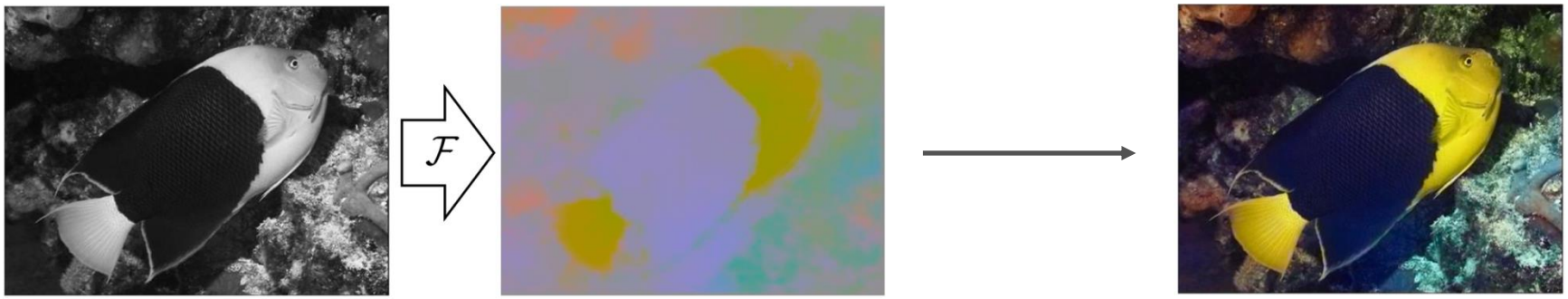
Pretext tasks based on images (1/3)

- **Classification of rotated images** *(Gidaris, 2018)*
  - ***Goal*** ➜ *Predict which rotation is applied to an image*
  - ***Decoder*** ➜ *Classification head (classes are 0°, 90°, 180°, 270°)*
  - ***Training*** ➜ *For each image of a sampled batch*
    - *Apply all rotations and generate the corresponding labels*
    - *Encode each image and decode the respective class probabilities*
    - *Update the model parameters using the cross-entropy loss*



Pretext tasks based on images (2/3)

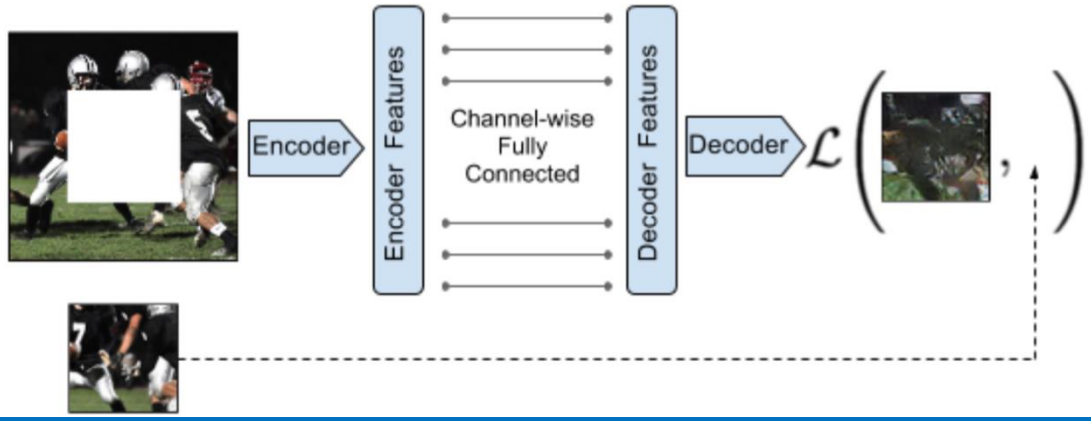
- **Colorization** *(Zhang, 2016)*
  - ***Goal*** ➜ *Predict colors from grayscale image*
  - ***Decoder*** ➜ *Pixel-wise classification head (with a fixed color palette)*
  - ***Training*** ➜ *For each image of a sampled batch*
    - *Split image into L-channel (grayscale) ab-channels (colors)*
    - *Encode grayscale image and decode color probabilities*
    - *Update the model parameters using the weighted cross-entropy loss*



Grayscale image: *L* channel | Color information: *ab* channels | Concatenate (*L*,*ab*) channels

Pretext tasks based on images (3/3)

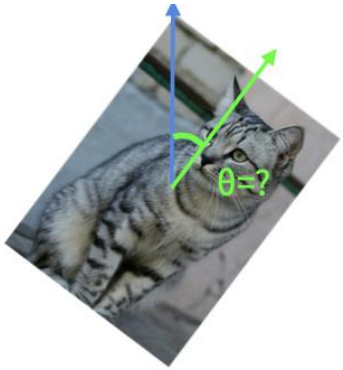
- **Inpainting** *(Pathak, 2016)*
  - ***Goal*** ➜ *Predict missing parts of an image*
  - ***Decoder*** ➜ *Regression head for missing pixels*
  - ***Training*** ➜ *For each image of a sampled batch*
    - *Remove a random patch and use its pixels as regression targets*
    - *Encode the modified image and decode the missing pixels*
    - *Update parameters with a combination of L2 loss and adversarial loss*



Moving away from pretext tasks

- **Limitations**
  - *Creating pretext tasks requires knowledge and experimentation*
  - *A single pretext task is often not enough to learn representations that perform well on unrelated downstream tasks*

image completion



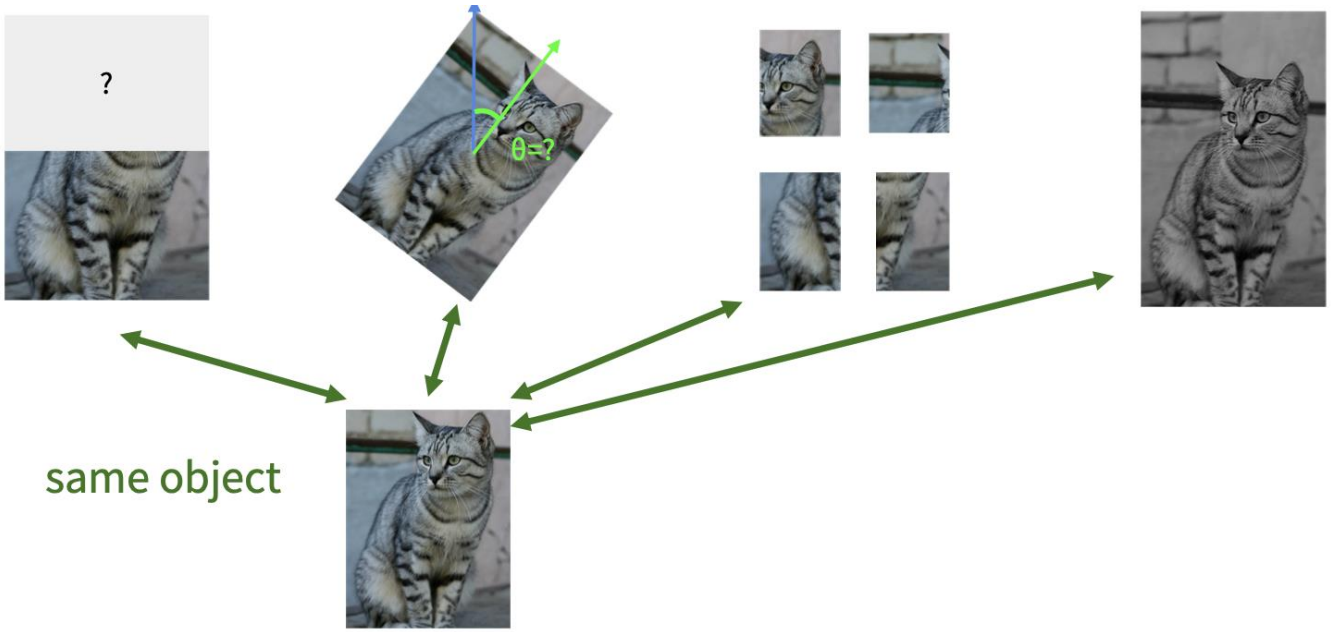
rotation prediction

"jigsaw puzzle"

colorization

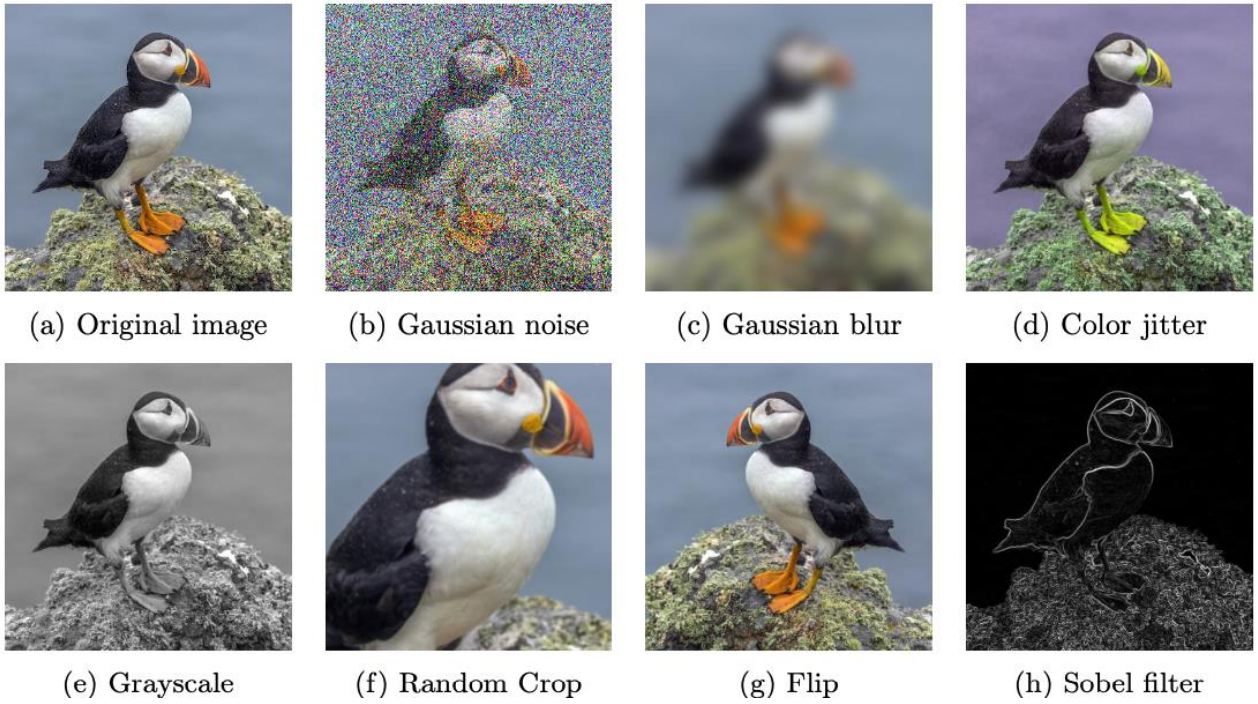
Learning invariant representations

- **Key idea ➜ Focus on invariance**
  - *Learn representations that are **invariant** to some transformations, rather than learning how to predict them explicitly !!!*
  - *This ensures that representations are general and transferable*



Self-supervision for computer vision

- **Invariance in the image domain**
  - *Image representations should be robust to various operations, such as distortion, rescaling, translation, color alteration, …*

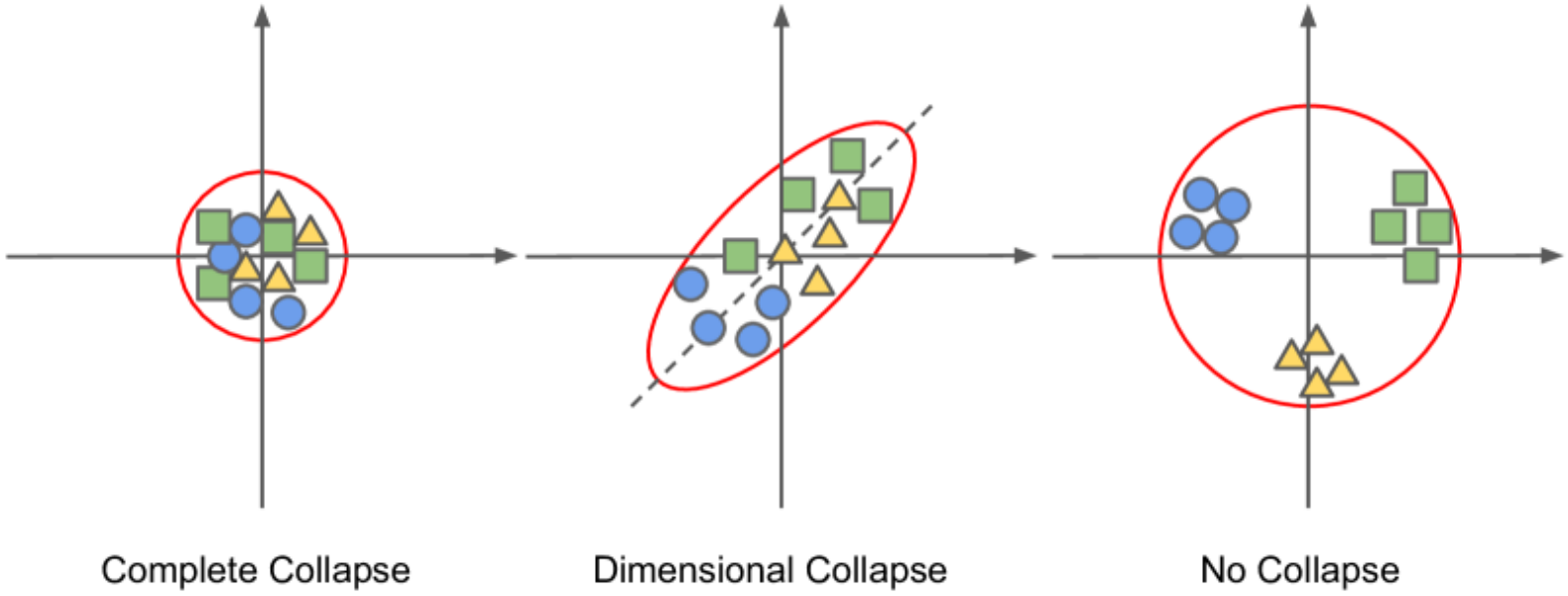


(a) Original image (b) Gaussian noise (c) Gaussian blur (d) Color jitter

(e) Grayscale (f) Random Crop (g) Flip (h) Sobel filter

Representation collapse

- There is a **pitfall** in learning invariant representations
  - *Invariance can be **trivially** achieved by mapping all data points to the same representation* *(constant values for all inputs)*
  - *Collapsed representations are useless for downstream tasks*



Branches of self-supervised learning

- Strategies to avoid representation collapse *(Uelwer, 2023)*
  - ***Contrastive learning***
    - *Push away representations of unrelated data points*
  - ***Clustering-based methods***
    - *Incorporate clustering objectives when learning representations*
  - ***Information maximization***
    - *Ensure high diversity and independence in representations*
  - ***Teacher-student methods***
    - *Use asymmetric networks with different learning objectives*

Summary

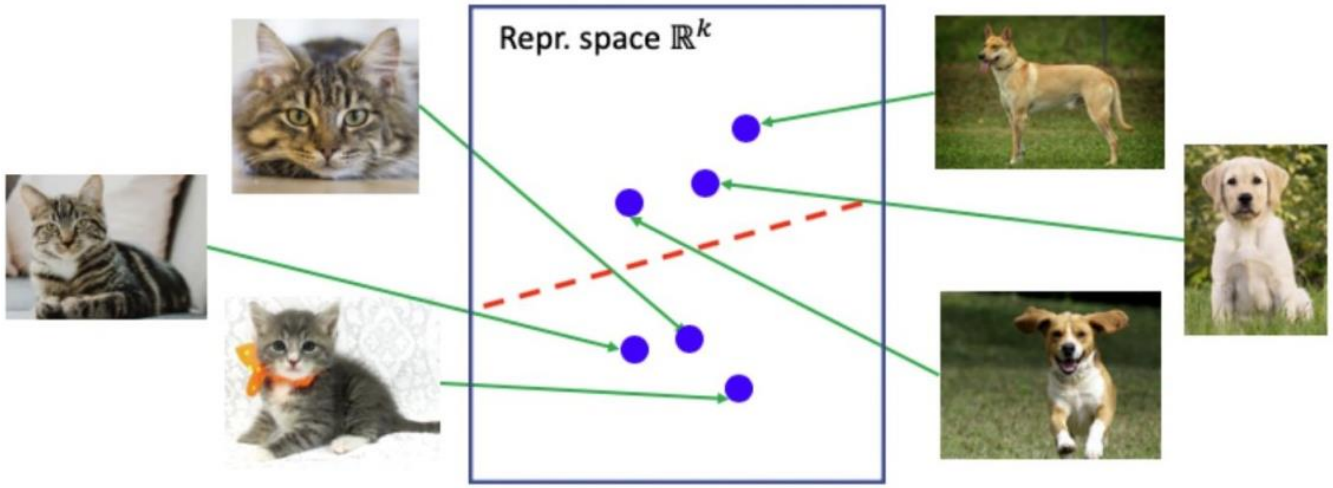
- **Self-supervised (representation) learning**
  - *How to extract meaningful representations from unlabeled data?*
  - *Design a supervised task using labels created from the data itself*
- **Pretext tasks**
  - *Early attempts at self-supervised learning*
  - *Introduced foundational ideas about invariance*
- **Learning invariant representations**
  - *Representations should remain consistent across transformations*
  - *The greatest difficulty is to avoid representation collapse*
  - *Still an open problem that can be addressed in several ways*

Contrastive Learning

- What is contrastive learning?
- Loss function (infoNCE)
- SimCLR framework

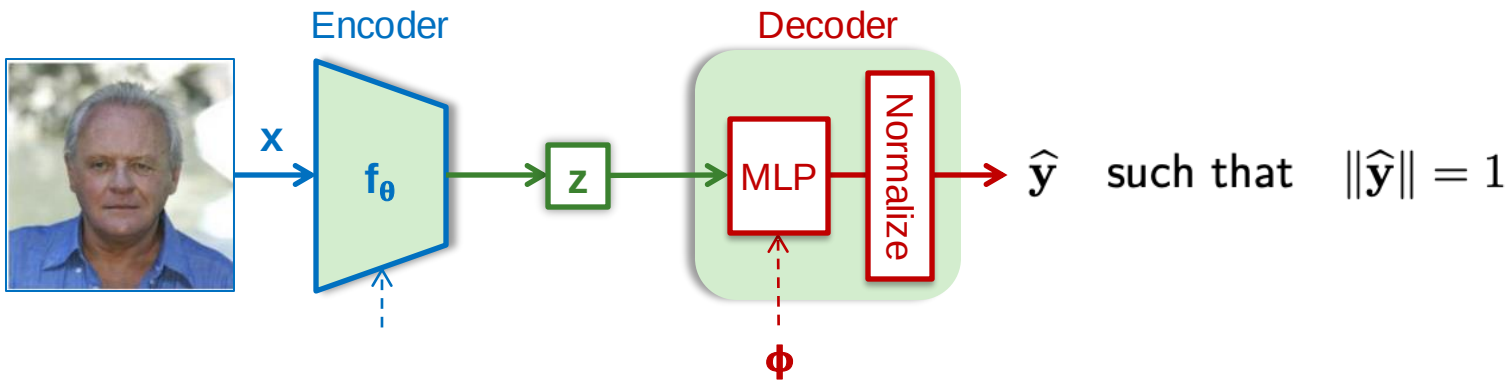
What is contrastive learning?

- **Contrastive learning**
  - *Self-supervised learning approach that aims to structure the representation space based on the similarity of the input data*
    - *The representations of similar data points are pulled closer together, while the representations of dissimilar ones are pushed farther apart*
    - *Same principle as metric learning, but without the labels*



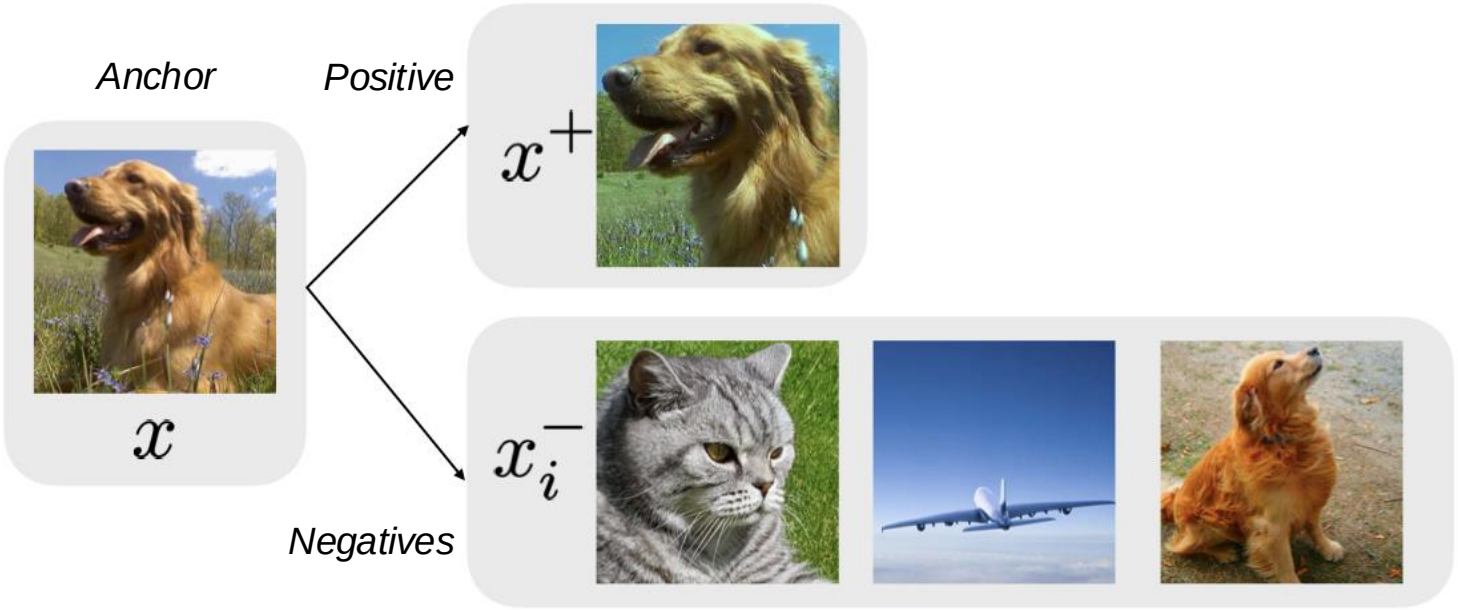
Architecture for contrastive learning

- Network **architecture** for contrastive learning
  - ***Encoder*** → *Model tailored to a domain or a task*
    - ***Image domain:*** *Convolutional backbone, Visual transformer, …*
  - ***Decoder*** → *Projection head for contrastive learning*
    - ***MLP:*** *Series of fully-connected layers and nonlinear activations*
    - ***Normalizer:*** *Layer that divides each input by its norm*



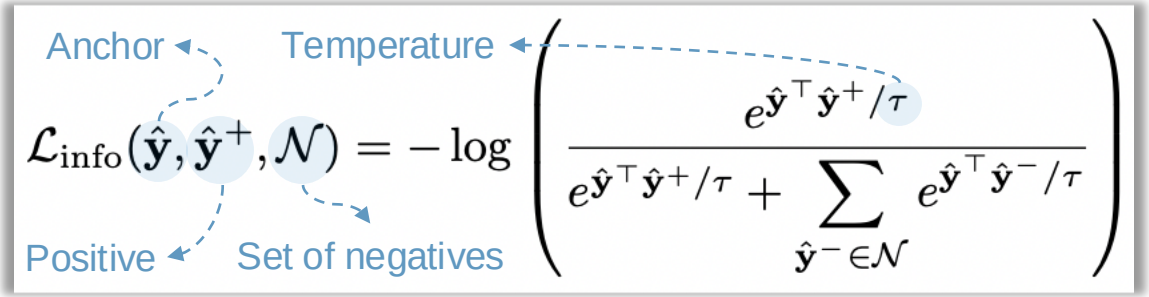
InfoNCE loss (1/2)

- **InfoNCE →** Loss function used in contrastive learning
  - *For a given point (anchor), the loss needs to know **one similar point** (positive) and **multiple dissimilar points** (negatives)*



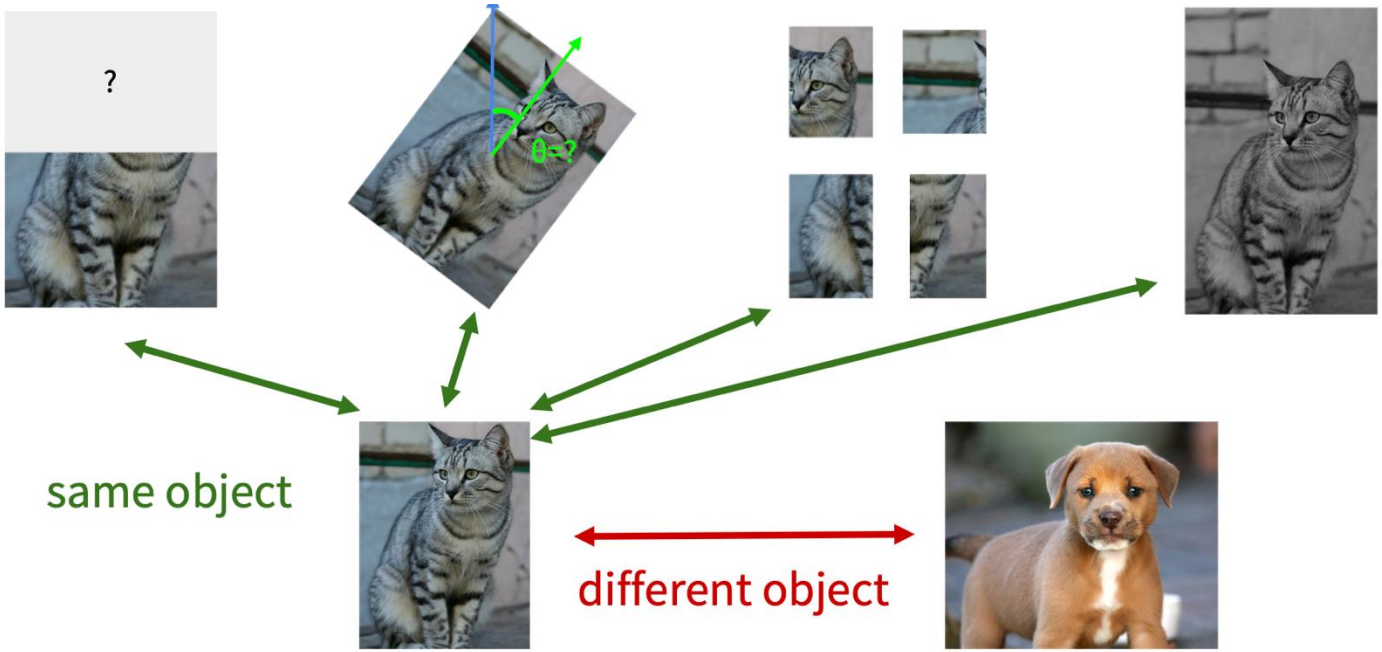
InfoNCE loss (2/2)

- InfoNCE is formulated as a **classification** task
  - ***Anchor*** ➜ *Point to classify*
  - ***Positive + Negatives*** ➜ *Classes*
  - ***Loss function*** ➜ *Softmax + Cross-entropy*
    - *As the loss maximizes the probability that the anchor belongs to the "positive" class, the model learns to increase the anchor-positive similarity, while decreasing the anchor-negative similarities*



Data augmentation (1/2)

- How to define **similarity** without explicit labels ?
  - ***Positive*** → *Two augmented views of the same data point*
  - ***Negative*** → *Any other pair of data points* *(augmented or not)*



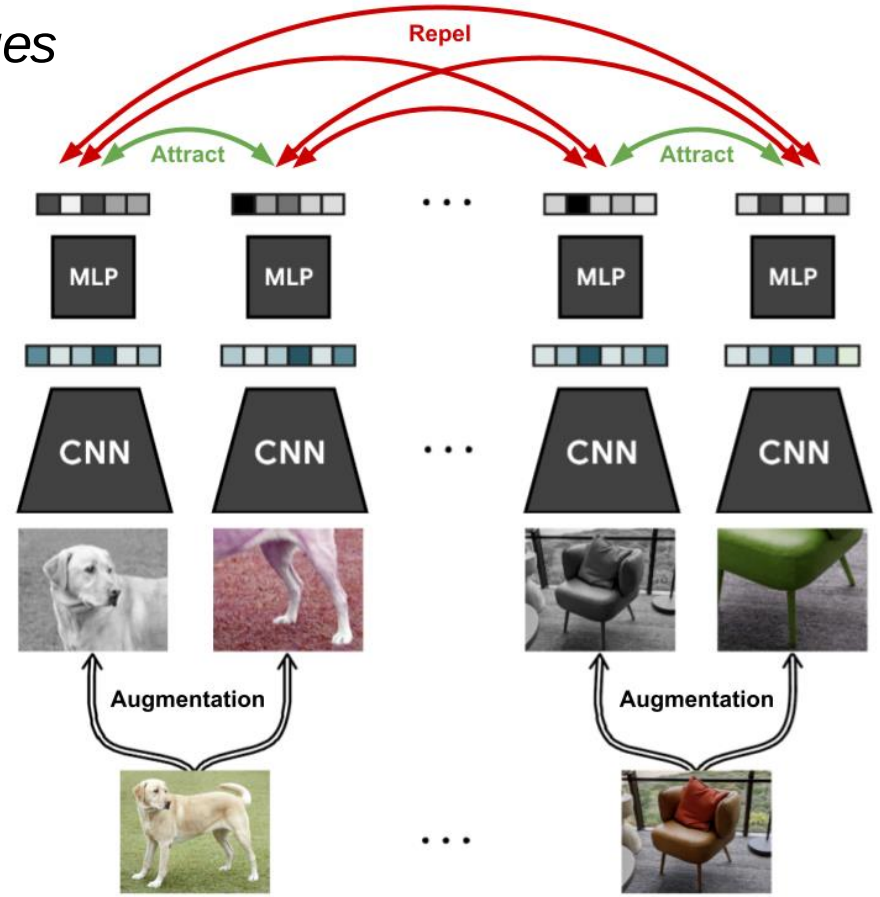
Data augmentation (2/2)

- Augmentations for the **image domain** *(Chen, 2020)*
  - *Serial composition of 3 transformations*
    - *Random crop and flip, then resize to original dimensions*
    - *Random color distortion (drop or jitter)*
    - *Random gaussian blur*
  - *Other transformations may not work as well*

(a) Original (b) Gaussian blur (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter)

SimCLR framework (1/2)

- **SimCLR→** Contrastive learning for visual representations
  - *For each sampled batch of images*
    - *Apply **two** random augmentations to each image and pass them through the encoder-decoder*
    - *Compute the infoNCE loss of each view as the anchor, using the other view as the positive, and anything else as negatives*
    - *Update the parameters of the encoder and the decoder*
    - *Repeat until training is over*



SimCLR framework (2/2)

- The **loss function** used by SimCLR *(Chen, 2020)*
  - *Both views of a positive pair are used as anchor in InfoNCE*

$$\mathcal{L}_{\text{SimCLR}}(\theta, \phi) = \frac{1}{2N} \sum_{n=1}^{N} \mathcal{L}_{\text{info}}\left(\hat{\mathbf{y}}_n^{(1)}, \hat{\mathbf{y}}_n^{(2)}, \mathcal{B}_n\right) + \mathcal{L}_{\text{info}}\left(\hat{\mathbf{y}}_n^{(2)}, \hat{\mathbf{y}}_n^{(1)}, \mathcal{B}_n\right)$$

  - *For a positive pair, the negatives are all other augmented views*

$$\mathcal{B}_n = \left\{\hat{\mathbf{y}}_1^{(1)}, \hat{\mathbf{y}}_1^{(2)}, \ldots, \hat{\mathbf{y}}_N^{(1)}, \hat{\mathbf{y}}_N^{(2)}\right\} \setminus \left\{\hat{\mathbf{y}}_n^{(1)}, \hat{\mathbf{y}}_n^{(2)}\right\}$$

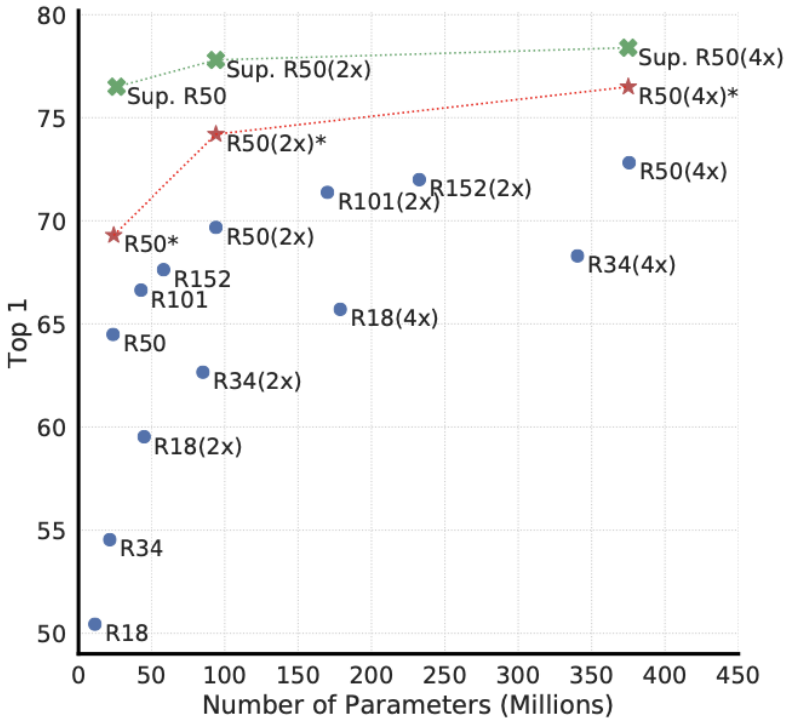
  - *This loss is more effective when the **batch size is large***

Key design choices

- **Data Augmentations**
  - *Strong augmentations force the model to learn representations that are invariant to transformations and thus more transferable*
- **Projection Head** (decoder)
  - *The use of a non-linear projection head improves performance*
- **Batch Size**
  - *A large batch size increases the pool of negative samples, which improves contrastive learning by creating harder negatives*
  - *Strong performance requires batches as large as **8'192 samples***

SimCLR evaluation (1/2)

- **Linear probing** on ImageNet
  - *Train encoder-decoder with SimCLR on ImageNet (full train set)*
  - *Train linear classifier with frozen encoder on ImageNet*



ResNet encoder with varied depth and width.

- Models in blue dots are trained for 100 epochs.
- Models in red stars are trained for 1000 epochs.
- Models in green crosses are supervised.

SimCLR evaluation (2/2)

- **Transfer learning** to other datasets
  - *Train encoder-decoder with SimCLR on ImageNet (full train set)*
  - *Train linear classifier with encoder on a **supervised** dataset*
    - ***Linear evaluation*** ➔ *Encoder is frozen and classifier is trained*
    - ***Fine-tuning*** ➔ *Both encoder and classifier are trained*
  - *Compare with a convolutional network pretrained on ImageNet*
    - *All models use ResNet-50 (4x) as encoder or backbone*

	Food	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Aircraft	VOC2007	DTD	Pets	Caltech-101	Flowers
Linear evaluation:												
SimCLR (ours)	**76.9**	**95.3**	80.2	48.4	**65.9**	60.0	61.2	**84.2**	**78.9**	89.2	**93.9**	**95.0**
Supervised	75.2	**95.7**	**81.2**	**56.4**	64.9	**68.8**	**63.8**	83.8	**78.7**	**92.3**	**94.1**	94.2
Fine-tuned:												
SimCLR (ours)	**89.4**	**98.6**	**89.0**	**78.2**	**68.1**	**92.1**	**87.0**	**86.6**	**77.8**	92.1	**94.1**	97.6
Supervised	88.7	98.3	**88.7**	**77.8**	67.0	91.4	**88.0**	86.5	**78.8**	**93.2**	**94.2**	**98.0**
Random init	88.3	96.0	81.9	**77.0**	53.7	91.3	84.8	69.4	64.1	82.7	72.5	92.5

Summary

- **SimCLR** *(Chen, 2020)*
  - *Learns representations from images without labels*
  - *No need for specialized architecture*
  - *Works well with large datasets and batch sizes*

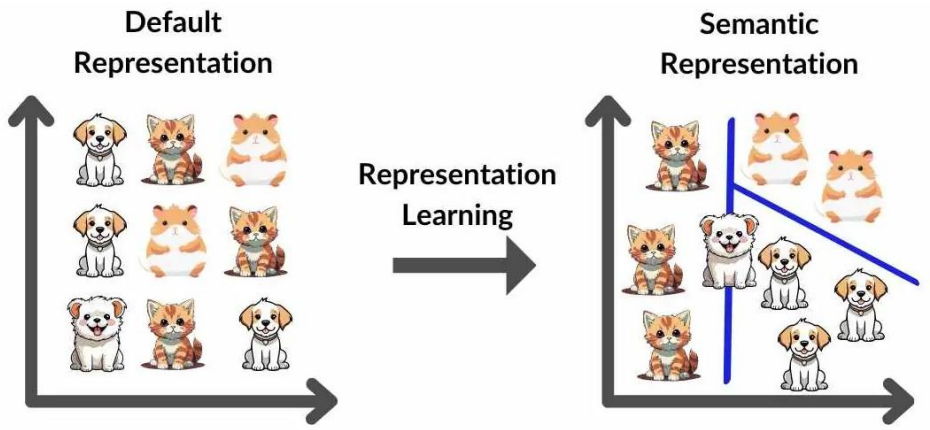
- **Only works with…**
  - *A large nonlinear projection head*
  - *The right augmentations*
    - *Performance heavily depends on the quality of augmentations*
  - *Large batch size and long training*
    - *Requires large memory and computational resources*

Conclusion

- Transfer learning
- Metric learning
- Contrastive learning

Representation learning

- **Objective**
  - *Learn representations that simplify downstream tasks*
- **Methods Discussed**
  - *Transfer Learning*
  - *Metric Learning*
  - *Contrastive Learning*



- **Current SOTA**
  - *DINOv2 (Oquab, 2023)*
  - *Self-supervised teacher-student method*
  - *Encoder implemented as vision transformer*

Transfer learning

- **Core Idea**
  - *Reuse a model trained on a large general dataset (e.g., ImageNet)*
  - *Fine-tune or extract the learned representations on a new task*

- **Benefits**
  - *Reduced Training Time*
    - *Speeds up convergence by starting from a strong initialization*
  - *Performance Boost*
    - *Often yields better results than training from scratch, especially with limited supervised data in the target domain or task*

Metric learning

- **Key Idea**
  - *Learn representations so that similar data points are mapped close together and dissimilar data points are mapped far apart*
- **Benefits**
  - *Easily adapt to new tasks by comparing distances*
  - *Useful for clustering, nearest-neighbor search, retrieval, …*
- **Challenges**
  - *Choosing the right loss function* *(triplet loss, cosine loss, arc loss, …)*

Contrastive learning

- **Core Concept**
  - *Learn representations by contrasting different augmentations of the same image as well as different images in the dataset*
- **Strengths**
  - *Self-supervised, needs no explicit labels*
  - *Produces representations useful for diverse downstream tasks*
- **Limitations**
  - *Performance heavily depends on the quality of augmentations*
  - *Training requires many computational resources*