# LESS DATA

## Deep Learning for Computer Vision

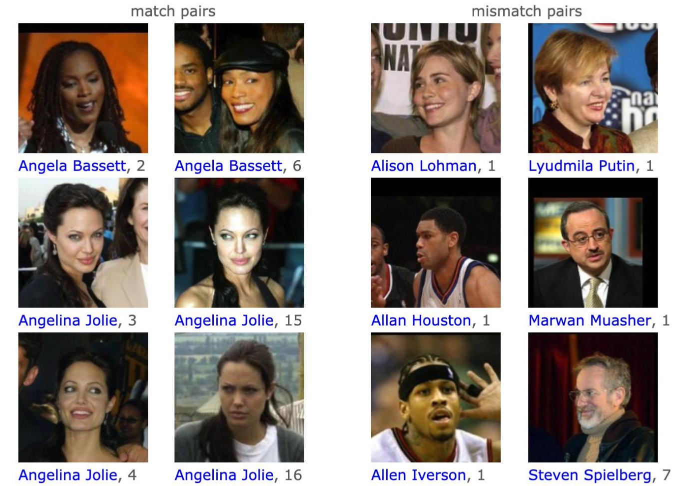
Arthur Douillard

https://arthurdouillard.com/deepcourse

# Few-Shot Learning

# Few-Shots Learning

[LFW: Labeled Face in the Wild]



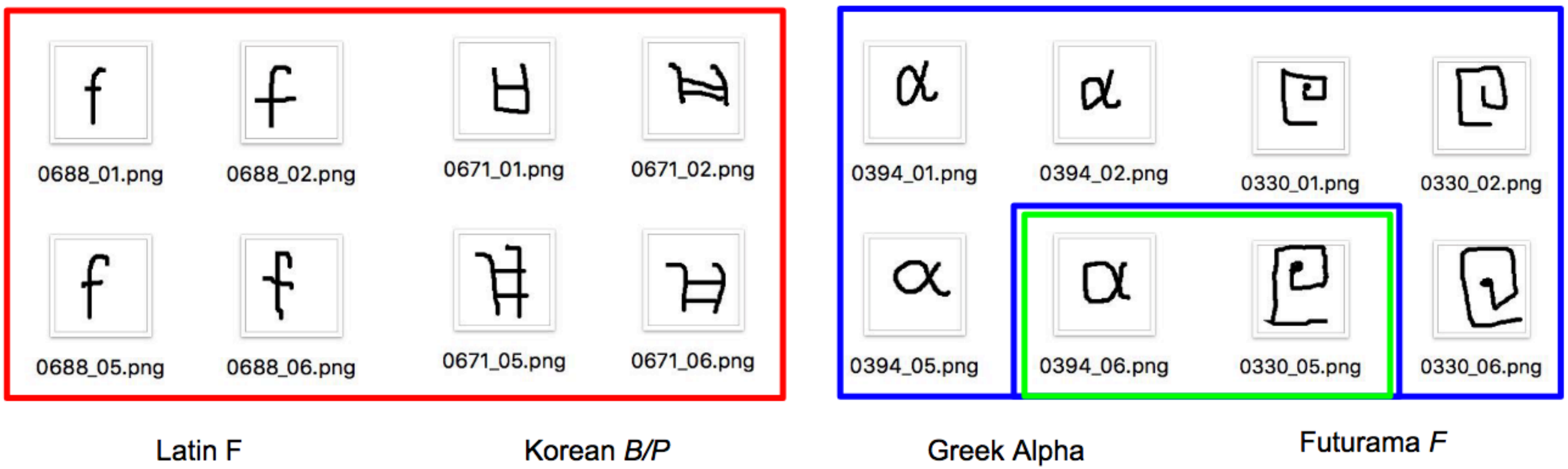

[Omniglot]


[Kaggle's Humpback Whale identification]

# Few-Shots Learning

1. Learn to classify the few labeled samples in the **background set**
2. With a a few labeled samples in **support set**, classify the **query set**

Training: Testing:




Number of labeled samples / class: **K-shots**

Number of classes in testing: **N-ways**

# Metric Learning

A huge, potentially growing number of classes.

Less than a dozen labeled samples per class.

Discriminative model is impossible.

**What if we learn a metric instead?**

# Metric Learning

Distance:

$$d(x_1, x_2) = \|f(x_1) - f(x_2)\|_2 \in \mathbb{R}^+$$

Similarity:

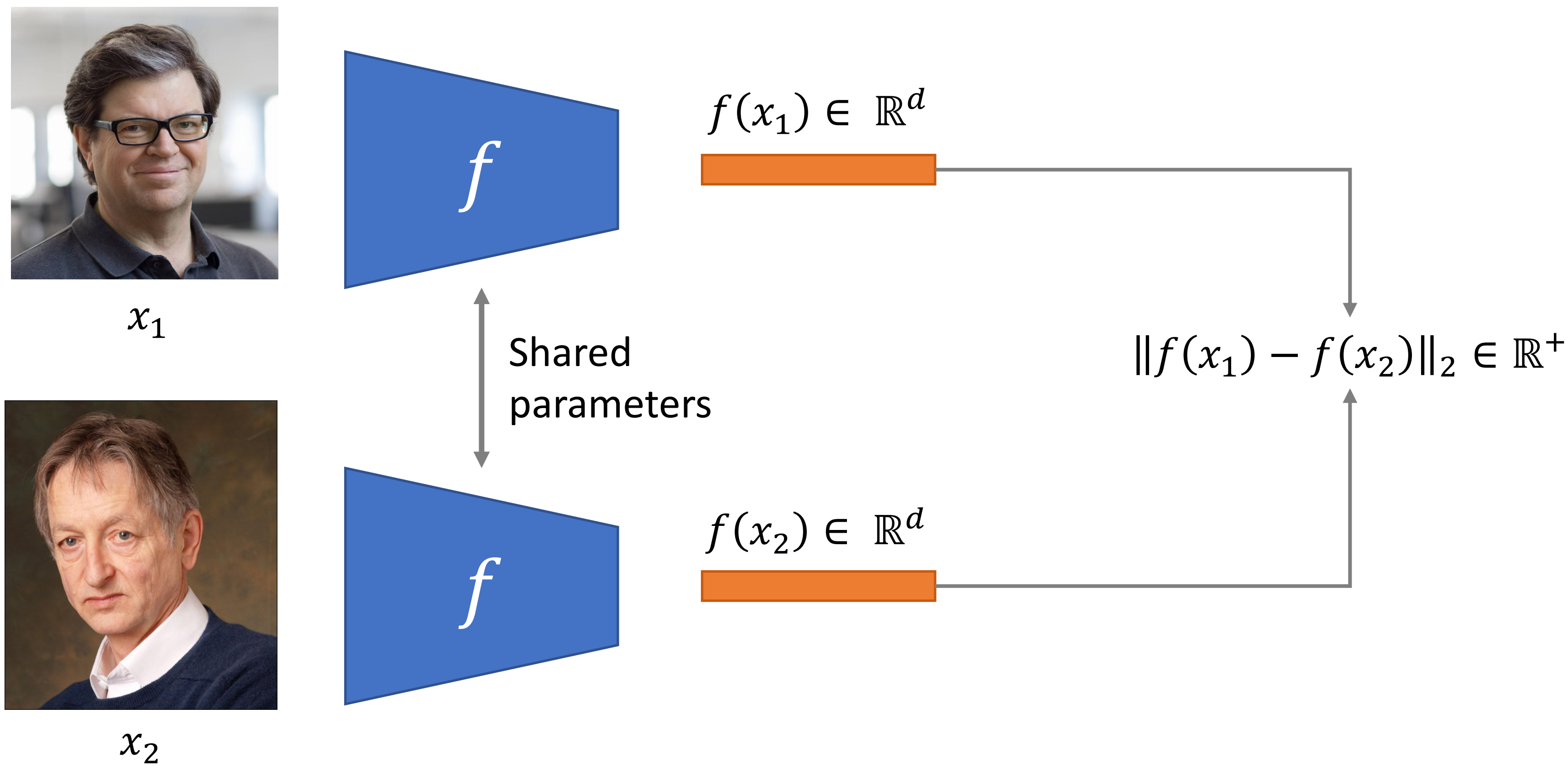
$$s(x_1, x_2) = \cos\big(f(x_1), f(x_2)\big) \in [-1, +1]$$

With $f(x) \in \mathbb{R}^d$ a features extractor (e.g. ConvNet).

Given two images of the class, we want:
- Minimize distance
- Maximize similarity

Given two images of different classes, we want:
- Maximize distance
- Minimize similarity

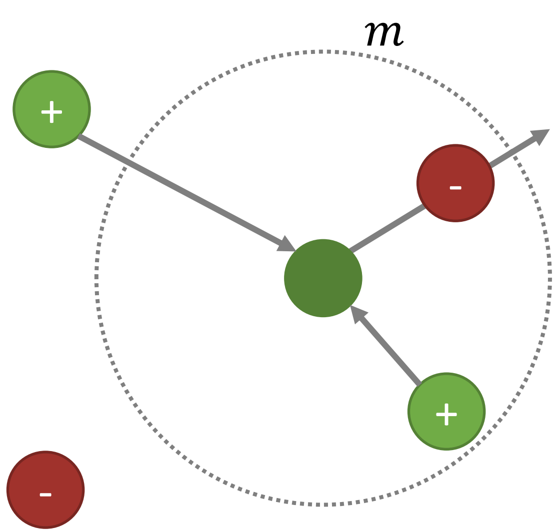
# Siamese Network



# Contrastive Loss

$$\boldsymbol{D} = \|f(x_1) - f(x_2)\|_2 \in \mathbb{R}^+$$

$$\mathcal{L}_{contrastive}(\boldsymbol{y}, \boldsymbol{D}) = \frac{1}{2}(1 - \boldsymbol{y})\boldsymbol{D}^2 + \frac{1}{2}\boldsymbol{y}\max(m - \boldsymbol{D}, 0)^2$$



# Weaknesses of Pairwise Models

- The margin $m$ may be hard to tune, especially because distributions can change through training
- A double-margin may improve to avoid collapsing all positive samples together
- Try to learn an absolute distance between images

# Triplet Network

We want to learn **relative distance** between samples

Given an anchor $x_a$, we want to have a small distance with a positive (same class) $x_+$:

$$\min \|f(x_a) - f(x_+)\|_2$$

And maximize with a negative (different class) $x_-$:

$$\max \|f(x_a) - f(x_-)\|_2$$

Therefore we want that:

$$\|f(x_a) - f(x_-)\|_2 > \|f(x_a) - f(x_+)\|$$
$$\|f(x_a) - f(x_-)\|_2 - \|f(x_a) - f(x_+)\| > 0$$

Add a margin $m$ to ensure extra separability:

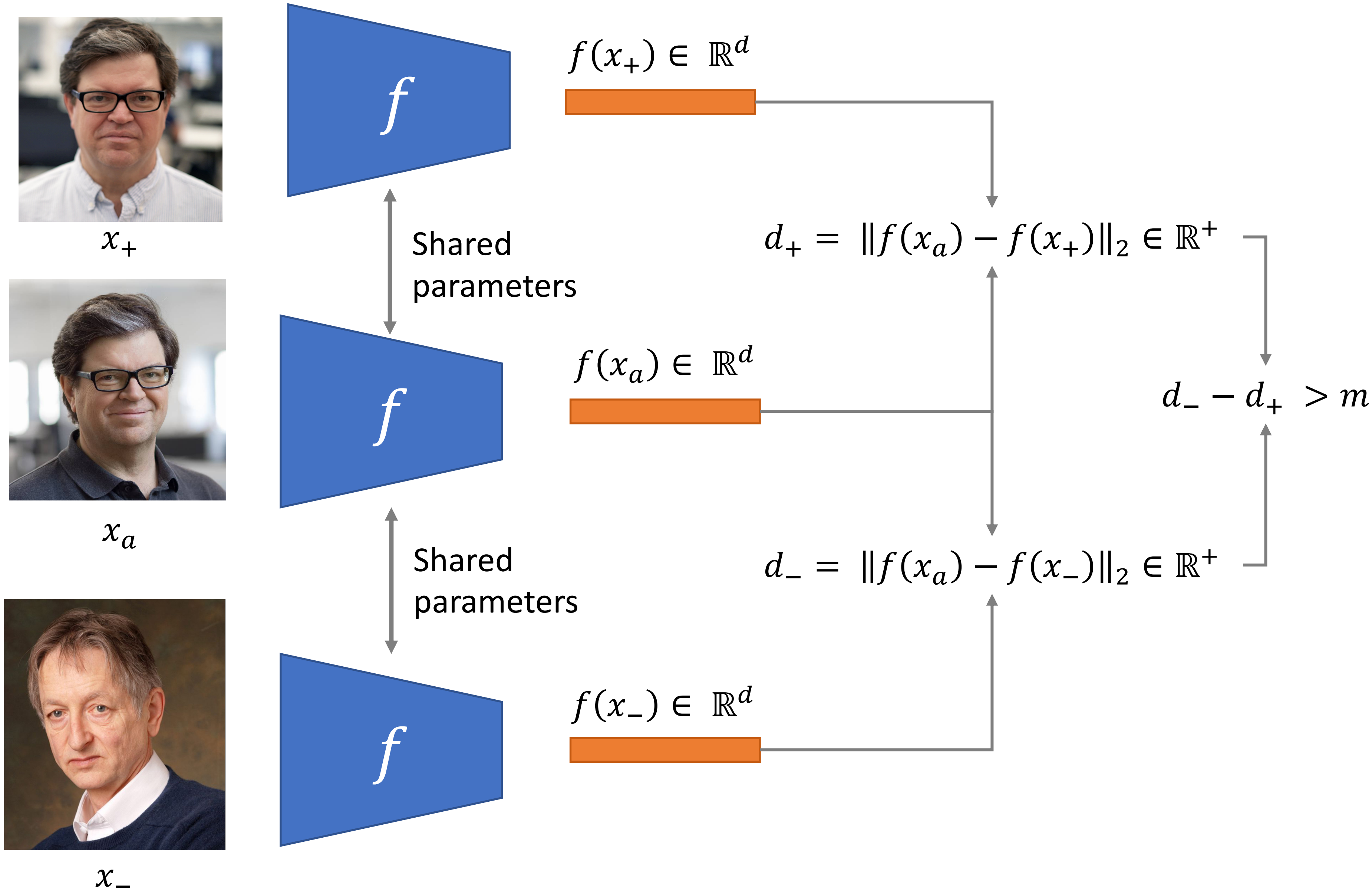
$$\|f(x_a) - f(x_-)\|_2 - \|f(x_a) - f(x_+)\| > m$$

Thus the loss is:

$$\min \max(\|f(x_a) - f(x_+)\|_2 - \|f(x_a) - f(x_-)\| + m, 0)$$

[Hoffer et al. SIMBAD 2015]

# Triplet Network



# Hard Negative Mining

Most triplets are easy.

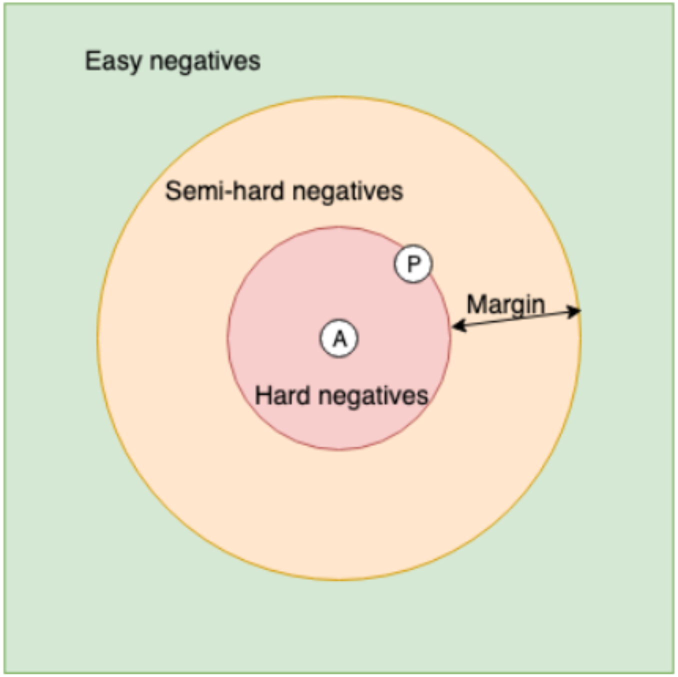
We want to sample either:

**Hard negatives**:

$$\|f(x_a) - f(x_+)\|_2 > \|f(x_a) - f(x_-)\| + m$$

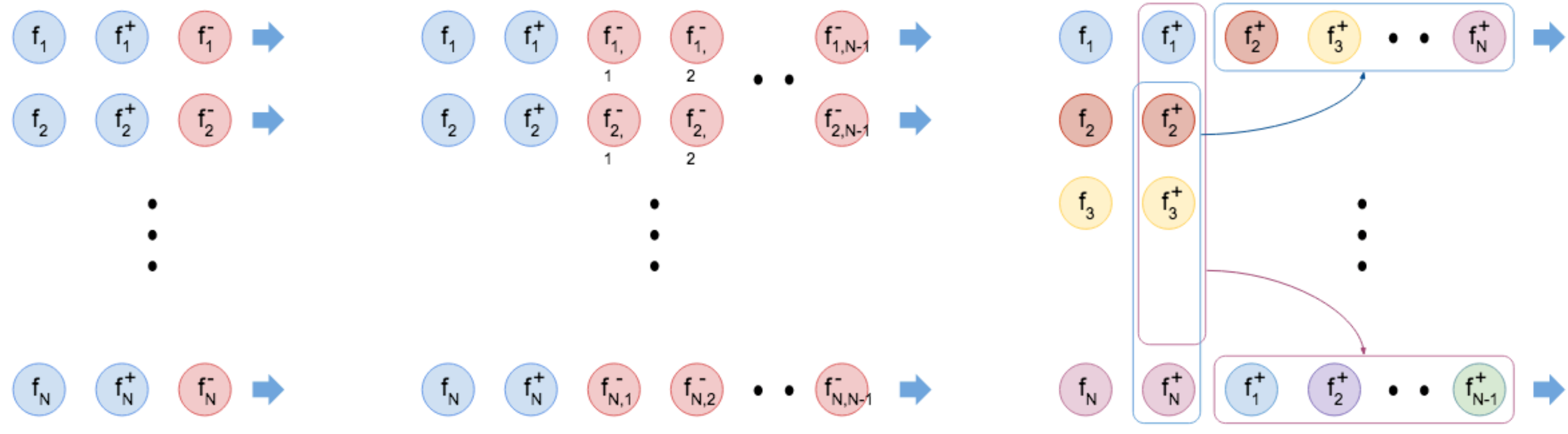
**Semi-Hard negatives**:

$$\|f(x_a) - f(x_+)\|_2 > \|f(x_a) - f(x_-)\|$$




Pretty much essential to have State-of-the-Art performance with Triplet Networks!

# N-Pairs

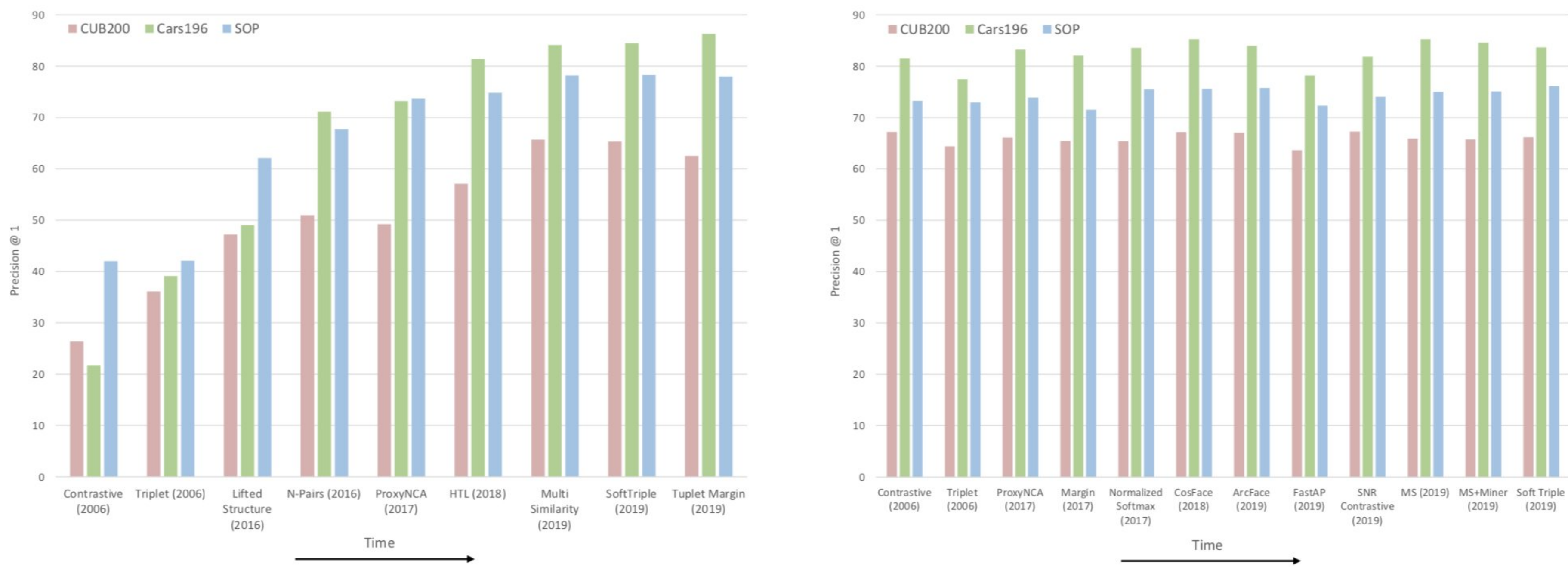


(a) Triplet loss (b) $(N+1)$-tuplet loss (c) $N$-pair-mc loss

Efficient generalization of Triplet networks that uses the whole batch.

[Sonh, NeurIPS 2016]

# Troubles in Metric Learning…




(a) The trend according to papers

(b) The trend according to reality

**Fig. 4.** Papers versus Reality: the trend of Precision@1 of various methods over the years.

The gain of more recent Metric Learning models <u>may</u> come from:

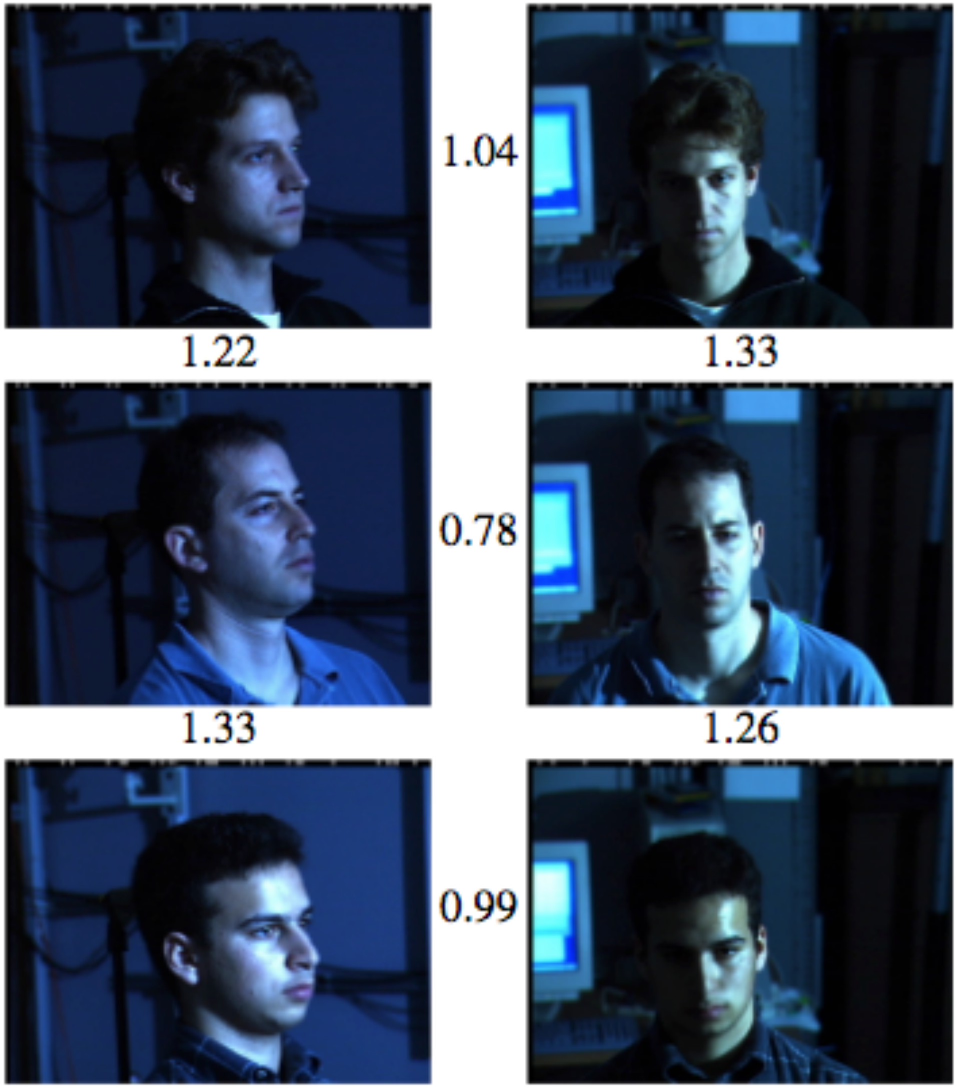
- Better backbone
- Better hyperparameters tuning
- Better data augmentation

[Musgrave et al. arXiv 2020]

# FaceNet

Triplet Network with semi-hard negative mining.

Pretty much solved the LFW dataset.




Distance between pairs

[Schroff et al. CVPR 2015]

# Meta-Learning MAML

Learning to learn:

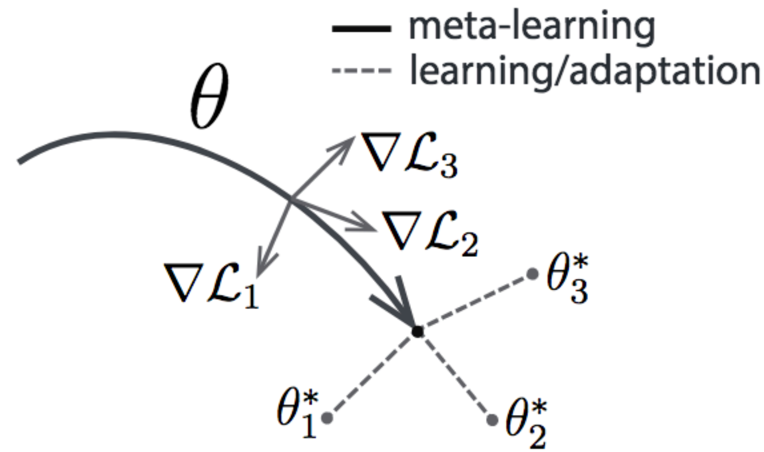
→ Learn a model that will be able to learn quickly given a few samples

Outer and inner loop:

- Inner loop learns to classify well a few labeled samples.

- Outer loop learns to have good weights for the inner loop.

During inference, perform only inner loop.




**Algorithm 1** Model-Agnostic Meta-Learning

**Require:** $p(\mathcal{T})$: distribution over tasks
**Require:** $\alpha$, $\beta$: step size hyperparameters

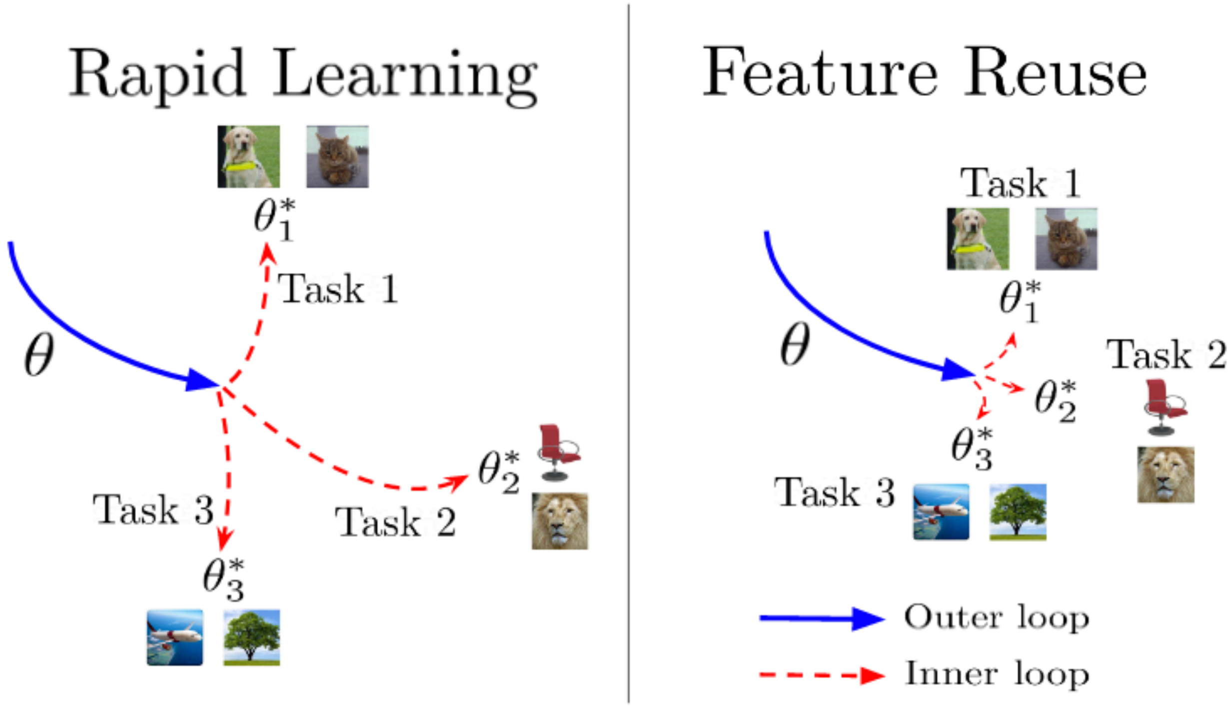
```
randomly initialize θ
while not done do
    Sample batch of tasks T_i ~ p(T)
    for all T_i do
        Evaluate ∇_θ L_{T_i}(f_θ) with respect to K examples
        Compute adapted parameters with gradient descent: θ'_i = θ − α∇_θ L_{T_i}(f_θ)
    end for
    Update θ ← θ − β∇_θ Σ_{T_i~p(T)} L_{T_i}(f_{θ'_i})
end while
```

[Finn et al. ICML 2017]

# Meta-Learning MAML

Does MAML *really* learns to learn (**rapid learning**)?

More probably it manages to learn features that generalize well (**feature reuse**)




[Raghu et al. ICLR 2020]

# Self-Supervised Learning

(also Unsupervised Learning)

# Setting

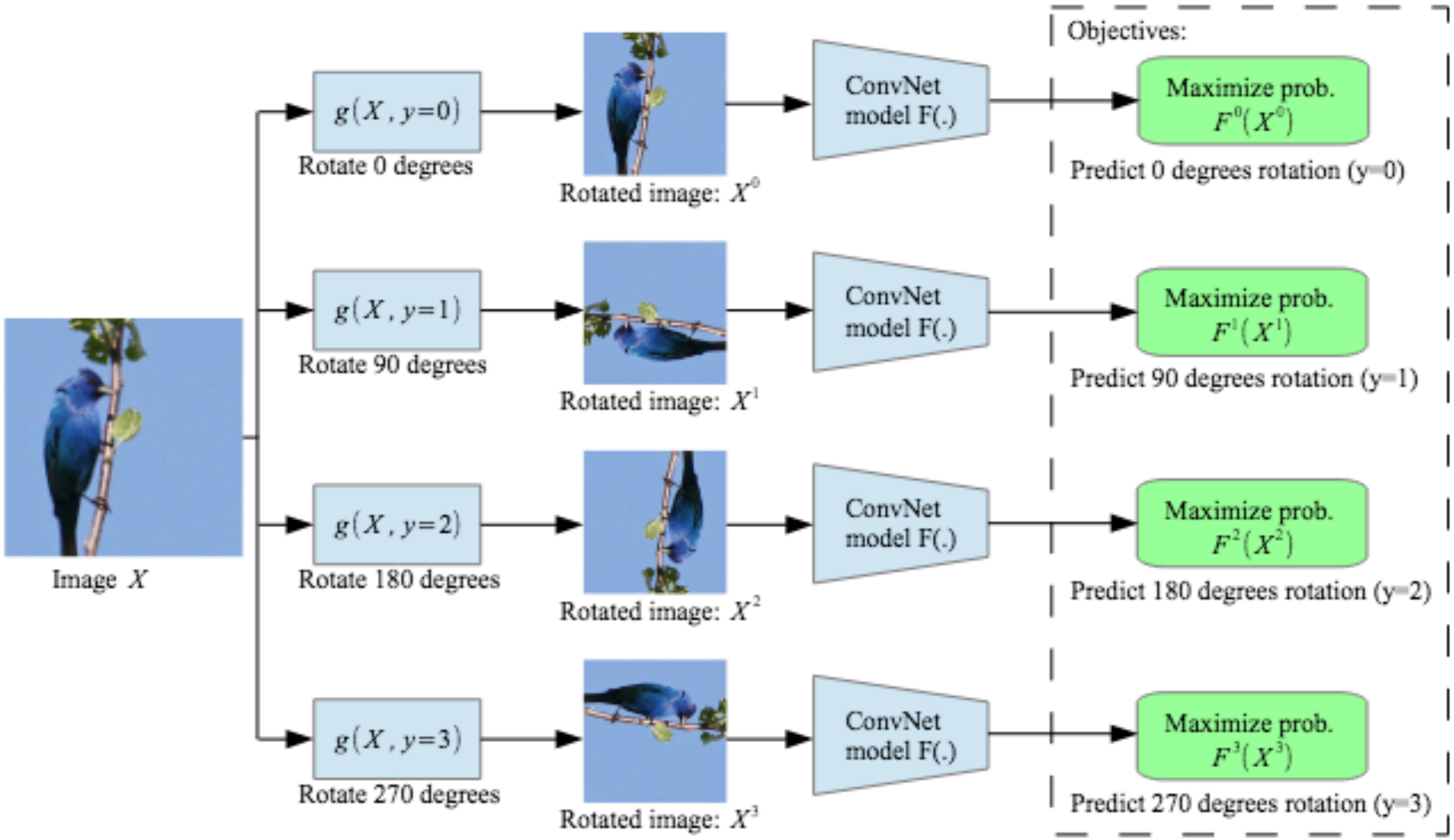
1. Learn a backbone with a lot of **unlabeled data** using a self-designed objective

2. Freeze the backbone
3. Learn a classifier on top of it on labeled data.

→ Evaluate how much the learned representation (features extracted by ConvNet) is useful.

It can be useful to have a good model pretraining in order to do transfer learning later.

# Rotation




Learn what is the rotation applied.

→ Thus it must learn what is the structure of the visual world.

[Gidaris et al. ICLR 2018]

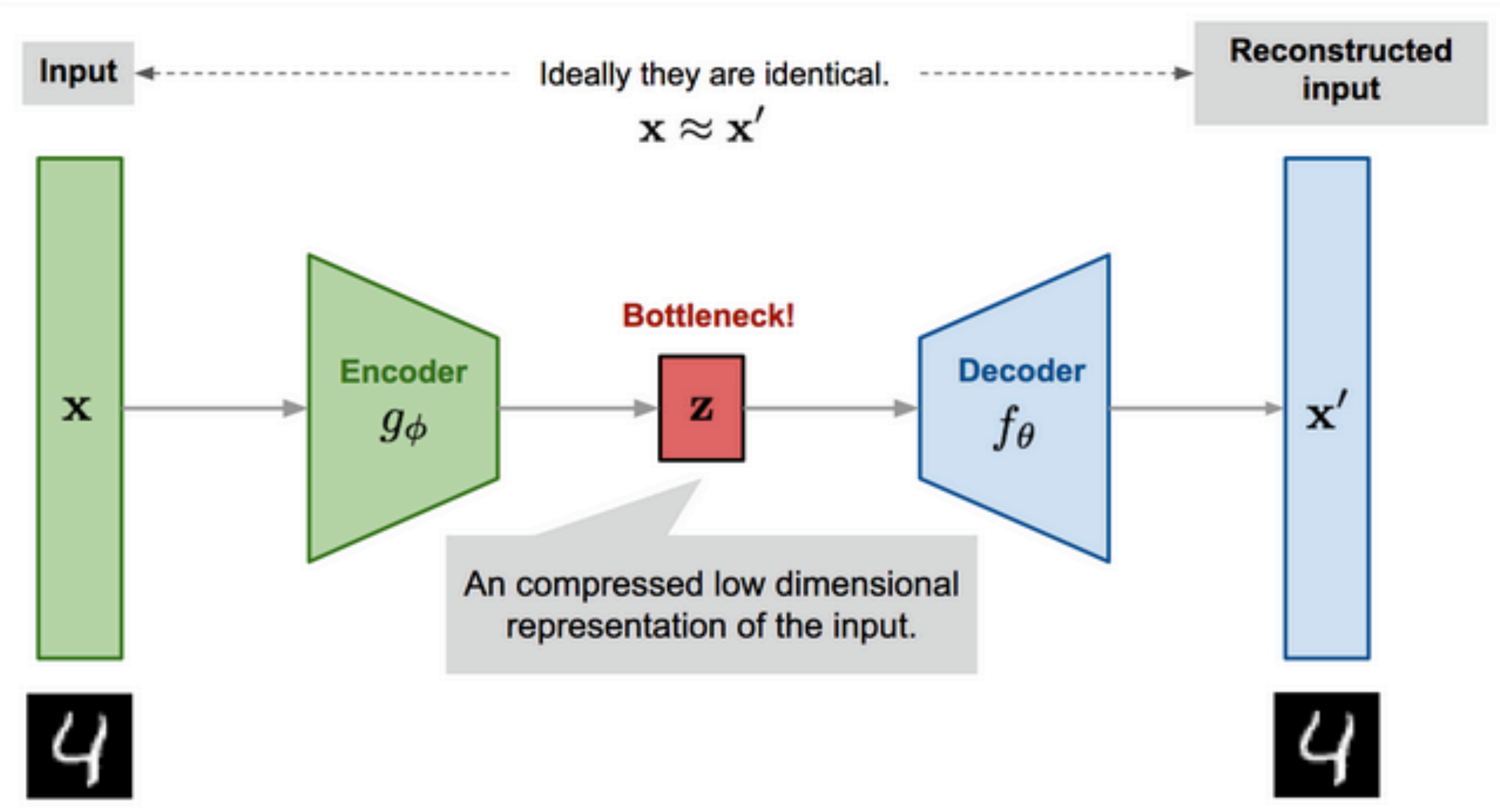
# Colorization

Learns to predict the colors from “*grayscale*” images.

→ Regression problem
→ Done in the LAB space instead of RGB

[Zhang et al. ECCV 2018]

# Auto-Encoder



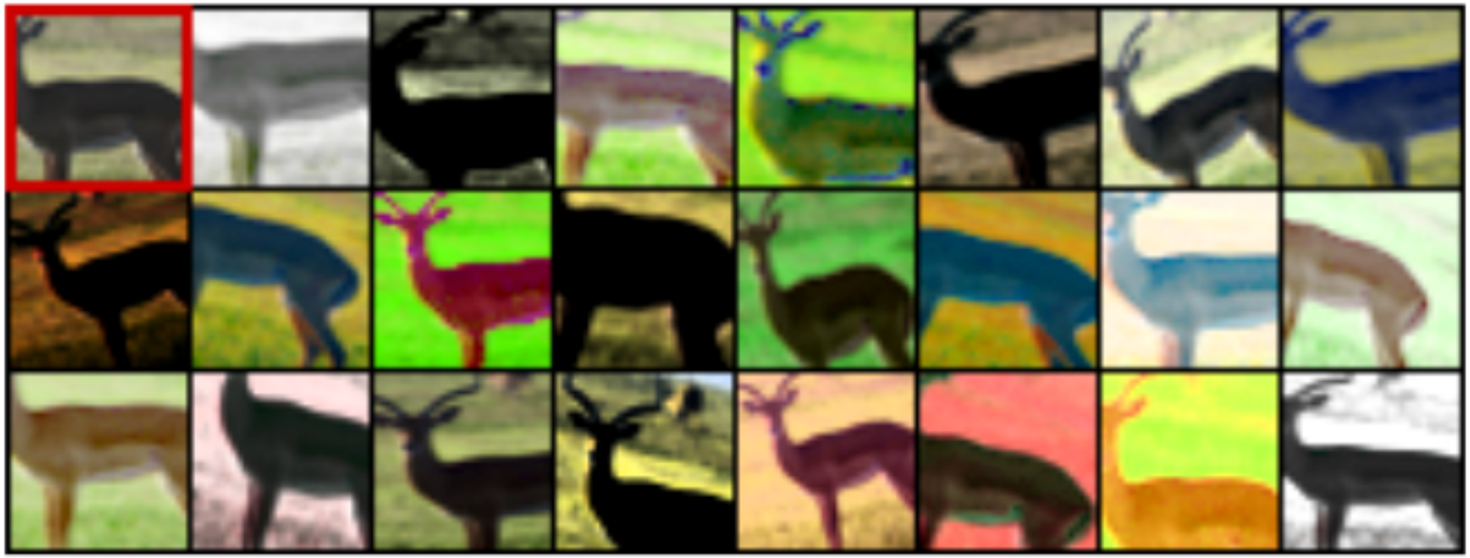

Compress an image and then reconstruct it.

→ Similarly to method of dimensionality reduction like PCA
→ It must only keep important features.

Image from the excellent [blog of Lilian Weng](#).

# Constrastive

**Each image is considered as a class.**
New samples of this “class” are generated with heavy data augmentation.
Trained with usual softmax + cross-entropy.

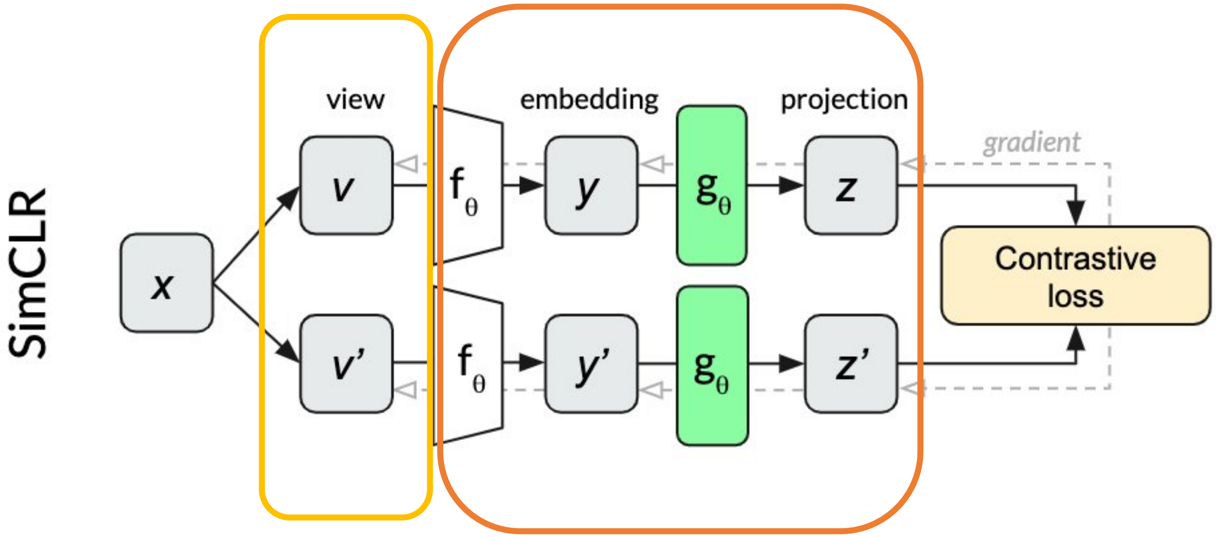


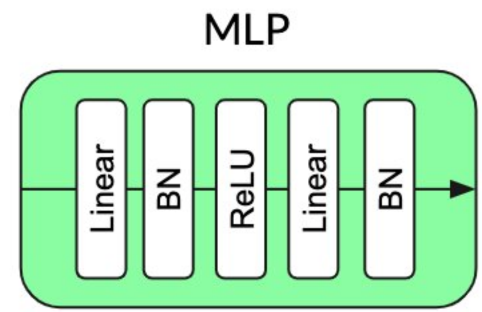
Most of future self-supervised models also try to:
- Bring together the same image augmented differently
- Push away all others images

[Dosovitskiy et al. NeurIPS 2014]

# SimCRL

**Batch of images is augmented twice differently**




**Extract features with ConvNet $f_\theta$, then project it with a small MLP to produce $z \in \mathbb{R}^d$**

Alternative version of the constrastive loss.
→ Bring together the same image augmented differently
→ Push away all others images of the batch

$$-\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq i]} \exp(s_{i,k}/\tau)}$$

[Chen et al. ICML 2020], image from Raffin's twitter thread.

# SimCRL

Alternative version of the constrastive loss.

$$-\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$$

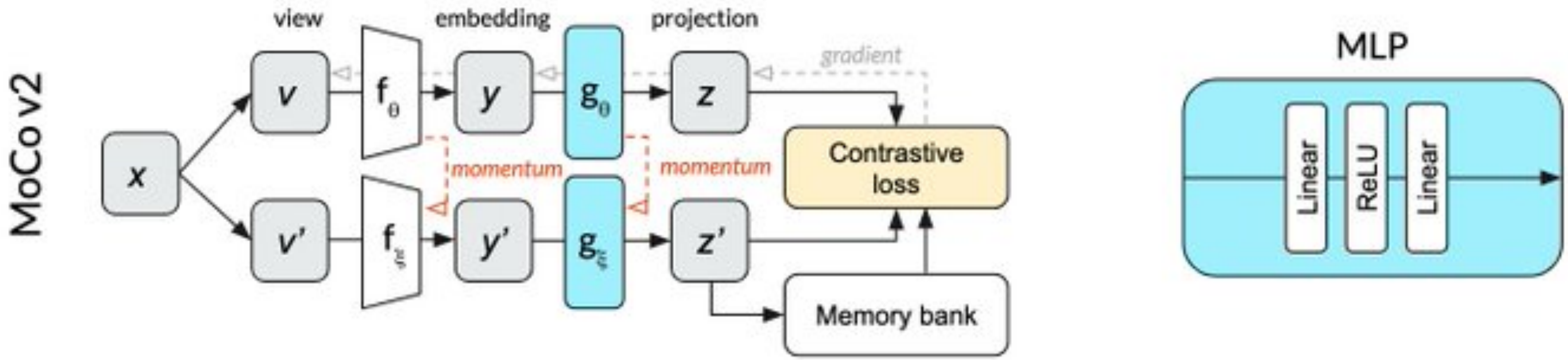
→ Bring together the same image augmented differently
→ Push away all others images of the batch

**It needs a very large batch size >= 1024!**

**The MLP that does the projection is essential.**
→ Learns useful transformations for the contrastive task
→ But it's discarded during the finetuning phase

[Chen et al. ICML 2020]

# MoCo

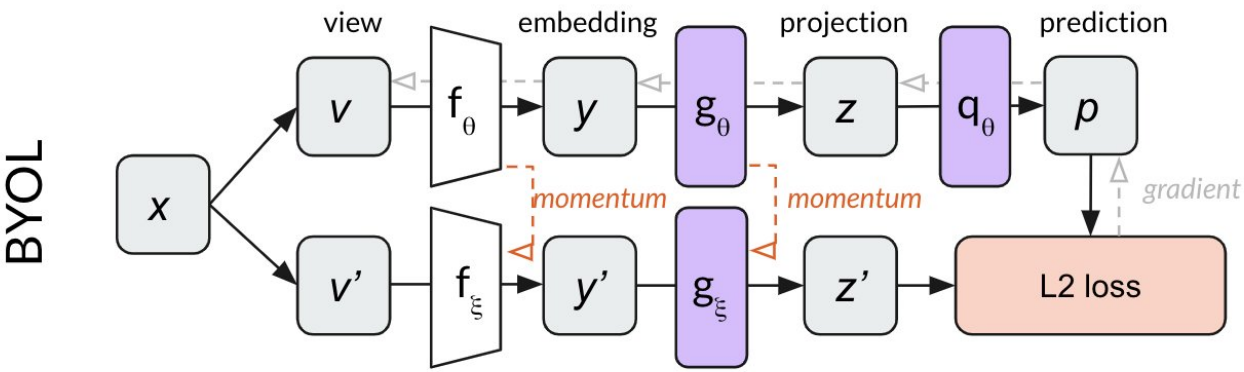


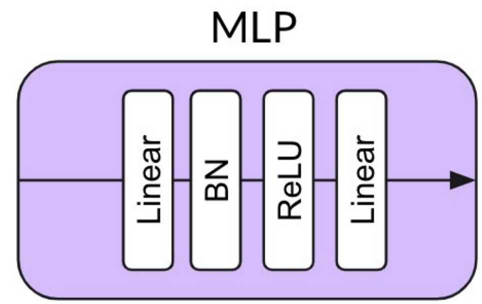

Reduces the need of large batch size with a **memory bank**:
→ Stores previously computed projections $z'$
→ Means more negative in the contrastive loss

The gradient is backpropagated only through one version of the network:
→ The other network is, as in RL, a **target network**
→ It is updated with momentum $\theta_t \leftarrow \alpha\, \theta_t + (1-\alpha)\, \theta_s$
→ Enforce stability in the memory bank representations

[Chen et al. arXiv 2020], image from Raffin's twitter thread.

# BYOL




L2 distance between **only positive examples**, **not negative examples are used**!

Why does the representation do not collapse?
→ Meaning only producing a zero vector for any input would minimize the loss

Still an active area of research, but some intuitions:
→ Asymmetrical architecture with another MLP $q_\theta$
→ Momentum for the target network

[Grill et al. NeurIPS 2020] but extremely similar to Mean Teacher [Tarvainen and Valpola, NeurIPS 2017], image from Raffin's twitter thread.

# What do we actually need for self-supervision?

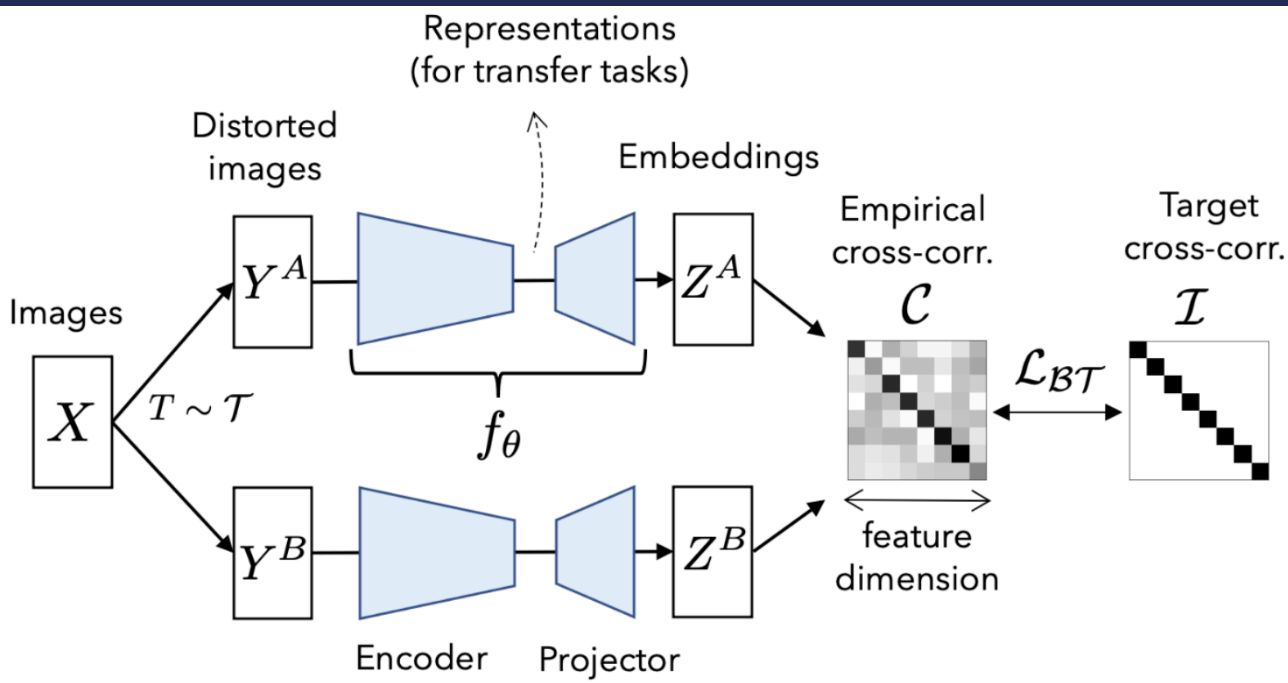
**Invariance to transformations**:

→ Two augmentations of the same image should produce the same representation

**Disentangling of the dimensions**:

→ Each dimension of the representation should encode a different info

# Barlow Twins




Cross-correlation along the batch dimension:

$$\mathcal{C}_{ij} \triangleq \frac{\sum_b z^A_{b,i} z^B_{b,j}}{\sqrt{\sum_b \left(z^A_{b,i}\right)^2} \sqrt{\sum_b \left(z^B_{b,j}\right)^2}}$$

$$\mathcal{L_{BT}} \triangleq \underbrace{\sum_i (1 - \mathcal{C}_{ii})^2}_{\text{invariance term}} + \lambda \underbrace{\sum_i \sum_{j \neq i} {\mathcal{C}_{ij}}^2}_{\text{redundancy reduction term}}$$

**Invariance to transformations**:

→ First term forces each dimension $i$ from both views to be very correlated (+1) despite the views were generated by different transformations

**Disentangling of the dimensions**:

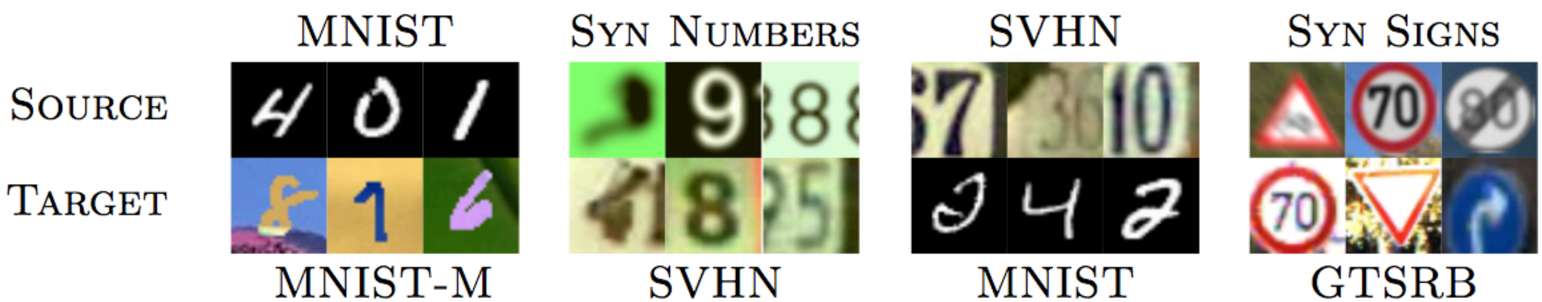
→ Second term forces each dimension $i$ to be orthogonal (0) with dimension $j \neq i$ so that each dimension encodes a different information, aka no collapse

[Zbontar and Jing et al. ICML 2021]

# Domain Adaptation

# Setting

- Source domain/dataset is fully labeled
- Target domain/dataset is unlabeled
- Both represent the same classes
- Huge discrepancy in the pixels distribution



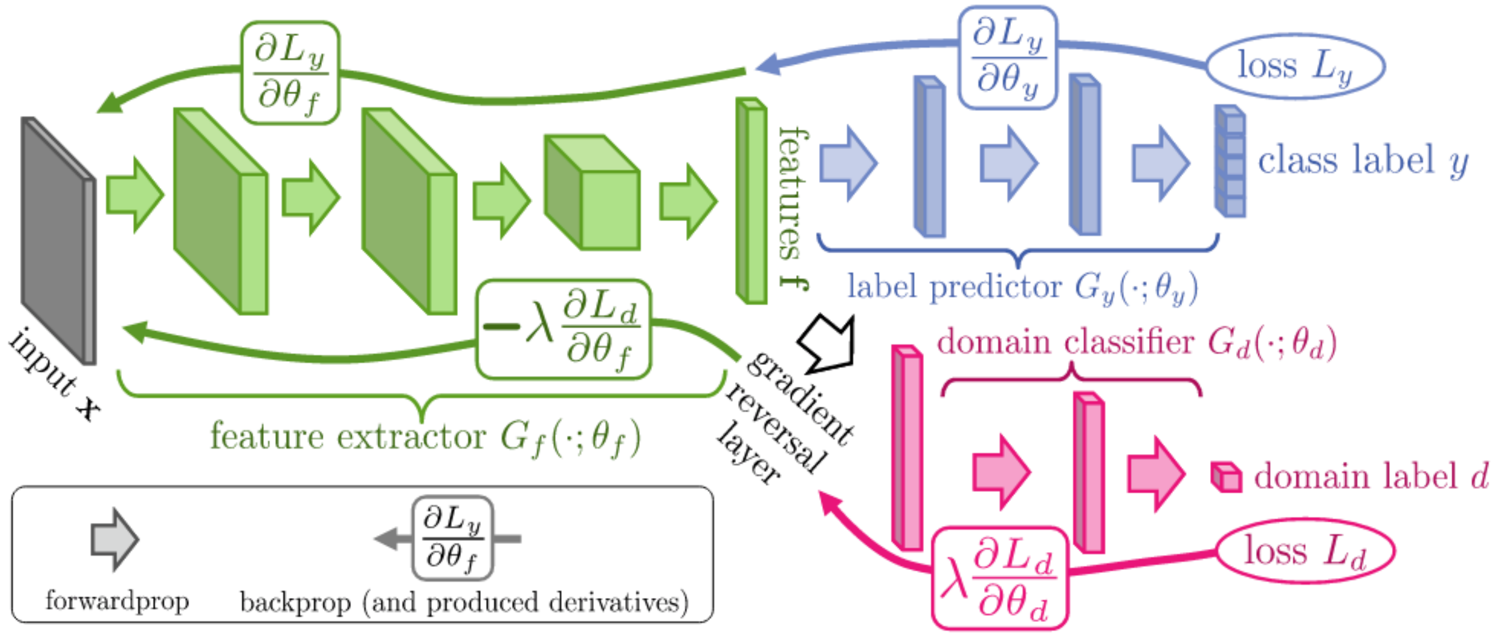

Source Domain

GTA5 (yes the game)

Target Domain

Cityscapes

# DANN: Gradient Reversal Layer

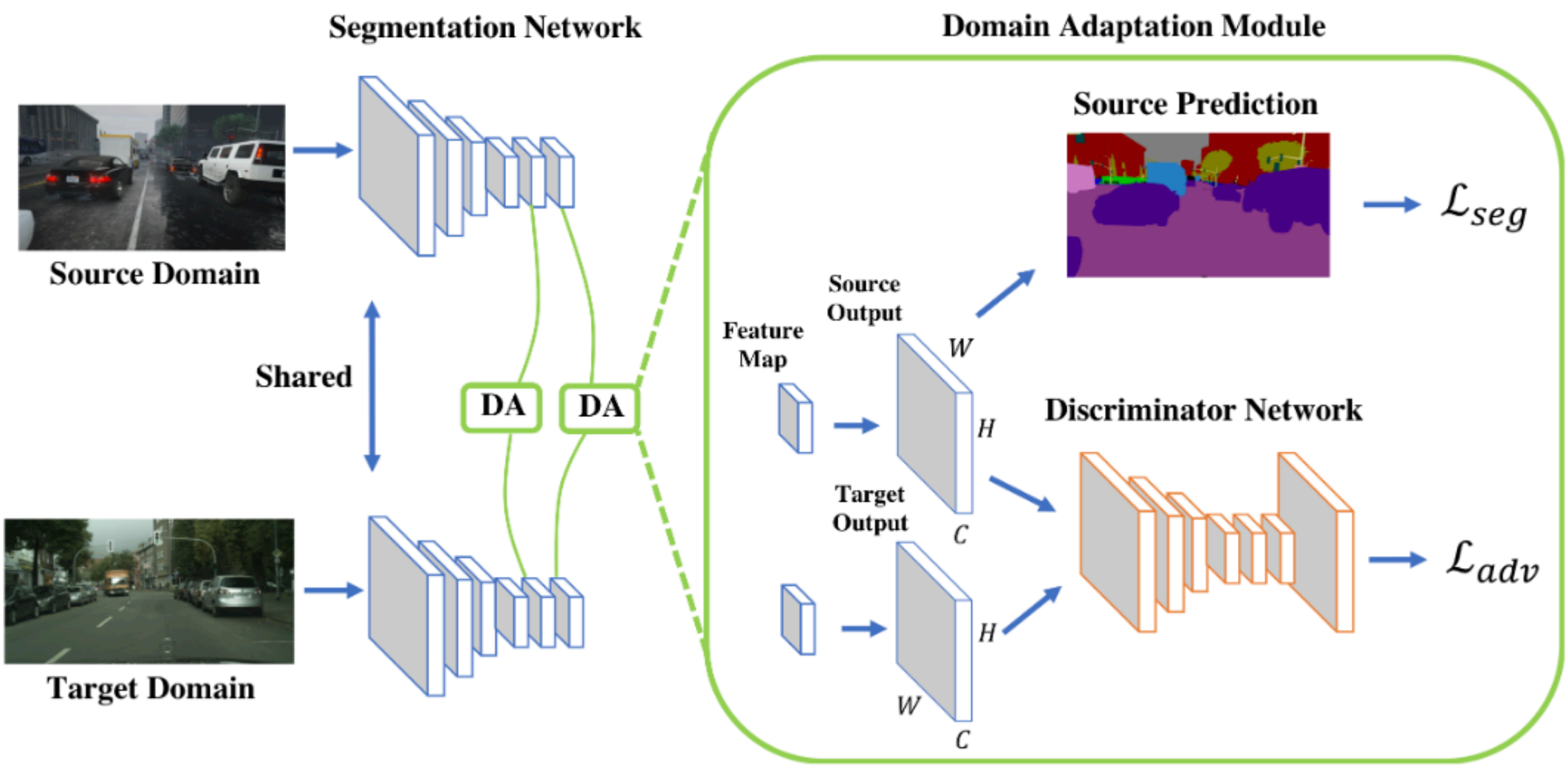



**Gradient Reversal Layer** (GRL) forces the ConvNet to maximize the loss of the Domain Classifier.

→ Force to learn domain agnostic features

[Ganin et al. JMLR 2015]

# AdaptSegNet




- Train the Discriminator on the probabilities of the source and target without the gradient flowing backward
- Train the Segmentation Network on source for classification and also force the discriminator to predict source given target images

$$\max_{\mathbf{D}} \min_{\mathbf{G}} \mathcal{L}(I_s, I_t). \qquad \mathcal{L}(I_s, I_t) = \mathcal{L}_{seg}(I_s) + \lambda_{adv}\mathcal{L}_{adv}(I_t)$$

[Tsai et al. CVPR 2018]

# Two key ideas

Two key ideas for domain adaption in segmentation:

1. **Adversarial loss** forcing a similar representation for both source and target domains

2. **Pseudo-labeling** to generates labels for the unlabeled target domain

# Other Problems

# Not covered in this lecture

**Zero-shot Learning**:
→ Not a single image of the class to predict, but access to metadata
→ *Ex: understand a Wikipedia description to classify a never-seen before animal*

**Semi-Supervised:**
→ A few amount (~10%) of the data is labeled, while the remaining is unlabeled but present
→ Most of the recent self-supervision literature took a lot of inspiration from it
→ Often solved with contrastive, weights averaging, and pseudo-labeling with consistency

**Weak supervision:**
→ Labels are imperfect
→ *Ex: training a model to predict the hashtag on Instagram photos*

Small break,
then coding session!