Challenging deep learning models in real-world applications

Learning with few or no labeled data

Céline Hudelot

February 10, 2023

Sources and references

- Slides from Yassine Ouali, PhD student, MICS Laboratory, *Label efficient Deep Learning*.
- ICCV Tutorial 2019 : *Learning with Limited Labels*[1]
- CVPR 2021 Tutorial : *Data- and Label-Efficient Learning in An Imperfect World*[2]
- Deep Learning Course, CentraleSupélec, Hervé Le Borgne - Frugality in AI.
- Lilian Weng Blog : `https://lilianweng.github.io/`

[1] `https://sites.google.com/view/learning-with-limited-data/schedule`
[2] `https://vita-group.github.io/cvpr_2021_data_efficient_tutorial.html`

Overview

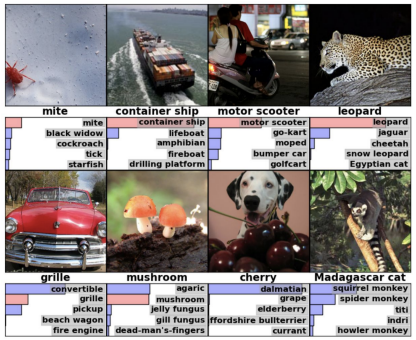
Data is fueling deep learning !

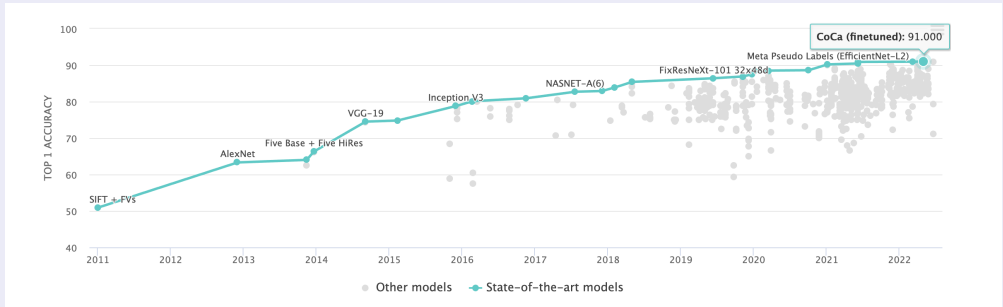
Training accurate deep models usually requires lots of well-labeled data

ImageNet Challenge

IMAGENET

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.





Source : `https://paperswithcode.com/sota/image-classification-on-imagenet`

(Yu et al, 2022) CoCa: Contrastive Captioners are Image-Text Foundation Models : an encoder-decoder architecture trained with both **contrastive loss** and captioning (generative) loss. It uses a pre-trained image-transformer (VIT) on ImageNet-21K [3](14,197,122 images, divided into 21,841 classes).

[3] `https://arxiv.org/pdf/2104.10972.pdf`

Training accurate deep models

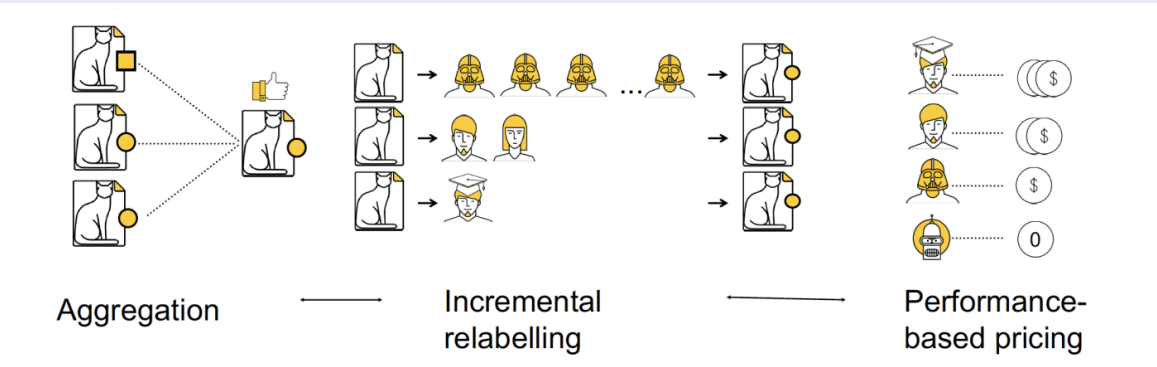
Usually requires lot of well-labeled data

- Data collection and annotation is expensive, tedious, time consuming.
  - e.g. Average time taken to label a single image in the Cityscapes dataset is 90 minutes.

Training accurate deep models

Usually requires lots of well-labeled data

- Data collection and annotation is expensive, tedious, time consuming.
  - e.g. Average time taken to label a single image in the Cityscapes dataset is 90 minutes.
  - Often involves crowdsourcing[a]



Source : (Drutsa et al) Practice of Efficient Data Collection via Crowdsourcing at Large-Scale[b]

[a] (Kovashka et al, 2016) Crowdsourcing in Computer Vision

[b] https://arxiv.org/abs/1912.04444 and https://research.yandex.com/tutorials/crowd/kdd-2019

Training accurate deep models

Usually requires lots of well-labeled data

- Data collection and annotation is expensive, tedious, time consuming.
- Crowd-sourcing may be infeasible for proprietary data.

Training accurate deep models

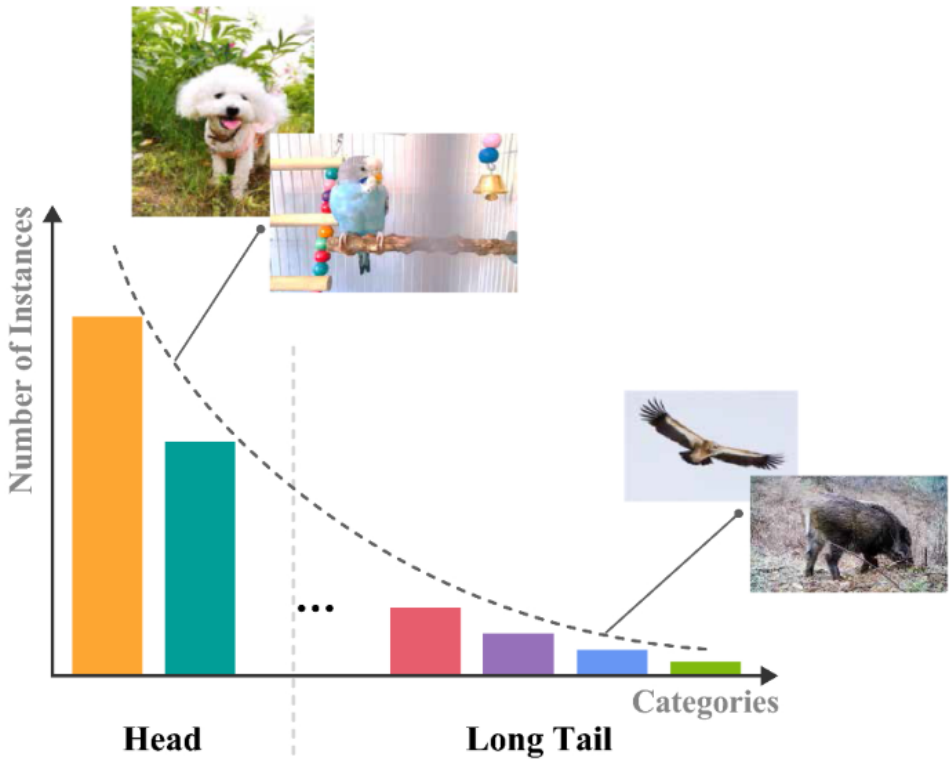
Usually requires lots of well-labeled data

- Data collection and annotation is expensive, tedious, time consuming.
- Crowd-sourcing may be infeasible for proprietary data.
- For some tasks, data may not be available at all (long tail distribution)

Training accurate deep models

Usually requires lots of well-labeled data

- Data collection and annotation is expensive, tedious, time consuming.
- Crowd-sourcing may be infeasible for proprietary data.
- For some tasks, data may not be available at all (long tail distribution)



Training accurate deep models

Usually requires lots of well-labeled data

- Data collection and annotation is expensive, tedious, time consuming
- Crowd-sourcing may be infeasible for proprietary data.
- For some tasks, data may not be available at all (long tail distribution)

Assuming the availability of large well-labeled dataset is not a realist assumption

Availability of labeled data

Synset: mushroom
Definition: any of various fleshy fungi of the subdivision Basidiomycota consisting of a cap at the end of a stem arising from an underground mycelium.
Popularity percentile:: 84%
Depth in WordNet: 7

Synset: mushroom
Definition: mushrooms and related fleshy fungi (including toadstools, puffballs, morels, coral fungi, etc.).
Popularity percentile:: 82%
Depth in WordNet: 8

Synset: mushroom
Definition: fleshy body of any of numerous edible fungi.
Popularity percentile:: 82%
Depth in WordNet: 6

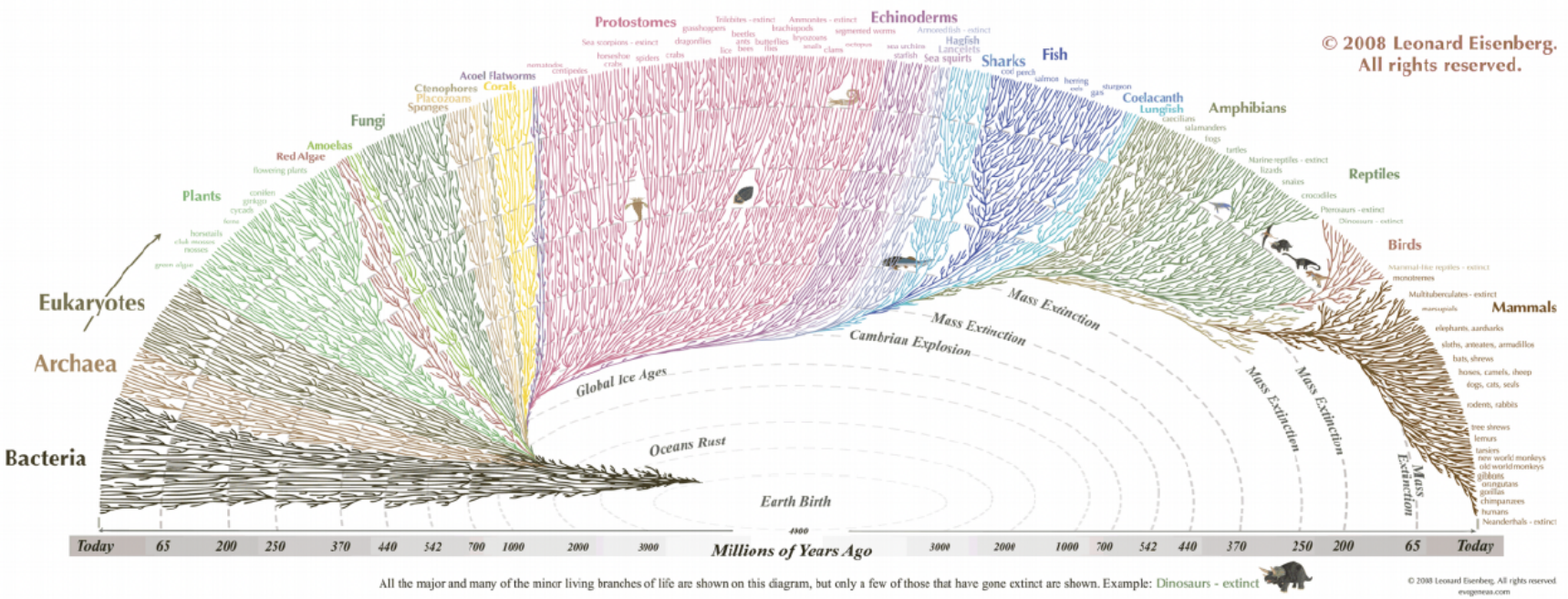
Synset: stuffed mushroom
Definition: mushrooms stuffed with any of numerous mixtures of e.g. meats or nuts or seafood or spinach.
Popularity percentile:: 69%
Depth in WordNet: 8

Synset: mushroom sauce
Definition: brown sauce and sauteed mushrooms.
Popularity percentile:: 69%
Depth in WordNet: 9

ImageNet has 30 mushroom synsets, each with ≈1000 images.

Slide credit: Christoph Lampert

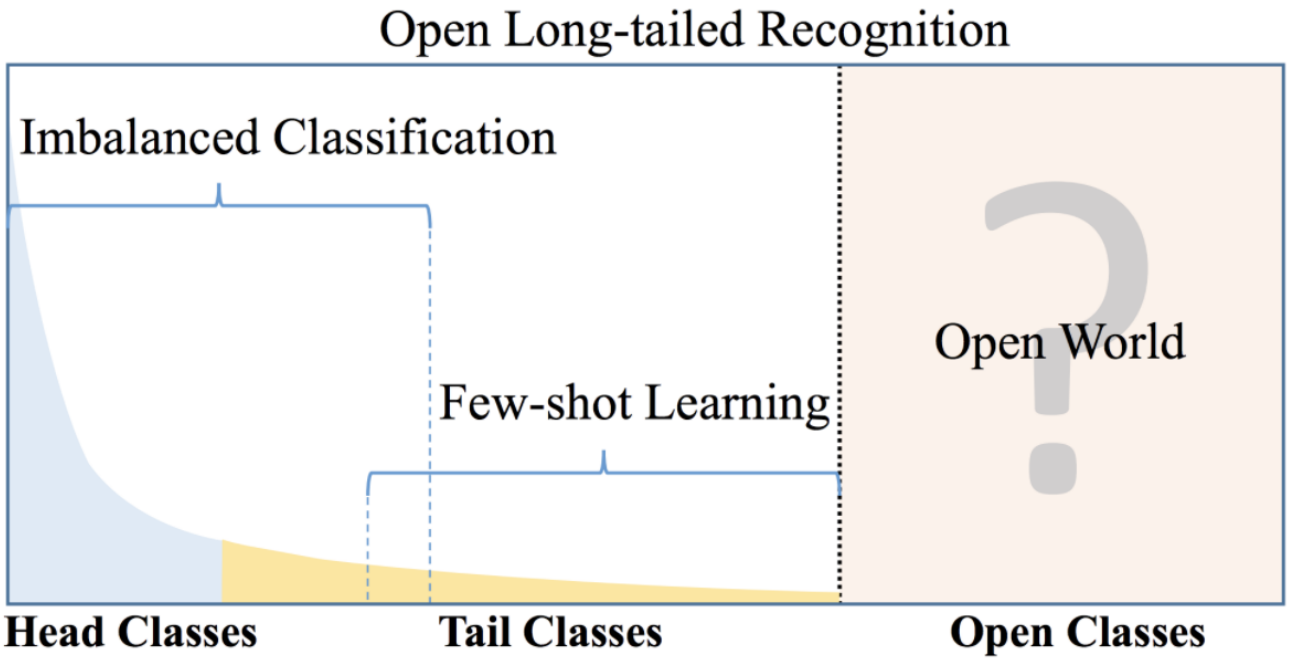
Availability of labeled data



In nature, there are ≈14,000 mushroom species !

Availability of labeled data

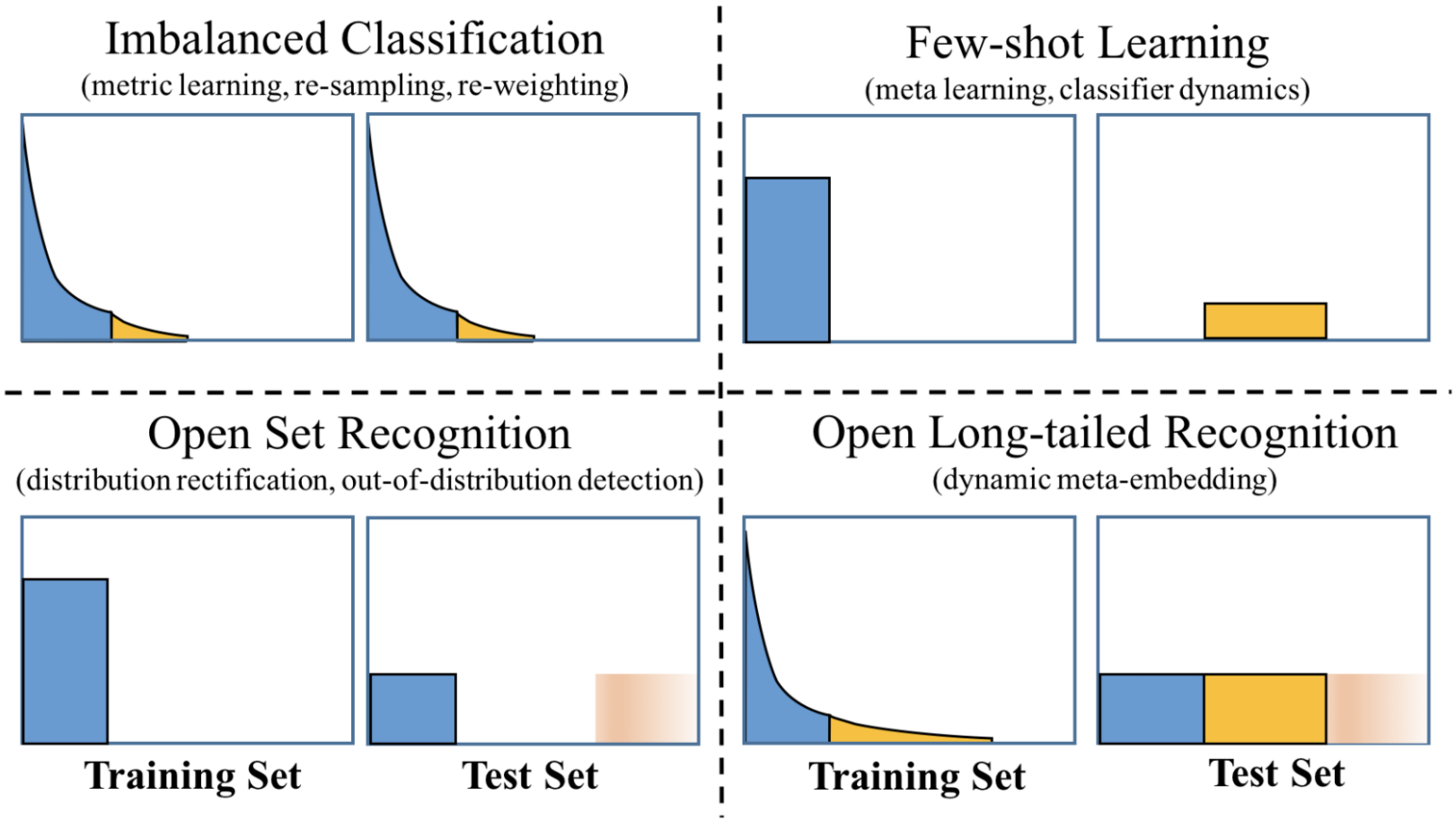
Large-Scale Long-Tailed Recognition in an Open World : Existing Computer Vision Setting v.s. Real-World Scenario



Source : `https://bair.berkeley.edu/blog/2019/05/13/oltr/`

Availability of labeled data

Large-Scale Long-Tailed Recognition in an Open World : Existing Computer Vision Setting v.s. Real-World Scenario



Source : `https://bair.berkeley.edu/blog/2019/05/13/oltr/`

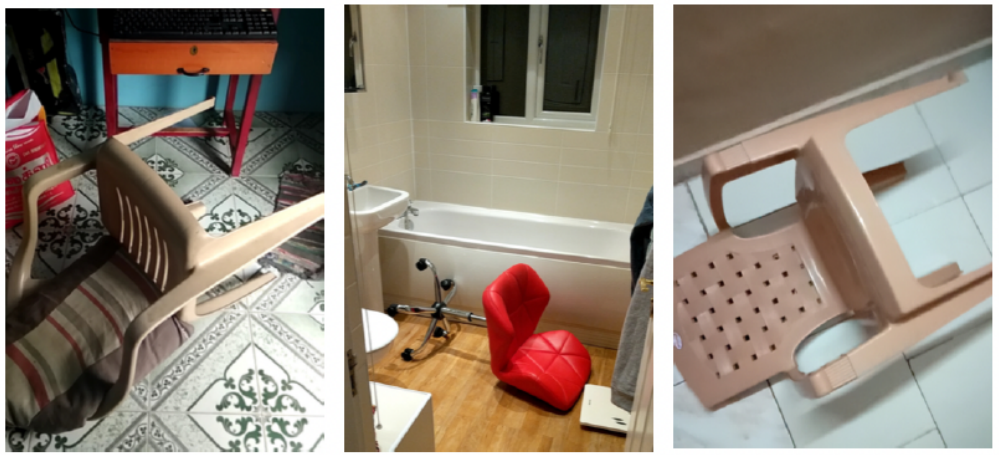
Availability of labeled data

Problem of bias in dataset

Classes with many examples may still suffer from bias !

ImageNet chairs

- Few rotations/viewpoints
- Typical backgrounds



Images from the ObjectNet dataset [Barbu et al., NeurIPS 2019]

ImageNet classifier fails to detect these chairs

See (Torralba et al, 2011) Unbiased Look at Dataset Bias[4]

[4] https://people.csail.mit.edu/torralba/publications/datasets_cvpr11.pdf

How to learn with limited labeled data ?

Several strategies

- **Data augmentation.**

How to learn with limited labeled data ?

Several strategies

- **Data augmentation.**
  - Leveraging ad-hoc input transformations that preserve corresponding output labels

How to learn with limited labeled data ?

Several strategies

- **Data augmentation.**
  - Leveraging ad-hoc input transformations that preserve corresponding output labels
  - By using generative models

How to learn with limited labeled data ?

Several strategies

- **Data augmentation.**
  - Leveraging ad-hoc input transformations that preserve corresponding output labels
  - By using generative models
- **By learning a pre-trained representation and transfer learning.**

How to learn with limited labeled data ?

Several strategies

- **Data augmentation.**
  - Leveraging ad-hoc input transformations that preserve corresponding output labels
  - By using generative models
- **By learning a pre-trained representation and transfer learning.**
  - Supervised : but we need annotated data for the source domain.

How to learn with limited labeled data ?

Several strategies

- **Data augmentation.**
  - Leveraging ad-hoc input transformations that preserve corresponding output labels
  - By using generative models
- **By learning a pre-trained representation and transfer learning.**
  - Supervised : but we need annotated data for the source domain.
  - Unsupervised or self-supervised : need to define a good pre-text task.

How to learn with limited labeled data ?

Several strategies

- **Data augmentation.**
  - Leveraging ad-hoc input transformations that preserve corresponding output labels
  - By using generative models
- **By learning a pre-trained representation and transfer learning.**
  - Supervised : but we need annotated data for the source domain.
  - Unsupervised or self-supervised : need to define a good pre-text task.
- **Weak (or noisy) supervision.**

How to learn with limited labeled data ?

Several strategies

- **Data augmentation.**
  - Leveraging ad-hoc input transformations that preserve corresponding output labels
  - By using generative models
- **By learning a pre-trained representation and transfer learning.**
  - Supervised : but we need annotated data for the source domain.
  - Unsupervised or self-supervised : need to define a good pre-text task.
- **Weak (or noisy) supervision.**
  - Leveraging web-based images or image captioning.

How to learn with limited labeled data ?

Several strategies

- **Data augmentation.**
  - Leveraging ad-hoc input transformations that preserve corresponding output labels
  - By using generative models
- **By learning a pre-trained representation and transfer learning.**
  - Supervised : but we need annotated data for the source domain.
  - Unsupervised or self-supervised : need to define a good pre-text task.
- **Weak (or noisy) supervision.**
  - Leveraging web-based images or image captioning.
  - Annotation for other tasks : classification labels used for segmentation.

How to learn with limited labeled data ?

Several strategies

- **Data augmentation.**
  - Leveraging ad-hoc input transformations that preserve corresponding output labels
  - By using generative models
- **By learning a pre-trained representation and transfer learning.**
  - Supervised : but we need annotated data for the source domain.
  - Unsupervised or self-supervised : need to define a good pre-text task.
- **Weak (or noisy) supervision.**
  - Leveraging web-based images or image captioning.
  - Annotation for other tasks : classification labels used for segmentation.
- **Alternatives to supervised learning or specific learning setting.**

How to learn with limited labeled data ?

Several strategies

- **Data augmentation.**
  - Leveraging ad-hoc input transformations that preserve corresponding output labels
  - By using generative models
- **By learning a pre-trained representation and transfer learning.**
  - Supervised : but we need annotated data for the source domain.
  - Unsupervised or self-supervised : need to define a good pre-text task.
- **Weak (or noisy) supervision.**
  - Leveraging web-based images or image captioning.
  - Annotation for other tasks : classification labels used for segmentation.
- **Alternatives to supervised learning or specific learning setting.**
  - Semi-supervised learning, active learning, unsupervised learning

How to learn with limited labeled data ?

Several strategies

- **Data augmentation.**
  - Leveraging ad-hoc input transformations that preserve corresponding output labels
  - By using generative models
- **By learning a pre-trained representation and transfer learning.**
  - Supervised : but we need annotated data for the source domain.
  - Unsupervised or self-supervised : need to define a good pre-text task.
- **Weak (or noisy) supervision.**
  - Leveraging web-based images or image captioning.
  - Annotation for other tasks : classification labels used for segmentation.
- **Alternatives to supervised learning or specific learning setting.**
  - Semi-supervised learning, active learning, unsupervised learning
  - Few shot learning

How to learn with limited labeled data ?

Several strategies

- **Data augmentation.**
  - Leveraging ad-hoc input transformations that preserve corresponding output labels
  - By using generative models
- **By learning a pre-trained representation and transfer learning.**
  - Supervised : but we need annotated data for the source domain.
  - Unsupervised or self-supervised : need to define a good pre-text task.
- **Weak (or noisy) supervision.**
  - Leveraging web-based images or image captioning.
  - Annotation for other tasks : classification labels used for segmentation.
- **Alternatives to supervised learning or specific learning setting.**
  - Semi-supervised learning, active learning, unsupervised learning
  - Few shot learning
- **Using ad-hoc external data**

How to learn with limited labeled data ?

Several strategies

- **Data augmentation.**
  - Leveraging ad-hoc input transformations that preserve corresponding output labels
  - By using generative models
- **By learning a pre-trained representation and transfer learning.**
  - Supervised : but we need annotated data for the source domain.
  - Unsupervised or self-supervised : need to define a good pre-text task.
- **Weak (or noisy) supervision.**
  - Leveraging web-based images or image captioning.
  - Annotation for other tasks : classification labels used for segmentation.
- **Alternatives to supervised learning or specific learning setting.**
  - Semi-supervised learning, active learning, unsupervised learning
  - Few shot learning
- **Using ad-hoc external data**
  - Data from others modalities

How to learn with limited labeled data ?

Several strategies

- **Data augmentation.**
  - Leveraging ad-hoc input transformations that preserve corresponding output labels
  - By using generative models
- **By learning a pre-trained representation and transfer learning.**
  - Supervised : but we need annotated data for the source domain.
  - Unsupervised or self-supervised : need to define a good pre-text task.
- **Weak (or noisy) supervision.**
  - Leveraging web-based images or image captioning.
  - Annotation for other tasks : classification labels used for segmentation.
- **Alternatives to supervised learning or specific learning setting.**
  - Semi-supervised learning, active learning, unsupervised learning
  - Few shot learning
- **Using ad-hoc external data**
  - Data from others modalities
  - Data from a different domain.

Overview

Data Augmentation

Main assumption : more information can be extracted from the original dataset through augmentations.

Definition

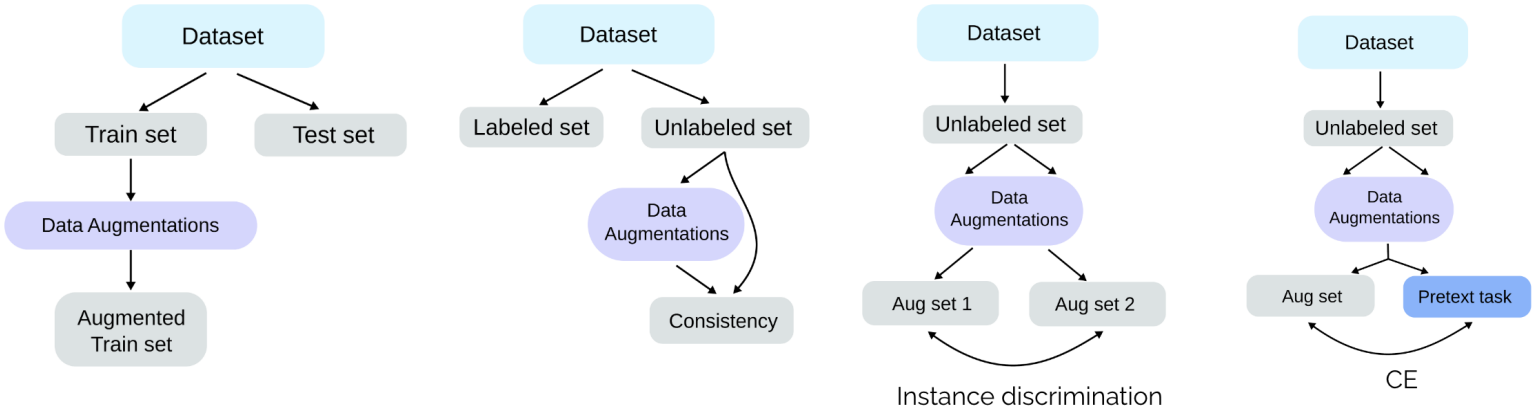
Data augmentation is a process of artificially increasing the amount of data by generating new data points from existing data.

- Best method to improve generalization = learn from more data
  - Allow a better sampling of the unknown data distribution.
- Answer (partially) to the practical limitation of data availability.
- A very common strategy for image classification:
  - The label must be robust to transformations of the input data x.

Data Augmentation

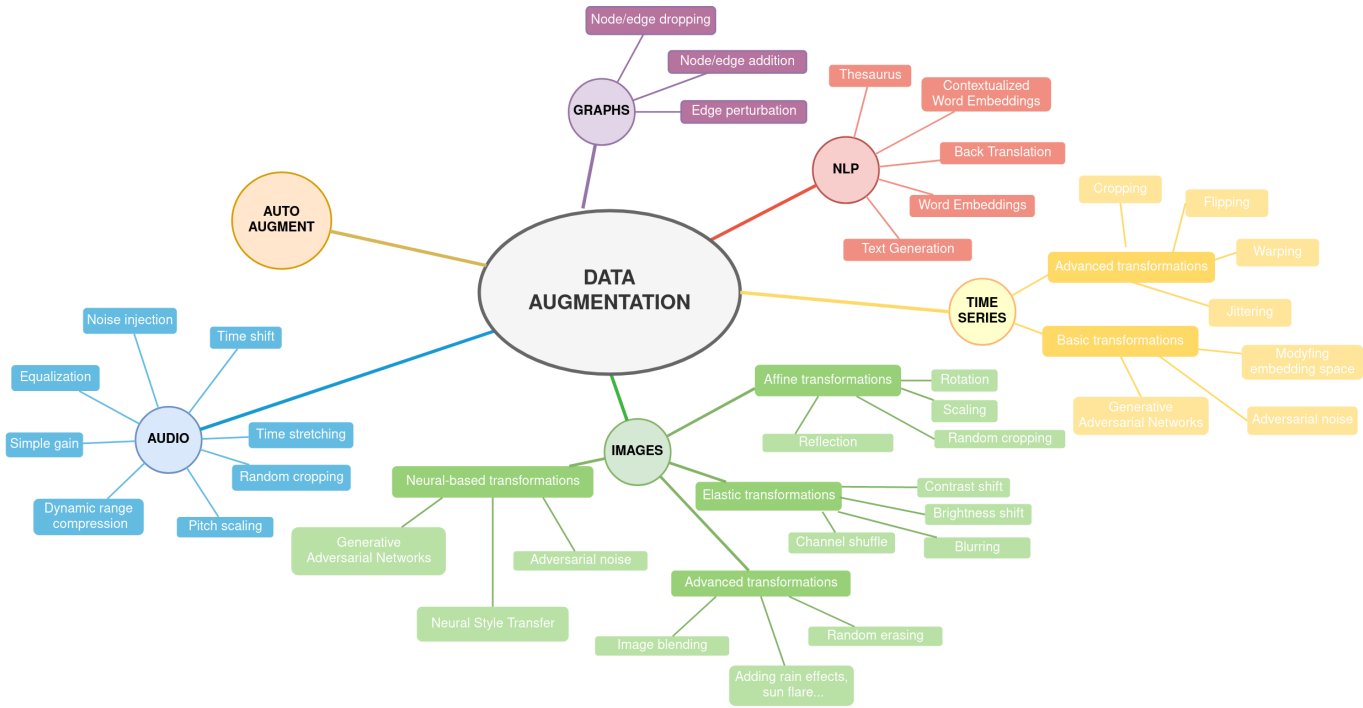
Data augmentation plays a critical role in deep learning:

- Increases the diversity of data, inducing a **regularization effect and a better generalization**.
- **Consistency**, can be used to extract training signal from unlabeled data.
- Instance discrimination, can be used for **contrastive pre-training**.
- **Self-supervision**, can be used to create pre-text tasks.



Data Augmentation

A common practice for all types of data



Source : `https://github.com/AgaMiko/data-augmentation-review`

Data Augmentation

A large catalog of approaches

Data Augmentation : Image Mixing

Principle

Blending two input images and their corresponding labels according to the following equations :

$$\tilde{x} = B \odot x_i + (I - B) \odot x_j$$

$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j$$

with x_i, x_j original input images, y_i, y_j respective one-hot label encodings, λ a mixing ratio, B a mixing mask matrix (for both pixel-wise and patch-wise mixing) and I the identity matrix of the same dimensionality of B.

Approaches are built around these equations and mainly differ by :

- the λ selection method
- Construction of matrix B.

Data Augmentation : Image Mixing

Core approach : mixup

(Zhang et al, ICLR 18) mixup: BEYOND EMPIRICAL RISK MINIMIZATION[a]

- Generates a weighted combinations of random image pairs from the training data.
- Given two images x_i, x_j and their ground truth labels y_i, y_j, a synthetic training example $(\hat{x}, \hat{y})$ is generated as:

$$\hat{x} = \lambda x_i + (1 - \lambda) x_j$$
$$\hat{y} = \lambda y_i + (1 - \lambda) y_j$$

[a] https://arxiv.org/pdf/1710.09412v2.pdf

Data Augmentation : Image Mixing

Core approach : mixup - From Empiral risk minimization to mixup

Given a training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ where $(x_i, y_i) \sim P$ for all $i = 1, ..., n$. We may approximate P by :

- Empirical distribution (Empirical Risk Minimization)

$$P_\delta(x, y) = \frac{1}{n} \sum_{i=1}^{n} \delta(x = x_i, y = y_i)$$

  with $\delta(x = x_i, y = y_i)$ a dirac centered at (x_i, y_i)

- Vicinal distribution (Vicinal Risk Minimization) (Chapelle, 2020)

$$P_\nu(\tilde{x}, \tilde{y}) = \frac{1}{n} \sum_{i=1}^{n} \nu(\tilde{x}, \tilde{y}|x_i, y_i)$$

  with ν a vicinity distribution, i.e. probability of finding the target pair $(\tilde{x}, \tilde{y})$ in the vicinity of (x_i, y_i)

- In practice, augmentation with a Gaussian Noise $\nu(\tilde{x}, \tilde{y}) = \mathcal{N}(\tilde{x} - x, \sigma^2)\delta(\tilde{y} = y)$ and empirical vicinal risk.
- mixup proposes a generic vicinal distribution

$$\mu(\tilde{x}, \tilde{y}|x_i, y_i) = \frac{1}{n} \sum_{i=j}^{n} \mathbb{E}_\lambda[\delta(\tilde{x} = \lambda.x_i + (1 - \lambda).x_j, \tilde{y} = \lambda.y_i + (1 - \lambda).y_j)]$$

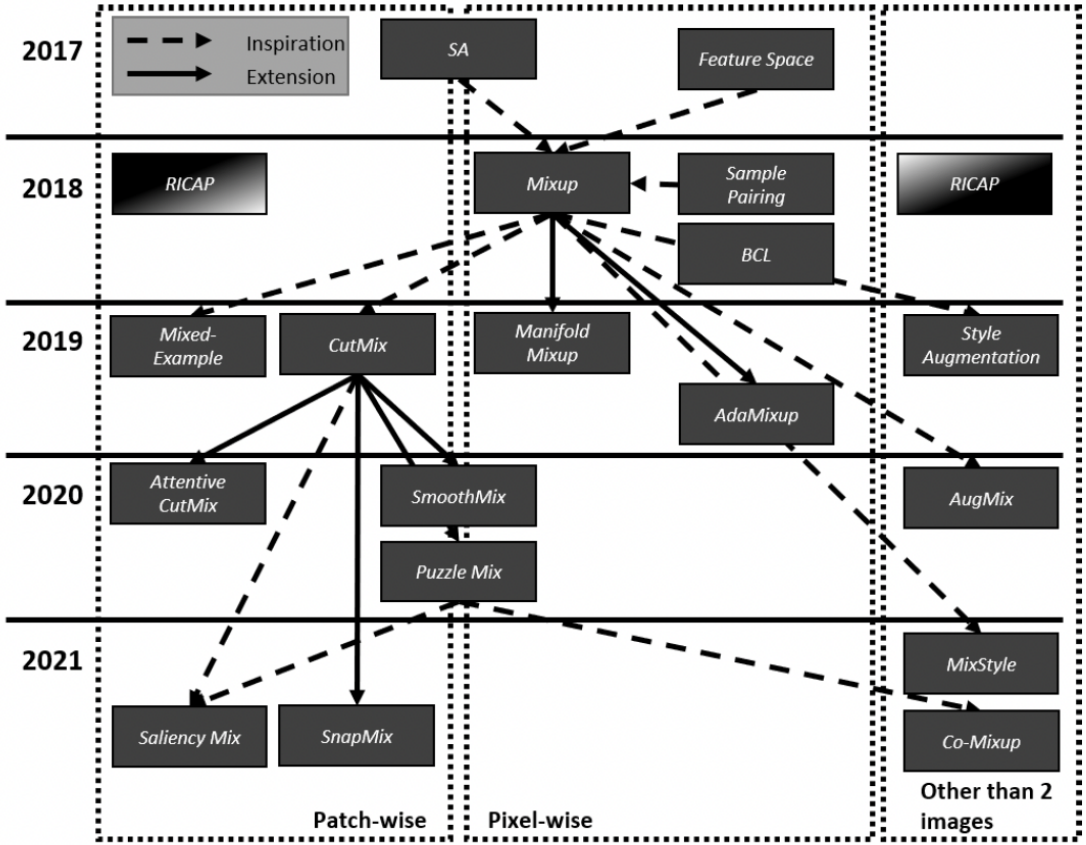
Data Augmentation : Image Mixing

Core approach : mixup - From Empiral risk minimization to mixup

- It introduces globally linear behavior in between training examples (generalization).
- Mixup trained models are relatively robust to input perturbations/corruptions and at same time are calibrated better than their non-mixup counterparts.
- Use in many works.

Data Augmentation : Image Mixing

A lot of other approaches



Source : (Lewy et al, 22) An overview of mixing augmentation methods and augmentation strategies[5]

[5] https://arxiv.org/pdf/2107.09887.pdf

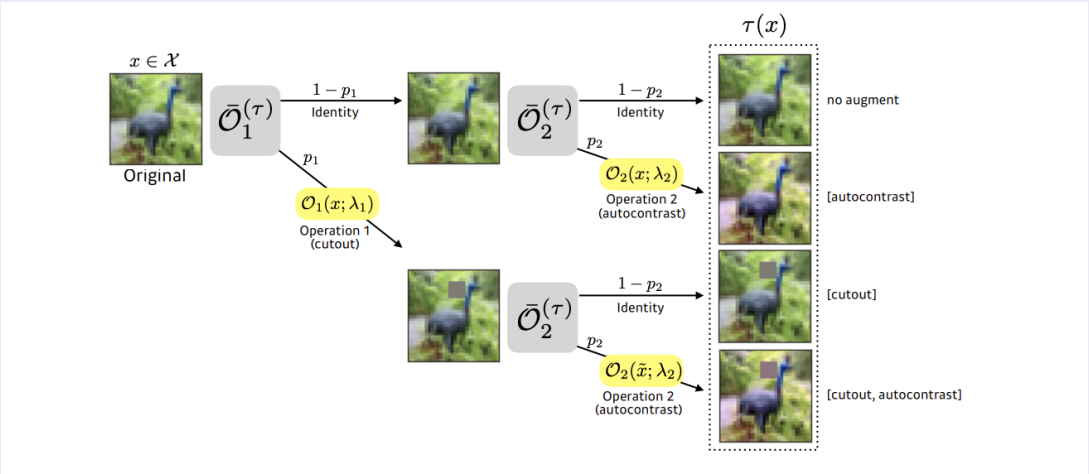
Data Augmentation : General image augmentation for DL

Recently, great advances in general purpose image augmentation for increasing quantity and diversity of training data.

AutoAugment (Cubuk et al, 2019)[a] , **Fast AutoAugment (Lim et al, 2019)[b]**

[a] https://arxiv.org/abs/1805.09501
[b] https://arxiv.org/abs/1905.00397

Formulate the problem of finding the best augmentation policy as a discrete search problem.



- $\mathbb{O}$: set of augmentation operations $\mathcal{O} : \mathcal{X} \to \mathcal{X}$ on the input space $\mathcal{X}$
- Each operation $\mathcal{O}$ has two parameters : calling probability p and magnitude λ (variability of the operation).
- $\mathcal{S}$ set of sub-policies τ, with τ that consists of N_τ consecutive operations applied sequentially to the image with probability p.

Data Augmentation : General image augmentation for DL

Fast AutoAugment (Lim et al, 2019)[6] : Search by density matching and bayesian optimisation

Search strategy

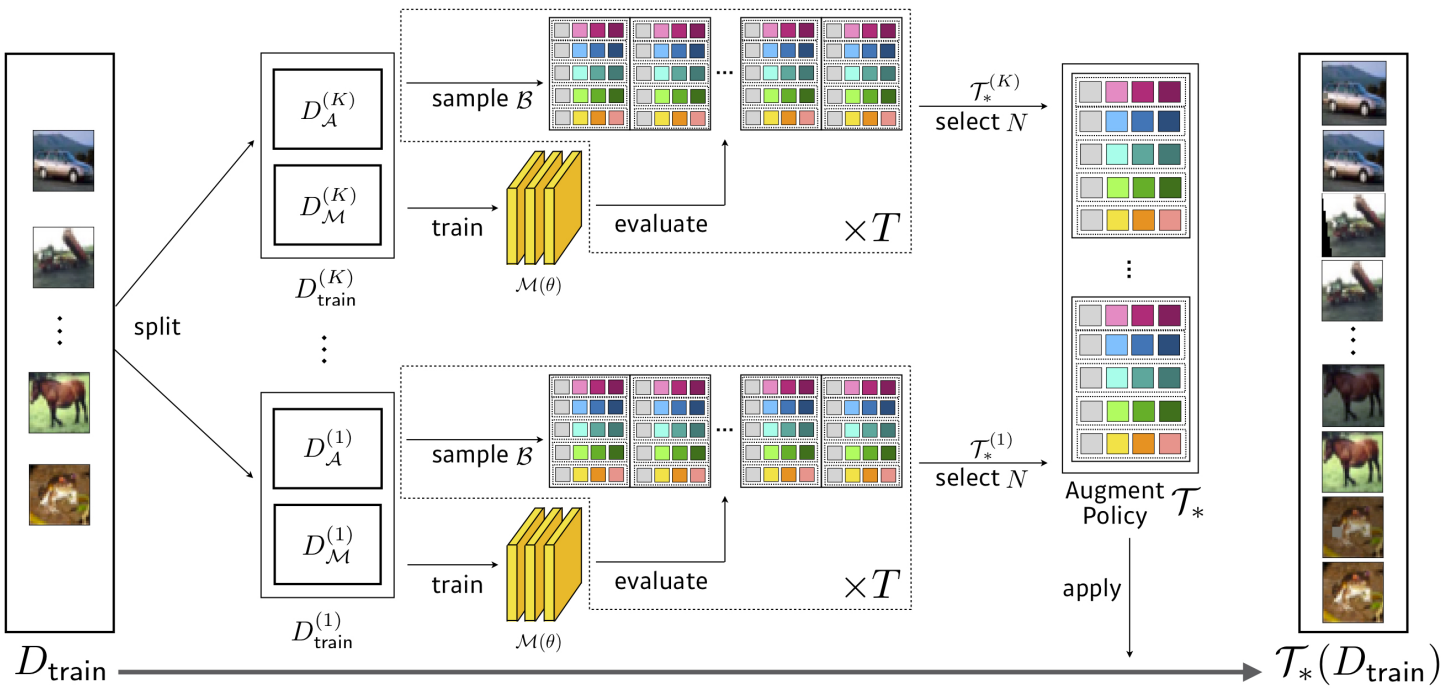
- Let D a probability distribution on $\mathcal{X} \times \mathcal{Y}$. We assume dataset $\mathcal{D}$ is sampled from this distribution.
- For a classification model $\mathcal{M}(.|\theta) : \mathcal{X} \rightarrow \mathcal{Y}$, expected accuracy : $\mathcal{R}(\theta|\mathcal{D})$ and expected loss $\mathcal{L}(\theta|\mathcal{D})$
- Goal: for a given pair $(\mathcal{D}_{\texttt{train}}, \mathcal{D}_{\texttt{valid}})$, improve the generalization ability by searching augmentation policies that match the density of $\mathcal{D}_{\texttt{train}}$ with density of augmented $\mathcal{D}_{\texttt{valid}}$: **search by density matching**
- It is evaluated by measuring how much one dataset follows the pattern of the other by using the model predictions on both datasets.
- Objective used :

$$\mathcal{T}_* = \texttt{argmax}_{\mathcal{T}} \mathcal{R}(\theta^*|\mathcal{T}(\mathcal{D}_A))$$

  where θ^* is trained on $\mathcal{D}_M$ with a splittting of $\mathcal{D}_{\texttt{train}}$ on $\mathcal{D}_M$ and $\mathcal{D}_A$
- Efficient strategy with a $k-$fold stratified shuffling : we train parameters on $\mathcal{D}_M$, evaluate them on $\mathcal{D}_A$ and we explore $\mathcal{B}$ candidate policies via Bayesian optimization (sample a sequence of sub-policies and tune their parameters to minimize the expected loss $\mathcal{L}(\theta||.)$ (exploration - exploitation).
- Select top$-N$ policies for each fold and merge them
- At the end, we augment the whole dataset $\mathcal{D}_{\texttt{train}}$ and retrain the model parameter.

Data Augmentation : General image augmentation for DL

Fast AutoAugment (Lim et al, 2019) : Search by density matching and bayesian optimisation



Algorithm 1: Fast AutoAugment

Input : $(\theta, D_{\text{train}}, K, T, B, N)$

Split D_{train} into K-fold data $D_{\text{train}}^{(k)} = \{(D_{\mathcal{M}}^{(k)}, D_{\mathcal{A}}^{(k)})\}$ // stratified shuffling

for $k \in \{1, \ldots, K\}$ **do**

  $\mathcal{T}_*^{(k)} \leftarrow \emptyset$, $(D_{\mathcal{M}}, D_{\mathcal{A}}) \leftarrow (D_{\mathcal{M}}^{(k)}, D_{\mathcal{A}}^{(k)})$ // initialize

  Train θ on $D_{\mathcal{M}}$

  for $t \in \{0, \ldots, T-1\}$ **do**

    $\mathcal{B} \leftarrow \text{BayesOptim}(\mathcal{T}, \mathcal{L}(\theta|\mathcal{T}(D_{\mathcal{A}})), B)$ // explore-and-exploit

    $\mathcal{T}_t \leftarrow$ Select top-N policies in $\mathcal{B}$

    $\mathcal{T}_*^{(k)} \leftarrow \mathcal{T}_*^{(k)} \cup \mathcal{T}_t$ // merge augmentation policies

return $\mathcal{T}_* = \bigcup_k \mathcal{T}_*^{(k)}$

Data Augmentation : GANs for data augmentation

(Shrivastava el al, 2017)[7] Learning from Simulated and Unsupervised Images through Adversarial Training (Best Paper CVPR 2017)

Simulated+Unsupervised (S+U) learning

Goal : improve the realism of synthetic images from a simulator using unlabeled real data while preserving the annotation.

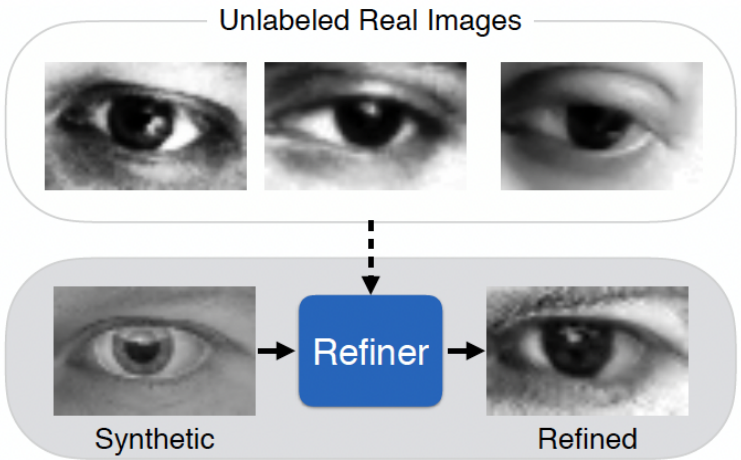


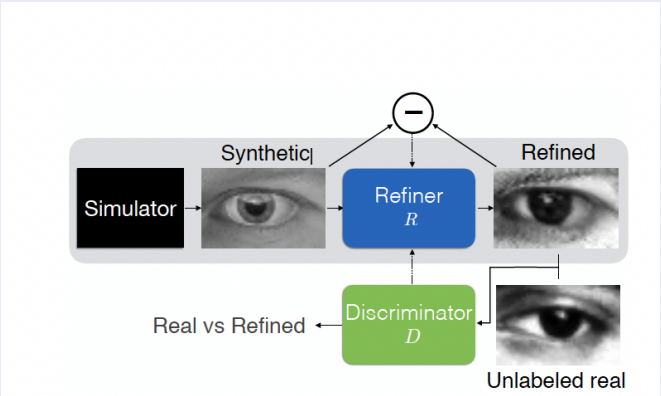
Figure 1. Simulated+Unsupervised (S+U) learning. The task is to learn a model that improves the realism of synthetic images from a simulator using unlabeled real data, while preserving the annotation information.

[7] https://arxiv.org/pdf/1612.07828.pdf

Data Augmentation : GANs for data augmentation

(Shrivastava el al, 2017) Learning from Simulated and Unsupervised Images through Adversarial Training (Best Paper CVPR 2017)

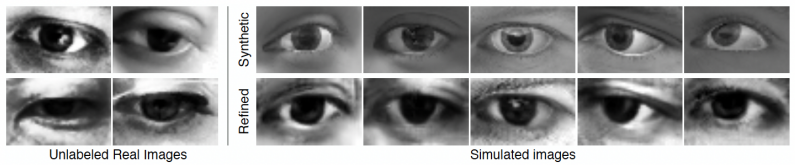
Simulated+Unsupervised (S+U) learning

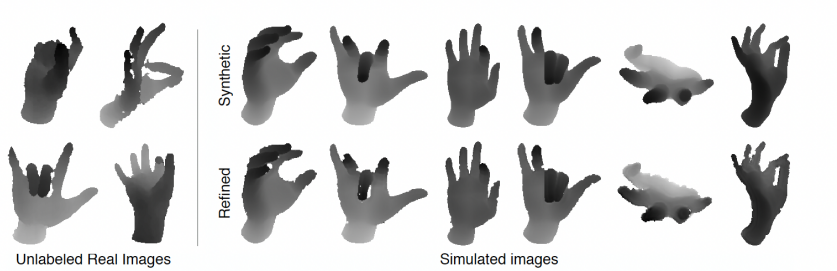


- Refiner network to add realism to synthetic images
- Discriminator classifies an image as real or refined
- Self-regularization term minimizes the image difference between the synthetic and the refined images
- The refiner is trained using a combination of an adversarial loss and a self-regularization loss

Data Augmentation : GANs for data augmentation

(Shrivastava el al, 2017) Learning from Simulated and Unsupervised Images through Adversarial Training (Best Paper CVPR 2017)





Data Augmentation : GANs for data augmentation

(Choi et al, 2020) StarGAN v2: Diverse Image Synthesis for Multiple Domains[8]

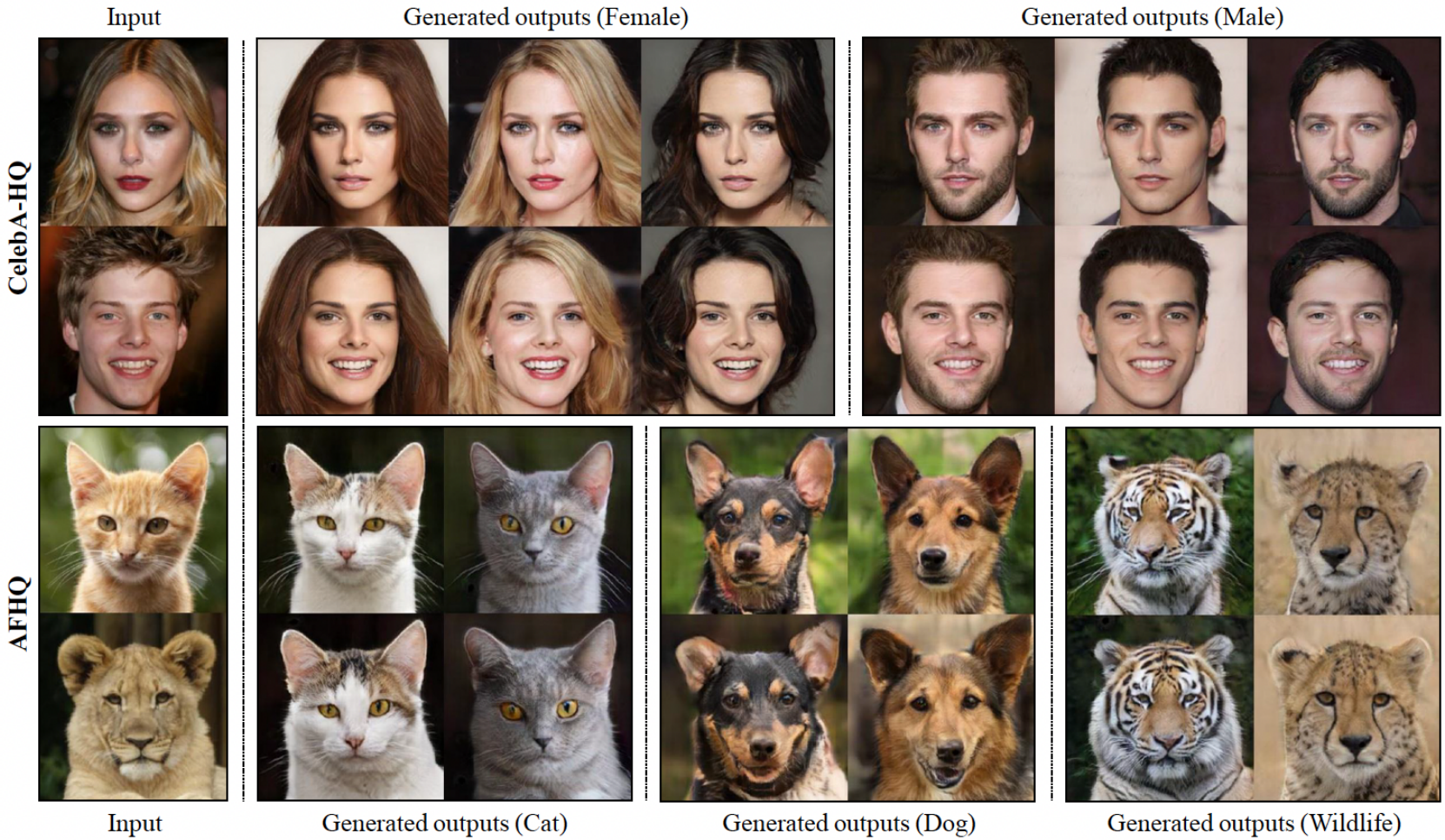
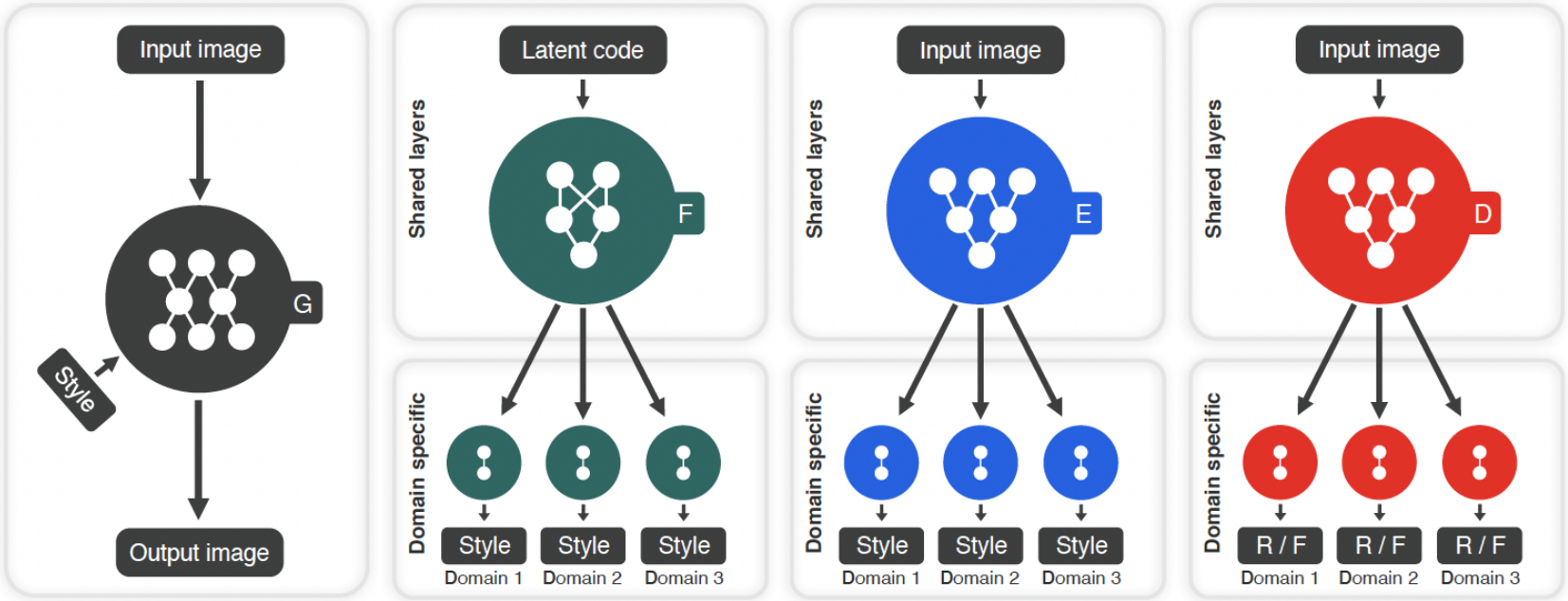


Figure 1. Diverse image synthesis results on the CelebA-HQ dataset and the newly collected animal faces (AFHQ) dataset. The first column shows input images while the remaining columns are images synthesized by StarGAN v2.

[8] https://openaccess.thecvf.com/content_CVPR_2020/papers/Choi_StarGAN_v2_Diverse_Image_Synthesis_for_Multiple_Domains_CVPR_2020_paper.pdf

Data Augmentation : GANs for data augmentation

(Choi et al, 2020) StarGAN v2: Diverse Image Synthesis for Multiple Domains[9]



(a) Generator (b) Mapping network (c) Style encoder (d) Discriminator

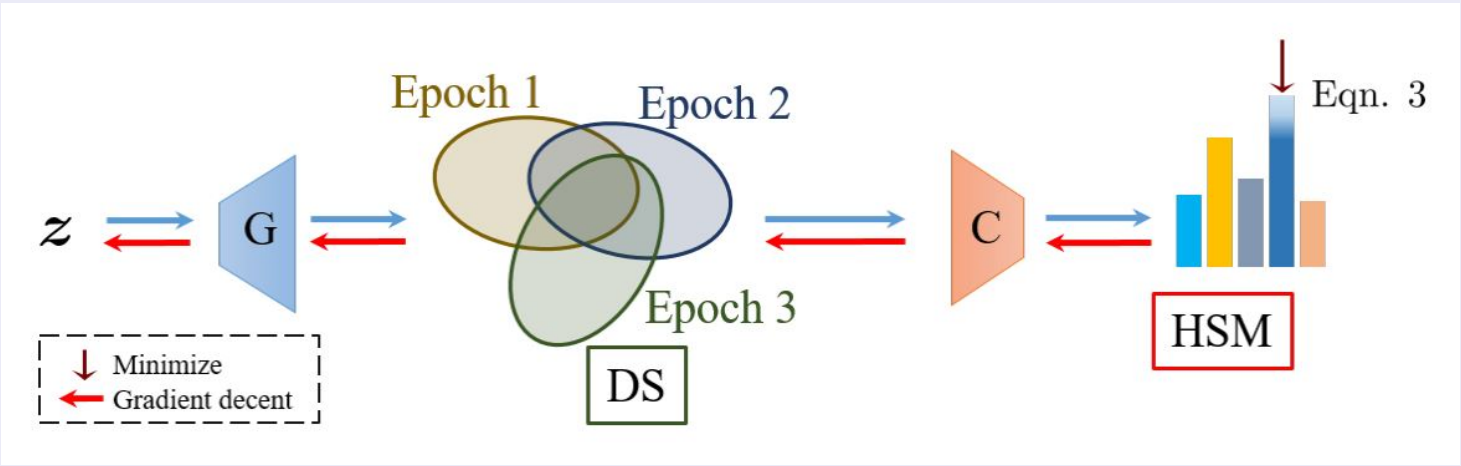
Figure 2. Overview of StarGAN v2, consisting of four modules. **(a)** The generator translates an input image into an output image reflecting the domain-specific style code. **(b)** The mapping network transforms a latent code into style codes for multiple domains, one of which is randomly selected during training. **(c)** The style encoder extracts the style code of an image, allowing the generator to perform reference-guided image synthesis. **(d)** The discriminator distinguishes between real and fake images from multiple domains. Note that all modules except the generator contain multiple output branches, one of which is selected when training the corresponding domain.

[9] https://openaccess.thecvf.com/content_CVPR_2020/papers/Choi_StarGAN_v2_Diverse_Image_Synthesis_for_Multiple_Domains_CVPR_2020_paper.pdf

Data Augmentation : GANs for data augmentation

Generated data : sufficient ?

(Besnier et al, ICASSP, 2020) THIS DATASET DOES NOT EXIST: TRAINING MODELS FROM GENERATED IMAGES[a]



- **Hard Sample Mining (HSM)** with latent code optimization : when training from generated images, we can use the generator to produce more informative images rather than drawing mere samples
  - The more informative or harder images are specific to the current state of the classifier C_θ^i
  - The hard images for C_θ^i are the images which are difficult to classify correctly by C_θ^i
  - Idea : optimize iteratively an original random latent code z to minimize the score of the class predicted by C_θ^i
- **Dataset smoothing** : progressively change the dataset by only partially replacing the generated training data with new samples every epoch, aiming for a diverse but gradually changing dataset

[a] https://valeoai.github.io/blog/publications/gan-dataset/

Data Augmentation : GANs for data augmentation

Generated data : sufficient ?

(Besnier et al, ICASSP, 2020) THIS DATASET DOES NOT EXIST: TRAINING MODELS FROM GENERATED IMAGES[a]

	Fixed dataset				Continuous sampling				Long training	real images
HSM	-	✓	-	✓	-	✓	-	✓	-	
DS	-	-	-	-	✓	✓	✓	✓	✓	
BNA	-	-	✓	✓	-	-	✓	✓	✓	
Top-1 Accuracy	70.6	73.6	76.4	78.2	78.6	81.8	84.2	**85.6**	**88.8**	88.4

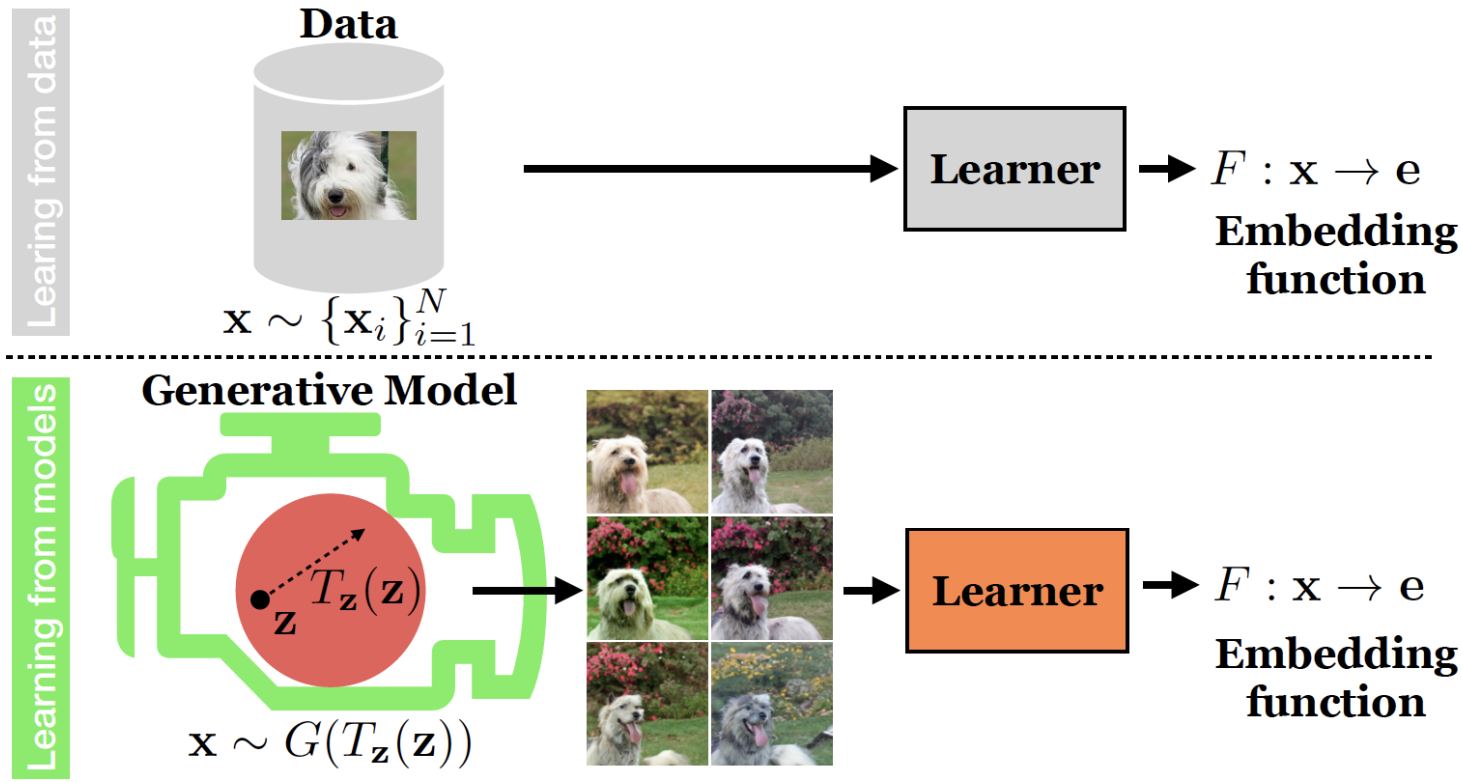
Table 1: **Results for ImageNet-10 real test images.** Performance of classifiers trained on generated images with all combinations of the proposed methods. Each classifier is trained for 150 epochs (except *Long training*, where we let DS run for 1500 epochs) over a set of $N = 13$K images; in case of *continuous sampling* we replace 50% (*i.e.*, $6,500$) of the images every epoch, while *fixed dataset* is the usual setup where no images are replaced during training. In all setups we use N images per epoch. First column, without applying any of the proposed methods, is the baseline. Each of the proposed methods individually shows improvement over the baseline. The combination of the methods further improves the results.

Generator : BiGANs Classifier : Resnet18

[a] https://valeoai.github.io/blog/publications/gan-dataset/

Data Augmentation : GANs for data augmentation

(Jahanian et al, ICLR 22): GENERATIVE MODELS AS A DATA SOURCE FOR MULTIVIEW REPRESENTATION LEARNING[10]

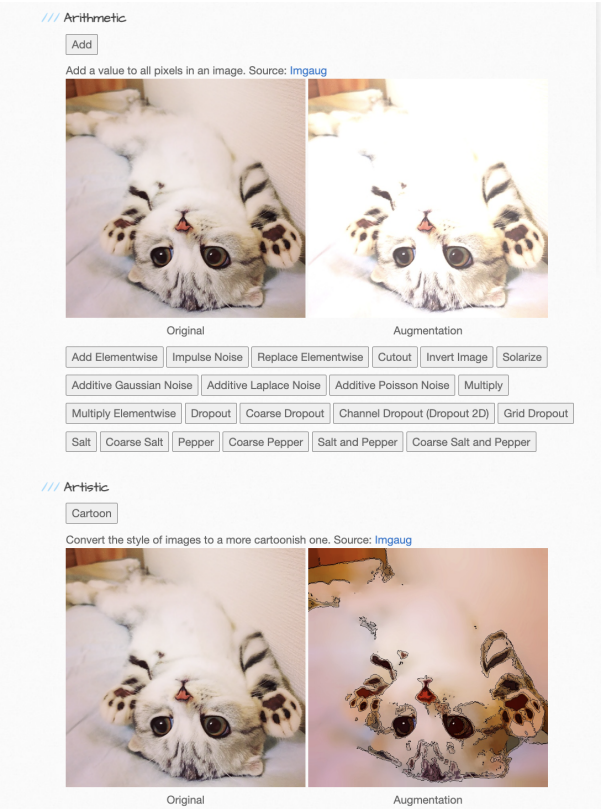


[10] https://ali-design.github.io/GenRep/

Data Augmentation

Many ressources

- https://brunokrinski.github.io/awesome-data-augmentation/
- Albumentations (https://github.com/albumentations-team/albumentations)
- Imgaug (https://github.com/aleju/imgaug)
- Augly : data augmentations library that currently supports four modalities (https://github.com/facebookresearch/AugLy)

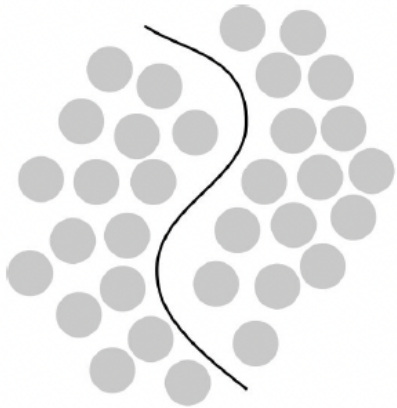


Overview

Alternatives to supervised learning

Supervised Learning

Semi-Supervised Learning



Unsupervised Learning

Richness of unlabeled data

- Data privacy friendly
- Cheaper
- Scales better

Learning with limited labeled data

- Semi-supervised deep learning
- Few-shot learning
- Unsupervised and self-supervised learning
- Transfer learning and domain adaptation (robust ML)

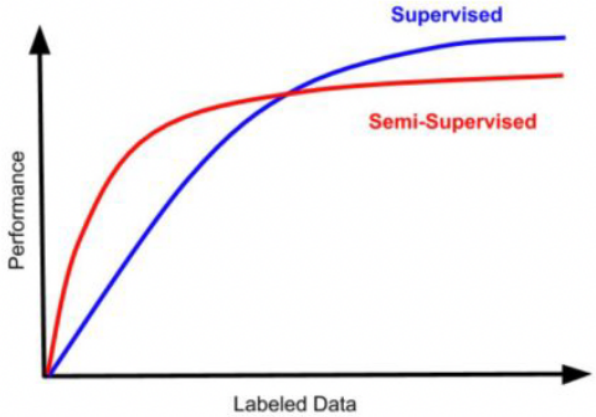
Overview

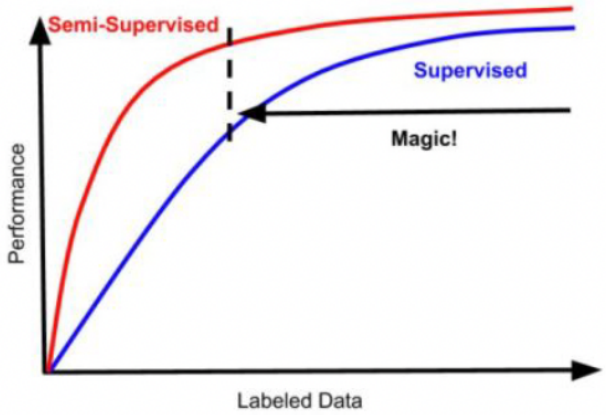
Sources and references

- (Ouali et al, 2020) An Overview of Deep Semi-Supervised Learning (https://arxiv.org/abs/2006.05278)
- (Yang et al, 2021) A Survey on Deep Semi-supervised Learning (https://arxiv.org/pdf/2103.00550.pdf)
- Lil'Log : Learning with not Enough Data Part 1: Semi-Supervised Learning (https://lilianweng.github.io/posts/2021-12-05-semi-supervised/)

Semi-supervised Learning

> Semi-supervised learning (SSL) is halfway between supervised and unsupervised learning. In addition to unlabeled data, the algorithm is provided with some supervision information – but not necessarily for all examples. Often, this information will be the targets associated with some of the examples. In this case, the data set $X = (x_i)\,; i \in [n]$ can be divided into two parts: the points $X_l := (x_1, \ldots, x_l)$, for which labels $Y_l := (y_1, \ldots, y_l)$ are provided, and the points $X_u := (x_{l+1}, \ldots, x_{l+u})$, the labels of which are not known.
>
> — ***Chapelle et al.*** — *SSL book*





Reference : SSL book :
`https://mitpress.mit.edu/books/semi-supervised-learning`

Semi-supervised Learning

Main assumptions about the structure of data

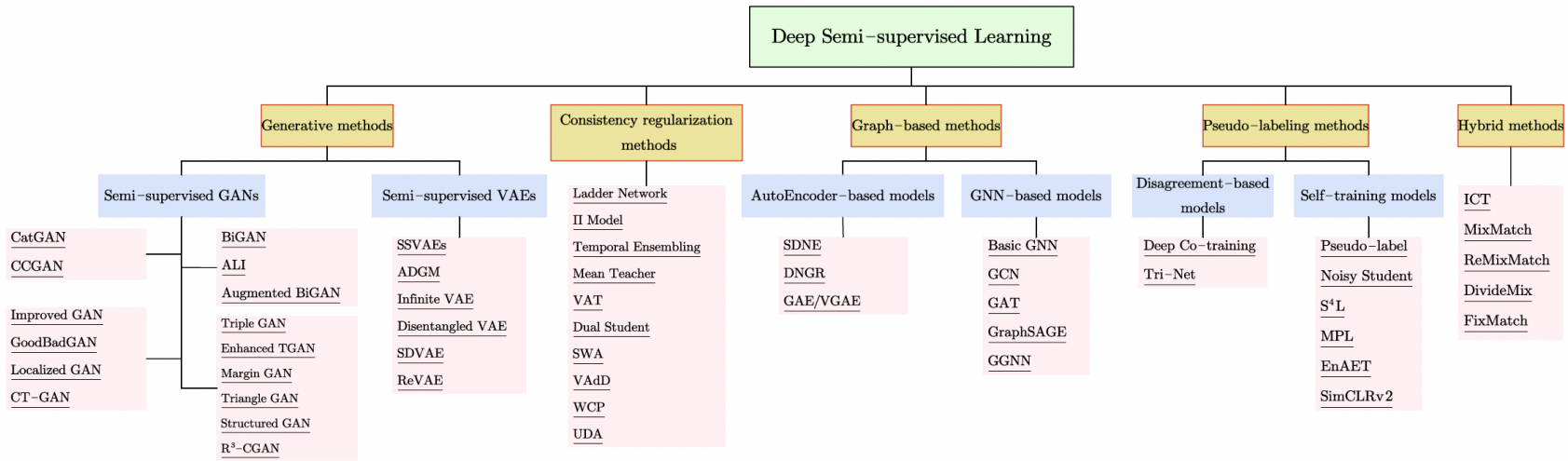
- **H1 : The Smoothness Assumption**: if two input points that reside in a high-density region are close then so should be their corresponding outputs.
- **H2 : The Cluster Assumption** : If points are in a same cluster, they are likely to be of the same class (particular case of the Smoothness Assumption in which we suppose that input data points form cluters)
- **H3 : Low-density Separation Assumption** : The decision boundary between classes tends to be located in the sparse, low density regions, because otherwise the decision boundary would cut a high-density cluster into two classes, corresponding to two clusters, which invalidates H1 and H2
- **H4 : The Manifold Assumption** : The (high dimensional) data (roughly) lie on a low-dimensional manifold, i.e. discover the low dimensional representation from unlabeled data and use the labeled data to solve the simplified task in the lower dimensional space.

Semi-supervised Learning

Main methods

- **Consistency training** : a realistic perturbation applied to the unlabeled point should not change significantly the prediction, i.e. the model can be trained to have a consistent prediction on a given unlabeled example and its perturbed version.
- **Pseudo-labeling methods** : leveraging of a model trained on a labeled set to produce additional training examples by labeling instances of the unlabeled set based on some heuristics.
- **Generative models** : If a generative model is able to generate realistic images from the data distribution $p(x)$, then it must learn transferable features to a supervised task $p(y|x)$
- **Graph-based methods** : Labeled and unlabeled data are considered as nodes of a graph and the objective is to propagate the labeled nodes to the unlabeled ones using the similarity between nodes and edge information.

Deep Semi-supervised Learning : an overview



(Yang et al, 2021) A Survey on Deep Semi-supervised Learning[11]

[11] https://arxiv.org/abs/2103.00550

Consistency training

Consistency training

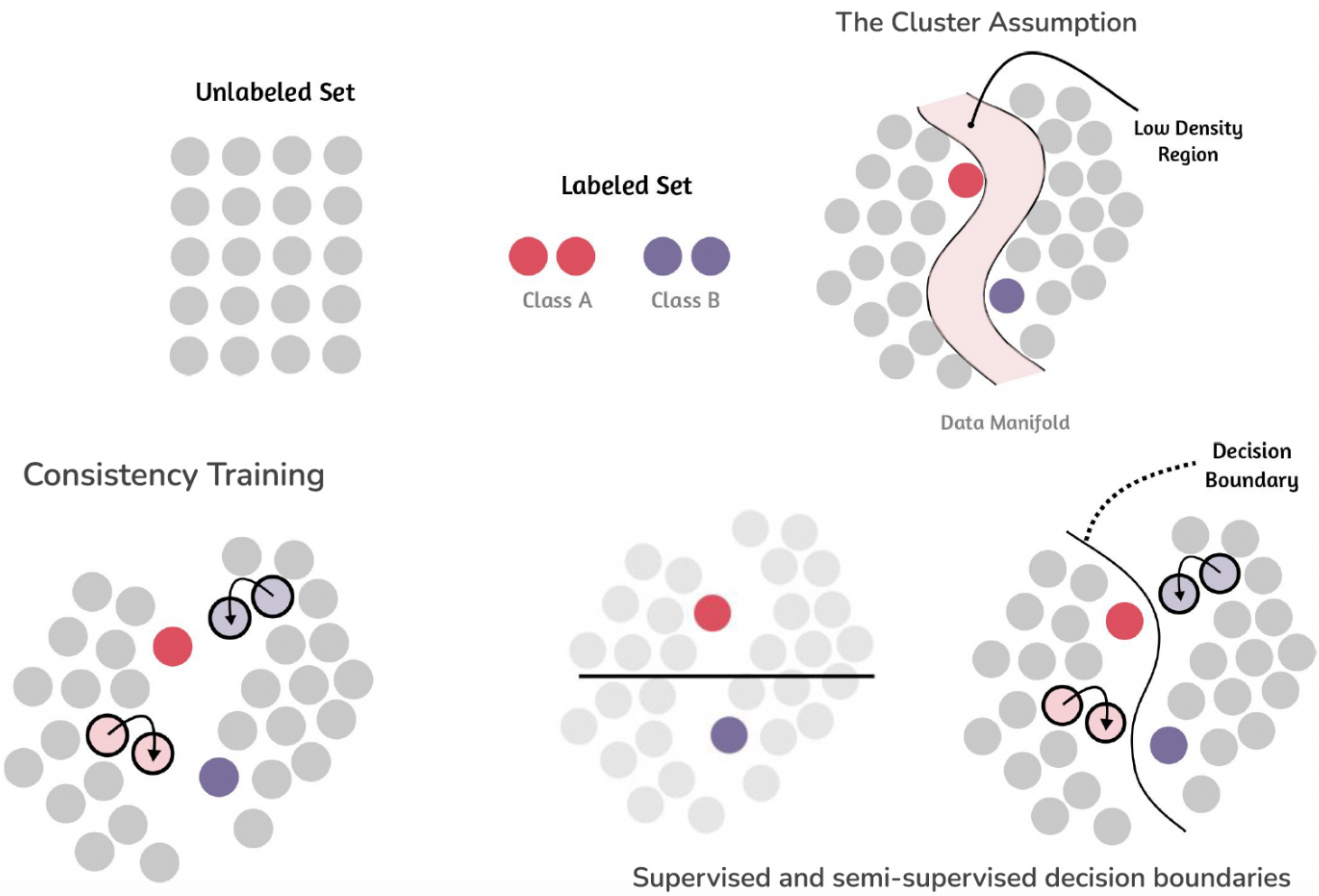
Consistency Regularization

$$\|\mathrm{p}_{\mathrm{model}}(y \mid \mathrm{Augment}(x); \theta) - \mathrm{p}_{\mathrm{model}}(y \mid \mathrm{Augment}(x); \theta)\|_2^2$$

Main assumption : randomness within a neural network (e.g. with Dropout or data augmentation transformations) should not modify model predictions given the same input : use of a consistency regularization loss as $\mathcal{L}_u$

Consistency training

A realistic perturbation applied to the unlabeled point should not change significantly the prediction, i.e. the model can be trained to have a consistent prediction on a given unlabeled example and its perturbed version.



Consistency training

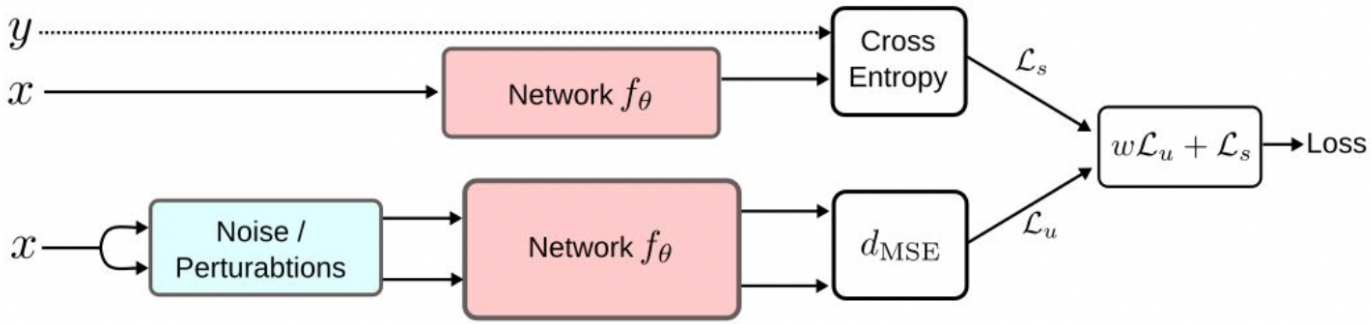
In **consistency training**, we are favoring models that give consistent predictions for similar unlabeled data points.
Given a set of unlabeled data $\mathcal{D}_u$ and a set of labeled data $\mathcal{D}_l$

- Consistency training methods generally consists into adding an unsupervised loss term $\mathcal{L}_u$ in addition to the supervised one $\mathcal{L}_s$ (here cross-entropy)

$$\mathcal{L} = \mathcal{L}_u + \mathcal{L}_s = \mathcal{L}_u + \frac{1}{|\mathcal{D}_l|} \sum_{x,t \in \mathcal{D}_l} H(\tilde{y}, t)$$

- The unsupervised loss is generally a distance between two predictions $(\tilde{y_1}, \tilde{y_2})$ over the unlabeled data.

$$\mathcal{L} = w \frac{1}{|\mathcal{D}_u|} \sum_{x \in \mathcal{D}_u} d_{MSE}(\tilde{y_1}, \tilde{y_2}) + \frac{1}{|\mathcal{D}_l|} \sum_{x,y \in \mathcal{D}_l} H(y, f(x))$$



Ladder networks

Ladder networks

- Combine supervised learning with unsupervised learning in deep neural networks.
- Ladder networks are trained to **simultaneously minimize the sum of supervised and unsupervised cost functions** by backpropagation, avoiding the need for layer-wise pre-training

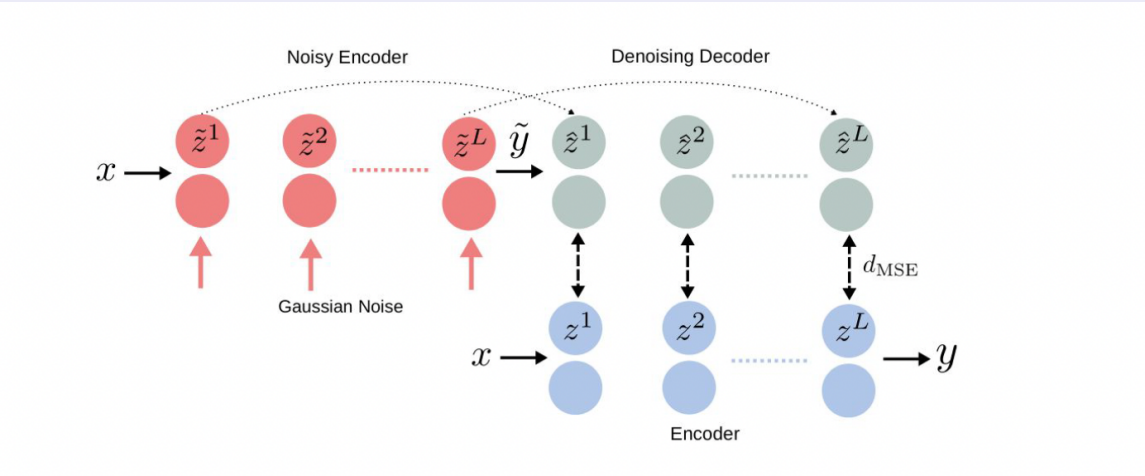
(Rasmus et al , 2015) Semi-supervised Learning with Ladder Networks (`https://arxiv.org/abs/1507.02672`). See also `https://rinuboney.github.io/2016/01/19/ladder-network.html`

Ladder networks

Principle

- At each training iteration, the input is passed through two encoders, **a clean one** and **a corrupted one** which adds Gaussian noise to the layers $\Rightarrow$ two predictions.
- A **decoder** can invert the mappings on each layer of the encoder. It uses a denoising function to reconstruct the activations of each layer given the corrupted version. The target at each layer is the clean version of the activation and the difference between the reconstruction and the clean version serves as the denoising cost of that layer.
- The unsupervised training loss is then computed as the MSE between the activations of the clean encoder and the reconstructed activations.

$$\mathcal{L}_u = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \sum_{l=0}^{L} \lambda_l d_{MSE}(z^{(l)}, \hat{z}^{(l)})$$



Pi-Model

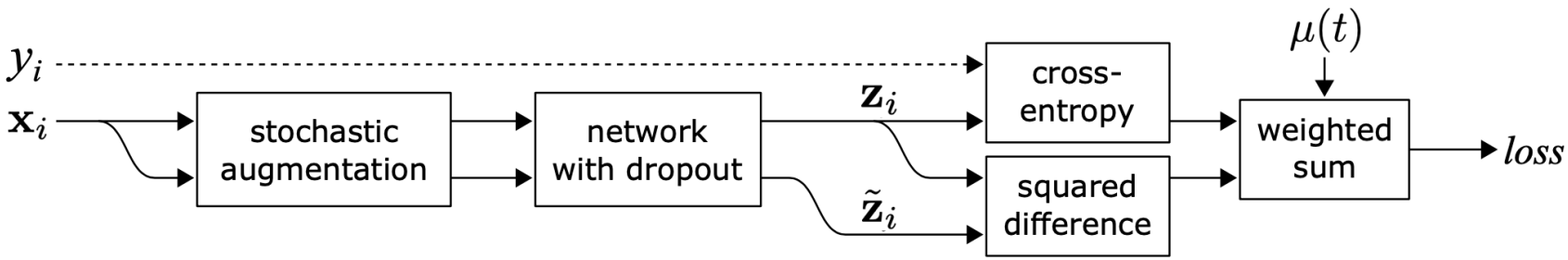
Idea

Ensemble method : a set of labels is more reliable than a simple one and thus consistency regularization : two augmentations should result in the same prediction.

The $\Pi-$Model is a simplification of Ladder networks where the corrupted encoder is removed and the same network is used to get prediction for both corrupted and uncorrupted inputs.

$$\mathcal{L}_u^{\Pi} = \sum_{x \in \mathcal{D}} MSE(f_\theta(x), f_\theta^{'}(x))$$

with $f^{'}$ the same neural network with different stochastic augmentation or dropout masks.



(Laine and Aila, 2017) Temporal Ensembling for Semi-Supervised Learning (`https://arxiv.org/abs/1610.02242`)

Pi-Model

Algorithm 1 Π-model pseudocode.

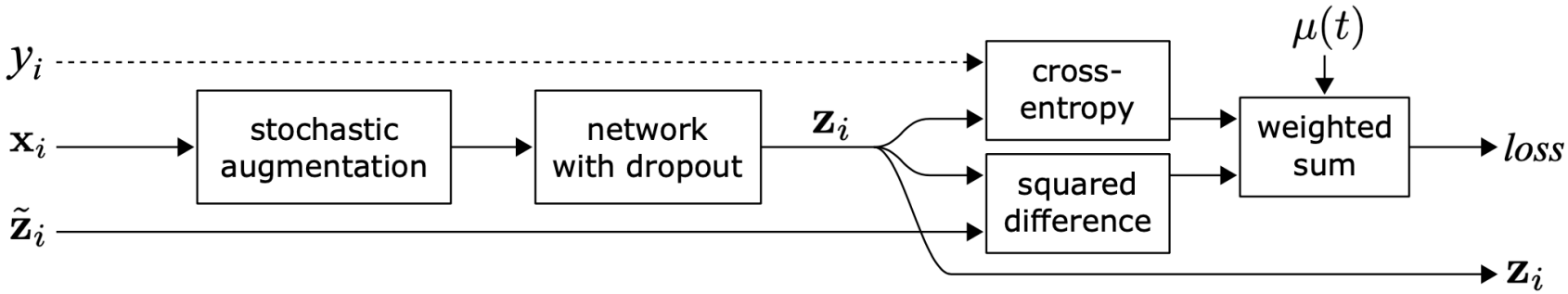
Require: x_i = training stimuli
Require: L = set of training input indices with known labels
Require: y_i = labels for labeled inputs $i \in L$
Require: $w(t)$ = unsupervised weight ramp-up function
Require: $f_\theta(x)$ = stochastic neural network with trainable parameters θ
Require: $g(x)$ = stochastic input augmentation function
for t in $[1, num\_epochs]$ **do**
  for each minibatch B **do**
    $z_{i \in B} \leftarrow f_\theta(g(x_{i \in B}))$ ▷ evaluate network outputs for augmented inputs
    $\tilde{z}_{i \in B} \leftarrow f_\theta(g(x_{i \in B}))$ ▷ again, with different dropout and augmentation
    $loss \leftarrow -\frac{1}{|B|} \sum_{i \in (B \cap L)} \log z_i[y_i]$ ▷ supervised loss component
      $+ w(t) \frac{1}{C|B|} \sum_{i \in B} ||z_i - \tilde{z}_i||^2$ ▷ unsupervised loss component
    update θ using, e.g., ADAM ▷ update network parameters
  end for
end for
return θ

Temporal ensembling

The Π−Model requests the network to run two passes per sample, doubling the computation cost. **Temporal Ensembling** maintains an exponential moving average (EMA) of the model prediction in time per training sample as the learning target, which is only evaluated and updated once per epoch.

$$\tilde{\mathbf{z}}_i^{(t)} = \frac{\alpha \tilde{\mathbf{z}}_i^{(t-1)} + (1-\alpha)\mathbf{z}_i}{1-\alpha^t}$$

Moving average and bias correction *a la Adam*



(Laine and Aila, 2017) Temporal Ensembling for Semi-Supervised Learning (`https://arxiv.org/abs/1610.02242`)

Temporal ensembling

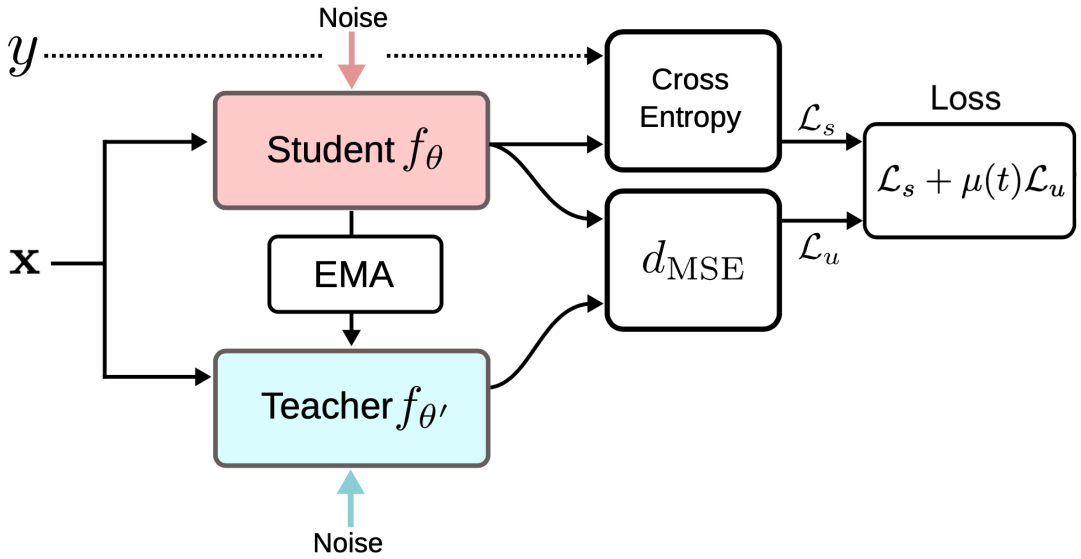
Algorithm 2 Temporal ensembling pseudocode. Note that the updates of Z and $\tilde{z}$ could equally well be done inside the minibatch loop; in this pseudocode they occur between epochs for clarity.

Require: x_i = training stimuli
Require: L = set of training input indices with known labels
Require: y_i = labels for labeled inputs $i \in L$
Require: α = ensembling momentum, $0 \leq \alpha < 1$
Require: $w(t)$ = unsupervised weight ramp-up function
Require: $f_\theta(x)$ = stochastic neural network with trainable parameters θ
Require: $g(x)$ = stochastic input augmentation function

$Z \leftarrow \mathbf{0}_{[N \times C]}$ ▷ initialize ensemble predictions
$\tilde{z} \leftarrow \mathbf{0}_{[N \times C]}$ ▷ initialize target vectors
for t in $[1, num\_epochs]$ **do**
  for each minibatch B **do**
    $z_{i \in B} \leftarrow f_\theta(g(x_{i \in B}, t))$ ▷ evaluate network outputs for augmented inputs
    $loss \leftarrow -\frac{1}{|B|} \sum_{i \in (B \cap L)} \log z_i[y_i]$ ▷ supervised loss component
      $+ w(t) \frac{1}{C|B|} \sum_{i \in B} ||z_i - \tilde{z}_i||^2$ ▷ unsupervised loss component
    update θ using, e.g., ADAM ▷ update network parameters
  end for
  $Z \leftarrow \alpha Z + (1 - \alpha) z$ ▷ accumulate ensemble predictions
  $\tilde{z} \leftarrow Z / (1 - \alpha^t)$ ▷ construct target vectors by bias correction
end for
return θ

Mean Teachers

In temporal ensembling, we kept an exponential moving average over the target of each input. But why not keep an exponential moving average over the model itself, which can then be used to generate stable targets over the unlabeled examples ? We call this model the **teacher model**.

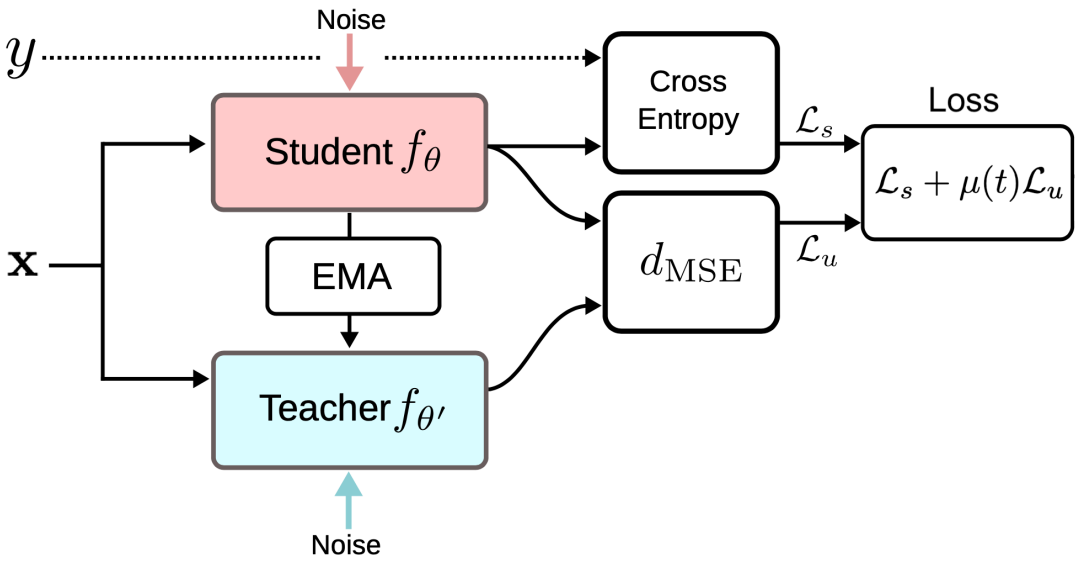


Mean Teachers

The loss is the sum of the supervised and unsupervised loss, where the teacher model is used to obtain the targets for the unsupervised loss for a given input.

$$\mathcal{L} = w\frac{1}{|\mathcal{D}_u|}\sum_{x\in\mathcal{D}_u} d_{MSE}(f_\theta(x), f_{\theta'}(x)) + \frac{1}{|\mathcal{D}_l|}\sum_{x,y\in\mathcal{D}_l} H(y, f_\theta(x))$$

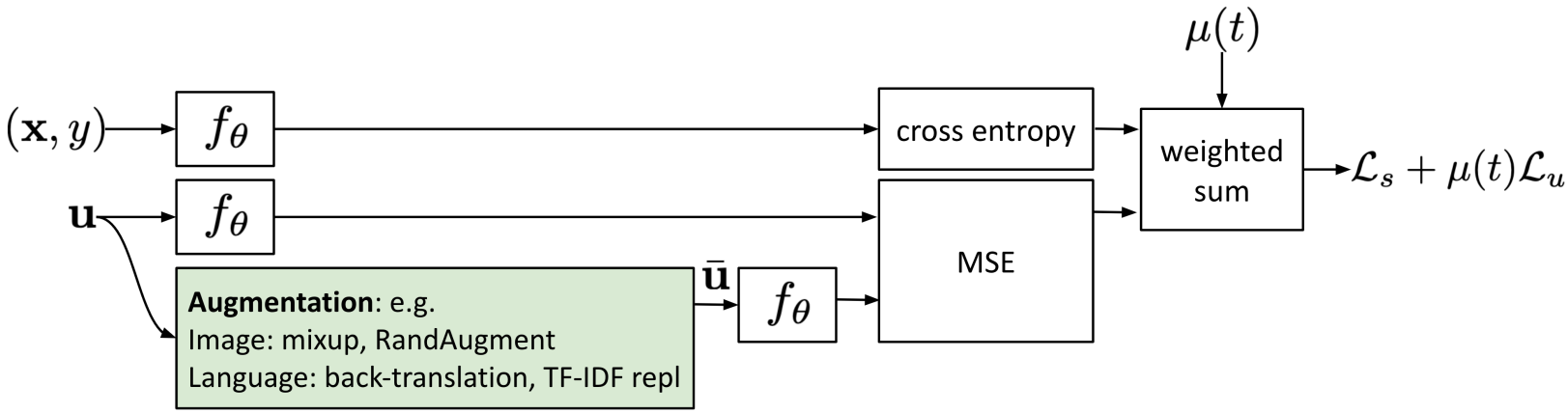
with $\theta'_t = \alpha\theta'_{t-1} + (1-\alpha)\theta_t$



(Tarveinen and Valpola,2017) Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results (`https://arxiv.org/abs/1703.01780`)

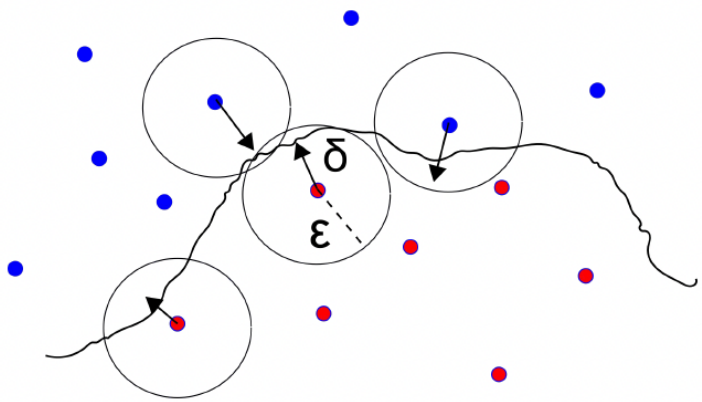
Noisy samples as learning targets

Recent consistency training methods learn to minimize prediction difference between the original unlabeled sample and its corresponding augmented version. It is quite similar to previous models but the consistency regularization loss is only applied to the unlabeled data.



VAT : Virtual Adversarial Training

- Until now : random perturbations to each input to generate artificial input points, encouraging the model to assign similar outputs to the unlabeled data points and their perturbed versions
- Random noise and random data augmentation leave the predictor particularly vulnerable to small perturbations in a specific direction, the **adversarial direction** : direction in the input space in which the label probability $p(y|x)$ of the model is most sensitive.
- Why not find the perturbation that the model is most sensitive towards, and use it to perturb the inputs?



1. $r \sim \mathcal{N}(0, \frac{\xi}{\sqrt{\dim(x)}} I)$
2. $grad_r = \nabla_r d_{\mathrm{KL}}(f_\theta(x), f_\theta(x+r))$
3. $r_{adv} = \epsilon \frac{grad_r}{\|grad_r\|}$

VAT : Virtual Adversarial Training

The loss computation consists of two steps, first find the adversarial perturbation, then use this perturbation to perturb the input and get the perturbed prediction to compute the unsupervised loss.

$$\mathcal{L}_u^{\text{VAT}} = w \frac{1}{|\mathcal{D}_u|} \sum_{x \in \mathcal{D}_u} d_{MSE}(f_\theta(x), f_\theta(x + r_{adv}))$$

(Miyato et al, 2018) Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning (`https://arxiv.org/abs/1704.03976`)

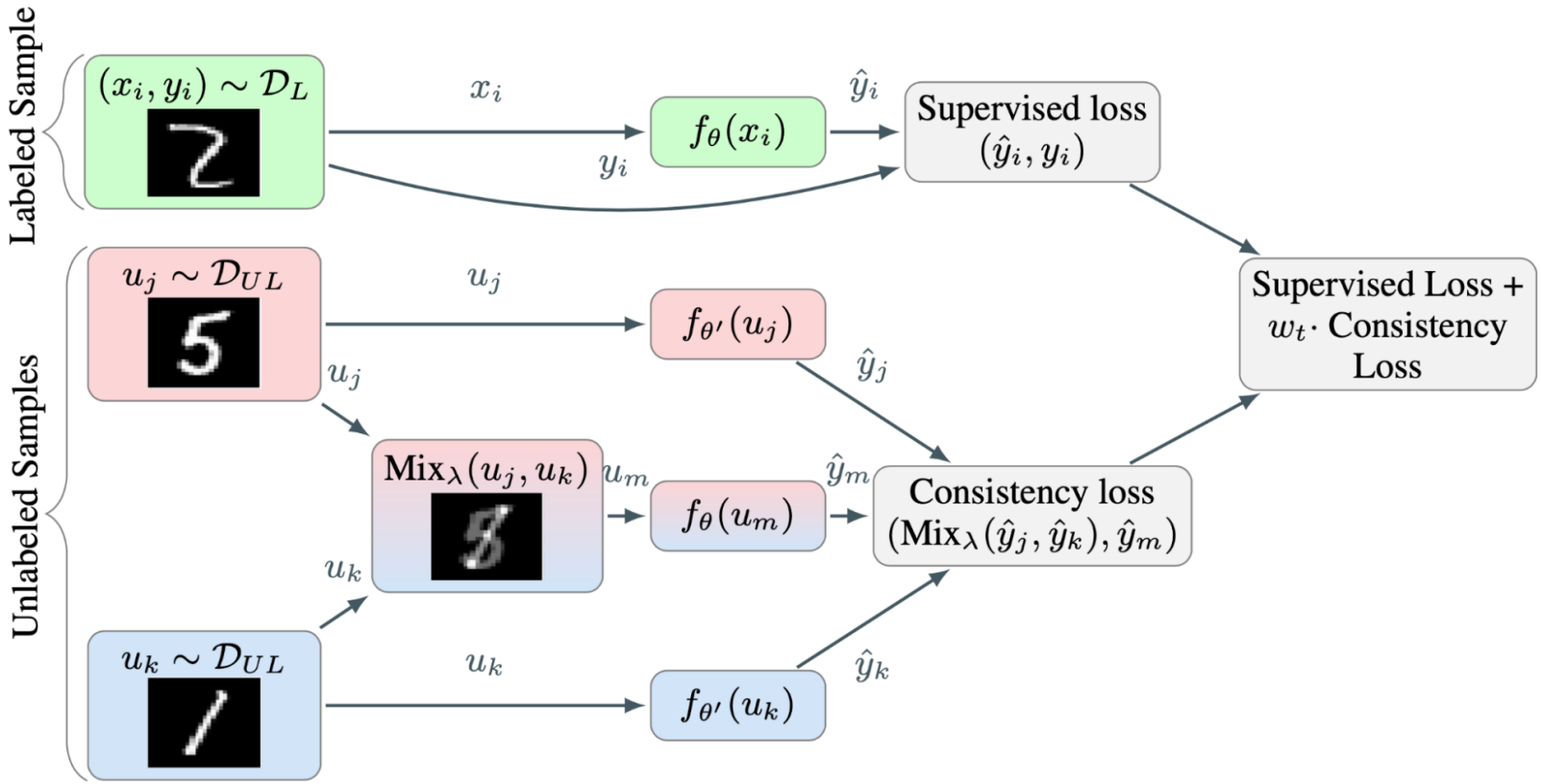
ICT : Interpolation Consistency Training

Principle

- Enhancement of the dataset by adding more interpolations of data points and expects the model prediction to be consistent with interpolations of the corresponding labels.
- Use of the mixup operation that mixes two images via a simple weighted sum and combines it with label smoothing.
- ICT expects the prediction model to produce a label on a mixup sample to match the interpolation of predictions of corresponding inputs.

$$\text{mixup}_\lambda(\mathbf{x}_i, \mathbf{x}_j) = \lambda \mathbf{x}_i + (1-\lambda)\mathbf{x}_j$$
$$p(\text{mixup}_\lambda(y \mid \mathbf{x}_i, \mathbf{x}_j)) \approx \lambda p(y \mid \mathbf{x}_i) + (1-\lambda) p(y \mid \mathbf{x}_j)$$

ICT : Interpolation Consistency Training



(Verma et al, 2019) Interpolation Consistency Training for Semi-Supervised Learning (`https://arxiv.org/abs/1903.03825`)

Entropy Minimization

Entropy Minimization

H3 : Low-density Separation Assumption : The decision boundary between classes tends to be located in the sparse, low density regions, i.e. decision boundary should not pass through high-density regions of the marginal data distribution.

Entropy Minimization

Entropy minimization encourages more confident predictions on unlabeled data i.e the classifier outputs low-entropy predictions on unlabeled data.
Entropy computation

$$\sum_{k=1}^{K} f_\theta(x)_k \log f_\theta(x)_k$$

with K the number of classes and $f_\theta(x)_k$ is the confidence with which the model predicts that input x belongs to class k.

Adding of a loss term that encourages the network to make "confident" (low-entropy) predictions for all unlabeled examples, regardless of their class.

Entropy Minimization

If we consider an input x and classes $[y_1, y_2, y_3]$.

- Classifier A : probabilities $[0.1, 0.8, 0.1]$
- Classifier B : probabilities $[0.1, 0.6, 0.3]$

Classifier A is more confident and has lower entropy.
By minimizing the entropy loss, we learn a model closer to classifier A then classifier B.

Entropy Minimization

(Grandvalet et al, 2004) Semi-supervised Learning by Entropy Minimization[12]
Entropy minimization will discourage the decision boundary from passing near data points where it would otherwise be forced to produce a low-confidence prediction.

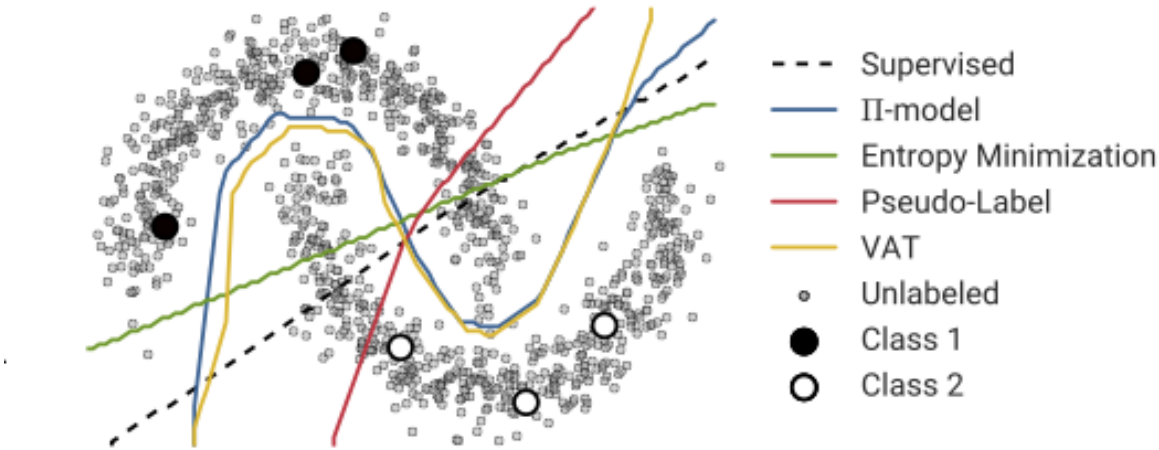


Figure: Decision Boundary found by various SSL approaches on the "two moons dataset". From Paper: Realistic Evaluation of Deep Semi-Supervised Learning Algorithms.

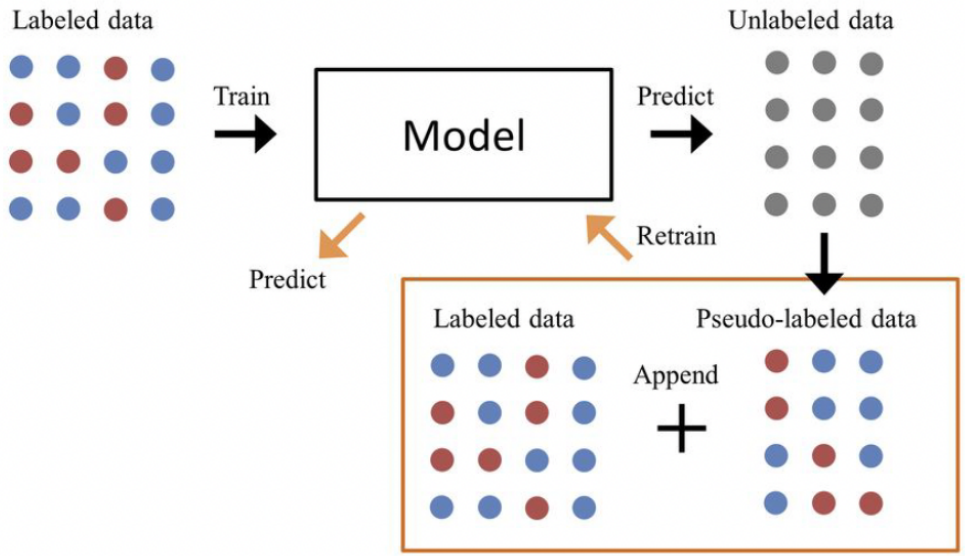
(Oliver et al, 2019) Realistic Evaluation of Deep Semi-Supervised Learning Algorithms (`https://arxiv.org/pdf/1804.09170.pdf`)

[12] `https://proceedings.neurips.cc/paper/2004/file/96f2b50b5d3613adf9c27049b2a888c7-Paper.pdf`

Pseudo Labeling

Pseudo Labeling

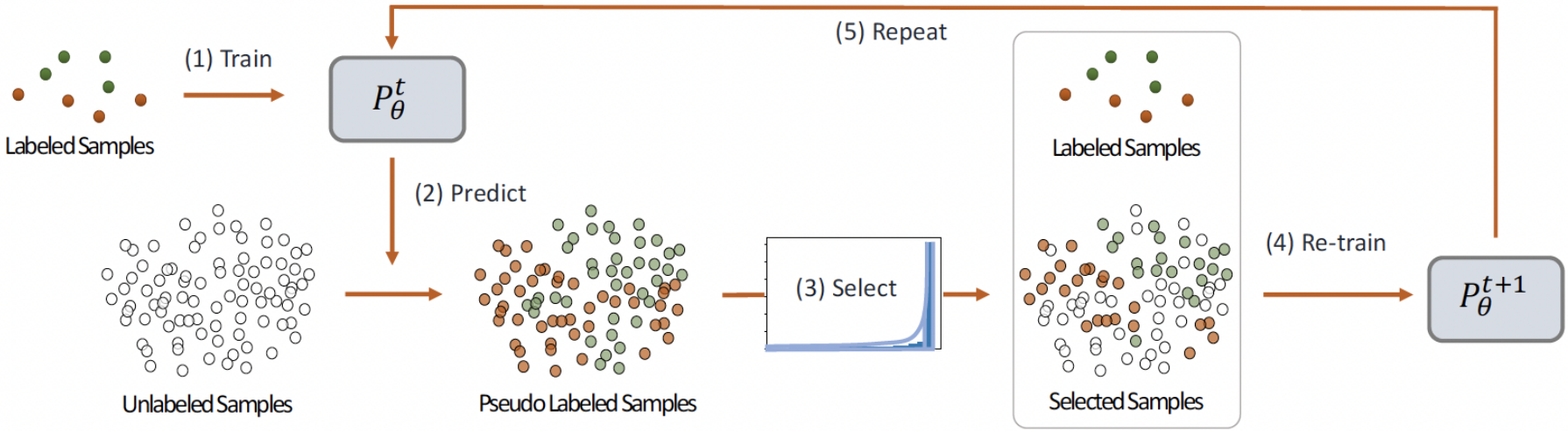
In pseudo labeling, a model is trained on labeled data and used to predict pseudo-labels for the unlabeled data. The model is then trained on both ground truth labels and pseudo-labels simultaneously.



One important question is, what unlabeled examples do we select to add to the labeled data ?

Pseudo Labeling

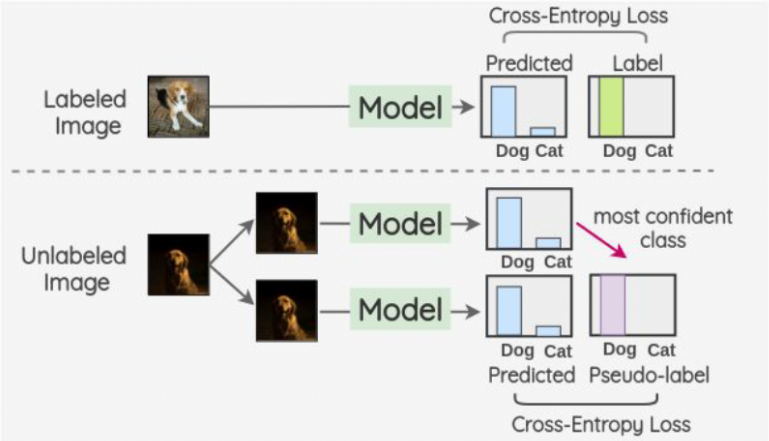
In pseudo labeling, a model is trained on labeled data and used to predict pseudo-labels for the unlabeled data. The model is then trained on both ground truth labels and pseudo-labels simultaneously.



Pseudo Labeling

Principle

- Train a model simultaneously on a batch of both labeled and unlabeled images.
- The model is trained on labeled images in usual supervised manner with a cross-entropy loss.
- The same model is used to get predictions for a batch of unlabeled images and the maximum confidence class is used as the pseudo-label.
- Then, cross-entropy loss is calculated by comparing model predictions and the pseudo-label for the unlabeled images



(Lee, 2013) Pseudo-Label: The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks (`http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.664.3543`)

Pseudo Labeling

- **Pseudo-Label** are target classes for unlabeled data as if they were true labels. The class, which has maximum predicted probability predicted using a network for each unlabeled sample, is picked up.

$$y'_i = \begin{cases} 1 & \text{if } i = \text{argmax}_{i'} f_{i'}(x) \\ 0 & \text{otherwise} \end{cases}$$

- Pseudo-Label is used in a fine-tuning phase with Dropout. The pre-trained network is trained in a supervised fashion with labeled and unlabeled data simultaneously.

$$L = \frac{1}{n} \sum_{m=1}^{n} \sum_{i=1}^{C} L(y_i^m, f_i^m) + \alpha(t) \frac{1}{n'} \sum_{m=1}^{n'} \sum_{i=1}^{C} L(y_i'^m, f_i'^m),$$

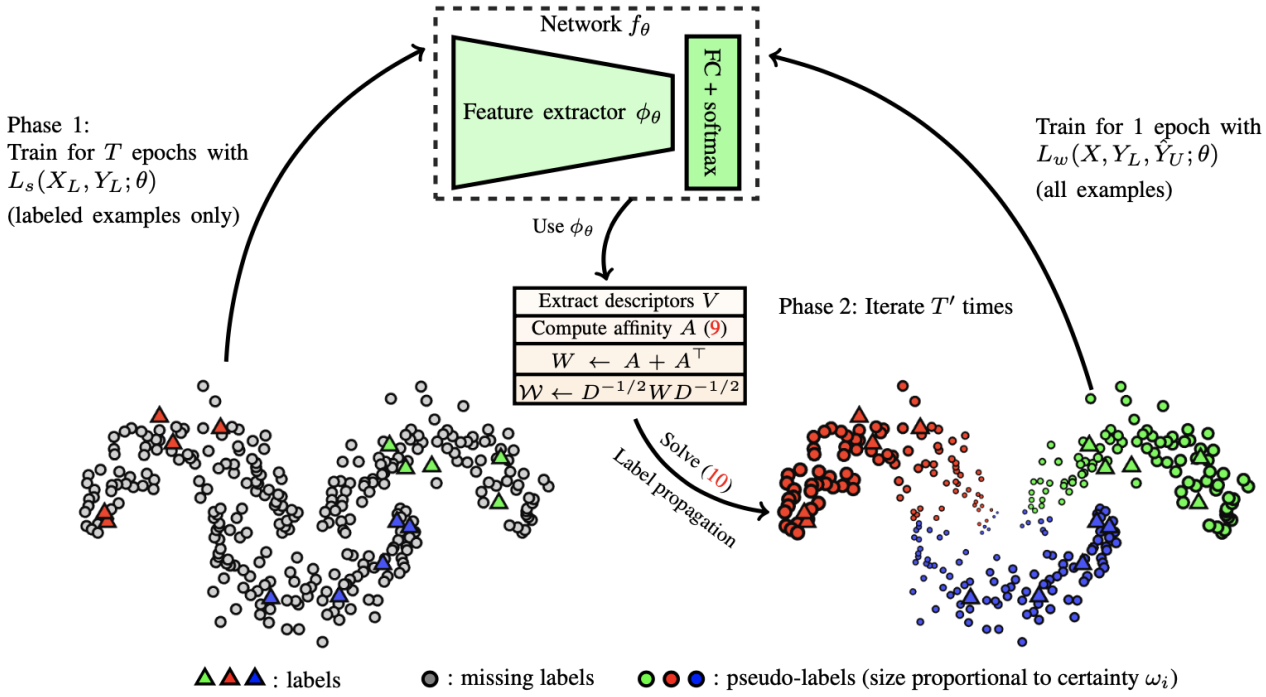
- $\alpha(t)$ is a coefficient balancing them at epoch t. If too high, it disturbs training even for labeled data. If too small, we cannot use benefit from unlabeled data.

Pseudo label is equivalent to Entropy Regularization.

Label propagation

Main ideas

- Construct a similarity graph among samples based on feature embedding
- Pseudo labels are "diffused" from known samples to unlabeled ones where the propagation weights are proportional to pairwise similarity scores in the graph



(Iscen et al, 2019) Label Propagation for Deep Semi-supervised Learning (`https://arxiv.org/abs/1904.04717`)

Hybrid approaches

Hybrid approaches : Pseudo Labeling with Consistency Regularization

- MixMatch
- ReMixMatch
- DivideMix
- FixMatch
- ...

Hybrid approaches : MixMatch

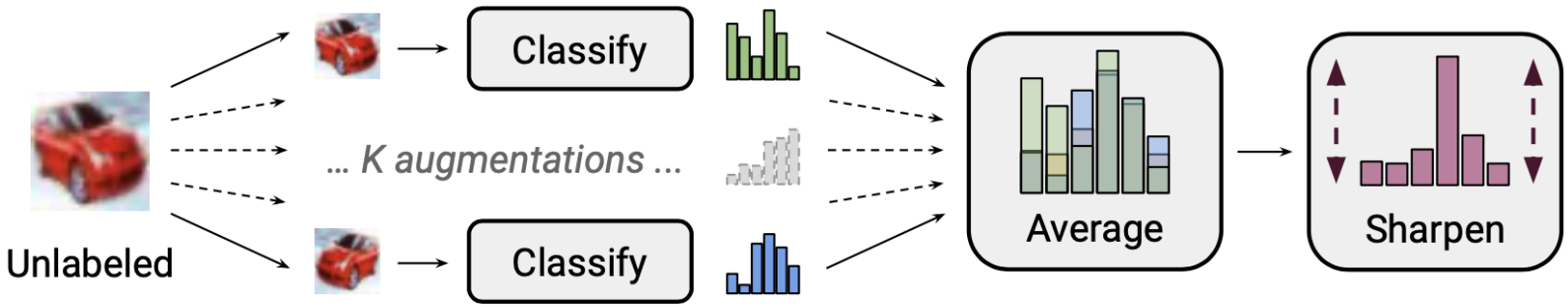
Principle : Merging of

- **Consistency regularization**: Encourage the model to output the same predictions on perturbed unlabeled samples.
- **Entropy minimization**: Encourage the model to output confident predictions on unlabeled data.
- **MixUp augmentation**: Encourage the model to have linear behaviour between samples.
- **Label guessing** : pseudo-labelling

(Berthelot et al, 2019) MixMatch: A Holistic Approach to Semi-Supervised Learning (`https://arxiv.org/abs/1905.02249`)

Hybrid approaches : MixMatch

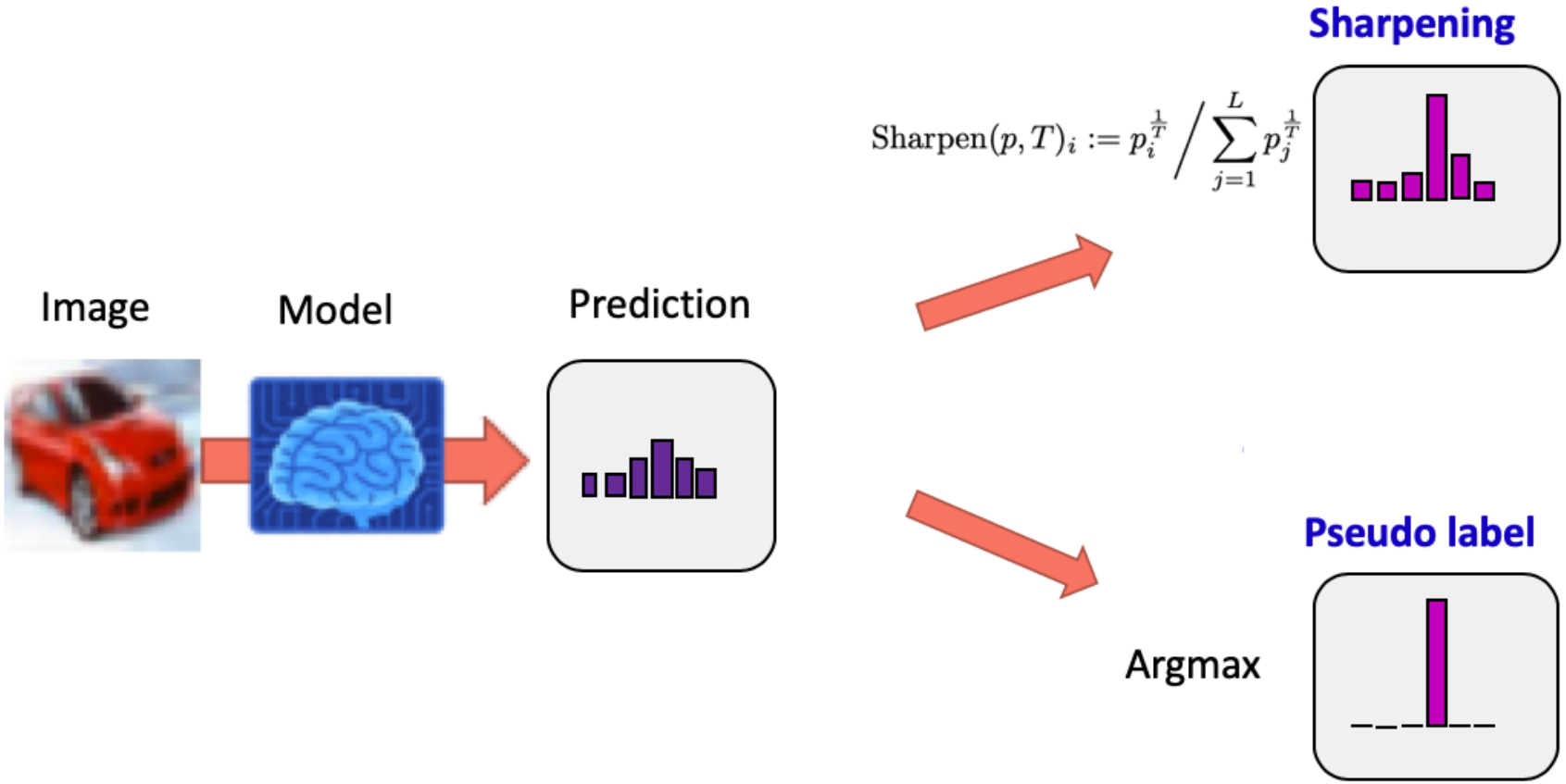
Label-guessing principle in MixMatch



- For each $\mathbf{u}$, K augmentations are generated, $\bar{\mathbf{u}}^{(k)} = \text{Augment}(\mathbf{u})$ for $k = 1, \ldots, K$ and the pseudo label is guessed based on the average : $\hat{y} = \frac{1}{K} \sum_{k=1}^{K} p_\theta(y \mid \bar{\mathbf{u}}^{(k)})$
- Then, temperature sharpening function to reduce the entropy of the label distribution

Hybrid approaches : MixMatch

Temperature sharpening



Hybrid approaches : MixMatch

Algorithm 1 MixMatch takes a batch of labeled data $\mathcal{X}$ and a batch of unlabeled data $\mathcal{U}$ and produces a collection $\mathcal{X}'$ (resp. $\mathcal{U}'$) of processed labeled examples (resp. unlabeled with guessed labels).

Input: Batch of labeled examples and their one-hot labels $\mathcal{X} = \big((x_b, p_b); b \in (1, \dots, B)\big)$, batch of unlabeled examples $\mathcal{U} = \big(u_b; b \in (1, \dots, B)\big)$, sharpening temperature T, number of augmentations K, Beta distribution parameter α for MixUp.
for $b = 1$ **to** B **do**
  $\hat{x}_b = \text{Augment}(x_b)$ *// Apply data augmentation to* x_b
  for $k = 1$ **to** K **do**
    $\hat{u}_{b,k} = \text{Augment}(u_b)$ *// Apply* k^{th} *round of data augmentation to* u_b
  end for
  $\bar{q}_b = \frac{1}{K} \sum_k \text{p}_{\text{model}}(y \mid \hat{u}_{b,k}; \theta)$ *// Compute average predictions across all augmentations of* u_b
  $q_b = \text{Sharpen}(\bar{q}_b, T)$ *// Apply temperature sharpening to the average prediction (see eq. (7))*
end for
$\hat{\mathcal{X}} = \big((\hat{x}_b, p_b); b \in (1, \dots, B)\big)$ *// Augmented labeled examples and their labels*
$\hat{\mathcal{U}} = \big((\hat{u}_{b,k}, q_b); b \in (1, \dots, B), k \in (1, \dots, K)\big)$ *// Augmented unlabeled examples, guessed labels*
$\mathcal{W} = \text{Shuffle}\big(\text{Concat}(\hat{\mathcal{X}}, \hat{\mathcal{U}})\big)$ *// Combine and shuffle labeled and unlabeled data*
$\mathcal{X}' = \big(\text{MixUp}(\hat{\mathcal{X}}_i, \mathcal{W}_i); i \in (1, \dots, |\hat{\mathcal{X}}|)\big)$ *// Apply* MixUp *to labeled data and entries from* $\mathcal{W}$
$\mathcal{U}' = \big(\text{MixUp}(\hat{\mathcal{U}}_i, \mathcal{W}_{i+|\hat{\mathcal{X}}|}); i \in (1, \dots, |\hat{\mathcal{U}}|)\big)$ *// Apply* MixUp *to unlabeled data and the rest of* $\mathcal{W}$
return $\mathcal{X}', \mathcal{U}'$

Hybrid approaches : MixMatch

Given a batch of labeled data $\mathcal{X}$ and unlabeled data $\mathcal{U}$, we create augmented versions of them via MixMatch(.), $\bar{\mathcal{X}}$ and $\bar{\mathcal{U}}$, containing augmented samples and guessed labels for unlabeled examples.

$$\bar{\mathcal{X}}, \bar{\mathcal{U}} = \text{MixMatch}(\mathcal{X}, \mathcal{U}, T, K, \alpha)$$

$$\mathcal{L}_s^{\text{MM}} = \frac{1}{|\bar{\mathcal{X}}|} \sum_{(\bar{\mathbf{x}}^l, y) \in \bar{\mathcal{X}}} H[y, p_\theta(y \mid \bar{\mathbf{x}}^l)]$$

$$\mathcal{L}_u^{\text{MM}} = \frac{1}{L|\bar{\mathcal{U}}|} \sum_{(\bar{\mathbf{u}}, \hat{y}) \in \bar{\mathcal{U}}} \|\hat{y} - p_\theta(y \mid \bar{\mathbf{u}})\|_2^2$$

$$\mathcal{L} = \mathcal{L}_s^{\text{MM}} + \lambda_u \mathcal{L}_u^{\text{MM}}$$

Hybrid approaches : ReMixMatch

Improves MixMatch by introducing two new mechanisms:

- **Distribution alignment** : encourages the marginal distribution to be close to the marginal distribution of the ground truth labels.
- **Augmentation anchoring** :Given an unlabeled sample, it first generates an *anchor* version with weak augmentation and then averages K strongly augmented versions using CTAugment (Control Theory Augment).

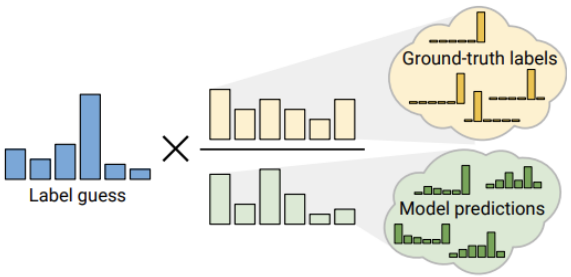


Figure 1: Distribution alignment. Guessed label distributions are adjusted according to the ratio of the empirical ground-truth class distribution divided by the average model predictions on unlabeled data.

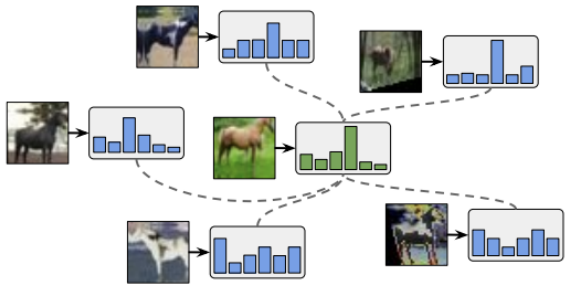
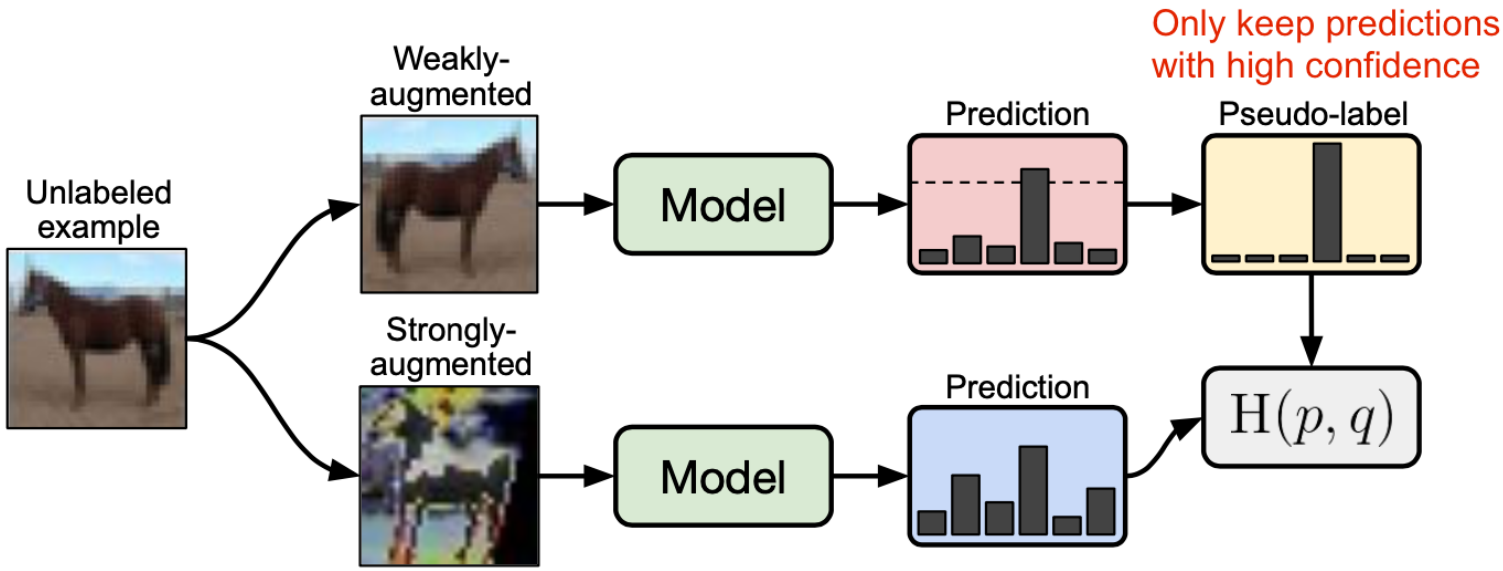


Figure 2: Augmentation anchoring. We use the prediction for a weakly augmented image (green, middle) as the target for predictions on strong augmentations of the same image (blue).

(Berthelot et al, 2020) ReMixMatch: Semi-Supervised Learning with Distribution Alignment and Augmentation Anchoring (`https://arxiv.org/abs/1911.09785`)

Hybrid approaches : FixMatch

It generates pseudo labels on unlabeled samples with weak augmentation and only keeps predictions with high confidence.



(Sohn et al, 2020) : FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence (`https://arxiv.org/abs/2001.07685`)

Hybrid approaches : FixMatch

$$\mathcal{L}_s = \frac{1}{B} \sum_{b=1}^{B} \text{CE}[y_b, p_\theta(y \mid \mathcal{A}_{\text{weak}}(\mathbf{x}_b))]$$

$$\mathcal{L}_u = \frac{1}{\mu B} \sum_{b=1}^{\mu B} \mathbf{1}[\max(\hat{y}_b) \geq \tau] \, \text{CE}(\hat{y}_b, p_\theta(y \mid \mathcal{A}_{\text{strong}}(\mathbf{u}_b)))$$

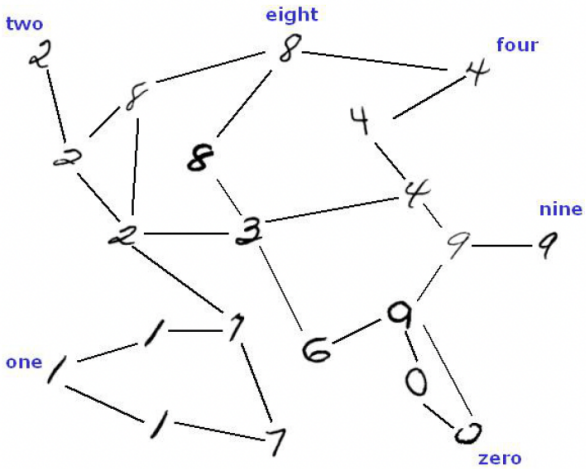
where $\hat{y}_b$ is the pseudo label for an unlabeled example; μ is a hyperparameter that determines the relative sizes of $\mathcal{X}$ and $\mathcal{U}$.

- Weak augmentation : A standard flip-and-shift augmentation
- Strong augmentation : AutoAugment, Cutout, RandAugment, CTAugment

Graph-based semi-supervised learning

Graph-based semi-supervised learning

Labeled and Unlabeled Data as a Graph

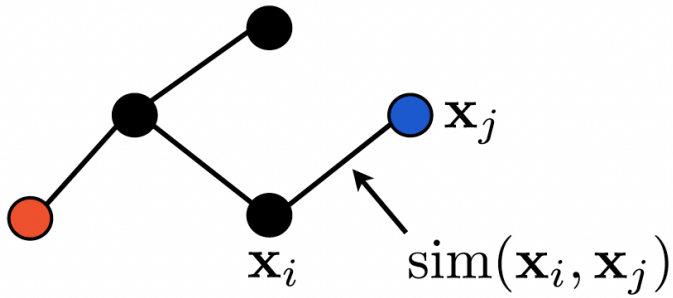


- Idea: Construct a graph connecting similar data points
- Let the hidden/observed labels be random variables on the nodes of this graph (i.e. the graph is an MRF)
- Intuition: Similar data points have similar labels
- Information “propagates” from labeled data points
- Graph encodes intuition

Graph-based semi-supervised learning

Smoothness Assumption

If two instances are similar according to the graph, then output labels should be similar



Two stages :

- Graph construction :
  - Neighborhood methods, metric learning
- Inference : a lot of approaches.

To go deeper, some reviews (Song et al 2021)[13]

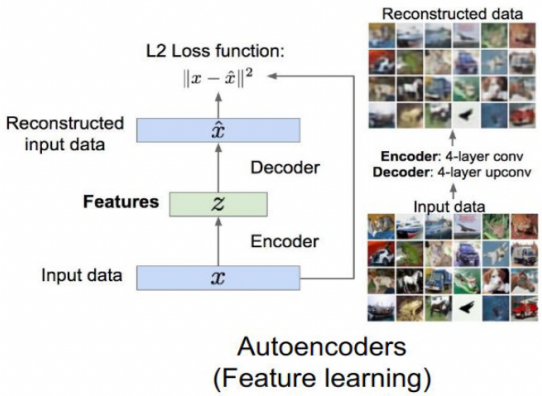
[13] https://arxiv.org/pdf/2102.13303.pdf

Overview

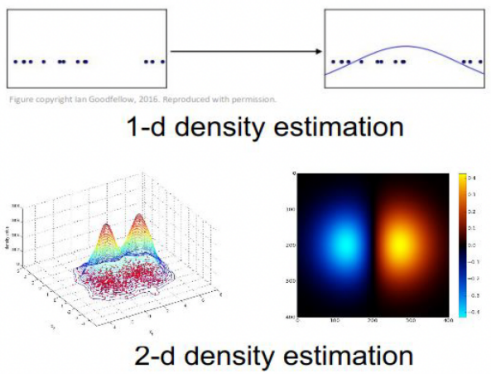
Unsupervised Learning

In unsupervised learning, we are interested in extracting information from the unlabeled data x_i (without using labels y_i).

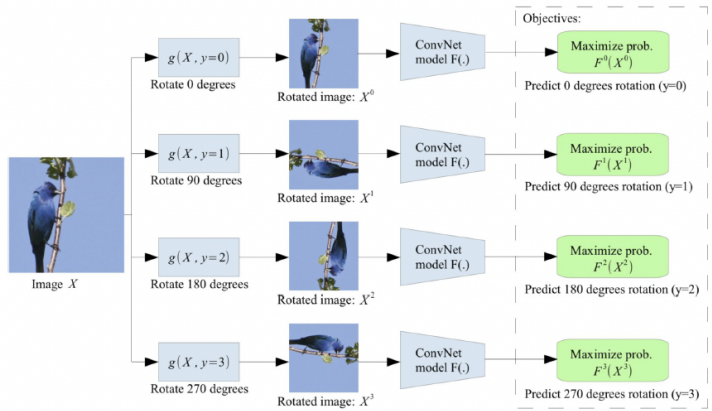
- Low-dimensional embeddings of the data : **Autoencoders**.
- Generative model able of approximating the data distribution $P_X(x)$: **Generative adversarial networks**
- **Self-supervision** : defining new training objectives based on the data itself to train the model.



Auto-encoder and dimensionality reduction



GaNs and density estimation



Self-supervision

Unsupervised Learning : Auto-encoders

Auto encoders can be used for self-supervised learning by either reconstructing the initial image, or only parts of it.

Sparse auto-encoders

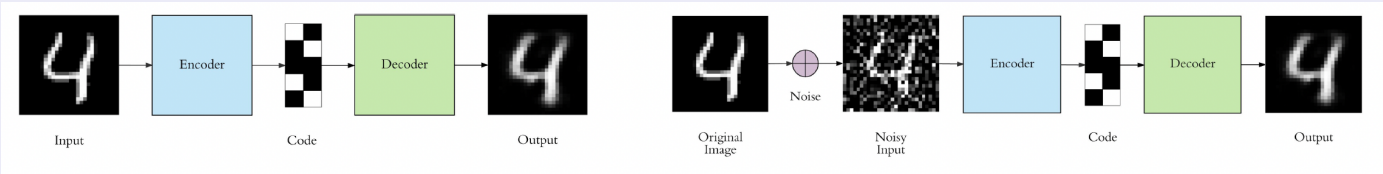
Find a sparse representation of the input data[a]

$$\min_{\boldsymbol{f},\boldsymbol{g}} \underbrace{\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}\left(\boldsymbol{x}_i, \boldsymbol{g}\left(\boldsymbol{h}_i\right)\right)}_{\text{loss}} + \underbrace{\lambda\left\|\boldsymbol{h}_i\right\|_1}_{\text{regularizer}} \quad \text{with} \quad \boldsymbol{h}_i = \boldsymbol{f}\left(\boldsymbol{x}_i\right), \text{ for all } i \in [n].$$

[a] https://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf

Denoising auto-encoders

The model must be robust to noise[a]



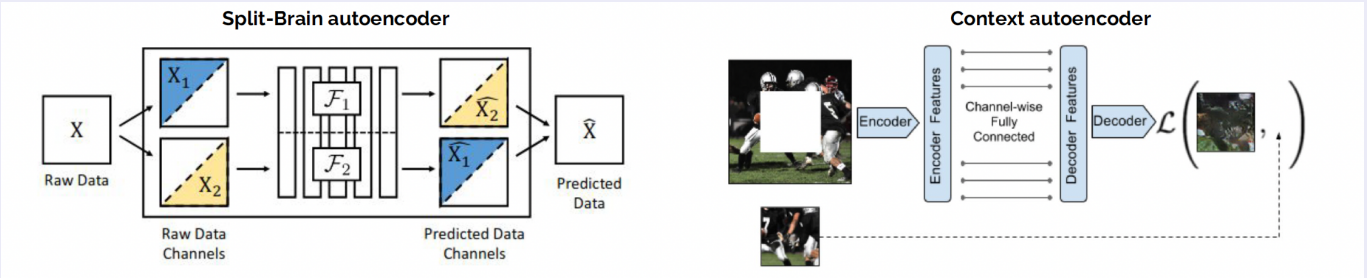
[a] https://www.jmlr.org/papers/volume11/vincent10a/vincent10a.pdf

Unsupervised Learning : Auto-encoders

Auto encoders can be used for self-supervised learning by either reconstructing the initial image, or only parts of it.

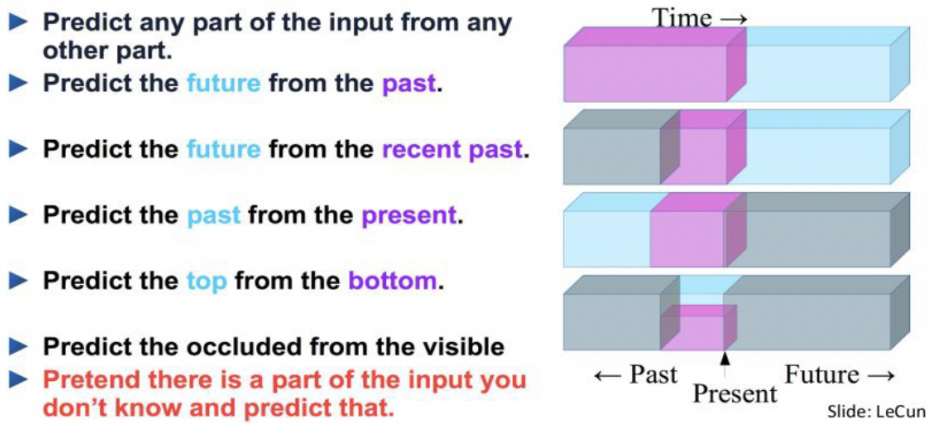
Other approaches[a]

[a] https://arxiv.org/abs/1611.09842, https://arxiv.org/abs/1604.07379



What is Self-Supervision ?

- A form of unsupervised learning where the **data provides the supervision**.
- Usually, define a pretext task for which the network is forced to learn what we really care about.
- For most pretext tasks, a part of the data is withheld and the network has to predict it.
- The features/representations learned on the pretext task are subsequently used for a different downstream task, usually where some annotations are available.



Source : LeCun talk

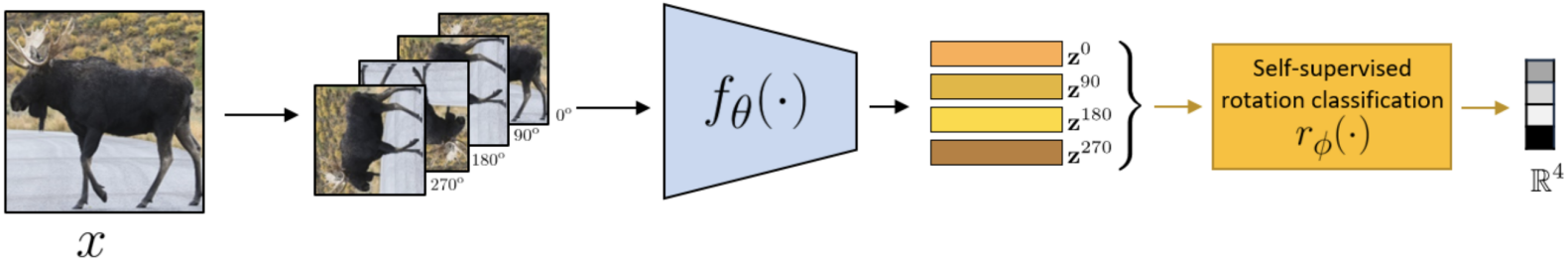
Self-Supervision

Goals

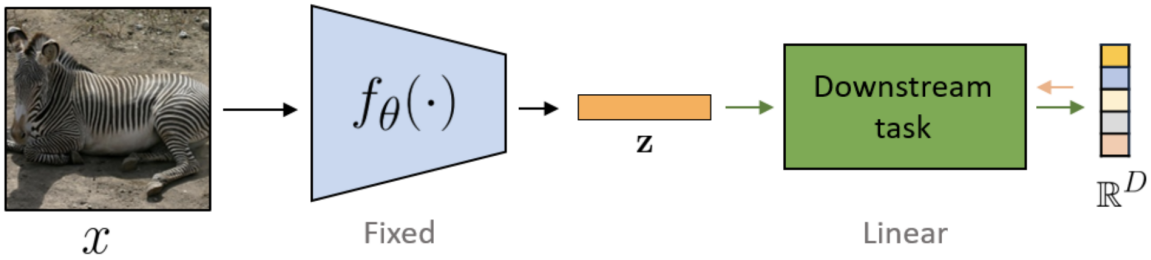
- Learn equally good (if not better) features without supervision.
- Be able to deploy similar quality systems without relying on too many labels for the downstream tasks.
- Generalize better potentially because you learn more about the world.

Self-Supervision

Stage 1: Train network on pretext task (without human labels)

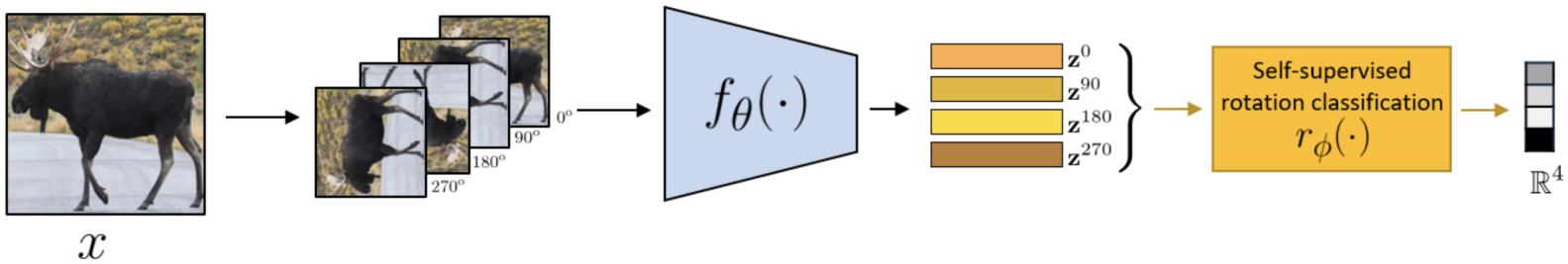


Stage 2: Train classifier on learned features for new task with fewer labels

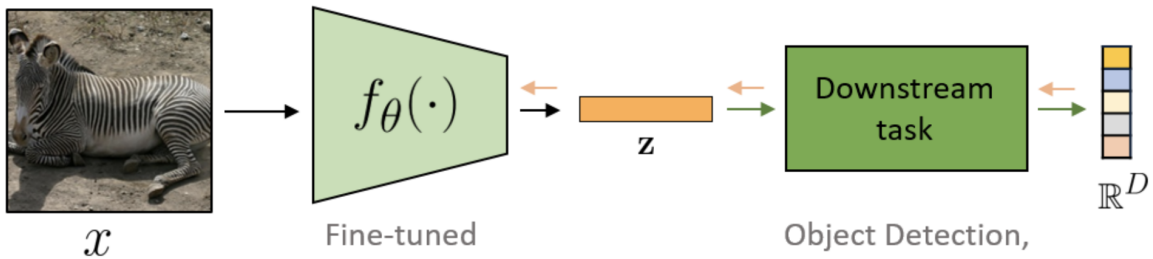


Self-Supervision

Stage 1: Train network on pretext task (without human labels)



Stage 2: Fine-tune network for new task with fewer labels

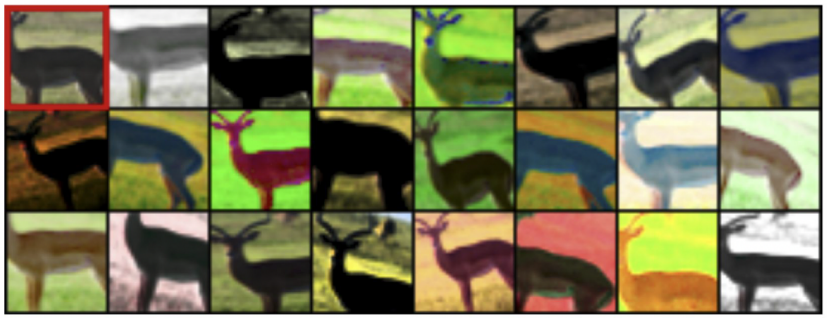


Self-Supervision : Image distorsions

Exemplar CNNs[a]

[a] https://arxiv.org/abs/1406.6909

Create a class by augmenting the same image multiple times.



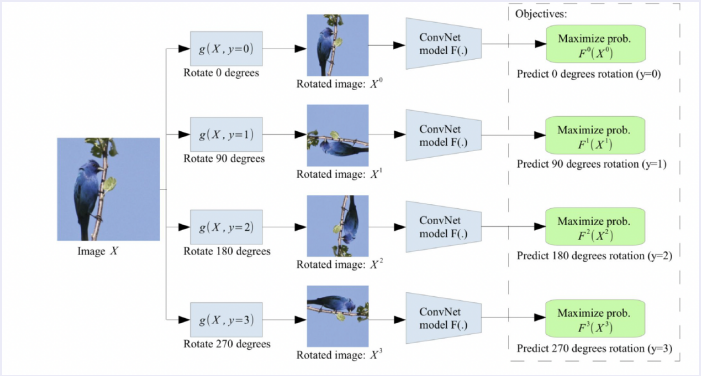
- Sample N patches of size 32×32 pixels from different images at varying positions and scales, only from regions containing considerable gradients as those areas cover edges and tend to contain objects or parts of objects. They are **exemplary** patches.
- Each patch is distorted by applying a variety of random transformations (i.e., translation, rotation, scaling, etc.). All the resulting distorted patches are considered to belong to the same surrogate class.
- The pretext task is to discriminate between a set of surrogate classes. We can arbitrarily create as many surrogate classes as we want.

Self-Supervision : Image distorsions

Rotation[a]

[a] https://arxiv.org/pdf/1803.07728.pdf

Create a class by augmenting the same image multiple times.



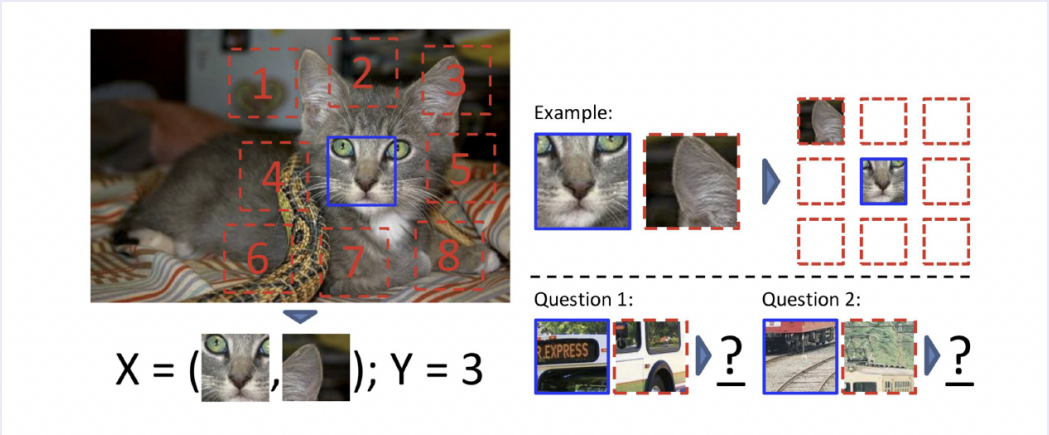
- Rotation of an entire image is another interesting and cheap way to modify an input image while the semantic content stays unchanged.
- Each input image is first rotated by a multiple of $90^{\check{r}}$ at random, corresponding to $[0, 90, 180, 270]$
- The model is trained to predict which rotation has been applied, thus a $4-$class classification problem.

# Rotations	Rotations	CIFAR-10 Classification Accuracy
4	0°, 90°, 180°, 270°	**89.06**
8	0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°	88.51
2	0°, 180°	87.46
2	90°, 270°	85.52

Self-Supervision : patches

Relative position[a]

[a] https://arxiv.org/abs/1505.05192



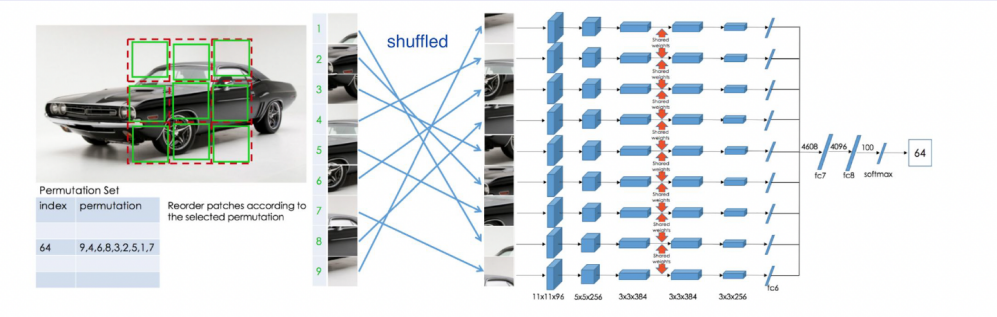
- Randomly sample the first patch without any reference to image content
- Considering that the first patch is placed in the middle of a 3×3 grid, and the second patch is sampled from its 8 neighbooring locations around it.
- The model is trained to predict which one of 8 neighbooring locations the second patch is selected from, a classification problem over 8 classes.

Self-Supervision : patches

Permutation prediction [a]

[a] https://arxiv.org/abs/1603.09246

Patches can be used in other ways, such as applying a given permutation from a set of fixed permutations, say 64 permutations, and the models predict which of these 64 permutation was applied

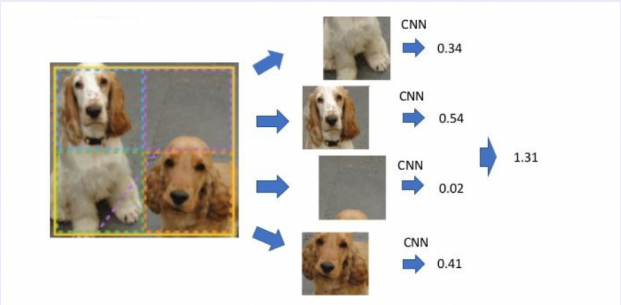


Self-Supervision : patches

Learning to count [a]

[a] https://arxiv.org/pdf/1708.06734.pdf

Another idea is to consider *feature* or *visual primitives* as a scalar-value attribute that can be summed up over multiple patches and compared across different patches. Then the relationship between patches can be defined by counting features and simple arithmetic.



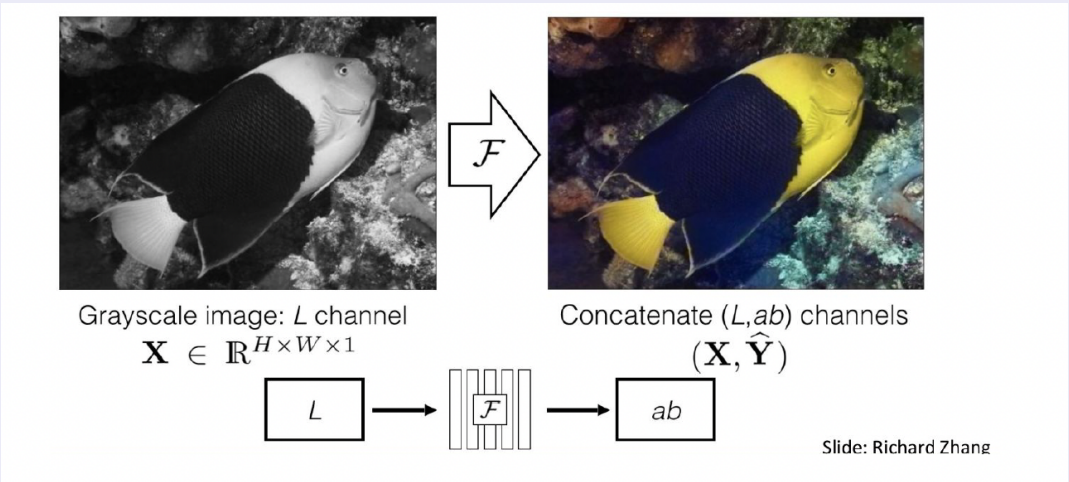
Self-Supervision : Colorization

Colorization [a]

[a] https://arxiv.org/abs/1603.08511

Colorization can be used as a powerful self-supervised task.

- A model is trained to color a grayscale input image
- the task is to map this image to a distribution over quantized color value outputs

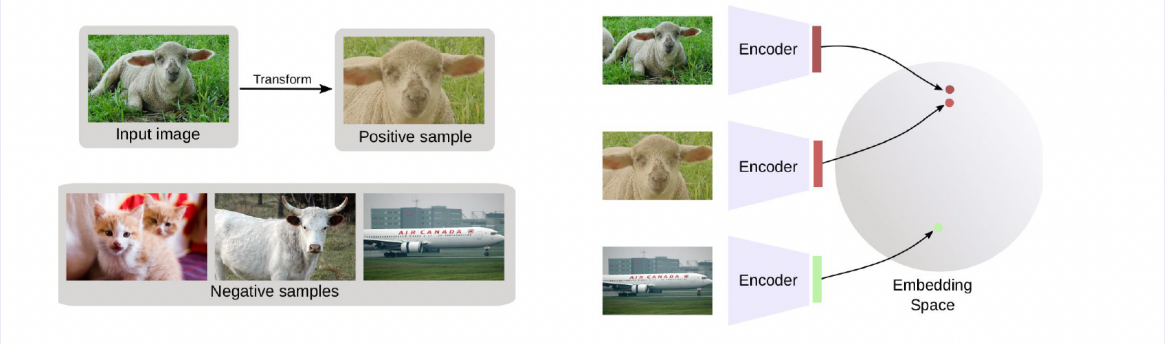


- The model outputs colors in the Lab color space. The Lab color is designed to approximate human vision, while, in contrast, RGB or CMYK models the color output of physical devices.

Self-Supervision : Contrastive Learning

Contrastive Learning

Contrastive learning aims at embedding augmented versions of the same sample close to each other while trying to push away embeddings from different samples.



Objective

- Learn a a deep neural network, that maps semantically nearby points (i.e. **positive pairs**) closer together in the embedding space, while pushing apart points that are dissimilar (i.e. **negative pairs**)
- One of the major design choices in contrastive learning is how to select the positive and negative pairs.
- The standard approach for generating positive pairs without additional annotations is to create multiple views of each example, for instance, splitting an image into luminance and chrominance, applying different random crops and data augmentations, or using different patches within a single image.
- Negative pairs, on the other hand, can be generated by randomly sampling images and

Self-Supervision : Contrastive Learning

Contrastive Learning Losses

Contrastive loss (Chopra et al, 2015)[a]

[a] http://yann.lecun.com/exdb/publis/pdf/chopra-05.pdf

- Given a list of input samples $\{\mathbf{x}_i\}$, each has a corresponding label $y_i \in \{1, \ldots, L\}$ among L classes
- We would like to learn a function $f_\theta(.) : \mathcal{X} \to \mathbb{R}^d$ that encode x_i into an embedding vector such that examples from the same class have similar embeddings and samples from different classes have very different ones.
- Contrastive loss takes a pair of inputs (x_i, x_j) and minimizes the embedding distance when they are from the same class but maximizes the distance otherwise

$$\mathcal{L}_{\text{cont}}(\mathbf{x}_i, \mathbf{x}_j, \theta) = \mathbb{1}[y_i = y_j] \|f_\theta(\mathbf{x}_i) - f_\theta(\mathbf{x}_j)\|_2^2 + \mathbb{1}[y_i \neq y_j] \max(0, \epsilon - \|f_\theta(\mathbf{x}_i) - f_\theta(\mathbf{x}_j)\|_2)^2$$

Self-Supervision : Contrastive Learning

Contrastive Learning Losses

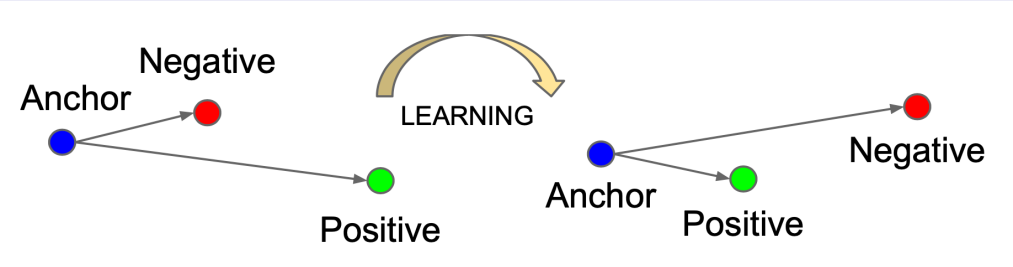
Triplet Loss (Schroff et al, 2015) [a]

[a] https://arxiv.org/abs/1503.03832

- Given one given anchor $\{\mathbf{x}\}$, we select one positive sample $\mathbf{x}^+$ (same class) and one negative sample $\mathbf{x}^-$ (another different class)
- Triplet loss learns to minimize the distance between the anchor $\{\mathbf{x}\}$ and positive $\mathbf{x}^+$ and maximize the distance between the anchor $\{\mathbf{x}\}$ and negative $\mathbf{x}^-$ at the same time

$$\mathcal{L}_{\text{triplet}}(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-) = \sum_{\mathbf{x} \in \mathcal{X}} \max\left(0, \|f(\mathbf{x}) - f(\mathbf{x}^+)\|_2^2 - \|f(\mathbf{x}) - f(\mathbf{x}^-)\|_2^2 + \epsilon\right)$$

- It is crucial to select challenging $\mathbf{x}^-$ to improve the model.



Self-Supervision : Contrastive Learning

Contrastive Learning Losses

Lifted Structured Loss (Song et al, 2015)[a]

[a] https://arxiv.org/abs/1511.06452

Use of all the pairwise edges within one training batch for better computational efficiency.

- Given $\mathcal{P}$ the set of positive pairs and $\mathcal{N}$ the set of negative pairs
- $D_{ij} = |f(\mathbf{x}_i) - f(\mathbf{x}_j)|_2$
- The structured loss function is

$$\mathcal{L}_{\text{struct}} = \frac{1}{2|\mathcal{P}|} \sum_{(i,j)\in\mathcal{P}} \max(0, \mathcal{L}_{\text{struct}}^{(ij)})^2$$

$$\text{where } \mathcal{L}_{\text{struct}}^{(ij)} = D_{ij} + \max\left(\max_{(i,k)\in\mathcal{N}} \epsilon - D_{ik}, \max_{(j,l)\in\mathcal{N}} \epsilon - D_{jl} \right)$$

- The red part is used for hard negative mining and as it is not smooth, it is relaxed to prevent to converge to a bad local optimum

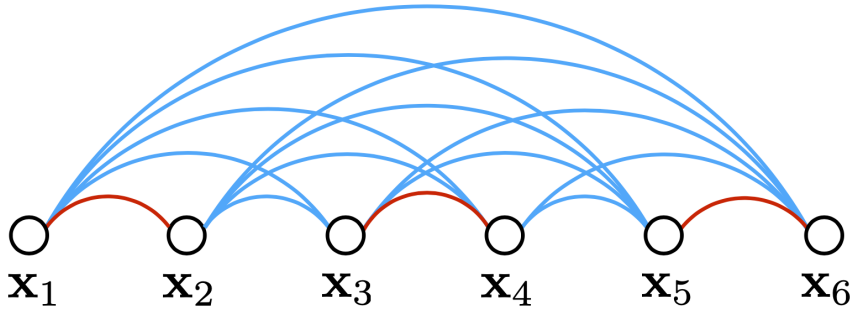
$$\mathcal{L}_{\text{struct}}^{(ij)} = D_{ij} + \log\left(\sum_{(i,k)\in\mathcal{N}} \exp(\epsilon - D_{ik}) + \sum_{(j,l)\in\mathcal{N}} \exp(\epsilon - D_{jl}) \right)$$

Self-Supervision : Contrastive Learning

Contrastive Learning Losses

(a) Contrastive embedding

(b) Triplet embedding



(c) Lifted structured embedding

Self-Supervision : Contrastive Learning

Contrastive Learning Losses

N-pair Loss (Sohn et al, 2016)[a]

[a] https://papers.nips.cc/paper/2016/file/6b180037abbebea991d8b1232f8a8ca9-Paper.pdf

Generalizes triplet loss to include comparison with multiple negative samples

- Given a $(N+1)-$tuplet of training samples $\{\mathbf{x}, \mathbf{x}^+, \mathbf{x}_1^-, \ldots, \mathbf{x}_{N-1}^-\}$ including one positive and $N-1$ negatives ones.
- $N-$pair loss is defined as

$$\mathcal{L}_{\text{N-pair}}(\mathbf{x}, \mathbf{x}^+, \{\mathbf{x}_i^-\}_{i=1}^{N-1}) = \log\left(1 + \sum_{i=1}^{N-1} \exp(f(\mathbf{x})^\top f(\mathbf{x}_i^-) - f(\mathbf{x})^\top f(\mathbf{x}^+))\right)$$

$$= -\log \frac{\exp(f(\mathbf{x})^\top f(\mathbf{x}^+))}{\exp(f(\mathbf{x})^\top f(\mathbf{x}^+)) + \sum_{i=1}^{N-1} \exp(f(\mathbf{x})^\top f(\mathbf{x}_i^-))}$$

- If we only sample one negative sample per class, it is equivalent to the softmax loss for multi-class classification.

Self-Supervision : Contrastive Learning

InfoNCE - Noise Contrastive Estimation (van den Oord, 2018)

- Given a context vector $\mathbf{c}$, the positive sample should be drawn from the conditional distribution $p(\mathbf{x}|\mathbf{c})$
- $N-1$ negative samples are drawn from the proposal distribution $p(\mathbf{x})$ independant from the context.
- We label all the samples $X = \{\mathbf{x}_i\}_{i=1}^N$ among which one of them is a positive sample $\mathbf{x}_{\text{pos}}$
- The probability of detecting the positive sample correctly is :

$$p(C = \text{pos}|X, \mathbf{c}) = \frac{p(x_{\text{pos}}|\mathbf{c}) \prod_{i=1,\dots,N;i\neq \text{pos}} p(\mathbf{x}_i)}{\sum_{j=1}^N \left[p(\mathbf{x}_j|\mathbf{c}) \prod_{i=1,\dots,N;i\neq j} p(\mathbf{x}_i) \right]} = \frac{\frac{p(\mathbf{x}_{\text{pos}}|c)}{p(\mathbf{x}_{\text{pos}})}}{\sum_{j=1}^N \frac{p(\mathbf{x}_j|\mathbf{c})}{p(\mathbf{x}_j)}} = \frac{f(\mathbf{x}_{\text{pos}}, \mathbf{c})}{\sum_{j=1}^N f(\mathbf{x}_j, \mathbf{c})}$$

  with the scoring function $f(\mathbf{x}, \mathbf{c}) \propto \frac{p(\mathbf{x}|\mathbf{c})}{p(\mathbf{x})}$

- The InfoNCE loss optimizes the negative log probability of classifying the positive sample correctly:

$$\mathcal{L}_{\text{InfoNCE}} = -\mathbb{E}\left[\log \frac{f(\mathbf{x}, \mathbf{c})}{\sum_{\mathbf{x}' \in X} f(\mathbf{x}', \mathbf{c})} \right]$$

- Connection with mutual information optimization

$$I(\mathbf{x}; \mathbf{c}) = \sum_{\mathbf{x},\mathbf{c}} p(\mathbf{x}, \mathbf{c}) \log \frac{p(\mathbf{x}, \mathbf{c})}{p(\mathbf{x})p(\mathbf{c})} = \sum_{\mathbf{x},\mathbf{c}} p(\mathbf{x}, \mathbf{c}) \log \frac{p(\mathbf{x}|\mathbf{c})}{p(\mathbf{x})}$$

See : https://arxiv.org/abs/1807.03748

Self-Supervision : Contrastive Learning

Common setup (Wang et al, 2020)

- Let $p_{\text{data}}(.)$ the data distribution over $\mathbb{R}^n$ and $p_{\text{pos}}(.,.)$ the distribution of positive pairs over $\mathbb{R}^{n \times n}$
- These two distributions should satisfy
  - Symmetry : $\forall \mathbf{x}, \mathbf{x}^+, p_{\text{pos}}(\mathbf{x}, \mathbf{x}^+) = p_{\text{pos}}(\mathbf{x}^+, \mathbf{x})$
  - Matching marginal : $\forall \mathbf{x}, \int p_{\text{pos}}(\mathbf{x}, \mathbf{x}^+) d\mathbf{x}^+ = p_{\text{data}}(\mathbf{x})$

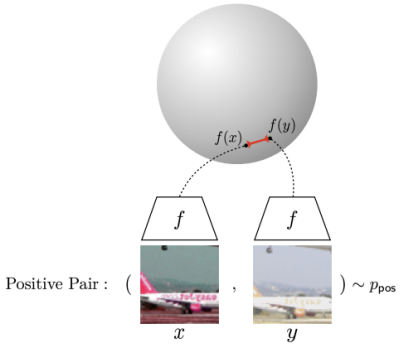
$$
\begin{aligned}
\mathcal{L}_{\text{contrastive}} &= \mathbb{E}_{(\mathbf{x},\mathbf{x}^+)\sim p_{\text{pos}},\{\mathbf{x}_i^-\}_{i=1}^M \overset{\text{i.i.d}}{\sim} p_{\text{data}}} \Big[-\log \frac{\exp(f(\mathbf{x})^\top f(\mathbf{x}^+)/\tau)}{\exp(f(\mathbf{x})^\top f(\mathbf{x}^+)/\tau) + \sum_{i=1}^M \exp(f(\mathbf{x})^\top f(\mathbf{x}_i^-)/\tau)} \Big] \\
&\approx \mathbb{E}_{(\mathbf{x},\mathbf{x}^+)\sim p_{\text{pos}},\{\mathbf{x}_i^-\}_{i=1}^M \overset{\text{i.i.d}}{\sim} p_{\text{data}}} \Big[-f(\mathbf{x})^\top f(\mathbf{x}^+)/\tau + \log \Big(\sum_{i=1}^M \exp(f(\mathbf{x})^\top f(\mathbf{x}_i^-)/\tau) \Big) \Big] \qquad \text{; Assuming infinite negatives} \\
&= -\frac{1}{\tau} \mathbb{E}_{(\mathbf{x},\mathbf{x}^+)\sim p_{\text{pos}}} f(\mathbf{x})^\top f(\mathbf{x}^+) + \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \Big[\log \mathbb{E}_{\mathbf{x}^-\sim p_{\text{data}}} \big[\sum_{i=1}^M \exp(f(\mathbf{x})^\top f(\mathbf{x}_i^-)/\tau) \big] \Big]
\end{aligned}
$$

(Wang et al, 2020) Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere[14]

[14] https://arxiv.org/abs/2005.10242

Self-Supervision : Contrastive Learning

- Two key properties related to the contrastive loss :
  - Alignement : Similar samples have similar features
  - Uniformity: Preserve maximal information
- The contrastive loss optimizes these properties.
- They have a positive effects on the downstream tasks.



Alignment: Similar samples have similar features.
(Figure inspired by Tian et al. (2019).)

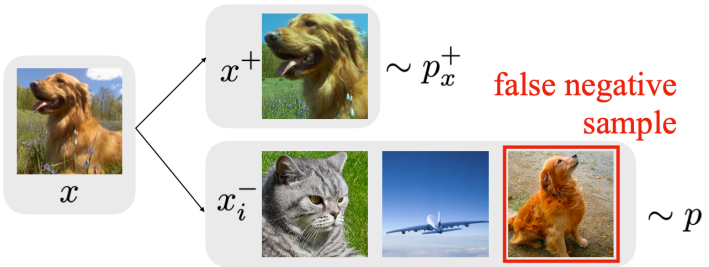
Uniformity: Preserve maximal information.

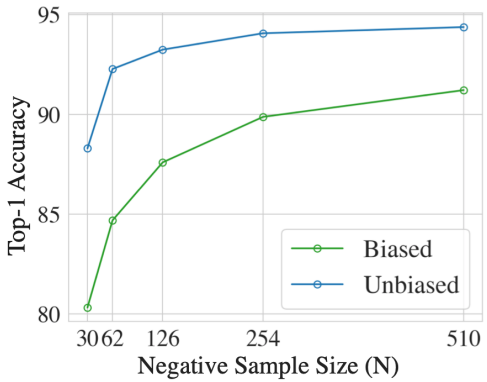
Figure 1: Illustration of alignment and uniformity of feature distributions on the output unit hypersphere. STL-10 (Coates et al., 2011) images are used for demonstration.

Self-Supervision : Contrastive Learning

Key ingredients

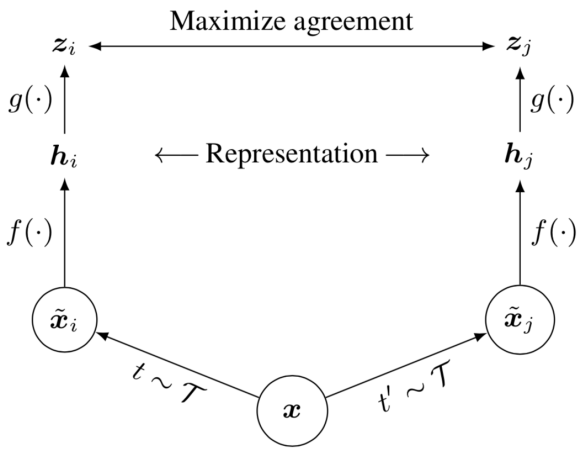
- **Heavy Data Augmentation** : introduces the non-essential variations into examples without modifying semantic meanings and thus encourages the model to learn the essential part of the representation.
- **Large Batch Size** : especially when it relies on in-batch negatives to enable the loss function can cover a diverse enough collection of negative samples.
- **Hard negative Mining** : Hard negative samples should have different labels from the anchor sample, but have embedding features very close to the anchor embedding





Self-Supervision : Contrastive Learning

SimCLR (Chen et al, 2020)[15] : Learns representations for visual inputs by maximizing agreement between differently augmented views of the same sample via a contrastive loss in the latent space



- Randomly sample a minibatch of N samples.
- Each sample is applied with two different data augmentations operations (same family) : $2N$ data points
- Given one positive pair, other $2(N-1)$ points are negative samples, the representation is produced by a base encoder $f(.)$

$$\mathbf{h}_i = f(\tilde{\mathbf{x}}_i), \quad \mathbf{h}_j = f(\tilde{\mathbf{x}}_j)$$

- The contrastive learning loss is defined using cosine similarity sim(., .)

$$\mathbf{z}_i = g(\mathbf{h}_i), \quad \mathbf{z}_j = g(\mathbf{h}_j)$$

$$\mathcal{L}_{\text{SimCLR}}^{(i,j)} = -\log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)}$$

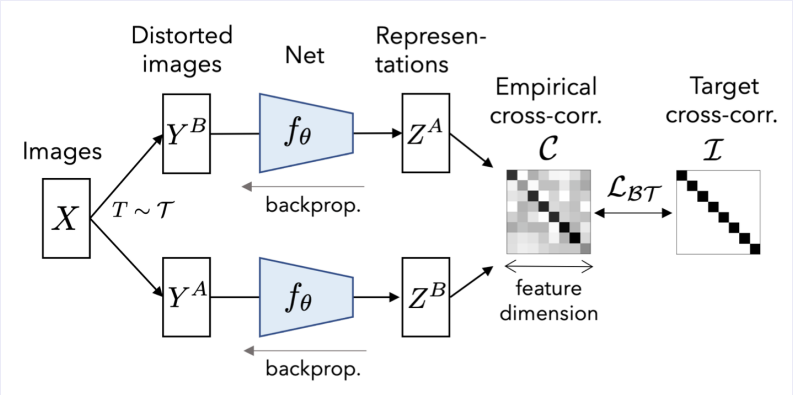
- The representation **h** is used for downstream tasks (it requires a different post-processing head)

[15] https://arxiv.org/abs/2002.05709

Self-Supervision : Contrastive Learning

And many other approaches

- Barlow Twins (Zbontar et al. 2021)[a] feeds two distorted versions of samples into the same network to extract features and learns to make the cross-correlation matrix between these two groups of output features close to the identity.



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

$$\text{where } \mathcal{C}_{ij} = \frac{\sum_b \mathbf{z}^A_{b,i} \mathbf{z}^B_{b,j}}{\sqrt{\sum_b (\mathbf{z}^A_{b,i})^2} \sqrt{\sum_b (\mathbf{z}^B_{b,j})^2}}$$

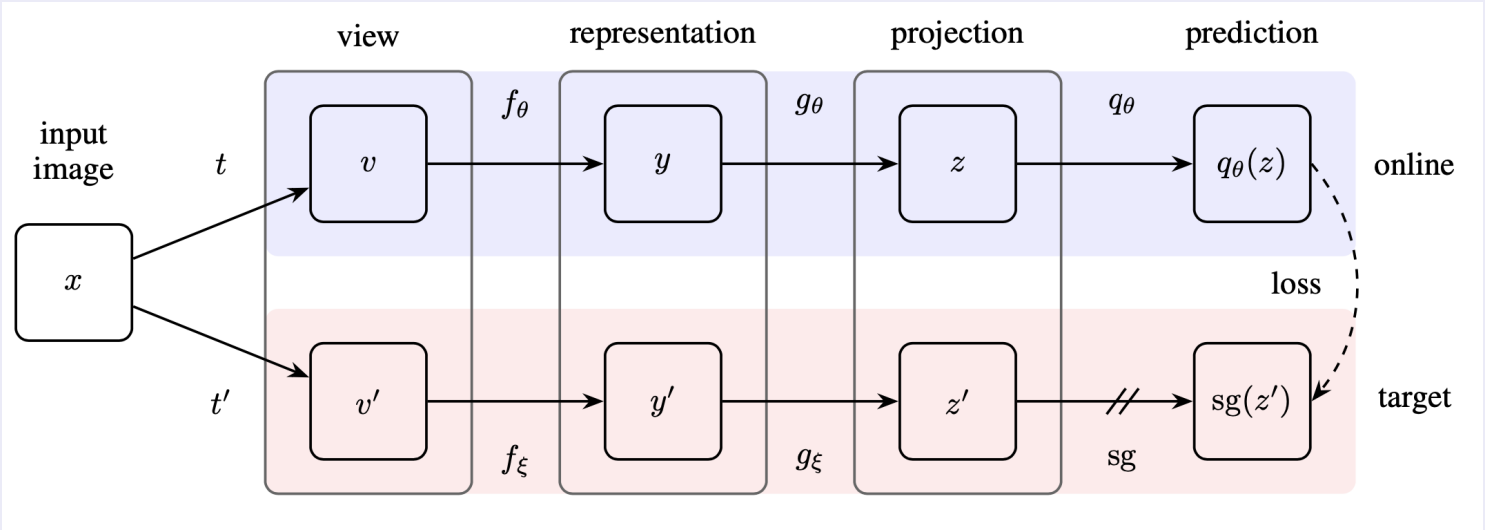
$\mathcal{C}_{ij}$ cosine similarity between network output vector dimension at index i, j and batch index b.

[a] https://arxiv.org/abs/2103.03230

Self-Supervision : Contrastive Learning

And many other approaches

- BYOL (Bootstrap Your Own Latent; Grill, et al 2020)[a] which not use **negative samples**. It relies on two neural networks, referred to as **online** and **target** networks that interact and learn from each other. The target network (parameterized by ξ) has the same architecture as the online one (parameterized by θ), but with polyak averaged weights $\xi \leftarrow \tau\xi + (1-\tau)\theta$
- The goal is to learn a representation y that can be used in downstream tasks.
- The online network parameterized by θ contains: an encoder f_θ, a projector g_θ and a predictor q_θ.
- $\mathcal{L}_\theta^{\text{BYOL}}$ is MSE between L2-normalized prediction $\bar{q}_\theta(\mathbf{z})$ and $\bar{\mathbf{z}}'$
- Symmetric loss $\tilde{\mathcal{L}}_\theta^{\text{BYOL}}$ by switching views and final loss $\mathcal{L}_\theta^{\text{BYOL}} + \tilde{\mathcal{L}}_\theta^{\text{BYOL}}$



[a] https://arxiv.org/abs/2006.07733

Self-Supervision : Contrastive Learning

And many other approaches

- BYOL (Bootstrap Your Own Latent; Grill, et al 2020) : not use **negative samples**. It relies on two neural networks, referred to as **online** and **target** networks that interact and learn from each other. The target network
- BYOL generally performs no better than random when batch normalization is removed.[a]
- Interpretation : presence of batch normalization implicitly causes a form of contrastive learning : using negative samples is important for avoiding model collapse.

---

[a] https://generallyintelligent.ai/blog/2020-08-24-understanding-self-supervised-contrastive-learning/

Self-Supervision : Contrastive Learning

And many other approaches

- Some approaches have been proposed to improve the computing embeddings for a large number of negative samples : with memory bank (Instance contrastive learning (Wu et al, 2018)[a]), with a dynamic dictionary look-up (Momentum Contrast (MoCo; He et al, 2019)[b])
- Evaluation : Transfer learning of a lot of downstream tasks.
- Some frameworks : A library for state-of-the-art self-supervised learning from images `https://vissl.ai/`

---

[a] `https://arxiv.org/abs/1805.01978`
[b] `https://arxiv.org/abs/1911.05722`

Overview

Motivations

(Lake et al, 2015) Human-level concept learning through probabilistic program induction[16]

People learning new concepts can often generalize successfully from just a single example, yet machine learning algorithms typically require tens or hundreds of examples to perform with similar accuracy

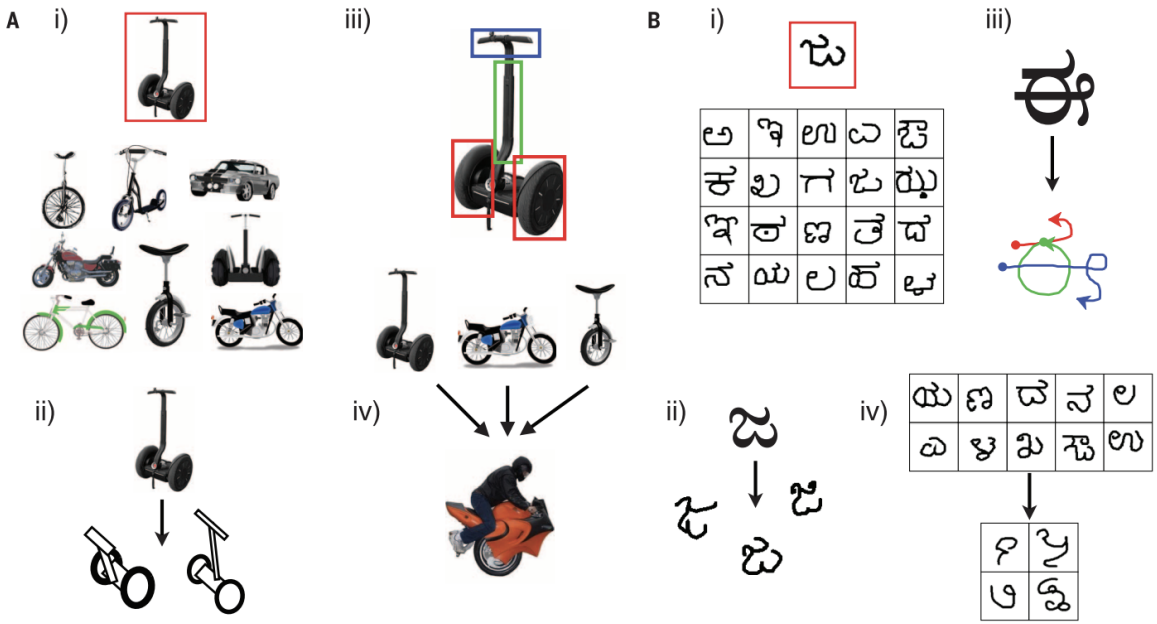


Fig. 1. People can learn rich concepts from limited data. (**A** and **B**) A single example of a new concept (red boxes) can be enough information to support the (i) classification of new examples, (ii) generation of new examples, (iii) parsing an object into parts and relations (parts segmented by color), and (iv) generation of new concepts from related concepts. [Image credit for (A), iv, bottom: With permission from Glenn Roberts and Motorcycle Mojo Magazine]

[16] https://www.cs.cmu.edu/~rsalakhu/papers/LakeEtAl2015Science.pdf

Motivations

Few-shot learning

Equip the learner with ability to rapidly learn new concept with few training samples.

?????

Motivations

Few-shot learning

Equip the learner with ability to rapidly learn new concept with few training samples.

?????

Elephant shrew.

Motivations

Few-shot learning

Equip the learner with ability to rapidly learn new concept with few training samples.

Elephant shrew rediscovered in Africa after 50 years

By Helen Briggs
BBC Environment correspondent

18 August 2020

The animal is small enough to fit in the palm of your hand

A little-known mammal related to an elephant but as small as a mouse has been rediscovered in Africa after 50 years of obscurity.

Motivations

Few-shot learning

Equip the learner with ability to rapidly learn new concept with few training samples.

Elephant Shrew

Few training samples

Motivations

"Few-Shot Learning", why do we need it?

some applications ...

Brand Logos

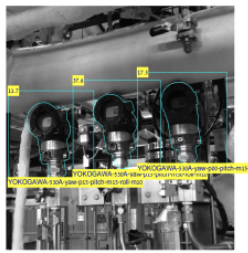
new brands can be added on the fly

with just one or two examples

Smart appliances

Retail

Industrial



Catering

Few shot learning : general setting

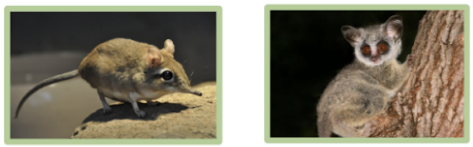
A few-shot classification task

Each classification task consists of a training set, also called support set, and a testing set, also called query set. A $N-$way $K-$shot classification task has :

- N: number of classes
- K: number of examples per class (small)

5–way (classes) 1–shot (example per class) Task

Train set (support set)

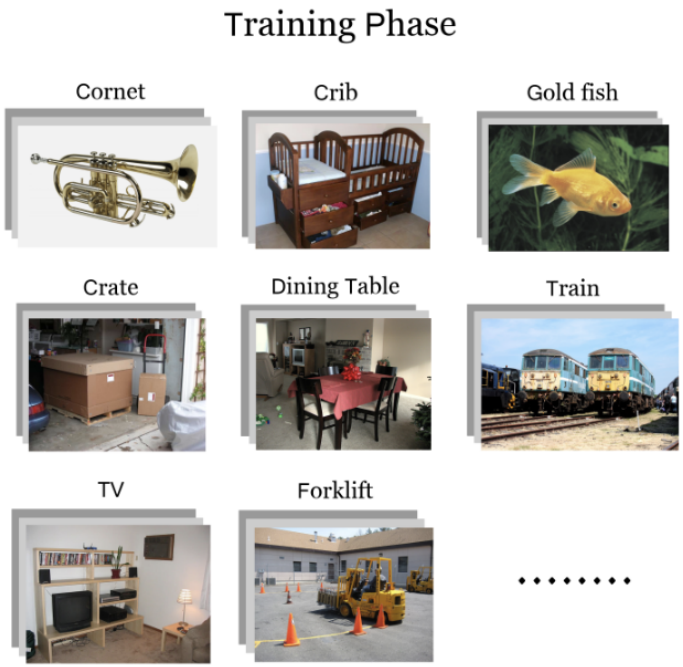


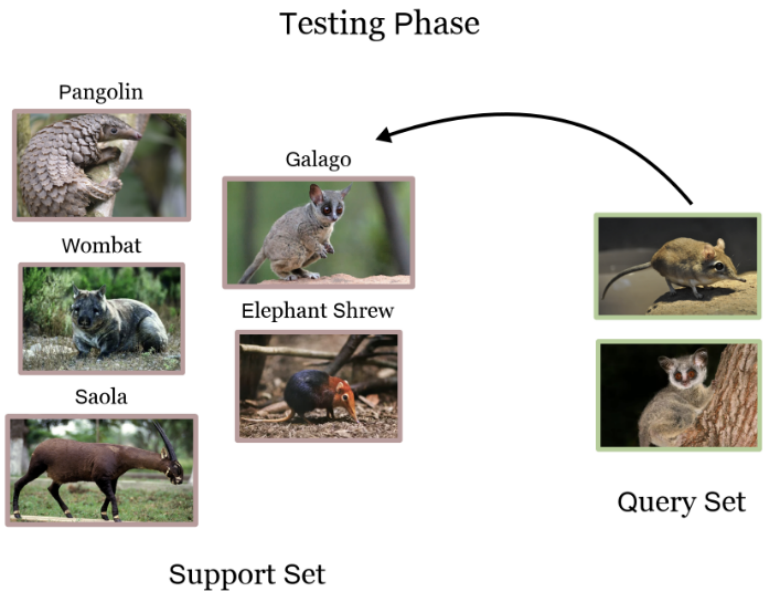
Test set (query set)

Few shot learning : general setting

Few shot classification usually involves two stage :

- a **training phase** where we have access to a meta training set (or base set) with a large number of classes.
- a **testing phase** where we evaluate the learner on a set of tasks with novel classes not seen during training.



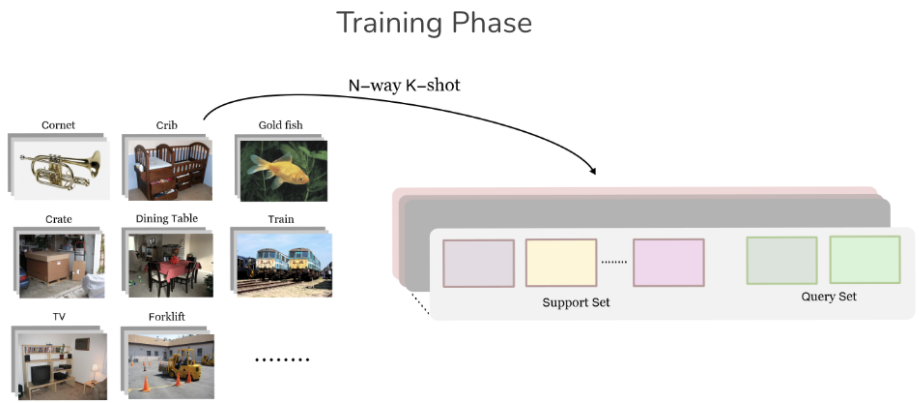


Meta-learning and Episodic Training

Principle

Match the training environment to the testing environment and learn to avoid overfitting. The learner in trained on a number of sampled tasks created from the meta training set.
Meta-learner : trained over a variety of learning tasks (represented by a dataset $\mathcal{D}$) and optimized for the best performance on a distribution of tasks, including potentially unseen tasks.

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{\mathcal{D}\sim p(\mathcal{D})}[\mathcal{L}_\theta(\mathcal{D})]$$



Meta-learning and Episodic Training

Classical Learning

- A dataset $\mathcal{D} = \{x_i, y_i\}$, pairs of feature vectors and labels with each label belongs to a known label set $\mathcal{L}^{\text{label}}$.
- A classifier f_θ with parameter θ outputs a probability of a data point belonging to the class y given the feature vector $\mathbf{x}$: $P_\theta(y|\mathbf{x})$
- The optimal parameters should maximize the probability of true labels across multiple training batches $B \subset \mathcal{D}$

$$\theta^* = \arg\max_\theta \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}}[P_\theta(y|\mathbf{x})]$$

$$\theta^* = \arg\max_\theta \mathbb{E}_{B\subset\mathcal{D}}[\sum_{(\mathbf{x},y)\in B} P_\theta(y|\mathbf{x})] \quad \text{; trained with mini-batches.}$$

Meta-learning and Episodic Training

Few shot Learning

"Fake" datasets with a subset of labels to avoid exposing all the labels to the model and modify the optimization procedure accordingly to encourage fast learning:

- Sample a subset of labels $L \subset \mathcal{L}^{\text{label}}$
- Sample a support set $S^L \subset \mathcal{D}$ and a training batch $B^L \subset \mathcal{D}$ that only contain data points with labels belonging to L
- Do several sampling : each pair of sampled dataset (S^L, B^L) will be consider as one data point

$$\theta = \arg\max_{\theta} \mathbb{E}_{L \subset \mathcal{L}}[\mathbb{E}_{S^L \subset \mathcal{D}, B^L \subset \mathcal{D}}[\sum_{(x,y) \in B^L} P_\theta(x, y, S^L)]]$$

Meta-learning and Episodic Training

Common approaches

Depends on how $P_\theta(y|\mathbf{x})$ is modeled.

- Metric-based : The predicted probability over a set of known labels y is a weighted sum of labels of support set samples.
- Model-based : No assumption on the form of $P_\theta(y|\mathbf{x})$: model designed specifically for fast learning.
- Optimization-based : Adjust gradient-based optimization algorithm so that the model can be good at learning with a few examples.

Metric-based approches for FSL

Principle

The predicted probability over a set of known labels y is a weighted sum of labels of support set samples. The weight is generated by a kernel function k_θ , measuring the similarity between two data samples.

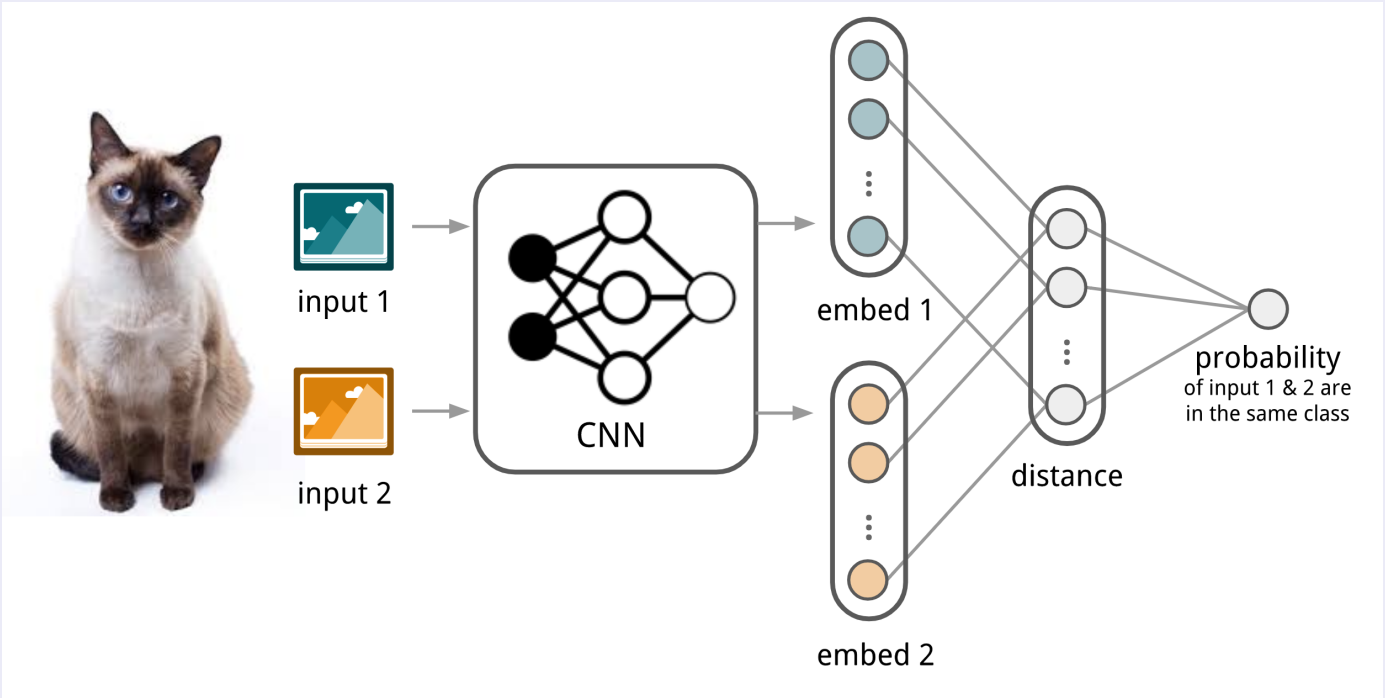
$$P_\theta(y|\mathbf{x}, S) = \sum_{(\mathbf{x}_i, y_i) \in S} k_\theta(\mathbf{x}, \mathbf{x}_i) y_i$$

Principle : learn embedding vectors of input data explicitly and use them to design proper kernel functions.

Metric-based approches for FSL

Convolutional Siamese Neural Network (Koch et al, 2015) [a]

[a] http://www.cs.toronto.edu/~rsalakhu/papers/oneshot1.pdf



Given a support set S and a test image $\mathbf{x}$, the final predicted class is:

$$\hat{c}_S(\mathbf{x}) = c(\arg\max_{\mathbf{x}_i \in S} P(\mathbf{x}, \mathbf{x}_i))$$

Metric-based approches for FSL

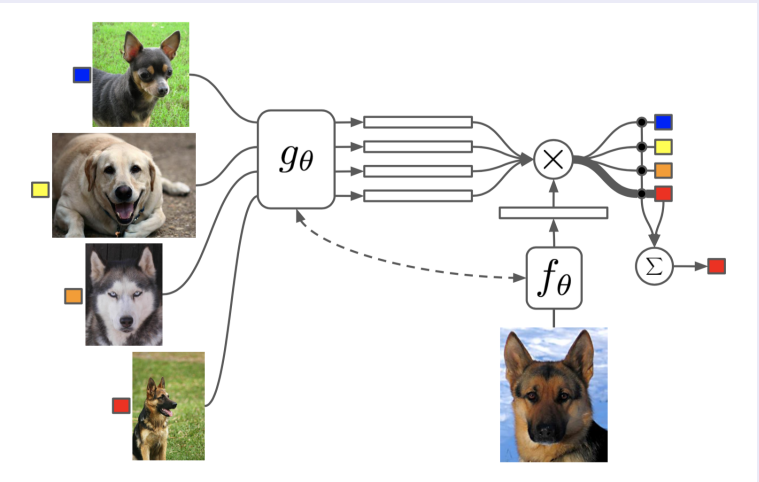
Matching Network (Vinyals et al , 2015) [a]

[a] https://proceedings.neurips.cc/paper/2016/file/90e1357833654983612fb05e3ec9148c-Paper.pdf

Goal : learn a classifier c_S for any given small support set $S = \{x_i, y_i\}_{i=1}^k$. The classifier output is a sum of labels of support samples weighted by attention kernel $a(\mathbf{x}, \mathbf{x}_i)$ proportional to the similarity between $\mathbf{x}$ and $\mathbf{x_i}$

$$c_S(\mathbf{x}) = P(y|\mathbf{x}, S) = \sum_{i=1}^{k} a(\mathbf{x}, \mathbf{x}_i) y_i, \text{ where } S = \{(\mathbf{x}_i, y_i)\}_{i=1}^k$$

$$a(\mathbf{x}, \mathbf{x}_i) = \frac{\exp(\cos(f(\mathbf{x}), g(\mathbf{x}_i))}{\sum_{j=1}^{k} \exp(\cos(f(\mathbf{x}), g(\mathbf{x}_j))}$$



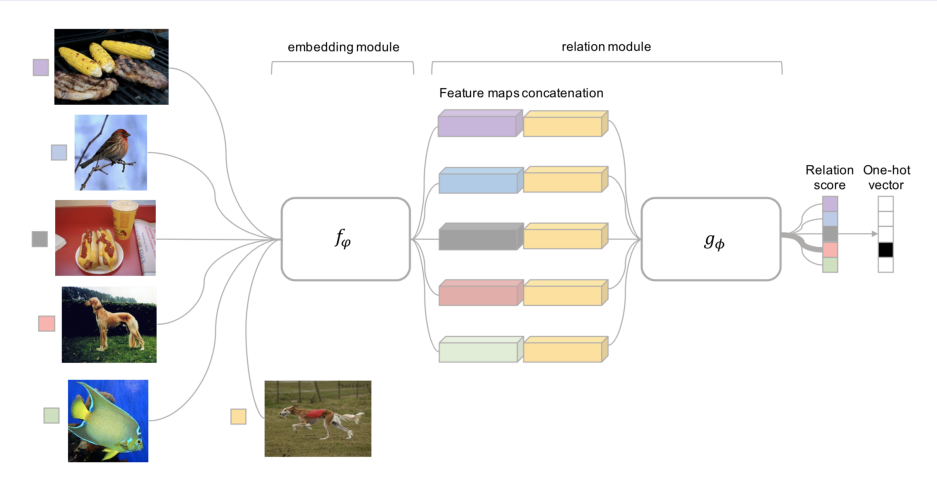
Metric-based approches for FSL

Relation Network (Sung et al , 2018) [a]

[a] https://openaccess.thecvf.com/content_cvpr_2018/papers_backup/Sung_Learning_to_Compare_CVPR_2018_paper.pdf

Similar to siamese network but :

- The relationship is not captured by a simple distance in the feature space, but predicted by a CNN classifier g_ϕ
- The objective function is MSE loss instead of cross-entropy, because conceptually RN focuses more on predicting relation scores which is more like regression, rather than binary classification.



Metric-based approches for FSL

Prototycal Network (Snell et al , 2017) [a]

[a] hhttps://proceedings.neurips.cc/paper/2017/file/cb8da6767461f2812ae4290eac7cbc42-Paper.pdf

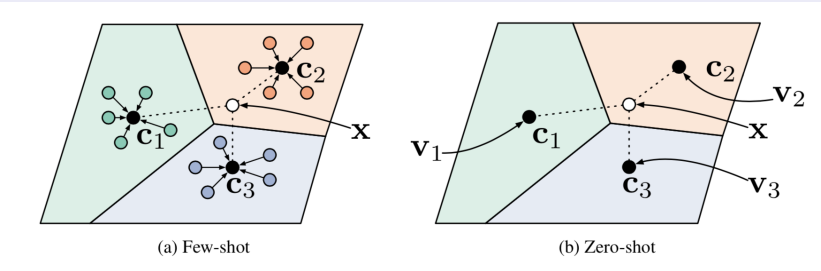
- An embedding function f_θ is used to encode each input into a $M-$dimensional feature vector.
- A prototype feature vector is defined for every class as the mean vector of the embedded support data samples in this class.

$$\mathbf{v}_c = \frac{1}{|S_c|} \sum_{(\mathbf{x}_i, y_i) \in S_c} f_\theta(\mathbf{x}_i)$$

- The distribution over classes for a given test input $\mathbf{x}$ is a softmax over the inverse of distances between the test data embedding and prototype vectors.

$$P(y = c|\mathbf{x}) = \text{softmax}(-d_\varphi(f_\theta(\mathbf{x}), \mathbf{v}_c)) = \frac{\exp(-d_\varphi(f_\theta(\mathbf{x}), \mathbf{v}_c))}{\sum_{c' \in \mathcal{C}} \exp(-d_\varphi(f_\theta(\mathbf{x}), \mathbf{v}_{c'}))}$$

- The loss function is the negative log-likelihood: $\mathcal{L}(\theta) = -\log P_\theta(y = c|\mathbf{x})$



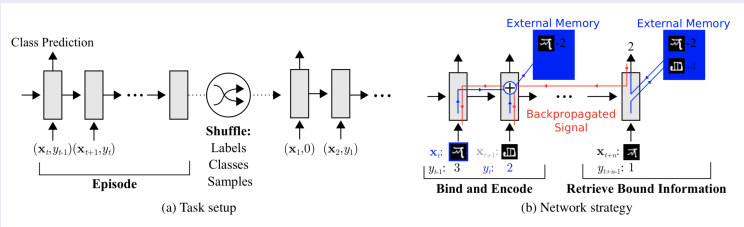
Model-based meta-learning for FSL

Principle

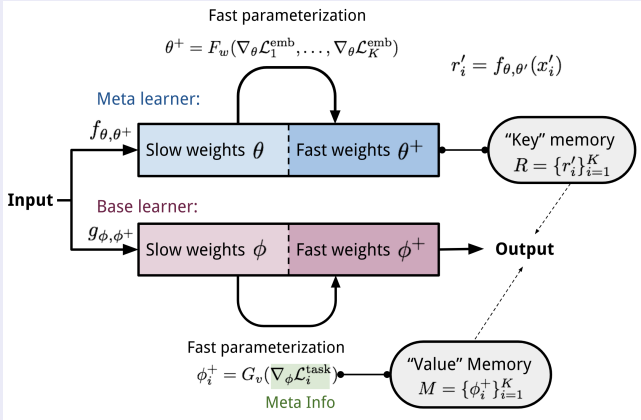
- Model designed specifically for fast learning — a model that updates its parameters rapidly with a few training steps.
- This rapid parameter update can be achieved by its internal architecture or controlled by another meta-learner model.
  - Memory-augmented Neural Networks (Santoro el al, 2016)[a]
  - Meta Networks (Munkhdalai el al, 2017)[b]

[a] http://proceedings.mlr.press/v48/santoro16.pdf
[b] hhttps://arxiv.org/abs/1703.00837

Memory-augmented Neural Networks



Meta Networks



Optimization-based meta-learning for FSL

Model-Agnostic Meta-Learning (Finn el al, 2017)[17] : find a robust initial set of parameters that makes gradient descent work across episodes

Principle

- Our model : f_θ with parameters θ.
- Given a task τ_i and its associated dataset $(\mathcal{D}^{(i)}_{\text{train}}, \mathcal{D}^{(i)}_{\text{test}})$, we update the model parameters by one or more gradient descent steps :

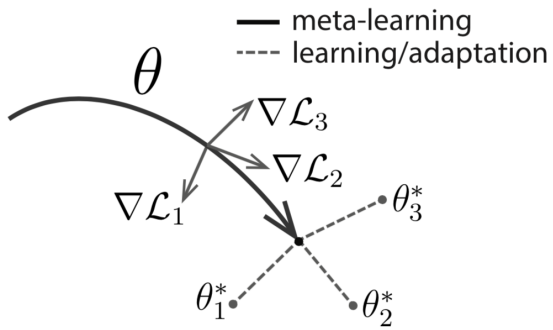
$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}^{(0)}_{\tau_i}(f_\theta)$$

with $\mathcal{L}^{(0)}$ the loss computed using the mini data batch with id (0)

- We sample a new data batch with id (1) for updating the meta-objective

$$\theta^* = \arg\min_\theta \sum_{\tau_i \sim p(\tau)} \mathcal{L}^{(1)}_{\tau_i}(f_{\theta_i'}) = \arg\min_\theta \sum_{\tau_i \sim p(\tau)} \mathcal{L}^{(1)}_{\tau_i}(f_{\theta - \alpha \nabla_\theta \mathcal{L}^{(0)}_{\tau_i}(f_\theta)})$$

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\tau_i \sim p(\tau)} \mathcal{L}^{(1)}_{\tau_i}(f_{\theta - \alpha \nabla_\theta \mathcal{L}^{(0)}_{\tau_i}(f_\theta)})$$



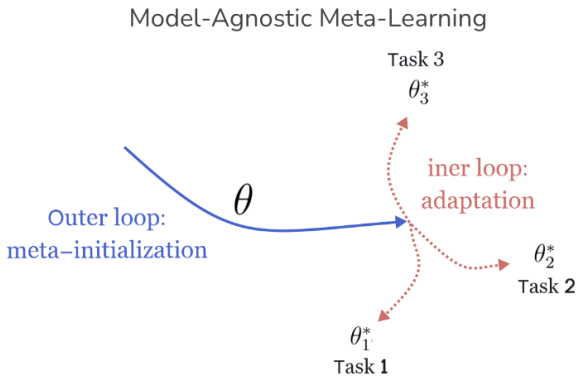
[17] https://arxiv.org/abs/1703.03400

Optimization-based meta-learning for FSL

Understanding the effectiveness of MAML (Raghu et al,2020)[18]

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
6: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
7: **end for** Note: the meta-update is using different set of data.
8: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
9: **end while**



Two loop structure

- outer loop : meta initialization for the networks parameters to a setting that enables fast adaptation to new tasks.
- inner loop : adapts these parameters for each task separately

[18] https://openreview.net/pdf?id=rkgMkCEtPB

Optimization-based meta-learning for FSL

Understanding the effectiveness of MAML (Raghu et al,2020)[19]

- Is this inner/outer loop necessary?

[19] https://openreview.net/pdf?id=rkgMkCEtPB

Optimization-based meta-learning for FSL

Understanding the effectiveness of MAML (Raghu et al,2020)[19]

- Is this inner/outer loop necessary?
- How much of the effectiveness of such methods is contingent on the inner/outer loop structure?

[19] https://openreview.net/pdf?id=rkgMkCEtPB

Optimization-based meta-learning for FSL

Understanding the effectiveness of MAML (Raghu et al,2020)[19]

- Is this inner/outer loop necessary?
- How much of the effectiveness of such methods is contingent on the inner/outer loop structure?
- Which of the two loop is more important ?

[19] https://openreview.net/pdf?id=rkgMkCEtPB

Optimization-based meta-learning for FSL

Understanding the effectiveness of MAML (Raghu et al,2020)[19]

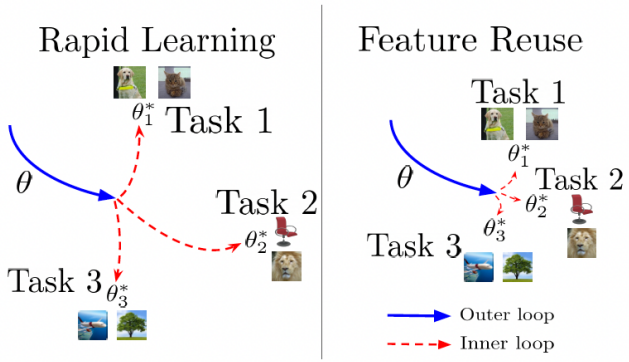
- Is this inner/outer loop necessary?
- How much of the effectiveness of such methods is contingent on the inner/outer loop structure?
- Which of the two loop is more important ?
  - Rapid learning : the representations change dramatically for each task and the inner loop plays an important role.

[19] https://openreview.net/pdf?id=rkgMkCEtPB

Optimization-based meta-learning for FSL

Understanding the effectiveness of MAML (Raghu et al,2020)[19]

- Is this inner/outer loop necessary?
- How much of the effectiveness of such methods is contingent on the inner/outer loop structure?
- Which of the two loop is more important ?
  - Rapid learning : the representations change dramatically for each task and the inner loop plays an important role.
  - Feature reuse : the outer loop gives rise to general purpose representations that require little adaption for each task.

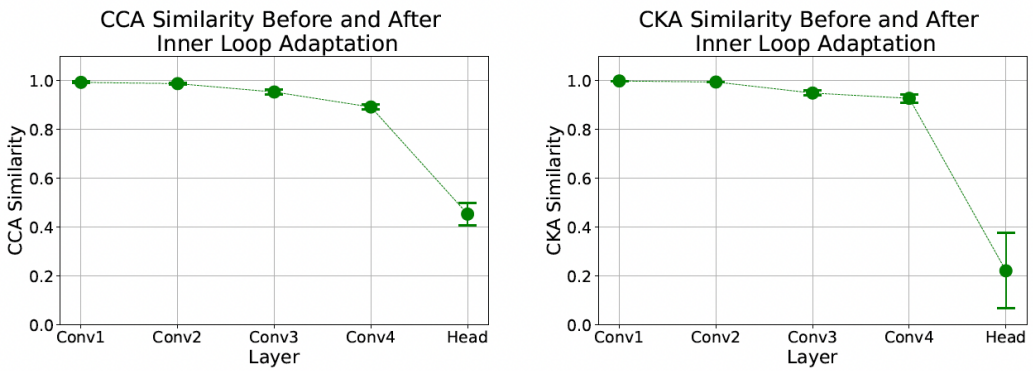


[19] https://openreview.net/pdf?id=rkgMkCEtPB

Optimization-based meta-learning for FSL

Understanding the effectiveness of MAML (Raghu et al,2020)[20]

Similarity of the representation before and after adaptation



- Representation remain largely the same expect for the final classification layer.
- Importance of having a good initialization of the representations for fast adaptation : **general-purpose representation**.

[20] https://openreview.net/pdf?id=rkgMkCEtPB

Meta-learning for FSL : a small reacp

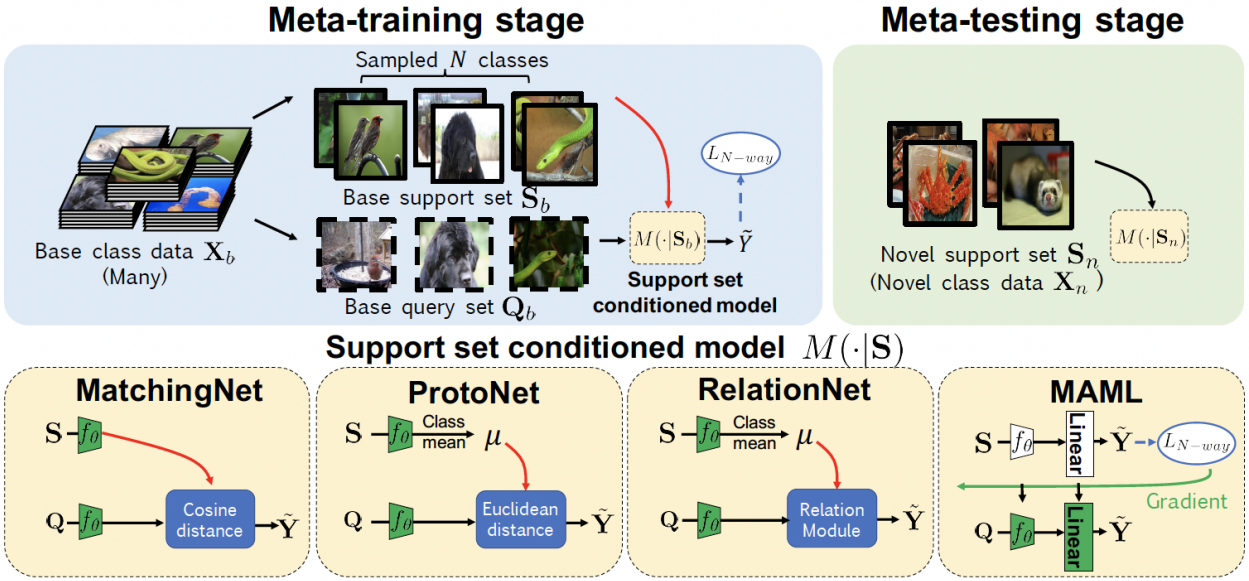


Figure 2: **Meta-learning few-shot classification algorithms.** The meta-learning classifier $M(\cdot|\mathbf{S})$ is conditioned on the support set $\mathbf{S}$. (*Top*) In the meta-train stage, the support set $\mathbf{S}_b$ and the query set $\mathbf{Q}_b$ are first sampled from random N classes, and then train the parameters in $M(.|\mathbf{S}_b)$ to minimize the N-way prediction loss $L_{\text{N-way}}$. In the meta-testing stage, the adapted classifier $M(.|\mathbf{S}_n)$ can predict novel classes with the support set in the novel classes $\mathbf{S}_n$. (*Bottom*) The design of $M(\cdot|\mathbf{S})$ in different meta-learning algorithms.

Source : (Chen et al, 2019) A closer look at few-shot classification[21]

[21] https://arxiv.org/pdf/1904.04232.pdf

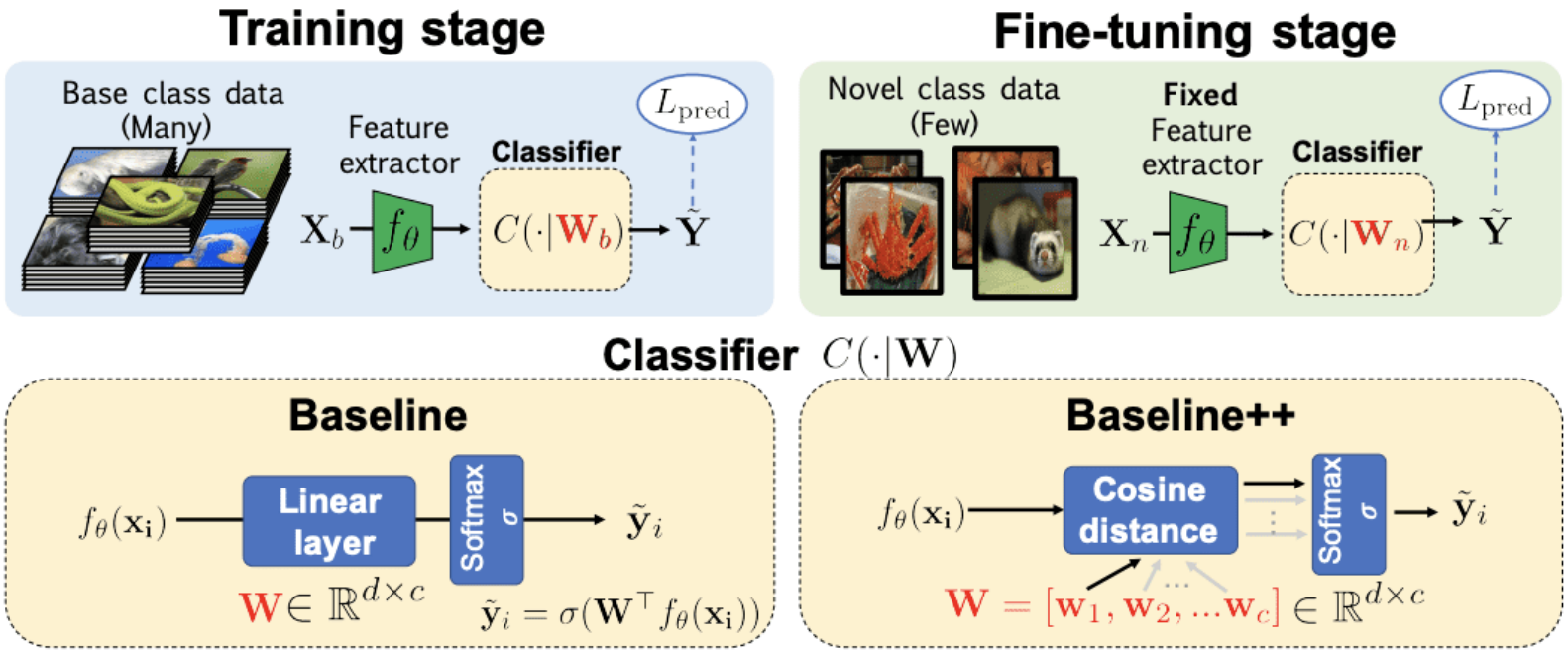
Transfer learning approaches

(Chen et al, 2019) A closer look at few-shot classification[22]

Principle

Follows the standard transfer learning procedure of network pre-training and fine-tuning

- **Training stage** : train a feature extractor f_θ and the classifier $C(.|W_b)$ from scratch by minimizing a standard CE classification loss L_{pred} using the training examples in the base classes.
- **Fine-tuning stage** : adapt the model to recognize novel classes in the fine-tuning stage : the pre-trained network parameters θ of f_θ are fixed and we train a new classifier $C(.|W_n)$ by minimizing L_{pred} using the support set with the novel classes X_n.



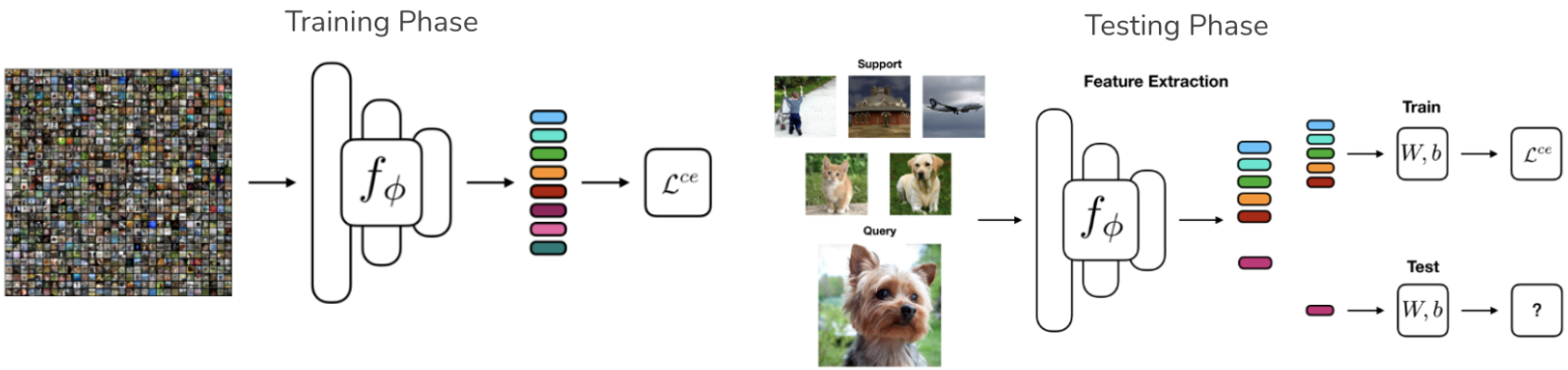
[22] https://arxiv.org/pdf/1904.04232.pdf

Transfer learning approaches

(Tian et al, 2020) Rethinking Few-Shot Image Classification: a Good Embedding Is All You Need? [23]

Principle

- **Training stage** : train a feature extractor f_θ on the whole meta training set without episodic training
- **Test-time** : For each sampled task, we first extract the features of the support and query examples using the pre-trained and frozen feature extractor, then we train a linear classifier on the L_2 normalized features of the support set and apply on the features of the test set.



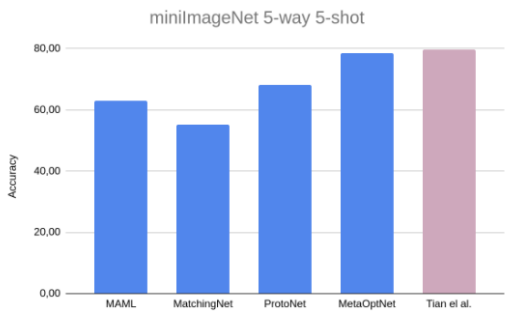
[23] https://arxiv.org/pdf/2003.11539.pdf

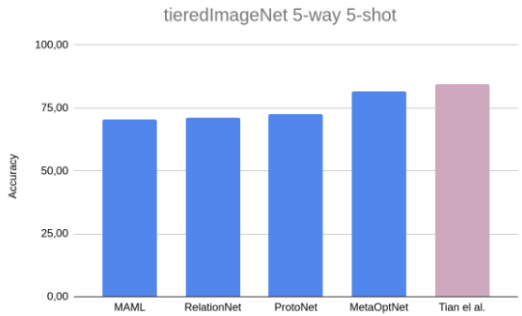
Transfer learning approaches

(Tian et al, 2020) Rethinking Few-Shot Image Classification: a Good Embedding Is All You Need? [24]

Principle

- **Training stage** : train a feature extractor f_θ on the whole meta training set without episodic training
- **Test-time** : For each sampled task, we first extract the features of the support and query examples using the pre-trained and frozen feature extractor, then we train a linear classifier on the L_2 normalized features of the support set and apply on the features of the test set.





[24] https://arxiv.org/pdf/2003.11539.pdf

Transductive Few Shot Learning

Principle

- The model has access to all the unlabeled data that we want to classify, and it only needs to produce labels for those samples (as opposed to every possible input sample).
- Transduction is used as a form of semi-supervised learning, where we have some unlabeled samples from which the model can obtain extra information about the data distribution to make better predictions.
- In few-shot learning, transductive algorithms make use of all the queries in an episode instead of treating them individually.

Transductive Few Shot Learning

(Liu et al, 2019) Learning to propagate labels : transductive propagation network for few-shot learning [25]

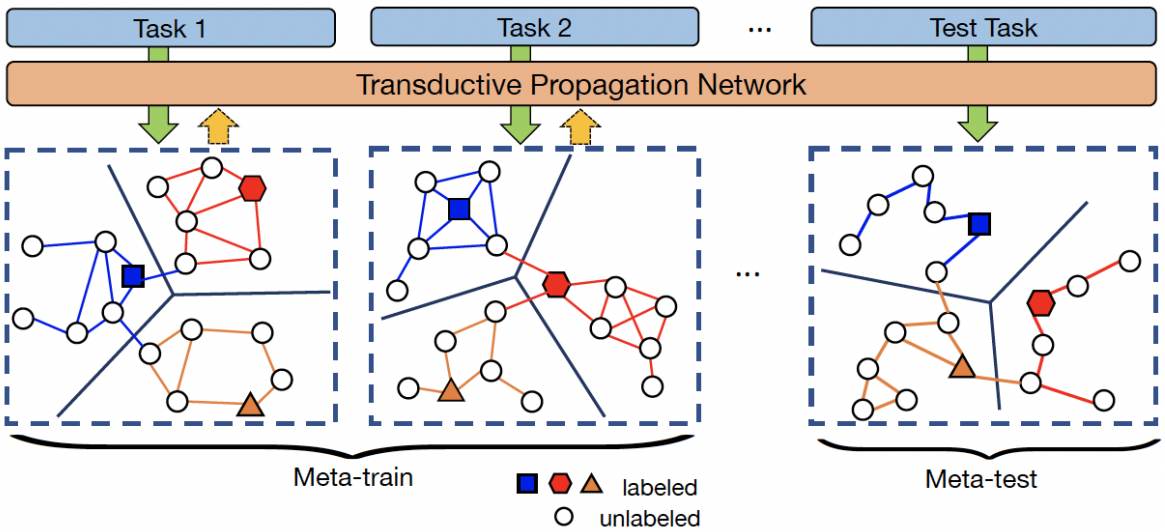


Figure 1: A conceptual illustration of our transductive meta-learning framework, where lines between nodes represent graph connections and their colors represent the potential direction of label propagation. The neighborhood graph is episodic-wisely trained for transductive inference.

[25] https://arxiv.org/pdf/1805.10002.pdf

Transductive Few Shot Learning

(Liu et al, 2019) Learning to propagate labels : transductive propagation network for few-shot learning [26]

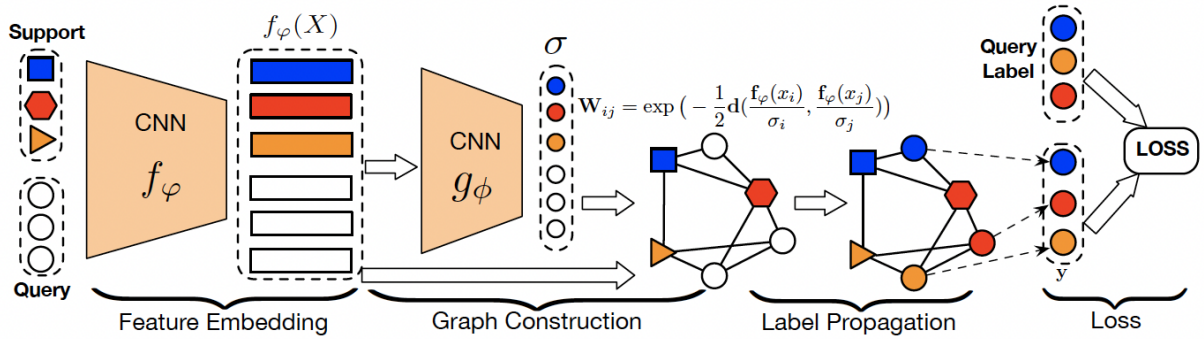


Figure 2: The overall framework of our algorithm in which the manifold structure of the entire query set helps to learn better decision boundary. The proposed algorithm is composed of four components: feature embedding, graph construction, label propagation, and loss generation.

- Graph construction : Gaussian similarity function with σ to select carefully for the best label propagation

$$W_{ij} = \exp\left(-\frac{d(x_i, x_j)}{2\sigma^2}\right)$$

- Convolutional network to produce an example-wise length-scale parameter

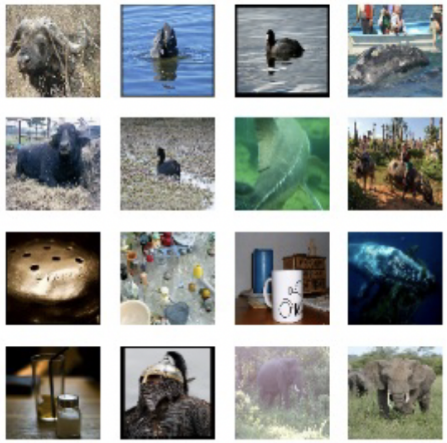
$$W_{ij} = \exp\left(-\frac{1}{2}d\left(\frac{f_\varphi(x_i)}{\sigma_i}, \frac{f_\varphi(x_j)}{\sigma_j}\right)\right)$$

- Label propagation with the normalized graph Laplacians $(I - \alpha S)^{-1} Y$

[26] https://arxiv.org/pdf/1805.10002.pdf

Benchmarks

Mini-Imagenet (Vinyals et al., 2016)



- 100 classes from Imagenet
- 64 training classes
- 16 validation classes
- 20 test classes
- 600 examples per class

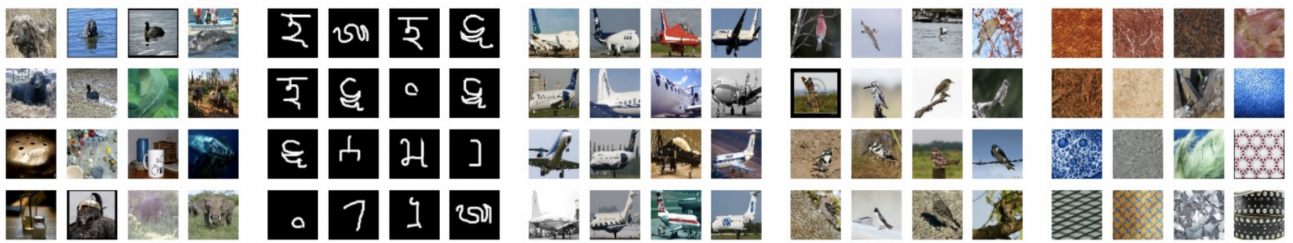
Omniglot (Lake et al., 2015)

- 1623 handwritten characters
- 50 alphabets
- 20 examples per character

Benchmarks

Large scale experiments

(Triantafillou et al, 2019)[a] Meta-dataset: a dataset of datasets for learning to learn from few examples



(a) ImageNet (b) Omniglot (c) Aircraft (d) Birds (e) DTD

(f) Quick Draw (g) Fungi (h) VGG Flower (i) Traffic Signs (j) MSCOCO

[a] https://arxiv.org/abs/1903.03096

Benchmarks

Large scale experiments

(Triantafillou et al, 2019) Meta-dataset: a dataset of datasets for learning to learn from few examples[a]

- Many diverse image datasets
- Hierarchically aware
- Heterogeneous episodes : different number of classes, different numbers of support examples per class
- Test generalization across domains : traffic signs and MSCOCO never seen
- Optimization over a wide range of hyperparameters to allow to initialize from pre-trained features.
- Pre-training : on ImageNet only or train on all datasets (except traffic signs and MSCOCO)

[a] `https://github.com/google-research/meta-dataset`

Some open issues in Few shot learning

Open set few shot recognition

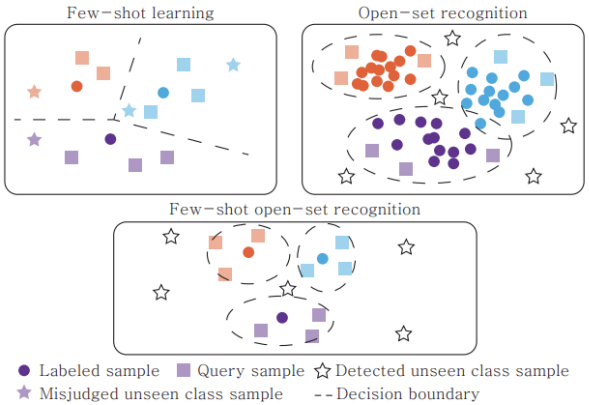


Figure 1. A visualization of the few-shot open-set recognition task. Few-shot learning methods fail to recognize unseen class samples, and open-set recognition methods require a large amount of datasets. Few-shot open-set recognition is a generalized few-shot learning task, where the model has to identify unseen class queries while classifying seen class queries correctly.

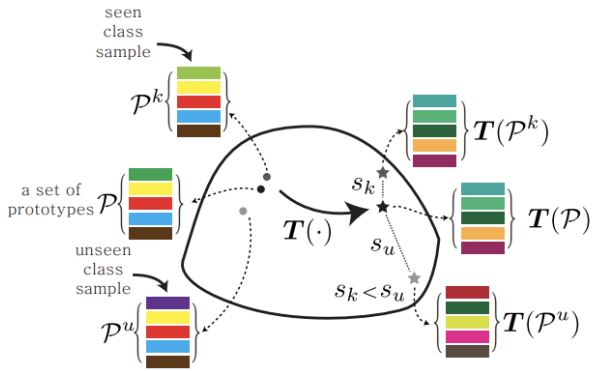


Figure 3. A visualization of our method. Each colored box represents a prototype or a query feature. SnaTCHer replaces a predicted class prototype to the query sample in the set of prototypes ($\mathcal{P}^k$ and $\mathcal{P}^u$), then it measures differences between the prototype set and the replaced set after the feature transformation $\boldsymbol{T}(\cdot)$. SnaTCHer rejects samples by the distance from the transformed prototype set $\boldsymbol{T}(\mathcal{P})$. Our method alters estimating feature distribution of unseen class samples for the detection to the relative feature transformation problem.

(Jeong et al, 2021) Few-shot Open-set Recognition by Transformation Consistency [27] (Liu et al, 2020) Few-Shot Open-Set Recognition using Meta-Learning [28]

[27] https://arxiv.org/pdf/2103.01537.pdf

[28] https://openaccess.thecvf.com/content_CVPR_2020/papers/Liu_Few-Shot_Open-Set_Recognition_Using_Meta-Learning_CVPR_2020_paper.pdf

Some open issues in Few shot learning

Distribution shift between the query and the support set

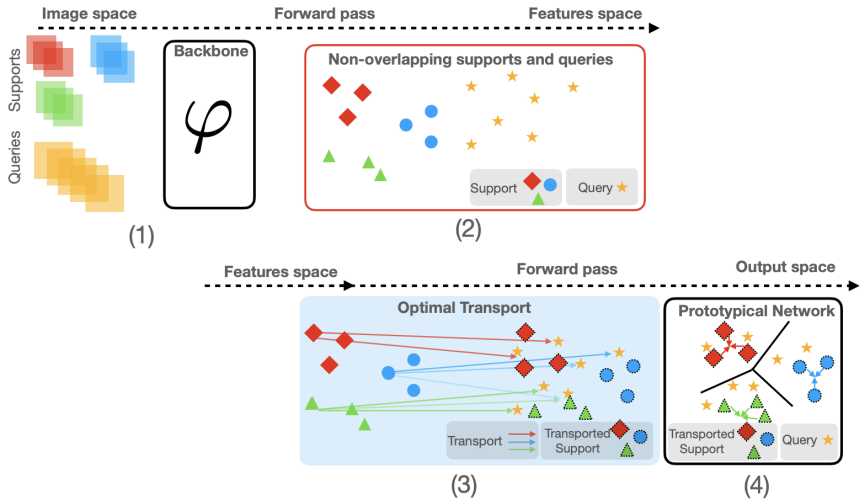


Fig. 3. Overview of *Transported Prototypes*. (1) A support set and a query set are fed to a trained backbone that embeds images into a feature space. (2) Due to the shift between distributions, support and query instances are embedded in non-overlapping areas. (3) We compute the Optimal Transport from support instances to query instances to build the transported support set. Note that we represent the transport plan only for one instance per class to preserve clarity in the schema. (4) Provided with the transported support, we apply the Prototypical Network [28] *i.e.*, L^2 similarity between transported support and query instances.

(Bennequin et al, 2021)[29] Bridging Few-Shot Learning and Adaptation: New Challenges of Support-Query Shift

[29] https://arxiv.org/pdf/2105.11804.pdf

Conclusion

- A short overview of the main approaches for few shot learning
- To practice : EasyFSL : `https://github.com/sicara/easy-few-shot-learning`