Aalto University

CS-E4890: Deep Learning
Learning with few labeled examples

Alexander Ilin

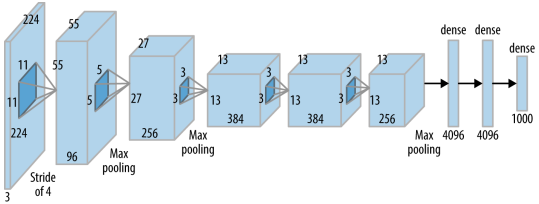
Motivation

- Deep learning is data hungry. To learn to classify handwritten digits, we need thousands of samples, to learn to classify natural images we need millions of images.
- Suppose we have a custom classification task, for example, we need to classify images to custom classes (not covered by imageNet). What can we do?
  - Collect a lot of training examples and label them.
    - Time consuming and expensive.
    - Sometimes collecting new examples is impossible.
  - Transfer learning
  - Semi-supervised learning
  - Self-supervised learning
  - Few-shot learning (meta learning)

Transfer learning

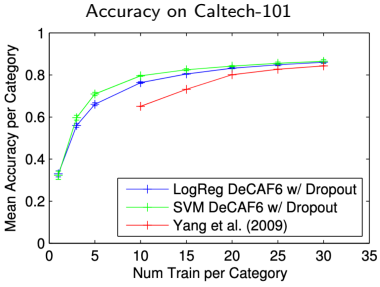
Transfer learning: Image classification

- Features that are useful for some tasks can be useful for other tasks in the same domain.
- For image classification tasks, it is common to fine-tune the last layer of a deep neural network pre-trained on imageNet (there is a bunch of them in PyTorch).
- For example, we can take the AlexNet (Krizhevsky, 2012) pre-trained on ImageNet and fine-tune the last two layers.



Example results: Transfer learning ImageNet → Caltech-101

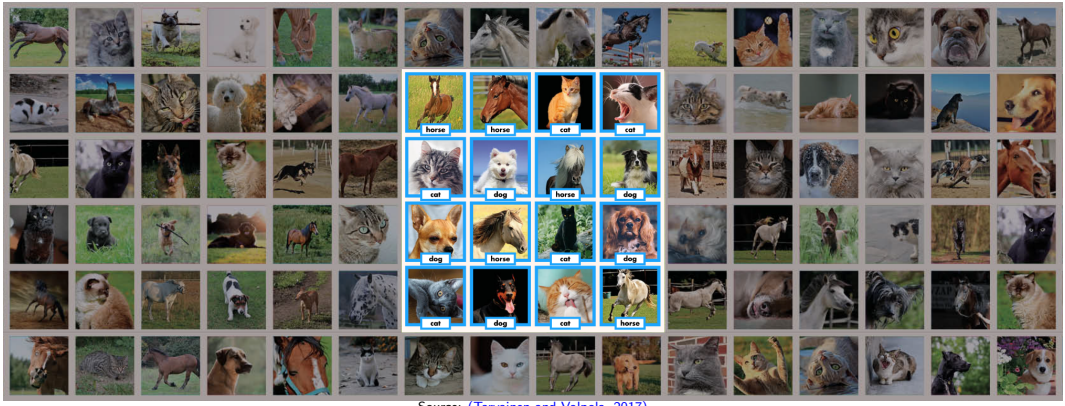
- Example results: Classification of images from the Caltech-101 dataset (Donahue et al., 2013):
  - Yang et al. (2009): a method employing a combination of five traditional hand-engineered image features followed by a multi-kernel based classifier.
  - DeCAF6: Features from the sixth layer of AlexNet pre-trained on ImageNet dataset.



Semi-supervised learning

Semi-supervised classification

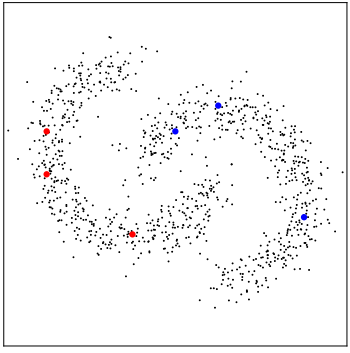
- Semi-supervised classification: few labeled examples, many unlabeled examples.



Source: (Tarvainen and Valpola, 2017)

When semi-supervised learning is possible?

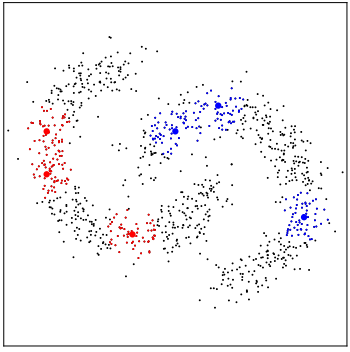
- Semi-supervised learning: few labeled examples, many unlabeled examples.
- Semi-supervised learning is possible when the knowledge on $p(\mathbf{x})$ that one gains through the unlabeled data carries information that is useful in the inference of $p(\mathbf{y} \mid \mathbf{x})$.
- In the hypothetical example on the right, modeling the distribution of unlabeled data can improve the classification accuracy.



The labels can be propagated to the unlabeled data in the same cluster yielding better classification accuracy.

When semi-supervised learning is possible?

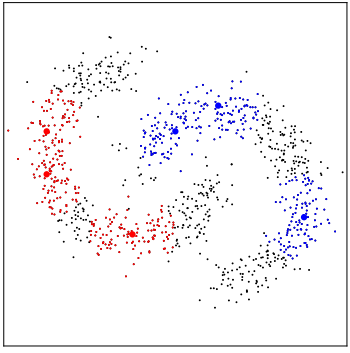
- Semi-supervised learning: few labeled examples, many unlabeled examples.
- Semi-supervised learning is possible when the knowledge on $p(\mathbf{x})$ that one gains through the unlabeled data carries information that is useful in the inference of $p(\mathbf{y} \mid \mathbf{x})$.
- In the hypothetical example on the right, modeling the distribution of unlabeled data can improve the classification accuracy.



The labels can be propagated to the unlabeled data in the same cluster yielding better classification accuracy.

When semi-supervised learning is possible?

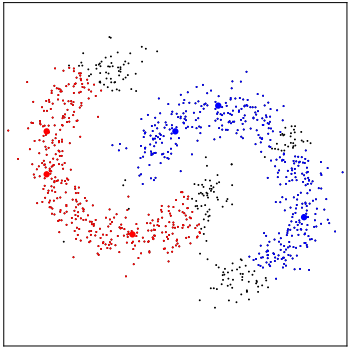
- Semi-supervised learning: few labeled examples, many unlabeled examples.
- Semi-supervised learning is possible when the knowledge on $p(\mathbf{x})$ that one gains through the unlabeled data carries information that is useful in the inference of $p(\mathbf{y} \mid \mathbf{x})$.
- In the hypothetical example on the right, modeling the distribution of unlabeled data can improve the classification accuracy.



The labels can be propagated to the unlabeled data in the same cluster yielding better classification accuracy.

When semi-supervised learning is possible?

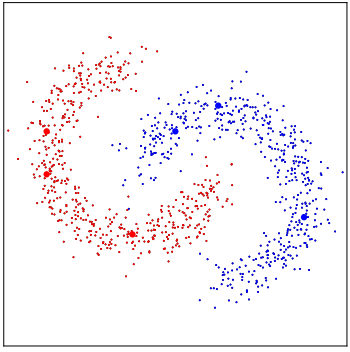
- Semi-supervised learning: few labeled examples, many unlabeled examples.
- Semi-supervised learning is possible when the knowledge on $p(\mathbf{x})$ that one gains through the unlabeled data carries information that is useful in the inference of $p(\mathbf{y} \mid \mathbf{x})$.
- In the hypothetical example on the right, modeling the distribution of unlabeled data can improve the classification accuracy.



The labels can be propagated to the unlabeled data in the same cluster yielding better classification accuracy.

When semi-supervised learning is possible?

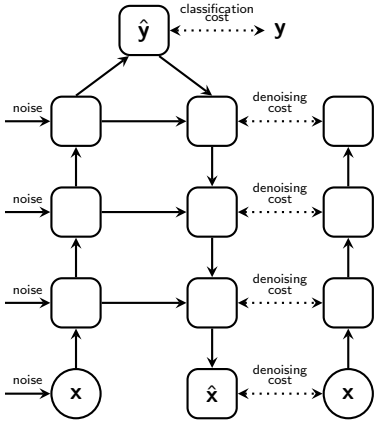
- Semi-supervised learning: few labeled examples, many unlabeled examples.
- Semi-supervised learning is possible when the knowledge on $p(\mathbf{x})$ that one gains through the unlabeled data carries information that is useful in the inference of $p(\mathbf{y} \mid \mathbf{x})$.
- In the hypothetical example on the right, modeling the distribution of unlabeled data can improve the classification accuracy.



The labels can be propagated to the unlabeled data in the same cluster yielding better classification accuracy.

Ladder networks (Rasmus et al., 2015)

- The architecture resembles a ladder (or a U-net).
- The bottom-up pass produces label **y** for a given input **x**.
- For labeled examples, we can compute the standard classification (e.g., cross-entropy) loss using the network output $\hat{\mathbf{y}}$ and the correct label **y**.

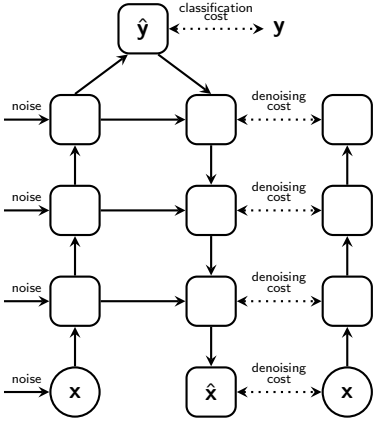


Ladder networks: Denoising

- The inputs **x** are corrupted by noise during training (e.g, we never use clean images as inputs).
- The top-down pass tries to reconstruct the clean (without noise) input **x**.
- For all examples (both labeled and unlabeled), we compute the denoising cost at the bottom:

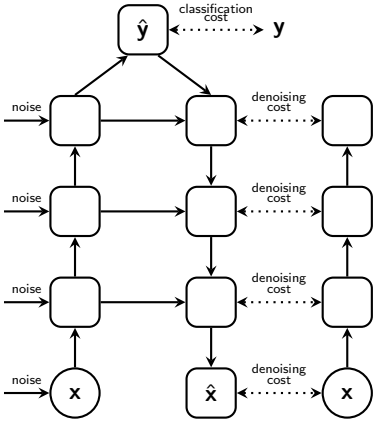
$$\text{denoising cost} = \|\mathbf{x} - \hat{\mathbf{x}}\|^2$$

- The minimized cost is the sum of the classification and denoising costs.



Ladder networks: Intuitions

- Intuition: In order to reconstruct the clean image from a noisy image, one has to understand what features are commonly present in images, that is we need to model the data distribution $p(\mathbf{x})$ inside the network.
- Therefore, denoising is an auxiliary task that helps model $p(\mathbf{x})$ and hopefully develop features useful for the primary classification task.
- The label itself cannot contain enough information to reconstruct the input. We need skip connections to pass low-level details from the bottom-up pass.
- Note: corruption and denoising happens on multiple levels.



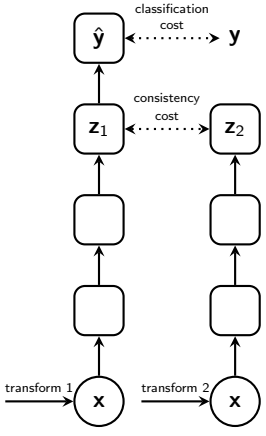
Π-model (Laine and Aila, 2016)

- The problem with the Ladder network: We need to model the whole distribution $p(\mathbf{x})$ which can contain lots of details irrelevant for the classification task.

- Laine and Aila (2016) proposed a simplification of the Ladder networks that does not contain the top-down pass.
- There are two copies of the same network performing computations for $\mathbf{x}_1$ and $\mathbf{x}_2$, which is the same training example changed with different transformations.
- The cost for unlabeled data is

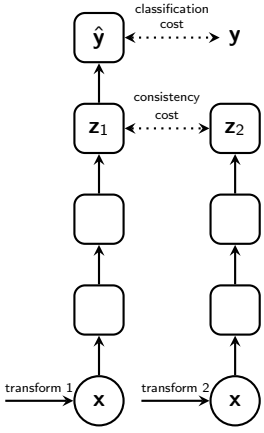
$$\text{consistency cost} = \|\mathbf{z}_1 - \mathbf{z}_2\|^2 = \|f(\mathbf{x}_1) - f(\mathbf{x}_2)\|^2$$

- In the Π-model, the gradients propagate through both networks, $\mathbf{z}$ is the input of the softmax (logit).



Consistency-based semi-supervised learning methods

- The intuition: we do not know the correct output for unlabeled data but we know that the output should not change for a transformed input.
- Since the introduction of the Π-model, the majority of the semi-supervised methods have been based on optimizing consistency between different transformations of the same training examples.
- The idea resembles siamese networks (Bromley et al., 1993).



Mean Teacher (Tarvainen and Valpola, 2017)

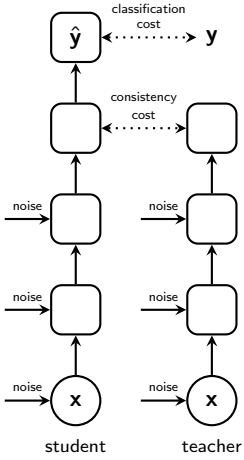
- Instead of using two copies of the same network, one of the networks (teacher) is obtained by computing exponential moving average of the weights of the other (student) network:

$$\boldsymbol{\theta}'_t = \gamma \boldsymbol{\theta}'_{t-1} + (1-\gamma)\boldsymbol{\theta}_t$$

- The same consistency cost is minimized on both labeled and unlabeled data:

$$\text{consistency cost} = \left\| f(\mathbf{x}_1, \boldsymbol{\theta}_t) - f(\mathbf{x}_2, \boldsymbol{\theta}'_t) \right\|^2$$

- The teacher is more accurate than the student. The gradients propagate only through the student.



Evolution of consistency-based semi-supervised methods

Algorithm	Student augment.	Teacher augment.	Teacher label post-processing	CIFAR-10 accuracy 4K labels (from here)
Fully supervised Wide ResNet with 50K labels	–	–	–	94.6
Π-Model (Laine and Aila, 2016)	Weak	Weak	–	87.84
VAT (Miyato et al., 2017)	Adversarial	–	–	88.64
Mean Teacher (Tarvainen and Valpola, 2017)	Weak	Weak	–	93.72
UDA (Xie et al., 2019)	Strong	Weak	Sharpening	94.73
MixMatch (Berthelot et al., 2019)	Weak	Weak	Sharpening	93.76
ReMixMatch (Berthelot et al., 2019)	Strong	Weak	Sharpening	94.86
FixMatch (Sohn et al., 2020)	Strong	Weak	Pseudo-labeling	95.69

- Weak augmentations:
  - Translation, flip, Gaussian noise.
  - MixMatch uses MixUp (Zhang et al., 2018).
- Adversarial: Use adversarial examples for data transformation.
- Strong augmentations: uniformly sample all image processing transformations from the Python Image Library (PIL).

Evolution of consistency-based semi-supervised methods

Algorithm	Student augment.	Teacher augment.	Teacher label post-processing	CIFAR-10 accuracy 4K labels (from here)
Fully supervised Wide ResNet with 50K labels	–	–	–	94.6
Π-Model (Laine and Aila, 2016)	Weak	Weak	–	87.84
VAT (Miyato et al., 2017)	Adversarial	–	–	88.64
Mean Teacher (Tarvainen and Valpola, 2017)	Weak	Weak	–	93.72
UDA (Xie et al., 2019)	Strong	Weak	Sharpening	94.73
MixMatch (Berthelot et al., 2019)	Weak	Weak	Sharpening	93.76
ReMixMatch (Berthelot et al., 2019)	Strong	Weak	Sharpening	94.86
FixMatch (Sohn et al., 2020)	Strong	Weak	Pseudo-labeling	95.69

- Label sharpening: Use (small) temperature parameter τ to compute the targets:

$$p_i^{\text{teacher}} = \frac{\exp(z_i/\tau)}{\sum_j \exp(z_j/\tau)}$$

- Pseudo-labeling: use "hard" labels (i.e., the arg max of the model's output) and only retain the teacher's labels whose largest class probability fall above a predefined threshold.

Evolution of consistency-based semi-supervised methods

Algorithm	Student augment.	Teacher augment.	Teacher label post-processing	CIFAR-10 accuracy 4K labels (from here)
Fully supervised Wide ResNet with 50K labels	–	–	–	94.6
Π-Model (Laine and Aila, 2016)	Weak	Weak	–	87.84
VAT (Miyato et al., 2017)	Adversarial	–	–	88.64
Mean Teacher (Tarvainen and Valpola, 2017)	Weak	Weak	–	93.72
UDA (Xie et al., 2019)	Strong	Weak	Sharpening	94.73
MixMatch (Berthelot et al., 2019)	Weak	Weak	Sharpening	93.76
ReMixMatch (Berthelot et al., 2019)	Strong	Weak	Sharpening	94.86
FixMatch (Sohn et al., 2020)	Strong	Weak	Pseudo-labeling	95.69

- Mean Teacher: Use exponential moving average of parameters to get the teacher model

$$\boldsymbol{\theta}'_t = \gamma \boldsymbol{\theta}'_{t-1} + (1-\gamma)\boldsymbol{\theta}_t$$

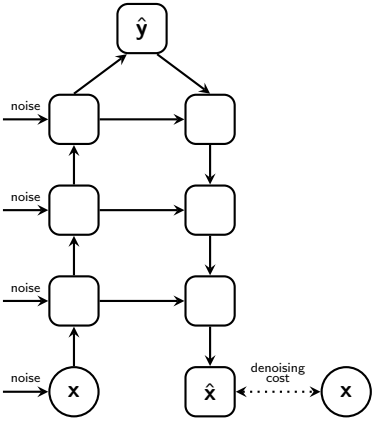
- MixMatch: The target is computed as the average prediction obtained for K augmentations.
- ReMixMatch: Distribution alignment encourages the marginal distribution of predictions on unlabeled data to be close to the marginal distribution of ground-truth labels.

Self-supervised learning

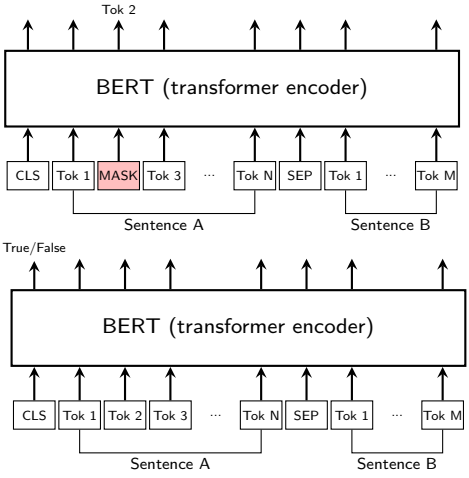
Self-supervised learning

- The assumption that is made in semi-supervised learning: The unlabeled examples belong to the same classes. This can be difficult to assure in many practical applications.
- Can we lean useful representations in a completely unsupervised manner?
- Self-supervised learning: Invent an auxiliary task that can be learned in an unsupervised manner and use the developed features for the downstream task.
- In order to succeed, the auxiliary task should be relevant for the downstream task.

Examples of self-supervised learning models



Ladder networks

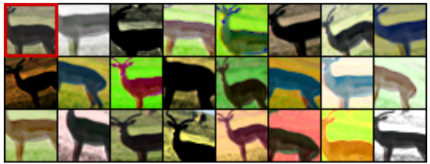


Discriminative unsupervised feature learning (Dosovitskiy et al., 2014)

- One of the early works on self-supervised training.
- A convolutional neural network is pre-trained using an artificially created learning task.
  - N patches of size 32×32 are sampled from different images at varying positions and scale.
  - Each patch is transformed multiple times using (a composition of elementary) transformations.
  - The task is to classify a transformed image patch to one of the N classes that correspond to the original N patches.
- The features produced by the CNN are used as inputs of a support vector machine classifier.

sampled patches

transformed patches



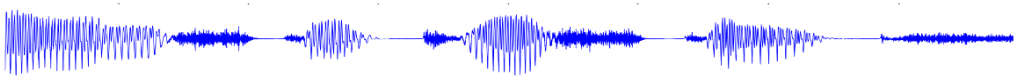
elementary transformations used: translation, scaling, rotation, contrast (raise S and V components of the HSV color representation), color (change the H component of the HSV representation)

Contrastive Predictive Coding

(van den Oord et al., 2018)

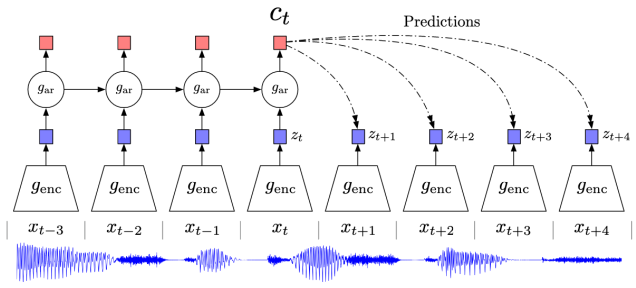
Motivation

- The goal is to learn representations that encode the underlying shared information between different parts of the (high-dimensional) signal. At the same time we want to discard low-level information and noise that is more local.
- When predicting further in the future, the amount of shared information becomes much lower, and the model needs to infer more global structure. These 'slow features' that span many time steps are often more interesting (e.g., phonemes and intonation in speech, objects in images, or the story line in books.)



Contrastive Predictive Coding (CPC): The model

- A non-linear encoder g_{enc} maps the input sequence of observations x_t to a sequence of latent representations $z_t = g_{\text{enc}}(x_t)$, potentially with a lower temporal resolution.
- Next, an autoregressive model g_{ar} summarizes all $z \leq t$ in the latent space and produces a context latent representation $c_t = g_{\text{ar}}(z \leq t)$.

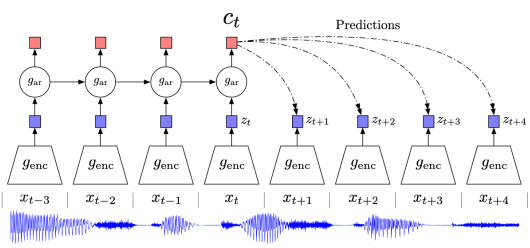


CPC: Contrastive loss

- The task: Given the context c_t, select the correct future code $z_{t+k} = g_{\text{enc}}(x_{t+k})$ after k steps among N alternatives $\{z_{t+k}, z_{\tau_1}, ..., z_{\tau_{N-1}}\}$.
- The alternatives $z_\tau = g_{\text{enc}}(x_\tau)$ are selected as encoder outputs produced for inputs x_τ randomly selected from the dataset (for example, from the same input sequence).
- The loss is the categorical cross-entropy of selecting the correct encoding:

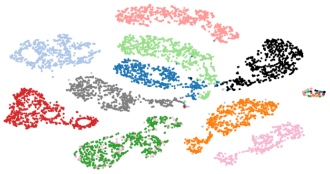
$$\mathcal{L} = -\log \frac{\exp(z_{t+k}^\top W_k c_t)}{\sum_j \exp(z_{\tau_j}^\top W_k c_t)}$$

- The logits produced by the classifier are postulated to have the form $z_\tau^\top W_k c_t$.



CPC: Results for audio data

- The quality of developed representations are tested by training a classifier using representations c_t as features on the phone classification task:
  - linear classifier: 64.6
  - MLP classifier: 72.5
- CPC captures both speaker identity and speech contents.



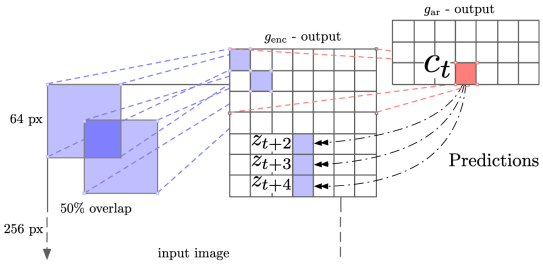
t-SNE visualization of audio (speech) representations for a subset of 10 speakers (out of 251). Every color represents a different speaker.

Method	ACC
Phone classification	
Random initialization	27.6
MFCC features	39.7
CPC	64.6
Supervised	74.6
Speaker classification	
Random initialization	1.87
MFCC features	17.6
CPC	97.4
Supervised	98.5

Classification accuracy on audio data. Phone classification: 41 classes Speaker classification: 251 classes. random initialization: random g_{enc} and g_{ar}

CPC for image data

- Contrastive Predictive Coding for images:



- The quality of developed representations are tested by training a linear classifier using RNN outputs c_t as input features.

Method	Top-5 ACC
Motion Segmentation (MS)	48.3
Exemplar (Ex)	53.1
Relative Position (RP)	59.2
Colorization (Col)	62.5
Combination of MS + Ex + RP + Col	69.3
CPC	**73.6**

ImageNet top-5 unsupervised classification results.

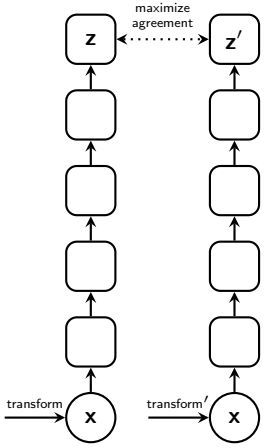
A Simple Framework for Contrastive Learning (SimCLR)

(Chen et al., 2020a, 2020b)

SimCLR (Chen et al., 2020a)

- SimCLR can be seen as the siamese networks processing two different augmentations of the same image (as in the Π-model, Mean Teacher) combined with the CPC contrastive loss.
  - Sample a minibatch of N examples.
  - Augment each example with two different transformations, which results in $2N$ data points.
  - Process each example with a deep neural network $\mathbf{z} = g(f(\mathbf{x}))$.
  - The training task is similar to CPC: For each image in the minibatch, we need to select the other image that was produced with the other transformation of the same original image among $2N - 1$ alternatives

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



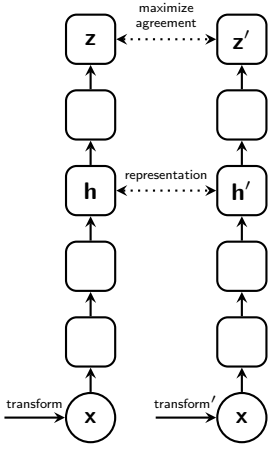
SimCLR (Chen et al., 2020a)

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

- The cosine similarity is used:

$$\text{sim}(\mathbf{u}, \mathbf{v}) = \mathbf{u}^\top \mathbf{v} / \|\mathbf{u}\| \, \|\mathbf{v}\|$$

- As representation **h**, we take the output of an intermediate layer (two layers before the output).
  - The intuition: The formulated task of contrastive learning needs development of the right features but the downstream task is likely to be different and to require a different post-processing head.



SimCLR augmentations

- Three augmentations are sequentially applied:

original

1) random cropping followed by resize to the original size

2) random color distortions

3) random Gaussian blur

- Random cropping and color distortions give the largest boost in performance.
- It is important to apply both transformations for good performance. Why?

SimCLR augmentations

- Three augmentations are sequentially applied:

original

1) random cropping followed by resize to the original size

2) random color distortions

3) random Gaussian blur

- Random cropping and color distortions give the largest boost in performance.
- It is important to apply both transformations for good performance. Why?
  Because with one transformation only it is relatively easy to find matching pairs of images.

Linear classifier trained on top of SimCLR representations

Method	Architecture	Param (M)	Top 1	Top 5
Methods using ResNet-50:				
Local Agg.	ResNet-50	24	60.2	-
MoCo	ResNet-50	24	60.6	-
PIRL	ResNet-50	24	63.6	-
CPC v2	ResNet-50	24	63.8	85.3
SimCLR (ours)	ResNet-50	24	**69.3**	**89.0**
Methods using other architectures:				
Rotation	RevNet-50 (4×)	86	55.4	-
BigBiGAN	RevNet-50 (4×)	86	61.3	81.9
AMDIM	Custom-ResNet	626	68.1	-
CMC	ResNet-50 (2×)	188	68.4	88.2
MoCo	ResNet-50 (4×)	375	68.6	-
CPC v2	ResNet-161 (∗)	305	71.5	90.1
SimCLR (ours)	ResNet-50 (2×)	94	74.2	92.0
SimCLR (ours)	ResNet-50 (4×)	375	**76.5**	**93.2**

ImageNet accuracies of linear classifiers trained
on representations learned with different self-supervised methods

Semi-supervised training with SimCLR

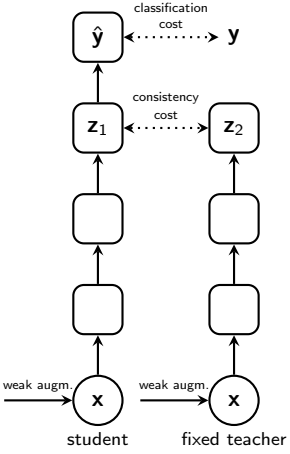
- The whole base network is fine-tuned on the few labeled data without regularization.

Method	Architecture	Label fraction (Top 5) 1%	10%
Supervised baseline	ResNet-50	48.4	80.4
Methods using other label-propagation:			
Pseudo-label	ResNet-50	51.6	82.4
VAT+Entropy Min.	ResNet-50	47.0	83.4
UDA (w. RandAug)	ResNet-50	-	88.5
FixMatch (w. RandAug)	ResNet-50	-	89.1
S4L (Rot+VAT+En. M.)	ResNet-50 (4×)	-	91.2
Methods using representation learning only:			
InstDisc	ResNet-50	39.2	77.4
BigBiGAN	RevNet-50 (4×)	55.2	78.8
PIRL	ResNet-50	57.2	83.8
CPC v2	ResNet-161(∗)	77.9	91.2
SimCLR (ours)	ResNet-50	75.5	87.8
SimCLR (ours)	ResNet-50 (2×)	83.0	91.2
SimCLR (ours)	ResNet-50 (4×)	**85.8**	**92.6**

ImageNet accuracy of models trained with few labels

SimCLRv2 (Chen et al., 2020b)

- SimCLRv2 improves the performance in the semi-supervised scenario by the following procedure:
  1. Pre-train SimCLR in an unsupervised way.
  2. Fine-tine the model using the few labeled examples (this is same as in SimCLRv1).
  3. Perform "knowledge distillation" using both labeled and unlabeled data.
- This last step is essentially equivalent to consistency-based semi-supervised training.
  - The teacher is fixed to the network obtained after step 2.
  - Only weak augmentations are used.
  - The student can have a smaller architecture.



Bootstrap your own latent (BYOL)

(Grill et al., 2020)

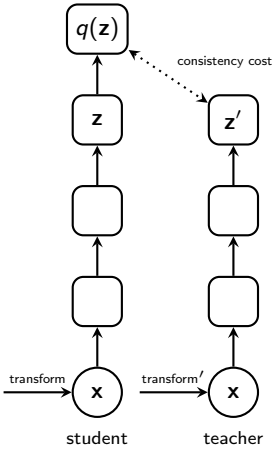
Bootstrap your own latent (BYOL) (Grill et al., 2020)

- BYOL can be seen as an extension of the Mean Teacher to the fully unsupervised scenario:
  - The teacher network is obtained by computing exponential moving average of the weights of the student.
  - The two networks process two different transformations of the same example.
  - The objective is

$$\text{consistency cost} = \left\| q(\mathbf{z}) - \mathbf{z}' \right\|^2$$

    where $\mathbf{z} = f(\mathbf{x}, \boldsymbol{\theta}_t)$, $\mathbf{z}' = f(\mathbf{x}', \boldsymbol{\theta}'_t)$

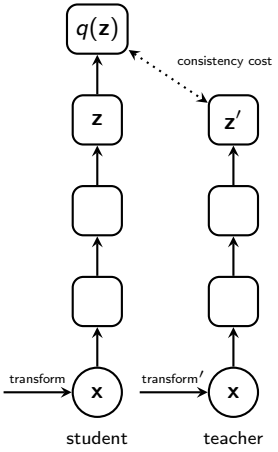
- Differences:
  - Use strong augmentations (same set of augmentations as in SimCLR).
  - Use an extra predictor MLP $q(\mathbf{z})$ in the student network.



Bootstrap your own latent (BYOL) (Grill et al., 2020)

$$\text{consistency cost} = \left\| q(\mathbf{z}) - \mathbf{z}' \right\|^2$$

- The representations can collapse: the network can learn to produce the same output $\mathbf{z}$ for any input $\mathbf{x}$, which would yield a zero consistency cost.
- In practice, this does not happen for the given architecture. In fact, the representations collapse if the predictor network $q(\mathbf{z})$ is removed.



BYOL: Linear evaluation

Method	Top-1	Top-5
Local Agg.	60.2	-
PIRL [35]	63.6	-
CPC v2 [32]	63.8	85.3
CMC [11]	66.2	87.0
SimCLR [8]	69.3	89.0
MoCo v2 [37]	71.1	-
InfoMin Aug. [12]	73.0	91.1
BYOL (ours)	**74.3**	**91.6**

(a) ResNet-50 encoder.

Method	Architecture	Param.	Top-1	Top-5
SimCLR [8]	ResNet-50 (2×)	94M	74.2	92.0
CMC [11]	ResNet-50 (2×)	94M	70.6	89.7
BYOL (ours)	ResNet-50 (2×)	94M	77.4	93.6
CPC v2 [32]	ResNet-161	305M	71.5	90.1
MoCo [9]	ResNet-50 (4×)	375M	68.6	-
SimCLR [8]	ResNet-50 (4×)	375M	76.5	93.2
BYOL (ours)	ResNet-50 (4×)	375M	78.6	94.2
BYOL (ours)	ResNet-200 (2×)	250M	**79.6**	**94.8**

(b) Other ResNet encoder architectures.

ImageNet accuracies of linear classifiers trained
on representations learned with different self-supervised methods

BYOL: Semi-supervised training

- The whole base network is fine-tuned on the few labeled data with spatial augmentations, i.e., random crops with resize and random flips.

Method	Top-1		Top-5	
	1%	10%	1%	10%
Supervised [77]	25.4	56.4	48.4	80.4
InstDisc	-	-	39.2	77.4
PIRL [35]	-	-	57.2	83.8
SimCLR [8]	48.3	65.6	75.5	87.8
BYOL (ours)	**53.2**	**68.8**	**78.4**	**89.0**

(a) ResNet-50 encoder.

Method	Architecture	Param.	Top-1		Top-5	
			1%	10%	1%	10%
CPC v2 [32]	ResNet-161	305M	-	-	77.9	91.2
SimCLR [8]	ResNet-50 (2×)	94M	58.5	71.7	83.0	91.2
BYOL (ours)	ResNet-50 (2×)	94M	62.2	73.5	84.1	91.7
SimCLR [8]	ResNet-50 (4×)	375M	63.0	74.4	85.8	92.6
BYOL (ours)	ResNet-50 (4×)	375M	69.1	75.7	87.9	92.5
BYOL (ours)	ResNet-200 (2×)	250M	**71.2**	**77.7**	**89.5**	**93.7**

(b) Other ResNet encoder architectures.

ImageNet accuracy of models trained with few labels

Few-shot learning

Few-shot learning

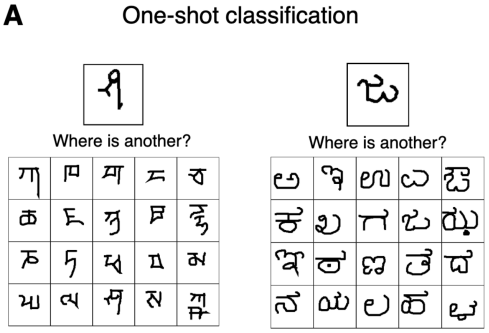
- People can learn new concepts from just a single example:

drone

segway

- Yet machine learning algorithms typically require thousands of examples to perform with similar accuracy (Lake et al., 2015).
- Few-shot learning: How can we train an accurate model using a very small amount of training data?

- The problem of few-shot learning attracted a lot of attention after releasing the Omniglot challenge (Lake et al., 2015).
- The authors also proposed a (non-deep) model that represents concepts as simple programs that best explain observed examples under a Bayesian criterion (BPL).
- BPL achieves human-level performance on the one-shot classification task.



Part A of Omniglot challenge: Two trials of one-shot classification, where a single image of a new character is presented (top) and the goal is to select another example of that character amongst other characters from the same alphabet (in the grid below).

Status of the Omniglot challenge (Lake et al., 2019)

- There has been a great progress of deep learning methods in few-shot learning tasks but deep learning seems to be behind BPL and human level.
- Deep learning methods rely heavily on data augmentation.

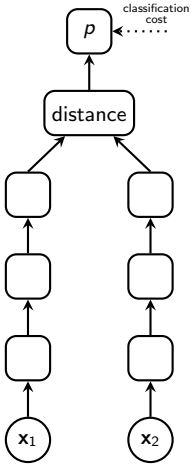
One-shot classification error rate

	Original		**Augmented**	
	Within alphabet	Within alphabet (minimal)	Within alphabet	Between alphabet
background set				
# alphabets	30	5	30	40
# classes	964	146	3,856	4,800
2015 results				
Humans	$\leq 4.5\%$			
BPL	**3.3%**	**4.2%**		
Simple ConvNet	13.5%	23.2%		
Siamese Net			8.0%*	
2016-2018 results				
Prototypical Net	13.7%	30.1%	**6.0%**	4.0%
Matching Net				6.2%
MAML				4.2%
Graph Net				**2.6%**
ARC			**1.5%***	**2.5%***
RCN	7.3%			
VHE	18.7%			4.8%

* results used additional data augmentation beyond class expansion

Siamese networks for one-shot learning (Koch et al., 2015)

- Consider the task of *one-shot* classification: building a classifier from one training example.
- A siamese neural network (Bromley et al., 1993) is trained to compare a pair of examples to decide whether they belong to the same class or not (binary classification):
  - Twin networks which accept distinct inputs
  - The output is computed using a distance (e.g., Euclidean) between the highest-level feature representations of the twin networks.
  - The network is trained on pairs of positive (same class) and negative (distinct classes) examples.
- Works for one-shot learning, few-shot requires tweaking.

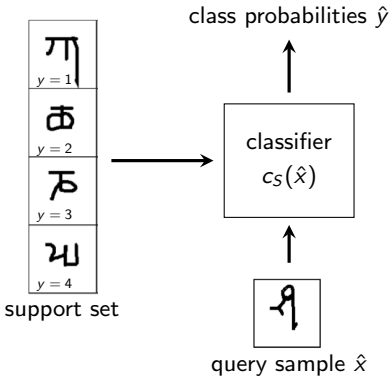


Matching networks (Vinyals et al., 2016)

- We wish to map from a (small) support set of k examples of image-label pairs $S = \{(x_i, y_i)\}_{i=1}^{k}$ to a classifier $c_S(\hat{x})$.

$$S \rightarrow c_S(\hat{x})$$

- The classifier $c_S(\hat{x})$ defines a probability distribution $\hat{y}$ over classes in the support set for a query example $\hat{x}$.
- Matching networks: Parameterize this mapping $S \rightarrow c_S(\hat{x})$ as a neural network.



Matching networks: The model

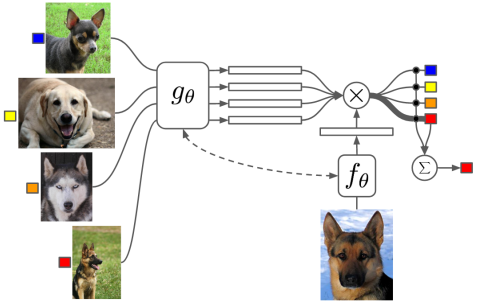
- The output of the classifier $c_S(\hat{x})$:

$$\hat{y} = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i$$

with an attention mechanism a:

$$a(\hat{x}, x_i) = \frac{\exp(c(f(\hat{x}), g(x_i)))}{\sum_{j=1}^{k} \exp(c(f(\hat{x}), g(x_j)))}$$

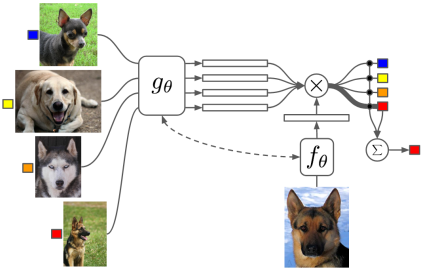
- f and g are parameterized as neural networks.



- Intuition: support samples whose representations $g(x_i)$ are close to the representation $f(\hat{x})$ of the query sample contribute more to the output.
- We tune the representations to work well in the few-shot learning scenario.

Matching networks: Episodic training

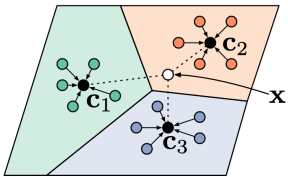
- N-way K-shot classification task: Each training example is a classification task with N classes and K training examples per class.
- One iteration of episodic training:
  - Select N classes by randomly sampling from the training set.
  - Select a support set by taking K random samples for each of the selected classes.
  - Select a query set (a few samples from the same classes as in the support set).
  - Do forward computations and compute the classification loss using the query samples.
  - Do backpropagation and update the parameters of the network.



Prototypical networks (Snell et al., 2017)

- Prototypical networks compute an M-dimensional representation $\mathbf{c}_k$, or prototype, of each class through an embedding function f_{θ}. Each prototype is the mean vector of the embedded support points belonging to its class:

$$\mathbf{c}_k = \frac{1}{|S_k|} \sum_{(x_i, y_i) \in S_k} f_{\theta}(x_i)$$



- Then they produce a distribution over classes for a query point x based on a softmax over distances to the prototypes in the embedding space:

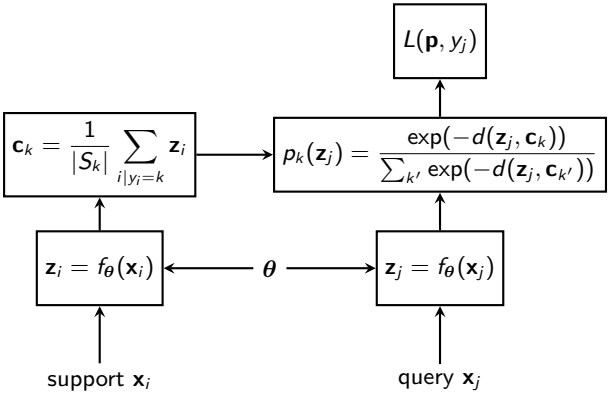
$$p(y = k \mid x) = \frac{\exp(-d(f_{\theta}(x), \mathbf{c}_k))}{\sum_{k'} \exp(-d(f_{\theta}(x), \mathbf{c}_{k'}))}$$

- In one-shot learning scenario, prototypical networks are equivalent to matching networks.

Prototypical networks: Training

One iteration of episodic training:

- The support set is used to compute the prototypes.
- The query set is used to compute the loss.
- Compute the gradients of the loss and update the parameters $\boldsymbol{\theta}$ of the embedding network.



Computational graph

Model-Agnostic Meta-Learning (MAML; Finn et al., 2017)

- We want to train a classifier $y = f_{\boldsymbol{\theta}}(\mathbf{x})$ to solve a new few-shot learning task.
- Learning can be done by performing a few iterations of gradient descent (GD).
- In the case of one iteration of GD:

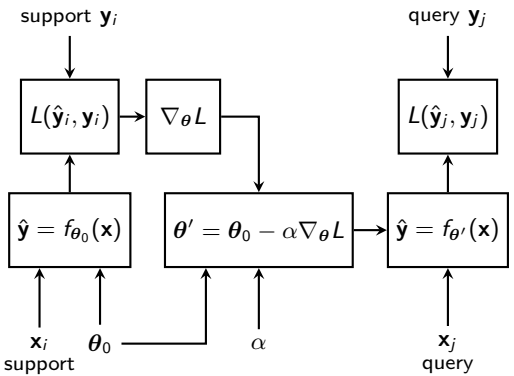
$$\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta}_0 - \alpha \nabla_{\boldsymbol{\theta}} L((\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_k, \mathbf{y}_k))$$

  - $\{(x_i, y_i)\}_{i=1}^{k}$ are the few training examples (support set)
  - L is the loss function (for example, cross-entropy for classification tasks)
  - α is the learning rate
  - $\boldsymbol{\theta}_0$ is the vector of the initial values of the parameters
- The idea of meta-learning: we can learn initialization $\boldsymbol{\theta}_0$ and the learning rate α to minimize the loss (on the query set) after the GD-adaptation.

Model-Agnostic Meta-Learning: Training

One iteration of episodic training:

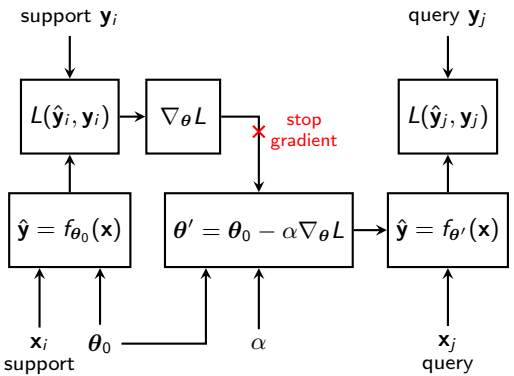
- Use the support set to compute the loss and its gradient $\nabla_{\boldsymbol{\theta}} L$.
- Compute adapted values $\boldsymbol{\theta}'$ of the parameters (as part of computational graph) with one (or a few) iteration of gradient descent.
- Use the query set to compute the loss with the adapted parameters $\boldsymbol{\theta}'$.
- Compute the loss on the query set.
- Perform backpropagation and update parameters $\boldsymbol{\theta}_0$ and learning rate α.



Computational graph

Model-Agnostic Meta-Learning: First-order approximation

- MAML requires computation of gradient through gradient, which can be computationally expensive.
- The first-order approximation (which stops gradient propagation through $\nabla_{\boldsymbol{\theta}} L$) works almost equally well.



Computational graph

Reptile (Nichol et al., 2018)

- Simplification of MAML: Instead of backpropagating through the computational graph of MAML, we update the initial parameter values $\boldsymbol{\theta}_0$ towards the adapted parameter values $\boldsymbol{\theta}_T$ with a small step ϵ.

  for iteration 1,2,3,... **do**
  Randomly sample a task T
  Perform $k > 1$ steps of SGD on task T starting from $\boldsymbol{\theta}_0$: $\boldsymbol{\theta}_0 \rightarrow \boldsymbol{\theta}_T$
  Update: $\boldsymbol{\theta}_0 \leftarrow \boldsymbol{\theta}_0 + \epsilon(\boldsymbol{\theta}_T - \boldsymbol{\theta}_0)$
  return $\boldsymbol{\theta}_0$

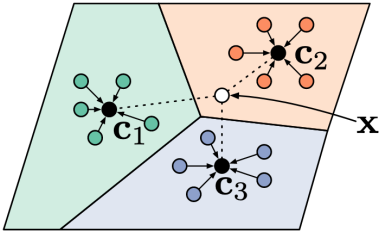
- Reptile demo

Home assignment

- You need to implement prototypical networks (Snell et al., 2017).

$$\mathbf{c}_k = \frac{1}{|S_k|} \sum_{(x_i, y_i) \in S_k} f_\theta(x_i)$$

$$p(y = k \mid x) = \frac{\exp(-d(f_\theta(x), \mathbf{c}_k))}{\sum_{k'} \exp(-d(f_\theta(x), \mathbf{c}_{k'}))}$$



Recommended reading

- Papers cited in the lecture slides.