Transfer learning

Pablo Arias, Gabriele Facciolo

ML&IA

25/11/2021

Contents

Contents

Review: supervised training

We have a set of data points with corresponding **labels**

$$(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m), \quad \text{with } x_i \in \mathcal{X}, y_i \in \mathcal{Y}, \quad \text{(e.g. } \mathcal{X} = \mathbb{R}^d, \mathcal{Y} = \mathbb{R}^t\text{)}$$

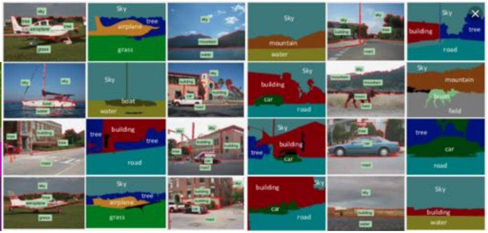
We assume that the data pairs are IID samples of a joint PDF: $(x_i, y_i) \sim p(x, y)$.

We want a function $\mathcal{F}_\theta : \mathcal{X} \to \mathcal{Y}$ such that $\mathcal{F}_\theta(x)$ "$=$" y, for $(x, y) \sim p(x, y)$.

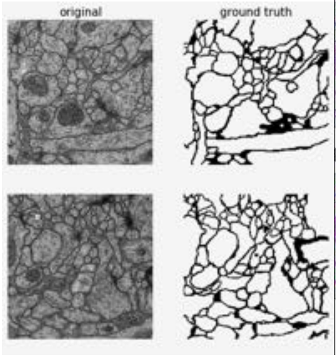
To that end we define a **loss function** ℓ which measures the error between $\mathcal{F}_\theta(x)$ and y, and set the **parameters** θ to minimize the expected loss:

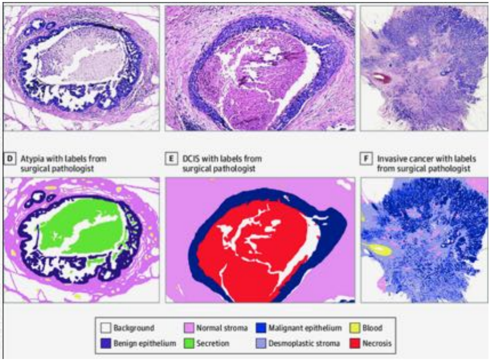
$$\underbrace{\mathcal{R}^{\text{emp}}(\theta) = \frac{1}{m}\sum_{i=1}^{m} \ell(\mathcal{F}_\theta(x_i), y_i)}_{\text{empirical risk}} \to_{m\to\infty} \underbrace{\mathbb{E}\{\ell(\mathcal{F}_\theta(x), y)\} = \mathcal{R}(\theta)}_{\text{risk}}.$$

Supervised training needs labels

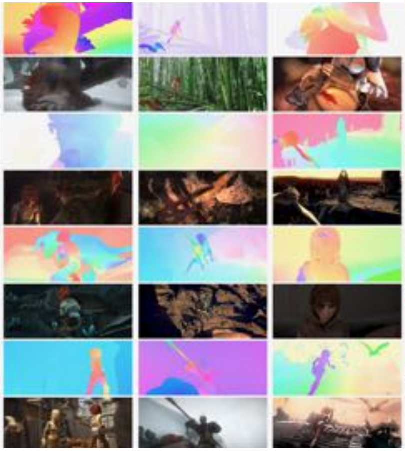


Supervised training needs labels




D Atypia with labels from surgical pathologist
E DCIS with labels from surgical pathologist
F Invasive cancer with labels from surgical pathologist
□ Background
🟪 Normal stroma
🟦 Malignant epithelium
🟨 Blood
🟦 Benign epithelium
🟩 Secretion
🟪 Desmoplastic stroma
🟥 Necrosis

Supervised training needs labels



Labelling is expensive (or impossible)

In the best case label is expensive: requires several hours of effort.

- classification
- segmentation

Sometimes, it requires expert knowledge:

- medical imaging
- satellite imagery

For some problems the ground truth is not know, or very difficult to measure:

- motion estimation
- depth estimation
- image restoration

In practice: small labeled dataset

We need strategies to cope with small datasets. We have see already:

- data augmentation: synthetically generate new data by applying transforms to existing data
- regularization: prevent overfitting to the dataset

These help, but are insufficient if dataset is very small.

Transfer learning

A function (e.g. a network) $\mathcal{F}_S : \mathcal{X}_S \to \mathcal{Y}_S$ has been trained to trained to solve a **source problem**:

$$S = (\mathcal{X}_S, \mathcal{Y}_S, p_S(x, y) = p_S(x)p_S(y|x), \ell_S).$$

Can it be used to help training a **second target problem**?

$$T = (\mathcal{X}_T, \mathcal{Y}_T, p_T(x, y) = p_T(x)p_T(y|x), \ell_T).$$

Transfer learning

Most usual cases:

(Inductive) transfer learning: Input spaces are the same, but task changes:

$$\begin{cases} (\mathcal{X}_S, p_S(x)) = (\mathcal{X}_T, p_T(x)), \\ \mathcal{Y}_T \neq \mathcal{Y}_S \text{ or } p_S(y|x) \neq p_T(y|x) \text{ or } \ell_T \neq \ell_S. \end{cases}$$

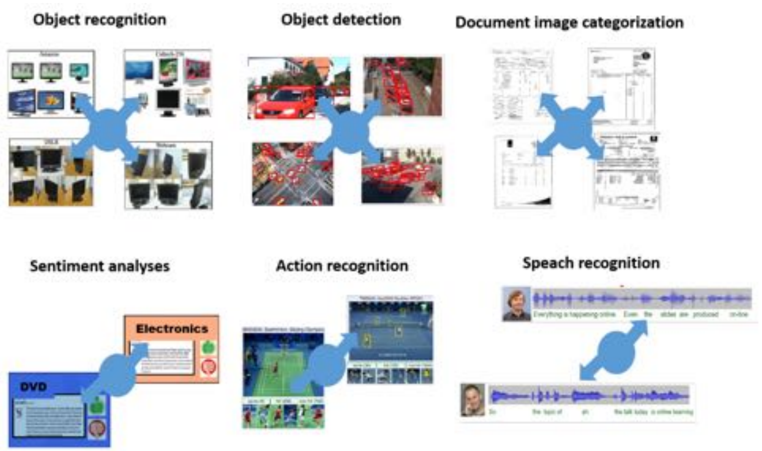
Example: detect objects in natural images $\rightarrow$ segmentation of natural images

Domain adaptation: Input spaces are different, but task is the same:

$$\begin{cases} \mathcal{X}_S \neq \mathcal{X}_T \text{ or } p_S(x)) \neq p_T(x), \\ \mathcal{Y}_T = \mathcal{Y}_S \text{ and } \ell_T = \ell_S. \end{cases}$$

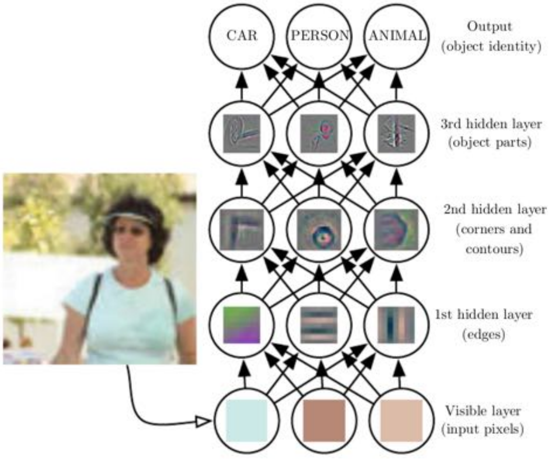
Example: sentiment classification in hotel reviews $\rightarrow$ sentiment classification in reviews of technological items

Examples of domain adaptation problems


Object recognition
Object detection
Document image categorization
Sentiment analyses
Electronics
DVD
Action recognition
Speach recognition
Everything is happening online. Even the slides are produced on-line
So the topic of the talk today is online learning

Contents

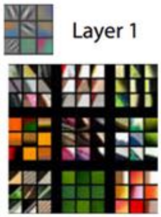
Intuition: why is transfer learning possible



Weights in first layers are low-level features (related to input domain). In deeper layers more complex higher level concepts begin to appear (related to task).

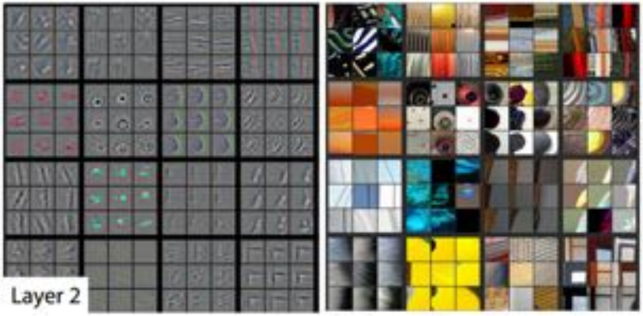
Figure from [Deep Learning, Goodfellow, Bengio and Courville, 2016]

Intuition: why is transfer learning possible



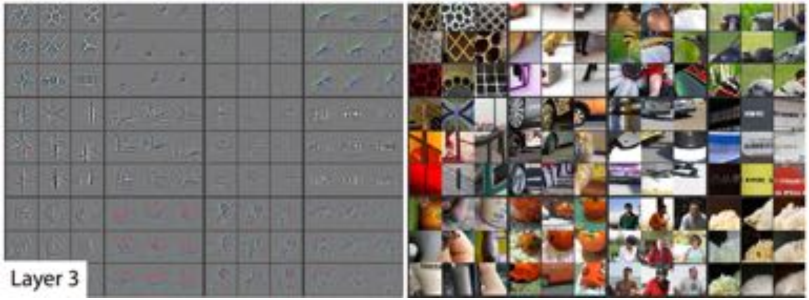
Figures from [Visualizing and Understanding Convolutional Networks. Zeiler, Fergus, 2013]

Intuition: why is transfer learning possible



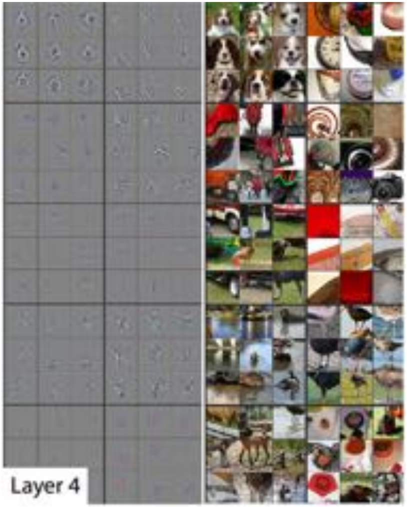
Figures from [Visualizing and Understanding Convolutional Networks. Zeiler, Fergus, 2013]

Intuition: why is transfer learning possible



Figures from [Visualizing and Understanding Convolutional Networks. Zeiler, Fergus, 2013]

Intuition: why is transfer learning possible



Figures from [Visualizing and Understanding Convolutional Networks. Zeiler, Fergus, 2013]

Intuition: why is transfer learning possible

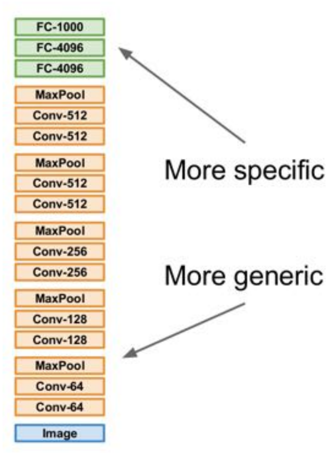
Figures from [Visualizing and Understanding Convolutional Networks. Zeiler, Fergus, 2013]

Intuition: why is transfer learning possible

https://www.youtube.com/watch?v=AgkfIQ4IGaM

Demo video from one of the authors of [Understanding Neural Networks Through Deep Visualization, Yosinski et al. 2015.]

Transfer learning in practice



Early layers

- low-level features (edges, textures, corners, blobs)
- seem to be independent from the task
- networks for different tasks have similar filters

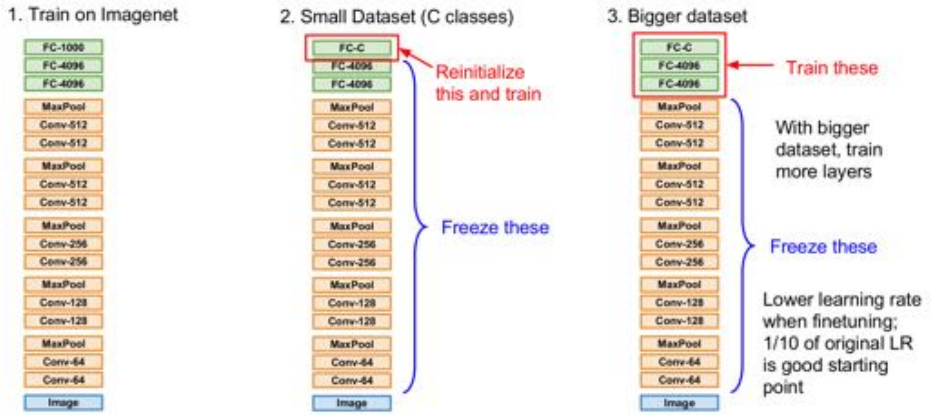
Deeper layers

- higher level concepts (faces, text, clothing)
- could be transfered between tasks requiring similar semantic concepts

Final layers (head of the network)

- computes the required output: classifier, localization, etc.

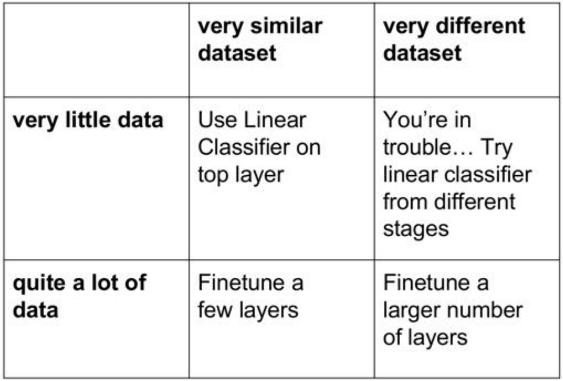
Transfer learning in practice: fine-tune from ImageNet



Final layers have to be replaced with new ones and trained from scratch.

The weights of the “freezed” layers can be kept constant, or used as initialization for a **fine-tuning** with a small learning rate. The larger the training set, the more layers we can fine-tune. This is a trial-and-error process.

Transfer learning in practice

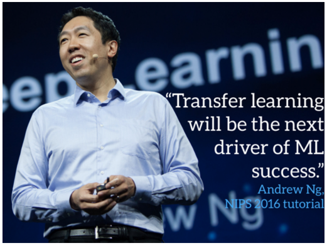


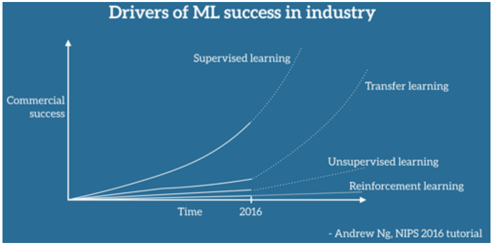
	very similar dataset	**very different dataset**
very little data	Use Linear Classifier on top layer	You’re in trouble… Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

This and the previous 2 slides were taken from [CS231n course of Stanford University]

Contents

Transfer learning is widely used


"Transfer learning will be the next driver of ML success."
Andrew Ng, NIPS 2016 tutorial

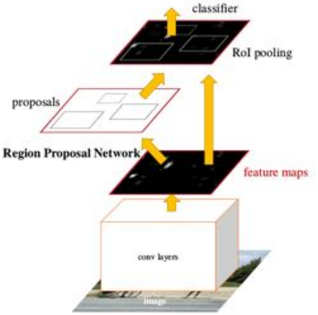


Andrew Ng likes transfer learning and plots with unlabeled axes.

Transfer learning in detection and localization

. . . when labeled training data is scarce, supervised pre-training for an auxiliary task, followed by domain-specific fine-tuning, yields a significant performance boost.

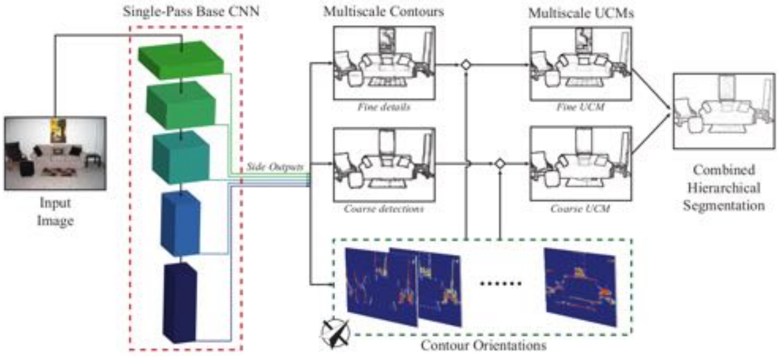
[R-CNN, Girshick et al. 2014]



All other layers (. . .) are initialized by pre-training a model for ImageNet classification [36], as is standard practice.

[Faster R-CNN, Ren et al. 2016]

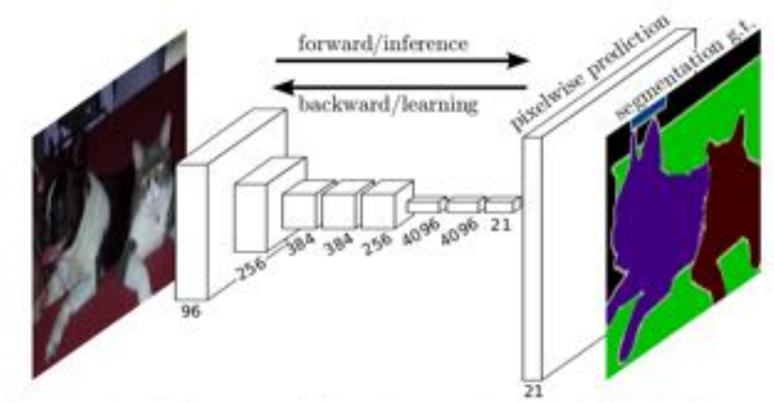
Transfer learning in boundary detection



Features are extracted from different scales of a base CNN, and fed to a contour detection network. The base CNN is initialized using a classification network trained on ImageNet.

[Convolutional Oriented Boundaries: From Image Segmentation to High-Level Tasks, Maninis et al. 2017]

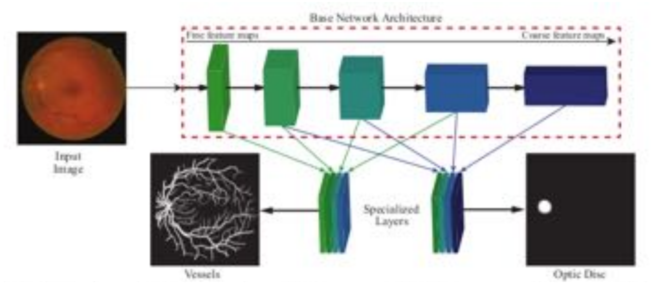
Transfer learning in semantic segmentation



We adapt contemporary classification networks (AlexNet [22], the VGG net [34], and GoogLeNet [35]) into fully convolu- tional networks and transfer their learned representations by fine-tuning [5] to the segmentation task.

[Fully Convolutional Networks for Semantic Segmentation, Long et al. 2014]

Transfer learning in medical image segmentation

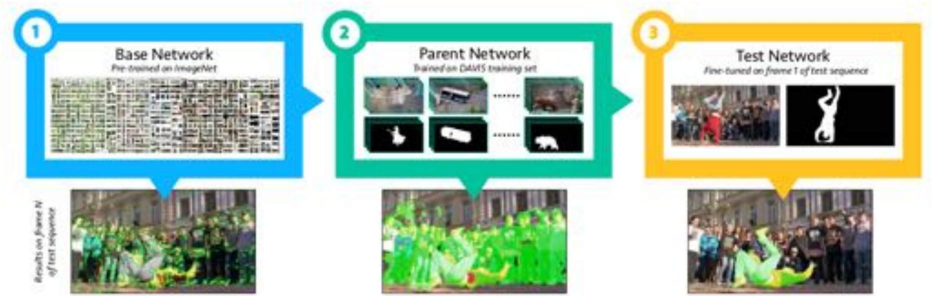


We start from the VGG [18] network (. . .) the fully connected layers at the end of the network are removed.

(. . .) we fine-tune the entire architecture for 20000 iterations. Due to the lack of data, the learning rate is set to a very small number.

[Deep retinal image understanding, Maninis et al. 2016]

Transfer learning in video object segmentation


1
Base Network
Pre-trained on ImageNet
2
Parent Network
Trained on DAVIS training set
3
Test Network
Fine-tuned on frame 1 of test sequence
Results on frame N of test sequence

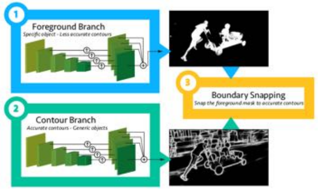

1
Foreground Branch
Specific object - Less accurate contours
2
Contour Branch
Accurate contours - Generic objects
3
Boundary Snapping
Snap the foreground mask to accurate contours

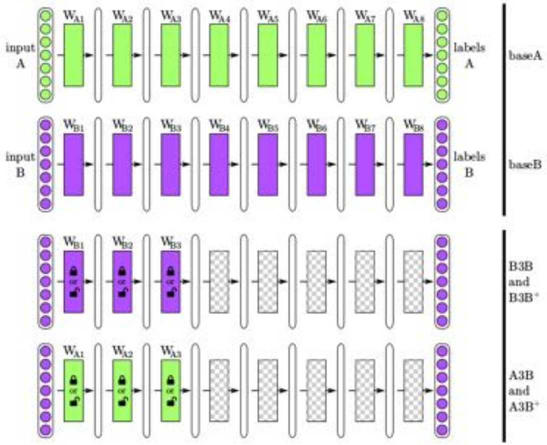
Figure 4. **Two-stream FCN architecture**: The main foreground branch (1) is complemented by a contour branch (2) which improves the localization of the boundaries (3).

The base CNN is pre-trained on ImageNet for image labeling, which has proven to be a very good initialization to other tasks.

[One-Shot Video Object Segmentation, Caelles et al. 2017]

Contents

How transferable are features in a deep NN?

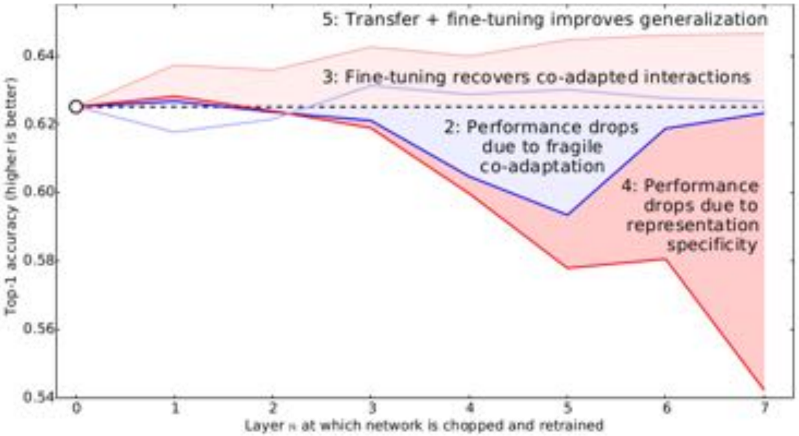


[Yosinski et al., 2014] transfer classificaction tasks A to B (two halfs of ImageNet).

XnY: first n layers trained for task A, remaining final layers trained from scratch for B.

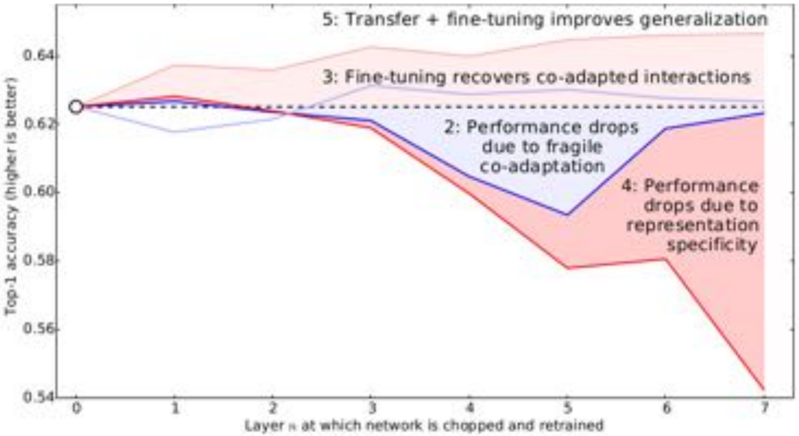
XnY$^+$: same, with fine-tuning.

How transferable are features in a deep NN?



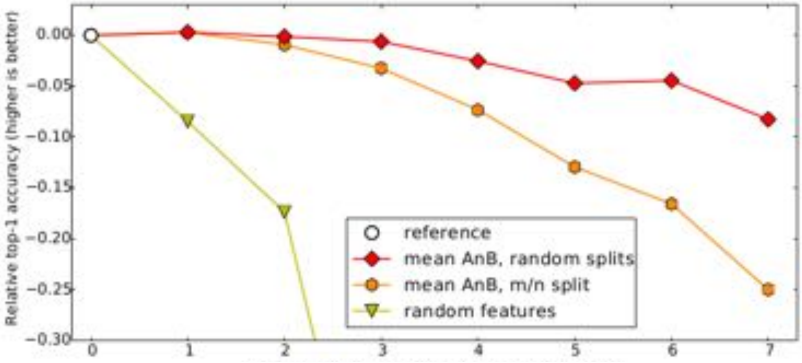
Strong purple: BnB drops at $n = 4, 5$. Surprinsing: re-training last layers from scratch can't recover original performance. *Fagile co-apadaptation*: neurons in consecutive layers adapt to each other. Retraining from scratch is not able to find the adapted weigths. Soft purple (BnB$^+$): Fine-tuning finds the co-adapted weights.

How transferable are features in a deep NN?



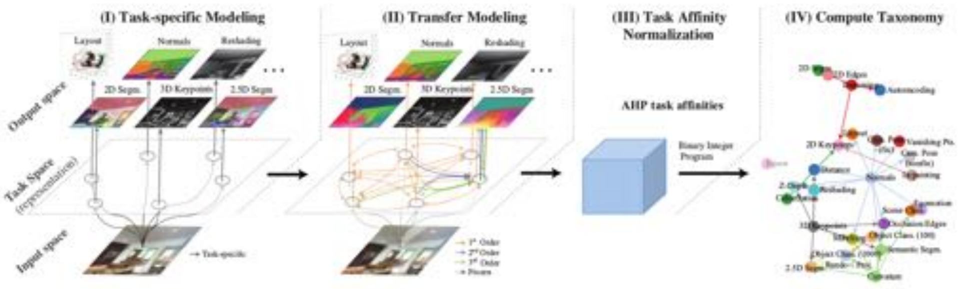
Strong red: AnB starts decreasing at $n = 4$ (lost co-adapted neurons), and continue to drop for larger n. This shows that deeper layers are specific for task A and do not transfer well to B. Soft red (AnB$^+$): Fine-tuning fixes this, and it **even performs better than the network trained on B**, suggesting that transfer learning helps in generalization.

How transferable are features in a deep NN?



Red: tasks A and B chosen as random splits of ImageNet. Tasks A and B are similar. Orange: tasks A and B chosen as splits of ImageNet in different classes (natural objects vs. man-made objects). As A and B differ, tansfer performs worse.

Taskonomy: a taxonomy of learning tasks



[Zamir et al., 2018]: compute transferability between different tasks.

1. Train encoder-decoder networks for 26 tasks using large training set (120k)
2. Compute all possible transfers: freeze encoder from source task, train decoder from scratch using small traning set (1k or 16k)
3. The performance of each transfered network, is used to create normalized task affinity matrix
4. Compute a graph selecting which are the best n tasks to transfer to the rest

Taskonomy: a taxonomy of learning tasks

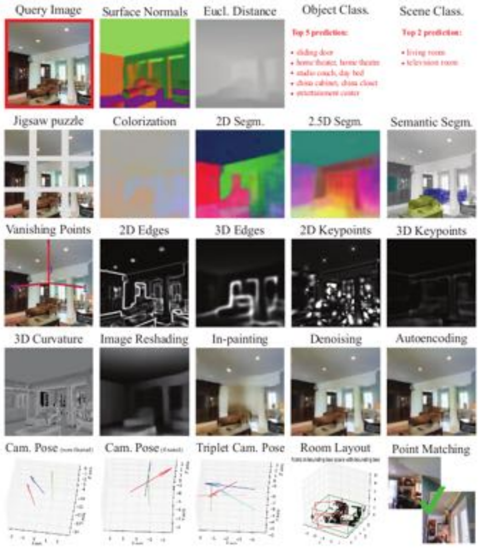
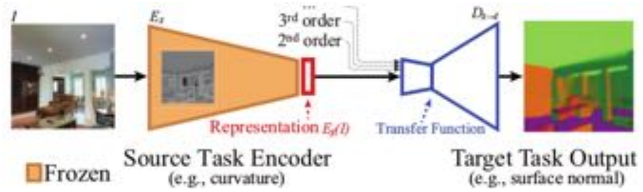


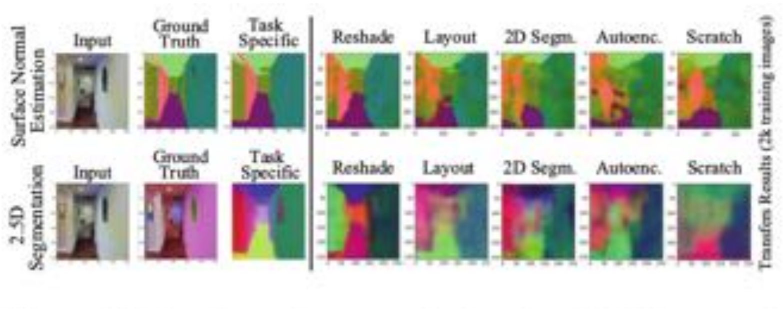
Figure 3: **Task Dictionary.** Outputs of 24 (of 26) task-specific networks for a query (top left). See results of applying frame-wise on a video here.

Taskonomy: a taxonomy of learning tasks



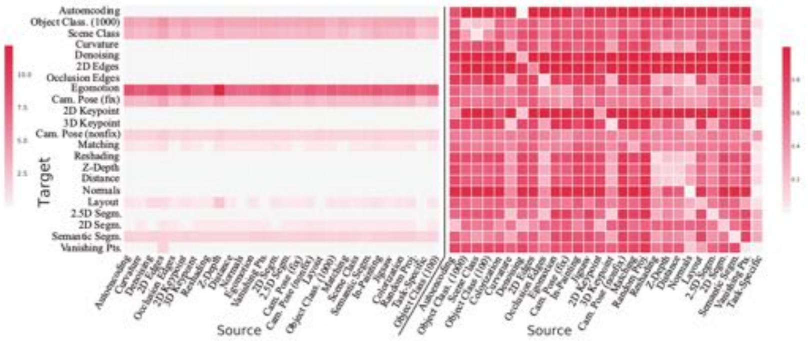
Enconder-decoder networks are used. The encoder is a Resnet-50 and is transfered between task. A decoder network is then trained for the target task (15 layer CNN for when the output is an image, 2FC layers if output is of smaller dimension). The authors also study higher order transfers, when the outputs of more than one encoder are used by the decoder.

Taskonomy: a taxonomy of learning tasks



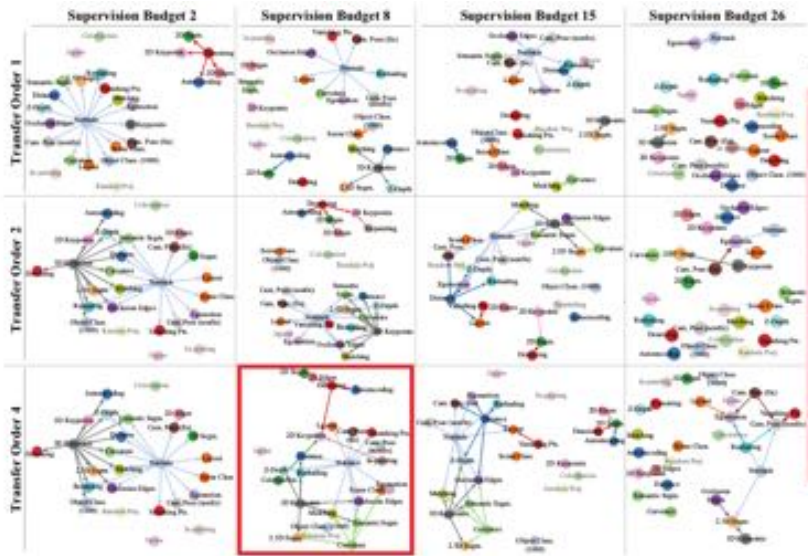
Example results obtained when transfering from 5 source tasks to surface normal estimation and 2.5D segmentation for different transfers. Clearly some tasks are more similar than other.

Taskonomy: a taxonomy of learning tasks



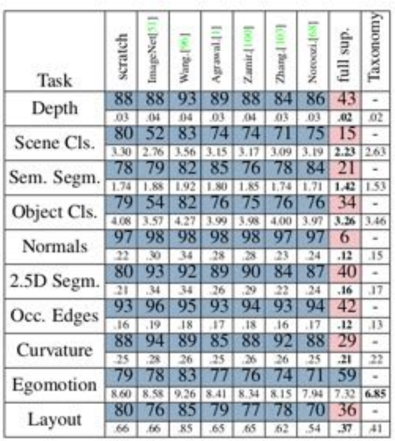
Affinity matrix for first order transfers, before and after normalization (the normalization is needed to accound for the different magnitudes in task losses). Each row shows the transferability from a task to the others. Lower values means better transferability.

Taskonomy: a taxonomy of learning tasks



Optimal transfer graphs found for different transfer orders and "supervision budget" (how many source tasks are trained from scratch with supervision).

Taskonomy: a taxonomy of learning tasks



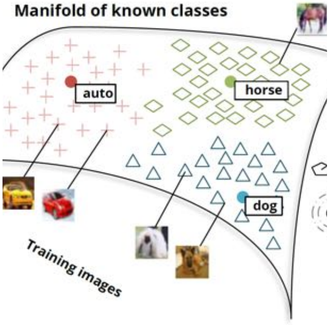
Task	scratch	ImageNet[51]	Wang.[96]	Agrawal.[1]	Zamir.[100]	Zhang.[107]	Noroozi.[68]	full sup.	Taxonomy
Depth	88	88	93	89	88	84	86	43	-
	.03	.04	.04	.03	.04	.03	.03	**.02**	.02
Scene Cls.	80	52	83	74	74	71	75	15	-
	3.30	2.76	3.56	3.15	3.17	3.09	3.19	**2.23**	2.63
Sem. Segm.	78	79	82	85	76	78	84	21	-
	1.74	1.88	1.92	1.80	1.85	1.74	1.71	**1.42**	1.53
Object Cls.	79	54	82	76	75	76	76	34	-
	4.08	3.57	4.27	3.99	3.98	4.00	3.97	**3.26**	3.46
Normals	97	98	98	98	98	97	97	6	-
	.22	.30	.34	.28	.28	.23	.24	**.12**	.15
2.5D Segm.	80	93	92	89	90	84	87	40	-
	.21	.34	.34	.26	.29	.22	.24	**.16**	.17
Occ. Edges	93	96	95	93	94	93	94	42	-
	.16	.19	.18	.17	.18	.16	.17	**.12**	.13
Curvature	88	94	89	85	88	92	88	29	-
	.25	.28	.26	.25	.26	.26	.25	**.21**	.22
Egomotion	79	78	83	77	76	74	71	59	-
	8.60	8.58	9.26	8.41	8.34	8.15	7.94	7.32	**6.85**
Layout	80	76	85	79	77	78	70	36	-
	.66	.66	.85	.65	.65	.62	.54	**.37**	.41

Comparison against tranfer from ImageNet and from networks trained with self-supervised approaches.

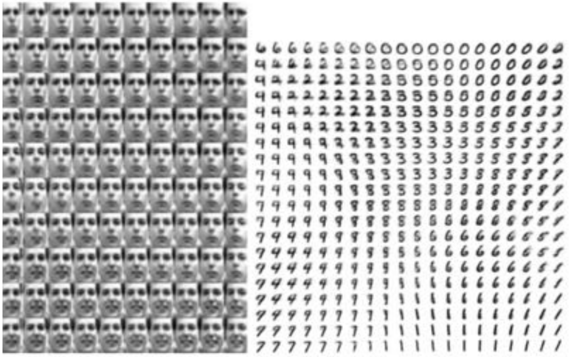
Contents

Representation (or feature) learning

An old problem in machine learning: find a representation of the data $\varphi(x)$ which is compact and useful for posterior applications.



For example: object classes should be easily (linearly) separable



The coordinates in $\varphi(x) \in \mathbb{R}^f$ should capture the main modes of variation in the data.

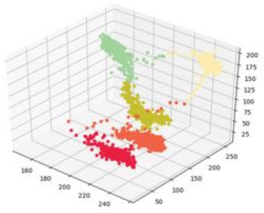
Figures from [Socher et al. 2013] and [Kingma, Welling, 2013]

Unsupervised representation learning

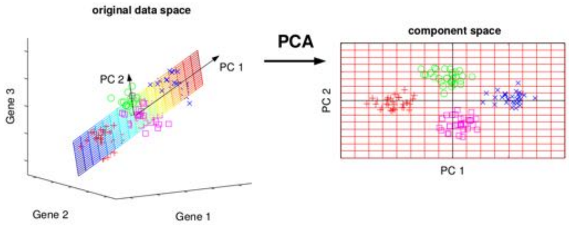
Representation learning has been typically addressed using **unsupervised training**:

- No external labels are required
- The goal of the model is not to predict an output (with notable exceptions)
- Learn structure, patterns, modes of variation from unlabeled dataset
- Since no labels are needed, a lot of data is available

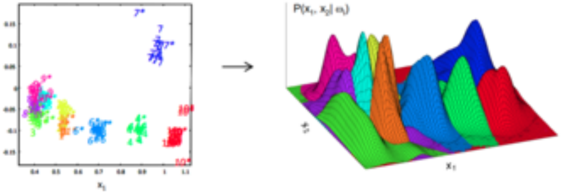
Examples of unsupervised learning



Find clusters in dataset



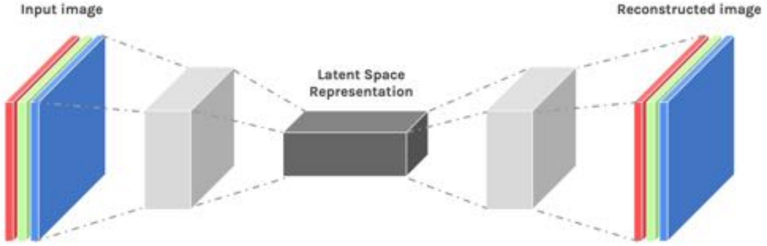
Dimensionality reduction (e.g. PCA)



Density estimation

Figures from [Swagatam Das, 2019], [Eulertech] and [Matthias Scholz, 2006]

Example of unsupervised representation learning: autoencoders



Autoencoders have been proposed for learning a feature representation:

1. Given an image x, the encoder networks coputes the code $\varphi(x)$ (also embedding, latent representation, etc).
2. The decoder network reconstruct the image $\hat{x} = \psi(\varphi(x))$ from the code $\varphi(x)$
3. Both networks are trained end-to-end by minimizing $\|\hat{x} - x\|^2$
4. Even if the training requires a loss, this is considered often unsupervised training
5. The decoder is mainly used for training, and then it is discarded. The encoder $\varphi(x)$ can then be used for different tasks by appending a small network.

Problems of autoencoders for representation learning

- The squared L^2 reconstruction loss does not encourage learning the high level semantic concepts in the representation.
- To reproduce the same input image, the representation $\varphi(x)$ needs to encode **all** factors or variation in the data.
- The relevance of the factors depends on the application: illumination changes are irrelevant for face recognition, but relevant for head pose estimation.

Figure from [Pattabhi Ramaiah et al. 2015]

Self-supervised representation learning

Self-supervised representation learning aims at solving these issues:

- Train a network to solve an auxiliary task for which we know the labels (the **pretext** task)
- General principle: hide some information from the input, and train the network to recover it.
- It is supervised learning, but does not require an external label, since the label is part of the data (the hidden information)!!
- The pretext task has to be related to the real task we need to solve, so that we can **transfer**.

Ten examples of pretext tasks

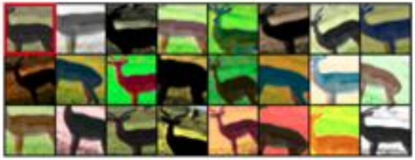


Fig. 2. The original patch of a cute deer is in the top left corner. Random transformations are applied, resulting in a variety of distorted patches. All of them should be classified into the same class in the pretext task. (Image source: Dosovitskiy et al., 2015)

All figures and captions captured from [Lilian Weng, 2019]

Ten examples of pretext tasks

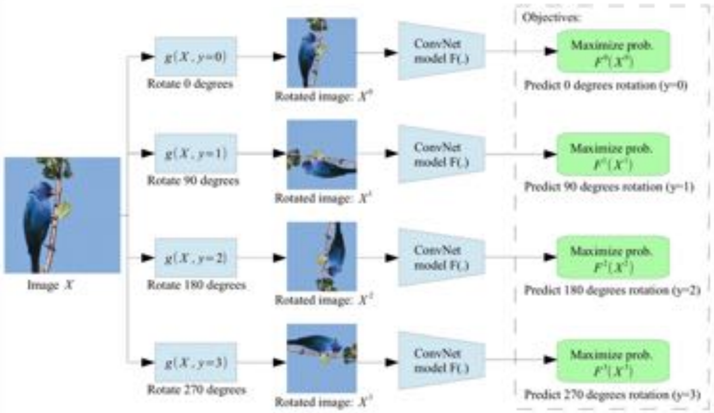


Fig. 3. Illustration of self-supervised learning by rotating the entire input images. The model learns to predict which rotation is applied. (Image source: Gidaris et al. 2018)

All figures and captions captured from [Lilian Weng, 2019]

Ten examples of pretext tasks

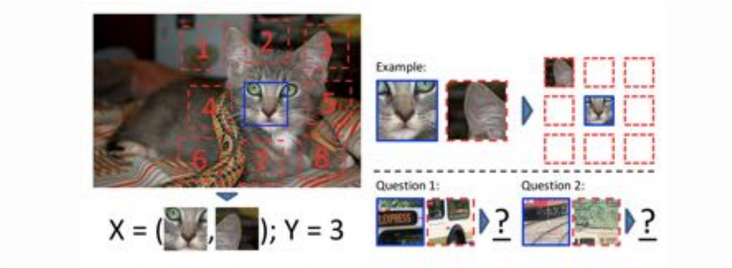


Fig. 4. Illustration of self-supervised learning by predicting the relative position of two random patches. (Image source: Doersch et al., 2015)

All figures and captions captured from [Lilian Weng, 2019]

Ten examples of pretext tasks

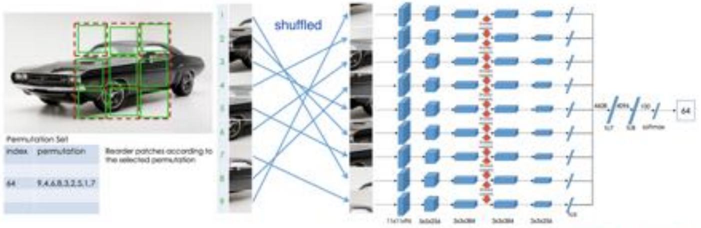


Fig. 6. Illustration of self-supervised learning by solving jigsaw puzzle. (Image source: Noroozi & Favaro, 2016)

Ten examples of pretext tasks

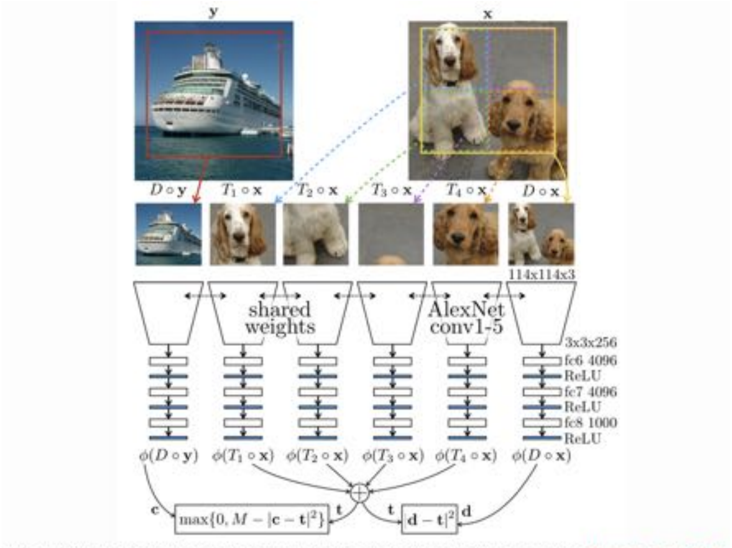

y
x
$D \circ \mathbf{y}$
$T_1 \circ \mathbf{x}$
$T_2 \circ \mathbf{x}$
$T_3 \circ \mathbf{x}$
$T_4 \circ \mathbf{x}$
$D \circ \mathbf{x}$
114x114x3
shared weights
AlexNet conv1-5
3x3x256
fc6 4096
ReLU
fc7 4096
ReLU
fc8 1000
ReLU
$\phi(D \circ \mathbf{y})$
$\phi(T_1 \circ \mathbf{x})$
$\phi(T_2 \circ \mathbf{x})$
$\phi(T_3 \circ \mathbf{x})$
$\phi(T_4 \circ \mathbf{x})$
$\phi(D \circ \mathbf{x})$
c
t
d
$\max\{0, M - |\mathbf{c} - \mathbf{t}|^2\}$
$|\mathbf{d} - \mathbf{t}|^2$

Fig. 7. Self-supervised representation learning by counting features. (Image source: Noroozi, et al, 2017)

All figures and captions captured from [Lilian Weng, 2019]

Ten examples of pretext tasks

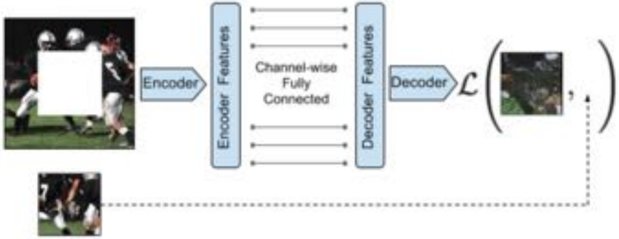


Fig. 8. Illustration of context encoder. (Image source: Pathak, et al., 2016)

All figures and captions captured from [Lilian Weng, 2019]

Ten examples of pretext tasks

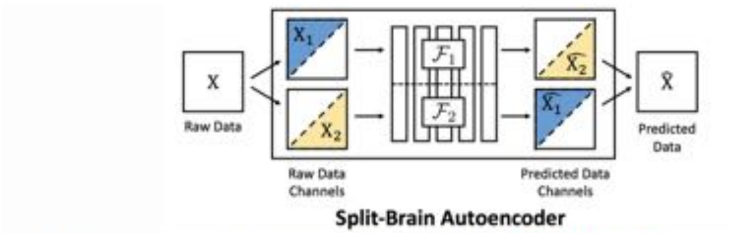


Fig. 9. Illustration of split-brain autoencoder. (Image source: Zhang et al., 2017)

Ten examples of pretext tasks

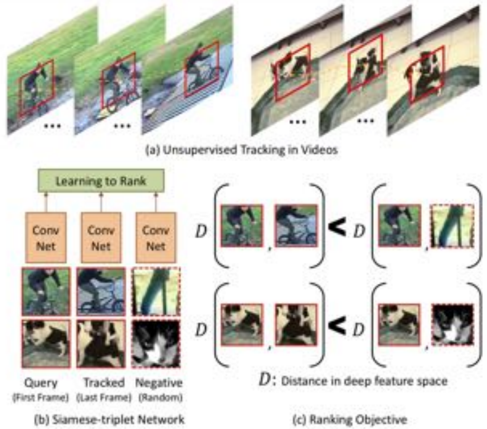


Fig. 11. Overview of learning representation by tracking objects in videos. (a) Identify moving patches in short traces; (b) Feed two related patched and one random patch into a conv network with shared weights. (c) The loss function enforces the distance between related patches to be closer than the distance between random patches. (Image source: Wang & Gupta, 2015)

All figures and captions captured from [Lilian Weng, 2019]

Ten examples of pretext tasks

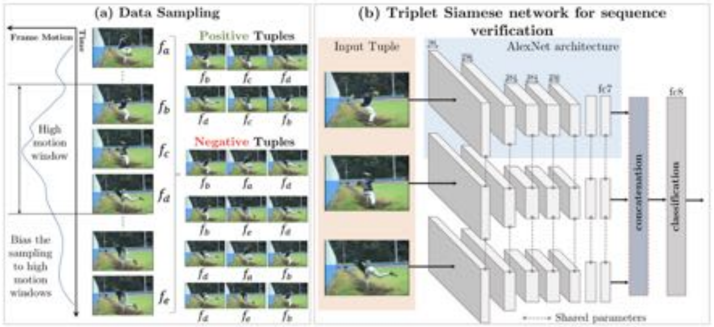


Fig. 12. Overview of learning representation by validating the order of video frames. (a) the data sample process; (b) the model is a triplet siamese network, where all input frames have shared weights. (Image source: Misra, et al 2016)

Ten examples of pretext tasks

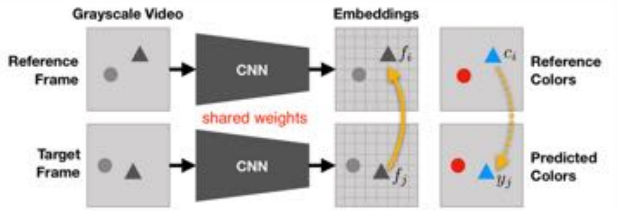


Fig. 14. Video colorization by copying colors from a reference frame to target frames in grayscale. (Image source: Vondrick et al. 2018)

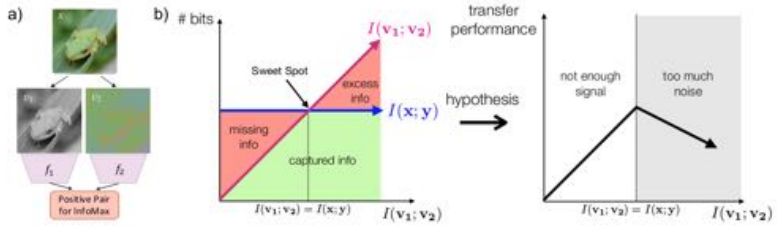
State-of-the-art: contrastive representation learning

Problem with pre-text tasks

- Pre-text tasks are interesting, but they are far away from the supervised methots (in image classification, object detection, segmentation)
- The choice of the pre-text class is crucial. How to choose the optimal pre-text class for a given target task?

Contrastive representation learning: a more systematic approach to self-supervised learning. (although there are a lot of cooking details that matter!!!)

Intuition about the pre-text class



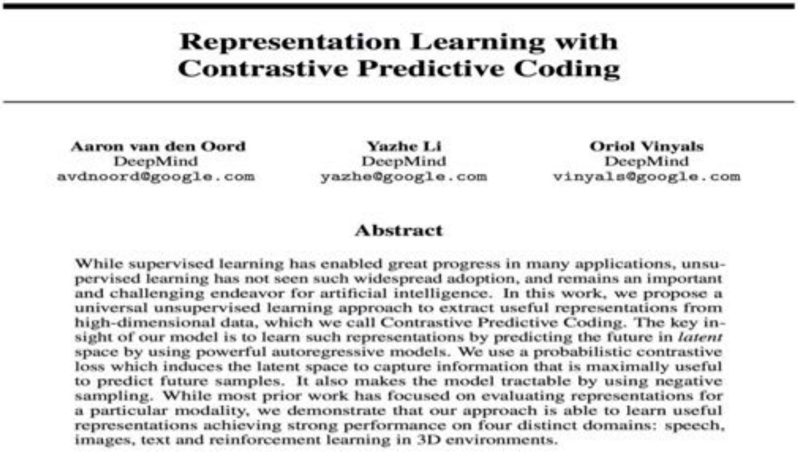
a)
x
v_1
v_2
f_1
f_2
Positive Pair for InfoMax
b)
# bits
$I(\mathbf{v_1};\mathbf{v_2})$
Sweet Spot
excess info
$I(\mathbf{x};\mathbf{y})$
missing info
captured info
$I(\mathbf{v_1};\mathbf{v_2}) = I(\mathbf{x};\mathbf{y})$
$I(\mathbf{v_1};\mathbf{v_2})$
hypothesis
transfer performance
not enough signal
too much noise
$I(\mathbf{v_1};\mathbf{v_2}) = I(\mathbf{x};\mathbf{y})$
$I(\mathbf{v_1};\mathbf{v_2})$

Figure 1: **(a)** Schematic of multiview contrastive representation learning, where an image is split into two views, and passed through two encoders to learn an embedding where the views are close relative to views from other images. **(b)** When we have views that maximize $I(\mathbf{v_1};\mathbf{y})$ and $I(\mathbf{v_2};\mathbf{y})$ (how much task-relevant information is contained) while minimizing $I(\mathbf{v_1};\mathbf{v_2})$ (information shared between views, including both task-relevant and irrelevant information), there are three regimes: *missing information* which leads to degraded performance due to $I(\mathbf{v_1};\mathbf{v_2}) < I(\mathbf{x};\mathbf{y})$; *excess noise* which worsens generalization due to additional noise; *sweet spot* where the only information shared between $\mathbf{v_1}$ and $\mathbf{v_2}$ is task-relevant and such information is complete.

- signal: variation factors related to the task we want to solve
- noise: variation factors unrelated to the task (nuisance factors)

[Yonglong Tian et al, 2020]

Contrastive Predictive Coding for audio processing



Representation Learning with Contrastive Predictive Coding

Aaron van den Oord
DeepMind
avdnoord@google.com

Yazhe Li
DeepMind
yazhe@google.com

Oriol Vinyals
DeepMind
vinyals@google.com

Abstract

While supervised learning has enabled great progress in many applications, unsupervised learning has not seen such widespread adoption, and remains an important and challenging endeavor for artificial intelligence. In this work, we propose a universal unsupervised learning approach to extract useful representations from high-dimensional data, which we call Contrastive Predictive Coding. The key insight of our model is to learn such representations by predicting the future in *latent* space by using powerful autoregressive models. We use a probabilistic contrastive loss which induces the latent space to capture information that is maximally useful to predict future samples. It also makes the model tractable by using negative sampling. While most prior work has focused on evaluating representations for a particular modality, we demonstrate that our approach is able to learn useful representations achieving strong performance on four distinct domains: speech, images, text and reinforcement learning in 3D environments.

Contrastive Predictive Coding - main concept

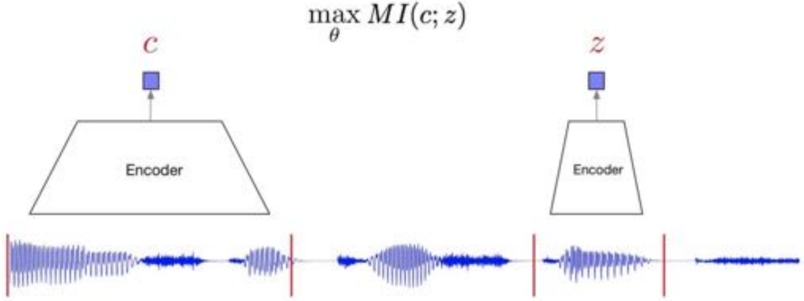


Figure from Alex Graves

Contrastive Predictive Coding - main concept

$$\frac{\exp f(c, z_i)}{\sum_j \exp f(c, z_j)}$$

$$f_k(x_{t+k}, c_t) = \exp\left(z_{t+k}^T W_k c_t\right)$$

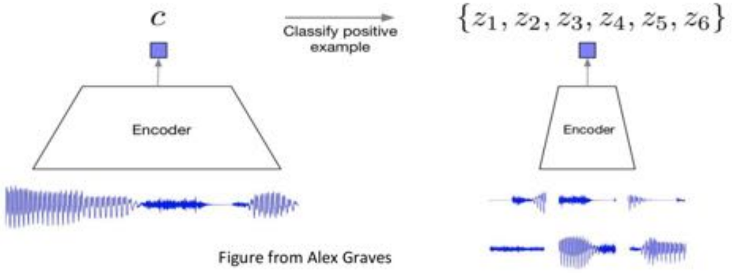
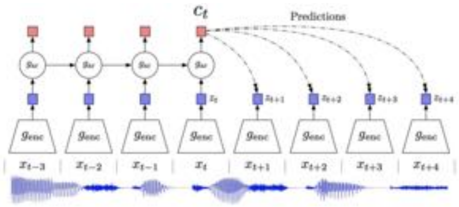


Figure from Alex Graves

Contrastive Predictive Coding - implementation

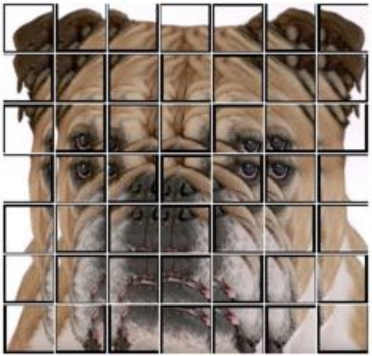


$$f_k(x_{t+k}, c_t) = \exp\left(z_{t+k}^T W_k c_t\right)$$

$$\mathcal{L}_{\mathrm{N}} = -\mathop{\mathbb{E}}_{X}\left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)}\right]$$

Figure from Alex Graves

Contrastive Predictive Coding for image patches



Contrastive Predictive Coding for image patches

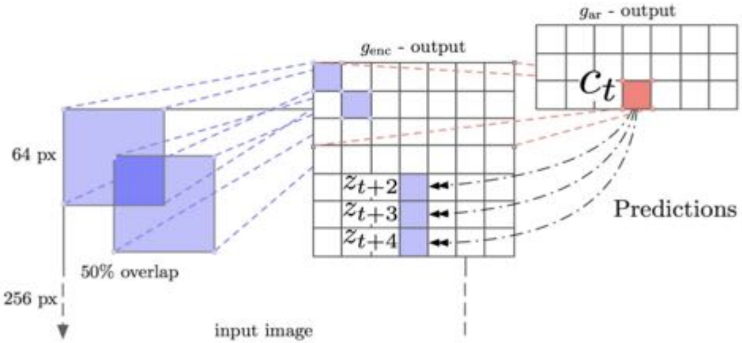
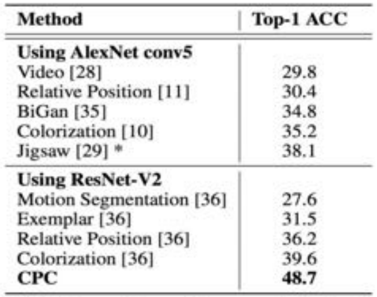


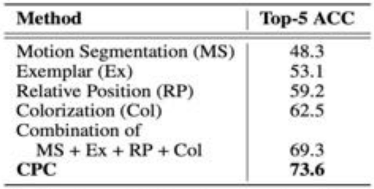
Figure 4: Visualization of Contrastive Predictive Coding for images (2D adaptation of Figure 1).

Contrastive Predictive Coding for image patches



Method	Top-1 ACC
Using AlexNet conv5	
Video [28]	29.8
Relative Position [11]	30.4
BiGan [35]	34.8
Colorization [10]	35.2
Jigsaw [29] *	38.1
Using ResNet-V2	
Motion Segmentation [36]	27.6
Exemplar [36]	31.5
Relative Position [36]	36.2
Colorization [36]	39.6
CPC	**48.7**

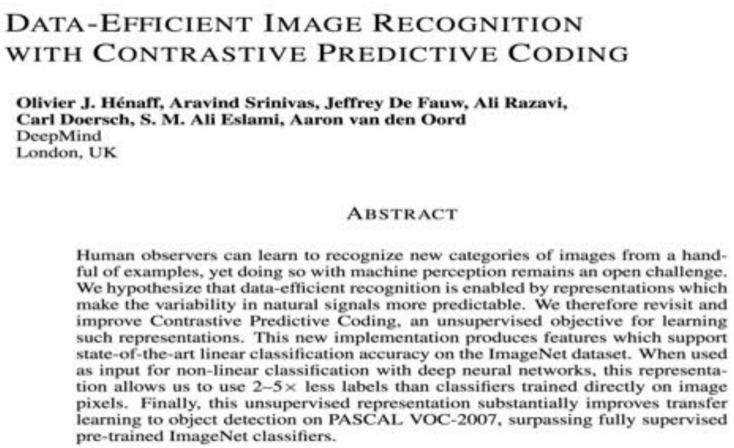
Table 3: ImageNet top-1 unsupervised classification results. *Jigsaw is not directly comparable to the other AlexNet results because of architectural differences.



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**

Table 4: ImageNet top-5 unsupervised classification results. Previous results with MS, Ex, RP and Col were taken from [36] and are the best reported results on this task.

Contrastive Predictive Coding version 2



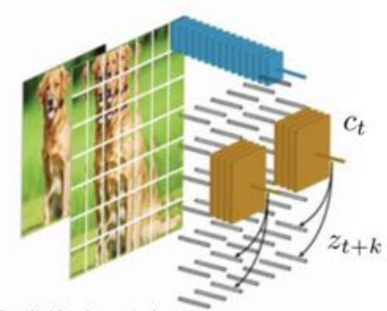
DATA-EFFICIENT IMAGE RECOGNITION WITH CONTRASTIVE PREDICTIVE CODING

Olivier J. Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, S. M. Ali Eslami, Aaron van den Oord
DeepMind
London, UK

ABSTRACT

Human observers can learn to recognize new categories of images from a handful of examples, yet doing so with machine perception remains an open challenge. We hypothesize that data-efficient recognition is enabled by representations which make the variability in natural signals more predictable. We therefore revisit and improve Contrastive Predictive Coding, an unsupervised objective for learning such representations. This new implementation produces features which support state-of-the-art linear classification accuracy on the ImageNet dataset. When used as input for non-linear classification with deep neural networks, this representation allows us to use 2–5× less labels than classifiers trained directly on image pixels. Finally, this unsupervised representation substantially improves transfer learning to object detection on PASCAL VOC-2007, surpassing fully supervised pre-trained ImageNet classifiers.

Contrastive Predictive Coding version 2



Parallel Implementation
with PixelCNN (masked conv) and 1x1 conv

InfoNCE Loss

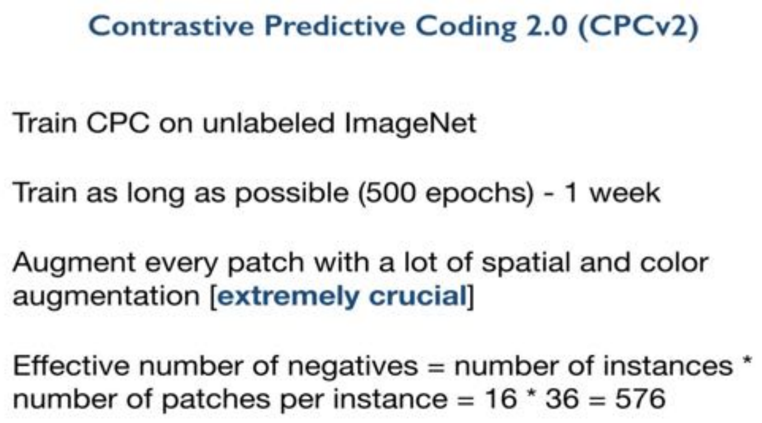
$$f_k(x_{t+k}, c_t) = \exp\left(z_{t+k}^T W_k c_t\right)$$

$$\mathcal{L}_\mathrm{N} = -\mathop{\mathbb{E}}_{X}\left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)}\right]$$

$\downarrow$ Negatives

1. Other patches within image
2. Patches from other images

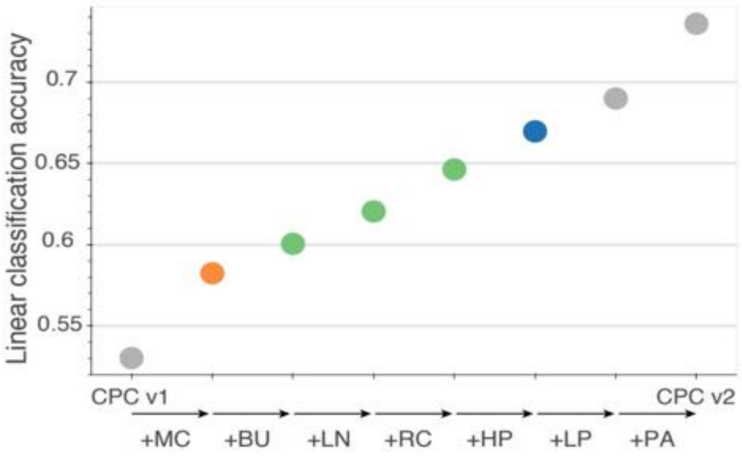
Contrastive Predictive Coding: details matter!



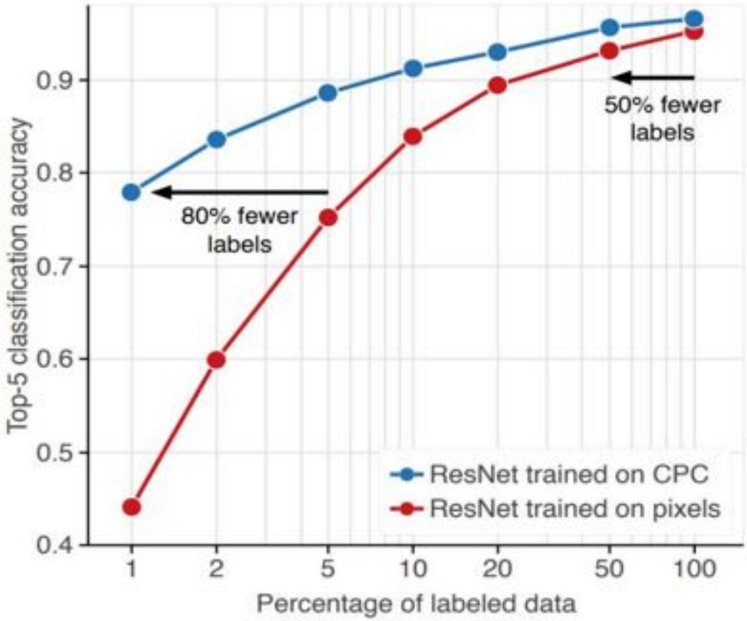
Contrastive Predictive Coding 2.0 (CPCv2)

- Train CPC on unlabeled ImageNet
- Train as long as possible (500 epochs) - 1 week
- Augment every patch with a lot of spatial and color augmentation [**extremely crucial**]
- Effective number of negatives = number of instances * number of patches per instance = 16 * 36 = 576

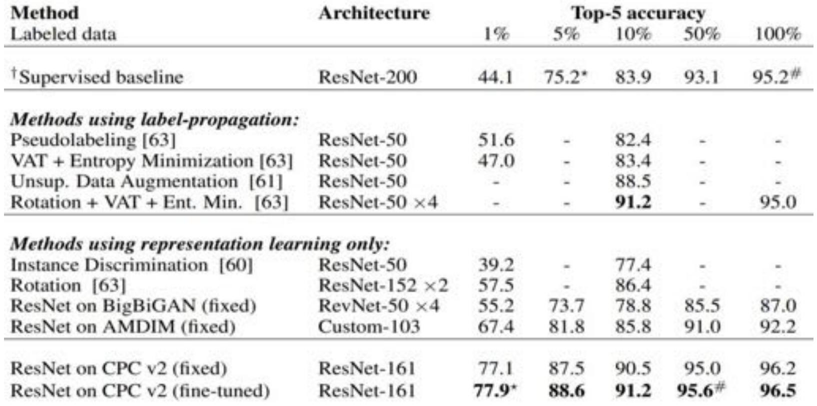
Contrastive Predictive Coding: details matter!



Contrastive Predictive Coding - results

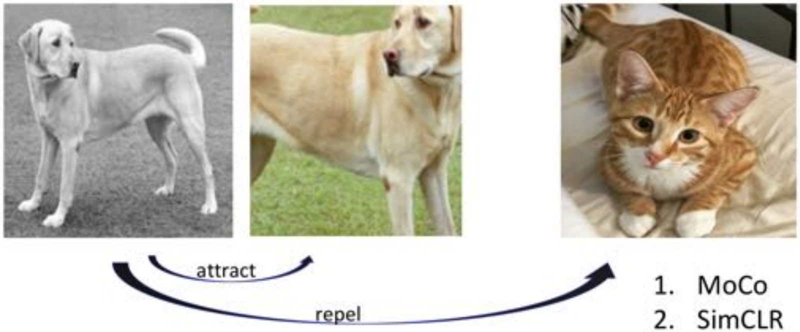


Contrastive Predictive Coding - more results



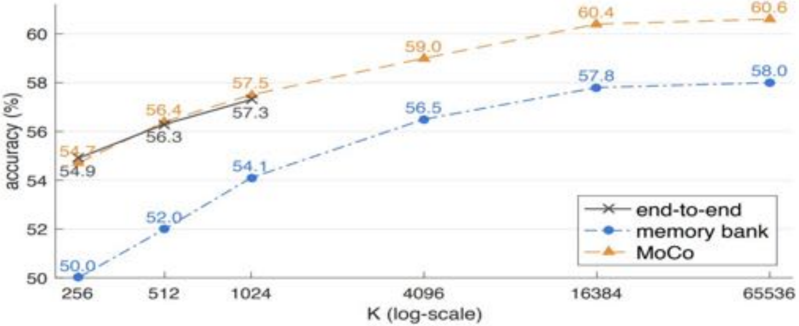
Method	**Architecture**	**Top-5 accuracy**				
Labeled data		1%	5%	10%	50%	100%
†Supervised baseline	ResNet-200	44.1	75.2*	83.9	93.1	95.2#
Methods using label-propagation:						
Pseudolabeling [63]	ResNet-50	51.6	-	82.4	-	-
VAT + Entropy Minimization [63]	ResNet-50	47.0	-	83.4	-	-
Unsup. Data Augmentation [61]	ResNet-50	-	-	88.5	-	-
Rotation + VAT + Ent. Min. [63]	ResNet-50 ×4	-	-	**91.2**	-	95.0
Methods using representation learning only:						
Instance Discrimination [60]	ResNet-50	39.2	-	77.4	-	-
Rotation [63]	ResNet-152 ×2	57.5	-	86.4	-	-
ResNet on BigBiGAN (fixed)	RevNet-50 ×4	55.2	73.7	78.8	85.5	87.0
ResNet on AMDIM (fixed)	Custom-103	67.4	81.8	85.8	91.0	92.2
ResNet on CPC v2 (fixed)	ResNet-161	77.1	87.5	90.5	95.0	96.2
ResNet on CPC v2 (fine-tuned)	ResNet-161	**77.9***	**88.6**	**91.2**	**95.6#**	**96.5**

Contrastive losses on full images: instance discrimination

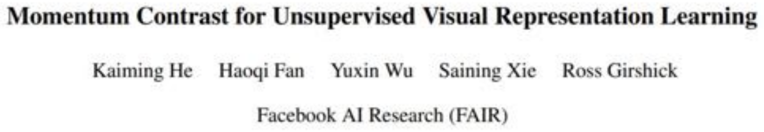


1. MoCo
2. SimCLR

The problem: we need a lot of negative samples



MoCo

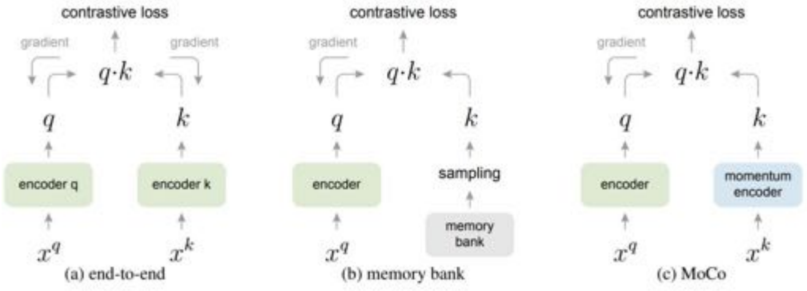


Momentum Contrast for Unsupervised Visual Representation Learning

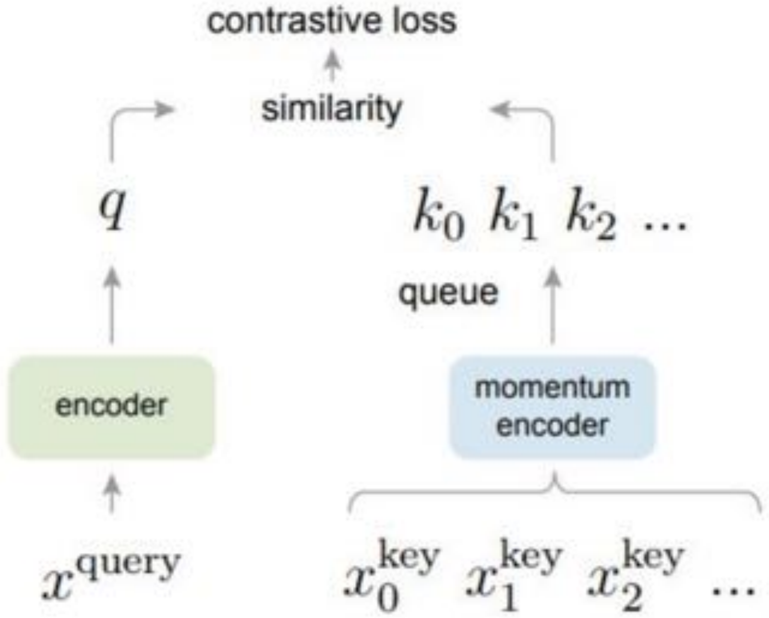
Kaiming He Haoqi Fan Yuxin Wu Saining Xie Ross Girshick

Facebook AI Research (FAIR)

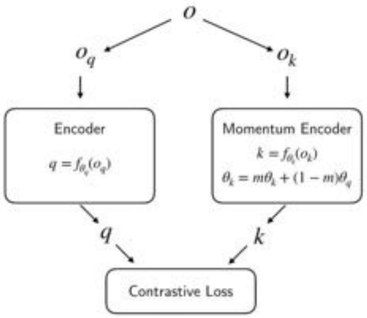
Three different approaches for intance discrimination



MoCo: main idea

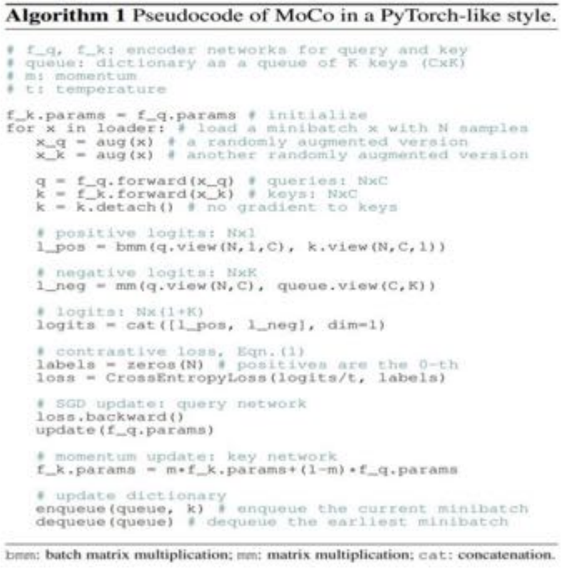


MoCo: main idea



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

MoCo: pseudo-code

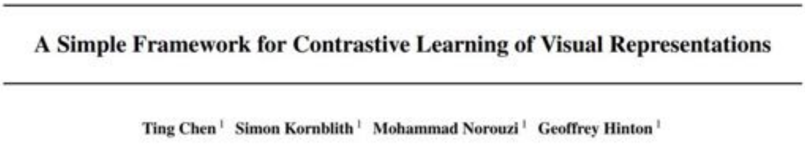


Algorithm 1 Pseudocode of MoCo in a PyTorch-like style.

```
# f_q, f_k: encoder networks for query and key
# queue: dictionary as a queue of K keys (CxK)
# m: momentum
# t: temperature

f_k.params = f_q.params # initialize
for x in loader: # load a minibatch x with N samples
   x_q = aug(x) # a randomly augmented version
   x_k = aug(x) # another randomly augmented version

   q = f_q.forward(x_q) # queries: NxC
   k = f_k.forward(x_k) # keys: NxC
   k = k.detach() # no gradient to keys

   # positive logits: Nx1
   l_pos = bmm(q.view(N,1,C), k.view(N,C,1))

   # negative logits: NxK
   l_neg = mm(q.view(N,C), queue.view(C,K))

   # logits: Nx(1+K)
   logits = cat([l_pos, l_neg], dim=1)

   # contrastive loss, Eqn.(1)
   labels = zeros(N) # positives are the 0-th
   loss = CrossEntropyLoss(logits/t, labels)

   # SGD update: query network
   loss.backward()
   update(f_q.params)

   # momentum update: key network
   f_k.params = m*f_k.params+(1-m)*f_q.params

   # update dictionary
   enqueue(queue, k) # enqueue the current minibatch
   dequeue(queue) # dequeue the earliest minibatch
```

bmm: **batch matrix multiplication;** mm: **matrix multiplication;** cat: **concatenation.**

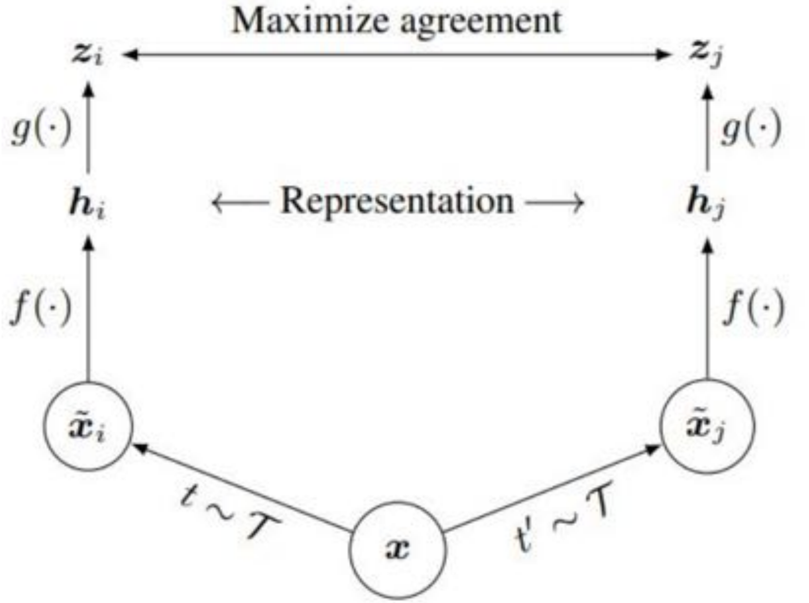
SimCLR: what if we just use a lot of TPUs?



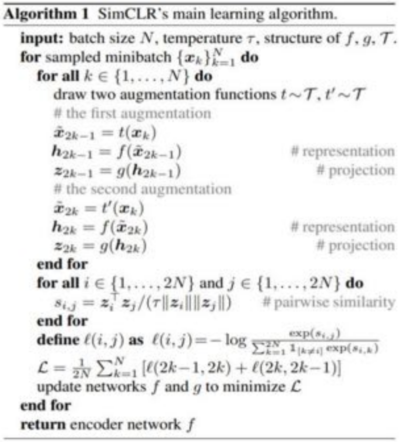
A Simple Framework for Contrastive Learning of Visual Representations

Ting Chen[1] Simon Kornblith[1] Mohammad Norouzi[1] Geoffrey Hinton[1]

SimCLR: additional transform before the loss



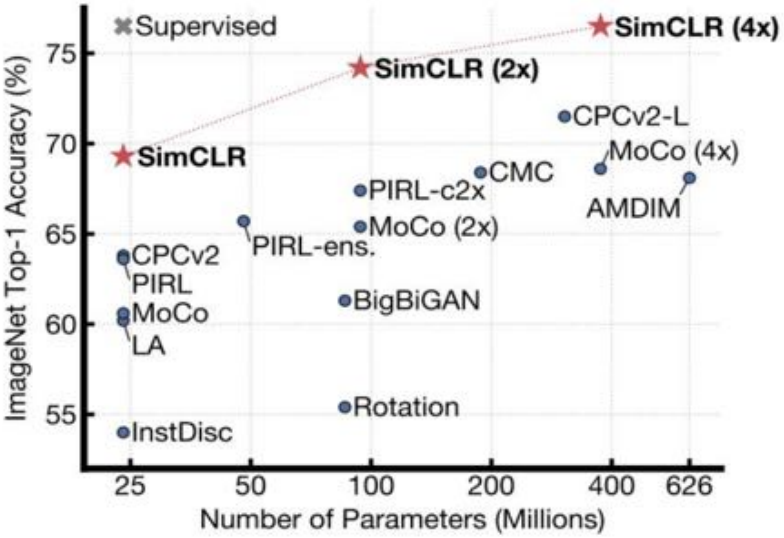
SimCLR: pseudo-code



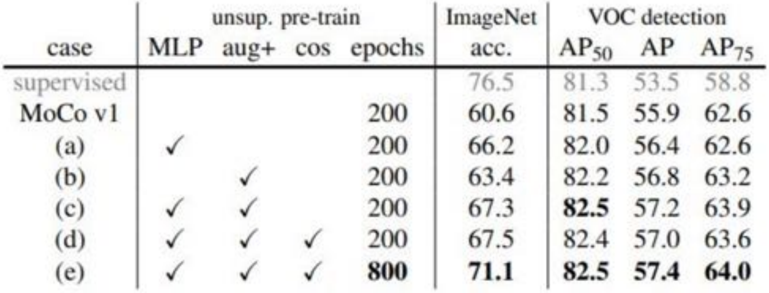
Algorithm 1 SimCLR's main learning algorithm.

```
input: batch size $N$, temperature $\tau$, structure of $f$, $g$, $\mathcal{T}$.
for sampled minibatch $\{\boldsymbol{x}_k\}_{k=1}^{N}$ do
    for all $k \in \{1, \ldots, N\}$ do
        draw two augmentation functions $t \sim \mathcal{T}$, $t' \sim \mathcal{T}$
        # the first augmentation
        $\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)$
        $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$                # representation
        $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$                        # projection
        # the second augmentation
        $\tilde{\boldsymbol{x}}_{2k} = t'(\boldsymbol{x}_k)$
        $\boldsymbol{h}_{2k} = f(\tilde{\boldsymbol{x}}_{2k})$                    # representation
        $\boldsymbol{z}_{2k} = g(\boldsymbol{h}_{2k})$                            # projection
    end for
    for all $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$ do
        $s_{i,j} = \boldsymbol{z}_i^\top \boldsymbol{z}_j / (\tau \|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)$      # pairwise similarity
    end for
    define $\ell(i,j)$ as $\ell(i,j) = -\log \frac{\exp(s_{i,j})}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k})}$
    $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
    update networks $f$ and $g$ to minimize $\mathcal{L}$
end for
return encoder network $f$
```

SimCLR: performance

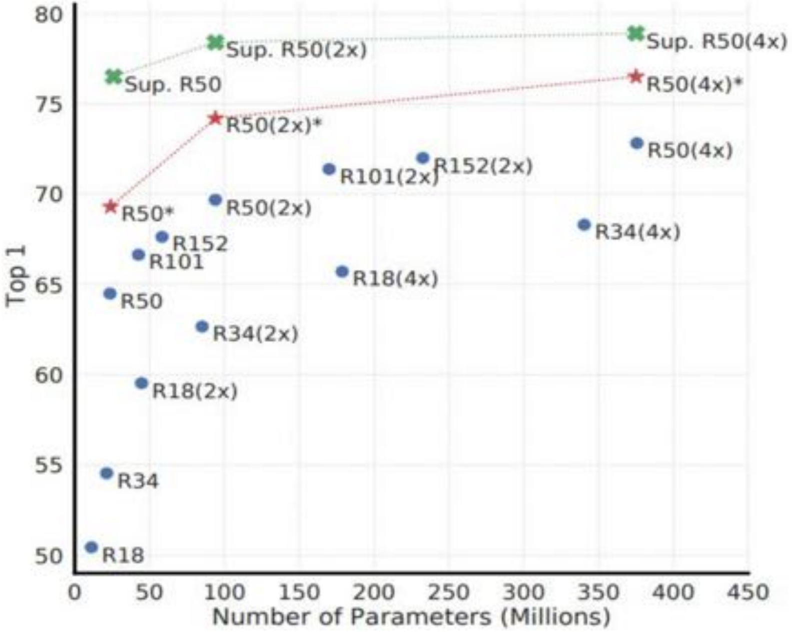


MoCo v2: takes some ideas from SimCLR

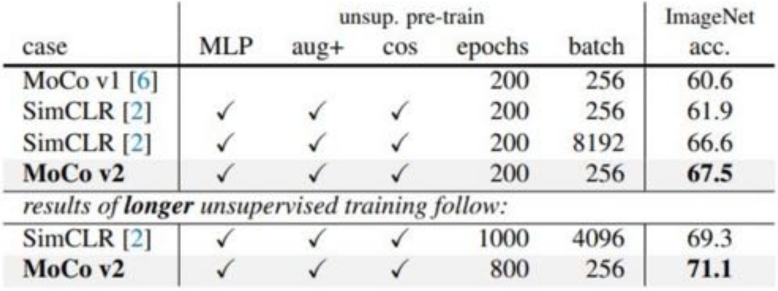


	unsup. pre-train				ImageNet	VOC detection		
case	MLP	aug+	cos	epochs	acc.	AP_{50}	AP	AP_{75}
supervised					76.5	81.3	53.5	58.8
MoCo v1				200	60.6	81.5	55.9	62.6
(a)	✓			200	66.2	82.0	56.4	62.6
(b)		✓		200	63.4	82.2	56.8	63.2
(c)	✓	✓		200	67.3	**82.5**	57.2	63.9
(d)	✓	✓	✓	200	67.5	82.4	57.0	63.6
(e)	✓	✓	✓	**800**	**71.1**	**82.5**	**57.4**	**64.0**

MoCo v2: takes some ideas from SimCLR



MoCo v2: takes some ideas from SimCLR



	unsup. pre-train					ImageNet
case	MLP	aug+	cos	epochs	batch	acc.
MoCo v1 [6]				200	256	60.6
SimCLR [2]	✓	✓	✓	200	256	61.9
SimCLR [2]	✓	✓	✓	200	8192	66.6
MoCo v2	✓	✓	✓	200	256	**67.5**
results of ***longer*** *unsupervised training follow:*						
SimCLR [2]	✓	✓	✓	1000	4096	69.3
MoCo v2	✓	✓	✓	800	256	**71.1**

Summary

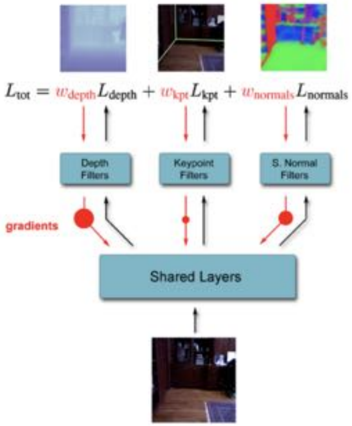
- Representations learned with self-supervision using contrastive losses rival with supervised representation learning, and supervised training.
- Active field of research, with a lot of involvement from main tech companies (Google & Facebook)
- Although contrastive learning seems elegant and generic, there are still a lot of details that matter: what augmentation, which learning rate schedule, architecture, hyperparameters, etc.
- All results are very, very recent (some from a few days!!) - all must be taken with a grain of salt.
- There are many methods we didn't cover. In particular:
  - BYOL: doesn't use negative samples!!
  - SwAV: claims state-of-the-art results without huge batches nor memory bank

Contents

Multi-task learning

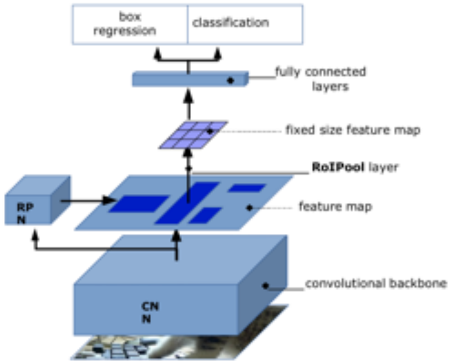
Multi-task network: A single network with shared layers and tasks specific outputs

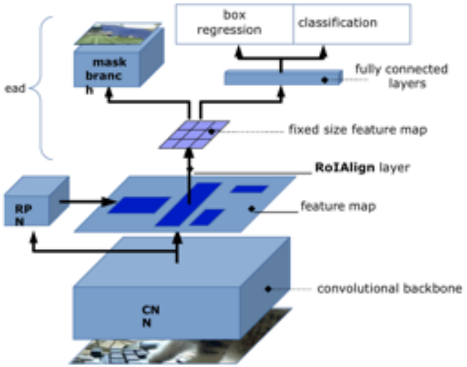
- Multi-task loss combining individual losses
- Don't need to have all labels for all training samples
- If tasks are related, the shared weights benefit from the training samples for all tasks
- Related to transfer learning, but different: tasks are learned simultaneously, and works better if number of training samples is similar for all tasks



$L_{\text{tot}} = w_{\text{depth}} L_{\text{depth}} + w_{\text{kpt}} L_{\text{kpt}} + w_{\text{normals}} L_{\text{normals}}$
Depth Filters
Keypoint Filters
S. Normal Filters
gradients
Shared Layers

Multi-task learning: famous examples

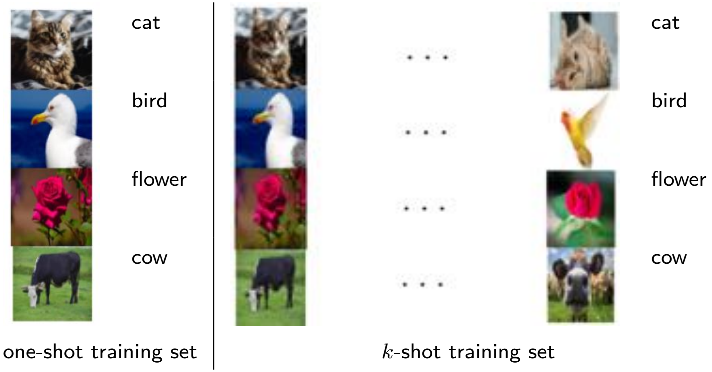




Faster R-CNN [Ren et al. 2015] and Mask R-CNN [He et al. 2017] are multi-task networks. The convolutional backbone is shared for different tasks (object classification, boundix box prediction and object mask).

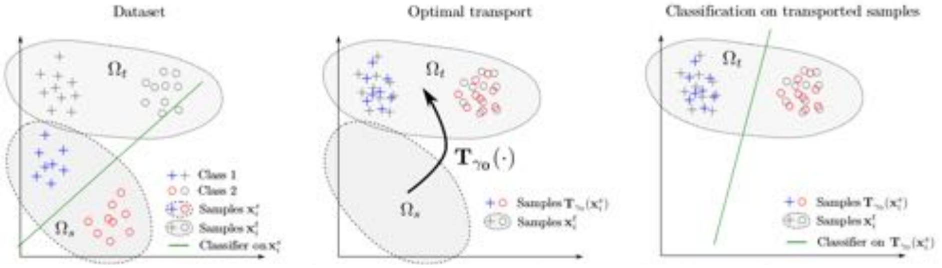
Figure from [Ildoo Kim]

Few-shot learning and meta-learning



- **Few-shot learning** algorithms aim at learning new tasks (e.g. classify new classes) only from a few labeled examples (humans have this ability).
- Few-shot learning from scratch is impossible: start from a network pre-trained for related tasks!
- **Meta-learning:** pre-train a network so that it's ready to learn from few examples (learning to learn).

Domain adaptation


Dataset
Optimal transport
Classification on transported samples
Ω_t
Ω_s
$\mathbf{T}_{\gamma_0}(\cdot)$
Class 1
Class 2
Samples $\mathbf{x}_i^s$
Samples $\mathbf{x}_i^t$
Classifier on $\mathbf{x}_i^s$
Samples $\mathbf{T}_{\gamma_0}(\mathbf{x}_i^s)$
Classifier on $\mathbf{T}_{\gamma_0}(\mathbf{x}_i^s)$

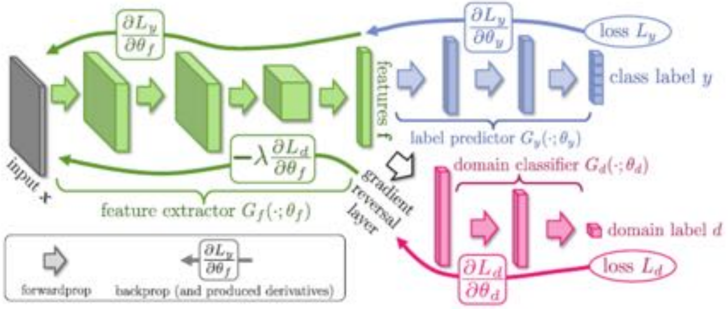
Solve the same task on two different domains. **Homogeneous** domain adaptation is when both feature spaces coincide $\mathcal{X}_S = \mathcal{X}_T$, but the data distributions differ $p_S(x) \neq p_T(x)$ (**domain drift**).

Some approaches try to reduce the domain shift by computing a mapping φ such that

$$p_S(\varphi^{-1}(x)) = p_T(x).$$

Figure from [Optimal transport for domain adaptation, Courty et al. 2016]

Domain adaptation



$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
class label y
features f
label predictor $G_y(\cdot;\theta_y)$
$-\lambda\frac{\partial L_d}{\partial \theta_f}$
domain classifier $G_d(\cdot;\theta_d)$
input x
feature extractor $G_f(\cdot;\theta_f)$
gradient reversal layer
domain label d
loss L_d
$\frac{\partial L_d}{\partial \theta_d}$
forwardprop
backprop (and produced derivatives)

Find a common representation of the data φ such that

$$p_S(\varphi^{-1}(x)) \approx p_T(\varphi^{-1}(x)).$$

A **domain discriminator** is trained simultaneously to classify $\varphi(x)$ in source and target domains. The feature extraction network is trained to confound the discriminator, while at the same time solving the task.

[Domain-adversarial training of neural networks, Gannin et al. 2016]