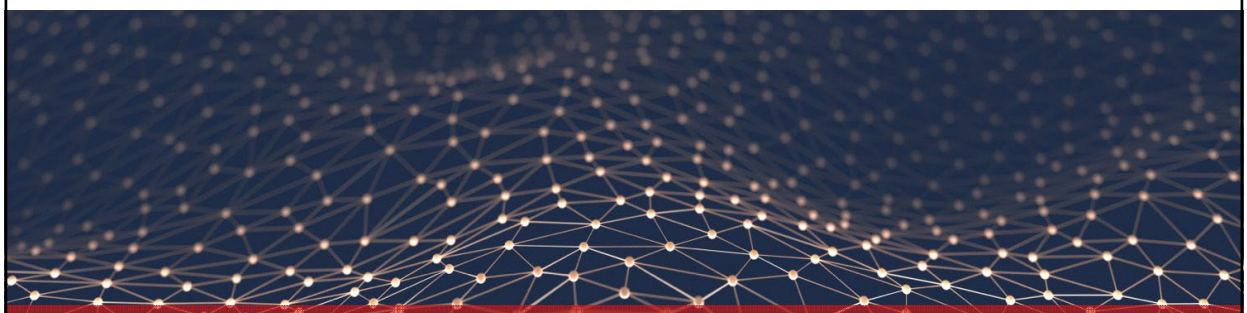


Module 10: Beyond Supervised learning: Semi/Self Supervised Learning

5LSM0: Convolutional neural networks for computer vision

Farhad G. Zanjani (f.ghazvinian.zanjani@tue.nl)

Electrical Engineering / VCA research group

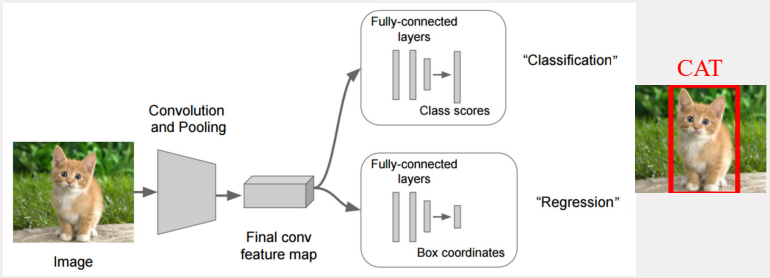
Types of learning algorithms

- Supervised learning
- Semi-supervised learning
- Self-supervised learning
- Unsupervised learning
- Reinforcement learning

Introduction: Supervised learning algorithms

Building a mathematical model from a set of data that contains both the inputs and outputs.

- Classification algorithms
- Regression algorithms
- Similarity learning



Introduction: Semi-Supervised learning algorithms

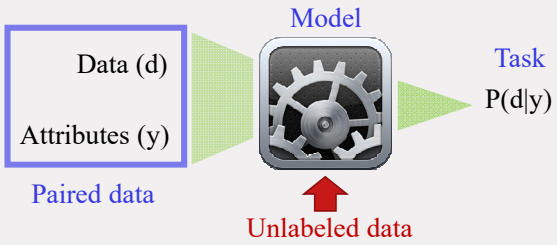
While the supervised learning works pretty well, why bother ourselves?

Because people want better performance for free!

- Unlabeled data is cheap!
- Labeled data can be hard to get:
  - Human annotation is time-consuming and boring
  - Not scalable to large datasets
  - Labeling may require experts (e.g. most of medical images)
  - Labeling may require special devices
  - Might not be well-defined (e.g. annotating actions in videos)

Introduction: Semi-Supervised learning algorithms

Building a mathematical model from a set of labeled and unlabeled data.

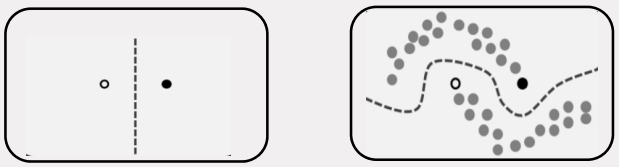


Introduction: Semi-Supervised learning algorithms

Building a mathematical model from a set of labeled and unlabeled data.

Question: How unlabeled data helps for learning the task better?

Answer: Links between distribution of unlabeled data and the target attributes



Introduction: Semi-Supervised learning algorithms

Inductive vs. **Transductive** semi-supervised learning

Definitions: Assume a given training set $\{(x_i, y_i)\}_{i=1}^{l}, \{x_j\}_{j=l+1}^{l+u}$

Inductive setting predicts the labels on the future test data. So the predictor may deploy on data beyond the $\{x_j\}_{j=l+1}^{l+u}$ sample set.

Transductive setting aims to predict labels only within the unlabeled instances $\{x_j\}_{j=l+1}^{l+u}$

An interesting analogy: inductive semi-supervised learning is like an in-class exam, where the questions are not known in advance, and a student needs to prepare for all possible questions; in contrast, transductive learning is like a take-home exam, where the student knows the exam questions and needs not prepare beyond those.

Introduction: Semi-Supervised learning algorithms

Popular semi-supervised learning methods:

- Self-training
- Probabilistic Generative Models
- Cluster-then-Label Methods
- Co-Training and Multiview Learning
- Graph-Based Methods
- Semi-Supervised Support Vector Machines
- …

Introduction: Semi-Supervised learning algorithms

- **Self-training** (also called self-teaching or bootstrapping): learning process uses its own predictions to teach itself

Algorithm

Inputs: Labeled data $\{(x_i, y_i)\}_{i=1}^{l}$, unlabeled data $\{x_j\}_{j=l+1}^{l+u}$
(1) Initially , let $L=\{(x_i, y_i)\}_{i=1}^{l}$ and $U = \{x_j\}_{j=l+1}^{l+u}$
Repeat:
(2) Train the predictor (f) from L using supervised learning
(3) Apply f to the unlabeled instances in U
(4) Remove a subset S from U; add $\{(x, f(x))|x \in S\}$ to L
Until: stopping criteria

Introduction: Semi-Supervised learning algorithms

- **Self-training**

Advantages:

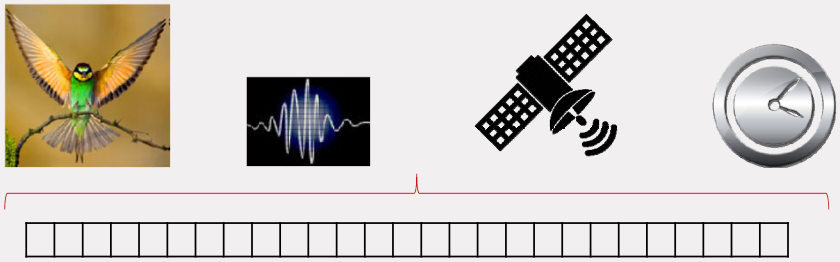
- The simplest semi-supervised learning method.
- A wrapper method, applies to existing (complex) classifiers.
- Often used in real tasks like natural language processing

Disadvantages:

- Early mistakes could reinforce themselves.
- Cannot say too much in terms of convergence (except in special cases like linear functions).

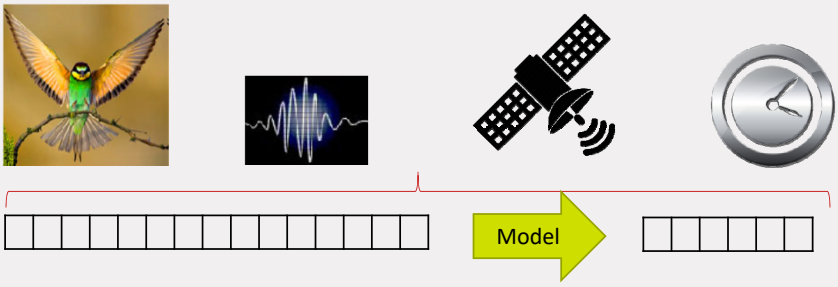
Self-Supervised learning algorithms

- Consider all the data associated to a sample
- For example, we have an image with any other additional measured signal (audio, GPS, time of the day, etc.)
- Collect each piece of information into a vector



Self-Supervised learning algorithms

- Transform the vector into two parts: data and targets



- Train a model to retrieve the **missing** observations (the targets) in the data

Self-Supervised learning algorithms

- Prior works
- Self-supervised task formulation is not new

Examples of earlier works in the literature

- Caruana, R. and de Sa, V. R. "Promoting poor features to supervisors: Some inputs work better as outputs." NIPS 1996
- Ando, R.K. and Zhang, T. "A framework for learning predictive structures from multiple tasks and unlabeled data." JMLR 2005

Self-Supervised learning algorithms

- Blurry line between Self-Supervised Learning (SSL) and Unsupervised Learning (UL)
- In UL we build an approximate model of $p(x)$
- In SSL split x into x_1 and x_2 and train a model ($\emptyset$) for

  $\emptyset(x_1) = x_2$ or $\emptyset(x_1) = p(x_2|x_1)$
- In the SSL formulation we can therefore write:

  $p(x) = p(x_2|x_1) \,.\, p(x_1) = \emptyset(x_1) \,.\, p(x_1)$

Learning a Self-Supervised Task

- The previous relations define a **supervision signal**
- The conditional probability $\emptyset(x_1) = p(x_2|x_1)$ might be a convolutional network so that it can adapt to any image size
- We take the representation from this conditional probability function

Self-Supervised Learning Loss

- General notation: Use transformations T_1 and T_2 of the input to get x_1 and x_2

$$\emptyset(T_1(x)) = T_2(x) \quad \text{or} \quad \emptyset(T_1(x)) = p(T_2(x)|T_1(x))$$

- The loss function is therefore either

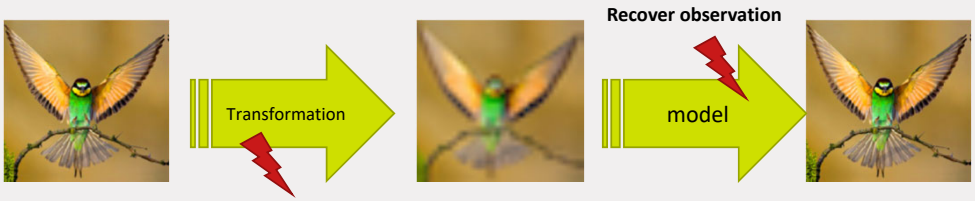
$$L(\emptyset) = \frac{1}{m}\sum_{i=1}^{m}\left|\emptyset\left(T_1\left(x^{(i)}\right)\right) - T_2\left(x^{(i)}\right)\right|$$

Or

$$L(\emptyset) = -\frac{1}{m}\sum_{i=1}^{m}\log\emptyset\left(T_1\left(x^{(i)}\right)\right).T_2\left(x^{(i)}\right)$$

Learning by Dropping & Retrieving Observations

The data transformations need to **drop observations** and the model needs to **recover observations**



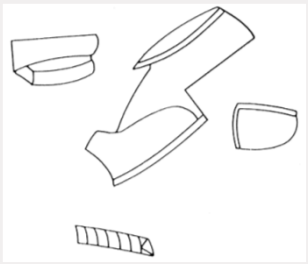
Thus, we can recover dropped observations only if they **depend** on the transformed data (we call this **redundancy**)

Self-Supervised Learning

Task: rearrange parts to form a familiar object

No additional information is made available to us in addition to the photo

What knowledge do we need to be able to solve the puzzle?



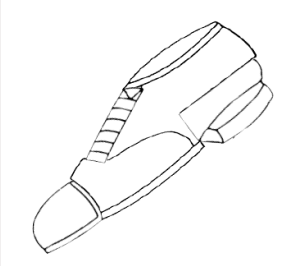
Self-Supervised Learning

Task: rearrange parts to form a familiar object

No additional information is made available to us in addition to the photo

What knowledge do we need to be able to solve the puzzle?

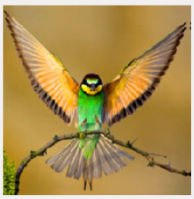
Is it **necessary** to know **how objects are made**?



Restrictions on the Data Transformations

After training the SSL model how do we use it?

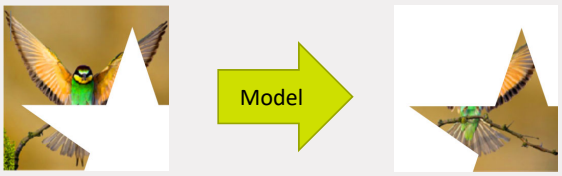
The way we split the data determines the format of the new data and this is what we need to use also later



Restrictions on the Data Transformations

After training the SSL model how do we use it?

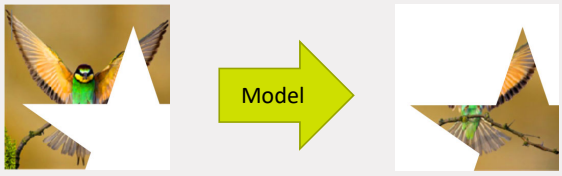
The way we split the data determines the format of the new data and this is what we need to use also later



Restrictions on the Data Transformations

After training the SSL model how do we use it?

The way we split the data determines the format of the new data and this is what we need to use also later



To feed an image to the model later on, choose the split so that the data term is an image or the cropping of an image

Learning by Dropping & Retrieving Observations

Example (1) **Context Prediction**

Drop the relative location of two tiles

$$x = \begin{bmatrix} r \\ g \\ b \\ u \\ v \end{bmatrix}$$

Image tiles learning localization

Unsupervised Visual Representation Learning by Context Prediction. C. Doersch, A. Gupta, and A. A. Efros. *ICCV 2015*

Learning by Dropping & Retrieving Observations

Example (2) **Jigsaw puzzles**

Drop the ordering of all the puzzle tiles

$$x = \begin{bmatrix} r \\ g \\ b \\ u \\ v \end{bmatrix}$$

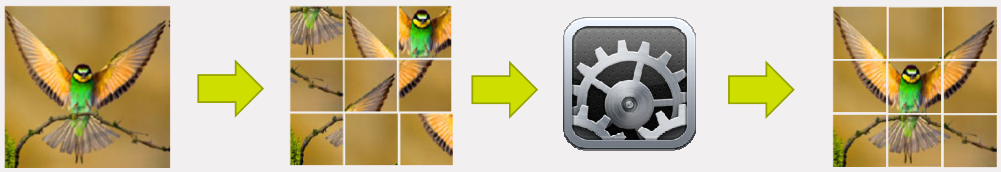


Image puzzle learning ordered

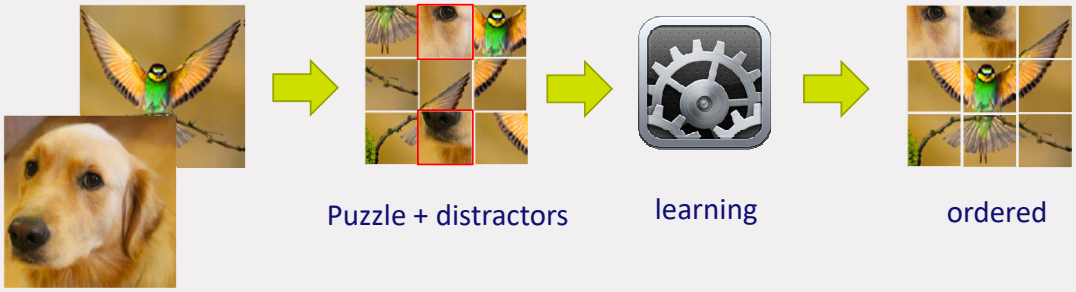
Unsupervised learning of visual representations by solving jigsaw puzzles M. Noroozi and P. Favaro, ECCV 2016

Learning by Dropping & Retrieving Observations

Example (3) **Jigsaw ++**

Drop the ordering of all the puzzle tiles and feed outlier(s)

$$x = \begin{bmatrix} r \\ g \\ b \\ u \\ v \end{bmatrix}$$



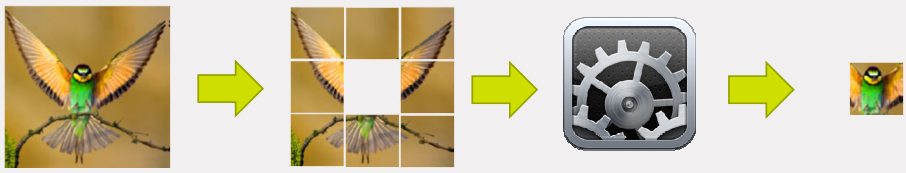
Boosting Self-Supervised Learning via Knowledge Transfer, M. Noroozi, A. Vinjimoor, P. Favaro, H. Pirsiavash. CVPR 2018

Learning by Dropping & Retrieving Observations

Example (4) **context auto-encoder**

Drop an entire image tile

$$x = \begin{bmatrix} r \\ g \\ b \\ u \\ v \end{bmatrix}$$



Context encoders: Feature learning by inpainting D. Pathak et al., CVPR 2016

Self-supervised learning

Example (5) **colorization**

Drop the data color

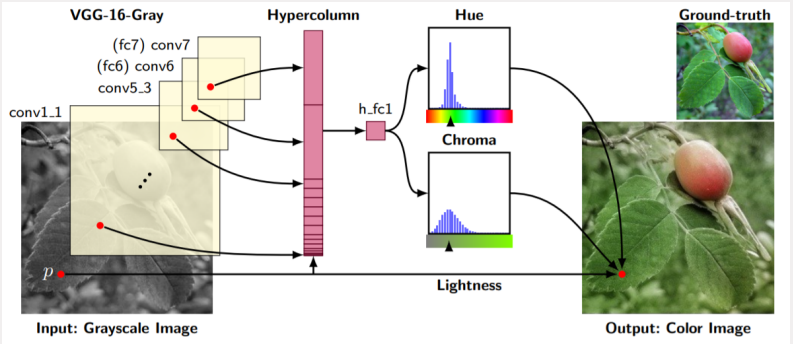
image de-color learning colorized

Learning representations for automatic colorization
G. Larsson et al., ECCV 2016

Self-supervised learning

Example (5) **colorization**

Drop the data color

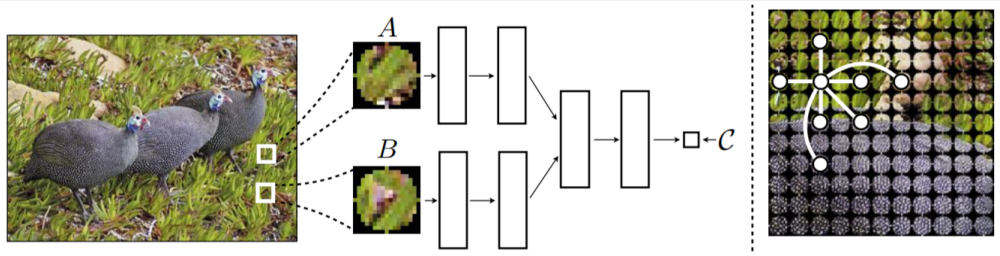


Learning representations for automatic colorization
G. Larsson et al., ECCV 2016

Self-supervised learning

Example (7) **co-occurrences in space and time**

Classifying adjacent patches vs. nonadjacent patches

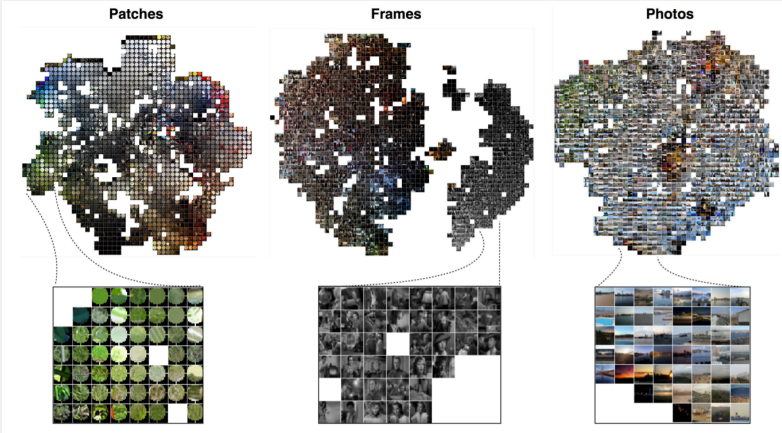


Learning visual groups from co-occurrences in space and time
P. Isola, ICLR workshop 2016

Self-supervised learning

Example (7) co-occurrences in space and time

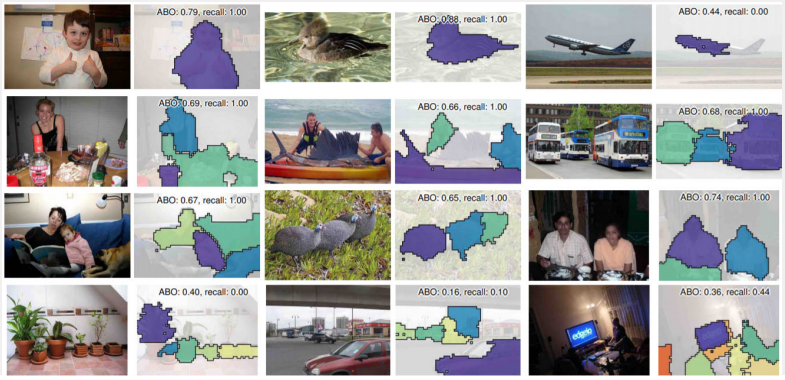
Learnt affinities in three different domains (t-SNE visualization)



Learning visual groups from co-occurrences in space and time
P. Isola, ICLR workshop 2016

Self-supervised learning

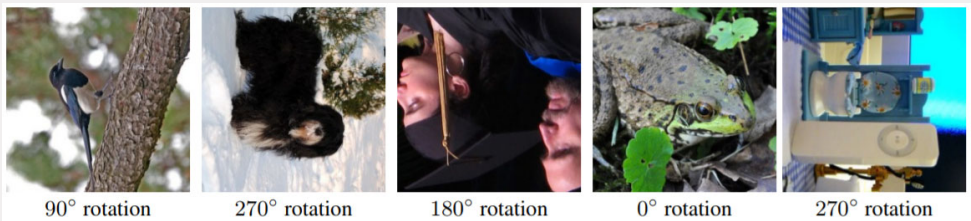
Example (7) co-occurrences in space and time



Learning visual groups from co-occurrences in space and time
P. Isola, ICLR workshop 2016

Self-supervised learning

Example (8) **Rotation prediction**



Unsupervised representation learning by predicting image rotations.
Gidaris et al., ICLR 2018.

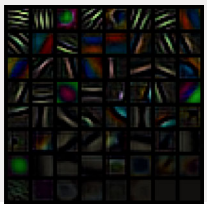
Self-supervised learning

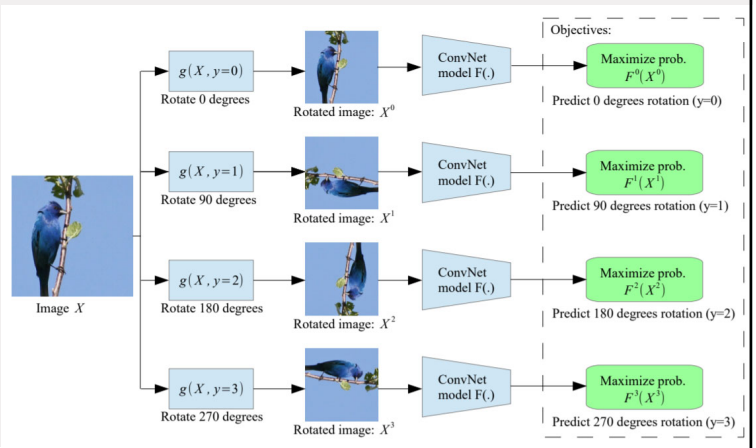
Example (8) Rotation prediction

Learnt features by network

supervised

SSL by rotation





Unsupervised representation learning by predicting image rotations. Gidaris et al., ICLR 2018.

Self-supervised learning

Example (9) **learning to count objects** (without specifying what objects are)

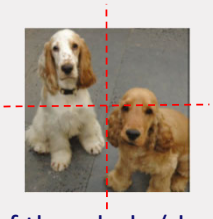
$$\emptyset\left(\quad\right) = \begin{bmatrix} * \\ * \\ * \\ * \\ * \\ * \\ * \end{bmatrix}$$

Number of objects/parts/clusters

Self-supervised learning

Example (9) learning to count objects

If the feature counts objects and we split an image



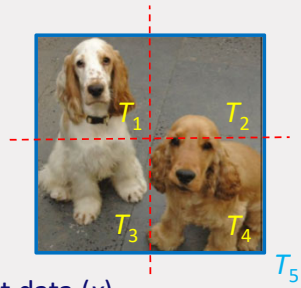
Then, the total number of objects in each tile must match that of the whole (down-sampled) image

$$\phi\left(\text{  }\right) + \phi\left(\text{  }\right) + \phi\left(\text{  }\right) + \phi\left(\text{  }\right) = \phi\left(\text{  }\right)$$

Representation learning by learning to count. Noroozi et al., ICCV 2017

Self-supervised learning

Example (9) learning to count objects



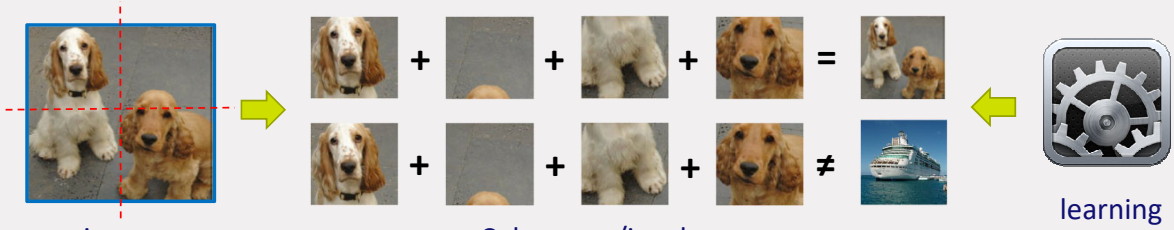
- The self-supervision signal is a relation

$$G\left(\emptyset(T_1(x)), \ldots, \emptyset(T_k(x))\right) = 0$$

- Have a set of transformations T_1, …, T_k applied to the input data (x)
- The relation G between the features of such transformations is known
- **Equivariance** is a special case

Self-supervised learning

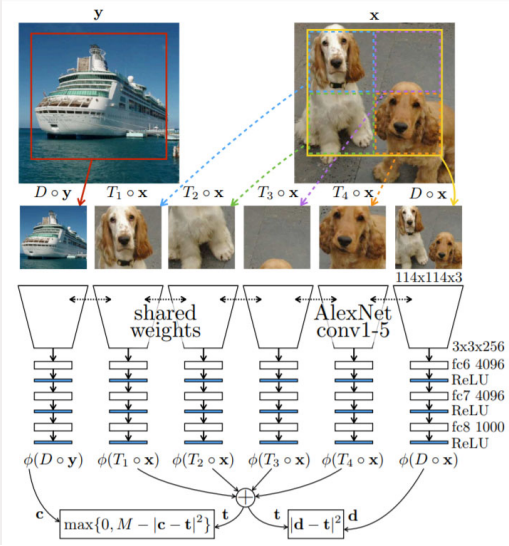
Example (9) learning to count objects



Representation learning by learning to count.
Noroozi et al., ICCV 2017

Self-supervised learning

Example (9) learning to count objects



Representation learning by learning to count.
Noroozi et al., ICCV 2017

Self-supervised learning

Example (10) **Temporal structure in video**

How learning from the sequence of data?

Give a triplet of images and then check if the middle one can lie in between.

Shuffle and Learn: Unsupervised Learning using Temporal Order Verification, I. Misra et al., ECCV 2016

Self-supervised learning

Data can be **heterogeneous**

- Data can be collected synchronously from multiple sensors

For example

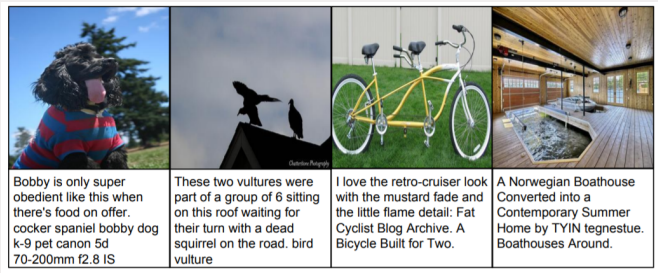
- Videos have images and audio
- Photos/videos with GPS tagging

Unsupervised Visual Representation Learning by Context Prediction. C. Doersch, A. Gupta, and A. A. Efros. *ICCV 2015*

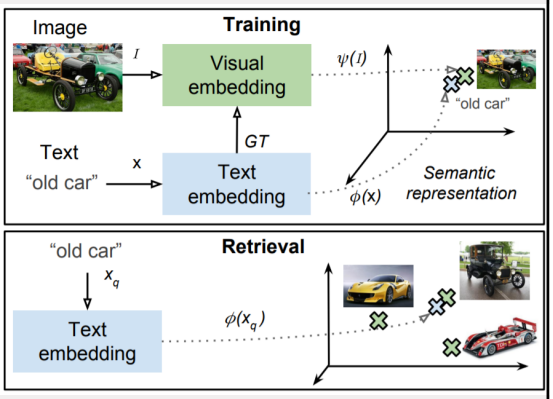
Self-supervised learning

Data can be **heterogeneous**

Example (11) **text to image domain**



Example of training data



Pipeline of the model

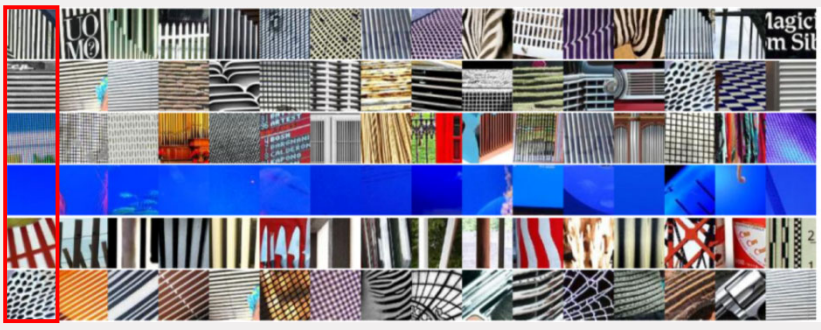
Self-Supervised Learning from Web Data for Multimodal Retrieval
R. Gomez et al., 2019.

Evaluation of Self-supervised learning

Quantitative evaluation

1) considering the units at each layer as object part detectors

Showing the top 16 activation (lowest *L1* distance with query) per unit for layer **Conv1**



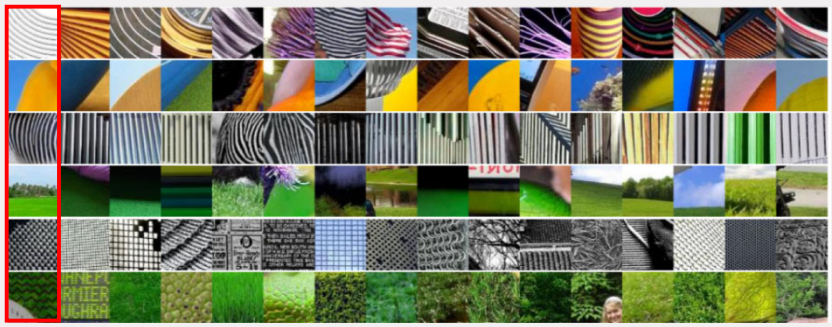
Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles.
M. Noroozi and P. Favaro, ECCV 2016

Evaluation of Self-supervised learning

Quantitative evaluation

1) considering the units at each layer as object part detectors

Showing the top 16 activation (lowest *L1* distance with query) per unit for layer **Conv2**



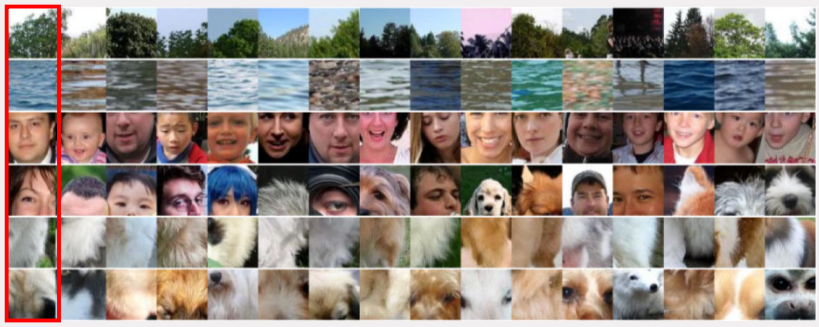
Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. M. Noroozi and P. Favaro, ECCV 2016

Evaluation of Self-supervised learning

Quantitative evaluation

1) considering the units at each layer as object part detectors

Showing the top 16 activation (lowest L1 distance with query) per unit for layer **Conv5**



Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. M. Noroozi and P. Favaro, ECCV 2016

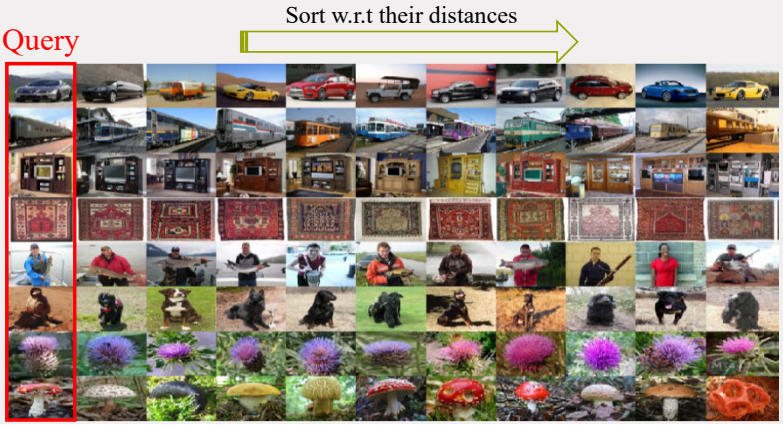
Evaluation of Self-supervised learning

Quantitative evaluation
2) Image retrieval

- To see what feature space we have learned, we can use image retrieval
- We compute features for a whole dataset of images (e.g. ImageNet)
- Take the feature vectors of query images and compute their Euclidean distance to all the other feature vectors in the dataset
- Rank images based on the distances

Evaluation of Self-supervised learning

Quantitative evaluation
2) Image retrieval



Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. M. Noroozi and P. Favaro, ECCV 2016

Self-supervised learning

Two common quantitative evaluations:

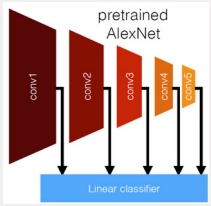
1) Freeze features and train a linear classifier on top of them

2) Transfer the knowledge learned with a SSL task via fine-tuning to Pascal VOC on the classification, detection and segmentation tasks

Note: The classifier is not strictly linear. There is a pooling layer to resize the features and also a softmax transformation

Self-supervised learning

Quantitative evaluation



- First training on the self-supervised task we freeze the weights of the network
- Then we train a linear classifier (includes a ReLU + feature resize through pooling + softmax) for each intermediate feature

ImageNet top-1 classification with linear layers

Method	Conv1	Conv2	Conv3	Conv4	Conv5
ImageNet labels	19.3	36.3	44.2	48.3	50.5
Random	11.6	17.1	16.9	16.3	14.1
Random rescaled Krähenbühl et al. (2015)	17.5	23.0	24.5	23.2	20.6
Context (Doersch et al., 2015)	16.2	23.3	30.2	31.7	29.6
Context Encoders (Pathak et al., 2016b)	14.1	20.7	21.0	19.8	15.5
Colorization (Zhang et al., 2016a)	12.5	24.5	30.4	31.5	30.3
Jigsaw Puzzles (Noroozi & Favaro, 2016)	18.2	28.8	34.0	33.9	27.1
BIGAN (Donahue et al., 2016)	17.7	24.5	31.0	29.9	28.0
Split-Brain (Zhang et al., 2016b)	17.7	29.3	35.4	35.2	32.8
Counting (Noroozi et al., 2017)	18.0	30.6	34.3	32.5	25.7
(Ours) RotNet	**18.8**	**31.7**	**38.7**	**38.2**	**36.5**

Unsupervised representation learning by predicting image rotations. Gidaris et al., ICLR 2018.

Self-supervised learning

Quantitative evaluation

PASCAL VOC 2007 classification and detection results, and PASCAL VOC 2012 segmentation results.

	Classification (%mAP)		Detection (%mAP)	Segmentation (%mIoU)
Trained layers	fc6-8	all	all	all
ImageNet labels	78.9	79.9	56.8	48.0
Random		53.3	43.4	19.8
Random rescaled Krähenbühl et al. (2015)	39.2	56.6	45.6	32.6
Egomotion (Agrawal et al., 2015)	31.0	54.2	43.9	
Context Encoders (Pathak et al., 2016b)	34.6	56.5	44.5	29.7
Tracking (Wang & Gupta, 2015)	55.6	63.1	47.4	
Context (Doersch et al., 2015)	55.1	65.3	51.1	
Colorization (Zhang et al., 2016a)	61.5	65.6	46.9	35.6
BIGAN (Donahue et al., 2016)	52.3	60.1	46.9	34.9
Jigsaw Puzzles (Noroozi & Favaro, 2016)	-	67.6	53.2	37.6
NAT (Bojanowski & Joulin, 2017)	56.7	65.3	49.4	
Split-Brain (Zhang et al., 2016b)	63.0	67.1	46.7	36.0
ColorProxy (Larsson et al., 2017)		65.9		38.4
Counting (Noroozi et al., 2017)	-	67.7	51.4	36.6
(Ours) RotNet	**70.87**	**72.97**	**54.4**	**39.1**

Unsupervised representation learning by predicting image rotations. Gidaris et al., ICLR 2018.

Self-supervised learning pitfall

Our target tasks need features related to objects Is all the data redundancy useful to the target tasks?

* No, Some redundancy is about low-level statistics (such as edges and corners)

* We are interested in high-level statistics (e.g., object parts and their co-location)

Self-supervised learning pitfall

- We need mechanisms to avoid that the model solves the tasks by learning low-level statistics

- We need to select what information to throw away

- Let's see an example

Self-supervised learning pitfall

Recall the Context Prediction

➢ Drop the relative location of two tiles

$$x = \begin{bmatrix} r \\ g \\ b \\ u \\ v \end{bmatrix}$$

Image tiles learning localization

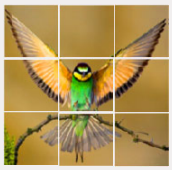
Unsupervised Visual Representation Learning by Context Prediction. C. Doersch, A. Gupta, and A. A. Efros. *ICCV 2015*

Self-supervised learning pitfall

Recall the Context Prediction

What redundancy is needed to solve the task?

- Geometric information (location)



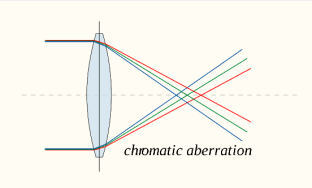
Possible dependencies:

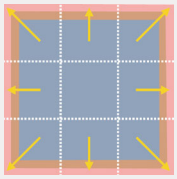
1. **No need to look at other tiles:** The absolute location (relative to the image) is embedded in each tile
2. **Need to look at other tiles**: The relative location is obtained through tile comparisons

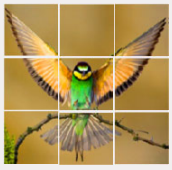
Unsupervised Visual Representation Learning by Context Prediction. C. Doersch, A. Gupta, and A. A. Efros. *ICCV 2015*

Self-supervised learning pitfall

Recall the Context Prediction







Chromatic aberration encodes absolute location. This might be a shortcut path for the learning model to solve the task!

Unsupervised Visual Representation Learning by Context Prediction. C. Doersch, A. Gupta, and A. A. Efros. *ICCV 2015*

Self-supervised learning pitfall

Recall the Jigsaw puzzles

Drop the ordering of all the puzzle tiles

$$x = \begin{bmatrix} r \\ g \\ b \\ u \\ v \end{bmatrix}$$

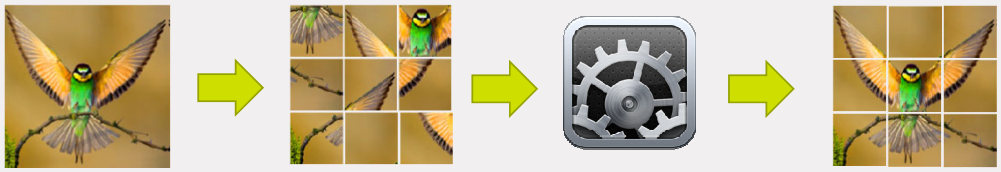
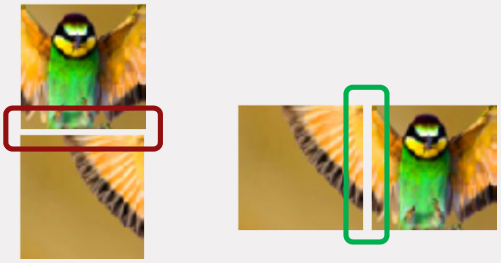


Image puzzle learning ordered

Unsupervised learning of visual representations by solving jigsaw puzzles
M. Noroozi and P. Favaro, ECCV 2016

Self-supervised learning pitfall

finding the relative position can be solved by comparing **only the borders**!



Unsupervised learning of visual representations by solving jigsaw puzzles
M. Noroozi and P. Favaro, ECCV 2016

Self-supervised learning pitfall

Recall the learning to count objects

- We train a model to count objects by enforcing

- If the model can distinguish tiles from down-sampled images, then it could learn two different families of features
- How can it tell the difference?
  1. Down-sampling method leaves a **footprint**
  2. Less noise in the down-sampled image
  3. Chromatic aberration (again)
  4. …

Self-supervised learning pitfall

These undesired learning behaviors are called **shortcuts**

- They are **specific** to each SSL task
- E.g.: Context/puzzle (chrom. aber.), counting (down-sampling, noise, chrom. aber.), time-arrow (video compression), artifacts (decoder artifacts), landmarks (image interpolation)
- Need to analyze and understand all the shortcuts problems in the specific tasks

Self-supervised learning

Dealing with Shortcuts

The general strategy is to **drop the unwanted redundancy** (through noise) that allows solutions through shortcuts

Example

- To reduce chromatic aberration generate new images by randomly and independently scaling and shifting (by very small amounts) the color channels of each image

- This will randomize chromatic aberration and not allow a model to learn from it

TU/e

Self-supervised learning

References:

(1) Recent publications which mostly were cited on the slides.

(2) Some SSL slides were borrowed from the Prof. P. Favaro presentation.

(3) New direction in semi-supervised learning, A.B. Goldberg, PhD thesis, 2010.

TU/e